AALBORG UNIVERSITY
DENMARK

KU LEUVEN

Master thesis Project

**Design of Mechanical Systems**

# Thermo-mechanical simulation for fiber-optic measurement systems

Nicolas Henrik Javier Jensen

AALBORG UNIVERSITET
STUDENT REPORT

**Faculty of Engineering and Science**
Materials and Production
Fibigerstræde 14
9220 Aalborg Øst
http://www.aau.dk/

**Title:**

Thermo-mechanical simulation for fiber-optic measurement systems

**Project:**

4. Semester
Design of Mechanical Systems (DMS)

**Project Period:**

13. February - 1. June 2023

**Project Group:**

Group 13

**Student:**

Nicolas Henrik Javier Jensen

**Supervisor at AAU:**

Sergey Sorokin

**Supervisor at KU Leuven:**

Frank Naets

**Mentor at KU Leuven:**

Sander Neeckx

**Censor:**

Lasse Søgaard Ledet

**Report page number:** 57
**Appendix page number:** 14
**Total page number:** 83
**Completed 31. May 2023**

**Abstract:**

The ability of monitoring the health of a structure is crucial in engineering. Beyond the calculation and simulation done in the design phase, properly assessing the in-life health of a structure is essential. Nowadays, existing technologies like strain gages permit this type of measurement. However, this kind of technology comes with noticeable limitations. Strain gages are sensitive to electromagnetic fields, corrosive environments, and only yield measurements in the direct vicinity of the sensor. The novel fiber-optic sensing technology, which can provide quasi-distributed measurements over a structure, even in rough environments, is studied in this work. This novel sensing technology brings new technical challenges. In this project, the decoupling challenge of the thermo-mechanical behavior of the fiber-optic sensors, is resolved.
Based on the physical properties of this sensing technology, a virtual version of a fiber-optic sensor is developed, within a research-oriented Finite Element Method algorithm. Then, two new different numerical approaches to the decoupling challenge are established. Finally, for validation purposes, an experimental campaign is derived to confirm the robustness of the created solutions.

# Preface

This report is written by Nicolas Henrik Javier Jensen on the $4^{th}$ semester of the Design of Mechanical Systems master's degree education at Aalborg University. This report covers the realisation of a project initially proposed by the university of KU Leuven in Belgium. The project started in February 2023 and finished end of May 2023.

I would like to express my gratitude to my supervisor Frank Naets from KU Leuven, for offering me the opportunity of doing this project and for providing me valuables inputs during the entire project. I would also like to thank my mentor Sander Neeckx, for his help and constructive criticism during the project. I am grateful to my supervisor from Aalborg University, Sergey Sorokin, who supported me to make this abroad adventure possible. I would also like to thank my family, who encouraged me all along my Master with wise counsels and support. Last but no least, I would like to thank Maria Bernal Govaerts, who, perhaps against her will, has become an expert in fiber-optic sensors.

*Venice, 27 May 2023*

# List of Symbols

$[\partial]$ Derivative operator matrix.

$\alpha_f$ Thermal expansion coefficient.

$\Delta T$ Temperature change.

$\Delta\lambda_B$ Wavelength shift.

$\Delta\lambda_{B\Delta T}$ Wavelength shift induced by a temperature change.

$\Lambda$ Grating period.

$\lambda_B$ Initial wavelength shift.

$\nu$ Poisson's ratio.

$\phi$ Field variable.

$\rho$ Density.

$\xi$ Thermo-optic coefficient.

$\xi, \eta, \zeta$ Element local coordinates.

$\{d_c\}$ Prescribed displacement vector.

$\{F_c\}$ Prescribed force vector.

$\{F_f\}$ Free force vector.

$\{F_f\}$ Unknown displacement vector.

$\{F_n\}$ Nodal load.

$a$ Strain proportionality factor.

$E$ Young's modulus.

$F$ Force.

$N$ Newton.

$n$ Node numbering within an element.

$n_{eff}$ Effective refractive index.

$p_{eff}$ Effective photo-elastic coefficient.

$q$ Distributed load.

$T$ Temperature.

$W$ Weight factor.

$x, y, z$ Cartesian coordinates.

$y$ Shift ratio.

$y_{\Delta T}$ Shift ratio induced by a temperature change.

$[B]$ Strain-displacement matrix.

$[D]$ Element constitutive matrix.

$[K]$ Global stiffness matrix.

$[k]$ Element stiffness matrix.

$[K_{ff}]$ Stiffness of free nodes matrix.

$[N]$ Shape functions matrix.

$\{\epsilon\}$ Strain vector.

$\{\sigma\}$ Stress vector.

$\{d\}$ Nodal displacement vector.

# List of Acronyms

**CAD** computer-aided design

**DMS** Design of Mechanical Systems
**DOF** degree of freedom

**EFPI** Extrinsic Fabry-Perot Interferometric
**EMI** Electromagnetic Interference

**FBG** Fiber Bragg Grating
**FE** Finite Element
**FEM** Finite Element Method
**FM** Flanders Make

**MSD** Mecha(tro)nic System Dynamics

**POI** Point of interest

**SHM** Structural Health Monitoring

**UG** Universiteit Gent

**VUB** Vrije Universiteit Brussel

# List of Tables

# List of Figures

# Contents

# Chapter 1

# Introduction

*Since the beginning of engineering, the ability of creating reliable structures has always been strongly supported by different physical and mathematical interpretations, that allowed to safely create boats, planes, nuclear power plants, and much more... Being able to predict the resistance of a structure is key to ensure the quality of it.*
*However, physical and mathematical interpretations remains simplifications of the so-called, "real-life". Having the capacity of measuring and assessing the behavior of a structure, allows to reduce the gap between the numerical tools and the reality.*
*The availability of sensor measurements allows Structural Health Monitoring (SHM), in order to evaluate the state of the operating system.*
*Nowadays, many types of sensors exists, strain gages being the first choice in term of deformation measurement. The focus of this work is centred on a novel sensing technology, called Fiber Bragg Grating (FBG) sensor. This technology enables quasi-distributed sensing of deformation of a structure, even in rough conditions. However, this arising sensing technology comes with new challenges, e.g. decoupling of the thermal and mechanical sensing.*
*In order to robustly apply fiber optical sensors, a proper correction for the connection between the thermal and structural behavior of the fibers and system under study is key. Assessing the thermo-mechanical behavior of this coupled system will allow to properly design and calibrate the measurement approach.*

To this end, the problem formulation is:

> *How can the thermal and mechanical measurements of the Fiber Bragg Grating sensor be decoupled, to enhance and robustly apply this novel sensing technology in structural health monitoring?*

## 1.1 Context of the project

The project described in this report is embedded in a Belgian multi-university collaborative research project. The organizations that are collaborating for this project are the Mecha(tro)nic System Dynamics (MSD) research group from KU Leuven, the B-phot research group from Vrije Universiteit Brussel (VUB), the Universiteit Gent (UG) and Flanders Make (FM). The idea is that these organizations share their knowledge and strengths to improve the novel FBG sensing technology. The work subsequently discussed in this report is part of the MSD research group participation.

## 1.2 Scope and objectives of the project

In the following lines, the scope of the project is defined, to narrow the research directions span and ensure the feasibility of the project in the time frame. Subsequently, the main objectives are discussed.

### 1.2.1 Scope

To ensure that the thermo-mechanical behavior of the FBG sensors is properly assessed in the allocated time of this project, the focus is centred on several aspects, while other trajectories are discarded. The scope is then transformed into a list of objectives, which is enumerated in the following sub-section. As this project is part of an ongoing bigger project, as mentioned earlier, the scope of this investigation needs to comply with the already defined scope of the main project.

This project's main objective is to decouple the thermal from the structural deformation, measured by FBG sensors. The behavior of the sensor in the non-linear span of the material is not studied, as in SHM most of the structures are considered to have failed if the yielding limit is exceeded. The assessment of the dynamical behavior of the structure is not considered to limit the scope of this work. Lastly, the temperature range of interest, which is inherited from the ongoing project at MSD research group, ranges between $0°C$ and $85°C$.

### 1.2.2 Objectives

Leading from the scope of the project and the problem statement presented in the previous section, the main objectives of the project are derived in the list below.

- Development of a numerical toolbox to enable the use of virtual FBG sensors in combination with Finite Element Method (FEM) models.
- Development of a method to estimate separately the temperature and the strain, based on the output of one or several FBG sensors.
- Development of an experimental campaign to confirm and challenge the derived method.

This list of objectives serves as key guidelines for the investigation process and development of the solutions presented along this report. The above objectives are essential in the organization of the project and the project's report. Moreover, they will also be used to evaluate the achievements of the project once the results are obtained, as shown in Figure 1.1, in section 1.3.

## 1.3 Proceeding of the project

In this section, the chapters of the project are enumerated and briefly introduced. Each chapter is the reflection of a milestone in the project, from the introduction to the key principles to the ultimate temperature assessment techniques and the derivation of the experimental procedure. Here the reader can step back and have an overview of the different steps taken in this project.

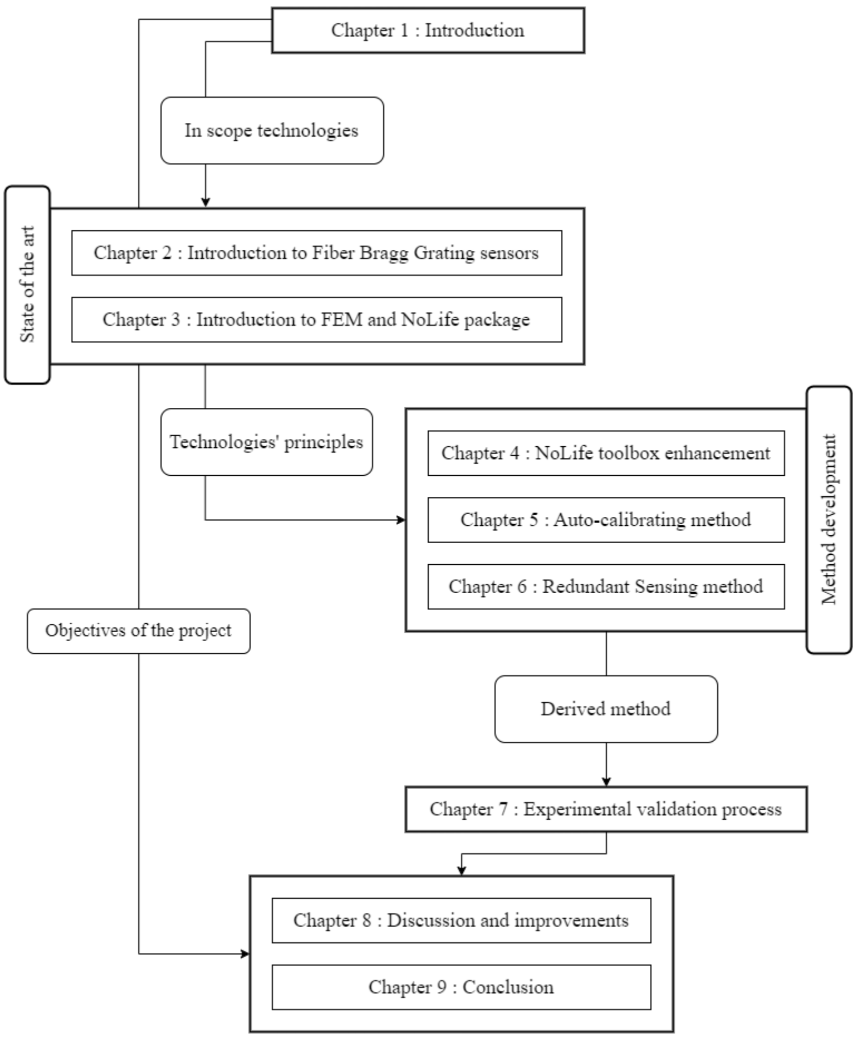



***Figure 1.1.*** Organization and dependency of chapters.

Chapter 1 is meant to introduce the project. The problem statement, the context of the project, and its scope are presented. The introduction presents the in-scope technologies that are then explained in detail in the *State of the art* chapters, i.e. Chapter 2 and Chapter 3. The introduction also provides the reader with the list of main objectives that will guide the project´s evolution. These objectives are assessed at the end of the project to ensure its completion, as illustrated above in Figure 1.1. In Chapter 3, the basic principles of FBG sensors, their thermo-mechanical coupled behavior, and the main advantages and drawbacks of this technology, are presented to the reader. Thirdly, the FEM is presented, and the NoLife library, an in-house built FEM tool that is later used for this project, is introduced, see Chapter 3.

The *State of the art* chapters then provide to the *Method development* chapters, see Figure 1.1, the technologies' principles that allows the creation of different tools and methods to solve the problem presented. In Chapter 4, several essential sub-functions are created. These sub-function are key for the proper functioning of the decoupling methods discussed later. Then, in Chapter 5, the first decoupling technique is revealed. This first method uses the linear behavior of the material and a calibration approach to estimate the unknown temperature. Subsequently, in Chapter 6, a second approach to the thermo-mechanical decoupling of the FBG sensor's terms is discussed. This second solution is based on the use of FEM and analytical methods.

The *Method development* provides the derived method for the experimental validation, i.e. the Chapter 7, as shown above in Figure 1.1. In this chapter the experimental campaign meant to validate the theoretical methods is presented. Finally, the results of the overall project are compared to the objectives in Chapter 8, where a discussion of the different results, and a suggestion of improvements paths are provided. Lastly, in Chapter 9, the project is summarized and the problem statement resolved.

# Chapter 2

# Introduction to Fiber Bragg Gratings sensors

*This chapter introduces the Fiber Bragg Grating (FBG) sensing technology, which this project aspires to improve. In engineering, the capability of assessing the health of a structure or a mechanism is key in terms of performance and safety. Many mature technologies already exist regarding Structural Health Monitoring (SHM), e.g. strain gages and thermocouples. However, the measurement range of former technologies is limited to the immediate vicinity of the sensor, requiring many sensors to assess the entire structure. Moreover, these sensors are sensitive to the environment, e.g. Electromagnetic Interference (EMI), corrosion and high-voltages can alter the efficiency of these technologies. Optical fibers arise in the SHM field with numerous advantages, one of them being the capacity to do quasi-distributed measurements, without adding intrusive mass.*

The basics of optical fiber technologies are introduced in the section 2.1, followed in section 2.2 with an explanation of the principle of Fiber Bragg Grating (FBG) sensor. Furthermore, the section 2.3 assesses the measurement of the FBG sensor in depth, while the section 2.4 introduces existing technologies to counteract the drawbacks of this technology.

## 2.1 Optical fiber technology

To properly understand the FBG sensing technology, the basic operation of optical fibers must first be explained. An optical fiber is a transparent fiber, made of a core in silica glass, that is used to transmit light. This core is enveloped by a transparent cladding material. This technology is broadly used in telecommunication due to its low loss of information over long distances and its ability to transmit information at a higher speed than electrical wires.

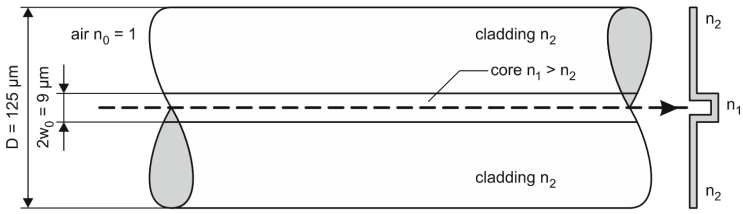



***Figure 2.1.*** Optical fiber.[1]

The index of refraction from the cladding surrounding material ($n_2$) is lower than the refractive index of the core ($n_1$). By using the total reflection principle, [1, p. 28], the light reflects on the cladding and is guided inside the core, as shown in Figure 2.2.

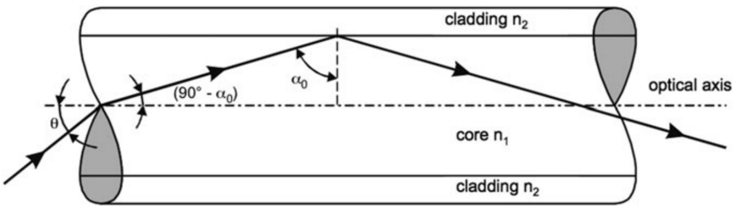



***Figure 2.2.*** Light guided in the core of an optical fiber, using the total reflection principle.[1]

## 2.2 Fiber Bragg Grating technology

A FBG sensor allows to survey the health of a structure, and assess physical phenomena e.g. strain and temperature. The FBG sensor is glued to the surface of the structure under study, as a strain gage, and follows its deformations.

However, to transform the aforementioned optical fibers in a FBG sensor, some slight modifications, that are depicted in the following section, are required.

An FBG sensor is an optical fiber which includes Bragg gratings, see Figure 2.3. The core of the optical fiber is locally modified with the FBG inscription method [2, p. 78], at the location where a sensor is desired

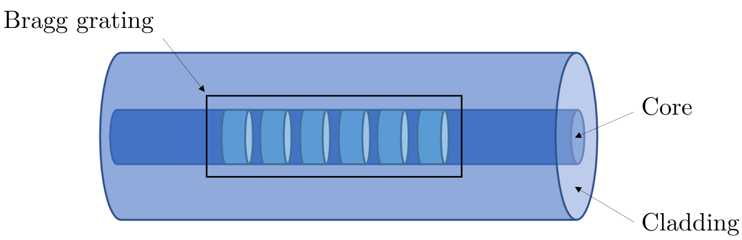
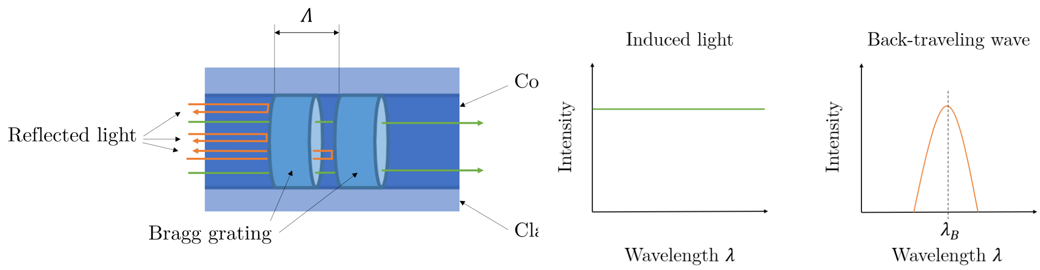



***Figure 2.3.*** Bragg grating. *Figure adapted from Figure 3.22 in* [2].

By locally changing the composition of the core, via Bragg grating, the refractive index changes. When broadband light is sent into the fiber, a narrow bandwidth is reflected at the Bragg grating and travels back to the light emitting source, see Figure 2.4a. This back-traveling wavelength is called a Bragg wavelength ($\lambda_B$), and its value depends on the location and the length of the Bragg grating on the fiber, as shown in Figure 2.4b.




*(a)* *(b)*

***Figure 2.4.*** Bragg grating back-traveling wave principle : (a) Refraction of the light on the Bragg Grating. *Figure adapted from Figure 3.22 in* [2], (b) induced light and back-traveling wave.

The Bragg wavelength ($\lambda_B$), illustrated in Figure 2.4b, is calculated with the following equation (2.1), [2, p. 77].

$$\lambda_B = 2n_{eff}\Lambda \tag{2.1}$$

where $\Lambda$ is the grating period, and $n_{eff}$ is the effective refractive index encountered by the light going through the Bragg grating, see Figure 2.4a. This Bragg wavelength ($\lambda_B$) is used as a reference, it stands for the initial configuration of the FBG sensor, where the strain $\epsilon = 0$, and the change of temperature $\Delta T = 0$.

## 2.3 Fiber Bragg Grating sensing principle

The FBG sensor assesses the physical behavior of the structure under study by being glued upon the surface of the structure, where it follows the deformations and temperature changes of the loaded structure. The sensor retrieves these changes as a shift in the wavelength $\lambda_B$, which is expressed as $\Delta\lambda_B$. Relation (2.2), [3, p 278], depicts the influence of temperature and strain on the shift in wavelength $\Delta\lambda_B$.

$$\Delta\lambda_B = \lambda_B[\Delta T(\alpha + \xi) + \epsilon(1 - p_{eff})] \tag{2.2}$$

where $\Delta T$ stands for the difference between the initial temperature and the final temperature. $\alpha$ correspond to the thermal expansion coefficient of the fiber, and $\xi$ to the thermo-optic coefficient. The thermo-optic coefficient $\xi$ associate the change of refractive index $n_{eff}$, with the temperature change $\Delta T$, [2, p. 83]. Regarding the strain $\epsilon$, $p_{eff}$ is the effective photo-elastic coefficient, which *"relates the change of refractive index"* $n_{eff}$ *"to the mechanical strain"*, [4, p. 1] and [2, p. 85].

As shown in the aforementioned equation (2.2), a change in temperature, strain, or both, induces a change in the value of $\Delta\lambda_B$. The difference between the initial wavelength $\lambda_B$, and the shifted wavelength $\Delta\lambda_B$, called the *"shift ratio"* $y$, depicts the total amount of strain $\epsilon$, and temperature change $\Delta T$, see equation (2.3), and Figure 2.5. However, the shift ratio $y$ does not provide sufficient information to dissociate the amount of strain $\epsilon$, and the amount of temperature change $\Delta T$, responsible of the shift in wavelength $\Delta\lambda_B$.

$$y = \frac{\Delta\lambda_B}{\lambda_B} = \Delta T(\alpha + \xi) + \epsilon(1 - p_{eff}) \tag{2.3}$$

The figure 2.5 illustrates the behavior of the FBG sensor, described by the relation (2.3). When the FBG sensor is subject to strain $\epsilon$ or temperature change $\Delta T$, the wavelength shifts in position. The proportion of both parameters is not observable on the output of the sensor.

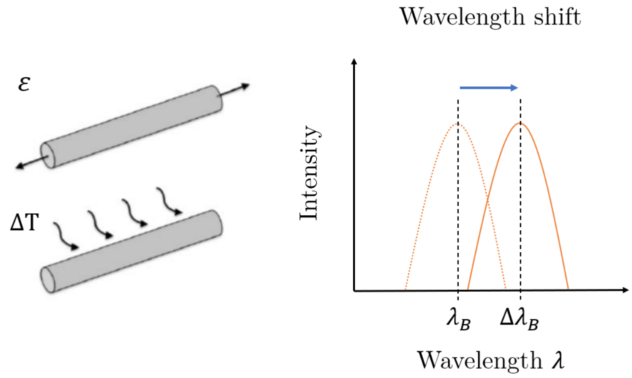



***Figure 2.5.*** Effect of strain and temperature on wavelength. *Figure adapted from Figure 3.18 in* [2].

## 2.4 State of the art in decoupling techniques

As highlighted previously, the strain measurements $\epsilon$ are heavily affected by the temperature $\Delta T$. Decoupling both parameters is key in the application of FBG sensors. Two techniques for this decoupling exist in the state of the art: (i) the use of two fibers to self-compensate for temperature, (ii) the usage of a combination of FBG with an Extrinsic Fabry-Perot Interferometric (EFPI) sensor.

### 2.4.1 Double fiber setup

This technique consists in using two fibers, as illustrated below in Figure 2.6. One fiber measures the strain and temperature, while the second one, closely located to the first, only measures the temperature. The second fiber is mounted in such a way that it is free to move ($\epsilon = 0$), to avoid strain affecting the wavelength shift $\Delta\lambda_B$ when the structure under study deforms, [5].

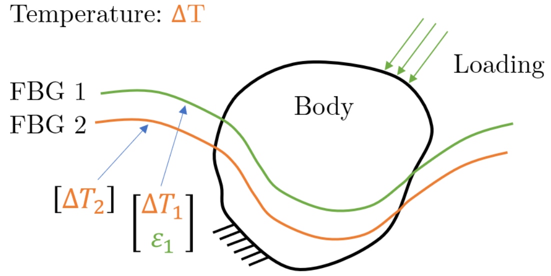



***Figure 2.6.*** Structure with double FBG sensors setup.

The measurement retrieved by the FBG sensor in the first fiber, the shift in wavelength $\Delta\lambda_{B1}$, is affected by the strain $\epsilon$ and the temperature change $\Delta T_1$. The second shift in wavelength $\Delta\lambda_{B2}$, given by the FBG sensor in the second fiber, only measures the temperature change $\Delta T_2$. By subtracting $\Delta\lambda_{B2}$, from $\Delta\lambda_{B1}$, the strain $\epsilon$ measured on the surface of the structure is obtained, and the influence of the temperature on the measurement is removed, see equation (2.4).

$$\Delta\lambda_{B1} - \Delta\lambda_{B2} = \lambda_B[\Delta T_1(\alpha + \xi) - \Delta T_2(\alpha + \xi) + \epsilon_1(1 - p_{eff})] \tag{2.4}$$

### 2.4.2 FBG and EFPI coupled setup

This approach, uses a combination of two sensors, leading to the FBG/Extrinsic Fabry-Perot Interferometric (EFPI) hybrid sensor, as illustrated in Figure 2.7. The interested reader can find more details in [3].

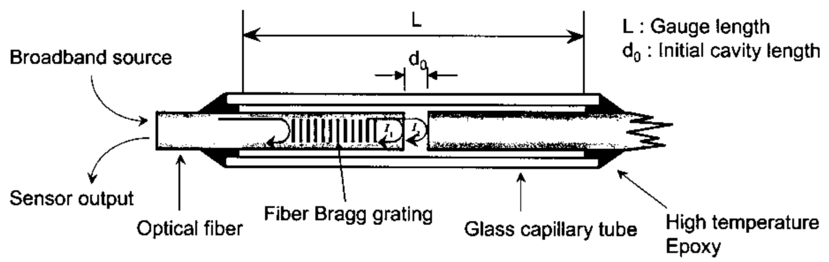



***Figure 2.7.*** Configuration of a FBG/EFPI hybrid sensor. [3].

The FBG sensor is *"encapsulated in a silica capillary tube"*, which isolates it from the strain-effects of the assessed structure. On the other side, the EFPI sensor is affected by both temperature and strain. When the structure deforms, due to a temperature variation and/or strain, the cavity length $d_0$ changes, see Figure 2.7. The temperature measured by the free-strain FBG can then be subtracted from the measurement of the EFPI sensor.

### 2.4.3 Weaknesses of the existing decoupling techniques

Despite of offering a solution, both methods have several disadvantages. The complexity of their installation, which directly influence the cost, and the qualification level of the operator installing these type of sensors. The manufacturing of the sensor is also a constraint, especially for the FBG and EFPI solution.

The main drawback of the first solution, the double fiber setup, is the complexity of the installation. The two fibers needs to be parallel, as close as possible to each other, while one of them is independent of strain $\epsilon$. Moreover, repeatability issues are expected, coming from the difficulty of aligning two fibers perfectly on a potentially non-flat surface. Finally, the level of qualification, and skills, of the operator installing these types of sensors is high, influencing again the repeatability of the solution, but also its cost.

For the FBG/EFPI hybrid sensor, the complexity for manufacturing the sensor itself is the main concern. This sensor is a combination of a FBG sensor, and an *EFPI cavity*, surrounded by a silica capillary tube, [3, p. 279]. Moreover, the fiber used for enclosing the cavity, seen on the right in Figure 2.7, is coated at its extremity with gold or aluminium, to enhance the reflectivity, i.e. the finesse of the results. Finally, the sealing of the overall assembly needs to be perfect [3], and genuinely, the cavity free of impurities.

## 2.5 Conclusion

The distributed nature of the measurement done by FBG sensors, the immunity to Electromagnetic Interference (EMI) and high-voltages, and mostly, the capacity to measure both strain and temperature, even in rough environments, emphasize the attractiveness of this technology for SHM.

Investigating a numerical approach to decouple the strain of the temperature measurements of a FBG sensor, such that measuring these quantities with only one fiber becomes possible, shows all its relevance here.

# Chapter 3

# Introduction to FEM and NoLife library

*This chapter presents the basics of the FEM and how this method is exploited for the sake of this project. The FEM tool used for this project is a MATLAB ® library called NoLife, which has been internally developed at the MSD research group. One of the objectives of this project is to extend the NoLife library with a tool that allows the usage of FBG sensors. This introduction is meant to provide the reader with an overview of the computation process, which yields the researched values that are key for the development of the toolbox.*

As a starting point, the FEM basic principles are presented in section 3.1. Subsequently, the main steps for computing the strain within a structural analysis are explained in section 3.2. Moreover, an academic case illustrates the usage of the NoLife library in section 3.3.

## 3.1 FEM introduction

The Finite Element Method (FEM) is a *"method for numerical solution of field problems"*, [6, p. 1]. It is nowadays extensively used for solving engineering challenges, from structural, electromagnetic, to thermal problems and much more. FEM permits to numerically approximate solutions to engineering problems, going beyond the limitations of analytical solutions. This work uses the FEM to solve structural problems, which will be the focus of the short introduction on the broad topic of FEM.

### 3.1.1 Modeling and discretization

To assess the behavior of a structure of interest with a FEM approach, the *"geometrical model"* coming from the computer-aided design (CAD) software needs to be transformed to a *"mathematical model"*. This mathematical model approximates the behavior of the geometrical model by using *"differential equations and boundary conditions"*, [6, p. 3]. To create the mathematical model, the geometrical model is discretized. The continuous body of the structure under study is split into small pieces, i.e. finite elements, see Figure 3.1. The continuous body is now a mesh of finite elements, where each element can be computed separately. Even if the mesh tends to follow the shape of the geometrical model, an error relative to the discretization process is induced, highly correlated to the size of the elements used.

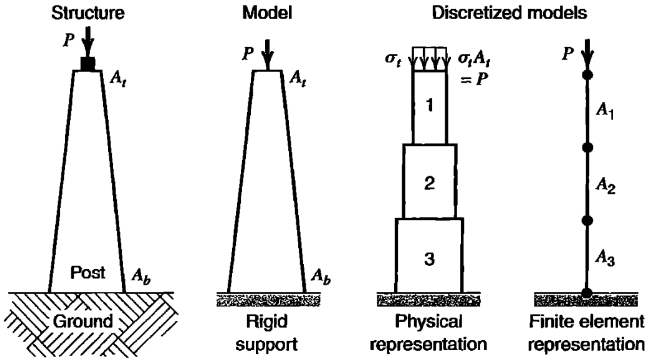



***Figure 3.1.*** Discretization example of a 1D model. [6].

### 3.1.2 Elements, nodes and DOFs

After discretizing the model, the mesh is composed of several elements, see *1*, *2* and *3* in the aforementioned 1D example, in Figure 3.1. If *1* is isolated, a spring element is obtained, as shown in Figure 3.2. Moreover, each element comes with a stiffness $k$, derived from the material properties of the model.

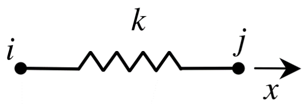



***Figure 3.2.*** Spring element. [7].

In a mesh, each element is surrounded by at least two nodes. The number of nodes depends on the dimensions of the problem, for example in Figure 3.2 above, the element is 1D and has two nodes, $i$ and $j$. Each node has no less than one degree of freedom (DOF). The number of DOFs are relative to the dimensions of the problem, and the type of element used, i.e. some elements allow nodes to rotate. In the Figure 3.2, the problem is 1D and rotation of the node is not permitted, so the node is only allowed to translate in one direction. The node has then one DOF. For 2D or 3D problems, and if rotations of the nodes are allowed, the number of DOFs increases, going up to six DOFs for 3D problems with rotating nodes.

## 3.2 Strain computation in structural mechanics

To compute the strain within a structure using the FEM, some key step needs to be followed. The element properties' matrices, the stiffness matrix $[k]$, the constitutive matrix $[D]$ and the strain-displacement matrix $[B]$ firstly needs to be generated. Subsequently, the elements need to be connected together, and the stiffness matrix $[K]$ of the overall structure needs to be constructed. Then, the loaded nodes are provided with their respective forces, and the DOFs of the nodes at the boundary conditions are set to zero. Next, the algebraic equations to obtain the nodal displacement $\{d\}$ are solved. Finally, the strain is computed, based on the nodal displacement $\{d\}$ and the strain-displacement matrix $[B]$.

### 3.2.1 Step 1: Element stiffness matrix $[k]$ generation

To properly assess the behavior of the structure the properties of each element needs to be defined. The stiffness matrix $[k]$ of the element is generated, as depicted in equation (3.1).

$$[k] = \int_V [B]^T [D][B]\, dV \tag{3.1}$$

The stiffness matrix $[k]$ is based on the constitutive matrix $[D]$, see (3.2), which contains the structure's material properties, i.e. the Young's modulus $E$ and the Poisson's ratio $\nu$.

$$[D] = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{(1-2\nu)}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{(1-2\nu)}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{(1-2\nu)}{2} \end{bmatrix} \tag{3.2}$$

The strain-displacement matrix $[B]$, shown in (3.3), is used to convert the displacements of the nodes of the element to the strain within this element. To construct the $[B]$ matrix, the shape function matrix $[N]$ needs to be constructed.

$$[B] = [\partial][N] \tag{3.3}$$

$[\partial]$ is the derivative operator matrix, see below (3.4).

$$[\partial] = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \end{bmatrix} \tag{3.4}$$

And the shape function matrix $[N]$ stores the shape functions of the element, as depicted in the following $[N]$ matrix (3.5).

$$[N] = \begin{bmatrix} N1 & 0 & N2 & 0 & ... & N_n & 0 \\ 0 & N1 & 0 & N2 & ... & 0 & N_n \end{bmatrix} \tag{3.5}$$

Where the value of $N_n$ is computed for the $n$ nodes of the element. $n$ varies between 2 for a 1D element, 4 to a 2D element, and 8 for a 3D element. More complex elements with a higher number of nodes exist, but they are out of the scope for this work. In the equation (3.6) below, each terms $\xi$, $\eta$ and $\zeta$ correspond to a direction of the local coordinate of the element. For instance, for a 2D element like the one in the Figure (3.3) underneath, $\zeta$ disappear.

$$N_n = \frac{1}{n}(1 \pm \xi)(1 \pm \eta)(1 \pm \zeta) \tag{3.6}$$

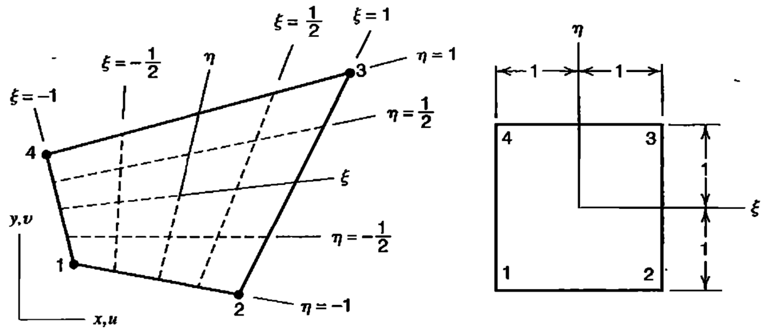



***Figure 3.3.*** Isoparametric element. [6].

To compute the element stiffness matrix $[k]$ by numerical integration, the Gauss quadrature rule is used. *A polynomial of degree* $2n-1$ *is integrated exactly by* $n$*-points Gauss quadrature* [6, pp. 211]. Using Gauss quadrature rule enables to compute the exact value of an integral of a polynomial, by making a summation of the polynomial assessed at specific points. These points, called Gauss or sampling points, see $\xi$ in Figure 3.4, are balanced by weight factors $W_n$, see the equation (3.7) below.

$$I = \int_{-1}^{1} \phi \, d\xi \approx W_1\phi_1 + W_2\phi_2 + ... + W_n\phi_n \tag{3.7}$$

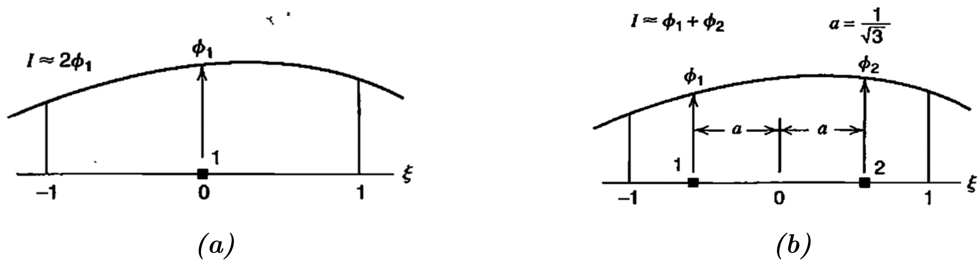



***Figure 3.4.*** Gauss quadrature rules of order 1 and 2.[6, pp. 210]

This principle naturally also applies for 2D and 3D elements. For a 2D element, the integration is written as seen below in (3.8), and the Gauss sampling points are located as depicted in Figure 3.5.

$$I = \int_{-1}^{1}\int_{-1}^{1} \phi(\xi,\eta)\, d\xi \; d\eta \approx \sum_{i}\sum_{j} W_i W_j \phi(\xi_i, \eta_j) \tag{3.8}$$

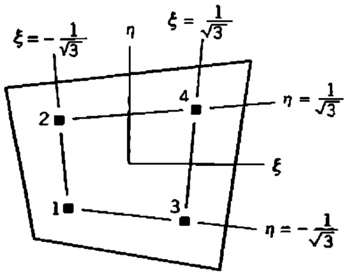



***Figure 3.5.*** Gauss point locations, using Gauss rules of order 2 to approximate equation (3.8).[6, pp. 210]

### 3.2.2 Step 2: Element connection and structure's stiffness matrix $[K]$ assembly

To create the stiffness matrix of the overall structure $[K]$, all elements' stiffness matrices $[k]$ are summed as shown beneath in (3.9).

$$[K] = \sum_{n=1}^{N_n} [k_n] \tag{3.9}$$

The elements are connected to each other by the nodes surrounding them, see Figure 3.1. When two, or more elements share a node, their stiffness is summed at this particular node. The relations $k_{n-1} + k_n$, seen in the diagonal of the matrix below in (3.10), reflect this connection for a 1D model where elements only have two nodes. The element $n-1$ and $n$ share one common node, and at this location, the stiffness of the two nodes are summed.

$$[K] = \begin{bmatrix} k_1 & -k_1 & 0 & \dots & \dots & 0 \\ -k_1 & k_1 + k_2 & -k_2 & \dots & \dots & 0 \\ 0 & -k_2 & k_2 & \dots & \dots & 0 \\ & & & \ddots & & \\ \vdots & \vdots & \vdots & k_{n-1} & -k_{n-1} & 0 \\ \vdots & \vdots & \vdots & -k_{n-1} & k_{n-1} + k_n & -k_n \\ 0 & 0 & 0 & 0 & -k_n & k_n \end{bmatrix} \tag{3.10}$$

### 3.2.3 Step 3: Application of the forces on the relevant nodes

Now that the properties of the model are set-up, the loads can be applied to the desired locations on the studied structure. The geometrical model has been split in a mesh of elements, as explained at the start of this section and shown in the example in Figure 3.6. It is necessary, in a first step, to find which nodes are part of the loaded surfaces or edges, what is the amount and direction of the loading applied to these locations, and how the load is spread over these locations.

In the FEM, a distributed load $q$ on an edge or a surface needs to be integrated over the loaded elements, see Figure 3.6. The nodal load $F_n$ of element $n$ is computed based on this integral, see equation (3.11) beneath.

$$F_n = \int_S [N]^T \{q\} \, dS \tag{3.11}$$

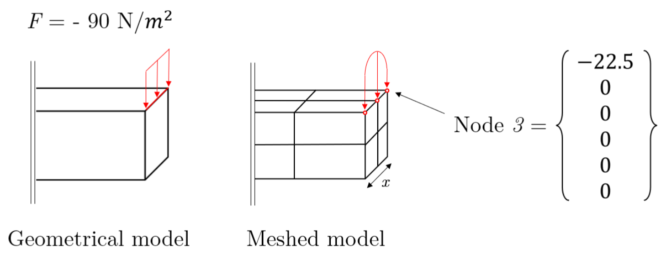



***Figure 3.6.*** Force application at node level.

The nodal force $F_n$ applied to these nodes is then stored in the column vector of prescribed forces $\{F_c\}$ of the model, where the loading information of each loaded node of the model is stored. Opposite to that, the unknown forces vector $\{F_f\}$ stores all the nodes with unknown loading, they are computed in a later step.

### 3.2.4 Step 4: Application of the boundary conditions

After applying the loading, the boundary conditions needs to be applied in the format a fixed nodes. The same procedure as for the loaded nodes applies here. The locations where the structure is fixed or constrained is depicted as a list of nodes, see Figure 3.7, and their DOFs are known, e.g. zero in case of fixed boundary conditions.

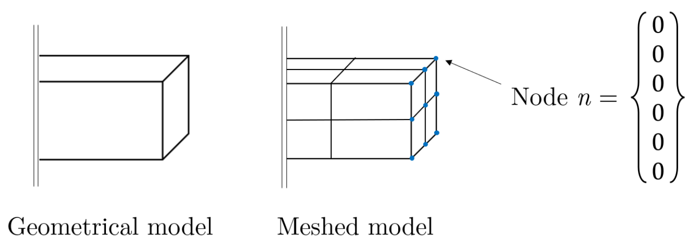



***Figure 3.7.*** Constrains application at node level.

The constrained DOFs are stored in the prescribed displacement vector $\{d_c\}$. Same as the loading vectors, the unknown displacement vector $\{d_f\}$ stores the nodes with unknown values, they are computed in the next step with the unknown forces.

### 3.2.5 Step 5: Computation of the nodal displacement $\{d\}$

The model's stiffness matrix $[K]$, prescribed force vector $\{F_c\}$ and prescribed displacement's vector $\{d_c\}$ are now defined. The unknown displacements vector $\{d_f\}$ can be computed by using the following equation (3.12).

$$[K]\{d\} = \{F\} \tag{3.12}$$

Which could be partitioned as follows..

$$\begin{bmatrix} K_{ff} & K_{fc} \\ K_{fc} & K_{cc} \end{bmatrix} \begin{bmatrix} d_f \\ d_c \end{bmatrix} = \begin{bmatrix} F_f \\ F_c \end{bmatrix} \tag{3.13}$$

The nodal displacement of every free node is computed likewise, since the constrained DOFs are fixed (equal to zero) in this example.

$$\{d_f\} = [K_{ff}]^{-1}\{F_f\} \tag{3.14}$$

The displacement of the free nodes $\{d_f\}$ and constrained nodes $\{d_c\}$ are merged in the nodal displacement matrix $\{d\}$.

### 3.2.6 Step 6: Computation of the strain $\{\epsilon\}$ within the structure

Finally the strain is computed for each element. The nodal displacement $\{d\}$ is know, the stiffness of each element $[k]$, and the strain-displacement matrix $[B]$, which is key here, allows to compute the strain for each element. The $[B]$ matrix varies for each element, so the relation (3.15) needs to be computed over the all model, to obtain the strain field in the overall structure.

$$[B]\{d\} = \{\epsilon\} \tag{3.15}$$

## 3.3 Academic case analysis using the NoLife library

To test the functions of the FBG sensor toolbox within NoLife, an academic example is derived. This trivial case allows to enhance and confirm the efficiency of this new solutions to SHM, before applying it to more complex industrial situations.

### 3.3.1 Academic case

The academic case is a cantilever beam of 300 $mm$ length, 25 $mm$ width and 3 $mm$ thickness. This basic sample easily exhibit important strain $\epsilon$ for low loading. This facilitates the recording made by the sensor attach to it, which is of great interest for future experimental work. Moreover, The beam is made of steel, to avoid damping issues coming from the material, for potential future vibro-acoustics and dynamical experiments. The beam is loaded at x = 300 $mm$, and fixed at x = 0 $mm$, see Figure 3.8.

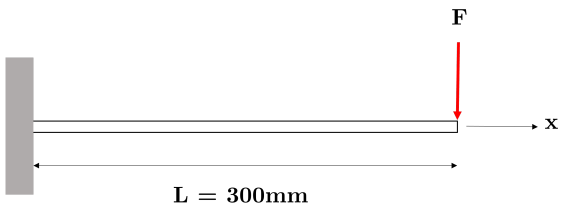



***Figure 3.8.*** Academic case.

#### Creation of the Finite Element (FE) model of the academic case

As a starting point, the CAD file of the structure of interest is converted to a FEM file, using the modeler of NX ® . The objective here is to convert the continuous structure in a meshed model, as explained in section 3.1. The continuous structure is split based on the desired size or number of elements, which correspond to 150 elements for the length, 12 elements for the width, and three elements for the thickness. The boundary conditions are also established in this step, see Figure 3.9. The file is ready to be exported, containing the list of nodes that are free, constrained, or loaded.

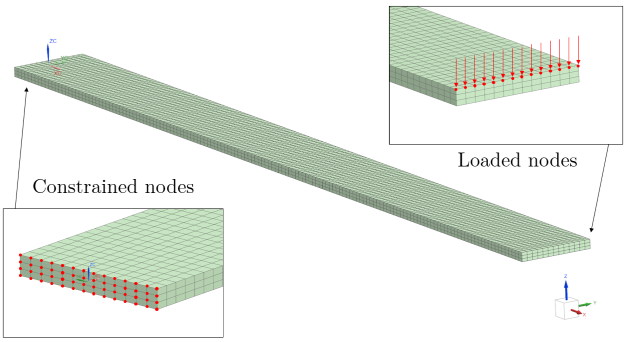



***Figure 3.9.*** Sample model meshed in NX ® , with constrained and loaded nodes defined.

#### Model data within NoLife

The model previously shown is exported from NX ® as a structure, which contains a list of nodes. Each node comes with its own coordinates and own ID. Moreover, a list of the loaded nodes, and of the constrained nodes are exported separately, so the boundary conditions can be reestablished within NoLife. The element IDs and particularly the ID of the nodes composing each element are also provided. This structure is illustrated below in

Figure 3.10. Lastly, the properties of the material of the model are merged to the exported structure.

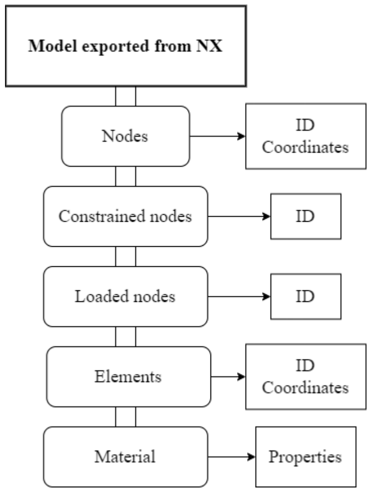



***Figure 3.10.*** Structure of the data exported from NX ® .

### DOFs ID

Based on the ID of the nodes and on the number of DOFs of the used element for the simulation, the DOFs ID's are computed. The list of node ID is multiplied by three or by six, for 3D elements, depending if only translation of the nodes are allowed, or if rotations are also permitted. For instance. see the relation (3.16) below, a node with six DOFs, having an ID equals to 342, leads to a DOF ID of 2047 for the $X$ DOF.

$$\begin{aligned} X_{ID} &= node_{ID} * 6 - 5 \\ Y_{ID} &= node_{ID} * 6 - 4 \\ Z_{ID} &= node_{ID} * 6 - 3 \\ RX_{ID} &= node_{ID} * 6 - 2 \\ RY_{ID} &= node_{ID} * 6 - 1 \\ RZ_{ID} &= node_{ID} * 6 \end{aligned} \tag{3.16}$$

## 3.4 Conclusion

FEM allows to compute the strain at the Point of interest (POI) upon the academic case derived earlier, but also over more complex industrial cases if needed. Furthermore, the use of the NoLife library permits the extraction of the essential matrices to the computation of the sought physical quantities, which enables the development of the main functions of the toolbox that is to be incremented to the NoLife library.

The virtual application of the FBG sensors relies on the use of the FEM, and this brief introduction serves as a background for the next three chapters.

# Chapter 4

# NoLife toolbox enhancement

*The existing FEM NoLife library is enhanced by adding several new tools. These tools allow the extraction of the essential information from the FE model. Properly locating the virtual FBG sensors is ensured by developing a node and element finder function, which links the relevant nodes and elements with the location of the selected POI. Once the location of the key elements is known, the strain at the location of interest is computed. Subsequently, the FBG sensor output is simulated, by using the strain previously find at the POI and the temperature.*

In the first section 4.1, the node and element finding procedure based on the user selection is presented. Then, in section 4.2, the process for computing the targeted values and matrices is explained. Lastly, the wavelength shift calculator function is described.

## 4.1 Node and element finder

The data retrieved by the FEM simulation done with the NoLife library needs post-processing for the user to obtain information at a pre-set location. For that, the user needs to go through the entire list of elements of the model, to find the element having matching coordinates with the location of interest. This seeking process is tedious and time consuming. To ensure that the researched physical quantities are easily assessed at the desired location on the FEM model, a node and element finder function is created. This function enables the user to find the ID of an element of interest, based on the location of its nodes. The elements linked to this node are then extracted and yielded to the user. The entire algorithm of this function can be found in appendix A.

The *node and element finder* is composed of two sub-functions, the first one, find the ID of the node of interest based on the inputs of the user. Then, the elements connected to that specific node could be found. The developed code is further explained below. It is important to note that the *node and element finder* is a sub-function that is meant to be called by another code, presented afterwards. This sub-function only receives the coordinates of interest and retrieves the elements and node IDs. The coordinates themselves are not defined at this layer of the code, but inputted as an argument.

### 4.1.1 Variable stream

#### Node finder

The sub-function *node and element finder* is split in two groups. To begin, the algorithm searches for the ID of the node located at the point of interest. It uses the `node_location` variable and the `model.NODES.CO` list, which contains the ID and coordinate of each node of the model. The *Node finder* compares the variable and the list, and yield the `node_ID`, i.e. the ID of the node located at the POI when the coordinate matches, see below in Figure 4.1.

#### Element finder

The same principle as above is applied here. The `model.ELEMENTS` list, which contains the ID, the coordinates, and the ID of the surrounding nodes of each element of the model, is injected. By comparing the previously obtained `node_ID`, with the `model.ELEMENTS` list, the *Element finder* provides the `element_ID`, i.e. the ID of the element of interest.

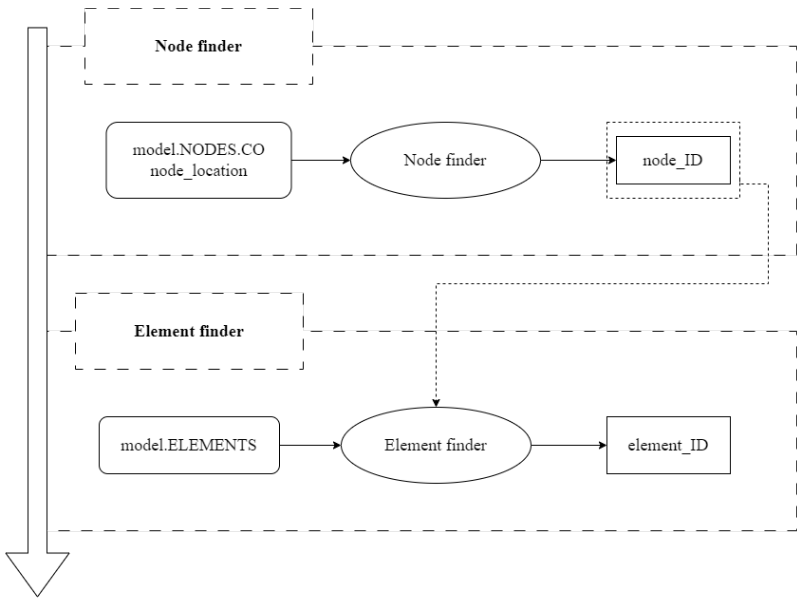



***Figure 4.1.*** Node and element finder function's variable stream.

### 4.1.2 Node finder

The search algorithm to find the `node_ID` starts by defining the coordinates of the location of interest. As mentioned earlier, these coordinates are inputted as an argument of the *node and element finder* sub-function. The coordinates of the location of interest are contained in the variable `node_location`.

```
% Loop to find the ID of the selected node, based on its coordinates
for i = 1:numel(model.NODES)

    if [node_location] == [model.NODES(i).CO]
        node_ID = i;
```

```
6          else
7
8      end
9  end
10
11 % Print status of the research process at the end of the loop
12 if exist('node_ID','var') == 1
13     node_ID;
14 else
15     Message = "Error: Node not found"
16     node_ID = [ ]
17 end
```

Now that the coordinates of the node of interest are known, the code loops over all the nodes of the model, as seen in line 2 with $i$ ranging from 1 to `numel(model.NODES)`. The variable `node_location`, which contains the coordinates of the selected node, is compared to each node in the `model.NODES(i).CO` list. This list contains the coordinates of all the nodes of the model, along with their ID, see Figure 3.10, in section 3.3. When the coordinates matches, the ID of the node is outputted, see line 5. If the node is found, its ID is provided to the user, see line 13. Otherwise, an error message arises, as shown in line 15.

### 4.1.3 Elements connected to node

From the known `node_ID` obtained earlier, the search for the element of interest can begin. As illustrated in Figure 3.10, the elements exported from NX ®, comes with their own ID but also with the ID of the nodes surrounding them. To find the element of interest, the list of surrounding nodes for each element is compared to the `node_ID` obtained in the previous sub-section. As shown below, the code loops over all the elements of the model, see `numel(model.ELEMENTS)` in line 3, to find the ones in the list that contains this particular `node_ID`, see the function `ismember(node_ID, model.ELEMENTS(i).NODES)` in line 5. Each time the code finds an element comporting this specific `node_ID`, it stores it.

```
1  % Loop to find the IDs of the 4 elements connected to the selected node
2  n = 0;
3  for i = 1:numel(model.ELEMENTS)
4
5      if ismember(node_ID, model.ELEMENTS(i).NODES);
6          n = 1+n;
7          element_ID(n,1) = model.ELEMENTS(i).ID;
8          else
9
10     end
11 end
12
13 % Print status of the research process at the end of the loop
14 if exist('element_ID','var') == 1
```

```
15     element_ID = element_ID(1); % First element (of 4) is chosen
16 else
17     Message = "Error: Element not found"
18     element_ID = [ ]
19 end
```

When an element is found, the variable $n$ is incremented by 1, see line 6. Then, the code loops over the remaining elements of the model. When the loop finishes, the first element of the list is allocated to `element_ID`, as shown in line 15. This choice is based on the assumption that the strain $\epsilon$ and the temperature change $\Delta T$ won't change drastically for neighboring elements. This assumptions could be enhanced by averaging these two physical values over the four elements, but this improvement couldn't be made in the time frame of the project. If the loop ends without yielding any element IDs, the code provides an error message, see line 17.

The *element finder* code is then able to provide the ID of the selected node on the cantilever beam, see Figure 4.3 in section 4.2. The ID of the neighboring elements to this particular node, and the ID of the surrounding nodes of these neighboring elements are also obtained. This allows to extract the physical quantities at the POI, i.e. where the FBG sensor are located.

## 4.2 Strain computation at element scale

This function is meant to compute strain $\epsilon$ and stress $\sigma$ within the elements of interest. These elements are located at the POIs, and their ID is extracted with the *node and element finder* sub-function, see section 4.1. The element-level matrix $[B]$, to compute stress $\sigma$ and strain $\epsilon$, is constructed based on the pre-computed node and element IDs, see section 3.2. The complete algorithm of the *Strain computation at element scale* function can be found in appendix B.

### 4.2.1 Variable stream

#### Node and element finder

The *Strain computation at element scale* function is divided in five groups. To begin, the *Node and element finder* receives the `ELEMENTS` list, see Figure 4.2 beneath, which contains information about the elements used within the model. The `NODES` list, which stores the ID and coordinates of each nodes, is injected in the function. The `AUX` list, containing the nodal information related to the FEM model, is also inputted. Additionally, the `cursor_info` variable, which hold the information regarding the selections made by the user, see next subsection, is added. Finally, the `igg2aa` and the `iLocalGG2AA` variables are provided. They are used to specify the number of DOFs analyzed in the FEM simulation.

Subsequently, the *Node and element finder* sub-function yields the following variables. The `node_ID` and the `element_ID`, which stands for the ID of the node and element of interest. The `element_NID` and `Node_Co`, which contains the ID and coordinates of the nodes surrounding the studied element. Lastly, the element properties stored in the `element` structure are provided.

### Computation of $[D]$

The computation of the constitutive relation matrix $[D]$ is realized by injecting the material properties stored in the `model` structure, see `model.MATERIALS` in the Figure 4.2 beneath. The matrix $[D]$ is outputted.

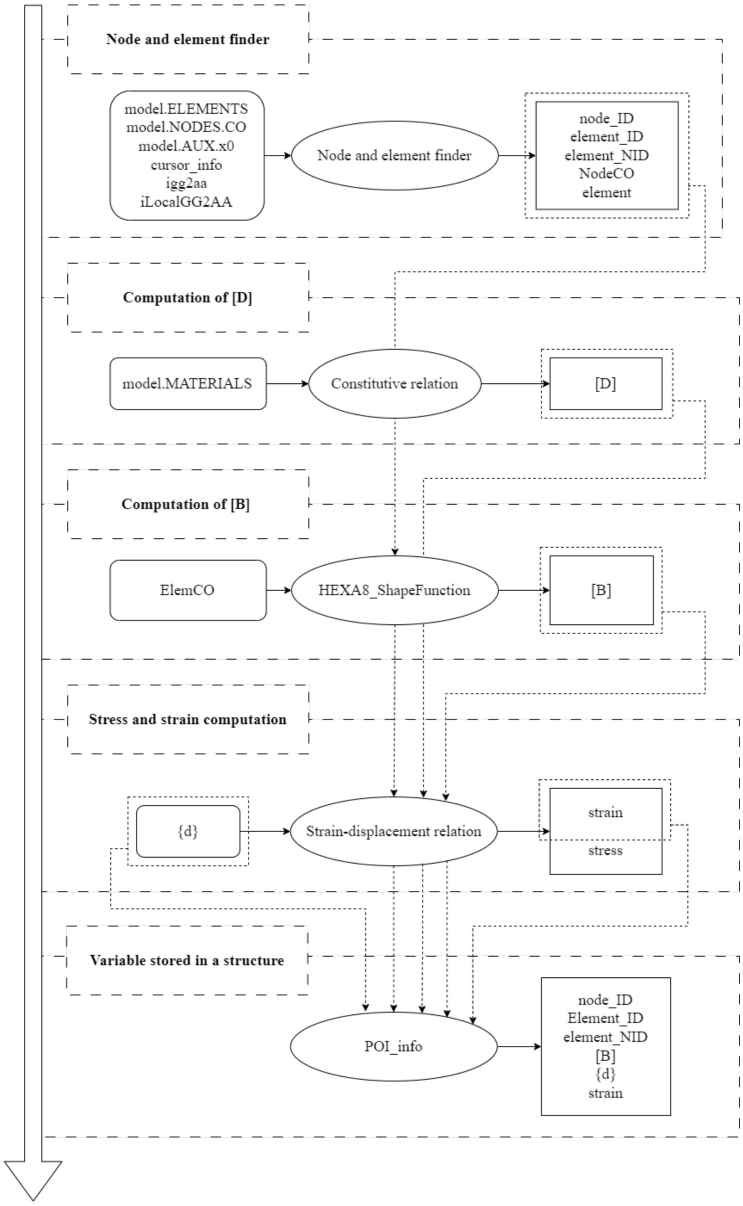



***Figure 4.2.*** Strain computation at element scale function's variable stream.

### Computation of $[B]$

The `NodeCo` variable computed above is injected in the built-in `HEXA8_ShapeFunction` of the NoLife library. The `ElemCO` variable, which contain the ID of the node of interest within the element-node scale, is also inputted. Then, the shape-function matrix $[N]$ and the derivatives $dN$ are computed, which enable the generation of the strain-displacement matrix $[B]$, at the location of the node of interest.

#### Stress and strain computation

To compute the stress and strain at the node of interest, the displacement $\{d\}$ is inputted. As shown in the equation (3.15), the computation uses the $[B]$ matrix, which is computed above.

#### Storage of the variables

The last step consist in storing the main variables, shown in the Figure 4.2, in a new structure called `POI_info`.

### 4.2.2 Loop over the elements of interest

The physical quantities are computed at the location of the selected POI. To find the ID of the node located at the POI, the coordinates need to be firstly defined. When the FEM analysis of the structure converges, a plot of the cantilever beam is displayed, as illustrated below in Figure 4.3. As shown in this figure, the user is able to right click on the node of interest on the cantilever beam, and choose to store the displayed coordinates under the variable `cursor_info.Position`. This variable can then be injected in the code as the variable `node_location`, as shown below.

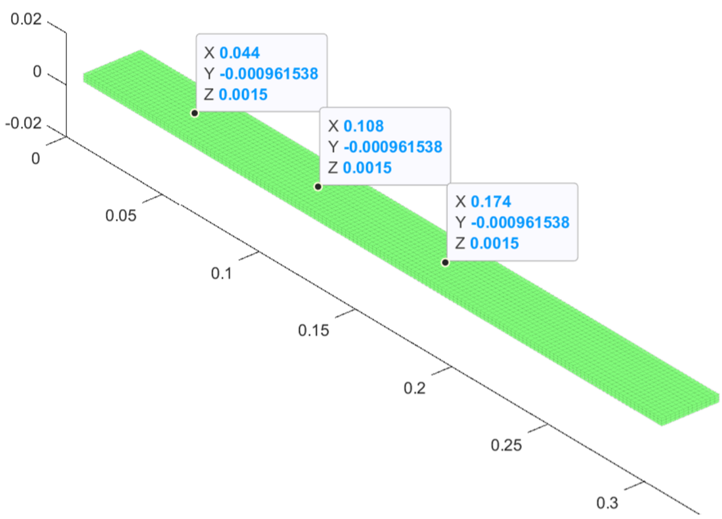



***Figure 4.3.*** Node selection on the plot of the cantilever beam.

The selection process, explained in section 4.1 and symbolized by the `[...]` in line 4 below, retrieves the coordinates of the selected node based on its coordinate. The code loops over the number of selected nodes $j$, which is counted with the built in `numel` function, see `numel(cursor_info)` in the first line below. The location of each POI, stored as the variable `cursor_info(j).Position` with $j = 1, 2, 3$ (three selections), is inputted in the loop, see line 2.

```
1 for j = 1:numel(cursor_info)
2     node_location = cursor_info(j).Position;
3
4   [...]
5
6 end
```

### 4.2.3 Search for element and node information

To begin, the element finder function provides the node ID and the element ID, see `[node_ID, element_ID]` in line 2. The element ID is used here to extract the IDs of the $n$ nodes of the considered element, in addition to the coordinates of this element, as shown in line 5 and 6. In line 4, the variable `elementNID` is an array containing the ID of the nodes surrounding the element under study. The variable `element` in line 6 gets assigned a structure containing the following essential information about the element under study. The type of element, its ID, and the surrounding nodes IDs are stored in this `element` structure, along with the DOFs and material IDs.

```
1 % Research of node ID and element ID
2 [node_ID, element_ID] = node_and_element_finder(model,node_location);
3
4 % Fetch current element from the elements struct, using element ID and node ID
5 elementNID = model.ELEMENTS(element_ID).NODES;
6 element = model.ELEMENTS(element_ID);
```

At element scale, the DOFs from the nodes are stored from 1 to $n * 6$, where 1 to 6 correspond to the DOFs of the $n$ node, as aforementioned in the relation (3.16) in section 3.3. As the element used in the FEM simulation has only displacement DOFs, the algorithm needs to get rid of the rotating DOFs. The `iLocalGG2AA` sub-function, see line 3, sort the DOFs. It removes the unwanted rotating DOFs. The variable `igg2aa` then contains only the displacement DOFs of the element.

```
1 % Indexing gg & aa set
2 nNoD = numel(element.NODES); % Auxiliary variable
3 igg2aa = iLocalGG2AA(nNoD);
4
5 % Extracting element coordinates from struct
6 NodeCO = model.AUX.x0(element.iDOF(igg2aa));
```

The `igg2aa` contains the row numbers of the DOFs of interest. The `element.iDOF` contains the ID of all the DOFs of the nodes surrounding the element of interest. By crossing these two lists, see `element.iDOF(igg2aa)` in line 6, the `igg2aa` filters the list `element.iDOF`. Only the ID of the DOFs $x, y, z$ of the nodes surrounding the targeted element are kept. The `model.AUX.x0` which contains the coordinates of all the nodes of the model, is then also filtered by this list, see `model.AUX.x0(element.iDOF(igg2aa))`. The `NodeCo` variable receives then the $x, y, z$ coordinates of each nodes of the considered element.

### 4.2.4 Computation of constitutive relation matrix

To pursue the computation, the properties of the material composing the model needs to be inputted. The specifications of the elected material are imported from NX ®. They correspond to a classic steel, with a Young's modulus $E = 210\ GPa$, a Poisson's ratio $\nu = 0.3$ and a density $\rho = 7800\ \frac{Kg}{m^3}$. These values are inputted in the code in line 2 to 4.

```
1  % Material parameters
2  E   = model.MATERIALS(1).PARAM(3);
3  nu  = model.MATERIALS(1).PARAM(4);
4  rho = model.MATERIALS(1).PARAM(2);
5
6  % compute D matrix (constitutive relation)
7  D = ...
8      (E/((1+nu)*(1-2*nu)))*...
9         [1-nu,  nu ,  nu ,      0      ,      0      ,      0      ; ...
10          nu  , 1-nu, nu ,      0      ,      0      ,      0      ; ...
11          nu  ,  nu , 1-nu,     0      ,      0      ,      0      ; ...
12          0   ,  0  ,  0  , 0.5*(1-2*nu),     0      ,      0      ; ...
13          0   ,  0  ,  0  ,     0      , 0.5*(1-2*nu),      0      ; ...
14          0   ,  0  ,  0  ,     0      ,      0      , 0.5*(1-2*nu)];
```

Then, the constitutive relation matrix $[D]$ is computed, see line 7. By using the relation (3.2), explained in section 3.2.

### 4.2.5 Computation of the strain-displacement matrix

To begin, the node of interest is found within the element, see line 2 below. The variable `ElemCoID` stands for the ID of this specific node, within the element-node scale. Based on this ID, the coordinates of this specific node are extracted, see line 3 to 18 below. These coordinate stands within the coordinate system of the element, i.e. $\xi, \eta$, and $\zeta$, see Figure 4.4.

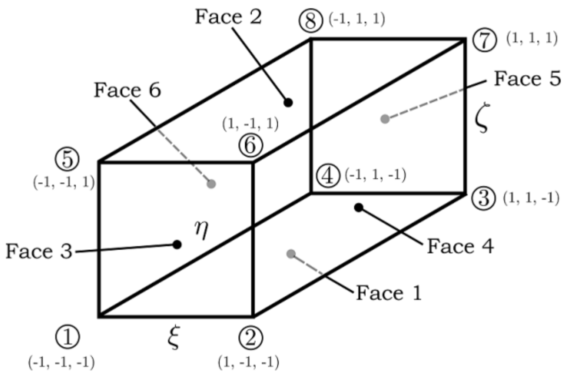



***Figure 4.4.*** Node ID at element scale.

```
% Extraction of the node coordinates within the element coordinate system
ElemCoID = find(element.NODES==node_ID)

if ElemCoID == 1
    ElemCO = [-1,-1,-1]
elseif ElemCoID == 2
    ElemCO = [1,-1,-1]
elseif ElemCoID == 3
    ElemCO = [1,1,-1]
elseif ElemCoID == 4
    ElemCO = [-1,1,-1]
elseif ElemCoID == 5
    ElemCO = [-1,-1,1]
elseif ElemCoID == 6
    ElemCO = [1,-1,1]
elseif ElemCoID == 7
    ElemCO = [1,1,1]
elseif ElemCoID == 8
    ElemCO = [-1,1,1]
else
    Message = "Error : node not found"
end
```

As seen in line 2 beneath, the built-in function `HEXA8_ShapeFunction` is used here. This function is based on the theory presented in section 3.2. The shape function matrix $[N]$ and the derivative $dN$ are computed for the node of interest.

```
1 % Shape function related computations
2 [N,dN] = HEXA8_ShapeFunction(ElemCO,NodeCO,1,1,1);
3
4 % Strain displacement matrix B
5 B = B_3D_linear(dN,nNoD);
6 B = B([1:6],[3*ElemCoID-2:3*ElemCoID])
```

Subsequently, the strain-displacement matrix $[B]$ is calculated, by computing the derivative $dN$ for the number of node `nNoD` of the element, see line 5. Then, the value of the $[B]$ matrix for the node of interest is extracted. For that, the ID of the node `ElemCoID` is used, as shown in line 6.

### 4.2.6 Computation of strain and stresses

Now that the strain-displacement matrix $[B]$ is known, the strain $\epsilon$ at the POI is assessed. The displacement $\{d\}$ of the node of interest, see line 2, is extracted. As seen below, the displacement $\{d\}$ is only considered in the $x, y, z$ coordinates, following the relation (3.16), in section 3.3. The variable `d_node` stores the displacement in an array, which is transposed afterwards, in line 4.

```
1 % Computation of the displacement and strain
2 d_node = [d(elementNID(ElemCoID)*3−2), d(elementNID(ElemCoID)*3−1),...
3     d(elementNID(ElemCoID)*3)];
4 d_node = d_node';
5
6 strain = B*d_node; % Strain computation for the entire element
7 stress = D*strain; % stress computed for the entire element
```

Finally, the strain is computed in line 6. The relation (3.15), discussed in section 3.2, is used. The strain-displacement matrix $[B]$ is multiplied with the nodal displacement $\{d\}$ of the node of interest. Then, the stress is also computed, see line 7, by using the Hooke's law, reminded below in (4.1).

$$\{\sigma\} = [D]\{\epsilon\} \tag{4.1}$$

### 4.2.7 Storing the essential variables in a structure

After computing the searched values at the POIs, the results are stored in a new structure called `POI_info`. The matrices $[B]$, but also the strain $\epsilon$ and the displacement of the nodes of interest are saved. The `POI_info` structure allows to extract these key information for future computation, see line 3 to 9. The sub-structure `POI1`, stores the values computed at the location of the first element of interest, which corresponds to the first sensor.

```
1 if j == 1
2
3         POI_info.POI1.B1 = B;
4         POI_info.POI1.strain1 = strain;
5         POI_info.POI1.d_node1 = d_node;
6         POI_info.POI1.node_ID1 = node_ID;
7         POI_info.POI1.element_ID1 = element_ID;
8         POI_info.POI1.element_NID1 = elementNID;
9         POI_info.POI1.strain_F1_1 = strain_F1;
10
11     elseif j == 2
12
13          [...]
14
15     elseif j == 3
16
17          [...]
18
19     else
20         Message = 'Error'
21 end
```

For each iteration, i.e. each element of interest, a sub-structure `POI` is created. The `[...]` term line 13 and 17 symbolize the repetition of the above lines 3 to 9, but for the second and third POI. At this stage of the project, the number of selection on the model are limited to three POIs, which condition the number of sub-structure that can be created. If the number of POI, illustrated by the variable $j$ in line one, is below zero or exceed three, an error is displayed, see line 20.

### 4.2.8 Plotting of the selected nodes on the beam

The deflected cantilever beam, resulting from the FEM simulation, is then provided, as illustrated below in the Figure 4.5. The nodes of interest, selected with the method presented in section 4.1, are highlighted in red. The POIs are displayed on the plotting of the deflected cantilever beam.

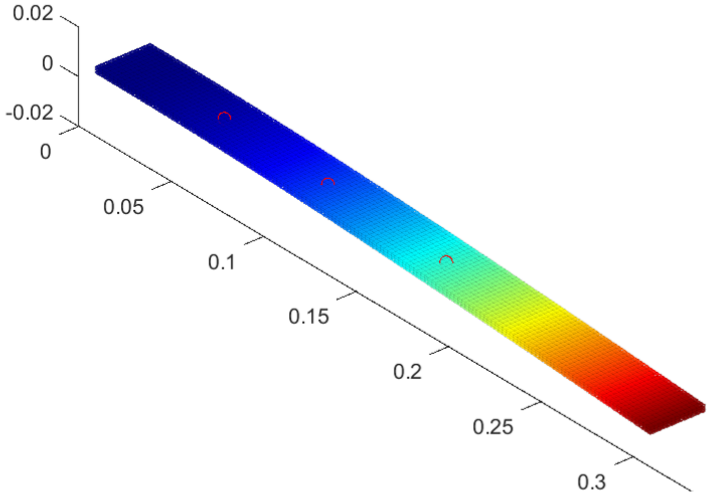



***Figure 4.5.*** Deflected beam with element's nodes highlighted.

## 4.3 Wavelength shift calculator

To simulate the output of a FBG sensor, a function based on the relation (2.2) in section 2.3 is created. The wavelength shift calculator uses as inputs, in line 1, the strain $\epsilon$ computed earlier at the location of the POI, i.e. sensor, and the temperature change $\Delta T$ of the overall structure. The constants `Cstrain` and `Ctemp` stands for the subsequent variables, $C_{strain} = 1 - p_{eff}$ and $C_{temp} = \alpha + \xi$, see details in section 2.3.

### 4.3.1 Variable stream

The variable stream within the wavelength shift trivial, it reflects the (2.2) and (2.3) from section 2.3. The wavelength shift $\Delta\lambda_B$ is computed based on aforementioned inputs. Then, based on the wavelength shift $\Delta\lambda_B$ and the initial wavelength $\lambda_B$, the shift ratio $y$ is computed.

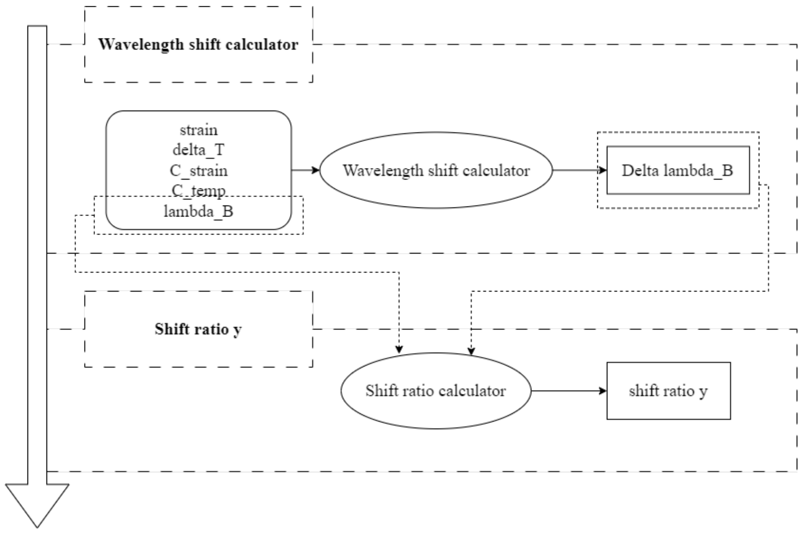



***Figure 4.6.*** Wavelength shift calculator function's variable stream.

### 4.3.2 Main code

$\Delta\lambda_B$ is computed in line 4, using the previously mentioned relation (2.2). The initial wavelength $\lambda_B$, explained in section 2.3, is multiplied to the measured strain $\epsilon$ and temperature $\Delta T$. The constants of the sensor are also added.

```
1 function [y] = calc_wavelength_shift(strain, deltaT, Cstrain, Ctemp, lambdaB)
2
3 % Wavelength shift (induced by a variation of strain and/or temperature) calculation
4 deltaLambdaB = lambdaB*((Ctemp*deltaT)+(Cstrain*strain));
5
6 % Shift ratio between wavelength shift and initial wavelength
7 y = deltaLambdaB/lambdaB;
```

Subsequently to that, the shift ratio $y$ is computed, by using the equation (2.3), also in section 2.3. The shift ratio $y$ is finally outputted.

## 4.4 Conclusion

The three tools presented in this chapter enable the proper assessment of the model at the location of the POIs, i.e. the location of the FBG sensors on the model. The user is now able to selected a node and obtain the information relative to the identification of this node and of the neighboring elements. Then, the strain and stress at this precise location can be extracted. Finally, the user can simulate the output of a FBG sensor located at the POI previously selected.

These key functions will be critical in the development of the two strain-temperature decoupling tools presented in the subsequent chapters.

# Chapter 5

# Auto-calibrating method

*In this chapter, the so-called auto-calibrating method is presented. This technique is the first approach to assess the thermo-mechanical behavior of the FBG sensors. It is based on a calibration technique, using the linearity aspect of the constitutive equations to evaluate the temperature around the sensor, by comparing the outputs of the FBG sensor for an increasing load.*

The principle behind this method is introduced in the section 5.1, to provide to the reader an overview of the strategy. Subsequently, the variable stream and the structure of the solution is discussed, in section 5.2. Details of the code are present in section 5.3.

## 5.1 Auto-calibrating principle based on linear constitutive equations

The auto-calibrating technique follows a simple structure explained in the following lines. Each module, depicted below in Figure 5.1, contains key functions that allow to estimate the temperature $\Delta T$, and then deduce the strain $\epsilon$. This estimation is based on the shift ratio $y$, and the increased load.

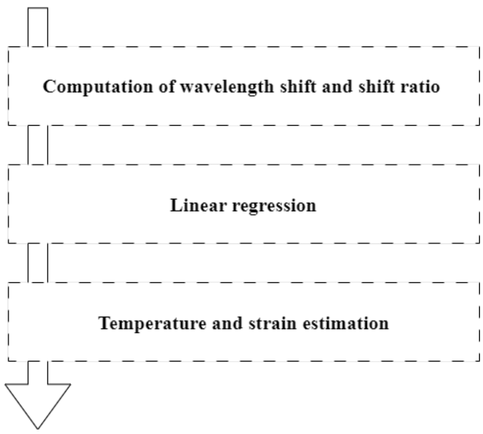



***Figure 5.1.*** Structure of the strain estimation code.

The first step gathers the needed inputs, and compute the wavelength shift $\Delta\lambda_B$ and the shift ratio $y$. Based on the shift ratio $y$, and the strain $\epsilon$ directly correlated to the load magnitude, the linear regression is now able to proceed. Lastly, the temperature $\Delta T$ and strain $\epsilon$ are estimated, based on the result of the aforementioned linear regression.

The auto-calibration principle is a technique that compares the output of a FBG sensor, located at a particular POI, for different magnitudes of loading. By using the constitutive equations in the linear domain, the temperature $\Delta T$ can be deduced at the location of the sensor. The shifted wavelength $\Delta\lambda_{B\Delta T}$, yielding from the difference in temperature $\Delta T$, produces an offset that can be recorded in certain conditions.

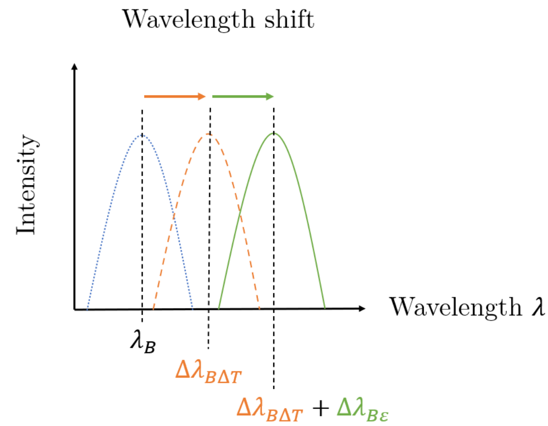



***Figure 5.2.*** Offset in wavelength shift, due to temperature and/or strain.

This offset, the wavelength shift due to temperature illustrated below in Figure 5.2, can then be compensated. This allows to differentiate the wavelength shift due to the strain $\Delta\lambda_{B\epsilon}$. As explained in section 2.3, FBG sensors retrieve the strain and temperature change by a wavelength shift, following the equation (2.2), see Figure 2.5. A structure subject to a uniform and steady temperature change $\Delta T$, over its whole body, will induce an equal and steady offset in wavelength shift to every sensors attached to it. Moreover, if the material of the loaded structure remains in its elastic regime, the measured strain $\epsilon$ on the structure changes linearly. The wavelength shifts, see equation (2.2), are then behaving linearly.

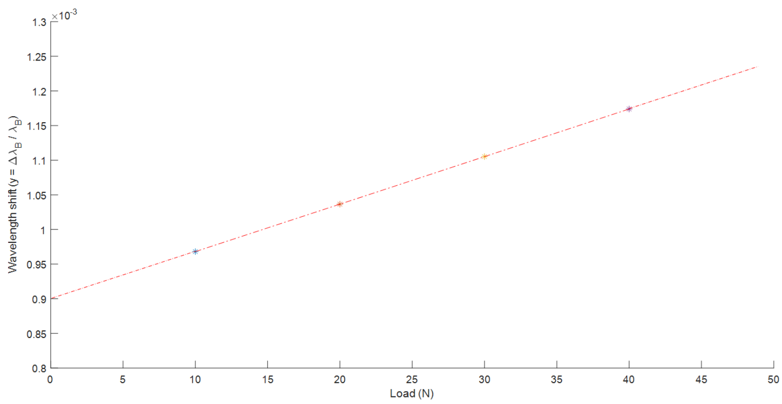



***Figure 5.3.*** Wavelength shift against loading, with linear regression.

Naturally, by increasing the load, the strain in the body increases. By evaluating the wavelength shift ratio, see equation (2.3), against the loading, and by using a linear regression analysis, the thermally-induced wavelength shift $\Delta\lambda_{B\Delta T}$ can be approximated. The shift ratio yielding from this thermal wavelength shift, expressed below in the equation (5.1), appears when the strain $\epsilon = 0$ at $F = 0$ N, as illustrated above in Figure 5.3.

$$\frac{\Delta\lambda_{B\Delta T}}{\lambda_B} = \Delta T(\alpha + \xi) + \epsilon_0(1 - p_{eff}) \tag{5.1}$$

This thermally-induced offset in every wavelength shift $\Delta\lambda_{B\Delta T}$ is used afterwards to deduce the strain at the desired POI. The temperature change $\Delta T$ is computed by rewriting the relation (5.1) above as shown below, where the strain $\epsilon_0$ is null.

$$\Delta T = \frac{y_{\Delta T}}{(\alpha + \xi)} \tag{5.2}$$

Finally, the strain $\epsilon$ can be computed, as demonstrated below in equation (5.3).

$$\epsilon = \frac{\Delta\lambda_B - \Delta\lambda_{B\Delta T}}{\lambda_B(1 - p_{eff})} \tag{5.3}$$

## 5.2 Variable stream in auto-calibrating method algorithm

The definition of the variable stream, presented in the subsequent lines, is meant to highlight the connections between the key functions and the path followed by the different variables. The overall structure of the code described in this section can be distinguished in two group, the *Simulation of the sensor output* group and the *Temperature estimation by linear regression* group.

### 5.2.1 Simulation of the sensor output

The simulation of the sensor output begins with the input of the FBG sensor's properties `FBG_properties`, presented in 2.3. The initial wavelength $\lambda_B$, described by equation (2.1), and the temperature change $\Delta T$ are also injected in the code here, see Figure 5.4. Moreover, the strain `strain_F1` corresponding to the strain $\epsilon$ at the POI for the applied force `F1` on the structure, and the strain `strain_F2`, to the loading `F2`, are inputted.

By using the relation (2.2) discussed in section 2.3, the shift ratio `y_POI_F1` and `y_POI_F2` are computed, based respectively on the strain `strain_F1`, the `strain_F2` and the temperature change $\Delta T$.

### 5.2.2 Temperature estimation by linear regression

#### Linear regression

To proceed, the linear regression is provided with the shift ratio `y_POI_F1` and `y_POI_F2`, but also with the corresponding loading over the assessed structure, i.e. `F1` and `F2`. The shift ratio per loading curve is then derived, and the equation `P(1)*x+P(2)` yielding form the linear regression displayed. The value of `P(2)` here is key, it correspond to the offset induced by the temperature change $\Delta T$ on the shift ratio $y$ of each FBG sensor over the structure, see previous section 5.1.

#### Temperature and strain estimation

As seen beneath in Figure 5.4, the temperature change $\Delta T$ is estimated based on the value of `P(2)` and the properties of the FBG sensor, by using the relation (5.2). The strain estimated `strain_estimated_F1` and `strain_estimated_F2` at loading `F1` and `F2` are then computed, based on the shift ratio `y_POI_F1` and `y_POI_F2`.

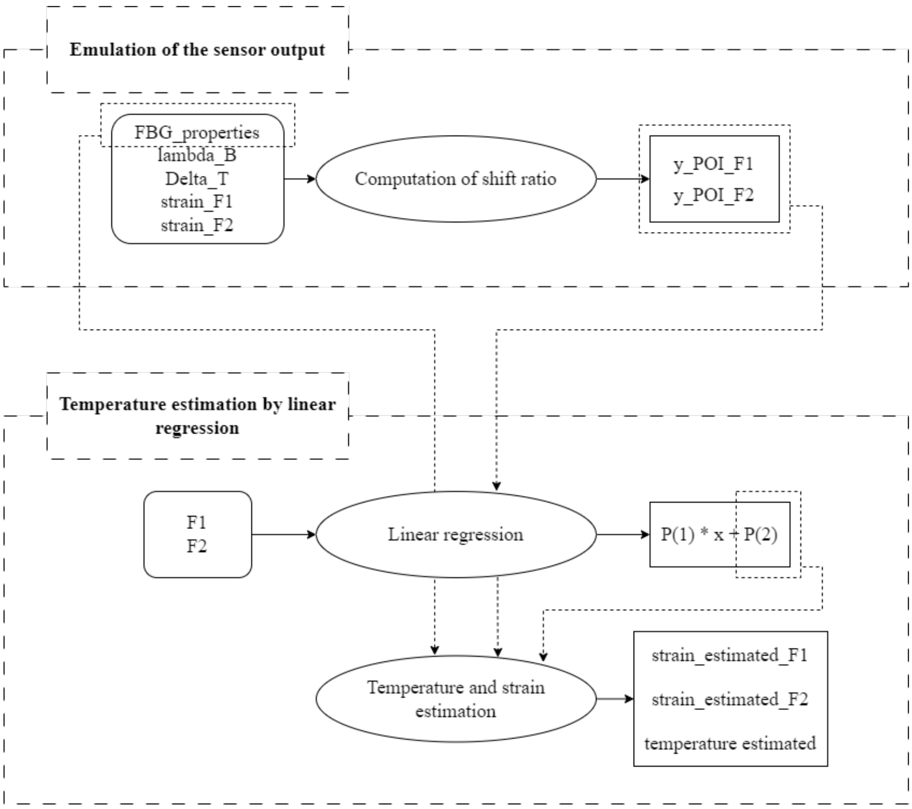



***Figure 5.4.*** Variable stream in the auto-calibrating method.

## 5.3 Algorithm of the auto-calibrating solution

In the following lines, the algorithm of the auto-calibrating method is presented. Each steps and functions are explained, in echo to the previous section and the Figure 5.4. The complete algorithm of this method can be found in appendix D.

### 5.3.1 Inputs

The starting point of the code is the loading of the variables. The properties of the FBG sensor, $\alpha_f$ , $\xi$ , $p_{eff}$ and $\lambda_B$ are loaded to the code. Moreover, the physical quantities that the sensor is intended to measure, i.e. the strain $\epsilon$ and temperature $\Delta T$, are simulated.

#### FBG sensor's constants

The variables seen below in table 5.1, introduced in the Chapter 2, in section 2.3, stands for the physical properties of the FBG sensor. The values of the thermal expansion coefficient of the specimen's material $\alpha_f$, the thermo-optical coefficient $\xi$, and the effective photo-elastic coefficient $p_{eff}$ are chosen based on [2, p. 85].

***Table 5.1.*** FBG sensor properties

| Variable | Value | Units |
|---|---|---|
| $\alpha_f$ | 11e-06 | [Kelvin^-1] |
| $\xi$ | 6.5e-06 | [Kelvin^-1] |
| $p_{eff}$ | 0.22 | / |

#### Initial wavelength $\lambda_B$

This method focus on the wavelength shift $\Delta\lambda_B$, see equation (2.2), which depicts the change of the measured physical properties of the analyzed structure. The initial wavelength $\Lambda_B$, is chosen arbitrary based on the common values observed in the literature. They range between 1530 $\mu m$ to 1580 $\mu m$, see [2] [5] [3]. Considering the low benefit of knowing precisely the value of the initial wavelength for the development of this solution, and regarding the easiness of inputting a "realistic" and measured initial wavelength in the future, the initial wavelength value is chosen arbitrarily to be 1550 $\mu m$.

#### Temperature difference $\Delta T$

Subsequently, the temperature $\Delta T$ is added, see Figure 5.4. The goal of this project is to assess the behavior of the FBG sensor in a temperature span of $0°C$ to $85°C$. For instance, the value of $80°C$ is selected for the explanation of the code contained in the following lines.

#### Strain $\epsilon$

Furthermore, the strain $\epsilon$ values imported to the code yield from a simulation of the academic example shown in section 3.3. At this stage of the project, the FEM simulation providing the strain $\epsilon$ values shown in the table 5.2 beneath is made in a commercial FEM software. The temperature is also taken into account in this FEM simulation, but in the targeted range of temperature, a material like steel does not exhibit noticeable thermal expansion effects. The strain $\epsilon$ values are stored in the matrix `strainFEM`.

***Table 5.2.*** Strain computed with a commercial FEM software at the POIs

| | Strain at point of interest (POI) | | |
|---|---|---|---|
| Load $F$ | POI 1 | POI 2 | POI 3 |
| 10 | 1.3797E-04 | 1.1244E-04 | 8.7428E-05 |
| 20 | 2.7492E-04 | 2.2487E-04 | 1.7492E-04 |
| 30 | 4.1223E-04 | 3.3735E-04 | 2.6242E-04 |
| 40 | 5.4964E-04 | 4.4985E-04 | 3.4992E-04 |

### 5.3.2 Computation of the wavelength shift and shift ratio

To compute the wavelength shift, the `fbg_wavelength_shift_calculator` function, see 4.3, is used. It is contained in a loop that iterates its calculation following the number of load steps found in the data file. At each iteration, the value of $\Delta\lambda_B$ is stored in a matrix. At the same time, the shift ratio $y$ is computed and stored as well in a matrix following the same procedure. Finally, the $\Delta\lambda_B$ matrix and the $y$ matrix are printed.

The `fbg_wavelength_shift_calculator` function seen in line 2 below correspond to the equation (2.2) expressed earlier in this report and reminded beneath. The variables previously loaded goes through this function at each iteration.

$$\Delta\lambda_B = \lambda_B[\Delta T(\alpha + \xi) + \epsilon(1 - p_{eff})]$$

```
1 % Wavelength shift calculation
2 deltaLambdaB = lambdaB*((deltaT*(alphaf+xi))+(strain*(alphaf+xi)));
3
4 % Computation of deltaLambdaB and shift ratio
5 for i=1:rowstrainFEM
6     dlb = fbg_wavelength_shift_calculator(strainFEM(i,1),deltaT,alphaf,xi,lambdaB,peff);
7     deltaLambdaB(i) = dlb;
8     y(i) = deltaLambdaB(i)/lambdaB;
9 end
```

In line 6, the variables goes through the function above. Then in line 7, the matrix stores the computed values for $\Delta\lambda_B$ and in line 8 the $y$ matrix stores the shift ratio for each iteration. The iteration number, set by the `rowStrainFEM` value, correspond to the number of load steps. The subsequent matrices can then be used for the linear regression.

### 5.3.3 Linear regression

To deduce the temperature at the location of the FBG sensor, the linear aspect of the constitutive equation is used. The linear regression allow to estimate the shift ratio for the strain $\epsilon = 0$ at $F = 0$ N, by extrapolating the linear behavior of the material at the POIs.

```
% Linear regression to obtain the slope of the y to strain curve
P = polyfit(load,y,1); % Polynomial curve fitting of degree 1
eqn = string(" Linear: eqn = " + P(1)) + "x + " + string(P(2)); % Write the equation
end
```

In the second line, the load and the shift ratio $y$ are curve fitted with a polynomial regression. This method uses the linear least-squares regression technique, which allows to do a curve fit of a data set, which is here the load and the shift ratio $y$. In line 3, the equation produced by the linear regression is written, as shown below. `P(1)` correspond to the slope coefficient, genuinely multiplied by $x$, and `P(2)` to the shift ratio $y_{\Delta T}$ induced by the temperature, when the strain $\epsilon = 0$.

### 5.3.4 Temperature and strain estimation

With the value `P(2)` corresponding to the shift ratio $y_{\Delta T}$, the temperature $\Delta T$ estimation can proceed. Subsequently, this will enable to estimate the strain $\epsilon$.

Firstly, the value of `P(2)` is assigned to `yDeltaT`. Then, as seen in line 3, the `ambientTempEstim` is computed, using the equation (5.2), highlighted in section 5.1 and reminded below.

$$\Delta T = \frac{y_{\Delta T}}{(\alpha + \xi)}$$

```
1 % Ambient temperature estimation
2 yDeltaT = P(2);
3 tempEstim = yDeltaT/(alphaf+xi)
4
5 % Strain estimation based on the temperature
6 for i=1:size(strainComp)
7     strainEstim(i) = (deltaLambdaB(i)/(lambdaB)-yDeltaT)/((1-peff));
8 end
```

Based on this result, the strain $\epsilon$ is extracted. For each loading step, an iteration is made within the loop. The strain $\epsilon$ is determined for each load, see line 7, by using the expression equation (5.3), reminded beneath.

$$\epsilon = \frac{\Delta\lambda_B - \Delta\lambda_{B\Delta T}}{\lambda_B(1 - p_{eff})}$$

Finally, the `strainEstim` obtained with this method are compared to the `strainFEM` ones coming from the FEM simulation, as shown in the following section 5.4.

## 5.4 Results of the auto-calibrating method

The solution presented above provides the estimated strain `strainEstim` and estimated temperature `tempEstim`, based on two measurements and a linear regression. These results are computed based on the *"perfect"* values of strain $\epsilon$ and temperature $\Delta T$ inputted in the algorithm. Before comparing the estimated strain `strainEstim` with the numerically computed strain `strainFEM`, it is important to highlight that several sources of errors are disregarded. Noise resulting from the sensors or recording device, and slight variation of temperature or strain are not considered.

Shown below in Figure 5.5, the estimated strain `strainEstim` and the numerically obtained strain `strainFEM` are compared. The difference $\frac{\epsilon_{estim}}{\epsilon_{theor}}$ is plotted as blue circles. On the right of the figure, the estimated temperature `tempEstim` (orange crosses) and the theoretical temperature `tempMeasured` (orange stars) are compared. These values stands against the increase of the load $F$, seen on the x-axis.

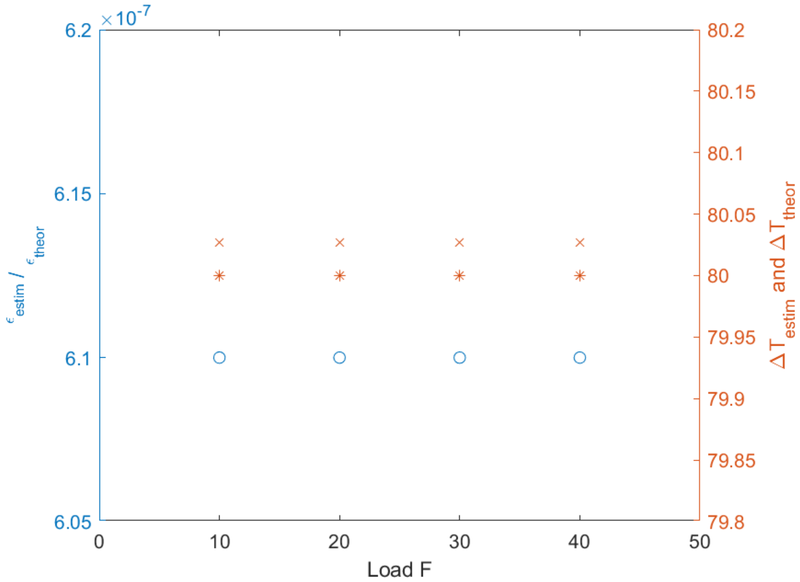



***Figure 5.5.*** Comparison of estimated and theoretical strain and temperature.

As seen above, the values are consistent. This highlight that this comparison is purely theoretical, i.e. the inputs are numerical and not experimental. The data used here is lacking of external error sources. The only source of deviation seen in these results yields from the error induced by the linear regression technique. To validate the reliability of this method, a comparison with experimental results is needed.

## 5.5 Conclusion

The auto-calibration method depicted in the previous lines is based on the linearity of the constitutive relations, while the material is in its elastic region. The strain $\epsilon$ is considered to actuate linearly, which allows to estimate the temperature based on the slope of the increasing shift ratio $y$.

However, this solution implies to compare two steps of loading, where noise and slight variations of physical quantities can disturb and reduce the robustness of the solution. Moreover, this method is not suitable for structures exposed to fast changing temperature or steady loads. A second approach, which is discussed in the next chapter, goes beyond these calibration limitations.

# Chapter 6

# Redundant Sensing method

*This chapter presents the redundant sensing method, which is another ingress to solve the thermo-mechanical decoupling challenge of this project. This approach combines analytical methods with FEM to estimate the temperature measured by the sensor, and subsequently the strain. Moreover, this combination of methods allows to find the magnitude of load applied on the structure under study, which is of great value in SHM.*

To begin, the theoretical principles are depicted in section 6.1. Furthermore, the stream of the variables of the code is elaborated in section 6.2. Finally, the algorithm of the developed method is studied in more detail in section 6.3.

## 6.1 Theoretical principles

Combining FEM with analytical methods allows to go beyond the limitation of the aforementioned method, depicted in chapter 5. As shown below in Figure 6.1, this new approach firstly analyzes the properties of the structure with a FEM analysis. Then, these properties are extrapolated in the elastic range of the material of the structure, and analytical methods are used to estimate the temperature change $\Delta T$. Finally, the loading $F$ is obtained, along with the strain $\epsilon$ measured at the POI.

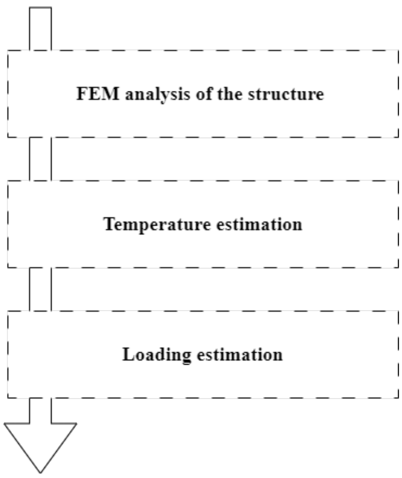



***Figure 6.1.*** Structure of the redundant sensing method, simplified.

### 6.1.1 FEM analysis of the structure

To begin, the structure under study is analyzed with a FEM approach. The first analysis of the structure is made with a known load of $F = 1$ N. This allows to have a first strain $\epsilon$ value for each POI, i.e. each sensor. Based on the linearity aspect of the constitutive relation, and considering that the beam is loaded in the elastic region of the material, a proportionality between the strain $\epsilon$ measurement is derived. The strain $\epsilon_1$ measured at $POI_1$ is equal to the strain $\epsilon_2$ measured at $POI_2$ multiplied by a factor $a$, see the relation (6.1).

$$a = \frac{\epsilon_1}{\epsilon_2} \tag{6.1}$$

### 6.1.2 Temperature estimation

The system of equation shown below in (6.2) is composed of the equation (2.2), presented in section 2.3. By using the $a$ factor, mentioned in the previous section, see (6.1), it is solved.

$$\begin{cases} y_1 = \Delta T(\alpha + \xi) + \epsilon_1(1 - p_{eff}) \\ y_2 = \Delta T(\alpha + \xi) + \epsilon_2(1 - p_{eff}) \end{cases} \tag{6.2}$$

The $a$ factor allows to get rid of an unknown, as shown below in the system (6.3).

$$\begin{cases} y_1 = \Delta T(\alpha + \xi) + \epsilon_1(1 - p_{eff}) \\ y_2 = \Delta T(\alpha + \xi) + a\ \epsilon_1(1 - p_{eff}) \end{cases} \tag{6.3}$$

In the system of equation (6.3) the first equation can be rewritten as beneath, see equation (6.4).

$$\epsilon_1 = \frac{y_1 - \Delta T(\alpha + \xi)}{1 - p_{eff}} \tag{6.4}$$

The term $\epsilon_1$ is then inserted in the second relation, see (6.5) below. The terms $1 - p_{eff}$ are simplified.

$$y_2 = \Delta T(\alpha + \xi) + a\ (y_1 - \Delta T(\alpha + \xi)) \tag{6.5}$$

Now the equation (6.5) is rewritten.

$$\Delta T = \frac{y_2 - a\ y_1}{(1 - a)(\alpha + \xi)} \tag{6.6}$$

Finally, the temperature change $\Delta T$ is computed, based on the shift ratio $y_1$ from the sensor at $POI_1$, the shift ratio $y_2$ from the sensor at $POI_2$ and the $a$ factor depicted earlier.

### 6.1.3 Load estimation

Now that the temperature change $\Delta T$ is known, the strain $\epsilon$ can be easily computed, using the relation (2.2) from section 2.3, rewritten as below, see equation (6.7).

$$\epsilon = \frac{y - \Delta T(\alpha + \xi)}{(1 - p_{eff})} \tag{6.7}$$

The assumption that states that the material is within its elastic region permit to estimate the force applied to the model, based on the proportion between the strain $\epsilon_{F1}$ at the POI for $F = 1$ N, and the strain $\epsilon_S$ at the POI retrieved by the FBG sensors. The proportion is calculated as beneath, see the relation (6.8).

$$F = \frac{\epsilon_{F1}}{\epsilon_S} \tag{6.8}$$

## 6.2 Variable stream in redundant sensing method algorithm

The variable stream description of the redundant sensing method is presented below. The flow and the interaction between the different functions are then visualised. The code is separated in two main groups, see Figure 6.1. On one hand, the *Simulation of the sensor output*, which is meant to simulate the output of a real FBG sensor. On the other hand, the *Computation of the estimated temperature and force*, which based on several variables coming from the first group, computes the temperature $T$ and the force $F$.

### 6.2.1 Simulation of the sensor output

#### Inputs

To begin, the sensor output is simulated. From the location of the sensor, contained in the `cursor_info`, the properties of the sensor `FBG_properties`, presented in 2.3, but also the initial wavelength $\lambda_B$, see equation (2.1) and the temperature change $\Delta T$.

The location of the sensor `cursor_info`, serves to obtain the node ID `node_ID` of the node at the location of the POI. The element `Element_ID` located close to the POI, and the element node ID `Element_NID`, which contain the ID of the surrounding nodes of the element under study. Moreover, the displacement $\{d\}$ at the POI are obtained.

Lastly, the strain `strain_F1`, which is the strain at the loading $F = 1$ N, and the strain `strain_F?`, which is the strain for an *"unknown"* loading, are computed.

#### Computation of the shift ratio

Using the previously find strain `strain_F1` and `strain_F?`, the properties of the sensor `FBG_properties`, but also $\lambda_B$ and $\Delta T$, and adding the strain direction `strain_dir` which indicate the direction of the FBG sensor, the shift ratio `y_POI_F1` for the strain `strain_F1`, and `y_POI` for the strain `strain` are calculated. The sensor's properties `FBG_properties`, the strain `strain_F1`, and the shift ratio `y_POI_F1` are sent to the next *group*.

### 6.2.2 Computation of the estimated temperature and force

#### Temperature estimation

From the three variables aforementioned, the temperature is estimated. The proportionality due to the linear constitutive relations, see the relation (6.1), yield the $a$ variable. The system of equation (6.3) is then solved, and the estimated temperature `temperature_estimated` is computed with equation (6.6).

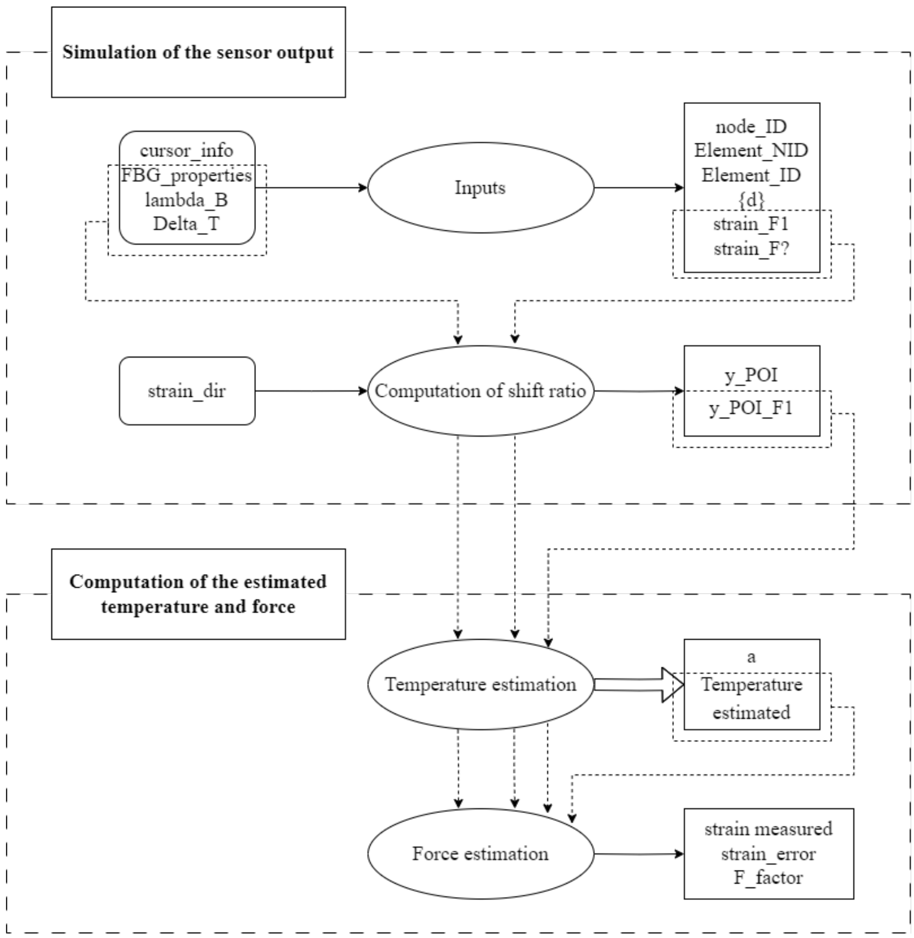



***Figure 6.2.*** Variable stream in redundant sensing method.

#### Force estimation

Now that the strain `strain_F1`, the shift ratio `y_POI` due to the strain `strain_F?` at *"unknown"* load, the properties of the sensor `FBG_properties` and the estimated temperature `temperature_estimated` are known, the last key variables are obtained. The strain at the POI `strain_measured`, and the strain measurement error `strain_error` between the theoretical strain computed in the *Simulation of the sensor output* part and the measured strain `strain_measured` are derived. Finally, the loading applied to the structure `F_factor` is computed, using the relation (6.8), explained in section 6.1.

## 6.3 Algorithm of the redundant sensing solution

The subsequent sub-sections introduce the created code for the *Redundant Sensing method*. Step by step, the calculation is explained, following the stream shown in Figure 6.2. The complete algorithm of this solution can be found in appendix E.

### 6.3.1 Inputs

As a starting point, the parameters of the FBG sensor are introduced in the code, see the table 6.1 below. The constants $C_{strain}$ and $C_{temp}$ are then calculated, using the equations (6.9) and (6.10), leading from the relation (2.2) in section 2.3.

***Table 6.1.*** FBG sensor properties

| Variable | Value | Units |
|---|---|---|
| $\alpha_f$ | 11e-06 | [Kelvin^-1] |
| $\xi$ | 6.5e-06 | [Kelvin^-1] |
| $p_{eff}$ | 0.22 | / |

$$C_{strain} = 1 - p_{eff} \tag{6.9}$$

$$C_{temp} = \alpha + \xi \tag{6.10}$$

Moreover, the initial wavelength $\lambda_B$, which is presented in section 2.3, is inputted. The initial wavelength $\lambda_B$ here is theoretical. As mentioned in the previous chapter, *the initial wavelength is chosen arbitrary based on the common values observed in the literature, ranging between 1530 $\mu m$ to 1580 $\mu m$, see [2] [5] [3].* The *real* initial wavelength $\lambda_B$ corresponding to the sensor used for the experimental approach, can be calculated with the relation (2.1), in section 2.3. The initial wavelength $\lambda_B$ used here is equal to 1550 $\mu m$.

Then, the initial temperature $T_0 = 0$, and the final temperature $\Delta T = 80$ are inputted. The difference in temperature, i.e. the temperature change $\Delta T$, is obtained.

Subsequently, as shown in line 2, the variables `[strain, strain_F1, POI_info]` are computed, using the function described in the chapter 4, section 4.2. The variable `strain`, stands for the strain $\epsilon_S$, see the previous section 6.1, at the POI for the *"unknown"* load $F$. The load $F$ is chosen upstream, when the FEM analysis is done. The FEM simulation allows to simulate the strain $\epsilon_S$ at the location of the sensors. Moreover, the strain $\epsilon_{F1}$ allocated in the variable `strain_F1`, see 6.1, is also computed with the same FEM model, but with a load $F = 1$ N. The `POI_info` structure contains information related to the POI, as shown in section 4.2.

```
% Strain computation at location of each sensor for F = 1 and F = x N
[strain, strain_F1, POI_info] = strain_computation_element_scale...
    (cursor_info, model, d);
```

### 6.3.2 Computation of the shift ratio

Subsequently, the shift ratio $y$ is computed. Firstly, the direction of assessment of the strain $\epsilon$, which correspond to the orientation of the sensor, is stated in line 6. The strain directions $x, y, z$ are related to the number one to three, as depicted in line 2 to four below.

```
1 % Choose the direction [x, y, z] of the measure
2 dir_x = 1; % Coordinate of x direction in strain matrix
3 dir_y = 2; % Coordinate of y direction in strain matrix
4 dir_z = 3; % Coordinate of z direction in strain matrix
5
6 strain_dir = dir_x ; % Choose the direction of the strain [x, y, z]
```

The shift ratio $y_{F1}$ and $y_s$ are then computed, in line 4 and 13, using the relation (2.2) from section 2.3, through the function `calc_wavelength_shift`. The along the direction `strain_dir` chosen previously are used, as the temperature change $\Delta T$, the constants $C_{strain}$ and $C_{temp}$, and the initial wavelength shift $\lambda_B$.

```
1 % Computation of the shifting ratio at F = 1 N
2 for j = 1:numel(cursor_info)
3
4     yf1 = calc_wavelength_shift(strain_F1(strain_dir,j), deltaT, ...
5         Cstrain, Ctemp, lambdaB);
6     y_POI_F1(j,1) = yf1;
7
8 end
9
10 % Computation of the shifting ratio at F = ? N (searched value)
11 for j = 1:numel(cursor_info)
12
13     yf = calc_wavelength_shift(strain(strain_dir,j), deltaT, ...
14         Cstrain, Ctemp, lambdaB);
15     y_POI(j,1) = yf;
16
17 end
```

These two loops iterates over $j$, which is the quantity of selected POI calculated by `numel(cursor_info)`, see line 2 and 11. The `cursor_info` extraction process is discussed in section 4.1. The `y_POI_F1` and `y_POI` variables store the shift ratio $yf1$ and $yf$ respectively, for each iteration, i.e. each sensor.

### 6.3.3 Temperature estimation

From the strain $\epsilon_S$ and $\epsilon_{F1}$, the $a$ factor is obtained in line 2, using the relation (6.1) shown in section 6.1.

```
1 % The factor a between strain 1 and 2 is computed
2 a = strain_F1(1,2)/strain_F1(1,1);
3
4 % Temperature estimated based on factor a and output of the two sensors
5 deltaTEstim = (y_POI_F1(2)-a*y_POI_F1(1))/(Ctemp-a*Ctemp);
```

The temperature change $\Delta T$, allocated within the variable `deltaTEstim`, is obtained with on the wavelength shifts, as shown in the line 5 above. This relation correspond to the equation (6.6) explained in section 6.1.

### 6.3.4 Force estimation

To begin, the temperature measurement error is assessed, in line 2, and stored in variable `temp_error`. Then, in line 5 to eight, the strain $\epsilon_S$ from the simulated sensors are computed. The strain error is then estimated, by comparing the strain $\epsilon_S$ retrieved by the sensors and the strain $\epsilon$ coming from the FEM analysis, in line 10 to 13.

```
1 % Temperature error
2 temp_error = deltaT/deltaTEstim-1;
3
4 % Strain measured at POIs, based on the shift ratios, FBG cst. and deltaT
5 strain_measured1 = (y_POI(1)-Ctemp*deltaTEstim)/Cstrain;
6 strain_measured2 = (y_POI(2)-Ctemp*deltaTEstim)/Cstrain;
7 strain_measured3 = (y_POI(3)-Ctemp*deltaTEstim)/Cstrain;
8
9 % Strain measurement compared to theoretical strain
10 strain_error1 = strain(1,1)-strain_measured1;
11 strain_error2 = strain(1,2)-strain_measured2;
12 strain_error3 = strain(1,3)-strain_measured3;
```

Finally, the force $F$ applied on the model is estimated, by computing the proportionality between the strain $\epsilon_S$ and the strain $\epsilon_{F1}$, using the equation (6.8) in section 6.1. The force $F$ is computed for each POI, see `F_factor1`, `F_factor2`, and `F_factor3`, in line 3 to five. The force $F$ is an average of these three forces, see `force_measured` in line 8.

```
1 % Force estimation based on comparison between measured strain at POIs
2 % and F = 1 N strain at POIs.
3 F_factor1 = strain_F1(1,1)/strain_measured1;
4 F_factor2 = strain_F1(1,2)/strain_measured2;
5 F_factor3 = strain_F1(1,3)/strain_measured3;
```

```

% Average of the measured forces
force_measured = 3/(F_factor1 + F_factor2 + F_factor3);
```

The value of the measured force $F$ and the estimated temperature change $\Delta T$, respectively `force_measured` and the value of `deltaTEstim`, are then retrieved. The values yielding from this method are discussed in the following section 6.4.

## 6.4 Results of the redundant sensing method

The redundant sensing method discussed in the previous sections yields promising results when the inputs are purely numerical. As shown below in Figure 6.3, the estimated load $F$ (blue dots) and estimated temperature `tempEstim` (orange crosses) are obtained with a negligible error. This error is directly proportional to the refinement of the mesh of the model, i.e. the number of elements, see table 6.2.

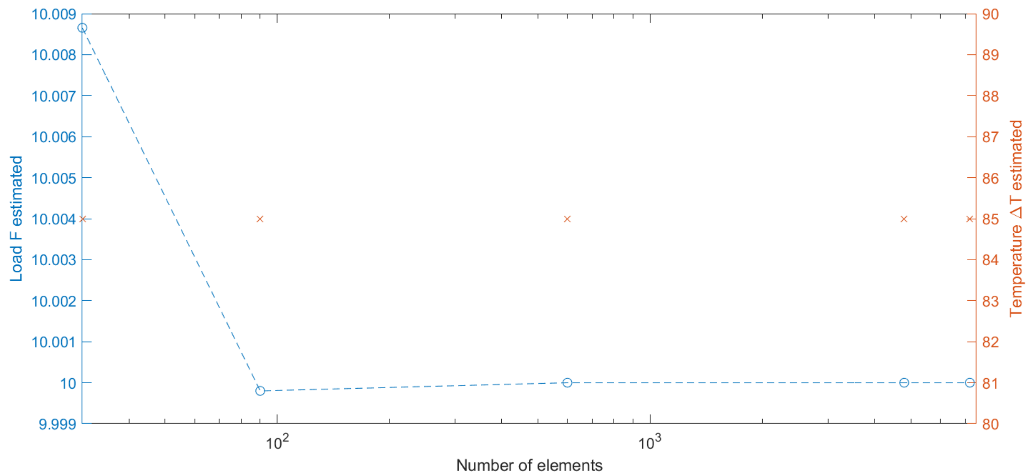



***Figure 6.3.*** Comparison of estimated load and estimated temperature against mesh refinement.

***Table 6.2.*** Effect of mesh refinement on force and temperature estimation.

| | Test 1 | Test 2 | test 3 | Test 4 | Test 5 |
|---|---|---|---|---|---|
| Number of elements | 30 | 90 | 600 | 4800 | 7200 |
| Estimated load $F$ | 10.0087 | 9.9998 | 10.0001 | 9.9999 | 10.0001 |
| Estimated temperature $\Delta T$ | 85 | 85 | 85 | 85 | 85 |

As shown above, the error is insignificant. However, external perturbations, like noise during the measuring process, is not considered in this comparison. The verification campaign of this method reaches its limit here. Experimental data is needed to assess the behavior of the algorithm.

## 6.5 Conclusion

This methodology, *in fine*, allows to obtain the temperature of the structure under study, by solving a system of equations based on the linear properties of the constitutive relations, and a FEM model. Moreover, the loading applied on the structure is estimated, by comparing the results of a FEM analysis, and the measurement made by the sensors. This solution then combines the FEM with *real life* measurements, which allows with redundant information to extrapolate and obtain the quantities of interest.

This method, the redundant sensing method, is used within the experimental validation procedure, which is presented in the next chapter.

# Chapter 7

# Experimental validation process

*The objective behind the decoupling of the thermo-mechanical behavior of the FBG sensor is to develop a novel approach for sensing in SHM. Until now, the development of the methodologies to solve this challenge are theoretical, and experimental confirmations is genuinely needed to confirm the viability of the proposed solution. Crossing the data coming from the theoretical solution using the FEM data, with the experimental results which are subject to noise and external alteration, is then critical. In the following lines, the experimental approach is derived.*

Firstly, the section 7.1 introduces the experimental procedure to be followed, introducing the key steps and goals of this experiment. Then, in section 7.2, the technical details, for instance the location of the different sensors, is discussed.

## 7.1 Experimental testing procedure

As the FBG sensor needs to be assessed in a variety of situations, with temperature changes and load variations, a rigorous organization of the experimental campaign is key. The structure under study here is the academic case presented in section 3.3, the technical details of the set-up are explained in the next section 7.2.

### 7.1.1 Main steps

The experimental approach follows key step, illustrated below in Figure 7.1. To begin, the FBG sensors, but also the strain gages are installed. The FBG sensors are installed on the samples by a fellow researcher from VUB, while the strain gages are installed in-house at KU Leuven. The location of each sensor is highlighted in the subsequent section 7.2. When the sample is equipped with the sensing equipment, the entire set-up is installed in a climate chamber, where the temperature is regulated based on the targeted temperatures, discussed in the following sub-section. Finally, the experiment is realised, and the outputs of the sensors are gathered and compared.

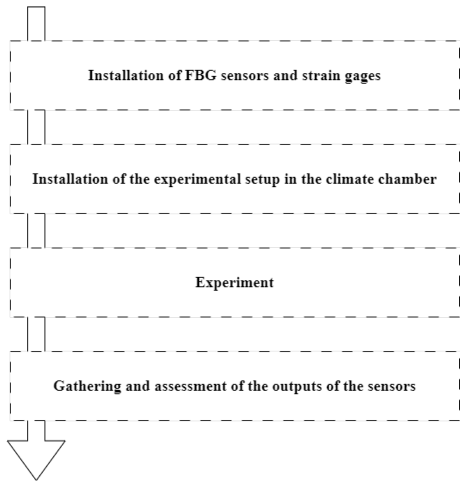



***Figure 7.1.*** Experiment procedure.

### 7.1.2 Temperature change $\Delta T$

As the equipped sample needs to be assessed in various temperature conditions, a predefined heating pattern is derived. Regarding the temperature $\Delta T$, the span ranges in between $0°C$ and $85°C$. To facilitate the experimental approach, and despite of the lower bound being $0°C$, the experiment starts at ambient temperature, i.e. $20°C$. Then the temperature rise up to the upper bound of the temperature range, by increment of $10°C$, and $15°C$ for the last one. Here it is key to stabilize the temperature $\Delta T$ at each step, see Figure 7.2 beneath. This ensure that an homogeneous temperature over the sample is reached when the measurement has to be done. Each measurement is made 10 minutes after the temperature rise. Then, the remaining 10 minutes are used to apply the increasing load.

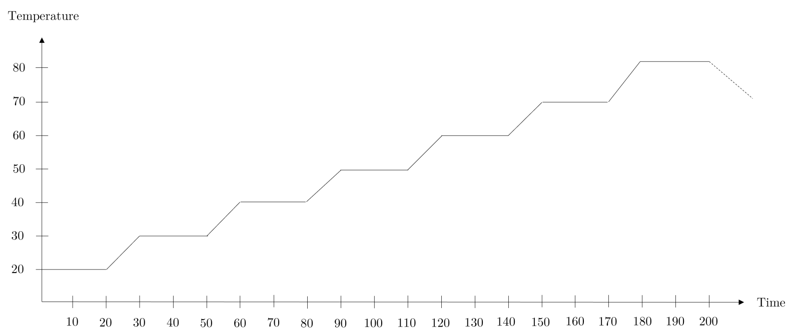



***Figure 7.2.*** Notched temperature increase.

### 7.1.3 Loading of the structure

The loading is key in this experiment. It is applied by steps, after stabilisation of the temperature. At each temperature increase, see Figure 7.2 above, a stage is reached and maintained during 20 minutes. After 10 minutes in a stage, the first load of 10 N is applied. Then, by increments of 10 N every 2 minutes, the objective of 40 N is reached, as illustrated in Figure 5.3, and in table 5.2, in section 5.1.

### 7.1.4 Searched values

The searched values behind this experiment is genuinely the wavelength shift $\Delta\lambda_B$, for each load step and each rise in temperature, at each POI. But additionally, the strain $\epsilon$ measurement from the installed strain gages, as the retrieved temperature $T$ from the thermocouple, are critical to confirm the relevancy of the results yielded by the FBG sensors. These supplementary data allows to wisely compare the results of the experiment.

## 7.2 Technical specifications

**Location of the POI**

The sample is equipped with three FBG sensors, modeled in red in the Figure 7.3 underneath, which measure the strain $\epsilon$ and temperature $\Delta T$ at three POI. These quantities are retrieved as a shift ratio $y$ for each sensor. The location of the three sensors is shown below, in Table 7.1.

***Table 7.1.*** POI location on the sample.

| | x | y | z |
|---|---|---|---|
| POI 1 | 80 | 12.5 | 1.5 |
| POI 2 | 120 | 12.5 | 1.5 |
| POI 3 | 160 | 12.5 | 1.5 |

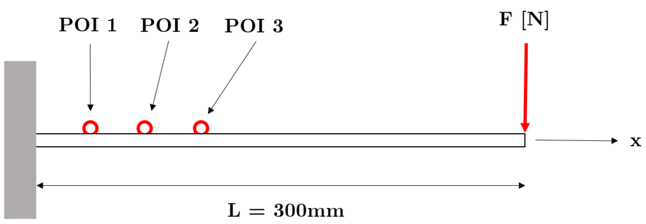



***Figure 7.3.*** Location of the FBG sensors on the sample.

### 7.2.1 Strain gages parallel measurements

To confirm the strain $\epsilon$ measurement of the FBG sensors at the POIs shown in Figure 7.3 above, strain gages are also installed on the model. The goal here is to measure the strain $\epsilon$ on the opposite side of the beam, illustrated in Figure 7.4 by the location of the strain gages $SG1$, $SG2$, and $SG3$, on the inner surface. The strain $\epsilon$ being the same due to the symmetry, but negative because of the compression, the strain $\epsilon$ measured on the upper side is double checked.

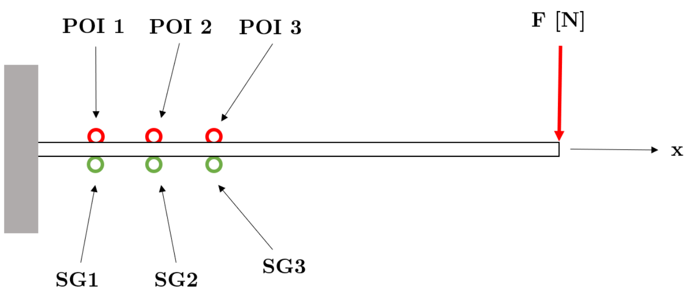



***Figure 7.4.*** Location of the strain gages on the sample.

### 7.2.2 Temperature control

The temperature $T$ of the experiment is controlled by a climate chamber, where each step, illustrated in the Figure 7.2, are rigorously reached. The sample studied, is installed within the climate chamber as illustrated below in Figure 7.5. The temperature $T$ of the sample is assessed with a thermocouple sensor.

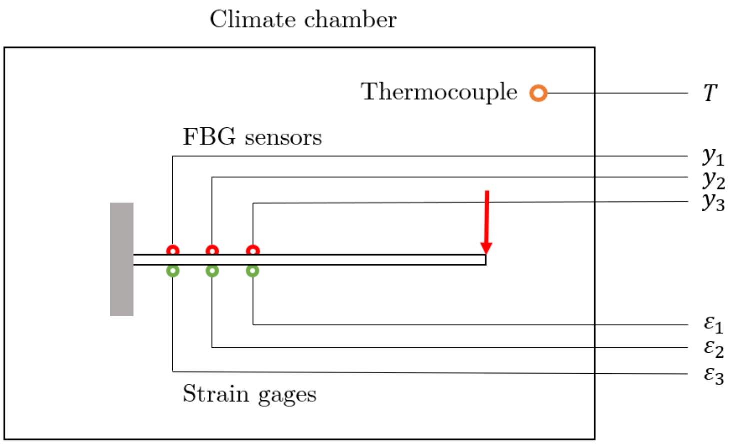



***Figure 7.5.*** Experimental configuration and sensors outputs.

## 7.3 Conclusion

The experimental procedure to assess the thermo-mechanical behavior of the FBG sensors, explained in the previous sections, allows to evaluate the robustness and reliability of the created methods, presented in Chapter 5 and 6. The outputs of the FBG sensors are processed using these two methods. Then, they are compared with the strain $\epsilon$ measured by the strain gage and the temperature $T$ measured by the thermocouple. This comparison permits to estimate the validity of the *auto-calibrating method* and the *redundant sensing method*.

The experimental campaign is ready to be proceeded.

# Chapter 8

# Discussion and improvements

*At this step of the project, the completion of the objectives is evaluated, and the solution provided by this work discussed. This step is crucial to state the evolution of the work, but also to orientate the future investigation for the next step above the end of this project. Some suggestion are then provided, to improve the methods developed, but also to reopen the scope of the investigation and to propose new directions of research.*

As a starting point, the completion of the objectives is stated in section 8.1, by correlating the objectives with the main chapters and achievements of this project. Subsequently, the results leading from the different methods, and their contribution to the state of the art, is discussed. Then, some improvements suggestions are provided, and future research directions are suggested.

## 8.1 Completion of the objectives

As depicted in Figure 1.1 in section 1.3, the objectives are brought from the introduction to the end of the project where the progression is stated. Each chapter corresponds to one or more main achievement, which counter the goals of the project.

- Development of a numerical toolbox to enable the use of virtual FBG sensors in combination with FEM models.

  To begin, in Chapter 3, the NoLife FEM library internally developed at MSD is assessed. This ensure that the toolbox designed during this project will be properly and robustly merged to the existing library. Then in Chapter 4, several functions are derived. These functions are key to extract the physical quantities, like the strain, at precise location over the FEM model. These values are then interpreted by the virtual FBG sensors to provide a virtual output. This toolbox is used subsequently to develop the decoupling methods needed for the enhancement of the FBG sensing technology.

- Development of a method to estimate separately the temperature and the strain, based on the output of one or several FBG sensors.

By using the toolbox derived earlier, two different approach to temperature estimation can be derived, see Chapter 5 and Chapter 6. The first one, the *auto-calibrating method*, focuses on a calibration approach while the second one, the *redundant sensing method*, combines a FEM and an analytical method. These two solutions are tested theoretically and numerically. The results of these tests are summed in the next section 8.2. However, the *auto-calibrating method*, and the *redundant sensing method* needs to be assessed experimentally.

- Development of an experimental campaign to confirm and challenge the derived method.

  Lastly, an experimental campaign is derived, see Chapter 7. This experimental procedure is meant to challenge and confirm the validity of the two theoretical methods created along this project. The robustness of these two methods could then also be assessed, seeing the influence of external error sources, like noise for example. Despite of this, the experimental campaign could not be proceeded in the time frame of the project, due to manufacturing delays of the test samples.

As depicted above, it is seen that most of the goals are covered by the work done in this report. Acquiring the experimental data would permit to complete the development of this solution. The results leading from the two methods developed during this project are discussed in the next section 8.2, before being compared with the existing solutions, presented in section 2.4.

## 8.2 Results comparison and contribution to the state of the art

Firstly, the *auto-calibrating method* and the *redundant sensing method* are compared to state the level of improvement provided by these two solutions. This allows to subsequently evaluate the contribution to the state of the art of these two methods.

### 8.2.1 Auto-calibrating method and redundant sensing method

To start, these two approaches share one main limitation. They are only valid within the elastic regime of the material used within the structure under study. They use the linear property of the constitutive laws to estimate the search physical quantities. Despite of this, the two solutions shows important differences. On one hand, the *auto-calibrating method* follows a calibrating approach, see Chapter 5, while on the other hand the *redundant sensing method* comes with a numerical and analytical hybrid method, see Chapter 6. The *auto-calibrating method* allows to obtain the strain $\epsilon$ and temperature $\Delta T$ by using only one sensor. However, this solution requires two steps where the loading changes, which induces limitations in SHM of structure with fast changing temperature or steady loading. On the other side, the *redundant sensing method* estimates the strain $\epsilon$ and temperature $\Delta T$ by comparing the output of two sensors and does not need two loading steps. Nevertheless, this method needs a pre-process step done with a FEM analysis. The behavior of the structure understudy needs to be assessed before the outputs of the sensors can be processed for the first time. Even so, the FEM analysis only needs to be run once. The use of a glsFEM model as a background can however induce sources of errors if the FBG sensors are not located properly on the structure, i.e. at the same location as in the FEM model.

The two solutions shows their own advantages and limitations. The *auto-calibrating method* does not depends of a numerical simulation and only needs one sensor, which allows to extract temperature at the direct vicinity of the sensor. Still, two different loading steps are needed. Aside, the *redundant sensing method* does not depend on several loading steps, and can provide temperature without any limitations. However, two sensors are required, and a FEM analysis is needed as a starting point to compare the output of the sensors.

### 8.2.2 Contribution to the state of the art

Presented in Chapter 2, section 2.4, two main solutions already exists to decouple the thermo-mechanical behavior of FBG sensors. First, the double fiber setup, which consist in two fibers laying close one to the other. One of the fiber measures the strain $\epsilon$ and the temperature $\Delta T$, while the other one only measures the temperature $\Delta T$. By subtracting the two results, the strain $\epsilon$ is obtained. The other solution is an FBG and EFPI coupled setup. Here the two technologies are merged to create a sensor that decouples the strain $\epsilon$ and the temperature $\Delta T$. The common withdraw of these two solution is their manufacturing complexity. On one side, with the first solution, the two fibers need to lay really close, while one is independent of the deformation of the structure. On the other hand, the FBG and EFPI coupled setup is a blend of two sensors, which needs to be precisely combined and sealed, see section 2.4.

The novel approaches presented in this work goes beyond this manufacturing limitation. The strain $\epsilon$ and the temperature $\Delta T$ can be obtained by numerical decoupling methods, which allow to use only one fiber to assess the health of the structure under study. This permits to install the FBG sensors in more complex locations over a structure, and reduce the cost of manufacturing and installation of this sensing technology.

## 8.3 Improvement suggestions and broadening of the scope

A list of improvements is suggested in the following lines. Some of the propositions could be implemented in the nearby next step of the project, while other would take place in the long term objectives. The scope is reopened.

### 8.3.1 Nearby step objectives

The *computation of the strain at element scale* function, computes the strain $\epsilon$ value at the first element, see section 4.2. An improvement would be to compute the strain within the four elements and average the result. This would yield more relevant results at the node of interest. An essential improvement would be to generate some experimental data. That would enable the comparison of the theoretical data from the numerical methods with the experimental results. The estimation method could then be adjusted, and the theoretical method confirmed. Regarding temperature, the range under study could be extended to negative but also higher positive temperatures. However, some technical limitation appears, in particular with the adhesive in the high temperature span.

### 8.3.2 Long term objectives

Furthermore, the system of equation shown in the relation (6.2) in section 6.3, could be expanded, by adding more FBG sensors. The goal would be to gather more information about the structure under study, and cross them to more robustly estimate the strain and temperature unknowns. An other improvement would be to go beyond the linear limitation, by implementing the material property's curve. Then, the proportional factors used to estimated the strain and forces over the structure, would vary with the load. Moreover, the assessment of dynamically loaded structure would be a great improvement. FBG sensors allows quasi-distributed measurement, so mode shapes could be more easily observed. Lastly, an optimization algorithm could also be developed to chose the optimum location and number of FBG sensor used for the SHM of the structure under study. The optimization could be based on the initial wavelength of each sensor and the range of strain expected at each POI.

# Chapter 9

# Conclusion

*The main challenge of this project was to enhance the FBG sensing technology and more precisely getting through the decoupling challenge of the thermo-mechanical behavior of the FBG sensor. To ensure that the problem frame was properly defined, a list of objectives and a reduced-realistic scope has been derived so the problem can be solve in the time frame of the project. Subsequently, the theory behind the FBG sensors has been assessed and the key equations and physical properties acknowledged. Moreover, the FEM NoLife library work flow and theory has been studied to ensure that the different functions designed and used for the FBG sensor toolbox are robustly aggregated to the NoLife library. These essential functions then allowed the development of two temperature estimation methods. The strain has then been found at the location of each sensor under study, and in the second method, the force applied on the model has been estimated, theoretically.*

*To confirm the robustness of these solutions, an experimental campaign needs to be realised. Unfortunately, manufacturing delays has postponed the acquisition of the experimental data, which will be available in the near future. Nevertheless, after comparing the initial objectives and the achievements of the project, it is conclude that the main goals are reached, and that a solution has been provided for the decoupling of the thermal and mechanical measurements of the FBG sensor.*

# Bibliography


[1] HP FISCHER-HIRCHERT ULRICH. *PHOTONIC PACKAGING SOURCEBOOK: FIBER-CHIP COUPLING FOR OPTICAL COMPONENTS, BASIC CALCULATIONS, MODU.* SPRINGER, 2015.

[2] Sidney Goossens. "Barely visible impact damage detection in aerospace-grade carbon fibre reinforced polymer components with optical fibre sensors". PhD thesis. June 2021. DOI: 10.13140/RG.2.2.23149.05606.

[3] Hyun-Kyu Kang et al. "Simultaneous measurement of strain and temperature of structures using fiber optic sensor". In: *Journal of intelligent material systems and structures* 12.4 (2001), pp. 277–281.

[4] A. Bertholds and R. Dandliker. "Determination of the individual strain-optic coefficients in single-mode optical fibres". In: *Journal of Lightwave Technology* 6.1 (1988), pp. 17–20. DOI: 10.1109/50.3956.

[5] Manfred Kreuzer. "Strain measurement with fiber Bragg grating sensors". In: *HBM, Darmstadt, S2338-1.0 e* 12 (2006).

[6] Robert D Cook et al. *Concepts and applications of finite element analysis.* John wiley & sons, 2007.

[7] Peter I Kattan. *MATLAB guide to finite elements: an interactive approach.* Springer Science & Business Media, 2010.

# Appendix A

# Node and element finder

```
function [node_ID, element_ID] = node_and_element_finder(model,node_location)

% ============================================= %

%                  NODE AND ELEMENT ID FINDER

% ============================================= %

%   The goal of this function is to find the element connected to the node
%   selected. The node ID is firslty found, then the four element sharing
%   this node are listed, and the first element of this list is selected.

% ============================================= %

clc, clear node_ID element_ID i

% ============================================= %
%             SEEKING NODE BASED ON ITS COORDINATES
% ============================================= %

% Loop to find the ID of the selected node, based on its coordinates
for i = 1:numel(model.NODES)

    if [node_location] == [model.NODES(i).CO]
        node_ID = i;
          else
    end

end

% Print status of the research process at the end of the loop
if exist('node_ID','var') == 1
    node_ID;
else
```

```
    Message = "Error: Node not found"
    node_ID = [ ]
end

% ============================================ %
%              SEEKING THE 4 ELEMENTS CONNECTED BY THIS NODE
% ============================================ %

% Loop to find the IDs of the 4 elements connected to the selected node
n = 0;
for i = 1:numel(model.ELEMENTS)

     if ismember(node_ID, model.ELEMENTS(i).NODES);
          n = 1+n;
          element_ID(n,1) = model.ELEMENTS(i).ID;
          else
     end

end

% Print status of the research process at the end of the loop
if exist('element_ID','var') == 1
    element_ID = element_ID(1); % First element (of 4) is chosen
else
    Message = "Error: Element not found"
    element_ID = [ ]
end
```

# Appendix B

# Strain computation at element scale

```
function [strain, strain_F1, POI_info] = strain_computation_element_scale(cursor_info, model, d)

% ================================================ %

%                    STRAIN COMPUTATION AT ELEMENT SCALE

% ================================================ %

% This function compute the strain-displacement matrix [B], the strain and the stress for elements located at some point of interest.

% To use this function, the user needs to firstly run the FEM model. Then, the user
% needs to select 3 nodes on the ploting of the model, and export them (with
% right-click) as a cursor_info. Then, this function can run.

% This function retrieves the strain at |F| = 1 N, and at the desired F,
% for the point of interest selected previously. When a point of interest
% is selected, the node is used to find the adjacent elements. The first
% element in the list of adjacent element is used to compute the strain at the node of interest.
% The strain is retrieved for this specific point of interest.

% ================================================ %

clc, clf, close all, clear element_ID node_ID strain stress i

% ================================================ %
%                         INPUTS
% ================================================ %

load('d_F1'); % Import displacement of nodes at F = 1, from .mat file

% ================================================ %
```

```
% LOOP OVER ELEMENTS OF INTEREST
% ============================================================ %

for j = 1:numel(cursor_info)
    node_location = cursor_info(j).Position;

% ============================================================ %
% SEEKING FOR ELEMENT AND NODE INFORMATION
% ============================================================ %

clear d_node_F1 d_node

% Research of node ID and element ID
[node_ID, element_ID] = node_and_element_finder(model,node_location);

% Fetch current element from the elements struct, using element ID and node ID
elementNID = model.ELEMENTS(element_ID).NODES;
element = model.ELEMENTS(element_ID);

% Indexing gg & aa set
nNoD = numel(element.NODES); % Auxiliary variable
igg2aa = iLocalGG2AA(nNoD);

% Extracting element coordinates from struct
NodeCO = model.AUX.x0(element.iDOF(igg2aa));

% ============================================================ %
% COMPUTATION OF [D]
% ============================================================ %
% Material parameters
E   = model.MATERIALS(1).PARAM(3);
nu  = model.MATERIALS(1).PARAM(4);
rho = model.MATERIALS(1).PARAM(2);

% compute D matrix (constitutive relation)
D = ...
    (E/((1+nu)*(1-2*nu)))*...
      [1-nu, nu , nu ,     0     ,     0     ,     0     ; ...
       nu , 1-nu, nu ,     0     ,     0     ,     0     ; ...
       nu , nu , 1-nu,     0     ,     0     ,     0     ; ...
       0  , 0  , 0  , 0.5*(1-2*nu),     0     ,     0     ; ...
       0  , 0  , 0  ,     0     , 0.5*(1-2*nu),     0     ; ...
       0  , 0  , 0  ,     0     ,     0     , 0.5*(1-2*nu)];

% ============================================================ %
% COMPUTATION OF [B]
% ============================================================ %

```

```
% Memory allocation
nDOFD = length(NodeCO); % Auxiliary variable
d_node = zeros(1,24);

% Extraction of the node coordinates within the element coordinate system
ElemCoID = find(element.NODES==node_ID)
if ElemCoID == 1
    ElemCO = [-1,-1,-1]
elseif ElemCoID == 2
    ElemCO = [1,-1,-1]
elseif ElemCoID == 3
    ElemCO = [1,1,-1]
elseif ElemCoID == 4
    ElemCO = [-1,1,-1]
elseif ElemCoID == 5
    ElemCO = [-1,-1,1]
elseif ElemCoID == 6
    ElemCO = [1,-1,1]
elseif ElemCoID == 7
    ElemCO = [1,1,1]
elseif ElemCoID == 8
    ElemCO = [-1,1,1]
else
    Message = "Error : node not found"
end

% Shape function related computations
[N,dN] = HEXA8_ShapeFunction(ElemCO,NodeCO,1,1,1);

% Strain displacement matrix B
B = B_3D_linear(dN,nNoD);
B = B([1:6],[3*ElemCoID-2:3*ElemCoID])

% ============================================== %
%                STRAIN AND STRESS COMPUTATION
% ============================================== %

% Computation of the displacement and strain at F = 1 N
d_node_F1 = [d_F1(elementNID(ElemCoID)*3-2), d_F1(elementNID(ElemCoID)*3-1),...
    d_F1(elementNID(ElemCoID)*3)];

d_node_F1 = d_node_F1';
strain_F1 = B*d_node_F1; % Strain computation for the entire element

% Computation of the displacement and strain at F = X N
d_node = [d(elementNID(ElemCoID)*3-2), d(elementNID(ElemCoID)*3-1),...
    d(elementNID(ElemCoID)*3)];
```

```
d_node = d_node';
strain = B*d_node; % Strain computation for the entire element

stress = D*strain % stress computed for the entire element

% ================================================ %
%                    STORING VARIABLES IN STRUCTURE
% ================================================ %

% Here the key informations are stored in a struct.

if j == 1

        POI_info.POI1.B1 = B;

        POI_info.POI1.strain1 = strain;
        POI_info.POI1.d_node1 = d_node;

        POI_info.POI1.node_ID1 = node_ID;
        POI_info.POI1.element_ID1 = element_ID;
        POI_info.POI1.element_NID1 = elementNID;

        POI_info.POI1.strain_F1_1 = strain_F1;

    elseif j == 2

        POI_info.POI2.B2 = B;

        POI_info.POI2.strain2 = strain;
        POI_info.POI2.d_node2 = d_node;

        POI_info.POI2.node_ID2 = node_ID;
        POI_info.POI2.element_ID2 = element_ID;
        POI_info.POI2.element_NID2 = elementNID;

        POI_info.POI2.strain_F1_2 = strain_F1;

    elseif j == 3

        POI_info.POI3.B3 = B;

        POI_info.POI3.strain3 = strain;
        POI_info.POI3.d_node3 = d_node;

        POI_info.POI3.node_ID3 = node_ID;
        POI_info.POI3.element_ID3 = element_ID;
        POI_info.POI3.element_NID3 = elementNID;

```

```
            POI_info.POI3.strain_F1_3 = strain_F1;

        else
            Message = 'Error'
end

end

% ============================================= %
%                 RESULTS AND PLOTING
% ============================================= %

% The strain is stored here in matrix form.
strain_F1 = [POI_info.POI1.strain_F1_1 POI_info.POI2.strain_F1_2 ...
    POI_info.POI3.strain_F1_3];
strain  = [POI_info.POI1.strain1 POI_info.POI2.strain2 ...
    POI_info.POI3.strain3];

% Ploting of the model, with and without the load applied, with the nodes
% selected higlighted.
plot_3D_mesh(model, [POI_info.POI1.node_ID1, ...
    POI_info.POI2.node_ID2, POI_info.POI3.node_ID3], ...
    model.AUX.x0, model.AUX.x0, 0, 0, 0, [0 0.4], [-0.02 0.02],  ...
    [-0.02 0.02], 0.1, 0.8, 0);
plot_3D_mesh(model, [POI_info.POI1.node_ID1, ...
    POI_info.POI2.node_ID2, POI_info.POI3.node_ID3], ...
    model.AUX.x0, model.AUX.x1, 0, 0, 0, [0 0.4], [-0.02 0.02],  ...
    [-0.02 0.02], 0.1, 0.8, 0);

% ============================================= %

clear j y_j strain1 strain2 strain3 strain_F1_1 strain_F1_2 ...
    strain_F1_3 element_NID1 element_NID2 element_NID3 ...
    element_NID element_ID1 element_ID2 element_ID3 ...
    element_ID nDOFD nNoD node_ID NodeCO NodeCO_RS N n ...
    Ip1 Ip2 Ip3 IntegrationFactor igg2aa i GP_W GP_CO ...
    elementNID elementNumber ElemCO dN disp_per_node ...
    disp_per_node_F1 dim detJ d_node_F1 d_node rho nu E ...

end
```

# Appendix C

# Wavelength shift and shift ratio calculation

```
function [y] = calc_wavelength_shift(strain, deltaT, Cstrain, Ctemp, lambdaB)

% =============================================== %

%                 WAVELENGTH SHIFT AND SHIFT RATIO CALCULATOR

% =============================================== %

%  The goal of this function is to compute the wavelength shift
%  retrived by a FBG sensor measuring the strain and the temperature
%  change at a point of interest. This function retrieves a shift ratio.

% =============================================== %

% Wavelength shift (induced by a variation of strain and/or temperature) calculation
deltaLambdaB = lambdaB*((Ctemp*deltaT)+(Cstrain*strain));

% Shift ratio between wavelength shift and initial wavelength
y = deltaLambdaB/lambdaB;

end
```

# Appendix D

# Auto-calibrating method

```
% ================================================ %

%                    AUTO-CALIBRATION METHOD

% ================================================ %

clc, clf, clear all

% ================================================ %

% Inputs

% FBG properties
alphaf = 11e-06; % Coefficient of thermal expansion of the specimen material [Kelvin^-1]
xi = 6.5e-06; % Thermo-optical coefficient [Kelvin^-1]
peff = 0.22; % Effective photoelastic coefficient

% Initial wavelength
lambdaB = 1550*1e-6; % Initial wavelength retrieved by the FBGS [m]

% Temperature change
deltaT = 80; % Ambient temperature, used to compute the theoretical wavelength shift

% Strain measured at POI, for different temperature T, data imported from excel sheet
sheetName = string(deltaT); % Sheet name correspond to the ambient temperature (T + ambient temperature)
strainFEM = xlsread('strain_data_fea.xlsx',sheetName,'C4:C7'); % Data imported from table and put in matrix form
load = xlsread('strain_data_fea.xlsx',sheetName,'B4:B7'); % Load data imported from table and put in matrix form

% ================================================ %

% Computation of deltaLambdaB and shift ratio
```

```
[rowstrainFEM, colstrainFEM] = size(strainFEM); % Extract number of row and columns of the matrix for the loop
deltaLambdaB = zeros(rowstrainFEM,colstrainFEM); % Allocate space for matrix

for i=1:rowstrainFEM

    dlb = fbg_wavelength_shift_calculator(strainFEM(i,1),deltaT,...
        alphaf,xi,lambdaB,peff); % Wavelength shift calculator
    deltaLambdaB(i) = dlb; % Fill up deltaLambdaB matrix with value at each loop iteration
    y(i) = deltaLambdaB(i)/lambdaB; % Fill up y matrix with value at each loop iteration

end

% ================================================ %

% Loop for ploting the results
[rowy, coly] = size(y); % Extract number of row and columns of the matrix for the loop

for i=1:coly

        scatter(load(i),y(i),"*") % Plot y against the load
        hold on

end

xlabel('Load (N)')
ylabel('Wavelength shift (y = \Delta\lambda_B / \lambda_B)')

% ================================================ %

% Linear regression to obtain the slope of the y to strain curve
P = polyfit(load,y,1); % Polynomial curve fitting of degree 1
eqn = string(" Linear: eqn = " + P(1)) + "x + " + string(P(2)); % Write the linear curve equation

% Plot of the linear regression line
plot(load,y,'r-.'); % Plot each value of y against the load
axis([0 50 1.4e-3 1.9e-3]) % Set the limit of the axis for the ploting

% ================================================ %

% Ambient temperature estimated based on the FBGS measurement
yDeltaT = P(2); % Shift ratio due to temperature at F = 0
tempEstim = yDeltaT/(alphaf+xi)

% ================================================ %

% Strain estimation based on the temperature
```

```
strainComp = zeros(4,2); % Allocate space for matrix
strainEstim = zeros(4,1); % Allocate space for matrix
strainComp(:,1) = strainFEM; % Fill up first column w. strain from FEA

for i=1:size(strainComp)

    strainEstim(i) = (deltaLambdaB(i)/(lambdaB)-yDeltaT)/((1-peff));
    strainComp(i,2) = strainEstim(i);

end

strainComp % Comparison between FEA strain and estimated strain
```

# Appendix E

# Redundant Sensing method

```
% ================================================== %

%                    REDUNDANT SENSING METHOD

% ================================================== %

%   This code uses the shift ratio yielding from the wavelength shift
%   outputted by a certain amount of FBG sensors, and combine them with
%   the estimated temperature of the sample and the known constitutive
%   equations of engineering to compute the magnitude of loading.

% ================================================== %

clc, clf

% ================================================== %
%                         INPUTS
% ================================================== %

% Import cursor info (location of FBG sensors) to avoid manual selection
load('cursor_info');

% FBG properties
alphaf = 0.55e-06; % alphaf is the coefficient of thermal expansion [K^-1]
xi = 6.5e-06; % xi is the thermo-optical coefficient [K^-1]
peff = 0.22; % Effective photoelastic coefficient
Cstrain = 1-peff;
Ctemp = alphaf+xi;

% Generating sensor output (theoretical) - to be replaced by output of
% the FBG sensor in real life.

% Initial wavelength - output of sensor at strain = 0 and temperature change = 0
lambdaB = 1550*1e-6;
```

```
% Temperature change of the sample
initialTemp = 0;
finalTemp = 85;
deltaT = finalTemp - initialTemp;

% Strain computation at location of each sensor for F = 1 and F = x N
[strain, strain_F1, POI_info] = strain_computation_element_scale...
    (cursor_info, model, d);

% ==================================================== %
%                COMPUTATION OF THE SHIFTING RATIO
% ==================================================== %

% Choose the direction [x, y, z] of the measure
dir_x = 1; % Coordinate of x direction in strain matrix
dir_y = 2; % Coordinate of y direction in strain matrix
dir_z = 3; % Coordinate of z direction in strain matrix

strain_dir = dir_x ; % Choose the direction of the strain [x, y, z]

% Computation of the shifting ratio at F = 1 N
for j = 1:numel(cursor_info)

    yf1 = calc_wavelength_shift(strain_F1(strain_dir,j), deltaT, ...
        Cstrain, Ctemp, lambdaB);
    y_POI_F1(j,1) = yf1;

end

% Computation of the shifting ratio at F = ? N (searched value)
for j = 1:numel(cursor_info)

    yf = calc_wavelength_shift(strain(strain_dir,j), deltaT, ...
        Cstrain, Ctemp, lambdaB);
    y_POI(j,1) = yf;

end

% ==================================================== %
%                   TEMPERATURE ESTIMATION
% ==================================================== %

% The strain1 at the POI1 is compared with the strain2 at the POI2.
% Strain2 increases proportionally to strain1, if the material stays in
% elastic region. A system of equation is solved to get the temperature.

% The factor a between strain 1 and 2 is computed
```

```
a = strain_F1(1,2)/strain_F1(1,1);

% Temperature estimated based on factor a and output of the two sensors
% deltaTEstim = (y_POI_F1(2)-Cstrain*strain_F1(1,1)*a)/(Ctemp);
deltaTEstim = (y_POI_F1(2)-a*y_POI_F1(1))/(Ctemp-a*Ctemp);

% ========================================================== %
%                        FORCE ESTIMATION
% ========================================================== %

% Temperature error
temp_error = deltaT/deltaTEstim-1;

% Strain measured at POIs, based on the shift ratios, FBG cst. and deltaT
strain_measured1 = (y_POI(1)-Ctemp*deltaTEstim)/Cstrain;
strain_measured2 = (y_POI(2)-Ctemp*deltaTEstim)/Cstrain;
strain_measured3 = (y_POI(3)-Ctemp*deltaTEstim)/Cstrain;

% Strain measurement compared to theoretical strain
strain_error1 = strain(1,1)-strain_measured1;
strain_error2 = strain(1,2)-strain_measured2;
strain_error3 = strain(1,3)-strain_measured3;

% Force estimation based on comparison between measured strain at POIs
% and F = 1 N strain at POIs.
F_factor1 = strain_F1(1,1)/strain_measured1;
F_factor2 = strain_F1(1,2)/strain_measured2;
F_factor3 = strain_F1(1,3)/strain_measured3;

% Strain measured and strain error stored in matrices
strain_measured = [strain_measured1; strain_measured2; strain_measured3]
strain_error = [strain_error1; strain_error2; strain_error3]

% Average of the measured forces
force_measured = 3/(F_factor1 + F_factor2 + F_factor3);

% Result
Message = ...
    'The force applied is of +/- ' + string(force_measured) + ...
    ' N and ' + 'the temperature delta measured is of ' + ...
    string(deltaTEstim)'

% ========================================================== %

clear j yf1 yf strain_measured1 strain_measured2 strain_measured3 ...
    strain_error1 strain_error2 strain_error3 dir_x dir_y dir_z ...
    strain_dir F_factor1 F_factor2 F_factor3 Message lambdaB ...
    deltaTEstim Cstrain Ctemp xi peff alphaf
```