



POLITECNICO DI TORINO

Dipartimento di Ingegneria Meccanica ed Aerospaziale

Master Thesis in Aerospace Engineering

Academic Year 2017/2018

Inverse heat transfer problem applied to CH₄/O₂ single and multi-element rocket thrust chambers

Author:

Alessandro Villani

Advisor:

Prof. Dario Giuseppe Pastrone

Supervisor TUM:

Nikolaos Perakis, M. Sc.

Luglio 2018

Declaration of Authorship

Name: Alessandro Villani

I hereby declare that this thesis is my own work prepared without the help of a third party. No other than the listed literature and resources have been used. All sources transferred literally or analogously to this work have been labeled accordingly.

Additionally, I hereby certify that this thesis has not been underlain in any other examination procedure up to the present.

.....
Date, Sign

Abstract

The present study intends to investigate and analyze the main issues affecting a conjugate gradient optimisation method (CGM) for parameter heat flux estimating applied to some experimental tests carried out on single- and multi-injector combustion chambers at the Institute of Propulsion of Flight (LFA) at the Technische Universität München (TUM). Each optimisation steps implemented in a Matlab code by Perakis et al. have been explained in detail. Hence, the code has been modified with the purpose to make it faster and more accurate and than it has been tested with a numerical function simulating heat flux rising along combustion chamber. An analysis of the nozzle boundary condition has been performed to estimate errors that occur by assuming it adiabatic. Finally, same modifications have been applied to five element combustion chamber with particular attention to the interpolation of the heat flux along combustion chamber.

Keywords:

Inverse Heat Conduction Problem, IHCP, Parameter Estimation, Conjugate Gradient Method, MoRaP, Single-Element, Multi-Element, Rocket Engine Combustion

Contents

1	Introduction	1
1.1	Fundamentals of Rocket Propulsion	3
1.1.1	Definitions	4
1.1.2	Thrust	5
1.1.3	Exhaust Velocity	5
1.1.4	Energy and Efficiencies	6
1.2	Inverse Method	7
1.3	Why Oxygen/Methane Combustion	8
1.4	Scope of this thesis	10
2	Single and Multi Element Hardware Description	12
2.1	Single-Element Combustion Chamber	12
2.1.1	Sensors	16
2.1.1.1	Pressure Transducers	16
2.1.1.2	Thermocouples	16
2.2	Multi-Element Combustion Chamber	18
2.3	Experimental Inputs and Output	20
3	Inverse Method	23
3.1	The Direct Problem	24
3.1.1	Boundary Conditions	25
3.1.1.1	Natural Convection on Outer Walls	25
3.1.1.2	Forced Convection on Inner Walls	26
3.1.1.3	Forced Convection on Cooling Channels Walls	27
3.1.1.4	Adiabatic Walls	28
3.1.2	Finite Difference Method for 3D Problems	28
3.2	The Inverse Problem	30
3.3	The Iterative Procedure	31

3.3.1	The Sensitivity Concept	31
3.3.2	The Direction of Descent	33
3.3.3	The Step Size	33
3.4	The Stopping Criterion	34
4	RoqFITT Code and its Modifications	37
4.1	RoqFITT Code	37
4.1.1	Loading Experimental Data and User Inputs	37
4.1.2	Initializations	38
4.1.3	Inverse Loop Start	48
4.1.4	Code Outputs	51
4.2	Code Modifications	54
4.2.1	Modifications of Time Segments	56
4.2.2	Newton Rapshon Method	60
4.2.3	Analysis and Differences Between Jacobi Matrix and Sensitivity Matrix	62
4.3	Comparison of Results	66
5	Validation of RoqFITT	71
5.1	Effect of the Different Nozzle Boundary Conditions	79
6	Heat Flux Interpolation in Five-Elements Combustion Chamber	83
6.1	Matlab Tools for Interpolation	84
6.2	A Different Way of Computing New Heat Flux Values for Interpolation	88
6.3	Inverse Distance Weighted Interpolation	90

List of Figures

1.1	Differences between Direct and Inverse Problems	2
1.2	Schematic Flow Diagram of a Liquid Propellant Rocket Engine (a) and of a Thrust Chamber (b)[22]	4
2.1	MoRaP Combustion Chamber	14
2.2	Measurement Systems	15
2.3	Modelled Combustion Chamber	16
2.4	Multi-Element Combustion Chamber and Thermocouples Arrangement in a Plane	19
2.5	Thermocouples Positions above the Upper Hot Gas Wall	20
2.6	Transient Pressure Measurements	22
2.7	Transient Temperature Measurements	22
3.1	Discretization for an Internal Node [18]	28
3.2	Discretization for Particular Nodes [18]	30
3.3	Transient Optimisation Procedure	35
3.4	Steps of the Parameter Update	36
4.1	Time Discretization	40
4.2	Thermocouples and Parameters Positions on $z - y$ Plane in Single-Element	41
4.3	Initial Temperature condition Assigned to all Nodes	42
4.4	Example of How the Code Scans the Domain	42
4.5	Example of Heat Flux First Guess	43
4.6	Example of How the MM Matrix is Filled (a), Reference Node for the Example Above (b)	45
4.7	Optimization Process on Segments	48
4.8	Example of Heat Flux Interpolation on Internal Nodes for Single-Element .	50
4.9	Example of Heat Flux Interpolation on Internal Nodes for Five-Element . .	51
4.10	Example of How New Points are added for Interpolation	52

4.11	Calculated and Measured Temperatures Match in Thermocouples Positions Over Time	53
4.12	Heat Flux Over Time in Middle Point at All Distances from the Injector Plate	54
4.13	Heat Flux Over z in Middle Point for Several Times	55
4.14	Heat Flux Over x on the Upper Hot Gas Wall at Evaluation Time for Different Planes	56
4.15	Computed Temperature Along x on the Upper Hot Gas Wall at Evaluation Time for Different Planes	57
4.16	Computed Temperature Along z at 1 mm from the Hot Gas Wall in Single-Element	58
4.17	Computed Temperature Along z at 1 mm from the Hot Gas Wall at Evaluation Time in Five-Elements, 3 R	58
4.18	Computed Temperature Along z at 1 mm from the Hot Gas Wall at Evaluation Time in Five-Elements	59
4.19	Computed Temperature Along z at 1 mm from the Hot Gas Wall at Evaluation Time in Five-Elements	60
4.20	New Optimization Process on Time Segments	61
4.21	Example of Connection Between Computed Temperature and Applied Heat Flux Parameters in a Segment	61
4.22	Sensitivity Coefficients	63
4.23	Sensitivity Matrices Elements Comparison	64
4.24	Comparison of sensitivity and Jacobi matrix All Over the Time Domain	65
4.25	Comparison of Computed and Measured Temperature for the Three Different Codes	67
4.26	Heat Flux Over Time at Middle Point $z = 170 mm$	68
4.27	Heat Flux Over z at Middle Point at Evaluation Time	69
4.28	Heat Flux Over x in Five-Element Combustion Chamber at Evaluation Time for Several Planes	70
4.29	Iteration necessary to convergence for each time-step	70
5.1	Numerical Heat Flux Evolution Along z for several time steps	72
5.2	Precision Errors in Thermocouples Readings Before the Ignition	73
5.3	Normal Distribution Errors Added to Numerical Temperatures in Thermocouples Position	75
5.4	Heat Flux Evolution Along z at Different Time for the Three Different Simulations	75

5.5	Heat Flux Evolution Over time at Different z for the Three Different Simulations	76
5.6	Root Mean Square Error Calculated Along z for Different Time Steps . . .	77
5.7	Thermocouples Arrangement After Their Movement	78
5.8	Heat Flux Evolution Over z at Evaluation Time	79
5.9	Heat Flux Boundary Condition on Nozzle Plane for the Different Tests . .	80
5.10	Estimated Heat Flux Over Time for the Upper Middle Point at $z = 0.289\text{ m}$	81
5.11	Estimated Heat Flux Over z in the Upper Middle Points at $t = 8.3215\text{ s}$.	81
5.12	Computed Temperature Over z in Middle Point at $t = 8.3215\text{ s}$	82
6.1	Example of Heat Flux Oscillations on the Upper Hot Gas Wall in Five-Element Combustion Chamber	84
6.2	Interpolation and Extrapolation Domains on the Upper Hot Gas Wall . . .	85
6.3	Example of Heat Flux Parameters Interpolation on the Upper Hot Gas Wall	85
6.4	Example of Heat Flux Parameters Extrapolation on the Upper Hot Gas Wall	86
6.5	New Parameters Points Arrangement on the Upper Hot Gas Wall	87
6.6	Heat Flux Interpolation All Over The Upper Hot Gas Wall Starting From The New Parameters Points Arrangement	87
6.7	New Points Added for Interpolation, the Heads of the Arrows Indicate the Points Involved in the Average	89
6.8	Heat Flux Evolution Along z in Middle Point of Horizontal Side at Evaluation Time	90
6.9	Heat Flux Evolution Along x for Different Planes at Evaluation Time . . .	90
6.10	Heat Flux Points Product of The Interpolation: Blue Points Indicate the Heat Flux Values Obtained With a Cubic Interpolation Along x , Grey Points Indicate Those Obtained From Inverse Distance Weighted Interpolation	91
6.11	Definition of Influence Domain for the <i>Node (1,2)</i> Lying on The Upper Hot Gas Wall	93
6.12	Definition of Influence Domain for the <i>Node (1,11)</i> Lying on The Upper Hot Gas Wall	93
6.13	Heat Flux Evolution along z in Middle Points at Evaluation Time, Comparison Between Original Code and Modified Code with IDW	94
6.14	Heat Flux Evolution along x for Different Parameters Planes At Evaluation Time, Comparison Between Original Code and Modified Code with IDW .	95
6.15	Iterations Necessary for Convergence	95

List of Tables

1.1	Most propallants used in in early 2000s	9
2.1	Combustion chamber dimensions	13
2.2	Material properties	13
2.3	Type T thermocouples features	17
2.4	Thermocouples positions	17
2.5	Thermocouples nomenclature	18
2.6	Thermocouples position	19
2.7	Ignition condition	21
4.1	Variables counting nodes	39
4.2	CGSM algorithm	46
4.3	CGSM algorithm with Cholesky factorization	47
4.4	Elapsed computational time [h] Intel Core i5-7200U CPU @2.50GHz	68
5.1	Numerical experiment	71
5.2	Skewnwss and Kurtosis comparison	74
5.3	Thermocouples coordinates and movements [mm]	78
5.4	Nozzle heat flux boundary condition MW/m^2	80
6.1	Number of points added in planes	88

Chapter 1

Introduction

In the last years, the Institute of Flight Propulsion (LTF) of the Technical University of Munich (TUM) approached the study and the development of several combustion chambers (single- and multi-injector configuration) for rocket engine applications. Their main purposes are to investigate the combustion of different propellants (oxygen/methane, GO_2/GCH_4) from those usually used in the aerospace environment as well as to test different injection systems and new techniques for heat flux estimation. Since heat transfer problems are addressed in a large number engineering fields, the realisation and improvement of trustworthy tools for heat flux estimation appear to be some of the most important purposes in the field of research and for this reason it will be the main theme of this thesis. A detailed and accurate valuation of heat flux helps for optimizing the development of state-of-the-art combustion chambers. Thanks to several experiments it is possible to provide further information in this context and solid ground for future research. More specifically, a correct heat flux estimation is important for understanding the temperature distribution and thermal loads that arise in a combustion chamber. This goal is usually done, as in most of the modern thermodynamic problems, with CFD tools which solve direct problems, but often the differences between computed and measured temperatures testify a lack of information about the boundary conditions of the direct problem, hence, an inverse problem is necessary to be solved in order to replace the missing of the inputs, as well as to reconstruct them and to minimize the deviation between computational and experimental data [18]. It is important now to understand the differences between this two concepts, while a direct problem can be explained as the determination of the effects from the causes, an inverse problem is quite the opposite. Figure 1.1 shows that in a direct problem experimental tests and computational simulations move on in parallel and in the end the results are compared. If there is no matching of these results, assuming that both the experiment and the direct computational tool has been perfectly performed, it means

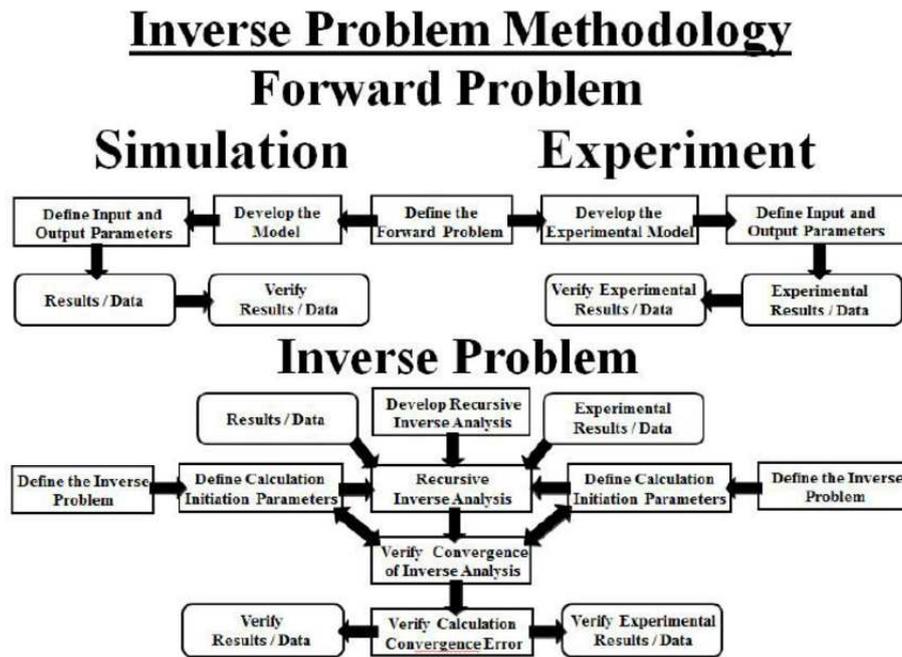


Figure 1.1: Differences between Direct and Inverse Problems

that some information used to solve the direct problem are not correct. Hence the problem becomes inverse because it is necessary to fill this gap of information.

In particular, this thesis deals with an Heat Transfer Conduction Problem (HTCP) in which some boundary conditions, i.e. heat flux on some boundaries (causes), are unknown. Hence an inverse problem must be solved to reconstruct them in detail. All methods used for this purpose involve the use of thermocouples measurements, i.e. it is necessary to know the temperature (effects) in some points of the domain. Therefore they depend extremely on the accuracy and precision of the instruments used in the experiment. This work is exactly intended to show how an inverse method works to reconstruct the heat flux along combustion chamber walls and how experimental data can affect its solution, as well as if it represents a trustworthy tool for heat flux estimation. The applied method has been implemented in Matlab environment and has experimental data from two test carried out on two different type of combustion chambers developed in TUM's laboratories as inputs.

1.1 Fundamentals of Rocket Propulsion

In order to give a complete picture of the problem this small section introduces the basic principles of rocket propulsion.

Rockets propulsion systems belong to a class of engines called non-airbreathing in which the combustion of propellants provides both the energy and the mass flow to create thrust. They can be classified into different categories, for example on the basis of energy source:

- chemical combustion;
- solar radiation;
- nuclear reaction;

or by the method of producing thrust:

- by thermodynamic expansion (nozzle is required);
- by magnetic and/or electric fields;

This thesis is interested in chemical rocket propulsion since it is used in the most of mission characterized by low-medium specific impulse (I_s). The energy released in this propulsion from the high-pressure combustion reaction of propellant chemicals permits to reach very high temperatures (up to $4100^\circ C$) and high velocities (up to 4300 m/sec) [22]. Considering that these gas temperatures are usually well above the melting point of the materials in which combustion chambers are built, it is necessary to cool or insulate all the surfaces that are exposed to the hot gases. However, since both single- and multi-injector combustion chambers do not have particular coatings and since only the five-element has a cooling system which has not been activated during the testing, the testing time of the experiments has been kept less than four seconds. A typical pressure-fed liquid propellant rocket engine system together with its thrust chamber is schematically shown in Fig. 1.2. The combustion chambers addressed in this thesis are gaseous bipropellant consisting of a gaseous oxidizer and a gaseous fuel. Inside the thrust chamber, the propellants react turning into hot gases, which subsequently are accelerated and ejected at a high velocity through a nozzle. In order to compare two or more combustion chambers, basic definitions and relations of propulsion, the exhaust velocity, and the efficiencies of converting the energy as well as other basic parameters are introduced.

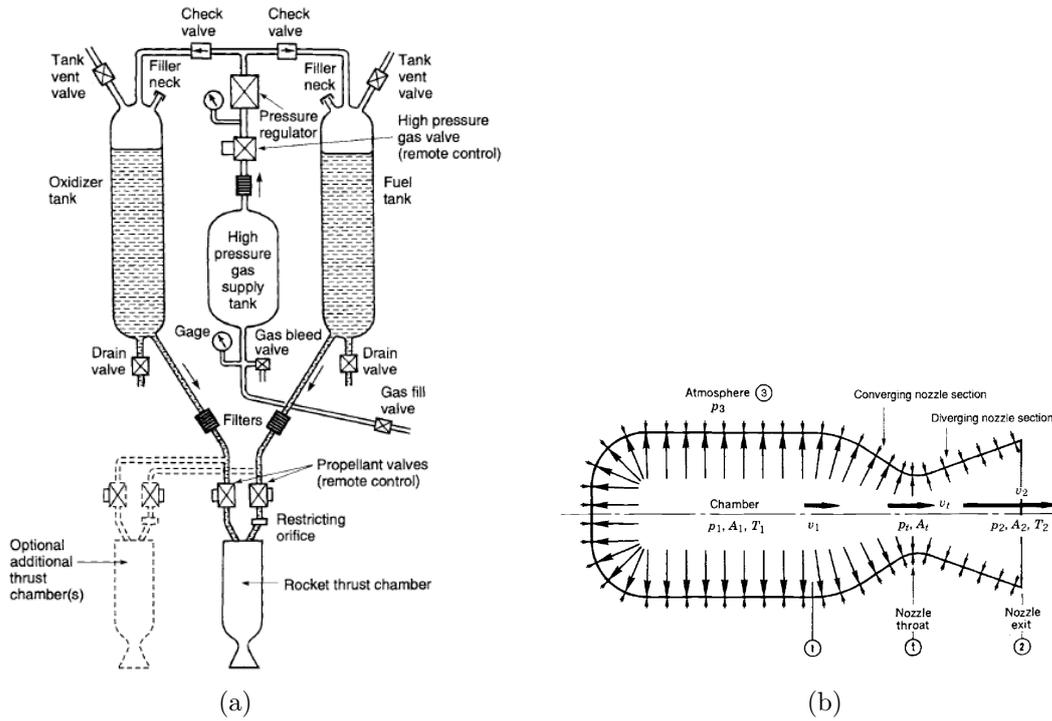


Figure 1.2: Schematic Flow Diagram of a Liquid Propellant Rocket Engine (a) and of a Thrust Chamber (b)[22]

1.1.1 Definitions

The total impulse I_t is defined as the thrust force F integrated over the burning time t .

$$I_t = \int_0^t F dt \quad (1.1)$$

I_t is related to the total energy released by all the propellant after the reaction. The specific impulse I_s is the total impulse per unit weight of propellant. A higher value means better performance.

$$I_s = \frac{\int_0^t F dt}{g_0 \int \dot{m} dt} \quad (1.2)$$

Another parameter that describes the performances of an engine system is the hot gas velocity on the nozzle exit-cross section. However, due to viscosity and to a non-uniform combustion this velocity is not uniform over the entire exit cross-section and it is difficult to estimate it accurately. For this reason in literature a uniform axial velocity c is introduced as the average equivalent velocity at which propellant is ejected from the vehicle.

$$c = I_s g_0 = F / \dot{m} \quad (1.3)$$

The impulse-to-weight ratio of a complete propulsion system is defined as the total impulse I_t divided by the initial vehicle weight w_0 and give an idea of the efficient design. It can be expressed as:

$$\frac{I_t}{w_0} = \frac{I_s}{m_f/m_p + 1} \quad (1.4)$$

where m_f is the mass of the vehicle when all the useful propellant mass m_p has been consumed and ejected.

1.1.2 Thrust

The thrust is the force produced by a rocket propulsion system acting upon a vehicle and is generated in consequence of the ejection of matter (reacted propellants) at high velocity.

$$F = \frac{dm}{dt} v_2 \quad (1.5)$$

This force represents the total propulsion force under the condition of adapted nozzle. The external pressure (local atmosphere), in fact, influences the thrust (Fig. ??). The size of the arrows indicates the relative magnitude of the pressure forces. The axial thrust can be determined by integrating all the pressures acting on areas and by projecting them on a plane normal to the nozzle axis. For a steadily operating rocket propulsion system, it can be expressed as

$$F = \dot{m}v_2 + (p_2 - p_3)A_2 \quad (1.6)$$

where the first term is the momentum thrust generated by the product of the propellant mass flow rate and its exhaust velocity relative to the vehicle and the second term represents the pressure thrust consisting of the product of the cross-sectional area at the nozzle exit A_2 and the difference between the exhaust gas pressure at the exit and the ambient fluid pressure.

1.1.3 Exhaust Velocity

For constant propellant mass flow, the effective exhaust velocity can be modified to

$$c = v_2 + (p_2 - p_3)A_2/\dot{m} \quad (1.7)$$

When p_2 is equal to p_3 , the effective exhaust velocity c is equal to the average actual exhaust velocity of the propellant gases v_2 and the thrust can be rewritten as

$$F = \dot{m}c \quad (1.8)$$

The characteristic velocity has been used frequently in the rocket propulsion literature in order to compare the relative performance of different chemical rocket propulsion system designs and propellants. It relates to the efficiency of the combustion and is independent of nozzle characteristics.

$$c^* = p_1 A_t / \dot{m} \quad (1.9)$$

1.1.4 Energy and Efficiencies

Efficiencies permit an understanding how energy is distributed in a rocket system. Those presented in this subsection are satisfactory in evaluating energy losses. Generally, two types of energy conversion processes occur in any propulsion system:

- the generation of energy (conversion of stored energy into available energy);
- the conversion to the form in which a reaction thrust can be obtained

The kinetic energy of ejected matter is the final product used for propulsion. The power of the jet P_{jet} is the time rate of expenditure of this energy, and for a constant gas ejection velocity v_2 this is a function of I_s and F .

$$P_{jet} = \frac{1}{2} F v_2 \quad (1.10)$$

For chemical rockets the energy is created by combustion and the maximum energy available per unit mass of chemical propellants is the heat of the combustion reaction Q_R ; the power unleashed by combustion is

$$P_{chem} = \dot{m} Q_R J \quad (1.11)$$

where J is a conversion constant which depends on the units used. A large portion of the energy of the exhaust gases is unavailable for conversion into kinetic energy and leaves the nozzle as residual enthalpy, in this way only a portion of this power is transmitted to the vehicle at any one time. This power is defined in terms of the thrust of the propulsion system F and the vehicle velocity u :

$$P_{vehicle} = F u \quad (1.12)$$

The propulsive efficiency determines how much of the total power produced by combustion has been converted in kinetic energy by the exhaust and is useful for propelling a vehicle

$$\begin{aligned}\eta_P &= \frac{\textit{vehicle power}}{\textit{vehicle power} + \textit{residual kinetic jet power}} = \\ &= \frac{Fu}{Fu + \frac{1}{2}(\dot{w}/g_0)(c - u)^2} = \frac{2u/c}{1 + (u/c)^2}\end{aligned}\tag{1.13}$$

where F is the thrust, u the absolute vehicle velocity, c the effective rocket exhaust velocity with respect to the vehicle, \dot{w} the propellant weight flow rate, and η_p the propulsive efficiency. The propulsive efficiency is a maximum when the forward vehicle velocity is exactly equal to the exhaust velocity.

1.2 Inverse Method

The aerospace environment has played a significant role in developing new tools concerning the resolution of inverse problems. The impossibility to make direct measurements of heat flux, for example, at lower surface of a space shuttle or all over combustion chamber walls, has always pushed toward new effective solutions. This thesis deals with an Inverse Heat Transfer Problems (IHTP). As will be better discussed in chapter 3, there are many different kinds of inverse problems, for example, for estimation of thermophysical properties of materials, or initial condition, or source terms or even geometric characteristics of a heated body [16], they, hence, depend on what type of information is missing, the treated one concerns the heat flux estimation. In a classical Direct Heat Conduction Problem (DHCP) the causes (boundary heat fluxes) are well known and the effects (temperature field in the body) are simply determined. The IHTP, instead, aims to estimate the unknown heat flux on some boundaries from the knowledge of temperature in the domain. An advantage of IHTP is that it enables a much closer collaboration between experimental and theoretical researchers, in order to obtain the maximum of information regarding the physical problem under study. These kinds of problem can be one-, two- or three-dimensional, linear or nonlinear, but the most important aspect is that, by not fulfilling all Hadamard conditions, they belong to a class called ill-posed, which determines the impossibility to find only and only one solution. The definition of a well-posed problem was originally introduced by Hadamard, in order to illustrate the points on which it is based on, the universal formulation of an inverse problem is assumed [17]:

$$Y = A(P) \in \mathbf{P}, Y \in \mathbf{Y}\tag{1.14}$$

where P is the unknown of interest (heat flux), that has to be estimated and Y (temperature) is the observed output. It is considered that the operator $A : \mathbf{P} \rightarrow \mathbf{A}$ is known, that $A\mathbf{P} \subset Y$ and \mathbf{P} and \mathbf{Y} are metric spaces.

A problem can be accepted to be well-posed if its solution satisfy the following three conditions:

- The solution of equation (1.14) must exist for any $Y \in \mathbf{Y}$;
- The solution of equation (1.14) must be unique;
- The solution of equation (1.14) must be stable under small changes to the input data (stability condition)

If only one of the conditions required for well-posedness is not satisfied, the problem would be unsolvable or the results obtained from such a solution would be meaningless. Due to this, the resolution of an inverse problem passes through a reformulation of the same as the approximate well-posed problem. The IHCP are ill-posed problems since conditions (2) and (3) are not fulfilled and thus finding a solution requires special techniques to be used. In this sense, A. N. Tikhonov's regularization procedure first and Alifanov's iterative regularization technique and Beck's function estimation approach than have contributed to develop extremely effective algorithms for resolving IHTP [16]. In all these cases the solution is found only if some information about physics of the problem are acquired and, hence, as said previously, considering the investigation on the heat flux in an heat conduction problem, it is reflected in the knowledge of temperature inside the structure of combustion chamber in some points over time, leading in this way to the importance of using thermocouples in the experiments. Furthermore, since an inverse method involves a large number of iterations and variables, it requires an huge computational cost, but the availability of high speed and capacity computers made successful solutions of IHCT being now reachable.

1.3 Why Oxygen/Methane Combustion

The use of hydrocarbons as fuels, especially for the propellant combination oxygen/methane, for rocket engines has been the subject of studies in Europe lately with the goal of employing it in future launchers [7]. Since is a field of research unexplored, a significant knowledge gap for these propellant combinations and a lack of test data of methane combustion tests at engine operating conditions exist. Nowadays, liquid propellant rocket engine for aerospace missions, like orbit transfers or space exploration, are principally

based on cryogenic propellants like liquid oxygen/liquid hydrogen (LO_x/LH_2), unsymmetrical dimethyl-hydrazine/nitrogen tetroxide ($UDMH/NTO$) and $LO_x/kerosene$ [14]. Main features of these propellants are summarised in table 1.1.

Table 1.1: Most propellants used in in early 2000s

<i>Combination</i>	<i>Pros</i>	<i>Cons</i>
LO_x/LH_2	highest specific impulse	low density and low boiling point
$UDMH/NTO$	hypergolic and storable	toxic, great pollution and low performances
$LO_x/kerosene$	low cost, low pollution and high performances	not usable in cooling system

Because of the low temperatures of cryogenic propellants (LO_x/LH_2), they are difficult to store over long periods of time and less desirable for use. Furthermore, this combination has a very low density and, therefore, requires a storage volume many times greater than other fuels. Despite these drawbacks, the high efficiency of liquid oxygen/liquid hydrogen makes these problems worth since the specific impulse is about 30% – 40% higher than most other rocket fuels [2]. The strengths of $UDMH/NTO$ propellants are based on their hypergolic and storable nature, but being toxic and causing great pollution make it unemployable in modern engines. However, $LO_x/kerosene$ seems to be a good compromise between its low costs, low pollution and high performance and although has some disadvantages in comparison to oxygen/methane combination it is still used in liquid rockets engines. Nevertheless, the idea of using hydrocarbon as a propellant, in particular, methane, is spreading especially for its [21]:

- reducing operational costs;
- low pollution;
- ease in handling;
- high thermal conductivity;
- and specific heat and low viscosity

Comparing methane and kerosene performances, it turns out that one of the most important limitation of the second one is its widely coking temperature (590 K) which unlike those of methane (970 K), makes it inoperable in cooling system [14]. Furthermore, recent research has shown that methane, compared to other hydrocarbons, exhibits both

an higher specific impulse and an improved heat transfer performance thanks to its high thermal conductivity, specific heat and low viscosity, as well as no risks for human health [7]. Since future missions to Mars would likely use methane fuel because it could be manufactured partly from Martian in-situ resources, and since this propellant has no flight history and very limited ground-test history [2], it will be subject to research in the next years at the Institute for Flight Propulsion (LTF) at the Technical University of Munich.

1.4 Scope of this thesis

In the present work, main issues affecting Inverse Heat Transfer Problems (IHTP) have been analyzed by applying an iterative technique based on Conjugate Gradient Method (CGM) in order to estimate heat flux passing through the hot gas walls of an oxygen/methane single and multi-injector combustion chamber. The experimental set up will be discussed in detail in chapter 2.

An optimization process is carried out on *RoqFITT* code (Rocket \dot{q} Flux Inverse Thermal Tool), with a view to reducing computational costs and oscillations of the solutions obtained. The code has been developed in Matlab environment by Perakis et al. specifically for single- and five-element combustion chambers. It implements the CGM, which is a direct descendant of the resolution proposed by Tikhonov, and like most methods for solving the inverse problem, it is based on the minimization of the ordinary least squares norm between measured and estimated temperature. Whereas the first are acquired by thermocouples spaced along the combustion chamber walls, the latter comes from the resolution of a three-dimensional direct transient heat conduction problem in which some of the boundary conditions are the unknown investigated heat fluxes. The experimental data investigated come from several tests carried out in TUM's laboratories on *MoRaP* (Mobile Raketen Prüfstand) and *BKM* chambers, respectively single and multi-coaxial shear injector chambers. In particular, this work is focused on the analysis of two tests that have been returning big oscillations in space and time, but since codes appear to respect roughly the expected results, the optimization process that has been applied and that will be reported in chapter 4, was aimed at improving the quality of the solution by reviewing the iterative procedure for updating the heat flux and by implementing the Newton-Raphson method (NRM). Results from modified codes have been compared with the original code, and subsequently, a numerical experiment has been performed to validate these modifications and generally to investigate the behavior of the CGM under specific inputs such as the way of smoothing thermocouple measurements or time step used for computing. Due to ill-posedness, in fact, taking into account that small changes

in temperature result in great changes of heat flux, the solution is particularly affected by thermocouples measurements, hence the acquisition system must be accurate and precise to ensure stability and to avoid oscillations. At this purpose, the concept of Sensitivity Matrix will be explained to show most important characteristics of CGM that prove also its weakness. Since an inverse method involves using a direct solver, in order to simplify the computation, an adiabatic boundary condition on the face plate between chamber and nozzle has been assumed in most of simulations. This assumption is far from the reality, hence a more accurate analysis has been necessary. Finally, since the Matlab code applied to multi-element is equal to the single-element one, it involves a larger number of thermocouple measurements, hence it is much more sensitive to the errors coming from data acquisition and/or interpolation methods. Therefore a study on the interpolation of the heat flux along the combustion chamber walls is necessary to contain them after that same modification mentioned earlier has been introduced.

Chapter 2

Single and Multi Element Hardware Description

The two tests investigated (*data_02-05-17_16-57_0*, *Data_CCC2D-20-26-25-01*) in this thesis has been conducted at two high pressure combustion test facilities of TUM-LFA, single-element and five-element combustion chambers, respectively. In this chapter a brief description of both hardware and measurement equipment is provided in accordance to [21] and [19].

2.1 Single-Element Combustion Chamber

The heat-sink chamber (MoRaP) is a modular single-element capacitive cooled combustion chamber with a square cross section. Its simplicity, low structural costs, ease to manufacture as well as high accessibility for measurement installation made it extremely effective for assisting LFA researches. The mobile facility is supported by two tanks (fuel and oxidizer), a pilot ignition system, a purge system and a control/data acquisition system (DAQ). While its modular layout allows more than one configuration, giving the possibility to choose the best one depending on the experiment conducted, its simple rectangular geometry makes boundary conditions to be straightforwardly modeled. Furthermore, MoRaP, despite its essential set-up and minimal instrumentations, provides for experimental data very similar to those of more complex combustion chambers. Generally, the combustion temperature reached inside the chamber cannot exceed a certain value otherwise a reduction of the nozzle throat is caused, for this reason, the pressure, the mixture ratio and the burning times must be contained. In fact, it has been designed for testing time up to 4 s at chamber pressure up to 20 *bar* as well as mixture ratio up to 3.4, in this way no special internal coatings or cooling systems are necessary. As can be seen

in Fig. 2.1(a) the MoRap is composed of three different segments, the first two measure 174 mm and 145 mm and form the real zone where combustion develops, the third is the nozzle segment, it differs from the others for its rectangular throat cross section 4.8 mm x 12 mm , the whole inner chamber dimensions are summarised in table 2.1. The external structure measures 50 mm in height and 85 mm in width.

Table 2.1: Combustion chamber dimensions

<i>Chamber length</i>	[mm]	290
<i>Chamber width</i>	[mm]	12
<i>Chamber height</i>	[mm]	12
<i>Throat height</i>	[mm]	4.8
<i>Throat width</i>	[mm]	12
<i>Contraction ratio</i>	[-]	2.5

On the basis of its several advantages, the material selected for construction of both combustion chambers is oxygen-free copper ($Cu - HCP$):

- good electrical and thermal conductivity;
- excellent corrosion resistance and formability;
- good weld ability and recyclable;

Its material properties are shown in table 2.2.

Table 2.2: Material properties

<i>Thermal conductivity</i>	λ	385	[W/mK]
<i>Density</i>	ρ	8940	[Kg/m ³]
<i>Specific heat capacity</i>	c_p	393	[J/KgK]

As reported in [21] more than one injector configuration is possible, but for the test considered, a single shear coaxial injector element is integrated as shown in Fig. 2.1(b), it consists of:

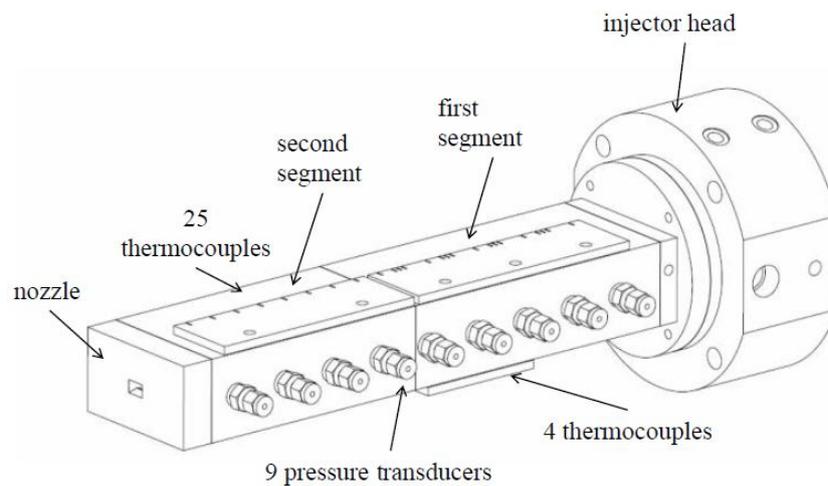
- an inner channel with 4 mm diameter, from which gaseous oxygen at 278 K and at velocities of about 122 m/s is expelled;
- a 0.5 mm annular gap from which gaseous methane at 269 K and at velocities of about 122 m/s enters the combustion chamber [9]

In order to simplify the analysis, the GO_2 post has been flush mounted with respect to the injection faceplate (no recess). Furthermore, to ensure homogeneous injection conditions in terms of temperature and pressure, MoRaP provides for two porous plates placed in the oxidizer and fuel manifolds.

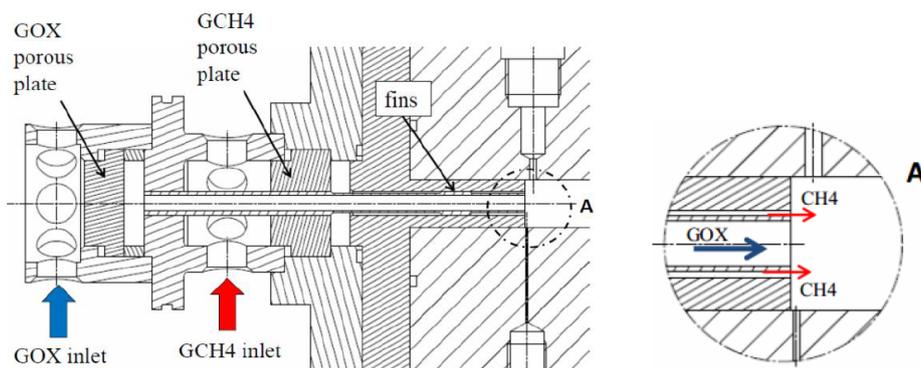
Nondimensional numbers such as the velocity ratio VR and the momentum flux ratio J can be introduced to characterize injection conditions:

$$VR = \frac{v_{GCH_4}}{v_{GO_x}} \quad ; \quad J = \frac{(\rho v^2)_{GCH_4}}{(\rho v^2)_{GO_x}} \quad (2.1)$$

The values of these quantities range from 0.89 to 1.1 and from 0.38 to 0.62, respectively, and are based on propellant temperatures and pressure at injection conditions.



(a) Single Element Components



(b) Injection System

Figure 2.1: MoRaP Combustion Chamber

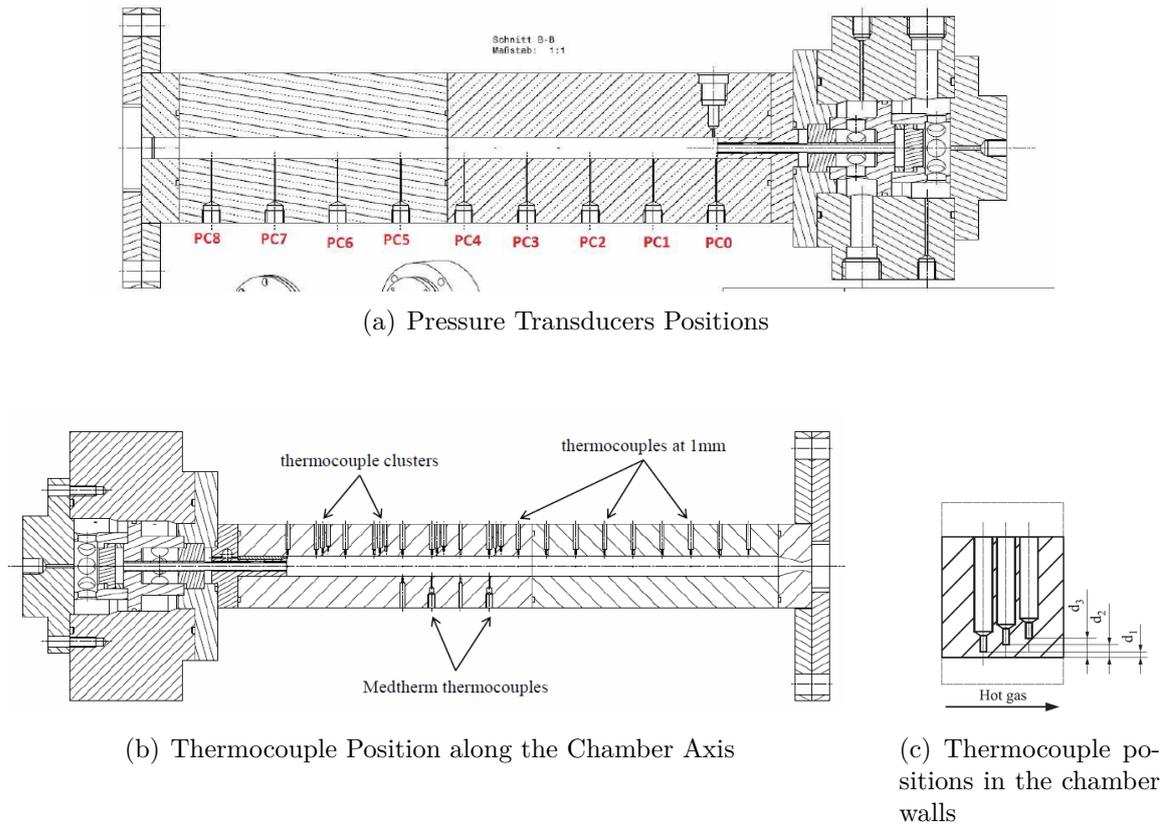


Figure 2.2: Measurement Systems

Reference System

As said in the previous chapter the conjugate gradient method requires that throughout the optimization process several direct problems must be solved. To do this a Finite Difference Method (FDM) has been adopted and thus it is necessary to introduce the reference system on which the discretization has been based. Figure 2.3 shows that its origin is set on the left down corner at nozzle entry cross-section, the x axis extends in latitude and y axis in longitude. From now on all sensor positions refer to this reference system. Moreover, figure 2.3 faithfully reports also how the combustion chamber has been modeled for resolution of the direct problem. Note that the domain has been extremely simplified, in fact, while the whole injector system together with all its particular components has been replaced with a simple copper block, both nozzle and all drill holes (the ones for the temperature sensors, those for the pressure transducers as well as the openings for the torch-igniter) have been excluded from modeling. This simplification is supported by the fact that the error resulting from excluding the mentioned parts mostly affects the small-scale temperature results in the far-field which have only very little influence on the heat flux values in the

region of interest as well as by the fact that the holes have a volume fraction of less than 4.907% of the total control volume [17].

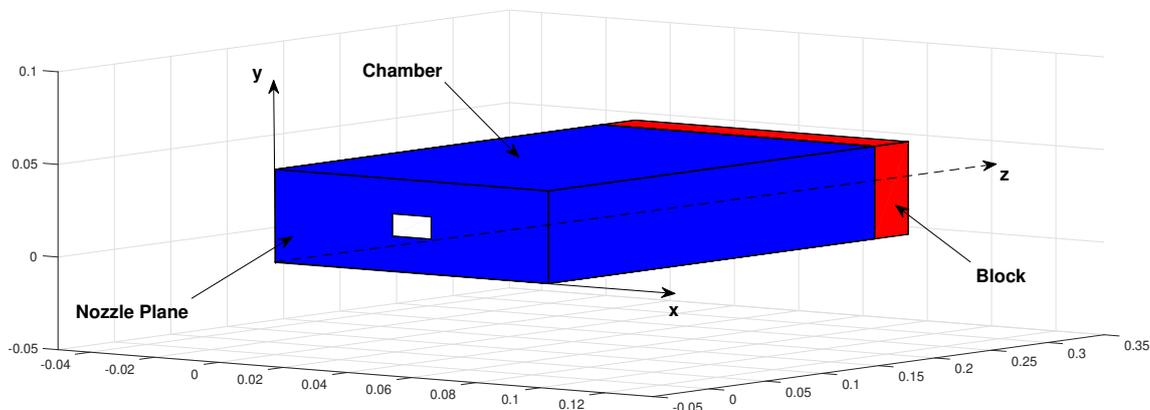


Figure 2.3: Modelled Combustion Chamber

2.1.1 Sensors

This section describes the two measurement systems used during the test. In particular, since the CGM requires only transient temperature measurements for IHCP resolution, the temperature acquisition system has been described in more detail than the pressure one. Hence a particular attention has been paid to thermocouples features as well as to their arrangement.

2.1.1.1 Pressure Transducers

Since the knowledge of pressure is generally essential for better understanding a combustion process as well as for other types of analysis, equally spaced pressure transducers (WIKA A10, Fig. 2.2(a)) on the side wall provide for a well-resolved measurement of the wall pressure distribution $p(x)$ along the chamber axis (PC0. . .PC8). These sensors are individually calibrated and operated at a data acquisition rate of 100 Hz.

2.1.1.2 Thermocouples

The temperature acquisition system hardware includes the use of following thermocouples provided by Electronic Sensor company:

- type T thermocouples with 0.5 mm diameter located in the chamber walls at 1-, 2-, and 3-millimeter distance to the hot wall;

- two coaxial Type T thermocouples (Medtherm) flush mounted with the hot wall;
- Type K surface thermocouples attached to the external surface

Type T thermocouples at 1 *mm* from hot gas wall are the only ones used in this particular test investigated, their main features are provided in the following table

Table 2.3: Type T thermocouples features

<i>Material</i>	<i>Cu/CuNi</i>
<i>Class</i>	2
<i>Temperature Range</i> [$^{\circ}C$]	-40 to 350
<i>Standard deviation</i> [$^{\circ}C$]	± 1

All sensors are mounted with a regular path of 17 *mm* in the upper surface of the first and second segments, along the center plane of the combustion chamber. Their location is given table 2.4 and can be appreciated in figure 2.2(b) and 2.2(c).

Table 2.4: Thermocouples positions

<i>Thermocouples Names</i>	x [m]	y [m]	z [m]
<i>TCLU01</i>	0.0425	0.032	0.289
<i>TCLU11</i>	0.0425	0.032	0.272
<i>TCLU21</i>	0.0425	0.032	0.255
<i>TCLU31</i>	0.0425	0.032	0.238
<i>TCLU41</i>	0.0425	0.032	0.221
<i>TCLU51</i>	0.0425	0.032	0.204
<i>TCLU61</i>	0.0425	0.032	0.187
<i>TCLU71</i>	0.0425	0.032	0.170
<i>TCLU81</i>	0.0425	0.032	0.153
<i>TCSU01</i>	0.0425	0.032	0.136
<i>TCSU11</i>	0.0425	0.032	0.119
<i>TCSU21</i>	0.0425	0.032	0.102
<i>TCSU31</i>	0.0425	0.032	0.085
<i>TCSU41</i>	0.0425	0.032	0.068
<i>TCSU51</i>	0.0425	0.032	0.051
<i>TCSU61</i>	0.0425	0.032	0.034
<i>TCSU71</i>	0.0425	0.032	0.017

In accordance to [18] the nomenclature shown in table 2.5 is used in RoqFITT code to identify each thermocouple position.

Table 2.5: Thermocouples nomenclature

T	Thermocouple
C	Chamber not water cooled
L/S	First segment (long) otherwise S second segment (short)
U/D	Upper side otherwise down side
<i>First digit</i>	Axial position, the first segment goes from 0 to 8, the second from 0 to 7
<i>Second digit</i>	Distance from the hot gas side (1, 2 or 3 mm)

From figure 2.2(c) one can note that each thermocouple is inserted in precisely manufactured cylindrical holes and furthermore, in order to ensure better contact, they are kept in position by a spring loaded system as well as their tip is polished to match the flat surface of the chamber. The spring loading of all thermocouples provides a constant force of about 2 N, this should guarantee a continuous contact between the thermocouples tip and the base of the hole, leaving their position unaltered. In reality, as will be better seen in chapter 5, despite these precautions aim to prevent the chance of potential loss of contact as the material undergoes expansion and contraction due to changes in temperature or vibrations during the hot run, it is possible that their position changes slightly.

2.2 Multi-Element Combustion Chamber

The RoqFITT code has also been developed to solve the heat flux transfer problem inside the BKM chamber (Fig. 2.4). This chamber represents an evolution of the previous one, its structure made slightly more complex by the presence of five injectors enables to examine heat transfer problems closer to those of the most sophisticated modern propulsion systems. The material used as said previously is oxygen-free copper. Its rectangular geometry, also in this case, allows to simply model the boundary conditions. The reference system is the same one adopted for the single-element, its origin is set on the left down corner at nozzle entry cross-section, the x axis extends in latitude and y axis in longitude. Hence, also in this case, all thermocouples positions refer to it. The modeling of the entire chamber is similar to that carried out on single-element, i.e. the same parts have been excluded making discretization less complex. The BKM chamber has a rectangular cross-section with dimensions 48 mm x 12 mm, a length of 277 mm and a contraction ratio of 2.5 ensuring in this way a Mach number of approximately 0.24, typical for rocket engine applications. The shape of the nozzle is a truncated trapezoidal prism with a rectangular throat section of 4.8 mm x 48 mm [19].

The injector system, visible from the figure 2.4 in the right, is made up of five identical coaxial injector elements placed next to each other along a line. The thermocouples are

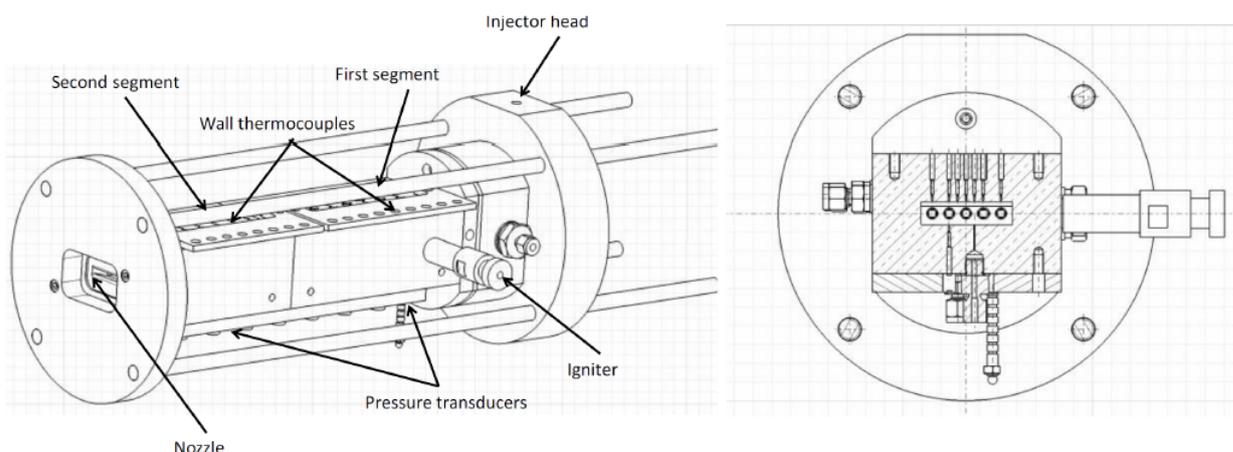


Figure 2.4: Multi-Element Combustion Chamber and Thermocouples Arrangement in a Plane

numbered from 1 to 5 starting from the left and going to the right. The distance between the centers of injectors circle cross-section is 6 mm and the injector-wall distance is 3 mm . Like in the single-element also this chamber is equipped with two porous plates, placed in the oxidizer and fuel manifolds respectively, to ensure homogeneous injection conditions. The entire instrumentation used for temperature and pressure acquisition is the same one discussed previously, however, the arrangement of both thermocouples and pressure transducers changes. Also in this case, only type T thermocouples positioned at 1 mm away from the hot gas wall with an axial resolution of 17 mm . From the chamber cross-section (Fig. 2.4) it is possible to appreciate the maximum number of thermocouple (7) that can be installed in each plane, while those directly placed on the injector are marked with an index C , the others mounted between two elements are identified with index L or R depending on whether they are left or right of vertical symmetry axis. Hence, the names and positions of the 7 thermocouples in a plane given in accordance to the reference system are shown in table 2.6).

Table 2.6: Thermocouples position

<i>Thermocouple name</i>	1C	2C	3L	3C	3R	4C	5C
<i>x Position [mm]</i>	32.0	41.0	45.5	50.0	54.5	59.0	68.0

The total number of thermocouples used in the investigated test is 66 and they are exclusively arranged along the combustion chamber upper wall, figure 2.5 shows their positions. The use of such a large number of sensors is necessary for the calculation process, since the presence of 5 injectors develops a more complex heat transfer compared to the single injector chamber.

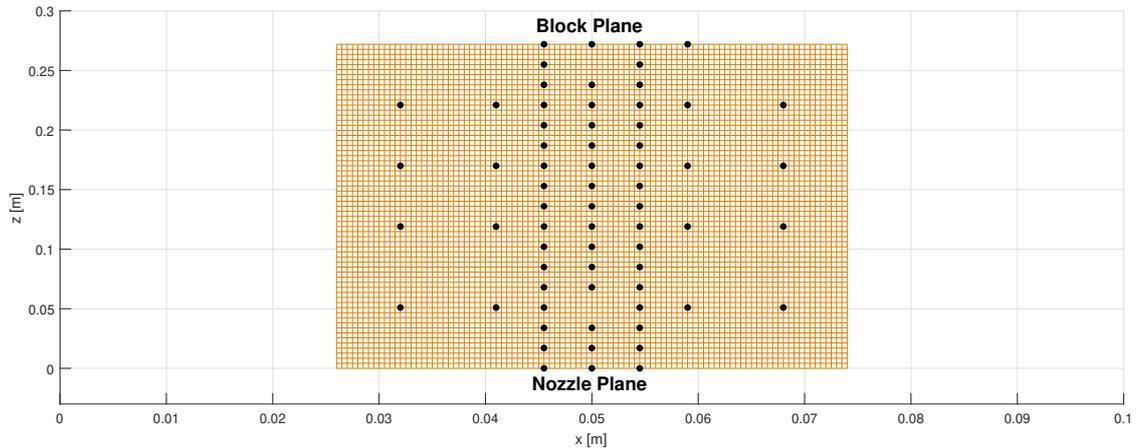


Figure 2.5: Thermocouples Positions above the Upper Hot Gas Wall

Unlike the single element, the experimental data acquisition system requires that both the thermocouples and the pressure transducers are connected to a *PCI eXtensions for Instrumentation* or *PXI*, that is an electronic modular platform used as a basis for building electronic test equipment as well as automation systems. In particular, of 15 available modules (M1, M2,..., M15), only 4 have been used for the acquisition (M5, M6, M8 and M9) of the temperature with a scan rate set at 90 Hz.

2.3 Experimental Inputs and Output

This section provides a picture of inputs and outputs of the two tests investigated. Table 2.7 shows some tests features such as the acquisition and evaluation time as well as the mixture ratio. Some of these parameters (e.g. those for time management) can be set and controlled by a code written in LabVIEW environment, others (e.g. mixture ratio) instead can be set manually by MoRaP facility, better explanations are provided in [12]. Since both single- and five-element outputs are similar, the considerations that can be made for one also apply to the other, hence, the following figures 2.6 and 2.7 show just classical single-element thermocouple/pressure transducer outputs. The evaluation phase of the unsteady heat-transfer starts at a chamber pressure of 90% of its stationary value, shown as t_{start} and finishes at the end of the stationary combustion, shown as t_{end} . For both combustion chambers, the wall temperature increases over time and the slopes of the wall temperatures as well as heat flux profiles vary with the time of firing. Hence, the heat flux changes accordingly and the local wall heat flux results need to be considered along with the corresponding local wall temperatures. Towards the end of the firing time a quasi-stationary combustion is reached then evaluating the heat flux at t_{eval} yield to

Table 2.7: Ignition condition

<i>Chamber</i>	Single-Element	Multi-Elements	
<i>Data</i>	<i>data_02 - 05 - 17_16 - 57_0</i>	<i>Data_CCC2D - 20 - 26 - 25 - 01</i>	
<i>Acquisition time</i>	18.5939	35.0777	[s]
<i>Start ignition</i>	5.3165	21.4556	[s]
<i>End ignition</i>	8.3548	25.4778	[s]
<i>Burning time</i>	3.0383	4.0222	[s]
<i>Evaluation time</i>	7.3421	25.286	[s]
<i>Mixture ratio</i>	2.9822	2.6213	[-]
<i>O₂ Mass flow rate</i>	0.0463	0.1772	[Kg/s]
<i>CH₄ Mass flow rate</i>	0.0155	0.0677	[Kg/s]
<i>Cooling system</i>	-	off	[-]

better results and for this reason, it is chosen as reference for the current study. Note that generally the burning time never exceeds 4 s and that the temperature reached do not overcome 440 K in single-element and 573 K in five-element chamber remaining always far from oxygen-free copper melting point (1356 K). Figure 2.6 not only shows that, once ignition starts, in each transducer position pressure rises rapidly and then remain at a constant value until shut down, but also that pressure drops slightly along chamber axis. Figure 2.7, instead, shows that starting from ignition in each thermocouple position temperature rises and then once chamber is shut down it decreases slowly. Additionally, an increase in temperature readings is detected by moving along chamber axis, this is due to the fact that chemical reaction of propellants is complete towards the nozzle entry section. In the same figure, from the zoomed plot it is possible to appreciate also the causes of one of the most important problems affecting a CGM, i.e. oscillations in temperature readings with a magnitude equal to thermocouples standard deviation can lead to uncertainty in the solution, i.e. heat flux oscillations too. This aspect will be further explored in chapter 5.

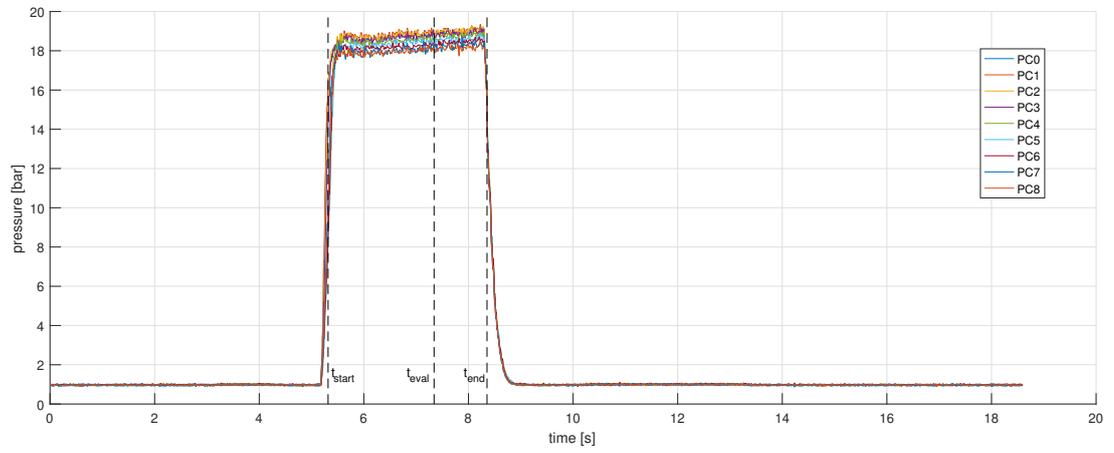


Figure 2.6: Transient Pressure Measurements

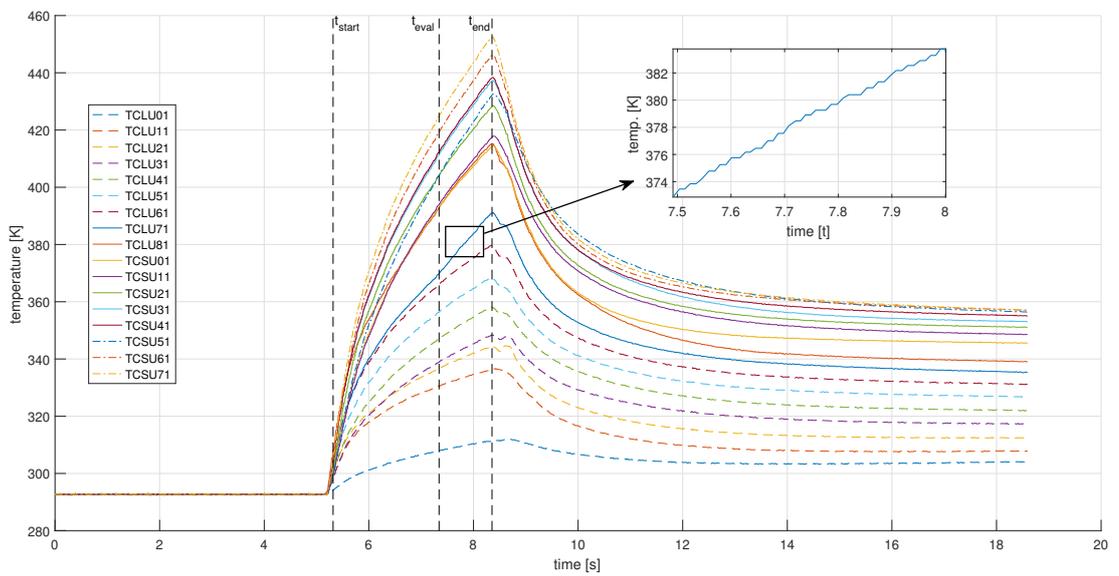


Figure 2.7: Transient Temperature Measurements

Chapter 3

Inverse Method

This chapter aims to explain in detail how an inverse method works as well as to provide additional informations about each step that characterizes it. In particular, as said in chapter 1, since an inverse method can be used for different types of estimations, among all available methods listed in [16], the Conjugate Gradient Method (CGM) is the optimization technique chosen and implemented in RoqFITT code for estimating the heat flux along combustion chamber internal walls. It is an iterative method for solving linear and non-linear inverse problems. In figure 3.3 there is a clear scheme explaining the steps in which this method is divided:

- The Direct Problem;
- The Inverse Problem;
- The Iterative Procedure;
- The Stopping Criterion;

In order to find a solution for the Inverse Heat Conduction Problem (IHCP), the CGM carries out an iterative optimization to the unknown boundary conditions. More specifically, the heat flux resulting from the combustion and passing through the surface of the hot-gas-wall is the only one boundary condition optimized iteratively. The all other boundary conditions remain unaltered throughout the whole process as well as thermophysical properties of the material. The problem is discretized in the spatial and time domain. The time domain is divided into I discrete steps. Each timestep consists in several iterations in which the heat flux is updated till the matching of measured and computed temperature is satisfied. While the former comes from thermocouple readings, the latter comes from the solution of a direct problem in which the optimized heat flux is set as the missing boundary condition. A peculiarity of this method is that instead of computing the heat flux for every

node of the hot wall boundary, the heat flux is discretized in just N parameters. Hence the inverse problem estimates just N unknown parameters, because they are, then, interpolated at all other points in order to be used as input for direct problem. Whenever the convergence is reached, the optimization process switches to another time step. Whether in the iterations and time steps the first guess for the estimated heat flux is based on the previous solution, allowing in this way a faster convergence of the optimization procedure. The figure 3.3 gives an idea of how the separate time steps are linked together. At the beginning of the process this method, like most inverse methods, requires two informations: an initial guess for heat flux and the initial medium temperature for solving the first direct problem. The robustness of this method allows to set any initial heat flux value, in fact, the convergence is reached in any case, but obviously, a better initial guess leads to faster convergence. The initial temperature is, instead, taken from thermocouple readings. At this point the first direct problem is solved and once the temperature field is known, it is compared with measured temperature in thermocouple positions, if the convergence is reached another time step is switched and the heat flux from the previous solution is set as the initial guess, otherwise an iterative procedure starts to update the heat flux on boundary wall. Figure 3.4 shows in detail how this iterative procedure works. First of all, a variable, k , counts the number of performed iterations, it is reset to 1 whenever a time step is switched, the N parameters are collected in a vector \mathbf{P}^k and applied to the direct problem, the stopping criterion that will be discussed below determines if a new update is necessary, if that's the case the actual iterative procedure starts by computing sequentially the Gradient Direction (GD), the Conjugation Coefficient (CC), the Direction of Descent (DD) and the Step Size (SS). Hence as will be seen below, by combining them a new estimate of heat flux is available \mathbf{P}^{k+1} and set as boundary condition for another direct problem, from which new temperatures will be available. These temperatures will be again compared with measured ones, and the procedure follows the same path. The whole procedure just introduced will be discussed in detail below.

3.1 The Direct Problem

As, said before the process of the estimation of the N parameters heat flux passes through the resolution of a heat conduction direct problem, whose main objective is to compute the transient temperature field $T(x, y, z, t)$ into a solid when all characteristics(i.e., boundary conditions BC, initial condition, thermophysical properties of the medium and energy generation term) are specified. Once the N parameters heat flux are updated by the iterative process, they are interpolated and applied as the inner boundary condition to

the direct conduction problem. As said before the computed temperatures $T_{calc} \equiv T$ have to match the measured temperatures $T_{meas} \equiv Y$ in each thermocouples positions $(x_{meas}, y_{meas}, z_{meas})$ at all times $t_i, i = 1, 2, \dots, I$ and if this matching is not satisfied, a new iteration starts. At this point, it is important to introduce the mathematical formulation for a three-dimensional conduction problem in a Cartesian reference system, known as the Heat Diffusion Equation (HDE), on which the direct solver is based.

$$\frac{\partial}{\partial x} \left(\lambda \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y} \left(\lambda \frac{\partial T}{\partial y} \right) + \frac{\partial}{\partial z} \left(\lambda \frac{\partial T}{\partial z} \right) + \dot{q} = \rho c_p \frac{\partial T}{\partial t} \quad (3.1)$$

In this particular examined case, some simplifications can be applied. The thermal conductivity λ is assumed constant and no source terms \dot{q} are present, in this way the heat diffusion equation is reduced to:

$$\nabla^2 T = \frac{1}{\alpha} \frac{\partial T}{\partial t} \quad (3.2)$$

where $\alpha = k/\rho c_p$ is the thermal diffusivity. This partial differential equation (PDE) (3.2) has been solved in RoqFiTT code by using a Finite Difference Method (FDM) forward in time central in space. More details will be provided below. The domain concerning this equation has been introduced in chapter 2 (fig. 2.3), it is extremely simplified to save high computational costs and to make discretization more immediate.

3.1.1 Boundary Conditions

This small section wants to explain the boundary conditions necessary to close the problem and to justify convective coefficients used in both direct solvers:

- Natural convection on outer walls;
- Forced convection on inner walls;
- Forced convection on cooling channel walls;
- Adiabatic block and nozzle planes

3.1.1.1 Natural Convection on Outer Walls

The outer wall of the chamber is subjected to natural convection, since it is in contact with quiet atmosphere. The following proposed study has been conducted in [18] for evaluating convection coefficient h_∞ . First, the Grashof number has been evaluated, for a typical

wall temperature range of $300 \div 600 \text{ K}$, by considering a temperature of the air at 20°C and a typical length equal to 50 mm as the outer vertical MoRaP wall:

$$Gr = \frac{(T_{wall} - T_\infty)gL^3}{T_\infty\nu^2} = 130205 \div 5450000 \quad (3.3)$$

Then, the Rayleigh number has been calculated, by assuming a Prandtl number equal to 0.7:

$$Ra = Gr \cdot Pr = 91143 \div 3815000 \quad (3.4)$$

Since the Rayleigh number Ra is less than 10^9 , the flow appear to be laminar and hence the calculation of Nusselt number is performed with following formula:

$$Nu = 0.68 + \left[1 + (2 \cdot Pr)^{-\frac{9}{16}}\right]^{-\frac{4}{9}} \cdot Ra^{\frac{1}{4}} = 13.97 \div 34.85 \quad (3.5)$$

which leads to:

$$Nu = \frac{h \cdot L}{\lambda} \rightarrow h_\infty = 7.2 \div 18 \frac{W}{m^2 K} \quad (3.6)$$

The analysis is quite similar for five-element since the geometry varies slightly, hence, the convection coefficient has been chosen equal to $10 \frac{W}{m^2 K}$ in both codes and it is kept constant throughout the optimization process.

3.1.1.2 Forced Convection on Inner Walls

This boundary condition, as said before, is the only one unknown that has to be estimated. The conjugate gradient method doesn't need any previous estimation in terms of convection coefficients, since it returns, at the end of each iteration, the N parameters heat flux values investigated. However, a small description of the aerodynamics and thermodynamics in accordance to [18] is given just to understand the physic of the problem. Inside the combustion chambers, where oxidizer and methane burn together, the heat transfer on the wall occurs via forced convection. Considering that the combustion may be incomplete, the mixture can be either in gas state or in liquid state, and the heat transfer is hard to describe. In general, the most important factors influencing the heat transfer in a combustion chamber are the chamber pressure, the geometry factors (injectors, igniter and chamber walls shape), type and speed of chemical reactions, turbulence and radiation. Cinjarew formula shows how each factor affects the convection coefficient, i.e. the heat transfer:

$$h = 0.0198 \frac{\lambda_{gas}(\dot{m}c_p)^{0.82}}{d^{1.82}} \left(\frac{T_{aw}}{T_w} \right)^{0.35} \quad (3.7)$$

where λ_{gas} is the heat conduction coefficient of the gas close the wall, c_p is the specific heat capacity of gas mixture, \dot{m} is the mass flow rate, d is the radius of the chamber, T_w is the wall temperature and T_{aw} is the recovery temperature of the hot gas side. From the formula (3.7), it is clear that the heat convection coefficient increase after the combustion has developed, i.e. after small distance from the injectors. For a combustion chamber with a constant radius, when the combustion can be considered complete, the heat transfer coefficient is constant along the length in first approximation. The examined chambers have square or rectangular cross-sections, thus, a relatively constant heat flux is expected after the point of optimal mixture is achieved (logarithmic trend).

3.1.1.3 Forced Convection on Cooling Channels Walls

The five-element combustion chamber also provides for the use of water cooling channels, hence, there is another forced convection that takes place into the analysis. In the present work, neither for single-element nor for five-element chamber analysis, the cooling system has been used, but a brief description of the physic of the problem is given in accordance to [18]. Also in this case the mechanism of heat transfer between copper wall and liquid medium absorbing the heat is described by several semi-empirical formulas. In this case, the convection coefficient is a function of medium (water/chopper walls) temperatures, as well as water mass flow and channels dimensions. The empirical Gnielski (3.8) and Dittus-Bölder (3.9) relationships express well these dependences:

$$Nu = \frac{\frac{\xi}{8}(Re - 1000)Pr}{1 + 12.7\sqrt{\frac{\xi}{8}}(Pr^{\frac{2}{3}} - 1)} \quad (3.8)$$

where $\xi = (0.79 \cdot \ln(Re) - 1.64)^{-2}$ is the loss term according to Petukhov.

$$Nu = 0.0223 Pr^{\frac{1}{3}} Re^{\frac{4}{5}} \quad (3.9)$$

The convection coefficient h_{water} can be estimated by knowing the Reynolds and Prandtl numbers of the water flowing in the channels. The more conservative formula (3.9) has been chosen for the purposes of these simulations, since it returns a smaller value of the convection coefficient and hence a less efficient cooling of the material:

$$h_{cooling} = 20000 [W/m^2K] \quad (3.10)$$

3.1.1.4 Adiabatic Walls

As mentioned previously, these boundary conditions are the result of the discretization. The only two boundary walls that have been assumed adiabatic in each direct problem are the copper block faceplate in contact with the atmosphere and the plane separating the actual chamber structure and nozzle block (fig. 2.3).

3.1.2 Finite Difference Method for 3D Problems

Below it is presented how heat diffusion equation has been discretized for several particular nodes. In a finite difference method the partial derivatives presents in a differential equation are approximated through differences of the variable's value in specific points. This expedient brings to define a large but finite algebraic system of equations that replaces the differential heat diffusion equation and that can be performed on a computer. Each finite-difference equations have been developed by using the energy balance method, i.e. by applying the conservation of energy to a control volume about the nodal region [5]. Since the actual direction of heat flux (into or out of the node) is unknown, it is suitable to formulate the energy balance by assuming that all the heat flux is into the node. Take as an example the discretization carried out on a generic internal node (3.1), the heat diffusion equation becomes the following algebraic expression:

$$T_{m,n,p}^{k+1}(1 + 2F_{O_z} + 4F_{O_x}) - F_{O_x}T_{m-1,n,p}^{k+1} - F_{O_x}T_{m,n+1,p}^{k+1} - F_{O_x}T_{m+1,n,p}^{k+1} - F_{O_x}T_{m,n-1,p}^{k+1} - F_{O_z}T_{m,n,p+1}^{k+1} - F_{O_z}T_{m,n,p-1}^{k+1} = T_{m,n,p}^k \quad (3.11)$$

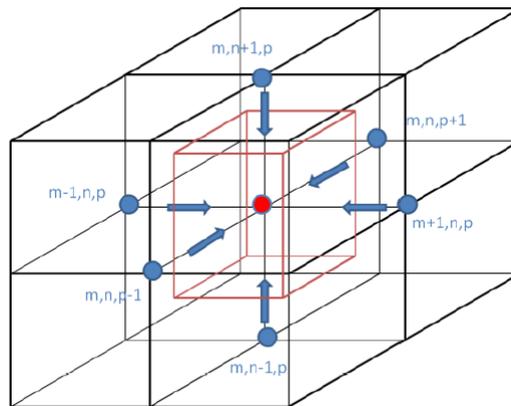


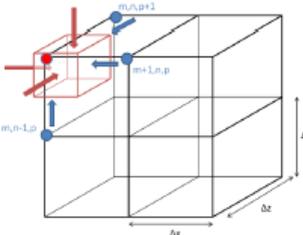
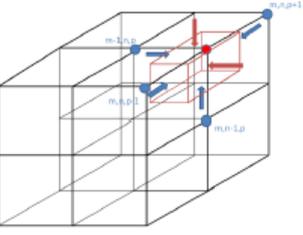
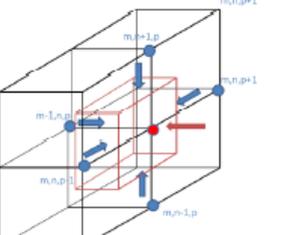
Figure 3.1: Discretization for an Internal Node [18]

Fourier numbers appearing in the equation (3.11) are different according to cartesian direction considered:

$$Fo_x = Fo_y = \frac{\lambda \Delta t}{\rho c_p \Delta x^2} \quad , \quad Fo_z = \frac{\lambda \Delta t}{\rho c_p \Delta z^2} \quad (3.12)$$

Note that the Fourier number is the same for x and y direction since the discretization has been carried out with same spacial step ($dx = dy$). The other algebraic equations as product of discretization are presented below in figure 3.2, they refers to specific nodes. Note that Biot numbers appear whenever an equation is related to a boundary node:

$$Bi_x = Bi_y = \frac{h \Delta x}{\lambda} \quad , \quad Bi_z = \frac{h \Delta z}{\lambda} \quad (3.13)$$

Node	Geometric Configuration	Formula
Outside Corner, First plane		$T_{m,n,p}^{(k+1)} (1 + 2Fo_z Bi_z + 4Fo_x Bi_x + 2Fo_z + 4Fo_x)$ $- 2Fo_x T_{m+1,n,p}^{(k+1)}$ $- 2Fo_x T_{m,n-1,p}^{(k+1)}$ $- 2Fo_z T_{m,n,p+1}^{(k+1)}$ $= T_{m,n,p}^{(k)}$ $+ 2Fo_z Bi_z T_{\infty}$ $+ 4Fo_x Bi_x T_{\infty}$
Outside Corner, Middle plane		$T_{m,n,p}^{(k+1)} (1 + 4Fo_x Bi_x + 2Fo_z + 4Fo_x)$ $- 2Fo_x T_{m-1,n,p}^{(k+1)}$ $- 2Fo_x T_{m,n-1,p}^{(k+1)}$ $- Fo_z T_{m,n,p+1}^{(k+1)}$ $- Fo_z T_{m,n,p-1}^{(k+1)}$ $= T_{m,n,p}^{(k)}$ $+ 4Fo_x Bi_x T_{\infty}$
Outside/Inside Middle plane		$T_{m,n,p}^{(k+1)} (1 + 2Fo_x Bi_x + 2Fo_z + 4Fo_x)$ $- 2Fo_x T_{m-1,n,p}^{(k+1)}$ $- Fo_z T_{m,n-1,p}^{(k+1)}$ $- Fo_z T_{m,n,p+1}^{(k+1)}$ $- Fo_x T_{m,n+1,p}^{(k+1)}$ $- Fo_z T_{m,n,p-1}^{(k+1)}$ $= T_{m,n,p}^{(k)}$ $+ 2Fo_x Bi_x T_{\infty}$

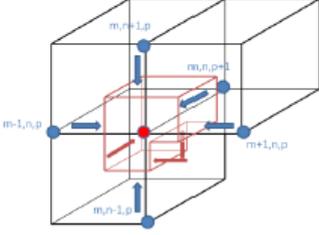
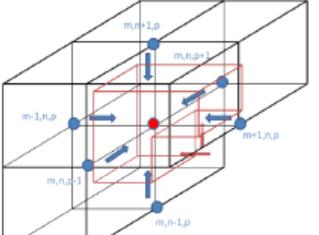
Inside Corner, First plane		$ \begin{aligned} T_{m,n,p}^{(k+1)} & \left(1 + 2Fo_z Bi_z + \frac{4}{3} Fo_x Bi_x + 2Fo_x \right. \\ & \left. + 4Fo_x \right) \\ & - \frac{4}{3} Fo_x T_{m-1,n,p}^{(k+1)} \\ & - \frac{4}{3} Fo_x T_{m,n+1,p}^{(k+1)} \\ & - \frac{2}{3} Fo_x T_{m+1,n,p}^{(k+1)} \\ & - \frac{2}{3} Fo_x T_{m,n-1,p}^{(k+1)} \\ & - 2Fo_z T_{m,n,p+1}^{(k+1)} \\ & = T_{m,n,p}^{(k)} \\ & + 2Fo_z Bi_z T_{in} \\ & + \frac{4}{3} Fo_x Bi_x T_{in} \end{aligned} $
Inside Corner, Middle plane		$ \begin{aligned} T_{m,n,p}^{(k+1)} & \left(1 + \frac{4}{3} Fo_x Bi + 2Fo_x + 4Fo_x \right) \\ & - \frac{4}{3} Fo_x T_{m-1,n,p}^{(k+1)} \\ & - \frac{4}{3} Fo_x T_{m,n+1,p}^{(k+1)} \\ & - \frac{2}{3} Fo_x T_{m+1,n,p}^{(k+1)} \\ & - \frac{2}{3} Fo_x T_{m,n-1,p}^{(k+1)} \\ & - Fo_z T_{m,n,p+1}^{(k+1)} \\ & - Fo_z T_{m,n,p-1}^{(k+1)} \\ & = T_{m,n,p}^{(k)} \\ & + \frac{4}{3} Fo_x Bi_x T_{in} \end{aligned} $

Figure 3.2: Discretization for Particular Nodes [18]

3.2 The Inverse Problem

The update of the N parameters heat flux starts when the difference between the computed temperatures $T_{calc} \equiv \mathbf{T}$, evaluated exactly at the M thermocouple positions $(x_{meas}, y_{meas}, z_{meas})$, and the measured temperatures $T_{meas} \equiv \mathbf{Y}$ can reach local minima. Hence to reduce this difference a minimization of the ordinary least squares norm, the residual functional, is performed:

$$S(\mathbf{P}) = \sum_{i=1}^I [Y_i - T_i(\mathbf{P})]^2 = [\mathbf{Y} - \mathbf{T}(\mathbf{P})]^T [\mathbf{Y} - \mathbf{T}(\mathbf{P})] \quad (3.14)$$

where:

S is the sum of squares error;

$\mathbf{P} = [P_1, P_2 \dots, P_N]$ is the vector of unknown parameters;

$\mathbf{T}(\mathbf{P})$ are the estimated temperatures at the thermocouples positions at time t_i ;

\mathbf{Y} are the measured temperature by thermocouples at time t_i ;

N is the total number of unknown parameters;

I is the total number of measurements;

k is the number of the iteration.

In those particular cases where the M sensors located at x_{meas} , y_{meas} , z_{meas} acquire at high frequency, the measured temperatures can be approximated as continuous and the residual functional (3.14) can be modified in the following way:

$$S(\mathbf{P}) = \int_{t=0}^{t_f} [Y(t) - T(x_{meas}, y_{meas}, z_{meas}, t; \mathbf{P})]^2 dt \quad (3.15)$$

3.3 The Iterative Procedure

Whenever the discrepancy principle, which will be discussed in section 3.4 indicates that another iteration is necessary, the actual updating process starts (fig.3.4). To minimize the least squares given by equation (3.15), the N parameters \mathbf{P} are varied in the following way:

$$\mathbf{P}^{k+1} = \mathbf{P}^k - \beta^k \mathbf{d}^k \quad (3.16)$$

where $k + 1$ is the next iteration, β^k is the step width and \mathbf{d}^k is the direction of descent and both are computed by using the Sensitivity Matrix (SM).

3.3.1 The Sensitivity Concept

The Sensitivity Concept is introduced in order to measure the sensitivity of the estimated temperature \mathbf{T} with regard to changes in the parameter P_j with $j = 1, 2, \dots, N$. This step is accomplished by varying one by one all N parameters and by computing the variation of temperature dT that they cause in each thermocouple position. The results are expressed in a compact form, i.e. the Sensitivity/Jacobian Matrix, $\mathbf{J}(\mathbf{P})$:

$$\mathbf{J}(\mathbf{P}) = \begin{bmatrix} \frac{\partial T_1}{\partial P_1} & \frac{\partial T_1}{\partial P_2} & \frac{\partial T_1}{\partial P_3} & \cdots & \frac{\partial T_1}{\partial P_N} \\ \frac{\partial T_2}{\partial P_1} & \frac{\partial T_2}{\partial P_2} & \frac{\partial T_2}{\partial P_3} & \cdots & \frac{\partial T_2}{\partial P_N} \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \frac{\partial T_M}{\partial P_1} & \frac{\partial T_M}{\partial P_2} & \frac{\partial T_M}{\partial P_3} & \cdots & \frac{\partial T_M}{\partial P_N} \end{bmatrix} \quad (3.17)$$

where M is the number of sensors and N is the number of unknown parameters. The elements of the sensitivity matrix given by equation (4.15) are called Sensitivity Coefficients:

$$J_{ij} = \frac{\partial T_i}{\partial P_j} \quad (3.18)$$

These coefficients J_{ij} are thus defined as the first derivative of the estimated temperature at i th thermocouple position with respect to the unknown parameter P_j .

Note that if the sensitivity coefficients J_{ij} are small, it means that large changes in P_j yield small changes in T_i , hence the estimation of the parameters P_j is extremely difficult because same values for temperatures would be obtained with a wild range of values of P_j . Whenever the sensitivity coefficients are small, the problem does not satisfy the Identifiability Condition:

$$|\mathbf{J}^T \mathbf{J}| \neq 0 \quad (3.19)$$

and it appear to be ill-conditioned. The expression above was introduced in one of the first developed inverse methods, the Levenberg-Marquardt Method (LMM), in which the iterative procedures could be performed only if the matrix $\mathbf{J}^T \mathbf{J}$ was non-singular. The non-singularity comes if any column of $\mathbf{J}(\mathbf{P})$ can be expressed as a linear combination of the other columns. Hence, it is better to have linearly-independent sensitivity coefficients J_{ij} with large magnitudes, in this way, the inverse problem is not very sensitive to measurement errors and accurate estimate of the parameters can be obtained. While the LMM bypassed the ill-conditioning by introducing the terms μ^k and Ω^k to damp oscillations and instability, the CGM overcomes the need of a non-singularity matrix by the direction of descent \mathbf{d}^k and search step size β^k . However, the uncertainties in solutions remain since these terms depend on magnitude values of sensitivity coefficients.

In literature, there are many different expedients to determine the values of $\mathbf{J}(\mathbf{P})$, but since the inverse problem considered is non-linear a Finite Difference Approximation (FDA) is required. More specifically, a forward finite difference approach of first order has been selected, thus the sensitivity coefficient with respect to parameter P_j is approximated by

$$J_{ij} \cong \frac{T_i(P_1, P_2, \dots, P_j + \varepsilon P_j, P_N) - T_i(P_1, P_2, \dots, P_j, P_N)}{\varepsilon P_j} \quad (3.20)$$

where ε can be set equal to 10^{-5} or to 10^{-6} . It is noteworthy that the computation of $\mathbf{J}(\mathbf{P})$ needs $N + 1$ solutions of the direct problem and thus it is consequently very time-consuming. Hence, as it will be seen in chapter 5 it is not advisable to perform this calculation more than one time.

3.3.2 The Direction of Descent

The direction of descent is a vector that search the direction to move the equation (3.15) toward a local minimum. It is a linear combination of the gradient direction, $\nabla S(\mathbf{P}^k)$, and the direction of descent of the previous iteration, \mathbf{d}^{k-1} :

$$\mathbf{d}^k = \nabla S(\mathbf{P}^k) + \gamma^k \cdot \mathbf{d}^{k-1} \quad (3.21)$$

The strength of this link is expressed by the conjugation coefficient γ^k whose value, hence, indicates the influence of the descent direction from previous iteration. For $\gamma = 0$, the conjugate gradient method is identical to the steepest descent method. Several different expression for computing the conjugation coefficient are available in litterature, the most frequently used are the Polak-Ribiere expression [20] [3]:

$$\gamma^k = \frac{\sum_{j=1}^N \left\{ \left[\nabla S(\mathbf{P}^k) \right]_j \left[\nabla S(\mathbf{P}^k) - \nabla S(\mathbf{P}^{k-1}) \right]_j \right\}}{\sum_{j=1}^N \left[\nabla S(\mathbf{P}^{k-1}) \right]_j^2} \quad \text{for } k = 1, 2, \dots \quad (3.22)$$

with $\gamma^0 = 0$ for $k = 0$, and the Fletcher-Reeves expression [13] [3]:

$$\gamma^k = \frac{\sum_{j=1}^N \left[\nabla S(\mathbf{P}^k) \right]_j^2}{\sum_{j=1}^N \left[\nabla S(\mathbf{P}^{k-1}) \right]_j^2} \quad (3.23)$$

Both formulations guarantee that the angle between the direction of descent and the negative gradient direction is less than 90° , in this way, the function $S(\mathbf{P})$ is certainly minimized [11].

Once the sensitivity matrix $\mathbf{J}(\mathbf{P})$ is computed, the gradient direction, which is a differentiation of the equation (3.14) with respect to the unknown parameters \mathbf{P} , can be calculated as:

$$\nabla S(\mathbf{P}^k) = -2(\mathbf{J}^k)^T \cdot [\mathbf{Y} - \mathbf{T}(\mathbf{P}^k)] \quad (3.24)$$

3.3.3 The Step Size

The search stepsize β^k gives the magnitude of change of the heat flux at parameter points and it is determined by minimizing the function $S(\mathbf{P}^{k+1})$ with respect to β^k :

$$\beta^k = \frac{[\mathbf{J}^k \mathbf{d}^k]^T [\mathbf{T}_c(\mathbf{P}^k) - \mathbf{T}_{meas}]}{[\mathbf{J}^k \mathbf{d}^k]^T [\mathbf{J}^k \mathbf{d}^k]} \quad (3.25)$$

Since the members of $\mathbf{J}(\mathbf{P})$ are of very small order of magnitude, in accordance to [17] the expression above yields very large values for the step size and, consequently, huge parameter oscillations can be observed together with no convergence of the optimization or a very slow one. In order to prevent these oscillations and to stabilize the optimization in the beginning, when residual functional value is large, the overall variation of the heat $\beta^k \mathbf{d}^k$ is limited to a certain interval.

3.4 The Stopping Criterion

Since the minimization of the residual functional, theoretically, can be carried forward indefinitely, a stopping criterion is necessary to exit from the iterative procedure and move on the next time step. Hence, after computing the sensitivity matrix \mathbf{J} , the gradient direction $\nabla S(\mathbf{P}^k)$, the conjugation coefficient γ^k and the search step size β^k , the updating process given by equation (3.16) is performed until a stopping criterion based on the Discrepancy Principle is satisfied:

$$S(\mathbf{P}^{k+1}) < \varepsilon = \sum_{i=0}^I \sigma_i^2 + \psi \quad (3.26)$$

In the expression above σ_i is the standard deviation of the measurement error at time t_i and ψ is an additional term due to numeric and discretization errors. It is precisely for these errors that the theoretic absolute convergence ($S = 0$) cannot be reached and the stopping criterion is necessary. Note this approach based on the discrepancy principle, requires the a priori knowledge of the standard deviation of the measurement errors.

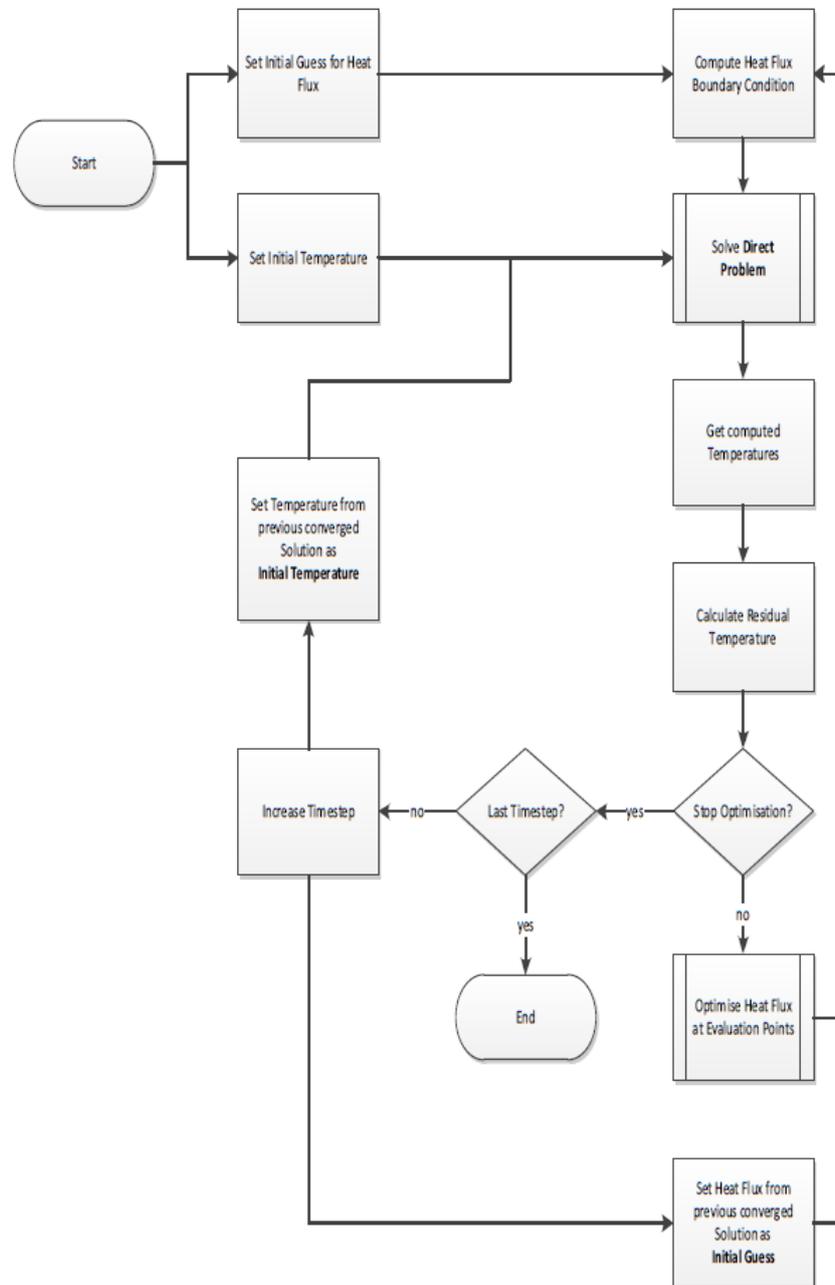
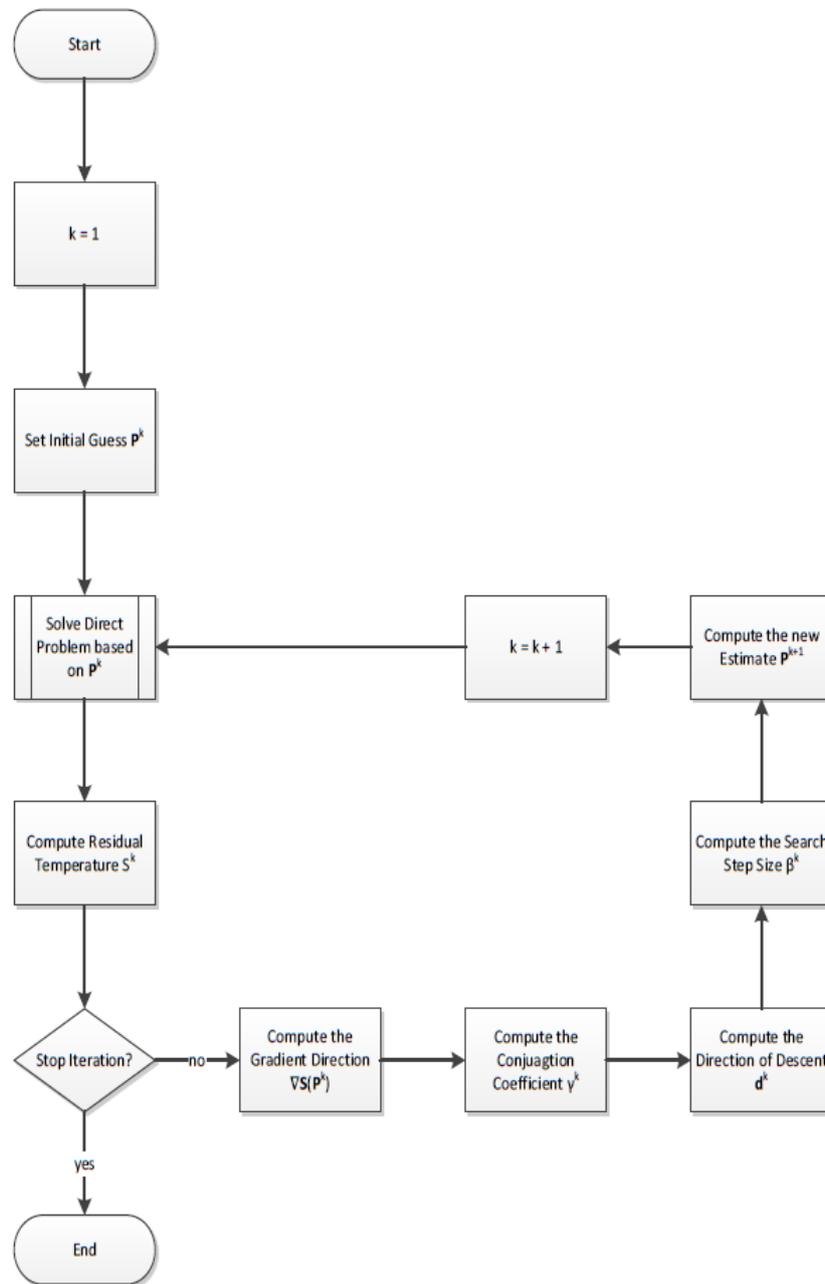


Figure 3.3: Transient Optimisation Procedure

**Figure 3.4:** Steps of the Parameter Update

Chapter 4

RoqFITT Code and its Modifications

This chapter is divided into two parts: in the first one it is explained how RoqFITT (Rocket \dot{q} Flux Inverse Thermal Tool developed by Perakis [8] at the TUM) works in detail by exploring most important functions with particular attention to their inputs and outputs, in the second one, instead, the modifications made to the code will be presented together with their results.

As said in Section 1.4, the RoqFITT code has been developed in Matlab environment for solving the heat transfer problem exclusively for single- and five-element combustion chambers. Therefore, since both Matlab codes are twins code, the one solving MoRaP will be taken as a reference and the possible differences with BKM code (mainly due to geometric reasons) will be specially highlighted.

4.1 RoqFITT Code

4.1.1 Loading Experimental Data and User Inputs

The inverse method solver consists in a main, *MainInverseSingleElement*, in which every single function is recalled. The first part is devoted to loading data and user inputs as well as folder arrangements. The user can set the following variables:

- *TestInfo* is used to load the experimental data file coming from the DAQ acquisition system;
- *create_sensitivity* provides for computing a new sensitivity matrix or for loading an old one (*Optimization.Sens_matrix_file*), since the number of thermocouples used in each experiment is certainly variable;

- *interp_method* establishes how to interpolate in each plane the heat flux parameters on the whole upper wall (where all thermocouples are located), i.e. the heat flux can assume a parabolic or a constant shape from a corner to another corner.
- *Thermocouples.delete* allows the user to exclude from simulation those thermocouples found to be defective during the experiment;
- *Thermocouples.smooth* is used to smooth the thermocouples measurements, it will be seen in Chapter 5 what are the consequences if no smooth is applied on thermocouples readings;
- *SolutionSettings.N_iterations* is used to set the maximum number of iteration in a single time step for the simulation;
- *TimeSettings* structure manages the whole computational time in accordance with the experimental time acquisition, user can establish the *starting_time* and *ending_time* of the simulation as well as can set the computational time step in *TimeSettings.dt_user*;
- *MaterialProperties* structure allows to set the thermal conductivity, density, and specific heat capacity of the medium (in this case oxygen-free copper);
- *Geometry* structure let user set all combustion chamber dimensions;
- *BoundaryCondions* structure allows user to set the convection heat transfer coefficient h_∞ , discussed in Section 3.1.1.1, and outside temperature value T_∞ as well as to indicate to the code the distance from hot gas side of the thermocouples used in the experiment (*use_1mm*, *use_2mm* or *use_3mm*);
- *Results* folder is created in order to save all variables and plots at the end of simulation;

4.1.2 Initializations

This part of the code precedes the updating process and it is necessary to initialize all variables that will be used in the simulation. The following paragraphs introduce the functions recalled in the main essential to set up the whole analysis.

Definition and Discretization of the Domain

The first function recalled is:

buildGeometry

The main purpose of this function is to discretize the entire domain by computing the spatial steps (*Geometry.dx* and *Geometry.dz*) so as to ensure the presence of nodes on all boundaries including the corners. Recalling figure 2.3, it is clear that the combustion chamber schematization used for the computation is much simpler than the reality. All numbers of nodes involved in the resolution of the direct problem are listed in table 4.1.

Table 4.1: Variables counting nodes

<i>Variables</i>	<i>Description</i>
<i>Geometry.Nx</i> and <i>Geometry.Ny</i>	number of nodes in x and y direction
<i>Geometry.Nx_in</i> and <i>Geometry.Ny_in</i>	number of nodes in hot side, x and y direction
<i>Geometry.Nxy</i>	number of nodes in each plane
<i>Geometry.Nz</i>	number of nodes in z direction (combustion chamber)
<i>Geometry.N_block</i>	number of node in z direction (heat sink)
<i>Geometry.n</i>	number of nodes along the hot gas wall perimeter

Temperature Readings Modification and Time Management

The following function:

modifyThermocoupleMeasurements

first, aims to fit together thermocouples acquisition times and computational time vector chosen by the user, then it selects from loaded data only the thermocouples used in the test and loads all their positions. The whole experiment acquisition time is initialized as a vector variable *time_MORAP* and it is representative of all thermocouples readings. By using the previous user settings (*TimeSettings.dt_user*, *TimeSettings.starting_time* and *TimeSettings.ending_time*) a computational time vector for the simulation, *TimeSettings.time*, is created. Figure 4.1 shows that the whole time domain is divided into $N-1$ segments, i.e. in N time steps, one can note that just the first segment coincides with a time step while all the others are enclosed between two time-steps, i.e. blue and red for start and end respectively.

Subsequently, the function identifies the thermocouples collected in the cell arrays *data.AI* and interpolates their readings on the computational time. After the matching has been completed, if *Thermocouples.smooth* marks 1 all temperatures are smoothed in time using

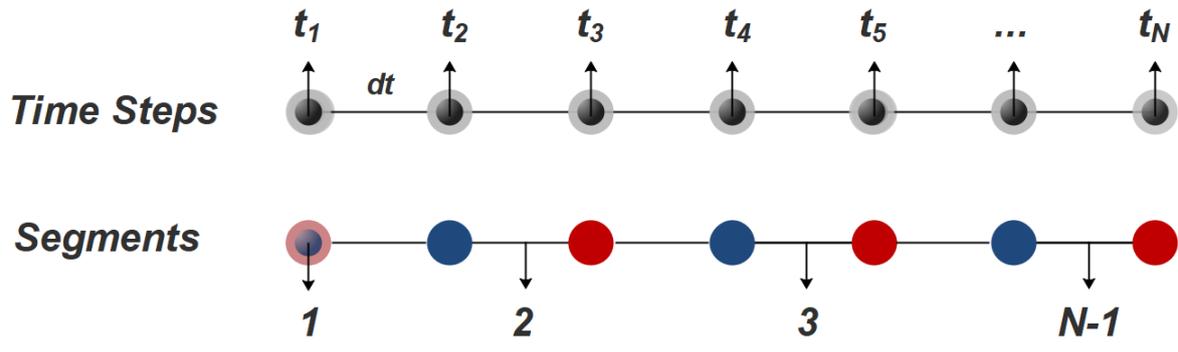


Figure 4.1: Time Discretization

a moving average on 3 points. Hence, the used thermocouples are grouped under the vector variable *Thermocouples.Names* as well as their readings under *Thermocouples.Tmeasured*. Finally, recalling the nomenclature introduced in table 2.5, the function calculates their position and put it into *Thermocouples.Locations*.

Define Optimization Paramters Points

defineParameterPoints

This function defines the numbers, *Parameters.N* and the positions, *Parameters.Locations*, of the parameters points, i.e. the points in which the heat flux is updated throughout the process. In all simulations carried out in this thesis, their number has been set equal to the number of thermocouples ($N = M$). Hence, their positions are essentially the projection of thermocouples locations on the hot gas wall (Fig. 4.2). Anyway, the user is allowed to enter more parameters by adding manually their positions.

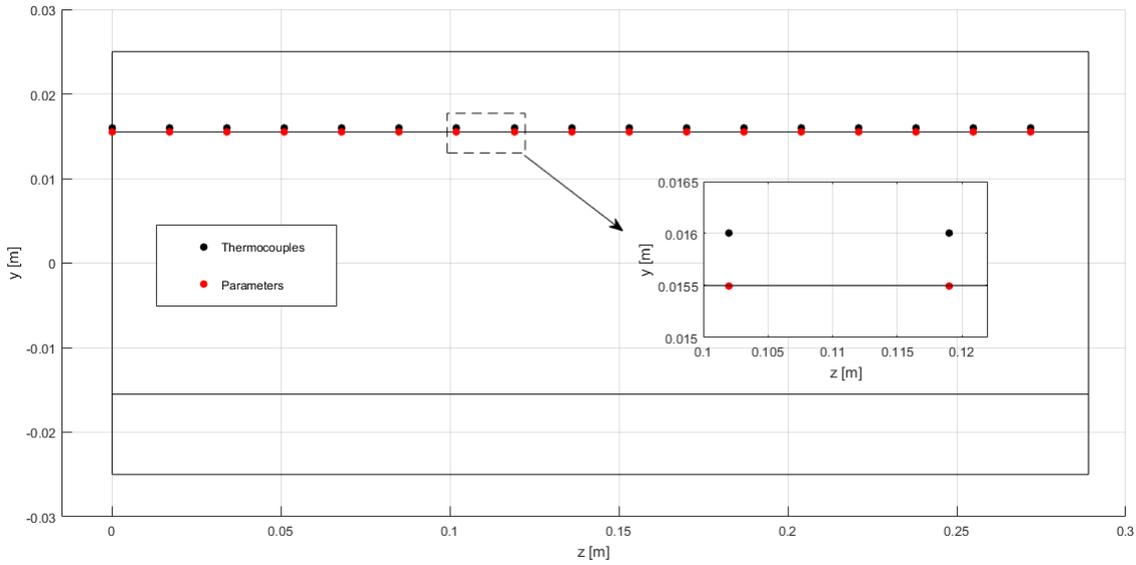


Figure 4.2: Thermocouples and Parameters Positions on $z - y$ Plane in Single-Element

Temperature Initialization

initializeTemperature

Throughout the optimization process, the code uses a variable *Temperature.allNodes* for saving computed temperatures in each node at the end of each segment. In order to solve the direct problem at the first iteration of the first time segment, it is necessary to provide the initial condition. In this regards, this function assign to those nodes located in the same plane of a thermocouple the measured temperature by that thermocouple before the ignition, otherwise, instead, to the all others node allocate a temperature value equal to the average between all thermocouples measurements before the ignition (*Temperature.initialT*, Fig. 4.3).

Figure 4.4 helps to understand how all nodes are scanned: once the last node in the plane is reached, the plane is switched, and the scanning is repeated in the same way.

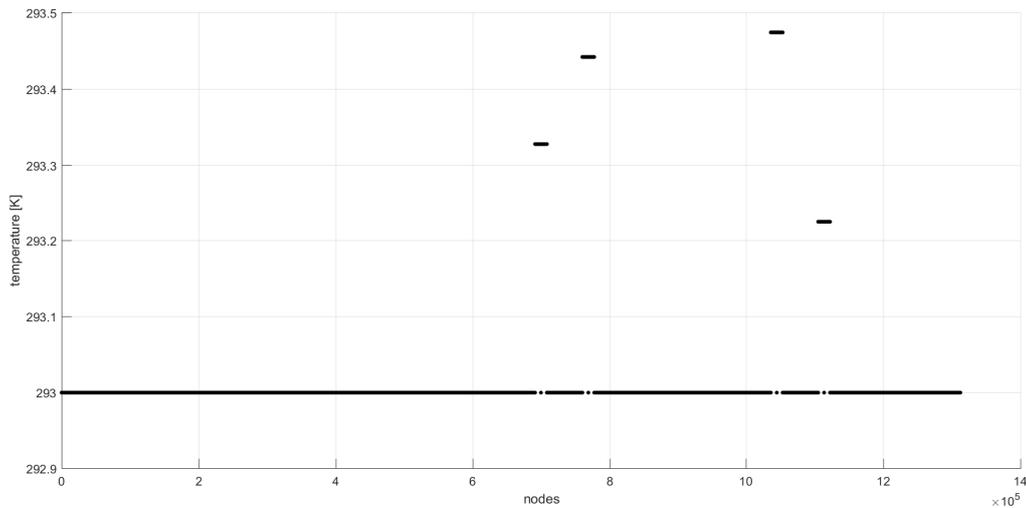


Figure 4.3: Initial Temperature condition Assigned to all Nodes

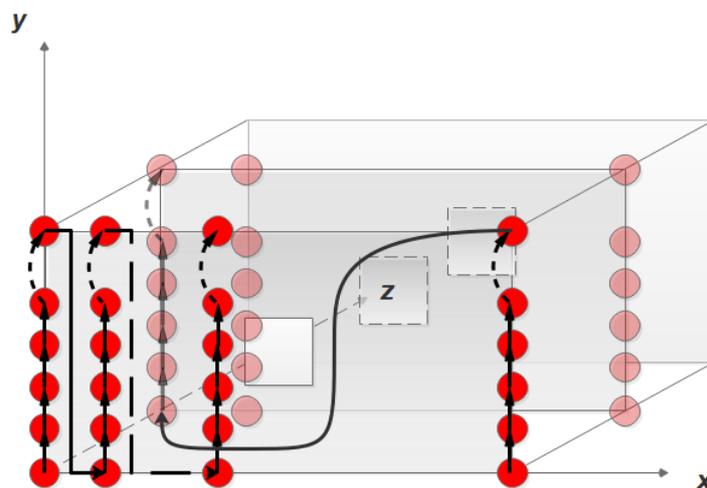


Figure 4.4: Example of How the Code Scans the Domain

Stopping Criterion Initialization

With following function:

defineStoppingCriterion

in accordance with literature, the threshold value ε present in the discrepancy principle, introduced in Section 3.4, is set equal to 0.3 , i.e. equal to the type T thermocouples standard deviation.

Heat Flux First Estimation and Initialization

initializeHeatFlux

The heat flux is, here, initialized for all hot gas wall nodes at the first iteration of each segment, it has a linear profile along z axis with the highest guess value (q_0) in correspondence of the nozzle. Heat flux is assumed to be constant for all nodes belonging to the same x - y plane, that explains the presence of several flat parts. The first guess is absolutely arbitrary (Fig. 4.5), from this point of view code shows a robustness, since it is capable of achieving the convergence with any first assigned value.

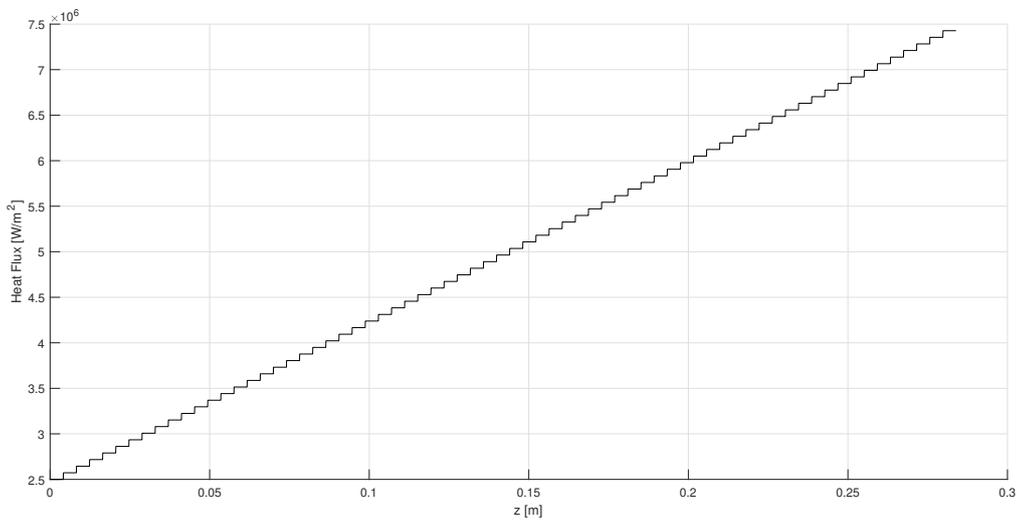


Figure 4.5: Example of Heat Flux First Guess

Linear System Assembling

After that some variables, necessary for computing the updating process are initialized under the structure *Optimization*, the following function:

buildGeometryMatrix

is recalled in the main to implement the discretization of the heat diffusion equation (3.2). Remember that the forward in time central in space finite difference scheme has been performed by applying the energy balance method to each nodes obtaining in this way the following linear system:

$$\underline{MMT}_s = \underline{T}_{s-1} + \underline{CC} \quad (4.1)$$

Where \mathbf{T}_s is the vector of the unknown nodal temperatures at time t_s , \mathbf{T}_{s-1} is the vector of known nodal temperatures coming from solution at previous time step t_{s-1} , \mathbf{MM} is the matrix of coefficients and \mathbf{CC} is the vector of known terms. Because of discretization it is clear that the \mathbf{MM} matrix is a sparse diagonal matrix with seven non-zero diagonals (Fig. 4.6(a), the elements of its diagonal are the nodes to which the heat diffusion equation is applied, the other non-zero elements are the adjacent nodes involved in the same equation). It is obtained as a combination of the connectivity matrix, $LinearSystem.S$, and the boundary conditions vectors, $LinearSystem.CC_in$, $LinearSystem.CC_out_x$, $LinearSystem.CC_out_z$, $LinearSystem.CC_out_nozzle$ in this way:

$$\begin{aligned} MM &= S + (CC_out_x \cdot Fo_x \cdot Bi_out_x + CC_out_z \cdot Fo_z \cdot Bi_out_z) \\ CC &= diag(CC_out_x \cdot Fo_x \cdot Bi_out_x \cdot T_inf + CC_out_z \cdot Fo_z \cdot Bi_out_z \cdot T_inf) \end{aligned}$$

To save mamory, the connectivity matrix \mathbf{S} is built with *spdiags* Matlab command having as iput the BB matrix of coefficients coming from the discretization and a d_vec vector difining their position in the matrix.

$$\begin{aligned} &BB(Geometry.N + Geometry.N_block \cdot Geometry.Nxy, \gamma) \\ d_vec &= [-Geometry.Nxy, -Geometry.Ny, -1, 0, 1, Geometry.Ny, Geometry.Nxy] \\ S &= spdiags(BB, d_vec, N+N_block \cdot Nxy, N+N_block \cdot Nxy) \end{aligned}$$

To understand how these variables are built, the following example consider a generic node ($i = 17271$, $j = 2$, where i goes from 1 to Nxy , and j from 1 to Nz), second plane from the nozzle) located at the corner outside middle plane (Fig. 4.6(b)). The discretized heat diffusion equation related to this particular node is

$$\begin{aligned} T_n^{k+1} \cdot (1 + 4 \cdot Fo_x Bi_x + 2 \cdot Fo_z + 4 \cdot Fo_x) - 2 \cdot Fo_x \cdot T_{n-1}^{k+1} - 2 \cdot Fo_x \cdot T_{n-N_y}^{k+1} \\ - Fo_z \cdot T_{n+N_{xy}}^{k+1} - Fo_z \cdot T_{n-N_{xy}}^{k+1} = T_n^k + 4 \cdot Fo_x Bi_x T_\infty \end{aligned} \quad (4.2)$$

which brings to define a BB matrix and a CC_out_x boundary condition vector:

$$\begin{aligned} BB(17271 - N_{xy}; 1) &= -Fo_z \\ BB(17271 - N_y, 2) &= -2 \cdot Fo_x \\ BB(17271 - 1, 3) &= -2 \cdot Fo_x \\ BB(17271, 4) &= 1 + 4 \cdot Fo_x + 2 \cdot Fo_z \\ BB(17271 + N_{xy}; 7) &= -Fo_z \\ CC_out_x(17271, 1) &= 4 \cdot Fo_x \end{aligned}$$

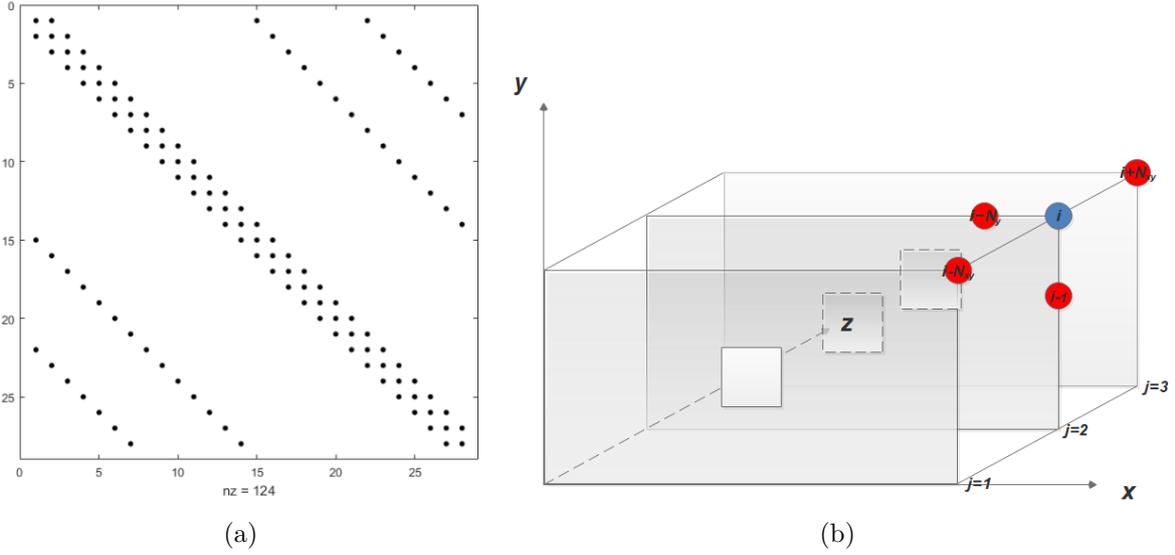


Figure 4.6: Example of How the MM Matrix is Filled (a), Reference Node for the Example Above (b)

Cholesky Factorization and Conjugate Gradient Method for Solving Linear Systems

Once the linear system (4.1) is defined, to solve it, first a Cholesky factorization and then a Conjugate Gradient Squared Method (CGSM) is performed reducing in this way both computational cost and occupied memory. In order to give further information about how these expedients work, the following linear system recalling the equation (4.1) is considered:

$$\underline{\mathbf{A}} \mathbf{T}_s = \mathbf{T}_{s-1} + \mathbf{b} \quad (4.3)$$

Since the two equations (4.1) and (4.3) are identical, each term has the same meaning already explained in previous paragraph: $\underline{\mathbf{A}}$ is the matrix of coefficients, i.e. the MM matrix used in the code, depending both on the discretization of the equation and boundary conditions and, \mathbf{b} is the same vector \mathbf{CC} discussed before depending on the boundary conditions. It is clear that the matrix of coefficients is an $N \times N$ matrix with N number of nodes in which the domain is discretized. A linear system can be solved in many different methods, they are classified as direct or iterative methods. When dealing with sparse matrices of coefficient the former compared to the latter imply high computational cost, hence, for this reason, the CSGM, which presents an iterative approach, has been chosen to solve the linear system (4.1). A brief description of this method is given in accordance with [10] considering the following linear system:

$$\underline{\mathbf{A}} \mathbf{x} = \mathbf{b} \quad (4.4)$$

The algorithm of resolution, already implemented in Matlab with *cgs* command, can be divided in the following steps:

Table 4.2: CGSM algorithm

<i>Steps</i>	<i>Calculations</i>
1	With a first approximation \mathbf{x}_0 calculate the residual $\mathbf{r}_0 = \mathbf{b} - \underline{\mathbf{A}}\mathbf{x}_0$ with $\mathbf{p}_0 = \mathbf{r}_0$
2	For $j = 0, 1, \dots$, until convergence perform:
3	$\alpha_j = \frac{\mathbf{r}_j \cdot \mathbf{r}_j}{\underline{\mathbf{A}}\mathbf{p}_j \cdot \mathbf{p}_j}$
4	$\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{p}_j$
5	$\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j \underline{\mathbf{A}}\mathbf{p}_j$
6	$\beta_j = \frac{\mathbf{r}_{j+1} \cdot \mathbf{r}_{j+1}}{\mathbf{r}_j \cdot \mathbf{r}_j}$
7	$\mathbf{p}_{j+1} = \mathbf{r}_{j+1} + \beta_j \mathbf{p}_j$

Before performing it, to improve the CGSM, the code uses a preconditioning on the matrix of coefficient. It consists in a transformation of the original linear system into an other one which accepts the same solution but it is easier to solve by an iterative method. Hence, the first step in preconditioning is to find a preconditioning non-singular matrix $\underline{\mathbf{M}}$ which is more or less equal to $\underline{\mathbf{A}}$ and enables to express the system (4.4) as:

$$\underline{\mathbf{M}}^{-1} \underline{\mathbf{A}} \mathbf{x} = \underline{\mathbf{M}}^{-1} \mathbf{b} \quad (4.5)$$

In order to ensure symmetry and positive definiteness of $\underline{\mathbf{A}}$, the preconditioner $\underline{\mathbf{M}}$ is split using the Cholesky factorization (CF):

$$\underline{\mathbf{M}}^{-1} = \underline{\mathbf{L}}\underline{\mathbf{L}}^T \quad (4.6)$$

with $\underline{\mathbf{L}}$ non-singular matrix. Hence the system (4.4) can be rewritten as:

$$\underline{\mathbf{A}} \mathbf{x} = \mathbf{b} \Leftrightarrow \underline{\mathbf{L}}^T \underline{\mathbf{A}} \underline{\mathbf{L}} \underline{\mathbf{L}}^{-1} \mathbf{x} = \underline{\mathbf{L}}^T \mathbf{b} \quad (4.7)$$

The Cholesky factorization is performed in Matlab with *ichol* command. In this way the CGSM is modified with following algorithm:

Table 4.3: CGSM algorithm with Cholesky factorization

<i>Steps</i>	<i>Calculations</i>
1	With a first approximation x_0 calculate the residual $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$; $\mathbf{q}_0 = \mathbf{L}^{-1}\mathbf{r}_0$; $\mathbf{p}_0 = \mathbf{L}^{-T}\mathbf{q}_0$
2	For $j = 0, 1, \dots$, until convergence perform:
3	$\alpha_j = \frac{\mathbf{q}_j \cdot \mathbf{q}_j}{\mathbf{A}\mathbf{p}_j \cdot \mathbf{p}_j}$
4	$\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{p}_j$
5	$\mathbf{q}_{j+1} = \mathbf{q}_j - \alpha_j \mathbf{L}^{-1} \mathbf{A}\mathbf{p}_j$
6	$\beta_j = \frac{\mathbf{q}_{j+1} \cdot \mathbf{q}_{j+1}}{\mathbf{q}_j \cdot \mathbf{q}_j}$
7	$\mathbf{p}_{j+1} = \mathbf{L}^{-T} \mathbf{q}_{j+1} + \beta_j \mathbf{p}_j$

Create Sensitivity Matrix

createSensitivityMatrix

is the last function to be recalled in the main before the inverse loop starts, it is necessary to compute the sensitivity matrix involved in the optimization process if it is not directly loaded. The method used for this scope has already been described in Section 3.3, however, it has to be mentioned that this computation is completed outside the loop, hence, this matrix does not vary step by step or segment by segment. As already mentioned the sensitivity coefficients $J_{i,j}$ are obtained by solving $N + 1$ times the heat conduction direct problem. The computational time vector considered for this purpose is:

$$time_sensi = [0 : dt_sensi : 1]$$

with $dt_sensi = dt_user$ but it can be modified by user. The initial temperature or initial condition is the same for each problem and is set equal to the temperature distribution computed in the function *initalizeTemperature*. The first direct problem is solved by setting in all N parameters points a constant heat flux equal to $q_level = 5$ MW, the other N remaining direct problems are solved by varying the heat flux parameters one by one by an amount equal to $q_epsilon = 10^{-5}$. Once $N + 1$ different temperature distributions are computed, the sensitivity coefficient $J_{i,j}$ is obtained by subtracting the temperature at i -th parameter point, result of the problem in which the j -th parameter heat flux has been varied, to the temperature at the same location resulting from the

direct problem in which the heat flux has been set constant, i.e. the first direct problem. In this way, the whole sensitivity matrix is obtained.

4.1.3 Inverse Loop Start

From now the real updating process starts. The code scrolls each *segment* one at the time and whenever, after several iterations in which the heat flux parameters have been updated, the convergence is reached, the variable *SolutionSettings.complete(segment)* switches from *false* to *true*, the segment is switched too and the variable counting the iterations is restored to one. Recalling the scheme used in figure 4.1, figure 4.7 helps to understand when the code performs the direct problem and when the updating process. Note that while the direct problem is solved in each time step the updating procedure is solved at the end of each time segment keeping the heat flux parameters P_{seg}^k (with k referring to iterations and seg to segments) constant all over the time segment.

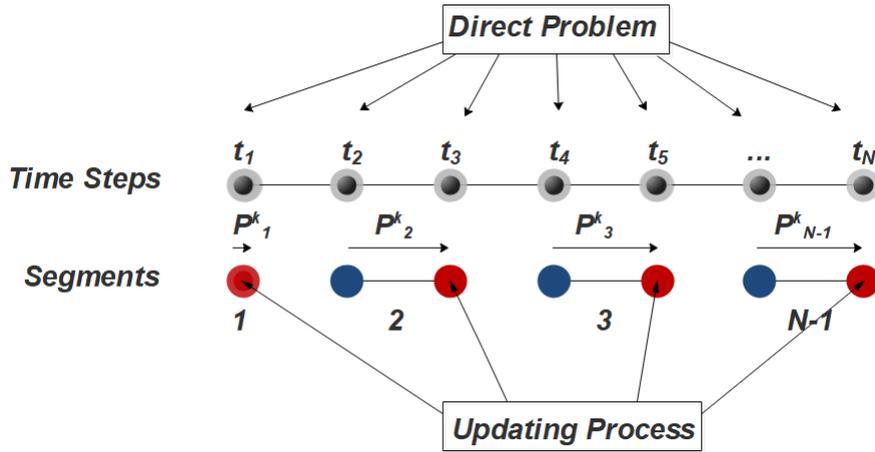


Figure 4.7: Optimization Process on Segments

At the beginning of each time segment the heat flux (HF) that has to be applied as boundary condition to the direct problem is initialized by combining the converged solution found in the previous time segment and the heat flux estimated at the ongoing iteration:

$$HF_{seg}^k = (1 - par_2) \cdot HF_{seg}^k + par_2 \cdot HF_{seg-1}^{optIter(seg-1)} \quad (4.8)$$

In the above equation par_2 is set in the code equal to 0.9 and $optIter_{seg-1}$ is the converged iteration of the previous time segment. At the beginning of each iteration, instead, the heat flux is initialized by combining the heat flux of previous iteration and heat flux estimated at the ongoing iteration:

$$\mathbf{HF}_{seg}^k = (1 - par_1) \cdot \mathbf{HF}_{seg}^k + par_1 \cdot \mathbf{HF}_{seg-1}^k \quad (4.9)$$

with par_1 set in the code equal to 0.25. These expedients are introduced to assist convergence and to avoid oscillations. After solving the linear system and computing the square difference between measured and computed temperatures at thermocouples positions with the latter coming from the function *calcTemperatureAtThermocoupleFromDistribution*, the code performs the calculation of the residual functional (*Optimization.ResidualFun*) as integral of the square sum over time and checks continuously if the discrepancy principle is satisfied. If the *Optimization.ResidualFun* is greater than *SolutionSettings.epsilon*, the heat flux parameters have to be updated. Hence, once *Optimization.DescentDirection* and *Optimization.beta* (step size) are available all new heat flux parameters are computed following each step discussed in chapter 3. If *Optimization.ResidualFun* decrease continuously, it means that the solution is continuously improved otherwise the code recognizes if this variable fluctuates around a value and stops the updating process and switches the segment. For the way code works, the heat flux parameters can be also negative, but since it cannot be physical, anytime a negative heat flux is found, it is set equal to zero. The last function of optimization loop:

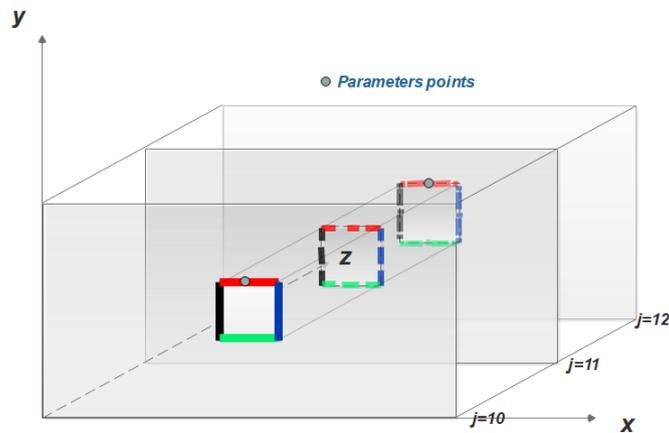
transformHeatFlux

is one of the most important functions that user can find in this code. It has the purpose of interpolating the updated heat flux parameters all over the walls of the combustion chamber and make it usable for the direct problem:

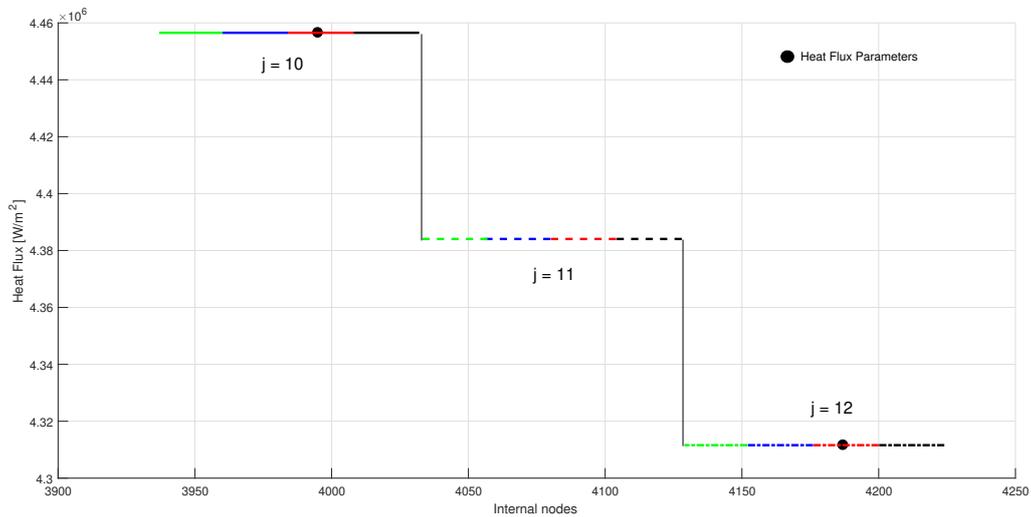
$$\mathbf{P}_{seg}^k \rightarrow \mathbf{HF}_{seg}^k \quad (4.10)$$

It is a crucial step since the inverse problem solution, as it will be seen in chapter 6, depends directly upon the type of interpolation carried out. In this regards, figures 4.8 and 4.9 help to understand how the interpolation is carried out after that the parameters heat flux are estimated. In this example, 3 generic planes $j = 10, 11$ and 12 , are considered (they are not related with any real planes coming from the discretization). The estimated parameters points belong to the two planes $j = 10$ and $j = 12$. With regards to single-element, since only one thermocouples is located in planes and therefore only one parameter point, it is simply assigned to all nodes belonging to the same plane of a parameters point the same heat flux estimated in that parameter point and subsequently a cubic interpolation between the planes is carried out 4.8(b). Instead, with regards to five-elements, since the presence of five injectors makes the heat flux evolution more complex along x and since more than one thermocouple is located in a plane, therefore more than one parameter

point (max 7 Fig. 2.5), to provide a more faithful reproduction of the heat flux along x , in those planes where the parameter points are less than 7 more points to be interpolated are added by averaging the estimated heat flux parameters. In this regards figure 4.10 helps to understand this procedure, the points A , B , D and E lying not directly over an injector are obtained by averaging the parameters points $3L$ and $3R$, instead for those points lying directly on an injector it has been taken the value if the parameter point $3C$. Then a cubic interpolation is performed along x and subsequently along z . An example of the result of this approach is given in figure 4.9, one can note that nodes belonging to upper and lower hot gas wall have the same heat flux.

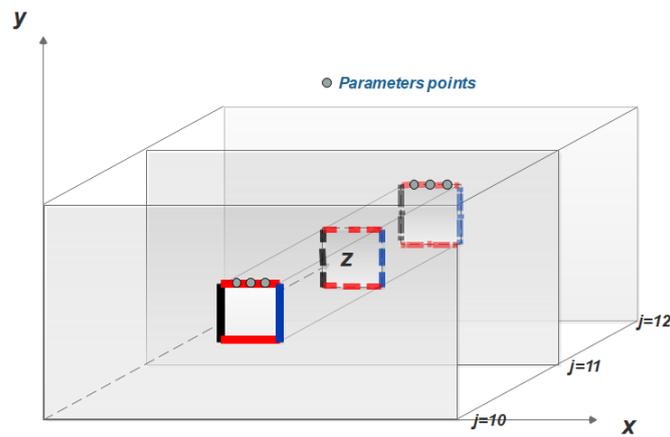


(a) Schematization of 3 Generic Planes

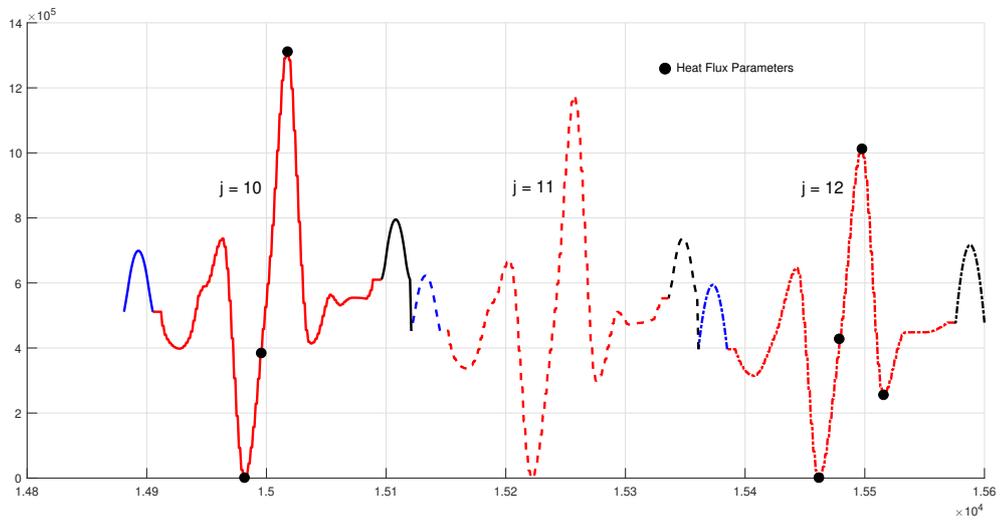


(b) Heat Flux Evolution

Figure 4.8: Example of Heat Flux Interpolation on Internal Nodes for Single-Element



(a) Schematization of 3 Generic Planes



(b) Heat Flux Evolution

Figure 4.9: Example of Heat Flux Interpolation on Internal Nodes for Five-Element

4.1.4 Code Outputs

In this section, the most important outputs, referring to the two investigated tests, released by the code are presented. The evaluation time has been set equal to $t = 7.341 s$ and $t = 23.398 s$ for single- and five-element respectively.

For logical reasons figure 4.11 shows the matching between measured and computed temperatures in just few thermocouples positions, but it is clear that this kind of plot exists for each thermocouple used in the test. This is one of the most important output because it allows you to check if the code has reached the convergence in a correct manner.

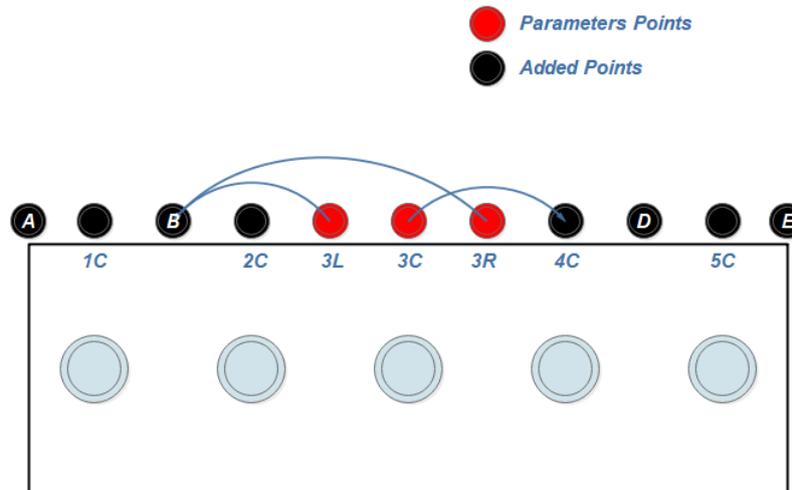


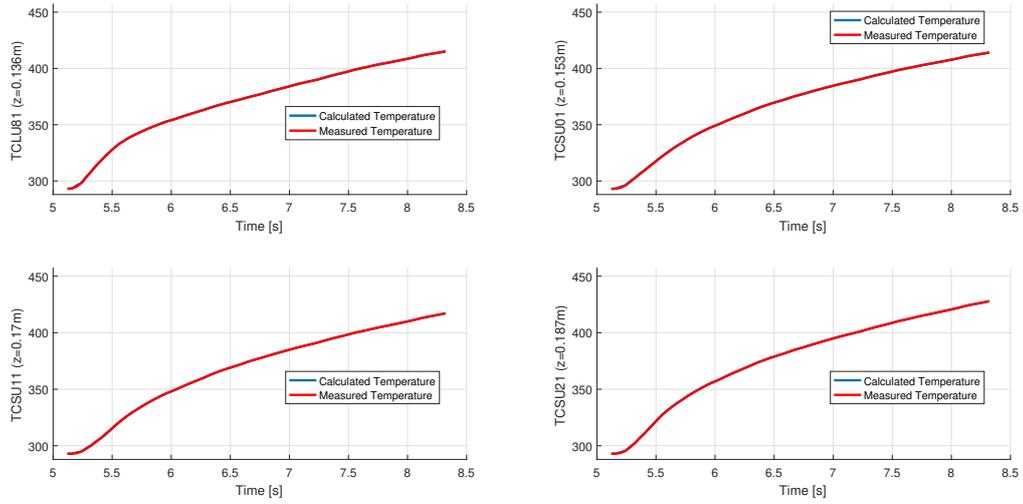
Figure 4.10: Example of How New Points are added for Interpolation

Figure 4.12 explains the evolution of heat flux over time for all distances from injector plate. One can note that in both chambers the heat flux reaches an almost stationary value after about 1.5 s.

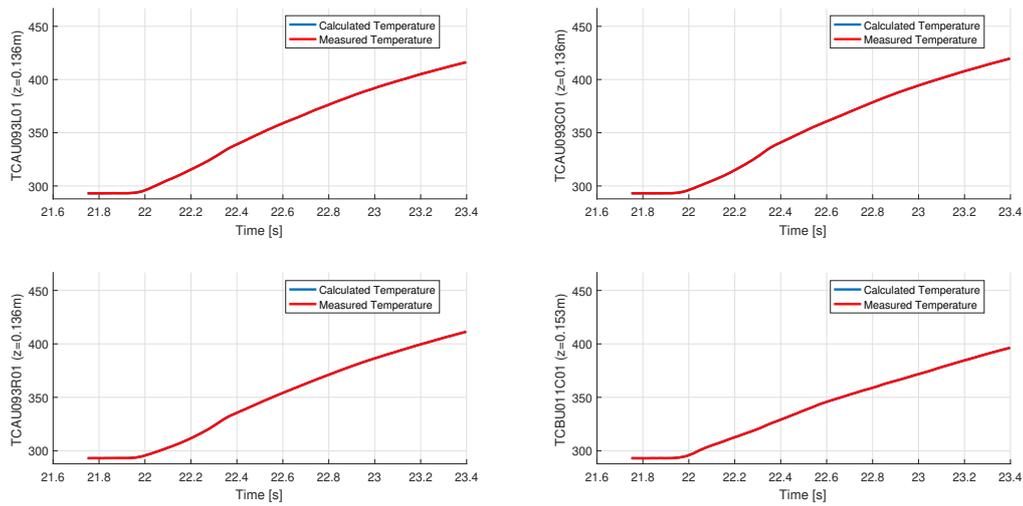
From figure 4.13 is possible to appreciate the heat flux in middle point along the chambers. One can note the big oscillations in space mentioned in chapter 1, which can reach up to 2 MW and 6 MW in magnitude, in single- and five-element respectively. These oscillations could be related to the presence of the injectors and therefore in this sense, they could be allowed except they arise also far from the copper block and they remain also after the shutdown of the injectors (over 0.5 s from ignition) and do not disappear. While in the first case this value can be more or less accepted, but still investigated in chapter 5, in the second one the suspicion that these oscillations are produced by the code has encouraged a study, conducted in chapter 6, on how the heat flux can be interpolated from parameters points to hot gas wall points. Figure 4.14 demonstrates what already anticipated in Section 4.1.3, i.e. the interpolation of the heat flux along x ensures a constant trend in single-element and an oscillating evolution in five-element.

Figure 4.15 shows the evolution of computed temperature along x on the upper hot gas wall at evaluation time. One can note that while in the single-element the temperature exhibits a parabolic trend from a corner to another one, in the five-element they reveal an oscillating trend. These oscillations, however, are consistent with the measured temperatures by thermocouples at 1 mm from the hot gas wall (Fig. 4.15(b)).

Figures from 4.16 to 4.18 show the evolution of computed temperatures along z at 1 mm from the hot gas wall directly on the injectors (Fig. 4.16, 4.18 and 4.19(a)) and between



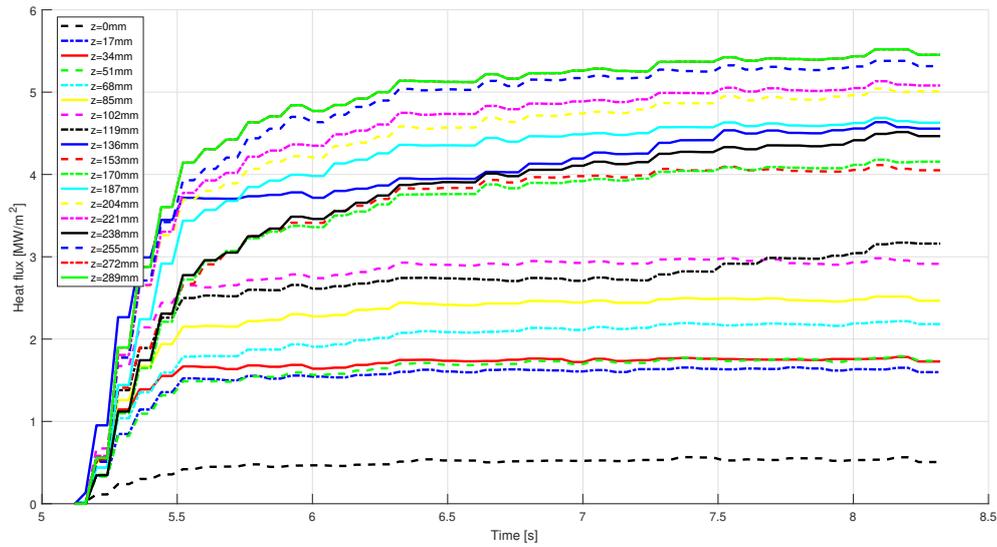
(a) Single-Element



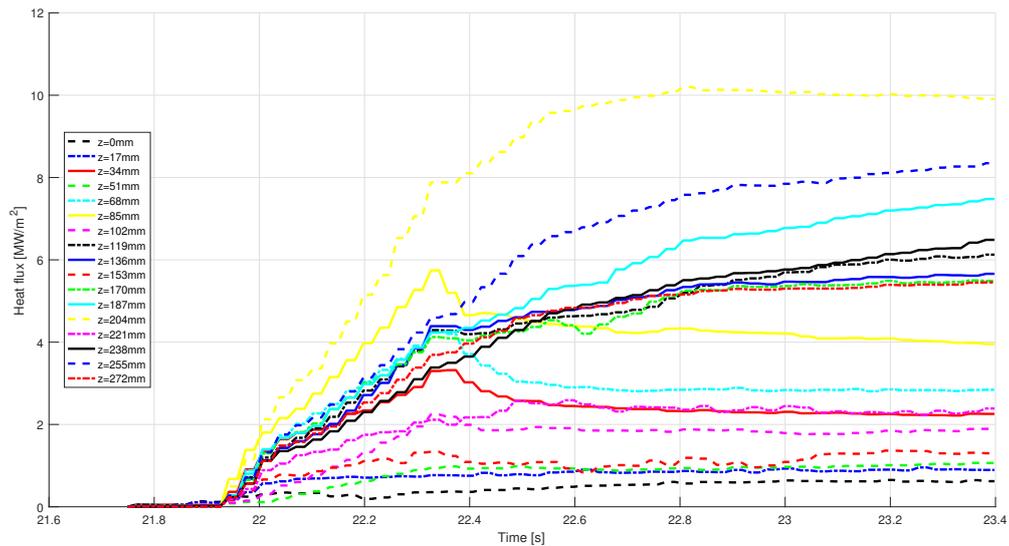
(b) Five-Elements

Figure 4.11: Calculated and Measured Temperatures Match in Thermocouples Positions Over Time

two injectors in the case of multi-elements (Fig. 4.17 and 4.19(b)).



(a) Single-Element

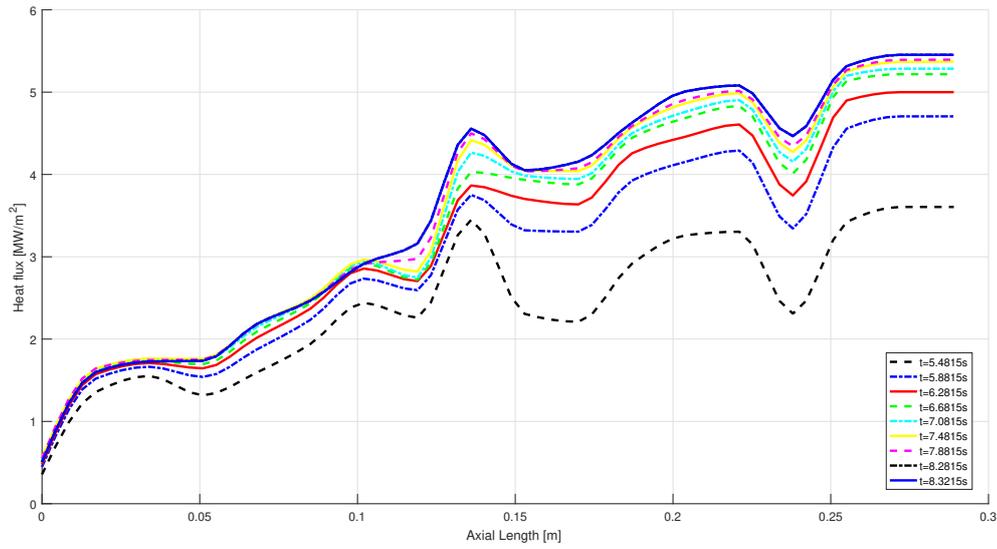


(b) Five-Elements

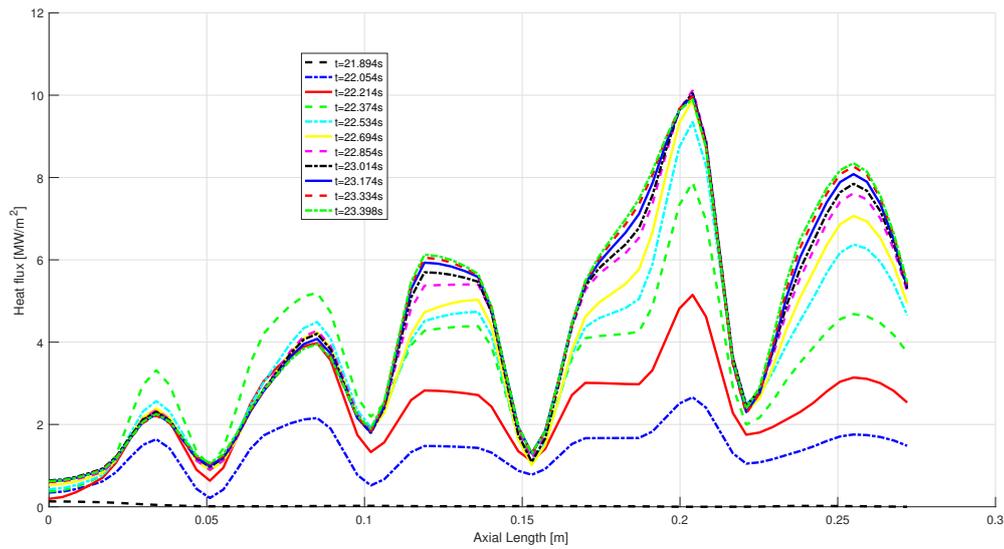
Figure 4.12: Heat Flux Over Time in Middle Point at All Distances from the Injector Plate

4.2 Code Modifications

This section presents the modifications made on the code solving the heat transfer problem on MoRaP. These modifications have been done to improve its performances in terms of computational time and quality of the solution. Since the code developed for MoRaP is



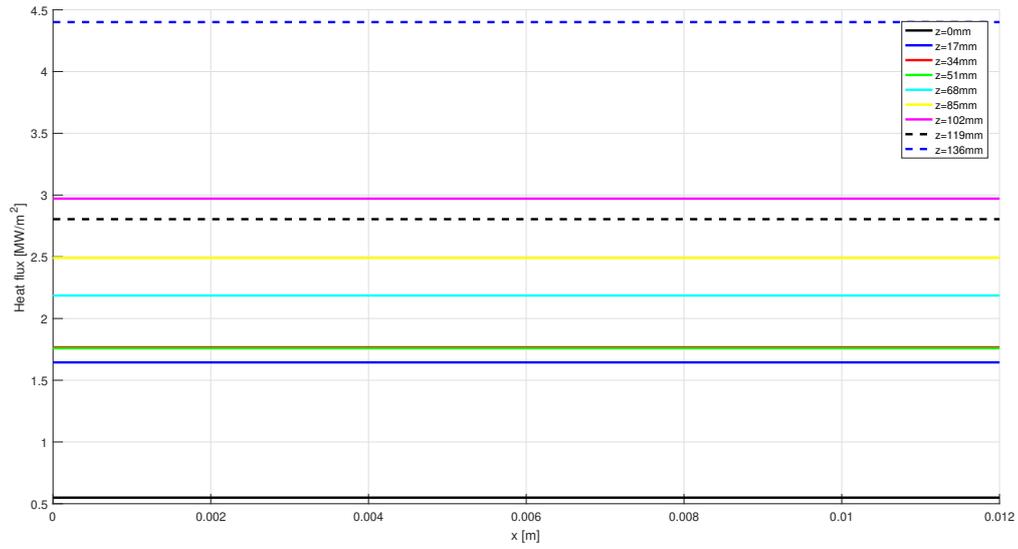
(a) Single-Element



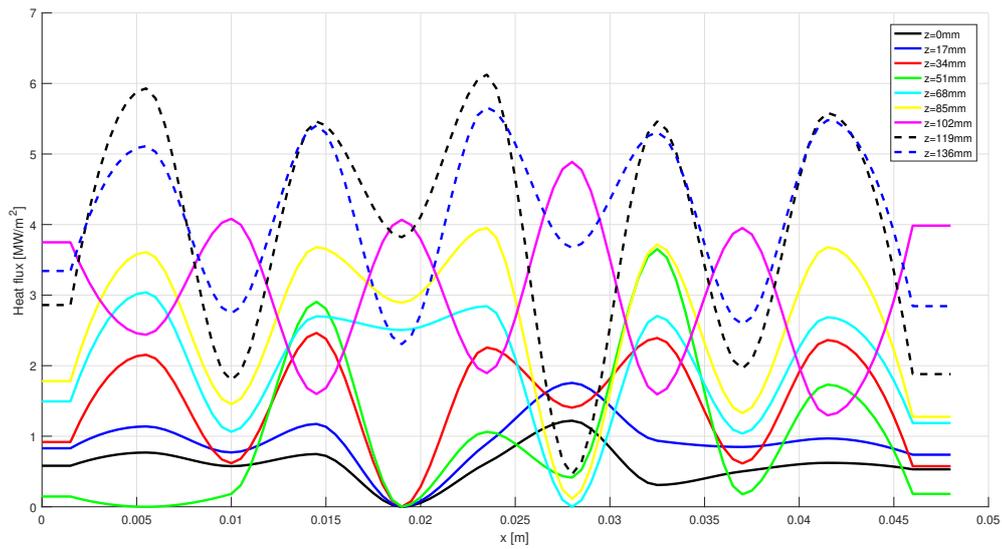
(b) Five-Elements

Figure 4.13: Heat Flux Over z in Middle Point for Several Times

quite similar to the one developed for BKM combustion chamber, the same modifications have been applied also to the latter.



(a) Single-Element

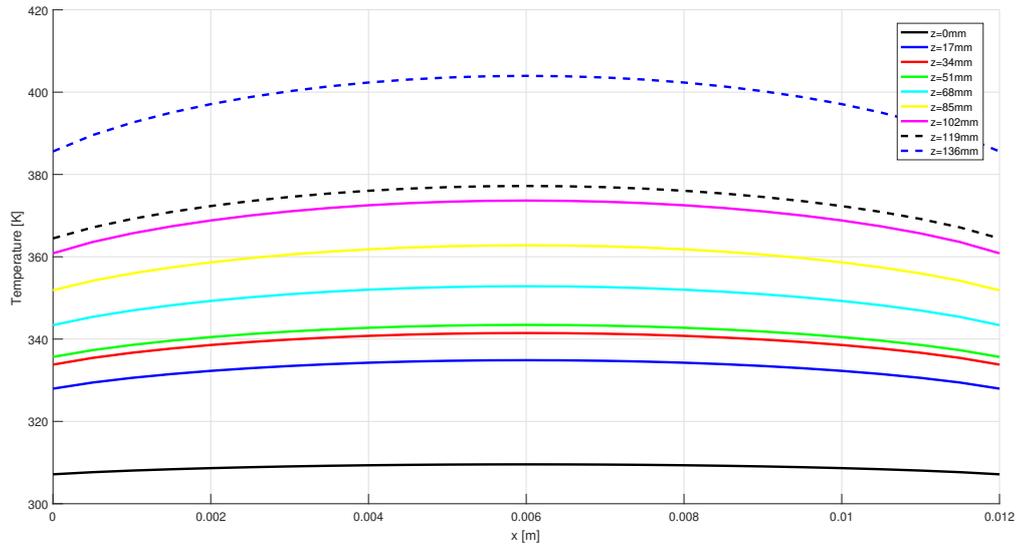


(b) Five-Elements

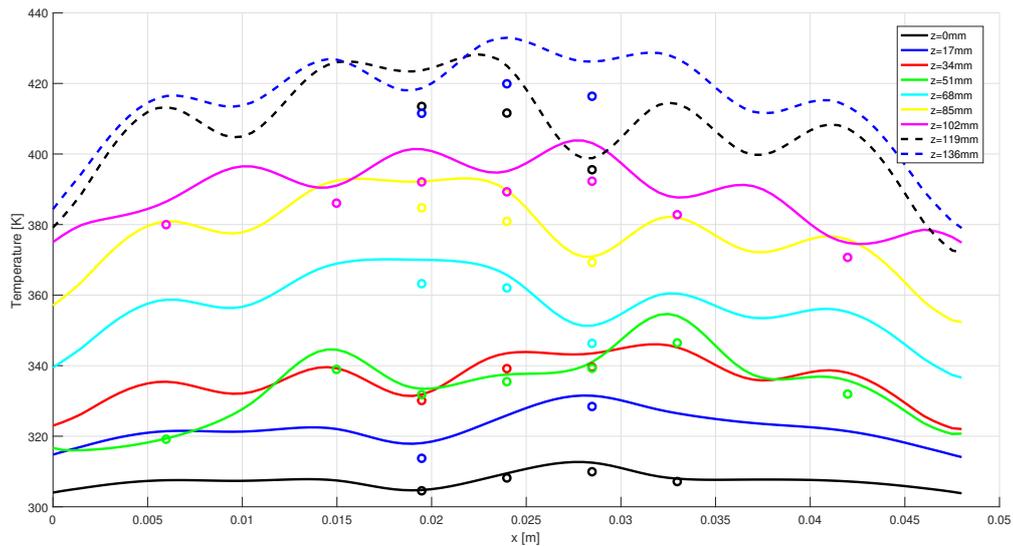
Figure 4.14: Heat Flux Over x on the Upper Hot Gas Wall at Evaluation Time for Different Planes

4.2.1 Modifications of Time Segments

From Section 4.1.3, in particular, from figure 4.7 it is clear that the direct problem is solved in each time steps, unlike the inverse problem which is solved at the end of each



(a) Single-Element



(b) Five-Elements

Figure 4.15: Computed Temperature Along x on the Upper Hot Gas Wall at Evaluation Time for Different Planes

time segment. The first important modification applied to the code is clear from figure 4.20.

One can note that now the updating process is performed in each time step as the direct problem. In this way, the computational time is significantly reduced but the obtained results with this new time handling do not show a significant deviation from previous ones,

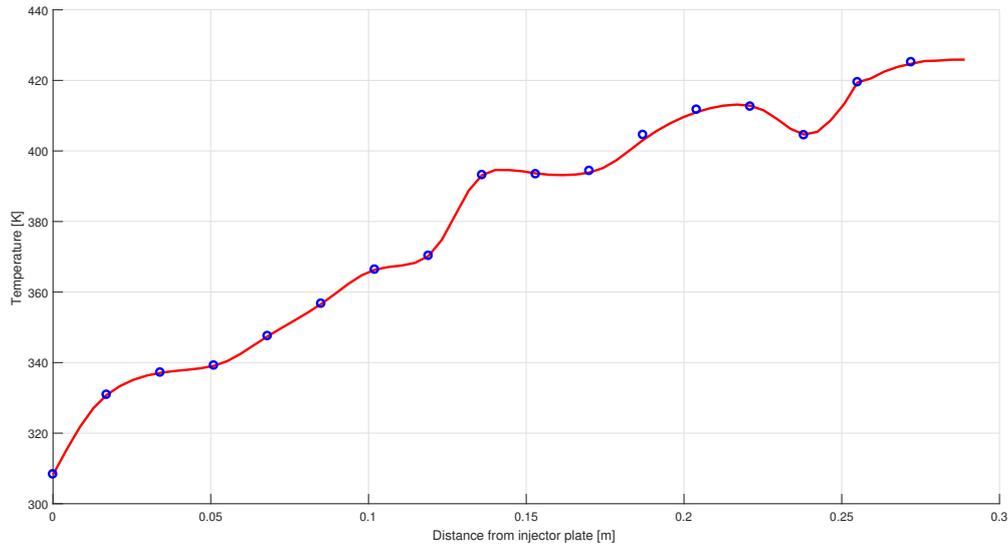


Figure 4.16: Computed Temperature Along z at 1 mm from the Hot Gas Wall in Single-Element

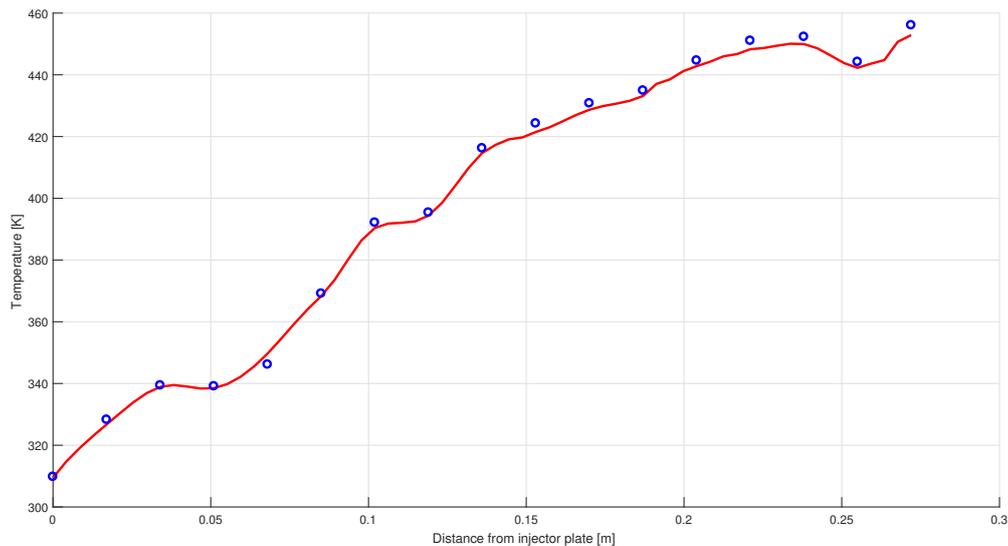
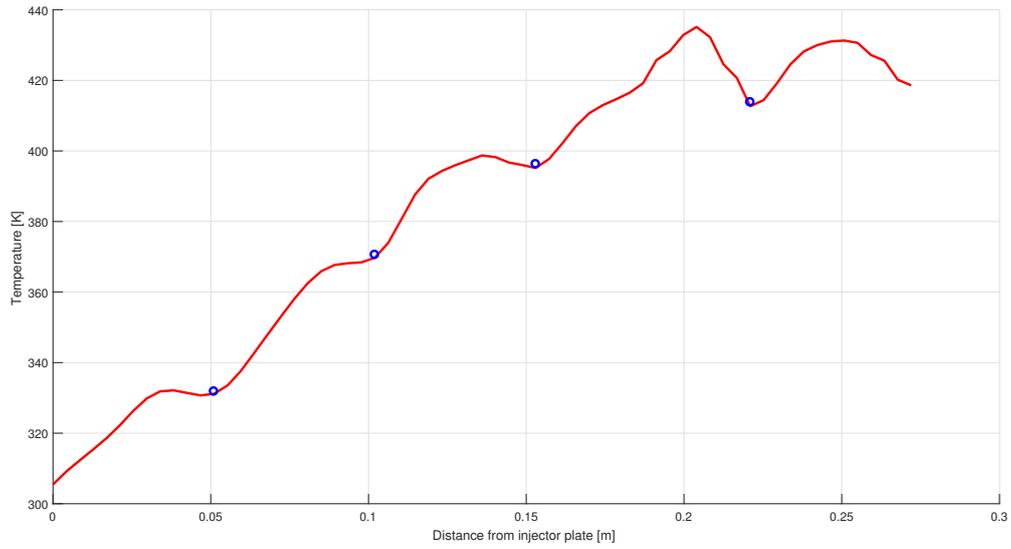
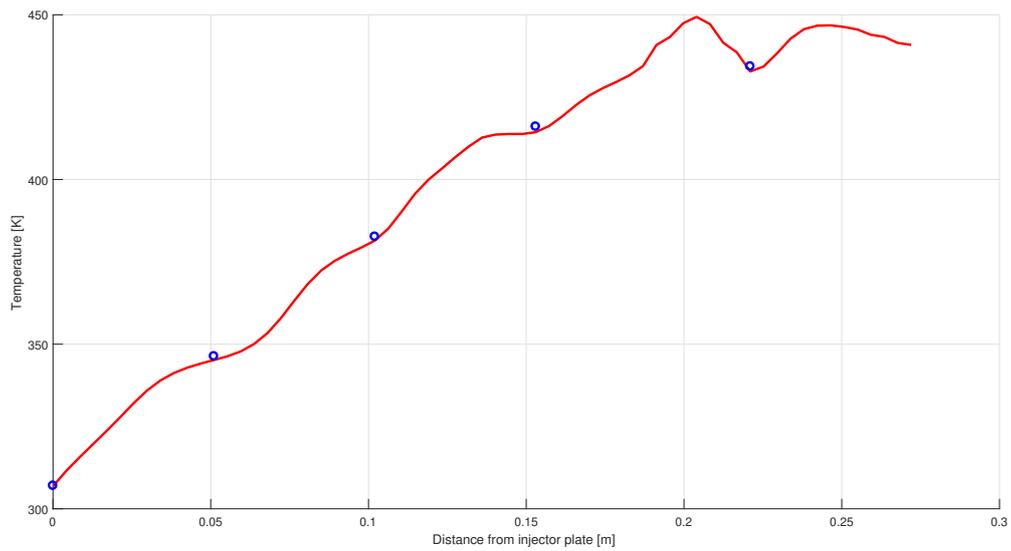


Figure 4.17: Computed Temperature Along z at 1 mm from the Hot Gas Wall at Evaluation Time in Five-Elements, 3 R

as will be clear from Section 4.3. In some tests carried out in [10], by choosing a time-step equal to 0.02, i.e. equal to the sampling interval of the thermocouples, one simulation may cost half the time with respect to the old code. This is because in each iteration only one direct problem is solved instead of two. Furthermore, it has to be noted that if the temperature is calculated in two time steps instead of one, smaller discretization errors



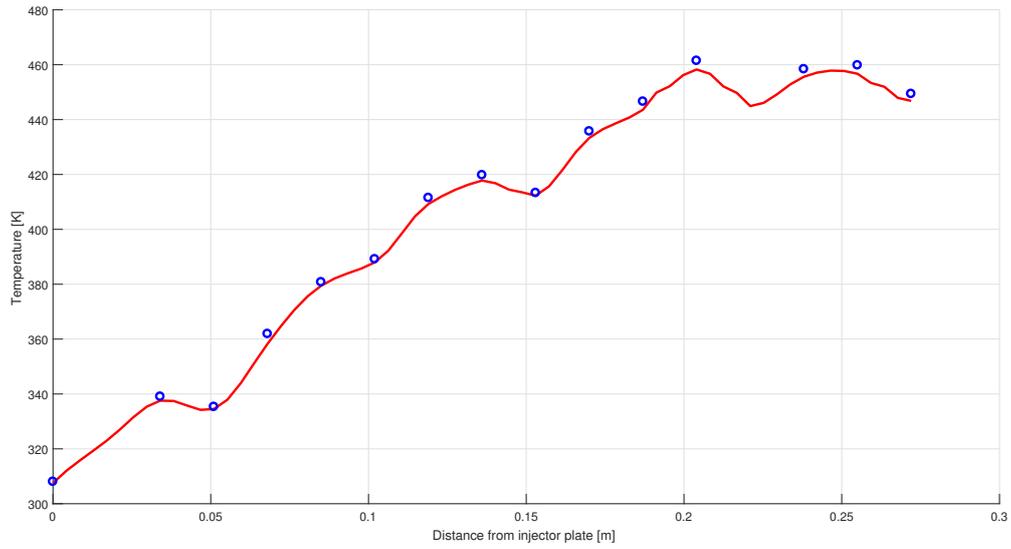
(a) 1 C



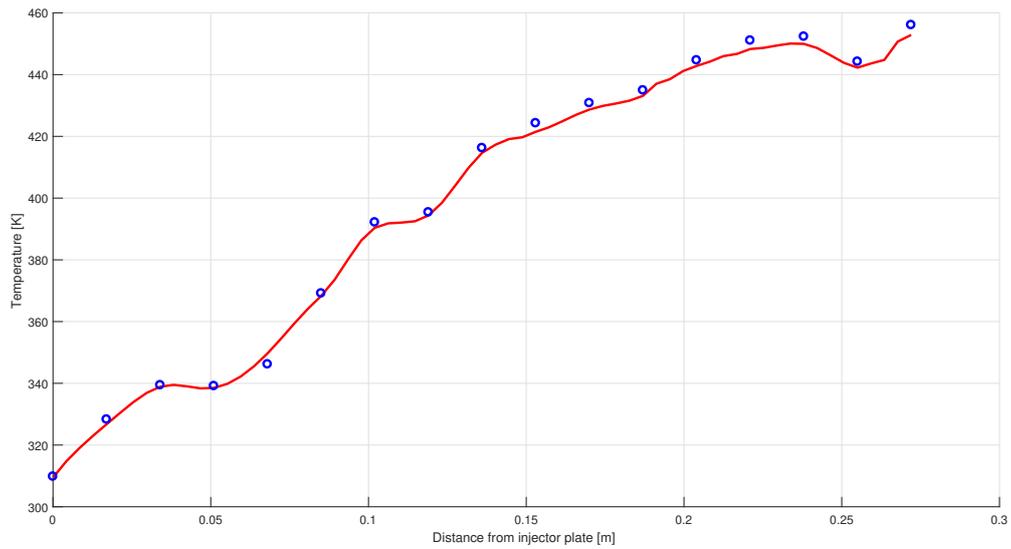
(b) 2 C

Figure 4.18: Computed Temperature Along z at 1 mm from the Hot Gas Wall at Evaluation Time in Five-Elements

are made, hence from this point of view, the previous time handling was more accurate, but the study on the errors conducted in [10] has proved that these sources of error can be negligible.



(a) 3 C



(b) 3 L

Figure 4.19: Computed Temperature Along z at 1 mm from the Hot Gas Wall at Evaluation Time in Five-Elements

4.2.2 Newton Rapshon Method

The resolution of an inverse problem in a time step implies the resolution of a non-linear system. As mentioned before, the computational temperatures T_s in thermocouples positions at time t_s depend on the values of the optimized heat flux parameters:

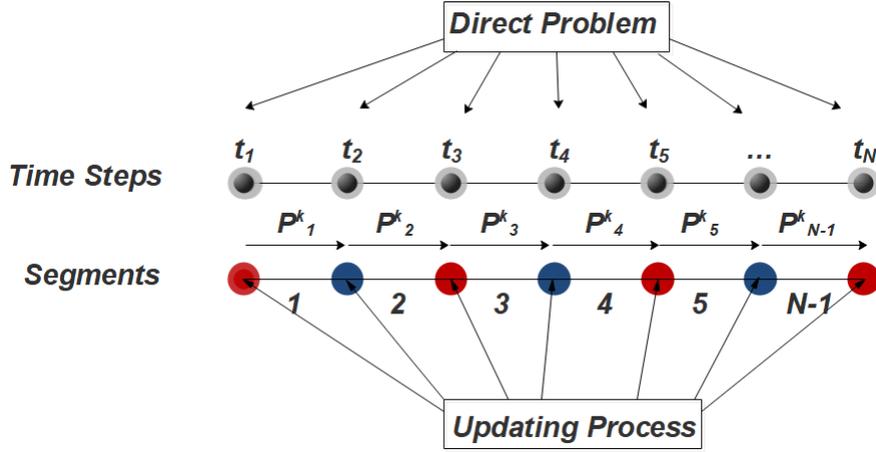


Figure 4.20: New Optimization Process on Time Segments

$$\mathbf{T}_s = \mathbf{F}(\mathbf{P}_{s-1}) \quad (4.11)$$

where \mathbf{P}_{s-1} is the vector of heat flux parameters which have to be applied as boundary conditions on the initial condition temperature \mathbf{T}_{s-1} all over the time segment $s - 1$ to obtain \mathbf{T}_s . Hence, the time segment s starts in t_{s-1} and finishes in t_s (Fig. 4.21).

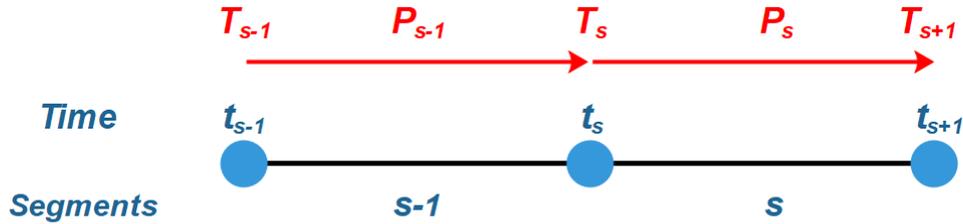


Figure 4.21: Example of Connection Between Computed Temperature and Applied Heat Flux Parameters in a Segment

Since the inverse problem aims to close the gap between computed and measured temperatures (\mathbf{Y}), the non-linear system to be solved is:

$$\mathbf{F}(\mathbf{P}_{s-1}) = \mathbf{T}_s(\mathbf{P}_{s-1}) - \mathbf{Y}_s = \mathbf{0} \quad (4.12)$$

This system can be solved using the Newton Rapshon Method (NRM) [15] by linearizing the equations starting from an initial guess and then updating the parameters until the norm of the vector $\mathbf{F}(\mathbf{P})$ is smaller than a threshold value, i.e. the value defined by stopping criteria in Section 3.4. Hence, referring to a solution of the inverse problem for a single time-step, the system (4.12) becomes:

$$\mathbf{F}(\mathbf{P}^k) + \underline{\mathbf{J}}_J^k \cdot (\mathbf{P}^{k+1} - \mathbf{P}^k) = \mathbf{0} \quad (4.13)$$

where k counts the iterations and $\underline{\mathbf{J}}_J$ is the Jacobi Matrix (JM), which conceptually is similar to the sensitivity matrix $\underline{\mathbf{J}}_S$ mentioned in Section 3.3, but their differences will be clear in Section 4.2.3 (from this moment the nomenclature adopted above will be used just to identify each matrix since the two terms are general synonyms). Hence, to obtain the heat flux parameters for the $k + 1$ iteration the linear system that has to be solved is:

$$\underline{\mathbf{J}}_J^k \cdot (\mathbf{P}^{k+1}) = -\mathbf{F}^k(\mathbf{P}^k) + \underline{\mathbf{J}}_J^k \cdot (\mathbf{P}^k) \quad (4.14)$$

where the second term on the right represents the increased heat flux in each iteration.

4.2.3 Analysis and Differences Between Jacobi Matrix and Sensitivity Matrix

In this study, some insights have been conducted on sensitivity matrix. First of all, figure 4.22 shows how a sensitivity matrix appears by computing it with the method introduced in Section 3.3, i.e. how the elements of a row of the sensitivity matrix evolve in time for this particular problem. The plot explains how much a raising in a point of the domain can affect the variation of temperature in another point. In particular, in the figure is plotted how all heat flux parameters affect the temperature raising in the parameter point 1. Higher values mean that that particular heat flux parameter has a higher influence. Hence because the parameter points 2 and 3 are physically near the parameter point 1, only the variation of the heat flux in those points produces a significant variation in temperature in the parameter point 1.

On the basis of these considerations, attempts have been made to discover if the sensitivity in thermocouples positions would have been different by computing the sensitivity matrix on the same time domain but with converged heat flux parameters. The method used for computing this new sensitivity matrix is the same discussed previously but few differences have to be highlighted: considering still a generic time segment s in figure 4.21, the first direct problem is solved step by step from the time step t_1 to t_{s+1} by imposing T_1 as the initial condition and by applying now the converged parameters heat flux for each time step and not a constant heat flux. The others N direct problems are solved on the same time domain (t_1-t_{s+1}), by perturbing one by one the heat flux parameters. The temperature obtained in this way is the result of this perturbation during the whole time domain and not only the result of a perturbation in a single time step. Figure 4.23 shows the result of

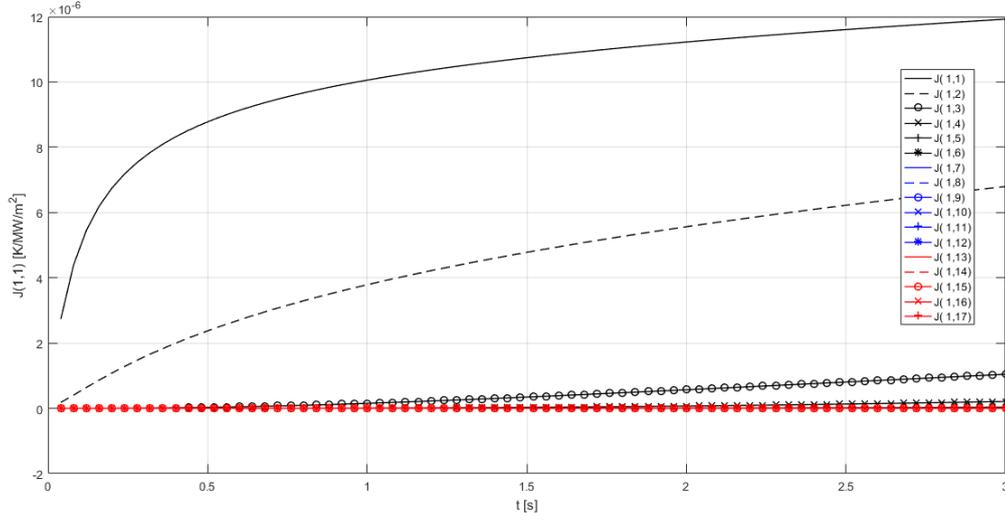


Figure 4.22: Sensitivity Coefficients

this study. The evolution of two diagonal elements of the two sensitivity matrices discussed before, i.e. the one computed using converged heat fluxes and the one using constant heat fluxes, explains that the sensitivity coefficients present no significant differences in trend in the whole time domain, except the one computed using real converged heat fluxes has smaller values than the other one. Hence this small difference in values suggests that computing just one time the sensitivity matrix is more than enough for the resolution of the problem.

The Jacobi Matrix $\underline{\mathbf{J}}_J$ unlike the sensitivity matrix $\underline{\mathbf{J}}_S$ is calculated in a single time-step but it is still a $M \times N$ (n° of thermocouples \times n° of parameters) square matrix, because the number of parameters points is equal to the number of thermocouples ($N = M$):

$$\underline{\mathbf{J}}_J(\mathbf{P}) = \begin{bmatrix} \frac{\partial T_1}{\partial P_1^k} & \frac{\partial T_1}{\partial P_2^k} & \frac{\partial T_1}{\partial P_3^k} & \cdots & \frac{\partial T_1}{\partial P_N^k} \\ \frac{\partial T_2}{\partial P_1^k} & \frac{\partial T_2}{\partial P_2^k} & \frac{\partial T_2}{\partial P_3^k} & \cdots & \frac{\partial T_2}{\partial P_N^k} \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \frac{\partial T_M}{\partial P_1^k} & \frac{\partial T_M}{\partial P_2^k} & \frac{\partial T_M}{\partial P_3^k} & \cdots & \frac{\partial T_M}{\partial P_N^k} \end{bmatrix} \quad (4.15)$$

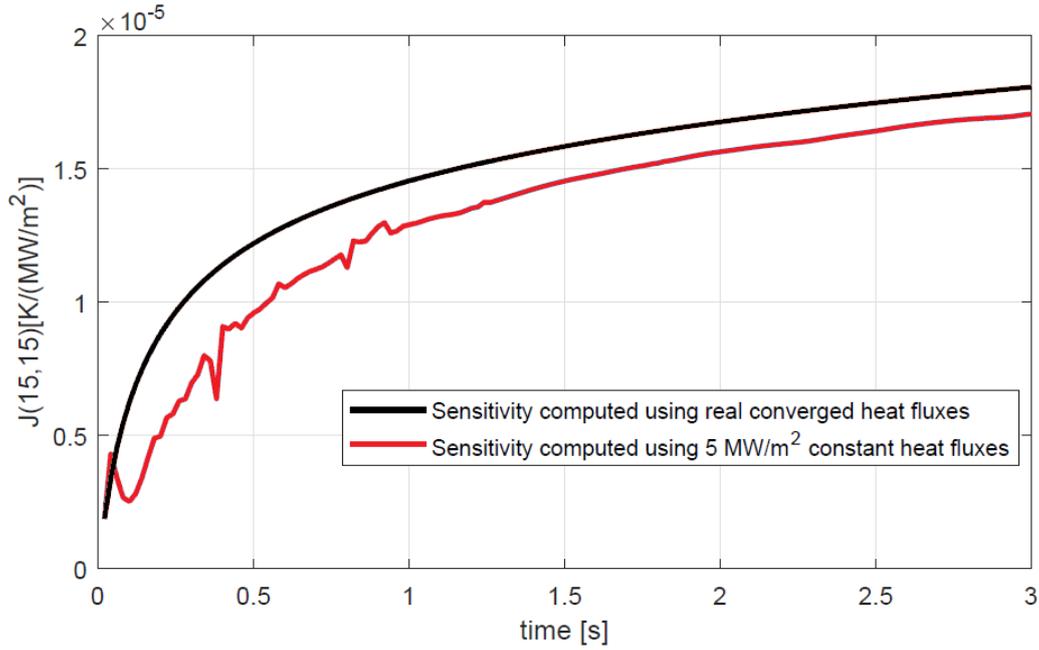


Figure 4.23: Sensitivity Matrices Elements Comparison

The method used for computing it is the same described in Section 3.3 and is presented by the equation (4.16), i.e. a forward finite difference approach of first order in which the temperatures T_i are the computational temperatures calculated at i -th thermocouple position and the parameters P_j are the heat flux parameters at j -th parameter point. But now there is a substantial difference, in fact, unlike the sensitivity matrix, in this case, the temperatures T_i are the temperatures at time t_s resulting from applying the heat flux parameters \mathbf{P} as boundary conditions from time t_{s-1} to time t_s . Hence, since this matrix depends on \mathbf{P}^k (k is the iteration), each Jacobi matrix should be calculated at each iteration.

$$J_{J_{ij}}^s \cong \frac{T_i^s(P_1^s, P_2^s, \dots, P_j^s + \varepsilon P_j^s, P_N^s) - T_i^s(P_1^s, P_2^s, \dots, P_j^s, P_N^s)}{\varepsilon P_j^s} \quad (4.16)$$

However, for this particular heat transfer problem, it has been noted that also if Jacobi matrix is computed with converged heat flux parameters, it appears to be almost constant not only if the heat flux parameters vary in a single time step but also for different time segments. The result is that since this kind of calculation implies high computational cost, it can be calculated just one time, outside the optimization loop, by applying a constant heat flux.

In this regards figure 4.24 shows that computing the Jacobi matrix in each time-step with the real starting condition of temperatures for each time-step and real converged heat flux

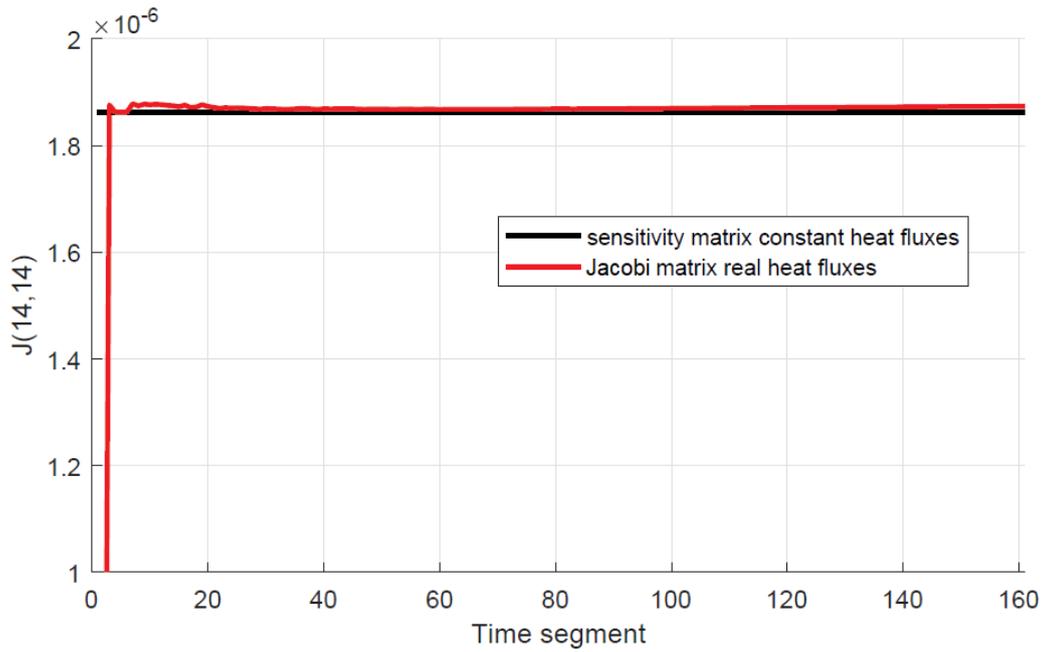


Figure 4.24: Comparison of sensitivity and Jacobi matrix All Over the Time Domain

parameters or computing the sensitivity matrix just one time outside the loop using a constant set of heat flux parameters and as the initial condition the temperatures taken from thermocouples measurement before the ignition do not imply substantial differences. Hence the Jacobi matrix can be calculated just one time, outside the loop and can be used to solve the inverse problem in all time steps.

Therefore, these studies suggest that the Jacobi matrix is almost independent on heat fluxes and time step and it can be calculated only one time out of the cycle and that the sensitivity matrix is almost independent on heat flux, but it is dependent on the time step used for calculation.

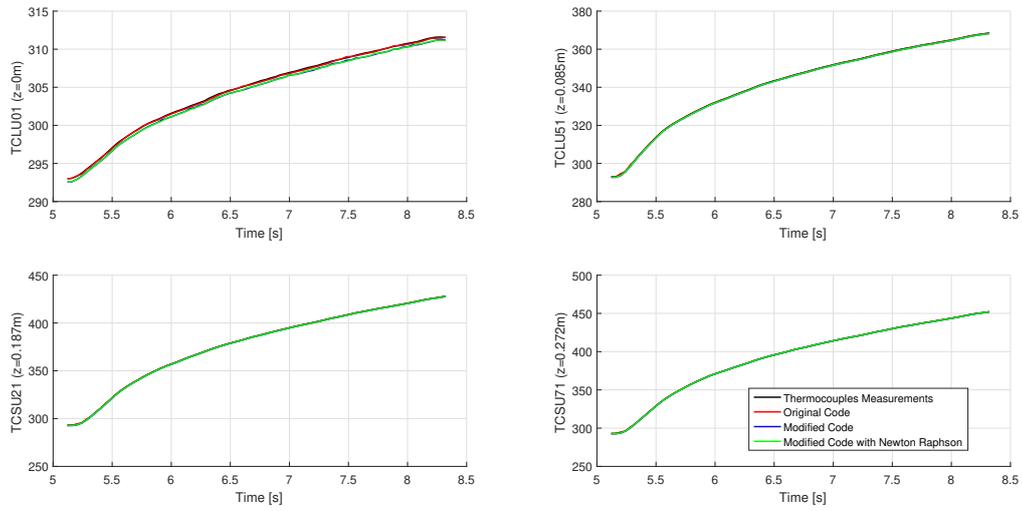
4.3 Comparison of Results

The following part presents the results of modifications made to the code. In particular two modified codes have been tested and compared to the *Original Code* (OC): one denoted as *Modified Code* (MC) to which has been modified only the time management keeping the iterative procedure unchanged and one, the *Modified Code with Newton Rapshon* (MCNR), to which in addition to the modifications on time management the Newton-Rapshon method for updating the heat flux parameters has been implemented. The investigated tests have been already introduced in Section 2.3: *data_02-05-17_16-57_0* for single-element and *Data_CCC2D - 20 - 26 - 25 - 01* for five-element. Obviously, the modifications made had as the main objective the disposal of the oscillations both in time and space or at least their limitation. As said in Section 4.1.4 the evaluation time has been set equal to $t = 341 \text{ s}$ and $t = 23.398 \text{ s}$ for single- and five-element respectively. The plot in figure 4.25, already seen in Section 4.1.4, shows the measured and computed temperatures over time, one can note that in single-element contrary to five-element the matching is quite perfect. In particular, in figure 4.25(b) there are the thermocouples positions which have been exhibit a relevant deviation between measured and computed temperatures. One can note that both in *Modified Code* and *Modified Code with Newton-Rapshon* this deviation show up at 0.5 s from ignition time and that it rises up to about 7°C and 10°C respectively at the end of simulation, however, there is to say that neither the *Original Code* has an absolute convergence.

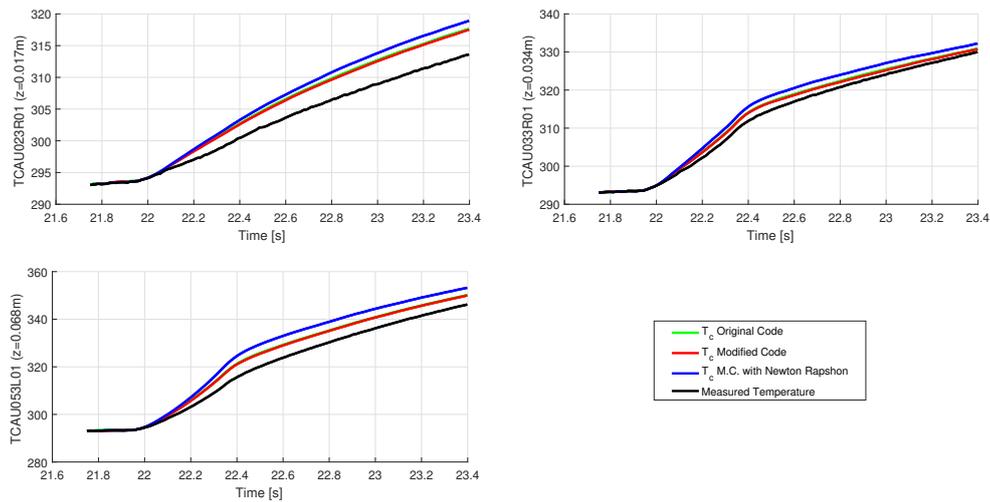
Figure 4.26 explains the heat flux evolution over time in the middle point at $z = 170 \text{ mm}$, the modifications have introduced some little oscillations in time, this is probably due to the fact that the direct problem is solved at each time step, so more discretization errors could be present. However, the heat flux appears to be the same for all codes.

Figure 4.27 shows, instead, the heat flux evolution at middle points over z at evaluation time. The *Modified Codes* have returned for all times the same heat flux evolution over z of *Original Code*. Therefore it is noticeable that despite these modifications, the oscillations still remain.

Figure 4.28 shows the heat flux profile along x at evaluation time and at several distances from injector plate, one can note that, generally, in each plane where thermocouples are located the heat flux profile is roughly the same, however, at $z = 170 \text{ mm}$ the MC appears to return a more damped heat flux profile and between $x = 30 \text{ mm}$ and $x = 40 \text{ mm}$ both *Modified Codes* return an heat flux slightly higher. This is confirmed also by the mismatching of temperatures. However, more of these variations in profiles along x have



(a) Single-Element

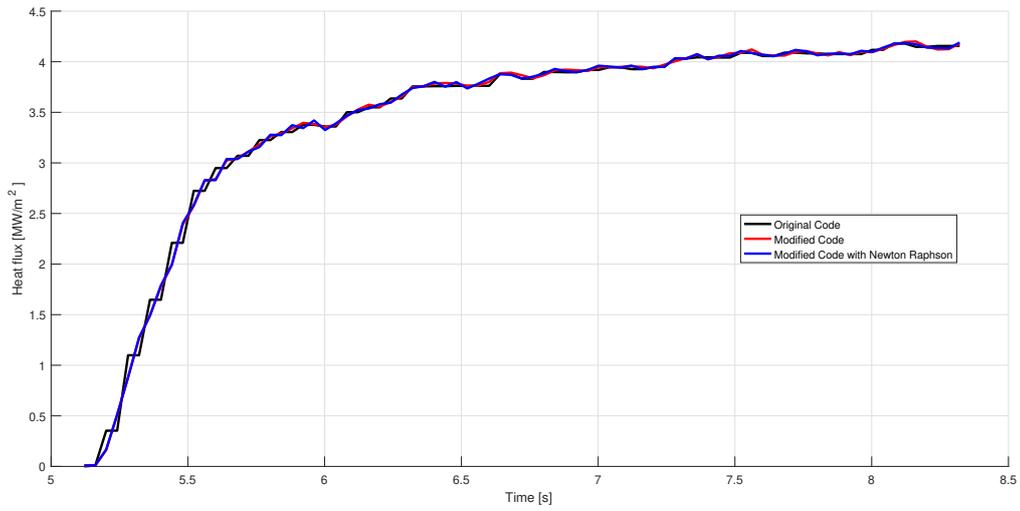


(b) Five-Element

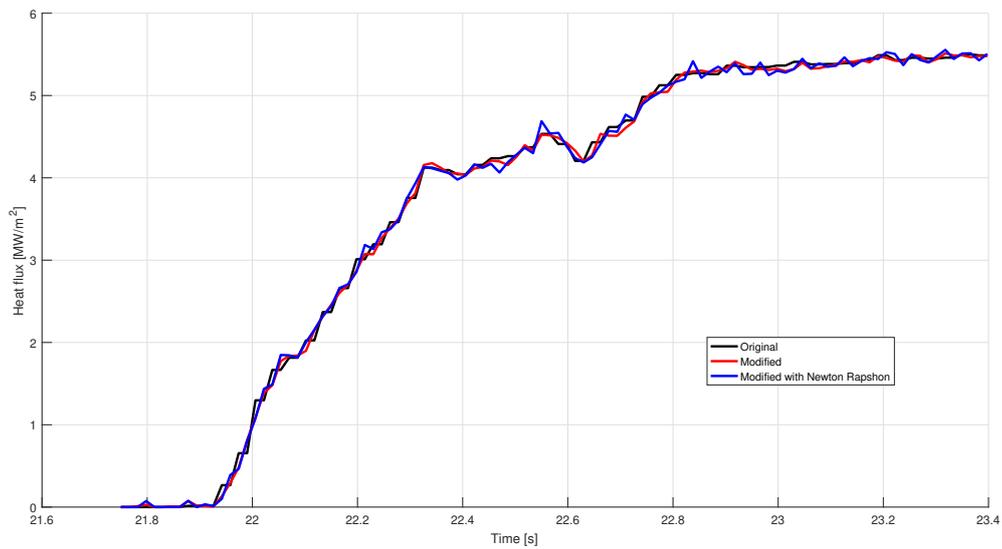
Figure 4.25: Comparison of Computed and Measured Temperature for the Three Different Codes

been found in other planes but for their small amplitude, they have been classified as non-relevant.

As far as the oscillations, even if no significant improvements have been observed, the computational time has been drastically reduced (table 4.4). In particular with regards to single-element, the computational time has been reduced by about 46% and 76% for MC and MCNR respectively. As concern the five-element, instead, the improvement has been approximately around 26% and 49%.



(a) Single-Element

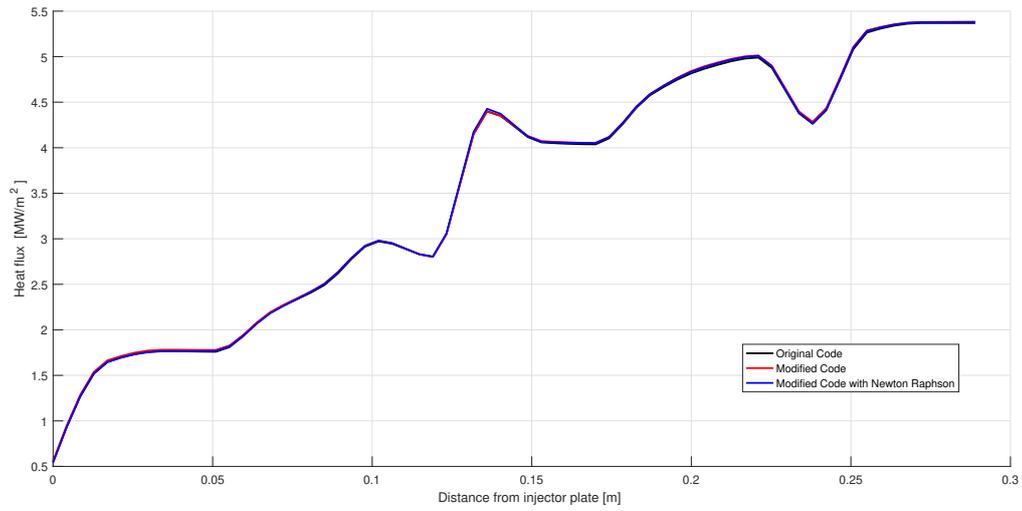


(b) Five-Element

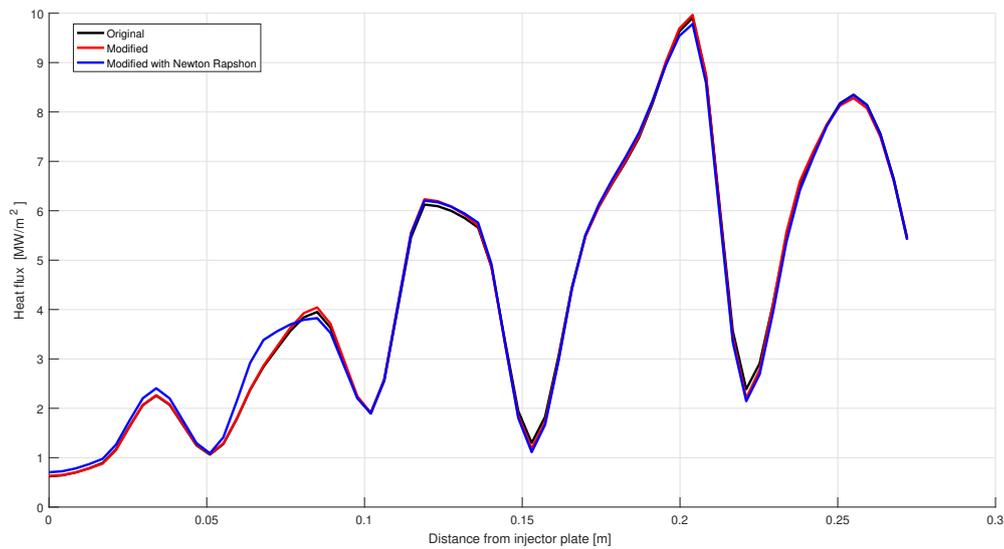
Figure 4.26: Heat Flux Over Time at Middle Point $z = 170 \text{ mm}$

Table 4.4: Elapsed computational time [h]
Intel Core i5-7200U CPU @2.50GHz

	Original Code	Modified Code	Modified Code with Newton Raphson
Single-Element	2.44859	1.33237	0.59218
Five-Element	16.889	10.88	8.67



(a) Single-Element



(b) Five-Element

Figure 4.27: Heat Flux Over z at Middle Point at Evaluation Time

This improvement is confirmed also by figure 4.29 in which the iterations necessary to convergence are plotted. In particular, one can note that these iterations progressively decrease passing from *Modified Code* to *Modified Code with Newton-Rapshon*.

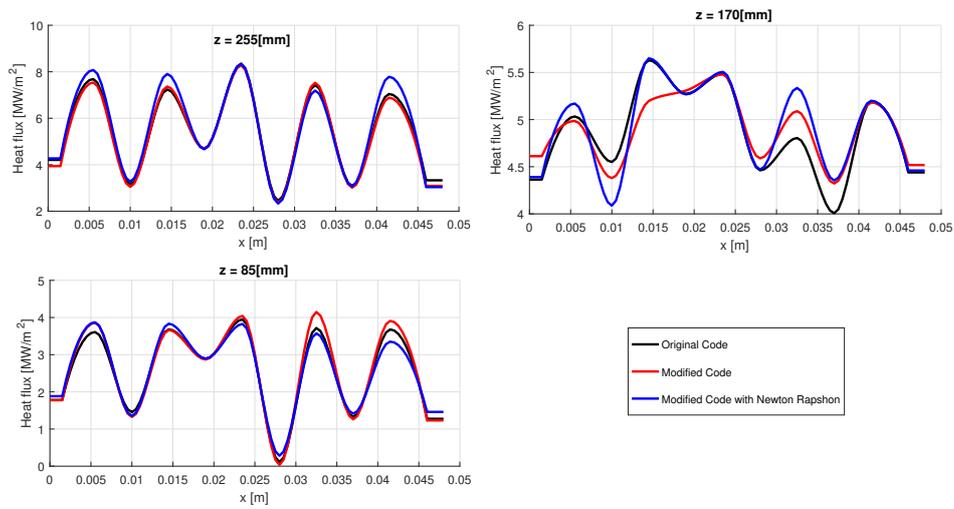
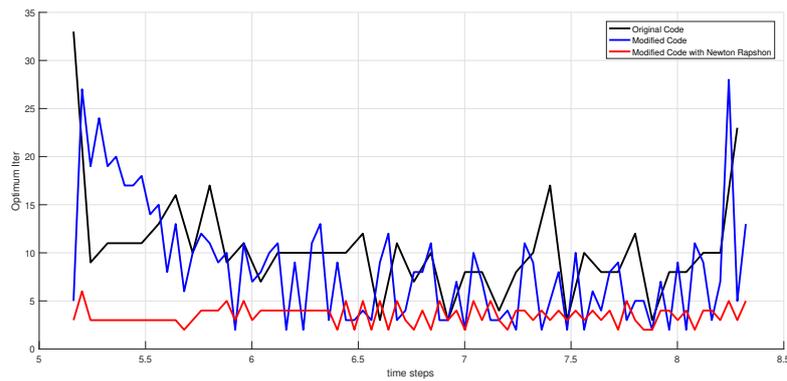
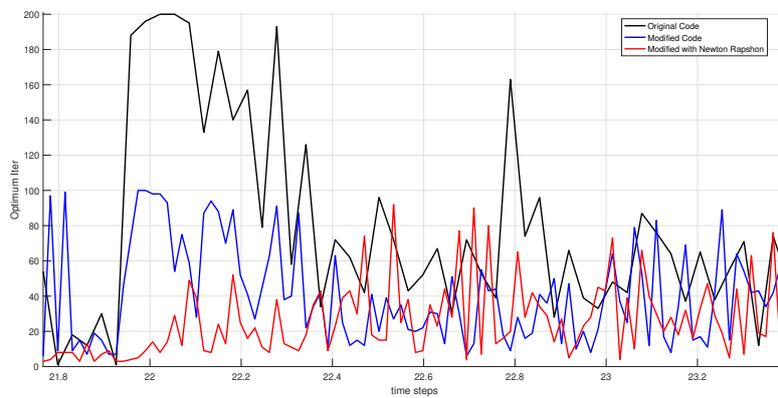


Figure 4.28: Heat Flux Over x in Five-Element Combustion Chamber at Evaluation Time for Several Planes



(a) Single-Element



(b) Five-Element

Figure 4.29: Iteration necessary to convergence for each time-step

Chapter 5

Validation of RoqFITT

Although the modifications made have been improved the performances of the code in term of computational time, the oscillations in space still affect the solution in both cases, whether it is the single- or five-element. Hence, it is not possible now to say with certainty that these oscillations are the product of the inverse code or are due to experimental data. For this reason, a validation of the inverse code has been conducted by performing several numerical experiments. The uncertainty is due to the fact that the boundary condition heat flux on the hot gas wall is unknown, so it is difficult to establish if an inverse transient heat method reproduces with precision the real heat flux. The following table explains well the passages made to carry out the numerical experiments.

Table 5.1: Numerical experiment

Passages	Instructions
1	Assign a known boundary condition on the hot gas wall for each timestep
2	Assign a starting condition temperature for the all domain
3	Solve the direct problem with assigned boundary condition for each time step
4	Once the temperature evolution in space and in time is available from the resolution of the direct problem, estimate the temperatures in thermocouples position and assume them as measured temperatures by thermocouples
5	Perform the inverse code
6	Compare the heat flux boundary conditions evaluated by inverse code with the boundary condition assigned to solve the direct problem

The numerical heat flux set as the boundary condition on the hot gas wall is shown in figure 5.1. In the reality when a combustion chamber is ignited in the first time instants (about 0.5 s) there is always an oscillation in heat flux caused by the igniter that progressively vanishes as the temperatures increase. The fact that these oscillations still remain in the

solution provided by the inverse code has encouraged to insert in the numerical experiment a big oscillation over z which appears and disappears gradually. The goal was to verify if the code can follow this changes in heat flux as well as to ensure that it does not introduce any numerical meaningless heat flux, i.e. relevant numerical errors.

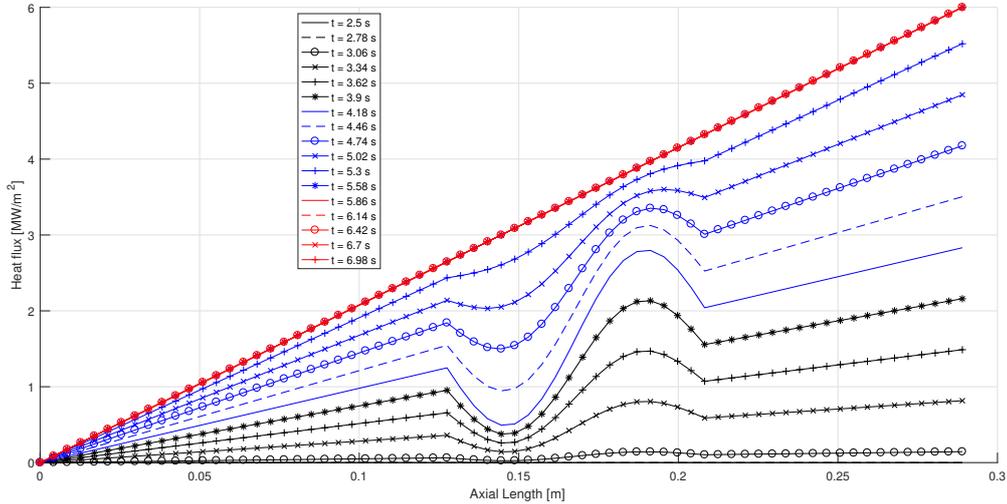
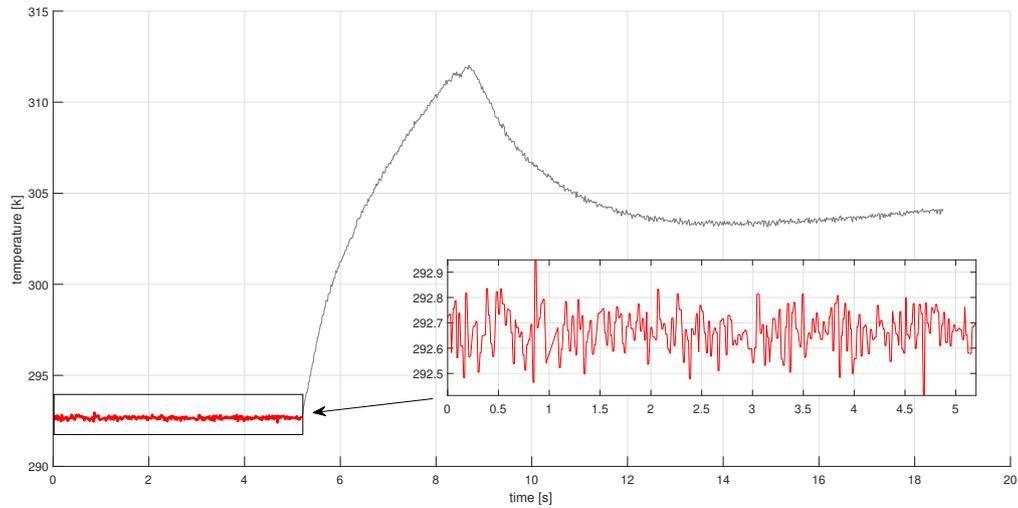
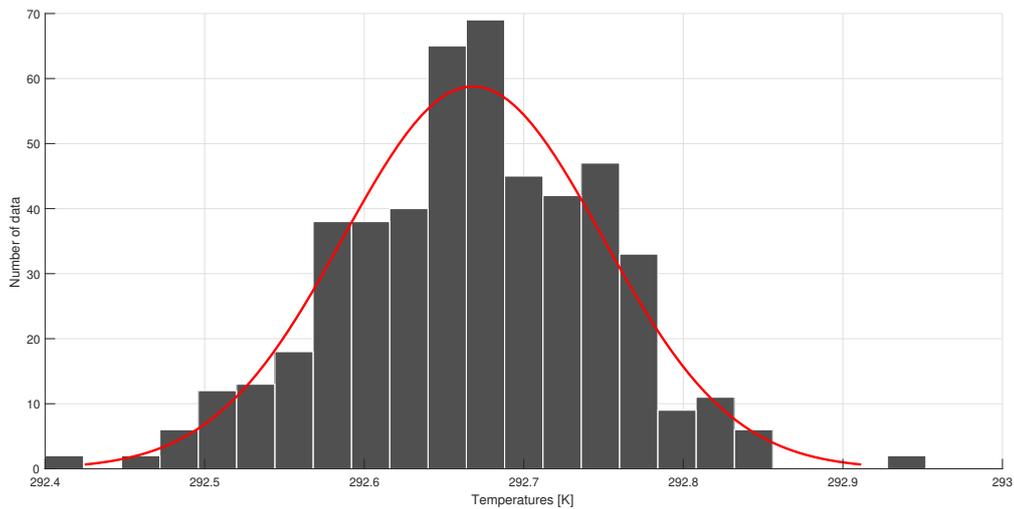


Figure 5.1: Numerical Heat Flux Evolution Along z for several time steps

The computational time domain of the direct problem goes from 2.5 s to 7 s. The heat flux evolution along z has been obtained from the sum of a linear function and a sinusoidal function. It presents a big oscillation which rises gradually after 1 s from the start and vanishes completely at 6 s, over the last 1 s the heat flux returns linear. Furthermore, it is possible to appreciate the sudden change in shape along z when the oscillation reaches the maximum amplitude, this not smoothed effect has been intentionally inserted to verify if these spatial rapid changes are detected by the code and/or amplified. By performing the direct problem with this heat flux applied on the hot gas wall, the temperatures are computed in thermocouples positions and set as measured temperatures (Fig. 5.3). Once the temperatures in thermocouples positions have been available, random errors have been added under the assumption that they present a Gaussian distribution. The following study supports this hypothesis. The thermocouples, like all sensors, present precision errors, by analyzing the evolution of their readings before the ignition (Fig. 5.2(a)) it has been noted that it exhibits a distribution close to the normal distribution (Fig. 5.2(b)). To verify this assumption, the two quantities, Skewness (S) and Kurtosis (K), have been calculated for several sets of readings.



(a) Measured Temperatures



(b) Distribution of the Readings

Figure 5.2: Precision Errors in Thermocouples Readings Before the Ignition

In accordance to [6] the Skewness value identifies if the data distribution is symmetric or skewed to one side. If the value is positive, it means that the distribution is skewed right, otherwise, with a negative value, it is skewed left. If $S = 0$, the data are perfectly symmetric. Instead, the Kurtosis describes how tall and sharp the central peak of the distribution is. As [6] refers in his article "higher kurtosis means more of the variance is the result of infrequent extreme deviations, as opposed to frequent modestly sized deviations.". If its value is exactly 3 the distribution is called *mesokurtic*.

For a set of N measurements the Skewness formula is:

$$S = \frac{(1/N) \sum_{i=1}^N (x_i^3 - \bar{x}^3)}{\sigma^3} \quad (5.1)$$

where σ is the standard deviation.

$$\sigma = \sqrt{(1/N) \sum_{i=1}^N (x_i^2 - \bar{x}^2)} \quad (5.2)$$

The Kurtosis is:

$$S = \frac{(1/N) \sum_{i=1}^N (x_i^4 - \bar{x}^4)}{\sigma^4} \quad (5.3)$$

The Skewness and Kurtosis of a normal distribution are 0 and 3 respectively. Hence these values are taken as reference and compared with the values estimated from the data. The results of this analysis are shown in table 5.2. The Skewness and Kurtosis of the set of measurements are very close to the values of the normal distribution and for this reason, it has been possible to approximate the distribution of the errors as the normal distribution.

Table 5.2: Skewness and Kurtosis comparison

	<i>Data</i>	<i>Gaussian Distribution</i>
<i>Skewness</i>	-0.1423	0
<i>Kurtosis</i>	3.1950	3

Therefore, this expedient has been done by using the Matlab command *normrnd*. The standard deviation inserted as input has been calculated ($\sigma = 0.08$) considering again the set of data before the ignition (Fig. 5.2(a)), but to emphasize any possible oscillation in solution, it has been set equal to 0.15 in all simulations.

Once the temperatures coming from direct problem have been available, three inverse codes have been performed, the goal was to discover what kind of effects in solution could be introduced by the oscillations in temperatures readings with different magnitude:

- *Not smoothed*, the numerical temperatures have not been smoothed;
- *Smoothed on 3 point*, the numerical temperatures have been smoothed in time using a moving average on 3 points;
- *Smoothed on 10 point*, the numerical temperatures have been smoothed in time using a moving average on 10 points;

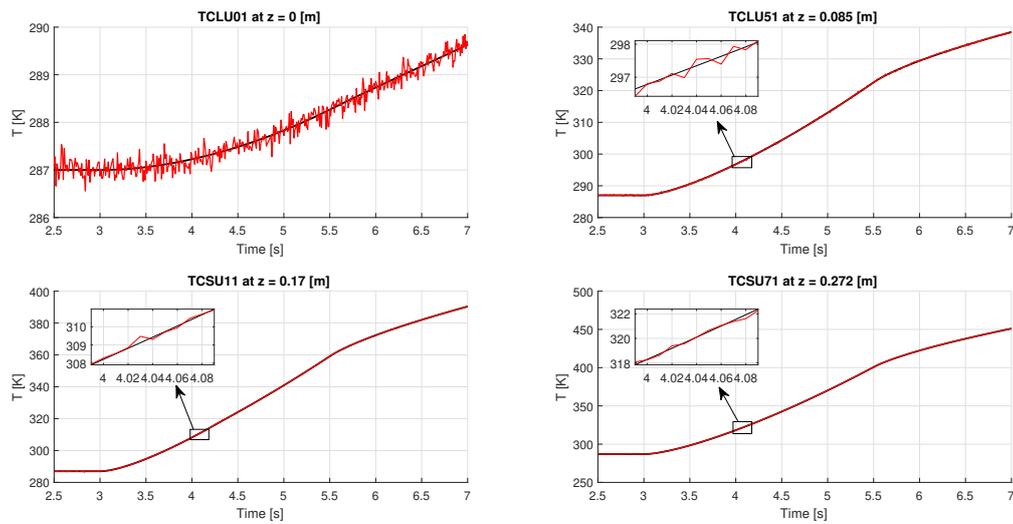


Figure 5.3: Normal Distribution Errors Added to Numerical Temperatures in Thermocouples Position

Furthermore, one can note that these errors are denoted as precision errors and that no accuracy errors have been considered in this analysis. The results are presented below.

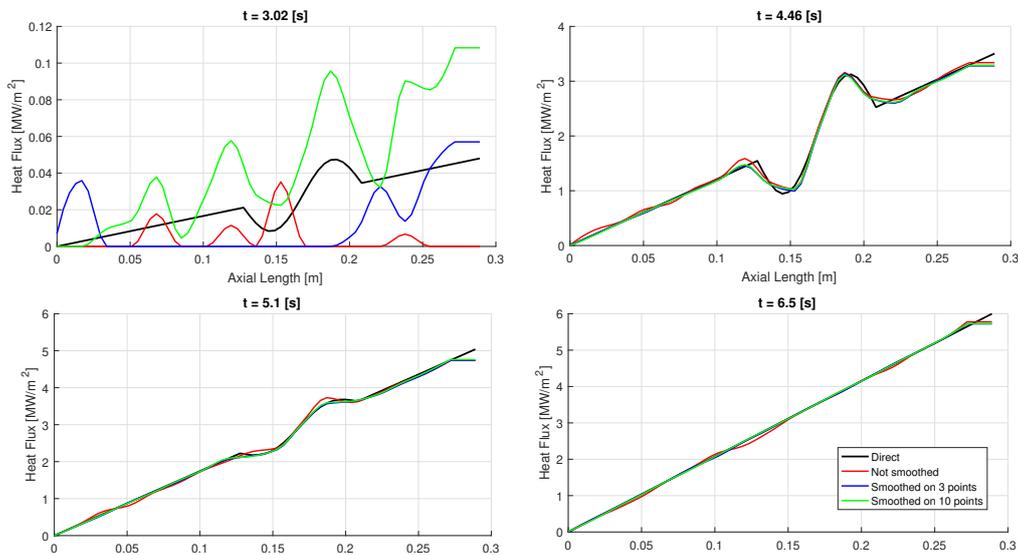


Figure 5.4: Heat Flux Evolution Along z at Different Time for the Three Different Simulations

Figure 5.4 compare the solutions provided by the three different simulations with original known heat flux set as the boundary condition to the direct problem. The first thing that becomes clear is that all three codes reproduce extremely well the original heat flux. In the first time segments ($t = 3.02$ s) when the heat flux is small in magnitude, one can appreciate that all three simulations fail to estimate the right heat flux, this is probably

due to the fact that the magnitude of the heat flux is comparable with the magnitude of numerical errors (about 10^4 W/m^2). The flat in heat flux lying at the end of the combustion chamber is due to the fact that the code assigns a constant heat flux between the last thermocouple and the end of the combustion chamber. Looking at the solution for $t = 4.46 \text{ s}$ it is possible to note that the code recognizes the sudden change in value of the heat flux along z , this big oscillation do not introduce any kind of problem for the code. Instead, the figure representing the heat flux at $t = 5.1 \text{ s}$ demonstrates that the code also recognizes when this oscillation drops gradually. Finally the plot at $t = 6.5 \text{ s}$ shows that, the whole time the heat flux remains constantly linear in space, the simulations do not introduce any kind of significant errors. Apart from the big oscillation intentionally introduced and well detected by the inverse code, the small oscillations appearing along z and over time are caused by the different choices about the smoothing. However, these oscillations are not so relevant compared to the big one, but one can note that the *Not smoothed* solution, compared to the other two, generally exhibits more oscillating parts. This result is confirmed by analyzing their magnitude through the Root Mean Square Error (RMSE) calculated along z for all time steps (Fig. 5.6). Generally, one can note that, by smoothing in time using a moving average on 10 points, the errors made in each time step are lower in amplitude than in the other cases.

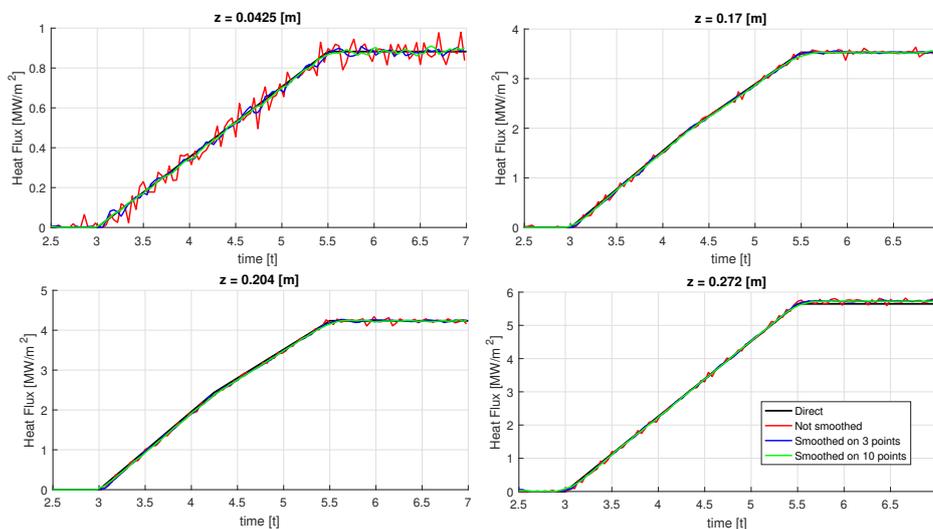


Figure 5.5: Heat Flux Evolution Over time at Different z for the Three Different Simulations

Figure 5.5 shows the attitude of the codes to reproduce the heat flux over time. One can note that, since the solutions of the inverse method are very sensitive to the measurement errors of the sensors, the most oscillating solution is provided by the code performed by

not smoothing the thermocouples measurements. Hence, from this study some important conclusions can be highlighted:

- The code detects well any kind of heat flux oscillation which gradually appears and disappears;
- The code can reproduce well the heat flux also if it presents sudden changes both in time and in space;
- Smoothing the thermocouples measurements is essential to reduce the small oscillations introduced by the oscillations in temperature readings;

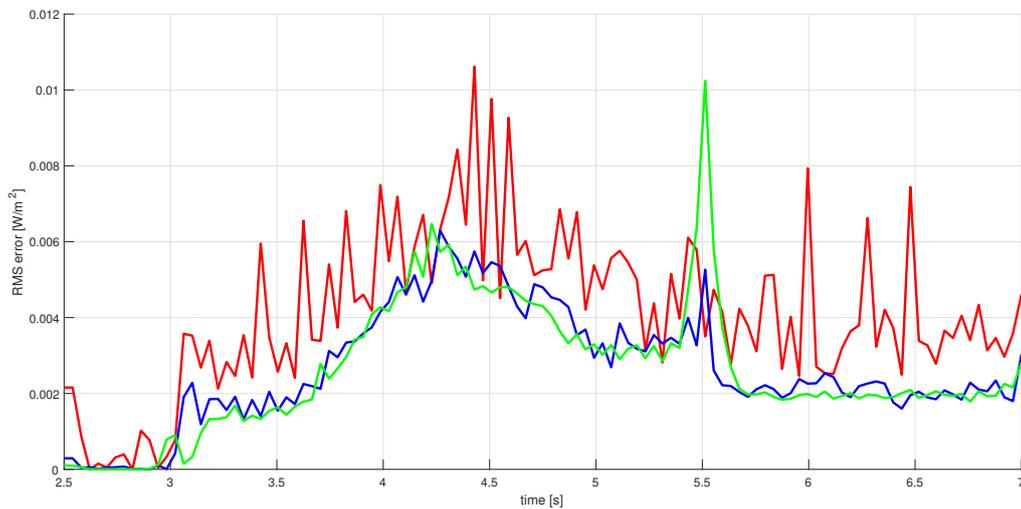


Figure 5.6: Root Mean Square Error Calculated Along z for Different Time Steps

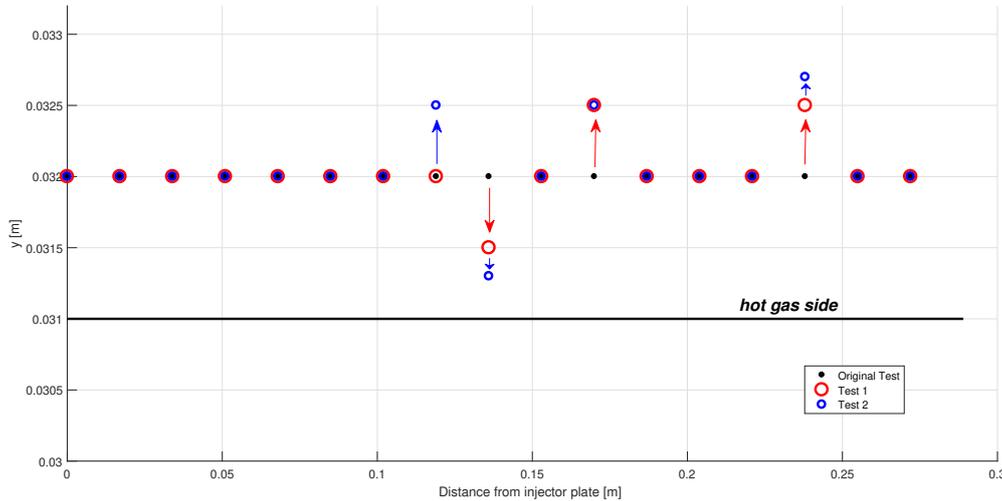
Therefore, if the problem of these oscillations does not lay in the way code works, there is something else that should cause them. The suspicion that, during the hot run, the thermocouples could move from their original position has led to the idea that by varying slightly their location into the code (*Thermocouples.Locations*) the oscillations could be mitigated. Since it is not sure if during the hot run a thermocouple is moving away from the hot gas wall or closer to it, the idea was to relocate the thermocouples on the basis of the magnitude of the heat flux in that position. The coordinate y has been the only one to be modified, the others (x and z) have been left unchanged. The criterion used to move them was: looking at the heat flux over z , whenever a big oscillation is detected if its overshoot appears to be anomalous it can mean that that thermocouple is recording a temperature value too high for that position, hence it has to be moved closer to the hot

Table 5.3: Thermocouples coordinates and movements [mm]

Thermocouples	Original Test		Test 1		Test 2	
	y	z	y_{new}	Movement	y_{new}	Movement
TCLU71	32	170	32	0	32.5	+0.5
TCLU81	32	153	31.5	-0.5	31.3	-0.7
TCSU11	32	119	32.5	+0.5	32.5	+0.5
TCSU11	32	51	32.5	+0.5	32.7	+0.7

gas side. Figure 5.7 and table 5.3 show exactly how the thermocouples have been moved. The *Original Test* is the test of Section 4.1.4.

From table 5.3 it is clear that the number of thermocouples moved in *Test 1* is 3, while 4 are the thermocouples moved in *Test 2*.

**Figure 5.7:** Thermocouples Arrangement After Their Movement

The result of these movements is presented in figure 5.8. Immediately, it is clear that the new thermocouples arrangement has produced the desired effect. In fact, it is possible to appreciate that, by moving the thermocouples away from the hot gas wall ($z = 119$ and 51 mm in Test 1 and $z = 170$, 119 and 51 mm in Test 2), the estimated heat flux on the hot gas wall appears to be higher than the ones estimated in the *Original Test*. This is due to the fact that in that positions the thermocouples were recording temperatures which are too lower compared to the normal trends. Instead, by moving the thermocouples closer to the hot gas wall ($z = 153$ mm in both tests) the opposite effect is produced, i.e. the estimated heat flux appears to be lower and the peak is reduced. Since it is impossible to judge the magnitude of thermocouples movements, it is not sure

that it is the right way to get rid of the oscillations over in space. Anyway, it is clear that the vibrations appearing during the hot run can seriously cause the movements of the thermocouples from their original position, and since the code is really sensitive to the smallest variations of temperatures, this phenomenon can be a real problem which calls into question the effectiveness of an inverse method.

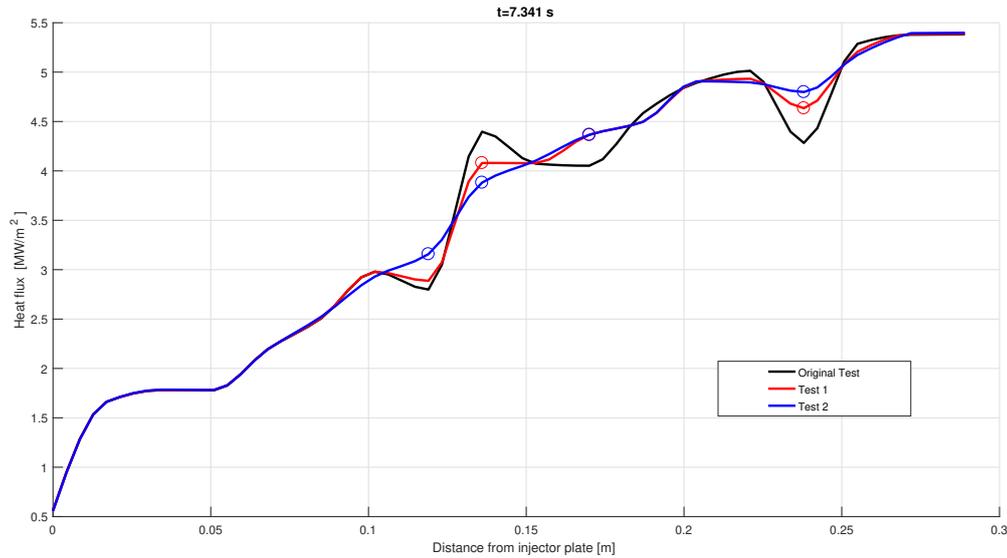


Figure 5.8: Heat Flux Evolution Over z at Evaluation Time

5.1 Effect of the Different Nozzle Boundary Conditions

This small section is dedicated to analyzing the effects on the solutions carried out by the code caused by different boundary conditions on nozzle plane. Until now all performed simulations have been characterized by the assumption of an adiabatic nozzle plane, i.e. no heat flux has been assumed to flow from the combustion chamber to nozzle block or vice-versa. The boundary condition used in the following simulations is a Neumann boundary condition, the heat flux flowing through the wall is a priori set. Looking at the table 5.4 the simulations have been conducted, first, by applying a constant heat flux, but then a more realistic behavior of the heat flux has been set. Recalling the heat diffusion equation (3.2), the Neumann boundary condition can be expressed as:

$$\frac{\partial T(\mathbf{x}, t)}{\partial \mathbf{n}} = f(\mathbf{x}, t) \quad \forall \mathbf{x} \in \partial\Omega \quad (5.4)$$

where \mathbf{n} denotes the normal to the boundary $\partial\Omega$ (Nozzle plane), f is a known scalar function and $\Omega \subset R^3$.

Hence a total of 5 simulations have been carried out as you can see from table 5.4, in the first 4 different constant heat fluxes have been set on nozzle boundary, the last simulation instead provide for a more realistic heat flux boundary condition, i.e. it is considered that this heat flux must change in space and in time. All these boundary conditions are visible in figure 5.9.

Table 5.4: Nozzle heat flux boundary condition MW/m^2

<i>Tests</i>	<i>Boundary Conditions</i>	
<i>Original Test</i>	0	<i>Constant</i>
1	0.3	
2	0.6	
3	0.8	
4	1	
5	$f(\mathbf{x}, t)$	<i>Variable</i>

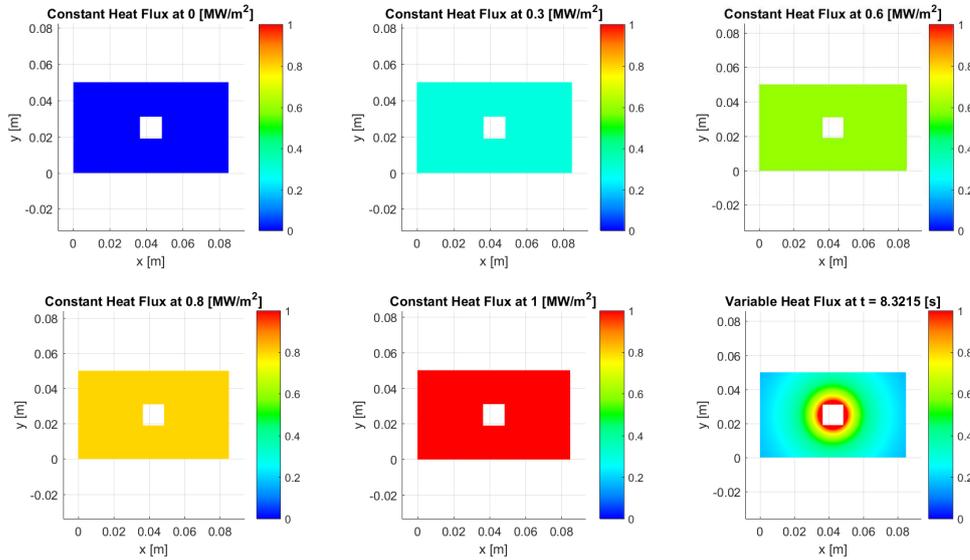


Figure 5.9: Heat Flux Boundary Condition on Nozzle Plane for the Different Tests

The results of the simulations are presented below from figure 5.11 to figure 5.12.

Figure 5.10 shows the evolution of the heat flux over time for the upper middle point for the different tests. Figure 5.11 shows the heat flux trend over z in the upper middle points at $t = 8.3215$ s. Finally, figure 5.12 shows the evolution of the computed temperature over

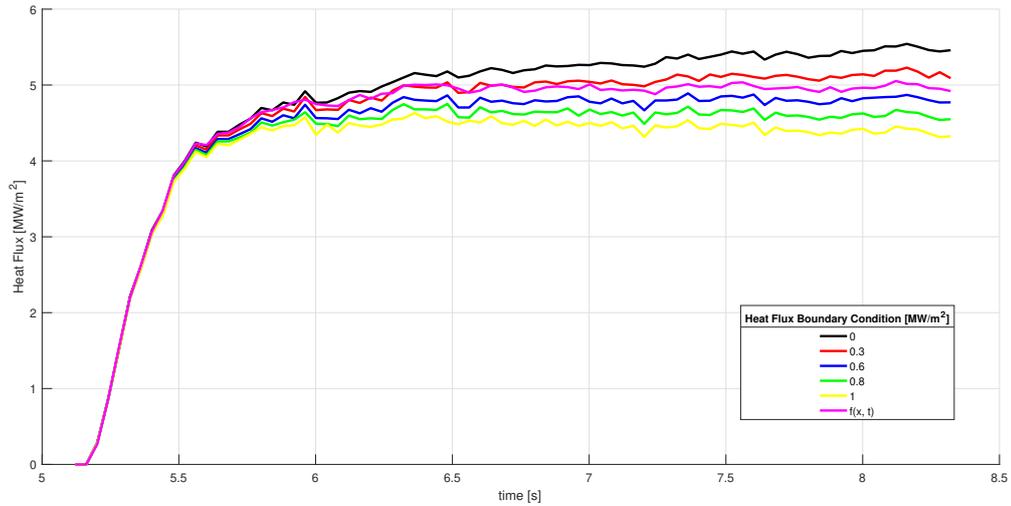


Figure 5.10: Estimated Heat Flux Over Time for the Upper Middle Point at $z = 0.289 \text{ m}$

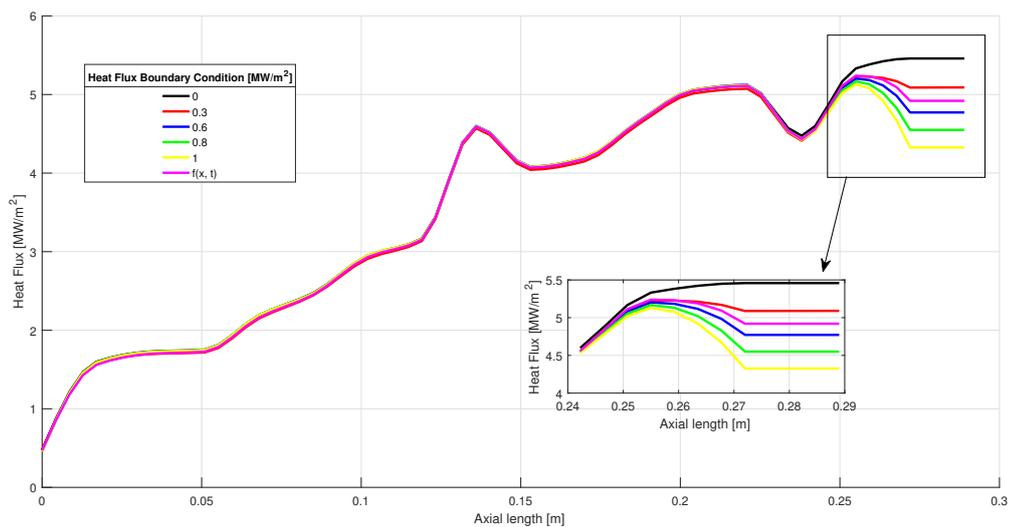


Figure 5.11: Estimated Heat Flux Over z in the Upper Middle Points at $t = 8.3215 \text{ s}$

z at the time step $t = 8.3215$. The first thing that becomes clear from these figures is that assuming an adiabatic boundary condition on nozzle plane ensures an error in heat flux and temperature estimation only restricted to the boundary zone, i.e. near the nozzle. In fact, from injector to approximately $z = 0.25 \text{ m}$ the heat flux evolution of all tests (1 to 5) is perfectly equal to the estimated heat flux coming from the *Original Test*, while from $z = 0.25 \text{ m}$ to the end of combustion chamber the heat flux starts to take different values. In particular, one can note that while assuming an adiabatic boundary condition leads to overestimate the heat flux, setting a constant heat flux of 1 MW/m^2 causes the opposite

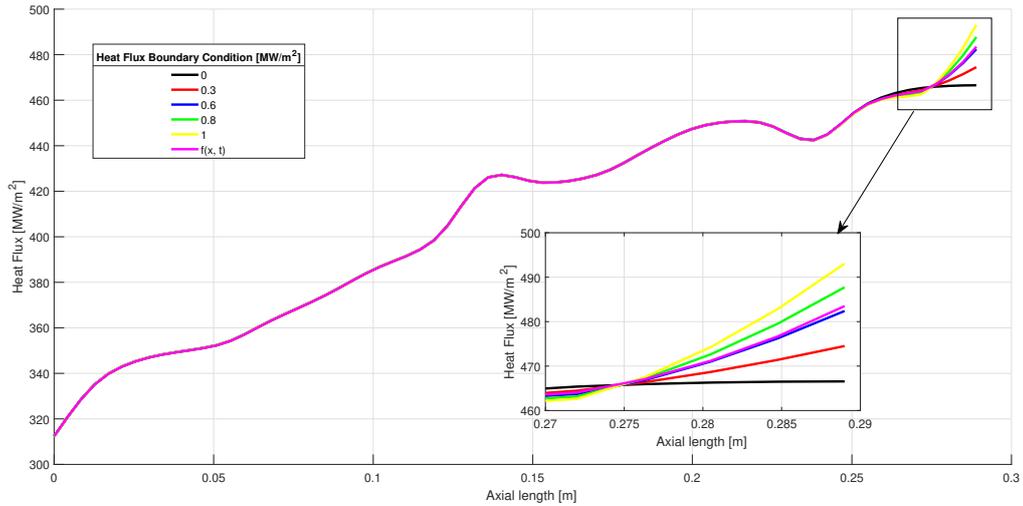


Figure 5.12: Computed Temperature Over z in Middle Point at $t = 8.3215$ s

effect, i.e. leads to underestimate the heat flux. Instead, assuming a logarithmic evolution in space with the peak in the center and a progressive raising in time of the heat flux ensures that neither an overestimation nor an underestimation of the heat flux is made. One can note also that assuming a constant heat flux boundary condition of $0.6 \text{ MW}/\text{m}^2$ can produce the same results of the more real variable boundary condition. Looking at temperatures evolution is instead clear that higher heat flux in value comes from nozzle plane higher temperature is reached on the boundary. The reason for which a high heat flux boundary condition in value causes a lower heat flux on the hot gas wall is due to the fact that since the code has to match the measured and computed temperatures in thermocouples positions, by introducing more heat flux in the domain from the nozzle, a lower heat flux in value has to come from nozzle to allow the matching of the temperatures in thermocouples positions. In the case of constant heat flux of $1 \text{ MW}/\text{m}^2$, the error in heat flux estimation made by this overestimating analysis reaches values up to 12% compared with the case in which the heat flux is set variable. Instead, if the boundary condition is assumed adiabatic the error reach values up to 10%.

Chapter 6

Heat Flux Interpolation in Five-Elements Combustion Chamber

Section 4.1.4 has returned that the problem of the oscillations in space mainly affects the inverse method applied to the five-element combustion chamber. Figure 6.1 testify that these oscillations can reach up to an amplitude of $10 \text{ MW}/\text{m}^2$. However, Chapter 5 has proved that the RoqFITT code works well even when the heat flux which has to be estimated can present an anomalous behavior. Therefore, since the most relevant difference between single- and five-element is the geometry (the presence of five injectors instead of one), possible errors in heat flux estimation can be attributed to the approach used to interpolate the estimated heat flux parameters all over the boundary hot gas wall. The approach used into the code to execute this step has been already introduced in Section 4.1.3.

Taking these previous ideas as the starting point, several attempts have been made to get rid of these oscillations or at least to mitigate them. These approaches are all listed below:

- *Matlab Tools for interpolation*: the first idea was to avoid adding more points in parameters planes for improving the interpolation and to interpolate directly the parameters values all over the hot gas wall by using different Matlab tools for two dimensions interpolation such as *scatteredInterpolant* and *griddata*;
- the second approach employed consists in *a different expedient of adding more points* in parameters planes for improving the quality of the interpolation. This different approach has been devised to mitigate the oscillations over z and at the same time to preserve the original shape of the heat flux along x . Each further point has been obtained by averaging not only the estimated heat flux parameters lying in the same parameters plane, but by involving into the same average also the heat flux parameters located along the combustion chamber;

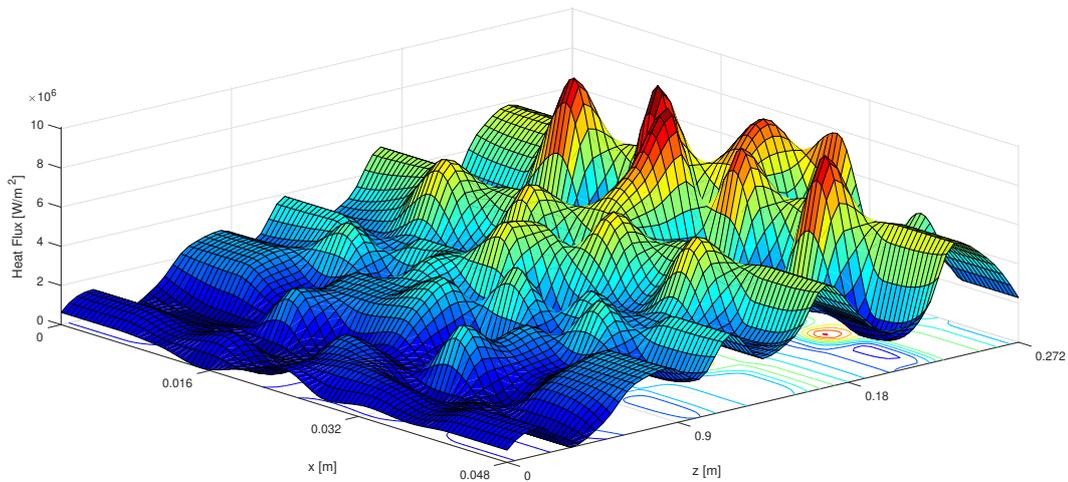


Figure 6.1: Example of Heat Flux Oscillations on the Upper Hot Gas Wall in Five-Element Combustion Chamber

- *Inverse Distance Weighting Interpolation Method* (IDWM) is the last approach to be implemented to improve the solutions of the inverse code.

The user can choose among all approaches listed above by simply switching some variables into the new Matlab function, *Parameters_Interpolation*. The following sections go into detail of each of these approaches.

6.1 Matlab Tools for Interpolation

The first idea was simply to interpolate directly the 66 estimated heat flux parameters all over the hot gas wall by the 2D Matlab interpolation tools *scatteredInterpolant* or *griddata*. Figure 6.3 shows the results obtained by using the first one Matlab command. The direct interpolation of the heat flux parameters all over the upper hot gas wall returned a good resolution of the heat flux but it raised a big issue: since the heat flux parameters points were not located all over the upper hot gas wall, the points in which the heat flux had to be estimated were outside the interpolation domain, hence it was impossible to determine the heat flux in that points. Figure 6.2 helps to understand the reason for this lack of information. Over the upper hot gas wall, according to the parameters points arrangement, it is possible to distinguish between:

- *Interpolation Domain* and;
- *Extrapolation Domain*

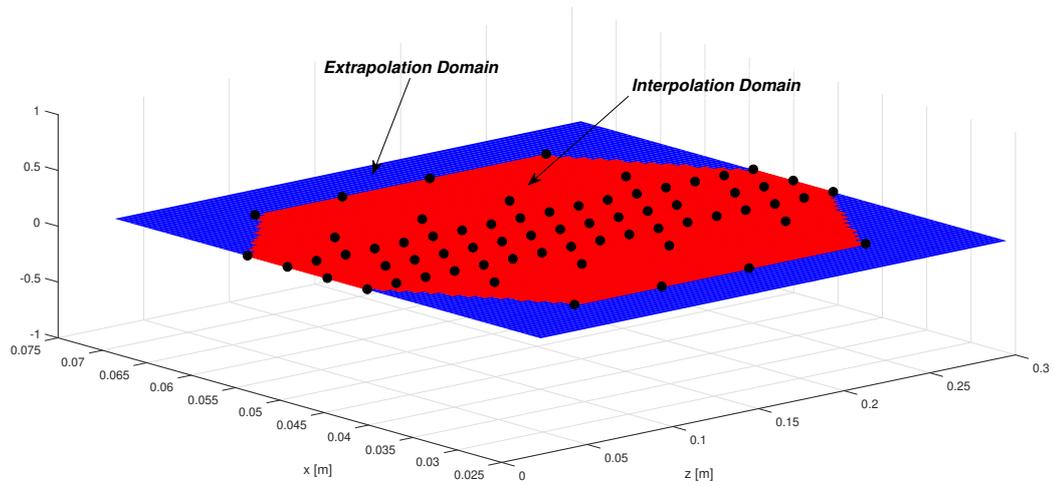


Figure 6.2: Interpolation and Extrapolation Domains on the Upper Hot Gas Wall

The first domain is represented by the entire physical upper hot gas wall enclosed between the furthest parameters points, the second one, instead, lay between the furthest parameters points and the edges of the upper hot gas wall. For this reason, the problem passed from being an interpolation problem to an extrapolation one.

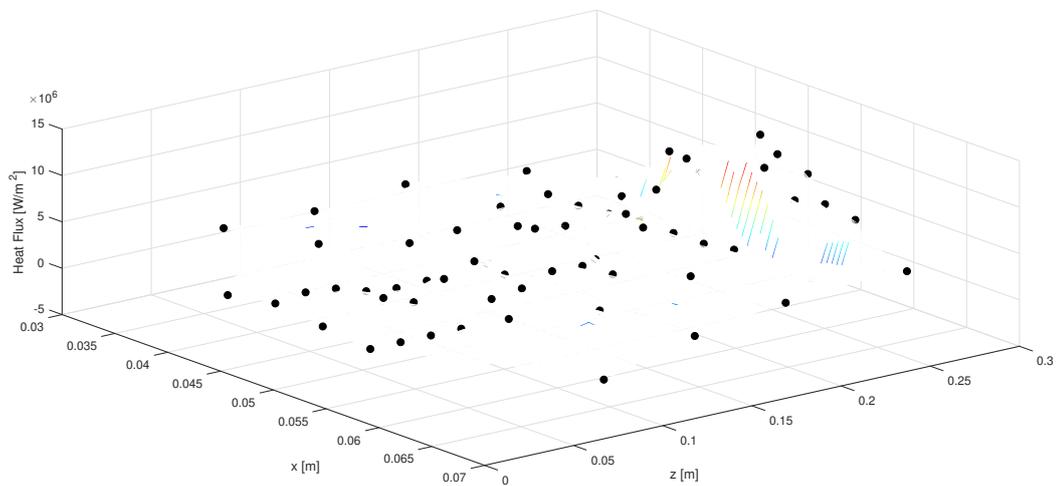


Figure 6.3: Example of Heat Flux Parameters Interpolation on the Upper Hot Gas Wall

At this point the extrapolation of the heat flux was conducted by using the *griddata* Matlab command. An example of how the heat flux would presents on the upper hot gas wall as a result of the extrapolation is shown in figure 6.4. It is immediately clear that also this expedient presented some issues: since the extrapolation domain is too much wider than

the interpolation domain, the heat flux, product of the extrapolation, appeared too high or too low in values on the outer edges. In addition to these peaks clearly not physical, the reason for which also this approach was abandoned is that looking at the trend of the heat flux inside the interpolation domain it also appeared to be poor of information, i.e. the linear interpolation between the points did not reproduce faithfully a real heat flux trend.

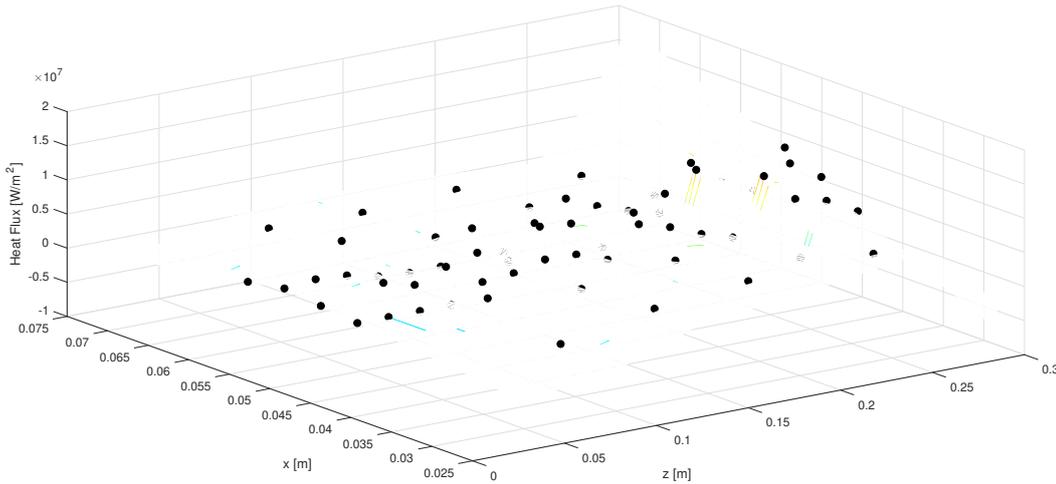


Figure 6.4: Example of Heat Flux Parameters Extrapolation on the Upper Hot Gas Wall

In order to extend the interpolation domain all over the upper hot gas wall and hence to overcome the issues mentioned previously, the next step was to add more parameters points. The code, in fact, provides for adding more parameters points without any limitation. Figure 6.5 plots the new parameters arrangement obtained by implementing a new function into the code *Modifie_Parameters_Points*. With a total number of 185 parameters points instead of 66, it was possible to let the interpolation domain coincide with the upper hot gas wall and to use again the interpolation tool *scatteredInterpolant* to implement a *cubic* interpolation all over the upper hot gas wall. In this way, the shape of the heat flux was well preserved. Figure 6.6 shows how the new heat flux interpolation appeared at a generic time segment, not only the oscillations still persisted but in most of the parameters points the code did not return any value except 0. This result can be explained by looking at the new Jacobi matrix \mathbf{J}_J which had to be computed since the number of parameters points changed with respect to the previous simulations.

Figure ?? compares the Jacobi coefficients $J_{1,j}$ and $J_{67,j}$ with $j = 1, 2, 3, \dots, 66$. As already explained in previous Chapters, a Jacobi or sensitivity coefficient, for example $J_{1,j}$, gives the information about how much the temperature raises in the j -th parameter point as

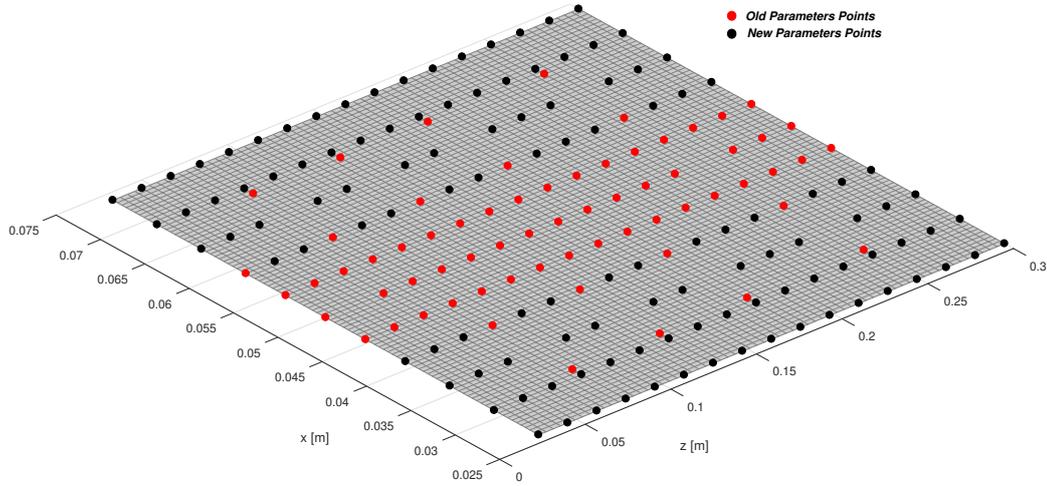


Figure 6.5: New Parameters Points Arrangement on the Upper Hot Gas Wall

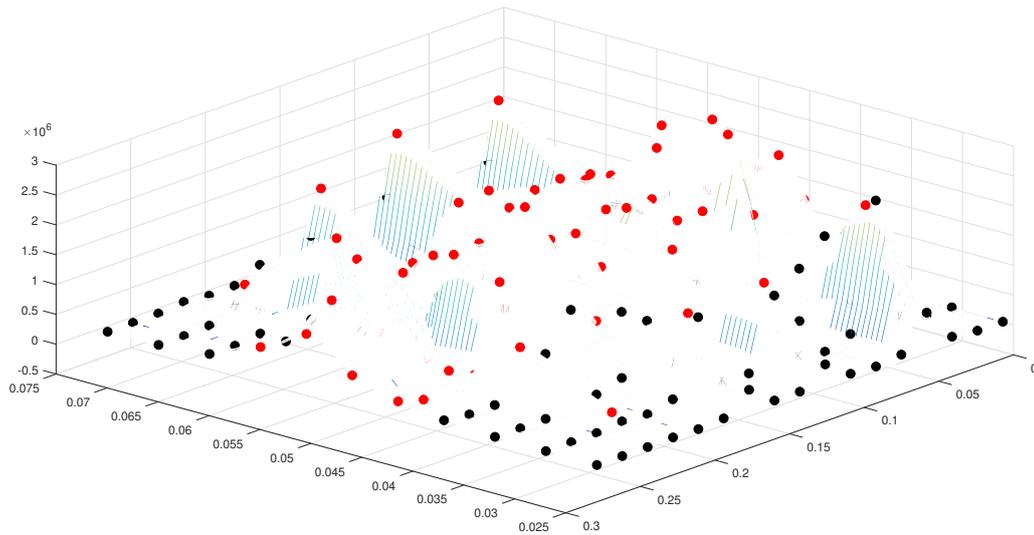


Figure 6.6: Heat Flux Interpolation All Over The Upper Hot Gas Wall Starting From The New Parameters Points Arrangement

a consequence of a variation of the heat flux in 1-*st* parameter point, hence following the same reasoning the Jacobi coefficient $J_{67,j}$ defines the effect on the j -*th* parameter point in terms of variation of temperature caused by the variation of the heat flux in the first parameter point added manually, i.e. the 67-*th* parameter point. Remember that in all simulation carried out till now the number of parameters points was set equal to the number of thermocouples ($N = 66$). Therefore, it is noticeable that since the magnitude of the all new Jacobi coefficients is *zero* the code do not recognize the presence of any new

parameters point, i.e. the variation of the heat flux in that points do not influence the temperature field at all. Therefore, since adding new parameters points did not contribute to the heat flux estimation, this approach had to be abandoned.

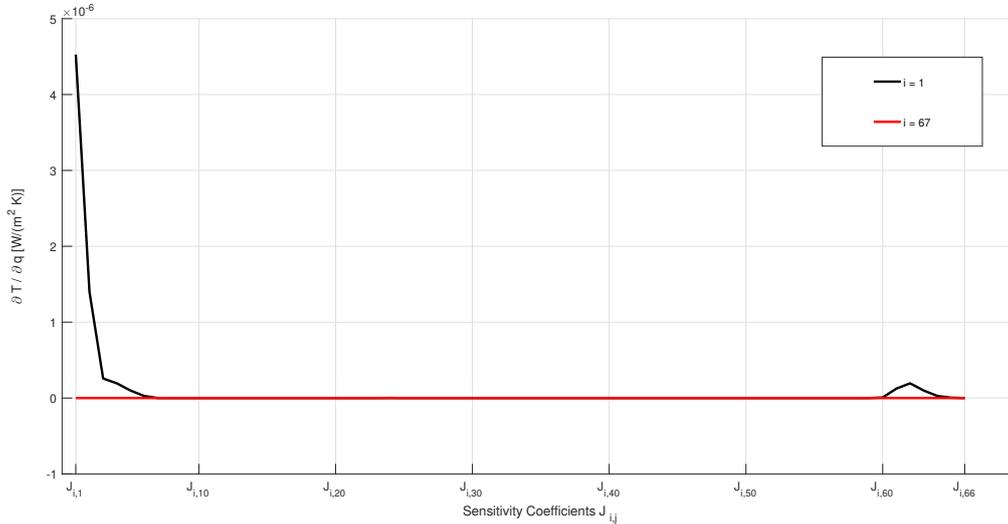


Figure 6.7: Jacobi Coefficients $J_{1,j}$ and $J_{67,j}$ with $j = 1, 2, 3, \dots, 66$ belonging to the new Jacobi Matrix \underline{J}_J

6.2 A Different Way of Computing New Heat Flux Values for Interpolation

In Section 4.1.3 it has been explained in detail the method used by the code to improve the interpolation of the heat flux parameters. The presence in most planes of few parameters points necessitated adding more heat flux points and the result is a more precise interpolation of the heat flux. The same criteria were adopted in this approach but now the new heat flux points were obtained not only on the basis of the information coming from the parameters lying on the same plane but also involving the heat flux parameters located along the combustion chamber. Figure 6.7 explains through the use of arrows which are the parameters points involved in the averages.

The number of heat flux points added in each plane depends on the number of parameters points lying in the same plane (table 6.1). Hence since the total number of heat flux points in each plane was set equal to 11, the number of points added was the 11's complement. The results obtained with this method are presented below. Figures 6.8 and 6.9 show the heat flux evolution along z in middle point of horizontal side and the heat flux evolution

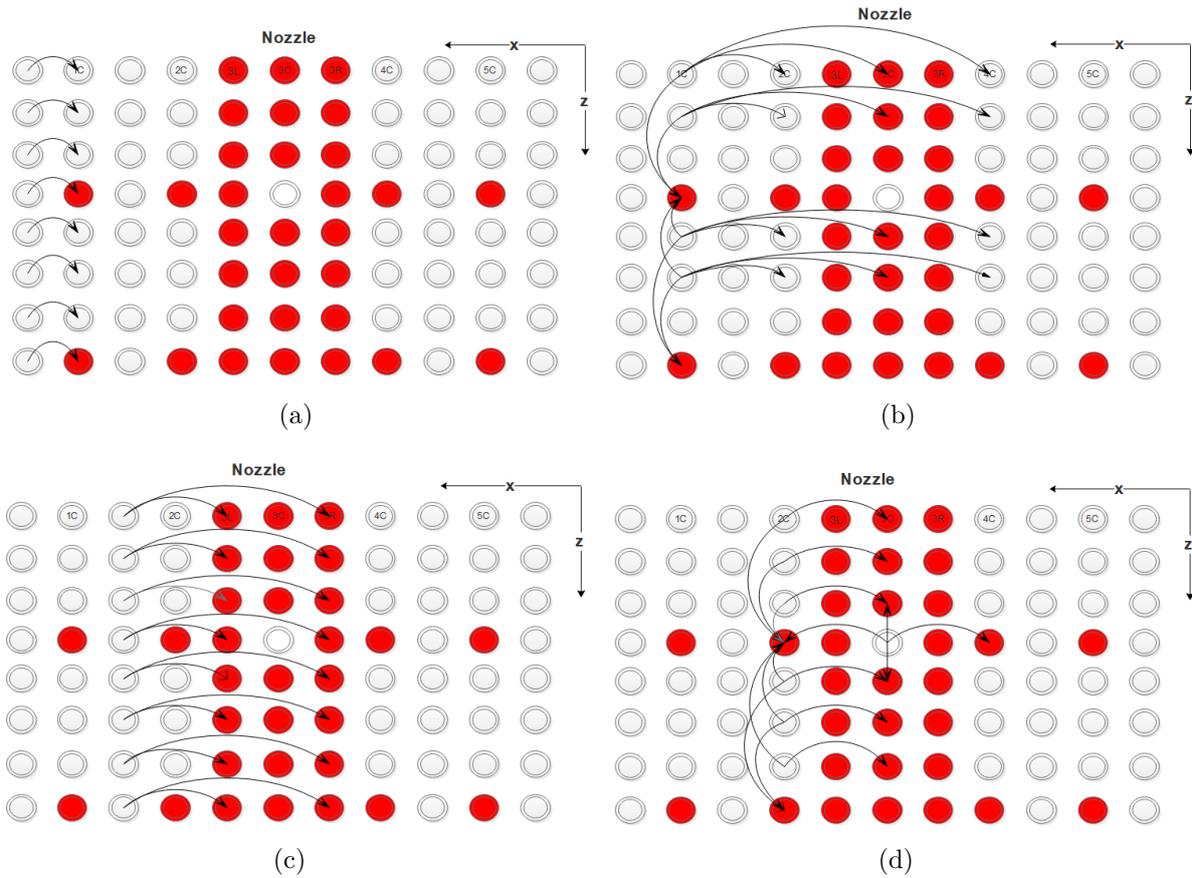


Figure 6.8: New Points Added for Interpolation, the Heads of the Arrows Indicate the Points Involved in the Average

Table 6.1: Number of points added in planes

<i>N^o of parameters points</i>	<i>N^o of points added</i>
3	8
4	7
6	5
7	4

along x for several planes both at evaluation time $t = 23.398$ s. In particular, figure 6.8 testify that by using this new interpolation the oscillations along z appear damped in small size, i.e. the highest oscillation located between $z = 0.15$ m and $z = 0.2$ m from injector plate is reduced from 10 MW/m^2 to 9 MW/m^2 . Looking at figure 6.9, instead, it is clear that by involving in the average also the parameters points along z , the heat flux product of this average appears clearly too damped. By using the original interpolation, the evolution along x testified the presence of five injectors, now the heat flux appears flat in many points and do not allow us to identify the presence of injectors, for example,

at $z = 255 \text{ m}$ from injector plate, the five oscillations have been replaced by just one big oscillation with two flats..

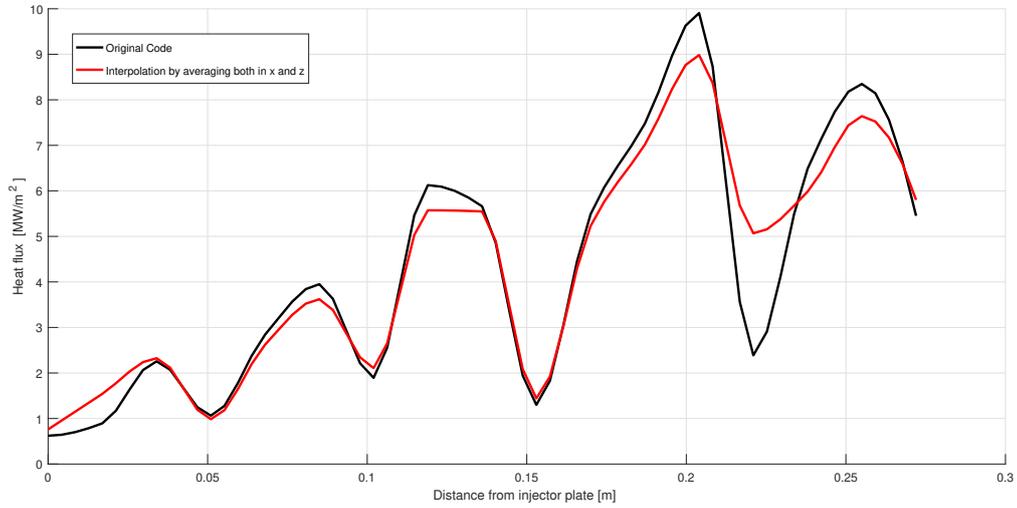


Figure 6.9: Heat Flux Evolution Along z in Middle Point of Horizontal Side at Evaluation Time

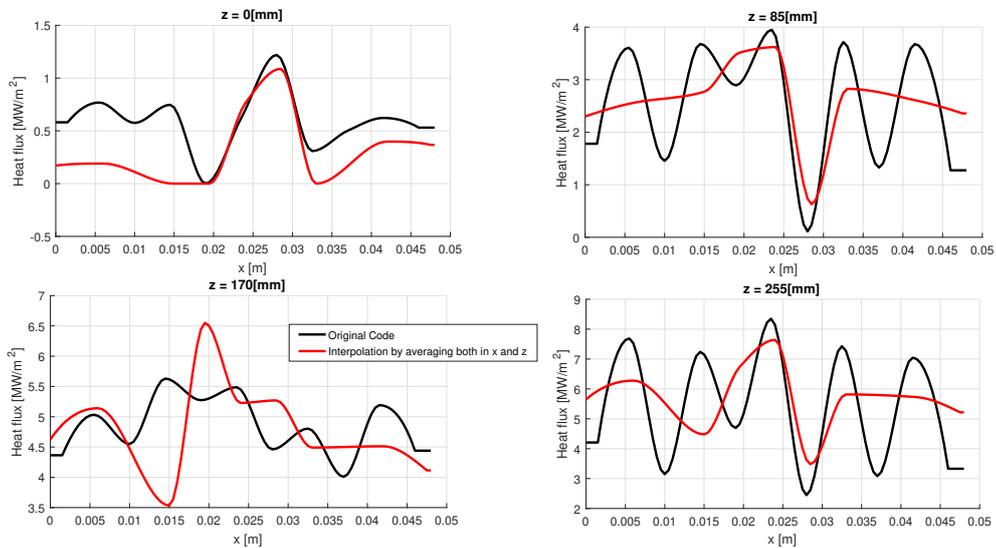


Figure 6.10: Heat Flux Evolution Along x for Different Planes at Evaluation Time

6.3 Inverse Distance Weighted Interpolation

Even if the previous interpolation did not return any significant improvement of the solution, the fact that the heat flux appeared damped in more points suggested finding a more natty method for interpolating the heat flux taking into account also the information provided by the other planes. Figure 6.10 helps to understand how the final approach has been conceived:

- in each of those places where the heat flux parameters stand, further heat flux points were added following the same procedure used in the original code and a *cubic interpolation* along x was performed (blue points in figure 6.10);
- to estimate the heat flux between each parameters plane (grey points in figure 6.10) an *Inverse Distance Weighted Interpolation Method* (IDW) was performed along z in *one dimension*

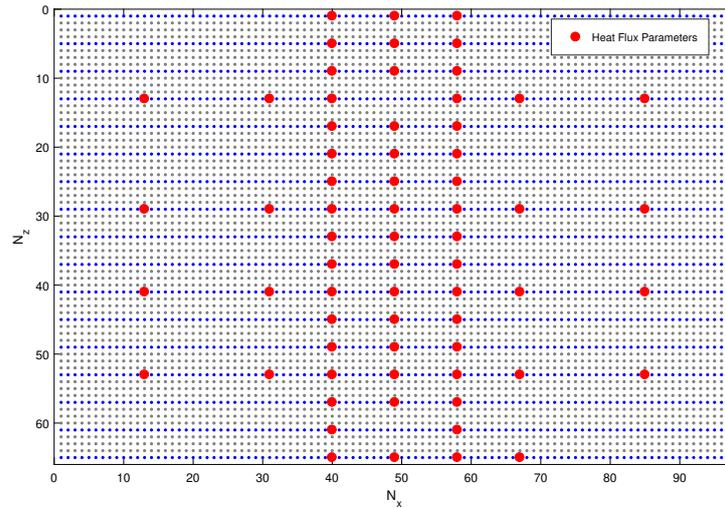


Figure 6.11: Heat Flux Points Product of The Interpolation: Blue Points Indicate the Heat Flux Values Obtained With a Cubic Interpolation Along x , Grey Points Indicate Those Obtained From Inverse Distance Weighted Interpolation

The Inverse Distance Weighted Interpolation is a deterministic interpolation technique mainly used in the geo-statistical analysis for creating surfaces of maps from measured points, it is based on either the extent of similarity or the degree of smoothing [1]. These techniques can be divided into two categories:

- *Global techniques* use the entire data-set to calculate the new values;

- *Local techniques* calculate the new values by using only the measured points lying in a delimited area, which is smaller than the study area

In accordance with [1] an IDW is essentially based on the idea that "things that are close to one another are more alike than those that are farther apart". Each value that has to be predicted is obtained by using the measured values surrounding the prediction location, the measured values that are closest to prediction location have more influence on the predicted value than those farther away. The magnitude of this influence diminishes with the distance.

In accordance with [4] let us assume the following general model for interpolation of Heat flux values (hf):

$$hf_{x,z} = f(x,z) \quad (6.1)$$

It is recalled that the heat flux is estimated only on the upper hot gas wall where the thermocouples lay, that's way hf is a function of x and z . The generic equation for IDW interpolations is:

$$hf_{x,z} = \frac{\sum_{i=1}^n hf_i w_i}{\sum_{i=1}^n w_i} \quad (6.2)$$

$hf_{x,z}$ is the heat flux point to be estimated, hf_i are the known heat flux values belonging to the interpolation domain, w_i are the weights which define the relative importance of individual heat flux values in the interpolation procedure. In this particular case as mentioned previously the weights are inversely proportional to the distance:

$$w_i = d_{x,z,i}^{-\beta} \quad (6.3)$$

$d_{x,z,i}$ is the distance between $hf_{x,z}$ and hf_i , and β is an exponent that can be set by the user and defines the rate at which the weights decrease with distance. If β is equal to 0 there is no decrease with distance, while higher values of β lead to a rapidly decrease of weights for distant points. When β is equal to 2 the method is called Inverse Distance Squared Weighted Interpolation (IDSW). The equation (6.2) becomes:

$$hf_{x,z} = \frac{\sum_{i=1}^n hf_i d_{x,z,i}^{-\beta}}{\sum_{i=1}^n d_{x,z,i}^{-\beta}} \quad (6.4)$$

The following part explain in detail how this method was implemented into the code. It must be kept in mind that this method was applied only for estimating the heat flux points (grey points in figure 6.10) between the parameter planes and was performed only along the z direction by defining a local one dimension *influence domain* and hence a local

maximum radius of influence (r). The exponent β has been set equal to 2. Figures 6.11 and 6.12 help to understand how the influence domains was defined for each node, it shows the influence domain of the two nodes:

- *Node (1,2)* located at $x = 0.074\text{ m}$ and $z = 0.0043\text{ m}$;
- *Node (1,11)* located at $x = 0.074\text{ m}$ and $z = 0.0425\text{ m}$

It is clear that the maximum radius r vary point-to-point and can be set by the user. In this simulation it has been set equal to 0.0468 m for external points and equal to 0.017 m for internal points.

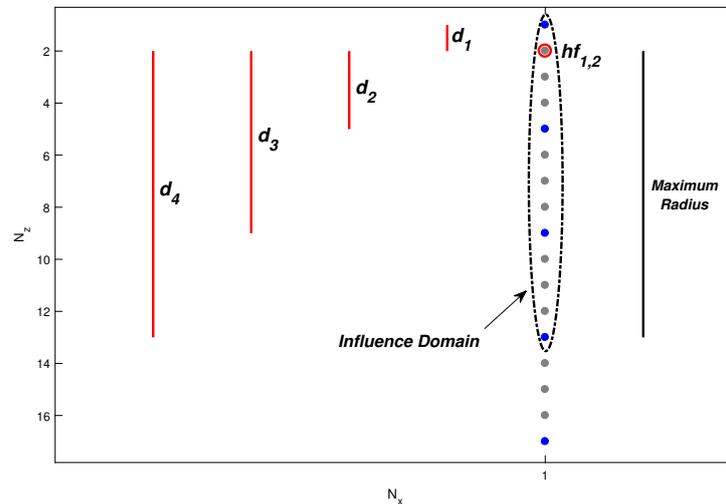


Figure 6.12: Definition of Influence Domain for the *Node (1,2)* Lying on The Upper Hot Gas Wall

Once that the 66 heat flux parameters are available from the updating process the function *Parameters_Interpolation* arranges them in a $N_z \times N_x$ matrix (*Parameters.plane_x_z*) which simulate the whole upper hot gas wall (Fig. 6.10). Hence further points are added in each parameters plane to improve the cubic interpolation along x . Once the interpolation along x is complete in all the 17 parameters planes the IDW interpolation starts. The code scrolls the points from the nozzle to the block, when a node is selected the code calculates the 4 distances d_1, \dots, d_4 and proceeds with the interpolation. These distances indicate how far apart the considered node is from the 4 closest parameters planes. The results are shown in the figures 6.14, 6.13 and 6.15.

Firstly, the Code has shown some difficulty in reaching the convergence in the first time segments, more than once the threshold value of 100 iterations was reached (Fig. 6.15). Instead, Figure 6.13 shows the evolution of the estimated heat flux along z for middle points

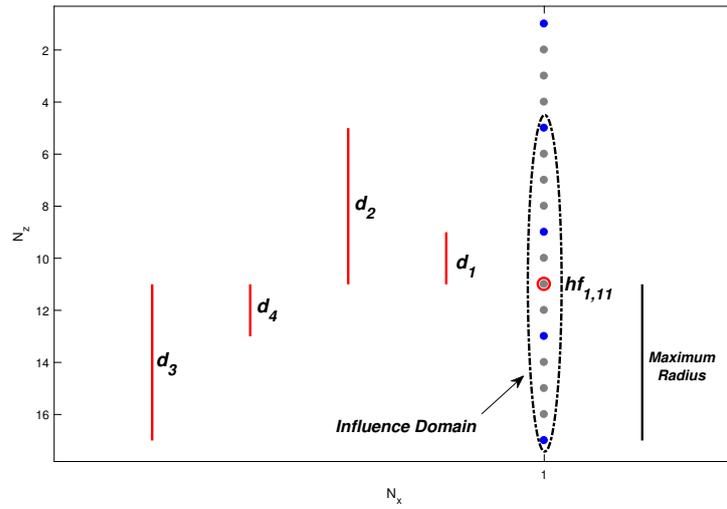


Figure 6.13: Definition of Influence Domain for the *Node (1,11)* Lying on The Upper Hot Gas Wall

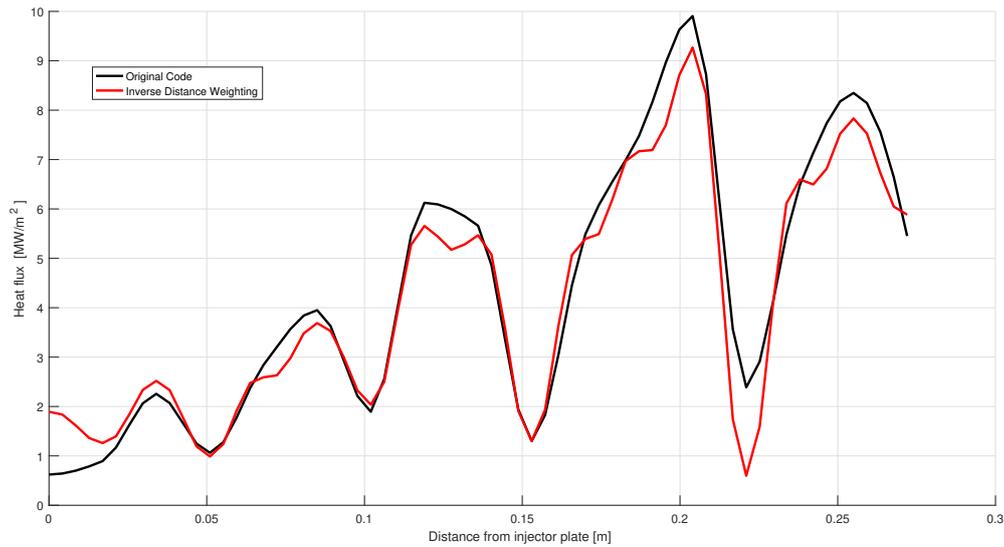


Figure 6.14: Heat Flux Evolution along z in Middle Points at Evaluation Time, Comparison Between Original Code and Modified Code with IDW

of the horizontal side at evaluation time $t = 23.398$ s, unfortunately the method applied did not mitigate the oscillation along z , on the contrary in some points the heat flux appears higher ($z = 0$ m) or lower ($z = 0.21$ m) than the heat flux estimated in the original code. Some peaks were damped but the trend appears chopped up ($z = 0.12, 0.2, 0.25$ m). Instead, figure 6.14 shows the heat flux evolution along x for several parameters plane. In most planes, the heat flux is alike to the original heat flux, but in some cases ($z = 204$ m)

the heat flux seems to not recognizes the presence of the five elements.

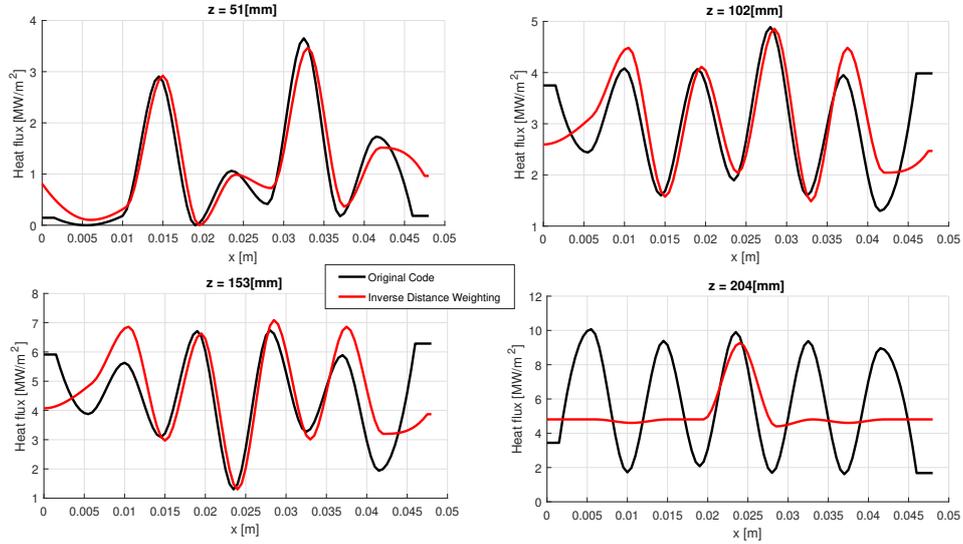


Figure 6.15: Heat Flux Evolution along x for Different Parameters Planes At Evaluation Time, Comparison Between Original Code and Modified Code with IDW

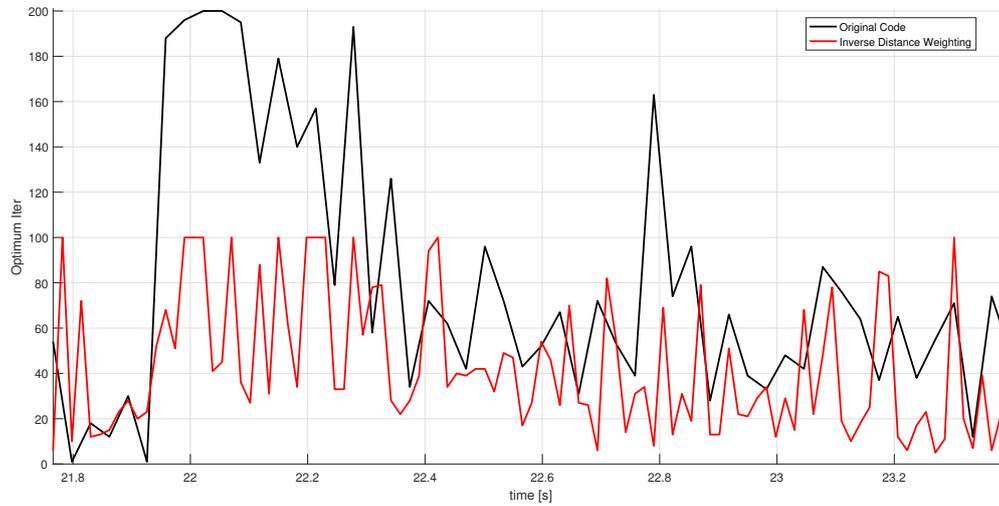


Figure 6.16: Iterations Necessary for Convergence

The application of the three methods described above did not produce any acceptable results. However, one thing is certain, the solution of the inverse code is very sensitive to the approach used to transfer the information to all hot gas walls. The problem of interpolation affects both the convergence and the shape of the heat flux.

Conclusion

The modifications applied to the *RoqFITT* code (new time handling and Newton Rapshon method) have produced great results in terms of computational time-saving. With the new time segment handling the code performs only one direct problem for each iteration instead of two. By using an *Intel Core i5-7200U CPU 2.50GHz* the computational time has been reduced up to 76% in single-element and up to 49% in five-element. The Newton Rapshon method implemented into the code uses the Jacobi Matrix instead of the Sensitivity Matrix. The difference between these two matrices is slight, the first is calculated in one time-step equal to the computational dt , the second goes from *zero* to a generic value chosen by the user depending on the phenomenon characteristic time. The constancy of the Jacobi Matrix over time allows the code to calculate it only one time outside the optimization loop. If the Jacobi Matrix was not constant it would have been computed at each iteration and the computation would not be certainly feasible in terms of time.

However, with these modifications no significant improvements have been detected with regards to mitigation of the oscillations. Both in single- and five-element the heat flux over time as well as over z has been almost the same of those obtained by the original code.

Therefore, the validation of the code has proved to be necessary to verify its ability to detect any progressive changes of the heat flux over time in conjunction with its sudden variations over z . The modified code returned absolutely good results in terms of convergence despite the test was conducted by overestimating the normal random errors (the standard deviation was set equal to $0.15 K$) added into the numerical thermocouples measurements. However, smoothing thermocouples measurements over time by including into the average more than three points allows reducing precisions errors, but even if the code is really sensitive to the smallest variations in temperature measurements, precision errors do not affect much the ability of the code of tracking the oscillation over z (it is well detected and it is not amplified nor dampened). Greats variations in the results were obtained, instead, by changing the thermocouples positions. The possibility that,

during the hot run, vibrations can move the thermocouples from their original location is testified by the last test carried out in the *Validation Code Chapter*. The code appears very sensitive to the variations of thermocouples locations even if these are less than 1 mm.

Assuming an adiabatic boundary condition on nozzle plane implies errors in heat flux estimation up to 10% in respect of assuming a more real heat flux distribution raising both in space and in time. However, the errors are exclusively contained in the area surrounding the nozzle.

The main goal of this thesis was to mitigate the oscillations appearing in the five-element solutions. The heat flux shape as well as the convergence of the code are very sensitive to the method used for interpolating the heat flux from parameter points to the hot gas wall points. One of the most important findings in this study has been that even if the *RoqFITT* code provided for adding more parameter points into the calculation, the sensitivity problem returned that a raising in temperature in these points did not cause any variation in heat flux. This aspect of the problem leads to the impossibility to increase the density of parameter points and consequently to make the interpolation easier and more accurate. The problem in this case is overdefined, the number of parameter points are more than the number of thermocouples. Adding more parameter points implies adding more thermocouples. The *Inverse Distance Weighted Interpolation method* tried to combine the information coming both from parameter points along x and from parameter points along z , i.e. belonging to other planes but with the same (x,y) coordinates. Although this method is more esteemed than the others it has not returned any good results.

The present study has revealed that an inverse method is a trustworthy tool for heat flux estimation as long as the thermocouples measurements are precise and accurate. The quality of the solution depends on the number of thermocouples used into the experiment. It is advisable the use of more thermocouples in such a way to cover the entire boundary domain and make the interpolation easier without any doubt on the heat flux shape.

Bibliography

- [1] *How inverse distance weighted interpolation works.* <http://pro.arcgis.com/en/pro-app/help/analysis/geostatistical-analyst/how-inverse-distance-weighted-interpolation-works.htm>, . – Accessed: 2018-05-28
- [2] A. BRAEUNING, Robert: Rocket propellants. In: *Retrieved from <http://www.braeunig.us/space/propel.htm>*, 2008
- [3] ALIFANOV, Oleg M.: *Inverse heat transfer problems.* Springer Science & Business Media, 2012
- [4] BARTIER, Patrick M. ; KELLER, C P.: Multivariate interpolation to incorporate thematic surface data using inverse distance weighting (IDW). In: *Computers & Geosciences* 22 (1996), Nr. 7, S. 795–799
- [5] BERGMAN, Theodore L. ; INCROPERA, Frank P. ; DEWITT, David P. ; LAVINE, Adrienne S.: *Fundamentals of heat and mass transfer.* John Wiley & Sons, 2011
- [6] BROWN, Stan: Measures of Shape: Skewness and Kurtosis. In: *Retrieved from <http://https://brownmath.com/stat/shape.htm>*, 2016
- [7] BURKHARDT, Holger ; SIPPEL, Martin ; HERBERTZ, Armin ; KLEVANSKI, Josef: Kerosene vs. methane: a propellant tradeoff for reusable liquid booster stages. In: *Journal of Spacecraft and Rockets* 41 (2004), Nr. 5, S. 762–769
- [8] CELANO, MP ; SILVESTRI, S ; PAUW, J ; PERAKIS, N ; SCHILY, F ; SUSLOV, D ; HAIDN, OJ: Heat flux evaluation methods for a single element heat-sink chamber. In: *6th European Conference of Aeronautics and Space Science, Krakow, Poland, 2015*
- [9] CELANO, MP ; SILVESTRI, S ; SCHLIEBEN, G ; KIRCHBERGER, C ; HAIDN, OJ ; KNAB, O: Injector characterization for a gaseous oxygen-methane single element combustion chamber. In: *Progress in Propulsion Physics* 8 (2016), S. 145–164

- [10] CIRACI, Cosimo: Inverse heat conduction method and heat flux error estimation in rocket engines. In: *Master degree thesis*, 2018
- [11] DANIEL, James W.: The approximate minimization of functionals. (1971)
- [12] DIFFICILE, Pasquale: Performance and emission imaging of a coaxial single element GO₂/GCH₄ rocket combustion chamber. In: *Master degree thesis*, 2018
- [13] FLETCHER, Reeves ; REEVES, Colin M.: Function minimization by conjugate gradients. In: *The computer journal* 7 (1964), Nr. 2, S. 149–154
- [14] HONG, Ye ; LIU, Zhanyi ; SILVESTRI, Simona ; CELANO, Maria P. ; HAIDN, O ; YU, Zhendong: Numerical simulation and measurements of wall heat fluxes in a single-element GO₂/GCH₄ rocket combustor. In: *European Conference for Aeronautics and Space Sciences*, 2017
- [15] MONEGATO, Giovanni: *Metodi e algoritmi per il CALCOLO NUMERICO*. Clut, 2008
- [16] OZISIK, M N.: *Inverse heat transfer: fundamentals and applications*. CRC Press, 2000
- [17] PAUW, Julian D.: Inverse Heat Conduction Problems applied to Rocket Combustion Chambers. In: *Master degree thesis*, 2015
- [18] PERAKIS, Nikolaos: Inverse method applied to rocket engine. In: *Bachelor degree thesis*, 2014
- [19] PERAKIS, Nikolaos ; CELANO, Maria P. ; HAIDN, Oskar J.: Heat flux and temperature evaluation in a rectangular multi-element GOX/GCH₄ combustion chamber using an inverse heat conduction method. In: *7th European Conference for Aerospace Sciences*, 2017
- [20] POLAK, Elijah ; RIBIERE, Gerard: Note sur la convergence de méthodes de directions conjuguées. In: *Revue française d'informatique et de recherche opérationnelle. Série rouge* 3 (1969), Nr. 16, S. 35–43
- [21] SILVESTRI, S ; CELANO, MP ; SCHLIEBEN, G ; KIRCHBERGER, C ; HAIDN, OJ: Characterization of a GOX-GCH₄ single element combustion chamber. In: *4th Space Propulsion Conference, Köln (Germany)*, 2014
- [22] SUTTON, George P. ; BIBLARZ, Oscar: *Rocket propulsion elements*. John Wiley & Sons, 2016