

**ALMA MATER STUDIORUM - UNIVERSITÀ DI
BOLOGNA**

SCUOLA DI INGEGNERIA E ARCHITETTURA

Dipartimento di Ingegneria Industriale

Corso di Laurea Magistrale in Ingegneria Energetica

TESI DI LAUREA

in

Modeling and Simulation Techniques for Energy Engineering

**Coupling of fem and fvm codes for optimal control of
the heat equation**

Candidato:

SAMUELE BALDINI

Relatore:

Prof. Ing. SANDRO MANSERVISI

Correlatori:

Prof. Ing. ANTONIO CERVONE

Dott. Ing. GIACOMO BARBI

Dott. Ing. LUCIA SIROTTI

Dott. Ing. FEDERICO GIANGOLINI

Anno Accademico 2023/2024

Contents

1	Introduction	5
2	Finite Elements	9
2.1	Elements of Functional Analysis	9
2.2	Numerical Approximation	17
2.2.1	One-Dimensional Case	17
2.2.2	Two-Dimensional Case	25
2.2.3	Gaussian Integration	30
2.3	Numerical Solution Techniques for PDEs	33
2.3.1	Finite Element Method	33
2.3.2	Finite Volume Method	36
3	Optimal Control	41
3.1	Theory of Optimal Control	41
3.1.1	Adjoint Method	42
3.1.2	Numerical solution	45
3.2	Distributed Control	51
3.3	Dirichlet Boundary Control	54
3.4	Neumann Boundary Control	58

4 Coupling	61
4.1 Coupling structure	62
4.2 Coupling algorithm	64
4.2.1 Interpolation algorithm	67
4.2.2 Interpolation errors	70
5 Numerical results	71
5.1 Uncoupled results	75
5.1.1 Case 1	75
5.1.2 Case 2	78
5.1.3 Minimization methods comparison	80
5.2 Coupled results	82
5.2.1 Distributed control	83
5.2.2 Dirichlet boundary control	90
5.2.3 Neumann boundary control	94
6 Conclusions	99
List of figures	101
List of tables	105
Bibliography	107

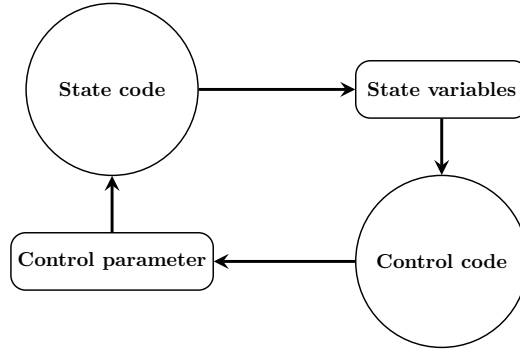
Chapter 1

Introduction

Mathematical modeling of physical phenomena is a well-established approach, often advantageous in terms of time and resource efficiency, compared to experimental methods for studying complex systems. These problems, modeled through partial differential equations (PDEs), depend on various *input data*, such as material properties, boundary conditions, and source terms. While, usually in simulations, these inputs are treated as fixed data, they can also be considered as system parameters in search of an optimal design. An *optimal control problem* involves manipulating these parameters, now termed *control variables*, to minimize a *cost function*, the minimum of which represents the desired condition of the system.

Optimal control theory has become a topic of significant interest in engineering applications, especially for solving inverse problems. The traditional *trial and error* approach can result in high computational and time costs, whereas solving the inverse problem allows for faster attainment of the desired outcome. Additionally, the mathematical framework of optimal control enables the identification of innovative and non-intuitive solutions to complex problems.

In contrast to direct simulations, solving an optimal control problem requires addressing the physical system and the entire control system, composed of the state (physical system), adjoint, and control equations. This process must be iterated multiple times. Validated and commercial codes increase the reliability of the optimization process. A possible approach is to employ one code for solving the physical system and another for the control system to address the solution with a validated state solver. Since each code depends on each variable of the optimal system, these variables must be coupled.



This work presents a temperature optimization achieved by coupling the in-house FEMuS code with the widely used OpenFOAM code. This coupling is facilitated through the open-source MED and MEDCOUPLING libraries. Previous studies have demonstrated the feasibility of code coupling using MED libraries in conjunction with the FEMuS and OpenFOAM codes [4, 14].

This approach is demonstrated through examples targeting different objectives, each employing distributed and boundary control techniques. The control parameter may be a volumetric heat source or, depending on the control location, either the wall temperature (Dirichlet type) or a heat flux (Neumann type).

The thesis is structured as follows: it begins with an introduction to el-

ements of functional analysis [35], followed by the numerical approximation of function spaces and the numerical solution of PDEs within these spaces [36]. Subsequently, the theory of optimal control is explored [30], along with various minimization methods for its resolution [18]. The optimality system is then derived for the three types of control: distributed, Dirichlet, and Neumann. The focus then shifts to the coupling between the FEMuS and OpenFOAM codes. The structure and internal coupling methods are illustrated [4, 14]. Finally, the numerical results are presented in two phases: first, the uncoupled results are shown to validate the control system implemented within FEMuS during this work, followed by the coupled results, which validate and analyze the effectiveness of the coupling itself. This last chapter also presents a nondimensional formulation for the control system. Lastly, some conclusions are drawn.

Chapter 2

Finite Elements

2.1 Elements of Functional Analysis

In this section we briefly introduce some fundamental concepts of functional analysis necessary for the numerical modeling of partial differential equations (PDEs): the notion of vector spaces, normed vector spaces, and Banach spaces. Then, we introduce the inner product for a vector space and Hilbert spaces, followed by integrable function spaces and Sobolev spaces. Finally, we also cover the strong and weak formulations of PDEs and the variational derivatives of infinite-dimensional functionals.

Normed Vector Spaces. A space X is defined as a vector space over a scalar field K if it is equipped with two operations that satisfy the following properties:

1. A sum such that for $u, v \in X$, $u + v \in X$,
2. A scalar multiplication such that for $u \in X$ and a scalar $a \in K$, $au \in X$.

Let X be a real vector space. A function $\|\cdot\| : X \rightarrow \mathbb{R}_0^+$, $x \mapsto \|x\|$, is called

a norm if and only if the following conditions are satisfied:

1. $\|x\| = 0 \iff x = 0$, for all $x \in X$,
2. $\|x + y\| \leq \|x\| + \|y\|$, for all $(x, y) \in X^2$,
3. $\|\lambda x\| = |\lambda|\|x\|$, for all $x \in X$, and for all $\lambda \in \mathbb{R}$.

If $\|\cdot\|$ is a norm on X , then $(X, \|\cdot\|)$ is called a *normed vector space*.

Let $(x_n)_{n \in \mathbb{N}}$ be a sequence in this normed vector space. The sequence is said to be *convergent* in X if and only if there exists a *limit of the sequence* $x \in X$ such that

$$\lim_{n \rightarrow \infty} \|x_n - x\| = 0. \quad (2.1)$$

This definition of convergence is called strong convergence. Moreover, the sequence is said to be a *Cauchy sequence* if and only if for every $\epsilon > 0$, there exists an $n_0 \in \mathbb{N}$ such that, for every $(m, n) \in \mathbb{N}^2$ with $n \geq n_0$ and $m \geq n_0$,

$$\|x_n - x_m\| \leq \epsilon. \quad (2.2)$$

Finally, a normed space $(X, \|\cdot\|)$ is said to be *complete* or a *Banach space* if and only if every Cauchy sequence in X is convergent in X .

Hilbert Spaces. An inner product on a vector space X is a symmetric bilinear form that associates with two vectors $x, y \in X$ a scalar in the real field \mathbb{R}

$$\langle x, y \rangle : X^2 \rightarrow \mathbb{R}, \quad (2.3)$$

such that the following properties are satisfied:

1. $\langle x, x \rangle > 0$,
2. $\langle x, y \rangle = \langle y, x \rangle$,

3. $\langle ax + y, z \rangle = a\langle x, z \rangle + \langle y, z \rangle$, for all $a \in \mathbb{R}$.

If $\langle \cdot, \cdot \rangle$ is an inner product on a real vector space X , then $(X, \langle \cdot, \cdot \rangle)$ is called a *pre-Hilbert space*. A pre-Hilbert space is called a *Hilbert space* if and only if $(X, \langle \cdot, \cdot \rangle)$ is a Banach space, where the norm $\|\cdot\|$ is defined from the inner product by $\|x\| := \sqrt{\langle x, x \rangle}$.

Spaces of Integrable Functions. Let E be a measurable subset of \mathbb{R}^m , $m \in \mathbb{N}$. For every $p \in [1, \infty)$, $L^p(E)$ is defined as the space of all measurable functions $f : E \rightarrow \mathbb{R}$ such that

$$\int_E |f(x)|^p dx < \infty. \quad (2.4)$$

For every $f \in L^p(E)$, its norm is defined as

$$\|f\|_p := \left(\int_E |f(x)|^p dx \right)^{\frac{1}{p}}, \quad (2.5)$$

and the space $(L^p(E), \|\cdot\|_p)$ is a Banach space.

Given $E \subseteq \mathbb{R}^m$ measurable and $f : E \rightarrow \mathbb{R}$ measurable, f is defined as *integrable* if and only if $f \in L^1(E)$.

It is observed that for every measurable $E \subseteq \mathbb{R}^m$, the map $(f, g) \mapsto \int_E fg$ defines an inner product on $L^2(E)$, thus $L^2(E)$ is a Hilbert space.

Fundamental Lemma of the Calculus of Variations. The *support* of a function $f : \Omega \rightarrow \mathbb{R}$ defined on a space $\Omega \subseteq \mathbb{R}^m$, called $\text{supp}(f)$, is the closure of the set of points where f is not null

$$\text{supp}(f) := \overline{\{x \in \Omega : f(x) \neq 0\}} \subseteq \overline{\Omega}. \quad (2.6)$$

Let $\Omega \subseteq \mathbb{R}^m$ be open and $k \in \mathbb{N}_0$.

1. Let $C^k(\Omega)$ be the space of all functions $f : \Omega \rightarrow \mathbb{R}$ such that f has continuous partial derivatives up to order k . Functions in $C^k(\Omega)$ are said to be of class $C^k(\Omega)$.
2. Let $C^k(\overline{\Omega})$ be the space of all functions in $C^k(\Omega)$ such that their partial derivatives extend continuously to $\overline{\Omega}$.
3. Let $C_0^k(\Omega)$ be the space of all functions in $C^k(\overline{\Omega})$ that have compact support contained in Ω , in other words, it is the space of functions that vanish on the boundary $\partial\Omega$.

If $\Omega \subseteq \mathbb{R}^m$ is open and $f \in L^1(\Omega)$, then

$$\int_{\Omega} f\phi = 0, \quad \forall \phi \in C_0^\infty(\Omega), \quad (2.7)$$

implies that $f = 0$ everywhere.

As a consequence, if $f, g \in C^1(\Omega)$ and $f \circ g \in C_0^1(\Omega)$, then the following *integration by parts* formula holds

$$\int_{\Omega} g \frac{\partial f}{\partial x_i} = - \int_{\Omega} f \frac{\partial g}{\partial x_i}, \quad \forall i \in \{1, \dots, m\}. \quad (2.8)$$

Sobolev Spaces. Given $f \in L^1(\Omega)$ and $k \in \mathbb{N}_0$, the function $g \in L^1(\Omega)$ is called the *weak derivative* of order k if and only if

$$\int_{\Omega} f \frac{\partial^k \phi}{\partial x^k} = (-1)^k \int_{\Omega} g\phi, \quad \forall \phi \in C_0^\infty(\Omega). \quad (2.9)$$

If this is true, $\frac{\partial^k \phi}{\partial x^k}$ can be written in place of g .

Let $\Omega \subseteq \mathbb{R}^m$ be open, and $(k, p, \alpha) \in \mathbb{N}_0 \times [1, \infty]$. Let $W^{k,p}(\Omega)$ be the subspace of $L^1(\Omega)$ consisting of all functions f such that its weak derivatives

$\frac{\partial^\alpha f}{\partial x^\alpha}$ exist and lie in $L^p(\Omega)$ for every $\alpha < k$. The spaces $W^{k,p}(\Omega)$ are called *Sobolev spaces*. The following norm is defined for every $p < \infty$

$$\|F\|_{W^{k,p}(\Omega)} := \left(\sum_{\alpha=0}^k \int_{\Omega} \left| \frac{\partial^\alpha f}{\partial x^\alpha} \right|^p \right)^{\frac{1}{p}} = \left(\sum_{\alpha=0}^k \left\| \frac{\partial^\alpha f}{\partial x^\alpha} \right\|_p^p \right)^{\frac{1}{p}}, \quad (2.10)$$

in the case $p = 2$, it is called $H^k(\Omega) := W^{k,2}(\Omega)$.

For every $(k, p, \alpha) \in \mathbb{N}_0 \times [1, \infty]$, the Sobolev space $W^{k,p}(\Omega)$ is a Banach space. In particular, each $H^k(\Omega)$ is a Hilbert space concerning the following inner product

$$\langle f, g \rangle_{H^k(\Omega)} = \sum_{\alpha=0}^k \int_{\Omega} \frac{\partial^\alpha f}{\partial x^\alpha} \frac{\partial^\alpha g}{\partial x^\alpha} = \sum_{\alpha=0}^k \left\langle \frac{\partial^\alpha f}{\partial x^\alpha}, \frac{\partial^\alpha g}{\partial x^\alpha} \right\rangle_{L^2(\Omega)}. \quad (2.11)$$

Strong and Weak Formulation. The weak formulation of partial differential equations is introduced by taking the Poisson equation with homogeneous Dirichlet boundary conditions as an example. Let $\Omega \subseteq \mathbb{R}^m$ be open, $\phi \in C^2(\Omega) \cap C^0(\bar{\Omega})$, and $q \in C^0(\Omega)$. The Poisson equation

$$-\nabla \cdot (\nabla \phi(x)) = q(x), \quad x \in \Omega, \quad (2.12)$$

can be rewritten as

$$-\sum_{i=1}^m \frac{\partial}{\partial x_i} \left(\frac{\partial \phi}{\partial x_i} \right) = q(x), \quad x \in \Omega. \quad (2.13)$$

We define the following system of equations for the unknown $\phi \in C^2(\Omega)$

$$\begin{cases} -\sum_{i=1}^m \frac{\partial}{\partial x_i} \left(\frac{\partial \phi}{\partial x_i} \right) = q(x), & x \in \Omega, \\ \phi(x) = 0, & x \in \partial\Omega, \end{cases} \quad (2.14)$$

as the Poisson problem in *strong form* with *homogeneous Dirichlet* boundary conditions. A function $\phi \in C^2(\Omega)$ that satisfies the system is called a *strong* solution of the problem. In general, a strong solution to the problem has conditions that are too restrictive to be of practical interest, so the search is extended to *generalized solutions* through a reformulation of the problem.

Let $\phi \in C^2(\Omega)$ be a solution of (2.14). Multiply (2.13) by $\psi \in C_0^\infty(\Omega)$ and integrate over Ω , obtaining

$$-\int_{\Omega} \psi \left(\sum_{i=1}^m \frac{\partial}{\partial x_i} \left(\frac{\partial \phi}{\partial x_i} \right) \right) dx = \int_{\Omega} \psi (q(x)) dx. \quad (2.15)$$

Bringing everything to the left

$$-\int_{\Omega} \psi \left(\sum_{i=1}^m \frac{\partial}{\partial x_i} \left(\frac{\partial \phi}{\partial x_i} \right) - q(x) \right) dx = 0, \quad (2.16)$$

the *Fundamental Lemma of Variational Calculus* can be applied to retrieve equation (2.13). Integrating by parts, the equation (2.15) can be rewritten as

$$\int_{\Omega} \left(\sum_{i=1}^m \frac{\partial \psi}{\partial x_i} \frac{\partial \phi}{\partial x_i} \right) dx = \int_{\Omega} \psi (q(x)) dx. \quad (2.17)$$

Let $\Omega \subseteq \mathbb{R}^m$ be open and $q \in L^2(\Omega)$. We define the following system of equations for the unknown $\phi \in H_0^1(\Omega)$

$$\int_{\Omega} \left(\sum_{i=1}^m \frac{\partial \psi}{\partial x_i} \frac{\partial \phi}{\partial x_i} \right) dx = \int_{\Omega} \psi (q(x)) dx, \quad \forall \psi \in H_0^1(\Omega), \quad (2.18)$$

as the Poisson problem in *weak form* with *homogeneous Dirichlet* boundary conditions. A function $\phi \in H_0^1(\Omega)$ that satisfies the system is called a *weak* solution of the problem. It is important to note that the condition imposed on the solution is no longer to belong to C^2 , i.e., to be continuous with

continuous first and second derivatives, but to H^1 , and thus integrable with the first derivative integrable.

Variational Derivative. An operator $A : X \rightarrow Y$ is defined as a function between two normed vector spaces $(X, \|\cdot\|_X)$ and $(Y, \|\cdot\|_Y)$. A is called bounded if and only if there exists a constant $C \geq 0$ such that

$$\|A(x)\|_Y \leq C\|x\|_X, \forall x \in X.$$

Operators with real values are called *functionals*. In particular, linear operators with real values are called linear functionals. The vector space of all bounded linear operators between two normed vector spaces $(X, \|\cdot\|_X)$ and $(Y, \|\cdot\|_Y)$ is called $\mathcal{L}(X, Y)$. In particular, the space $\mathcal{L}(X, \mathbb{R})$ of all bounded linear functionals on X is called the *dual* space of X . The dual space of X is denoted by X^* .

Let $U = (X, \|\cdot\|_X)$ and $V = (Y, \|\cdot\|_Y)$ be normed vector spaces, $f : U \rightarrow V$, and $u \in U$.

1. Let $h \in U$. If $u+th \in U$ for every sufficiently small positive t , and if the limit

$$\delta f(u, h) := \lim_{t \rightarrow 0} \frac{f(u + th) - f(u)}{t}, \quad (2.19)$$

exists in V , then $\delta f(u, h)$ is called the *directional derivative* of f with respect to u in the direction h .

2. Let $u \in U$. If the directional derivative $\delta f(u, h)$ exists for every $h \in U$, then the operator

$$\delta f(u, \cdot) : U \rightarrow V, h \mapsto \delta f(u, h), \quad (2.20)$$

is called the *first variation* of f with respect to u . If it is also a bounded linear operator, it is called the *Gâteaux derivative* and is denoted as $f'_G(u)$. In this case, f is said to be *differentiable according to Gâteaux*.

3. Finally, f is said to be *differentiable according to Fréchet* if and only if there exists a linear operator $A : U \rightarrow V$ such that for every $h \neq 0 \in U$ with $\|h\|_U \rightarrow 0$

$$\frac{\|f(u+h) - f(u) - Ah\|_V}{\|h\|_U} \rightarrow 0.$$

In this case, we write $f'_F(u)$ instead of A and it is called the *Fréchet Derivative* of f with respect to u .

If $f : U_{ad} \rightarrow V$ has a directional derivative with respect to u in the direction h (respectively, first variation, Gâteaux, and Fréchet derivative), then the following properties hold

1. Let f be as above and $\alpha \in \mathbb{R}$

$$\begin{aligned} \delta(\alpha f)(u, h) &= \alpha \delta(f)(u, h), \\ (\alpha f)'_G(u) &= \alpha f'_G(u), \\ (\alpha f)'_F(u) &= \alpha f'_F(u). \end{aligned} \tag{2.21}$$

2. Let f be as above and $g : U_{ad} \rightarrow V$ which has a variational derivative with respect to u in the direction h (respectively, first variation, Gâteaux, and Fréchet derivative)

$$\begin{aligned} \delta(f+g)(u, h) &= \delta(f)(u, h) + \delta(g)(u, h), \\ (f+g)'_G(u) &= f'_G(u) + g'_G(u), \\ (f+g)'_F(u) &= f'_F(u) + g'_F(u). \end{aligned} \tag{2.22}$$

2.2 Numerical Approximation

When seeking the solution of a partial differential equation numerically, it is necessary to approximate the infinite-dimensional spaces defined previously to finite-dimensional spaces. The choice of how to approximate the functional space determines the numerical solution method used, such as the *finite element method* or the *finite volume method*. The spaces of primary interest, which will be addressed in the approximation, are the Sobolev spaces H^1 and L^2 .

2.2.1 One-Dimensional Case

A preliminary overview is provided for the one-dimensional case, before moving on to the two-dimensional case. The three-dimensional case is analogous.

Approximation in S_h^0 . The space L^2 of integrable functions, since it does not impose conditions on their differentiability, can be represented by a basis of infinite piecewise constant functions. The simplest approximation of this space is through a finite basis of piecewise constant functions, see Figure 2.1.

Given a domain $\Omega \subseteq \mathbb{R}^1$, it can be divided into N intervals defined by $N + 1$ points

$$x_0 < x_1 < \dots < x_i < \dots < x_{N-1} < x_N,$$

a function ϕ_i^0 is defined such that

$$\begin{cases} \phi_i^0 = 1 & \text{for } x_i < x < x_{i+1}, \\ \phi_i^0 = 0 & \text{otherwise.} \end{cases} \quad (2.23)$$

The space S_h^0 is then defined as the N -dimensional functional space generated

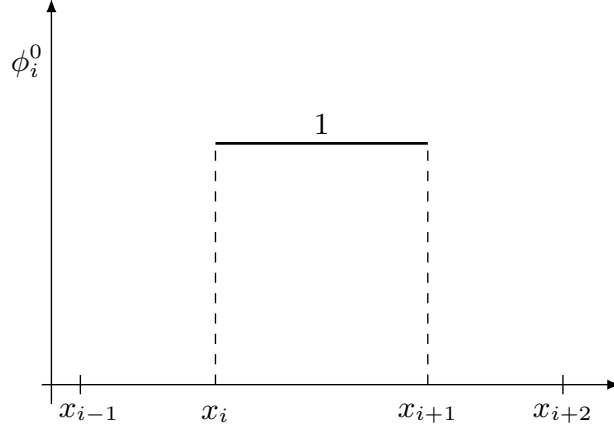


Figure 2.1: Piecewise constant function.

by this basis

$$\mathbf{S}_h^0 = \left\{ f \in L^2(\Omega) : f = \sum_{i=0}^{N-1} \alpha_i \phi_i^0, \alpha_i \in \mathbb{R} \right\}, \quad (2.24)$$

which, as $N \rightarrow \infty$, coincides with L^2 .

In this way, a function $f \in L^2(\Omega)$ can be approximated by another function $f_h \in S_h^0(\Omega)$ defined as

$$f_h = \sum_{i=0}^{N-1} \alpha_i \phi_i^0, \quad (2.25)$$

where α_i depends on how you want to approximate the function f . In the case of a left approximation, the value of α_i is given by $f(x_i)$, whereas for a right approximation, α_i is defined as $f(x_{i+1})$. Alternatively, for a central approximation α_i is computed as the average $\frac{f(x_i)+f(x_{i+1})}{2}$, an example is reported in Figure 2.2.

Considering each interval $\Omega_i = [x_i, x_{i+1}] = [x_1^i, x_2^i]$, it is useful to pass from this global formulation to a local one by transforming global coordinates into

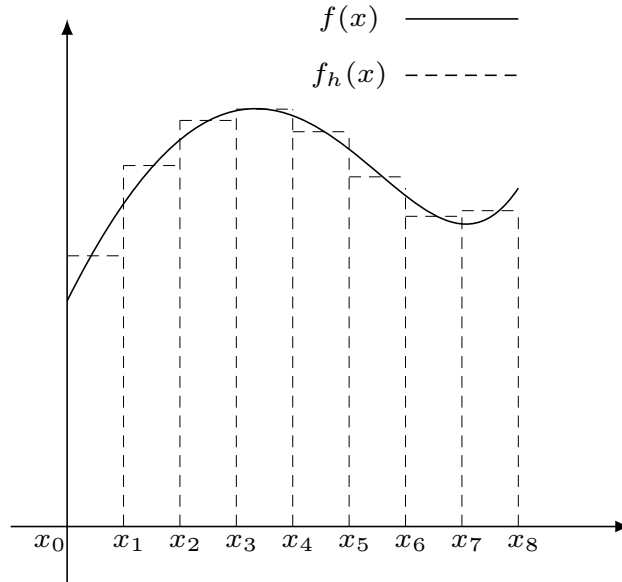


Figure 2.2: Piecewise constant approximation.

local *canonical* coordinates

$$\xi_i = -1 + 2 \frac{x - x_i}{x_{i+1} - x_i}. \quad (2.26)$$

Then, defining the shape function N_1^0 corresponding to the restriction of ϕ_i^0 on the i -th element

$$\begin{cases} N_1^0 = 1 & \text{for } -1 < \xi < 1, \\ N_1^0 = 0 & \text{otherwise,} \end{cases} \quad (2.27)$$

it is possible to define the approximation of f on Ω_i as

$$f_h^i = \alpha_i N_1^{0,i}, \quad (2.28)$$

where α_i is calculated as before. Obviously, in the case of piecewise constant functions, only one shape function is needed.

Approximation in X_h^1 . Unlike L^2 , the space H^1 consists of continuous functions since their first derivatives must also be integrable. Therefore, the most general representation of this space is through a basis of infinite linear polynomials, see Figure 2.3. It is still possible to approximate the infinite-dimensional space through a finite basis.

Using the same partition into N intervals of the domain $\Omega \subseteq \mathbb{R}^1$ defined earlier, the function ϕ_i^1 , called the *first-order Lagrange polynomial*, is defined as

$$\begin{cases} \phi_i^1 = \frac{x - x_{i-1}}{x_i - x_{i-1}} & \text{for } x_{i-1} < x < x_i, \\ \phi_i^1 = \frac{x_{i+1} - x}{x_{i+1} - x_i} & \text{for } x_i < x < x_{i+1}, \\ \phi_i^1 = 0 & \text{otherwise.} \end{cases} \quad (2.29)$$

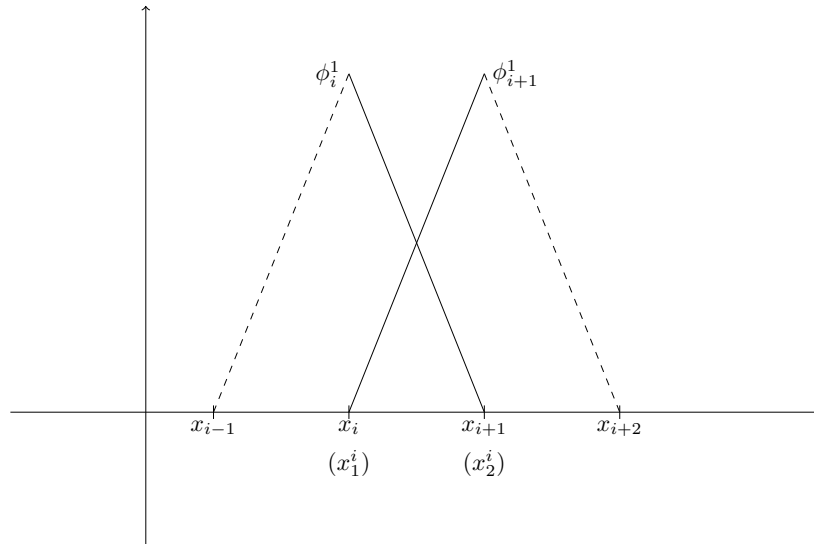


Figure 2.3: Linear Lagrange polynomial.

The space X_h^1 is then defined as the N -dimensional functional space generated by this basis, such that

$$\mathbf{X}_h^1 = \left\{ f \in H^1(\Omega) : f = \sum_{i=0}^N \alpha_i \phi_i^1, \alpha_i \in \mathbb{R} \right\}. \quad (2.30)$$

As before, as $N \rightarrow \infty$, X_h^1 coincides with H^1 .

Given a function $f \in H^1$, it can be approximated by an element of the space X_h^1 defined as

$$f_h = \sum_{i=0}^N \alpha_i \phi_i^1. \quad (2.31)$$

In this case, it is noted that at each point x_i , only the function ϕ_i^1 is non-zero and specifically equal to 1. Therefore, α_i corresponds to the value of the approximated function at that point $f(x_i)$, an illustration is reported in Figure 2.4.

The restriction of the functions ϕ_i^1 and ϕ_{i+1}^1 on Ω_i defines the two canonical shape functions

$$\begin{cases} N_1^{1,i} = \frac{1 - \xi_i}{2} & \text{for } -1 < \xi_i < 1, \\ N_1^{1,i} = 0 & \text{otherwise,} \end{cases} \quad (2.32)$$

$$\begin{cases} N_2^{1,i} = \frac{1 + \xi_i}{2} & \text{for } -1 < \xi_i < 1, \\ N_2^{1,i} = 0 & \text{otherwise.} \end{cases} \quad (2.33)$$

Therefore, the representation of the function f on the i -th element becomes

$$f_h^i(x) = \sum_{j=1}^2 N_j^1(\xi(x)) f(x_j^i). \quad (2.34)$$

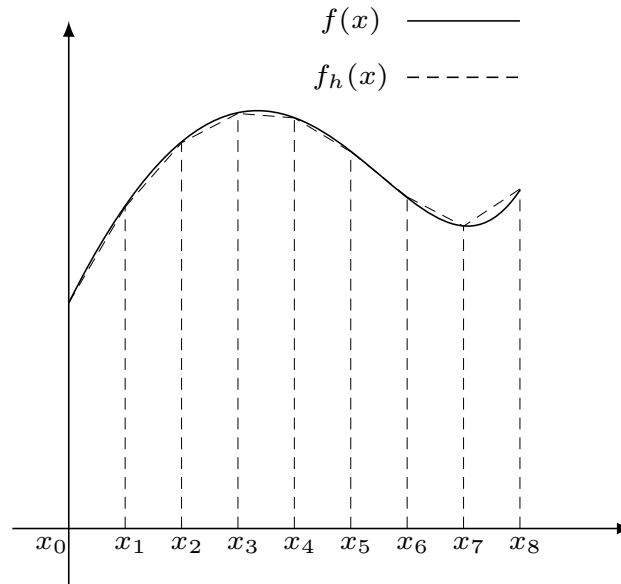


Figure 2.4: Linear approximation.

Approximation in X_h^2 . A second possible approximation of the space H^1 is given by using quadratic polynomials, known as *quadratic Lagrange polynomials*. They are reported in Figure 2.2.1.

Using the usual partition of Ω into N intervals, where in this case N must be even and the $N + 1$ points are divided between vertices and nodes according to the criterion

$$\begin{cases} i = 0, 2, \dots, N \rightarrow \text{vertices,} \\ i = 1, 3, \dots, N - 1 \rightarrow \text{nodes.} \end{cases} \quad (2.35)$$

Each element $\Omega_i = [x_{2i}, x_{2i+2}]$ consists of two vertices and one central node. These are assigned the following local nomenclature: the vertex x_{2i} is denoted as x_1^i , the vertex x_{2i+2} is denoted as x_2^i , and the central node x_{2i+1} is denoted

as x_3^i . A graphical representation is shown in Figure 2.5.

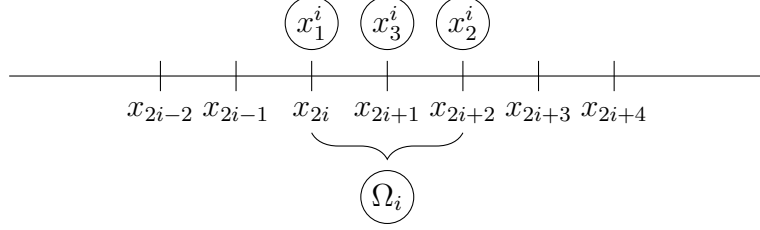


Figure 2.5: Local nomenclature of a 1D quadratic element.

The Lagrange polynomials ϕ_i^2 that form the basis of the space X_h^2 are defined based on which point x_i they belong to. If x_i is a vertex

$$\begin{cases} \phi_i^2 = \frac{(x - x_{i-1})(x - x_{i-2})}{(x_i - x_{i-1})(x_i - x_{i-2})} & \text{for } x_{i-2} < x \leq x_i, \\ \phi_i^2 = \frac{(x - x_{i+1})(x - x_{i+2})}{(x_i - x_{i+1})(x_i - x_{i+2})} & \text{for } x_i < x < x_{i+2}, \\ \phi_i^2 = 0 & \text{otherwise,} \end{cases} \quad (2.36)$$

and if it is a central node

$$\begin{cases} \phi_i^2 = \frac{(x - x_{i-1})(x - x_{i+1})}{(x_i - x_{i-1})(x_i - x_{i+1})} & \text{for } x_{i-1} < x < x_{i+1}, \\ \phi_i^2 = 0 & \text{otherwise.} \end{cases} \quad (2.37)$$

The space X_h^2 is then defined as the $N+1$ dimensional functional space generated by this basis

$$\mathbf{X}_h^2 = \left\{ f \in H^1(\Omega) : f = \sum_{i=0}^N \alpha_i \phi_i^2, \alpha_i \in \mathbb{R} \right\}, \quad (2.38)$$

which, as $N \rightarrow \infty$, coincides with H^1 . Given a function $f \in H^1$, it can be

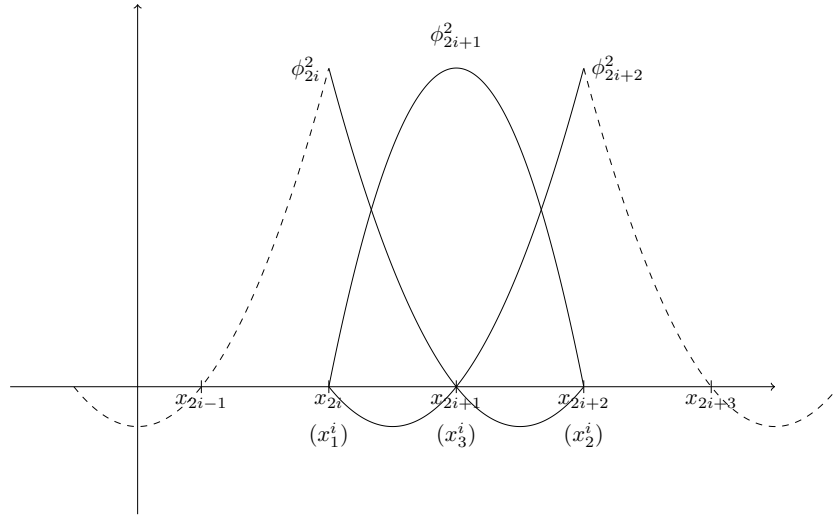


Figure 2.6: Quadratic Lagrange polynomials.

approximated by

$$f_h = \sum_{i=0}^N \alpha_i \phi_i^2, \quad (2.39)$$

where $\alpha_i = f(x_i)$. A case of quadratic approximation is displayed in Figure 2.7.

The canonical shape functions are obtained by restricting the basis of X_h^2 to the element Ω_i

$$\begin{cases} N_1^{2,i} = \frac{1}{2}(1 - \xi)\xi & \text{for } -1 < \xi < 1, \\ N_1^{2,i} = 0 & \text{otherwise,} \end{cases} \quad (2.40)$$

$$\begin{cases} N_2^{2,i} = \frac{1}{2}(1 + \xi)\xi & \text{for } -1 < \xi < 1, \\ N_2^{2,i} = 0 & \text{otherwise,} \end{cases} \quad (2.41)$$

$$\begin{cases} N_3^{2,i} = (1 - \xi)(1 + \xi) & \text{for } -1 < \xi < 1, \\ N_3^{2,i} = 0 & \text{otherwise,} \end{cases} \quad (2.42)$$

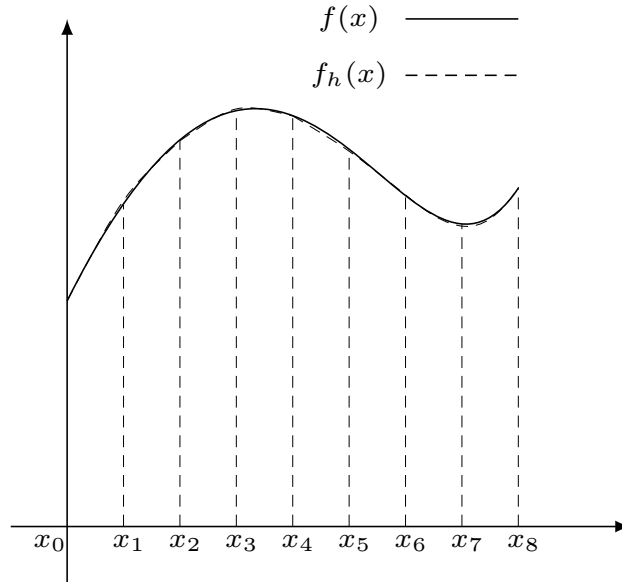


Figure 2.7: Quadratic approximation.

and the representation of the function on the i -th element can be written as

$$f_h^i = \sum_{j=1}^3 N_j^2(\xi(x)) f(x_j^i). \quad (2.43)$$

2.2.2 Two-Dimensional Case

The discussion is extended to the two-dimensional case by considering a convex domain $\Omega \subseteq \mathbb{R}^2$ with boundary Γ . A partition of this domain into N subdomains, or elements, Ω_e is performed such that the union of all elements satisfies $\cup_{e=0}^{N-1} \Omega_e = \Omega$. Furthermore, each element is distinct, ensuring that $\Omega_j \neq \Omega_k$ for all $j \neq k$. This subdivision of the domain is also called a *mesh* and is made up of elements, nodes, and vertices (which in Figure 2.8 are represented respectively by squares and circles): each point $\{P_n\}_{n=0}^{N_n}$

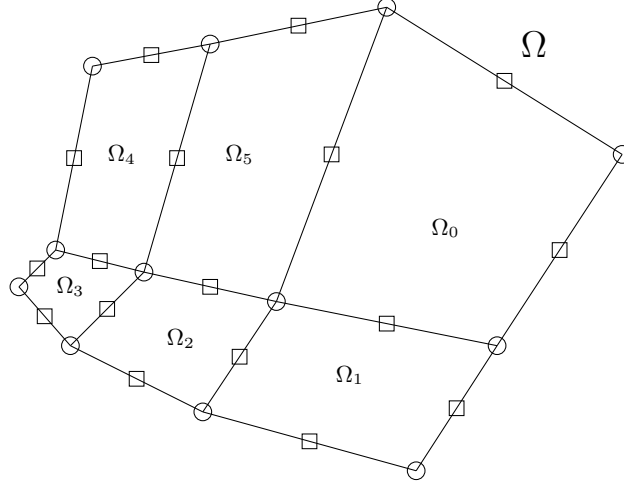


Figure 2.8: Two-dimensional domain partition.

is represented by its coordinates \mathbf{x}_n . Additionally, it is useful to define a connectivity map $P_n(P_i, e)$ that returns the global numbering P_n of each point given its local numbering P_i and the element to which it belongs e . In this way, through also the appropriate formulation of canonical elements, it is possible to reformulate the global problem into different local problems, significantly simplifying it.

Approximation in S_h^0 . There are no particular differences from the one-dimensional case: the space $L^2(\Omega)$ is approximated by $S_h^0(\Omega)$, and the elements of the N -dimensional basis that generate this space are the functions

$$\begin{cases} \phi_i^0(\mathbf{x}) = 1 & \text{for } \mathbf{x} \in \Omega_i, \\ \phi_i^0(\mathbf{x}) = 0 & \text{otherwise.} \end{cases} \quad (2.44)$$

Therefore, given any function $f \in L^2(\Omega)$, it can be approximated by

$$f_h(\mathbf{x}) = \sum_{i=0}^{N-1} \alpha_i \phi_i^0(\mathbf{x}), \quad (2.45)$$

where α_i is the average value of f over the element Ω_i .

Approximation in X_h^1 . For simplicity, we start from the local formulation and then move to the global formulation. Consider the partition of $\Omega \subseteq \mathbb{R}^2$ described earlier, and take an element Ω_i defined by the four vertices $[(x_1^i, y_1^i), (x_2^i, y_2^i), (x_3^i, y_3^i), (x_4^i, y_4^i)]$, this type of element is called a *QUAD4*.

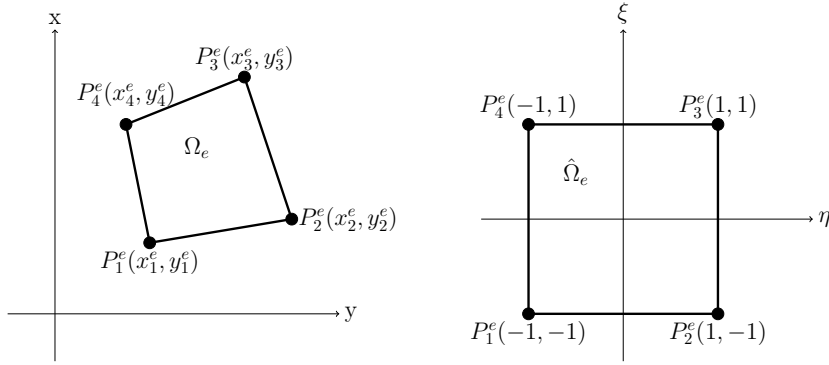


Figure 2.9: Transformation from global coordinates to canonical coordinates for a QUAD4 element.

Starting from this element, we transform it to the canonical element by converting global coordinates to canonical coordinates (see figure 2.9), associating the four vertices of the element to the four vertices of the canonical square $[(-1, -1), (-1, 1), (1, 1), (1, -1)]$. Then, by taking the tensor product of the one-dimensional linear shape functions previously seen, we obtain the four shape functions for the two-dimensional case

$$\begin{aligned}
 N_1^1(\xi, \eta) &= N_1^1(\xi)N_1^1(\eta) = \frac{1-\xi}{2} \frac{1-\eta}{2}, \\
 N_2^1(\xi, \eta) &= N_2^1(\xi)N_1^1(\eta) = \frac{1+\xi}{2} \frac{1-\eta}{2}, \\
 N_3^1(\xi, \eta) &= N_2^1(\xi)N_2^1(\eta) = \frac{1+\xi}{2} \frac{1+\eta}{2}, \\
 N_4^1(\xi, \eta) &= N_1^1(\xi)N_2^1(\eta) = \frac{1-\xi}{2} \frac{1+\eta}{2},
 \end{aligned} \tag{2.46}$$

which are equal to 1 at their respective vertices and 0 at all other points, see figure 2.10.

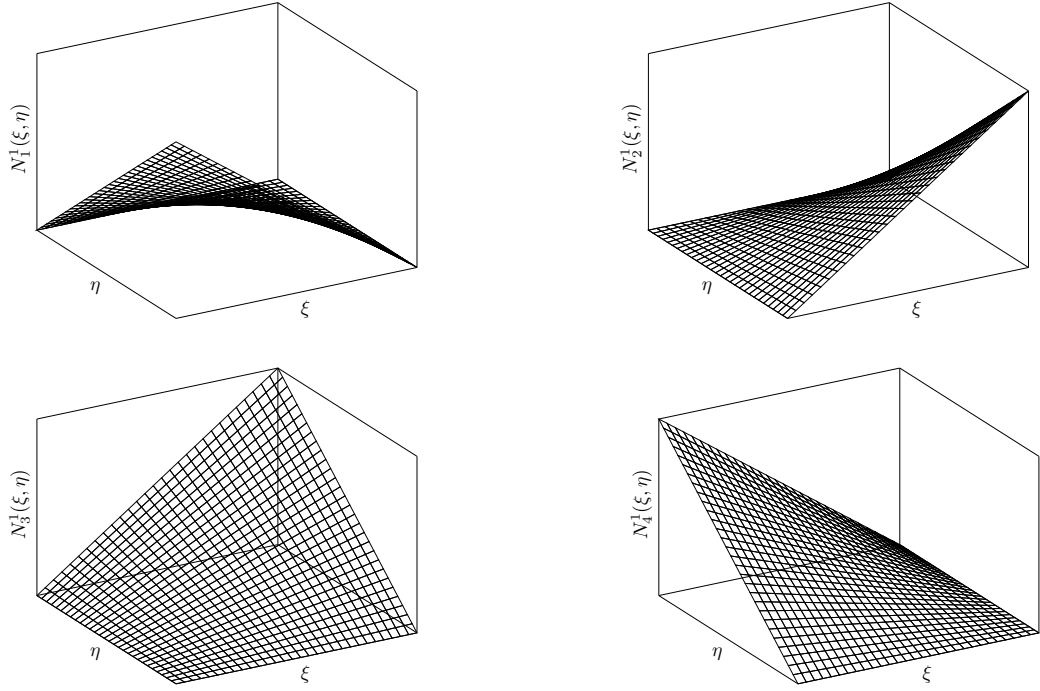


Figure 2.10: Shape functions for X_h^1 on the 2D canonical domain.

Returning to global coordinates via the transformation

$$\mathbf{x}^i(\xi, \eta) = \sum_{j=1}^4 \mathbf{x}_j^i N_j^{1,i}(\xi, \eta), \quad (2.47)$$

one can approximate any function f over the element Ω_i , knowing its values $\{f(\mathbf{x}_j^i)\}_{j=1}^4$ at the vertices of the element Ω_i

$$f_h^i(\mathbf{x}^i(\xi, \eta)) = \sum_{j=1}^4 f(\mathbf{x}_j^i) N_j^1(\xi, \eta). \quad (2.48)$$

Finally, to extend the treatment to the entire domain Ω , it is necessary to

sum the contributions of all N individual elements

$$f_h(\mathbf{x}) = \sum_{i=0}^{N-1} f_h^i(\mathbf{x}^i(\xi, \eta)). \quad (2.49)$$

Approximation in X_h^2 . In this case, consider a *QUAD9* element Ω_i defined by the vertices $[(x_1^i, y_1^i), (x_2^i, y_2^i), (x_3^i, y_3^i), (x_4^i, y_4^i)]$ and the nodes $[(x_5^i, y_5^i), (x_6^i, y_6^i), (x_7^i, y_7^i), (x_8^i, y_8^i), (x_9^i, y_9^i)]$, and then transform it into the canonical element reported in Figure 2.11. As before, the shape functions, which must be equal to

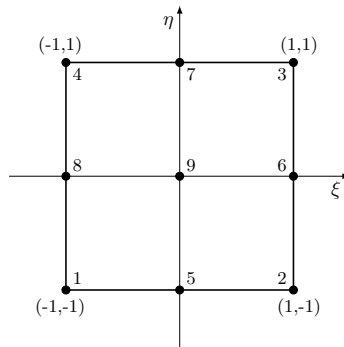


Figure 2.11: Canonical QUAD9 element.

1 at their respective nodes and 0 at all other nodes, are defined by the tensor product of the quadratic one-dimensional shape functions. Their graphical

representation is shown in Figure 2.12.

$$\begin{aligned}
N_1^2(\xi, \eta) &= N_1^2(\xi)N_1^2(\eta) = \xi \frac{1-\xi}{2} \eta \frac{1-\eta}{2}, \\
N_2^2(\xi, \eta) &= N_2^2(\xi)N_1^2(\eta) = \xi \frac{1+\xi}{2} \eta \frac{\eta-1}{2}, \\
N_3^2(\xi, \eta) &= N_2^2(\xi)N_2^2(\eta) = \xi \frac{1+\xi}{2} \eta \frac{1+\eta}{2}, \\
N_4^2(\xi, \eta) &= N_1^2(\xi)N_2^2(\eta) = \xi \frac{\xi-1}{2} \eta \frac{1+\eta}{2}, \\
N_5^2(\xi, \eta) &= N_3^2(\xi)N_1^2(\eta) = (1-\xi^2) \eta \frac{\eta-1}{2}, \\
N_6^2(\xi, \eta) &= N_2^2(\xi)N_3^2(\eta) = \xi \frac{1+\xi}{2} (1-\eta^2), \\
N_7^2(\xi, \eta) &= N_3^2(\xi)N_2^2(\eta) = (1-\xi^2) \eta \frac{1+\eta}{2}, \\
N_8^2(\xi, \eta) &= N_1^2(\xi)N_3^2(\eta) = \xi \frac{\xi-1}{2} (1-\eta^2), \\
N_9^2(\xi, \eta) &= N_3^2(\xi)N_3^2(\eta) = (1-\xi^2)(1-\eta^2).
\end{aligned} \tag{2.50}$$

Finally, the approximation of a function $f \in H^1(\Omega)$ in the space X_h^2 on the element is

$$f_h^i(\mathbf{x}^i(\xi, \eta)) = \sum_{j=1}^9 f(\mathbf{x}_j^i) N_j^2(\xi, \eta), \tag{2.51}$$

while at the global level

$$f_h(\mathbf{x}) = \sum_{i=0}^{N-1} f_h^i(\mathbf{x}^i(\xi, \eta)). \tag{2.52}$$

2.2.3 Gaussian Integration

On the canonical element, the *test functions* ϕ_i is represented by the polynomial shape functions N_j . The Gaussian quadrature method is highly effective for integrating on canonical coordinates. It approximates the integral by a weighted sum of the integrated function evaluated at n points. Its formula-

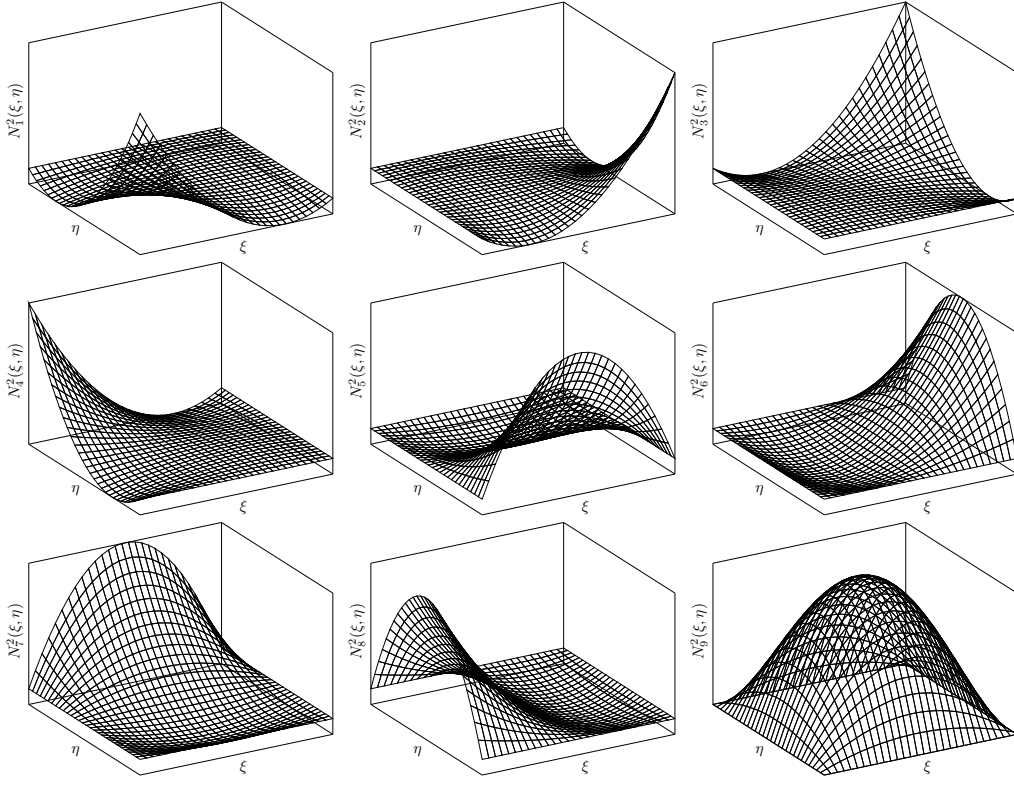


Figure 2.12: Shape functions for X_h^2 on the 2D canonical domain.

tion in the two-dimensional case is

$$\int_{\hat{\Omega}_e} f(\xi, \eta) d\xi d\eta = \sum_{g=1}^n \omega_g f(\xi_g, \eta_g), \quad (2.53)$$

and it is exact for polynomials of degree less than or equal to $2n - 1$ in the one-dimensional case and $2\sqrt[n]{n} - 1$ in the k -dimensional case. The weights ω_g and the coordinates (ξ_g, η_g) must be chosen appropriately, according to the criterion shown in figure 2.13. Due to its accuracy and simplicity, this technique is widely used for numerical applications as in the finite element method, which will be discussed later.

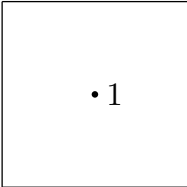
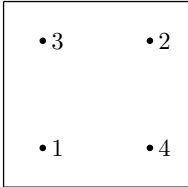
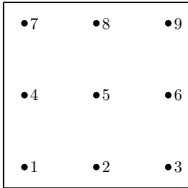
m	n_p	p	ξ_p	η_p	W_p	Position of points
1	1	1	0	0	4	
3	4	1	$-\frac{1}{\sqrt{3}}$	$-\frac{1}{\sqrt{3}}$	1	
		2	$\frac{1}{\sqrt{3}}$	$-\frac{1}{\sqrt{3}}$	1	
		3	$-\frac{1}{\sqrt{3}}$	$\frac{1}{\sqrt{3}}$	1	
		4	$\frac{1}{\sqrt{3}}$	$\frac{1}{\sqrt{3}}$	1	
5	9	1	$-\sqrt{\frac{3}{5}}$	$-\sqrt{\frac{3}{5}}$	$\frac{25}{81}$	
		2	0	$-\sqrt{\frac{3}{5}}$	$\frac{40}{81}$	
		3	$\sqrt{\frac{3}{5}}$	$-\sqrt{\frac{3}{5}}$	$\frac{25}{81}$	
		4	$-\sqrt{\frac{3}{5}}$	0	$\frac{40}{81}$	
		5	0	0	$\frac{64}{81}$	
		6	$\sqrt{\frac{3}{5}}$	0	$\frac{40}{81}$	
		7	$-\sqrt{\frac{3}{5}}$	$\sqrt{\frac{3}{5}}$	$\frac{25}{81}$	
		8	0	$\sqrt{\frac{3}{5}}$	$\frac{40}{81}$	
		9	$\sqrt{\frac{3}{5}}$	$\sqrt{\frac{3}{5}}$	$\frac{25}{81}$	

Figure 2.13: Weights and coordinates for two-dimensional Gaussian quadrature.

2.3 Numerical Solution Techniques for PDEs

The previous sections provided an overview of the functional spaces needed for partial differential problems, particularly in their weak formulation, and described their discretization. With this knowledge, it is possible to introduce some tools for the numerical solution of a partial differential equation. Specifically, we will discuss the *Finite Element Method* and the *Finite Volume Method*. Both methods solve the problem starting from its weak formulation, allowing solutions with realistic regularity requirements for engineering applications.

2.3.1 Finite Element Method

This technique belongs to the *Galerkin Method* class, where the solution is not obtained through the approximation of differential operators but through the approximation of the space in which the solution is sought. The method is based on partitioning the domain into elements, within which the solution is given by a linear combination of the solution at the nodes weighted by the shape functions

$$f_h^i(\mathbf{x}^i(\xi, \eta)) = \sum_{j=1}^{n_e} f(\mathbf{x}_j^i) N_j(\xi, \eta). \quad (2.54)$$

The degrees of freedom, i.e., the unknowns, of the problem are the values of the function at the nodes, which allows converting a differential problem into an algebraic one.

Poisson Equation. Consider a domain $\Omega \subseteq \mathbb{R}^2$, and let us address the solution of the Poisson problem defined in its weak formulation in (2.17)

$$\int_{\Omega} \nabla \phi \cdot \nabla f d\mathbf{x} = \int_{\Omega} \phi q d\mathbf{x}, \quad \forall \phi \in H_{\Gamma}^1(\Omega). \quad (2.55)$$

Approximate the solution space H^1 with X_h^1

$$\begin{aligned} f \in H^1(\Omega) &\rightarrow f_h \in X_h^1(\Omega), \\ \phi \in H^1(\Omega) &\rightarrow \phi_h \in X_h^1(\Omega), \end{aligned} \quad (2.56)$$

and rewrite the problem as

$$\int_{\Omega} \nabla \phi_h \cdot \nabla f_h \, d\mathbf{x} = \int_{\Omega} \phi_h q \, d\mathbf{x}, \quad \forall \phi_h \in X_h^1(\Omega). \quad (2.57)$$

Since this must hold for any test function $\phi_h \in X_h^1$, we choose the shape functions ϕ_i^1 as test functions, where $i = 0, 1, \dots, N_{dof} - 1$.

Next, divide the integral over the entire domain into the sum of integrals over individual elements

$$\sum_{e=0}^{N-1} \int_{\Omega_e} \nabla \phi_i^1 \cdot \nabla f_h \, d\mathbf{x} = \sum_{e=0}^{N-1} \int_{\Omega_e} \phi_i^1 q \, d\mathbf{x}, \quad (2.58)$$

and express f_h as a linear combination of the values at the nodes weighted by the shape functions

$$\sum_{e=0}^{N-1} \sum_{j=1}^{n_e} f(\mathbf{x}_j^e) \int_{\Omega_e} \nabla \phi_i^1 \cdot \nabla_{\mathbf{x}} N_j^1(\xi_e(\mathbf{x}), \eta_e(\mathbf{x})) \, d\mathbf{x} = \sum_{e=0}^{N-1} \int_{\Omega_e} \phi_i^1 q \, d\mathbf{x}. \quad (2.59)$$

Considering each integral separately, we note that the only non-zero test functions ϕ_i^1 on element Ω_e are those corresponding to the nodes of the element. Therefore, we can rewrite the previous equation as

$$\sum_{j=1}^{n_e} f(\mathbf{x}_j^e) \int_{\Omega_e} \nabla_{\mathbf{x}} N_i^1(\xi(\mathbf{x}), \eta(\mathbf{x})) \cdot \nabla_{\mathbf{x}} N_j^1(\xi(\mathbf{x}), \eta(\mathbf{x})) \, d\mathbf{x} = \quad (2.60)$$

$$= \int_{\Omega_e} N_i^1(\xi(\mathbf{x}), \eta(\mathbf{x})) q \, d\mathbf{x}. \quad (2.61)$$

Perform a change of variables to transform the integral over element Ω_e into an integral over the canonical element $\hat{\Omega}_e$, calculating the Jacobian of the coordinate transformation $\mathbf{x} = (x, y) \rightarrow (\xi, \eta)$ defined as

$$J = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{n_e} x_j^e \frac{\partial N_j^1}{\partial \xi} & \sum_{j=1}^{n_e} x_j^e \frac{\partial N_j^1}{\partial \eta} \\ \sum_{j=1}^{n_e} y_j^e \frac{\partial N_j^1}{\partial \xi} & \sum_{j=1}^{n_e} y_j^e \frac{\partial N_j^1}{\partial \eta} \end{bmatrix}, \quad (2.62)$$

such that

$$\nabla_{\mathbf{x}} N_i^1 = \nabla N_i^1 \cdot J. \quad (2.63)$$

Thus, the integral in canonical coordinates is written as

$$\sum_{j=1}^{n_e} f(\mathbf{x}_j^e) \int_{\hat{\Omega}_e} \nabla N_i^1 \cdot \nabla N_j^1 |J| d\xi d\eta = \int_{\hat{\Omega}_e} N_i^1 q(\mathbf{x}^e(\xi, \eta)) |J| d\xi d\eta, \quad (2.64)$$

which, when solved using Gaussian quadrature, becomes

$$\begin{aligned} \sum_{j=1}^{n_e} f(\mathbf{x}_j^e) \sum_{g=1}^{n_g} \omega_g \nabla N_i^1(\xi_g, \eta_g) \cdot \nabla N_j^1(\xi_g, \eta_g) |J(\xi_g, \eta_g)| = \\ \sum_{g=1}^{n_g} \omega_g N_i^1(\xi_g, \eta_g) q(\mathbf{x}^e(\xi_g, \eta_g)) |J(\xi_g, \eta_g)|. \end{aligned} \quad (2.65)$$

Defining

$$\begin{aligned} a_{i,j}^e &= \sum_{g=1}^{n_g} \omega_g \nabla N_i^1(\xi_g, \eta_g) \cdot \nabla N_j^1(\xi_g, \eta_g) |J(\xi_g, \eta_g)|, \\ b_i^e &= \sum_{g=1}^{n_g} \omega_g N_i^1(\xi_g, \eta_g) q(\mathbf{x}^e(\xi_g, \eta_g)) |J(\xi_g, \eta_g)|, \end{aligned} \quad (2.66)$$

the local algebraic system can be written as

$$\begin{bmatrix} a_{1,1}^e & a_{1,2}^e & a_{1,3}^e & a_{1,4}^e \\ a_{2,1}^e & a_{2,2}^e & a_{2,3}^e & a_{2,4}^e \\ a_{3,1}^e & a_{3,2}^e & a_{3,3}^e & a_{3,4}^e \\ a_{4,1}^e & a_{4,2}^e & a_{4,3}^e & a_{4,4}^e \end{bmatrix} \begin{bmatrix} f(\mathbf{x}_1^e) \\ f(\mathbf{x}_2^e) \\ f(\mathbf{x}_3^e) \\ f(\mathbf{x}_4^e) \end{bmatrix} = \begin{bmatrix} b_1^e \\ b_2^e \\ b_3^e \\ b_4^e \end{bmatrix}, \quad (2.67)$$

which condenses into

$$A^e x = b. \quad (2.68)$$

Using the connectivity map $P_n(P_i, e)$, which, given the local numbering of the node and the reference element, returns the global node numbering, it is possible to assemble the global system

$$Ax = b. \quad (2.69)$$

In figure 2.14, the process of transferring the local matrix A^e to the global matrix A is shown. The procedure for the vector b is analogous.

2.3.2 Finite Volume Method

This method does not belong to the *Galerkin method* class anymore, even though it can be grouped in the *Discontinuous Galerkin method* class. In the same way as the Finite Element Method, the Finite Volume Method approximates the space in which the solution is sought partitioning the domain into elements, but this time the solution is considered to be constant within each element creating a piece-wise constant field. In this case, the unknowns of the problems are the values of the function in the cells.

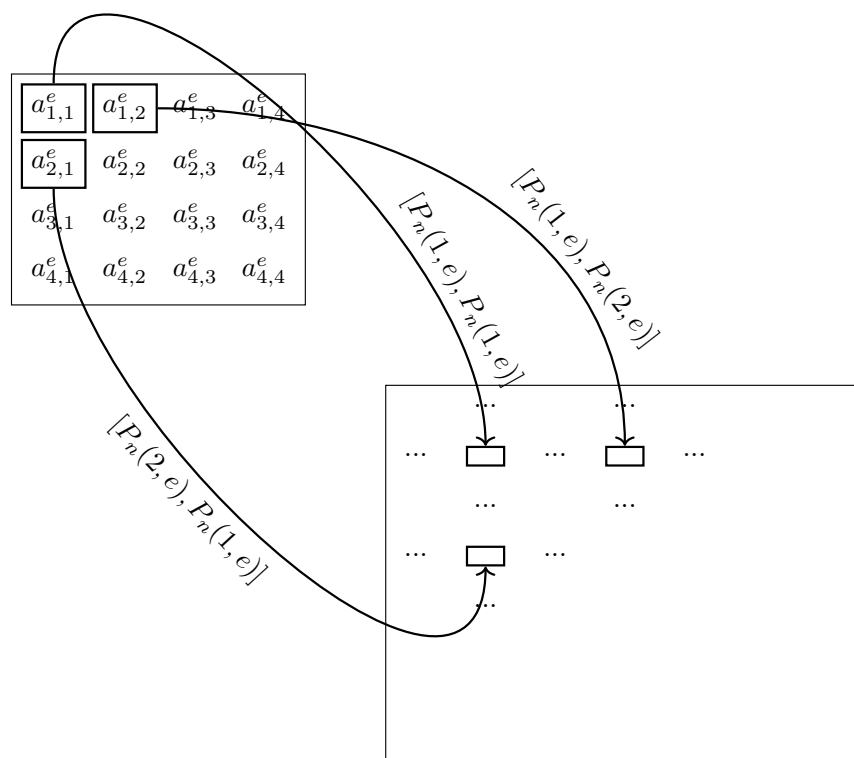


Figure 2.14: Assembly of the local matrix A^e into the global matrix A .

Poisson Equation. Consider a domain $\Omega \subseteq \mathbb{R}^2$, and let's address the solution of the Poisson problem defined in its weak formulation in (2.17)

$$-\int_{\Omega} \phi \Delta f d\mathbf{x} = \int_{\Omega} \phi q d\mathbf{x}, \quad \forall \phi \in L^2(\Omega), \quad (2.70)$$

where Δ is addressed to the Laplacian operator.

This time the solution f is sought in L^2 instead of in H^1 , the solution

space L^2 is approximated with S_h^0

$$\begin{aligned} f \in L^2(\Omega) &\rightarrow f_h \in S_h^0(\Omega), \\ \phi \in L^2(\Omega) &\rightarrow \phi_h \in S_h^0(\Omega), \end{aligned} \quad (2.71)$$

and the problem is rewritten as

$$-\int_{\Omega} \phi_h^0 \Delta f_h \, d\mathbf{x} = \int_{\Omega} \phi_h^0 q \, d\mathbf{x}, \quad \forall \phi_h^0 \in S_h^0(\Omega). \quad (2.72)$$

Since this must hold for any test function $\phi_h^0 \in S_h^0$, we choose the shape functions ϕ_i^0 as test functions, where $i = 0, 1, \dots, N_{dof} - 1$.

As the first step, we rewrite the equation performing the *Reconstruction of the Gradient*

$$\int_{\Omega} \phi_i^0 \nabla \cdot J_h \, d\mathbf{x} = \int_{\Omega} \phi_i^0 q \, d\mathbf{x}, \quad \forall \phi_i^0 \in S_h^0(\Omega), \quad (2.73)$$

where $J_h = -\nabla f_h$. Next, divide the integral over the entire domain into the sum of integrals over individual elements

$$\sum_{e=0}^{N-1} \int_{\Omega_e} \phi_i^0 \nabla \cdot J_h^e \, d\mathbf{x} = \sum_{e=0}^{N-1} \int_{\Omega_e} \phi_i^0 q \, d\mathbf{x}, \quad \forall \phi_i^0 \in S_h^0(\Omega), \quad (2.74)$$

and since the only ϕ_i^0 non-null on the element Ω_e is $N^{0,e}$, we can rewrite it as

$$\sum_{e=0}^{N-1} \int_{\Omega_e} N^{0,e} \nabla \cdot J_h^e \, d\mathbf{x} = \sum_{e=0}^{N-1} \int_{\Omega_e} N^{0,e} q \, d\mathbf{x}. \quad (2.75)$$

Then we can integrate by parts the first term of the equation by obtaining

$$\int_{\Omega} N^{0,e} \nabla \cdot J_h \, d\mathbf{x} = \int_{\Gamma} N^{0,e} (J_h \cdot \mathbf{n}) \, d\mathbf{s} - \int_{\Omega} \nabla N^{0,e} \cdot J_h \, d\mathbf{x}, \quad (2.76)$$

and considering that $N^{0,e} = 1$ over Ω_e it becomes

$$\int_{\Omega_e} \nabla \cdot J_h \, d\mathbf{x} = \int_{\Gamma_e} (J_h \cdot \mathbf{n}) \, ds. \quad (2.77)$$

Since the discrete solution f_h is piece-wise constant the reconstructed gradient J_h needs to be modeled, i.e., with a finite difference scheme, considering a mesh made up from square elements the integral above Γ_e can be rewritten as:

$$\int_{\Gamma_e} (J_h \cdot \mathbf{n}) \, ds = -\frac{f_h^e - f_h^{e,S}}{\Delta_S y} \Gamma_S + \frac{f_h^e - f_h^{e,E}}{\Delta_E x} \Gamma_E + \frac{f_h^e - f_h^{e,N}}{\Delta_N y} \Gamma_N - \frac{f_h^e - f_h^{e,O}}{\Delta_O x} \Gamma_O, \quad (2.78)$$

where $f_h^{e,S}, f_h^{e,E}, f_h^{e,N}, f_h^{e,O}$ are respectively the value of f in the element under, on the right, over and on the left of the element Ω_e , where $\Gamma_S, \Gamma_E, \Gamma_N, \Gamma_O$ are considered to be the length of the four frontiers of the square element, and where $\Delta_S y, \Delta_E x, \Delta_N y, \Delta_O x$ are the distance in the four directions from the near elements. Then defining the local matrix A^e terms

$$\begin{aligned} a_0^e &= -\frac{\Gamma_S}{\Delta_S y} + \frac{\Gamma_E}{\Delta_E x} + \frac{\Gamma_N}{\Delta_N y} - \frac{\Gamma_O}{\Delta_O x}, \\ a_1^e &= \frac{\Gamma_S}{\Delta_S y}, \\ a_2^e &= \frac{\Gamma_E}{\Delta_E y}, \\ a_3^e &= \frac{\Gamma_N}{\Delta_N y}, \\ a_4^e &= \frac{\Gamma_O}{\Delta_O y}, \end{aligned} \quad (2.79)$$

and the right-hand side term b^e

$$b^e = \int_{\Omega_e} q \, d\mathbf{x}, \quad (2.80)$$

the local algebraic system can be written as

$$\begin{bmatrix} a_0^e & a_1^e & a_2^e & a_3^e & a_4^e \end{bmatrix} \begin{bmatrix} f^e \\ f^{e,S} \\ f^{e,E} \\ f^{e,N} \\ f^{e,O} \end{bmatrix} = \begin{bmatrix} b^e \end{bmatrix}, \quad (2.81)$$

which condenses into

$$A^e x = b. \quad (2.82)$$

Again, using the connectivity map $P_n(P_i, e)$, which, given the relative position to the element Ω_e (the element itself, the southern element and so on), returns the global elements numbering, it is possible to assemble the global system

$$Ax = b. \quad (2.83)$$

Chapter 3

Optimal Control

3.1 Theory of Optimal Control

An optimization problem aims to control an output variable through the appropriate variation of an input variable [30]. The input variable is called *control*. In the case of the application considered, i.e. heat transport, it can take the form of a volumetric heat source or a temperature or thermal flux imposed on the wall. The output variable is called *state variable*. The desired behavior of the state variable is measured through the appropriate definition of a function \mathcal{F} , called the *objective functional*, generally formulated to be minimized at the optimal state.

Given Hilbert spaces U and H , a control $q \in U$, a state variable $\phi \in H$, $\phi_0 \in H$, and $\lambda \in \mathbb{R}_0^+$, the functional of the optimization problem can be defined as

$$\mathcal{F}(q) = \frac{1}{2} \|\phi - \phi_0\|_{L^2}^2 + \frac{\lambda}{2} \|q\|_U^2. \quad (3.1)$$

The first term is called the *objective term* as it ensures the effectiveness of the imposed control, and the second term is called the *control regularization*

term, as it helps to keep it within finite values. The parameter λ controls the relative importance of the two terms.

It is also possible to add constraints arising from practical needs, such as limiting the control within a certain range of values:

$$a \leq q \leq b. \quad (3.2)$$

The additional constraints to which the system is subjected are primarily represented by the physics to be controlled, and in the case of temperature control the state variable must satisfy the heat equation and its boundary conditions:

$$A\phi = Bq. \quad (3.3)$$

In cases where a partial differential equation forms the constraint, A can be seen as some differential operator [35].

3.1.1 Adjoint Method

The optimization problem can be formulated by finding the functional (3.1) minimum, which is referred to as

$$\mathcal{F}(\phi(q), q) \rightarrow \min,$$

thus the optimal control \bar{q} must satisfy the Euler equation

$$\mathcal{F}'_F(\bar{q}) = 0. \quad (3.4)$$

By explicitly writing the Frechét derivative of the functional concerning the control parameter, we obtain

$$\mathcal{F}'_F(q) \tilde{q} = \int_{\Omega} (\phi - \phi_0) (\phi(q + \tilde{q}) - \phi(q)) d\Omega + \int_{\Omega} q \tilde{q} d\Omega = 0, \quad \forall \tilde{q} \in L^2. \quad (3.5)$$

Although the Euler equation provides a definition of the optimal control, it is not suitable for the numerical solution of the problem. Firstly, because it requires the knowledge of the state variable ϕ both at the control q and at $q + \tilde{q}$, and because it does not provide a formulation for $\nabla \mathcal{F}(q)$, necessary to determine the descent direction in the minimization methods.

For this reason, the functional \mathcal{F} is rewritten using the Lagrange equation, also known as the Lagrangian, an auxiliary function that allows all constraints of the optimization problem to be concentrated into a single equation

$$\mathcal{L}(\phi, q, p) = \mathcal{F}(\phi, q) - \langle A\phi - Bq, p \rangle, \quad (3.6)$$

where p is called the *Lagrange multiplier*. Considering the Poisson problem, it is explicitly rewritten as

$$\mathcal{L} = \frac{1}{2} \int_{\Omega_d} (\phi - \phi_0)^2 d\Omega + \frac{\lambda}{2} \int_{\Omega} q^2 d\Omega + \int_{\Omega} \Delta\phi p d\Omega + \int_{\Omega} q p d\Omega. \quad (3.7)$$

Again, differentiating with respect to q , we obtain

$$\begin{aligned} \mathcal{L}'_F(q) \tilde{q} &= \int_{\Omega} (\phi - \phi_0) (\phi(q + \tilde{q}) - \phi(q)) d\Omega + \lambda \int_{\Omega} q \tilde{q} d\Omega + \\ &+ \int_{\Omega} (\Delta\phi(q + \tilde{q}) - \Delta\phi(q)) p d\Omega + \int_{\Omega} \tilde{q} p d\Omega, \end{aligned} \quad (3.8)$$

from this, integrating by parts twice and substituting $\tilde{\phi} = \phi(q + \tilde{q}) - \phi(q)$, it follows

$$\begin{aligned} \mathcal{L}'_F(q) \tilde{q} &= \int_{\Omega} (\phi - \phi_0) \tilde{\phi} \, d\Omega + \lambda \int_{\Omega} q \tilde{q} \, d\Omega + \int_{\Omega} \tilde{\phi} \Delta p \, d\Omega + \\ &+ \int_{\Omega} \tilde{q} p \, d\Omega. \end{aligned} \quad (3.9)$$

By appropriately choosing the Lagrange multiplier p , it is possible to eliminate $\tilde{\phi}$, in fact, if

$$\int_{\Omega} (\phi - \phi_0) \psi \, d\Omega = \int_{\Omega} \Delta p \psi \, d\Omega \quad , \forall \psi \in H^1, \quad (3.10)$$

then

$$\mathcal{L}'_F(q) \tilde{q} = \int_{\Omega} (\lambda q + p) \tilde{q} \, d\Omega. \quad (3.11)$$

Note that differentiating the Lagrangian with respect to the state variable ϕ would yield the same formulation for the Lagrange multiplier p , this approach will be used later to determine the adjoint system in different types of control. Since $\mathcal{L}'_F(q) \tilde{q}$ represents the action of $\mathcal{L}'_F(q)$ on \tilde{q} as the inner product of $\lambda q + p$ with \tilde{q} , we can write the gradient of the Lagrangian with respect to the control parameter as

$$\nabla \mathcal{L}(q) = \lambda q + p. \quad (3.12)$$

Furthermore, the new formulation of the Euler equation is derived as

$$\mathcal{L}'_F(\bar{q}) \tilde{q} = \int_{\Omega} (\lambda \bar{q} + p) \tilde{q} \, d\Omega = 0 \quad \forall \tilde{q} \in L^2, \quad (3.13)$$

which is equivalent to

$$\bar{q} = -\frac{p}{\lambda}. \quad (3.14)$$

3.1.2 Numerical solution

The numerical solution of optimal control problems relies on iterative algorithms that determine an optimal control \bar{q} through a sequence $(q^{(n)})_{n \in \mathbb{N}}$. Cases where the control q is sought within the whole space \mathbb{R}^n can be distinguished from cases constrained within a subset $K \subset \mathbb{R}^n$. This discussion will be limited to the first case. For solving this type of optimization, the family of *gradient descent* methods will be shown, these methods require knowledge of the gradient of the functional \mathcal{F} , which makes them unsuitable for applications where the functional is difficult to differentiate or even non-differentiable.

Given a solution $q^{(n)}$ at the n -th step, the solution at the next step is obtained by moving from this starting point in the descent direction $\mathbf{d}^{(n)}$ according to the formula

$$\mathbf{q}^{(n+1)} = \mathbf{q}^{(n)} + \rho^{(n)} \mathbf{d}^{(n)}, \quad (3.15)$$

where $\rho^{(n)}$ is called the step size. In gradient descent methods, the direction $\mathbf{d}^{(n)}$ is obtained at each iteration through the calculation of the gradient $\nabla \mathcal{F}$, which indicates the direction of maximum increase of the functional. Below, two different strategies for choosing $\mathbf{d}^{(n)}$ will be briefly discussed: the *steepest gradient* method and the *conjugate gradient* method [18]. Note that, in the previous equations the bold symbol has been used for scalar variables. Nevertheless, for the minimization method, the vectors must be interpreted as numeric vectors, i.e. the collection of the variable values on the degrees of freedom of the computational grid. Therefore, \mathbf{q} represents the numeric vector that contains this value for each node. The same comment can be drawn for \mathbf{d} .

Steepest gradient. The descent direction is set equal to the negative gradient of the functional

$$\mathbf{d}^{(n)} = -\nabla \mathcal{F}(\mathbf{q}^{(n)}), \quad (3.16)$$

if the functional is sufficiently differentiable $\mathcal{F} \in C^2(\mathbb{R}^n)$ and the step size ρ satisfies the following *Wolfe conditions* [30]

$$\begin{aligned} \mathcal{F}(\mathbf{q}^{(n)} + \rho^{(n)} \mathbf{d}^{(n)}) &\leq \mathcal{F}(\mathbf{q}^{(n)}) + \sigma \rho^{(n)} (\mathbf{d}^{(n)})^T \nabla \mathcal{F}(\mathbf{q}^{(n)}), \\ (\mathbf{d}^{(n)})^T \nabla \mathcal{F}(\mathbf{q}^{(n)} + \rho^{(n)} \mathbf{d}^{(n)}) &\geq \delta (\mathbf{d}^{(n)})^T \nabla \mathcal{F}(\mathbf{q}^{(n)}), \end{aligned} \quad (3.17)$$

where $0 < \sigma < \delta < 1$ are known constants, then the method is globally convergent. The first is called the *Armijo rule* and requires that the change in the functional value is proportional, according to a factor σ , to the product of the step size ρ and the directional derivative of the functional $(\mathbf{d}^{(n)})^T \nabla \mathcal{F}(\mathbf{q}^{(n)})$. The second condition ensures that the variational derivative of the functional along the direction $\mathbf{d}^{(n)}$ at $\mathbf{q}^{(n+1)}$ is greater, in absolute value, than that computed at $\mathbf{q}^{(n)}$ weighted by δ . Moreover, in the case of a quadratic functional, the method converges linearly, i.e.

$$|\mathbf{q}^{(n+1)} - \bar{\mathbf{q}}| \leq C |\mathbf{q}^{(n)} - \bar{\mathbf{q}}|, \quad (3.18)$$

with $C < 1$ and n sufficiently large.

Conjugate Gradient. This method improves the convergence of the classical steepest gradient method by moving along the directions

$$\mathbf{d}^{(n)} = -\nabla \mathcal{F}(\mathbf{q}^{(n)}) + \beta^{(n)} \mathbf{d}^{(n-1)}. \quad (3.19)$$

The parameter $\beta^{(n)} \in \mathbb{R}$ is obtained such that the directions $\mathbf{d}^{(n)}$ and $\mathbf{d}^{(n-1)}$ are conjugate with respect to the Hessian matrix \mathbf{H} , satisfying

$$\left(\mathbf{d}^{(n)}\right)^T \mathbf{H} \mathbf{d}^{(n-1)} = 0, \quad (3.20)$$

and where \mathbf{H} is constant, i.e. for a quadratic functional, we obtain the following expressions for β

$$\beta^{(n)} = \frac{\nabla \mathcal{F}(\mathbf{q}^{(n)})^T \mathbf{H} \mathbf{d}^{(n-1)}}{\left(\mathbf{d}^{(n-1)}\right)^T \mathbf{H} \mathbf{d}^{(n-1)}}. \quad (3.21)$$

In order to make the computation easier, the formula (3.21) can be rewritten with the *Fletcher-Reeves* expression [19]

$$\beta_{FR}^{(n)} = \frac{\|\nabla \mathcal{F}(\mathbf{q}^{(n)})\|^2}{\|\nabla \mathcal{F}(\mathbf{q}^{(n-1)})\|^2}, \quad (3.22)$$

that has been implemented in the numerical algorithm.

If the functional $\mathcal{F} \in C^1(\mathbb{R}^n)$ and its gradient is *Lipschitz continuous*, and the step size ρ satisfies the *strong Wolfe conditions*

$$\begin{aligned} \mathcal{F}\left(\mathbf{q}^{(n)} + \rho^{(n)} \mathbf{d}^{(n)}\right) &\leq \mathcal{F}\left(\mathbf{q}^{(n)}\right) + \sigma \rho^{(n)} \left(\mathbf{d}^{(n)}\right)^T \nabla \mathcal{F}\left(\mathbf{q}^{(n)}\right), \\ \left|\left(\mathbf{d}^{(n)}\right)^T \nabla \mathcal{F}\left(\mathbf{q}^{(n)} + \rho^{(n)} \mathbf{d}^{(n)}\right)\right| &\geq -\delta \left(\mathbf{d}^{(n)}\right)^T \nabla \mathcal{F}\left(\mathbf{q}^{(n)}\right), \end{aligned} \quad (3.23)$$

where $0 < \sigma < \delta < \frac{1}{2}$ are known constants, then the method converges globally and linearly.

Steepest Gradient and Conjugate Gradient methods. In this paragraph, the two descent methods Steepest gradient (SG) and Conjugate Gradient (CG) are briefly compared. An application of the two minimization

methods is presented, with an objective function in a two-dimensional domain to be minimized, $f : \mathbb{R}^2 \rightarrow \mathbb{R}$, such that

$$f(x, y) = 3x^2 + 2y^2 + 3xy + 2x + 6y + 5. \quad (3.24)$$

The function has a local minimum in $\bar{\mathbf{x}} = (\frac{2}{3}, -2)$, and its minimum is $f(\bar{\mathbf{x}}) = -\frac{1}{3}$. In Figure 3.1 the trajectories of each method have been reported, this initial comparison aims to qualitatively appreciate the differences in the descent trajectories between the two methods. The convergence has considered reached when $\|\mathbf{x}_{n+1} - \mathbf{x}_n\| \leq \epsilon$, where $\epsilon = 10^{-6}$. The step size ρ is considered fixed.

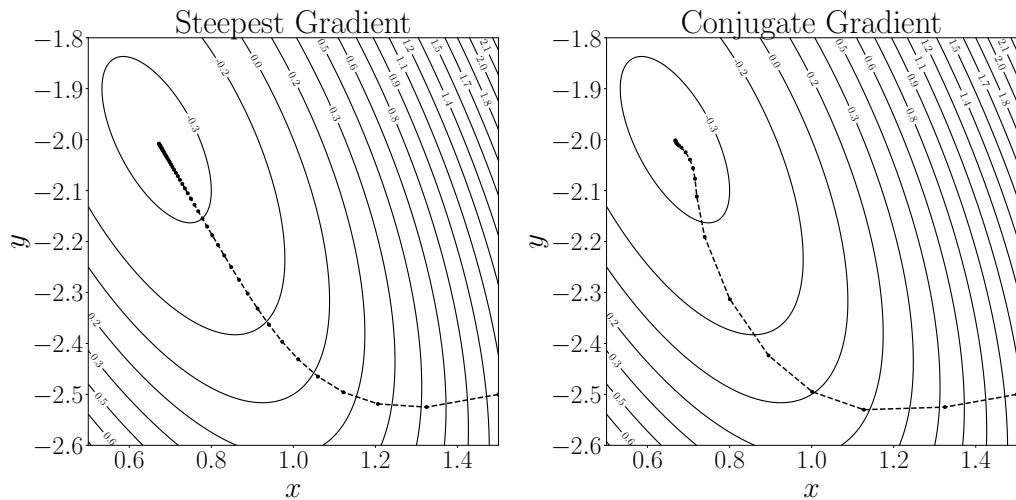


Figure 3.1: Trajectory comparison between the two descent methods: SG and CG.

In Figure 3.1 it is possible to notice how the CG method direction at each step is influenced by the previous direction, this is what leads to the sinuous behavior. In Table 3.1 some data, such as the number of iterations, the functional and the coordinates reached, and their relative error with respect to the theoretical results are reported.

Method	N. of iterations	$f(\mathbf{x})$	$\frac{f(\mathbf{x})-f(\bar{\mathbf{x}})}{\mathbf{x}}$	\mathbf{x}	$\ \frac{\mathbf{x}-\bar{\mathbf{x}}}{\bar{\mathbf{x}}}\ $
SG	46	-0.333248	$2.55 \cdot 10^{-4}$	(0.672, 2.008)	0.009
CG	18	-0.333330	$9.71 \cdot 10^{-6}$	(0.668, 2.002)	0.002

Table 3.1: N. of iterations, functional and coordinates of the minimum, and relative errors of the two minimizations realized with SG and CG methods.

Table 3.1 shows how the CG method consents to obtain better results: both in terms of convergence, since it has the lower number of iterations, and in the quality of the result.

The results above refer to a straightforward case, namely a two-dimensional domain, and aim to provide a qualitative comparison of the two methods in terms of efficiency and trajectories.

Numerical algorithm In this section, the numerical algorithm used for the minimization problem is described. At first the state \mathbf{T} , the adjoint \mathbf{T}_a and control \mathbf{q} variable vectors are initialized, along with the functional \mathcal{F} and the auxiliary variable \mathbf{g}_{old} , that represents the functional gradient at the past iteration. For the first iteration of the time loop the descent direction $\mathbf{d}^{(0)}$ is initialized as the anti-gradient direction, similarly to the steepest gradient method.

After that, the time loop starts and the variables \mathbf{T} , \mathbf{T}_a , \mathbf{q} and the functional \mathcal{F} are updated. Naturally, the time loop must be interpreted as an iteration loop since our system is considered stationary. At this point, if the convergence condition has not been satisfied yet, the new search direction \mathbf{d} is found through the calculation of the current functional \mathbf{g}_{new} and the parameter β . Otherwise, if the convergence is reached, the **while** loop is terminated and the number of iterations and the minimum of the reached convergent functional are collected.

The Algorithm 3.1 illustrates only the mathematical aspect of the application since the coupling algorithm between the codes is shown in the next section. Therefore, this scheme does not take into account which code solves a specific numerical field but shows the logical order for finding the numerical solution considering the necessary steps of the minimization procedure.

Algorithm 3.1 Numerical Algorithm

1: **procedure** main()

Variables initialization.

2: $n \leftarrow 0$
3: $\mathbf{u}^{(0)} \leftarrow 0$
4: $\mathbf{T}^{(0)} \leftarrow \mathbf{T}(\mathbf{u}^{(0)})$
5: $\mathbf{T}_a^{(0)} \leftarrow \mathbf{T}_a(\mathbf{T}^{(0)})$
6: $\mathcal{F}^{(0)} \leftarrow \mathcal{F}(\mathbf{T}^{(0)}, \mathbf{u}^{(0)})$
7: $\mathbf{g}_{old} \leftarrow \nabla \mathcal{F}(\mathbf{u}^{(0)}, \mathbf{T}_a^{(0)})$
8: $\mathbf{s} \leftarrow -\mathbf{g}_{old}$

Time loop.

9: **while** not **stop** **do**
10: $\mathbf{u}^{(n+1)} \leftarrow \mathbf{u}^{(n)} + \rho \cdot \mathbf{s}$
11: $\mathbf{T}^{(n+1)} \leftarrow \mathbf{T}(\mathbf{u}^{(n+1)})$
12: $\mathbf{T}_a^{(n+1)} \leftarrow \mathbf{T}_a(\mathbf{T}^{(n+1)})$
13: $\mathcal{F}^{(n+1)} \leftarrow \mathcal{F}(\mathbf{T}^{(n+1)}, \mathbf{u}^{(n+1)})$
14: **if** $\frac{|\mathcal{F}^{(n+1)} - \mathcal{F}^{(n)}|}{|\mathcal{F}^{(n+1)}|} < \epsilon$ **then**
15: **stop**
16: **end if**
17: $\mathbf{g}_{new} \leftarrow \nabla \mathcal{F}(\mathbf{u}^{(n+1)}, \mathbf{T}_a^{(n+1)})$
18: $\beta \leftarrow \frac{\langle \mathbf{g}_{new}, \mathbf{g}_{new} \rangle}{\langle \mathbf{g}_{old}, \mathbf{g}_{old} \rangle}$
19: $\mathbf{g}_{old} \leftarrow \mathbf{g}_{new}$
20: $\mathbf{s} \leftarrow -\mathbf{g}_{new} + \beta \cdot \mathbf{s}$
21: $n \leftarrow n + 1$
22: **end while**
23: **end procedure**

3.2 Distributed Control

Distributed control refers to when the control parameter acts on the internal part of the solution domain Ω .

The optimal system is derived for stationary heat conduction physics, applied to a domain Ω with boundary Γ . Specifically, a Dirichlet boundary condition is imposed on the partition of Γ denoted by Γ_d , while a Neumann boundary condition is imposed on the partition Γ_n . The state equation is written in strong form as:

$$\begin{cases} -\alpha\Delta T = Q & \text{on } \Omega, \\ T = g_d & \text{on } \Gamma_d, \\ \alpha\nabla T \cdot \mathbf{n} = g_n & \text{on } \Gamma_n. \end{cases} \quad (3.25)$$

Analogous to the Poisson equation, the functional becomes

$$\mathcal{F}(T, g) = \frac{1}{2} \int_{\Omega_d} (T - T_d)^2 d\Omega + \frac{\lambda}{2} \int_{\Omega} Q^2 d\Omega, \quad (3.26)$$

where the regularization norm is defined on L^2 , the space in which the control will be sought.

Euler Equation. By substituting ϕ with T , q with Q , and the Lagrange multiplier p with T_a , the adjoint temperature, we reformulate the Lagrangian as follows:

$$\begin{aligned} \mathcal{L} = & \frac{1}{2} \int_{\Omega_d} (T - T_d)^2 d\Omega + \frac{\lambda}{2} \int_{\Omega} Q^2 d\Omega + \int_{\Omega} (\alpha\Delta T) T_a d\Omega + \\ & + \int_{\Omega} (Q) T_a d\Omega - \int_{\Gamma_d} (T - g_d) T_a d\Gamma - \int_{\Gamma_n} (\alpha\nabla T \cdot \mathbf{n} - g_n) T_a d\Gamma, \end{aligned} \quad (3.27)$$

where the boundary conditions of the problem have also been included. Note that differentiating the Lagrangian with respect to the adjoint variable T_a yields the state equation again:

$$\begin{aligned} \mathcal{L}'_F(T_a) \tilde{T}_a &= \int_{\Omega} (\alpha \Delta T) \tilde{T}_a \, d\Omega + \int_{\Omega} (Q) \tilde{T}_a \, d\Omega - \int_{\Gamma_d} (T - g_d) \tilde{T}_a \, d\Gamma - \\ &\quad - \int_{\Gamma_n} (\alpha \nabla T \cdot \mathbf{n} - g_n) \tilde{T}_a \, d\Gamma, \end{aligned} \quad (3.28)$$

which must hold for every variation \tilde{T}_a , and is equivalent to the state problem (3.25).

Adjoint Problem Differentiating the Lagrangian with respect to the state variable T again, we obtain the system of equations for the adjoint variable T_a :

$$\begin{aligned} \mathcal{L}'_F(T) \tilde{T} &= \int_{\Omega_d} (T - T_d) \tilde{T} \, d\Omega + \int_{\Omega} (\alpha \Delta \tilde{T}) T_a \, d\Omega + \int_{\Omega} Q T_a \, d\Omega - \\ &\quad - \int_{\Gamma_d} (\tilde{T}) T_a \, d\Gamma - \int_{\Gamma_n} (\alpha \nabla \tilde{T} \cdot \mathbf{n}) T_a \, d\Gamma. \end{aligned} \quad (3.29)$$

We integrate by parts to isolate the variation \tilde{T} :

$$\begin{aligned} \int_{\Omega} (\alpha \Delta \tilde{T}) T_a \, d\Omega &= \int_{\Gamma} (\alpha \nabla \tilde{T} \cdot \mathbf{n}) T_a \, d\Gamma - \int_{\Omega} \alpha \nabla \tilde{T} \cdot \nabla T_a \, d\Omega \\ &= \int_{\Gamma} (\alpha \nabla \tilde{T} \cdot \mathbf{n}) T_a \, d\Gamma - \int_{\Gamma} \alpha \tilde{T} (\nabla T_a \cdot \mathbf{n}) \, d\Gamma + \\ &\quad + \int_{\Omega} \alpha \tilde{T} \Delta T_a \, d\Omega, \end{aligned}$$

where, due to the boundary conditions on T , the boundary terms are zero

$$\begin{aligned} \int_{\Gamma} \left(\alpha \nabla \tilde{T} \cdot \mathbf{n} \right) T_a \, d\Gamma &= \int_{\Gamma_d} \left(\alpha \nabla \tilde{T} \cdot \mathbf{n} \right) T_a \, d\Gamma \\ &= \int_{\partial\Gamma_d} \alpha \tilde{T} T_a \, d\partial\Gamma - \int_{\Gamma_d} \alpha \tilde{T} (\nabla T_a \cdot \mathbf{n}) \, d\Gamma = 0, \end{aligned}$$

$$\begin{aligned} \int_{\Gamma} \alpha \tilde{T} (\nabla T_a \cdot \mathbf{n}) \, d\Gamma &= \int_{\Gamma_n} \alpha \tilde{T} (\nabla T_a \cdot \mathbf{n}) \, d\Gamma \\ &= \int_{\partial\Gamma_n} \alpha \tilde{T} T_a \, d\partial\Gamma - \int_{\Gamma_n} \alpha T_a (\nabla \tilde{T} \cdot \mathbf{n}) \, d\Gamma = 0. \end{aligned}$$

To obtain the boundary condition on Γ_n , the respective boundary term is reformulated:

$$\begin{aligned} \int_{\Gamma_n} \left(\alpha \nabla \tilde{T} \cdot \mathbf{n} \right) T_a \, d\Gamma &= \int_{\partial\Gamma_n} \alpha \tilde{T} T_a \, d\partial\Gamma_n - \int_{\Gamma_n} \alpha \tilde{T} \nabla T_a \cdot \mathbf{n} \, d\Gamma \\ &= - \int_{\Gamma_n} \alpha \tilde{T} \nabla T_a \cdot \mathbf{n} \, d\Gamma. \end{aligned}$$

Rewriting equation (3.29) with the above modifications yields the *adjoint problem* in weak form

$$\begin{aligned} \mathcal{L}'_F(T) \tilde{T} &= \int_{\Omega} \alpha \Delta T_a \tilde{T} \, d\Omega + \int_{\Omega_d} (T - T_d) \tilde{T} \, d\Omega - \int_{\Gamma_d} T_a \tilde{T} \, d\Gamma \\ &\quad + \int_{\Gamma_n} (\alpha \nabla T_a \cdot \mathbf{n}) \tilde{T} \, d\Gamma, \end{aligned} \tag{3.30}$$

and in strong form

$$\begin{cases} \alpha \Delta T_a = -(T - T_d) \Theta_{\Omega_d} & \text{on } \Omega, \\ T_a = 0 & \text{on } \Gamma_d, \\ \alpha \nabla T_a \cdot \mathbf{n} = 0 & \text{on } \Gamma_n, \end{cases} \tag{3.31}$$

where Θ_{Ω_d} is addressed to the *Heaviside* function, defined nonzero on the target region Ω_d .

Control Equation In an analogous manner to what was done for the Poisson equation, we differentiate the Lagrangian with respect to the control parameter Q and then integrate by parts:

$$\begin{aligned} \mathcal{L}'_F(Q) \tilde{Q} &= \int_{\Omega} (T - T_d) \tilde{T} \, d\Omega + \lambda \int_{\Omega} Q \tilde{Q} \, d\Omega + \\ &+ \int_{\Omega} \tilde{T} \, \Delta T_a \, d\Omega + \int_{\Omega} \tilde{Q} \, T_a \, d\Omega. \end{aligned} \quad (3.32)$$

Then we can simplify using the adjoint formulation in (3.30), obtaining the expression for the control Q in weak form:

$$\mathcal{L}'_F(Q) \tilde{Q} = \int_{\Omega} (\lambda Q + T_a) \tilde{Q} \, d\Omega, \quad (3.33)$$

and in strong form

$$Q = -\frac{T_a}{\lambda}. \quad (3.34)$$

Starting from equation (3.33), the gradient of the functional can be expressed as:

$$\nabla \mathcal{F}(Q) = \lambda Q + T_a. \quad (3.35)$$

3.3 Dirichlet Boundary Control

In the case of Dirichlet boundary control, the control parameter is the temperature imposed on the wall T_c .

Control Problem System The state system is rewritten as

$$\begin{cases} -\alpha\Delta T = Q & \text{on } \Omega, \\ T = g_d & \text{on } \Gamma_i, \\ T = g_d + T_c & \text{on } \Gamma_c, \\ \alpha\nabla T \cdot \mathbf{n} = g_n & \text{on } \Gamma_n, \end{cases} \quad (3.36)$$

and the functional

$$\mathcal{F}(\mathbf{u}, T, g) = \frac{1}{2} \int_{\Omega_d} (T - T_d)^2 d\Omega + \frac{\lambda}{2} \int_{\Gamma_c} |T_c|^2 d\Gamma. \quad (3.37)$$

The Euler and the Adjoint Problem equation. The formulation of the Euler equation, referring to the functional (3.36) and the state problem (3.37), becomes

$$\begin{aligned} \mathcal{L} = & \frac{1}{2} \int_{\Omega_d} (T - T_d)^2 d\Omega + \frac{\lambda}{2} \int_{\Gamma_c} |T_c|^2 d\Gamma + \int_{\Omega} (\alpha\Delta T) T_a d\Omega - \\ & - \int_{\Gamma_c} (T - g_d - T_c) T_a d\Gamma - \int_{\Gamma_i} (T - g_d) T_a d\Gamma - \\ & - \int_{\Gamma_n} (\alpha\nabla T \cdot \mathbf{n} - g_n) T_a d\Gamma. \end{aligned} \quad (3.38)$$

Taking the derivative with respect to the state variable T , we get

$$\begin{aligned} \mathcal{L}'_F(T) \tilde{T} = & \int_{\Omega} (\alpha\Delta \tilde{T}) T_a d\Omega - \int_{\Omega_d} (T - T_d) \tilde{T} d\Omega - \\ & - \int_{\Gamma_i + \Gamma_c} (T_a) \tilde{T} d\Gamma - \int_{\Gamma_n} (\alpha\nabla T_a \cdot \mathbf{n}) \tilde{T} d\Gamma, \end{aligned} \quad (3.39)$$

and performing the same steps as in the distributed case, the adjoint problem

can be written in weak form as

$$\begin{aligned} \mathcal{L}'_F(T) \tilde{T} &= \int_{\Omega} (\alpha \Delta T_a) \tilde{T} \, d\Omega - \int_{\Omega_d} (T - T_d) \tilde{T} \, d\Omega - \\ &\quad - \int_{\Gamma_i + \Gamma_c} (T_a) \tilde{T} \, d\Gamma - \int_{\Gamma_n} (\alpha \nabla T_a \cdot \mathbf{n}) \tilde{T} \, d\Gamma, \end{aligned} \quad (3.40)$$

and in strong form

$$\begin{cases} \alpha \Delta T_a = (T - T_d) \Theta_{\Omega_d} & \text{on } \Omega, \\ T_a = 0 & \text{on } \Gamma_i, \Gamma_c, \\ \alpha \nabla T_a \cdot \mathbf{n} = 0 & \text{on } \Gamma_n. \end{cases} \quad (3.41)$$

Optimality system. Taking the derivative of the Lagrangian, this time with respect to the control parameter T_c , we get

$$\begin{aligned} \mathcal{L}'_F(T_c) \tilde{T}_c &= \int_{\Omega_d} (T - T_d) \tilde{T} + \lambda \int_{\Gamma_c} T_c \tilde{T}_c \, d\Gamma + \int_{\Omega} \alpha T_a \Delta \tilde{T} \, d\Omega - \\ &\quad - \int_{\Gamma_c} T_a (\tilde{T} - \tilde{T}_c) \, d\Gamma - \int_{\Gamma_i} T_a \tilde{T} \, d\Gamma - \\ &\quad - \int_{\Gamma_n} \alpha (\nabla \tilde{T} \cdot \mathbf{n}) T_a \, d\Gamma, \end{aligned} \quad (3.42)$$

where

$$\tilde{T} = T (T_c + \tilde{T}_c) - T (T_c). \quad (3.43)$$

To isolate the variation \tilde{T} , we integrate twice by parts

$$\begin{aligned} \int_{\Omega} \alpha T_a \Delta \tilde{T} \, d\Omega &= \int_{\Gamma} \alpha T_a (\nabla \tilde{T} \cdot \mathbf{n}) \, d\Gamma - \int_{\Omega} \alpha \nabla T_a \cdot \nabla \tilde{T} \, d\Omega \\ &= \int_{\Gamma} \alpha T_a (\nabla \tilde{T} \cdot \mathbf{n}) \, d\Gamma - \int_{\Gamma} \alpha \tilde{T} (\nabla T_a \cdot \mathbf{n}) \, d\Gamma + \\ &\quad + \int_{\Omega} \alpha \tilde{T} \Delta T_a \, d\Omega, \end{aligned} \quad (3.44)$$

and using the boundary conditions imposed on T and T_a , we obtain

$$\int_{\Omega} \alpha T_a \Delta \tilde{T} \, d\Omega = - \int_{\Gamma_c} \alpha \tilde{T}_c \nabla T_a \cdot \mathbf{n} \, d\Gamma + \int_{\Omega} \alpha \tilde{T} \Delta T_a \, d\Omega. \quad (3.45)$$

Similarly

$$\begin{aligned} - \int_{\Gamma_n} \alpha (\nabla \tilde{T} \cdot \mathbf{n}) T_a \, d\Gamma &= - \int_{\partial\Gamma_n} \alpha \tilde{T} T_a \, d\partial\Gamma + \int_{\Gamma_n} \alpha \tilde{T} (\nabla T_a \cdot \mathbf{n}) \, d\Gamma \\ &= \int_{\Gamma_n} \alpha \tilde{T} (\nabla T_a \cdot \mathbf{n}) \, d\Gamma. \end{aligned} \quad (3.46)$$

Thus, the equation (3.42) becomes

$$\begin{aligned} \mathcal{L}'_F(T_c) \tilde{T}_c &= \int_{\Omega_d} (T - T_d) \tilde{T} + \lambda \int_{\Gamma_c} T_c \tilde{T}_c \, d\Gamma - \int_{\Gamma_c} \alpha \tilde{T}_c (\nabla T_a \cdot \mathbf{n}) \, d\Gamma + \\ &+ \int_{\Omega} \alpha \tilde{T} \Delta T_a \, d\Omega - \int_{\Gamma_c} T_a (\tilde{T} - \tilde{T}_c) \, d\Gamma - \int_{\Gamma_i} T_a \tilde{T} \, d\Gamma + \\ &+ \int_{\Gamma_n} \alpha \tilde{T} (\nabla T_a \cdot \mathbf{n}) \, d\Gamma, \end{aligned} \quad (3.47)$$

which can be further simplified using the adjoint system (3.40) to obtain

$$\mathcal{L}'_F(T_c) \tilde{T}_c = \lambda \int_{\Gamma_c} T_c \tilde{T}_c \, d\Gamma - \int_{\Gamma_c} \alpha \tilde{T}_c (\nabla T_a \cdot \mathbf{n}) \, d\Gamma + \int_{\Gamma_c} T_a \tilde{T}_c \, d\Gamma. \quad (3.48)$$

Finally, using the T_a boundary conditions

$$\int_{\Gamma_c} T_a \tilde{T}_c \, d\Gamma = 0, \quad (3.49)$$

the control equation can be rewritten in weak form as

$$\mathcal{L}'_F(T_c) \tilde{T}_c = \lambda \int_{\Gamma_c} \tilde{T}_c T_c \, d\Gamma - \int_{\Gamma_c} \alpha \tilde{T}_c \nabla T_a \cdot \mathbf{n} \, d\Gamma, \quad (3.50)$$

and in strong form as

$$T_c - \alpha \frac{\nabla T_a \cdot \mathbf{n}|_{\Gamma_c}}{\lambda} = 0. \quad (3.51)$$

Gradient of the Functional. Starting from the equation (3.50), the gradient of the functional can be expressed as

$$\nabla \mathcal{F}(T_c) = \lambda T_c - \alpha \nabla T_a \cdot \mathbf{n}|_{\Gamma_c}. \quad (3.52)$$

3.4 Neumann Boundary Control

In Neumann boundary control, the control parameter is the thermal flux h imposed on the boundary.

Control Problem System. The state system is rewritten as

$$\begin{cases} -\alpha \Delta T = Q & \text{on } \Omega, \\ T = g_d & \text{on } \Gamma_d, \\ \alpha \nabla T \cdot \mathbf{n} = g_n & \text{on } \Gamma_i, \\ \alpha \nabla T \cdot \mathbf{n} = H & \text{on } \Gamma_c, \end{cases} \quad (3.53)$$

and the functional

$$\mathcal{F}(\mathbf{u}, T, g) = \frac{1}{2} \int_{\Omega_d} (T - T_d)^2 d\Omega + \frac{\lambda}{2} \int_{\Gamma_c} H^2 d\Gamma. \quad (3.54)$$

Euler and adjoint Equations. The Lagrangian is reformulated for the state system (3.58) and for the functional (3.54)

$$\begin{aligned} \mathcal{L} = & \frac{1}{2} \int_{\Omega_d} (T - T_d)^2 d\Omega + \frac{\lambda}{2} \int_{\Gamma_c} H^2 d\Gamma + \int_{\Omega} \alpha \Delta T T_a d\Omega - \\ & - \int_{\Gamma_d} (T - g_d) T_a d\Gamma - \int_{\Gamma_i} (\alpha \nabla T \cdot \mathbf{n} - g_n) T_a d\Gamma - \\ & - \int_{\Gamma_c} (\alpha \nabla T \cdot \mathbf{n} - H) T_a d\Gamma. \end{aligned} \quad (3.55)$$

Again, by differentiating with respect to the state variable T

$$\begin{aligned} \mathcal{L}'_F(T) \tilde{T} = & \int_{\Omega_d} (T - T_d) \tilde{T} d\Omega + \int_{\Omega} \alpha \Delta \tilde{T} T_a d\Omega - \\ & - \int_{\Gamma_d} \tilde{T} T_a d\Gamma - \int_{\Gamma_i + \Gamma_c} (\alpha \nabla \tilde{T} \cdot \mathbf{n}) T_a d\Gamma, \end{aligned} \quad (3.56)$$

and performing the same substitutions as in the distributed case, the adjoint problem for Neumann control is obtained in weak form

$$\begin{aligned} \mathcal{L}'_F(T) \tilde{T} = & \int_{\Omega_d} (T - T_d) \tilde{T} d\Omega + \int_{\Omega} (\alpha \Delta T_a) \tilde{T} d\Omega - \\ & - \int_{\Gamma_d} (T_a) \tilde{T} d\Gamma - \int_{\Gamma_i + \Gamma_c} (\alpha \nabla T_a \cdot \mathbf{n}) \tilde{T} d\Gamma, \end{aligned} \quad (3.57)$$

and in strong form

$$\begin{cases} \alpha \Delta T_a = (T - T_d) \Theta_{\Omega_d} & \text{on } \Omega, \\ T_a = 0 & \text{on } \Gamma_d, \\ \alpha \nabla T_a \cdot \mathbf{n} = 0 & \text{on } \Gamma_i, \Gamma_c. \end{cases} \quad (3.58)$$

Control Equation. By differentiating the Lagrangian with respect to the control parameter H one obtains

$$\begin{aligned} \mathcal{L}'_F(H) \tilde{H} &= \int_{\Omega_d} (T - T_d) \tilde{T} + \lambda \int_{\Gamma_c} H \tilde{H} \, d\Gamma + \int_{\Omega} \alpha T_a \Delta \tilde{T} \, d\Omega - \\ &\quad - \int_{\Gamma_d} T_a \tilde{T} \, d\Gamma - \int_{\Gamma_i} \alpha \nabla \tilde{T} \cdot \mathbf{n} T_a \, d\Gamma - \\ &\quad - \int_{\Gamma_c} \left(\alpha \nabla \tilde{T} \cdot \mathbf{n} - \tilde{H} \right) T_a \, d\Gamma, \end{aligned} \quad (3.59)$$

where $\tilde{T} = T(H + \tilde{H}) - T(H)$. In a manner analogous to the previous cases, the equation can be simplified to obtain the control equation in weak form

$$\mathcal{L}'_F(H) \tilde{H} = \lambda \int_{\Gamma_c} H \tilde{H} \, d\Gamma + \int_{\Gamma_c} T_a \tilde{H} \, d\Gamma, \quad (3.60)$$

and in strong form

$$H = -\frac{T_a}{\lambda}. \quad (3.61)$$

Starting from the equation (3.60), the gradient of the functional can be expressed as

$$\nabla \mathcal{F}(H) = \lambda H + T_a. \quad (3.62)$$

Chapter 4

Coupling

This section details the approach used to couple the two codes, FEMuS and OpenFOAM, which are the focus of the subsequent numerical results.

FEMuS [3] is an in-house multigrid finite element library developed in C++ that uses a variety of open-source libraries, including PETSc for linear algebra and LibMesh for mesh hierarchy management. The FEMuS library has been extended to support coupling with a MED-compatible C++ interface, utilizing the SALOME platform for enhanced interoperability.

OpenFOAM [25] is a widely recognized open-source, object-oriented C++ library primarily developed for computational fluid dynamics (CFD) simulations. It is maintained separately by the ESI Group and the OpenFOAM Foundation. The OpenFOAM library is based on the finite volume method (FVM), which discretizes the computational domain into elements, or cells, where PDEs are solved.

Next the coupling structure and an explanation of the coupling algorithm are reported, for further details see [4].

4.1 Coupling structure

To enable communication between the two codes, and more generally between any codes, it is necessary to create interfaces for each code capable of translating internal data structures into a common format, i.e., the MED format.

The MED and MEDCOUPLING libraries are structured to be optimized for code coupling, meaning that the data structures in which the fields are stored contain enough data to be correctly interpolated. Additionally, they do not require external programs to be used, simplifying their use, and they adopt all high-performance computing (HPC) paradigms by optimizing resource use and managing data exchange directly in memory without the need for external file reading and writing.

These interfaces communicate through a central *hub*, forming a *hub-and-spoke* model illustrated in Figure 4.1. This approach facilitates the addition of further codes since, for each new code, only its interface needs to be created to transfer data in the MED format, greatly reducing the development resources required compared to an approach where each code is coupled separately.

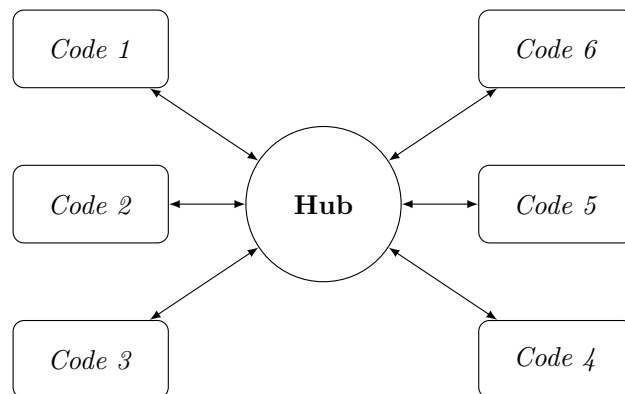


Figure 4.1: Hub-and-Spoke structure representation.

The coupling application thus uses three different classes for data transfer. The first, *OF_interface*, serves to translate OpenFOAM's internal data structures into the common MED format. Similarly, the second class, *F_interface*, acts as an interface between FEMuS and the MED library. Finally, the third class, *MED_class*, is responsible for managing operations within the MED library itself, such as storing, retrieving, and manipulating data. A particularly interesting feature of this class is its ability to interpolate a field on a different mesh. The interpolation of a field, passing from a source field ϕ_s defined on a source mesh Ω_s into a target field ϕ_t defined on a target mesh Ω_t , is a fundamental action when coupling two codes that use different resolution techniques, such as finite elements and finite volumes, or that simply use different meshes.

At a higher level, the interaction between the codes and their operation is managed. First, the two codes are initialized and configured, ensuring they are ready to interact through the interface structures with an exchange mesh and initialization numerical fields. Second, the synchronization of time steps between the two codes is managed, ensuring that the two simulations remain consistent with each other. At each time step, fields are exchanged, the problem is solved, and convergence is monitored.

Through this platform, simulations that require exchanging volumetric data, defined over the entire mesh, or boundary data can be performed. In this discussion, both alternatives will be explored by solving a coupled distributed control problem for volumetric data exchange and boundary control problems for boundary data exchange. In both cases, the general structure of the code remains the same.

4.2 Coupling algorithm

The object of this coupling is the solution of an optimal control system for the heat transport equation. In particular, the system has been split into the state system, solved by OpenFOAM, and the combined adjoint and control system, solved by FEMuS. A graphical representation is reported in figure 4.2.

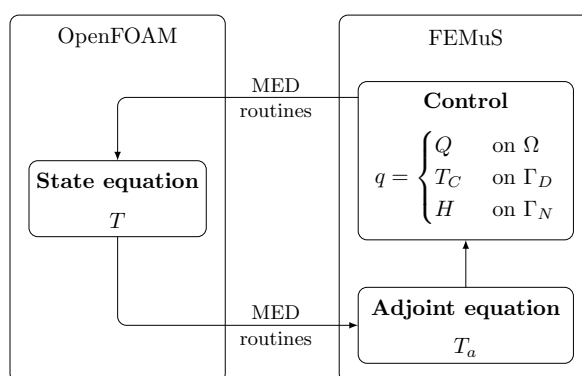


Figure 4.2: Coupling diagram.

As shown in the Algorithm 4.1, in the first step, both OpenFOAM and FEMuS generate a copy of the mesh in MED format, where the field to be passed to the other code will be set for interpolation.

Both interface classes, *OF_interface* and *F_interface*, have the function `init_interface()`, which extracts information such as connectivity and coordinates from the original mesh of the two codes to create the copy in the MED format. FEMuS, employing the Finite Element Method, handles bi-quadratic fields and therefore a bi-quadratic mesh. Nonetheless to exchange data with OpenFOAM which uses a linear mesh the FEMuS interface needs a linear mesh as well. For this reason, the function `init_interface()` from the *F_interface*, holding the information from the original bi-quadratic mesh extracts the necessary data needed to create a linear mesh for the FEMuS

Algorithm 4.1 Coupling Algorithm

```

1: procedure main()
2:   Initialization of OpenFOAM and FEMuS structures.

```

Initialization of interfaces both for FEMuS and OpenFOAM.

```

3:   function init_interface()
4:     Set interface name for reference at the supervisor level.
5:     conn ← get_mesh_connectivity()      ▷ Get interface mesh connectivity
6:     coords ← get_mesh_coordinates()    ▷ Get coordinates of mesh nodes
7:     set_map_CodeFromToMED()           ▷ Map mesh nodes ↔ MED mesh nodes
8:   end function
9:   function create_mesh()
10:    insert cells with conn information into the MED mesh structure.
11:    setup coords information into the MED mesh structure.
12:    creation of MED mesh copy from the mesh of FEMuS/OpenFOAM.
13:  end function
14:  function init_med_field_on_nodes/cells()
15:    assigns the MED field to the corresponding interface MED mesh.
16:    allocate_med_array()                ▷ MED array memory allocation
17:    init_med_field()                    ▷ Set MED field values to zero
18:  end function

```

Time loop

```

19:  it = 0
20:  while non stop do
21:    Solve system of equations (T) of OpenFOAM.
22:    get_field_from_OF()                 ▷ Extract field T solution from OpenFOAM
23:    fill_med_array()                   ▷ Write field T solution into MED array
24:    update_med_field()                 ▷ Set MED array values into MED field T
25:    interpolation()                    ▷ Interpolation  $P_0 \rightarrow P_0$  (from OF to F)
26:    set_field_to_F()                  ▷ Set field T solution into FEMuS
27:    Evaluate Functional.
28:    if convergence then
29:      stop
30:    end if
31:    Solve system of equations ( $T_a, q$ ) of FEMuS.
32:    get_field_from_F()                 ▷ Extract field q solution from FEMuS
33:    fill_med_array()                   ▷ Write field q solution into MED array
34:    update_med_field()                 ▷ Set MED array values into MED field q
35:    interpolation()                    ▷ Interpolation  $P_0 \rightarrow P_0$  (from F to OF)
36:    set_field_to_OF()                 ▷ Set field q solution into OpenFOAM
37:    it += 1
38:  end while
39: end procedure

```

coupling interface.

At this point, once the necessary data are stored, is possible to generate copies of the two original meshes through the function `create_mesh()` that belongs to the *med_class*.

Now both codes have a copy of their mesh in the MED format, so the field that needs to be exchanged (T for OpenFOAM and q for FEMuS) is initialized on these meshes. To do that are used the functions `init_med_field_on_cells()`, for OpenFOAM, and `init_med_field_on_nodes()`, for FEMuS: both functions belong to the *med_class*. Here an array is generated for each field, allocating the memory for it and enabling the MED library to set the values of the MED fields. Now both codes have completed the interface initialization, configuring MED mesh copies and MED fields, so it is possible to proceed with the time loop.

The time loop begins with solving the state equation for T on OpenFOAM, once the code has completed the calculation and has obtained a solution this is extracted from the solver using the function `get_field_from_OF()` belonging to the *OF_interface* class. Then the solution is translated into a MED array and transferred into the corresponding MED field, respectively through the functions `fill_med_array()` and `update_med_field()` of the *med_class* class.

At this point, the field from OpenFOAM needs to be interpolated from its MED mesh into the FEMuS MED mesh. The performed interpolation is from P0 to P0, since FEMuS solves for biquadratic fields, there is a second interpolation from P0 to P2 to set the field on FEMuS. The algorithm used for this last interpolation is implemented in FEMuS [14]. So the field is firstly interpolated from the P0 MED mesh of OpenFOAM into the P0 MED mesh of FEMuS using the function `interpolation()` of the *med_class* and in a

second step into the biquadratic FEMuS field using an in-house interpolation. At this point, the solution can be set into FEMuS as its own solution using the `set_field_to_F()` of *F_interface*.

Once FEMuS has the T solution performed by OpenFOAM the system functional is evaluated and the convergence condition is checked.

Now is possible to solve the FEMuS equation system, formed by the adjoint temperature T_a , which uses the temperature T obtained from OpenFOAM, and the control q . Once the calculations are performed is possible to follow the same scheme reported for passing the data from OpenFOAM to FEMuS backwards. That means using the function `get_field_from_F()` to extract the solution from FEMuS, then interpolate it from P2 to P0. At this step, the P0 solution is used to fill a MED array through the function `fill_med_array()` and then transferred into the corresponding MED field using `update_med_field()`.

Now the `interpolation()` function is used again in order to interpolate the MED field of FEMuS onto the OpenFOAM MED mesh and finally set the interpolated field on OpenFOAM using `set_field_to_OF()`. At this point is possible to re-iterate the procedure for each iteration of the time loop.

4.2.1 Interpolation algorithm

One of the key aspects of coupling is the interpolation of fields between the grids of the different codes. In addition to being one of the most challenging and delicate aspects of coupling codes, it is also necessary to study its impact on the solutions and the errors it generates. As the coupling routine has been defined, it is clear that there are two interpolations: the first from the OpenFOAM cell-wise field to the FEMuS point-wise biquadratic field and the second the opposite.

Point-wise to cell-wise. How this interpolation is done depends on whether it is between volume or boundary fields.

If it is between volume fields, the average value of each element of the point-wise field is assigned to each cell of the cell-wise field. This average value is calculated as the integral of the field above the element divided by its area (or volume if the mesh is three-dimensional)

$$f_h^e = \frac{\int_{\Omega_e} f_h d\mathbf{x}}{\int_{\Omega_e} d\mathbf{x}}. \quad (4.1)$$

Expressing the field f_h as the linear combination of the values at the nodes weighted by the shape function we can write its integral above the element as

$$\int_{\Omega_e} f_h d\mathbf{x} = \sum_{j=1}^{n_e} f(\mathbf{x}_j^e) \int_{\Omega_e} N_j^{2,e} d\mathbf{x}. \quad (4.2)$$

An example of interpolation is reported in Figure 4.3.

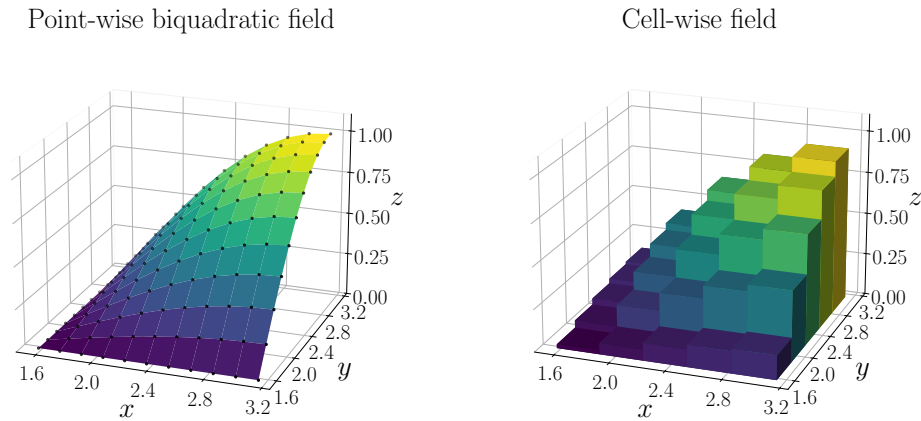


Figure 4.3: Interpolation from point-wise to cell-wise field.

When the interpolated field is a boundary field, we assign the face central node value of the point-wise field, see Figure 4.4.

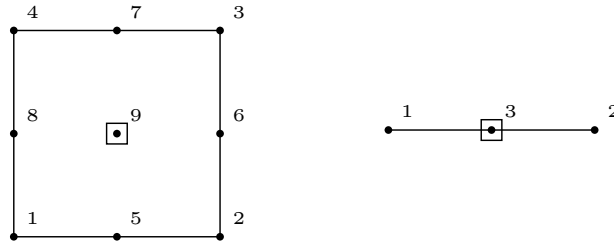


Figure 4.4: Central point of a two-dimensional (left) or one-dimensional (right) boundary face.

Cell-wise to point-wise. In this case, the procedure is similar to that in the opposite case. Still, instead of assigning the mean node value to each cell, each node is given the mean value among the near cells values. If the node is a corner the mean value is taken between all the four cells that share the corner, otherwise only between the two confining cells. Instead, when a node is a central one then its proper cell value is conserved. An example is reported in Figure 4.5.

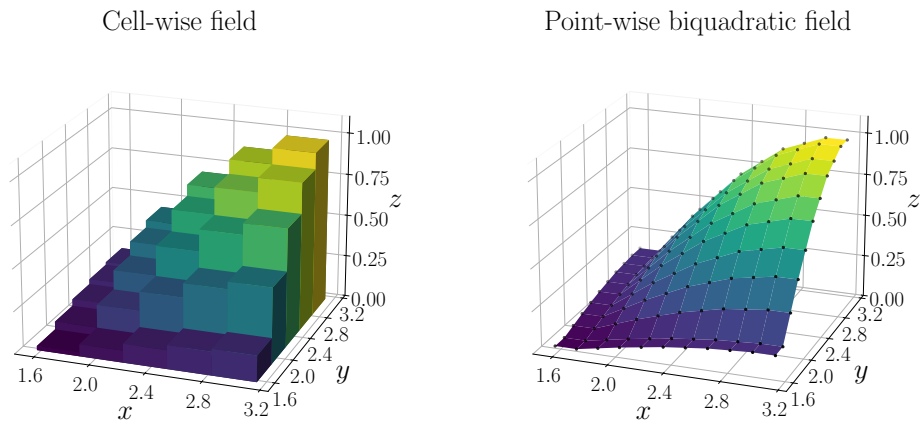


Figure 4.5: Interpolation from cell-wise to point-wise field.

4.2.2 Interpolation errors

In the coupling routine, the interpolation errors occur two times, one for each interpolation process. The first is on the temperature field received by FEMuS interpolation from the OpenFOAM grid to its own. It is important to remember that this means interpolating a P0 field into a P2 field. Defining an *interpolation function* $\mathcal{I}(T) : T_{OF} \rightarrow T_F$ it is possible to rewrite the functional \mathcal{F} as

$$\mathcal{F}(\mathcal{I}(T), q) = \frac{1}{2} \int_{\Omega_d} |\mathcal{I}(T) - T_d|^2 d\Omega + \frac{1}{2} \int_{\Omega_c} |q|^2 d\Omega_c, \quad (4.3)$$

and the adjoint variable T_a as

$$\Delta \tilde{T}_a = \mathcal{I}(T) - T_d. \quad (4.4)$$

After that, the control q will also be affected by the error in T_a

$$\tilde{q} = q(\tilde{T}_a). \quad (4.5)$$

This affects the functional computation since the control error will accumulate iteratively.

Moreover, we should also consider the error resulting from the control interpolation q from the FEMuS grid to the OpenFOAM's one. Now, we use a biquadratic to piece-wise element field approximation. In Chapter 5 some considerations about the relative importance of the two interpolation processes and their dependence on the computational grid are made.

Chapter 5

Numerical results

This chapter presents the numerical results for an optimal control application solved by coupling FEMuS and OpenFOAM as described in Chapter 4. For each case solved, the result obtained is compared with the result obtained without coupling on FEMuS, which serves as the reference result.

The steady-state heat transport equation without convective contribution, which consists of a Laplacian with a volumetric heat source on the *right-hand side*, gives the state of the system,

$$-\alpha\Delta T = Q. \quad (5.1)$$

The state described above is solved on a two-dimensional square domain Ω shown in Figure 5.1 with boundary Γ . On the partition Γ_d , a Dirichlet condition is imposed, and on Γ_n , a Neumann condition is imposed

$$\begin{cases} -\alpha\Delta T = Q & \text{in } \Omega, \\ T = g_t & \text{on } \Gamma_d, \\ \alpha\nabla T \cdot \mathbf{n} = g_n & \text{on } \Gamma_n. \end{cases} \quad (5.2)$$

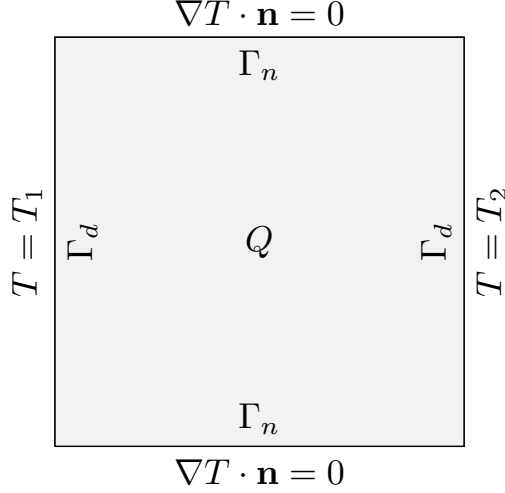


Figure 5.1: State of the problem.

Concerning the value of Q and the boundary conditions, in the following sections, these parameters are described considering the different types of optimal control. Nevertheless, the boundary conditions g_d, g_n are considered to be constant on Γ_D, Γ_N respectively. Note that, considering a boundary-type control the involved boundary conditions are still initialized with g_d or g_n . So at every step, we impose at the boundary g_d (or g_n) plus T_c (or H), where the control variable is initialized to zero.

Anyway, the objective functional which relates the state variable T with the desired temperature T_d remains the same for each type of control. In particular, we have

$$\mathcal{F}(T) = \frac{1}{2} \int_{\Omega_d} |T - T_d|^2 d\Omega, \quad (5.3)$$

where Ω_d is the target region with target T_d .

Regarding the controlled regions, for each simulation, the regularization term is assumed to be solved only on $\Omega_c \subseteq \Omega$. Therefore, the considered

functional is

$$\mathcal{F}(T, q) = \frac{1}{2} \int_{\Omega_d} |T - T_d|^2 d\Omega + \frac{\lambda}{2} \int_{\Omega_c} |q|^2 d\Omega. \quad (5.4)$$

We recall that in the case of boundary control, the second integral in the right-hand side of Equation (5.4) has meaning naturally only on the controlled portion of the domain which is a portion of Γ , thus on the Dirichlet boundaries or on the Neumann ones.

For each type of control, the results obtained with the coupled and uncoupled algorithms have been compared. In particular, we have considered the same λ , ρ , and the same criterion for the convergence solution. Specifically, considering the average functional of ten iterations $\bar{\mathcal{F}}_i$ at the iteration i , the convergence has been obtained following

$$\frac{|\bar{\mathcal{F}}_i - \bar{\mathcal{F}}_{i-1}|}{\bar{\mathcal{F}}_i} < \varepsilon, \quad (5.5)$$

where ε has been set equal to 10^{-9} .

Non-dimensional system In the following, a non-dimensional formulation for the optimal control system is proposed. By using this formulation is possible to compare results from similar problems but with different dimensions or physical constants. Thus, the state variable T , the adjoint variable T_a and the control q have been reported in the dimensionless formulation

$$T^* = \frac{T - g_d}{T_{ref} - g_d}, \quad T_a^* = \frac{T_a \alpha}{T_{ref} L^2}, \quad q^* = \begin{cases} Q^* = \frac{QL^2}{\alpha T_{ref}}, \\ T_c^* = \frac{T_c}{T_{ref}}, \\ H^* = \frac{HL}{\alpha T_{ref}}, \end{cases} \quad (5.6)$$

where $T_{ref} = \max(|T_d|)$. Similarly, the space coordinates \mathbf{x} are transformed into $\mathbf{x}^* = \mathbf{x}/L$, with L the length of the squared domain side. In order for the optimal control system to remain coherent during the transition to these non-dimensional variables, also the regularization factor λ must be considered. In fact, λ has a specific dimensionality that depends on the type of control, and it turns into its dimensionless form too. Specifically, the following dimensionless transformations apply to the three types of control

$$\lambda_{dist}^* = \frac{\lambda \alpha^2}{L^4}, \quad \lambda_{Dir}^* = \frac{\lambda}{L}, \quad \lambda_{Neu}^* = \frac{\lambda \alpha^2}{L^3}. \quad (5.7)$$

In the following results, for the sake of simplicity, the parameter λ will always be presented with its dimensional value, i.e., the value used in the simulations. However, its dimensionless value can be derived using formulas (5.7), with $L = 0.01 [m]$ and $\alpha = 1.433 \cdot 10^{-7} \left[\frac{m^2}{s} \right]$.

As said above, this approach allows for scaling the problem to different materials or geometries. A brief demonstration of this concept is reported below, within a distributed control case, *case 1*.

5.1 Uncoupled results

A large part of the work has been dedicated to the implementation of the control system, along with the minimization techniques, in the finite element code FEMuS. For this reason, some introductory cases are presented to establish the proper functioning of the code and its functionalities. Two cases are presented, both with a 80×80 resolution grid is used. In addition, also a comparison between the two implemented minimization techniques, the steepest and the conjugate gradient methods, has been added.

5.1.1 Case 1

A first example is presented as a distributed control case, where the optimal control system is defined in Section 3.2, with a single target zone Ω_d with target T_d , and the controllable zone Ω_c coincide with the entire domain Ω . Defined Ω as a square $\Omega = [0, L] \times [0, L]$, then $\Omega_d = [L/5, 4L/5] \times [2L/5, 3L/5]$. The target zone Ω_d is reported in Figure 5.2. The parameter λ is taken equal to 10^{-2} .

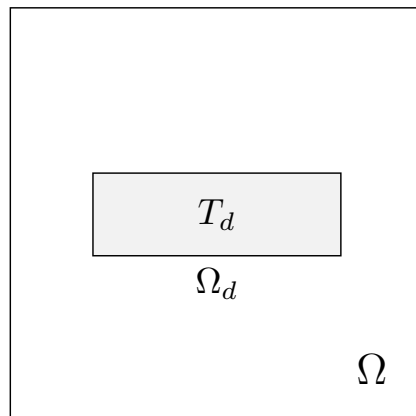


Figure 5.2: Single target region for the uncoupled problem.

In Figure 5.3 a surface plot of the non-dimensional temperature and the

non-dimensional control is reported. We can notice how the temperature

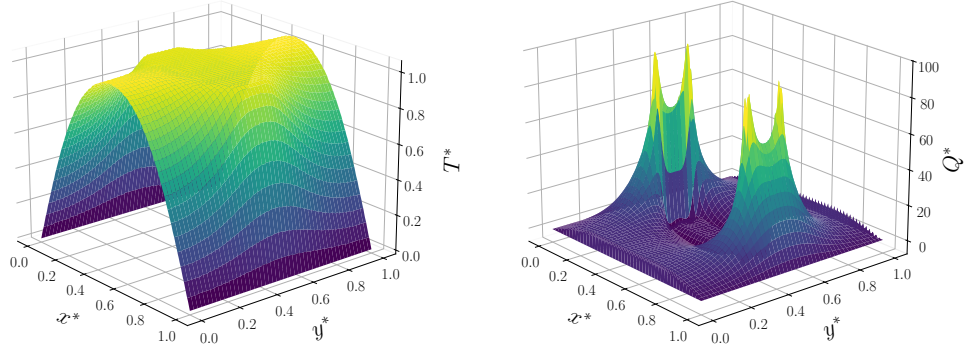


Figure 5.3: Three-dimensional representation of T^* (on the left) and Q^* (on the right) for the distributed control, considering the two-dimensional non-dimensional domain.

reaches very well the target value in the target region, assuming a flat profile. This is more noticeable in Figure 5.4, which shows the temperature plotted across the target zone as a function of x^* . The control Q is flat inside the target region since the temperature has to remain constant, on the contrary, it has a very sharp behavior in correspondence with the edges of the region near the left and right walls. In these walls we have a Dirichlet boundary condition, therefore the temperature gradient is maximum.

To have a better look at the quality of the results, i.e. how much we are near the real optimum, it is possible to use the Euler equation (3.34). If the two terms of the equation Q and $-\frac{T_a}{\lambda}$ are identical, then the optimum is reached. In Figure 5.4 are reported the two variables as a function of the non-dimensional coordinate x^* for $y^* = 0.5$. As we can see the two profiles are almost the same. This means that the optimum is near the real one.

Problem scaling. Starting from the problem defined above, with $\lambda = 10^{-2} [s^2]$, we scale it to a ten times bigger domain, i.e. considering $L_s =$

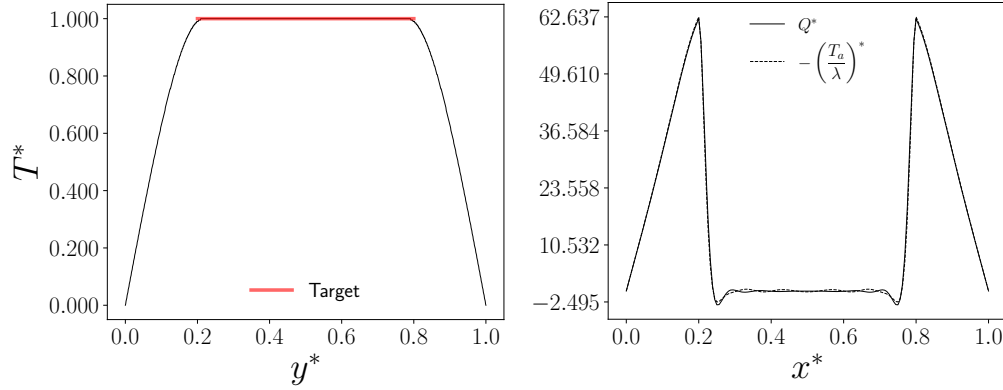


Figure 5.4: The non-dimensional temperature T^* (on the left), and the control Q^* compared with its analytic optimum expression (on the right), as a function of x^* , for $y^* = 0.5$.

0.1 [m]. To solve the problem on the new domain with the same *effective* regularization is sufficient to calculate the new dimensional λ as

$$\lambda_s = \lambda^* \frac{L_s^4}{\alpha^2} = 10^2 \text{ [s}^2\text{]}. \quad (5.8)$$

In Figure 5.5 we compare two cases, the first with λ , L and α and the second with λ_s , L_s and $\alpha_s = \alpha$. The state variable T and the control variable Q are reported across the square domain's diagonal. By keeping λ^* constant between the two cases, we can observe that the profile of the state variable T remains the same. Recall that its dimensionless form, T^* , depends solely on the target value. This consideration proves the consistency of the proposed non-dimensional system, which, as demonstrated, allows the optimal control problem to be scaled to others, considering geometric similar domains and other materials.

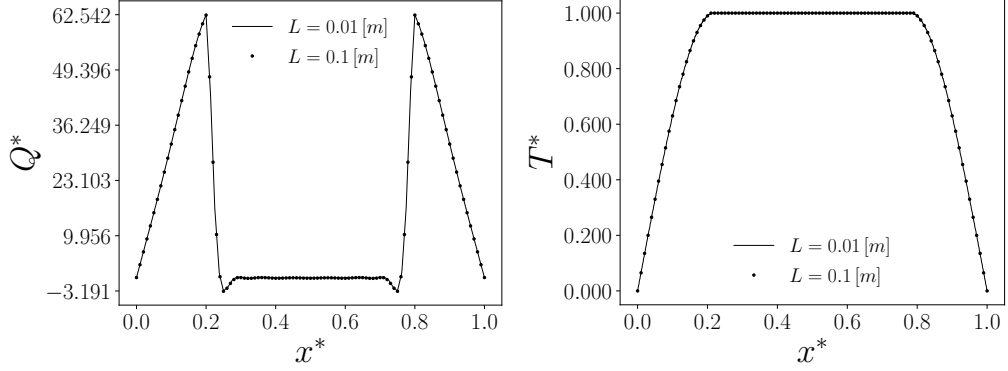


Figure 5.5: Dimensionless control Q^* (left) and temperature T^* (right) with the same λ^* but different domain, as a function of the non-dimensional diagonal r^* .

5.1.2 Case 2

In the code has been also implemented the possibility to have different target regions, each with different targets, and to restrict the controllable region Ω_c to a subset of the domain Ω . Both these requests increase the complexity of the system. Next, a Dirichlet boundary control case with two target regions and a restriction of the controllable region is reported. For a boundary control case, the target region is called Γ_c .

Defined Ω as a square $\Omega = [0, L] \times [0, L]$, the controllable region Γ_c is equal to half of the left wall, $\Gamma_c = [0, L/2]$. Instead, the target zone Ω_d is divided into two subzones $\Omega_{d,1}$ and $\Omega_{d,2}$, with target temperatures $T_{d,1}$ and $T_{d,2}$, respectively. Specifically, $\Omega_{d,1} = [L/5, 2L/5] \times [L/5, 2L/5]$ and $\Omega_{d,2} = [3L/5, 4L/5] \times [3L/5, 4L/5]$. A representation is reported in Figure 5.6. The parameter λ is taken equal to 10^{-8} . In Figure 5.7, a surface plot of the non-dimensional temperature is reported. Observing the surface plot on the left, it is evident that the temperature imposed at the wall is significantly higher than its value across the target region. Additionally, the right plot clearly shows that the efficiency of the control is much lower compared to the

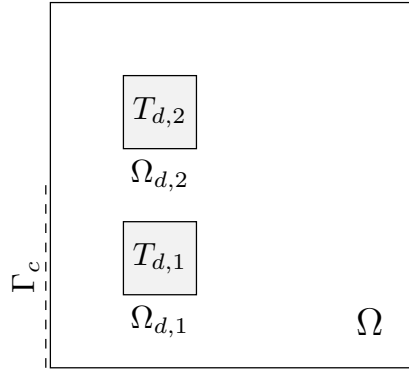


Figure 5.6: Double target region for the uncoupled problem.

previous case. As mentioned earlier, this reduction in efficiency is due to the increased complexity introduced by the restriction of the controllable region and the presence of multiple targets.

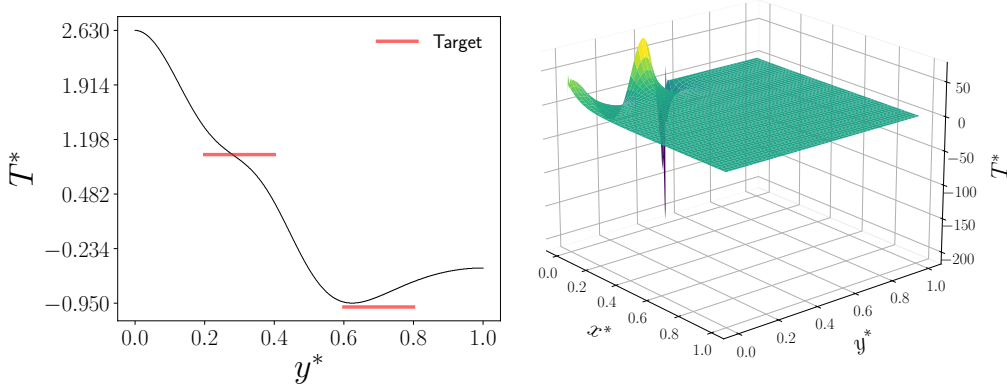


Figure 5.7: The state variable T^* as a function of y^* for $x^* = 0.3$, in the middle of the target region (on the left), and across the two-dimensional domain (on the right), for the Dirichlet boundary control.

In Figure 5.8, the obtained control profile is shown alongside a comparison with its analytical optimal expression (3.51). On the left, we observe a spike in the control profile near the boundary of the uncontrolled region. This is because the target zone $\Omega_{d,2}$, located farther from the controlled boundary Γ_c , is less effectively controlled, with the adjoint variable reaching very high

values in this region. The profile of the adjoint temperature T_a is reported in Figure 5.9. Despite the state does not perfectly match the target values, it remains close to the optimal state, as illustrated on the right side of Figure 5.8.

This example contrasts with the previous case, where the target values were met more accurately due to the problem's simpler nature. Nevertheless, both optimal control problems achieve their respective optimal solutions. It is important to note that the obtained control, T_c , follows the Euler equation only within the controllable domain Γ_c , as it is zero outside this region.

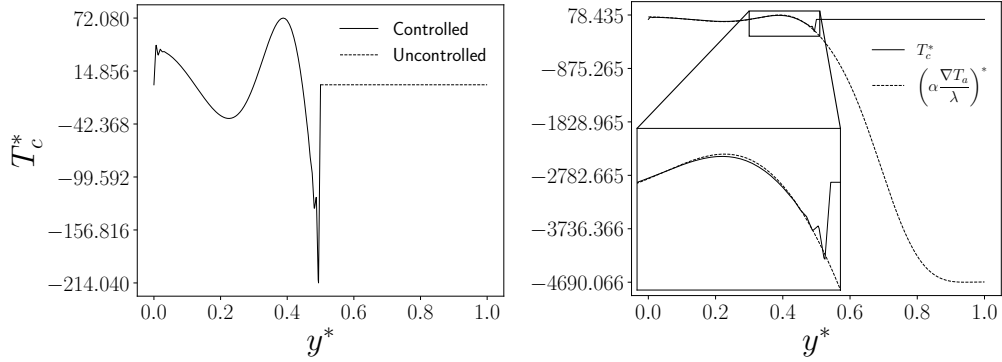


Figure 5.8: Control T_c (on the left) and its comparison with the analytic optimum expression (on the right) as a function of y^* , for $x^* = 0$.

5.1.3 Minimization methods comparison

Recalling the first case reported above, it has been performed both with the conjugate gradient (CG) and the steepest gradient (SG) methods, with various grid configurations. Next, a brief comparison of the performance of the two methods is reported.

Figure 5.10 shows the convergence ratio, defined as the ratio between the number of iterations of the CG method and the SG method $\frac{It_{CG}}{It_{SG}}$, as

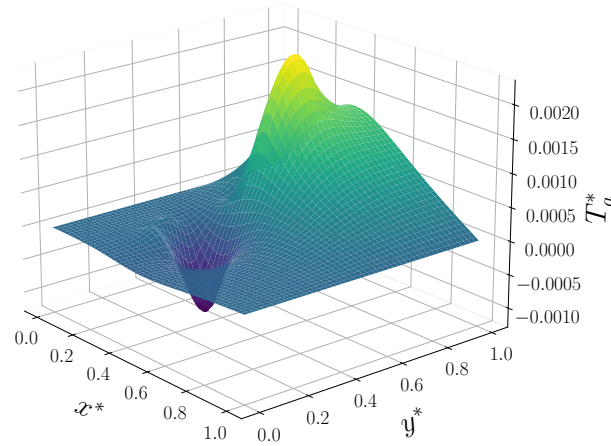


Figure 5.9: Adjoint temperature T_a across the bidimensional domain, for Dirichlet boundary control.

a function of the final value of the functional and the mesh grid used. In this case, threshold convergence was used instead of equilibrium convergence. Results are reported for various functional thresholds and grid configurations.

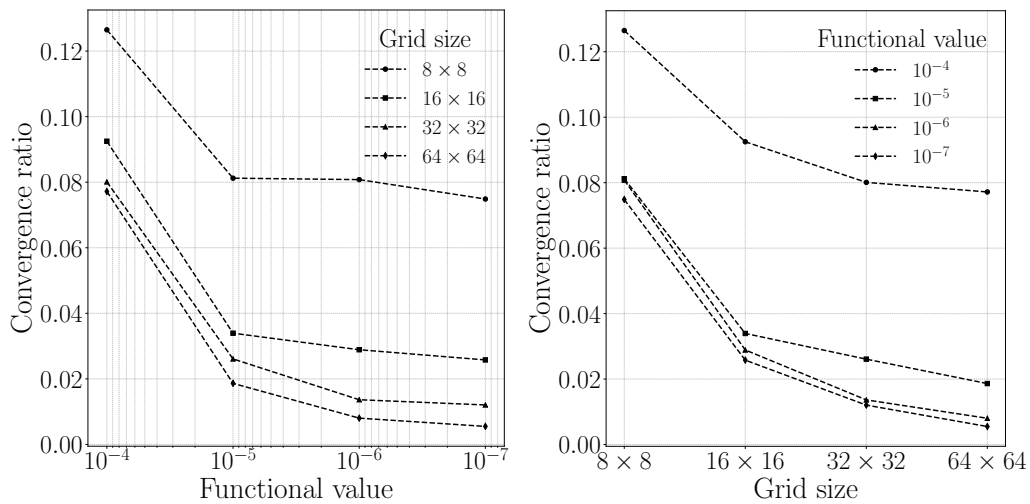


Figure 5.10: Convergence ratio between CG and SG methods, as a function of the functional value threshold and the mesh grid.

From Figure 5.10 we can notice how the convergence ratio decreases both

with the increase of the degrees of freedom of the problem and the decrease of the threshold functional value. In particular, from the graph on the left, we see the convergence ratio as a function of the functional threshold value while on the right graph as a function of the grid size. On the left, we can notice how the convergence ratio reaches a kind of *plateau* for finer grids. On the contrary, it continues to decrease with the increase of the degrees of freedom of the grid, as we can see on the right.

Taking the lowest functional value, 10^{-7} , and the finest grid, 64×64 , as examples, the advantage of using the CG method over the SG method is approximately two orders of magnitude. This demonstrates the significant impact that the choice of minimization technique can have in an optimal control application, or more generally, in optimization.

5.2 Coupled results

In this section, the coupled results are presented and validated using the uncoupled code as a benchmark.

In the following examples, the controllable zone Ω_c is considered to be the all domain Ω , or in the case of boundary control all Γ_d and Γ_n respectively. The target zone Ω_d is divided into two subzones $\Omega_{d,1}$ and $\Omega_{d,2}$, with target temperatures $T_{d,1}$ and $T_{d,2}$, respectively. Defined Ω as a square $\Omega = [0, L] \times [0, L]$, then $\Omega_{d1} = [L/5, 2L/5] \times [L/5, 2L/5]$ and $\Omega_{d2} = [3L/5, 4L/5] \times [3L/5, 4L/5]$. The target zones are schematically shown in Figure 5.11.

The coupled and uncoupled results are compared with the same λ and ρ parameters. Unless otherwise specified, the results are obtained using a 40×40 resolution grid on FEMuS and 81×81 on OpenFOAM so that the same number of degrees of freedom is used on both.

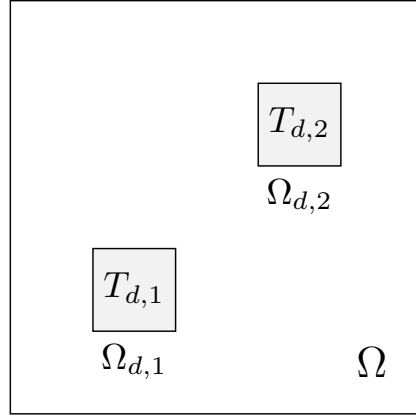


Figure 5.11: Target zones.

5.2.1 Distributed control

In this case, the control q represents the source Q on the *right-hand side* of the state equation, and the Dirichlet and Neumann boundary conditions imposed on Γ_d and Γ_n , respectively, are homogeneous. The optimal control system is defined in Section 3.2.

Coupling results. Three cases with three different values of λ were considered to appreciate different levels of regularization on the control. Specifically, λ assumes the values $10^2, 10^0, 10^{-2}$. For each case, the trend of the dimensionless control Q^* and the dimensionless state T^* along the main diagonal of the domain, from the bottom left to the top right corner, as a function of the dimensionless coordinate $r^* \in [0, \sqrt{2}]$ is reported.

Figure 5.12 shows the different behavior of the control parameter Q concerning λ , and the corresponding effect on the state variable inside the controlled regions. As λ decreases, the control Q^* assumes a shape less smooth, producing a sharp behavior close to the controlled areas. The case with $\lambda = 10^{-2}$ represents the less regularized solution, with corresponding temperatures inside the controlled region very close to the target value. Otherwise,

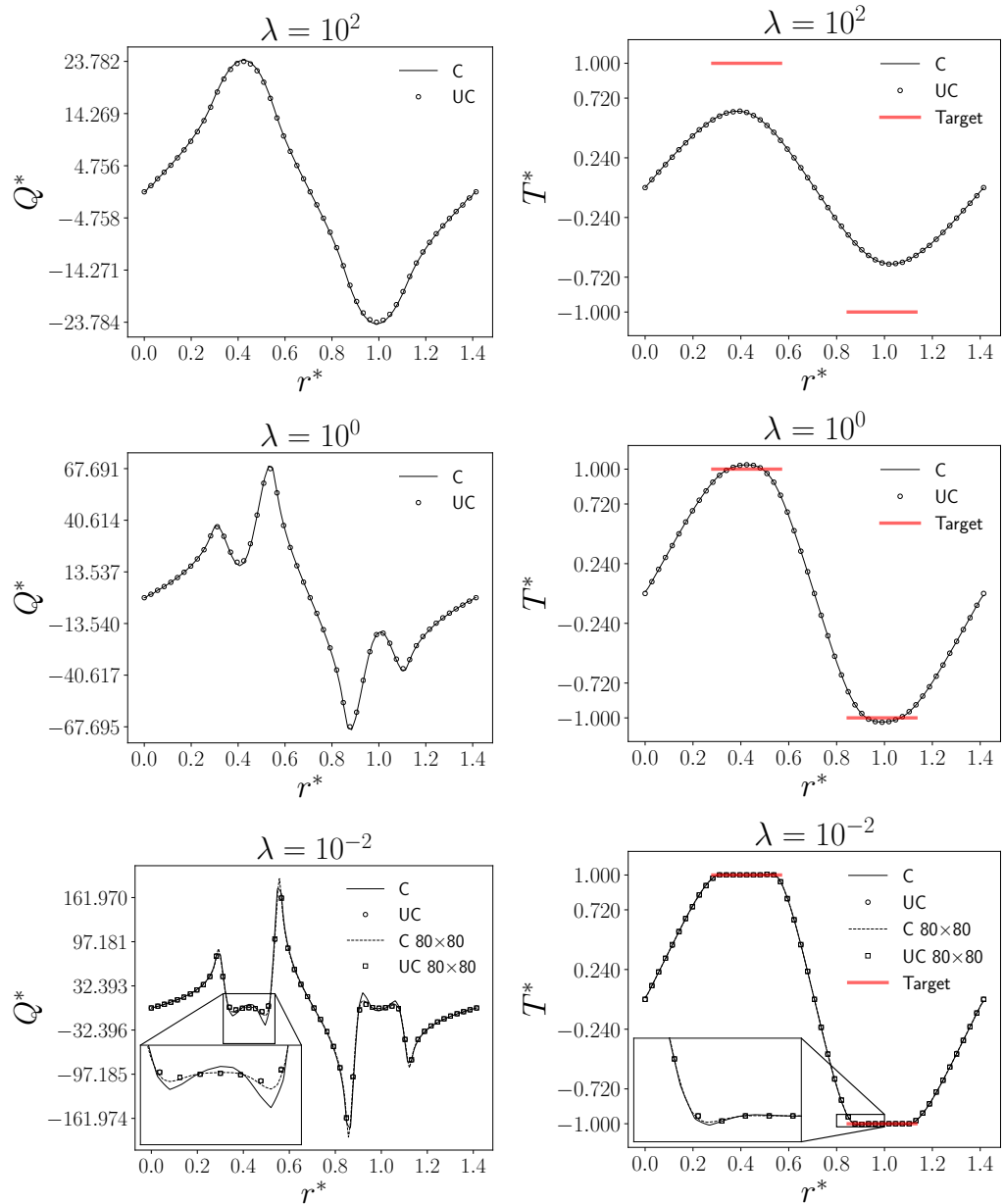


Figure 5.12: Distributed control Q^* and state T^* as a function of the dimensionless diagonal coordinate r^* , for $\lambda = 10^2, 10^0, 10^{-2}$.

with a $\lambda = 10^2$ the control Q^* has a low effect on the state variable that is not able to reach the target value on the controlled regions, as we can see from the distance with the red line.

In Figure 5.13 a surface plot of the non-dimensional temperature and the non-dimensional control is reported. Specifically, only the case with the lowest λ is shown, considering a 40×40 finite element grid. With the three-dimensional representation, the symmetric behavior of these variables can be noticed. Moreover, the sharp trend of Q^* is present close to the target regions, where several peaks can be observed. Regarding the temperature field, as expected, two plateaus are present in the target regions, with opposite values close to 1 and -1 .

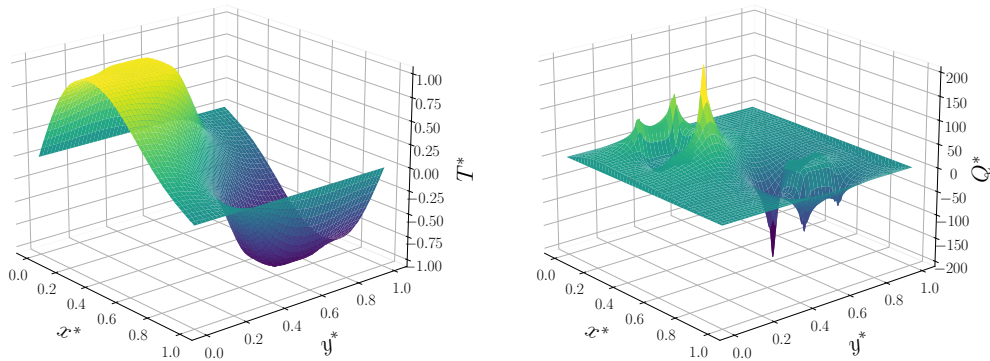


Figure 5.13: Three-dimensional representation of T^* (on the left) and Q^* (on the right) for the distributed control, considering the two-dimensional non-dimensional domain.

In Table 5.1 the functional and the number of iterations have been reported for the three different cases previously presented.

From Table 5.1, we better understand the numerical results depicted in Figure 5.12, in particular considering the functional minimum. Indeed, for each case of λ the first column represents the minimum of the functional, while the second one reports the number of iterations employed for reaching

λ	10^2		10^0		$10^{-2} - 10^{-2}(\text{refined})$	
	$\min(\mathcal{F}) \cdot 10^{-6}$	$n_{iter} \cdot 10^2$	$\min(\mathcal{F}) \cdot 10^{-7}$	$n_{iter} \cdot 10^2$	$\min(\mathcal{F}) \cdot 10^{-9}$	$n_{iter} \cdot 10^3$
C	6.97	50.9	1.87	19.9	3.24 – 3.02	2.1 – 4.1
UC	6.91	6	1.83	4	2.96 – 2.95	2.7 – 1.2

Table 5.1: Minimum functional and number of iterations for the different cases of the distributed control with both algorithms for a 40×40 grid. In the last column, for the lowest λ value, the results for the refined grid are also reported.

the convergence condition. Naturally, the first row reports the results for the coupled case, while the second row represents the uncoupled scenario. It is recalled that these results refer to a computational grid with 40×40 elements (considering the FEM grid), therefore an 81×81 grid for the FVM code has been used to have the same degrees of freedom (81×81). Regarding $\min(\mathcal{F})$, the first two values of λ show a good match for both algorithms, while a major difference can be noted for the lowest value of λ .

In fact, with $\lambda = 10^{-2}$, we observe some discrepancies between the coupled and the uncoupled results compared to higher values of λ . This phenomenon can be explained by the errors introduced during field interpolation when transferring data from one code to the other. As the optimal state is approached, the distance of the state from the objective $|T - T_d|$ decreases, making the interpolation error on T , at first negligible, increasingly important. This interpolation error limits the achievement of the real optimum state since it is not possible to obtain a sufficiently accurate control.

On the other hand, the interpolation error can be improved by increasing the number of degrees of freedom in the grid. In fact, in Figure 5.12 on the last row, are also reported the same case of $\lambda = 10^{-2}$ obtained with a double refined grid, with a dashed line for the coupled case and with squared markers for the uncoupled one. For the uncoupled case, slight differences

can be noticed because the simulation has already found its optimum, but for the coupled case an improvement has been obtained, especially for Q which is very close to the uncoupled results. These conclusions can be also drawn from Table 5.1 where the distance between the two minimum decreases considering the refined solution.

In addition, the relative errors between the functional minimum increase by decreasing the value of λ , going from 1 to 10 percent. However, the latter value corresponding to the lowest λ can be improved until around 2% with a double refined grid.

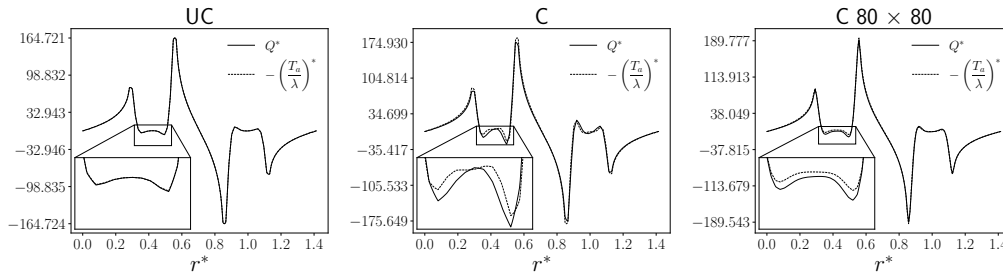


Figure 5.14: Optimal control Q and its analytic expression at the optimum state. The uncoupled case at the left, the coupled case in the middle, and the refined coupled case on the right.

A measure of the quality of the optimization convergence can be obtained using the Euler expression defined in (3.34). In Figure 5.14, the obtained control Q and its analytic optimal expression $-\frac{T_a}{\lambda}$ are compared. It is known from theory that the exact optimum is achieved when the two are identical. We can observe that for the uncoupled case, there is a perfect agreement between the two lines, while some discrepancies are present for the coupled case. This indicates that the numerical solution is slightly away from reaching the true optimum. As previously mentioned, better results are obtained considering the refined coupled case which is depicted on the right. This analysis perfectly reflects the considerations made earlier about the func-

tional minimum and also validates the accuracy of the uncoupled simulation, which is used as a reference case.

Regarding the number of iterations, the situation for the distributed control is different between the two algorithms. This is true considering the first two values of λ where the coupled case takes more iterations to satisfy the convergence condition. For example, considering $\lambda = 10^2$ there is an increase of almost one order of magnitude for the coupled case. However, this difference decreases when λ decreases, since for the most controlled case, i.e. $\lambda = 10^{-2}$ the number of iterations is similar.

In the last column, for the lowest λ the results with a refined mesh are reported. Refining the grid we can notice that on one hand, we reach a lower value of the functional, which means a better result and more similar to the uncoupled case, on the other hand, the number of iterations increases. On the opposite, the number of iterations for the uncoupled case decreases.

Grid ratios. For the distributed case, is investigated also the influence of the difference between the degrees of freedom of the two codes. This comparison aims to evaluate the impact of the interpolating routine on the overall behavior of the algorithm, especially for the shape of the control Q , which is the real output of the simulation. Therefore, three different grids have been considered for the finite volume code (OpenFOAM), by fixing the finite element computational grid with the same degree of freedom as the intermediate FVM grid. Moreover, the same simulations have been performed fixing the FVM grid and varying the FEM grid. The results have been reported in Figure 5.15, where with F is denoted the FEMuS grid, while with OF the one of OpenFOAM. For these results, only the case with $\lambda = 10^{-2}$ has been performed, and the dimensionless control Q^* is again reported as a function

of the dimensionless diagonal r^* . From Figure 5.15 we can note the difference

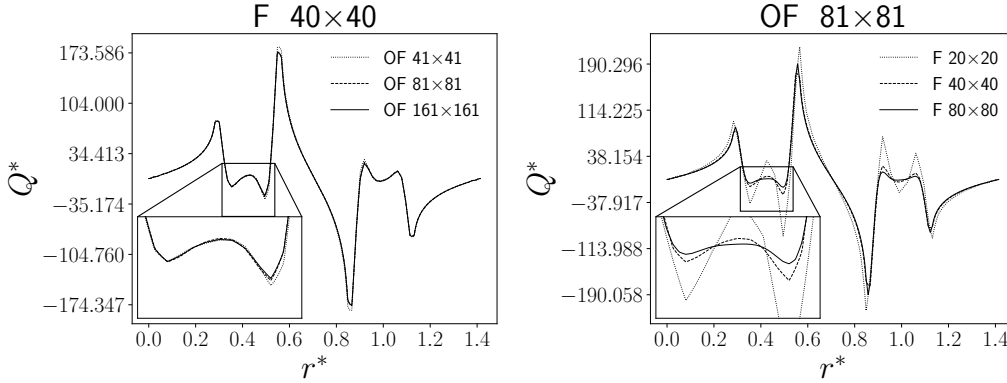


Figure 5.15: Control Q for different combinations of degrees of freedom between the two codes. On the left the FEM grid (F) is fixed, on the right the FVM grid (OF) is fixed.

of the control Q^* varying the computational grids of the CFD codes. If, on the left, the change of the FVM grid seems to have a small influence on the solution of the equation systems, on the right the situation is different. In this case, the FVM grid is fixed and the FEM grid is changed. It can be noticed that we have significant discrepancies when the coarsest FEM grid is considered, while with finer grids the results are comparable with the first case of grids combination.

It is observed that varying the OpenFOAM grid has much less influence compared to varying the FEMuS grid. The interpolation error occurs when the projected field no longer provides a good approximation of the original one, i.e., when the target grid is not fine enough to represent it. Reconfirming the observations made earlier, this reaffirms that the error occurs when the temperature field is projected from the P_0 grid to the P_2 grid of FEMuS, and when the latter is not sufficiently fine, the interpolation error becomes very significant.

In table 5.2 we can confirm the previous impressions about the effective-

	OF 41×41		OF 81×81		OF 161×161	
	$\min(\mathcal{F}) \cdot 10^{-9}$	$n_{iter} \cdot 10^3$	$\min(\mathcal{F}) \cdot 10^{-9}$	$n_{iter} \cdot 10^3$	$\min(\mathcal{F}) \cdot 10^{-9}$	$n_{iter} \cdot 10^3$
F 40×40	3.30	4.4	3.24	2.1	3.22	2.9
	F 20×20		F 40×40		F 80×80	
	$\min(\mathcal{F}) \cdot 10^{-9}$	$n_{iter} \cdot 10^3$	$\min(\mathcal{F}) \cdot 10^{-9}$	$n_{iter} \cdot 10^3$	$\min(\mathcal{F}) \cdot 10^{-9}$	$n_{iter} \cdot 10^3$
OF 81×81	4.47	6.2	3.24	2.1	3.04	3.0

Table 5.2: Minimum functional and number of iterations for the case $\lambda = 10^{-2}$ varying the grid ratio of the two codes. In the first row, the FEMuS mesh is kept fixed and the OpenFOAM mesh is varied, while in the second row, it's vice versa.

ness of the grid refinement in the two codes: we have a much greater impact on the minimum of the functional by varying the FEMuS mesh compared to varying the OpenFOAM mesh.

Moreover, it is interesting to note that the minimum number of iterations corresponds to the case where the two meshes are identical.

5.2.2 Dirichlet boundary control

In this section, the boundary control case for the Dirichlet boundary condition is reported. For the boundary control, the volumetric source of the state equation is zero, leading to a Laplace equation for T .

Grid convergence analysis. For the case $\lambda = 10^{-6}$, a grid convergence analysis was also performed for both the coupled and uncoupled cases.

From Figure 5.16 a good grid convergence can be noticed for both algorithms, confirming the reliability of the numerical solutions. The finest grid presents a solution that tends to be smooth also close to the boundary of the control region. The boundary of the boundary Γ , corresponding to the corner nodes of the domain, is the most problematic region since q is imposed

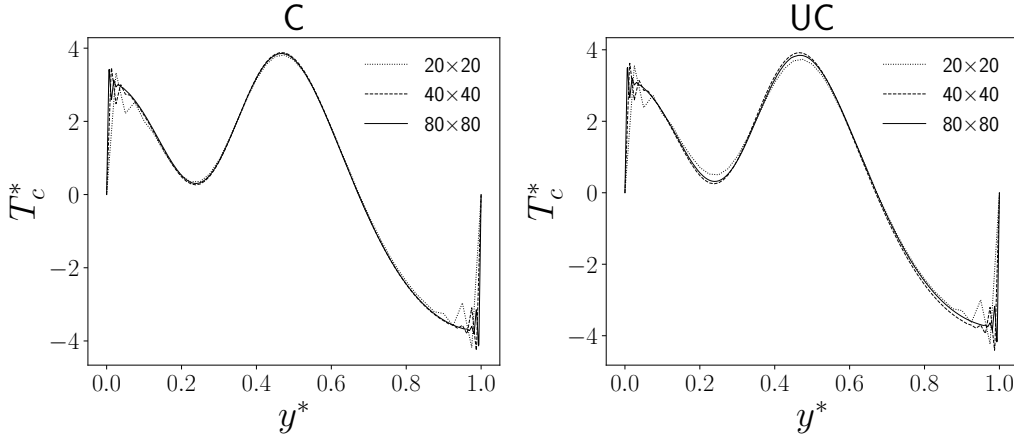


Figure 5.16: Control T_c^* for $\lambda = 10^{-6}$, grid convergence

equal to zero.

Coupling results. In this paragraph, the control is represented by the value imposed as a Dirichlet condition on Γ_d , with the Neumann condition on Γ_n and the volumetric source Q being null. The optimal control system is defined in Section 3.3. Three cases are shown with varying λ , assuming the values 10^{-4} , 10^{-6} , 10^{-8} . In each of these cases, the trend of the dimensionless control T_c^* on the left wall at $x^* = 0$ and the state T at the center of the target area $\Omega_{d,1}$ at $x^* = 0.3$ is reported as $y^* \in [0, 1]$ varies. Since the simulation is perfectly anti-symmetric to the line parallel to the y axis at $x^* = 0.5$, only a controlled wall and a target region are reported.

Also for the Dirichlet boundary control, the same conclusion of the Distributed case can be drawn for Figure 5.17. Specifically, a good match with the temperature target value can be obtained with a low value of λ , which corresponds to the less regular trend of the control T_c^* . For this case, it can be noticed a flat behavior of T^* very close to 1 in correspondence with the controlled region Ω_1 .

In Figure 5.18 a surface plot of the non-dimensional temperature is re-

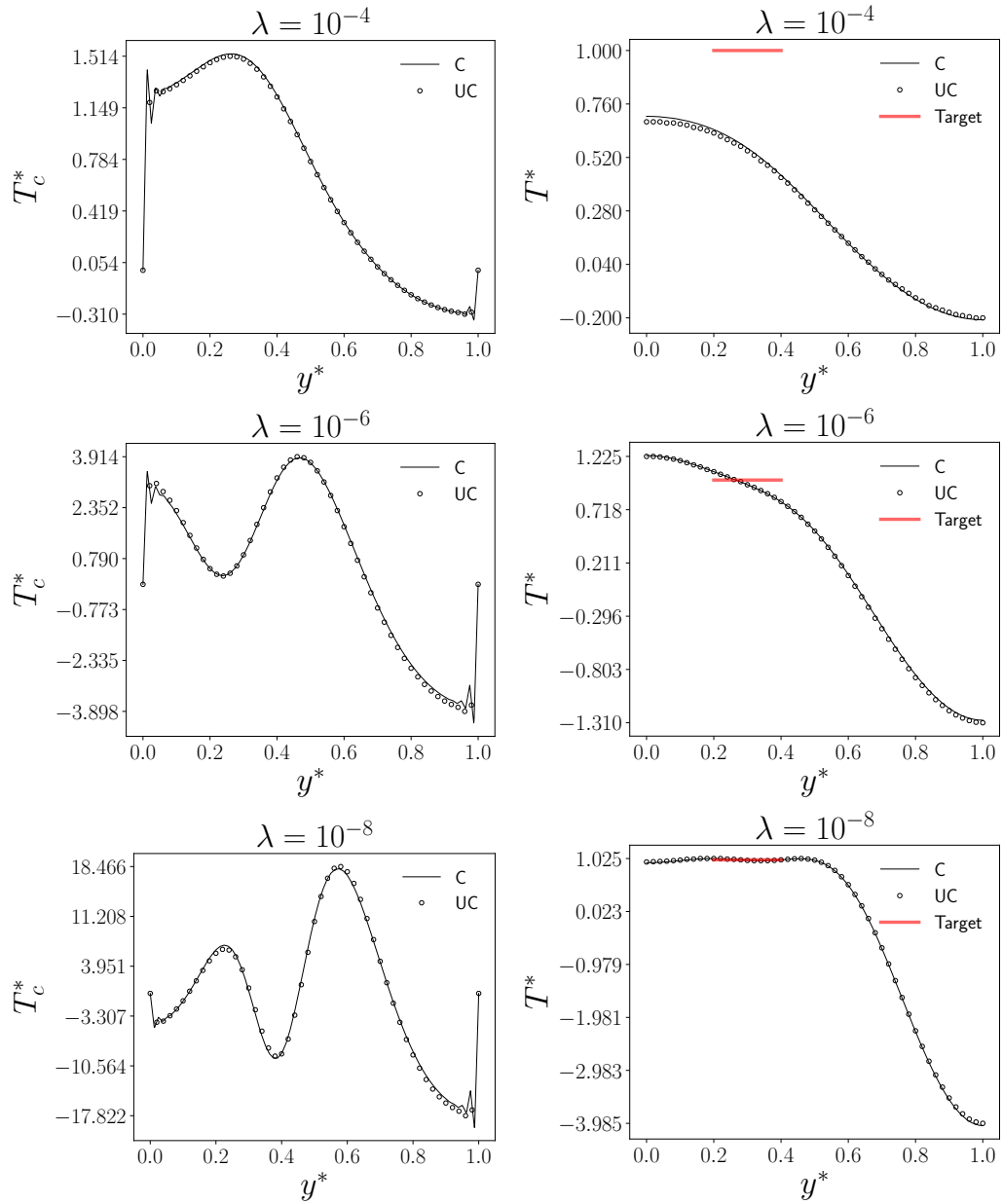


Figure 5.17: Dirichlet boundary control T_c^* and state T^* as a function of the dimensionless coordinate y^* , at $x^* = 0$ and $x^* = 0.3$, respectively, for $\lambda = 10^{-4}, 10^{-6}, 10^{-8}$.

ported. Specifically, only the case with the lowest λ is reported, considering a 40×40 finite element grid. With the three-dimensional representation, the

anti-symmetric behavior of these variables can be noticed. Moreover, we can appreciate how much the temperature at the controlled boundary increases with respect to the target value that the target zones become unrecognizable.

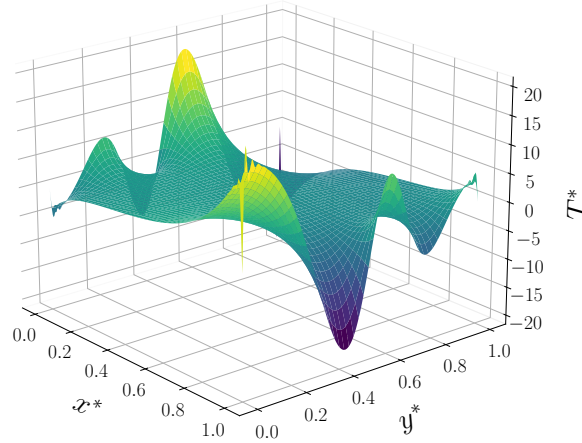


Figure 5.18: Three-dimensional representation of T^* for the boundary Dirichlet control, considering the two-dimensional non-dimensional domain.

λ	10^{-4}		10^{-6}		10^{-8}	
	$\min(\mathcal{F}) \cdot 10^{-4}$	$n_{iter} \cdot 10^2$	$\min(\mathcal{F}) \cdot 10^{-6}$	$n_{iter} \cdot 10^2$	$\min(\mathcal{F}) \cdot 10^{-8}$	$n_{iter} \cdot 10^3$
C	7.09	9	6.77	5	1.17	3.4
UC	7.18	5	6.91	13	1.19	8.3

Table 5.3: Minimum functional and number of iterations for the different cases of the Dirichlet boundary control with both algorithms for a 40×40 grid.

In Table 5.3 we can observe a better behavior of the coupled code with respect to the uncoupled, on both the minimum reached and the number of iterations. In any case, the differences between the two cases are very minimal.

5.2.3 Neumann boundary control

In this section, is reported the boundary control cases for the Neumann boundary condition. For the boundary control, the volumetric source of the state equation is equal to zero, leading to a Laplace equation for T .

Coupling results. Now are presented the numerical results obtained with the Neumann boundary control, following the system defined in Section 3.4. Also for this case, three values of λ have been employed to test the influence of the control H on the state equation.

Here, the control is represented by the value imposed as a Neumann condition on Γ_n , with the Dirichlet condition on Γ_d and the volumetric source Q being null. Three cases are shown with varying λ , assuming the values $10^1, 10^0, 10^{-1}$. In each of these cases, the trend of the dimensionless control H^* on the bottom wall at $y^* = 0$ and the state T at the center of the target area $\Omega_{d,1}$ at $y^* = 0.3$ is reported as $x^* \in [0, 1]$ varies. Once again, being the simulation anti-symmetric only one controlled wall and one target region have been reported.

The same comments of the previous cases can be also done for the Neumann boundary control considering the result in Figure 5.19. A perfect agreement between the state variable and the target temperature can be obtained only with a lower value of λ since for greater value of this parameter the control H^* does not have sufficient effect on the temperature T^* . On the other hand, for a lower value of λ , some discrepancies can be noticed for the control H^* on the controlled boundary. In particular, the major difference is present in the target region where for the uncoupled solution the control H^* seems to be more flat, concerning the coupled result. The explanation is analogous to the distributed case. Once again, is reported the same case

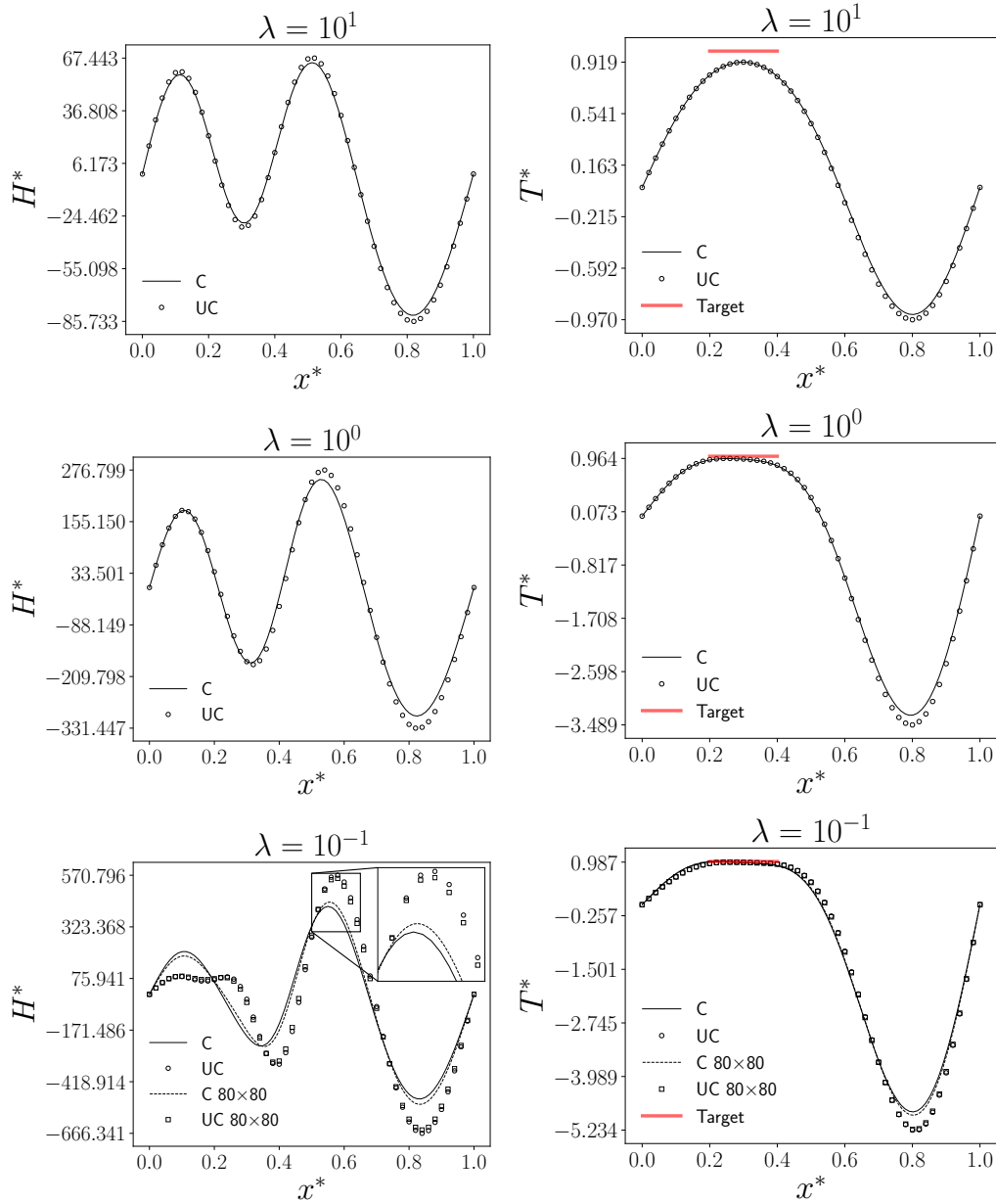


Figure 5.19: Neumann boundary control T_c^* and state T^* as a function of the dimensionless coordinate x^* , at $y^* = 0$ and $y^* = 0.3$, respectively, for $\lambda = 10^1, 10^0, 10^{-1}$.

of $\lambda = 10^{-1}$ obtained with a double refined grid, with a dashed line for the coupled case and with squared markers for the uncoupled one. Similarly,

as in the distributed case, we can observe that the coupled and uncoupled results become similar as the grid becomes finer. On the other hand, these discrepancies, do not influence the state temperature T^* which is in perfect agreement with the desired value for both algorithms. A slight difference can be noticed for $x^* \approx 0.8$, where for the uncoupled algorithm T^* reaches a lower value.

In Figure 5.20 a surface plot of the non-dimensional temperature is reported. Specifically, only the case with the lowest λ is reported, considering a 40×40 finite element grid. With the three-dimensional representation, the anti-symmetric behavior of these variables can be noticed. Once again, with boundary control at the wall, we obtain a much higher temperature than the target value.

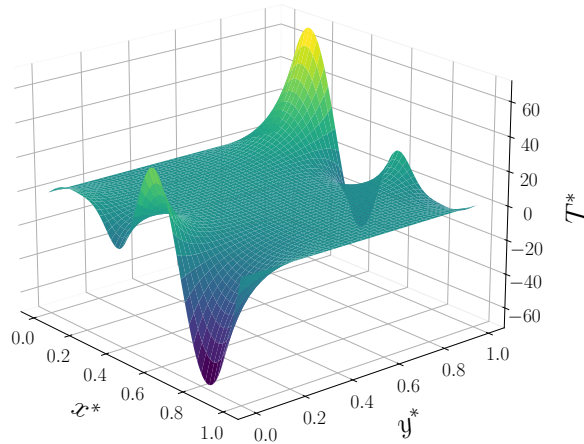


Figure 5.20: Three-dimensional representation of T^* , for the boundary Neumann control, considering the two-dimensional non-dimensional domain.

As one can see in Table 5.4, the number of iterations increases both on the coupled and in the uncoupled codes, decreasing the value of λ . For the case with $\lambda = 10^{-1}$ we again can observe the same pattern of the distributed case above: finer is the grid on the coupled code lower is functional minimum.

λ	10^1		10^0		$10^{-1} - 10^{-1}(\text{refined})$	
	$\min(\mathcal{F}) \cdot 10^{-6}$	$n_{iter} \cdot 10^3$	$\min(\mathcal{F}) \cdot 10^{-7}$	$n_{iter} \cdot 10^3$	$\min(\mathcal{F}) \cdot 10^{-7}$	$n_{iter} \cdot 10^3$
C	1.27	1.1	6.09	5.2	2.24 – 2.16	19.2 – 96.4
UC	1.28	3.8	6.12	18.1	1.99 – 1.97	39.0 – 10.5

Table 5.4: Minimum functional and number of iterations for the different cases of the Neumann boundary control with both algorithms for a 40×40 grid. In the last column, for the lowest λ value, the results for the refined grid are also reported.

Chapter 6

Conclusions

In this work, an optimal control problem for the heat equation has been implemented in the in-house code FEMuS and solved via coupling with the external code OpenFOAM. The optimal control was solved by adopting the adjoint method, while minimization was performed using the conjugate gradient method.

In this setup, the state equation was solved within OpenFOAM. The adjoint equation and the control were handled by FEMuS. Specifically, the state variable T was taken from OpenFOAM and used in the adjoint equation in FEMuS. Conversely, the control variable q was computed in FEMuS and transferred back to OpenFOAM, where it was applied either as a distributed control (RHS) or as a boundary condition (boundary control). Data transfer between the two codes was facilitated by the external library MEDCOUPLING.

The numerical results obtained through the coupled framework were compared with those produced by the stand-alone FEMuS solution. For all three types of control problems, a good agreement was observed between the two methods. The differences may be due to interpolation errors for lower values

of the regularization parameter λ since the numerical precision requested for the state field T was very high.

Future work will focus on improving the current status of the optimal system on FEMuS, implementing some regularization techniques [6] for better handling boundary control cases, as well as an adaptive step-size calculation and other minimization techniques [18]. Moreover, this framework can be extended to other systems of equations, ranging from Navier-Stokes and Fluid-Structure Interaction (FSI) [12] to turbulence modeling [28] and shape optimization [24]. The goal is to leverage existing multi-physics capabilities of open-source codes such as OpenFOAM, which, while robust in many areas, currently lacks optimal control functionalities based on the adjoint method.

List of Figures

2.1	Piecewise constant function.	18
2.2	Piecewise constant approximation.	19
2.3	Linear Lagrange polynomial.	20
2.4	Linear approximation.	22
2.5	Local nomenclature of a 1D quadratic element.	23
2.6	Quadratic Lagrange polynomials.	24
2.7	Quadratic approximation.	25
2.8	Two-dimensional domain partition.	26
2.9	Transformation from global coordinates to canonical coordinates for a QUAD4 element.	27
2.10	Shape functions for X_h^1 on the 2D canonical domain.	28
2.11	Canonical QUAD9 element.	29
2.12	Shape functions for X_h^2 on the 2D canonical domain.	31
2.13	Weights and coordinates for two-dimensional Gaussian quadrature.	32
2.14	Assembly of the local matrix A^e into the global matrix A	37
3.1	Trajectory comparison between the two descent methods: SG and CG.	48
4.1	Hub-and-Spoke structure representation.	62

4.2	Coupling diagram.	64
4.3	Interpolation from point-wise to cell-wise field.	68
4.4	Central point of a two-dimensional (left) or one-dimensional (right) boundary face.	69
4.5	Interpolation from cell-wise to point-wise field.	69
5.1	State of the problem.	72
5.2	Single target region for the uncoupled problem.	75
5.3	Three-dimensional representation of T^* (on the left) and Q^* (on the right) for the distributed control, considering the two- dimensional non-dimensional domain.	76
5.4	The non-dimensional temperature T^* (on the left), and the control Q^* compared with its analytic optimum expression (on the right), as a function of x^* , for $y^* = 0.5$	77
5.5	Dimensionless control Q^* (left) and temperature T^* (right) with the same λ^* but different domain, as a function of the non-dimensional diagonal r^*	78
5.6	Double target region for the uncoupled problem.	79
5.7	The state variable T^* as a function of y^* for $x^* = 0.3$, in the middle of the target region (on the left), and across the two- dimensional domain (on the right), for the Dirichlet boundary control.	79
5.8	Control T_c (on the left) and its comparison with the analytic optimum expression (on the right) as a function of y^* , for $x^* = 0$	80
5.9	Adjoint temperature T_a across the bidimensional domain, for Dirichlet boundary control.	81
5.10	Convergence ratio between CG and SG methods, as a function of the functional value threshold and the mesh grid.	81

5.11	Target zones.	83
5.12	Distributed control Q^* and state T^* as a function of the dimensionless diagonal coordinate r^* , for $\lambda = 10^2, 10^0, 10^{-2}$	84
5.13	Three-dimensional representation of T^* (on the left) and Q^* (on the right) for the distributed control, considering the two-dimensional non-dimensional domain.	85
5.14	Optimal control Q and its analytic expression at the optimum state. The uncoupled case at the left, the coupled case in the middle, and the refined coupled case on the right.	87
5.15	Control Q for different combinations of degrees of freedom between the two codes. On the left the FEM grid (F) is fixed, on the right the FVM grid (OF) is fixed.	89
5.16	Control T_c^* for $\lambda = 10^{-6}$, grid convergence	91
5.17	Dirichlet boundary control T_c^* and state T^* as a function of the dimensionless coordinate y^* , at $x^* = 0$ and $x^* = 0.3$, respectively, for $\lambda = 10^{-4}, 10^{-6}, 10^{-8}$	92
5.18	Three-dimensional representation of T^* for the boundary Dirichlet control, considering the two-dimensional non-dimensional domain.	93
5.19	Neumann boundary control T_c^* and state T^* as a function of the dimensionless coordinate x^* , at $y^* = 0$ and $y^* = 0.3$, respectively, for $\lambda = 10^1, 10^0, 10^{-1}$	95
5.20	Three-dimensional representation of T^* , for the boundary Neumann control, considering the two-dimensional non-dimensional domain.	96

List of Tables

3.1	N. of iterations, functional and coordinates of the minimum, and relative errors of the two minimizations realized with SG and CG methods.	49
5.1	Minimum functional and number of iterations for the different cases of the distributed control with both algorithms for a 40×40 grid. In the last column, for the lowest λ value, the results for the refined grid are also reported.	86
5.2	Minimum functional and number of iterations for the case $\lambda = 10^{-2}$ varying the grid ratio of the two codes. In the first row, the FEMuS mesh is kept fixed and the OpenFOAM mesh is varied, while in the second row, it's vice versa.	90
5.3	Minimum functional and number of iterations for the different cases of the Dirichlet boundary control with both algorithms for a 40×40 grid.	93
5.4	Minimum functional and number of iterations for the different cases of the Neumann boundary control with both algorithms for a 40×40 grid. In the last column, for the lowest λ value, the results for the refined grid are also reported.	97

Bibliography

- [1] Robert A Adams. Sobolev spaces. 1975, 1975.
- [2] Robert A Adams and John JF Fournier. *Sobolev spaces*. Elsevier, 2003.
- [3] Giacomo Barbi, Giorgio Bornia, Daniele Cerroni, Antonio Cervone, Andrea Chierici, Leonardo Chirco, RDA Viá, Valentina Giovacchini, Sandro Manservisi, R Scardovelli, et al. Femus-platform: A numerical platform for multiscale and multiphysics code coupling. In *9th International Conference on Computational Methods for Coupled Problems in Science and Engineering, COUPLED PROBLEMS 2021*, pages 1–12. International Center for Numerical Methods in Engineering, 2021.
- [4] Giacomo Barbi, Sandro Manservisi, Antonio Cervone, Federico Giangolini, Sandro Manservisi, and Lucia Sirotti. Numerical coupling between a fem code and the fvm code openfoam by using the med library. *Applied Science*, 2024.
- [5] Giorgio Bornia and Saikant Ratnavale. Different approaches for dirichlet and neumann boundary optimal control. *AIP Publishing LLC*, 2018.
- [6] Giorgio Bornia, Andrea Chierici, and Saikant Ratnavale. A comparison of regularization methods for boundary optimal control problems. *International journal of numerical analysis and modeling*, 2022.

- [7] Susanne Brenner and Ridgway Scott. *The mathematical theory of finite element methods*, volume 15. Springer Science & Business Media, 2007.
- [8] Daniele Cerroni. *Multiscale multiphysics coupling on a finite element plat-form*. PhD thesis, University of Bologna, 2015.
- [9] Andrea Chierici. *Mathematical and Numerical Models for Boundary Optimal Control Problems Applied to Fluid-Structure Interaction*. PhD thesis, University of Bologna, 2021.
- [10] L Chirco, R Da Vià, and S Manservigi. An optimal control method for fluid structure interaction systems via adjoint boundary pressure. *Journal of Physics: Conference Series*, 2017.
- [11] L Chirco, V Giovacchini, and S Manservigi. An adjoint-based temperature boundary optimal control approach for turbulent buoyancy-driven flows. *Journal of Physics: Conference Series*, 2020.
- [12] Leonardo Chirco. *On the optimal control of steady fluid structure interaction systems*. PhD thesis, University of Bologna, 2020.
- [13] P. G. Ciarlet. The finite element method for elliptic problems. *Society for Industrial and Applied Mathematics*, 2002.
- [14] Roberto Da Vià. *Development of a computational platform for the simulation of low Prandtl number turbulent flows*. PhD thesis, 2019.
- [15] J. Dennis and R. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, volume 1. Society for Industrial and Applied Mathematics (SIAM), 1996.

- [16] M. D’Elia and M. Gunzburger. The fractional laplacian operator on bounded domains as a special case of the nonlocal diffusion operator. *Computers & Mathematics with Applications*, 2013.
- [17] P. Farrell and J. Maddison. Conservative interpolation between volume meshes by local galerkin projection. *Computer Methods in Applied Mechanics and Engineering*, 200(1-4):89–100, 2011.
- [18] R. Fletcher. *Practical methods of optimization*. Wiley-Interscience (John Wiley & Sons), 2001.
- [19] Reeves Fletcher and Colin M Reeves. Function minimization by conjugate gradients. *The computer journal*, 7(2):149–154, 1964.
- [20] G. Folland. *Real analysis: modern techniques and their applications*. *Pure and applied mathematics*, 1999.
- [21] A Fursikov, M Gunzburger, LS Hou, and S Manservisi. Optimal control problems for the navier-stokes equations. *Lectures on applied mathematics (Munich, 1999)*, 2000.
- [22] Valentina Giovacchini. *Development of a numerical platform for the modeling and optimal control of liquid metal flows*. PhD thesis, 2022.
- [23] Max D Gunzburger. *Perspectives in flow control and optimization*. SIAM, 2002.
- [24] Max D Gunzburger, Hongchul Kim, and Sandro Manservisi. On a shape control problem for the stationary navier-stokes equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 2000.

- [25] Hrvoje Jasak. Openfoam: Open source cfd in research and industry. *International Journal of Naval Architecture and Ocean Engineering*, 1 (2):89–94, 2009.
- [26] Hrvoje Jasak, Aleksandar Jemcov, Zeljko Tukovic, et al. Openfoam: A c++ library for complex physics simulations. In *International workshop on coupled methods in numerical dynamics*, volume 1000, pages 1–20. Dubrovnik, Croatia), 2007.
- [27] Sandro Manservisi. *Course in modeling and simulation techniques for energy engineering*. 2022.
- [28] Sandro Manservisi and Filippo Menghini. Optimal control problems for the navier–stokes system coupled with the k- ω turbulence model. *Computers & Mathematics with Applications*, 2016.
- [29] Sandro Manservisi and Filippo Menghini. Numerical simulations of optimal control problems for the reynolds averaged navier–stokes system closed with a two-equation turbulence model. *Computers & Fluids*, 2016.
- [30] Andrea Manzoni, Alfio Quarteroni, and Sandro Salsa. *Optimal Control of Partial Differential Equations*. 2021.
- [31] John L Nazareth. Conjugate gradient method. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(3):348–353, 2009.
- [32] J. Nocedal and S.J. Wright. *Numerical optimization*. Springer, 2006.
- [33] Asim Önder and Johan Meyers. Optimal control of a transitional jet using a continuous adjoint method. *Computers & Fluids*, 126:12–24, 2016.

- [34] Carsten Othmer. A continuous adjoint formulation for the computation of topological and surface sensitivities of ducted flows. *International journal for numerical methods in fluids*, 58(8):861–877, 2008.
- [35] Peter Philip. *Optimal Control of Partial Differential Equations*. 2013.
- [36] Alfio Quarteroni and Silvia Quarteroni. *Numerical models for differential problems*, volume 2. Springer, 2009.
- [37] Andre Ribes and Christian Caremoli. Salome platform component model for numerical simulation. In *31st annual international computer software and applications conference (COMPSAC 2007)*, volume 2, pages 553–564. IEEE, 2007.
- [38] G. H. Silva, R. Le Riche, J. Molimard, and A. Vautrin. Exact and efficient interpolation using finite elements shape function. *European Journal of Computational Mechanics/Revue Européenne de Mécanique Numérique*, 18(3–4):307–331, 2009.
- [39] W. Sun and Y.-X. Yuan. *Optimization theory and methods. Nonlinear programming*, volume 1. Springer, 2006.
- [40] OC Zienkiewicz. The finite element method. vol. 1, basic formulation and linear problems. 1989.
- [41] Olgierd Cecil Zienkiewicz, Robert Leroy Taylor, Olgierd Cecil Zienkiewicz, and Robert Lee Taylor. *The finite element method*, volume 3. McGraw-hill London, 1977.