# Development of a block-coupled finite volume methodology for non-linear elasticity

Zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)
Genehmigte Dissertation von Lucas Ribeiro de Azevedo aus Rio de Janeiro, Brasilien
Tag der Einreichung: 8. Mai 2020, Tag der Prüfung: 15. Juli 2020

1. Gutachten: Prof. Dr. rer. nat. Michael Schäfer
2. Gutachten: Prof. Dr. Francisco J. Galindo-Rosales
3. Gutachten: Prof. Dr. Philip Cardiff
Darmstadt – D 17

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Mechanical Engineering
Department
FNB

Development of a block-coupled finite volume methodology for non-linear elasticity

Accepted doctoral thesis by Lucas Ribeiro de Azevedo

1. Review: Prof. Dr. rer. nat. Michael Schäfer
2. Review: Prof. Dr. Francisco J. Galindo-Rosales
3. Review: Prof. Dr. Philip Cardiff

Date of submission: May 8, 2020
Date of thesis defense: 15. Juli 2020

Darmstadt – D 17

*Dedicated to Karoline Pontes de Freitas,*
*an amazing human being and wife.*

# Acknowledgements

I would like to express my deepest gratitude to my supervisors Prof. Dr. rer. nat. Michael Schafer, Prof. Dr. Francisco J. Galindo-Rosales and Prof. Dr. Philip Cardiff for the opportunity, guidance and support.

I am thankful to FNB system administrator Michael Fladerer and to GSC-CE system administrator Christian Schmitt for helping me with technical matters.

All the financial support provided by the Brazilian National Research Council (CNPq) is gratefully acknowledged.

I would like to thank my family for all the support during my whole life. A special thanks to my wife for being the strongest support during this doctorate journey.

# Abstract

A *rheinforce* composite consists of a laminar sheet of compacted micro-agglomerated cork engraved by laser with a network of microchannels and filled with a concentrated aqueous solution of shear-thickening fluid. Using it for personal protection equipment, as an energy-absorbing material layer, is one of its important applications. An accurate numerical modeling of such material is not currently available and would be a valuable tool during design and manufacturing processes.

A virtual dynamic drop tower test can be used to shed lights on the dynamics of the energy dissipation of the *rheinforce* composite. From the continuum mechanics point of view, it can be translated into an initial-boundary value problem with enclosed-Fluid-Structure Interaction, contact boundary and rigid body dynamics.

The open-source OpenFOAM toolbox called Solids4foam (S4F) offers a very attractive starting point to build a solver for that initial-boundary value problem. Not only because no commercial software available (as far as the author is aware) is as customizable as needed, but also because S4F offers a strongly-coupled Fluid-Structure Interaction with fully customizable and replaceable solid and fluid solvers. Furthermore, these solvers are constructed based on only one numerical framework, the Finite Volume Method.

Unfortunately, numerical simulations revealed that the current finite volume methodologies for solid mechanics implemented in S4F, i.e. Segregated (SEG) and the recently developed Block-Coupled (BC), cannot simulate the solid part of the *rheinforce* composite under the finite strain regime. Moreover, the SEG method supports general materials (in theory any material law can be used), but it can be very slow. The BC method is fast, but only Hooke's law (an infinitesimal strain model) is supported.

The principal aim of the thesis is to present the development of a new BC methodology that preserves, in a great extent, the fast convergence of the original BC and material generality (only requiring the elasticity tensor to have right-minor symmetry) of SEG.

This work also demonstrates that the Field Operation and Manipulation (FOAM) concept should be implemented in a high-level programming environment to fill an important "prototyping gap" left untouched by OpenFOAM and S4F, that is, between concept development and concept implementation in a high-performance programming language.

# Zusammenfassung

Ein *Rheinforce*-Verbundwerkstoff besteht aus einer laminaren Schicht aus Kork, in welche mittels Lasertechnik ein Netzwerk von Mikrokanälen eingefräst wurde, die mit einer konzentrierten wässrigen Lösung einer scher-verfestigenden Flüssigkeit gefüllt sind. Eine der wichtigsten Anwendungen ist der Einsatz als energieabsorbierende Materialschicht in persönlichen Schutzausrüstungen. Bis heute gibt es keine genaue numerische Modellierung für ein solches Material, welche allerdings für den Prozess des Designs und der Herstellung sehr nützlich wäre.

Ein virtueller dynamischer Fallturmtest kann angewendet werden, um die Dynamik der Energiezerstreuung des *Rheinforce*-Verbundmaterials zu erforschen. Vom Standpunkt der Kontinuumsmechanik kann die Dynamik als Anfangs-Randwertproblem mit der Interaktion zwischen Struktur und eingeschlossener Flüssigkeit (Fluid-Strukturkopplung), Kontaktgrenze und Festkörperdynamik betrachtet werden.

Das Open-Source Werkzeug Solids4foam (S4F) von OpenFOAM bietet einen attraktiven Ausgangspunkt für die Entwicklung eines Programms zur Lösung des Anfangs-Randwertproblems. Gründe dafür sind, dass es – soweit dem Autor bekannt - keine kommerzielle Software gibt, welche so gut den jeweiligen Bedürfnissen angepasst werden kann, und dass S4F eine stark gekoppelte Fluid-Struktur-Interaktion mit anpassungsfähigen und austauschbaren Lösungsprogrammen für Feststoffe und Lösungen anbietet. Außerdem basieren diese Lösungsprogramme auf einem einzigen numerischen Grundgerüst, dem Finite-Volumen-Verfahren.

Leider zeigten numerische Simulationen, dass die aktuellen Finite-Volumen-Verfahren, die Teil des S4F Werkzeuges sind, d.h. das segregierte (SEG) und das kürzlich entwickelte Block-Coupled (BC) Verfahren nicht in der Lage sind, den Festkörperanteil des *Rheinforce*-Verbundmaterials unter endlichen Belastungskonditionen zu simulieren. Außerdem unterstützt die SEG-Methode allgemeine Materialien (theoretisch kann jedes Materialgesetz verwendet werden) aber die Methode ist sehr zeitaufwendig. Die BC-Methode

ist schnell, kann aber nur das Hookesche Gesetz (eine infinitesimale Spannungstheorie) unterstützen.

Das Hauptziel dieser Doktorarbeit ist die Entwicklung einer neuen BC-Methode, welche weitgehend die schnelle Konvergenz der ursprünglichen BC-Methode und die materielle Allgemeinheit (welche nur des Elastizitätstensors bedarf, um die rechte Nebensymmetrie zu erhalten) von SEG beibehält.

Diese Arbeit zeigt auch, dass das Field Operation And Manipulation (FOAM) Konzept in höheren Programmierumgebungen eingeführt werden sollte, um eine wichtige “Prototypisierungs-Lücke” zu schließen, welche bei OpenFOAM und S4F besteht, d.h. zwischen Konzeptentwicklung und Konzepteinsatz in einer höheren Programmiersprache.

# Contents

# Acronyms

| | |
|---|---|
| BC | Block-Coupled |
| CCM | Computational Continuum Mechanics |
| CSM | Computational Solid Mechanics |
| DSL | Domain-Specific Language |
| eFSI | enclosed-Fluid-Structure Interaction |
| FEM | Finite-Element Method |
| FOAM | Field Operation And Manipulation |
| FSI | Fluid-Structure Interaction |
| FV | Finite Volume |
| FVM | Finite-Volume Method |
| MMS | Method of Manufactured Solutions |
| nFVM | nano Finite Volume Method |
| NLBC | non-linear Block-Coupled |
| S4F | Solids4foam |
| SEG | Segregated |
| STF | Shear-Thickening Fluid |
| TL | Total Lagrangian |
| TLID | Total Lagrangian Incremental Displacement |
| TLTD | Total Lagrangian Total Displacement |
| UL | Updated Lagrangian |
| ULID | Updated Lagrangian Incremental Displacement |

# 1 Introduction

## 1.1 Motivation

Injuries due to accidents and violence are a major public health problem [1]. One in three European workers is exposed to vibrations at work, resulting in vibration syndromes and vibration-related injuries [2]. Prevention or minimization of many of these impact-related and vibration-related issues could be achieved by wearing adequate personal protection equipment, e.g. helmets or anti-vibration gloves. So, there is a strong motivation to develop advanced passive energy-absorbing materials. There is also a growing public awareness requiring eco-friendly, sustainable and recyclable materials.

The state-of-art of advanced passive energy-absorbing materials consists of impregnating open-cell foams with Shear-Thickening Fluid (STF) [3]. The majority of these materials deform by crushing, limiting their efficiency and use to just one single time due to a permanent deformation. Moreover, they are not environmentally sustainable because they are not renewable. Furthermore, the impregnation technique has two major drawbacks: i) it cannot be applied to closed-cell materials (like cork); and ii) the path network through which the fluid will flow is predetermined by the structure of the foam, which prevents the optimization of the amount of energy dissipated for a specific application.

Galindo-Rosales et al. has patented a technology [4, 5, 6], hereafter named *rheinforce*, that allows the reinforcement of the mechanical performance of any scaffold material by adding a complex fluid through embedded microfluidic patterns. Microfluidics is known for the manipulation of low amount of fluid in channels, with characteristic length scales below one millimeter [7]; but also for allowing the design of microfluidic rectifiers [8, 9, 10, 11]. Microfluidic rectifiers are microfluidic channels specifically designed to exploit the anisotropy in flow resistance [12]. If the flow resistance can be controlled, so can the energy dissipated. Moreover, the shape of the microfluidic rectifiers can be numerically optimized to maximize the flow resistance according to rheological properties of the fluid

[13, 14]. It is also important to highlight the fact that the non-linear response of the complex fluids is enhanced at micro scale. In the case of viscoelastic fluids, the elastic instabilities can be triggered at low Reynolds numbers, resulting in an extra pressure drop [15, 16, 16]; while shear thickening fluids show enhanced rise in the viscosity when flowing under confinement [17, 18]. When the composite is subjected to an impact or vibration, the energy is dissipated by the combined effect of: 1) the cork deformation; 2) the complex fluid flows confined in the microchannels producing viscous dumping; and 3) the fluid-structure interaction.

Thus, the *rheinforce* technology has been successfully applied to a closed-cell foam, i.e. micro-agglomerated cork, [19] and it allowed the development of a helmet liner with improved damping performance [20].

Optimization of the mechanical performance of the *rheinforce* composites cannot be done exclusively by experiments, because it would be costly and inefficient. The Computational Physics offers an alternative investigation framework through analytical and numerical modeling [21].

However, the optimization of the microfluidic rectifiers is focused solely on the fluid flow process and does not consider the deformation of the scaffold material nor the fluid-structure interaction. Consequently, the *rheinforce* composite has not been fully optimized yet considering the whole mechanical process of the impact (or vibration), as studies of the solid mechanics and the fluid-structure interaction, have failed to be considered in the design of composites [19, 22].

The *rheinforce* composites are tested experimentally in low velocity impact test machine, which consist of measuring the force with which the composite responds to an impact of an anvil with a controlled weight dropped from a certain height. Thus, the numerical simulation of a 3-D dynamic test resorting to a drop tower can be used to shed light on the dynamics of the energy dissipation of a *rheinforce* composite [23]. A natural question would arise: How to model each part and the interaction between them? As a first approximation, the object dropped on the brick can be modeled as a rigid body. Thus, a solver to simulate rigid body dynamics would be needed. Which parts of the rigid body and the composite are touching each other must be found by a contact algorithm [24, 25]. The solid part[1], undergoing large compression, should be modeled as viscoelastic solid, since cork is highly viscoelastic [28, 29, 30]. STF is a complex fluid which may be assumed to be incompressible [22]. The boundary of a cavity is the interface where the

[1]Cork is a natural foam [26]. Interestingly, the Poisson's ratio of the cork is close to zero [27], making it an ideal material for the *rheinforce* composites, because during impact the microchannels do not get jammed.

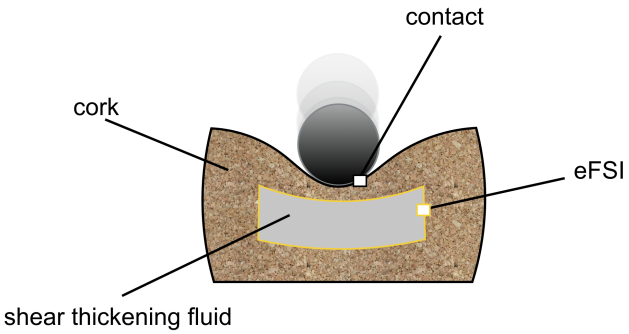



Figure 1.1: Schematic drawing of the initial-boundary value problem for a drop test.

fluid and the solid interact with each other, so to a more accurate model of the scenario a Fluid-Structure Interaction (FSI)[2] solver would also be necessary.

From the continuum mechanics point of view, the above simulation can be mapped into an initial-boundary value problem with enclosed-Fluid-Structure Interaction (eFSI), contact boundary and rigid body dynamics (see Fig. 1.1). As far as the author is aware, no commercial software available is as customizable as needed (e.g. no way to re-implement or adapt the FSI) to simulate such problem.

The open-source, therefore fully customizable, Finite Volume (FV) toolbox for solid mechanics simulations Solids4foam (S4F)[32] is an alternative. It implements: i) the classical FSI solver for initial-boundary value problems; ii) a Newtonian fluid solver that can be extended to support STFs; and iii) a contact solver for linearized elasticity, that can also be extended to support finite elasticity [24]. Currently, no viscoelastic solid model is available.

As hinted above, the Finite-Element Method (FEM) is not the only way to simulate solid mechanics. The Finite-Volume Method (FVM) can be considered as an alternative, specially if FSI is a must. Using FVM as a single framework to simulate FSI cases simplifies both understanding and programming, opens the possibility of using only one software algorithm and ensures a conservative resolution of the governing equations. This thesis is part of an effort made by an ongoing research which is extending the S4F toolbox, having the single-framework approach in mind, to simulate the composite.

[2] A special solver is actually needed in order to take into account the fact that the fluid domain is entirely enclosed by Dirichlet boundary condition [31].

After an extensive investigation, it was revealed that the current solid solvers implemented in S4F cannot handle the 3-D drop test[3], mentioned above, using the state-of-art non-linear material law commonly employed to simulate cork (namely Ogden-Storakers [33, 34]). So it was inferred that, in fact, the FV methodologies for solid mechanics implemented in S4F, namely Segregated (SEG) variants [35], are not able to simulate such case.

The SEG methodology for solid closely resembles the procedures commonly used in fluid dynamics where memory-efficient segregated solution algorithms are used in conjunction with iterative linear solvers. In practice, the linear momentum vector equation is temporarily decoupled into three scalar component equations that are independently solved, where outer Fixed-Point/Picard iterations provide the required coupling [36].

Besides the SEG approach, there is another FV methodology implemented in S4F called Block-Coupled (BC). Recently developed by Cardiff et al. [36], in this method, the entire momentum equation is arranged in a single linear system. The convergence rate is notably higher than that obtained with the SEG approach/variants. Therefore, the implementation resulted in less execution time and memory requirements than a FEM software for the set of cases tested by Cardiff et al. [36].

Regarding the weaknesses of both approaches, on the one hand BC methodology is restricted to linearized elasticity (currently, only Hooke's law is supported, i.e., the simplest constitutive equation). On the other hand, SEG can experience slow convergence rates whenever there is strong coupling between displacement directions. This is particularly disadvantageous in comparison with the BC approach when considering partitioned FSI methods, which requires solving for both solid and fluid domains multiple times per time step [37].

Hence, a natural question emerges: is there a possibility of combining both SEG and BC to produce a superior method so that its implementation is close enough to the existing implementations in S4F? The present work not only answer this question in affirmative, but it also shows a concrete sample that can be readily implemented in S4F.

## 1.2 Objectives

These are the objectives of the thesis:

[3] All solvers diverge for no apparent reason at the first time step.

- Create a fully coupled cell-centred FV methodology for the analysis of non-linear elasticity[4] in Computational Solid Mechanics (CSM). The new methodology, mostly based on BC and named non-linear Block-Coupled (NLBC), should be better than SEG and BC methodologies, i.e. being as general and as fast-convergent as SEG and BC, respectively, without importing their major weakness. Furthermore, this new methodology should allow the creation of a family of FVM methods for solid mechanics;
- Expose the development of the NLBC and of a method based on it. Note that only one NLBC variant is presented, and for practical purposes, it has the same name of the methodology;
- Discuss why the tensorial field approach Field Operation And Manipulation (FOAM) to Computational Continuum Mechanics (CCM) using class-oriented techniques, developed by Weller et al. [38], should be promoted to a high-level programming environment;
- Based on the previous discussion, create a new abstract high-level framework for prototyping FVM concepts, i.e. a "FVM-based CCM laboratory";
- Present the validation of this framework by means of a concrete implementation.

## 1.3 Contributions

The main contributions of this thesis are:

- The creation of a whole new methodology that closely resembles FEM in the sense that: i) all displacement components are solved at the same time in a big linear system generated by applying the Newton-Raphson procedure on an *out of balance* force function; and ii) very general constitutive laws are supported in the finite strain regime;
- The creation of a new method that validates the NLBC methodology. At least one of the major qualities of each method, e.g. fast convergence (BC) and material generality (SEG), is present in NLBC. This method can simulate cases that SEG can not handle and BC will never do (simply because it is limited to linearized elasticity).

[4] In this work, this non-linearity means large displacements and large strains, i.e. it does not include elastoplasticity or viscoplasticity, for example.

Moreover, the method has an impressive convergence rate when compared to SEG. As an example, in a specific linear test case (with strong coupling between displacement directions), SEG needs more than 23000 iterations to converge, while NLBC needs only one, just as BC. Furthermore, in all test cases, except one, NLBC exhibits errors with many orders of magnitude smaller than that of produced by SEG;

- The current traction boundary discretization, adopted from BC, restricts the applicability of NLBC, but once a robust discretization is developed, it might be the real opportunity, after four decades of development of FVM methods for solids, to have the long awaited unification: one robust numerical method for both solid and fluid simulations. The common split into "FEM for solid and FVM for fluid" would be surpassed, in particular, with a methodology having the important conservative property when FSI is a must;
- A validated abstract high-level framework for prototyping FVM concepts through a concrete successful implementation, in particular, in the shape of a new CCM toolbox, based on FV. As a Matlab [39] toolbox, nano Finite Volume Method (nFVM) uses the high-level programming environment designed for numerical computations to serve as an investigation and as a prototyping tool, e.g. to experiment, create and test FV methodologies. As far as the author is aware, this is the first time a general CSM solver for Matlab based on the FVM framework is depicted. This new toolbox adopts the tensorial approach in combination with a class-oriented design to provide flexibility when extension is needed. SEG, BC and NLBC are implemented in nFVM and verified against the analytical solutions of several test cases.

## 1.4 Outline of thesis

This document is organized as follows:

- In Chapter 2, the necessary knowledge of continuum mechanics for solid is reviewed. The kinematics and governing equations for structure are discussed followed by the definition of an elastic body. Thereafter, the initial-boundary value problem and the hyperelasticity framework, along with some constitutive equations are presented.
- The SEG and BC methodologies used as reference to build the new approach are given in Chapter 3. The two discretizations are applied to the initial-boundary value problem defined in the previous chapter. The whole presentation also serves as a basis for delineating the NLBC approach, which is shown in the next chapter.

- The Chapter 4 describes the fully coupled cell-centred FV discretization NLBC. The methodology encompasses the mathematical framework for incremental displacement description, momentum equation linearization, and discretization of (surface and body) forces, boundary conditions and system of linear equations. The iterative Newton-Raphson solution algorithm, applied to the new method, is also described. It is also outlined that, by restricting deformation scope to linearized elasticity framework, one can conclude that BC is a particular case of NLBC. In other terms, the latter generalizes the former in some sense.
- The Chapter 5 is devoted to testing NLBC's accuracy and convergence properties. Several representative benchmark test cases are examined for independent testing of different aspects of the new formulation. The results are compared with analytical solutions. Results from SEG and BC are also considered.
- The advantages and disadvantages of having the aforementioned tensorial approach implemented using the high-level programming environment designed for numerical computations are revealed in Chapter 6. The nFVM is also presented. Not only features and characteristics, such as support for mesh format of openFOAM, supported boundary conditions, available FV methods, portability, extendability, generality, usability, performance, but also the nFVM's architecture is shown.
- Finally, summary and outlook are given in Chapter 7.

# 2 Mathematical model

In this chapter, a mathematical model for a continuously deformable elastic solid body, based on continuum mechanics, is described. Thus, the mechanical behavior of such solid body is modeled as a continuous mass rather than as discrete particles and its deformation is governed by an **Elastodynamic** field Equation. Also, a concise introduction of the mathematical framework is presented.

## 2.1 Adopted mathematical framework

A subset of the continuum mechanics framework presented in Ref. [40] is adopted in this thesis with some punctual modifications and additions. Its definitions, theorems, physical laws, governing equations and (initial-)boundary problems are presented next. The main motivation to create this subset was to supply concise necessary and sufficient tools to the development of the new method.

In this work, standard differentiability assumptions sufficient to make an argument rigorous are assumed.

Throughout this chapter (and the next ones) the following adopted conventions are used:

(i) *Lightface* Latin and Greek letters generally denote **scalars**;

(ii) *Boldface lowercase* Latin and Greek letters generally denote **vectors**, with the exception that the letters $\mathbf{o}$, $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ are reserved for **points**;

(iii) *Boldface uppercase* Latin and Greek letters generally denote **second-order tensors**, with the exception that the letters $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$ are reserved for **points**;

(iv) *Boldface calligraphic* letters generally denote **fourth-order tensors**, e.g., $\boldsymbol{\mathcal{C}}$ and $\boldsymbol{\mathcal{M}}$;

(v) The Einstein summation convention is extensively used, especially when writing a tensor in terms of the basis vectors;

(vi) The symbol $\equiv$ is commonly used to introduce simplified notations to improve readability;

(vii) Sometimes it is important to distinguish a tensor of any order, say $\mathbf{x}$, from its coordinates $x_i$ (in a Cartesian frame $\mathbf{e}_i$). The connection between them, for this example, is given by means of the relation $\mathbf{x} = x_i\mathbf{e}_i$. Concerning a second-order tensor $\mathbf{T}$, it can be written as $\mathbf{T} = T_{ij}\mathbf{e}_i \otimes \mathbf{e}_j$ (note that the second-order tensor $\mathbf{e}_i \otimes \mathbf{e}_j$ will be sometimes written as $\mathbf{e}_i\mathbf{e}_j$). The components $T_{ij}$ can be arranged into a two-dimensional array $[\mathbf{T}]$, then the element at the $i$th-position and $j$th-position, along of the two dimensions respectively, is denoted by $[\mathbf{T}]_{ij}$ and, of course, this is equal to $T_{ij}$. High-order tensors are denoted and treated similarly.

## 2.2 Body configuration

Here, it is presented the mathematical description of body and its configuration based on the continuum hypothesis. Deformation, displacement and strain are also elaborated.

The **continuum hypothesis** [40] is assumed here: for any solid, the atomic nature of the material is ignored and it is assumed that it is infinitely divisible. This makes a body consisting of continuously distributed material.

The aforementioned hypothesis allows the identification of a material body with an open subset of the synthetic physical space in which the body lives. Thus, it is postulated that a material **body** occupies an open and bounded subset $B$ of the Euclidean point space $\mathbb{E}^3$ [41, 40]. In particular, each material particle is identified with a point $\mathbb{E}^3$. The subset $B$ is called a **configuration** of the body in $\mathbb{E}^3$ and the following is assumed: (i) the bounding surface $\partial B$ is piecewise smooth; (ii) The $\partial B$ is orientable in the sense that it clearly has two sides. The above assumptions make $B$ into a **regular** region. The Euclidean's associated vector space is denoted by $\mathcal{V}$.

The function $\boldsymbol{\varphi} : B \to B'$, named **deformation map** relative to the reference configuration $B$, which maps each point $\mathbf{X} \in B$ to a point $\mathbf{x} = \boldsymbol{\varphi}(\mathbf{X}) \in B'$ (see Fig. 2.1), describes the deformation of a body from a configuration $B$ onto another configuration $B'$. It is assumed that the function $\boldsymbol{\varphi}$ is one-to-one, $\det \nabla\boldsymbol{\varphi}(\mathbf{X}) > 0$ for all $\mathbf{X} \in B$ and possesses

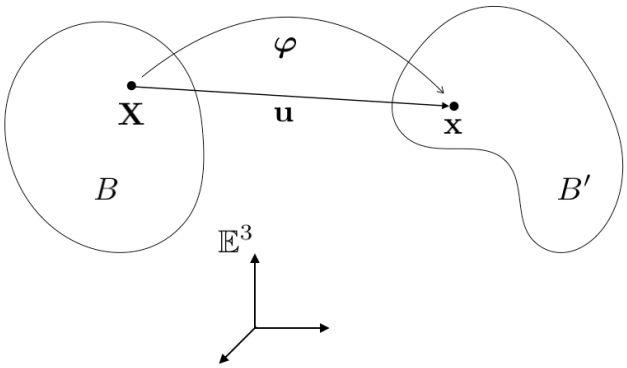



Figure 2.1: Schematic representation of a deformation. The reference configuration $B$ is deformed onto the new configuration $B'$. The black dot in both configurations represents one and the same material particle, $\boldsymbol{\varphi}$ represents the deformation map and $\mathbf{u}$ is the spatial field.

appropriate regularity properties for the ensuing analysis. The displacement of a material particle from its initial location $\mathbf{X}$ to its final location $\mathbf{x}$ is given by

$$\mathbf{U}(\mathbf{X}) = \boldsymbol{\varphi}(\mathbf{X}) - \mathbf{X} \iff U_\alpha = \varphi_i - X_\alpha, \quad \forall \mathbf{X} \in B,\ i = 1,2,3,\ \alpha = 1,2,3. \tag{2.1}$$

The mapping $\mathbf{U} : B \to \mathcal{V}$ is called **displacement field** associated with $\boldsymbol{\varphi}$ (see Fig. 2.1).

The displacement can also be given in terms of a **spatial field**[1] $\mathbf{u} : B' \to \mathcal{V}$ as

$$\mathbf{u}(\mathbf{x}) = \mathbf{x} - \boldsymbol{\psi}(\mathbf{x}) \iff u_i = x_i - \psi_\alpha, \quad \forall \mathbf{X} \in B,\ i = 1,2,3,\ \alpha = 1,2,3. \tag{2.2}$$

where $\boldsymbol{\psi} = \boldsymbol{\varphi}^{-1} : B' \to B$ (this is possible, since $\boldsymbol{\varphi}$ is assumed to be one-to-one). Of course the relation between them is given by

$$\mathbf{u}(\mathbf{x})\Big|_{\mathbf{x}=\boldsymbol{\varphi}(\mathbf{X})} = \mathbf{U}(\mathbf{X}). \tag{2.3}$$

There are many useful tensor fields which can be used to describe measures of deformation and strain. Those used in this thesis are described next. The gradient of $\boldsymbol{\varphi}$, which is a

[1] Any field expressed in terms of points $\mathbf{x}$ of $B'$.

second-order tensor field $\mathbf{F} : B \to \mathcal{V}^2$, is defined by

$$\mathbf{F}(\mathbf{X}) = \frac{\partial \boldsymbol{\varphi}(\mathbf{X})}{\partial \mathbf{X}} = \nabla \boldsymbol{\varphi}(\mathbf{X}) \iff F_{i\alpha} = \frac{\partial \varphi_i}{\partial X_\alpha} \equiv \varphi_{i,\alpha}, \quad \forall \mathbf{X} \in B \tag{2.4}$$

and referred to as the **deformation gradient tensor** field. Because $\mathbf{F}$ provides a complete description of homogeneous local deformation, it is considered to be the primitive measure of deformation. Combining Equations (2.1) and (2.4) results in

$$\mathbf{F}(\mathbf{X}) = \mathbf{I} + \nabla \mathbf{U}(\mathbf{X}) \iff F_{i\alpha} = \delta_{i\alpha} + \frac{\partial U_i}{\partial X_\alpha} = \delta_{i\alpha} + U_{i,\alpha}, \quad \forall \mathbf{X} \in B. \tag{2.5}$$

Other deformation measures, associated with $\mathbf{F}$, are provided by the **right Cauchy-Green deformation tensor** $\mathbf{C} : B \to \mathcal{V}^2$ and the **Green-Lagrange strain tensor** $\mathbf{E} : B \to \mathcal{V}^2$, which are given in terms of $\mathbf{F}$ as

$$\mathbf{C} = \mathbf{F}^T \cdot \mathbf{F}, \quad \mathbf{E} = \frac{1}{2}(\mathbf{F}^T \cdot \mathbf{F} - \mathbf{I}). \tag{2.6}$$

Note that since $\mathbf{C}^{-1} = \mathbf{F}^{-1} \cdot \mathbf{F}^{-T} = (\mathbf{F}^{-1} \cdot \mathbf{F}^{-T})^T = (\mathbf{C}^{-1})^T$, the $\mathbf{C}^{-1}$ is symmetric, i.e.

$$(C^{-1})_{KL} = (C^{-1})_{LK}. \tag{2.7}$$

Also observe that the argument of the tensor fields above were omitted, and it will be often the case from this point forward, to facilitate understanding.

Other important measures are the so-called principal invariants of a second-order tensor. They are defined (and will be used in the hyperelasticity framework later on), e.g. considering $\mathbf{C}$, as

$$\begin{aligned} I_1(\mathbf{C}) &= \text{tr } \mathbf{C} = \mathbf{C} : \mathbf{I} = \lambda_1 + \lambda_2 + \lambda_3 \\ I_2(\mathbf{C}) &= \text{tr } \mathbf{C} \cdot \mathbf{C} = \mathbf{C} : \mathbf{C} = \lambda_1^2 + \lambda_2^2 + \lambda_3^2 \\ I_3(\mathbf{C}) &= \det \mathbf{C} = J^2 = \lambda_1 \lambda_2 \lambda_3, \end{aligned} \tag{2.8}$$

where $\{\lambda_\alpha\}$ are the eigenvalues of $\mathbf{C}$. The derivatives of the principal invariants will also be necessary and are given as [42]

$$\begin{aligned} \frac{\partial I_1(\mathbf{C})}{\partial \mathbf{C}} &= \mathbf{I} \\ \frac{\partial I_2(\mathbf{C})}{\partial \mathbf{C}} &= 2\mathbf{C} \\ \frac{\partial I_3(\mathbf{C})}{\partial \mathbf{C}} &= J^2 \mathbf{C}^{-1}. \end{aligned} \tag{2.9}$$

## 2.3 Body motion

The definition of body motion, a continuous time sequence of displacements, is presented here. This notion is part of the composition of the **Kinematics** theory (the study of motion disregarding the influences of mass, force and stress). Material, spatial and density fields are also defined.

By **motion**, it meant to be understood a continuous deformation of a body over the course of time. Precisely, it is a continuous map

$$\boldsymbol{\varphi} : B \times [0, \infty) \to \mathbb{E}^3, \tag{2.10}$$

such that for each fixed $t \geq 0$ the function

$$\boldsymbol{\varphi}(\cdot, t) = \boldsymbol{\varphi}_t : B \to \mathbb{E}^3 \tag{2.11}$$

is a deformation of $B$. This means that at any time $t \geq 0$ the deformation $\boldsymbol{\varphi}_t$ maps the reference configuration $B$ onto a configuration $B_t = \boldsymbol{\varphi}_t(B)$. The subset $B_t$ is called the **current** or **deformed** configuration at time t.

It is assumed the existence of the identity map $\boldsymbol{\varphi}_0(\mathbf{X}) = \mathbf{X}$ for all $\mathbf{X} \in B$. Of course, this implies that $B_0 = B$. Thus, the continuous deformation of a body initially at configuration $B_0$ is represented by the defined concept of motion.

The **displacement** field associated to the motion is defined by

$$\begin{aligned} \mathbf{U} : B \times [0, \infty) &\to \mathcal{V} \\ (\mathbf{X}, t) &\mapsto \mathbf{U}(\mathbf{X}, t) = \boldsymbol{\varphi}(\mathbf{X}, t) - \mathbf{X}. \end{aligned} \tag{2.12}$$

As motion is already defined, the **velocity** $\mathbf{V}$ and **acceleration** $\mathbf{A}$ of a particle $\mathbf{X} \in B$ at a time $t \geq 0$ are defined indifferently by

$$\begin{aligned} \mathbf{V}(\mathbf{X}, t) &= \dot{\boldsymbol{\varphi}}(\mathbf{X}, t) = \dot{\mathbf{U}}(\mathbf{X}, t) \quad \text{and} \\ \mathbf{A}(\mathbf{X}, t) &= \ddot{\boldsymbol{\varphi}}(\mathbf{X}, t) = \ddot{\mathbf{U}}(\mathbf{X}, t), \end{aligned} \tag{2.13}$$

where the dot represents the material time derivative, i.e. $\dot{\mathbf{U}} = \frac{D\mathbf{U}}{Dt}$ and $\ddot{\mathbf{U}} = \frac{D^2\mathbf{U}}{Dt^2}$. In this particular case, since $\mathbf{U}$ is a **material field**[2], these derivatives reduce to the partial derivatives, i.e.

$$\dot{\mathbf{U}} = \frac{\partial \mathbf{U}}{\partial t} \text{ and } \ddot{\mathbf{U}} = \frac{\partial^2 \mathbf{U}}{\partial t^2}. \tag{2.14}$$

[2] Any field expressed in terms of points $\mathbf{X}$ (which obviously does not depend on $t$) of $B$.

### 2.3.1 Material and spatial fields

Some fields are better described in terms of the current configuration $B_t$ (or deformed configuration $B'$ in case of elastostatics problem) whose point is labeled by $\mathbf{x}$. However, since $\mathbf{x} = \boldsymbol{\varphi}(\mathbf{X}, \text{t})$, any function of $\mathbf{x} \in B_t$ can also be expressed as a function of $\mathbf{X} \in B$. In particular, to any spatial field $\Gamma(\mathbf{x}, t)$ we can associate a material field $\Gamma_m(\mathbf{X}, t)$ by the relation

$$\Gamma_m(\mathbf{X}, t) = \Gamma(\boldsymbol{\varphi}(\mathbf{X}, t), t). \tag{2.15}$$

In this case, $\Gamma_m$ is called **material description** of the spatial field $\Gamma$.

As an example, consider the following. The body has a mass density field per unit volume $\rho(\mathbf{x}, t) > 0$ for all $\mathbf{x} \in B_t$. The associated material field $\rho_m(\mathbf{X}, t) = \rho(\boldsymbol{\varphi}(\mathbf{X}, t), t)$ is related to the reference mass density $\rho_0(\mathbf{X})$ (i.e. the mass density in the reference field) through the relations[3]

$$\rho_m(\mathbf{X}, t) \det \mathbf{F}(\mathbf{X}, t) = \rho(\boldsymbol{\varphi}(\mathbf{X}, t), t) \det \mathbf{F}(\mathbf{X}, t) = \rho(\mathbf{X}, 0) = \rho_0(\mathbf{X}). \tag{2.16}$$

The opposite might also be convenient, i.e. given a material field, it can be associated a spatial field using the inverse of the deformation map $\boldsymbol{\psi}$. In this case, the subscript $m$ is replaced by $s$. The velocity and the acceleration fields are important examples of such construction and they are defined as

$$\begin{aligned} &\mathbf{v}(\mathbf{x}, t) = \mathbf{V}_s(\mathbf{x}, t) = \mathbf{V}(\mathbf{X}, t)\Big|_{\mathbf{X}=\boldsymbol{\psi}(\mathbf{x},t)} \quad \text{and} \\ &\mathbf{a} = \dot{\mathbf{v}} = \frac{\partial \mathbf{v}}{\partial t} + (\nabla \mathbf{v}) \cdot \mathbf{v} \end{aligned} \tag{2.17}$$

respectively.

## 2.4 System of forces

The inclusion of surface and body forces concepts to the Kinematics content allows accounting for the action of external influences to move and change the shape of a body. The first type of force is given by the **traction** or **surface force field**, a vector field $\mathbf{t}_{\hat{\mathbf{n}}} : \Gamma \to \mathcal{V}$,

[3] The conservation of mass law can be used to show this result [40].

such that $\Gamma$ is an arbitrary oriented surface in a configuration $B_t$ ($t \geq 0$) with unit normal field $\hat{\mathbf{n}} : \Gamma \to \mathcal{V}$. It is assumed that the function $\mathbf{t}_{\hat{\mathbf{n}}}$ gives the force, per unit area, exerted by material on one side of the surface upon material on the other side [40].

Now the Cauchy's postulate [40] is assumed in order to define the traction's dependence on surface geometry: the traction field $\mathbf{t}_{\hat{\mathbf{n}}}$ depends only pointwise on $\hat{\mathbf{n}}$. In particular, there exists a function $\mathbf{t} : N \times B_t \to \mathcal{V}$, where $N \subset \mathcal{V}$ denotes the set of all unit vectors, such that $\mathbf{t}_{\hat{\mathbf{n}}}(\mathbf{x}) = \mathbf{t}(\hat{\mathbf{n}}(\mathbf{x}), \mathbf{x})$

The **Cauchy-Poisson theorem** [43], one of the major results of continuum mechanics, says that $\mathbf{t}_{\hat{\mathbf{n}}}(\mathbf{x})$ is linear in $\hat{\mathbf{n}}$, that is, for each $\mathbf{x} \in B_t$ there is a second-order tensor $\boldsymbol{\sigma}(\mathbf{x}) \in \mathcal{V}^2$ such that

$$\mathbf{t}_{\hat{\mathbf{n}}}(\mathbf{x}) = \boldsymbol{\sigma}(\mathbf{x}) \cdot \hat{\mathbf{n}}. \tag{2.18}$$

This field is called **Cauchy stress tensor** field.

These allow the definition of arbitrary forces acting on parts or on the whole of the boundary surface $\Gamma = \partial B_t$ of $B_t$, accounting for external forces, thus also for traction boundary condition definition.

To couple with the second type of force, the concept of **body force** is used. It is defined here as a force vector and represented using the function $\mathbf{b} : B_t \to \mathcal{V}$.

## 2.5 Governing equations

A discussion about the equations that govern the motion and state of the body are presented here. Besides, the first and the second Piola-Kirchhoff stress tensor fields are defined, and then used in the Lagrangian form of the linear momentum equation, which is also described.

Irrespective of material properties, there are five fundamental laws (or principles) which govern the motion and state of the continuum bodies [44], they are: i) mass conservation; ii) angular momentum conservation; iii) linear momentum conservation; iv) energy conservation; and v) entropy inequality. In this study, only isothermal modeling of continuum bodies is considered. Thus, the energy conservation and the entropy inequality laws will be ignored, because under these circumstances they decouple from the mass, angular momentum and linear momentum conservation equations [40]. Moreover, the mass conservation law is also disregarded because the Lagrangian framework is adopted (instead of the Eulerian one as it is customary in fluid dynamics) [40].

In other words, it is postulated only the existence of the fundamental (linear and angular) momentum balance principles for the whole or arbitrary parts of a continuum body $B$.

It is assumed that the second-order Cauchy stress tensor field

$$\begin{aligned} \boldsymbol{\sigma} : B \times [0, \infty) &\to \mathcal{V}^2 \\ (\mathbf{x}, t) &\mapsto \boldsymbol{\sigma}(\mathbf{x}, t) \end{aligned} \tag{2.19}$$

is symmetric in order to satisfy the conservation of angular momentum law [40], i.e.

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}^T. \tag{2.20}$$

Therefore, the only law directly of interest is the linear momentum law. It is possible to use the Eulerian form of the linear momentum equation:

$$\rho \dot{\mathbf{v}} = \nabla \cdot \boldsymbol{\sigma} + \rho \mathbf{b}, \quad \text{in } B_t, t \geq 0, \tag{2.21}$$

where $\dot{\mathbf{v}}$ is the acceleration field as in Eq. $(2.17)_2$ and $\mathbf{b}(\mathbf{x}, t)$ in the spatial body force per unit mass. However, since the Lagrangian version of linear momentum Equation (2.21) is of interest in this work, it is also necessary to use the **first Piola-Kirchhoff stress** field $\mathbf{P}$. Its definition is given in terms of $\boldsymbol{\sigma}_m$ as

$$\mathbf{P}(\mathbf{X}, t) = (\det \mathbf{F}(\mathbf{X}, t)) \boldsymbol{\sigma}_m(\mathbf{X}, t) \cdot \mathbf{F}(\mathbf{X}, t)^{-T}, \quad \forall \mathbf{X} \in B, \mathbf{F} \in \mathcal{V}^2, \det \mathbf{F} > 0. \tag{2.22}$$

The other important stress measure, the **second Piola-Kirchhoff stress** field, can then be defined in terms of $\mathbf{P}$ as

$$\boldsymbol{\Sigma}(\mathbf{X}, t) = \mathbf{F}(\mathbf{X}, t)^{-1} \cdot \mathbf{P}(\mathbf{X}, t), \quad \forall \mathbf{X} \in B, \mathbf{F} \in \mathcal{V}^2, \det \mathbf{F} > 0. \tag{2.23}$$

Let $\rho_0(\mathbf{X})$ be the reference mass density field as defined in (2.16), $\mathbf{b}_m(\mathbf{X}, t)$ the reference body force field (see Eq. 2.15) and $\mathbf{P}(\mathbf{X}, t)$ the Piola-Kirchhoff stress tensor field defined above, then the balance of linear momentum in Lagrangian form requires

$$\rho_0 \ddot{\mathbf{U}} = \nabla \cdot \mathbf{P} + \rho_0 \mathbf{b}_m \quad \text{in } B, t \geq 0. \tag{2.24}$$

## 2.6 Elastic body

An elastic body can be defined through the following **elastic body axiom** [40]: a continuum body with reference configuration $B$ is elastic if $\exists\ \widehat{\boldsymbol{\sigma}} : \mathcal{V}^2 \times B \to \mathcal{V}^2$ such that

$$\begin{aligned} &\boldsymbol{\sigma}_m(\mathbf{X}, t) = \widehat{\boldsymbol{\sigma}}(\mathbf{F}(\mathbf{X}, t), \mathbf{X}), \quad \forall \mathbf{X} \in B, t \geq 0 \quad \text{and} \\ &\widehat{\boldsymbol{\sigma}}(\mathbf{F}, \mathbf{X})^T = \widehat{\boldsymbol{\sigma}}(\mathbf{F}, \mathbf{X}), \quad \forall \mathbf{X} \in B, \mathbf{F} \in \mathcal{V}^2, \det \mathbf{F} > 0. \end{aligned} \tag{2.25}$$

Since this work considers only homogeneous bodies, the **stress response function** $\widehat{\boldsymbol{\sigma}}$ is considered independent of $\mathbf{X}$, thus $\boldsymbol{\sigma}_m(\mathbf{X},t) = \widehat{\boldsymbol{\sigma}}(\mathbf{F}(\mathbf{X},t))$. Because of this axiom, there are two functions $\widehat{\mathbf{P}} : \mathcal{V}^2 \to \mathcal{V}^2$ and $\widehat{\boldsymbol{\Sigma}} : \mathcal{V}^2 \to \mathcal{V}^2$ such that

$$\mathbf{P}(\mathbf{X},t) = \widehat{\mathbf{P}}(\mathbf{F}(\mathbf{X},t)) \quad \text{and} \quad \boldsymbol{\Sigma}(\mathbf{X},t) = \widehat{\boldsymbol{\Sigma}}(\mathbf{F}(\mathbf{X},t)), \tag{2.26}$$

in particular, they must satisfy the relations

$$\widehat{\mathbf{P}}(\mathbf{F}) = (\det \mathbf{F})\widehat{\boldsymbol{\sigma}}(\mathbf{F}) \cdot \mathbf{F}^{-T} \quad \text{and} \quad \widehat{\boldsymbol{\Sigma}}(\mathbf{F}) = \mathbf{F}^{-1} \cdot \widehat{\mathbf{P}}(\mathbf{F}). \tag{2.27}$$

There is an axiom in physics which asserts that: any "observable quantity", e.g. any quantity with intrinsic character, such as mass density, an acceleration vector, etc., must be independent of the particular orthogonal basis in which it is computed. The **axiom of material frame-indifference**, a tailored version (applied to elastic materials) of this general axiom, implies that: $\exists\, \overline{\boldsymbol{\Sigma}} : \mathcal{V}^2 \to \mathcal{V}^2$ such that

$$\widehat{\mathbf{P}}(\mathbf{F}) = \mathbf{F} \cdot \overline{\boldsymbol{\Sigma}}(\mathbf{C}) \quad \text{and} \quad \widehat{\boldsymbol{\Sigma}}(\mathbf{F}) = \overline{\boldsymbol{\Sigma}}(\mathbf{C}). \tag{2.28}$$

Let $\mathbf{A} \in \mathcal{V}^2$ and assume the Definition (2.6), then

$$\frac{\partial \mathbf{C}}{\partial \mathbf{F}} : \mathbf{A} = \mathbf{A}^T \cdot \mathbf{F} + \mathbf{F}^T \cdot \mathbf{A}, \tag{2.29}$$

thus (using the chain rule and $\partial\mathbf{F}/\partial\nabla\mathbf{U} = \mathbf{I}$, i.e. the fourth-order identity tensor [4])

$$\begin{aligned} \frac{\partial \mathbf{C}}{\partial \nabla\mathbf{U}} : \mathbf{A} &= \frac{\partial \mathbf{C}}{\partial \mathbf{F}} : \left( \frac{\partial \mathbf{F}}{\partial \nabla\mathbf{U}} : \mathbf{A} \right) = \frac{\partial \mathbf{C}}{\partial \mathbf{F}} : \left( \mathbf{I} : \mathbf{A} \right) = \frac{\partial \mathbf{C}}{\partial \mathbf{F}} : \mathbf{A} \\ &= \mathbf{A}^T \cdot \mathbf{F} + \mathbf{F}^T \cdot \mathbf{A} \equiv 2 \cdot \mathrm{sym}(\mathbf{F}^T \cdot \mathbf{A}), \end{aligned} \tag{2.30}$$

where $\mathrm{sym}(\cdot)$ denotes the symmetric component of a tensor [5]. The field $\partial\mathbf{C}/\partial\nabla\mathbf{U}$ is used next.

The Green-Lagrange strain tensor $\mathbf{E} = \frac{1}{2}(\mathbf{C} - \mathbf{I})$ and the new function $\breve{\boldsymbol{\Sigma}}(\mathbf{E}) = \overline{\boldsymbol{\Sigma}}(\mathbf{C}(\mathbf{E}))$ can be used to find the **elasticity tensor** $\boldsymbol{\mathcal{C}} = \partial\, \breve{\boldsymbol{\Sigma}}\, /\partial\mathbf{E}$ in terms of $\partial\overline{\boldsymbol{\Sigma}}/\partial\mathbf{C}$ as

$$\begin{aligned} \frac{\partial\, \breve{\boldsymbol{\Sigma}}}{\partial \mathbf{E}} : \mathbf{A} &= \frac{\partial \overline{\boldsymbol{\Sigma}}}{\partial \mathbf{C}} : \left( \frac{\partial \mathbf{C}}{\partial \mathbf{E}} : \mathbf{A} \right) \qquad \text{(using again the chain rule)} \\ &= \frac{\partial \overline{\boldsymbol{\Sigma}}}{\partial \mathbf{C}} : \left( 2\mathbf{I} : \mathbf{A} \right) \\ &= 2\frac{\partial \overline{\boldsymbol{\Sigma}}}{\partial \mathbf{C}} : \mathbf{A}. \end{aligned} \tag{2.31}$$

[4] The definition is $\mathbf{I} \equiv \delta_{ac}\delta_{bd}\mathbf{e}_a \otimes \mathbf{e}_b \otimes \mathbf{e}_c \otimes \mathbf{e}_d$ which implies that $\mathbf{I} : \mathbf{A} = \mathbf{A}$.

[5] The last equation shows that $\mathrm{sym}(\mathbf{B}) \equiv \frac{1}{2}(\mathbf{B} + \mathbf{B}^T)$.

The arbitrariness of $\mathbf{A}$ implies that

$$\boldsymbol{\mathcal{C}} = \frac{\partial \breve{\boldsymbol{\Sigma}}}{\partial \mathbf{E}} = 2\frac{\partial \overline{\boldsymbol{\Sigma}}}{\partial \mathbf{C}}. \tag{2.32}$$

Using Equation (2.28) and the definition of the two new functions $\widetilde{\mathbf{P}}(\nabla\mathbf{U}) = \widehat{\mathbf{P}}(\mathbf{F}(\nabla\mathbf{U})) = \widehat{\mathbf{P}}(\mathbf{I}+\nabla\mathbf{U})$ and $\widetilde{\boldsymbol{\Sigma}}(\nabla\mathbf{U}) = \overline{\boldsymbol{\Sigma}}(\mathbf{C}(\nabla\mathbf{U}))$, the derivative of the stress response function $\widetilde{\mathbf{P}}$ is given as

$$\begin{aligned}
\frac{\partial \widetilde{\mathbf{P}}}{\partial \nabla\mathbf{U}} : \mathbf{A} &= \frac{\partial(\mathbf{F}\cdot\widetilde{\boldsymbol{\Sigma}})}{\partial\nabla\mathbf{U}} : \mathbf{A} \qquad \text{(using Equation (2.28))} \\
&= \left(\frac{\partial \mathbf{F}}{\partial\nabla\mathbf{U}} : \mathbf{A}\right)\cdot\widetilde{\boldsymbol{\Sigma}} + \mathbf{F}\cdot\left(\frac{\partial\widetilde{\boldsymbol{\Sigma}}}{\partial\nabla\mathbf{U}} : \mathbf{A}\right) \\
&= \mathbf{A}\cdot\widetilde{\boldsymbol{\Sigma}} + \mathbf{F}\cdot\left(\frac{\partial\widetilde{\boldsymbol{\Sigma}}}{\partial\nabla\mathbf{U}} : \mathbf{A}\right) \qquad \text{(using } \partial\mathbf{F}/\partial\nabla\mathbf{U} = \mathbf{I}) \\
&= \mathbf{A}\cdot\widetilde{\boldsymbol{\Sigma}} + \mathbf{F}\cdot\left[\frac{\partial\overline{\boldsymbol{\Sigma}}}{\partial\mathbf{C}} : \left(\frac{\partial\mathbf{C}}{\partial\nabla\mathbf{U}} : \mathbf{A}\right)\right] \qquad \text{(using the chain rule)} \\
&= \mathbf{A}\cdot\widetilde{\boldsymbol{\Sigma}} + \mathbf{F}\cdot\left[\frac{\partial\overline{\boldsymbol{\Sigma}}}{\partial\mathbf{C}} : \Big(2\cdot\mathrm{sym}(\mathbf{F}^T\cdot\mathbf{A})\Big)\right] \qquad \text{(using Eq. (2.30))} \\
&= \mathbf{A}\cdot\widetilde{\boldsymbol{\Sigma}} + \mathbf{F}\cdot\Big[\boldsymbol{\mathcal{C}} : \Big(\mathrm{sym}(\mathbf{F}^T\cdot\mathbf{A})\Big)\Big] \qquad \text{(using Eq. (2.32))}.
\end{aligned} \tag{2.33}$$

## 2.7 Initial-boundary value problem

The initial-boundary value problem for an elastic body is a set of equations for determining the motion and deformation of a given body subject to specified initial conditions in $B$ at time $t = 0$, and boundary conditions on $\partial B$ at times $t \geq 0$.

A standard initial-boundary value problem for an elastic body with reference configuration $B$ is the following: find the displacement function $\mathbf{U} : B \times [0,T] \to \mathcal{V}$ such that

$$\boxed{\begin{array}{ll}
\rho_0\ddot{\mathbf{U}} = \nabla\cdot\widetilde{\mathbf{P}}(\nabla\mathbf{U}) + \rho_0\mathbf{b}_m & \text{in } B \times [0,T] \\
\mathbf{U} = \overline{\mathbf{U}} & \text{in } \Gamma_U \times [0,T] \\
\widetilde{\mathbf{P}}(\nabla\mathbf{U})\cdot\mathbf{N} = \overline{\mathbf{T}} & \text{in } \Gamma_T \times [0,T] \\
\mathbf{U}(\cdot,0) = \mathbf{0} & \text{in } B \\
\dot{\mathbf{U}}(\cdot,0) = \dot{\mathbf{U}}_0 & \text{in } B.
\end{array}} \tag{2.34}$$

In the above system, $\Gamma_U$ and $\Gamma_T$ are subsets of $\partial B$ with the properties $\Gamma_T \cup \Gamma_U = \partial B$, and $\Gamma_T \cap \Gamma_U = \varnothing$, $\rho_0$ is the reference mass density, $\mathbf{b}_m$ is the material description of a spatial body force field per unit mass, $\mathbf{N}$ is the unit outward normal field on $\partial B$, and $\overline{\mathbf{U}}$, $\overline{\mathbf{T}}$ and $\dot{\mathbf{U}}_0$ are prescribed fields. Equation $(2.34)_1$ is the balance of linear momentum equation, $(2.34)_2$ is a displacement boundary condition on $\Gamma_U$, $(2.34)_3$ is a traction boundary condition on $\Gamma_T$, $(2.34)_4$ is an initial condition for the displacement $\mathbf{U}$ and $(2.34)_5$ is an initial condition for the material velocity field $\dot{\boldsymbol{\varphi}} = \dot{\mathbf{U}}$. The initial conditions should be compatible with the boundary conditions at time $t = 0$. This formulation is called **elastodynamics problem**.

It is important to note that, differently from the one given in [40], this non-linear initial-boundary value problem uses $\widetilde{\mathbf{P}}(\nabla\mathbf{U})$ and $\mathbf{U}$ instead $\widehat{\mathbf{P}}(\mathbf{F})$ and $\boldsymbol{\varphi}$. This is crucial in order to allow the linearization of $\widetilde{\mathbf{P}}(\nabla\mathbf{U})$ applied by the new algorithm (done in Chapter 4).

## 2.8 Boundary value problem

When time-dependent behaviour is ignored, the above problem turns into an **elastostatics problem** and it is simplified. In particular, the inertial $\rho_0\ddot{\mathbf{U}}$ and body force $\rho_0\mathbf{b}_m$ terms are excluded from the linear momentum equation. Of course initial boundary conditions are also ignored. And in this case, the problem is: find the displacement function $\mathbf{U} : B \to \mathcal{V}$ such that

$$\boxed{\begin{array}{ll} \nabla \cdot \widetilde{\mathbf{P}}(\nabla\mathbf{U}) = \mathbf{0} & \text{in } B \\ \mathbf{U} = \overline{\mathbf{U}} & \text{in } \Gamma_U \\ \widetilde{\mathbf{P}}(\nabla\mathbf{U}) \cdot \mathbf{N} = \overline{\mathbf{T}} & \text{in } \Gamma_T. \end{array}} \tag{2.35}$$

The meaning of boundary conditions is analogous to that in the previous problem.

## 2.9 Constitutive equations

The vector Equation (2.24) do not completely determine the first Piola-Kirchhoff stress tensor field $\mathbf{P}$ for a body in equilibrium. There are three partial differential equations in $(2.24)$ and three independent algebraic equations in (2.20) with which to determine the nine components of $\mathbf{P}$. This issue is addressed by adding a **constitutive equation** which characterizes the specific material properties of a body.

In this section the Hookean and the hyperelastic models Neo-Hookean and Ogden are presented. Only **compressible forms** of all material laws are considered.

### 2.9.1 Hyperelasticity

Hyperelasticity is a framework created to deal with large deformation and one of its pillars is purely elastic models. This work is concerned precisely with such models. The use of a **strain-energy function** (or **elastic potential**) $W$ (a stored energy per unit in the reference configuration) is offered by this framework to simulate the model. The definition for $W$ is given as

$$\begin{aligned} W : B \times \mathcal{V}^2 &\to \mathbb{R} \\ (\mathbf{X}, \mathbf{F}) &\mapsto W(\mathbf{X}, \mathbf{F}(\mathbf{X})). \end{aligned} \tag{2.36}$$

For every homogeneous material model, the strain energy $W$ depends only on the deformation $\mathbf{F}$. Furthermore, it is postulated that the first Piola-Kirchhoff stress tensor field $\mathbf{P}$ is given as

$$\mathbf{P}(\mathbf{X}) = \frac{\partial W(\mathbf{F}(\mathbf{X}))}{\partial \mathbf{F}} \iff P_{i\alpha} = \frac{\partial W}{\partial F_{i\alpha}} = W_{,i\alpha}, \quad \forall \mathbf{X} \in B. \tag{2.37}$$

Another implication of the axiom of material frame-indifference is: $\exists\ \overline{W} \mid W(\mathbf{F}) = \overline{W}(\mathbf{F}^T \cdot \mathbf{F}) = \overline{W}(\mathbf{C})$ [45]. In this case the partial derivative of this strain-energy function gives the second Piola-Kirchhoff stress tensor field $\mathbf{\Sigma}$ as

$$\mathbf{\Sigma}(\mathbf{X}) = 2\frac{\partial \overline{W}(\mathbf{C})}{\partial \mathbf{C}}, \quad \forall \mathbf{X} \in B. \tag{2.38}$$

To account for material isotropy, another restriction is added: “the constitutive behaviour is identical in any material direction” [42]. This implies that the strain-energy function $W$ must be independent of the material axes chosen and, consequently, $W$ must only be a function of the principal invariants of $\mathbf{C}$, in the sense that there exist a function $\widehat{W} : \mathbb{R}^3 \to \mathbb{R}$ such that

$$W(\mathbf{F}) = \overline{W}(\mathbf{C}) = \widehat{W}(I_1(\mathbf{C}), I_2(\mathbf{C}), I_3(\mathbf{C})) \equiv \widehat{W}(\mathcal{I}_{\mathbf{C}}). \tag{2.39}$$

The second Piola-Kirchhoff tensor $\mathbf{\Sigma}$ can be rewritten, as a result of the isotropic restriction from Equation (2.38) as

$$\begin{aligned}\mathbf{\Sigma}(\mathbf{X}) &= 2\frac{\partial \overline{W}(\mathbf{C})}{\partial \mathbf{C}} \\ &= 2\frac{\partial \widehat{W}(\mathcal{I}_{\mathbf{C}})}{\partial \mathbf{C}} \\ &= 2\frac{\partial \widehat{W}(\mathcal{I}_{\mathbf{C}})}{\partial I_1}\frac{\partial I_1(\mathbf{C})}{\partial \mathbf{C}} + 2\frac{\partial \widehat{W}(\mathcal{I}_{\mathbf{C}})}{\partial I_2}\frac{\partial I_2(\mathbf{C})}{\partial \mathbf{C}} + 2\frac{\partial \widehat{W}(\mathcal{I}_{\mathbf{C}})}{\partial I_3}\frac{\partial I_3(\mathbf{C})}{\partial \mathbf{C}}, \quad \forall \mathbf{X} \in B.\end{aligned} \tag{2.40}$$

Introducing Expressions (2.9) into Equation (2.40) allows the second Piola-Kichhof stress tensor field to be evaluated as

$$\mathbf{\Sigma}(\mathbf{X}) = 2\frac{\partial \widehat{W}(\mathcal{I}_{\mathbf{C}})}{\partial I_1}\mathbf{I} + 4\frac{\partial \widehat{W}(\mathcal{I}_{\mathbf{C}})}{\partial I_2}\mathbf{C} + 2J^2\frac{\partial \widehat{W}(\mathcal{I}_{\mathbf{C}})}{\partial I_3}\mathbf{C}^{-1}, \quad \forall \mathbf{X} \in B. \tag{2.41}$$

Because of the relations between principal invariants and eigenvalues (Eq. 2.8), and Equation (2.39), a function $\breve{W} : \mathbb{R}^3 \to \mathbb{R}$ can be defined such that

$$W(\mathbf{F}) = \breve{W}(\lambda_1, \lambda_2, \lambda_3), \tag{2.42}$$

where the set $\{\lambda_\alpha \,|\, \alpha = 1, 2, 3\}$ are the eigenvalues of $\mathbf{C}$.

Knowing this relation, one can define a strain energy function only in terms of the "most fundamental" measure of the stretches $\{\lambda_\alpha\}$. In this case, the expression for the second Piola-Kirchhoff stress tensor $\mathbf{\Sigma}(\mathbf{X})$ is [42]

$$\mathbf{\Sigma}(\mathbf{X}) = \sum_{\alpha=1}^{3} \breve{\Sigma}_{\alpha\alpha}\mathbf{N}_\alpha \otimes \mathbf{N}_\alpha, \quad \forall \mathbf{X} \in B, \tag{2.43}$$

where $\{\mathbf{N}_\alpha\}$ are the eigenvectors of $\mathbf{C}$ and $\breve{\Sigma}_{\alpha\alpha}$ is equal to

$$\breve{\Sigma}_{\alpha\alpha} = 2\frac{\partial \breve{W}}{\partial \lambda_\alpha}. \tag{2.44}$$

The elasticity tensor $\mathcal{C}$ for a material given in terms of $\breve{W}$ is [42]

$$\mathcal{C} = \sum_{\alpha,\beta=1}^{3} 4\frac{\partial^2 \breve{W}}{\partial \lambda_\alpha \partial \lambda_\beta}\boldsymbol{\mathcal{N}}_{\alpha\alpha\beta\beta} + \sum_{\alpha,\beta=1,\alpha\neq\beta}^{3} \frac{\breve{\Sigma}_{\alpha\alpha} - \breve{\Sigma}_{\beta\beta}}{\lambda_\alpha - \lambda_\beta}(\boldsymbol{\mathcal{N}}_{\alpha\beta\alpha\beta} + \boldsymbol{\mathcal{N}}_{\alpha\beta\beta\alpha}), \tag{2.45}$$

where

$$\begin{aligned}\boldsymbol{\mathcal{N}}_{\alpha\alpha\beta\beta} &= \mathbf{N}_\alpha \otimes \mathbf{N}_\alpha \otimes \mathbf{N}_\beta \otimes \mathbf{N}_\beta;\\ \boldsymbol{\mathcal{N}}_{\alpha\beta\alpha\beta} &= \mathbf{N}_\alpha \otimes \mathbf{N}_\beta \otimes \mathbf{N}_\alpha \otimes \mathbf{N}_\beta;\\ \boldsymbol{\mathcal{N}}_{\alpha\beta\beta\alpha} &= \mathbf{N}_\alpha \otimes \mathbf{N}_\beta \otimes \mathbf{N}_\beta \otimes \mathbf{N}_\alpha.\end{aligned} \tag{2.46}$$

When $\lambda_\alpha = \lambda_\beta$ the expression

$$\frac{\breve{\Sigma}_{\alpha\alpha} - \breve{\Sigma}_{\beta\beta}}{\lambda_\alpha - \lambda_\beta} \tag{2.47}$$

is substituted by

$$2\left(\frac{\partial^2 \widetilde{W}}{\partial\lambda_\beta \partial\lambda_\beta} - \frac{\partial^2 \widetilde{W}}{\partial\lambda_\alpha \partial\lambda_\beta}\right) \tag{2.48}$$

(see [42]).

**Proposition:** Every hyperelastic, homogeneous, frame-indifferent and isotropic model has a right-minor symmetric elasticity tensor (this is important for Chapter 4).

**Proof:** noting that

$$\begin{aligned}[\boldsymbol{\mathcal{N}}_{\alpha\alpha\beta\beta}]_{IJKL} &= [\mathbf{N}_\alpha \otimes \mathbf{N}_\alpha \otimes \mathbf{N}_\beta \otimes \mathbf{N}_\beta]_{IJKL}\\ &= [\mathbf{N}_\alpha]_I [\mathbf{N}_\alpha]_J [\mathbf{N}_\beta]_K [\mathbf{N}_\beta]_L\\ &= [\mathbf{N}_\alpha]_I [\mathbf{N}_\alpha]_J [\mathbf{N}_\beta]_L [\mathbf{N}_\beta]_K\\ &= [\mathbf{N}_\alpha \otimes \mathbf{N}_\alpha \otimes \mathbf{N}_\beta \otimes \mathbf{N}_\beta]_{IJLK}\\ &= [\boldsymbol{\mathcal{N}}_{\alpha\alpha\beta\beta}]_{IJLK}\end{aligned} \tag{2.49}$$

and

$$\begin{aligned}[\boldsymbol{\mathcal{N}}_{\alpha\beta\alpha\beta}]_{IJKL} &= [\mathbf{N}_\alpha \otimes \mathbf{N}_\beta \otimes \mathbf{N}_\alpha \otimes \mathbf{N}_\beta]_{IJKL}\\ &= [\mathbf{N}_\alpha]_I [\mathbf{N}_\beta]_J [\mathbf{N}_\alpha]_K [\mathbf{N}_\beta]_L\\ &= [\mathbf{N}_\alpha]_I [\mathbf{N}_\beta]_J [\mathbf{N}_\beta]_L [\mathbf{N}_\alpha]_K\\ &= [\mathbf{N}_\alpha \otimes \mathbf{N}_\beta \otimes \mathbf{N}_\beta \otimes \mathbf{N}_\alpha]_{IJLK}\\ &= [\boldsymbol{\mathcal{N}}_{\alpha\beta\beta\alpha}]_{IJLK}\end{aligned} \tag{2.50}$$

it is now straightforward to verify (using 2.45) that

$$[\boldsymbol{\mathcal{C}}]_{IJKL} = [\boldsymbol{\mathcal{C}}]_{IJLK}, \tag{2.51}$$

i.e. the elasticity tensor has the right-minor symmetry property:

$$
\begin{aligned}
[\boldsymbol{\mathcal{C}}]_{IJKL} &= \sum_{\alpha,\beta=1}^{3} 4\frac{\partial^2 \widetilde{W}}{\partial\lambda_\alpha \partial\lambda_\beta}[\boldsymbol{\mathcal{N}}_{\alpha\alpha\beta\beta}]_{IJKL} \\
&\quad + \sum_{\alpha,\beta=1,\alpha\neq\beta}^{3} \frac{\widetilde{\Sigma}_{\alpha\alpha} - \widetilde{\Sigma}_{\beta\beta}}{\lambda_\alpha - \lambda_\beta}([\boldsymbol{\mathcal{N}}_{\alpha\beta\alpha\beta}]_{IJKL} + [\boldsymbol{\mathcal{N}}_{\alpha\beta\beta\alpha}]_{IJKL}) \\
&= \sum_{\alpha,\beta=1}^{3} 4\frac{\partial^2 \widetilde{W}}{\partial\lambda_\alpha \partial\lambda_\beta}[\boldsymbol{\mathcal{N}}_{\alpha\alpha\beta\beta}]_{IJLK} \\
&\quad + \sum_{\alpha,\beta=1,\alpha\neq\beta}^{3} \frac{\widetilde{\Sigma}_{\alpha\alpha} - \widetilde{\Sigma}_{\beta\beta}}{\lambda_\alpha - \lambda_\beta}([\boldsymbol{\mathcal{N}}_{\alpha\beta\beta\alpha}]_{IJLK} + [\boldsymbol{\mathcal{N}}_{\alpha\beta\alpha\beta}]_{IJLK}) \quad \text{(Eq. 2.49/2.50)} \\
&= \sum_{\alpha,\beta=1}^{3} 4\frac{\partial^2 \widetilde{W}}{\partial\lambda_\alpha \partial\lambda_\beta}[\boldsymbol{\mathcal{N}}_{\alpha\alpha\beta\beta}]_{IJLK} \\
&\quad + \sum_{\alpha,\beta=1,\alpha\neq\beta}^{3} \frac{\widetilde{\Sigma}_{\alpha\alpha} - \widetilde{\Sigma}_{\beta\beta}}{\lambda_\alpha - \lambda_\beta}([\boldsymbol{\mathcal{N}}_{\alpha\beta\alpha\beta}]_{IJLK} + [\boldsymbol{\mathcal{N}}_{\alpha\beta\beta\alpha}]_{IJLK}) \quad \text{(commutativity of addition)} \\
&= [\boldsymbol{\mathcal{C}}]_{IJLK}. \quad \blacksquare
\end{aligned}
\tag{2.52}
$$

The discretization of the new FVM, i.e. NLBC, requires a new tensor, say $\mathbf{T}^d$ $(d = 1, 2, 3)$, which is a function of another new quantity called **transformed elasticity tensor** $\boldsymbol{\mathcal{M}} = \mathbf{F} \cdot \underset{(3)}{\boldsymbol{\mathcal{C}}} \cdot \mathbf{F}^T$ (the operator $\underset{(3)}{\cdot}$ is a contraction at the third index) and the face normal $\mathbf{N}$, and is defined as

$$
\begin{aligned}
\mathbf{T}^d &= \mathcal{M}_{aJdL} N_J \mathbf{e}_a \otimes \mathbf{e}_L \\
&= F_{aI}\mathcal{C}_{IJKL}F_{dK}N_J \mathbf{e}_a \otimes \mathbf{e}_L \\
&= F_{aI}\mathcal{C}_{IJKL}f^d_K N_J \mathbf{e}_a \otimes \mathbf{e}_L \qquad (\mathbf{f}^d = f^d_K \mathbf{e}_K = F_{dK}\mathbf{e}_K) \\
&\equiv (\mathbf{F} \cdot \boldsymbol{\mathcal{C}}) \underset{(2,3)}{:} (\mathbf{N} \otimes \mathbf{f}^d).
\end{aligned}
\tag{2.53}
$$

The full expression for $\mathbf{T}^d$ depends of course on the material model and it is shown next.

### 2.9.2 Neo-Hookean model

This compressible isotropic hyperelastic material model is able to describe the stress-strain response for a moderate strain [42, 46]. Its strain-energy function is defined as

$$\overline{W}(\mathbf{C}) = \widehat{W}(\mathcal{I}_{\mathbf{C}}) = \frac{\mu}{2}(I_1(\mathbf{C}) - 3) - \mu \ln J + \frac{\lambda}{2}(\ln J)^2, \tag{2.54}$$

where $I_3(\mathbf{C}) = \det \mathbf{C} = J^2$. The Lamé (material) coefficients $\lambda$ and $\mu$ relating to the Young's modulus $E$ and Poisson's ratio, $\nu$, are given respectively as: $\mu = \dfrac{E}{2(1+\nu)}$; $\dfrac{\nu E}{(1+\nu)(1-\nu)}$ for plane stress; and $\dfrac{\nu E}{(1+\nu)(1-2\nu)}$ for plane strain and 3-D. The second Piola-Kirchhoff stress tensor is obtained from Equation (2.41) as

$$\mathbf{\Sigma} = \overline{\mathbf{\Sigma}}(\mathbf{C}) = \mu(\mathbf{I} - \mathbf{C}^{-1}) + \lambda(\ln J)\mathbf{C}^{-1}. \tag{2.55}$$

The elasticity tensor can be obtained by differentiation of Equation (2.55) with respect to the components of $\mathbf{E}$ to give, after some algebra using $\partial I_3(\mathbf{C})/\partial \mathbf{C} = J^2\mathbf{C}^{-1}$, $\boldsymbol{\mathcal{C}}$ as

$$\boldsymbol{\mathcal{C}} = \frac{\partial \breve{\mathbf{\Sigma}}}{\partial \mathbf{E}} = 2\frac{\partial \overline{\mathbf{\Sigma}}}{\partial \mathbf{C}} = \lambda \mathbf{C}^{-1} \otimes \mathbf{C}^{-1} + 2(\mu - \lambda \ln J)\boldsymbol{\mathcal{J}}, \tag{2.56}$$

where the fourth-order tensor $\boldsymbol{\mathcal{J}}$ is defined as

$$\boldsymbol{\mathcal{J}} = -\frac{\partial \mathbf{C}^{-1}}{\partial \mathbf{C}} \iff \mathcal{J}_{IJKL} = \frac{1}{2}\Big[(C^{-1})_{IK}(C^{-1})_{JL} + (C^{-1})_{IL}(C^{-1})_{JK}\Big]. \tag{2.57}$$

It is straightforward to show that $\boldsymbol{\mathcal{J}}$, $\mathbf{C}^{-1} \otimes \mathbf{C}^{-1}$ (using Eq. 2.7) and therefore the elasticity tensor $\boldsymbol{\mathcal{C}}$ above has right-minor symmetry, i.e.

$$\mathcal{C}_{IJKL} = \mathcal{C}_{IJLK}. \tag{2.58}$$

The full expression for $\mathbf{T}^d$ is obtained by substituting Equation (2.56) into Equation (2.53) resulting in

$$\begin{aligned}
\mathbf{T}^d &= \lambda(\mathbf{F}\cdot\mathbf{C}^{-1}\cdot\mathbf{N}) \otimes (\mathbf{f}^d \cdot \mathbf{C}^{-1}) \\
&\quad + (\mu - \lambda \ln J)\Big[(\mathbf{F}\cdot\mathbf{C}^{-1}\cdot\mathbf{f}^d) \otimes (\mathbf{N}\cdot\mathbf{C}^{-1}) + (\mathbf{N}\cdot\mathbf{C}^{-1}\cdot\mathbf{f}^d)(\mathbf{F}\cdot\mathbf{C}^{-1})\Big] \\
&= \lambda(\mathbf{A}\cdot\mathbf{N}) \otimes (\mathbf{f}^d \cdot \mathbf{C}^{-1}) + (\mu - \lambda \ln J)\Big[(\mathbf{A}\cdot\mathbf{f}^d) \otimes \mathbf{b} + (\mathbf{b}\cdot\mathbf{f}^d)\mathbf{A}\Big] \\
&= \lambda(\mathbf{A}\cdot\mathbf{N}) \otimes (\mathbf{C}^{-1}\cdot\mathbf{f}^d) + (\mu - \lambda \ln J)\Big[(\mathbf{A}\cdot\mathbf{f}^d) \otimes \mathbf{b} + (\mathbf{b}\cdot\mathbf{f}^d)\mathbf{A}\Big] \qquad (\mathbf{C}^{-1} \text{ is symmetric}),
\end{aligned} \tag{2.59}$$

where $\mathbf{A} = \mathbf{F}\cdot\mathbf{C}^{-1}$ and $\mathbf{b} = \mathbf{N}\cdot\mathbf{C}^{-1} = \mathbf{C}^{-1}\cdot\mathbf{N}$.

### 2.9.3 Ogden-Storakers model

The following strain-energy function

$$\widetilde{W}(\lambda_1, \lambda_2, \lambda_3) = \sum_{i=1}^{q} \frac{\mu_i}{2\alpha_i^2} \left[ \lambda_1^{\alpha_i} + \lambda_2^{\alpha_i} + \lambda_3^{\alpha_i} - 3 + \frac{1}{\beta_i}(J^{-\alpha_i\beta_i} - 1) \right], \tag{2.60}$$

where $\{\lambda_\alpha\}$ are the eigenvalues of $\mathbf{C}$ and the values $q$, $\{\alpha_i\}$ and $\{\beta_i\}$ are disposable, has been proposed by [47] and [48] for describing the mechanical behaviour of highly compressible polymers. The value $q \in \{1, 2, 3\}$ being suitable in general to simulate real materials [33, 34]. The second Piola-Kirchhoff stress tensor $\mathbf{\Sigma}(\mathbf{X})$ for this material is computed using

$$\breve{\Sigma}_{ll} = 2\frac{\partial \widetilde{W}}{\partial \lambda_l} = \frac{1}{\lambda_l} \sum_{i=1}^{q} \frac{\mu_i}{\alpha_i} \left[ \lambda_l^{\alpha_i} - (J)^{-\alpha_i\beta_i} \right] \tag{2.61}$$

and Equation (2.43).

The elasticity tensor $\boldsymbol{\mathcal{C}}$ is given by Equation (2.45), thus the tensor $\mathbf{T}^d$ (see Eq. 2.53) for this model can be easily computed by noting that

$$(\mathbf{F} \cdot \boldsymbol{\mathcal{N}}_{\alpha\beta\gamma\delta}) \underset{(2,3)}{:} (\mathbf{N} \otimes \mathbf{f}^d) = (\mathbf{N}_\beta \cdot \mathbf{N})(\mathbf{N}_\gamma \cdot \mathbf{f}^d) \Big[ (\mathbf{F} \cdot \mathbf{N}_\alpha) \otimes \mathbf{N}_\delta \Big], \tag{2.62}$$

where $\mathbf{N}$ is the face normal and $\mathbf{f}^d = f_K^d \mathbf{e}_K = F_{dK}\mathbf{e}_K$ as seen before.

# 3 Finite Volume Methods

The preceding chapter presented a mathematical model, without connection with numerical methods, for a general elastic body. The task of this chapter is to present two discretizations of the proposed initial-boundary value problem (Box 2.34), which determines the motion and deformation of a general elastic body. Two FV discretization methods are presented next: SEG and BC. Almost all the content of this chapter provides a basis for developing the non-linear Block-Coupled (NLBC), which is done in the next chapter. Both presentations are based on [36, 35].

The numerical methods SEG and BC were the major inspiration sources to create the new method, NLBC, which tries to preserve the most relevant aspect of each one without importing their major weakness. The SEG is a very flexible method of discretization, in the sense that it does not constraint the constitutive equations and can be used in both non-linear and linear elasticity. But, its major drawback is that it can be very slow for many test cases, in particular, whenever there is a strong coupling between displacement components [36]. The method of discretization BC has shown itself to be much faster than SEG for those strongly coupling test cases [1]. Furthermore, the BC solver by Cardiff et. al [36] resulted in less execution time and memory requirements than a finite element software for the set of cases tested[2]. Nevertheless, the current BC formulation is tied to only one constitutive equation and linear elasticity [36].

A FV method is a discretization process which transforms one or more partial differential equations into a corresponding system of algebraic equations. The solution of the system produces a set of values that correspond to the solution of the original equations at some pre-determined locations in space and time, provided certain conditions are satisfied. The conditions for segregated can be seen in [49], but for BC there is no literature available yet. Future studies can focus on finding conditions for NLBC, since this topic will not be discussed in the current work.

[1] By a factor of 2.5-6 times [36].

[2] In fact it was almost 6 times faster and used 8 times less memory.

The FV methodologies that are presented in this thesis directly discretize the strong integral form of the governing equation (2.24), i.e. the discretization occurs directly in the physical space (subdivided into finite volumes). The stress term in the governing equation is turned into face fluxes and evaluated at the finite volume faces. The flux entering a given volume is made identical to that leaving the adjacent volume, generating their main advantage: strictly conservative, i.e. conservation of transported variable is guaranteed. Thus the presented FV methodologies can easily incorporate coupling, e.g. to simulate fluid-structure interaction. For a rather complete overview of FVM for solid mechanics analysis, for example, see [35, 50].

## 3.1 Solution domain discretization

The starting point for a FV discretization is to decompose the solution domain $B \times [0, T]$ (or only $B$ in case of non-transient simulation). The solution spatial domain $B$ is usually approximated by arbitrary and finite number $n_C$ of contiguous convex polyhedral cells (also known as finite volume) $\Omega_C$'s bounded by faces that do not overlap. But this thesis adopts a specific polyhedral: the rectangular cuboid. The reason for choosing rectangular cuboids is to avoid non-conformal (skewed and/or non-conjunctional) mesh [51] and the complexities that arise from it. This way, investigation efforts focus only on the "core" (i.e. minimal structure to be fully usable) of the NLBC algorithm. Non-essential extensions can be added to NLBC after an extensive investigation of the core.

The approximation mentioned above is written as

$$B \approx B_d = \bigcup_{C=1}^{n_C} \Omega_C, \tag{3.1}$$

i.e. the continuous body $B$ is approximated by the computational domain $B_d$ which is the union of $n_C$ cells. The Figure 3.1 shows (for two-dimensional case) the configuration of one cell $\Omega_C \in B_d$.

The cell's centroid, the computational node, stores the values of the displacement vector as well as the physical properties of the material (e.g. $\rho_0$). Because of the storage location, this discretization is classified as cell-centered (an alternative is vertex-centered) [51].

Concerning the solution temporal domain $[0, T]$ (for transient simulation), the total specified simulation time $T$ is divided into a finite number of time increments (of fixed size $\Delta t$) and the discretized governing equations are solved using time stepping procedure.

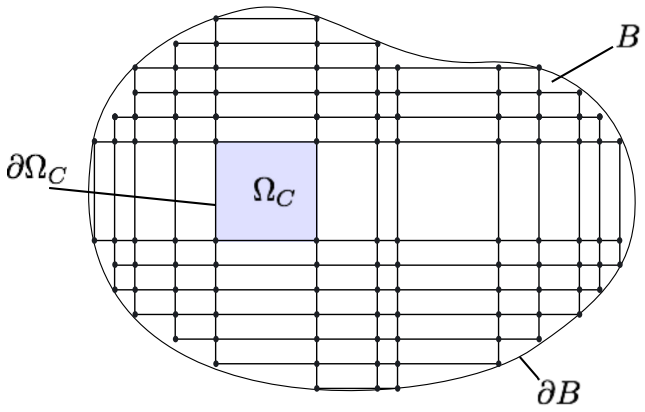



Figure 3.1: Discretization of a body $B$ into cells $\Omega_C$'s. Every cell $\Omega_C$ has a boundary $\partial\Omega_C$. Note that because of the rectangular cuboid restriction, the boundary domain $\partial B$ is approximated in a castellated *staircase* manner.

## 3.2 Discretization of governing equation

Only the linear momentum equation needs to be discretized, as discussed in the previous chapter. The discretization process of this equation can be seen as a transformation pipeline (see Fig. 3.2) such that each step converts one set of equations into another, the first set composed by the differential equation (see Box 2.34)

$$\rho_0 \ddot{\mathbf{U}} = \nabla \cdot \widetilde{\mathbf{P}} + \rho_0 \tag{3.2}$$

and the last set having the algebraic equations forming the final linear system where the unknowns are the displacements at the nodes.

To make things clearer, in the first step the Equation (3.2), which is valid for the whole discretized domain $B_d = \cup_{C=1}^{n_C} \Omega_C$, is integrated for each finite volume $\Omega_C$, producing a set of integral equations. Following this step, after applying some transformations, the set of integral equations are converted into a set of coupled (by face values[3]) equations, say, Discrete Algebraic Relations. Finally, interpolation profiles are used to express each face value as a function of cell's centroid values. The Figure (3.2) illustrates the block diagram for the discretization process.

[3]The set of **face values** is composed by variables stored at the centroid of the faces. The definition of the set of **volume values** is analogous.

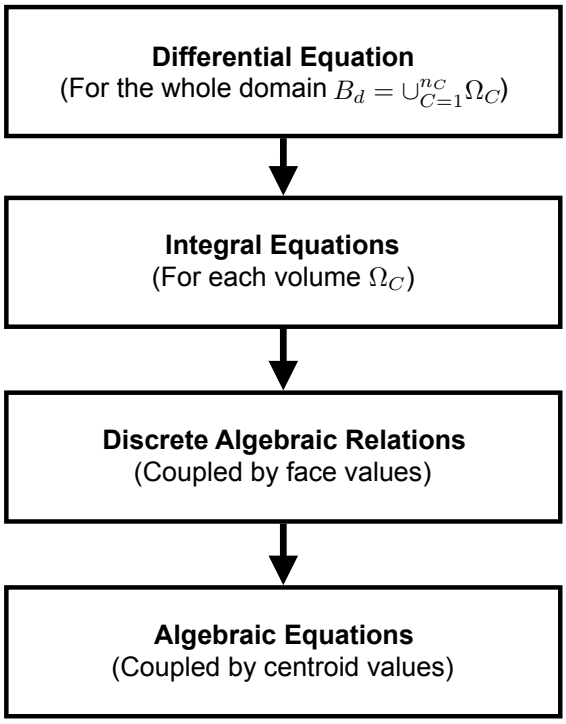



Figure 3.2: Block diagram illustrating the discretization processes of the FV methods SEG, BC and NLBC.

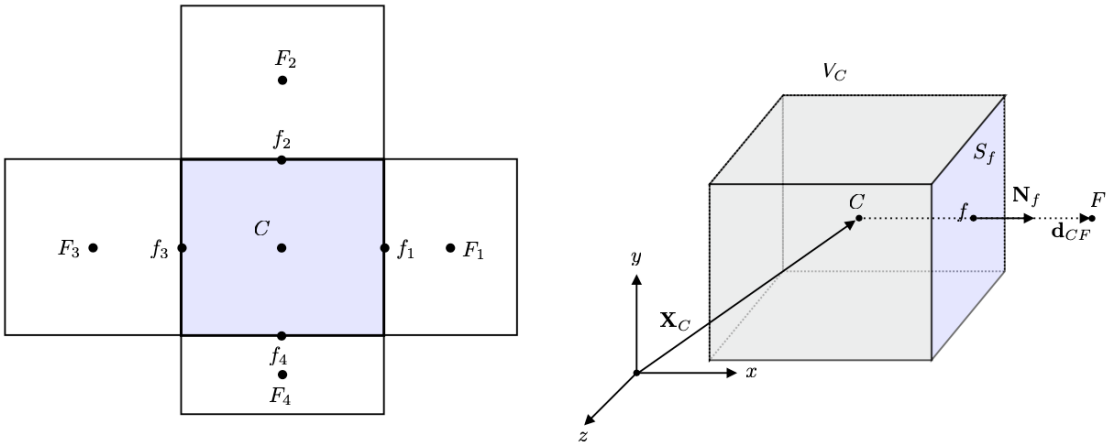



Figure 3.3: a) A cell $\Omega_C$, with centroid $C$ of a 2D discretized domain, and its neighbours $F_i$; b) A cell $\Omega_C$ with its geometric parameters used in the finite volume discretization. In style of [52].

Before proceeding, note the geometric parameters shown in the Figures (3.3a-b) which are needed in the FV discretization process of the governing equations. The Figure (3.3a) shows a cell with its neighbours $F$'s and their face centroids $f$'s. The other image (Fig. 3.3b) exemplifies a typical cuboid cell $\Omega_C$, having volume $V_C$ and the centroid, or computational node, located at the point C.

### 3.2.1 Integral Equations

In the first step of the discretization process, the conservation law (3.2) is applied to each finite volume $\Omega_C \in B_d$, rendering

$$\underbrace{\int_{\Omega_C} \rho_0 \ddot{\mathbf{U}}\, dV}_{\text{Inertial force}} = \underbrace{\int_{\Omega_C} \nabla \cdot \widetilde{\mathbf{P}}\, dV}_{\text{Surface force}} + \underbrace{\int_{\Omega_C} \rho_0 \mathbf{b}_m\, dV}_{\text{Body force}} \quad \forall\, \Omega_C \in B_d. \tag{3.3}$$

The next step is to use the divergence theorem to replace the volume integral of the surface force term by surface integral, thus the equation above becomes

$$\int_{\Omega_C} \rho_0 \ddot{\mathbf{U}}\, dV = \oint_{\partial\Omega_C} \widetilde{\mathbf{P}} \cdot \mathbf{N}\, dS + \int_{\Omega_C} \rho_0 \mathbf{b}_m\, dV \quad \forall\, \Omega_C \in B_d, \tag{3.4}$$

where $\mathbf{N}$ is the unit outward normal field on $\partial\Omega_C$. Because the FV is a polyhedral with flat faces, the equation above can be converted into a sum of integrals over all faces to become

$$\int_{\Omega_C} \rho_0 \ddot{\mathbf{U}}\, dV = \sum_{\Gamma_f \in \partial\Omega_C} \left( \int_{\Gamma_f} \widetilde{\mathbf{P}} \cdot \mathbf{N}\, dS \right) + \int_{\Omega_C} \rho_0 \mathbf{b}_m\, dV \quad \forall\, \Omega_C \in B_d, \tag{3.5}$$

where $\partial\Omega_C = \bigcup_f \Gamma_f$ such that $\Gamma_f$ is a flat face.

### 3.2.2 Discrete Algebraic Equations

This section describes the process of transforming the Equation (3.5) into the corresponding discrete algebraic relation, for each cell $\Omega_C \in B_d$, by means of eliminating surface and volume integrals using mid-point rule (1-point Gauss integration approximation). The relations that make up the resulting set of this process are coupled by face values (which are not the primary unknowns). This is the reason why another step is necessary in order to write these values, by assuming interpolation profiles, as a function of volume values (where primary unknowns “live”).

The discretization of the inertial and the body force terms are fully described in this section, since they are the same for all approaches (SEG, BC and NLBC). This is not the case for the surface force term though. The complete surface force term discretization is different for each method and it will be presented only partially here, i.e. only up to the point right before their discretization differs from each other. The missing part for SEG and BC will be presented in the next sessions, but that for NLBC will be presented later on, in the next chapter.

Essentially, mid-point integration approximation and Taylor expansions are used to express the inertial force (see Equation 3.3) with the aid of volume values at different times. The starting point is to use the mid-point rule to eliminate the volume integral as:

$$\int_{\Omega_C} \rho_0 \ddot{\mathbf{U}}\, dV \approx \rho_0(\mathbf{X_C})\ddot{\mathbf{U}}(\mathbf{X_C}, t)V_C \equiv \left(\rho_0 \ddot{\mathbf{U}} V\right)_{C,t} \equiv \rho_{0,C}\ddot{\mathbf{U}}_C^t V_C, \tag{3.6}$$

where $V_C$ is the volume of a cell $\Omega_C$ and $\mathbf{X_C}$ is the position of cell's centroid.

In order to calculate $\ddot{\mathbf{U}}$, a first-order backward fully implicit Euler scheme[4] is derived by expressing the values of $\mathbf{U}$ at times $t - \Delta t$, $t - 2\Delta t$ in terms of its value and the value of its derivative at time $t$ using Taylor series as

$$\mathbf{U}(\mathbf{X}, t - \Delta t) = \mathbf{U}(\mathbf{X}, t) - \Delta t\dot{\mathbf{U}}(\mathbf{X}, t) + \frac{\Delta t^2}{2}\ddot{\mathbf{U}}(\mathbf{X}, t) + O(\Delta t^3) \tag{3.7}$$

and

$$\mathbf{U}(\mathbf{X}, t - 2\Delta t) = \mathbf{U}(\mathbf{X}, t) - 2\Delta t\dot{\mathbf{U}}(\mathbf{X}, t) + \frac{(2\Delta t)^2}{2}\ddot{\mathbf{U}}(\mathbf{X}, t) + O(\Delta t^3). \tag{3.8}$$

Multiplying Equation (3.7) by $-2$ and subtracting the resulting equation from Equation (3.8), a first order representation of the second derivative is obtained as

$$\ddot{\mathbf{U}}(\mathbf{X}, t) = \frac{1}{\Delta t^2}\Big[\mathbf{U}(\mathbf{X}, t) - 2\mathbf{U}(\mathbf{X}, t - \Delta t) + \mathbf{U}(\mathbf{X}, t - 2\Delta t)\Big]. \tag{3.9}$$

Thus, considering the times $t_i$, $t_{i-1} = t_i - \Delta t$, $t_{i-2} = t_i - 2\Delta t$ and a cell $\Omega_C$, the final expression for the inertial force term is calculated using the first-order backward fully implicit Euler scheme above as

$$\int_{\Omega_C} \rho_0 \ddot{\mathbf{U}}\, dV \approx \left[\frac{\rho_0 V}{\Delta t^2}\left(\mathbf{U}^{t_i} - 2\mathbf{U}^{t_{i-1}} + \mathbf{U}^{t_{i-2}}\right)\right]_C. \tag{3.10}$$

[4] Actually, any appropriate finite difference scheme may be used [35].

### Body force

Also adopting the mid-point rule and considering a time $t_i$, the body force term (see Equation 3.3) is computed as

$$\int_{\Omega_C} \rho_0 \mathbf{b}_m \, dV \approx \rho_0(\mathbf{X_C})\mathbf{b}_m(\mathbf{X_C}, t_i)V_C \equiv \left[\rho_0 \mathbf{b}_m V\right]_{C,t_i} \equiv \rho_{0,C}\mathbf{b}^{t_i}_{m,C}V_C. \tag{3.11}$$

### Surface force term

The surface force term (see Equation 3.3) discretization also uses the mid-point rule, but this time to approximate surface integral instead of volume integral:

$$\begin{aligned}
\sum_{\Gamma_f \in \partial\Omega_C} \left[\int_{\Gamma_f} \widetilde{\mathbf{P}} \cdot \mathbf{N} \, dS\right] &\approx \sum_{f \in centroids(\partial\Omega_C)} \left[\widetilde{\mathbf{P}}(\nabla \mathbf{U}(\mathbf{X}_f, t_i)) \cdot \mathbf{N}(\mathbf{X}_f)\right] S_f \\
&\equiv \sum_f \left[\widetilde{\mathbf{P}}(\nabla \mathbf{U}(\mathbf{X}_f, t_i)) \cdot \mathbf{N}(\mathbf{X}_f)\right] S_f \\
&\equiv \sum_f \left[\widetilde{\mathbf{P}} \cdot \mathbf{N}\right]_{f,t_i} S_f \\
&\equiv \sum_f \left[\widetilde{\mathbf{P}} \cdot \mathbf{S}\right]_{f,t_i},
\end{aligned} \tag{3.12}$$

where $centroids(\partial\Omega_C)$ means the set having the centroids of all faces in $\partial\Omega_C = \bigcup_f \Gamma_f$, surface area field $\mathbf{S} = S\mathbf{N}$ and $t_i$ is a given time. To go beyond this point, it is necessary to choose a FV method.

Considering a cell $\Omega_C$ and using Equations (3.10)-(3.12), the final equation to be solved (hiding the prefix $C$) is

$$\frac{\rho_0 V}{\Delta t^2}\left[\mathbf{U}^{t_i} - 2\mathbf{U}^{t_{i-1}} + \mathbf{U}^{t_{i-2}}\right] = \sum_f \left[\widetilde{\mathbf{P}} \cdot \mathbf{S}\right]_{f,t_i} + \left[\rho_0 \mathbf{b}_m V\right]_{t_i}. \tag{3.13}$$

### 3.2.3 Segregated FV methodology

The current approach [32] to simulate general constitutive relations using the SEG methodology is to split the surface force term (see Equation 3.3) into implicit and explicit components, i.e. the Equation (3.13) becomes

$$\frac{\rho_0 V}{\Delta t^2}\left[\mathbf{U}^{t_i} - 2\mathbf{U}^{t_{i-1}} + \mathbf{U}^{t_{i-2}}\right] = \underbrace{\sum_f \left[\mathbf{Q_P} S\right]_{f,t_i}}_{\text{implicit}} + \underbrace{\sum_f \left[\widetilde{\mathbf{P}}\cdot\mathbf{S} - \mathbf{Q_P} S\right]_{f,t_i}}_{\text{explicit}} + \left[\rho_0 \mathbf{b}_m V\right]_{t_i}, \tag{3.14}$$

where $\mathbf{Q_P}$ approximates the traction field $\mathbf{Q} = \widetilde{\mathbf{P}}\cdot\mathbf{N}$, **implicit** indicates contribution to the matrix of the resulting discretized algebraic linear system and **explicit** indicates contribution to the source vector of the linear system. In the case of linear elastic law, the optimal approximation for $\mathbf{Q}$ is clear (see [32]), i.e. for Hooke's law

$$\widetilde{\mathbf{P}} = \mu\nabla\mathbf{U} + \mu\nabla\mathbf{U}^T + \lambda\,(\text{tr}\,\nabla\mathbf{U})\mathbf{I}, \tag{3.15}$$

where $\mu$ and $\lambda$ are the Lamé coefficients, is

$$\mathbf{Q_P} = K\nabla\mathbf{U}\cdot\mathbf{N}. \tag{3.16}$$

such that the so-called **implicit stiffness** $K$ is equal to $2\mu + \lambda$. For non-linear elasticity[5], the **scalar** $K$ can be obtained, e.g. using some invariant from the elasticity tensor $\boldsymbol{\mathcal{C}}$ [32].

Regardless the choice for $K$, $\mathbf{Q}$ is discretized on a face $\Gamma_f$ at a time $t_i$ using the standard central differencing as follows:

$$\begin{aligned}\left[\mathbf{Q_P}\right]_{f,t_i} &= \left[K\nabla\mathbf{U}\cdot\mathbf{N}\right]_{f,t_i} \\ &\approx \left[K\left(\frac{\mathbf{U}_C - \mathbf{U}_F}{|\mathbf{d}_{CF}|}\right)\right]_{f,t_i} \\ &\equiv K_f^{t_i}\left(\frac{\mathbf{U}_C^{t_i} - \mathbf{U}_F^{t_i}}{|\mathbf{d}_{CF}|}\right),\end{aligned} \tag{3.17}$$

where the vector connecting the centroids of the cells sharing the common face is $\mathbf{d}_{CF} = \mathbf{X}_F - \mathbf{X}_C$ (Fig. 3.3).

[5] Actually, the non-linear term, although quite common, is misleading, since elasticity cannot be linear [45]

To include boundary conditions, the surface force term discretization at the boundary faces is modified. In the case of a Neumann boundary condition, the specified traction $\overline{\mathbf{T}}$ is directly substituted in the surface stress expression $\widetilde{\mathbf{P}} \cdot \mathbf{N}$ (see Box $(2.34)_3$), while in the case of Dirichlet boundary condition, the face displacement gradients are calculated at the face using the specified displacement $\overline{\mathbf{U}}$, i.e. this value is directly used in Equation (3.17). Finally, the current available displacement values are used to calculate the displacement gradient by means of either the Green-Gauss or the Least Square methods [51], which in turns is used to evaluate the explicit component.

Given a material law, assembling Equation (3.14) using the procedure just described produces the following linear algebraic equation:

$$a_C \mathbf{U}_C^{t_i} + \sum_F a_F \mathbf{U}_F^{t_i} = \mathbf{R}_C \tag{3.18}$$

with one equation assembled for each finite volume $C$ and

$$\begin{aligned} a_T &= \frac{\rho_0 V}{\Delta t^2}, \\ a_F &= \frac{K_f^{t_i} S_f}{|\mathbf{d}_{CF}|}, \\ a_C &= a_T - \sum_F a_F, \\ \mathbf{R}_C &= a_T \left( 2\mathbf{U}_C^{t_{i-1}} - \mathbf{U}_C^{t_{i-2}} \right) + \sum_f \left[ \widetilde{\mathbf{P}} \cdot \mathbf{S} - \mathbf{Q}_{\mathbf{P}} S \right]_{f,t_i} + \left[ \rho_0 \mathbf{b}_m V \right]_{t_i}. \end{aligned} \tag{3.19}$$

The terms $K_f$ and $\left[\widetilde{\mathbf{P}} \cdot \mathbf{S} - \mathbf{Q}_{\mathbf{P}} S\right]_f$ are evaluated using the current available values of $\mathbf{U}$ and the linear interpolation profile:

$$[Y]_f = f_x [Y]_C + (1 - f_x)[Y]_F \quad f_x = \frac{|\mathbf{X}_C - \mathbf{X}_f|}{|\mathbf{d}_{CF}|}, Y \in \{K, \widetilde{\mathbf{P}} \cdot \mathbf{S} - \mathbf{Q}_{\mathbf{P}} S\}. \tag{3.20}$$

Because $\mathbf{U}_C^{t_i}$ depends on the values in the neighbouring cells, the following system of algebraic equations is created:

$$[A][\mathbf{U}] = [\mathbf{R}], \tag{3.21}$$

where $[A]$ is a sparse matrix, with coefficients $a_C$ on the diagonal and $a_F$ off the diagonal, $[\mathbf{U}]$ is the vector of $\mathbf{U}$'s for all FV's and $[\mathbf{R}]$ is the right-hand side vector. The above system is solved consecutively for three components of $\mathbf{U}$, i.e. in a segregated manner. It is to be

noted that $\mathbf{R}_C$ contains contribution from the explicit component, which creates coupling between displacement components, implying that an iterative solution (Picard/Fixed-Point) is needed to provide the coupling. Each such iteration is called **correction** and the number of corrections $n_{\text{corr}}$ needed to conform with a chosen residual tolerance is case-dependent.

The linear system (3.21) can be solved using an iterative solver, such as the incomplete Cholesky pre-conditioned conjugate gradient method (ICCG), or directly using Gaussian elimination or LU decomposition [36].

### 3.2.4 Block-coupled FV methodology

The BC method employs a solution procedure where all the three displacement components are simultaneously solved in a large block system, which resembles the traditional Finite Element Methods. Besides, it employs a complete linearization of the surface force in terms of the unknowns, i.e. fully implicit (see Eq. 3.14), which has shown significant speedups over the segregated methods for inter-component coupling cases [36]. In practical terms, the surface traction ($\mathbf{Q} = \widetilde{\mathbf{P}} \cdot \mathbf{N}$) is decomposed into normal ($\mathbf{Q}_n$) and tangential ($\mathbf{Q}_t$) components:

$$
\begin{aligned}
\sum_f \left[\widetilde{\mathbf{P}} \cdot \mathbf{S}\right]_{f,t_i} &= \sum_f S_f \left[\widetilde{\mathbf{P}} \cdot \mathbf{N}\right]_{f,t_i} \\
&= \sum_f S_f \left[\mathbf{I} \cdot \widetilde{\mathbf{P}} \cdot \mathbf{N}\right]_{f,t_i} \\
&= \sum_f S_f \left[\mathbf{I} \cdot \mathbf{Q}\right]_{f,t_i} \\
&= \sum_f S_f \left[(\mathbf{NN} + \mathbf{I} - \mathbf{NN}) \cdot \mathbf{Q}\right]_{f,t_i} \\
&= \sum_f S_f \left[\underbrace{\mathbf{NN} \cdot \mathbf{Q}}_{\mathbf{Q}_n} + \underbrace{(\mathbf{I} - \mathbf{NN}) \cdot \mathbf{Q}}_{\mathbf{Q}_t}\right]_{f,t_i} \\
&= \sum_f S_f \left[\underbrace{(2\mu + \lambda)\nabla \mathbf{U}_n \cdot \mathbf{N} + \lambda(\text{tr}\, \nabla_t \mathbf{U}_t)\mathbf{N}}_{\mathbf{Q}_n} + \underbrace{\mu \nabla \mathbf{U}_t \cdot \mathbf{N} + \mu \nabla_t U_n}_{\mathbf{Q}_t}\right]_{f,t_i},
\end{aligned}
\tag{3.22}
$$

where $\mathbf{NN} = \mathbf{N} \otimes \mathbf{N}$, $\mathbf{U}_n = \mathbf{NN} \cdot \mathbf{U}$, $U_n = \mathbf{N} \cdot \mathbf{U}$, $\mathbf{U}_t = (\mathbf{I} - \mathbf{NN}) \cdot \mathbf{U}$ and $\nabla_t = (\mathbf{I} - \mathbf{NN}) \cdot \nabla$ is a tangential derivative. The expressions for $\mathbf{Q}_n$ and $\mathbf{Q}_t$ are derived in [36]. The discretizations of normal and face tangential derivatives, $(\nabla[\ ] \cdot \mathbf{N})$ and $(\nabla_t[\ ])$, from the equation above are shown next by considering a cell $C$, any of its face $\Gamma_f$ and a time $t_i$.

### Normal derivative terms

The normal derivative terms $(\nabla[\ ] \cdot \mathbf{N})$ on a face $\Gamma_f$ are discretized using the standard central differencing as follows:

$$\begin{aligned} \Big[\nabla \mathbf{U}_n \cdot \mathbf{N}\Big]_{f,t_i} &\approx \left[\left(\frac{\mathbf{U}_n^C - \mathbf{U}_n^F}{|\mathbf{d}_{CF}|}\right)\right]_{f,t_i} \\ &\approx \left[\left(\frac{(\mathbf{NN} \cdot \mathbf{U})^C - (\mathbf{NN} \cdot \mathbf{U})^F}{|\mathbf{d}_{CF}|}\right)\right]_{f,t_i} \\ &\equiv \left[\mathbf{NN} \cdot \left(\frac{\mathbf{U}^C - \mathbf{U}^F}{|\mathbf{d}_{CF}|}\right)\right]_{f,t_i}; \end{aligned} \tag{3.23}$$

And analogously,

$$\Big[\nabla \mathbf{U}_t \cdot \mathbf{N}\Big]_{f,t_i} \approx \left[(\mathbf{I} - \mathbf{NN}) \cdot \left(\frac{\mathbf{U}^C - \mathbf{U}^F}{|\mathbf{d}_{CF}|}\right)\right]_{f,t_i}, \tag{3.24}$$

where the vector connecting the centroids of the cells sharing the common face $\mathbf{d}_{CF} = \mathbf{X}_F - \mathbf{X}_C$ (Fig. 3.3).

### Tangential derivative terms

The face tangential derivative $\nabla_t U_n$ is approximated using a face-Gauss/Finite Area method [36]:

$$\begin{aligned}\Big[\nabla_t U_n\Big]_{f,t_i} &\approx \frac{1}{S_f}\Big[\sum_{e\,\in\, edges(\Gamma_f)} (U_n)_e L_e \mathbf{M}_e\Big]_{t_i} \\ &= \frac{1}{S_f}\Big[\sum_{e\,\in\, edges(\Gamma_f)} (\mathbf{N}_f \cdot \mathbf{U})_e L_e \mathbf{M}_e\Big]_{t_i} \\ &= \frac{1}{S_f}\Big[\sum_{e\,\in\, edges(\Gamma_f)} L_e \mathbf{M}_e \mathbf{N}_f \cdot \mathbf{U}_e\Big]_{t_i} \\ &\equiv \frac{1}{S_f}\Big[\sum_{e} L_e \mathbf{M}_e \mathbf{N} \cdot \mathbf{U}_e\Big]_{f,t_i},\end{aligned} \tag{3.25}$$

where $edges(\Gamma_f)$ refers to the set of all edges enclosing the face $\Gamma_f$, $(U_n)_e$ is the edge-centre displacement component, $L_e$ is the length of the edge $e$ and $\mathbf{M}_e$ is the edge-centre unit bi-normal vector calculated using the cross product between $\hat{\mathbf{e}}$ (the unit vector parallel with the edge $e$) and the face normal $\mathbf{N}_f$, i.e.

$$\mathbf{M}_e = \hat{\mathbf{e}} \times \mathbf{N}_f. \tag{3.26}$$

The edge-centre displacement $\mathbf{U}_e$ is approximated using the average of the edge $e$ end-points as follows:

$$\mathbf{U}_e \approx \frac{1}{2}\Big[\mathbf{U}_{ep1} + \mathbf{U}_{ep2}\Big], \tag{3.27}$$

where the edge end-point displacements $(U_n)_{epi}$ are approximated in terms of the neighbouring cell-centre values using a weighted least squares interpolation.

$$\mathbf{U}_{ep} \approx \sum_{pc} w_{pc} \mathbf{U}^{pc}, \tag{3.28}$$

where $w_{pc}$ is a weighting factor. Using these interpolations in equation (3.25), the face tangential derivative $\nabla_t U_n$ is approximated as

$$\begin{aligned}\left[\nabla_t U_n\right]_{f,t_i} &\approx \frac{1}{S_f}\left[\sum_e L_e \mathbf{M}_e \mathbf{N} \cdot \left(\frac{1}{2}\sum_{pc} w_{pc}\mathbf{U}^{pc}\right)\right]_{f,t_i} \\ &= \frac{1}{S_f}\left[\sum_e \sum_{pc} \frac{1}{2} L_e w_{pc} \mathbf{M}_e \mathbf{N} \cdot \mathbf{U}^{pc}\right]_{f,t_i} \\ &\equiv \frac{1}{S_f}\left[\sum_{e,pc} \frac{1}{2} L_e w_{pc} \mathbf{M}_e \mathbf{N} \cdot \mathbf{U}^{pc}\right]_{f,t_i}.\end{aligned} \tag{3.29}$$

The other face tangential derivative $\nabla_t \mathbf{U}_t$ is approximated as

$$\left[\nabla_t \mathbf{U}_t\right]_{f,t_i} \approx \frac{1}{S_f}\left[\sum_e (\mathbf{U}_t)_e \mathbf{M}_e L_e\right]_{f,t_i}. \tag{3.30}$$

The term $(\text{tr}\ \nabla_t \mathbf{U}_t)\mathbf{N}$ can now be computed using the above expression as follows:

$$\begin{aligned}\left[(\text{tr}\ \nabla_t \mathbf{U}_t)\mathbf{N}\right]_{f,t_i} &\approx \left(\text{tr}\ \frac{1}{S_f}\left[\sum_e (\mathbf{U}_t)_e \mathbf{M}_e L_e\right]_{f,t_i}\right)\mathbf{N}_f \\ &= \frac{1}{S_f}\left[\sum_e \left[\text{tr}\ (\mathbf{U}_t)_e \mathbf{M}_e\right] L_e\right]_{f,t_i} \mathbf{N}_f \\ &= \frac{1}{S_f}\left[\sum_e \mathbf{M}_e \cdot \mathbf{U}_e L_e\right]_{f,t_i} \mathbf{N}_f \\ &= \frac{1}{S_f}\left[\sum_e L_e \mathbf{N}\mathbf{M}_e \cdot \mathbf{U}_e\right]_{f,t_i} \\ &= \frac{1}{S_f}\left[\sum_e \sum_{pc} \frac{1}{2} L_e w_{pc} \mathbf{N}\mathbf{M}_e \cdot \mathbf{U}^{pc}\right]_{f,t_i} \\ &\equiv \frac{1}{S_f}\left[\sum_{e,pc} \frac{1}{2} L_e w_{pc} \mathbf{N}\mathbf{M}_e \cdot \mathbf{U}^{pc}\right]_{f,t_i}\end{aligned} \tag{3.31}$$

where the fact $\mathbf{m}_e \cdot \mathbf{N}_f \equiv \mathbf{0}$ and hence $[\text{tr}\ (\mathbf{U}_t)_e \mathbf{M}_e] \equiv \mathbf{M}_e \cdot \mathbf{U}_e$ have been used.

### Boundary conditions

The BC method does not incorporate the boundary conditions (Box 2.34) directly into the discretized momentum equation for control volumes adjacent to the boundary as it is

done by the SEG method described in Section (3.2.3). Instead, the same discretization procedure is applied to the boundary faces, where boundary-face values are implicitly included as unknowns just as cell-centre values.

In the case of a Dirichlet boundary condition, the prescribed displacement $\overline{\mathbf{U}}$ is used to create an equation for the unknown displacement:

$$\mathbf{I} \cdot \mathbf{U}_f = \overline{\mathbf{U}}_f^{t_i} \tag{3.32}$$

where $\mathbf{U}_f$ and $\overline{\mathbf{U}}_f^{t_i} = \overline{\mathbf{U}}(\mathbf{X}_f, t_i)$ are the unknown and prescribed boundary-face centre displacements respectively. While in the case of Neumann boundary condition, considering that the traction on a boundary face is given by

$$\overline{\mathbf{T}}_f^{t_i} S_f = \int_{\Gamma_f} \left[ \underbrace{(2\mu + \lambda)\nabla \mathbf{U}_n \cdot \mathbf{N} + \lambda(\text{tr } \nabla_t \mathbf{U}_t)\mathbf{N}}_{\mathbf{Q}_n} + \underbrace{\mu \nabla \mathbf{U}_t \cdot \mathbf{N} + \mu \nabla_t U_n}_{\mathbf{Q}_t} \right]_{t_i} dS, \tag{3.33}$$

the discretization is applied in the same manner as for internal cell to

$$\overline{\mathbf{T}}_f^{t_i} = \left[ \underbrace{(2\mu + \lambda)\nabla \mathbf{U}_n \cdot \mathbf{N} + \lambda(\text{tr } \nabla_t \mathbf{U}_t)\mathbf{N}}_{\mathbf{Q}_n} + \underbrace{\mu \nabla \mathbf{U}_t \cdot \mathbf{N} + \mu \nabla_t U_n}_{\mathbf{Q}_t} \right]_{f,t_i}. \tag{3.34}$$

There is also the Symmetry plane boundary condition (a mixed Dirichlet-Neumann condition), which specifies a zero normal displacement and a zero normal gradient of tangential displacement as:

$$\begin{aligned} \mathbf{U}_n &\equiv \mathbf{0} \\ \nabla \mathbf{U}_t \cdot \mathbf{N} &\equiv \mathbf{0} \end{aligned} \tag{3.35}$$

where the condition is enforced at the centre of a boundary face $f$. The equations are discretized as:

$$\begin{aligned} \left[ \mathbf{NN} \cdot \mathbf{U}_f \right]_{f,t_i} &= \mathbf{0} \\ \left[ (\mathbf{I} - \mathbf{NN}) \cdot \left( \frac{\mathbf{U}_f - \mathbf{U}^C}{|\mathbf{d}_{CF}|} \right) \right]_{f,t_i} &= \mathbf{0} \end{aligned}, \tag{3.36}$$

where $\mathbf{U}^C$ is the cell-centre unknown of the face owner cell. These two equations can be summed to produce the final discretized boundary condition because the first only refers to the normal component of the displacement whereas the second only refers to the tangential component of the displacement.

### Linear system

Assembling Equation (3.13) using the full discretization of the surface force term $[\widetilde{\mathbf{P}} \cdot \mathbf{S}]_{f,t_i}$ given as (using Eq. 3.23, 3.24, 3.29 and 3.31)

$$\left[ S_f \Big\{ (2\mu + \lambda)\mathbf{NN} + \mu(\mathbf{I} - \mathbf{NN}) \Big\} \cdot \left( \frac{\mathbf{U}^C - \mathbf{U}^F}{|\mathbf{d}_{CF}|} \right) + \sum_{e,pc} \frac{1}{2} L_e w_{pc} \left( \lambda \mathbf{NM}_e + \mu \mathbf{M}_e \mathbf{N} \right) \cdot \mathbf{U}^{pc} \right]_{f,t_i}, \tag{3.37}$$

produces the following linear algebraic equation

$$\mathbf{A}_C \cdot \mathbf{U}_C^{t_i} + \sum_F \mathbf{A}_F \cdot \mathbf{U}_F^{t_i} = \mathbf{R}_C, \tag{3.38}$$

one for each control volume $C$, where $\mathbf{A}_C$ is the central tensor coefficient, $\mathbf{A}_F$ are tensor coefficients representing interactions with neighbour cell-centred unknowns or boundary-face-centred unknowns and $\mathbf{R}_C$ is the source vector contribution from inertia and body forces terms. It is to be noted that these coefficients are second-order tensors and not scalars as in Equation (3.18).

Let $N_b$ be the number of boundary faces, then an additional $N_b$ linear algebraic equations are originated from boundary discretization processes, one for each boundary-face centre. The linear equations format is analogous to that of above. Both sets of equations form a system of linear equations which can be solved using, e.g. Bi-Conjugate Gradient Stabilised (BiCGStab), Generalised Minimal Residual (GMRes) or even direct methods [36].

# 4 The new Finite Volume Methodology

As highlighted in the introduction of this work, the SEG method is not able to simulate a finite deformation test case of a three-dimensional virtual brick made of cork modeled using the Ogden-Storakers material law (Sec. 2.9.3). This revelation, along with the fact that current BC's formulation is tied to one and only one linear material law (Hooke's law), was the major motivation to create the new FV methodology NLBC.

In fact, the SEG method can be formulated using the Total Lagrangian (TL) or Updated Lagrangian (UL) formulations [35]. The Total and Updated attributes relate to the reference configuration choice. All static and kinematic variables are referred to the initial configuration at the initial time in the TL approach. The UL formulation is based on the same procedures, but in the solution all static and kinematic variables are referred to the last calculated configuration [53]. Furthermore, all kinematic non-linear effects, due to large displacements, large rotations, and large strains, are included in both the TL and UL formulations.

As far as displacement is concerned, each approach above can be subdivided. The choices are: Total (the displacement is referred to the first configuration) or Incremental (the displacement is referred to the last configuration). The combinations give rise to the following frameworks: Total Lagrangian Total Displacement (TLTD); Total Lagrangian Incremental Displacement (TLID); and Updated Lagrangian Incremental Displacement (ULID). Thus, the aforementioned manifestations are SEG-TLTD, SEG-TLID and SEG-ULID [54]. Thus, it becomes evident that BC method uses the TLTD. Further in this chapter, it will be shown that NLBC uses the TLID.

As mentioned in Chapter 1, the two FVM programs used for this work were the nFVM and the Solids4foam [32]. The former implements SEG-TLTD, BC and NLBC[1]. The latter implements all SEG manifestations and BC. The Figure (4.2) compares the TLTD and the TLID framework.

[1] A more complete acronym would be NLBC-TLID.

This chapter presents the NLBC methodology. As the name implies, the new methodology is (mostly) based on the BC approach, but it uses the TLID (due to Newton-Raphson method) instead of TLTD. The NLBC was developed to be as general as the SEG method and as fast-convergent as the BC method, without importing their major weakness (see Fig. 4.1). The first characteristic (generality) is largely retained, because the only imposition is that the elasticity tensor $\mathcal{C}$ of a chosen material law must have the right-minor symmetry property (Eq. 2.51). It was seen in Chapter 2 that every hyperelastic, homogeneous, frame-indifferent and isotropic material model has this property. Therefore, a large class of non-linear materials can be simulated. The numerical tests presented in the next chapter show that the second characteristic (convergence rate) is also retained.

| **Segregated** | **Block-Coupled** | **Non-Linear Block-Coupled** |
|---|---|---|
| Only the laplacian term is linearized in terms of the unknowns | Discretizes all terms in the Hookean's constitutive equation in terms of the unknowns | Apply a second-order Taylor expansion to the constitutive equation and linearize the second term in terms of the unknowns |
| Implicit + explicit discretization | Implicit discretization | Implicit + explicit discretization |
| Each displacement component is solved separately using three linear systems | All displacement components are solved at the same time in a big linear system | All displacement components are solved at the same time in a big linear system |
| Any constitutive equation | Only Hookean's constitutive equation | Any constitutive equation (with right-minor symmetric elasticity tensor) |

Figure 4.1: Differences and similarities between the presented FVM approaches considering only the stress tensor term in the momentum equation.

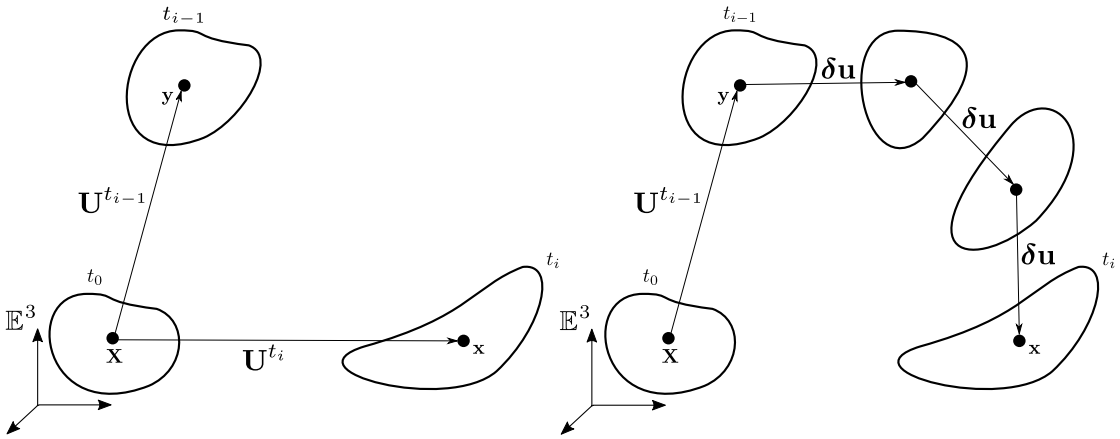



Figure 4.2: Two schematic representations of the same deformation. The image on the left illustrates the TLTD method, where the primary unknown is the displacement $\mathbf{U}$ . The other on the right illustrates the TLID, where the primary unknown is the delta displacement $\boldsymbol{\delta}\mathbf{u}$. The time-dependent boundary conditions are not modified in the intermediate configurations between time $t_{i-1}$ and $t_i$. The black dots represent one and the same material particle.

## 4.1 Mathematical framework for incremental description

To describe the incremental approach, let the following maps be defined:

$$\begin{aligned} &\boldsymbol{\varphi} : \mathbf{X} \in B \to B' \ni \mathbf{x}, \\ &\boldsymbol{\phi} : \mathbf{X} \in B \to B^{\circ} \ni \mathbf{y} \quad \text{and} \\ &\boldsymbol{\chi} : \mathbf{y} \in B^{\circ} \to B' \ni \mathbf{x}, \end{aligned} \tag{4.1}$$

where $B^{\circ}$ can be thought as an intermediate (also labeled as old) body state between the reference body state $B$ and the current body state $B'$ (see Fig. 4.3). Then, by using the composition $\boldsymbol{\chi} \circ \boldsymbol{\phi}$ , it is derived the relation between the deformation gradients associated with the mappings as

$$\mathbf{x} = \boldsymbol{\varphi}(\mathbf{X}) = \boldsymbol{\chi}(\boldsymbol{\phi}(\mathbf{X})) \implies \underbrace{\frac{\partial \boldsymbol{\varphi}}{\partial \mathbf{X}}}_{\mathbf{F}} = \underbrace{\frac{\partial \boldsymbol{\chi}}{\partial \mathbf{y}}}_{\boldsymbol{\delta}\mathbf{F}} \cdot \underbrace{\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}}}_{\mathbf{F}^{\circ}}, \tag{4.2}$$

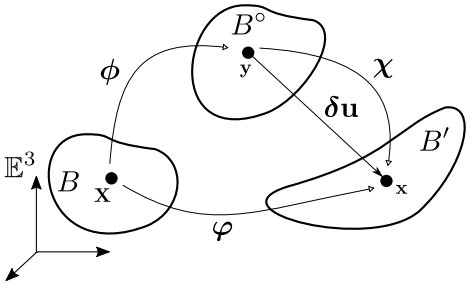



Figure 4.3: A deformation is illustrated by considering the reference configuration $B$, the old configuration $B^\circ$ and the current deformed configuration $B'$. The black dots represent one and the same material particle.

thus obtaining the relation between the deformation gradients as

$$\mathbf{F} = \boldsymbol{\delta}\mathbf{F} \cdot \mathbf{F}^\circ. \tag{4.3}$$

The symbol $\boldsymbol{\delta}\mathbf{F}$ is the well known [41, 55] **incremental** (or **relative**) **deformation gradient**, and its relation with the so-called **incremental** (or **relative**) **displacement gradient** $\nabla^\circ\boldsymbol{\delta}\mathbf{u}$ is found using the **incremental displacement** field $\boldsymbol{\delta}\mathbf{u} : B^\circ \to B'$ (Fig. 4.3) as:

$$\boldsymbol{\delta}\mathbf{u}(\mathbf{y}) = \boldsymbol{\chi}(\mathbf{y}) - \mathbf{y} \implies \underbrace{\frac{\partial \boldsymbol{\delta}\mathbf{u}}{\partial \mathbf{y}}}_{\nabla^\circ\boldsymbol{\delta}\mathbf{u}} = \frac{\partial \boldsymbol{\chi}}{\partial \mathbf{y}} - \frac{\partial \mathbf{y}}{\partial \mathbf{y}} = \boldsymbol{\delta}\mathbf{F} - \mathbf{I}, \tag{4.4}$$

therefore

$$\nabla^\circ\boldsymbol{\delta}\mathbf{u} = \boldsymbol{\delta}\mathbf{F} - \mathbf{I}. \tag{4.5}$$

The intermediate (or old) displacement field $\mathbf{U}^\circ(\mathbf{X}) = \boldsymbol{\phi}(\mathbf{X}) - \mathbf{X}$ gives rise to the intermediate (or old) displacement gradient

$$\nabla\mathbf{U}^\circ = \nabla\boldsymbol{\phi} - \mathbf{I} \implies \nabla\mathbf{U}^\circ = \mathbf{F}^\circ - \mathbf{I}. \tag{4.6}$$

It will be necessary later to linearize the stress tensor $\widetilde{\mathbf{P}}$ at the current body state $B'$. To accomplish that, there must be found a way to compute a gradient increment. This is

done using (4.3), (4.4) and (4.6) as

$$\begin{aligned} \mathbf{F} &= \boldsymbol{\delta}\mathbf{F}\cdot\mathbf{F}^{\circ} \\ \mathbf{I}+\nabla\mathbf{U} &= (\mathbf{I}+\nabla^{\circ}\boldsymbol{\delta}\mathbf{u})\cdot\mathbf{F}^{\circ} \implies \\ \nabla\mathbf{U} &= \mathbf{F}^{\circ}+\nabla^{\circ}\boldsymbol{\delta}\mathbf{u}\cdot\mathbf{F}^{\circ}-\mathbf{I} \implies \\ \nabla\mathbf{U} &= \nabla\mathbf{U}^{\circ}+\nabla^{\circ}\boldsymbol{\delta}\mathbf{u}\cdot\mathbf{F}^{\circ}, \end{aligned} \tag{4.7}$$

where is to be noted that a gradient increment is driven by $\nabla^{\circ}\boldsymbol{\delta}\mathbf{u}\cdot\mathbf{F}^{\circ}$.

## 4.2 Momentum equation linearization

To solve the Equation (3.4), it is first rearranged to

$$\boldsymbol{\mathcal{R}}(\mathbf{U},\nabla\mathbf{U}) = \int_{\Omega_C}\rho_0\ddot{\mathbf{U}}\,dV - \oint_{\partial\Omega_C}\widetilde{\mathbf{P}}(\nabla\mathbf{U})\cdot\mathbf{N}\,dS - \int_{\Omega_C}\rho_0\mathbf{b}_m\,dV = \mathbf{0}, \quad \forall\,\Omega_C\in B_d, \tag{4.8}$$

where $\boldsymbol{\mathcal{R}}:\mathcal{V}\times\mathcal{V}^2\to\mathcal{V}$ can be called **residual function**, since $\boldsymbol{\mathcal{R}}(\mathbf{U},\nabla\mathbf{U})$ is the so-called, in FEM terminology, residual or out-of-balance force [42]. The solution of this equation is sought using a Newton-Raphson iterative process whereby, given a solution estimate $(\mathbf{U}^{n-1},\nabla\mathbf{U}^{n-1})$ at iteration $n-1$, a new value $(\mathbf{U}^{n}=\mathbf{U}^{n-1}+\boldsymbol{\delta}\mathbf{u},\nabla\mathbf{U}^{n}=\nabla\mathbf{U}^{n-1}+\nabla^{\circ}\boldsymbol{\delta}\mathbf{u}\cdot\mathbf{F}^{n-1})$ is obtained by establishing the linear approximation[2]:

$$\begin{aligned} \boldsymbol{\mathcal{R}}(\mathbf{U}^{n},\nabla\mathbf{U}^{n}) \approx& \boldsymbol{\mathcal{R}}(\mathbf{U}^{n-1},\nabla\mathbf{U}^{n-1}) + \frac{\partial\boldsymbol{\mathcal{R}}(\mathbf{U}^{n-1},\nabla\mathbf{U}^{n-1})}{\partial\mathbf{U}}\cdot\boldsymbol{\delta}\mathbf{u} \\ &+ \frac{\partial\boldsymbol{\mathcal{R}}(\mathbf{U}^{n-1},\nabla\mathbf{U}^{n-1})}{\partial\nabla\mathbf{U}} : (\nabla^{\circ}\boldsymbol{\delta}\mathbf{u}\cdot\mathbf{F}^{n-1}) = \mathbf{0}. \end{aligned} \tag{4.9}$$

Or, using a simplified notation, the above relations are written as

$$\boldsymbol{\mathcal{R}}(\mathbf{U}^{\bullet},\nabla\mathbf{U}^{\bullet}) \approx \boldsymbol{\mathcal{R}}(\mathbf{U}^{\circ},\nabla\mathbf{U}^{\circ}) + \frac{\partial\boldsymbol{\mathcal{R}}(\mathbf{U}^{\circ},\nabla\mathbf{U}^{\circ})}{\partial\mathbf{U}}\cdot\boldsymbol{\delta}\mathbf{u} + \frac{\partial\boldsymbol{\mathcal{R}}(\mathbf{U}^{\circ},\nabla\mathbf{U}^{\circ})}{\partial\nabla\mathbf{U}} : (\nabla^{\circ}\boldsymbol{\delta}\mathbf{u}\cdot\mathbf{F}^{\circ}) = \mathbf{0}, \tag{4.10}$$

or even

$$\boldsymbol{\mathcal{R}}^{\bullet} \approx \boldsymbol{\mathcal{R}}^{\circ} + \frac{\partial\boldsymbol{\mathcal{R}}^{\circ}}{\partial\mathbf{U}}\cdot\boldsymbol{\delta}\mathbf{u} + \frac{\partial\boldsymbol{\mathcal{R}}^{\circ}}{\partial\nabla\mathbf{U}} : (\nabla^{\circ}\boldsymbol{\delta}\mathbf{u}\cdot\mathbf{F}^{\circ}) = \mathbf{0}, \tag{4.11}$$

[2]The approximation is just a truncated Taylor series.

where the first term is simply

$$\boldsymbol{\mathcal{R}}^{\circ} = \int_{\Omega_C} \rho_0 \ddot{\mathbf{U}}^{\circ}\, dV - \oint_{\partial\Omega_C} \widetilde{\mathbf{P}}^{\circ} \cdot \mathbf{N}\, dS - \int_{\Omega_C} \rho_0 \mathbf{b}_m\, dV. \tag{4.12}$$

The second and the third terms are calculated as

$$\begin{aligned} \frac{\partial \boldsymbol{\mathcal{R}}^{\circ}}{\partial \mathbf{U}} \cdot \boldsymbol{\delta}\mathbf{u} &= \frac{\partial}{\partial \mathbf{U}} \left[ \int_{\Omega_C} \rho_0 \ddot{\mathbf{U}}\, dV \right]^{\circ} \cdot \boldsymbol{\delta}\mathbf{u} = \int_{\Omega_C} \rho_0 \left. \frac{\partial \ddot{\mathbf{U}}}{\partial \mathbf{U}} \right|^{\circ} \cdot \boldsymbol{\delta}\mathbf{u}\, dV \\ &= \int_{\Omega_C} \rho_0 \frac{\partial^2}{\partial t^2} \left[ \frac{\partial \mathbf{U}}{\partial \mathbf{U}} \cdot \boldsymbol{\delta}\mathbf{u} \right]^{\circ} dV = \int_{\Omega_C} \rho_0 \frac{\partial^2 \boldsymbol{\delta}\mathbf{u}}{\partial t^2}\, dV = \int_{\Omega_C} \rho_0 \ddot{\boldsymbol{\delta}\mathbf{u}}\, dV, \end{aligned} \tag{4.13}$$

and as

$$\begin{aligned} \frac{\partial \boldsymbol{\mathcal{R}}^{\circ}}{\partial \nabla \mathbf{U}} : (\nabla^{\circ} \boldsymbol{\delta}\mathbf{u} \cdot \mathbf{F}^{\circ}) &= -\frac{\partial}{\partial \nabla \mathbf{U}} \left[ \oint_{\partial\Omega_C} \widetilde{\mathbf{P}}(\nabla \mathbf{U}) \cdot \mathbf{N}\, dS \right]^{\circ} : (\nabla^{\circ} \boldsymbol{\delta}\mathbf{u} \cdot \mathbf{F}^{\circ}) \\ &= -\oint_{\partial\Omega_C} \left[ \frac{\partial \widetilde{\mathbf{P}}(\nabla \mathbf{U})}{\partial \nabla \mathbf{U}} : (\nabla^{\circ} \boldsymbol{\delta}\mathbf{u} \cdot \mathbf{F}^{\circ}) \right]^{\circ} \cdot \mathbf{N}\, dS \\ &= -\oint_{\partial\Omega_C} \left[ \frac{\partial \widetilde{\mathbf{P}}^{\circ}}{\partial \nabla \mathbf{U}} : (\nabla^{\circ} \boldsymbol{\delta}\mathbf{u} \cdot \mathbf{F}^{\circ}) \right] \cdot \mathbf{N}\, dS, \end{aligned} \tag{4.14}$$

respectively. Assuming the above linearization and substituting the expressions (4.12)-(4.14) into Equation (4.11) give

$$\underbrace{\int_{\Omega_C} \rho_0 \left( \ddot{\mathbf{U}}^{\circ} + \ddot{\boldsymbol{\delta}\mathbf{u}} \right) dV}_{\text{inertial force}} \underbrace{- \oint_{\partial\Omega_C} \left[ \frac{\partial \widetilde{\mathbf{P}}^{\circ}}{\partial \nabla \mathbf{U}} : (\nabla^{\circ} \boldsymbol{\delta}\mathbf{u} \cdot \mathbf{F}^{\circ}) \right] \cdot \mathbf{N}\, dS}_{\text{surface force increment}} \underbrace{- \oint_{\partial\Omega_C} \widetilde{\mathbf{P}}^{\circ} \cdot \mathbf{N}\, dS}_{\text{old surface force}} \underbrace{- \int_{\Omega_C} \rho_0 \mathbf{b}_m\, dV}_{\text{body force}} = \mathbf{0}. \tag{4.15}$$

Note that “old” in this context can be read as “old iteration of the Newton-Raphson process”, but during the first iteration it coincides to the old time point (see Fig. 4.4 and 4.5). It is revealed in the next section how these terms are discretized. The body force term is the exception, its discretization is given in Section (3.2.2).

## 4.3 Discretization of the force terms

Each term of the Equation (4.15) needs a different discretization which are shown next.

### 4.3.1 Inertial force

Let the displacements $\mathbf{U}^{\circ}$, $\mathbf{U}^{t_{i-1}}$ and $\mathbf{U}^{t_{i-2}}$ refer to values obtained at a previous iteration, and at a previous and at two time-steps earlier, respectively. The discretization of the inertial force term, from Equation (4.15), follows Equation (3.10), i.e.

$$\int_{\Omega_C} \rho_0 \left( \ddot{\mathbf{U}}^{\circ} + \ddot{\boldsymbol{\delta}}\mathbf{u} \right) dV \approx \left[ \frac{\rho_0 V}{\Delta t^2} \left( (\mathbf{U}^{\circ} + \boldsymbol{\delta}\mathbf{u}) - 2\mathbf{U}^{t_{i-1}} + \mathbf{U}^{t_{i-2}} \right) \right]_C , \tag{4.16}$$

where $\mathbf{U}^{\circ} + \boldsymbol{\delta}\mathbf{u}$ is an approximation for the current unknown displacement $\mathbf{U}^{t_i}$ (see Fig. 4.2).

### 4.3.2 Surface force increment

Before proceeding to the discretization of the surface force increment, the Equation (2.33) needs to be extended by taking the $\boldsymbol{\mathcal{C}}$'s right-minor symmetry into consideration (see 2.51) as

$$\begin{aligned}
\frac{\partial \widetilde{\mathbf{P}}}{\partial \nabla \mathbf{U}} : \mathbf{A} &= \mathbf{A} \cdot \widetilde{\boldsymbol{\Sigma}} + \mathbf{F} \cdot \left[ \boldsymbol{\mathcal{C}} : \left( \text{sym}(\mathbf{F}^T \cdot \mathbf{A}) \right) \right] \\
&= \mathbf{A} \cdot \widetilde{\boldsymbol{\Sigma}} + \mathbf{F} \cdot \left[ \boldsymbol{\mathcal{C}} : \left( \mathbf{F}^T \cdot \mathbf{A} \right) \right] \qquad \text{(using } \boldsymbol{\mathcal{C}}\text{'s symmetric property)} \\
&= \mathbf{A} \cdot \widetilde{\boldsymbol{\Sigma}} + \mathbf{F} \cdot \left[ \mathcal{C}_{\alpha\beta\gamma\delta} F_{a\gamma} A_{a\delta} \mathbf{e}_\alpha \otimes \mathbf{e}_\beta \right] \\
&= \mathbf{A} \cdot \widetilde{\boldsymbol{\Sigma}} + \left( \mathbf{F} \cdot \underset{(3)}{\boldsymbol{\mathcal{C}}} \cdot \mathbf{F}^T \right) : \mathbf{A} \\
&= \mathbf{A} \cdot \widetilde{\boldsymbol{\Sigma}} + \boldsymbol{\mathcal{M}} : \mathbf{A}, \qquad \forall \mathbf{A} \in \mathcal{V}^2,
\end{aligned} \tag{4.17}$$

where $\boldsymbol{\mathcal{M}}$ is the transformed elasticity tensor defined in the paragraph preceding Equation (2.53). Note that the right-minor symmetry restriction, which could not be overcome, creates a class of supported materials. This is discussed in more details in Chapter 7.

Now, the integral of the surface force increment term is approximated as:

$$\begin{aligned}
&\oint_{\partial\Omega_C} \left[\frac{\partial\widetilde{\mathbf{P}}^\circ}{\partial\nabla\mathbf{U}} : (\nabla^\circ\boldsymbol{\delta}\mathbf{u}\cdot\mathbf{F}^\circ)\right]\cdot\mathbf{N}\,dS \\
&\qquad = \sum_{\Gamma_f\in\partial\Omega_C}\int_{\Gamma_f}\left[\frac{\partial\widetilde{\mathbf{P}}^\circ}{\partial\nabla\mathbf{U}} : (\nabla^\circ\boldsymbol{\delta}\mathbf{u}\cdot\mathbf{F}^\circ)\right]\cdot\mathbf{N}\,dS \qquad (\partial\Omega_C \text{ is a polyhedral}) \\
&\qquad \approx \sum_f S_f\left[\left\{\frac{\partial\widetilde{\mathbf{P}}^\circ}{\partial\nabla\mathbf{U}} : (\nabla^\circ\boldsymbol{\delta}\mathbf{u}\cdot\mathbf{F}^\circ)\right\}\cdot\mathbf{N}\right]_f \qquad \text{(mid-point rule integration)}.
\end{aligned} \tag{4.18}$$

Substituting for $\mathbf{A} = \nabla^\circ\boldsymbol{\delta}\mathbf{u}\cdot\mathbf{F}^\circ$ into Equation (4.17) yields

$$\begin{aligned}
\left[\left\{\frac{\partial\widetilde{\mathbf{P}}^\circ}{\partial\nabla\mathbf{U}} : (\nabla^\circ\boldsymbol{\delta}\mathbf{u}\cdot\mathbf{F}^\circ)\right\}\cdot\mathbf{N}\right]_f &= \left[\left(\nabla^\circ\boldsymbol{\delta}\mathbf{u}\cdot\mathbf{F}^\circ\cdot\widetilde{\boldsymbol{\Sigma}}^\circ\right)\cdot\mathbf{N}\right]_f \\
&\quad + \left[\left\{\boldsymbol{\mathcal{M}}^\circ : \left(\nabla^\circ\boldsymbol{\delta}\mathbf{u}\cdot\mathbf{F}^\circ\right)\right\}\cdot\mathbf{N}\right]_f,
\end{aligned} \tag{4.19}$$

where $\widetilde{\boldsymbol{\Sigma}}^\circ = \widetilde{\boldsymbol{\Sigma}}(\nabla\mathbf{U}^\circ)$ and $\boldsymbol{\mathcal{M}}^\circ = \boldsymbol{\mathcal{M}}(\nabla\mathbf{U}^\circ)$. Letting $\mathbf{NN} = \mathbf{N}\otimes\mathbf{N}$, the first term on the right-hand side of Equation (4.19) is

$$\begin{aligned}
\left[\nabla^\circ\boldsymbol{\delta}\mathbf{u}\cdot\mathbf{F}^\circ\cdot\widetilde{\boldsymbol{\Sigma}}^\circ\cdot\mathbf{N}\right]_f &= \left[(\nabla^\circ\delta u)_{ab}F^\circ_{bc}\Sigma^\circ_{cd}N_d\mathbf{e}_a\right]_f \\
&= \left[(\nabla^\circ\delta u)_{ab}v^\circ_b\mathbf{e}_a\right]_f \qquad (\mathbf{v}^\circ = v^\circ_b\mathbf{e}_b = F^\circ_{bc}\Sigma^\circ_{cd}N_d\mathbf{e}_b) \\
&= \left[\nabla^\circ\boldsymbol{\delta}\mathbf{u}\cdot\mathbf{v}^\circ\right]_f \\
&= \left[\nabla^\circ\boldsymbol{\delta}\mathbf{u}\cdot\mathbf{v}^\circ_n + \nabla^\circ\boldsymbol{\delta}\mathbf{u}\cdot\mathbf{v}^\circ_t\right]_f \quad (\mathbf{v}^\circ_n = (\mathbf{v}^\circ\cdot\mathbf{N})\mathbf{N},\ \mathbf{v}^\circ_t = (\mathbf{I}-\mathbf{NN})\cdot\mathbf{v}^\circ) \\
&= \left[(\mathbf{v}^\circ\cdot\mathbf{N})\nabla^\circ\boldsymbol{\delta}\mathbf{u}\cdot\mathbf{N} + \nabla^\circ\boldsymbol{\delta}\mathbf{u}\cdot\mathbf{v}^\circ_t\right]_f.
\end{aligned} \tag{4.20}$$

Note the projection of $\mathbf{v}^\circ$ onto the face normal direction and onto the face plane. This step creates the opportunity to apply the same discretization procedures, employed by

the BC method, to calculate the normal and tangential derivative terms (i.e. $\nabla^\circ \boldsymbol{\delta u} \cdot \mathbf{N}$ and $\nabla^\circ \boldsymbol{\delta u} \cdot \mathbf{v}_t^\circ$, see Section 3.2.4). The second term on the right-hand side of Equation (4.19) is computed as:

$$
\begin{aligned}
&\left[\left\{\boldsymbol{\mathcal{M}}^\circ : \left(\nabla^\circ \boldsymbol{\delta u} \cdot \mathbf{F}^\circ\right)\right\} \cdot \mathbf{N}\right]_f \\
&\quad = \left[\mathcal{M}^\circ_{abcd} (\nabla^\circ \delta u)_{ce} F^\circ_{ed} N_b \mathbf{e}_a\right]_f \\
&\quad = \left[\overline{\mathcal{M}}^\circ_{acd} (\nabla^\circ \delta u)_{ce} F^\circ_{ed} \mathbf{e}_a\right]_f \qquad (\overline{\mathcal{M}}^\circ_{acd} = \mathcal{M}^\circ_{abcd} N_b) \\
&\quad = \left[\sum_d \overline{\mathcal{M}}^\circ_{acd} (\nabla^\circ \delta u)_{ce} g^\circ_{ed} \mathbf{e}_a\right]_f \qquad (\mathbf{g}^\circ_d = g^\circ_{ed} \mathbf{e}_e = F^\circ_{ed} \mathbf{e}_e) \\
&\quad = \left[\sum_d \mathbf{T}^\circ_d \cdot \nabla^\circ \boldsymbol{\delta u} \cdot \mathbf{g}^\circ_d\right]_f \qquad (\mathbf{T}^\circ_d = \overline{\mathcal{M}}^\circ_{acd} \mathbf{e}_a \otimes \mathbf{e}_c) \\
&\quad = \left[\sum_d \mathbf{T}^\circ_d \cdot \nabla^\circ \boldsymbol{\delta u} \cdot ((\mathbf{g}^\circ_d \cdot \mathbf{N})\mathbf{N} + (\mathbf{I} - \mathbf{N}\mathbf{N}) \cdot \mathbf{g}^\circ_d))\right]_f \qquad (\text{project } \mathbf{g}^\circ_d) \\
&\quad = \left[\sum_d (\mathbf{g}^\circ_d \cdot \mathbf{N}) \mathbf{T}^\circ_d \cdot (\nabla^\circ \boldsymbol{\delta u} \cdot \mathbf{N})\right]_f + \left[\sum_d \mathbf{T}^\circ_d \cdot (\nabla^\circ \boldsymbol{\delta u} \cdot \mathbf{h}^\circ_d)\right]_f \qquad (\mathbf{h}^\circ_d = (\mathbf{I} - \mathbf{N}\mathbf{N}) \cdot \mathbf{g}^\circ_d).
\end{aligned}
\tag{4.21}
$$

Using (4.20) and (4.21), the term (4.19) is given as:

$$
\begin{aligned}
\left[\left\{\frac{\partial \widetilde{\mathbf{P}}^\circ}{\partial \nabla \mathbf{U}} : (\nabla^\circ \boldsymbol{\delta u} \cdot \mathbf{F}^\circ)\right\} \cdot \mathbf{N}\right]_f &= \left[(\mathbf{v}^\circ \cdot \mathbf{N}) \nabla^\circ \boldsymbol{\delta u} \cdot \mathbf{N} + \nabla^\circ \boldsymbol{\delta u} \cdot \mathbf{v}_t^\circ\right]_f \\
&\quad + \left[\sum_d (\mathbf{g}^\circ_d \cdot \mathbf{N}) \mathbf{T}^\circ_d \cdot (\nabla^\circ \boldsymbol{\delta u} \cdot \mathbf{N})\right]_f \\
&\quad + \left[\sum_d \mathbf{T}^\circ_d \cdot (\nabla^\circ \boldsymbol{\delta u} \cdot \mathbf{h}^\circ_d)\right]_f \\
&\quad = \underbrace{\left[\left\{(\mathbf{v}^\circ \cdot \mathbf{N})\mathbf{I} + \sum_d (\mathbf{g}^\circ_d \cdot \mathbf{N}) \mathbf{T}^\circ_d\right\} \cdot (\nabla^\circ \boldsymbol{\delta u} \cdot \mathbf{N})\right]_f}_{(1)} \\
&\quad + \underbrace{\left[\nabla^\circ \boldsymbol{\delta u} \cdot \mathbf{v}_t^\circ + \sum_d \mathbf{T}^\circ_d \cdot (\nabla^\circ \boldsymbol{\delta u} \cdot \mathbf{h}^\circ_d)\right]_f}_{(2)}.
\end{aligned}
\tag{4.22}
$$

The underlined terms (1) and (2) from the equation before are approximated using the same approach taken by the BC method (see Section 3.2.4), i.e. the normal derivative term (1) is discretized using the central differencing method as

$$\Bigg[\underbrace{\left\{(\mathbf{v}^\circ\cdot\mathbf{N})\mathbf{I}+\sum_d(\mathbf{g}_d^\circ\cdot\mathbf{N})\mathbf{T}_d^\circ\right\}}_{\mathbf{H}_n^\circ}\cdot\left(\frac{\boldsymbol{\delta}\mathbf{u}^C-\boldsymbol{\delta}\mathbf{u}^F}{|\mathbf{d}_{CF}|}\right)\Bigg]_f, \tag{4.23}$$

where the vector connecting the centroids of the cells sharing the common face $\mathbf{d}_{CF}=\mathbf{X}_F-\mathbf{X}_C$.

The tangential face derivative term (2) above is discretized using the face-Gauss/Finite Area method (Session 3.2.4) as

$$\begin{aligned}\left[\frac{1}{S}\sum_e L_e(\mathbf{M}_e\cdot\mathbf{v}_t^\circ)\boldsymbol{\delta}\mathbf{u}_e+\sum_d\mathbf{T}_d^\circ\cdot\left\{\frac{1}{|\mathbf{S}|}\sum_e L_e(\mathbf{M}_e\cdot\mathbf{h}_d^\circ)\boldsymbol{\delta}\mathbf{u}_e)\right\}\right]_f\\ =\left[\frac{1}{S}\sum_e\underbrace{L_e\left[(\mathbf{M}_e\cdot\mathbf{v}_t^\circ)\mathbf{I}+\sum_d(\mathbf{M}_e\cdot\mathbf{h}_d^\circ)\mathbf{T}_d^\circ\right]}_{\mathbf{H}_t^\circ}\cdot\boldsymbol{\delta}\mathbf{u}_e\right]_f.\end{aligned} \tag{4.24}$$

### 4.3.3 Old surface force

The discretization of this term uses the mid-point integration approximation as

$$\begin{aligned}\oint_{\partial\Omega_C}\widetilde{\mathbf{P}}^\circ\cdot\mathbf{N}\,dS&=\sum_{\Gamma_f\in\partial\Omega_C}\int_{\Gamma_f}\widetilde{\mathbf{P}}^\circ\cdot\mathbf{N}\,dS\qquad(\partial\Omega_C\text{ is a polyhedral})\\ &\approx\sum_f\left[\widetilde{\mathbf{P}}^\circ\cdot\mathbf{S}\right]_f\qquad(\text{mid-point rule integration}),\end{aligned} \tag{4.25}$$

where $\widetilde{\mathbf{P}}^\circ$ is the last known value of the first Piola-Kirchhoff stress tensor.

## 4.4 Boundary conditions

The boundary conditions (Box 2.34) are handled in the same way as in the BC method (Session 3.2.4), except by the fact that, instead of $\mathbf{U}_f$, $\mathbf{U}_f^\circ+\boldsymbol{\delta}\mathbf{u}_f$ is used. Thus, Equation

(3.32) becomes

$$\mathbf{I} \cdot \left(\mathbf{U}_f^{\circ} + \boldsymbol{\delta}\mathbf{u}_f\right) = \overline{\mathbf{U}}_f^{t_i} \implies \mathbf{I} \cdot \boldsymbol{\delta}\mathbf{u}_f = \overline{\mathbf{U}}_f^{t_i} - \mathbf{U}_f^{\circ}, \tag{4.26}$$

and Equation (3.34) becomes

$$\overline{\mathbf{T}}_f^{t_i} = \left[\widetilde{\mathbf{P}}(\nabla\mathbf{U}^{\circ}) \cdot \mathbf{N} + \left\{\frac{\partial\widetilde{\mathbf{P}}(\nabla\mathbf{U}^{\circ})}{\partial\nabla\mathbf{U}} : \left(\nabla^{\circ}\boldsymbol{\delta}\mathbf{u} \cdot \mathbf{F}^{\circ}\right)\right\} \cdot \mathbf{N}\right]_f \tag{4.27}$$

to be discretized using the same processes applied to the surface force increment and to the old surface force terms described before. The symmetry plane boundary condition is discretized analogously to BC's approach (Eq. 3.36).

## 4.5 Linear system

Assembling Equation (4.15) using (4.16), (4.23), (4.24), (4.25) and (3.2.2) along with a material law produces a linear algebraic equation with the same structure seen in the Block-Coupled method (Eq. 3.38). However, instead of solving for $\mathbf{U}^{t_i}$, it is solved for $\boldsymbol{\delta}\mathbf{u}$, i.e. the final equation becomes

$$\mathbf{A}_C \cdot \boldsymbol{\delta}\mathbf{u}_C + \sum_F \mathbf{A}_F \cdot \boldsymbol{\delta}\mathbf{u}_F = \mathbf{R}_C. \tag{4.28}$$

The contribution from the boundary discretization is also analogous to that of the BC method, and also contributes to the final system of linear equations, say $[\mathbf{A}][\boldsymbol{\delta}\mathbf{u}] = [\mathbf{R}]$, which can be solved using the same linear solvers already cited.

## 4.6 Newton-Raphson algorithm

Before the algorithm is presented, a residual formula should be given: let $a = \max_C |\mathbf{U}_C^n - \mathbf{U}_C^{t_{i-1}}|$, $b = \max\left(\max_C |\mathbf{U}_C^n|, \epsilon\right)$ and $d = a$ if $a > \epsilon$ or $d = b$ otherwise, then a residual can be defined as $\max_C \frac{|\mathbf{U}_C^n - \mathbf{U}_C^{n-1}|}{d}$, where the scope of max function is all the computational points and $\epsilon$ is a small value, e.g. $10^{-7}$ (the same value used in both nFVM and S4F).

In practice, the resulting Newton-Raphson algorithm is given in Figure (4.4), where each assignment happens to all computational nodes at the same time. The inner loop finishes when the residual is below a certain threshold or the number of iterations is equal to the maximum number of corrections $n_{\text{corr}}$ . A graphical illustration for one time increment is given by Figure (4.5).

**Newton-Raphson Algorithm**

- INPUT geometry, material law and solution parameters
- INITIALIZE $\mathbf{x} = \mathbf{X}$ // initial geometry
- INITIALIZE $\mathbf{U}^{t_i} = \mathbf{0}$, $\mathbf{U}^{\bullet} = \mathbf{0}$
- **LOOP** over time increments
  - SET $\mathbf{U}^{\circ} = \mathbf{U}^{t_i}$
  - UPDATE time-dependent boundary conditions
  - SET $n = 0$
  - **WHILE** $n < n_{\text{corr}}$ // Loop over $\boldsymbol{\delta}\mathbf{u}$'s
    - FIND $\nabla\mathbf{U}^{\circ}$ // Using Green-Gauss or least square approach [51]
    - SET $\mathbf{F}^{\circ} = \mathbf{I} + \nabla\mathbf{U}^{\circ}$
    - SET $[\mathbf{A}]$ and $[\mathbf{R}]$
    - SOLVE the linear system $[\mathbf{A}][\boldsymbol{\delta}\mathbf{u}] = [\mathbf{R}]$
    - UPDATE $\mathbf{x} = \mathbf{x} + \boldsymbol{\delta}\mathbf{u}$
    - SET $\mathbf{U}^{\circ} = \mathbf{U}^{\bullet}$
    - SET $\mathbf{U}^{\bullet} = \mathbf{x} - \mathbf{X}$ // current approximation of $\mathbf{U}^{t_i}$
    - COMPUTE residual
    - **IF** residual $<$ tolerance
      - BREAK // $\mathbf{U}^{\bullet}$ is considered close enough to $\mathbf{U}^{t_i}$
    - **ENDIF**
    - UPDATE $n = n + 1$
  - **ENDLOOP**
  - UPDATE $\mathbf{U}^{t_{i-1}} = \mathbf{U}^{t_i}$
  - UPDATE $\mathbf{U}^{t_i} = \mathbf{U}^{\bullet}$
- **ENDLOOP**

Figure 4.4: Note that each assignment happens to all computational nodes at the same time.

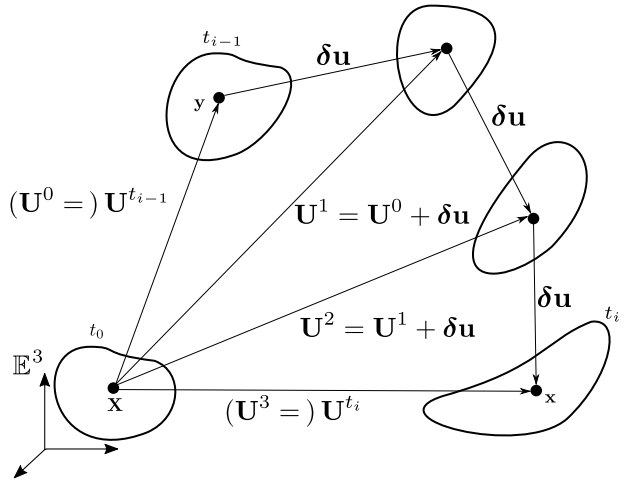



Figure 4.5: Schematic representation of an incremental deformation. The old reference configuration at the time $t_{i-1}$ is deformed onto the new configuration at the time $t_i$. To compute this deformation, it is supposed here that three iterations of the inner loop (**while** scope) in the Newton-Raphson algorithm (Fig. 4.4) suffice.

## 4.7 Linearized elasticity

As mentioned in the introduction of this chapter, the BC method is restricted to the linearized elasticity framework. The attribute "linearized", in this case, means that infinitesimally small displacements are assumed and linear elastic material laws, e.g. Hooke's law, are considered relatively good approximations. The displacement assumption implies that strains are small. Consequently, the difference between the initial and current dimensions becomes irrelevant and only one configuration is used [50]. Furthermore, the two Piola-Kirchhoff and Cauchy stress tensors become indistinguishable, i.e. $\mathbf{P} = \mathbf{\Sigma} = \boldsymbol{\sigma}$.

Precisely, a linear elastic material has the form

$$\mathbf{P} = \boldsymbol{\mathcal{C}} : \mathbf{E}, \tag{4.29}$$

which means that $\mathbf{P}$ is linear in $\mathbf{E}$. As the Hooke's elasticity tensor is given by

$$\boldsymbol{\mathcal{C}} = \lambda \mathbb{J} + 2\mu \mathbb{I} \iff \mathcal{C}_{abcd} = \lambda \delta_{ab}\delta_{cd} + \mu(\delta_{ac}\delta_{bd} + \delta_{ad}\delta_{bc}), \tag{4.30}$$

it is straightforward to see that by substituting this equation and the expression for $\mathbf{E}$ (Eq. 2.6) into Equation (4.29) gives (3.15). Besides, the Hooke's $\mathbf{T}^{\circ}_{d}$ is given by

$$(T^{d}_{ab})^{\circ} = \lambda N_a \delta_{bd} + \mu(N_d \delta_{ab} + \delta_{ad} N_b). \tag{4.31}$$

As an important fact, when NLBC is also restricted to linearized elasticity, it reduces to BC formulation. Thus, NLBC can be seen as a generalization of BC. The demonstration is as follows. Assume the reference configuration is undeformed ($\mathbf{F}^{\circ} = \mathbf{I} \implies \widetilde{\mathbf{\Sigma}}^{\circ} = \widetilde{\mathbf{P}}^{\circ} = \mathbf{0}$), the Hooke's law is employed ($\boldsymbol{\mathcal{C}}$ is given by Eq. 4.30) and neglect inertial and body forces for clarity. For that reason, it suffices to show that the surface force increment (left-hand side of Eq. 4.22 multiplied by $S$) is equal to Equation (3.37), i.e.

$$\begin{aligned} \underbrace{\left[\left\{\frac{\partial \widetilde{\mathbf{P}}^{\circ}}{\partial \nabla \mathbf{U}} : (\nabla^{\circ}\boldsymbol{\delta}\mathbf{u} \cdot \mathbf{F}^{\circ})\right\} \cdot \mathbf{S}\right]_{f,t_i}}_{\text{NLBC's surface force increment}} &= \underbrace{\left[\cancelto{\mathbf{0}}{\widetilde{\mathbf{P}}^{\circ}} \cdot \mathbf{S} + \left\{\frac{\partial \widetilde{\mathbf{P}}^{\circ}}{\partial \nabla \mathbf{U}} : (\nabla^{\circ}\boldsymbol{\delta}\mathbf{u} \cdot \mathbf{F}^{\circ})\right\} \cdot \mathbf{S}\right]_{f,t_i}}_{\text{NLBC's surface force (see Eq. 4.15)}} \\ &= \underbrace{\left[\widetilde{\mathbf{P}} \cdot \mathbf{S}\right]_{f,t_i}}_{\text{BC's surface force (Eq. 3.37)}}. \end{aligned} \tag{4.32}$$

And this is straightforward because the sum of the terms composing the increment (Eq. 4.23 and 4.24 ), i.e.

$$
\begin{aligned}
&\left[S_f \mathbf{H}_n^\circ \cdot \left(\frac{\boldsymbol{\delta}\mathbf{u}^C - \boldsymbol{\delta}\mathbf{u}^F}{|\mathbf{d}_{CF}|}\right)\right]_f \\
&\qquad = \left[S_f\Big\{(\mathbf{v}^\circ \cdot \mathbf{N})\mathbf{I} + \sum_d (\mathbf{g}_d^\circ \cdot \mathbf{N})\mathbf{T}_d^\circ\Big\} \cdot \left(\frac{\boldsymbol{\delta}\mathbf{u}^C - \boldsymbol{\delta}\mathbf{u}^F}{|\mathbf{d}_{CF}|}\right)\right]_f \\
&\qquad = \left[S_f\Big\{\sum_d (\mathbf{g}_d^\circ \cdot \mathbf{N})\mathbf{T}_d^\circ\Big\} \cdot \left(\frac{\boldsymbol{\delta}\mathbf{u}^C - \boldsymbol{\delta}\mathbf{u}^F}{|\mathbf{d}_{CF}|}\right)\right]_f \qquad (\boldsymbol{\Sigma}^\circ = \mathbf{0} \implies \mathbf{v}^\circ = \mathbf{0}) \\
&\qquad = \left[S_f\Big\{\sum_d N_d \mathbf{T}_d^\circ\Big\} \cdot \left(\frac{\boldsymbol{\delta}\mathbf{u}^C - \boldsymbol{\delta}\mathbf{u}^F}{|\mathbf{d}_{CF}|}\right)\right]_f \qquad (\mathbf{F}^\circ = \mathbf{I} \implies g_{ad}^\circ = \delta_{ad}) \\
&\qquad = \left[S_f\Big\{(2\mu + \lambda)\mathbf{NN} + \mu(\mathbf{I} - \mathbf{NN})\Big\} \cdot \left(\frac{\boldsymbol{\delta}\mathbf{u}^C - \boldsymbol{\delta}\mathbf{u}^F}{|\mathbf{d}_{CF}|}\right)\right]_f \qquad (\text{Equation 4.31})
\end{aligned}
\tag{4.33}
$$

and

$$
\begin{aligned}
&\left[\sum_e \mathbf{H}_t^\circ \cdot \boldsymbol{\delta}\mathbf{u}_e\right]_f \\
&= \left[\sum_e L_e \left[(\mathbf{M}_e \cdot \mathbf{v}_t^\circ)\mathbf{I} + \sum_d (\mathbf{M}_e \cdot \mathbf{h}_d^\circ)\mathbf{T}_d^\circ\right] \cdot \boldsymbol{\delta}\mathbf{u}_e\right]_f \\
&= \left[\sum_e L_e \left[\sum_d (\mathbf{M}_e \cdot \mathbf{h}_d^\circ)\mathbf{T}_d^\circ\right] \cdot \boldsymbol{\delta}\mathbf{u}_e\right]_f \qquad (\boldsymbol{\Sigma}^\circ = \mathbf{0} \implies \mathbf{v}^\circ = \mathbf{0}) \\
&= \left[\sum_e L_e \left[\sum_d (\mathbf{M}_e \cdot (\mathbf{I} - \mathbf{N}\mathbf{N}) \cdot \boldsymbol{\delta}^d)\mathbf{T}_d^\circ\right] \cdot \boldsymbol{\delta}\mathbf{u}_e\right]_f \qquad (\boldsymbol{\delta}^d = \delta_{ad}\mathbf{e}_a) \\
&= \left[\sum_e L_e \left[\sum_d (\mathbf{M}_e \cdot \boldsymbol{\delta}^d)\mathbf{T}_d^\circ\right] \cdot \boldsymbol{\delta}\mathbf{u}_e\right]_f \qquad (\mathbf{M}_e \cdot \mathbf{N} = \mathbf{0}) \\
&= \left[\sum_e L_e \left[\sum_d (M_e)_d \mathbf{T}_d^\circ\right] \cdot \boldsymbol{\delta}\mathbf{u}_e\right]_f \\
&= \left[\sum_e L_e \left(\lambda \mathbf{N}\mathbf{M}_e + \mu \mathbf{M}_e\mathbf{N}\right) \cdot \boldsymbol{\delta}\mathbf{u}_e\right]_f \qquad \text{(using 4.31)} \\
&= \left[\sum_{e,pc} \frac{1}{2} L_e w_{pc} \left(\lambda \mathbf{N}\mathbf{M}_e + \mu \mathbf{M}_e\mathbf{N}\right) \cdot \boldsymbol{\delta}\mathbf{u}^{pc}\right]_f \qquad \text{(using 3.27 and 3.28)}
\end{aligned}
\tag{4.34}
$$

is precisely the surface force given in Equation (2.35), and the proof is complete. Note that it also shows analytically that NLBC works in the linearized elastic framework.

# 5 Method verification

This chapter presents the application of the NLBC to several representative benchmark test cases. The goal is to use a sequence of benchmarks, with increasing complexity, to analyze the accuracy and robustness of NLBC. As reported in the introduction of this work, the methods SEG, BC and NLBC were implemented in the new Matlab toolbox nFVM (see Chapter 6). The S4F's SEG[1] implementation was also used for one specific test case in order to confirm the result given by nFVM's SEG implementation. Note that, for all test cases examined in this chapter, a solution is considered converged when the residual falls below $10^{-7}$.

The NLBC methodology is examined along three sections:

- Linearized elasticity – A representative benchmark, used to verify the NLBC method for this framework, is examined in this section. It also serves to show that SEG can experience slow convergence rates whenever there is strong coupling between displacement directions.

- Non-linear elasticity – Using this framework, which allows simulation of accurate "large displacement-large strain" models, it is presented in this section the comparison of NLBC with the SEG solution procedure.

- Conclusion – The main relevant aspects collected from the numerical results are summarized.

It is shown next only 2-D test cases, because the analysis of 3-D cases did not contribute to any extra significant insights.

[1] The SEG-TLTD manifestation was used (see introduction of Chapter 4).

## 5.1 Linearized elasticity

As shown in Section (4.7), when NLBC is restricted to the linearized elasticity framework, it reduces to BC formulation. Thus, the latter can be seen as a special case of the former. The results from the test case presented in this section, that of a slender 2-D cantilever undergoing bending, show this fact by means of numerical simulation. This case was used by Cardiff *et al.* [36] in their seminal work on the BC method.

The geometry of the test case, shown in Figure (5.1), consists of a rectangle beam 2 x 0.1 m with a Young's modulus $E$ of 200 GPa and a Poisson's ratio $\nu$ of 0.3. Three uniform quadrilateral meshes were considered: 60x3, 100x5 and 300x15 cells. The mesh with 100x5 cells is shown in Figure (5.2). The beam is fixed at the left end, by imposing the boundary displacement condition $\overline{\mathbf{U}} = [0\ 0]^T$m, and is subjected to a uniform distributed traction at the other end, by imposing the boundary traction condition $\overline{\mathbf{T}} = [0\ 1]^T$ MPa. The top and bottom boundaries are traction-free, i.e. $\overline{\mathbf{T}} = \mathbf{0}$. Plane strain conditions are assumed.

This problem has analytical solution and the deflection on the right-end of the beam is given as [56]:

$$\Delta = \frac{PL^3}{3\left(\frac{E}{1-\nu^2}\right)I} = 14.56 \times 10^{-3}\text{m} \tag{5.1}$$

where $P = 0.1 \times 10^6$ N is the applied load, $L = 2$ m is the length of the beam, and $I = \frac{bh^3}{12} = \frac{0.1^3}{12}$ m$^4$ m is the second moment of area of the beam about its bending axis. A metric defined as the difference between the predicted displacement and the analytical solution shows that both results from BC and NLBC match consistently (Fig. 5.3), reflecting the analytical proof of equivalence between the formulations inside the boundaries of the linearized elasticity framework.

The Neo-Hookean law was chosen for NLBC, but Hooke's law was also an option, since for small strain they are equivalent [42]. Only one correction step was sufficient for convergence of both methods. Finally, just for comparison, the SEG needs more than 23000 correction steps (using either nFVM or S4F) for mesh 60x3 cells.

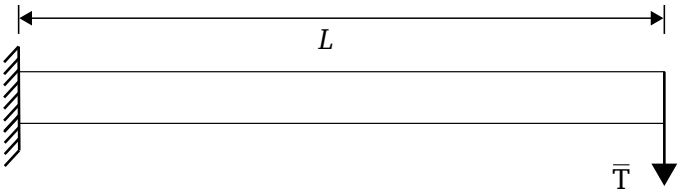



Figure 5.1: Geometry and boundary conditions for the slender cantilever beam in bending test case.

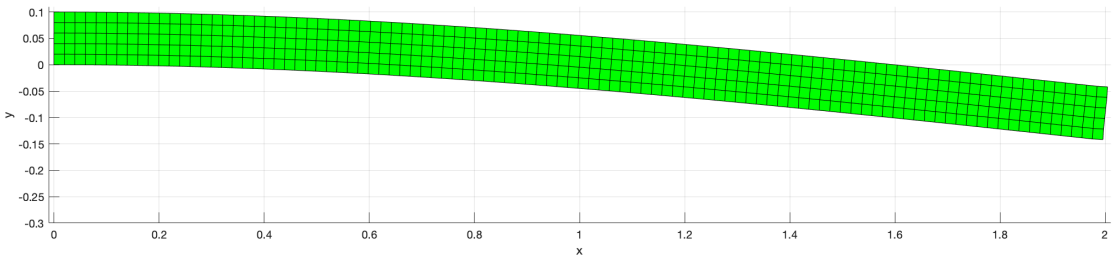



Figure 5.2: Deformed profile (scaled by factor of 10) for mesh 100x5 cells.

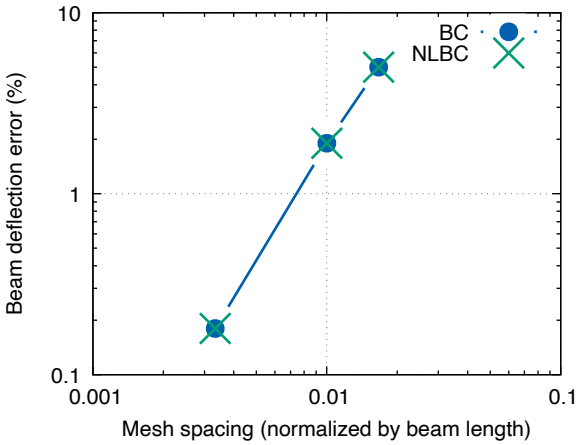



Figure 5.3: Error in cantilever end-deflection for different mesh refinements. Clearly the approaches match consistently.

## 5.2 Non-linear elasticity

The comparison of the NLBC method with the classical SEG solution procedure is given in this section by means of an examination of three representative test cases for large deformation, large rotation or both. In particular, uniaxial, shear and bending cases were investigated. All test cases were created using the accepted standard of verification testing, the Method of Manufactured Solutions (MMS), which allows validation against analytical solution [57]. A MMS test prescribes the deformation map $\varphi$, or any other map that allows one to recover it.

The Neo-Hookean and Ogden-Storakers laws were used to model the agglomerated cork AC216 [34] material. The former was used for all test cases. But, the latter was confined to the uniaxial compression case because only experimental data for compression was available. The following curve-fitting parameters

| $i$ | $\alpha_i[-]$ | $\beta_i[-]$ | $\mu_i[Pa]$ |
|---|---|---|---|
| 1 | 19.35 | 0.5423 | $1.982 \times 10^6$ |
| 2 | 28.58 | 2.0905 | $1.5478 \times 10^2$ |
| 3 | 19.9 | 0.3967 | $1.1045 \times 10^6$ |

(5.2)

were used to set Ogden-Storakers law. The density $\rho_0$ was set to 216 kg/m$^3$; the Young's Moduli $E$ and Poisson's ratio $\nu$ were set to 0.02 GPa and 0.3, respectively (for the Neo-Hookean law).

The first two tests (uniaxial and shearing) focus on elastostatics equations, where "time" defines the motion parametrically. Only one time step was considered. The body and inertia forces were disregarded. All simulations use the unit square domain and its five uniform discretization levels. In particular, five Cartesian meshes were considered: 3×3, 8×8, 16×16, 32×32 and 64×64 cells. The coarsest and finest meshes are shown in Figure (5.4).

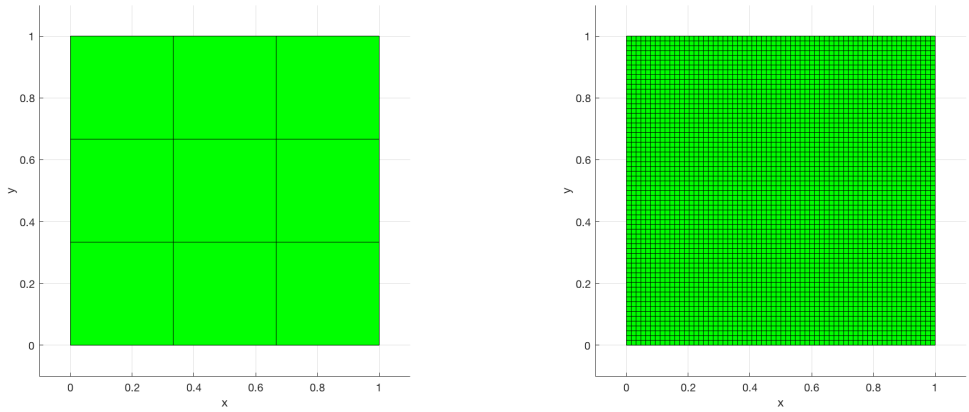



Figure 5.4: The Coarsest (3×3 cells) and the finest meshes (64×64 cells).

The following metrics were defined to quantify the difference between the predicted displacement and the analytical solution:

$$e_{\text{abs}} \begin{cases} \text{Mean error} = & \frac{1}{n_{\text{cells}}} \sum_{i=1}^{n_{\text{cells}}} r^i \\ \text{Max error} = & \max\{r^1, r^2, ..., r^{n_{\text{cells}}}\} \\ \text{Min error} = & \min\{r^1, r^2, ..., r^{n_{\text{cells}}}\}, \end{cases} \tag{5.3}$$

where $n_{\text{cells}}$ is the total number of cells composing the mesh, the sum is over all cells and considering a cell $C_a$, $r^a = |\mathbf{U}^{C_a}_{\text{calculated}} - \mathbf{U}^{C_a}_{\text{analytic}}|$. Every test case was split into two versions: one for displacement-only (Dirichlet) boundary conditions and another for traction-only boundary (Neumann) conditions (except for one boundary, which is set

to zero-displacement in order to avoid rigid-body motions). This split scheme isolates patterns which arise due to different boundary condition discretizations employed by NLBC and errors from each one can be investigated individually.

### 5.2.1 Uniaxial test cases

Two homogeneous uniaxial strain MMS were simulated using only nFVM. The deformation gradient $\mathbf{F}$ is the mapping prescribed for these cases and it is given as:

$$\mathbf{F} = \begin{bmatrix} \phi(t) & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \text{where} \quad \phi(t) = 1 + (\Lambda - 1)t \quad \text{and} \quad 0 \leq t \leq 1. \tag{5.4}$$

Note that $\mathbf{F}$ is homogeneous, i.e. does not depend on a material point $\mathbf{X}$. The deformation map is defined as: $\mathbf{x} = \boldsymbol{\varphi}(\mathbf{X}) \equiv \mathbf{F} \cdot \mathbf{X}$ and it is used to set the displacement boundary condition by imposing

$$\overline{\mathbf{U}} = \mathbf{x} - \mathbf{X} \tag{5.5}$$

at the boundary face centroids. A traction boundary counterpart can be set by noting that a traction $\mathbf{T}$ acting on the face with unit normal $\mathbf{N}$ is

$$\overline{\mathbf{T}} = \widetilde{\mathbf{P}}(\nabla \mathbf{U}) \cdot \mathbf{N} = \widetilde{\mathbf{P}}(\mathbf{F} - \mathbf{I}) \cdot \mathbf{N}. \tag{5.6}$$

#### Compression for displacement boundary

A variation of the homogeneous uniaxial strain test case described in [57] is presented in this section. However, instead of traction, displacement boundary condition was adopted. The Neo-Hookean and Ogden-Storakers laws were employed for this case. Two compression levels were investigated by assigning different values for the compression factor $\Lambda$, in particular, $\Lambda = 0.65$ and $= 0.1$ (see Fig. 5.5 and 5.6).

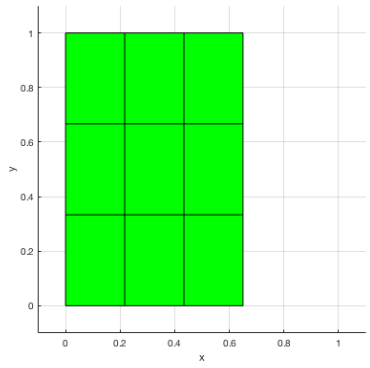



Figure 5.5: The final deformed domain at compression level $\Lambda = 0.65$.

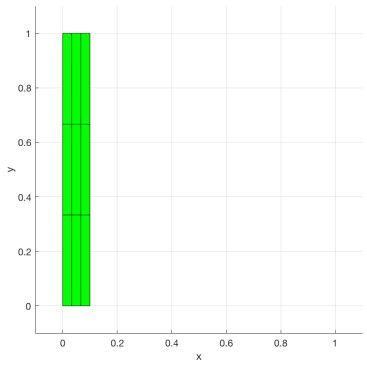



Figure 5.6: The final deformed domain at compression level $\Lambda = 0.1$.

The computed solution with the coarsest mesh was already enough to produce $e_{\text{abs}} < 10^{-16}$, regardless the method and material law, for $\Lambda = 0.65$ (see Fig. 5.7). The convergence in all scenarios was achieved with only one correction step, i.e. $n_{\text{corr}} = 1$. When $\Lambda$ is decreased to $0.1$, the SEG method produces $e_{\text{abs}} < 10^{-9}$. The errors for NLBC also increase when $\Lambda$ get smaller, but they are still relatively small ($e_{\text{abs}} < 10^{-13}$) and only one correction is needed, considering any mesh.

### Compression for traction boundary

Just changing from Dirichlet to Neumann makes the convergence a challenge for both methods, in particular, they are not able to simulate big compressions. The summarized results gathered from simulations are:

- The SEG method converges only when using the 3×3 cells mesh and $\Lambda \geq 0.8$, but with relatively high errors ($e_{\text{abs}} > 10^{-2}$).
- The NLBC method also converges only for $\Lambda \geq 0.8$ and provided that meshes are more refined than or equal to the mesh 16×16. For these scenarios, $e_{\text{abs}} < 10^{-7}$. The Neo-Hookean law was used.
- The NLBC method does not converge using Ogden-Storakers law (unless traction boundary condition on top and bottom boundaries is substituted by symmetry boundary condition).

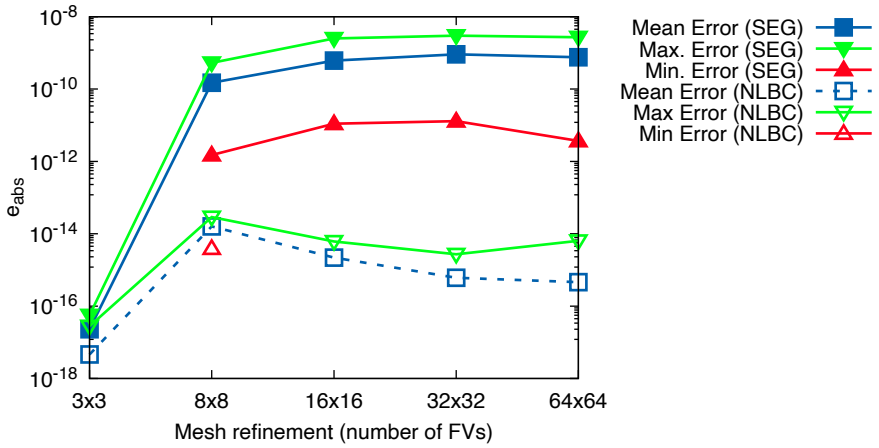



Figure 5.7: Errors from compression for displacement boundary test case using Neo-Hookean material. The missing data corresponds to when the difference between the solutions is below machine precision.

### Tension for displacement boundary

The cases above were repeated, but with $\Lambda > 1$ in order to simulate tension, in particular, $\Lambda = 2$ was adopted. The Figure (5.8) shows the final deformed domain for the coarsest mesh.

Interestingly, something changes when tension is simulated. Both methods converge for all meshes with errors $e_{\text{abs}} < 10^{-8}$ (see Fig. 5.9). Note that NLBC produces significantly smaller errors. Both methods converge with only one correction step.

### Tension for traction boundary

Once again, when traction is introduced, SEG does have convergence problems. In fact, it does not converge for $\Lambda$ much greater than one. And even when $\Lambda$ is close to one, e.g. 1.2, the errors are relatively high (either with nFVM or S4F). Regarding NLBC's results, they show good agreement with analytical solution. The method converges for all meshes, for any $\Lambda \in (1, 2]$ and the errors are relatively small ($e_{\text{abs}} < 10^{-6}$), but much higher than

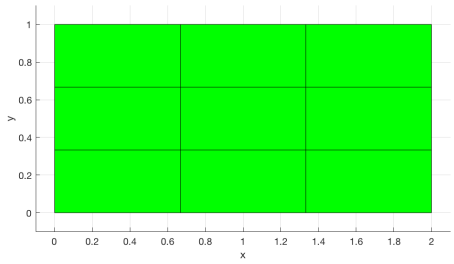



Figure 5.8: The final deformed domain for tensile strain case and for displacement boundary condition.

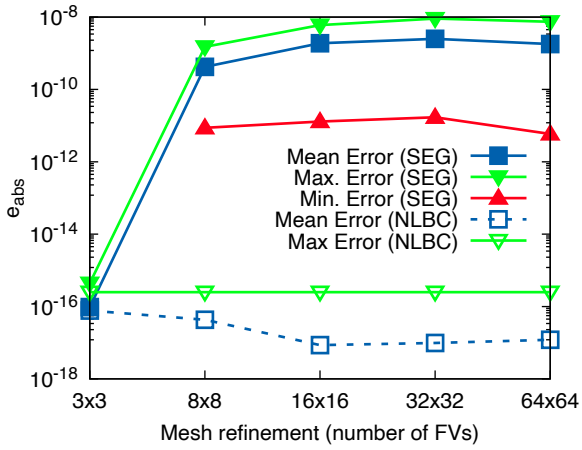



Figure 5.9: Errors from tension for displacement boundary test case. The missing data corresponds to when the difference between the solutions is below machine precision.

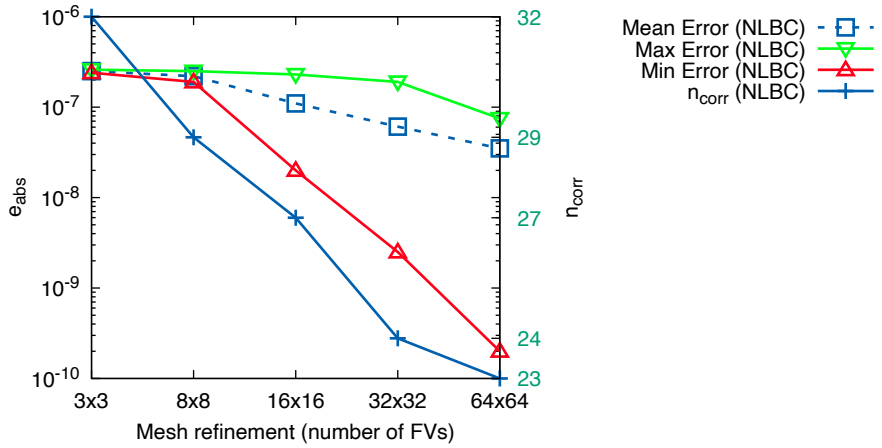



Figure 5.10: Error in tension for traction boundary test case as mesh is refined. The $n_{\text{corr}}$ as mesh is refined is also shown. Results are only for NLBC, since SEG method could not handle this case.

the corresponding test which uses displacement boundaries (see Fig. 5.10 and compare with Fig. 5.9).

### 5.2.2 Shear test cases

This test case consists of a simple shear [42]. The deformation gradient for this manufactured solution is very similar to that of the uniaxial test case and is given by (being the shear factor $\omega = 0.45$ chosen arbitrarily):

$$F = \begin{bmatrix} 1 & \phi(t) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \text{where} \quad \phi(t) = \omega t \quad \text{and} \quad 0 \leq t \leq 1. \tag{5.7}$$

The Figure (5.11) shows the deformed profile for mesh 16×16. The boundary condition is imposed in the same manner as it was done in uniaxial test cases.

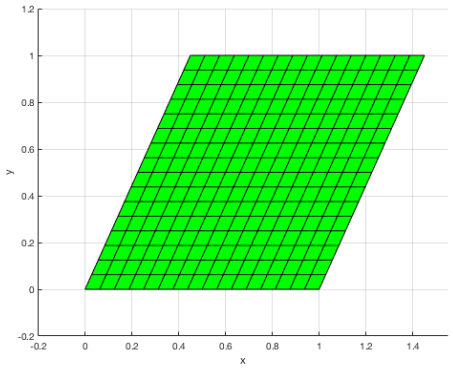



Figure 5.11: Deformed profile for mesh 16×16 in shear test case.

### Shear for displacement boundary

The results from simulations were qualitatively similar to that of the uniaxial compression, or tension, for displacement boundary (compare Fig. 5.7 and 5.9 with Fig. 5.12). The nFVM's SEG and NLBC needed only one correction for all meshes. The S4F's SEG was also tested and the output shows that as mesh gets refined, it needs more corrections to achieve convergence ($n_{\text{corr}} = 23, 30$ and $35$ for meshes 3×3, 8×8, 16×16 respectively). Besides, it did not converge for meshes finer than 16×16.

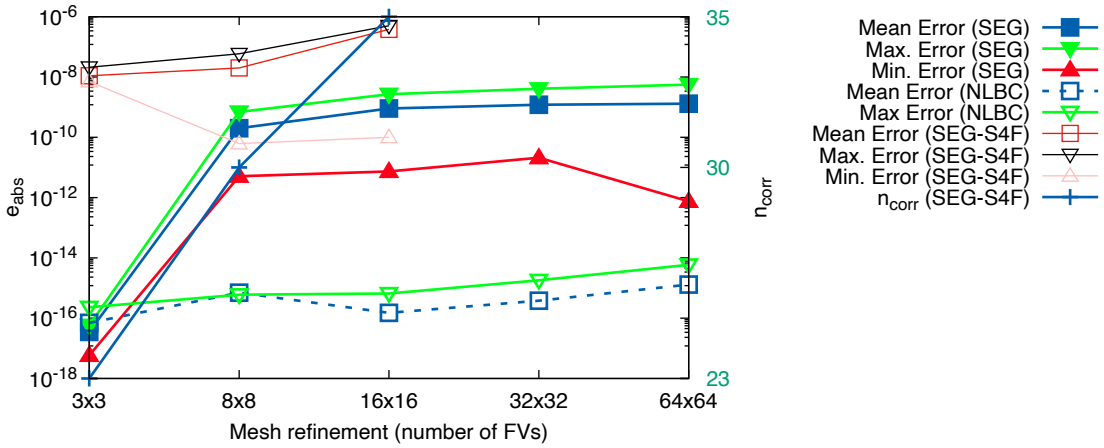



Figure 5.12: Error in shear for displacement boundary test case as mesh is refined. The number of correction $n_{\text{corr}}$ as mesh is refined for S4F's SEG is also shown. The SEG and NLBC implementations in nFVM needed only one correction to achieve convergence.

### Shear for traction boundary

Once more, when displacement boundaries are replaced by traction boundaries, the SEG method has convergence problems (both in nFVM and in S4F). In fact, convergence is achieved, however with relatively high errors (see Fig. 5.13). The SEG approach needs more than 100 correction steps to converge and for the finer the mesh, more correction steps are necessary for convergence (see Fig. 5.14).

The results from the NLBC method were in good agreement with analytical solutions and only one correction was needed in order to achieve convergence (see Fig. 5.14) using any mesh.

The final deformation domain (Fig. 5.15) for SEG is clearly "warped" (and refining the mesh does not reduce this spurious artifact).

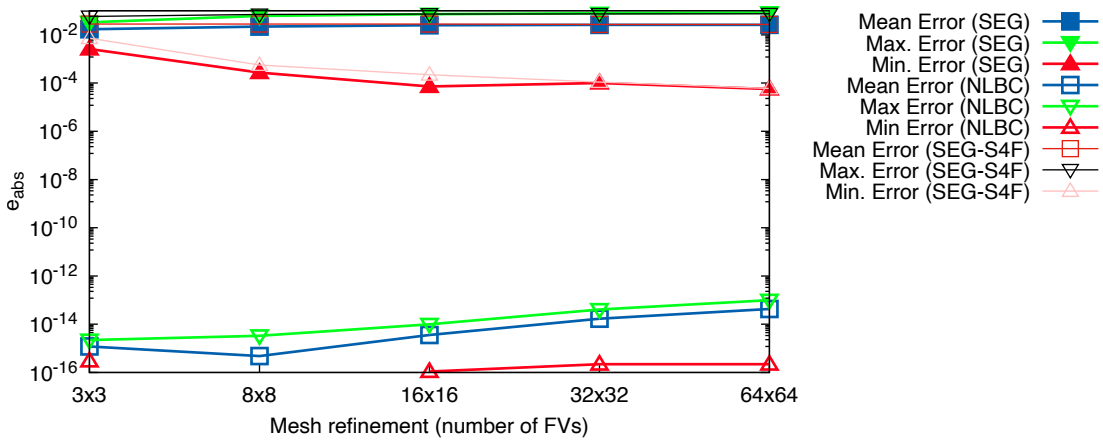



Figure 5.13: Error in shear for traction boundary test case as mesh is refined.

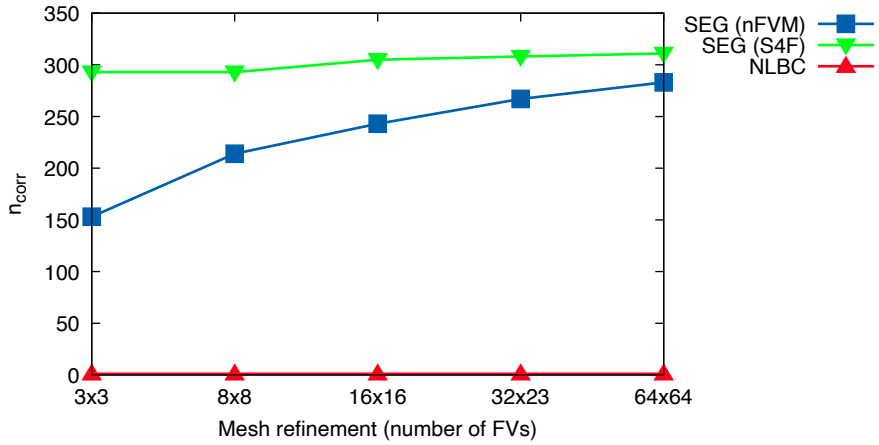



Figure 5.14: Number of corrections in shear for traction boundary test case as mesh is refined. The $n_{\text{corr}}$ for NLBC is 1.

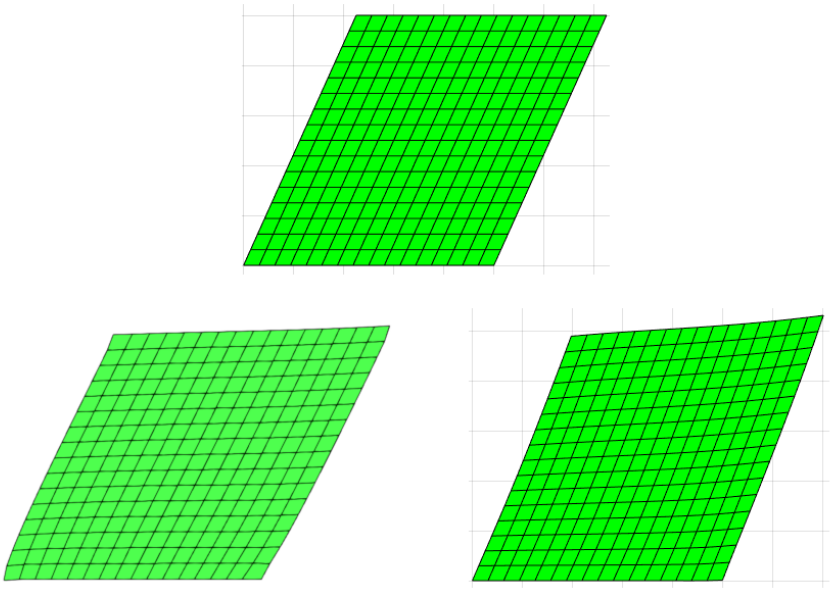


Figure 5.15: Deformed profiles for SEG using S4F (lower-left corner) and nFVM (lower-right corner). On top the result from NLBC. Traction boundary was used.

### 5.2.3 Bending test cases

In this section, it is examined the bending bar transient test case from [57]. This is more complete than the previous test cases because all terms in the momentum equation (Eq. 2.24) are considered. This MMS test comprises large deformation and large rotation of a thick vertical beam due to action of body force, acceleration and prescribed displacement (or traction) on the boundary. Effectively, all material points undergo an identical deformation mode: uniaxial strain with superimposed rotation (see Fig. 5.16).

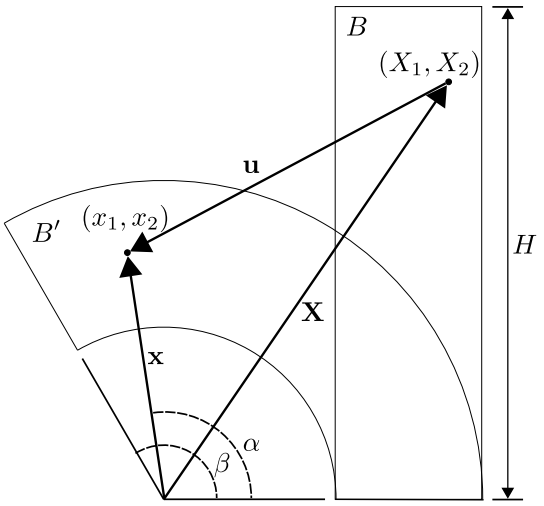



Figure 5.16: Snapshot of deformation in time for bending test case. In style of [57].

Unlike the previous, this case has a time and space-varying deformation gradient $\mathbf{F}$, which can be decomposed into rotation and stretch $\mathbf{F} = \mathbf{R} \cdot \mathbf{U}$, where $\mathbf{R}$ is the rotation tensor and $\mathbf{U}$ is the stretch tensor:

$$\mathbf{R} = \begin{bmatrix} \cos\alpha(t) & -\sin\alpha(t) & 0 \\ \sin\alpha(t) & \cos\alpha(t) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \Lambda(t) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{5.8}$$

$$\alpha(t) = \frac{\beta(t)X_2}{H}, \quad \Lambda(t) = \frac{\beta(t)X_1}{H} + 1 \quad \text{and} \tag{5.9}$$

$$\beta(t) = \frac{A(1 - \cos(\frac{2\pi t}{T}))}{2}, \tag{5.10}$$

where $\alpha(t)$ is the angle of rotation at the material point of interest, $\Lambda(t)$ is the amount of stretch in the 2-direction, $\beta(t)$ is the amplitude function, $A$ is the peak amplitude, $T$ is the stop time of the simulation, $t$ is time, $0 \leq t \leq T$ and $H$ is the height of the bar. The deformation map $\boldsymbol{\varphi}$ is given as

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \boldsymbol{\varphi}(\mathbf{X}, t) = \begin{bmatrix} -\frac{H}{\beta(t)} + \left(\frac{H}{\beta(t)} + X_1\right)\cos\left(\frac{\beta X_2}{H}\right) \\ \left(\frac{H}{\beta(t)} + X_1\right)\sin\left(\frac{\beta X_2}{H}\right) \\ X_3 \end{bmatrix}. \tag{5.11}$$

In this MMS, it is assumed that all terms in the momentum equation are known except the body force. The acceleration $\ddot{\mathbf{U}}$ can be obtained from Equation (5.11) and the stress tensor $\mathbf{P}$ can be obtained from the given deformation gradient $\mathbf{F}$. Thus, the body force $\mathbf{b}_m$ can be obtained for any location and time.

The parameters for the following simulations are: $H = 1$ (the domain is still the unit square); $A = \pi/7$, number of time steps $n_t = 3$; and $T = 1$. Only 0.5 second is simulated (see Figures 5.18 and 5.19).

### Bending for displacement boundary

The results from this test are:

- NLBC diverges for meshes more refined than the mesh 8×8. Changing the number of time steps does not change the scenario. It also diverges if the peak amplitude $A$ is greater than $\pi/7$;
- Still considering NLBC, the simulation with mesh 16×16 (and more refined) starts diverging after some corrections steps, but only at the last time step;
- The Segregated method converges for all meshes with relatively small errors (Fig. 5.17).

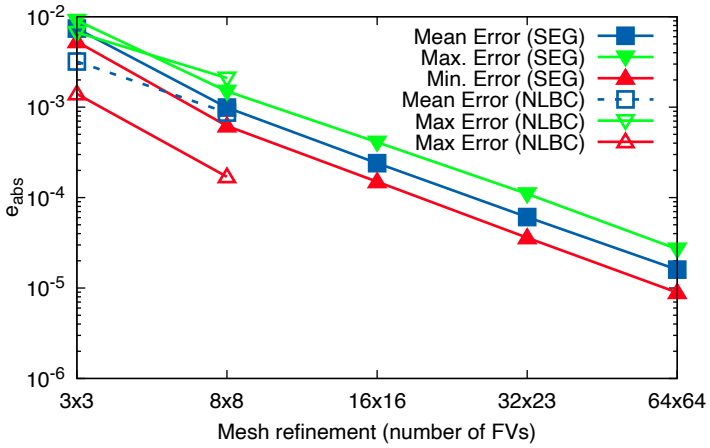



Figure 5.17: Errors from bending for displacement boundary test case as mesh is refined. The missing data corresponds to when NLBC diverges.

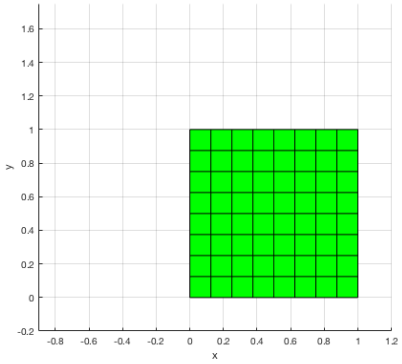


Figure 5.18: The initial undeformed mesh for mesh 8×8.

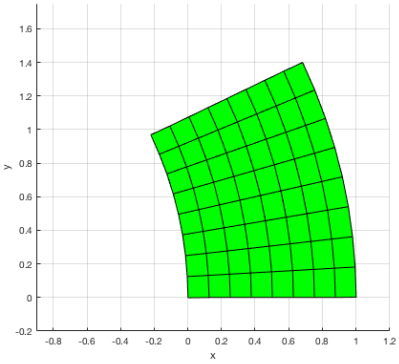


Figure 5.19: Deformed profile at the last time step for mesh 8×8.

## Bending for traction boundary

Both SEG and NLBC seems to be unable to run this test case. Either the simulations diverge at the second time step or the errors are very high. Again, the current traction boundary condition discretization, at least in NLBC, is causing divergence.

## 5.3 Conclusions

In this chapter, the NLBC methodology was investigated by means of numerical simulations. The other approaches were considered as well, namely, SEG and BC. Results from several test cases were examined and the findings are summarized as follows:

- The convergence rate of NLBC is impressive when compared to that of SEG's;
- The NLBC method has shown to be superior when comparing with BC, since the latter is restricted to linearized elasticity;
- In all test cases, except the bending problem, NLBC approach was much superior than SEG approach, i.e. in regard to convergence rate and divergence;
- Test cases without traction boundary condition, when converging for all time steps, NLBC does that with relatively small errors;
- The NLBC's convergence problems from traction discretization is case-dependent;
- The tests clearly showed that the adopted traction boundary condition discretization needs to be improved;
- The "borrowed" (from BC) traction boundary discretization for NLBC seems to cause divergence.

# 6 FOAM's approach in a high-level programming environment

## 6.1 Introduction

As already seem, FOAM stands for Field Operation And Manipulation, a concept conceived to be a C++ class library for Computational Continuum Mechanics (CCM). It was unveiled in the 1990s with a clear intention, as stated by Weller et al. [38]: "*to make it as easy as possible to develop reliable and efficient computational continuum-mechanics codes*". Roughly speaking, it would be a tool devoted to give numerical solution of partial differential equations on unstructured polygonal meshes by means of FVM. The breakthrough innovation consisted in a tensorial approach to CCM using object-oriented techniques and operator overloading. Therefore, instead of manipulating individual floating-point values (and arrays) and using only procedural programming, a new abstraction layer would be added. Furthermore, it is considered, from the viewpoint of tensor calculus, appropriate for FVM, where the discretization can be formulated in the physical space on unstructured polygonal meshes [38].

In the computer science field, when a language is specialized to a particular application domain, it is called Domain-Specific Language (DSL) [58]. Therefore, the FOAM concept creates an embedded[1] DSL, i.e. a language that has its own look, feel and semantics, where the application domain is: tensor field algebra/calculus and solution of partial-differential equations on meshes. Note that, although primarily conceived in C++, the FOAM concept can be implemented in any other programming language, provided there exists a minimum support for user-defined types and operator overloading.

[1] The C++ programming language would be the "host language".

As it is shown in Section (6.4), nFVM re-implements the FOAM-C++ DSL on top of another DSL, the Matlab[2] language, which has numerical computing and analysis as application domain. The relevant features to implement FOAM concept, present in the Matlab environment, i.e. combination of software and language, are analyzed in Section (6.3). Note that every statement in this chapter, related to Matlab, should be mostly valid for its open-sourced alternative called Octave, since the latter is largely compatible with the former [59].

OpenFOAM is the open-sourced FOAM library that, in the past decade, has become one of the most popular library for CFD and multi-physics simulations [60, 61]. Furthermore, it has a widespread adoption both in academia and in industry [32]. In OpenFOAM, solvers are executables, i.e. stand-alone programs, each designed for a specific group of problems, e.g. one for heat conduction, another for isothermal laminar Newtonian incompressible Navier-Stokes equations and also for small strain/rotations Hookean elastic equations.

Recently, the S4F toolbox (for OpenFOAM) has emerged. As Cardiff et al. [32] says, "[it] aims to generalize the OpenFOAM design further to allow straightforward implementation of advanced solid mechanics and fluid-solid interaction procedures". Instead of stand-alone programs, it adopts a class-oriented solvers approach, i.e. a solver is encapsulated in a class, opening the possibility of, when dealing with FSI, combining different solvers. In practice, this is mostly accomplished by defining abstract classes representing the concepts of fluid, solid and fluid-coupling. Since each abstract class defines an interface, a protocol, the implementation of a new, say fluid solver, is a matter of realizing[3] the fluid abstract class, or just subclassing an existing solver and making the necessary changes (overriding functions, for example).

OpenFOAM and S4F together have been successfully applied to many complex problems, such as, eFSI for finite elasticity [60], transient coupled temperature-displacement analysis, transient viscoelastic analysis [32], FSI benchmarks [37], metal forming [54], contact mechanics [24] and reactor dynamics [62], proving to be useful tools. Nevertheless, they leave untouched a gap that remains open, in particular, one between concept development and concept implementation in a high-performance programming language. Actually, both are sub-optimal tools for prototyping/experimenting new FV methodologies, models or algorithms. Two major factors for this scenario can be identified.

It is well known that the problem of error isolation and correction, i.e. debugging, is ubiquitous during the prototyping stage and it can take considerable amount of time.

[2] Matlab is a software and a language at the same time.
[3] In UML jargon, implement the behavior specified by the inherited abstract class.

This important process is indeed not trivial for OpenFOAM (and for S4F). To create a solver or other tool for OpenFOAM, as stated by Damián et al. [63], "*a deep knowledge is needed concerning with classes structure, for value storage in geometric fields and also for matrices resulting from equation systems, becoming a hard task for debugging*". For example, values examination gets hard when viewing the desired data involving polymorphism and inheritance connected with the virtual methods used by the library. For this, it is necessary to walk through the class tree looking for the desired data members. Once the members are found, these maybe do not directly represent the desired information. Furthermore, this exploratory task requires relatively strong knowledge in pointers, let alone the necessary usage of complex sentences during the inspection. Moreover, things can get even more complex when *inline* functions and/or macros are involved [63].

The problem above motivated the creation of a tool (or a macro layer) called gdbOF [63], attachable to the general, *defacto* standard, C++ debugger gdb (GNU debugger) [64]. The calling of a gdbOF macro triggers a sequence of actions that include OpenFOAM class tree navigation, data collection and reordering for representation in an user readable format or formated appropriately to be imported in Matlab/Octave, so that inspection can actually happen. Of course these also applies for S4F as well. See Listing (6.1) for comparison between debugging expressions.

```
1   $(gdb) p *(U.boundaryField_.ptrs_.v_[0].v_)    // gdb expression (two lines)
2      @(U.boundaryField_.ptrs_.v_[0].size_)
3
4   $(gdb) ppatchvalues U 0    // gdbOF expression
5
6   K>> U.boundary(1).data    // nFVM expression
```

Listing 6.1: View boundary values of a vector field $\mathbf{U}$ using gdb, gdbOF and nFVM, respectively. It is assumed that the first boundary is inspected (in Matlab, the indexing starts at 1). The gdb expression is long and complex. The gdbOF's is simpler, but the user has to memorize syntax and names (`ppatchvalues` stands for *print patch values*). In nFVM, tab completion in the Matlab Command Window facilitates expression composition. In addition, only in nFVM, the expression being written is the same found outside the debugging process.

The second factor relates to the nature of C++. It seems that academic researchers have found dynamically-typed programming languages (e.g. Matlab and Python) more productive than statically-typed languages (e.g. C++ and Java). A quote from Okon et

al. [65]: "*Many researchers today do their day-to-day work in dynamic languages*". As stated by Sargado et al. [66], "*it can be argued that there is now a clearer divide between academia and industry, with most of the programming work related to implementation of new approaches and models being done by academic researchers utilizing interpreted languages such as Matlab and Python*". Furthermore, "*The popularity of these platforms stems from the fact that they allow for rapid implementation, prototyping and visualization of results as well easier debugging due to access to intermediate states of variables during execution time. On the other hand, codes used to generate results in publications are often hand-tailored to the specific problems being solved and are impossible to apply without substantial modification to other cases*" [66]. In fact, these codes adopt a functional programming style, where a pre-determined flow of execution is used. More on dynamic languages in Section (6.2).

Taking the previous paragraphs into consideration, it seems reasonable to require that in a high level prototyping environment for CCM, using FV framework, three aspects must be present: I) a debugging process as easy as possible; II) the modular class-oriented solvers approach adopted by S4F; and III) a dynamic language must be preferably the primary one having numerical computing and analysis as application domain. To the present time, no such environment has been unveiled as far as the author is aware.

This chapter shows that the implementation of FOAM approach using Matlab[4], i.e. the high-level programming environment designed for numerical computations and analysis, fills the current open gap. As a concept demonstration, the nFVM, a CCM Matlab toolbox based on FVM, is presented. As said in the previous chapter, this library has the three FV methodologies already discussed, SEG, BC and NLBC. They were implemented following the same class-oriented solvers approach taken by S4F.

## 6.2 Dynamically-typed programming languages

The convenience of dynamic languages comes from their support of the duck typing feature, also known as cross-hierarchy polymorphism [67, 68]. It indicates that one variable may bind to objects whose types are unrelated, i.e. do not have any common parent implementing/requiring a method invoked on that variable. This feature matches conveniently with the prototyping processes, giving a lot of freedom to programmers, in the sense that they do not need to define interface classes (abstract ones in C++ and Matlab) at all. It is as if the language type-checking system was turned off in this case.

[4] As said before, the same applies to Octave.

Thus, no need to start the process by first elaborating abstract concepts, e.g. composing class hierarchies. This step can be postponed until after the moment when the code is already running as expected. That is, cycles of "implement, check if the results are consistent, extract abstract/general concepts from the implementation, then apply code refactoring[5]" can be applied during prototyping processes. This approach was successfully applied during the development of the nFVM toolbox.

On the other extreme, in C++ it is not possible to turn off the language type-checking system, i.e. it is completely rigid. Therefore, every change in types might be prevented by type constraints, even if the programmer intends just to test a localized piece of code for a moment. Furthermore, realizing these considerations, one can safely conclude that systems in dynamically languages with duck typing are intrinsically more extensible, thus more flexible for prototyping, than statically-typed languages. See Tratt et al. [69] for a survey of dynamic languages.

In classic dynamically-typed languages, all code is polymorphic, i.e. the types of any values are rarely restricted (only by explicitly checking types or when objects fail to support operations at run-time). Thus, type-incompatibility errors only happen at run-time. However, within the statically-typed language paradigm, those kind of errors are checked, as much as it is possible, at compile-time. Therefore, the errors can be found before even executing the program. From a statically-typed language user perspective, these two considerations would demonstrate one of the major disadvantages of dynamic languages [69, 70].

Nevertheless, still in the context of errors, the debugging process also should be taken into consideration. It can be defined as a two-step procedure: i) errors detecting; and ii) errors fixing. Therefore, when a compiler is issuing a type-incompatibility error, it is doing the first step for the programmer, which in turn has to finish the process by doing the next step on his/her own. But note that type-incompatibility errors found during the compilation process, or because of a runtime crash, are almost always trivially corrected in the codes of both classes of languages (statically and dynamically-typed), although not necessarily found immediately in dynamic-language codes. When it comes to debugging errors in algorithms (logic errors), both the compiler and the runtime are almost always helpless, and a low-level interactive debugging environment together with low-level codes can make things even worse, as seen in the introduction of this chapter. So, depending on the type of application, having the combination of a high-level debugging environment and also high-level code, as offered by the dynamic language approach, might more than compensate for the lack of compile-time type check.

[5] It is the process of restructuring existing computer code without changing its external behavior.

Another major disadvantage concerns the run-time speed. It is well known that high-level dynamic languages are usually slow. But depending on the main goal, e.g. fast prototyping numerical concepts, it might be irrelevant. One can first "approve" a prototype using many test cases with small degrees-of-freedom, then re-implement it in a language suitable for computationally intensive problems as statically-typed language such as C++ and Fortran.

Last, but not least, the Section (6.5) discusses how the analysis tools of Matlab can be used to create a high-level numerical dynamic analysis environment in nFVM.

## 6.3 Matlab/Octave environment for prototyping

Graydon Hoare, the author of Rust, said once [70]: *"Programming languages are mediating devices, interfaces that try to strike a balance between human needs and computer needs. Implicit in that is the assumption that human and computer need[s] are equally important, or[and] need mediating"*. Ultimately, programs consist of data (that is held during runtime) and operations on data[6]. As the computer knows more about the data, it is better at executing operations on this data. The programming language data types are exactly this metadata. For human to describe this metadata, i.e. the types, a real effort is necessary. The more a dynamically-typed language dispenses with type definition, the greater the productivity, but also the lower the performance, as the compiler and the runtime cannot benefit from metadata information that is crucial to generate fast code. This is why dynamically-typed language is generally better suitable for prototyping. In terms of the above needs, the following ordered list can be established: (assembly, C++, Fortran, Matlab), where on one extreme assembly language favors much more computer needs, and on the other, Matlab, by means e.g. of removing all variable type declarations, largely favors human needs. Note that C++ should be on the left of Fortran specially for numerical computing (one can just consider, for non-trivial programs, the usage of C++ pointers to handle numerical data).

Fortran stands for FORmula TRANslating system. It was developed in 1953 to be an alternative to assembly. The first motivation to develop such a programming language was related to the fact that, back then, it was very complex to do numerical computation in assembly. For example, there was not a floating-point type and no way to write mathematical formulae in algebraic notation [71].

[6] Arrays, lists, graphs, constants, etc.

The second motivation relates to debugging in assembly, in particular involving numerical codes. At that time, it was very time-consuming (from one quarter to one half of the computer's time was spent in debugging) and it demanded a significant knowledge of the computer hardware instructions (this is why it is said to be a very low-level programming language) [71].

By implementing an easier way to entry equations into computer, i.e. by means of a DSL, Fortran became the "original numerical computing language" [70]. As a result, Fortran dominated the scientific and engineering application areas for decades [72].

Later on, in the 90's, object-oriented programming (OOP) was generally recognized to produce code that was easier to write, validate, and maintain than procedural techniques [38] (at that point in time, Fortran was only a procedural language[7]). By opting for OOP, offered by C++, FOAM designers abdicated a numerics-oriented language, i.e. Fortran, with a built-in numerical computation environment (well established numerical libraries). And at the same time, it also offered to the programmer a framework written much more in favor of compiler needs (when compared to Fortran). As a side effect, the previously mentioned debugging complexities have emerged.

By implementing FOAM concept in Matlab, to fill the above gap (the absence of a high-level tool for prototyping/experiment new FV methodologies, models or algorithms), a numerical computation (plus, as a bonus, analysis) environment is brought back, one that is much richer than "the original" one. Note that, compared to C++/Fortran, Matlab is slow. Luckily, during prototyping, speed is usually not the major concern.

Matlab stands for MATrix LABoratory. Its developer wanted the students to have easy access to LINPACK and EISPACK (linear algebra libraries), without writing Fortran programs. The facilitation was promoted by not going through edit-compile-link-load-execute process that was ordinarily required when using Fortran.

After almost four decades of improvements, today's Matlab is a high-level numerical dynamic language (when compared to C++, much closer to human needs). The most important features favoring the implementation of FOAM-Matlab are the following:

- it supports OOP and operator overloading;
- it has a modern numerical computation and analysis environment with a:

[7] OOP could only be partially simulated in Fortran 90 [73].

  - REPL[8] for testing/creating/exploring code, therefore it offers an "interactive computer programming environment";
  - vast library for numerical analysis, that can be used during runtime. In particular, tightly integrated graph-plotting features and data, matrix decomposition, "eigen-analysis", different linear solvers and many others;
  - Computer Algebra System (CAS), i.e. it supports symbolic computing;
  - built-in support for complex tensor manipulations;
  - facility of changing class/function definitions at runtime;
- class/function definitions can be changed at runtime time, a language feature called intercession, which is mostly found in dynamic languages [69];
- it has interactive documentation (via shell);
- it has numerical-friendly debugging system.

## 6.4 The nFVM toolbox

By adopting the OOP design of FOAM-C++ (but using the modular class-oriented solvers approach adopted by S4F), nFVM refutes the generally adopted approach when developing FV-based CSM codes in dynamic languages (as in [74]), i.e. hand-tailored code to a specific problem being solved (therefore making it impossible to apply to other problems without substantial modification).

The computational tensor field algebra (a DSL) in nFVM is possible thanks to the class hierarchies and operator overloading methods implemented in the same fashion as in FOAM-C++ [38]. For each tensor order (0,1,2), there are four field classes (see Fig. 6.1). For example, considering zero-order tensors, the `scalarField` is just a list of scalars with the operator overloading feature implementing arithmetic operations at list level. The `volScalarField`[9] adds an abstraction layer on top of `scalarField` to creates the notion of discrete tensor field, in mathematical sense, where the domain is the set of computational nodes. This FV field (just like the `volVectorField` and

[8] REPL means read–eval–print loop, it is an interactive program that accepts single user inputs (i.e. single expressions), evaluates (executes) them, and returns the result to the user.

[9] The prefix `vol` stands for volume, or more precisely, finite volume.

volTensorField for first- and second-order tensors) has access to the polyhedral mesh via an object of the type fvMesh.

Still considering Figure (6.1), it can be seen that volScalarField class relates to scalarField and bScalarField classes by means of composition, i.e. an instance of the first contains one instance of each of the latter classes. Because volScalarField relates to scalarField via a "has-a" relationship, the former has to re-implement the operator overloading. An instance of bScalarField has a list of objects of the type patchScalarField, one for each collection of faces associated to a boundary condition. Furthermore, to implement a boundary condition, inheritance is used, e.g. a "fixed-displacement" constraint is implemented by the class fixedDisplacement, which is a sub-class of patchScalarField. Thus, as a modular approach, implementing a new boundary condition is just a matter of subclassing patchScalarField or maybe a subclass of it.

As a result, with this arrangement, the top-level syntax of the code is as close as possible to conventional mathematical notation for tensors fields. The Listing (6.2) shows the generally adopted in Matlab array-oriented style to compute, for example, a material law (in this case the Hooke's law). In contrast, the level of abstraction obtained by adopting the tensor-oriented approach is much higher and it can be seen in Listing (6.3). See Listing (6.4) for a complete function used to compute the Neo-Hookean material law, written in nFVM. Check also Listing (6.5) for an even more complex expression, in particular having even tensor product. Access to an individual component of a tensor field is also possible, permitting the computation of "low-level" material law expressions (see Fig. 6.2).

The Figure (6.3) presents the class hierarchies and the relations between packages composing the CSM framework. It was designed to be flexible and extensible for developers to implement new FV solid methodologies and material laws. The abstract class SolidModelSolver, in the solidModel package, defines an interface (i.e. a "contract") to be implemented by a solid solver. As it can be seen, the three solvers discussed in this work are implemented by creating classes (i.e. BC, NLBC and SEG) that realize this interface. In fact, two additional classes are necessary, a subclass of MechanicalModel (from mechanicalModels packages), which is used to decouple a solver from a material law, and a subclass of MaterialLaw (from materialLaws package). The decoupling opens the opportunity, e.g. to implement a multi-material solid model as in S4F.

Remember that each FV solid methodology defines a set of supported material laws. This constraint is enforced by creating a material law interface associated to each methodology, e.g. the NLBCMaterialLaw abstract class (which is a subclass of MaterialLaw) is such an interface associated to NLBC.

A material library is also available and trivially extensible (because every material is a class), thus promoting reusability. Furthermore, because of the class-oriented approach (instead of the statically-based one implemented in S4F), complex (e.g. dynamic) materials can be easily created. Moreover, materials can be selected using tab completion. The library is actually a package, called `materials`. Every material class must implement the interface defined, indirectly, by one or more material laws. For example, the `Cork216` material implements the `OgdenStorakerean` interface, which in turn is required by a material law, in this case, `OgdenStoraker`.

Differently from S4F, nFVM uses a class-oriented approach to define problems, therefore test cases. In addition, by exploring the dynamic-language nature of Matlab, instead of defining a test case as a set of parameter-value pairs (as in S4F), code written in Matlab can be used to set up a test case without resorting to external tools or to an edit-compile-execute process. The architecture is the following: there is an abstract class called `Problem` and, to a specific problem to be investigated, there must be a subclass, e.g. `Cantilever`, implementing the interface defined by `Problem` (see Fig. 6.3). Then, one or more test cases can be created instantiating the new class, in this case, `Cantilever`. Thus, such subclass defines, in a sense, a family of test cases. Furthermore, all members of a family can be changed at the same time, by just modifying one class. Moreover, making problems using inheritance promotes re-usability, e.g. a 2-D problem can be easily created by subclassing a 3-D problem class (see Fig. 6.3).

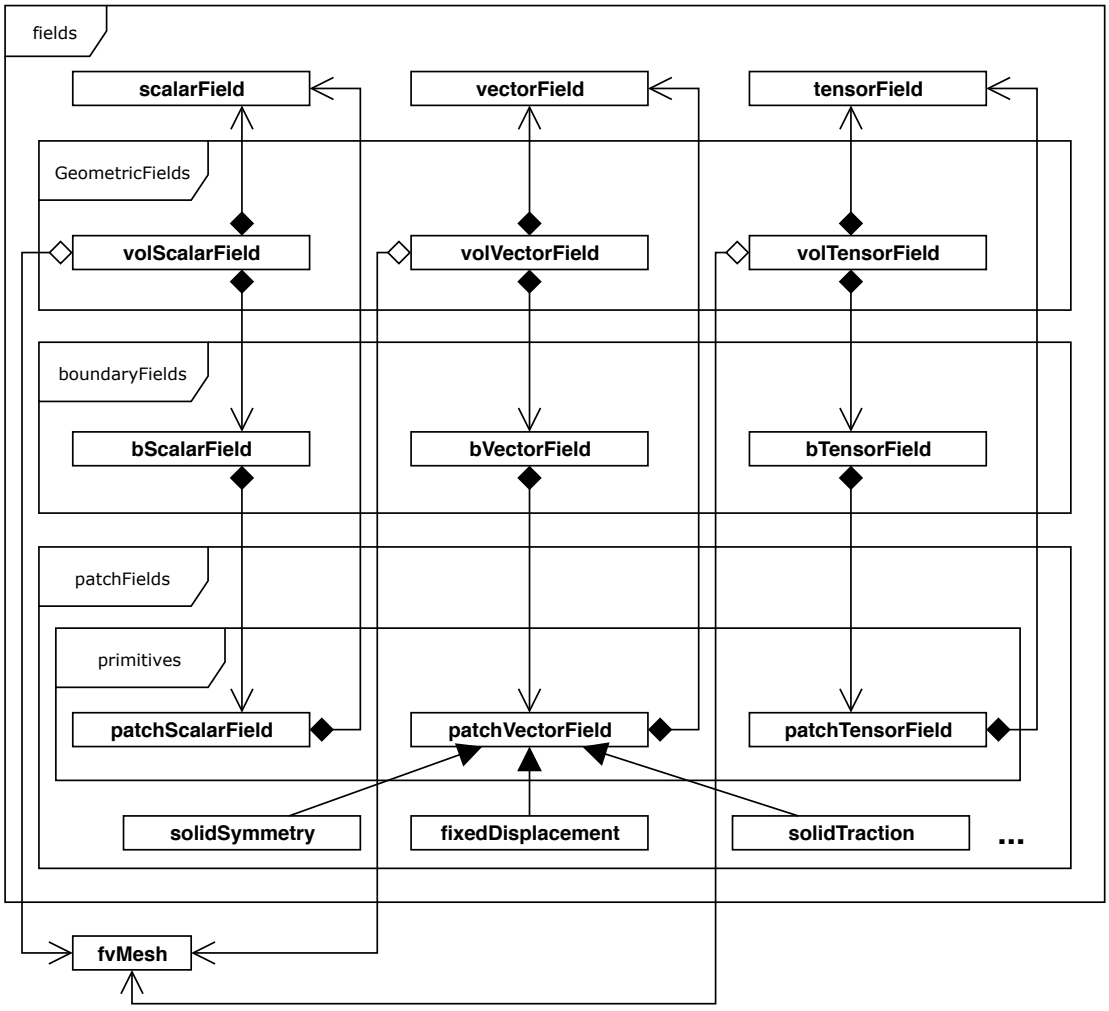



Figure 6.1: A simplified UML class diagram of the computational field classes.

```
1   [lambdaH_x,lambdaH_y,miuH_x,miuH_y]=...
2       Lambda_miu_Harmonic(N_x,N_y,lambda,miu);
3
4   for i=1:N_x+1
5       stress_xx(i,:)=-((2*miuH_x(i,:)+lambdaH_x(i,:)).*dUx_dx(i,:)+...
6           lambdaH_x(i,:).* dUy_dy_bar(i,:));
7       stress_xy(i,:)=-(miuH_x(i,:).*(dUy_dx(i,:)+dUx_dy bar(i,:)));
8   end
9
10  for j=1:N_y+1
11      stress_yy(:,j)=-((2*miuH_y(:,j)+lambdaH_y(:,j )).*dUy_dy(:,j)+...
12          lambdaH_y(:,j).*dUx_dx_bar(:,j));
13      stress_yx(:,j)=-(miuH_y(:,j).*(dUx_dy(:,j)+dUy_dx_bar(:,j)));
14  end
```

Listing 6.2: Matlab code excerpt from [74]. It shows the computation of the Hooke's law using the low-level and classical array-oriented approach. Note not only the profusion of indices and the usage of **for** loops (over computational points of the mesh), but also the necessary computation of each component of the stress tensor separately. Here, no DSL (for tensor field algebra) is used, only plain Matlab language.

```
1   [Mu, Lambda] = Lambda_miu_Harmonic(mesh, lambda, mu);
2
3   E = 0.5*(gradU + gradU.'); //  ≡ E = 0.5(∇U + (∇U)^T)
4   traceE = E.trace();
5
6   Sigma = 2*Mu*E + Lambda*traceE*I; //  ≡ Σ = 2μE + λ(tr E)I
```

Listing 6.3: Matlab code excerpt from nFVM. Operator overloading and tensor field classes concepts create a DSL that allows the syntax to closely resembles conventional mathematics notation. Compare with Listing (6.2). Here, no indices, **for** or mention to computational points of the mesh are needed. All happens at tensor field level.

```
1   H = tensorField(G.numberOfElements());
2
3   H(1,1) = G(2,2)*G(3,3) - G(2,3)*G(3,2);
4   H(1,2) = G(2,3)*G(3,1) - G(2,1)*G(3,3);
5   H(1,3) = G(2,1)*G(3,2) - G(2,2)*G(3,1);
6   H(2,2) = G(1,1)*G(3,3) - G(1,3)*G(3,1);
7   H(2,3) = G(3,1)*G(1,2) - G(1,1)*G(3,2);
8   H(3,3) = G(1,1)*G(2,2) - G(1,2)*G(2,1);
9
10  J3 = Sqrt(Det(G));
11  c = -(Beta + 2);
12
13  Sigma = mu*(I - (J3^c)*H);
```

$$
\begin{aligned}
H_{11} &= G_{22}G_{33} - G_{23}G_{32} \\
H_{12} &= G_{23}G_{31} - G_{21}G_{33} \\
H_{13} &= G_{21}G_{32} - G_{22}G_{31} \\
H_{22} &= G_{11}G_{33} - G_{13}G_{31} \\
H_{23} &= G_{31}G_{12} - G_{11}G_{32} \\
H_{33} &= G_{11}G_{22} - G_{12}G_{21} \\
J_3 &= \sqrt{\det \mathbf{G}} \\
c &= -(\beta + 2) \\
\boldsymbol{\Sigma} &= \mu \left( \mathbf{I} - J_3^c \mathbf{H} \right)
\end{aligned}
$$

Figure 6.2: Matlab code excerpt from nFVM and the corresponding mathematical expressions to show manipulation at tensor field component level. Given the `tensorField` $G$, the expression, say $G(1,1)$, returns a `scalarField`. Thus, access to, say $H(1,1)$, allows modification of all tensors, belonging to the tensor field $H$, at the same time, in contrast to modification of only one array component. This is the Blatz-Ko material law [75].

```
1   function Sigma = computeSigmaTensorField(obj, C, I)
2       % Computes the second piola stress given the right Cauchy-Green
3       %   strain tansor (as a tensorField). The equation is described
4       %   in [6.28, Bonet - 2008].
5       %
6       %   computeSigmaTensorField(C, I) returns a tensorField. The
7       %       arguments C and I must be tensorFields too.
8
9       lnJ = C.det().sqrt().log();
10      invC = C.inv();
11      Sigma = obj.mu()*(I-invC) + (obj.lambda()*lnJ)*invC;
12      //  ≡ Σ = μ(I − C^{-1}) + λ(ln J)C^{-1}
13  end
```

Listing 6.4: Matlab code excerpt from nFVM. It shows the implementation of the method that computes the Neo-Hookean material law (given by Equation (2.55) and shown in blue). The method belongs to the NeoHookean class. Note the absence of templates, type declarations, pointers and references that pervades the C++ (statically-typed) language.

```
1   function Td = Td(obj, F, n, t)
2       % Td returns (FC) : n t, | t = f^d,
3       %                (2,3)
4       % where n is the face normal and t is any vector. The inputs
5       % must be tensor, vector and vector fields respectively.
6
7       C = F.'*F;
8       lnJ = Log(Sqrt(Det(C));
9       invC = Inv(C);
10
11      mu = obj.mu();
12      lmb = obj.lambda();
13
14      one = Ones(lnJ);
15      A = F*invC;
16      b = invC*n;
17      Td = ((A*(lmb*n)) & (invC*t)) + ...
18           (mu*one - lmb*lnJ)*(((A*t) & b) + (b^t)*A);
19      //  ≡ T^d = λ(A · N) ⊗ (C^{-1} · f^d) + (μ − λ ln J)[(A · f^d) ⊗ b + (b · f^d)A]
20  end
```

Listing 6.5: Matlab code excerpt from nFVM comparing with the Neo-Hookean's $\mathbf{T}^d$ given by Equation (2.59) and shown in blue. Even a relatively complex tensor expression is readily understood. Note that the logical AND operator & is overloaded to create a tensor product operator for vector fields.

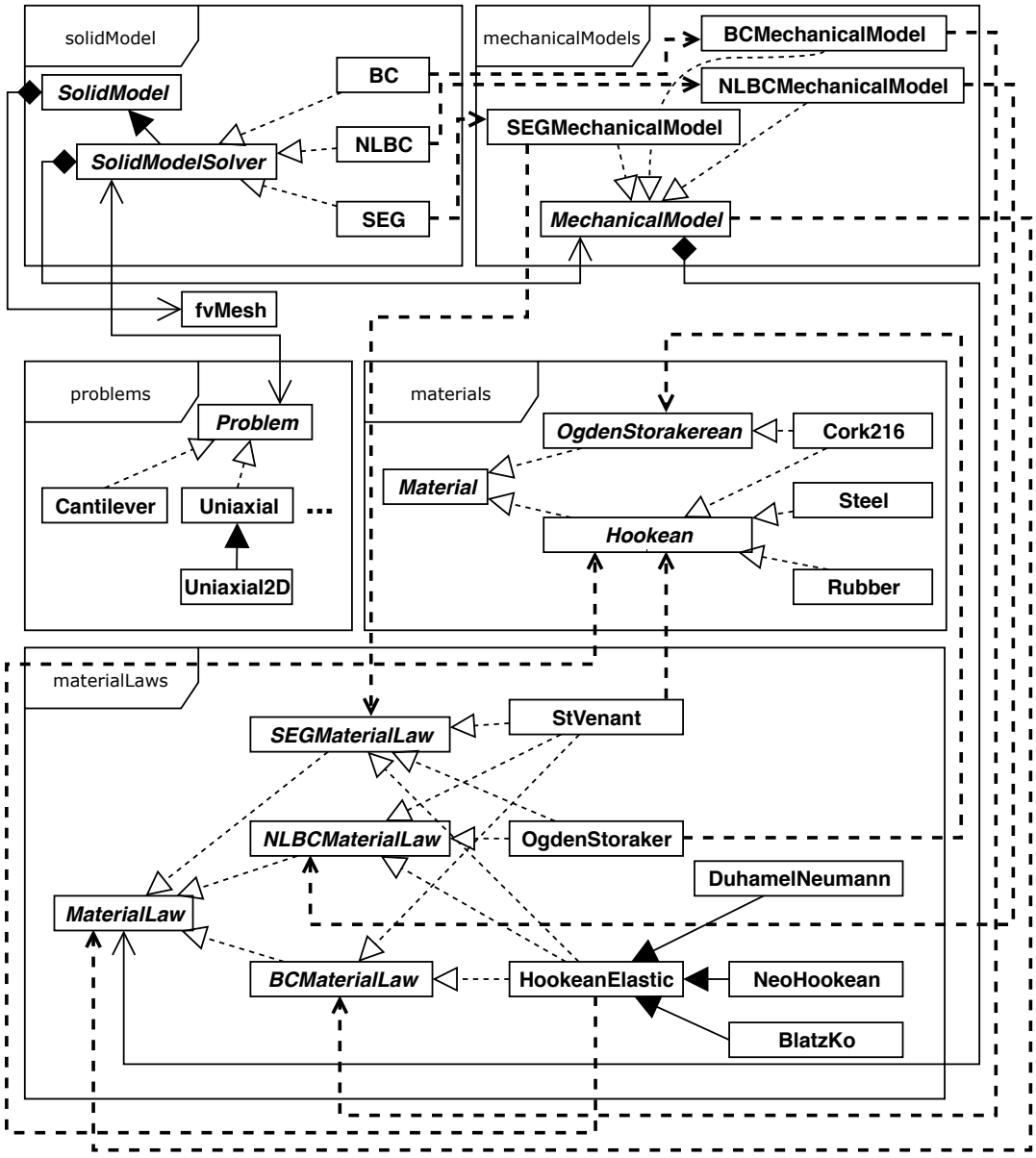



Figure 6.3: A simplified UML class diagram of the CSM framework implemented in nFVM.

## 6.5 A high-level analysis library in nFVM

In general, after an OpenFOAM simulation is finished, the generated data must be analyzed. There are many dedicated tools for such tasks, but paraView is the standard post-processing (and open source) tool that comes with OpenFOAM [61]. Note that, as a toolbox for OpenFOAM, S4F users usually resort to paraView too.

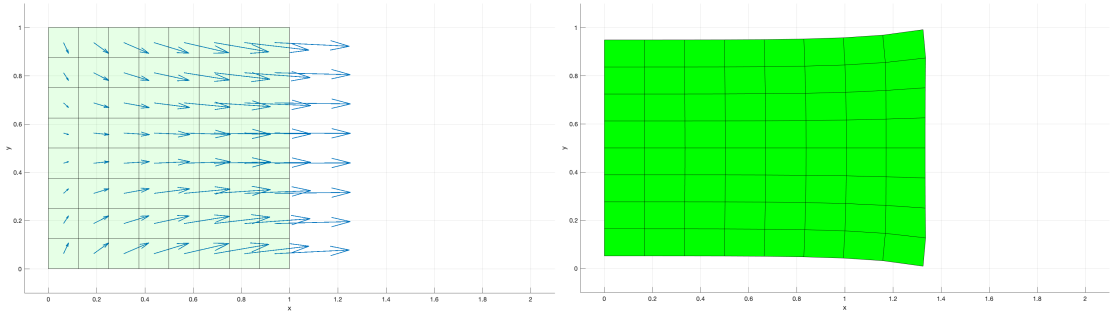
As already seen, Matlab has not only numerical computing, but also numerical analysis as application domain. This is due to the vast library for numerical analysis that can be used during runtime. Consequently, with nFVM the analysis can happen while simulating. That is, a running simulation can be paused at any time and inspection can take place. This characterizes a high-level analysis environment.

In addition, by wrapping some existing analysis functions in Matlab, higher-level functions can be created and grouped in a library. For example, instead of using the Matlab `plot` function, a new function called `plotVolVectorField` was created. As one can imagine, it accepts a `volVectorField` object, e.g. a displacement field, as input. This opens the opportunity to plot any vector field at any time during a simulation (see Fig. 6.4). And, consequently, a field resulting from a complex tensorial expression, written at simulation time, can be inspected using this function. Furthermore, no external library is needed and analysis functions can be easily incorporated/removed into or from the code.

## 6.6 Conclusion

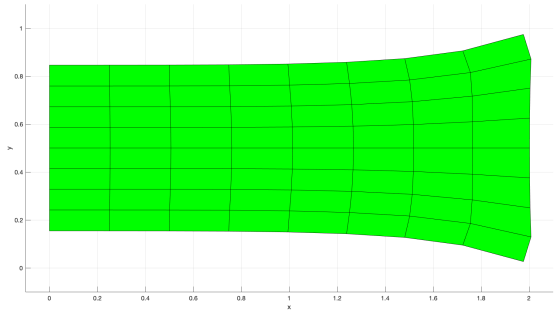
This chapter has presented:

- The core of the FOAM concept and why the current implementations in C++ are sub-optimal regarding (fast) prototyping;
- The reason dynamic languages should be used during prototyping phase;
- A “FVM-based CCM laboratory”, called nFVM, that fills the gap between theory and high-performance implementation. Furthermore, note that the development had the goal to make it as easy as possible not only to prototyping and investigation, but also to teach FV methodologies, concepts, algorithms and material laws at numerical computation level;



(a) Displacement field.

(b) Deformed profile.



(c) Final deformed profile.

Figure 6.4: One and the same uniaxial (with plane-symmetry on the left boundary) deformation case is shown. The images on the top (created using nFVM during runtime) show the deformed state at the first time step, while the lower one corresponds to the last time step. The displacement field shown was created using the `plotVolVectorField` function.

- The arguments that allow one to say: on the one hand OpenFOAM and S4F toolbox sacrifice prototyping in order to achieve high performance and, on the other hand, nFVM is exactly the opposite. In this sense, they complement each other;
- The improved problem definition system, where test cases are instances of classes that make parts of a hierarchy. Differently of a pair-value based problem system, the new one promotes re-use, modularity and flexibility;
- How nFVM creates a modular material library based on OOP.

# 7 Summary and outlook

The S4F toolbox offers a very attractive starting point to build a *rheinforce* composite solver. It is completely modifiable, has a strong-coupling FSI solver and its (fluid and solid) model solvers are modular (due to the adopted OOP technique). Furthermore, as the underlying methods are based on the FVM framework, the discretization is applied directly to the strong integral form of the governing equations, producing strictly conservativeness, a very important feature for FSI simulations. Moreover, its C++ code has been carefully written with high-performance as a goal.

Unfortunately, no solid solver available in S4F is able to compute the large strain necessary to simulate a large deformation of a three-dimensional virtual brick made of cork and modeled using the Ogden-Storakers material law. This limitation lies on the FVM methodologies themselves (all SEG manifestations and BC), not on their implementations. This frame originated the following motivation. Is there a possibility of combining both SEG and BC to produce a superior methodology so that its implementation is close enough to the existing implementations in S4F?

Creating a notable contribution to finite elasticity, this work gives an affirmative answer to the question above. The new methodology NLBC, which is mostly based on BC, has been designed for finite elasticity and it preserves, in a great extent, the fast convergence of BC and material generality (only requiring the elasticity tensor to have right-minor symmetry) of SEG. The first property is a consequence of the fact that all the displacement components are solved at the same time (instead of "segregating" them, as in SEG and as it is typically done in fluid dynamics) in a big linear system (similar to FEM). The second property emerges from the fact that NLBC uses an incremental approach in order to linearize a generic stress tensor (as opposed to BC, which fixes it) at each iteration.

As it is shown in Chapter 5, based on numerical results from representative test cases with analytical solutions (the presented MMS problems), the rate of convergence of NLBC is really impressive, i.e. when compared to that of SEG's. This turns out to be very important when considering partitioned FSI methods, which requires solving multiple times per

time step for both solid and fluid domains. Furthermore, it was demonstrated analytically and exemplified numerically (in Chapters 4 and 5, respectively) that NLBC has the same discretization as BC when the elasticity is restricted to the small strain regime.

Finally, as regards the implementation of NLBC in S4F, it can be readily done because the discretizations of the differential operators are the same employed by BC, and their codification are the only complex parts.

Considering the second goal of this work, it was clearly demonstrated that the tensorial approach to CCM using OOP with operator overloading and based on FVM, i.e. the FOAM concept, when promoted to a high-level numerical computation and analysis environment, offers huge benefits. In particular, it fills the very important "prototyping gap", i.e. the one between concept development and concept implementation in a high-performance programming language. The combination of a high-level dynamic language, which is a DSL with numerical computation and analysis domain, and the very popular Matlab/Octave desktop environment tuned for iterative analysis fills this gap.

An improved, and yet simplified, architecture of the flexible and extensible object-oriented framework present in the S4F toolbox has been established. This architecture, at the core of the nFVM toolbox, was verified since the three FVM methodologies were successfully used in a series of simulations.

Concerning the limitation of this work, because the current NLBC methodology requires the presence of a right-minor symmetric elasticity tensor, it defines a class of "officially" supported solid models (note that Hooke's material does not fulfill all the requirements, but nevertheless can give satisfactory results). In fact, it was demonstrated in Chapter 3 that a large set of important solid models (those that are hyperelastic, frame-indifferent, homogeneous and isotropic) belong to this class. As a matter of fact, frame-indifference should be required independently of the FVM methodology, since it is required in finite elasticity, otherwise different observers could collect different results [45]. This symmetric-related elasticity restriction should be subjected to investigation in order to establish if it can be removed or at least weakened.

The current traction discretization for NLBC ("borrowed" from BC) is causing divergence problems, as seen in Chapter 5. On the one hand, the BC method naturally "asks for" an implicit traction discretization because all the other discretizations do not require an iterative process. Therefore, it becomes necessary to solve only one linear system per time step. On the other hand, the NLBC is an iterative procedure (employs Newton-Raphson method). Thus it seems totally acceptable to use an explicit traction discretization. After all, iteration is inevitable anyway.

Note that the boundary discretizations employed in BC remove one of the most attractive aspect of the “classical” FVM: the ease to implement a variety of boundary conditions in a non-invasive manner. This is because the unknown variables are evaluated at the centroids of the volume, not at their boundaries. Using the suggested iterative approach would restore this rather precious property. Moreover, this would greatly reduce the number of computational points needed, reducing the size of the final linear system, since there would be no unknowns on the boundaries.

As regards mesh support, NLBC assumes that finite volumes are rectangular cuboids. However, by judging how other FV methodologies handle convex polyhedral, the modification to add support to it should be relatively straight-forward.

Ultimately, a convergence analysis of the NLBC has a great appeal, but it is far beyond the scope of the work.

# Bibliography


[1] EuroSafe. *Injuries in the European Union 2013-2015*. Tech. rep. Aug. 2017.

[2] V. H. P. Vitharana and T. Chinda. "Structural equation modelling of lower back pain due to whole-body vibration exposure in the construction industry". In: *International Journal of Occupational Safety and Ergonomics* 25.2 (2019), pp. 257–267.

[3] M. Dawson, G. Mckinley, and L. Gibson. "The Dynamic Compressive Response of an Open-Cell Foam Impregnated With a Non-Newtonian Fluid". In: *Journal of Applied Mechanics* 76 (Jan. 2009). DOI: 10.1115/1.3130825.

[4] F. J. Galindo-Rosales and L. Campo-Delaño. *Composite layer material for dampening external load, obtaining process, and uses thereof*. European Patent EP3201487A1.

[5] F. J. Galindo-Rosales and L. Campo-Delaño. *Composite layer material for dampening external load, obtaining process, and uses thereof*. U.S. Patent US10443678B2, Apr. 07, 2019.

[6] F. J. Galindo-Rosales and L. Campo-Delaño. *For inhibiting the composite material, its preparation method and purposes of external load*. Chinese Patent CN107208731B, May 31, 2019.

[7] G. M. Whitesides. "The Origins and the Future of Microfluidics". In: *Nature* 442 (July 2006), pp. 368–373. DOI: 10.1038/nature05058.

[8] P. C. Sousa et al. "Efficient microfluidic rectifiers for viscoelastic fluid flow". In: *Journal of Non-Newtonian Fluid Mechanics* 165.11 (2010), pp. 652–671. DOI: 10.1016/j.jnnfm.2010.03.005.

[9] P. C. Sousa et al. "High performance microfluidic rectifiers for viscoelastic fluid flow". In: *RSC Adv.* 2 (3 2012), pp. 920–929. DOI: 10.1039/C1RA00803J.

[10] C. Tsai et al. "High-performance microfluidic rectifier based on sudden expansion channel with embedded block structure". In: *Biomicrofluidics* 6 (June 2012), pp. 24108–241089. DOI: 10.1063/1.4704504.

[11] D. Kawale, J. Jayaraman, and P. Boukany. “Microfluidic rectifier for polymer solutions flowing through porous media”. In: *Biomicrofluidics* 13 (Feb. 2019), p. 014111. DOI: 10.1063/1.5050201.

[12] A. Groisman and S. R. Quake. “A Microfluidic Rectifier: Anisotropic Flow Resistance at Low Reynolds Numbers”. In: *Phys. Rev. Lett.* 92 (9 Mar. 2004), p. 094501. DOI: 10.1103/PhysRevLett.92.094501.

[13] K. Jensen et al. “Experimental characterisation of a novel viscoelastic rectifier design”. In: *Biomicrofluidics* 6 (Dec. 2012). DOI: 10.1063/1.4769781.

[14] K. Jensen, P. Szabo, and F. Okkels. “Topology optimization of viscoelastic rectifiers”. In: *Applied Physics Letters* 100.23 (2012), p. 234102. DOI: 10.1063/1.4728108.

[15] J. P. Rothstein and G. H. McKinley. “The axisymmetric contraction-ĂŞexpansion: the role of extensional rheology on vortex growth dynamics and the enhanced pressure drop”. In: *Journal of Non-Newtonian Fluid Mechanics* 98.1 (2001), pp. 33–63. DOI: 10.1016/S0377-0257(01)00094-5.

[16] F. J. Galindo-Rosales et al. “Microfluidic systems for the analysis of viscoelastic fluid flow phenomena in porous media”. In: *Microfluidics and Nanofluidics* 12.1 (2012), pp. 485–498. DOI: 10.1007/s10404-011-0890-6.

[17] E. Brown et al. “Shear thickening in densely packed suspensions of spheres and rods confined to few layers”. In: *Journal of Rheology* 54.5 (Aug. 2010), pp. 1023–1046. DOI: 10.1122/1.3474580.

[18] S. Majumdar, R. Krishnaswamy, and A. Sood. “Discontinuous shear thickening in confined dilute carbon nanotube suspensions”. In: *Proceedings of the National Academy of Sciences of the United States of America* 108.22 (May 2011), pp. 8996–9001. DOI: 10.1073/pnas.1018685108.

[19] F. J. Galindo-Rosales, S. Martínez-Aranda, and L. Campo-Delaño. “CorkSTF$\mu$fluidics – A novel concept for the development of eco-friendly light-weight energy absorbing composites”. In: *Materials & Design* 82.5 (May 2015), pp. 326–334. DOI: 10.1016/j.matdes.2014.12.025.

[20] F. J. Galindo-Rosales and L. Campo-Deaño. “Reinforced cork composites developed for helmet liner with improved”. In: *The 1st World Conference on Advanced Materials for Defense*. AuxDefense, Sept. 2018.

[21] J. Thijssen. *Computational Physics*. 2nd ed. Cambridge University Press, 2007.

[22] F. J. Galindo-Rosales. “Complex Fluids in Energy Dissipating Systems”. In: *Applied Sciences* 6.8 (2016). DOI: 10.3390/app6080206.

[23] F. Fernandes, R. J. S. Pascoal, and R. A. d. Sousa. “Modelling impact response of agglomerated cork”. In: *Materials and Design* 58 (June 2014), pp. 499–507. DOI: 10.1016/j.matdes.2014.02.011.

[24] P. Cardiff, A. Karač, and A. Ivanković. “Development of a finite volume contact solver based on the penalty method”. In: *Computational Materials Science* 64 (Nov. 2012), pp. 283–284. DOI: 10.1016/j.commatsci.2012.03.011.

[25] H. Jasak and H. G. Weller. “Finite volume methodology for contact problems of linear elastic solids”. In: *ICCSM*. CSM, Sept. 2000, pp. 253–260.

[26] L. Gibson and M. Ashby. *Cellular Solids: Structure and Properties*. 2nd ed. Cambridge University Press, 1999.

[27] M. Fortes and M. T. Nogueira. “The poison effect in cork”. In: *Materials Science and Engineering: A* 122.2 (1989), pp. 227–232. ISSN: 0921-5093. DOI: https://doi.org/10.1016/0921-5093(89)90634-5.

[28] F. A. O. Fernandes. “Biomechanical analysis of helmeted head impacts: novel materials and geometries”. PhD thesis. Mar. 2017.

[29] P. Santos et al. “Agglomerated cork: A way to tailor its mechanical properties”. In: *Composite Structures* (July 2017). DOI: 10.1016/j.compstruct.2017.07.035.

[30] R. Jardin et al. “Static and dynamic mechanical response of different cork agglomerates”. In: *Materials & Design* 68 (Mar. 2015). DOI: 10.1016/j.matdes.2014.12.016.

[31] U. Küttler, C. Förster, and W. A. Wall. “A solution for the incompressibility dilemma in partitioned fluid–structure interaction with pure Dirichlet fluid domains”. In: *Computational Mechanics* 38.4 (Mar. 2006), pp. 417–429. DOI: 10.1007/s00466-006-0066-5.

[32] P. Cardiff et al. *An open-source finite volume toolbox for solid mechanics and fluid-solid interaction simulations*. 2018. arXiv: 1808.10736 [math.NA].

[33] A. M. Brill et al. “Viscoelastic Shear Analysis of Polymeric Foam Midsoles”. In: *ANTEC*. SPE, May 2016.

[34] F. Fernandes et al. “Comparing the mechanical performance of synthetic and natural cellular materials”. In: *Materials & Design* 82.5 (Oct. 2015), pp. 335–341. DOI: 10.1016/j.matdes.2015.06.004.

[35] P. Cardiff and I. Demirdžić. *Thirty years of the finite volume method for solid mechanics*. 2018. arXiv: 1810.02105 [math.NA].

[36] P. Cardiff et al. “A block-coupled Finite Volume methodology for linear elasticity and unstructured meshes”. In: *Computers & Structures* 175.15 (Oct. 2016), pp. 100–122. DOI: 10.1016/j.compstruc.2016.07.004.

[37] Ž. Tuković et al. “OpenFOAM Finite Volume Solver for Fluid-Solid Interaction”. In: *Transactions of FAMENA* 42.3 (Oct. 2018), pp. 1–31. DOI: 10.21278/TOF.42301.

[38] H. G. Weller et al. “A Tensorial Approach to Computational Continuum Mechanics Using Object-Oriented Techniques”. In: *Computers in Physics* 12.6 (Nov. 1998), pp. 620–631. DOI: 10.1063/1.168744.

[39] *MATLAB version 9.5.0.1265761 (R2018b)*. The Mathworks, Inc. Natick, Massachusetts, 2018.

[40] O. Gonzalez and A. M. Stuart. *A First Course in Continuum Mechanics*. 1st ed. Cambridge University Press, 2008.

[41] R. W. Ogden. *Non-linear Elastic Deformations*. 2nd ed. Dover Publications, 1997.

[42] J. Bonet and R. D. Wood. *Nonlinear Continuum Mechanics for Finite Element Analysis*. 2nd ed. Cambridge University Press, 2008.

[43] S. Flügge, C. Truesdell, and J. W. Dally. *Encyclopedia of Physics: Mechanics of Solids II*. 1st ed. Springer-Verlag Berlin Heidelberg, 1972.

[44] G. A. Holzapfel. *Nonlinear Solid Mechanics. A Continuum Approach for Engineering*. 2nd ed. John Wiley & Sons, 2001.

[45] P. G. Ciarlet. *Mathematical elasticity, volume I: Three-dimensional elasticity*. 1st ed. Elsevier, 1988.

[46] I. Doghri. *Mechanics of Deformable Solids: Linear, Nonlinear, Analytical and Computational Aspects*. 1st ed. Springer, 2013.

[47] R. Hill. “Aspects of Invariance in Solid Mechanics”. In: vol. 18. Advances in Applied Mechanics. Elsevier, May 1979, pp. 1–75. DOI: 10.1016/S0065-2156(08)70264-3.

[48] B. Storåkers. “On material representation and constitutive branching in finite compressible elasticity”. In: *Journal of the Mechanics and Physics of Solids* 34.2 (1986), pp. 125–145. DOI: 10.1016/0022-5096(86)90033-5.

[49] H. Jasak. “Error Analysis and Estimation for the Finite Volume Method With Applications to Fluid Flows”. PhD thesis. Imperial College London, 1996.

[50] K. Maneeratana. “Development of the finite volume method for non-linear structural applications”. PhD thesis. Jan. 2000.

[51] F. Moukalled, L. Mangani, and M. Darwish. *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM and Matlab*. 1st ed. Springer, 2015.

[52] I. de Oliveira. "Using foam-extend to assess the influence of fluid-structure interaction on the rupture of intracranial aneurysms". PhD thesis. Aug. 2017.

[53] K.-J. Bathe. *Finite Element Procedures*. 1st ed. Prentice-Hall, 1996.

[54] P. Cardiff et al. "A Lagrangian Cell-Centred Finite Volume Method for Metal Forming Simulation". In: *International Journal for Numerical Methods in Engineering* (Aug. 2016). DOI: 10.1002/nme.5345.

[55] I.-S. Liu. "Successive linear approximation for boundary value problems of nonlinear elasticity in relative-descriptional formulation". In: *International Journal of Engineering Science* 49 (7 July 2011), pp. 635–645. DOI: 10.1016/j.ijengsci.2011.02.006.

[56] S. Timoshenko and J. N. Goodier. *Theory of elasticity*. 3rd ed. McGraw-Hill, 1970.

[57] K. Kamojjala et al. "Verification tests in solid mechanics". In: *Engineering with Computers* 31 (Apr. 2013), pp. 193–213. DOI: 10.1007/s00366-013-0342-x.

[58] M. Fowler. *Domain Specific Languages*. 1st ed. Addison-Wesley Professional, 2010.

[59] *GNU Octave*. https://www.octave.org.

[60] J. Nobrega and H. Jasak. *OpenFOAM®: Selected Papers of the 11th Workshop*. 1st ed. Springer, 2019.

[61] K. Mooney, T. Maric, and J. Höpken. *The OpenFOAM Technology Primer*. 1st ed. Sourceflux, 2014.

[62] I. Clifford and H. Jasak. "The Application of a Multi-Physics Toolkit to Spatial Reactor Dynamics". In: *Computational Methods & Reactor Physics*. American Nuclear Society, May 2009.

[63] S. M. Damián, J. M. Gimenez, and N. Nigro. "gdbOF: a debugging tool for openFOAM®". In: *Advances in Engineering Software* 47.1 (May 2012), pp. 17–23. DOI: 10.1016/j.advengsoft.2011.12.006.

[64] N. Matloff and P. J. Salzman. *The Art of Debugging with GDB, DDD, and Eclipse*. 1st ed. No Starch Press, 2008.

[65] S. Okon and S. Hanenberg. "Can we enforce a benefit for dynamically typed languages in comparison to statically typed ones? A controlled experiment". In: *ICPC*. IEEE, May 2016, pp. 1–10.

[66] J. Sargado. *A new object-oriented framework for solving multiphysics problems via combination of different numerical methods*. Apr. 2019. arXiv: 1905.00104 [cs.MS].

[67] R. Chugh, P. Rondon, and R. Jhala. "Nested Refinements: A Logic for Duck Typing". In: *SIGPLAN Not.* 47.1 (1 Jan. 2012), pp. 231–244. DOI: 10.1145/2103621.2103686.

[68] N. Milojković, M. Ghafari, and O. Nierstrasz. "It's Duck (Typing) Season!" In: *ICPC*. IEEE, May 2017, pp. 312–315.

[69] L. Tratt. "Dynamically Typed Languages". In: vol. 77. Advances in Computers. Elsevier, July 2009, pp. 149–184. DOI: 10.1016/S0065-2458(09)01205-4.

[70] J. Bezanson et al. "Julia: A Fresh Approach to Numerical Computing". In: *Society for Industrial and Applied Mathematics* SIAM Rev (Feb. 2017), pp. 65–98. DOI: 10.1137/141000671.

[71] J. Backus. "The History of FORTRAN I, II, and III". In: *SIGPLAN Not.* 13 (Aug. 1978), pp. 165–180. DOI: 10.1145/960118.808380.

[72] W. H. Press et al. *Numerical Recipes in Fortran 90*. 2nd ed. Cambridge University Press, 1996.

[73] A. C. Marshall. *Fortran 90 Course Notes*. The University Of Liverpool. 1997.

[74] M. Asadollahi. "Finite volume method for poroelasticity". MA thesis. Delft University of Technology, Mar. 2017.

[75] R. A. Brockman. "On the use of the Blatz-Ko constitutive model in nonlinear finite element analysis". In: *Computers & Structures* 24.4 (May 1986), pp. 607–611. DOI: 10.1016/0045-7949(86)90199-9.

# List of Figures