Demodulation of Fiber-Optic Chirped Fiber Bragg Grating Sensors for Thermal Pattern Detection

Yerzhan Orazayev, B. Eng

Submitted in fulfilment of the requirements for the degree of Master of Science in Electrical and Computer Engineering

School of Engineering

Department of Electrical and Computer Engineering

Nazarbayev University

53 Kabanbay Batyr Avenue,

Astana, Kazakhstan, 010000

Supervisors: Daniele Tosi, Ikaechi Ukaegbu

December 13

DECLARATION

I hereby, declare that this manuscript, entitled "*Demodulation of Fiber-Optic Chirped Fiber Bragg Grating Sensors for Thermal Pattern Detection",* is the result of my own work except for quotations and citations which have been duly acknowledged.

I also declare that, to the best of my knowledge and belief, it has not been previously or concurrently submitted, in whole or in part, for any other degree or diploma at Nazarbayev University or any other national or international institution.

Name: Yerzhan Orazayev

Date: 2018/12/13

Abstract

This report focuses on development of demodulation software for ultra-dense distributed chirped fiber Bragg grating optical sensors, which measure temperature and strain with 75 micron resolution over 1.5 cm for advanced medical applications. Initially, addressed problem is stated, thermal ablation technology is discussed, and basic background of optics is considered. Next, old and new reconstruction algorithms are reviewed in detail. It follows with discussion of MATLAB code to LabVIEW environment transition. Real spectrum measurements, which are used for verification of proposed reconstruction, were obtained during experiments conducted under supervision of Professor Daniele Tosi. The obtained data is used as software inputs to obtain thermal distribution. Moreover, significance and opportunities of temperature prediction for the proposed technology is discussed. Finally, the results of prediction based on linear regression model are analyzed and further improvements of technology are proposed.

Acknowledgments

Firstly, I would like to express my uttermost gratitude towards my supervisor Daniele Tosi. It has been his supervision and direction throughout the duration of my studies which has allowed me to successfully complete this Master's program. I am appreciative for all hours of discussion he has offered me, especially in the areas of optical communications, and signal processing.

Secondly, I would like to thank my family for faith and love through the whole education in Nazarbayev University. Although being in different city, I always knew that you always supported me no matter where I am.

Thirdly, I would like to express acknowledgements to my friends Sanzhar Korganbayev, Sultan Sovetov, Arman Aitkulov, and Bekarys Kuspan. Their encouragements during studies have helped me to withstand the pressure of all assignments and exams.

Table of Contents

List of Abbreviations

FBG – fiber Bragg grating

CFBG – chirped fiber Bragg grating

RFA – radio frequency ablation

MWA – microwave ablation

LA – laser ablation

HIFU – high frequency focused ultrasound

TI – thermal imaging

TC – thermocouples

FOS – fiber optical sensors

MR – magnetic resonance

CT – Computer tomographic

SLED – superluminescent light emitting diode

CMT – coupled mode theory

TM – transmission matrix

MSE – mean squared error

TA – thermal ablation

OBR – optical backscatter reflectrometry

SMF – single mode fiber

List of Publications

S. Korganbayev, Y. Orazayev, S. Sovetov, A. Bazyl, E. Schena, C. Massaroni, R. Gassino, A. Vallan, G. Perrone, P. Saccomandi, M. Arturo Caponero, G. Palumbo, S. Campopiano, A. Iadicicco and D. Tosi, "Detection of thermal gradients through fiber-optic Chirped Fiber Bragg Grating (CFBG): Medical thermal ablation scenario", Optical Fiber Technology, vol. 41, pp. 48-55, 2018.

S. Korganbayev, Y. Orazayev, S. Sovetov, A. Bazyl, D. Tosi, E. Schena, C. Massaroni, R. Gassino, A. Vallan, G. Perrone. “Thermal Gradient Estimation with Fiber-Optic Chirped FBG Sensors: Experiments in Biomedical Applications”, 2017 IEEE SENSORS, 2017.

Chapter 1 – Introduction & Literature Review

1.1 Introduction

Cancer is one of the ancient diseases that humanity still struggling. With science progress, many treatment procedures were suggested to encounter this lethal disease. Before, different primitive tools were used in surgery operations [6]. However, nowadays there are many different technologies for cancer tumor removal. One of the technologies is temperature ablation. The technology allows to selectively increase the temperature of a tissue by using electromagnetic energy [7-8]. The tumor cell can be subjected to a cellular damage if its temperature will be higher than 42-44 °C. If the temperature will achieve the value of 60 °C, the tumor cell is destroyed almost instantly due to protein coagulation effect. There are several thermal ablation technologies that can be clinically used for this purpose: radiofrequency ablation (RFA) [7] [9], microwave ablation (MWA) [10-11], laser ablation (LA) [12], and high-intensity focused ultrasound (HIFU) [13].

Although TA is a good instrument for destruction of cancer cells, it can destroy living cells likewise. Therefore, a temperature sensor should monitor the process of thermal ablation [14]. There are two methods that were used for this

purpose in late 90's – thermal imaging [15] [17] and thermocouples [16]. For instance, in [15] the study shows implementation of radiofrequency (RF) ablation, and thermal imaging methods such as computed tomographic (CT) and magnetic resonance (MR) images is used for assessment of the proposed treatment. In [16] the study also considers RF ablation, but thermocouples or thermistors are used for temperature measurement.

Nowadays new types of sensors are rapidly acquiring recognition in biomedicine – fiber optical sensors (FOS). These sensors have numerous advantages with respect to thermocouples [23] [24] – small size, lightweight, compatible to magnetic resonance (MR) because of resistance to electromagnetic interferences, good sensitivity and quick response to physical/biological influence, and conformity to ISO 10993 standard for biocompatibility of medical devices. The optical sensors were already applied and commercialized for measurement of blood vessel obstructions [18], intra-aortic balloon pumping therapy [19], intra-cranial pressure [20], solid state manometers [21], and biomechanics [22]. Numerous researches over FOS were conducted. In [24] FBG array was used for temperature sensing in hyperthermia treatment. In [23] and [25-26] the system was applied in RFA, MWA, and LA. In [27] the application of OBR is examined. The first reported investigation of CFBG was in [28]. In [29] CFBG optical sensing method was improved with development of a spectral reconstruction method.

This project concerns the development of demodulation technique that will assist in thermal treatment of cancer tumor using CFBG. Firstly, some background is discussed in Chapter 1. It will be followed by discussion of old demodulation technique. Next, new demodulation technique will be proposed in Chapter 2. Also, the developed LabVIEW software will be introduced in the chapter. In Chapter 3, some experimental setup and its results will be reviewed, and in Chapter 4 simulations of the software based on these results will be observed. Finally, some conclusions and future works will be discussed in Chapter 6.

1.2 Optics Background

1.2.1 CFBG model

Fiber Bragg Grating are reflective arrangement located in the core of an optical fiber. It has two index modulation profiles: periodic and quasi-periodic. The grating can be obtained by exposing the core of optical fiber to some pattern of ultraviolet light radiation. Different arrangements of grating are presented in the following figure:

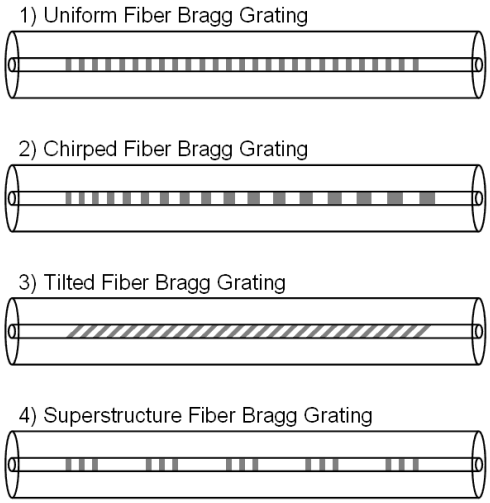


Figure 1.1. Variations of index profile [30].

Before explaining the working principles of CFBG, it is essential to consider the basics of FBG performance. Consider a broadband light injected into a fiber from a light source through fiber optic coupler. According to the Zeng and et al, the light will be reflected by a grating of the fiber at a specific wavelength, as long as Bragg condition is conserved. It states that the frequencies of incident and reflected light must be identical [32]. Consequently, all reflected light from each grating of the fiber assembles into an impulse of reflection with center wavelength λ_B, also called Bragg wavelength, which is specific to the grating parameters. The parameters can be analyzed by capturing the reflected light using spectrometer. The wavelength can be evaluated by the following equation:

$$\lambda_B = 2n_{eff} \cdot \Lambda \tag{1}$$

where n_{eff} is the effective refractive index and Λ is the period of grating.

Figure 1.2 and 1.3 clearly illustrates FBG performance. In case of inconsistency of the condition, the phases of reflected lights will not match and they will eliminate one another.

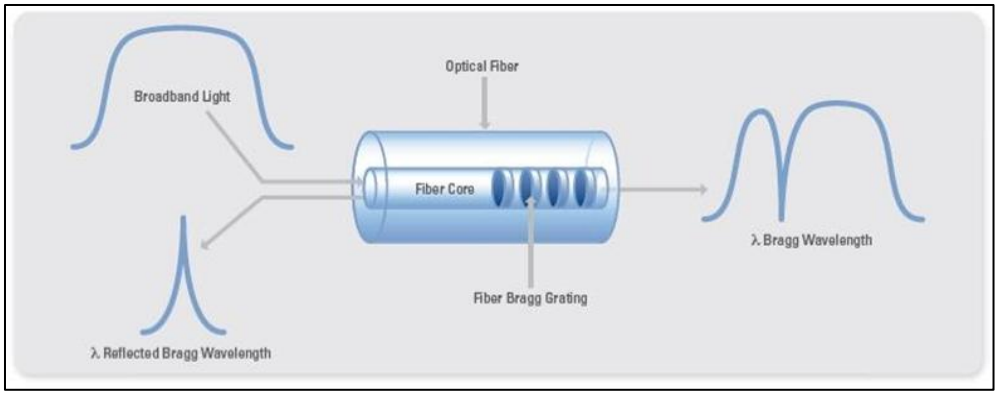


Figure 1.2. Reflection of Bragg wavelength for uniform profile [31].

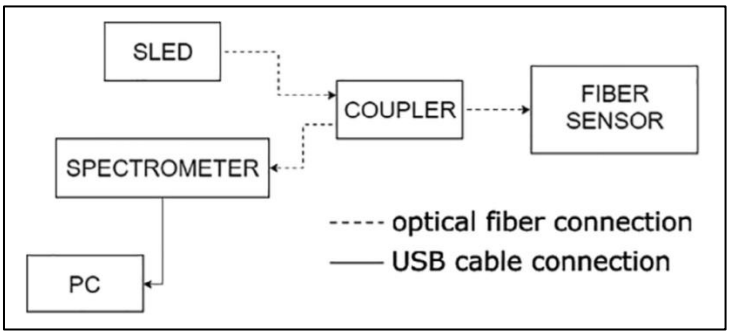


Figure 1.3. Light transmission and reflection diagram [4].

As most of other sensors, the change of external environment (temperature, strain, etc.) affects the parameters of the fiber (effective refractive index and grating period). Further, the changes of the parameters lead to the changes of reflectivity spectrum. The main principles of proposed sensing technique are analysis of reflected spectrum, application of designed demodulation algorithm for

obtaining grating parameters, and consequently discovering environmental affecting factors. For this project, the affecting factor is temperature change.

This report discusses Chirped Fiber Bragg Grating (CFBG), specifically its linear subtype, which has algebraic progression of increase in the spacing between grating planes (Figure 2.3). As a result, equation for Bragg wavelength (1) modifies to:

$$\lambda_B(z) = 2n_{eff}(z) * \Lambda(z) \tag{2}$$

where for linearly chirped

$$\Lambda(z) = \Lambda(0) + \Delta n(z) \tag{3}$$

where $\Lambda(0)$ is a starting period, K is the slope for period along the fiber.

As a result, for chirped grating of spectrum of different Bragg wavelengths, which are related to different grating periods, is reflected from gratings. Thus, distributed sensing is possible for chirped profile, which is considerable advantage of CFBG sensors.

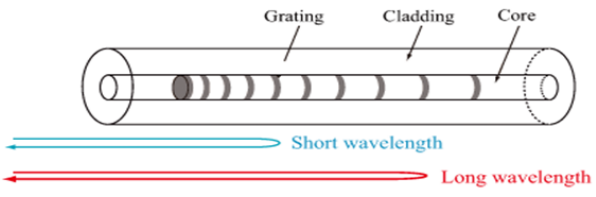


Figure 1.4. Chirped reflection index change profile [4].

Chapter 2. Simulation methodology.

To begin with, algorithm of the demodulation should be clarified. In this chapter two decoding techniques, i.e. old and new demodulation algorithms, and the development of LabVIEW software will be discussed.

2.1 First decoding technique.

2.1.1 Modulation of single FBG.

In order to review the whole algorithm, it is important to understand the modulation of single FBG. The proposed technique is designed for incoherent detection that implies assumption of the source (SLED) having a short coherency length. In other words, standing waves existing between each section of the grating are at minimum, so they can be neglected. This technique is alternative to its alternative more complex technique – Transmission matrix model [1], [33].

This method includes phase relationships in the detection, making it coherent. According to Harris, coherent detection cannot be more precise in detection than incoherent detection, owing to the fact that the phase detection in coherent results in measurement uncertainty [34].

It can be stated that the proposed technique of incoherent detection is adaptation of the transmission matrix model. Eventually, the modulation is achieved by using previously discussed equations (1-3):

$$R_i(\lambda) = \frac{\sinh^2\left(L_g\sqrt{k^2 - \sigma_i^2}\right)}{\cosh^2\left(L_g\sqrt{k^2 - \sigma_i^2}\right) - \frac{\sigma_i^2}{k^2}} \quad (4)$$

which evaluates the reflectivity on a certain wavelength. L_g and k are characteristic variables of a fiber that denotes the length of grating and AC coupling coefficient respectively. σ is DC coupling coefficient and can be evaluated from the equation (5).

$$\sigma_i(\lambda) = \frac{\pi}{\lambda}\delta n_{eff} + 2\pi n_{eff}\left(\frac{1}{\lambda} - \frac{1}{\lambda_{B,i}}\right) \quad (5)$$

It also depends on the Bragg wavelength, amplitude of refractive index modulation (δn_{eff}), effective refractive index (n_{eff}), and Bragg wavelength ($\lambda_{B,i}$). By using MATLAB, the equations were used to show the refraction of single FBG.

2.1.2 Modulation of CFBG.

Since modulation of single FBG was discussed, it is time to concern the array of FBGs. In this work, CFBG is considered as the array. In order to evaluate the reflected spectrum of CFBG transfer matrix method is used. It is important to note that the distance between each FBG is L_g, and the number of FBGs in the array is M. The amplitude (Λ) and Bragg wavelength (λ_B) varies for each FBG. Total reflection spectrum is evaluated from the estimation of separate reflection spectrums and conversion to transmission spectrums $(1 - R_i(\lambda))$ of i-th FBG,

multiplication of obtained spectrums and conversion back to the reflection spectrum. The final equation is:

$$R_{CFBG}(\lambda) = 1 - \prod_{i=1}^{M}[1 - R_i(\lambda)] \tag{6}$$

2.1.3 Demodulation of CFBG spectrum.

Next step after modulation of reflection spectrum is demodulation of it to the temperature profile. The algorithm that was developed for this purpose is applicable for the three types of temperature profiles: linear profile with two parameters – gradient and initial temperature value; Gaussian profile with three parameters – amplitude, variance, and central position; and super-Gaussian profile that has the same parameters of Gaussian profile and also super-Gaussian power factor. Since the thermal distribution on a living tissue of living organs is assumed to have Gaussian distribution, it was mainly considered in the work. In order to achieve the thermal map from the reflection spectrum, the following Gaussian equation was used:

$$\Delta T(z) = A \cdot \exp\left(\frac{-(z - z_0)^2}{2s^2}\right) \tag{7}$$

where is A is the amplitude (℃), z is the position (m), z_0 is the central position (m), s is the variance (m). For the case of super-Gaussian profile, the equation is:

$$\Delta T(z) = A \cdot \exp\left(\frac{-(z - z_0)^{2P}}{2s_0^{2P}}\right) \tag{8}$$

where s_0 is:

$$s_0 = \frac{s}{(0.5 \cdot \pi)^{\frac{2}{2P-1}}} \tag{9}$$

where is P is the super-Gaussian power factor.

Table 2.1. Variables of CFBG modulation process.

Symbol	Name	Unit	Value range
$\boldsymbol{M}$	Number of gratings (discretization)	-	1-500
$\boldsymbol{\psi}$	Chirp rate	nm/cm	-
$\boldsymbol{\delta n_{eff}}$	Amplitude of refractive index modulation	-	10^{-6}
$\boldsymbol{n_{eff}}$	Effective refractive index	-	1.5
$\boldsymbol{\xi}$	Thermal sensitivity coefficient	$pm/℃$	10.2
$\boldsymbol{L}$	Length of CFBG	m	0.015-0.05
$\boldsymbol{L_g}$	Length of single grating (L/M)	m	–
$\boldsymbol{\Lambda}$	Grating period	nm	-
$\boldsymbol{kL_g}$	Grating strength	-	0-0.9
$\boldsymbol{k}$	AC coupling coefficient	$1/nm$	-
$\boldsymbol{\sigma}$	DC coupling coefficient	$1/nm$	-
$\boldsymbol{N}$	Number of spectrum measurement	-	-
$\boldsymbol{\lambda_B}$	Bragg wavelength	nm	1520-1560

$\boldsymbol{\Delta T}$	Temperature	℃	-
$\boldsymbol{A_g}$	Amplitude guess	℃	-
$\boldsymbol{s_g}$	Variance guess	M^2	-
$\boldsymbol{\mu_g}$	Mean guess	m	-
$\boldsymbol{z}$	Distance axis	m	0-L
$\boldsymbol{R_{meas}}$	Measured reflected spectrum	-	0-1
$\boldsymbol{R_{calc}}$	Calculated reflected spectrum	-	0-1
$\boldsymbol{H}$	Equalizer function	-	-

The initialization part of the figure 2.3 is performed with no ΔT gradient along the CFBG, i.e. in reference condition. The goal is to obtain the model of the CFBG as in Eq. (1-6) based on coupled mode theory (CMT) [1], and therefore to populate all the grating parameters outlined in Table 2.1. From the manufacturer, the grating length (L) as well as the optical parameters (δn_{eff}, n_{eff}, ξ, L, $\lambda_{B,0,1}$) can be obtained. In detail, the initial and final wavelengths, i.e. $\lambda_{B,0,1}$ and are obtained through spectral observation to obtain an initial guess, and running an optimization based on the CMT theory until there is the best match between the CFBG spectra on the left and right side. The thermal sensitivity coefficient ξ, on the other hand, is set to 10.2 pm/°C as the sensor is calibrated as in [3].

The discretization factor M is chosen according to the selection of the reflectivity shape. In other words, large value of the discretization factor leads to rounding of the reflectivity figure, which leads to the difference from the actual reflectivity spectrum as shown in the figure 3.

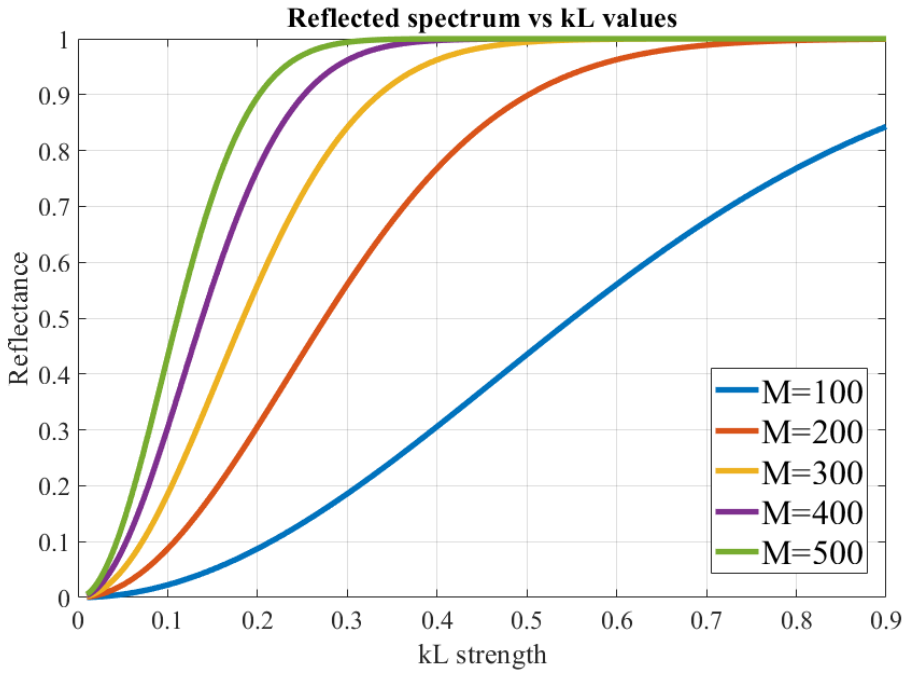


Figure 2.2. Representation of dependency of reflection spectrum on gratings number (M) and grating strength (kL_g).

An essential part of the algorithm is equalizer $H(\lambda)$, which is obtained from the division of measured spectrum by simulated spectrum and afterwards applied to the simulated CFBG in order to equalize its spectrum. This is necessary as the CMT returns an absolute value for the reflectivity of the CFBG, while the measured CFBG has amplitude that depends on the gain and exposure time of the

detector and is quantized. In addition, the CFBG spectrum may have spectral ripples which are not accounted in the CMT-model. Thus, after the CFBG model is generated and the spectrum is simulated, the CFBG spectrum is multiplied by the equalizer.

The grating strength kL_g determines the overall CFBG reflectivity, which needs to match the measured peak reflectivity. Fig. 2.2 serves as a calibration for the CMT-based CFBG model based on the discrete set of gratings. Calibration is done only with the first spectrum (at the beginning of the measurement process) to define M and kL_g.

Some values of our CFBG model parameters (number of modeled FBGs M, and their grating strength kL_g) can be obtained from developed code, illustrated in Fig. 2.2, that defines relation between M, kL_g and maximum reflectivity of simulated CFBG. Thus, maximum value of spectrum measured by spectrometer provides possible M and kL_g values for our CFBG model. Then, as it was mentioned about the optimization of wavelength range, CFBG spectrum is modulated using Eq. (1-6) and CFBG parameters with approximate wavelength range, defined from fist spectrum measurement. After, modulated CFBG wavelength range margins are optimized by iteration method to minimize difference between measured and modulated CFBG spectra.

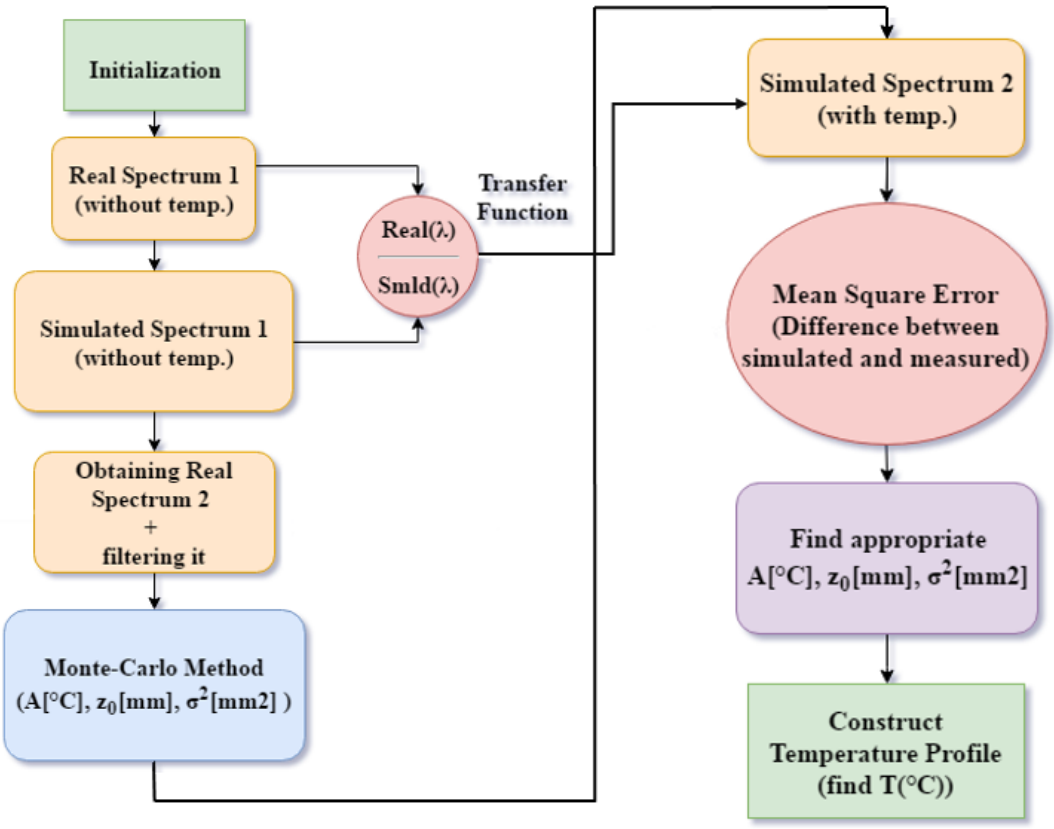


Figure 2.3. Flow chart of the first decoding algorithm [4].

2.1.5 Problems of the first decoding technique

The aim of the project is development of computer software that will process and measure temperature concurrently to the medical surgery process. This implies the smallest processing time. The processing period of the developed decoding technique is excessively long and requires an alternative solution for increasing the speed of processing.

2.2 Second decoding technique

2.2.1 Pre-processing functions

To decrease the complexity and increase the rate of demodulation, the algorithm was divided into two main parts. The first part of demodulation process is considered on the estimations of starting and ending wavelength values of reflection spectrum [15], evaluation of transfer function for aligning the top of reflection spectrum for compliance with the characteristics of real fiber reflection [4], and the simulation of reflection spectrum using combination of the gradient parameters – amplitude, center, variance, and super-Gaussian power. Initially, the spectrums are simulated by using a combination of the parameters in the equations [2-7]. Then, obtained reflection models are saved with the corresponding gradient parameters in the form of matrix in separate database. In order to distinguish each matrix, they are named according to the amplitude variations. The figure 2.4 briefly demonstrates first part algorithm.

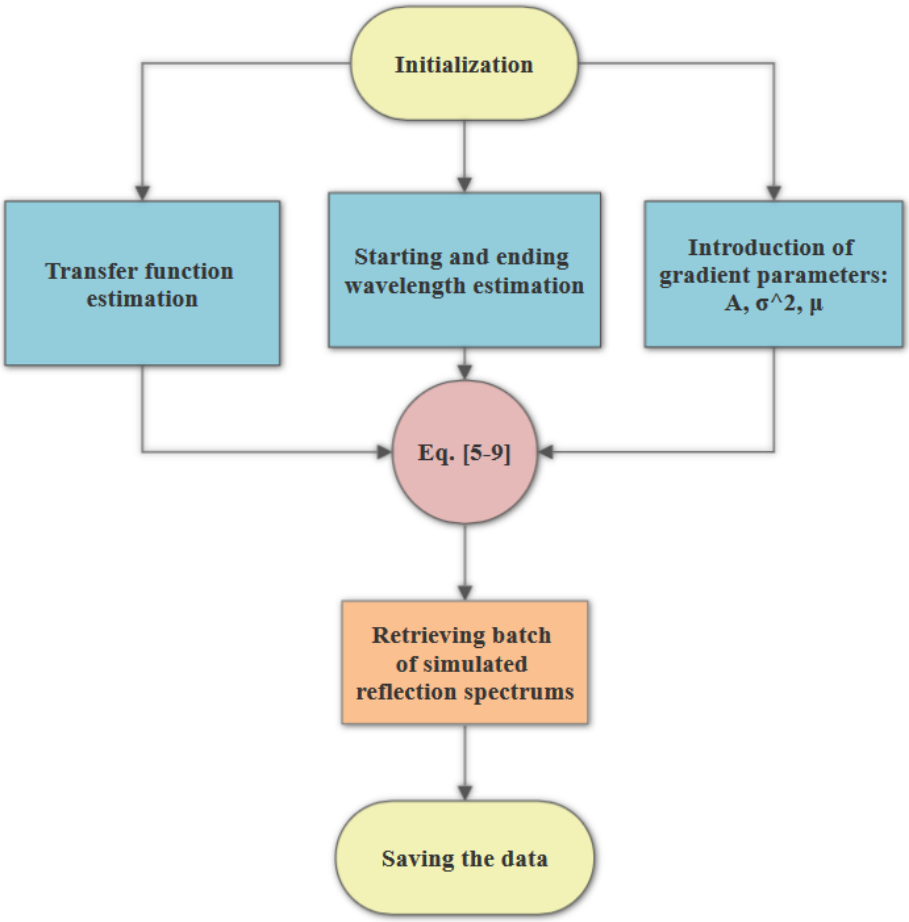


Figure 2.4. First part algorithm - preliminary simulations of reflection spectrums.

2.2.2 Post-processing functions

The second part of the demodulation is introduced during the ablation process. The algorithm works in a loop mode with feedback element – temperature change value, which is also the amplitude value of thermal gradient. Initially, the algorithm obtains the change of the temperature as input and for the first cycle it is assumed to be 0. According to the temperature change, it extracts corresponding matrix from database with simulated reflection models made in the previous part.

Then, all simulated models are compared with the measured model by using Mean Squared Error (MSE). The most related model ascertains the required gradient parameters for reconstruction of temperature distribution, and the amplitude value is then passed as the input for the next temperature measurement of TA process. If the thermal ablation stops, the algorithm also stops. This algorithm allows visualizing the temperature change across the sensor through the process. The steps of the second part are illustrated in the figure 2.5.

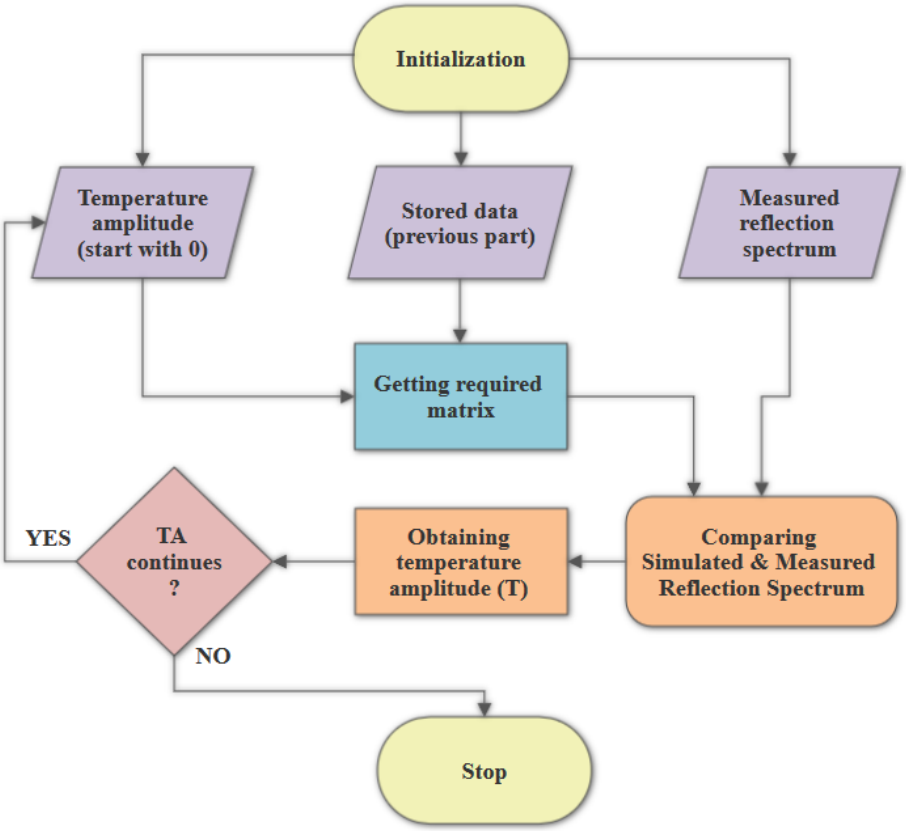


Figure 2.5. Second part algorithm – identification of gradient components during thermal ablation.

2.2.3 Problems of the second decoding technique

In order to improve the developed software, potential challenges must be acknowledged. For instance, storage of the simulated reflected spectrums obtained in the “pre-processing” part is a challenge for the future. It takes time to simulate and store each matrix. For a start, only amplitude was iterated for simulation of the reflection spectrum. However, other parameters such as variance, center position, and super-Gaussian power factor should be also iterated and considered in simulation of the reflection spectrums. Taking account of every possible scenario of distribution will greatly accuracy, but also will increase the pre-simulation time, thereby creating a challenge to deal with data loss or data corruption.

2.3 LabVIEW software development

2.3.1 Opportunities and challenges

In order to create software with convenient graphical user interface (GUI) that can read data from spectrometer, LabVIEW programming environment was used. According to [5] LabVIEW is more preferred for “acquiring, processing, and displaying signals” than MATLAB. On the other side, MATLAB is better in computational aspect. Therefore, the first part of the code that is responsible for preparation and estimation of reflection spectrum models remained in the MATLAB form. The second part that involves estimation of temperature concurrently to thermal ablation was ported from MATLAB to LabVIEW. Due to

the difference in the programming environments, porting process was challenging. For instance, an apparent difference is the matrix indexing in LabVIEW starts from 0, while MATLAB starts from 1. Another big challenge is inability of LabVIEW to use function on a matrix without applying loop. This increases the duration of processing.

2.3.2 Functions and representations

The program consists of one main code, and 5 functions that support it.

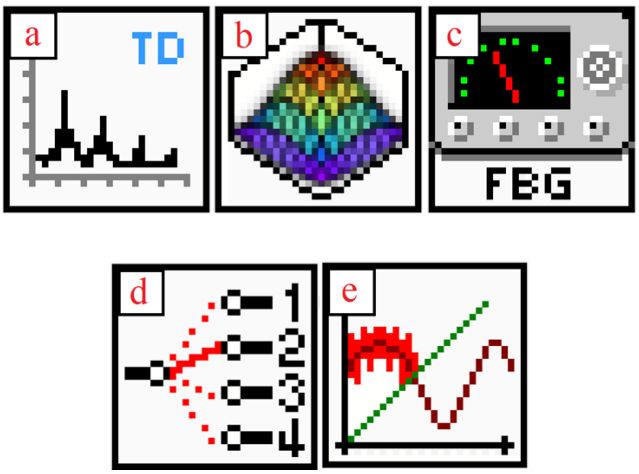


Figure 2.6. Icons of LabVIEW functions: a) Temperature distribution; b) Temperature reconstruction; c) FBG generation; d) Linspace; e) Offset removal;

First function is “Linspace”. Although it is present in MATLAB, there is no analogous function in LabVIEW. The function takes three numbers as inputs,

namely x1, x2, and n. It generates “n” points between “x1” and “x2” with spacing between points calculated by:

$$\Delta = \frac{x2 - x1}{n - 1}$$

Second function is called “Offset removal”. The function removes the DC offset noise presenting in the power reflectivity spectrum as a characteristic noise of the spectrometer. The spectrum contains 512 measurement points. The function takes initial 100 points with no reflections, evaluates mean of these points, and subtracts from all points.

Third function is called “FBG generation”. The function evaluates reflection spectrum by using characteristic variables of CFBG, and the equations (2-7) from the Chapter 2. One of the challenges of converting MATLAB code to LabVIEW schematic is square root function. Although LabVIEW has built-in root function, it gives not-a-number output for negative input. Therefore, built-in function that processes MATLAB code was used. The following figure illustrates the problem:

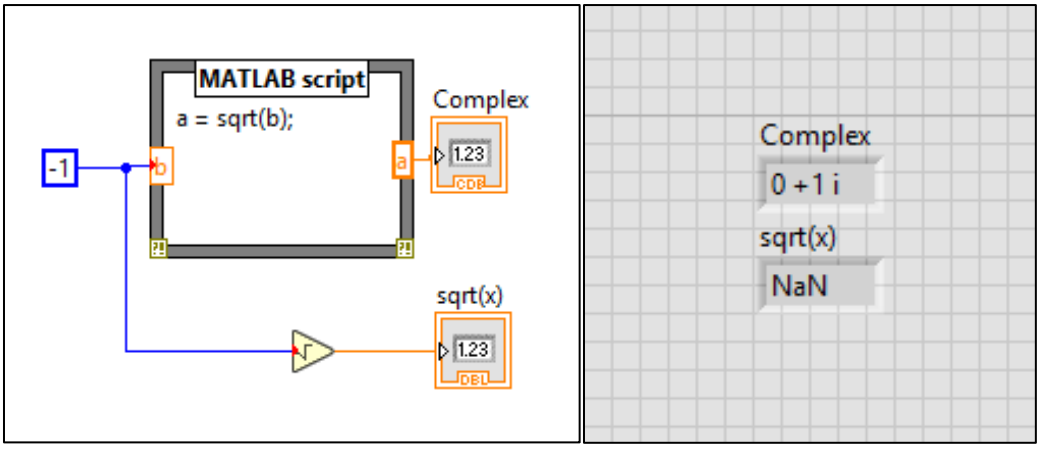


Figure 2.7. Built-in and MATLAB script square root function comparison.

Fourth function is “Temperature reconstruction”. This function is the core of post-processing part of decoding. The function obtains the change of the temperature as input. According to the temperature change, it extracts corresponding matrix from database of pre-defined “comma-separated values” (CSV) format with simulated reflection models made in the previous part. Then, all simulated models are compared with the measured model by using Mean Squared Error (MSE). The most related model ascertains the required gradient parameters for reconstruction of temperature distribution.

Last function is “Temperature distribution”. The function takes as input super-Gaussian power factor, number of gratings, amplitude, variance, and central position. The last three is the output of previous “Temperature reconstruction” function. Then, the equations (2-7) from the chapter 2 are implemented to build Gaussian distribution of temperature.

2.3.3 Final model of LabVIEW software

The main part of the software assembles all functions and performs the demodulation process. Initially, it reads the measurements from spectrometer. Spectrometer sends the measurement data with frequency of 10 Hz. In order to increase the speed of processing, the processed data size was reduced. Specifically saying, the software selects each 10th measurement to process. Then, selected data is passed through "offset removal" function to remove offset noise. Next, each time record of reflection model passed through loop. In loop, the data is passed through "temperature reconstruction" function to obtain characteristic values (amplitude, central position, and mean). The obtained values are penetrated through "temperature distribution" function to build Gaussian temperature profile.

Chapter 3 – Experiments and Simulations

Speaking about implementation of the proposed demodulation technique, experimental setup should be done. In this chapter, the components, interconnections of the setup, and results of the experiment will be illustrated and briefly discussed.

3.1 Experimental setup

In order to develop the algorithm and software, good input data are required. Moreover, some reference data also must be present to make the algorithm as precise as possible. For this purpose, real experiments should be conducted.

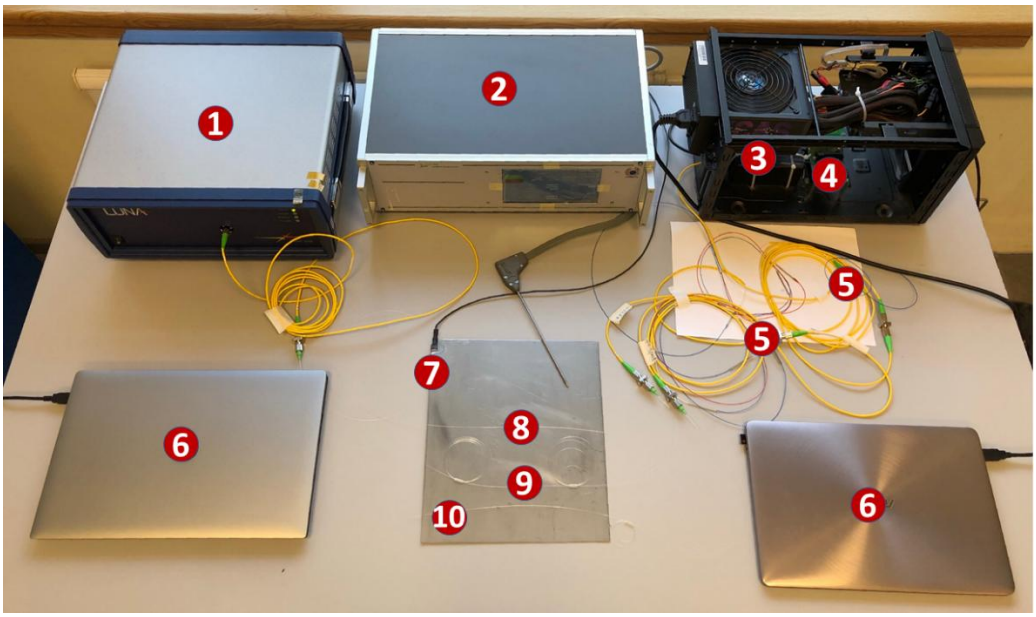


Figure 3.1. Photo of experimental setup: (1) Luna OBR 4600; (2) RF Generator; (3) Spectrometer; (4) SLED; (5) Coupler of 50/50 ratio; (6) PCs; (7) RF applicator; (8) SMF; (9) CFBG; (10) Chirped Fiber Bragg Grating.

The figure 3.1 shows setup photo of the experiments conducted to compare temperature sensing of FBG, CFBG, and optical backscatter reflectometry (OBR).

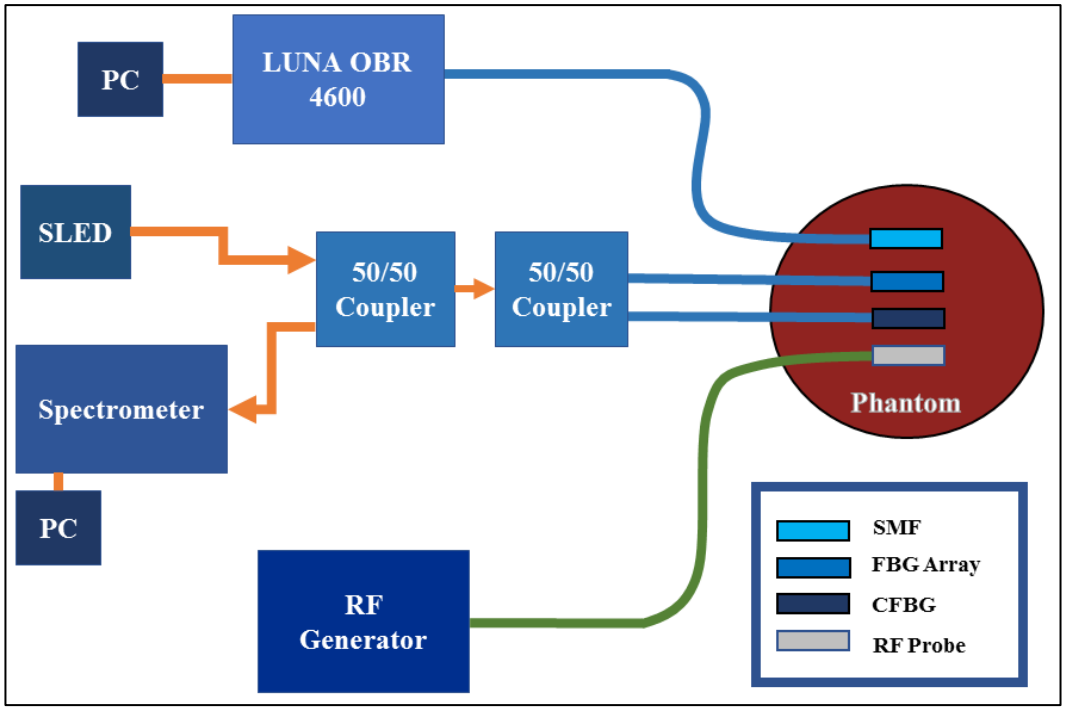


Figure 3.2. Schematic view for representation of interconnections of the setup.

Figure 3.2 illustrates schematic view of the setup in figure 3.1 to clarify interconnections. According to the figure, Single Mode Fiber (SMF) is a reference sensor of temperature change, and it is connected to LUNA OBR 4600, which is a reflectometry technology. LUNA OBR 4600 is connected to PC that reads and displays the data. Other two fibers are connected to Interferometer that consists of superluminescent LED source (EXS2100), and infrared spectrometer (I-MON-512 USB, Ibsen Photonics) that accepts the reflected light and transfer it to the connected PC. The last is RF probe placed in parallel to other fibers and connected

to RF generator, i.e. thermal ablation machine. Parallel placing of RF probe and fibers is necessary to achieve maximum likelihood in thermal distribution, and utilization of a real phantom makes the distribution close to reality.

3.2 Results of the experiment – reflection models

The aim of the experiment was comparison and evaluation of CFBG and FBG array. The measurements obtained from the LUNA OBR 4600 were considered as a reference due to relative reliability of the existing technology. The following figure 3.3 illustrates the results of the comparison.

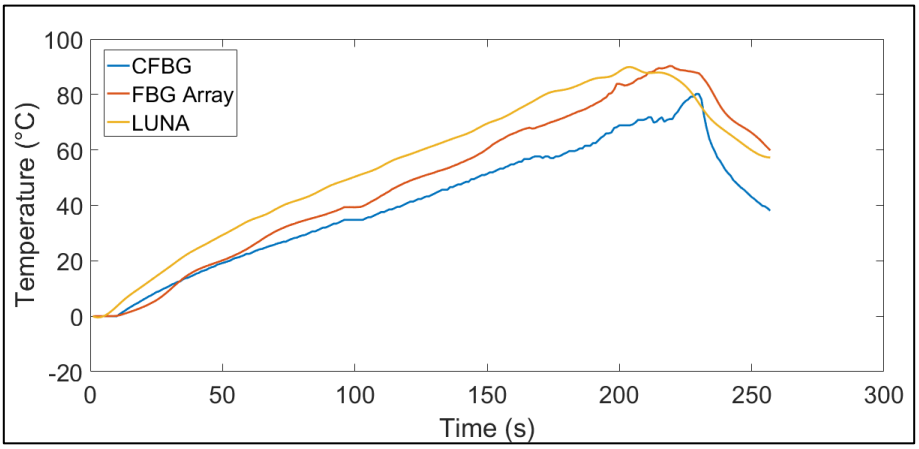


Figure 3.3. Evaluation of CFBG and FBG array performance with reference to LUNA.

From the obtained results, the main data is measurements obtained by CFBG sensor, since this is the main focus of the given project. Therefore, the data from the spectrometer containing required measurements were retrieved. The data is represented in the figures 3.4 and 3.5.

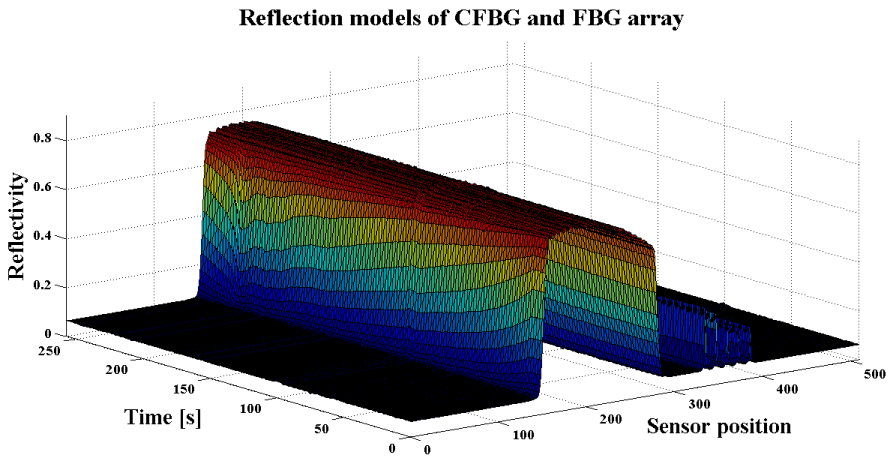


Figure 3.3. 3D representation of the obtained reflectivity results.

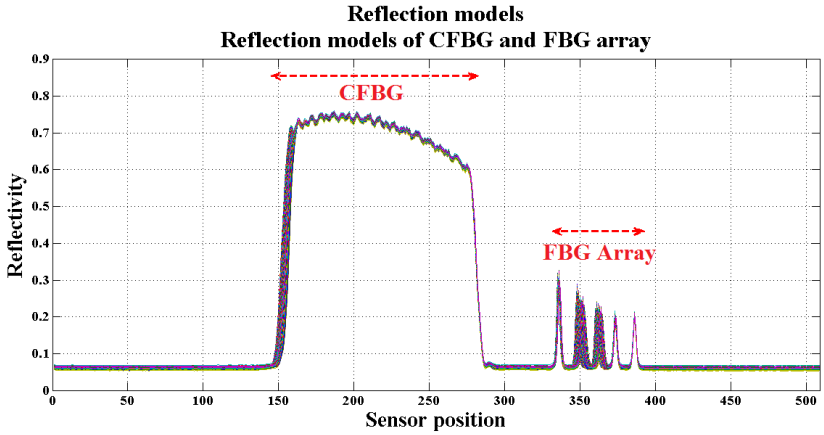


Figure 3.4. Reflection models of CFBG and FBG array.

Chapter 4 – Simulation results & discussion

After performing experiments and extracting data, simulations can be conducted to test the performance of demodulation algorithms. This chapter briefly discusses the results of the second demodulation algorithm based on LabVIEW, and illustrates the coincidence to the results of MATLAB demodulation algorithm.

4.1 Analysis of demodulation results

The results of MATLAB and LabVIEW post-processing demodulation is illustrated in the figure 4.1 and 4.2 respectively. The figures represents the ablation process, specifically saying the temperature distribution over the phantom liver used in the experiment. From the figure 4.3, the slow start of heating can be observed. The central point of the laser treatment was around 1 mm from a sensor's side. At the time $t \geq 170\ s$, the temperature reached 60 °C and continued to grow. The maximum temperature reached was slightly above 80 °C. Eventually, the temperature trend sharply dropped down due to the shut down of the laser.

From the obtained results it can be concluded that LabVIEW software demodulates in the same manner as MATLAB. However, the speed of processing

of LabVIEW is approximately 4 times slower than of MATLAB. It is a challenge for the future work.

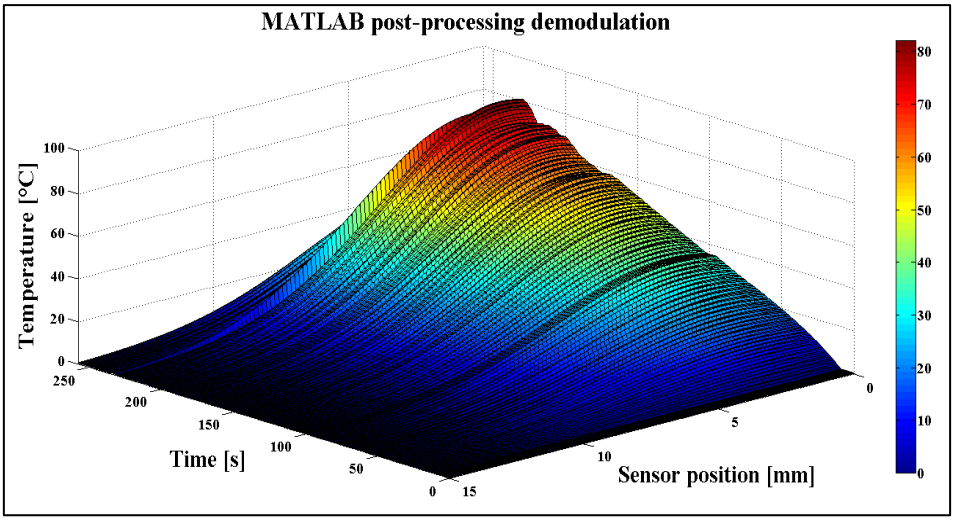


Figure 4.1. Temperature profile demodulated using MATLAB.

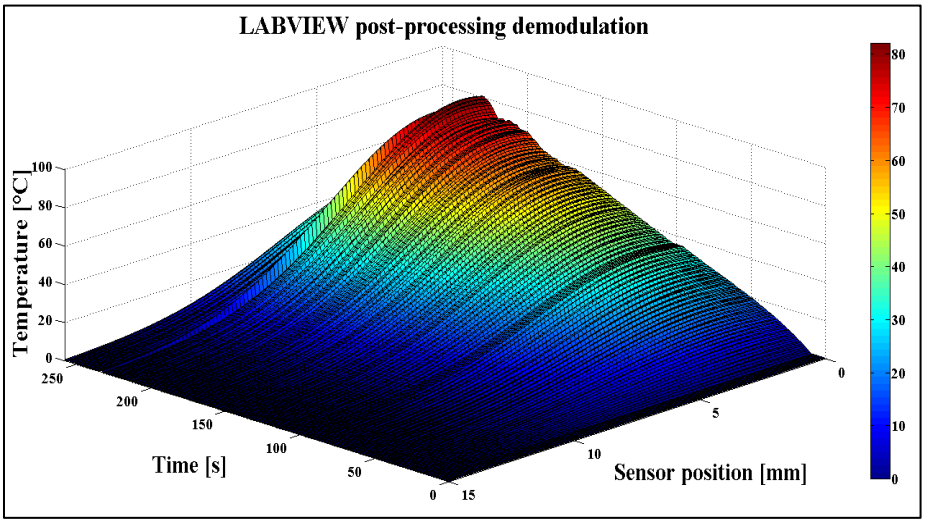


Figure 4.2. Temperature profile demodulated using LabVIEW.

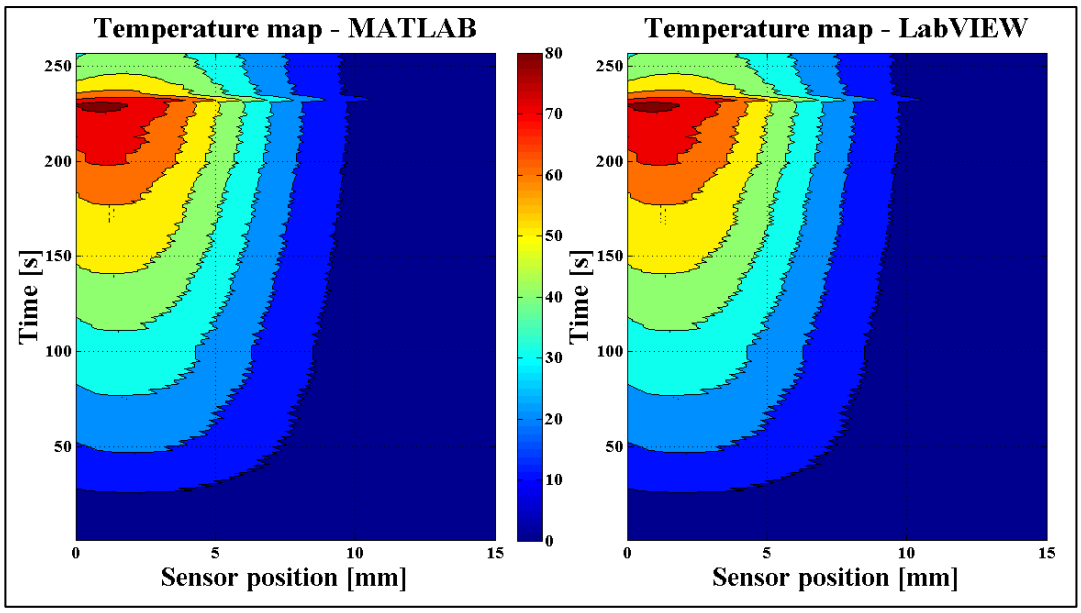


Figure 4.3. Comparison of thermal maps.

Chapter 5 – Temperature prediction using regression models

The results of the demodulation algorithm can be further implemented in the optimization of thermal pattern detection. In this chapter, linear regression model as a prediction algorithm is examined. Firstly, preliminary data processing is discussed. Next, MATLAB algorithm used in thermal distribution forecasting is introduced. Finally, results of the prediction algorithm are analyzed.

5.1 Prediction model

The prediction model of temperature starts with obtaining and reading data from spectrometer. Since the data below 60°C does not affect the cells, the prediction algorithm awaits the temperature record of equal or above 60°C, while the temperature measurements are stored. As soon as CFBG sensor senses needed temperature value, the algorithm is initiated. It processes all stored measurements data in the following manner:

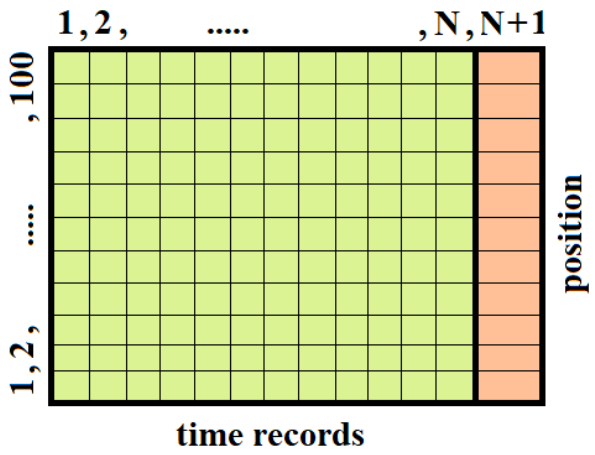


Figure 5.1. Preprocessing data for time-series prediction.

where N is the number of time records, i.e. one record per second or can be adjusted manually. Position represents gratings, which is for the case is 100 and can be also modified by the need. The temperature data from 1^{st} to N^{th} time records are used for prediction of $(N+1)^{th}$ temperature distribution. Although, the temperature at time of (N+2) is present, it is assumed to be unknown.

Next step the algorithm creates two datasets for prediction: train and test data. Both of these data must have identical dimensions - train data is data from 1^{st} to $(N + 1)^{th}$, and test data is from 2^{nd} to $(N+2)^{th}$ data. They are fed to the function, which is obtained from the built-in application “Regression learner” of MATLAB that automates the prediction. The obtained function produces a prediction function based on the training data, which in turn forecasts the value at (N + 2).

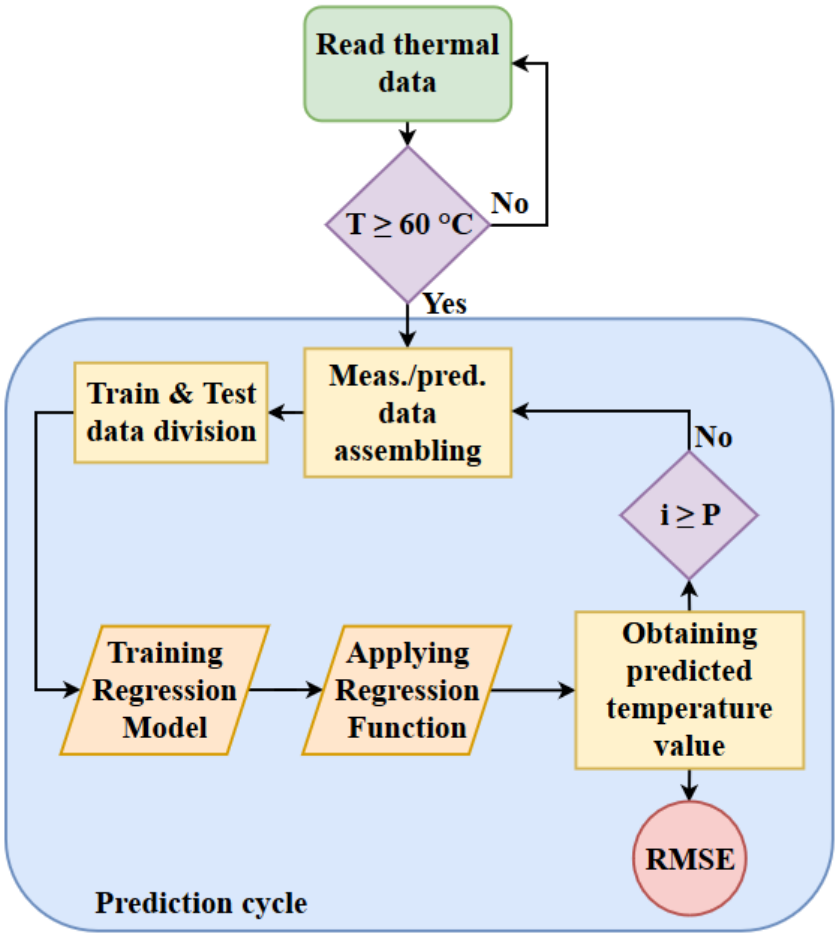


Figure 5.2. P - temperature step prediction algorithm.

The quality of the predictor can be observed by using RMSE:

$$RMSE = \sqrt{\frac{1}{N}(T_{pred} - T_{meas})^2} \tag{10}$$

Despite the fact that the algorithm can be used for any number of predictions (P), for this project 3 step prediction was examined. Larger step of prediction is assumed to be unnecessary. For that reason, the first predicted value is

concatenated to the 1-(N+1) measured data for further prediction as long as three predictions are made. Figure 5.2 illustrates the discussed algorithm in the form of flow chart, where T is temperature in Celsius, i is the prediction iteration, and P is the number of total prediction steps.

5.2 Analysis of obtained results

For predictions, thermal profiles obtained by LabVIEW demodulation software in chapter 3 were used. Three step prediction algorithm has produced three temperature distributions that are compared to the measured data. The figures 5.3 and 5.4 show prediction errors and comparison of predicted and measured distributions respectively.

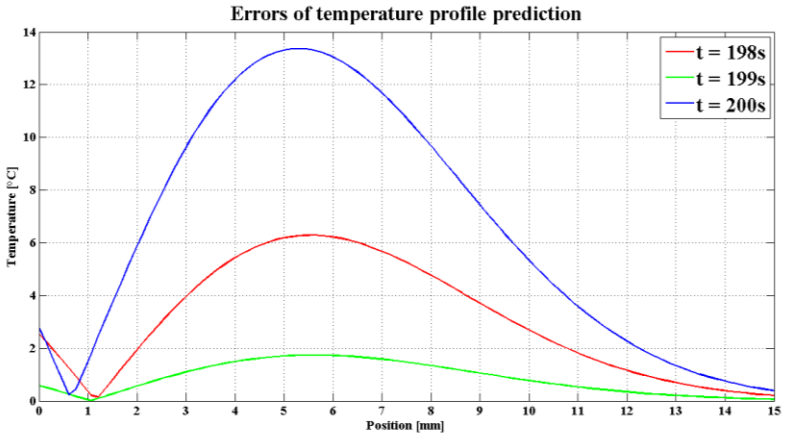


Figure 5.3. Errors of three step temperature prediction.

It can be seen that the last prediction based on the measured data and two predicted data produced maximum difference of approximately 13°C at 4.5-6 mm

sensor position. On the other side, relatively low error of 1.7°C and 6°C were achieved in the second and first predictions at the same sensor position accordingly. Furthermore, the figure 5.4 illustrates that the algorithm constructs relatively correct shape of temperature distribution. Nonetheless, the most important position of the sensor is the point of ablation. According to the thermal map illustrated in the previous chapter in the figure, the heat was concentrated around 0.5 – 1.5 mm sensor position. In other words, this range of sensor position needs accurate and precise prediction.

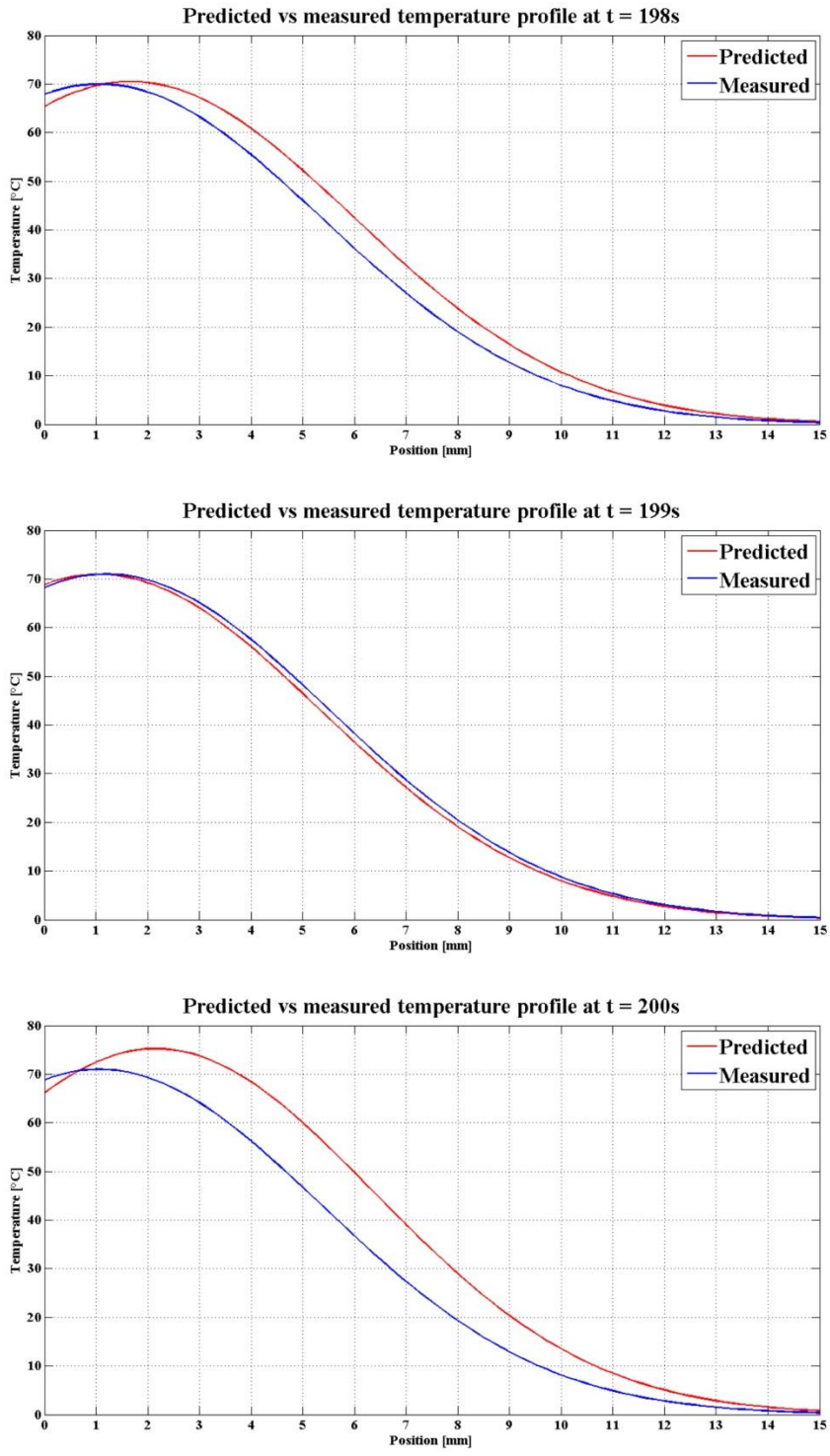


Figure 5.4. Temperature profiles comparison for 3 step prediction.

Another important aspect of prediction is forecasting the way the temperature distributes. For this purpose, the radius of distribution must be compared and analyzed. The figures 5.5 and 5.6 show the changes of predicted and measured temperature profiles, and the figure 5.7 shows the sequence of decrement followed by increment of temperature distribution. According to these figures, it can be noticed that the sequence was predicted correctly, but the amplitude and the radius of thermal distribution were predicted with errors.

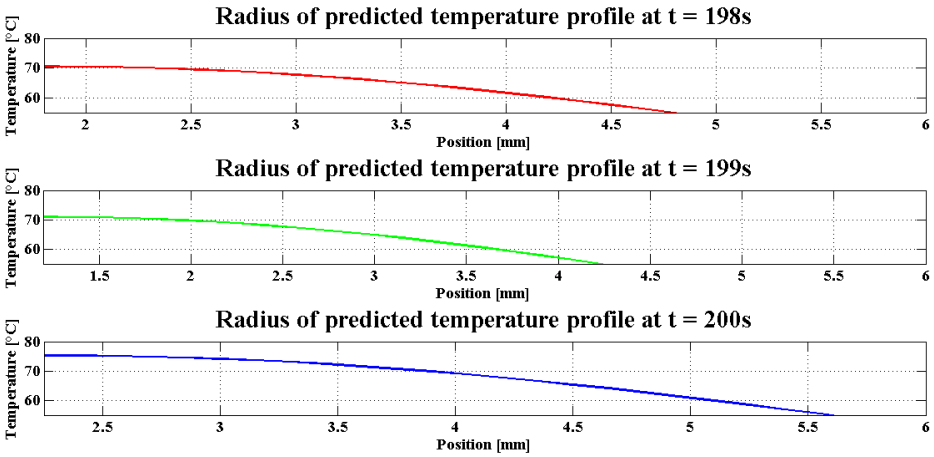


Figure 5.5. Radius change of predicted temperature profiles.

The difference between the measured and the predicted radius of distribution is illustrated in the figure 5.7. The errors are approximately 0.1, 0, and 0.3 millimeters. Even though it may seem that the errors are small, the slightest miscalculation of the thermal distribution can lead to the damage of living cells. Therefore, in the future the prediction algorithm is expected to forecast with the smallest error of nanometers.

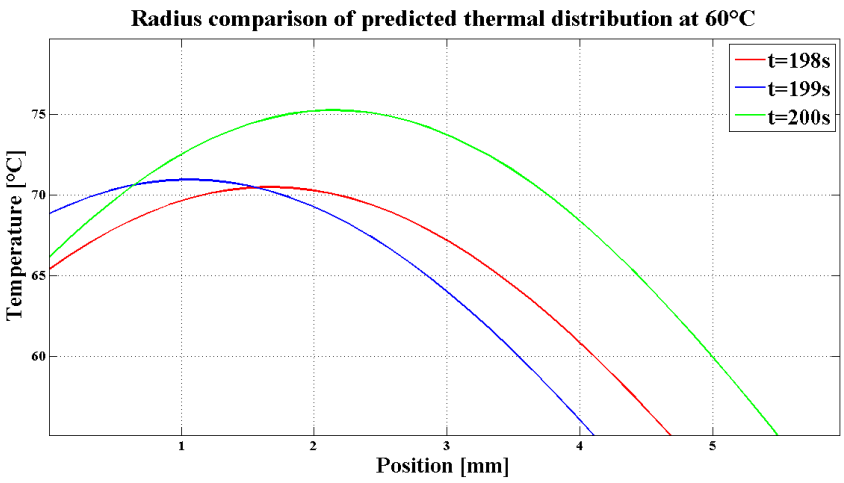


Figure 5.6. Predicted temperature profile change.

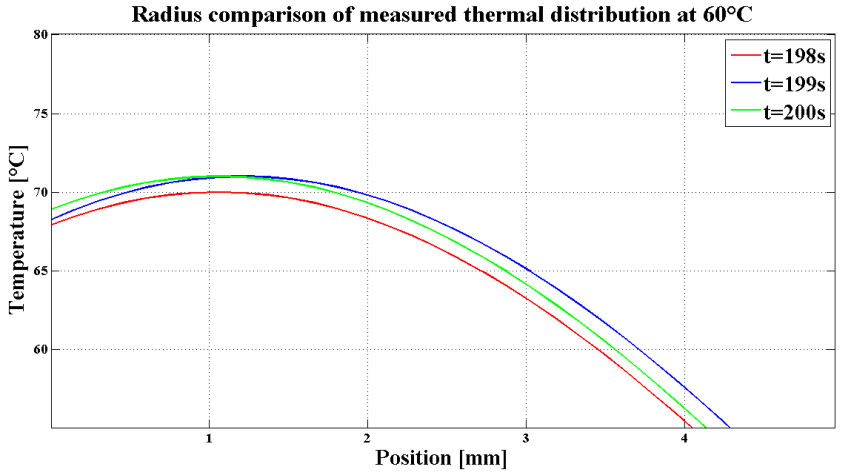


Figure 5.7. Measured temperature profile change during prediction.

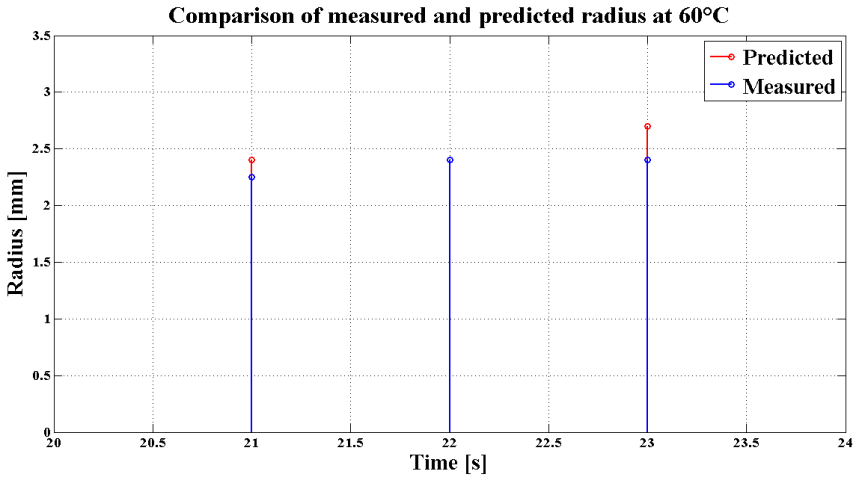


Figure 5.8. Difference between predicted and measured radius in millimeter scale.

For the purpose of examination, the prediction algorithm was applied to another thermal map of different experiment. The results of the prediction can be observed in the Appendix D. The results showed similar performance of prediction, therefore it can be concluded that prediction algorithm based on linear regression is not able to detect with anticipated accuracy. The alternative approaches that can be taken into account as a future work are non-linear regression model, or prominent in present day Neural Networks in case of massive amount of experiment data.

Chapter 6 – Conclusion

Developing of the code give to the developer good understanding of the work principles of almost any system, which in this case is combination thermal ablation process and Chirped Fiber Bragg Grating sensors. This knowledge obtained has supported me in writing of this report. The report considered the development of demodulation software on MATLAB and LabVIEW of power reflectivity spectrum of CFBG sensor in order to obtain the temperature pattern. Moreover, it examined feasibility of prediction of temperature profile distribution. Although the prediction of temperature pattern using linear regression still requires enhancements due to insufficiently precise forecasting results, the demodulation of few experiments have showed some convincing results. Specifically saying, the demodulation software performs their task both on MATLAB and LabVIEW. However, there are still some work to do. For the instance, the results of comparison to the existing LUNA OBR 4600 technology have shown that the precision of estimations must be improved. Also, the code needs acceleration of compilation process. In the near future, constant debugging and designing of faster, more precise and accurate demodulation and prediction algorithms are expected.

Bibliography

[1] T. Erdogan, Fiber grating spectra, J. Lightwave Technol. 15 (1997) 1277–1294.

[2] D. Tosi, "Review and Analysis of Peak Tracking Techniques for Fiber Bragg Grating Sensors", Sensors, vol. 17, no. 10, p. 2368, 2017.

[3] W. Chen, R. Gassino, Y. Liu, A. Carullo, G. Perrone, A. Vallan, D. Tosi, Performance assessment of FBG temperature sensors for laser ablation of tumors, in: Medical Measurements and Applications (MeMeA), 2015 IEEE International Symposium on, IEEE, pp. 324–328.

[4] S. Korganbayev, et al., Detection of thermal gradients through fiber-optic Chirped Fiber Bragg Grating (CFBG): medical thermal ablation scenario, Opt. Fiber Technol. 41 (2018) 48–55.

[5] Tašner, T., Lovrec, D., Tašner, F. and Edler, J., Comparison of LabVIEW and MATLAB for scientific research, Annals of the Faculty of Engineering Hunedoara - International Journal of Engineering, 10(3), 2012, pp.389-394.

[6] C. Parsons, J. Pimiento, D. Cheong, S. Marzban, R. Gonzalez, D. Johnson, G. Letson and J. Zager, "The role of radical amputations for extremity tumors: A single institution experience and review of the literature", Journal of Surgical Oncology, vol. 105, no. 2, pp. 149-155, 2011.

[7] Bortolotto C, Macchi S, Veronese L, Dore R, Draghi F, Rossi S. Radiofrequency ablation of metastatic lesions from breast cancer. Journal of ultrasound. 2012 Sep 1; 15(3):199-205.

[8] Chu KF, Dupuy DE. Thermal ablation of tumours: biological mechanisms and advances in therapy. Nature Reviews Cancer. 2014 Mar; 14(3):199.

[9] Rhim H, Goldberg SN, Dodd GD, Solbiati L, Lim HK, Tonolini M, Cho OK. Essential techniques for successful radio-frequency thermal ablation of malignant hepatic tumors. Radiographics. 2001 Oct; 21(suppl_1):S17-35.

[10] Simon CJ, Dupuy DE, Mayo-Smith WW. Microwave ablation: principles and applications. Radiographics. 2005 Oct; 25(suppl_1):S69-83.

[11] Liang P, Wang Y, Yu X, Dong B. Malignant liver tumors: treatment with percutaneous microwave ablation—complications among cohort of 1136 patients. Radiology. 2009 Jun; 251(3):933-40.

[12] Schwartz JA, Shetty AM, Price RE, Stafford RJ, Wang JC, Uthamanthil RK, Pham K, McNichols RJ, Coleman CL, Payne JD. Feasibility study of particle-assisted laser ablation of brain tumors in orthotopic canine model. Cancer research. 2009 Feb 15;69(4):1659-67.

[13] Zhang L, Zhu H, Jin C, Zhou K, Li K, Su H, Chen W, Bai J, Wang Z. High-intensity focused ultrasound (HIFU): effective and safe therapy for

hepatocellular carcinoma adjacent to major hepatic veins. European radiology. 2009 Feb 1;19(2):437.

[14] Schena E, Tosi D, Saccomandi P, Lewis E, Kim T. Fiber optic sensors for temperature monitoring during thermal treatments: an overview. Sensors. 2016 Jul 22;16(7):1144.

[15] Dromain C, de Baere T, Elias D, Kuoch V, Ducreux M, Boige V, Petrow P, Roche A, Sigal R. Hepatic tumors treated with percutaneous radio-frequency ablation: CT and MR imaging follow-up. Radiology. 2002 Apr; 223(1):255-62.

[16] Macchi EG, Gallati M, Braschi G, Cigada A, Comolli L. Temperature distribution during RF ablation on ex vivo liver tissue: IR measurements and simulations. Heat and Mass Transfer. 2015 May 1; 51(5):611-20.

[17] Diakides NA, Bronzino JD (2007) Medical Infrared Imaging. CRC Press, Boca Raton.

[18] Wu N, Tian Y, Zou X, Zhai Y, Barringhaus K, Wang X. A miniature fiber optic blood pressure sensor and its application in in vivo blood pressure measurements of a swine model. Sensors and Actuators B: Chemical. 2013 May 1; 181:172-8.

[19] Pinet E, Pham A, Rioux S. Miniature fiber optic pressure sensor for medical applications: an opportunity for intra-aortic balloon pumping (IABP) therapy.

In 17th International Conference on Optical Fibre Sensors 2005 May 23 (Vol. 5855, pp. 234-238). International Society for Optics and Photonics.

[20] Gelabert-Gonzalez M, Ginesta-Galan V, Sernamito-Garcia R, Allut AG, Bandin-Dieguez J, Rumbo RM. The Camino intracranial pressure device in clinical practice. Assessment in a 1000 cases. Acta neurochirurgica. 2006 Apr 1; 148(4):435-41.

[21] J. W. Arkwright, I, D. Underhill, S. A. Maunder, N. Blenman, M. M. Szczesniak, L. Wiklendt, I. J. Cook, D. Z. Lubowski, P. G. Dinning, "Design of a high-sensor count fibre optic manometry catheter for in-vivo colonic diagnostics," Opt. Expr., vol. 17, no. 25, pp. 22423-22431, 2009.

[22] Roriz P, Frazão O, Lobo-Ribeiro AB, Santos JL, Simões JA. Review of fiber-optic pressure sensors for biomedical and biomechanical applications. Journal of biomedical optics. 2013 May; 18(5):050903.

[23] Tosi D, Macchi EG, Braschi G, Gallati M, Cigada A, Poeggel S, Leen G, Lewis E. Monitoring of radiofrequency thermal ablation in liver tissue through fibre Bragg grating sensors array. Electronics Letters. 2014 Jun 27; 50(14):981-3.

[24] Webb DJ, Hathaway MW, Jackson DA, Jones S, Zhang L, Bennion I. First in-vivo trials of a fiber Bragg grating based temperature profiling system. Journal of biomedical optics. 2000 Jan;5(1):45-51.

[25] Jelbuldina M, Korobeinyk A, Korganbayev S, Tosi D, Dukenbayev K, Inglezakis VJ. Real time Temperature Monitoring in Liver during Magnetite Nanoparticle-enhanced Microwave Ablation with Fiber Bragg Grating sensors: Ex Vivo Analysis. IEEE Sensors Journal. 2018 Aug 16.

[26] Jelbuldina M, Korobeinyk AV, Korganbayev S, Inglezakis VJ, Tosi D. Fiber Bragg grating based temperature profiling in ferromagnetic nanoparticles-enhanced radiofrequency ablation. Optical Fiber Technology. 2018 Jul 31; 43:145-52.

[27] Macchi EG, Tosi D, Braschi G, Gallati M, Cigada A, Lewis E. Optical fiber sensors-based temperature distribution measurement in ex vivo radiofrequency ablation with submillimeter resolution. Journal of biomedical optics. 2014 Nov; 19(11):117004.

[28] Tosi D, Macchi EG, Gallati M, Braschi G, Cigada A, Rossi S, Leen G, Lewis E. Fiber-optic chirped FBG for distributed thermal monitoring of ex-vivo radiofrequency ablation of liver. Biomedical optics express. 2014 Jun 1; 5(6):1799-811.

[29] Korganbayev S, Orazayev Y, Sovetov S, Bazyl A, Schena E, Massaroni C, Gassino R, Vallan A, Perrone G, Saccomandi P, Caponero MA. Detection of thermal gradients through fiber-optic Chirped Fiber Bragg Grating (CFBG):

Medical thermal ablation scenario. Optical Fiber Technology. 2018 Mar 31; 41:48-55.

[30] "What is Fiber Bragg Grating?", Fosco Connect, 2018. [Online]. Available: https://www.fiberoptics4sale.com/blogs/archive-posts/95046406-what-is-fiber-bragg-grating. [Accessed: 19- Nov- 2018].

[31] "Archived: Fundamentals of Fiber Bragg Grating (FBG) Optical Sensing - National Instruments", Ni.com, 2018. [Online]. Available: http://www.ni.com/white-paper/11821/en/. [Accessed: 19- Nov- 2018].

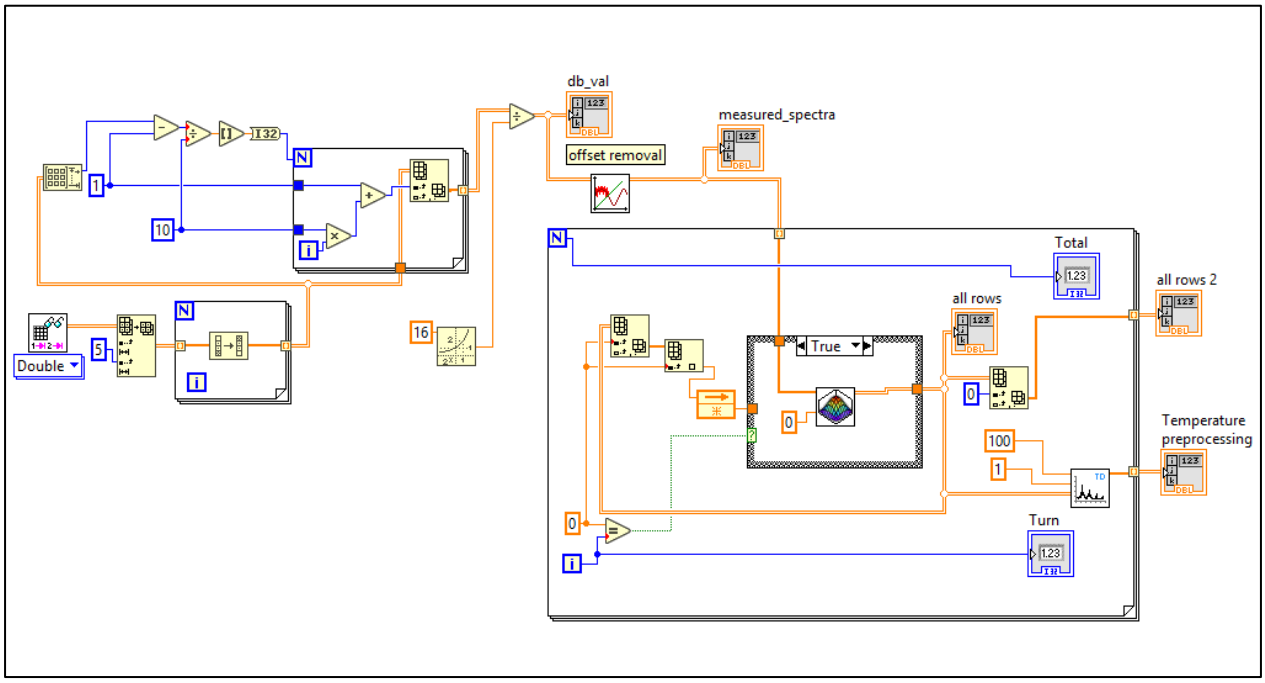
[32] M. Li, N. Zeng, C. Shi, M. Zhang and Y. Liao, 'Fiber Bragg grating distributed strain sensing: an adaptive simulated annealing algorithm approach', Optics & Laser Technology, vol. 37, no. 6, pp. 454-457, 2005.

[33] G. Palumbo, D. Tosi, A. Iadicicco, and S. Campopiano, “Analysis and design of chirped fiber Bragg grating for temperature sensing for possible biomedical applications,” IEEE Photonics J. 10(3), 1–15 (2018).

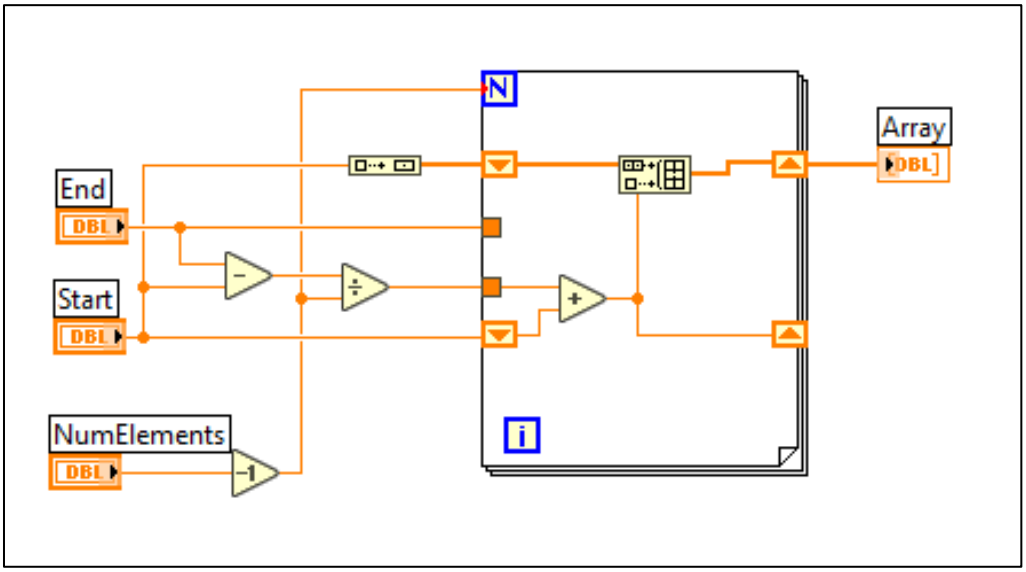
[34] A. I. Harris, "Coherent and incoherent detection", Proc. 29th Liege Int. Astrophys. Colloq. Ground-Based to Space-Borne Sub-mm Astronomy, pp. 165-169, 1990-Dec.

Appendix A

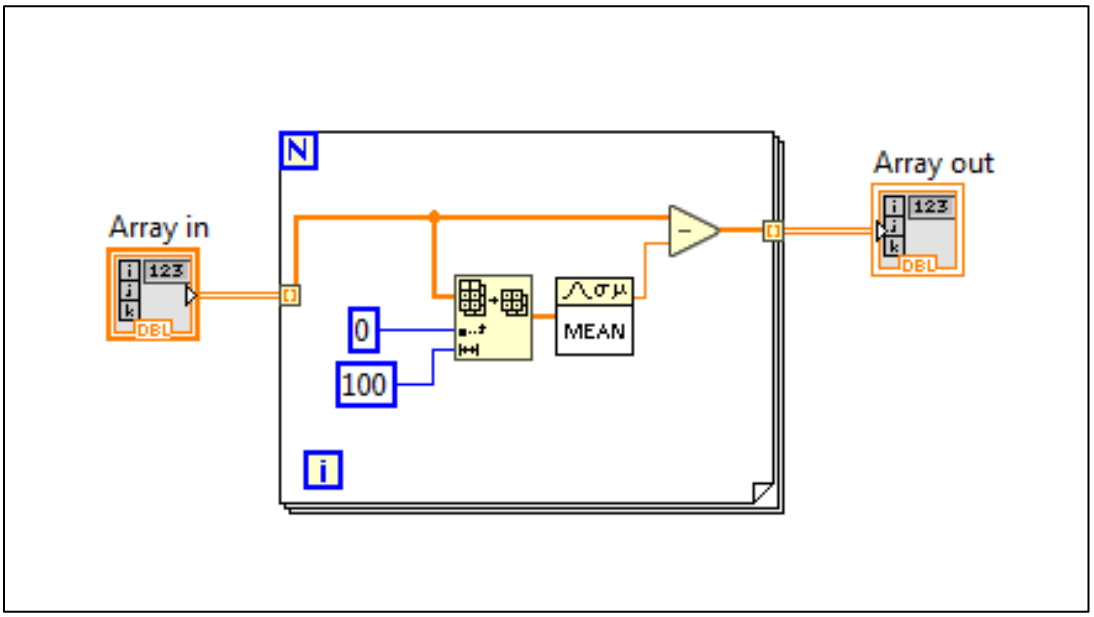
Main Code:



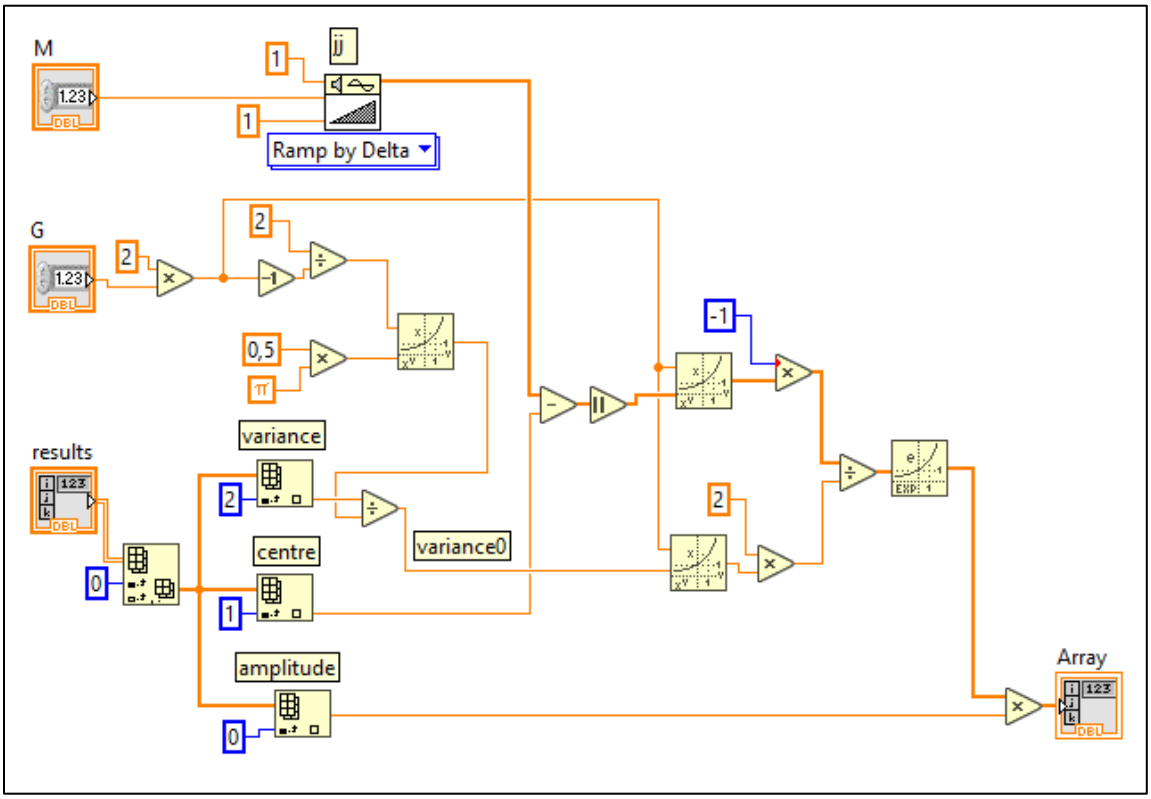
Linspace function:



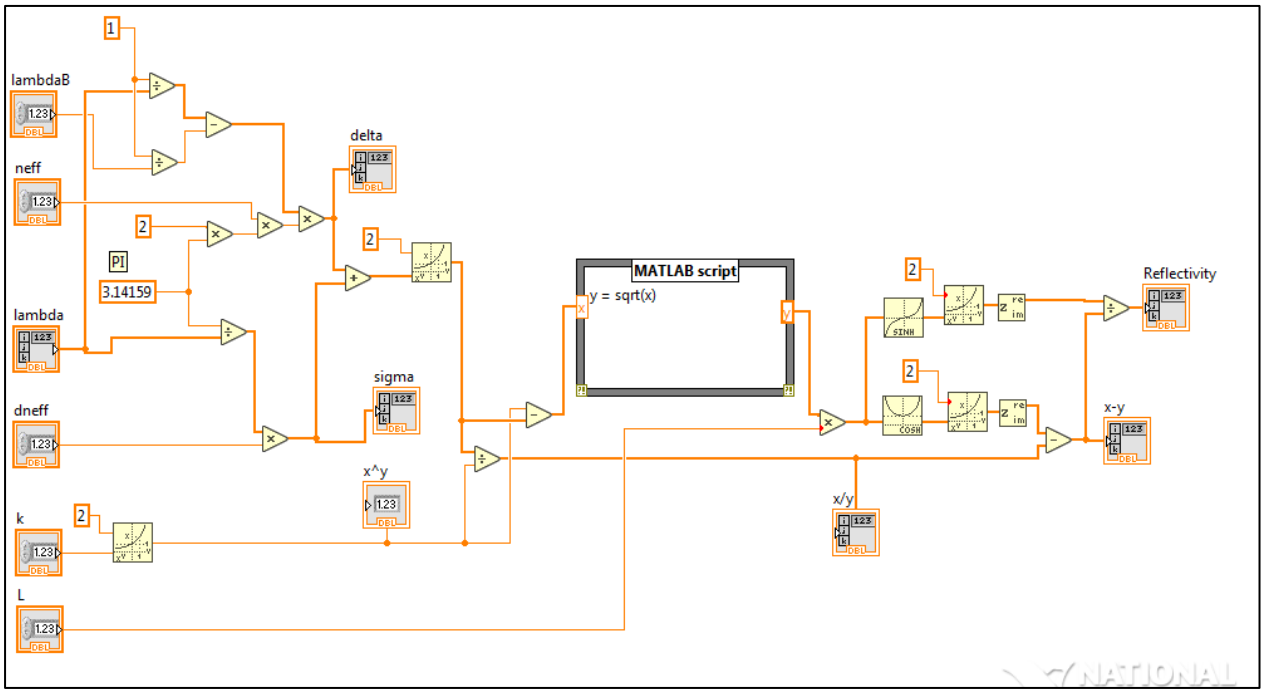
Offset removal:



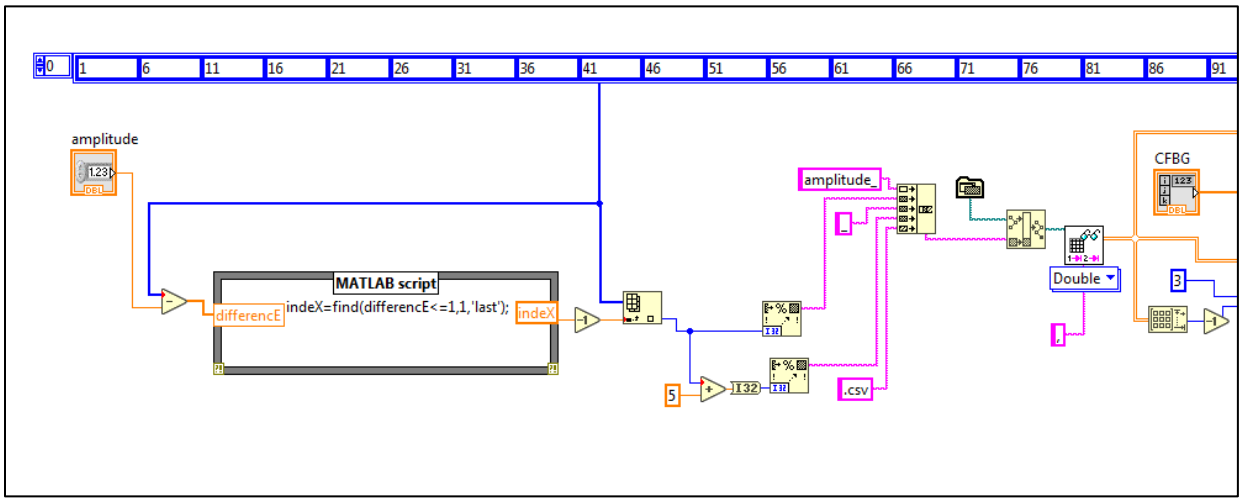
Temperature Distribution:



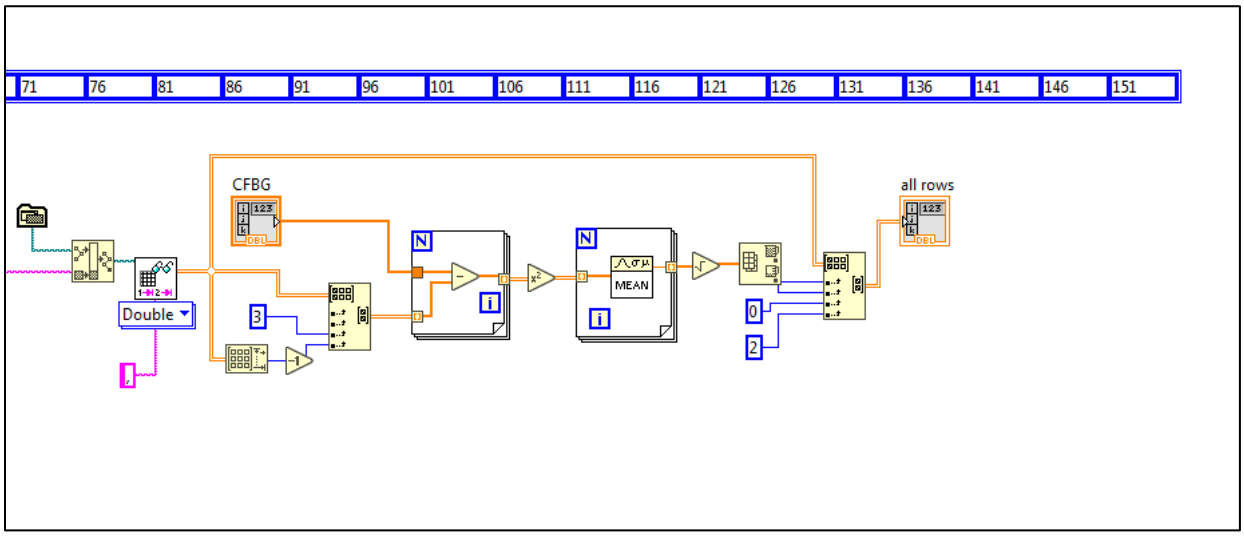
FBG generation function:



Temperature Reconstruction (a):



Temperature Reconstruction (b):



Appendix B

Simulation main code:

```
clc
close all
clear all
tic

%///////////////  READ SPECTRE
////////////////////////////////////////////
[db_val] = GetItalian; %take measurement of italian spectrometer
% lambdaS = 1510e-9;  %start of lambda, shown in plots
% lambdaE = 1595e-9;  %end of lambda, shown in plots

[scans,pixels] = size(db_val);

M = 100;              %number of sections, can be varied

%//////////////////  PARAMETERS OF GRATING
//////////////////////////////////
grating.kL= 0.62;
grating.dneff = 1e-6;          %change of refraction index
grating.neff = 1.5;            %refraction index

lengthofFBG = 0.1;  %length of LCFBG sensor   or Length can be
0.016
L = lengthofFBG/M;    %length of grating
k = grating.kL/L;           %strength


for ii = 1:scans
   db_val(ii,:)= db_val(ii,:)-mean(db_val(ii,1:100));
end


Xnm=(320)*(85e-9)/509;

CFBG=db_val(:,1:320);
FBG_array=db_val(:,321:end);


[CFBG_measurements,N_CFBG] = size(CFBG);
```

```
lambdaS = 1510e-9; %[m]
lambdaE = (1510e-9)+Xnm; %[m]
lambda_CFBG = linspace (lambdaS,lambdaE,N_CFBG);

guessL.LAMBDA1_guess=1534.4e-9;
guessL.LAMBDA2_guess=1557.7e-9;

lambdaLocal_final = linspace(guessL.LAMBDA1_guess,
guessL.LAMBDA2_guess, M);
ideal_spectrum_final = ones(1,length(lambda_CFBG));

for ii=1:M
    Reflectivity_final  = FBGgeneration( lambdaLocal_final(ii),
lambda_CFBG, L, grating,k); % specification
    Transmission_final = 1 - Reflectivity_final;
    ideal_spectrum_final = ideal_spectrum_final .*
Transmission_final; %transmission spectrum
end

ideal_Reflection_final=1-ideal_spectrum_final;


figure
plot(lambda_CFBG,ideal_Reflection_final,lambda_CFBG,CFBG(:,:));


%/////////////// PRE-FILTERING
////////////////////////////////////////////
[LAMBDA1_new, LAMBDA2_new,filter_final,Zero_filter_new] =
lambda_calculation2(CFBG,lambda_CFBG,M,L,grating,N_CFBG,guessL,k)
;

figure
plot(lambda_CFBG,filter_final)

for amplitude_val=1:5:150
amplitude_val
simulation_profiles(amplitude_val,lambda_CFBG, L, M, grating,
filter_final, LAMBDA1_new, LAMBDA2_new,k);
end

toc
```

Get_Italian function:

```
function [db_val] = GetItalian()

[nome,path]=uigetfile('*.txt','Spettro');
file=[path,nome];
s=load(file);

db_val=s(2:10:end,end:-1:6)/(2^16);

end
```

FBGgeneration function:

```
function [Reflectivity]=FBGgeneration(lambdaB,lambda,L,grating,k)

sigma = pi./lambda*grating.dneff;
delta = 2*pi*grating.neff*(1./lambda - 1./lambdaB);
sigmahat = delta + sigma;

Reflectivity = ( sinh(L*sqrt(k.^2 - sigmahat.^2)).^2 ) ./ ( (
cosh(L*sqrt(k.^2 - sigmahat.^2)).^2 ) - (sigmahat.^2)./(k.^2));

end
```

Lambda_calculation function:

```
function [LAMBDA1_new, LAMBDA2_new,filter_final,Zero_filter_new]
= lambda_calculation2(db_val_CFBG, lambda,M, L,
grating,N_CFBG,guessL,k)

%///////////////  OBTAIN FIRST SPECTRUM
////////////////////////////////////////////
new_real_spectrum=db_val_CFBG(1,:); %First spectrum (no
temperature profile) used for filtering

%/////////////// FILTERING
////////////////////////////////////////////

transfer_equalizer = equalizer(new_real_spectrum, lambda, M, L,
grating,guessL,k);
% creates ideal spectrum with given grating parameters
% obtain transfer_function=italian_filtered/simulated_ideal

% ITERATION ITERATION ITERATINO ITERATION

rangeLambda1=0.99*guessL.LAMBDA1_guess:0.1e-
9:1.01*guessL.LAMBDA1_guess;
ww=0;
size_lam=length(rangeLambda1);
B=zeros(2,size_lam);

for LAMBDA1=(0.99*guessL.LAMBDA1_guess):0.1e-
9:(1.01*guessL.LAMBDA1_guess)
    lambdaLocal = linspace(LAMBDA1, guessL.LAMBDA2_guess, M);
    Transm_partial_calc_LAMBDA1=ones(1,length(lambda));
    ww=ww+1;
    for ii = 1:M
        Reflectivity_calc  = FBGgeneration( lambdaLocal(ii),
lambda, L, grating,k);
        Transmission_calc = 1 - Reflectivity_calc;
        Transm_partial_calc_LAMBDA1 = Transm_partial_calc_LAMBDA1
.* Transmission_calc;
    end

    difference_a=sqrt((new_real_spectrum-(1-
Transm_partial_calc_LAMBDA1)).^2);

    B(ww,1)=mean(difference_a);
    B(ww,2)=LAMBDA1;
```

```
end

[~,LAMBDA1_position] = min(B(:,1));
LAMBDA1_new=B(LAMBDA1_position,2);

lambdaLocal_fin = linspace(LAMBDA1_new, guessL.LAMBDA2_guess, M);
ideal_spectrum_fin = ones(1,length(lambda));

for zz=1:M
    Reflectivity_fin  = FBGgeneration( lambdaLocal_fin(zz),
lambda, L, grating,k ); % specification
    Transmission_fin = 1 - Reflectivity_fin;
    ideal_spectrum_fin = ideal_spectrum_fin .* Transmission_fin;
%transmission spectrum
end

rangeLambda2=0.99*guessL.LAMBDA2_guess:0.1e-
9:1.01*guessL.LAMBDA2_guess;
ee=0;
sizelam2=length(rangeLambda2);
V=zeros(2,sizelam2);

for LAMBDA2=0.99*guessL.LAMBDA2_guess:0.1e-
9:1.01*guessL.LAMBDA2_guess
    lambdaLocal2 = linspace(LAMBDA1_new, LAMBDA2, M);
    Transm_partial_calc_LAMBDA2=ones(1,length(lambda));
    ee=ee+1;
    for ii = 1:M
        Reflectivity_calc  = FBGgeneration( lambdaLocal2(ii),
lambda,L, grating,k);
        Transmission_calc = 1 - Reflectivity_calc;
        Transm_partial_calc_LAMBDA2 = Transm_partial_calc_LAMBDA2
.* Transmission_calc;
    end

    difference_var=sqrt((new_real_spectrum-(1-
Transm_partial_calc_LAMBDA2)).^2);

    V(ee,1)=mean(difference_var);
    V(ee,2)=LAMBDA2;

end

[~,LAMBDA2_position] = min(V(:,1));

LAMBDA2_new=V(LAMBDA2_position,2);
```

```
display(LAMBDA1_new);
display(LAMBDA2_new);

lambdaLocal_final = linspace(LAMBDA1_new, LAMBDA2_new, M);
ideal_spectrum_final = ones(1,length(lambda));

for ii=1:M
    Reflectivity_final  = FBGgeneration( lambdaLocal_final(ii),
lambda, L, grating,k); % specification
    Transmission_final = 1 - Reflectivity_final;
    ideal_spectrum_final = ideal_spectrum_final .*
Transmission_final; %transmission spectrum
end

figure
plot(lambda*10e8,1-
ideal_spectrum_final,lambda*10e8,new_real_spectrum);
title('Measured and simulated spectra with new wavelength
range');
xlabel('Wavelength [nm]'); ylabel('Reflectivity')
legend('Simulated','Measured')

TF=new_real_spectrum./(1-ideal_spectrum_final);

for ff=1:(N_CFBG/2)
    if abs(new_real_spectrum(ff+1)-new_real_spectrum(ff))>0.05
        lambda_border1=ff;
        break;
    end
end

for gg=(N_CFBG-1):(-1):(N_CFBG/2)
    if abs(new_real_spectrum(gg+1)-new_real_spectrum(gg))>0.02
        lambda_border2=gg;
        break;
    end
end

Zero_filter_new = zeros(1,N_CFBG);
for rr=1:N_CFBG
    if (rr >= lambda_border1) && (rr <= lambda_border2)
        Zero_filter_new(rr) = 1;
    else
        Zero_filter_new(rr) = 0;
    end
end
```

```
%////////////////////////////////////////////////////////////////
/////////


filter_final = TF.*Zero_filter_new;

figure
plot(lambda, transfer_equalizer, 'b'); title('Filter Equalizer');
xlabel('Wavelength, nm');
ylabel('Measured Spectrum/Simulated Spectrum');

figure
plot(lambda, filter_final, 'b'); title('Filter FINAL');
xlabel('Wavelength, nm');
ylabel('Final filter = Equalizer * H filter');

end
```

simulation_profiles function:

```
function [] = simulation_profiles(amplitude_val,lambda_CFBG,  L,
M, grating, filter_final,LAMBDA1_new, LAMBDA2_new,k)

lambdaLocal = linspace(LAMBDA1_new, LAMBDA2_new, M);

amplitude_set=amplitude_val-2:1:amplitude_val+6;

variance_set=30:1:70;
centre_set=0:1:40;
superG_set=1;

sets = {amplitude_set, centre_set, variance_set, superG_set};
[amplitude_i, centre_j, variance_z, superG_k] = ndgrid(sets{:});
all_combinations = [amplitude_i(:) centre_j(:) variance_z(:)
superG_k(:)];
number_of_rows=size(all_combinations,1);

lambdaLocal_new1 = ones(1, M);
deltalambda_new1 = ones(1, M);

for rr=1:number_of_rows
    Transm_partial_calcREC = ones(1, length(lambda_CFBG));

    amplitude1=all_combinations(rr,1);
```

```
    centre1=all_combinations(rr,2);
    variance1=all_combinations(rr,3);
    G1=1;

     [delta_Gaussian1] = TempDistribution( G1, amplitude1, M,
variance1, centre1);

    for mm=0:(M-1)
        deltalambda_new1(mm+1) = delta_Gaussian1(mm+1)*10.2e-12;
        lambdaLocal_new1(mm+1) = lambdaLocal(mm+1) +
deltalambda_new1(mm+1);

        Reflectivity_calcREC  = FBGgeneration(
lambdaLocal_new1(mm+1), lambda_CFBG, L, grating,k);
        Transmission_calcREC = 1 - Reflectivity_calcREC;
        Transm_partial_calcREC = Transm_partial_calcREC .*
Transmission_calcREC;
    End

    Model_spectrumREC=1-Transm_partial_calcREC;

    filterMmodel=filter_final.*Model_spectrumREC;

saved_matrix(rr,:)=[amplitude1,centre1,variance1,filterMmodel];

end

filename= strcat('amplitude','_',
num2str(amplitude_set(1)+2),'_',num2str(amplitude_set(end)-1));

save(filename, 'saved_matrix');
```

TempDistribution function:

```
function [delta_Gaussian,ax] = TempDistribution(  G, amplitude,M,

variance, centre)


    jj=1:M;
    variance0=variance/((0.5*pi)^(2/(2*G-1)));
```

```
    delta_Gaussian = amplitude*exp((-(abs(jj-
centre)).^(2*G))/(2*variance0.^(2*G)));
    ax = exp((-(abs(jj-centre)).^(2*G))/(2*variance0.^(2*G)));

end
```

Main Code:

```
clc
close all
clear all

%///////////////  READ SPECTRE  ////////////////////////////
[db_val] = GetItalian; %take measurement of italian spectrometer

[scans,pixels] = size(db_val);

M = 100;              %number of sections, can be varied

%//////////////////  PARAMETERS OF GRATING  //////////////////
grating.kL= 0.62;
grating.dneff = 1e-6;          %change of refraction index
grating.neff = 1.5;            %refraction index

lengthofFBG = 0.1; %length of LCFBG sensor or Length can be 0.016
L = lengthofFBG/M;     %length of grating
k = grating.kL/L;            %strength


for ii = 1:scans
   db_val(ii,:)= db_val(ii,:)-mean(db_val(ii,1:100));
end

Xnm=(320)*(85e-9)/509;
CFBG=db_val(:,1:320);

lambdaS = 1510e-9; %[m]
lambdaE = (1510e-9)+Xnm; %[m]

guessL.LAMBDA1_guess=1534.4e-9;
guessL.LAMBDA2_guess=1557.7e-9;

lambdaLocal_final = linspace(guessL.LAMBDA1_guess,
guessL.LAMBDA2_guess, M);
%////////////  RECONSTRUCTION   ////////////////////////
```

```
YYmax = scans; % scans;

results(1)=0;
Temperature_preprocessing = zeros(YYmax, M);

%/////////////// RECONSTRUCTION  ////////////////////////
iteration=0;

for YY=1:YYmax

[results] = temperature_reconstruction(CFBG(YY,:),results(1));
  results2(YY, :) = results;
  [Temperature_preprocessing(YY,:), ax] = TempDistribution(1,
results(1), M, results(3), results(2));
  iteration=iteration+1;
  display(iteration)

end

return
```

temperature_reconstruction function:

```
function [results] = temperature_reconstruction(CFBG,amplitude)

 amps=[0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95
100 105 110 115 120 125 130 135 140 145]+1;
 differencE=amps-amplitude;
 indeX=find(differencE<=1,1,'last');
 amplitude_start=amps(indeX);

 filename= strcat('amplitude','_',
num2str(amplitude_start),'_',num2str(amplitude_start+5));

 A=load(filename);
 B=A.saved_matrix;
 CF=sqrt(mean(((CFBG-B(:,4:end)).^2)'));

[~,I]=min(CF);

results=B(I,1:3);


end
```

Appendix C

Temperature prediction MATLAB code:

```
clear;
clc;
close all;

ResultCSV = csvread('Result.csv');
[row, col] = find(ResultCSV >= 60);
ind = find(row == min(row));
heatP = [min(row), col(ind(1))];

steps = heatP(1);
% ResultCSV = csvread('Result.csv')';
% [rowsR, columnsR] = size(ResultCSV);

Name = {};

for (kk = 1:steps)
    Name{end+1} = strcat('VarName',int2str(kk));
end

%First

Train = ResultCSV(1:steps, :);
TrainT = array2table(Train');
TrainT.Properties.VariableNames = Name;

Test = ResultCSV(2:(steps + 1), :);
TestT = array2table(Test');
TestT.Properties.VariableNames = Name;

[modelT, RMSE] = trainRegressionModel(TrainT, Name);
predicVal = modelT.predictFcn(TestT);
error(1, :) = Test(steps, :) - predicVal';

% Second

Train = [Train; predicVal'];
TrainT = array2table(Train');
Name = [];
for (kk = 1:(steps + 1))
    Name{end+1} = strcat('VarName',int2str(kk));
end
TrainT.Properties.VariableNames = Name;

Test = [Train(2:end, :); ResultCSV((steps + 1), :)];
```

```
TestT = array2table(Test');
TestT.Properties.VariableNames = Name;

[modelT, RMSE] = trainRegressionModel(TrainT, Name);
predicVal2 = modelT.predictFcn(TestT);
error(2, :) = Test(steps, :) - predicVal2';

% Third

Train = [Train; predicVal2'];
TrainT = array2table(Train');
Name = [];
for (kk = 1:(steps + 2))
    Name{end+1} = strcat('VarName',int2str(kk));
end
TrainT.Properties.VariableNames = Name;

Test = [Train(2:end, :); ResultCSV((steps + 2), :)];
TestT = array2table(Test');
TestT.Properties.VariableNames = Name;

[modelT, RMSE] = trainRegressionModel(TrainT, Name);
predicVal3 = modelT.predictFcn(TestT);
error(3, :) = Test(steps, :) - predicVal3';

Train = [Train; predicVal3'];

figure;
surf(Train);
title('Temp distribution with prediction');
xlabel('Sensor points');
ylabel('Time steps');
zlabel('Temperature');
axis([1 100 1 22 0 80]);
grid on;

figure;
surf(ResultCSV(1:(steps+3), :));
title('Actual temp distribution');
xlabel('Sensor points');
ylabel('Time steps');
zlabel('Temperature');
axis([1 100 1 22 0 80]);
grid on;

error2 = abs(error);
figure;

for ii = 1:3
    plot(error2(ii, :));
    hold on;
end
```

```
title('Error across sensor through 3 step prediction');
legend('Error 1', 'Error 2', 'Error 3');
xlabel('Sensor points');
ylabel('Temp error');
axis([1 100 0 (max(error2(:)) + 1)]);
grid on;
hold off;
```

TrainRegressionModel function:

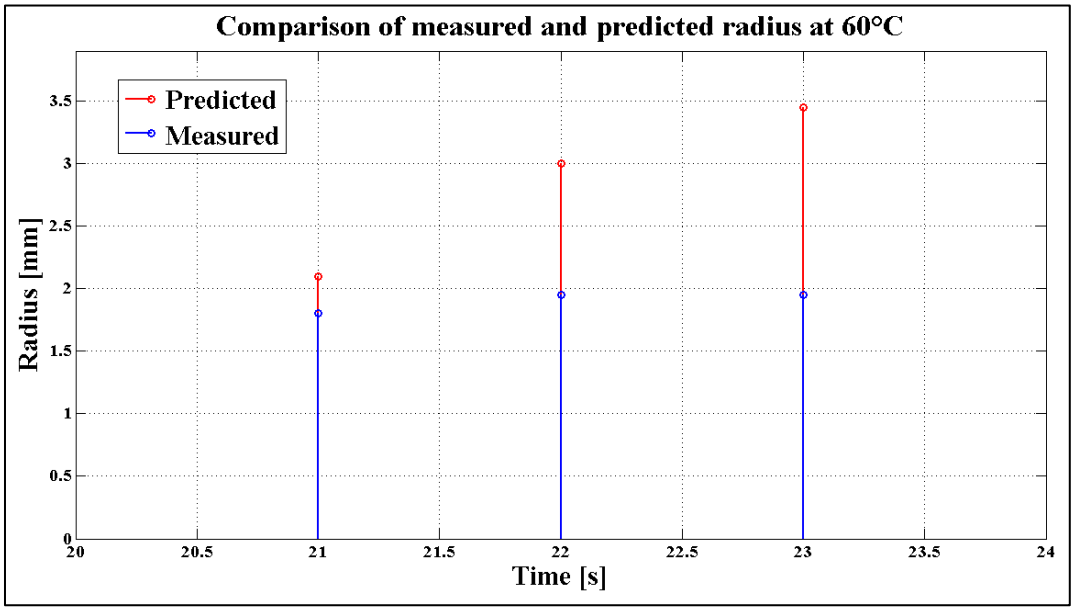
```
function [trainedModel, validationRMSE] =
trainRegressionModel(trainingData, predictorNames)
% [trainedModel, validationRMSE] = trainRegressionModel(trainingData)
% returns a trained regression model and its RMSE. This code recreates
the
% model trained in Regression Learner app. Use the generated code to
% automate training the same model with new data, or to learn how to
% programmatically train models.
%
%  Input:
%      trainingData: a table containing the same predictor and response
%       columns as imported into the app.
%
%  Output:
%      trainedModel: a struct containing the trained regression model.
The
%       struct contains various fields with information about the
trained
%       model.
%
%      trainedModel.predictFcn: a function to make predictions on new
data.
%
%      validationRMSE: a double containing the RMSE. In the app, the
%       History list displays the RMSE for each model.
%
% Use the code to train the model with new data. To retrain your model,
% call the function from the command line with your original data or
new
% data as the input argument trainingData.
%
% For example, to retrain a regression model trained with the original
data
% set T, enter:
%   [trainedModel, validationRMSE] = trainRegressionModel(T)
%
```

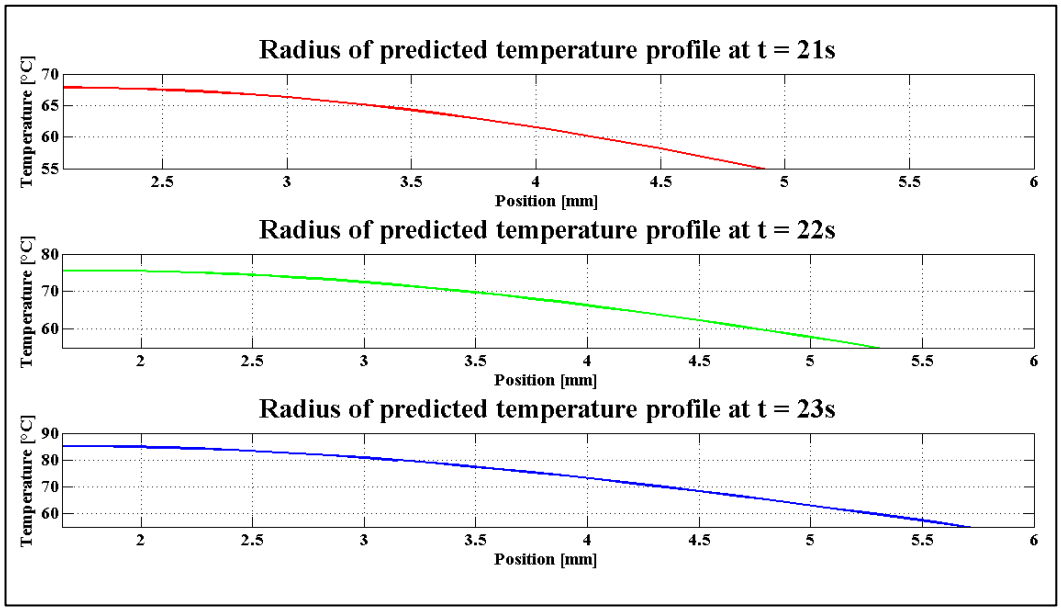
```
% To make predictions with the returned 'trainedModel' on new data T2,
use
%   yfit = trainedModel.predictFcn(T2)
%
% T2 must be a table containing at least the same predictor columns as
used
% during training. For details, enter:
%   trainedModel.HowToPredict

% Auto-generated by MATLAB on 30-Oct-2018 00:59:47


% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
inputTable = trainingData;
%predictorNames = {'VarName1', 'VarName2', 'VarName3'};
predictors = inputTable{:, 1:(end-1)};
response = inputTable{:, end};
isCategoricalPredictor = [false, false, false];

% Train a regression model
% This code specifies all the model options and trains the model.
% concatenatedPredictorsAndResponse = predictors;
% concatenatedPredictorsAndResponse{:, end + 1} = response;
concatenatedPredictorsAndResponse = inputTable;
concatenatedPredictorsAndResponse.Properties.VariableNames =
predictorNames;
linearModel = fitlm(...
    concatenatedPredictorsAndResponse, ...
    'linear', ...
    'RobustOpts', 'off');

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
linearModelPredictFcn = @(x) predict(linearModel, x);
trainedModel.predictFcn = @(x)
linearModelPredictFcn(predictorExtractionFcn(x));

% Add additional fields tVarName4o the result struct
trainedModel.RequiredVariables = predictorNames;
trainedModel.LinearModel = linearModel;
trainedModel.About = 'This struct is a trained model exported from
Regression Learner R2018b.';
trainedModel.HowToPredict = sprintf('To make predictions on a new
table, T, use: \n  yfit = c.predictFcn(T) \nreplacing ''c'' with the
name of the variable that is this struct, e.g. ''trainedModel''. \n
\nThe table, T, must contain the variables returned by: \n
c.RequiredVariables \nVariable formats (e.g. matrix/vector, datatype)
must match the original training data. \nAdditional variables are
ignored. \n \nFor more information, see <a
href="matlab:helpview(fullfile(docroot, ''stats'', ''stats.map''),
```

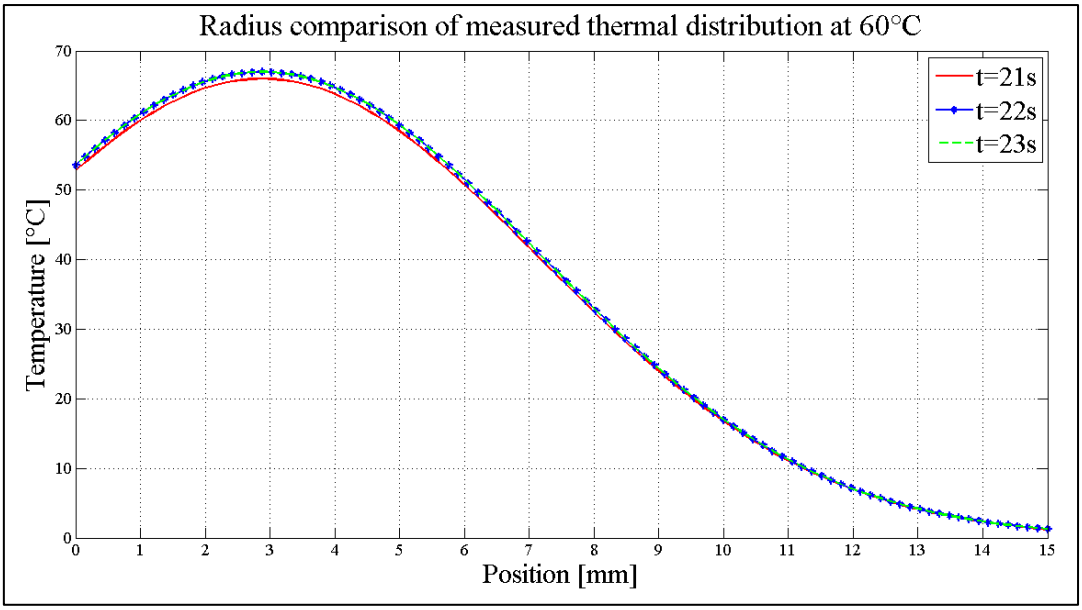
```
''appregression_exportmodeltoworkspace'')">How to predict using an
exported model</a>.');

% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
inputTable = trainingData;
%predictorNames = {'VarName1', 'VarName2', 'VarName3'};
predictors = inputTable(:, 1:(end - 1));
response = inputTable{:, end};
isCategoricalPredictor = [false, false, false];

% Perform cross-validation
KFolds = 10;
cvp = cvpartition(size(response, 1), 'KFold', KFolds);
% Initialize the predictions to the proper sizes
validationPredictions = response;
for fold = 1:KFolds
    trainingPredictors = predictors(cvp.training(fold), :);
    trainingResponse = response(cvp.training(fold), :);
    foldIsCategoricalPredictor = isCategoricalPredictor;

    % Train a regression model
    % This code specifies all the model options and trains the model.
    concatenatedPredictorsAndResponse = trainingPredictors;
    concatenatedPredictorsAndResponse{:, end + 1} = trainingResponse;
    concatenatedPredictorsAndResponse.Properties.VariableNames =
predictorNames;
    linearModel = fitlm(...
        concatenatedPredictorsAndResponse, ...
        'linear', ...
        'RobustOpts', 'off');

    % Create the result struct with predict function
    linearModelPredictFcn = @(x) predict(linearModel, x);
    validationPredictFcn = @(x) linearModelPredictFcn(x);

    % Add additional fields to the result struct

    % Compute validation predictions
    validationPredictors = predictors(cvp.test(fold), :);
    foldPredictions = validationPredictFcn(validationPredictors);

    tModel.predictFcn = validationPredictFcn;

    % Add additional fields tVarName4o the result struct
    tModel.RequiredVariables = predictorNames;
    tModel.LinearModel = linearModel;
    tModel.About = 'This struct is a trained model exported from
Regression Learner R2018b.';

    % Store predictions in the original order
```

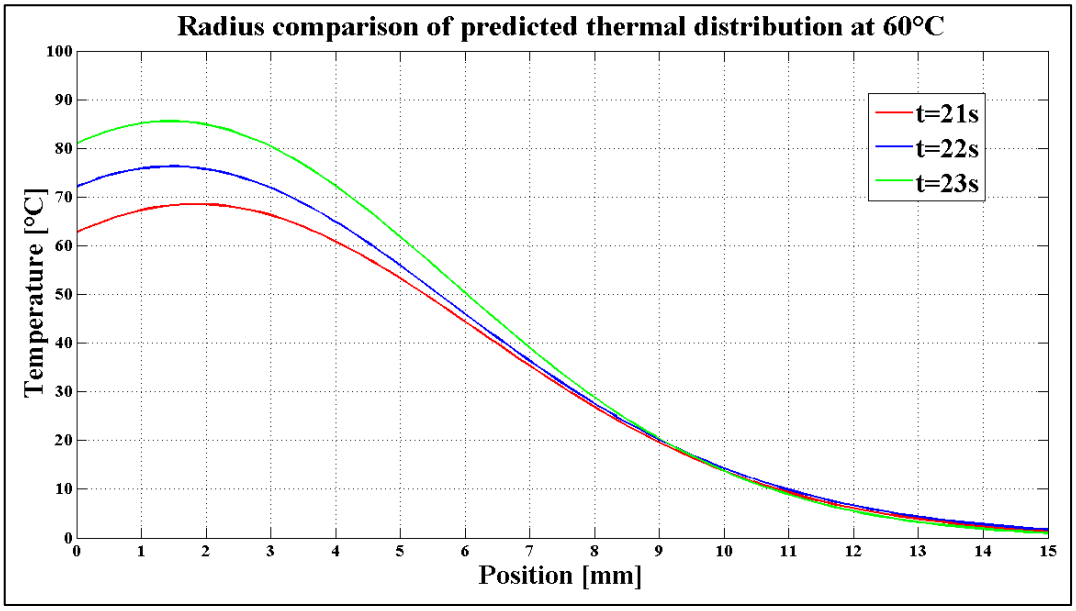
```
    validationPredictions(cvp.test(fold), :) = foldPredictions;
end


trainedModel = tModel;

% Compute validation RMSE
isNotMissing = ~isnan(validationPredictions) & ~isnan(response);
validationRMSE = sqrt(nansum(( validationPredictions - response ).^2) /
numel(response(isNotMissing) ));
```

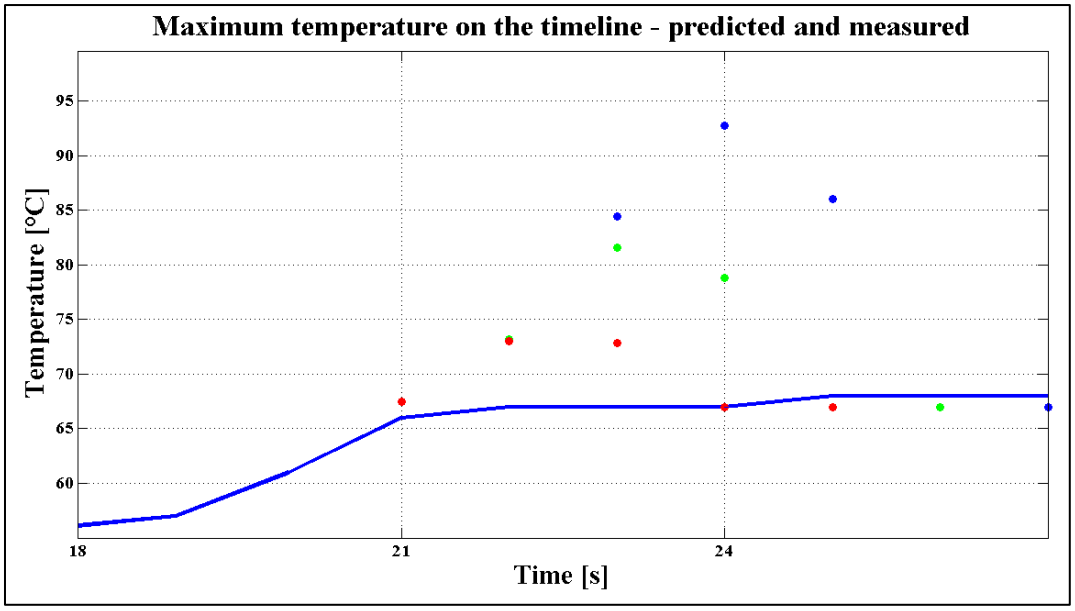
Appendix D

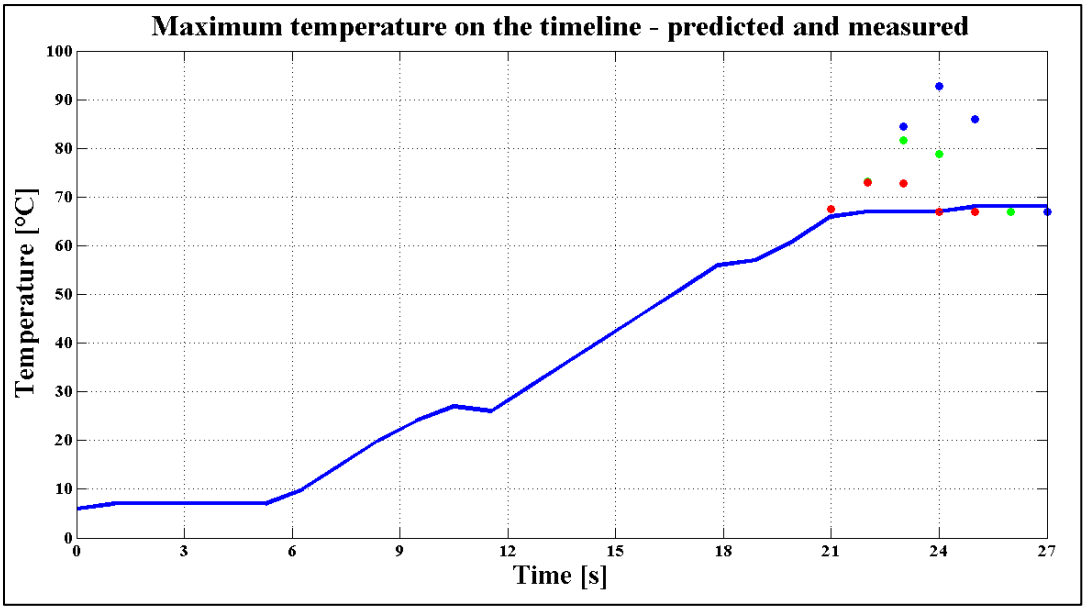

Comparison of measured and predicted radius at 60°C
Predicted
Measured
Radius [mm]
Time [s]
0
0.5
1
1.5
2
2.5
3
3.5
20
20.5
21
21.5
22
22.5
23
23.5
24


Radius of predicted temperature profile at t = 21s
Temperature [°C]
Position [mm]
Radius of predicted temperature profile at t = 22s
Temperature [°C]
Position [mm]
Radius of predicted temperature profile at t = 23s
Temperature [°C]
Position [mm]


Radius comparison of measured thermal distribution at 60°C
Temperature [°C]
Position [mm]
t=21s
t=22s
t=23s


Radius comparison of predicted thermal distribution at 60°C
Temperature [°C]
Position [mm]
t=21s
t=22s
t=23s


Maximum temperature on the timeline - predicted and measured
Temperature [°C]
95
90
85
80
75
70
65
60
18
21
24
Time [s]


Maximum temperature on the timeline - predicted and measured
Temperature [°C]
100
90
80
70
60
50
40
30
20
10
0
0
3
6
9
12
15
18
21
24
27
Time [s]