University of Pavia

Department of Civil Engineering and Architecture (DICAr)
Ph.D. course in Design, Modeling and Simulation in Engineering

Ph.D. Thesis

Computational Fluid Dynamics simulations of Laser Metal Deposition process exploring open source software

Candidate:
Mauro Murer

Thesis advisor:
Prof. F. Auricchio

Thesis co-advisors:
Prof. G. Formica
Prof. S. Morganti

XXXIII Cycle

Abstract

Laser Metal Deposition (LMD) is an innovative technology adopted in Additive Manufacturing (AM) processes and its use is becoming more and more popular in various application fields, such as part manufacturing, repair, and prototype fabrication. This technique is capable of creating several layers of solidified material, by the simultaneous delivery of metal powders and the laser beam, and offers an effective way to produce complicated geometries thanks to its high flexibility. However, complex physical phenomena occur during the additive process, which have a great impact on the success of the process, and many of these have yet to be fully understood. With the aim of shedding light on these aspects, a detailed numerical study, focused on LMD technology, will be conducted using three-dimensional models based on Computational Fluid Dynamics (CFD).

The particle flow problem regarding the coupling between the fluid phase (i.e., the carrier gas) and the solid phase (i.e., a metallic material powder) is first investigated using OpenFOAM, an open source software widely used in the CFD community. In particular, two different numerical approaches are investigated: the first approach is based on an Eulerian method to describe the carrier gas flow combined with a Lagrangian method to describe the particle flow (LE method), and the second approach is based on a pure Eulerian method to model both the carrier gas and the particle flow (EE method). Simulations results show the main features of the two approaches considered in terms of reliability in reproducing the key geometrical and physical features of the LMD process, together with a comparison with experimental evidences.

On the other hand, the thermal problem, that describes the interaction between particles flow and the laser beam, play a crucial role and cannot be neglected. For this purpose, the time-dependent Navier-Stokes equations for incompressible flows are coupled with the energy equation in order to represent the temperature field, whereas the Lagrangian description of the particle dynamic is enriched accounting the thermal evolution, and the consequent phase changing of the metallic powder due to the particle-laser interaction. This model is developed in a C++ in-house code using the open source Finite Element library deal.II and it is validated through consolidated results available in the literature. Furthermore, different schemes able to solve the Navier-Stokes equations, coupled with the heat transfer equation, are implemented and compared, so as to prove both accuracy and efficiency.

Then, with the aim of investigating the LMD process in detail, and in particular the thermal behaviour of the powder exiting from the nozzle, a sensitivity analysis is performed in terms of the parameters most meaningful from a technological viewpoint, i.e., the nozzle inclination, the carrier gas and powder flow rate, and the laser power. The results of such an analysis show that it is possible to predict both the configuration and the energy distribution that characterizes the flow of the powder leaving the nozzle until it reaches the substrate. In particular, the influence of both laser power and nozzle geometry to phase change conditions of powder flux

are analyzed in order to improve the set up of the printing process, which can lead to increased productivity and less material waste.

Contents

Introduction **1**

Additive Manufacturing: a focus on Laser Metal Deposition **5**
2.1 Laser Metal Deposition technology 5
2.1.1 LMD process 7

Physical problems **10**
3.1 An introduction of CFD methodologies for LMD 10
3.1.1 Lagrangian-Eulerian (LE) methods 10
3.1.2 Eulerian-Eulerian (EE) methods 13
3.2 Modeling coupled problems of flow and heat transfer with particles 15

Computational modeling and numerical schemes **18**
4.1 Navier-Stokes equations: the role of the pressure 18
4.1.1 Incompressible flow problem 18
4.1.2 Weak formulation and spatial discretization 21
4.2 Advection-diffusion equations: the necessity to stabilize 26
4.2.1 Variational Multiscale method 27
4.2.2 Advection-diffusion problem 29
4.2.3 Streamline Upwind Petrov-Galerkin stabilization 29
4.3 Time integration methods 33
4.3.1 The θ family methods 33
4.3.2 Multistep methods 35
4.4 Algorithms for solving linear systems 37
4.4.1 Direct Solvers 37
4.4.2 Iterative Solvers 38
4.5 Particle methods 43

Numerical approaches and implementation **47**
5.1 Available softwares: an overview 47
5.1.1 Commercial software 47
5.1.2 Free software 48
5.2 OpenFOAM: Problem formulation and implementation 52

5.2.1 Eulerian-Eulerian (EE) approach . . . 52
5.2.2 Lagrangian-Eulerian (LE) approach . . . 54
5.2.3 Implementation details . . . 55
5.3 Deal.II: Problem formulation and implementation . . . 58
5.3.1 Navier-Stokes equations: Newton-Raphson Method . . . 59
5.3.2 The stabilized heat equation . . . 61
5.3.3 Navier-Stokes equations: Projection Methods . . . 62
5.4 Deal.II: Benchmark tests . . . 67
5.4.1 Benchmark 1: Expansion channel . . . 67
5.4.2 Benchmark 2: Transient advection-diffusion . . . 70
5.4.3 Benchmark 3: Backward-facing step flow . . . 72
5.4.4 Benchmark 4: Flow over square obstacle through different pipes. . . . 75
5.4.5 Benchmark 5: Flow over square obstacle in a pipe . . . 78
5.5 Deal.II: Particles dynamic equations . . . 83
5.5.1 Particle-wall interaction . . . 83
5.5.2 Particles algorithm . . . 86
5.5.3 The fully coupled Particles-CFD algorithm . . . 93
5.5.4 Benchmark 6: OpenFOAM particle tracking method comparison . . . 95

Numerical results **97**
6.1 LMD simulations with OpenFOAM . . . 97
6.1.1 Simulation set up . . . 97
6.1.2 On the comparability of the two approaches . . . 98
6.1.3 Velocity and mass fields . . . 102
6.1.4 Powder shape cone . . . 105
6.1.5 Towards experimental validation . . . 107
6.2 LMD simulations with Deal.II . . . 112
6.2.1 Simulations set up . . . 112
6.2.2 Results and discussions . . . 116

Conclusions **125**
Future improvements . . . 128
Source code availability . . . 128
Acknowledgements . . . 128

References **128**

Nomenclature

α	Element Peclet number
α_e	Thermal expansion
α_p	Solid volume fraction
β	Momentum exchange coefficient
$\boldsymbol{\omega}_p$	Particle angular velocity
$\boldsymbol{\tau}_i$	Stress tensor field
$\boldsymbol{a}_\sigma$	Parcel acceleration
$\boldsymbol{f}$	Force term
$\boldsymbol{F}_{i,j}$	Momentum transfer term between the phases i and j
$\boldsymbol{g}$	Gravity force
$\boldsymbol{J}_p$	Impulsive force acting on the particle
$\boldsymbol{u}$	Velocity
$\boldsymbol{u}_p$	Particle velocity
$\boldsymbol{x}_p$	Particle position
$\dot{m}$	Mass flow rate
ϵ_i	Volumetric fraction of each i-th phase
η_p	Particle absorption coefficient
Γ	Domain boundary
γ	Heat source term
κ	Diffusivity coefficient
λ_i	Bulk viscosity
$\mathbf{J}$	Jacobian matrix
μ	Dynamic viscosity
ν	Kinematic viscosity
Ω	Domain
ω	Vorticity
ρ	Density
ρ_p	Density of the particle
τ	SU/PG stabilization parameter
θ	Angle measure
ξ	Particle distribution function
A_p	Area of the particle
$A_{p,p}$	Effective projected area of the particle
Bi	Biot number
C_D	Drag coefficient
c_p	Specific heat
D	Diameter of the laser beam
d_p	Diameter of the particle
e	Coefficient of restitution
f	Particle liquid mass fraction
F_i	Forces acting on the particle
f_r	Coefficient of kinetic friction
h	Heat convection
H_j	Heat terms acting on the particle
I_p	Particle moment of inertia
I_T	Laser total energy incident
k_p	Particle heat conduction coefficient
L_f	Latent heat of fusion

m_l	Particle liquid mass
m_p	Mass of the particle
Nu	Nusselt number
P	Laser power
p	Pressure
Pe	Peclet number
Pr	Prandtl number
r_p	Radius of the particle
Re	Reynolds number
Re_p	Particle Reynolds number
St	Strouhal number
T	Temperature
t	time
T_p	Temperature of the particle
T_{liq}	Liquidus temperature
T_{sol}	Solidus temperature
V_p	Volume of the particle
w	Radius of laser beam
AM	Additive Manufacturing
DED	Direct Energy Deposition
LMD	Laser Metal Deposition

Preamble

In recent years, the world has been experiencing the birth of new trends and technologies that are completely different from those that occurred in the last decades of the XX century and the beginning of the XXI. In particular, Additive Manufacturing (AM), also known as 3D printing technology, produces objects layer by layer (additively) rather than subtracting material and it represents a recent technological advance with a high potential for a new industrial evolution [1]. In fact, several strategic research programs in the European Union and in the United States are investing public and private resources on Additive Manufacturing to enhance technological aspects and to strengthen the competitiveness of the industrial network [2, 3]. In particular, one of the most important and active entities in the European AM context is Lombardy, the first Italian manufacturing region and the third in Europe by number of employed, which contributes to 2.71% of the European Union's GDP and 21% of the national GDP with a network of 820000 companies, of which 63% SMEs [4]. It's here worth mention MADE4LO (MEtal ADditive for LOmbardy), the first project dedicated to the development of metal 3D printing technologies in Italy that is completely *made in Lombardy* [5]. The research asset is costituted by a pool of 9 companies (3 GI and 6 SMEs) and 2 research centers (University of Pavia and Milan Polytechnic) focused on developing a Lombard alliance in advanced manufacturing, and dedicated to the chain of additive processes for the manufacturing of metal parts. The research proposal is based on the indications contained in the *Road Map* of AFIL (Associazione Fabbrica Intelligente Lombarda) [6], the Regional Technological Cluster promoted by Region Lombardy to manage the research and development needs of regional manufacturing companies. Among AFIL members of the Additive Manufacturing Working Group, additive processes are emerging as the frontier of advanced manufacturing and one of the enabling technologies of Industry 4.0 [7, 8].

Today, the Additive Manufacturing of metal parts is still a business with high access difficulties and, in terms of productivity, it is far from being a technology for mass customization, in particular for small manufacturing companies. The reasons for such difficulties are related to an incomplete knowledge of the process, to the high investment costs for limited production, and to the high cost of raw material [9]. In response to these issues, MADE4LO aims to develop a new model of business accounting also for the strong presence of small and medium companies on the Lombard territory. The key-concept is "shared physical Industry 4.0", where new digital production models, increasingly automated and interconnected, favour the process traceability and a collective, shared, and collaborative information management between the various partners. In this sense, MADE4LO fits perfectly the research investments of Region Lombardy funding innovation projects with a significant partnership and investments, and with positive impacts on the Lombard competitive network.

In this context, the University of Pavia strongly reflects these peculiarities, thanks to its research and innovation projects strongly dedicated to the additive world. In fact this partnership is no accident. The University of Pavia, an in particular the CompMech Group of DICAr (Department of Civil Engineering and Architecture), has been engaged for years in the additive manufacturing field [10, 11, 12], activating several 3D printing laboratories with a wide spectrum of technologies under the project named 3D@UniPV [13]. The goal of this innovative project is to give an even further boost toward innovative applications, in particular combining current technologies with two other fundamental ingredients, such as virtual modeling and advanced materials. In fact, the term 3D@UniPV indicates the choice of the University of Pavia to create a research center for Additive Manufacturing focused on 3D printing technology, capable of responding to the growing demand from various application fields of the industrial and medical world [14, 15, 16]. In addition, another laboratory was inaugurated in 2018, the 3DMetal@UniPV, dedicated to metal 3D printing and equipped with a powder bed laser melting system. The 3DMetal@UniPV aims to print metal components with an innovative design carring out research activities on devices optimization and additive manufacturing process simulation. Moreover, the CompMech Group has organized "The Second International Conference on Simulation for Additive Manufacturing (Sim-AM 2019)" in September 2019, with an high participation from both academic and industrial field [17].

In relation to this additive manufacturing specialization, various tasks have been assigned to the University of Pavia within the MADE4LO project. This collaboration mainly concerns the activity of numerical simulations, fluid-dynamic and thermo-mechanics, in order to support the development of new components and the optimization of those that already exist. In particular, the CFD (Computational Fluid Dynamics) analysis of a flow of metal powder which passes through a nozzle and it is melted by means of a laser beam and then solidifies, thus creating a layer of deposited material, represents a problem of relevant interest to the MADE4LO project, and it is the topic of this thesis. This technology is part of the DED (Direct Energy Deposition) processes, more specifically named LMD (Laser Metal Deposition), and allows the creation of metal objects through the deposition of several layers of solidified material. The goal of the CFD study is the redesign of the deposition head, in order to optimize the powder flow inside and outside the nozzle, which is influenced by many multiphysics aspects, such as the coupling with an energy field, generated by a laser beam, and the melting and solidification phenomena [18]. Therefore, in the next chapters, the LMD printing process will be explored, from both a physical and a modeling point of view, with the aim of expanding and contributing to the knowledge in the Additive Manufacturing field, a subject still at the beginning and with strong prospects of development.

Introduction

The sustainability and the competitiveness of the manufacturing system depends on the ability to introduce and manage innovative processes and take advantage of these [19]. In this context, Additive Manufacturing (AM) technologies offers numerous competitive benefits compared to traditional production processes. The high freedom of design in terms of geometric complexity, usable materials and properties of the products developed, shortens *time-to-market* [20, 21]. A complete digitalization of both the design and the production processes allowed by AM technologies [22] keeps the production flexibility extremely high [23], with significant advantages in terms of time and costs, especially for high-value prototypes and small series production [24, 25, 26]. The use of AM systems for the repair of worn components extends the life cycle of the product, with an evident impact on sustainability, especially in reference to the use of rare or precious materials [27, 28]. Last but not least is the impact on the supply chain, since AM can potentially produce a component when and where required, thus reducing stocks and simplifying the production chain logistics [29].

Although AM has been widely tested as a prototyping technique and it is assuming a consolidated role in some industrial sectors (*e.g.*, biomedical, dental, aerospace) [30, 31, 32], some features need to be improved to allow this technology to have a widespread diffusion in other industrial fields (*e.g.*, automotive, machinery and tooling, creative industries) [33, 34]. In particular, the main challenges that need to be addressed in relation to AM processes concern: (i) increase in productivity; (ii) optimize costs on the entire value chain (powders, systems, post-processing); (iii) reach *zero-defect* additive manufacturing; (iv) increase in sustainability (recycling, life-cycle assessment); (v) increase in the size of the components that can be printed; (vi) enlarge the range of workable materials, also employing *multi-material* solutions (functionally graded), and, (vii) integrate post-processing activities towards hybrid solutions [35, 36, 37, 38].

Among the various technologies developed in the AM context, one that is growing very rapidly and is widely used in the industrial field is Directed Energy Deposition (DED). This process has been used extensively and becomes more and more popular in part manufacturing, repairing and prototype fabrication of metal objects. It offers an effective way to deposit metallic material via simultaneous delivery of the desired material powders and laser beam. Complex physical events take place during the process and have great impacts on the success of the process, in which powder stream and its interaction with laser beam play a crucial role [39]. The complexity, linked to the multiphysical phenomena that characterize the printing process,

has highlighted many uncertainties and difficulties that compromise an efficient employment of the manufacturing technology. Therefore, to better study and optimize the powder stream process, many research projects have been created and funded [40, 41]. These have explored the DED process both from an experimental point of view, employing 3D printers produced in house or in collaboration with companies, or using numerical approaches, through the use of CFD.

In particular, many authors have published papers concerning the numerical modeling of DED additive processes (see Section 3.1 for more details). Many of these contributions represent the powder flow distribution, without including any thermal interaction or phase change phenomena, using commercial codes only [42], sometimes comparing the outcomes with experimental data [43]. Mostly the results show different output configurations when the input parameters vary, such as the use of different carrier fluids [44] injected varying their pressure values [45], the use of nozzles with variable geometry [46] and inclination angles [47], or the use of powders characterized by different granulometries [48] and mass flow rates [49]. Some have developed models capable of representing particular characteristics, such as stochastic models to consider realistic collisions that occur in powder flow [50, 51], or analytical models that describes the variation of powder stream concentration [52]. Another explored field specifically concerns the dynamics of the melt pool and in particular the description of the formation, flows and heat exchanges that characterize the melt pool itself [53].

These works are limited to single parts that constitute the entire DED process and, moreover, those that compare the use of different numerical schemes are rare, if not non-existent. In fact, computational aspects, for this particular type of application, are not investigated. Consequently the literature does not provide an exhaustive vision about the modeling of the physical process in question, as the problem of finding more efficient models has never been posed.

On the other hand, our CFD study led to interesting conclusions, both from a physical and numerical point of view, by exploring fields that have not yet been covered in the literature. In particular, this study is treated in two distinct phases, the former using the OpenFOAM software [54], in which only the fluid-particle interaction is studied, and the latter developing an in-house C++ code, which exploits the Finite Element library deal.ii [55], in which the influence of the thermal field on the powder flow enriches the description of the process, also including a phase change model. With OpenFOAM two different approaches are considered, with the aim of understanding which type of numerical model is more efficient to represent the flow of particles immersed in a fluid. The first approach use an Eulerian description to model the carrier gas flow combined with a Lagrangian description related to the particle flow (LE method), and the second approach is based on a pure Eulerian method to describe both the carrier gas and the particle flow (EE method) [56]. The main features of the two approaches are analyzed and discussed, also from a computational point of view, highlighting both reliability and validity of the models in reproducing the key geometrical and physical features of the powder flow. However, the applicability of OpenFOAM to the physical problem in question

is limited, because the implementation of the interaction between the powder flow and the laser beam, which cannot be neglected, results complicated to perform. For this purpose, a C++ in-house code using the open source Finite Element library deal.II is developed. Here, the Navier-Stokes equations are coupled with the heat equation in an Eulerian description, whereas the particle dynamic follows a Lagrangian description enriched with a thermal evolution model.

The Eulerian coupled problem handles the fully nonlinear formulation of the Navier-Stokes equations, solved with a modified Newton-Raphson scheme, in order to represent in accurately and efficiently way the advection-diffusion problem [57]. The proposed algorithm does not introduce any linearization, as usually done in stabilized methods, and it has been compared with two approaches coming from well consolidated numerical formulation, namely the projection-correction scheme (PC) [58] and the iterated projection-scheme (iPC) [59]. Moreover, the presence of high Péclet numbers requires the stabilization of the heat equation, which takes place by using the SUPG method [60]. As regard the treatment of the particles, in order to save computational costs, a parallel code is implemented. Then, each particle is governed both by a moment balance equation, which describes the exchange of forces between the particle itself and the fluid, and by a temperature equation, which instead describes its thermal evolution [61]. In particular, the heat exchange that occurs due to the action of the laser beam translates not only into an increase in the temperature of the particle, but also into a consequent phase change of the same. Therefore, the model implemented is able to trace, not only the dynamic of the particle, but also its thermal properties, taking into account the phase change from the solid to the fluid state. This aspect represents a novelty in the computational modeling of the physical process, as to the author knowledge, it has never been treated in the literature.

Finally, a sensitivity analysis of the DED process, performed in terms of the most meaningful parameters, describes both the configuration and the energy of the powder flow at different working conditions. The aim of the present study is certainly to improve the set up of the printing process parameters, but also to provide an efficient and reliable tool able to accurately represent the physical process in reasonable times. These conditions can lead to a considerable saving of time in the design phase, and to a consequent increase in productivity and a decrease in wasted material in the prototyping phase.

Thesis outline

This work aims to understand and simulate, through various computational approaches, a particular class of DED manufacturing technology, that is Laser Metal Deposition (LMD). The present research study, ranging from a general description of the process from a physical viewpoint to the numerical implementation details, is divided into the following chapters:

Chapter 2. A global view of the LMD process is given, thus pointing out the main physical aspects that characterize the printing process. Some fundamental working principles will be outlined, and materials, instruments and methods will be detailed.

Chapter 3. A literature review of the various Computational Fluid Dynamic (CFD) methods that have been used to describe the LMD process are presented. Then, the system of equations able to model the multiphysics phenomena of the printing process considered in this work are shown in a general way, giving also some theoretical background.

Chapter 4. The numerical problems related to the incompressibility constraint of a fluid in the Navier-Stokes system of equations, and the necessity to stabilize the temperature equation to avoid oscillations on the solution, are the topics of this chapter. The fully-discretized forms of the equations are derived, whereas the time integration methods and the algorithms employed to solve linear systems are detailed. Moreover, the equations that govern both the motion and the energy conservation of the particles immersed in a fluid are described.

Chapter 5. An in-depth comparison between commercial and opensource software is provided at the beginning of this chapter, focusing on the two libraries that are used to simulate the problem, OpenFOAM and deal.II. Then, the numerical features and the implementations details of the algorithms adopted to perform efficient simulations, first with OpenFOAM and then with our code implemented in deal.II, are presented and discussed. Moreover, several validations of the developed in-house C++ code with benchmarks proposed in the literature are shown, analysing and comparing the algorithms implemented.

Chapter 6. Results of the simulations obtained, first with OpenFOAM, and then with our code, are presented and described in details. OpenFOAM outcomes are mainly focused on the comparison between the Eulerian-Eulerian approach and the Eulerian-Lagrangian approach. Furthermore a brief comparison with experimental data is shown. On the other hand, the results obtained with our code highlight the coupling with the laser beam and the consequent phase change of the powder flux, modeled through a simplified approach.

Chapter 7. The main conclusions are summarized, together with a brief discussion about improvements operable in setting the LMD printing process on the basis of the outcomes obtained from CFD numerical simulations.

Additive Manufacturing: a focus on Laser Metal Deposition

2.1 Laser Metal Deposition technology

Additive Manufacturing (AM) refers to an engineering process in which objects are built up and then fabricated by means of adding, melting, and solidifying material layer by layer, opposed to the subtractive technologies [62]. The single layers are produced based on the direction controlled by the 3D CAD information. Such a process is particularly appropriate to produce complex and near-net shape geometries due to its high flexibility especially in terms of design [63]. There are over thirty different technologies that made up the additive manufacturing process, and recently the ASTM (American Society for Testing and Materials) International and the ISO (International Organization for Standardization) have grouped these technologies into seven main categories (see Figure 2.1) of AM: (i) Vat Photo Polymerisation; (ii) Material Extrusion, (iii) Material Jetting, (iv) Binder Jetting, (v) Powder Bed Fusion, (vi) Direct Energy Deposition (DED) and (vii) Sheet Lamination [64]. Among these categories, Direct Energy Deposition is suitable for producing metal parts via local layer by layer deposition of molten metal materials, with a simultaneous delivery of beam and raw material (typically powder or wire) [65]. More in general, DED processes are based on the heating and melting of a

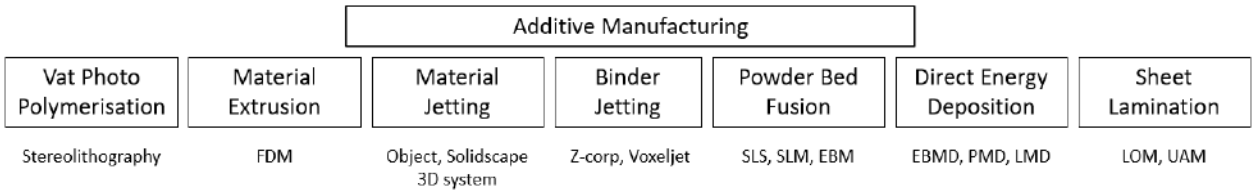

Figure 2.1: Classification of Additive Manufacturing technologies

substrate due to a direct energy source, as well as the simultaneous melting of raw material that is deposited on the substrate [66]. In contrast with techniques of powder bed fusion which melt a material that is prelaid in a powder bed [67, 68], DED processes melt materials as they are being deposited in the so-called melt pool. This feature allows to freely deliver both the powder and the laser beam according to any orientation (vertical, horizontal or inclined) and a robotic arm is usually employed to these purposes. Moreover, large-size components can be easily printed using this technique, since the powder does not have to be accommodated into

a carefully-laid flat bed inside an enclosed chamber [69]. Obviously DED can be considered as an interdisciplinary technology since it uses laser technology, computer-aided design and manufacturing (CAD/CAM), robotics, sensors, controls, and powder metallurgy [70]. In addition, depending on the thermal energy source, DED can be distinguished in Electron Beam Metal Deposition (EBMD), Plasma Metal Deposition (PMD), and Laser Metal Deposition (LMD) [18]. The present work focuses on this latter, where metallic powder material is first injected into a laser focal region, and then it is melted and solidified to form a fully dense track.

LMD emerged in the modern industry as one of the most important manufacturing technology, due its capability of producing complex geometries and light-weight components, repairing damaged components, creating coatings and depositing multi-material and functionally-graded compositions [71, 72]. The main advantages related to this AM process are, the ability to repair high valued component parts, and the capability to handle multiple materials simultaneously, with the possibility of using higher powder deposition rates. Instead, the geometrical accuracy and the quality of the finished surface, strongly dependent on the type of materials used and the processing parameters employed, is still a disadvantage [73]. In fact, it was demonstrated [74] that the surface roughness increased as the scanning speed of the laser beam was increased, and post-processing operations may are required to improve surface finishing and geometrical accuracy. However, this type of technology allows for producing functionally graded parts directly from the 3D CAD model in one single step [75]. Moreover, taking into account economical aspects, the LMD technology is attractive especially in terms of design flexibility and capability in reducing component weight, as well as in saving time and energy throughout the fabrication process [76]. Material waste and its recycle may be also considered from a sustainability viewpoint [77]. Lots of organizations have developed their own LMD machines using lasers and powder feeders, and Figure 2.2 shows the one built and used in the laboratories of Milan Polytechnic. Although the basic configuration is the same for all the LMD machines, several changes can be made to improve and adapt the printing process performance by modifying, for example, the laser type, the laser spot size, the laser power, the feedback control scheme, the powder and inert gas delivery method, and the type of motion control utilized.

Due to its nature, LMD is inherently a multi-physics process involving particle transportation, multiple flow interaction, particles-gas interaction with energy source and material phase-change (from solid powder to liquid phase and then a solidification again). Therefore, the printing process can be conveniently simplified in three different physical phenomena: (i) particles flow inside and outside the nozzle; (ii) interaction between laser beam and moving powders; (iii) particle deposition, fusion and solidification inside the melt pool, partially composed by molten substrate/layer material. In particular, the investigation of the particles flow inside and outside the nozzle, and the consequent interaction with the laser beam, is propaedeutic for the correct prediction of the phenomenon, since the powder delivery at the exit of the nozzle strongly affects the entire process [78]. The powder flow rate is directly connected to different factors, which are able to change the geometrical shape of the powder cone and its main properties, like the waist diameter of the powder and its position. In addition, powder cone

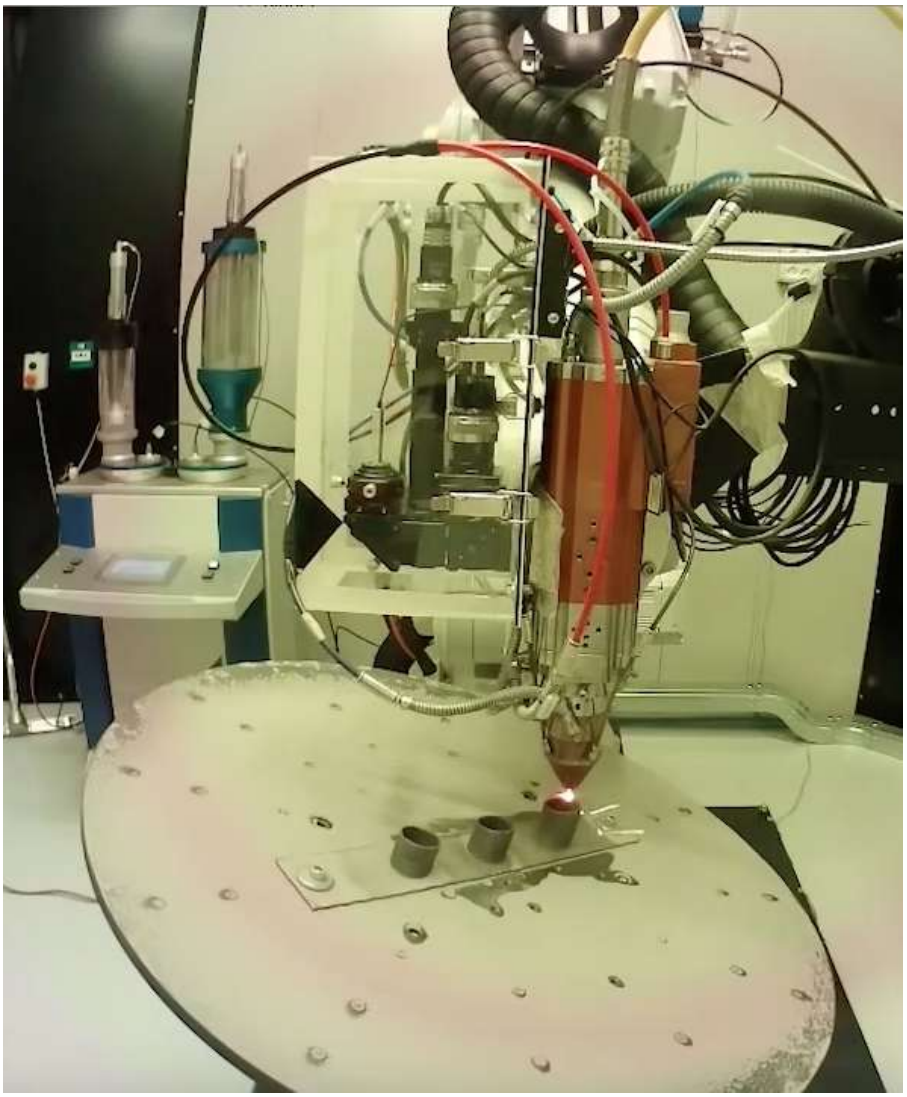

Figure 2.2: LMD machine built and used in the laboratories of Milan Polytechnic

alterations are able to produce a different behavior in terms of material processability. In this perspective, the simulation of powder delivery up to the nozzle is mandatory to control the process successfully.

2.1.1 LMD process

The target of the LMD process is to melt one or more layers of powder material to produce a solid part using a continuous wave or, less frequently, pulsed wave laser power sources [79]. A schematic view of the LMD process is shown in Figure 2.3. A deposition head is used to deposit the feedstock material, in form of discontinuous and spherical powder onto the substrate. A powder LMD system commonly integrates a laser head with laser optics which are able to deliver the laser beam at the target place. The laser head is commonly equipped with a nozzle(s) along with a carrier gas transports the powder up to the substrate. In order to prevent occurrences of oxidation phenomena, the deposition of metallic powder is protected by a coaxial shielding gas,

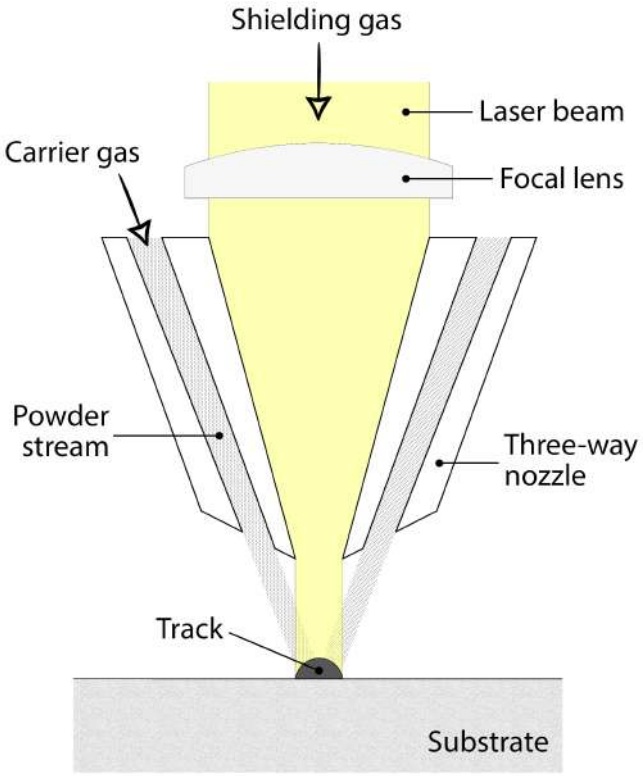

Figure 2.3: A sketch of Laser Metal Deposition (LMD) process

injected together with the carrier one [80]. The shielding gas is the same of carrier one, and the employed gases are inert like nitrogen or argon. Moreover, the right amount of gas flow rate is crucial for a proper success of the process. In fact, small amount of gas flow rate determines a poor protection of the material being deposited, if it is an highly reactive material, while a too high flow rate, which increases the particles velocity, may blow away the powder from the point where the powder has to be melted (melt pool area).

All these components together contribute in delivering metallic particles within a pool of molten material, generated by an energy source focused on a small region of the working surface, allowing the fusion and subsequent solidification of the deposited powder. Note that, with high temperatures generated by the laser energy around the deposition head, the powder can be fused arriving at the substrate in a molten state. However, the powder flow rate needs to be effectively controlled, because a large amount of metallic material flow rate could result in improper melting or even no melting at all. On the other hand, too low powder flow rate may cause the vaporization of the particles, which is not desirable in LMD [73].

In general, the deposition process is controlled by relative differential motion between the substrate and the nozzle. This differential motion is accomplished by moving either the nozzle or the substrate, or by a combination of both substrate and nozzle motion. Typically, a 3 axis systems are utilized, whereby the deposition process occurs in a vertical manner. However, 4 or 5 axis systems using either rotary tables or robotic arms are also available. In fact, nonvertical deposition is just as effective as vertical deposition, as kinetic energy of powder particles during the flight, from the nozzle to the melt pool, is greater than the effect of gravity. Therefore

a multiaxis deposition is possible and indeed quite useful. Moreover, the configuration of LMD machines can change, also during the deposition process, depends on the geometries and dimensions of the printed object. If the substrate is very large and/or heavy, it is easier to accurately control the motion of the deposition head than the substrate; on the other hand, if the substrate is a simple flat plate, it is easier to move the substrate than the deposition head.

An other important parameter to set optimally is the scanning velocity, that is the length of time at which the laser beam interacts with the substrate and the deposited materials. In fact, too low velocities could produce an high dilution or an evaporation of the deposited material, while high scanning movements produce a poor interaction between the energy source and the metallic material bringing to an incomplete melting. The metallic powder is typically formed by particles that have the diameter ranging from 40 μm to 150 μm. Different metallic materials can be used moving from steels up to titanium or aluminum powders.

Hence, a welded track is erected when the laser irradiation moves out of the melt pool, and then the solidification process of the molten material onto the substrate takes place. The dimensions of the printed track, generally characterized by a width and a height, can be different depending on the employed system and process parameters. Typical range of values are 1-5 mm and 0.2-1 mm for width and height, respectively [81]. The final object is then additively constructed, or 3D printed, by consecutive overlapping tracks that repeat all the steps of the process described above. The overlap is realized along the vertical axis between consecutive layers, so that the solidified layer becomes the substrate of the next to be deposited, which is partially re-melted.

This brief technological description reveals how a wide range of physical phenomena characterize the LMD process, and consequently, many sub-problems have to be tackled in order to optimize the entire LMD process.

Physical problems

3.1 An introduction of CFD methodologies for LMD

Laser Metal Deposition (LMD) is an emerging manufacturing process that integrates different advanced technologies such as lasers components, motion control systems and metal powder products. Due to the interactions of these various components, a complex combination of multiphysics and multiphase phenomena occurring during the process.

In particular, the motion of particles is influenced primarily by the surrounding carrier and shielding gas, which deliver the powder stream through the nozzle channels. The interaction between particles and the channels walls causes a trajectory deflection of the powder stream that must be accounted in the numerical simulations. Then, the in-flight powder exiting from the nozzle is heated by a radiated laser beam source, that together with the high temperatures of the surrounding environment, produce heat transfer phenomena. Finally, the influence of the melted substrate on particles catchment, as well as the action of both solidification and evaporation processes in the melt pool, characterize the rest of the additive production chain.

It is obvious that a complete numerical description of all the entire process is very difficult to perform, and therefore an individual analysis of the LMD phases is necessary. In literature, many authors have tried to simulate some of these multiphysics aspects that characterize the printing process, and below, a brief overview of the various CFD methods implemented for this purpose, is reported.

3.1.1 Lagrangian-Eulerian (LE) methods

Concerning the numerical methods able to predict the particle flow, the Lagrangian approach is one of the most commonly used in the literature, due to its ability to work as particle-tracking method. This approach is based on an Eulerian method to describe the carrier gas flow combined with a Lagrangian method to describe the particle flow.

One of the early work was done by Pan and Liou [50] who developed a stochastic Lagrangian model that accounts for particle shape effects, in particular non-spherical collisions through stochastic parameters, and that is able to quickly simulate realistic powder flow. Once validated with experiments, such a model allows the authors to evaluate various nozzle geometrical configurations, showing the dispersion of metallic powder due to the deviation from the sphere-shape in particle morphology. Such a dispersion evolution plays a key role in the focus-

ability of the powder stream and powder spatial concentration, and it turns out that width and outer diameter of the powder outlet passage are important dimensions in the determination of the powder stream structure. The same authors improved the stochastic model in a subsequent work [51], where effects of outer shielding gas directions and inner/outer shielding gas flow rate had been also considered in the model.

Zhang and Coddet [44] conducted a comprehensive numerical study that investigates the effect of pressure and nozzle dimension on particle distribution and velocity in laser cladding systems. They developed three-dimensional CFD models in Ansys Fluent, solved using a Discrete Phase Modeling (DPM) approach to compute particles acceleration while Navier-Stokes equations have been considered to model the inert gas. In particular the authors found out that during laser cladding processes higher particle velocity and more compact powder flow can be obtained by Helium vacuum environment than using Argon, and nozzle dimensions have significant effect on particle distribution and velocity. The same CFD numerical model, presented by Zeng et al. [46], is employed to better understand the powder deposition process and to analyze the influence of the geometrical and processing parameters such as the standoff distance, the volumetric gas flow rate, and the powder mass flow rate on the quality of the LMD printing technology.

Along the same research perspective, similar approaches aiming at optimizing nozzle design and validating particle flow experimental measurements can be previously found in [43, 52, 47]. In particular, Tabernero et al. [43] simulate the powder flux on a real continuous coaxial nozzle, capable of predicting the powder distribution shape and particle velocities and trajectories. Numerical simulations show that a stream with different powder flow shapes is generated by the nozzle, starting from an annular shape, when particles exit from the nozzle, to a Gaussian shape near the powder stream focal point. The predictions of the CFD model show good agreement with experimental results.

Only few other works have studied the real powder flux shape assuming a Gaussian distribution, such as Pinkerton and Li [52]. In particular, the authors developed an analytical model that describes the variation in powder stream concentration along the axis of a coaxial nozzle. The model, that supports the use of Gaussian equations for powder streams, is applicable both before and after the powder stream merges into a central stream and relates directly to nozzle dimensions and material mass and volume flow rates. Good agreement was found comparing experimentally results with those predicted by the model.

Arrizubieta et al. [47] investigated other aspects related to the LMD process. In particular, using the Ansys Fluent DPM method to simulate the powder flux inside the nozzle, the optimal values of the carrier and shielding gases flow rates have been studied. Here, it is clear that the powder velocity in the focal plane exhibits almost a linear increase when both carrier e shielding gas flow rates increase. Moreover they explored the influence of the water cooling system during the printing process and the nozzle working efficiency changing the inclination angle of the deposition head. The employment of water flow minimizes the maximum temperature reached by the plastic nozzle without a cooling system (around 250°C) below 50°C, avoiding the plastics

melt and the consequence bock of the powder flow and the interruption of the process. On the influence of the inclination angle on the deposition process they concluded that the designed nozzle works correctly until inclinations of 30°, increasing this value the particles concentration and the spot size are not adequate for a correct operation of the nozzle.

Also [48, 49, 45] have developed numerical strategies based on the already mentioned DPM approach, the particle-tracking method available in Ansys Fluent, in order to explore how nozzle geometry, powder properties, and feeding parameters can improve LMD process efficiency. From the results obtained by Balu et al. [48], on the characterization of the coaxial powder flow behavior of Ni–WC composite powders, emerged that: (i) the peak of the powder stream in the typical Gaussian distribution tends to be relatively flat in the multimaterial powder flow; (ii) strong influence on the particle concentration is due to particles rebound on the nozzle walls both in radial and axial directions; (iii) density, shape factor and diameter of the particle influence the average velocity of the powder particles that plays a key role in stable powder flow; and (iv) changing nozzle angle leads to a longer powder stream and alters the shape of the powder stream.

In order to improve the technological processes of cladding with powders, Grigoryants et al. [49] tried to determinate the optimum values of the flow rate of powder, the size of powder particles, the geometry of the powder flow and the distance from the outlet of the nozzle to the surface of the components. The authors found out the optimal fractional composition of the powder flow when particles size ranging between 20–120 μm. The upper limit is associated with the difficulty of melting larger mass particles, while the lower limit is imposed to avoid the scattering of the directional flow of the smaller particles. Moreover, both simulations and experimental results show that the minimum diameter, that ensures the minimum width of the powder stream, is reached when the convergence angle of the powder flow is 52°, whereas the maximum productivity of the process is reached when this angle is 64°, but in this case the width of the stream is greater than the diameter of the pool melted by the laser beam.

The work of Liu et al. [45] is mainly focused on the effects of the nozzle exit geometry on the characteristics of the powder flow. In fact, they proved the influence of the exit shape of the nozzle on the powder flow structure controlling the pressure distribution at one side of the nozzle exit and highlighting the benefits in providing more coherent powder flow. Furthermore, they noted that the powder concentration at the side of the nozzle is higher than at the central region, with the consequence that the distribution is not uniform and affects the deposited clad geometry. More uniform powder distribution at the outlet of the nozzle, and therefore a clad with a flat surface, could be achieved adjusting the carrier-gas flow rate. The shape of the powder flow at the nozzle outlet is also studied by Mazzucato et al. [82], who have experienced the usefulness of a nozzle configuration with a shielding gas external to the carrier gas. This design decreases the powder spread in correspondence to the deposition surface, leading to a significant improvement in the efficiency of LMD printing process.

A recent contribution to the knowledge of the gas/powder streams characteristics is made by Ferreira et al. [42]. The authors, in order to perform the numerical modeling of a jet flow, have

employed a 2D axisymmetric models of both the gas and powder streams, with RANS turbulent model implemented with COMSOL Multiphysics 5.3a software. The used Euler–Lagrange model indicates a good agreements between numerical and experimental results, pointing out the great impact of particle rebound conditions, that should be linked to the particle concentration, to correctly describe the powder stream structure, especially for nozzles with small exit diameters. The comparison with the experimental campaign also show an over-estimation of the gas velocity by pressure measurements.

Some interesting improvements are also presented in the work of Ibarra-Medina and Pinkerton [78], where a thermal coupling between powder flow and laser beam has been added. In particular, in addition to various interactions that occur during the printing process, powder stream formation, powder heating and mass deposition into the melt pool are also considered and analyzed using the commercial software CFD-ACE+, capable to model multiphysics problems. Numerical results prove that mass concentration within the powder stream, overall powder stream heating and mass deposition rate are strongly dependent by the distance from the nozzle tip to the substrate. Furthermore, process features as, dimensions of the powder stream, the amount of powder supplied to the process, properties of both powder and substrate, size of powder particles, laser power, substrate absorptivity, scanning speed and position of the substrate are the main parameters that influence the amount of mass incorporated into the melt pool.

The above mentioned methods can be framed in a so called Lagrangian-Eulerian formulation – sometimes indicated as CFD-DEM (Computational Fluid Dynamics - Discrete Element Method) – where a Lagrangian approach for the particles solution is effectively combined with an Eulerian one that is used for the flow solution. However, for an overview on Lagrangian numerical methods used to model powder flow dynamics, for coaxial powder type, the reader can refer to [83].

3.1.2 Eulerian-Eulerian (EE) methods

An alternative formulation consists of a pure Eulerian framework, namely Eulerian-Eulerian model, where both particles and flow are treated as Eulerian fluids. While the Eulerian-Lagrangian approach has been extensively adopted, the Eulerian-Eulerian approach has never been considered for simulating powder flow in LMD processes. Computational advantages/disadvantage of this latter, compared to the previous one, have been explored only in different application fields. To cite a few, we mention turbulent dispersion and coalescence of droplets within a spray [84], particle-laden flows subject to radiative heating [85], and fluidized bed systems [86].

In [84] the authors showed that both Lagrangian and Eulerian approaches are able to simulate droplet turbulent dispersion and coalescence for a wide range of droplet and gas flows, and for sprays from nozzles that produce different droplet-size distributions. They have compared Lagrangian and Eulerian methods to predict the axial mean velocity profiles of droplets at various axial locations downstream of the nozzle. Both models predict similar decay rates for

the axial mean velocity at the center-line and the spreading rates of droplets of different sizes are also similarly predicted by both Lagrangian and Eulerian models. Thus, both approaches are able to predict the main features of a turbulent spray. The computational time required for simulating coalescence within a steady axisymmetric spray is of a similar order of magnitude regardless of which formulation, Eulerian or Lagrangian, is adopted. However, the Lagrangian formulation is more practical in terms of the range of applicability and ease of implementation.

In [85] a good agreement between Lagrangian and Eulerian results is also proved. The authors employ an Eulerian method, namely Monokinetic moment method, as alternative to Lagrangian particle tracking, to solve the statistics of the disperse phase directly in order to describe a thermally two-way coupled system. They show that the Eulerian moment method is an accurate way to describe gas-particle flows in a two-way context, but a limiting factor has been found. In fact, the Eulerian approach can not capture more than one velocity per grid position, and it makes the method valid only if no particle trajectory crossing occurs. Authors also proved that the gas phase obtained from both methods is equivalent, while the statistics of the disperse phase are strongly effected by the number of particles. In particular, calculations are performed over a wide range of parameters by varying the particle Stokes number. In case of very small Stokes numbers, where results are highly sensitive to the number of particles involved, Lagrangian simulations are still the preferred method, and new developments are necessary for Eulerian methods.

In [86] both the Lagrangian and Eulerian approaches are employed to perform simulations of solids mixing in gas fluidized beds with various inlet velocities. Both the approaches predict similar fluidization behaviors and bubbling phenomena, that have a significant influence on solids mixing. Good agreement in fluidization behaviors simulated using the two different modeling approaches was observed, with slight discrepancies in the exact solid volume fractions at each point in time, due to inherent differences in the constitutive equations, drag models and in the parameters describing the motion of the phases. At low gas velocity, the rate of solids mixing predicted by CFD-DEM was higher than that predicted by the Eulerian-Eulerian model. At higher gas velocities, improved agreements in mixing behaviors are observed. However, severe bed expansions in the initial phases of the process are predicted only by CFD-DEM.

Not moving away from CFD numerical methods for Additive Manufacturing processes, pure Eulerian approach are also applied in order to simulate the dynamic of the melt pool, not covered by this thesis, but a topic of particular interest. In fact, the above mentioned authors, Ibarra-Medina and Pinkerton, in a subsequent work [53] improved the previous model taking into account problems related to melt pool formation, melt pool flows and heat transfer from powder to melt pool. In order to determine the free surface of the melt pool, the Volume of Fluid (VoF) method [87] is applied, exploiting the capabilities of the multiphase software CFD-ACE+. This method lends itself very well to represent complicated and realistic shapes, in terms of both transverse and longitudinal profile, such as those found in clads with high wetting angle or inter-clad porosity.

Another very interesting numerical simulation about LMD process via Eulerian approach

is presented by Wagner et al. [88]. In order to model complex free surface, fluid flow, thermal and laser interaction evolution, a novel multiphase thermo-fluid formulation based on a diffusive Level Set method [89] coupled with the Navier-Stokes, energy conservation and radiative transport equations is implemented. Results show that the penetration of the powder within the focal spot of the laser is favored by larger velocity of the particles, whereas the evaporation across the surface of the particles, due to laser absorption, drive powder motion either into or outside of the melt pool. Moreover, other features that affect the melt pool dimensions, such as the laser absorption by this latter, can be altered by large amount of powder material, that shields the melt pool from the energy source produced by the laser beam.

However, the literature is really vast, covering a broad variety of general numerical methods, and for a recent and sufficiently comprehensive reference bibliography see for instance [90, 91, 92, 93, 94, 95].

3.2 Modeling coupled problems of flow and heat transfer with particles

The general framework concerning convective heat transfer flows is particularly challenging since it represents a situation that involves intrinsically coupled problems, that is, problems involving multiple physical phenomena. In fact, it is necessary to employ two classical areas of applied mechanics, namely, fluid mechanics and heat transfer, in order to model a coupled system of equations. In realistic contexts, Computational Fluid Dynamics simulations have been used to solve a variety of industrial fluid flow and heat transfer problems and, in order to obtain suitable numerical results, the fluid behavior, often modeled by Navier-Stokes equations, should be coupled with heat transfer, governed by the energy equation. These equations consist of a set of coupled, nonlinear, partial differential equations in terms of the velocity components, pressure, and temperature. Industrial applications where such fluid-thermal coupling is numerically simulated have a diverse variety, *e.g.*, heat exchangers, spent fuel storage of nuclear power plants, solar collectors, crude oil storage tanks, energy storage devices, and modern crop dryers [96, 97].

In particular, such a wide span of physical phenomena may be categorized in two different fluid-thermal coupling problems, depending on the forces that are responsible for the fluid motion: i) *free or natural convection* problems, where the fluid motion is produced by temperature-induced buoyancy forces that generate small Reynolds numbers within the flow, then the nonlinear terms due to inertial effects can be neglected, resulting in a linear boundary value problem named Stokes flow; and ii) *forced convection* problems, where the application of pressure or viscous forces on the fluid boundary produce the fluid motion, generally in these cases the Reynolds number starts to grow and an advection-dominated problem characterizes the flow behavior, and thus the full Navier-Stokes equations must be employed. In this thesis we focus on this latter class of problems. Moreover, when the temperature has no significant effect on the fluid flow, the energy equation is uncoupled from the Navier-Stokes equations,

which can therefore be resolved independently [98].

The advection-diffusion problem addressed in the present work is a well-known multiphysics problem (see for instance [99, 100, 101]), composed by Navier-Stokes equations and thermal energy equation that describe the time-dependent heat transfer in a flow of a viscous incompressible fluid as follows:

$$\begin{cases} \partial_t \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} - \nu \Delta \boldsymbol{u} + \nabla p = \boldsymbol{f} \\ \nabla \cdot \boldsymbol{u} = 0 \\ \partial_t T + \boldsymbol{u} \cdot \nabla T - \kappa \Delta T = \gamma \end{cases} \tag{3.1}$$

where t is the time, $\boldsymbol{u}$ represents the velocity vector field of the flow, p the pressure field, ν the kinematic viscosity, $\boldsymbol{f}$ the force term, T the temperature field, κ the diffusivity coefficient, and γ is the heat source term. The system of equations need to be supplemented with initial and boundary conditions to be solved.

The first two equations of the system (3.1) are the Navier-Stokes equations. The former is the momentum conservation equation for a Newtonian fluid, based on the application of Newton's second law. It represents a relation equating the rate of change of momentum of a selected portion of fluid and the sum of all forces acting on that portion of fluid [100]. The latter is the mass-conservation equation, or continuity equation, and it expresses the conservation of the fluid mass contained in a material volume. These two equations together are used to describe the motion of an incompressible fluid flow and, contextualized in the LMD process, they are employed to model the flow of both the carrier gas, which delivers the powder particles, and the shielding gas. The third equation of the system (3.1) is the thermal energy equation, which accounts for the energy balance of the fluid. It represents the temperature field in the LMD process, whose high gradients are generated by the laser beam source and the melt pool thermal features.

Regarding the powder modeling, a *multiphase flow*, that describes the motion of particles in fluids, must be considered. Multiphase flows are modeled by equations in which one phase, the dispersed phase, is not materially connected, such as gas-particle and liquid-particle flows, where the particles constitute precisely the dispersed phase. In general, particle-fluid interaction refers to a process in which the information of both fluid and particles properties are exchanged between them, causing the coupling of the two phases [102]. More precisely, several coupling methods are possible: i) the *one-way coupling*, where the fluid phase affects the dispersed phase with no reverse effect, and ii) the *two-way coupling*, where a mutual effect between the two phases take place. In this model the one-way coupling is employed.

The coupling happens through mass, momentum and energy information transfer. In particular, the subtraction of mass caused by the evaporation of the particle surface is an example of mass coupling. Momentum coupling is instead the application of the Newton's second law on the particle dynamic, and it is described by the following conservation equation

$$m_p \frac{d\boldsymbol{u}_p}{dt} = \sum_i F_i \tag{3.2}$$

where m_p is the mass of the particle, $\boldsymbol{u}_p$ is the particle velocity and F_i are all the forces acting on the particle, subdivided into body forces and surface forces. Body forces act on the mass of the particle, such as gravity, while surface forces is due, for example, to drag, which represent a momentum coupling between the two phases.

Instead, the energy coupling, which occurs through heat transfer between phases, is governed by

$$c_p m_p \frac{dT_p}{dt} = \sum_j H_j \tag{3.3}$$

where c_p is the specific heat of the particle, T_p is the temperature of the particle at the time t and H_j represent all the terms that include phenomena such as energy absorption, internal conduction, external convection and radiation.

In the end, the current literature review shows that several numerical strategies have been implemented to aid the design of LMD set-ups, leading to possible optimal conditions of minimizing thermal gradients and speeding up the whole deposition process [103], but a comprehensive numerical tool, modeling the various multiphysics phenomena involved in fluid-particles interaction, still lacks.

From this perspective, the presented model is able to describe the multiphase and multiphysics flow that occur in the LMD process including both fluid-particles and particle-laser interactions. The system of Eulerian equations (3.1), which represents the velocity, pressure and temperature field of the fluid, provides the necessary information to model both the dynamic and the energy conservation of particles, described by the Lagrangian approaches (3.2) and (3.3), respectively. All details regarding the above mentioned equations are reported and discussed in the following chapter.

Computational modeling and numerical schemes

4.1 Navier-Stokes equations: the role of the pressure

Several numerical methods have been developed for the approximation of the Navier-Stokes equations, and among these, we consider the mixed Finite Element Method (FEM) to discretize the incompressible flow problem [104, 105]. This method, implemented for fluids, is based on the weak form Galerkin formulation [106, 107], using basic physical variables, namely, velocities and pressure. In particular, the Galerkin formulation of the Navier–Stokes system has to minimize a constrained problem, due to incompressibility condition of the fluid [98]. Hence, the approximations used for velocity and pressure field must satisfy certain restrictive conditions, that are discussed and analyzed in the forthcoming sections.

4.1.1 Incompressible flow problem

Navier-Stokes equations describe the motion of a fluid with constant density ρ in a domain $\Omega \subset \mathbb{R}^d$ (with $d = 2, 3$). These are written in the following form

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\, \boldsymbol{u} - \nabla \cdot \left[\nu \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^\top\right)\right] + \nabla p = \boldsymbol{f}, \quad \forall \boldsymbol{x} \in \Omega, t > 0, \tag{4.4}$$

$$\nabla \cdot \boldsymbol{u} = 0, \qquad \forall \boldsymbol{x} \in \Omega, t > 0, \tag{4.5}$$

where $\boldsymbol{u}$ is the fluid velocity, ρ the fluid density, p the fluid pressure divided by the density (that we simply call "pressure"), $\nu = \mu/\rho$ the kinematic viscosity, μ the dynamic viscosity and $\boldsymbol{f}$ the force term per unit mass. The first equation in the system is the momentum balance equation (4.4), the second is the mass conservation equation (4.5), also called the continuity equation. The term $(\boldsymbol{u} \cdot \nabla)\, \boldsymbol{u}$ describes the convective transport process, while $-\nabla \cdot \left[\nu \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^\top\right)\right]$ describes the molecular diffusion process. In the case where ν is constant, using the continuity equation, we get

$$\nabla \cdot \left[\nu \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^\top\right)\right] = \nu \left(\Delta \boldsymbol{u} + \nabla \boldsymbol{u}\right) = \nu \Delta \boldsymbol{u} \tag{4.6}$$

and the system can be written in the compact form

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\, \boldsymbol{u} - \nu \Delta \boldsymbol{u} + \nabla p = \boldsymbol{f}, \qquad \forall \boldsymbol{x} \in \Omega, t > 0, \tag{4.7}$$

$$\nabla \cdot \boldsymbol{u} = 0, \qquad \forall \boldsymbol{x} \in \Omega, t > 0. \tag{4.8}$$

Equations (4.7) and (4.8) are often called incompressible Navier-Stokes equations. More generally, fluids that satisfy the *incompressibility constraint* $\nabla \cdot \boldsymbol{u} = 0$, are called incompressible fluids.

In order for the problem to be well posed it is necessary to assign initial conditions

$$\boldsymbol{u}(\boldsymbol{x}, 0) = \boldsymbol{u}_0(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \Omega \tag{4.9}$$

together with suitable boundary conditions, for example,

$$\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{u}_D(\boldsymbol{x}, t) \qquad \forall \boldsymbol{x} \in \Gamma_D, \tag{4.10}$$

$$\left(\nu \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} - p\boldsymbol{n}\right)(\boldsymbol{x}, t) = \boldsymbol{t}(\boldsymbol{x}, t) \qquad \forall \boldsymbol{x} \in \Gamma_N, \tag{4.11}$$

where $\boldsymbol{u}_0$, $\boldsymbol{u}_D$ and $\boldsymbol{t}$ are assigned vector functions, whereas Γ_D e Γ_N are two disjoint subsets of the total boundary $\partial\Omega$ of Ω such that $\Gamma_D \cup \Gamma_N = \partial\Omega$, $\Gamma_D \cap \Gamma_N = \emptyset$ and $\boldsymbol{n}$ is the outer normal unit vector from $\partial\Omega$.

We wrote the Navier-Stokes equations in terms of the so-called primitive variables $\boldsymbol{u}$ and p, but it is possible to express these equations in terms of other variables, such as the stream function and vorticity. For example, in the two-dimensional case, we could write the momentum balance equation (4.7) in terms of the velocity $\boldsymbol{u}$ and vorticity ω as

$$\frac{\partial \omega}{\partial t} + (\boldsymbol{u} \cdot \nabla)\omega = \nu \Delta \omega \tag{4.12}$$

where the vorticity ω is a scalar function defined by

$$\omega = \mathrm{rot}\boldsymbol{u} = \frac{\partial u_2}{\partial x_1} - \frac{\partial u_1}{\partial x_2} \tag{4.13}$$

Notice that in deriving (4.12) the pressure term has been eliminated, because of the difficulties associated with this scalar variable in an incompressible flow. Moreover, a stream function ψ may be defined such that the continuity equation is identically satisfied. Using both the vorticity and stream function definitions, the Navier-Stokes equations can be expressed as the pair consisting of (4.12) and

$$\Delta \psi = -\omega \tag{4.14}$$

where the stream function is defined as

$$u_1 = -\frac{\partial \psi}{\partial x_2}, \qquad u_2 = -\frac{\partial \psi}{\partial x_1} \tag{4.15}$$

If required, the pressure can be derived form the following Poisson equation

$$\Delta p = \nabla \cdot (\boldsymbol{f} + \nu \Delta \boldsymbol{u} - \boldsymbol{u} \cdot \nabla \boldsymbol{u}) \tag{4.16}$$

which is obtained by taking the divergence of the momentum balance equation.

Since the original equations are computationally convenient compared to this alternative formulation, especially as regards the enforcement of boundary conditions, we will mainly deal with the Navier-Stokes equations written in the form (4.7) and (4.8).

However, obtaining an acceptable solution of the discrete Navier-Stokes system is generally difficult and, before introducing the finite element technique for the discretization of equations, we wish to discuss the numerical difficulties that have their origins in the physical description of the viscous, incompressible flow problem.

The presence of the nonlinear and non-symmetric convective term in the momentum balance equation is the first source of numerical difficulty, which increases with the value of the Reynolds number, defined as

$$Re = \frac{\rho \boldsymbol{u} L}{\mu} \tag{4.17}$$

where L is a characteristic length. In fact in convection dominated flow, characterized by high Reynold numbers, the standard Galerkin formulation become unstable and some stabilization techniques, such as SUPG, GLS, SGS or others, are usually employed [60, 108, 109, 110].

Another numerical complication related to the Navier-Stokes system is the incompressibility constraint, imposed by the continuity equation (4.8), which ensures the velocity field to be divergence free. In this context, the pressure scalar variable in the momentum balance equation represents the additional degree of freedom needed to satisfy the incompressibility condition. In fact, the special role that pressure plays in incompressible flows, is to adjust itself instantaneously in order to satisfy the condition of divergence-free velocity and thus, can be interpreted as a Lagrange multiplier that both enforces the continuity constraint on the velocity field and couples the velocity and pressure unknowns [111, 100].

In literature has been shown various formulation able to handle incompressible flow problems, such as penalty methods [112], but in this work we deal with the primitive variable formulation, which considers both velocity and pressure as unknowns, and leads to the so-called mixed finite element method [113, 60]. Recall that pressure acts as a Lagrangian multiplier of the incompressibility condition, such method presents some numerical difficulties caused by the saddle-point nature of the resulting variational problem. In fact, the system of equations that comes out from the Galerkin discretization is formed by a partitioned matrix with a null submatrix on the diagonal, and moreover, a proper choice of finite element spaces for velocity and pressure, that have to satisfy the Lady*ž*enskaja-Babu*š*ka-Brezzi (LBB) condition [114, 115, 116], is crucial for the solvability of the algebraic system. The LBB condition, a particular instance of the so-called discrete inf-sup condition, is necessary and sufficient to well-pose the discrete saddle point problems arising from discretization via Galerkin method [117], and suggest to use an higher order element for velocity than pressure. This type of elements belong to the Taylor-Hood family, as reported in [118], and are used to prevent an overconstrained system of discrete equations; the interpolation used for pressure (P1, for exemple) must be at least one order lower than that used for the velocity field (P2). However, in order to circumvent the LBB restrictions, alternative finite element formulations have been proposed and demonstrated [119, 120, 121], but these are not considered in this thesis. Hence, suitable finite element spaces for velocity and pressure will be delineate in the next section in order to develop a of finite element model for incompressible flows.

4.1.2 Weak formulation and spatial discretization

To get the the weak form of Navier-Stokes system we need to introduce classes of functions for both the velocity and pressure field [122]. Regarding the velocity field $\boldsymbol{u}$, the space of trial solutions is denoted by $\mathcal{S}$, and its functions have to satisfy a priori Dirichlet boundary conditions on Γ_D. Hence the trial solution space $\mathcal{S}$ can be defined by

$$\mathcal{S} := \left\{ \boldsymbol{u} \in \mathcal{H}^1(\Omega) \,|\boldsymbol{u} = \boldsymbol{u}_D \,\text{on}\, \Gamma_D \right\}. \tag{4.18}$$

The corresponding test functions $\boldsymbol{v}$ of the velocity field, belong to the space $\mathcal{V}$, which have the same characteristics as those in $\mathcal{S}$ but vanish on Γ_D, that is where the velocity is prescribed. The class $\mathcal{V}$ is thus symbolically characterized as follows

$$\mathcal{V} := \mathcal{H}^1_{\Gamma_D}(\Omega) = \left\{ \boldsymbol{v} \in \mathcal{H}^1(\Omega) \,|\boldsymbol{v} = 0 \,\text{on}\, \Gamma_D \right\}, \tag{4.19}$$

and we can observe the following relationship

$$\mathcal{S} = \mathcal{V} + \{\bar{\boldsymbol{u}}_D\} \tag{4.20}$$

where $\bar{\boldsymbol{u}}_D$ is any function in $\mathcal{H}^1(\Omega)$ such that $\bar{\boldsymbol{u}}_D = \boldsymbol{u}_D$ on Γ_D. Then, we introduce the space of functions $\mathcal{Q}$ for the pressure field, that is required to be square-integrable, because spatial derivatives of pressure are absent from the weak form of the Navier-Stokes system of equations. Before writing the definition of the space $\mathcal{Q}$, some consideration about the pressure variable must be addressed. If $\Gamma_D = \partial\Omega$, only Dirichlet boundary conditions are considered, and pressure is defined up to a constant; this means that if $(\boldsymbol{u}, p)$ is solution of (4.7) and (4.8), it is also $(\boldsymbol{u}, p + c)$, where c is a constant, since $\nabla(p + c) = \nabla p$. To avoid this indeterminacy we can impose that the average pressure value is zero or fix the value of p at a point $\boldsymbol{x}_0$ of the domain. On the other hand, if Γ_D is not empty, and then Neumann-type boundary conditions are also considered, there are no more problems about the pressure uncertainty, due to the condition on Γ_D where spatial derivatives of p do not appear. Hence, we can consider

$$\mathcal{Q} = \mathcal{L}^2(\Omega) \text{ if } \Gamma_D \neq \emptyset, \qquad \mathcal{Q} = \mathcal{L}^2_0(\Omega) \text{ if } \Gamma_D = \emptyset. \tag{4.21}$$

where both the trial solution space and the weighting function space for the pressure field are contained in $\mathcal{Q}$.

In order to obtain a weak formulation of the problem (4.7) - (4.8), we formally proceed multiplying (4.7) by the specified test function $\boldsymbol{v}$, belonging to the suitable space $\mathcal{V}$, and integrate it on Ω:

$$\int_\Omega \frac{\partial \boldsymbol{u}}{\partial t} \cdot \boldsymbol{v}\, d\Omega - \int_\Omega \nu \Delta \boldsymbol{u} \cdot \boldsymbol{v}\, d\Omega + \int_\Omega \left[(\boldsymbol{u} \cdot \nabla)\, \boldsymbol{u} \right] \cdot \boldsymbol{v}\, d\Omega \int_\Omega \nabla p \cdot \boldsymbol{v}\, d\Omega = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v}\, d\Omega \tag{4.22}$$

Using the Green's formula we can write

$$-\int_\Omega \nu \Delta \boldsymbol{u} \cdot \boldsymbol{v}\, d\Omega = \int_\Omega \nu \nabla \boldsymbol{u} \cdot \nabla \boldsymbol{v}\, d\Omega - \int_{\partial\Omega} \nu \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} \cdot \partial \boldsymbol{v}\, d\Gamma \tag{4.23}$$

$$\int_\Omega \nabla p \cdot \boldsymbol{v}\, d\Omega = -\int_\Omega p\, \nabla \cdot \boldsymbol{v}\, d\Omega + \int_{\partial\Omega} p \boldsymbol{v} \cdot \boldsymbol{n}\, d\Gamma \tag{4.24}$$

and substituting these into (4.7),

$$\begin{aligned}&\int_\Omega \frac{\partial \boldsymbol{u}}{\partial t}\cdot \boldsymbol{v}\, d\Omega + \int_\Omega \nu \nabla \boldsymbol{u}\cdot\nabla \boldsymbol{v}\, d\Omega + \int_\Omega \left[(\boldsymbol{u}\cdot\nabla)\,\boldsymbol{u}\right]\cdot \boldsymbol{v}\, d\Omega - \int_\Omega p\,\nabla\cdot \boldsymbol{v}\, d\Omega \\ &= \int_\Omega \boldsymbol{f}\cdot \boldsymbol{v}\, d\Omega + \int_{\partial\Omega}\left(\nu\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} - p\boldsymbol{n}\right)\cdot \boldsymbol{v}\, d\Gamma \qquad \forall \boldsymbol{v}\in\mathcal{V}.\end{aligned} \tag{4.25}$$

Similarly we can multiply (4.8) by a test function q, belonging to the space $\mathcal{Q}$, integrating on Ω and obtain

$$\int_\Omega q\,\nabla\cdot \boldsymbol{u}\, d\Omega = 0 \qquad \forall\, q\in\mathcal{Q}. \tag{4.26}$$

Finally, the weak formulation of equations (4.7), (4.8), (4.9), (4.10) and (4.11) is therefore: for every $t>0$, given $\boldsymbol{f}$, $\boldsymbol{u}_D$, $\boldsymbol{t}$, and $\boldsymbol{u}_0$, find $\boldsymbol{u}(\boldsymbol{x},t)\in\mathcal{S}$ and $p(\boldsymbol{x},t)\in\mathcal{Q}$, such that

$$\begin{aligned}&\int_\Omega \frac{\partial \boldsymbol{u}}{\partial t}\cdot \boldsymbol{v}\, d\Omega + \nu\int_\Omega \nabla \boldsymbol{u}\cdot\nabla \boldsymbol{v}\, d\Omega + \int_\Omega \left[(\boldsymbol{u}\cdot\nabla)\,\boldsymbol{u}\right]\cdot \boldsymbol{v}\, d\Omega - \int_\Omega p\,\nabla\cdot \boldsymbol{v}\, d\Omega \\ &= \int_\Omega \boldsymbol{f}\cdot \boldsymbol{v}\, d\Omega + \int_{\Gamma_N} \boldsymbol{t}\cdot \boldsymbol{v}\, d\Gamma \qquad \forall \boldsymbol{v}\in\mathcal{V} \\ &\\ &\int_\Omega q\,\nabla\cdot \boldsymbol{u}\, d\Omega = 0 \qquad \forall\, q\in\mathcal{Q}.\end{aligned} \tag{4.27}$$

with $\boldsymbol{u}(0)=\boldsymbol{u}_0$. The variational problem (4.27) can be rewritten in compact form in terms of velocity and pressure using the inner product notation $(\cdot,\cdot)$; find $\boldsymbol{u}(\boldsymbol{x},t)\in\mathcal{S}$ and $p(\boldsymbol{x},t)\in\mathcal{Q}$, for all $\boldsymbol{v}\in\mathcal{V}$ and $q\in\mathcal{Q}$, such that

$$(\boldsymbol{v},\boldsymbol{u}_t) + a(\boldsymbol{v},\boldsymbol{u}) + c(\boldsymbol{u};\boldsymbol{v},\boldsymbol{u}) + b(\boldsymbol{v},p) = (\boldsymbol{v},\boldsymbol{f}) + (\boldsymbol{v},\boldsymbol{t})_{\Gamma_N}, \tag{4.28}$$

$$b(\boldsymbol{u},q) = 0 \tag{4.29}$$

with the following definitions of the forms,

$$a(\boldsymbol{v},\boldsymbol{u}) = \nu\int_\Omega \nabla\boldsymbol{u}\cdot\nabla\boldsymbol{v}\, d\Omega, \qquad c(\boldsymbol{u};\boldsymbol{v},\boldsymbol{u}) = \int_\Omega \left[(\boldsymbol{u}\cdot\nabla)\,\boldsymbol{u}\right]\cdot\boldsymbol{v}\, d\Omega, \tag{4.30}$$

$$b(\boldsymbol{v},p) = -\int_\Omega p\,\nabla\cdot\boldsymbol{v}\, d\Omega, \qquad b(\boldsymbol{u},q) = \int_\Omega q\,\nabla\cdot\boldsymbol{u}\, d\Omega. \tag{4.31}$$

Now, we perform the spatial discretization using the Galerkin formulation of the Navier-Stokes problem that leads to a mixed Finite Element Method. At this point, we have to view the domain Ω as discretized into element domains. Let $\mathcal{T}^h(\Omega)$ be a regular partition, also called triangulation, of Ω into n_{el} convex subdomains $\Omega^e\neq\emptyset$, such that

$$\Omega = \bigcup_{e=1}^{n_{el}} \Omega^e \tag{4.32}$$

Each subdomain Ω^e has a piecewise smooth boundary $\Gamma^e=\partial\Omega^e$, and h is a characteristic mesh size. Then we need to introduce local approximations for both the velocity $\boldsymbol{u}^h$ and pressure p^h, as well as for their associated weighting functions $\boldsymbol{v}^h$ and q^h. Let $\mathcal{S}^h$ and $\mathcal{V}^h$ be finite dimensional

spaces' subsets of $\mathcal{S}$ and $\mathcal{V}$, and $\mathcal{Q}^h$ the finite dimensional subspace of $\mathcal{Q}$ respectively defined as

$$\mathcal{S}^h = \left\{ \boldsymbol{u}^h \in \mathcal{H}^1(\Omega) \,|\boldsymbol{u}^h|_{\Omega^e} \in \mathcal{P}_m(\Omega) \,\forall\, e \text{ and } \boldsymbol{u}^h = \boldsymbol{u}_D \text{ on } \Gamma_D \right\} \tag{4.33}$$

$$\mathcal{V}^h = \left\{ \boldsymbol{v}^h \in \mathcal{H}^1(\Omega) \,|\boldsymbol{v}^h|_{\Omega^e} \in \mathcal{P}_m(\Omega) \,\forall\, e \text{ and } \boldsymbol{v}^h = 0 \text{ on } \Gamma_D \right\} \tag{4.34}$$

$$\mathcal{Q}^h = \left\{ q^h \in \mathcal{L}^2(\Omega) \,|q^h|_{\Omega^e} \in \mathcal{P}_l(\Omega) \,\forall\, e \right\} \tag{4.35}$$

where $\mathcal{P}$ is the finite element interpolating space of polynomial functions of degree $m \geq 1$ and $l \geq 0$ $(m-1 \leq l \leq m)$.

The next step in the Galerkin formulation consists in approximating the velocity and pressure field, interpolating the continuous values in the discrete spaces mentioned above. We denote the number of global velocity node in the finite element mesh with $\eta = \{1, 2, \ldots n_{np}\}$, where n_{np} is the number of the nodal points, and we define by $\eta_D \subset \eta$ the subset of velocity nodes belonging to the Dirichlet portion of the boundary where the velocity is prescribed. The velocity is then approximated as follow

$$\boldsymbol{u}^h(\boldsymbol{x}) = \sum_{A \in \eta \setminus \eta_D} N_A(\boldsymbol{x}) \boldsymbol{u}_A + \sum_{A \in \eta_D} N_A(\boldsymbol{x}) \boldsymbol{u}_D(\boldsymbol{x}_A) \tag{4.36}$$

where $\boldsymbol{u}_A$ is the value of $\boldsymbol{u}$ at node number A and, N_A is the shape function (see [123, 124] for more details) associated with global node number A. In the Galerkin spatial discretization the test functions are defined such that

$$\boldsymbol{v}^h \in \mathcal{V}^h := \text{span}\{N_A\} \tag{4.37}$$

The pressure field is interpolated using a possibly different set of pressure nodes denoted by $\hat{\eta}$ and the shape functions $\hat{N}_{\hat{A}}$, as

$$p^h(\boldsymbol{x}) = \sum_{\hat{A} \in \hat{\eta}} \hat{N}_{\hat{A}}(\boldsymbol{x}) p_{\hat{A}} \tag{4.38}$$

where $\hat{A}$ is the global pressure node number and $p_{\hat{A}}$ is the pressure value at $\hat{A}$. Similarly, the weighting function q^h for the pressure is expressed as

$$q^h \in \mathcal{Q}^h := \text{span}\left\{ \hat{N}_{\hat{A}} \right\} \tag{4.39}$$

Upper-case letters, such as A and B, are used to represent global node numbers in the finite element mesh: $1 < A, B < n_{np}$, whereas lower-case letters, such as a and b, will be used to represent local node numbers in an element: $1 < a, b < n_{en}$, where n_{en} is the number of element nodes.

Then the Galerkin spatial discretization of the Navier-Stokes time-dependent system proceeds as follow. For each $t > 0$, we seek the velocity field $\boldsymbol{u}^h(\boldsymbol{x}, t) \in \mathcal{S}^h$ and pressure $p(\boldsymbol{x}, t) \in \mathcal{Q}^h$, such that, for all $(\boldsymbol{v}, q) \in \mathcal{V}^h \times \mathcal{Q}^h$,

$$\begin{aligned} &\left(\boldsymbol{v}^h, \boldsymbol{u}_t^h\right) + a\left(\boldsymbol{v}^h, \boldsymbol{u}^h\right) + c\left(\boldsymbol{u}^h; \boldsymbol{v}^h, \boldsymbol{u}^h\right) + b\left(\boldsymbol{v}^h, p^h\right) = \left(\boldsymbol{v}^h, \boldsymbol{f}^h\right) + \left(\boldsymbol{v}^h, \boldsymbol{t}^t\right)_{\Gamma_N} \\ &b\left(\boldsymbol{u}^h, q^h\right) = 0 \end{aligned} \tag{4.40}$$

with $\boldsymbol{u}^h(0) = \boldsymbol{u}_0^h$. Finally, the finite element discretization of this weak form yields the system of semi-discrete equations for $t > 0$

$$\begin{aligned} &\boldsymbol{M}\dot{\boldsymbol{u}}(t) + \left[\boldsymbol{K} + \boldsymbol{C}\left(\boldsymbol{u}(t)\right)\right]\boldsymbol{u}(t) + \boldsymbol{G}\boldsymbol{p}(t) = \boldsymbol{f}\left(t\right) \\ &\boldsymbol{G}^\top \boldsymbol{u}(t) = 0 \end{aligned} \tag{4.41}$$

where the coefficient matrices shown in (4.41) can be represented through the action of an assembly operator $\mathop{\mathrm{A}}$ acting on the local element matrix and nodal vector as follows:

$$\boldsymbol{M} = \mathop{\mathrm{A}}_{e}^{e} \boldsymbol{M}^e \qquad \boldsymbol{M}^e = \int_{\Omega^e} N_a N_b \, d\Omega \tag{4.42}$$

$$\boldsymbol{K} = \mathop{\mathrm{A}}_{e}^{e} \boldsymbol{K}^e \qquad \boldsymbol{K}^e = \nu \int_{\Omega^e} \nabla N_a \nabla N_b \, d\Omega \tag{4.43}$$

$$\boldsymbol{C} = \mathop{\mathrm{A}}_{e}^{e} \boldsymbol{C}^e \qquad \boldsymbol{C}^e = \int_{\Omega^e} N_a \left(\boldsymbol{u}^h \nabla N_b\right) \, d\Omega \tag{4.44}$$

$$\boldsymbol{G} = \mathop{\mathrm{A}}_{e}^{e} \boldsymbol{G}^e \qquad \boldsymbol{G}^e = -\int_{\Omega^e} \hat{N}_{\hat{a}} \nabla \cdot N_a \, d\Omega \tag{4.45}$$

$$\boldsymbol{f} = \mathop{\mathrm{A}}_{e}^{e} \boldsymbol{f}^e \qquad \boldsymbol{f}^e = \int_{\Omega^e} N_a \boldsymbol{f} \, d\Omega + \int_{\Gamma_N^e} N_a \boldsymbol{t} \, d\Gamma_N^e \tag{4.46}$$

that in matrix form can be written as

$$\begin{bmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{u}} \\ \dot{p} \end{bmatrix} + \begin{bmatrix} \mathbf{K} + \mathbf{C}(\mathbf{u}) & \mathbf{G} \\ \mathbf{G}^\top & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{0} \end{bmatrix} \tag{4.47}$$

or in a more symbolic format as

$$\bar{\mathbf{M}}\dot{\mathbf{x}} + \bar{\mathbf{K}}\left(\mathbf{u}\right)\mathbf{x} = \bar{\mathbf{f}} \tag{4.48}$$

where

$$\mathbf{x} = \begin{bmatrix} \mathbf{u} & p \end{bmatrix}^\top, \qquad \bar{\mathbf{M}} = \begin{bmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \qquad \bar{\mathbf{K}} = \begin{bmatrix} \mathbf{K} + \mathbf{C}(\mathbf{u}) & \mathbf{G} \\ \mathbf{G}^\top & \mathbf{0} \end{bmatrix} \tag{4.49}$$

Each term of the given conservation equations in matrix form represent a particular physical process and it is important to recognize its basic features that heavily influence the choice of a solution procedure: $\mathbf{M}$ is the mass , $\mathbf{K}$ is the viscous diffusion term, $\mathbf{C}$ is the convective transport term, $\mathbf{Q}$ is the pressure gradient operator, $\mathbf{Q}^\top$ is the divergence operator and $\mathbf{f}$ contains both body and surface forces. An examination of the system (4.47) indicate that both matrices $\mathbf{M}$ and $\mathbf{K}$ are symmetric, whereas $\mathbf{C}$ is non-symmetric, making $\bar{\mathbf{K}}$ non-symmetric, and thus, a non-symmetric system must be handle to find the solution of general flows. On the other hand, when material properties are constant and the Reynolds number is sufficiently small the system of equations becomes linear and symmetric thanks to the Stokes approximation.

As already mentioned at the beginning of this chapter, the presence of zeros on the matrix diagonals (see (4.47)), corresponding to the pressure variable, that not appear explicitly in the continuity equation, add some troubles related to the solution research in the mixed

finite element model. In fact, some type of pivoting strategy must be used if direct solver are employed, while poor convergence rates affect the use of iterative solvers, mainly due to the divergence-free condition. Other technical problems associated with the incompressibility condition concern, for example, the impossibility of impulsively started flows, and thus proper boundary conditions and suitable initial conditions for time-dependent computations have to be imposed to correctly implement the computational problem. Again, the special role that pressure plays in incompressible flows, also influence the choice of interpolation functions in the mixed finite element model. In fact, in order to satisfy the LBB conditions and, thus, prevent an overconstrained system, the interpolation functions used for pressure must be at least one order lower than that used for the velocity field. Moreover, the fact that the pressure is not a primary variable of the weak form it doesn't have to be continuous across elements.

4.2 Advection-diffusion equations: the necessity to stabilize

In literature many authors deal with the treatment of the Navier-Stokes equations or the heat equation. The momentum equation of the Navier-Stokes system and the heat equation are both represented with the advection-diffusion equation. The main difference is that, the former is a nonlinear vector value problem, while the latter is a linear scalar problem. Sometimes, when the velocity field increase, some oscillations affect the advection-diffusion equation, and rise up the need to stabilize the solution [110]. In fact, as mentioned in the previous section, the convective transport matrix $\mathbf{C}$, assembled through the Galerkin finite element discretization, is non-symmetric in fluid flow or heat transfer problems, and this can lead to spurious oscillations on the solution field. Typically these oscillations, or "wiggles", appear in case of high Peclet or Reynolds number, that is when the convection term dominates the flow behavior and downstream boundary condition forces a rapid change in the solution [60]. A significant mesh refinement is the only way to avoid the formation of wiggles, such that the convection term no longer dominates on an element level, but this choice is quite unpractical and an alternative approach is required.

This inconvenient is due to the central-difference type approximations of differential operators in the Galerkin finite element method, and in fact, oscillations problems also afflict central-difference finite difference solutions. However, it has been seen that using the upwind differencing technique, where the convective derivatives are approximated with solution values at the upstream and central nodes of a three-node stencil, the wiggles disappear from the solution in convection dominated flow problems. Thus, the difficulties related to the outflow boundary condition on the convective transport term can be removed considering the convective information only from upstream. However, upwind differences are only first-order accurate, while central differences are second-order accurate, and this leak of accuracy lead to an excessively diffuse solutions. In fact, if the solution obtained using a central difference manifest an underdiffuse behavior, on the other hand, the employment of upwinding has an overdiffuse effect on the solution.

Then, a new optimized method that implement a linear combination of central and upwind differences, based on Peclet or Reynolds number, it was found to be a better way rather than using upwind or central differences alone. Therefore, the construction of the upwinded convective term is performed simply by adding the proper amount of artificial diffusion to the central difference method. It is important to notice that the artificial diffusion is added in order to counterbalance the negative numerical diffusion introduced by the Galerkin approximation, and it is not relative to the actual physics of the problem. Using this technique the solution is nodally exact and this proves that the central-difference method is effectively affected by negative artificial diffusion. Apart from the use of the artificial diffusion technique, it is possible to achieve the upwind effect in the finite element method in other ways. Hughes [125] proposed a modified numerical quadrature rule for the convection term to achieve the upwind

effect, whereas weighting more heavily the element upwind of the node rather than the one downwind via Petrov-Galerkin weighting function was employed in [126, 127, 128, 129]. All the upwind finite element formulations described give the same system of matrix equations and exact nodal solution, but when generalized to more complicated situations, such as in multidimensional and transient cases, these methods give results much worse than those obtained by Galerkin's method. Unfortunately, in these cases, a spurious crosswind diffusion effect appear on the solution and an excessive diffusion perpendicular to the flow, when the flow direction is skew to the mesh, make the results unacceptable.

To overcome these problems the "streamline upwind Petrov-Galerkin method" (SU/PG) was presented in details in [60], where the upwind formulation is constructed through a proper choice of the Petrov-Galerkin weighting functions. This new innovative method keep all the robust qualities of a classical upwind formulation without introducing any of the issues related to the crosswind artificial diffusion. In particular, the spirit of the SU/PG method is to include artificial diffusion only in the flow direction, extending the standard Galerkin weighting functions to a Petrov-Galerkin formulation and adding a streamline upwind perturbation only in the flow direction. All terms of the equation are involved in this method, and thus a consistent weighted residual formulation is achieved. Obviously, the SUPG method have attracted considerable interest and have been widely utilized in both physical and engineering applications, especially with high Reynols and Peclet numbers, because it allows to obtain reliable results without instabilities and spurious oscillations.

Many of the stabilized methods, such as SU/PG, that are used in advection diffusion equations can be interpreted as a subclass of the largest variational multiscale method (VMS). In fact, before discussing the mathematical framework of the SU/PG method and its application to the heat equation, it is useful to briefly mention the variational multiscale method, which provides the theory behind the stabilized methods and helps to understand the basics and the implementation of such formulations.

4.2.1 Variational Multiscale method

The variational multiscale method (see Hughes [130] for a detailed explanation) provides the necessary mathematical basics for the construction of the SU/PG method and is based on the decomposition of the solution $\boldsymbol{u} = \bar{\boldsymbol{u}} + \boldsymbol{u}'$ where $\bar{\boldsymbol{u}}$ represents the coarse-scale component and $\boldsymbol{u}'$ is the fine-scale component. In this framework, the coarse-scale component $\bar{\boldsymbol{u}}$ is solved numerically by the standard finite element approximation, whereas the fine-scale component $\boldsymbol{u}'$ is determined analitically. It is worth noting that fine-scales solution $\boldsymbol{u}'$ effectively represents the error $\boldsymbol{u} - \bar{\boldsymbol{u}}$ of coarse-scales, while $\bar{\boldsymbol{u}}$ is the resolvable scale approximated by the finite element method.

The VMS method will be illustrated in the context of advection-diffusion equation, with homogeneous Dirichlet boundary conditions, in a simplified form. Hence, starting from the

standard weak form of this boundary value problem, we have to find $\boldsymbol{u} \in \mathcal{V}$ such that

$$a(\boldsymbol{v}, \boldsymbol{u}) + c(\boldsymbol{a}; \boldsymbol{v}, \boldsymbol{u}) = (\boldsymbol{v}, \boldsymbol{f}) \qquad \forall\, \boldsymbol{v} \in \mathcal{V}, \tag{4.50}$$

where $\boldsymbol{a}$ is a generic advection vector field.

Now we decompose the solution $\boldsymbol{u} = \bar{\boldsymbol{u}} + \boldsymbol{u}'$, the test function $\boldsymbol{v} = \bar{\boldsymbol{v}} + \boldsymbol{v}'$, and the space $\mathcal{V} = \bar{\mathcal{V}} \oplus \mathcal{V}'$ into a finite dimensional coarse-scale subspace and an infinite dimensional fine-scale subspace, respectively. Then, we can write the weak form as

$$a(\bar{\boldsymbol{v}} + \boldsymbol{v}', \bar{\boldsymbol{u}} + \boldsymbol{u}') + c(\boldsymbol{a}; \bar{\boldsymbol{v}} + \boldsymbol{v}', \bar{\boldsymbol{u}} + \boldsymbol{u}') = (\bar{\boldsymbol{v}} + \boldsymbol{v}', \boldsymbol{f}), \tag{4.51}$$

which, because of the linear independence of $\bar{\boldsymbol{v}}$ and $\boldsymbol{v}'$, we can split (4.51) into the two following equations:

$$a(\bar{\boldsymbol{v}}, \bar{\boldsymbol{u}}) + c(\boldsymbol{a}; \bar{\boldsymbol{v}}, \bar{\boldsymbol{u}}) + a(\bar{\boldsymbol{v}}, \boldsymbol{u}') + c(\boldsymbol{a}; \bar{\boldsymbol{v}}, \boldsymbol{u}') = (\bar{\boldsymbol{v}}, \boldsymbol{f}), \qquad \forall\, \bar{\boldsymbol{v}} \in \bar{\mathcal{V}}, \tag{4.52}$$

$$a(\boldsymbol{v}', \boldsymbol{u}') + c(\boldsymbol{a}; \boldsymbol{v}', \boldsymbol{u}')\, a(\boldsymbol{v}', \bar{\boldsymbol{u}}) + c(\boldsymbol{a}; \boldsymbol{v}', \bar{\boldsymbol{u}}) = (\boldsymbol{v}', \boldsymbol{f}), \qquad \forall\, \boldsymbol{v}' \in \mathcal{V}', \tag{4.53}$$

where the equation (4.52) governs the resolved scales and the equation (4.53) the unresolvable scales. The method require to solve analytically the fine-scale equation (4.53) to find $\boldsymbol{u}'$ as a function of $\bar{\boldsymbol{u}}$, and then substituting it in (4.52) to obtain an equation for $\bar{\boldsymbol{u}}$. In order to obtain the fine-scale solution the Green's function technique is applied, imposing $\boldsymbol{u}' = 0$ on the finite element borders to apply the fine-scale equation in the interior of each finite element. Hence, the Euler-Lagrange problem related to the fine-scale equation (4.53) is

$$\Pi L(\boldsymbol{u}') = -\Pi\left[L(\bar{\boldsymbol{u}} - \boldsymbol{f})\right] \qquad in\ \Omega^e \tag{4.54}$$

$$\boldsymbol{u}' = 0 \qquad on\ \Gamma^e \tag{4.55}$$

where Π is the $\mathcal{L}_2$ projection onto $\mathcal{V}'$ and $L(\boldsymbol{u}) = \boldsymbol{a} \cdot \nabla \boldsymbol{u} - \nabla \cdot (\nu \nabla \boldsymbol{u})$ is the differential operator associated with the advection—diffusion equation. Looking at (4.54) we can notice that the fine scales $\boldsymbol{u}'$ are driven by the residual of the coarse scales. In terms of the solution of this problem, $\boldsymbol{u}'$ can be written using the algebraic operator τ as

$$\boldsymbol{u}' = -\tau\left[L(\bar{\boldsymbol{u}}) - \boldsymbol{f}\right], \tag{4.56}$$

At this point we note that

$$a(\bar{\boldsymbol{v}}, \boldsymbol{u}') + c(\boldsymbol{a}; \bar{\boldsymbol{v}}, \boldsymbol{u}') = \sum_e \int_{\Omega^e} \left[-\boldsymbol{a} \cdot \nabla \bar{\boldsymbol{v}} - \nabla \cdot (\nu \nabla \bar{\boldsymbol{v}})\right] = \sum_e \left(L^*(\bar{\boldsymbol{v}}), \boldsymbol{u}'\right)_{\Omega^e} \tag{4.57}$$

where the differential operator $L^*(\boldsymbol{u}) = -\boldsymbol{a} \cdot \nabla \boldsymbol{u}$ is introduced and used in the SU/PG stabilization method and it corresponds to the perturbation of the test function of the streamline upwinding method. At this point, we substitute (4.56) in (4.57) and we write the coarse-scale equation (4.52) in the SU/PG stabilized form as

$$a(\bar{\boldsymbol{v}}, \bar{\boldsymbol{u}}) + c(\boldsymbol{a}; \bar{\boldsymbol{v}}, \bar{\boldsymbol{u}}) + \sum_e \tau\left(-L^*(\bar{\boldsymbol{v}}), L(\bar{\boldsymbol{u}}) - \boldsymbol{f}\right)_{\Omega^e} = (\bar{\boldsymbol{v}}, \boldsymbol{f}), \tag{4.58}$$

where the third term in the lhs of the equation represents the SU/PG stabilizing term.

Considering that, after a classical FEM spatial discretization (see Section 4.2.2), the coarse scales are piecewise linear and $L(\bar{\boldsymbol{u}})$ and $L^*(\bar{\boldsymbol{v}})$ are constants on each element, then the operator τ can be defined, likewise using the Green's function on the element, as

$$\tau = \frac{h}{2\boldsymbol{a}} \left(\coth \alpha - \frac{1}{\alpha} \right) \tag{4.59}$$

where h is the element length and α is the element Peclet number defined as

$$\alpha = \frac{\boldsymbol{a} h}{2\nu}. \tag{4.60}$$

Clarified the theory behind stabilized formulations we can proceed to the implementation of the SU/PG method to the heat equation.

4.2.2 Advection-diffusion problem

Following the explanation of Section 4.2.1 we apply the stabilized SU/PG formulation to the energy equation, discretized using the Finite Element Method and written in the temperature variable, as we intend to represent a temperature map in the domain of interest.

The strong form of the advection diffusion equation that describes the heat transfer of a fluid in a domain $\Omega \subset \mathbb{R}^d$ (with $d = 2, 3$) is written in the following form

$$\frac{\partial T}{\partial t} + \boldsymbol{u} \cdot \nabla T - \kappa \Delta T = \gamma \qquad \forall \boldsymbol{x} \in \Omega, t > 0, \tag{4.61}$$

where T represents the scalar temperature field, $\boldsymbol{u}$ the velocity vector field of the flow, κ the diffusivity coefficient, and γ the heat source term. The term $\boldsymbol{u} \cdot \nabla T$ describes the advection of the temperature field, while $-\kappa \Delta T$ describes the temperature diffusion process.
The necessary initial conditions for the boundary value problem are

$$T(\boldsymbol{x}, 0) = T_0(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \Omega \tag{4.62}$$

together with suitable boundary conditions,

$$T(\boldsymbol{x}, t) = T_D(\boldsymbol{x}, t) \qquad \forall \boldsymbol{x} \in \Gamma_D, \tag{4.63}$$

$$\kappa (\boldsymbol{n} \cdot \nabla) T(\boldsymbol{x}, t) = h(\boldsymbol{x}, t) \qquad \forall \boldsymbol{x} \in \Gamma_N, \tag{4.64}$$

where T_0, T_D and h are assigned scalar functions, whereas Γ_D e Γ_N are two disjoint subsets of the total boundary $\partial\Omega$ of Ω such that $\Gamma_D \cup \Gamma_N = \partial\Omega$, $\Gamma_D \cap \Gamma_N = \emptyset$ and $\boldsymbol{n}$ is the outer normal unit vector from $\partial\Omega$.

4.2.3 Streamline Upwind Petrov-Galerkin stabilization

As done for the Navier-Stokes problem, in order to achieve a discretization of the strong form of the advection diffusion equation, the next step is to associate an equivalent weak (or variational)

form to the boundary value problem (4.61), (4.62), (4.63) and (4.64).
Then, we define the trial solution space $\mathcal{T}$, consists of real-valued scalar functions T, that satisfy the Dirichlet condition, as

$$\mathcal{T} := \left\{ T \in \mathcal{H}^1(\Omega) \,|T = T_D \,\text{on}\, \Gamma_D \right\}, \tag{4.65}$$

and the trial solution space $\mathcal{W}$, where its function w vanish on Γ_D, such that

$$\mathcal{W} := \left\{ w \in \mathcal{H}^1(\Omega) \,|w = 0 \,\text{on}\, \Gamma_D \right\}, \tag{4.66}$$

The weak formulation of the advection-diffusion problem then takes the following form: for every $t > 0$, given γ , T_D, h, and T_0, find $T(\boldsymbol{x}, t) \in \mathcal{T}$ such that

$$\int_\Omega w \frac{\partial T}{\partial t}\, d\Omega + \int_\Omega w\,(\boldsymbol{u} \cdot \nabla T)\, d\Omega - \int_\Omega w \kappa \Delta T\, d\Omega = \int_\Omega w\gamma\, d\Omega \qquad \forall w \in \mathcal{W}, \tag{4.67}$$

then, using the Green's formula on the diffusion term and noting that $w = 0$ on Γ_D,

$$\begin{aligned} &\int_\Omega w \frac{\partial T}{\partial t}\, d\Omega + \int_\Omega w\,(\boldsymbol{u} \cdot \nabla T)\, d\Omega + \int_\Omega \kappa \nabla w \nabla T\, d\Omega = \int_\Omega w\gamma\, d\Omega + \int_{\Gamma_N} wh\, d\Gamma \\ &\quad \forall w \in \mathcal{W}. \end{aligned} \tag{4.68}$$

The variational form can be written in a compact version based upon the integral forms defined as

$$(w, T_t) = \int_\Omega w \frac{\partial T}{\partial t}\, d\Omega, \qquad c(\boldsymbol{u}; w, T) = \int_\Omega w\,(\boldsymbol{u} \cdot \nabla T)\, d\Omega, \tag{4.69}$$

$$a(w, T) = \int_\Omega \kappa \nabla w \nabla T\, d\Omega, \qquad (w, \gamma) = \int_\Omega w\gamma\, d\Omega, \tag{4.70}$$

$$(w, h)_{\Gamma_N} = \int_{\Gamma_N} wh\, d\Gamma. \tag{4.71}$$

$$\tag{4.72}$$

Then, the weak equation (4.68) written in a compact form becomes

$$(w, T_t) + c(\boldsymbol{u}; w, T) + a(w, T) = (w, \gamma) + (w, h)_{\Gamma_N} \tag{4.73}$$

At this point, using the SU/PG method described above, in which we test the residual of the differential heat equation, computed only for each element interior Ω^e, with test functions defined as $\tau\, \boldsymbol{u} \cdot \nabla w$, where τ is the stabilization parameter, we can write the following stabilized form

$$\begin{aligned} &(w, T_t) + c(\boldsymbol{u}; w, T) + a(w, T) \\ &+ \sum_e \left(T_t + \boldsymbol{u} \cdot \nabla T - \kappa \Delta T - \gamma, \tau \boldsymbol{u} \cdot \nabla w \right)_{\Omega^e} = (w, \gamma) + (w, h)_{\Gamma_N}\ . \end{aligned} \tag{4.74}$$

where τ is defined in (4.59). For sake of simplification, we will denote

$$(w, T_t)^S = \sum_e (T_t, \tau \boldsymbol{u} \cdot \nabla w)_{\Omega^e} \ , \tag{4.75}$$

$$a\,(w, T)^S = \sum_e (\kappa \Delta T, \tau \boldsymbol{u} \cdot \nabla w)_{\Omega^e} \ , \tag{4.76}$$

$$c\,(\boldsymbol{u}; w, T)^S = \sum_e (\boldsymbol{u} \cdot \nabla T, \tau \boldsymbol{u} \cdot \nabla w)_{\Omega^e} \ , \tag{4.77}$$

$$(w, \gamma)^S = \sum_e (\gamma, \tau \boldsymbol{u} \cdot \nabla w)_{\Omega^e} \ . \tag{4.78}$$

$$\tag{4.79}$$

so that equation (4.74) may be rewritten as

$$\begin{aligned} &(w, T_t) + c\,(\boldsymbol{u}; w, T) + a\,(w, T) \\ &+ (w, T_t)^S + c\,(\boldsymbol{u}; w, T)^S - a\,(w, T)^S - (w, \gamma)^S \\ &= (w, \gamma) + (w, h)_{\Gamma_N} \,. \end{aligned} \tag{4.80}$$

Let us introduce the approximated spaces $\mathcal{T}^h$ and $\mathcal{W}^h$ subsets of $\mathcal{T}$ and $\mathcal{W}$, and their respective trial functions $T^h \in \mathcal{T}^h$ and test functions $w^h \in \mathcal{W}^h$. The former satisfies the boundary condition T_D on Γ_D, while the latter vanish on Γ_D. Then, the Galerkin formulation obtained by restricting the weak form to the finite dimensional spaces, leads to find, for each $t > 0$, $T^h \in \mathcal{T}^h$ such that

$$\begin{aligned} &\left(w^h, T_t^h\right) + c\left(\boldsymbol{u}^h; w^h, T^h\right) + a\left(w^h, T^h\right) \\ &+ \left(w^h, T_t^h\right)^S + c\left(\boldsymbol{u}^h; w^h, T^h\right)^S - a\left(w^h, T^h\right)^S - \left(w^h, \gamma^h\right)^S \\ &= \left(w^h, \gamma^h\right) + \left(w^h, h^h\right)_{\Gamma_N} \qquad\qquad \forall w^h \in \mathcal{W}^h. \end{aligned} \tag{4.81}$$

where the approximated solution T^h is defined as

$$T^h(\boldsymbol{x}) = \sum_{A \in \eta \setminus \eta_D} N_A(\boldsymbol{x}) T_A + \sum_{A \in \eta_D} N_A(\boldsymbol{x}) T_D(\boldsymbol{x}_A) \tag{4.82}$$

here T_A is the nodal unknown at node number A and N_A is the shape function associated. Moreover the test functions w^h are defined such that

$$w^h \in \mathcal{W}^h := \operatorname{span}\{N_A\} \tag{4.83}$$

Finally, the finite element discretization of this weak form yields the system of semi-discrete equations written in terms of coefficient matrices

$$\begin{aligned} &\left(\boldsymbol{M} + \boldsymbol{M}^s\left(\boldsymbol{u}(t)\right)\right) \dot{T}(t) + \left[\left(\boldsymbol{K} - \boldsymbol{K}^s\left(\boldsymbol{u}(t)\right)\right) + \left(\boldsymbol{C}\left(\boldsymbol{u}(t)\right) + \boldsymbol{C}^s\left(\boldsymbol{u}(t)\right)\right)\right] T(t) \\ &= \boldsymbol{f}\,(t) + \boldsymbol{f}^s\,(t, \boldsymbol{u}(t)) \end{aligned} \tag{4.84}$$

or, in a more compact form

$$\bar{\boldsymbol{M}}\left(\boldsymbol{u}(t)\right) \dot{T}(t) + \left[\bar{\boldsymbol{K}}\left(\boldsymbol{u}(t)\right) + \bar{\boldsymbol{C}}\left(\boldsymbol{u}(t)\right)\right] T(t) = \bar{\boldsymbol{f}}\,(t, \boldsymbol{u}(t)) \tag{4.85}$$

where $\bar{\mathbf{M}}(\boldsymbol{u}(t)) = \boldsymbol{M} + \boldsymbol{M}^s(\boldsymbol{u}(t))$, $\bar{\mathbf{K}}(\boldsymbol{u}(t)) = \boldsymbol{K} - \boldsymbol{K}^s(\boldsymbol{u}(t))$, $\bar{\boldsymbol{C}}(\boldsymbol{u}(t)) = \boldsymbol{C}(\boldsymbol{u}(t)) + \boldsymbol{C}^s(\boldsymbol{u}(t))$ and $\bar{\boldsymbol{f}}(t, \boldsymbol{u}(t)) = \boldsymbol{f}(t) + \boldsymbol{f}^s(t, \boldsymbol{u}(t))$. The superscript S is referred to the SU/PG terms. Then defining the coefficient matrices through the action of the assembly operator $\mathop{\mathrm{A}}$ (see again [124]), acting on the local element matrix and nodal vector, as follows

$$\bar{\boldsymbol{M}} = \mathop{\mathrm{A}}_{e} \bar{\boldsymbol{M}}^e \qquad \bar{\boldsymbol{M}}^e = \int_{\Omega^e} N_a N_b \, d\Omega + \int_{\Omega^e} \tau \boldsymbol{u} \cdot \nabla N_a \, T \, d\Omega \tag{4.86}$$

$$\bar{\boldsymbol{K}} = \mathop{\mathrm{A}}_{e} \bar{\boldsymbol{K}}^e \qquad \bar{\boldsymbol{K}}^e = \kappa \left(\int_{\Omega^e} \nabla N_a \nabla N_b \, d\Omega - \int_{\Omega^e} \tau \boldsymbol{u} \cdot \nabla N_a \, \Delta T \, d\Omega \right) \tag{4.87}$$

$$\bar{\boldsymbol{C}} = \mathop{\mathrm{A}}_{e} \bar{\boldsymbol{C}}^e \qquad \bar{\boldsymbol{C}}^e = \int_{\Omega^e} N_a \left(\boldsymbol{u}^h \nabla N_b \right) \, d\Omega + \int_{\Omega^e} \tau \boldsymbol{u} \cdot \nabla N_a \left(\boldsymbol{u}^h \nabla T \right) \, d\Omega \tag{4.88}$$

$$\bar{\boldsymbol{f}} = \mathop{\mathrm{A}}_{e} \bar{\boldsymbol{f}}^e \qquad \bar{\boldsymbol{f}}^e = \int_{\Omega^e} N_a \gamma \, d\Omega + \int_{\Gamma_N^e} N_a h \, d\Gamma_N^e + \int_{\Omega^e} \tau \boldsymbol{u} \cdot \nabla N_a \gamma \, d\Omega \tag{4.89}$$

we can write the equation (4.85) in matrix form

$$\bar{\mathbf{M}}(\mathbf{u}) \, \dot{\mathbf{T}} + \left[\bar{\mathbf{K}}(\mathbf{u}) + \bar{\mathbf{C}}(\mathbf{u}) \right] \mathbf{T} = \bar{\mathbf{f}}(\mathbf{u}) \tag{4.90}$$

where $\mathbf{T}$ is the transpose vector of the unknown scalar components T_i and $\mathbf{u}$ is the transpose vector of the known velocity field components u_i.

In the end, the robustness of the classical upwind methods, in that spurious wiggles are not present, and the accuracy of wiggle-free Galerkin solutions, are the main features that characterize the streamline upwind/Petrov-Galerkin method. This latter is not affected by "artificial diffusion", many times living in the other upwind schemes, and its great success is mainly due to the streamline upwind concept, which neglect a crosswind diffusion behavior of the solution. Since it does not require higher-order weighting functions, the implementation of this method is quite simple and, moreover, it shows high accuracy in transient analysis.

4.3 Time integration methods

The spatial approximation of the original Navier-Stokes system (4.4) - (4.5) and the heat equation (4.61) are actually represented by the semi-discrete equations (4.48) and (5.166), respectively. Now a direct time integration procedure that replaces the continuous time derivative in (4.48) and (5.166) with an approximated form is required. In fact, in time-dependent problems the spatial finite element discretization needs to be transported in time by an appropriate algorithm able to balance both the spatial and the temporal approximations properly. Hence, an incremental procedure that provides numerical stability and accuracy have to be employed in order to advance the solution by discrete steps in time [100].

Though explicit methods have been widely used for the time integration of the Navier-Stokes equations, implicit methods are preferred to solve this kind of problems because explicit procedures include several difficulties related to the resolution of incompressible flows, such as: (i) the natural implicitness of the pressure variable in the system of equations; (ii) the strong restrictions on time step size to preserve the stability of the integration process; (iii) the issue of diagonalizing and inverting $\bar{\mathbf{M}}$ in a cost-effective manner for a variety of element types; and (iv) the decrease in accuracy given by the diagonalization of $\bar{\mathbf{M}}$. On the other hand, implicit integration methods, though computationally expensive, ensure greater stability and a consistent treatment of pressure for the Navier-Stokes system, and then they are widely implemented [111].

A better stability and a stable discretization, regardless of the length of the time step, are also achieved when implicit schemes are applied to time dependent advection-diffusion stabilized equations. In fact, it was proven in [131] that applying the streamline upwind stabilization operator in conjunction with implicit time integration can be considered as a safe separated discretization that does not lead to any additional stability restrictions on the Peclet or Courant numbers. Moreover, spurious oscillations are expected for small time steps, tipically used in explicit integration methods [132].

In the next paragraphs we present some possibilities for the time discretization of first-order differential equations. Usually a Newton's method is used to find the solution of the resulting algebraic system of nonlinear equations for each time step, but also some iterative procedures can be employed to solve the fully-discretized problem.

4.3.1 The θ family methods

Considering the Navier-Stokes system (4.48), we assume, for a generic time step $\Delta t = t_{n+1} - t_n$, the following time-discretization expression

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta t\, \dot{\mathbf{x}} \tag{4.91}$$

where we define $\dot{\mathbf{x}} := (1-\theta)\dot{\mathbf{x}}_n + \theta\, \dot{\mathbf{x}}_{n+1}$, and analogously $\mathbf{x} := (1-\theta)\mathbf{x}_n + \theta\, \mathbf{x}_{n+1}$, whereas θ is a parameter taken to be in the interval [0, 1]. By rewriting the equation (4.48) at the time

t_{n+1} and t_n substituting (4.91) we get

$$\bar{\mathbf{M}}\mathbf{x}_{n+1} = \bar{\mathbf{M}}\mathbf{x}_n + \Delta t\,\theta\left[\bar{\mathbf{f}} - \bar{\mathbf{K}}\left(\mathbf{u}_{n+1}\right)\mathbf{x}_{n+1}\right] + \Delta t\,(1-\theta)\left[\bar{\mathbf{f}} - \bar{\mathbf{K}}\left(\mathbf{u}_n\right)\mathbf{x}_n\right] \tag{4.92}$$

A set of nonlinear algebraic equations for the solution vector $\mathbf{x}$ at time t_{n+1} is obtained. Notice that, for different values of the parameter θ several time integration methods can be considered. In fact, conditionally stable schemes are obtained for values of $\theta < 1/2$, such as the Forward Euler method ($\theta = 0$). Instead, for values $\theta \geq 1/2$ the schemes are unconditionally stable, such as the Backward Euler method ($\theta = 1$), the Galerkin method ($\theta = 2/3$), and the Crank-Nicolson method ($\theta = 1/2$).

The Forward Euler method, or explicit Euler method, is probably the simplest time discretization algorithm that uses a forward differencing for the time derivative. Concerning its application in the context of incompressible flows, this method has two main disadvantages: (i) for a second order problem any explicit method enforces dt $\leq C\,h^2$ for some constant C and mesh size h, which results in considering a very high number of time steps, and (ii) the incompressibility condition cannot be satisfied exactly and the error increases linearly with the number of time steps. Hence, using an explicit scheme for incompressible flows it is usually not recommended, but in cases where a discrete subspace is implemented to exactly incorporate the incompressibility constraint and the viscosity ν is small, the time step restriction became larger (νh^2) and an explicit algorithm turns out to be competitive as an implicit one.

On the other hand, the Forward Euler method, or implicit Euler method, is a straightforward time discretization based on backward time differencing, where every time step a modified stationary Navier–Stokes system has been solved using, for example, a Newton iteration scheme. The main advantage of this method is the unconditional stability. Moreover, the solution at the previous time step $\mathbf{x}_n$ provide a good starting point for the nonlinear iteration if the time step is not too large, and a first order convergence in time, i.e, the error between the solution of the Navier–Stokes system and the solution of the semi-discrete problem is of order dt, is reached [98].

Finally, the semi-implicit Crank–Nicholson method is based on computing the average of the implicit and the explicit Euler methods in the parabolic equation, whereas enforcing the incompressibility constraint exactly. The main advantage of this scheme is the high order of convergence that is second order in the time step. At each time step, as seen for the implicit Euler method, a modified stationary Navier–Stokes problem has been solved, and therefore both methods have the same computational performance.

Similarly, the same time integration method can also be used to fully discretize the stabilized advection-diffusion equation. In particular, employing the θ-scheme to the approximated heat equation (5.166), that includes the SU/PG term, we can write

$$\begin{aligned}(1-\theta)\left(\bar{\mathbf{M}}\left(\mathbf{u}_n\right)\dot{\mathbf{T}}_n + \left[\bar{\mathbf{K}}\left(\mathbf{u}_n\right) + \bar{\mathbf{C}}\left(\mathbf{u}_n\right)\right]\mathbf{T}_n - \bar{\mathbf{f}}\left(\mathbf{u}_n\right)\right) + \\ \theta\left(\bar{\mathbf{M}}\left(\mathbf{u}_{n+1}\right)\dot{\mathbf{T}}_{n+1} + \left[\bar{\mathbf{K}}\left(\mathbf{u}_{n+1}\right) + \bar{\mathbf{C}}\left(\mathbf{u}_{n+1}\right)\right]\mathbf{T}_{n+1} - \bar{\mathbf{f}}\left(\mathbf{u}_{n+1}\right)\right) = 0.\end{aligned} \tag{4.93}$$

Now using the expression $(\mathbf{T}_{n+1} - \mathbf{T}_n) = \Delta t\, \dot{\mathbf{T}}$ where $\dot{\mathbf{T}} := (1-\theta)\dot{\mathbf{T}}_n + \theta\, \dot{\mathbf{T}}_{n+1}$ and $\mathbf{T} := (1-\theta)\mathbf{T}_n + \theta\, \mathbf{T}_{n+1}$, and remembering the sum $\bar{\mathbf{M}} = \mathbf{M} + \mathbf{M}^s$, we can write the mass matrix as

$$\begin{aligned} \mathbf{M}\Big((1-\theta)\dot{\mathbf{T}}_n + \theta\dot{\mathbf{T}}_{n+1}\Big) &= \mathbf{M}\left(\mathbf{T}_{n+1} - \mathbf{T}_n\right)/\Delta t \\ \mathbf{M}^s(\mathbf{u}_{n+1})\theta\dot{\mathbf{T}}_{n+1} &= \mathbf{M}^s(\mathbf{u}_{n+1})\left(\mathbf{T}_{n+1} - \mathbf{T}_n\right)/\Delta t - (1-\theta)\mathbf{M}^s(\mathbf{u}_{n+1})\dot{\mathbf{T}}_n \end{aligned} \tag{4.94}$$

so as to have

$$\begin{aligned} \Big(\mathbf{M} + \mathbf{M}^s(\mathbf{u}_{n+1})\Big)\left(\mathbf{T}_{n+1} - \mathbf{T}_n\right) + \Delta t(1-\theta)\Big(\left[\bar{\mathbf{K}}\left(\mathbf{u}_n\right) + \bar{\mathbf{C}}\left(\mathbf{u}_n\right)\right]\mathbf{T}_n - \bar{\mathbf{f}}\left(\mathbf{u}_n\right)\Big) \\ +\Delta t\,\theta\Big(\left[\bar{\mathbf{K}}\left(\mathbf{u}_{n+1}\right) + \bar{\mathbf{C}}\left(\mathbf{u}_{n+1}\right)\right]\mathbf{T}_{n+1} - \bar{\mathbf{f}}\left(\mathbf{u}_{n+1}\right)\Big) \\ -\Delta t(1-\theta)\Big(\mathbf{M}^s\left(\mathbf{u}_{n+1}\right) - \mathbf{M}^s(\mathbf{u}_n)\Big)\dot{\mathbf{T}}_n = 0 \end{aligned} \tag{4.95}$$

and finally

$$\begin{aligned} \bar{\mathbf{M}}(\mathbf{u}_{n+1})\mathbf{T}_{n+1} + \Delta t(\theta)\Big(\left[\bar{\mathbf{K}}\left(\mathbf{u}_{n+1}\right) + \bar{\mathbf{C}}\left(\mathbf{u}_{n+1}\right)\right]\mathbf{T}_{n+1} - \bar{\mathbf{f}}\left(\mathbf{u}_{n+1}\right)\Big) = \\ \bar{\mathbf{M}}(\mathbf{u}_{n+1})\mathbf{T}_n - \Delta t(1-\theta)\Big(\left[\bar{\mathbf{K}}\left(\mathbf{u}_n\right) + \bar{\mathbf{C}}\left(\mathbf{u}_n\right)\right]\mathbf{T}_n - \bar{\mathbf{f}}\left(\mathbf{u}_n\right)\Big) \\ +\Delta t(1-\theta)\Big(\mathbf{M}^s\left(\mathbf{u}_{n+1}\right) - \mathbf{M}^s(\mathbf{u}_n)\Big)\dot{\mathbf{T}}_n \end{aligned} \tag{4.96}$$

where

$$\dot{\mathbf{T}}_n = \frac{1}{\theta\Delta t}(\mathbf{T}_n - \mathbf{T}_{n-1}) - \frac{1-\theta}{\theta}\dot{\mathbf{T}}_{n-1} \tag{4.97}$$

Also here, depending on the choice of the θ parameter several time integration schemes can be used.

As done previously, choosing $\theta = 0$ an explicit forward Euler method is employed, which requires the matrix $\bar{\mathbf{M}}$ to be easily invertible. The explicit nature of the algorithm implies a stability condition on the maximum allowable time step, the material's thermal diffusivity and the finite element mesh size. The scheme is first-order accurate in time and despite these limitations is commonly used in conduction problems. The main advantages of explicit procedure is the minimal requirement of computer resources because the matrix solution storage is not required. Implicit methods, that require the solution of a matrix problem at each time step, appear for $\theta > 0$. The fully implicit backward Euler scheme ($\theta = 1$) is an unconditionally stable method without time step restriction and it is first-order accurate in time. The Crank—Nicolson method ($\theta = 1/2$) is the only second-order accurate scheme of the θ family methods. It is unconditionally stable and it is not affected by time step restriction for stability, but some oscillations could appear when a too large time step is employed.

4.3.2 Multistep methods

Multistep Backward Difference Formulas (BDF), and in particular the second-order accurate BDF scheme (BDF2), are implicit time integration schemes widely used in the computation of large-scale engineering flows, since they are known for their stability. For a given function, this family of methods approximates the time derivative using storage solution levels from previous

time steps, increasing the accuracy of the algorithm. Then, a linear or nonlinear set of equations are required to be solved at each time step. However, BDF methos can not be A-stable if an order greater than 2 is considered and, they are difficult to use with variable time steps [133].

The general implementation, for the Navier-Stokes equations (4.48), of a k-step BDF scheme is represented by

$$\bar{\mathbf{M}}\mathbf{x}_{n+k} = -\sum_{i=0}^{k-1} \alpha_i \, \bar{\mathbf{M}}\mathbf{x}_{n+i} + \Delta t \, \beta_k \left[\bar{\mathbf{f}} - \bar{\mathbf{K}}\left(\mathbf{u}_{n+k}\right)\mathbf{x}_{n+k}\right] \tag{4.98}$$

whereas, for the stabilized heat equation (5.166) we have to write

$$\begin{aligned} \bar{\mathbf{M}}\left(\mathbf{u}_{n+k}\right)\mathbf{T}_{n+k} = &-\sum_{i=0}^{k-1} \alpha_i \, \bar{\mathbf{M}}\left(\mathbf{u}_{n+i}\right)\mathbf{T}_{n+i} \\ &+ \Delta t \, \beta_k \left[\bar{\mathbf{f}}\left(\mathbf{u}_{n+k}\right) - \left(\bar{\mathbf{K}}\left(\mathbf{u}_{n+k}\right) + \bar{\mathbf{C}}\left(\mathbf{u}_{n+k}\right)\right)\mathbf{T}_{n+k}\right] \end{aligned} \tag{4.99}$$

Hence, BDF schemes require the computation of k-1 solutions of previous time steps, and then a linear or nonlinear equation is solved at each time step. For k=1 (BDF1) the scheme is equivalent to the Backward Euler method, whereas, for k=2 (BDF2) a second-order accurate scheme is obtained using the coefficients: $\alpha_0 = -4/3$, $\alpha_1 = 1/3$ and $\beta_2 = 2/3$.

4.4 Algorithms for solving linear systems

A system of m linear equations in n unknowns is a set of algebraic relations such as

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \qquad i = 1, \ldots, m \tag{4.100}$$

where x_j are the unknowns, a_{ij} the system coefficients and b_i the constant terms, The system (4.100) is commonly written in matrix form

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{4.101}$$

indicating with $\mathbf{A} = (a_{ij}) \in \mathbb{R}^{m \times n}$ the matrix of coefficients, with $\mathbf{b} = (b_i) \in \mathbb{R}^m$ the vector of constant terms and with $\mathbf{x} = (x_j) \in \mathbb{R}^n$ the vector of unknowns. The solution of (4.101) is any n-tuple of x_j values that verify (4.100). In this section we show numerical techniques able to solve the system (4.101) in the case of $m = n$, supposing $\mathbf{A}$ non singular, that is $\det \mathbf{A} \neq 0$.

Now, solving the linear system (4.101) using Cramer's rule requires unacceptable high computational cost, of the order of $(n+1)!$ operations. Hence, several alternative numerical methods to Cramer's rule have been developed, and they are divided in two categories: (i) *direct solvers*, which lead to the solution of the system with a finite number of operations, and (ii) *iterative solvers*, which require an infinite (theoretically) number of operations [134].

4.4.1 Direct Solvers

The solution of a linear system can be performed using the Gauss elimination method (MEG), in which the system (4.101) is transformed in an equivalent system of the form $\mathbf{A}^{(n)}\mathbf{x} = \mathbf{b}^{(n)}$, where $\mathbf{A}^{(n)} = \mathbf{U}$ is a non singular upper triangular matrix, and $\mathbf{b}^{(n)}$ is a new constant term. This can be solved, with a computational cost of the order of n^2 operations, using the following backward elimination algorithm

$$\begin{aligned} x_n &= \frac{b_n^{(n)}}{u_{nn}}, \\ x_i &= \frac{1}{u_{ii}} \left(b_i^{(n)} - \sum_{j=i+1}^{n} u_{ij} x_j \right), \qquad i = n-1, \ldots, 1 \end{aligned} \tag{4.102}$$

considering $\mathbf{A}^{(1)}\mathbf{x} = \mathbf{b}^{(1)}$ the original system. In the MEG algorithm the passage between the matrices $\mathbf{A}^{(k)}$ and $\mathbf{A}^{(k+1)}$ can be obtained with the following steps

$$\begin{aligned} m_{ik} &= a_{ik}^{(k)} / a_{kk}^{(k)}, \qquad i = k+1, \ldots, n, \\ a_{ij}^{k+1} &= a_{ij}^{(k)} - m_{ik} a_{jk}^{(k)}, \qquad i, j = k+1, \ldots, n, \\ b_i^{k+1} &= b_i^k - m_{ik} b_k^{(k)}, \qquad i = k+1, \ldots, n. \end{aligned} \tag{4.103}$$

Notice that the elements $a_{ij}^{(k+1)}$ with $i = k$ and $j = k+1, \ldots, n$ are all zeros. Elements m_{ik} are the *multipliers*, whereas the terms a_{kk} are the *pivotal elements*. Obviously the MEG scheme

works only if the pivotal elements are not zeros, and it is happen for diagonally dominant or positive-definite matrices. In general it is necessary to apply some pivoting technique in order to ensure that pivotal elements are not zeros. To complete the Gauss elimination $2(n-1)n(n+1)/3+n(n-1)$ *flops (floating point operations)* are needed, plus n^2 *flops* to solve the triangular system $\mathbf{Ux} = \mathbf{b}^{(n)}$ with the back substitution method. Then, the whole process related to the Gauss elimination requires $2n^3/3$ *flops*.

However, the MEG scheme is equivalent to factorize the matrix $\mathbf{A}$, that is writing $\mathbf{A}$ such as a product $\mathbf{LU}$ of two matrices. The upper triangular matrix $\mathbf{U}$ is the matrix $\mathbf{A}^{(n)}$ obtained with the elimination process, whereas $\mathbf{L}$ is the lower triangular matrix, whose elements are equal to one on the diagonal and equal to the *multipliers* in the remaining lower triangular portion. Once the matrices $\mathbf{L}$ and $\mathbf{U}$ are known, the resolution of the linear system is achieved solving the two triangular systems in sequence

$$\mathbf{Ly} = \mathbf{b}, \qquad \mathbf{Ux} = \mathbf{y}. \tag{4.104}$$

The computational cost of the factorization process is the same as that required by the MEG. The advantages of this reinterpretation are clear: since $\mathbf{L}$ and $\mathbf{U}$ depend only on $\mathbf{A}$, and not on the constant term $\mathbf{b}$, the same factorization can be used to solve several linear systems with matrix $\mathbf{A}$, but with a variable term $\mathbf{b}$. An example is the discretization of a linear parabolic problem in which, at each time step, it is necessary to solve a system with the same matrix $\mathbf{A}$ and different term $\mathbf{b}$. Consequently, since the computational cost is concentrated in the elimination procedure (about $2n^3/3$ *flops*), there is a considerable reduction in the number of operations if several linear systems having the same matrix have to be solved.

4.4.2 Iterative Solvers

Iterative methods aim to construct the solution $\mathbf{x}$ of a linear system as the limit of a succession of $\{\mathbf{x}^n\}$ vectors. In order to obtain the single succession element the calculation of the residual $\mathbf{r}^{(n)} = \mathbf{b} - \mathbf{Ax}^{(n)}$ is required.

In the case of a full matrix of order of n, the computational cost of an iterative method is of n^2 operations for each iteration. This latter must be compared with the $2n^3/3$ operations of a direct method. Consequently, iterative methods are competitive with direct methods only if the number of iterations necessary to reach convergence (within a fixed tolerance) is independent of n or depends on n in a sublinear way.

Other observations in choosing between an iterative method and a direct method must be considered if the matrix is sparse. A general strategy to build iterative schemes is based on the splitting of the matrix $\mathbf{A}$ in the form $\mathbf{A} = \mathbf{P} - \mathbf{N}$, where $\mathbf{P}$ and $\mathbf{N}$ are two appropriate matrices and $\mathbf{P}$ is non singular, also named *preconditioning matrix* or *preconditioner*.

Then, assigned $\mathbf{x}^{(0)}$, we obtain $\mathbf{x}^{(k)}$ for $k \geq 1$ solving the new systems

$$\mathbf{Px}^{(k+1)} = \mathbf{Nx}^{(k)} + \mathbf{b}, \qquad k \geq 0 \tag{4.105}$$

or, equivalently

$$\mathbf{x}^{(k+1)} = \mathbf{B}\mathbf{x}^{(k)} + \mathbf{P}^{-1}\mathbf{b}, \qquad k \geq 0 \tag{4.106}$$

indicating with $\mathbf{B} = \mathbf{P}^{-1}\mathbf{N}$ the *iteration matrix*. We are interested in convergent iterative methods that is such that $\lim_{k\to\infty} \mathbf{e}^{(k)} = \mathbf{0}$ for each choice of the *initial vector* $\mathbf{x}^{(0)}$, where $\mathbf{e}^{(k)} = \mathbf{x}^{(k)} - \mathbf{x}$ the error. Since with a recursive argument it is found

$$\mathbf{e}^{(k)} = \mathbf{B}^k\mathbf{e}^{(0)}, \qquad \forall k = 0, 1, \ldots \tag{4.107}$$

We can conclude that an iterative method of the form (4.105) is convergent if and only if $\rho(\mathbf{B}) \leq 1$, that is the spectral radius of the iteration matrix $\mathbf{B}$, *i.e.* the largest absolute value of its eigenvalues. The equation (4.105) can be written as

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{P}^{-1}\mathbf{r}^{(k)}, \tag{4.108}$$

indicating with $\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}$ the *residual vector* at the step k. Then, in order to update the solution at the $k+1$ step, it is necessary to solve a linear system of matrix $\mathbf{P}$, which must be non singular and, to get a quick scheme, invertible with low computational costs. Notice that if $\mathbf{P}$ were equal to $\mathbf{A}$ and $\mathbf{N} = 0$, the method (4.108) would converge in one iteration, but with computational cost of a direct method.

Now we introduce the iteration matrix

$$\mathbf{R}_\mathbf{P} = \mathbf{I} - \mathbf{P}^{-1}\mathbf{A} \tag{4.109}$$

associated to the scheme (4.108), which can be accelerated introducing an appropriate relaxation parameter α. In this way *Richardson methods* are obtained as follow

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha\mathbf{P}^{-1}\mathbf{r}^{(k)}, \qquad k \geq 0. \tag{4.110}$$

In general, supposing α dependent on the iteration step, *non stationary Richardson methods* are given

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k\mathbf{P}^{-1}\mathbf{r}^{(k)}, \qquad k \geq 0. \tag{4.111}$$

The matrix iteration at the k-step is

$$\mathbf{R}(\alpha_k) = \mathbf{I} - \alpha_k\mathbf{P}^{-1}\mathbf{A}. \tag{4.112}$$

In the case of $\mathbf{P} = \mathbf{I}$, the method is *not preconditioned*. Then, we can write equation (4.111) in a very interesting form for computational usage. Introducing the *preconditioned residual* $\mathbf{z}^{(k)} = \mathbf{P}^{-1}\mathbf{r}^{(k)}$, we get $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k\mathbf{z}^{(k)}$ and $\mathbf{r}^{(k+1)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k+1)} = \mathbf{r}^{(k)} + \alpha_k\mathbf{A}\mathbf{z}^{(k)}$. Hence, the $k+1$ step of the Richardson method requires the following operation:

solve the linear system $\mathbf{P}\mathbf{z}^{(k)} = \mathbf{r}^{(k)}$,

compute the relaxation parameter α_k,

update the solution $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k\mathbf{z}^{(k)}$,

update the residual $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} + \alpha_k \mathbf{A}\mathbf{z}^{(k)}$.

Going few steps back, it is worth noting that, in the case of $\mathbf{A}$ is a symmetric positive defined matrix, the resolution of the system (4.101) is equivalent to find the minimum of $\mathbf{x} \in \mathbb{R}^n$ of

$$\mathbf{\Phi}(\mathbf{y}) = \frac{1}{2}\mathbf{y}^\top \mathbf{A}\mathbf{y} - \mathbf{y}^\top \mathbf{b}, \tag{4.113}$$

that is the *energy of the system*. Therefore, the problem is to determine the minimum point $\mathbf{x}$ of $\mathbf{\Phi}$ starting from a point $\mathbf{x}^{(0)} \in \mathbb{R}^n$ and, consequently, choose appropriate directions in order to get closer, as quickly as possible, to the solution $\mathbf{x}$. The optimal direction, joining $\mathbf{x}^{(0)}$ and $\mathbf{x}$, is obviously not known a priori: then, we have to move from $\mathbf{x}^{(0)}$ along another direction $\mathbf{d}^{(0)}$, and on this fix a new point $\mathbf{x}^{(1)}$, from which repeat the procedure until convergence.

At the generic step k we therefore determinate $\mathbf{x}^{(k+1)}$ as

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}, \tag{4.114}$$

where α is the value of the step length along $\mathbf{d}^{(k)}$. The more natural idea, which consists in taking the maximum slope for $\mathbf{\Phi}$, given by $\mathbf{r}^{(k)} = -\nabla\mathbf{\Phi}(\mathbf{x}^{(k)})$, as the descent direction, leads to the *gradient method*, or *steepest descent method*.

This produce the following algorithm: given $\mathbf{x}^{(0)} \in \mathbb{R}^n$, write $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$, for $k = 0, 1, \ldots$ until convergence, we compute

$$\begin{aligned} \alpha_k &= \frac{{\mathbf{r}^{(k)}}^\top \mathbf{r}^{(k)}}{{\mathbf{r}^{(k)}}^\top \mathbf{A}\mathbf{r}^{(k)}} \\ \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \alpha_k \mathbf{r}^{(k)} \\ \mathbf{r}^{(k+1)} &= \mathbf{r}^{(k)} + \alpha_k \mathbf{A}\mathbf{r}^{(k)} \end{aligned} \tag{4.115}$$

Its preconditioned form is: given $\mathbf{x}^{(0)} \in \mathbb{R}^n$, write $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$, $\mathbf{z}^{(0)} = \mathbf{P}^{-1}\mathbf{r}^{(0)}$, for $k = 0, 1, \ldots$ until convergence, we compute

$$\begin{aligned} \alpha_k &= \frac{{\mathbf{z}^{(k)}}^\top \mathbf{r}^{(k)}}{{\mathbf{z}^{(k)}}^\top \mathbf{A}\mathbf{z}^{(k)}} \\ \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \alpha_k \mathbf{z}^{(k)} \\ \mathbf{r}^{(k+1)} &= \mathbf{r}^{(k)} + \alpha_k \mathbf{A}\mathbf{z}^{(k)} \\ \mathbf{P}\mathbf{z}^{(k+1)} &= \mathbf{r}^{(k+1)} \end{aligned} \tag{4.116}$$

An other more efficient alternative is the *conjugate gradient method*, in which descent directions no longer coincide with those of the residual. In particular, given $\mathbf{p}^{(0)} = \mathbf{r}^{(0)}$, we find directions of the form

$$\mathbf{p}^{(k+1)} = \mathbf{r}^{(k+1)} - \beta_k \mathbf{p}^{(k)}, \qquad k = 0, 1, \ldots, k. \tag{4.117}$$

Direction of this type are called A-orthogonal. The method in the preconditioned case assume the form: given $\mathbf{x}^{(0)} \in \mathbb{R}^n$, write $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$, $\mathbf{z}^{(0)} = \mathbf{P}^{-1}\mathbf{r}^{(0)}$ and $\mathbf{p}^{(0)} = \mathbf{r}^{(0)}$, the k step for

$k = 0, 1, \ldots$ is

$$\begin{aligned}
\alpha_k &= \frac{{\mathbf{p}^{(k)}}^\top \mathbf{r}^{(k)}}{{(\mathbf{A}\mathbf{p}^{(k)})}^\top \mathbf{p}^{(k)}} \\
\mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \alpha_k \mathbf{p}^{(k)} \\
\mathbf{r}^{(k+1)} &= \mathbf{r}^{(k)} - \alpha_k \mathbf{A}\mathbf{p}^{(k)} \\
\mathbf{P}\mathbf{z}^{(k+1)} &= \mathbf{r}^{(k+1)} \\
\beta_k &= \frac{{(\mathbf{A}\mathbf{p}^{(k)})}^\top \mathbf{z}^{(k+1)}}{{\mathbf{p}^{(k)}}^\top \mathbf{A}\mathbf{p}^{(k)}} \\
\mathbf{p}^{(k+1)} &= \mathbf{z}^{(k+1)} - \beta_k \mathbf{p}^{(k)}
\end{aligned} \tag{4.118}$$

The parameter α_k is chosen such that the error is minimized along the descent direction $\mathbf{p}^{(k)}$, whereas the parameter β_k ensure that the new direction $\mathbf{p}^{(k+1)}$ is A-conjugate with $\mathbf{p}^{(k)}$, that is ${(\mathbf{A}\mathbf{p}^{(k)})}^\top \mathbf{p}^{(k+1)} = 0$. For large dimension matrices the conjugate gradient method is employed such that the iterative process is terminated when the error is lower than a specified tolerance.

Generalizations of the gradient method, in case of the matrix $\mathbf{A}$ is not symmetric, lead to the so-called Krylov methods, among which notable examples are the GMRES (Generalized Minimal Residual Method), the conjugated big-gradient method, BiCG, and to its stabilized version, the BiCGSTAB method. In particular, the GMRES method is widely used to solve systems with non symmetric sparse matrices, such that the convective transport matrix in the Navier-Stokes equations. But, before focusing on the implementation of the algorithm, it is necessary a brief introduction on *Krylov subspaces*.

The m-th Krylov subspace for the problem (4.101) is

$$\mathcal{K}_m = \mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)}) = \text{span}\{\mathbf{r}^{(0)}, \mathbf{A}\mathbf{r}^{(0)}, \mathbf{A}^2\mathbf{r}^{(0)}, \ldots, \mathbf{A}^{m-1}\mathbf{r}^{(0)}\}, \qquad m = 1, 2, \ldots, \tag{4.119}$$

Projection methods on Krylov subspaces compute for each m an approximate solution $\mathbf{x}^{(m)}$ of the system $\mathbf{A}\mathbf{x} = \mathbf{b}$, that belong to $\mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)})$, where $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$ is the initial residual vector and $\mathbf{x}^{(0)}$ is the initial solution.

Unfortunately, the basis $\mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)})$ is numerically unstable because, as m increase, vectors tend to became linearly dependent and, it is necessary to extract an orthonormal basis of the Krylov basis. Then, the approximate solution has to be found in $\mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)})$, which increase the dimension at each iteration, such that the residual vector is orthogonal to a subspace, formed orthonormal vectors, named $\mathcal{L}_m$. In other words, we have to found the solution $\mathbf{x}^{(m)}$ in the subspace $\mathbf{x}^{(0)} + \mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)})$, imposing

$$\mathbf{b} - \mathbf{A}\mathbf{x}^{(m)} \perp \mathcal{L}_m \tag{4.120}$$

For this purpose, the Arnoldi iteration is used to find orthonormal vectors $\mathbf{q}^{(1)}, \mathbf{q}^{(2)}, \ldots, \mathbf{q}^{(m)}$ which form a basis for $\mathcal{K}_m(\mathbf{A}, \mathbf{r}^{(0)})$. The orthonormalization process is implemented as follow:

given $\mathbf{q}^{(1)} = \mathbf{r}^{(0)}/||\mathbf{r}^{(0)}||$, the k-step for $k = 1, \ldots, m$ is

$$\begin{aligned} \tilde{\mathbf{q}}^{(k+1)} &= \mathbf{A}\mathbf{q}^{(k)} - \sum_{i=1}^{k} \mathbf{h}^{(ik)}\mathbf{q}^{(i)} \\ \mathbf{q}^{(k+1)} &= \frac{\tilde{\mathbf{q}}^{(k+1)}}{||\tilde{\mathbf{q}}^{(k+1)}||} \end{aligned} \tag{4.121}$$

where $\mathbf{h}^{(ik)} = {\mathbf{q}^{(i)}}^{\top}\mathbf{A}\mathbf{q}^{(k)}$.

Now, we are ready to introduce the Generalized Minimal Residual (GMRES) method. This computes the solution of the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ minimizing the norm of the residual $\mathbf{r}^{(m)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(m)}$, with

$$\mathbf{x}^{(0)} + \mathcal{K}_m = \mathbf{x}^{(0)} + \mathbf{Q}_m\mathbf{y} \tag{4.122}$$

where $\mathcal{K}_m$ is the Krylov space generated by the initial residual $\mathbf{r}^{(0)}$ and $\mathbf{Q}_m$ is the matrix that establishes an orthonormal basis for $\mathcal{K}_m$. In the end, the GMRES algorithm has, first of all, orthogonalize the Krylov basis using the Arnoldi method through

$$\mathbf{A}\mathbf{q}^{(k)} = \sum_{i=1}^{k+1} \mathbf{h}^{(ik)}\mathbf{q}^{(i)} \tag{4.123}$$

that in matrix form is written

$$\mathbf{A}\mathbf{Q}_m = \mathbf{Q}_{m+1}\mathbf{H} \tag{4.124}$$

Then we define the residual that we want to minimize

$$\begin{aligned} \mathbf{r}^{(m+1)} &= \mathbf{b} - \mathbf{A}\mathbf{x}^{(m)} \\ &= \mathbf{b} - \mathbf{A}(\mathbf{x}^{(0)} + \mathbf{Q}_m\mathbf{y}) \\ &= \mathbf{r}^{(0)} - \mathbf{A}\mathbf{Q}_m\mathbf{y} \\ &= \beta\mathbf{q}^{(1)} - \mathbf{Q}_{m+1}\mathbf{H}\mathbf{y} \\ &= \mathbf{Q}_{m+1}(\beta\mathbf{e}^{(1)} - \mathbf{H}\mathbf{y}) \end{aligned} \tag{4.125}$$

where $\beta = ||\mathbf{r}^{(0)}||$ and $\mathbf{e}^{(1)} = (1, 0, 0, \ldots, 0)^{\top}$. Since the matrix $\mathbf{Q}_{m+1}$ is orthonormal we have

$$||\mathbf{r}^{(m)}|| = ||\mathbf{b} - \mathbf{A}(\mathbf{x}^{(0)} + \mathbf{Q}_m\mathbf{y})|| = ||\beta\mathbf{e}^{(1)} - \mathbf{H}\mathbf{y}|| \tag{4.126}$$

Finally, the GMRES algorithm minimizes (4.126), that is find the approximate solution

$$\mathbf{x}^{(m)} = \mathbf{x}^{(0)} + \mathbf{Q}_m\mathbf{y}^{(m)} \tag{4.127}$$

where

$$\mathbf{y}^{(m)} = \arg\min_{\mathbf{y}} ||\beta\mathbf{e}^{(1)} - \mathbf{H}\mathbf{y}|| \tag{4.128}$$

4.5 Particle methods

Numerical models for the particle phase are implemented in order to resolve the dynamics and thermal properties of each particle in the system [102]. At the current state of art, three different approaches have been developed for numerical simulation of particles immersed in a fluid domain: (i) the *Discrete Element Method* (DEM), (ii) the *Discrete Parcel Method* (DPM) and, (iii) the *Two-Fluid model* (TF).

In particular, the Discrete Element Method controls the motion of each individual particle, taking into account fluid dynamic forces, contact forces and collisions between particles. This method is very accurate but every single particle equation has to be solved and, this could lead to high computational costs when the number of particles increases considerably. The second approach, the Discrete Parcel Method, identify a parcel of particles that travel through the field represented by one *computational particle*. In this case the properties of each particle in the parcel, such as size, velocity and temperature, are the same. It seems obvious that this method allow to save calculation time, but it is slightly less accurate than the previous one. The last approach is the Two-Fluid model, where the properties of the particles are assumed to be continuous like those of a fluid. So, for every single node of the Eulerian field, a set of algebraic conservation equations have to be solved to get the properties of the particles cloud. Depending on the particles flow features, this method could be or not to be computationally competitive with the two previously mentioned.

The first two models, the DEM and the DPM, are *Lagrangian tracking approaches*, in which the single discrete particle, or the parcel of particles, is tracked though the Eulerian domain and its properties are determined form the point-wise values of the field. The TF model, where both properties of particles and fluid are considered as continuous fields, represents an *Eulerian approach*. In order to choose the most suitable approach in numerical simulations, the dense or dilute character of the disperse phase is a fundamental feature to consider. In particular, when the volume fraction of the disperse phase, that is the amount of particles volume respect to the fluid volume, is lower than 12% the flow is dilute, whereas, increasing this ratio, a dense flow has to be considered. In dilute flows the velocity response time of the particles, that is the time required for a particle to respond to a change in velocity, is lower than the time between particles collision. It means that particle-fluid interaction determinate the particle motion more than the particle-particle interaction. In this case, the particle information, such as size, velocity and temperature, travels along particle trajectories and therefore a Lagrangian approach must be used. On the other hand, the Eulerian approach, that treats particles as a continuous phase, is applicable in dense flows, where particle collisions allow information to travel in all directions.

After these preliminary considerations the conservation equations for single particles, concerning the evolution of both motion and thermal properties, are introduced [135, 136]. In particular, assuming the drag force and gravity as the factors that most influence the fluid-

particle interaction, the equation of motion of for each particle is written as follow

$$m_p \frac{d\boldsymbol{u}_p}{dt} = 0.5 A_p C_D \rho_p \left(\boldsymbol{u} - \boldsymbol{u}_p\right) \left|\boldsymbol{u} - \boldsymbol{u}_p\right| + m_p \boldsymbol{g} \tag{4.129}$$

where m_p is the mass of the particle, $\boldsymbol{u}_p$ is the particle velocity field, A_p is the area of the particle defined as $A_p = \pi d_p^2/4$, where d_p is the diameter of the particle, C_D is the drag coefficient, ρ_p is the density of the particle and $\boldsymbol{g}$ is the gravitational force. The first term in the right hand side of the equation (4.129) represents the drag force that the fluid impresses on the particles causing their motion, whereas the second term models the body forces, such as gravity. These two forces must balance the particle inertia, in the left hand side of the equation, that is the product of particle mass times its acceleration.

As explained in Section 2.1.1, particles that form the powder stream interact with also a thermal field, mainly generated by the laser source and the melt pool high temperatures. In fact, particles exiting form the nozzle intersect the path of the laser beam and are subjected to high energy radiation, causing a sudden increase of the powder temperature. Moreover, the energy reflected by both the melt pool and the substrate contributes on heating particles, that at the same time exchange heat with the surrounding fluid by convection. Notice that, the temperature of a single particle is mainly influenced by its thermal conductivity, and the *Biot number*, that is the ratio of convective to conductive heat transfer, can be considered as a decisive parameter to determinate the importance of the particle internal conduction [61]. In particular the Biot number is defined as

$$\mathrm{Bi} = \frac{h d_p}{k_p} \tag{4.130}$$

where h is the heat convection coefficient, d_p the particle diameter and k_p the particle heat conduction coefficient. If a particle is small, $\mathrm{Bi} < 0.1$, the temperature gradient within a particle in negligible and the assumption of infinite conduction can be made. In this thesis, where stainless-steel powder and nitrogen gas are considered, the Biot number is $\mathrm{Bi} \ll 0.01$.

Hence, under these conditions, the energy conservation of a particle, that include phenomena of energy absorption, external convection and radiation, can be described by the following relation

$$m_p c_p \frac{dT_p}{dt} = I_T \eta_p A_{p,p} - h A_p \left(T - T_p\right) - m_p L_f \frac{df}{dt} \tag{4.131}$$

where m_p is the mass of the particle again, c_p is the specific heat of powder particle, η_p is the particle material absorption coefficient, $A_{p,p}$ is the effective projected area of the particle on laser beam, A_p is surface area of the particle, T_p and T are the temperature of the powder particle and surrounding gas, L_f is the latent heat of fusion, that contributes to the heat transfer of powder particle due to phase change such as melting or partial melting, f is the liquid mass fraction, df/dt is its derivative respect to time, and I_T is the total energy of the laser incident on the particle define as:

$$I_T(x, y, z) = \frac{2P}{\pi w^2(x)} \exp\left[-\frac{2(y^2 + z^2)}{w^2(x)}\right] \tag{4.132}$$

where P is the laser power and $w(x)$ the effective radius of laser beam, that is obviously twice the effective diameter D described as follows

$$2w(x) = D(x) = D_0\sqrt{1 + \left(\frac{z - z_0}{z_R}\right)^2} = \sqrt{D_0^2 + \theta^2(z - z_0)^2} \tag{4.133}$$

where θ the divergence angle, z_0 the focal point distance and D_0 the diameter of the laser beam at the focal point (see Figure 4.4).

The term in the left hand side of the equation (4.131) represents the net energy stored by

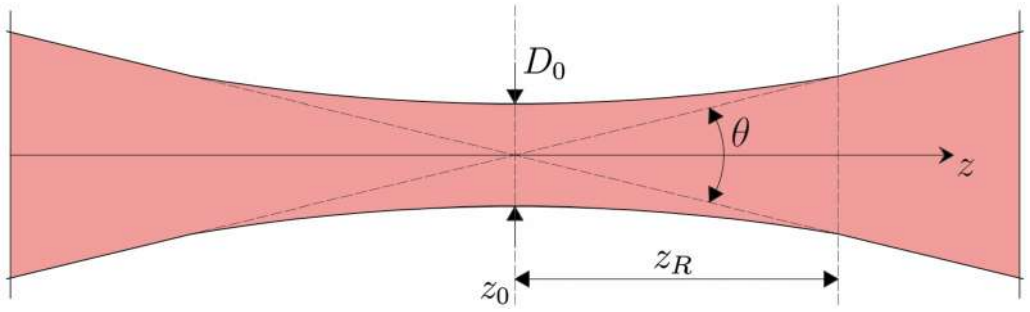

Figure 4.4: Trend of the propagation in free space of the effective diameter of the laser beam. Valid for Gaussian distribution.

the particle, whereas the terms in the right hand side model, the laser beam energy input, the heat loss by convection and, finally, the phase change.In particular a linear relationship is used towards f and particle temperature between the solidus temperature T_{sol} and the liquidus temperature T_{liq}, which can be expressed as:

$$f = \frac{m_l}{m_p} = \frac{T_p - T_{sol}}{T_{liq} - T_{sol}} \qquad T_{sol} < T_p < T_{liq} \tag{4.134}$$

where m_l is the liquid mass. When the temperature of the particle is smaller than solidus temperature and larger than liquidus temperature the liquid mass fraction f is equal to 0 or 1, respectively. Then, the time derivative of the liquid mass fraction can be expressed as a function of the time derivative of the particle temperature as:

$$\frac{df}{dt} = \frac{1}{m_p}\frac{dm_l}{dt} = \begin{cases} 0 & T_p < T_{sol} \\ \frac{1}{T_{liq} - T_{sol}}\frac{dT_p}{dt} & T_{sol} < T_p < T_{liq} \\ 0 & T_p > T_{liq} \end{cases} \tag{4.135}$$

or, in a more compact form, such as:

$$\frac{df}{dt} = \frac{1}{m_p}\frac{dm_l}{dt} = \frac{\delta_t}{T_{liq} - T_{sol}}\frac{dT_p}{dt} \qquad \text{with } \delta_t = \begin{cases} 0 & T_p \leq T_{sol} \text{ or } T_p \geq T_{liq} \\ 1 & T_{sol} < T_p < T_{liq} \end{cases} \tag{4.136}$$

where the Kronecker function δ_t is used to distinguish the temperature of the particle.
In the equations (4.129) and (4.131), both the velocity field $\boldsymbol{u}$ and the temperature field T at the

particle location are not available in discretized models. Then, interpolating the approximated solution vectors $\boldsymbol{u}^h$ and T^h, obtained solving the Navier-Stokes equations and the heat equation respectively, at the particle position, both fluid velocity and temperature can be evaluated. Operatively, in the FEM framework, a sum over the shape functions, belonging to the cell where the particle is located, must be employed to extract the eulerian field at the particle position. This procedure leads to the following equations

$$\boldsymbol{u} \simeq \boldsymbol{u}^h = \sum_{A \in \eta} N_A(\boldsymbol{x}_p) \boldsymbol{u}_A^h \tag{4.137}$$

$$T \simeq T^h = \sum_{A \in \eta} N_A(\boldsymbol{x}_p) T_A^h \tag{4.138}$$

where $\boldsymbol{x}_p$ is the particle location, $\eta = \{1, 2, \dots n_{np}\}$ is the number of global velocity node in the finite element mesh and N_A the shape function associated with global node number A.

Numerical approaches and implementation

5.1 Available softwares: an overview

Due to the complexity of modern fluid dynamics problems, their resolution is very difficult to obtain without the use of Computational Fluid Dynamics (CFD). CFD represents the science of solving partial differential equations (PDEs) of fluid motion numerically [134] in order to predict and analyze various kinds of flow phenomena [137]. This numerical approach, if used appropriately, is optimal to carry on parametric studies and flow-physics investigations, otherwise impossible or impractical to be achieved through theoretical or experimental efforts [138]. In fact, analytical solutions of fluid mechanics equations are very limited and, when the geometry of the physical domain becomes complicated, it is necessary to choose a numerical method to reach an acceptable results [139]. Recently, many numerical methods have been developed in order to solve the fluid mechanics PDEs, but historically, there are three fundamental formulations to CFD: the Finite Difference Method (FDM), the Finite Volume Method (FVM) and the Finite Element Method (FEM) [140]. These methods are essentially based on a spatial discretization of the domain into a grid or mesh of points or elements, and then, marching the numerical solution forward in discrete time steps [141]. In this context, numerous commercial and opensouce software have been developed, for many engineering applications, capable of automatically solving the equations governing the motion of both fluids and solids [142].

5.1.1 Commercial software

Commercial codes are available for solving a wide range of engineering problems. A user friendly interface and a easy learning process are the main advantages of these softwares which, often, are equipped with CAD tools. Moreover, support and training for their users are provided by the manufacturers, usually requiring the purchase of a license with a validity limited to a certain amount of time, or sometimes, unlimited expiration once purchased [143].

One of the most popular CFD commercial software is Ansys Fluent [144], which uses the Finite Volumes Method (FVM) to solve partial differential equations. Fluent is fully integrated into the Ansys Workbench environment, a platform designed for efficient and flexible workflows, CAD associativity and powerful capabilities in geometry modeling and meshing. It can solve

diverse multiphysics problems, *e.g.*, fluid–structure interaction (FSI), multiphase flows, heat transfer between fluids and solids, chemical reactions and combustions. The software places special emphasis on providing a wide range of turbulence models and, in the last few years, a High Performance Computing (HPC) package [145]. Other commercial software able to perform CFD simulations with excellent performance could be: STAR-CCM+, FLOW-3D, COMSOL Multiphysics and XFlow.

STAR-CCM+ [146], produced by CD-Adapco, is a multidisciplinary simulation tool capable of simulating complex problems related to fluid dynamics and heat transfer in a simple way for a wide range of applications. In STAR-CCM+ the entire workflow, from the creation of the geometry to the analysis of the results, takes place within a single user interface. In addition, the wide range of physical models offered, allows the analysis of multiphase systems, dynamics of interacting particles and reacting fluids. FLOW-3D [147] is an accurate, fast, proven CFD software that solves the toughest free-surface flow problems. A pioneer in the CFD industry, and a trusted leader, FLOW-3D is a highly-efficient, comprehensive solution for free-surface flow problems with human-centric support. Modeling with COMSOL Multiphysics [148] allows to move in a single software environment between simulations of electromagnetism, structural mechanics, acoustics, fluid dynamics, heat transfer and chemical reaction phenomena, or any other physics modeled by a PDEs system. XFlow [149] is a next generation CFD software system that uses a proprietary state-of-the-art Lattice Boltzmann technology for high fidelity Computational Fluid Dynamics (CFD) applications as a part of SIMULIA's Fluids Simulation portfolio. This software is specifically designed to address complex CFD workflows involving high frequency transient simulations with real moving geometries, complex multiphase flows, free surface flows and fluid-structure interactions.

However, the perpetual license fee for a commercial software typically ranges from 10000 to 50000 Euros, or more, depending on the number of extra modules required and the license type requested, *e.g.*, Named License or Floating Network License. As an alternative, CFD software houses can usually release an annual license agreement. Clearly the investment costs of a general purpose software are not small, but high-quality experimental facilities are not usually a less expensive investment [150].

5.1.2 Free software

On the other hand, with "zero-costs", the open source world takes a place in contrast with commercial solution, well beyond the numerical landscape.

In the previous sentence, the term "zero-costs" is referred to the cost of the license only, without considering all the costs in terms of learning, understanding and writing source codes. In fact, these codes are provided for free under several different legal and often standardized licenses, such as the MIT (from the same university) or the GNU General Public License [151], with source codes available for free but, usually, implemented for academic and research scopes [152, 153]. The reason for this limited field of application is due to the need for a highly specialized profile capable of managing these codes, both from an IT and physical point of view.

In general, most of open source software are designed and coded for specific scientific purposes [154], while only few of them can handle wide ranges of engineering problems. Moreover, modeling complicated geometry is quite impossible using the default tools, hence, these must be created and imported from separate CAD program, with various difficulties. User guides and tutorials with few examples are provided on web-sites, but private support and training for general users is too demanding as a request.

A user friendly interface is completely absent, and in some cases it is possible to buy it, *e.g.*, the OpenFOAM GUI SimFlow [155], so that the open source philosophy loses its sense. Other problems are related to the abandonment of the implementation of the code, due to the lack of funds that support the development or to the weak success of the code itself. Many times these codes rely on external libraries, always open source, which usually perform linear algebra operations, *e.g.*, PETSc [156], Trilinos [157], SLEPc [158] and Eigen [159], parallelization functions, *e.g.*, OpenMP [160], Threading Building Blocks (TBB) [161] and MPI [162], and elaboration and visualization of results, *e.g.*, Paraview [163]. This dependence often leads to conflicts between the updated versions of the libraries and those of open-source software, with the consequence that very often it is difficult, if not impossible, to simply compile the source code. All these complications make open source codes unsuitable for an usage in industrial companies, where fast and reliable results are needed and without too specialized user know-how; in fact, they usually prefer invest in buying commercial software rather than training or hiring specialized personnel able to handle with in-house code [164, 165].

On the other hand, the flexibility offered by an open source code is well suited into a context of research and experimentation. The possibility to directly access to the source code allows to know the numerical implementation details, to modify it if necessary, but especially, to be able to independently manage the entire calculation process, from creating the mesh to post processing calculations [166]. On the contrary, it is not possible to have access to the source codes of commercial software, and the user is forced to accept, without seeing or sometimes knowing, the implementation choices made by someone else. For this reason the experimentation of innovative numerical methods with commercial codes is often avoided. Examples of open source codes able to perform CFD analysis are: OpenFOAM [167], FEniCS [168], FreeFEM [169] and deal.II [170].

In this thesis two different open source software are used to perform CFD analysis of LMD process: OpenFOAM and deal.II.

OpenFOAM Given the great success of OpenFOAM, a toolbox implemented in the object-oriented language C++ [171] ISO/IEC14882:2017, this software was originally chosen to implement the additive process that is the subject of the present work. It uses finite volume numerics to solve systems of partial differential equations discretized on a 3D unstructured mesh of polyhedral cells. The idea was to exploit the potential of the code, now affirmed in the CFD community, to simulate the flow of particles exiting from the LMD nozzle and interacting with a laser beam, exploring both a Lagrangian and an Eulerian approach. In particular,

`MPPICFoam` is the solver that uses the Lagrangian approach and `twoPhaseEulerFoam` is the one for the Eulerian approach, and both applications are available by default in the OpenFOAM package. To faithfully represent the LMD process, however, it was necessary to manually integrate a temperature field, interacting with both the fluid and the particles, not present in any of the two applications.

Theoretically, since the source code is available, the operation appears possible, but many difficulties have been encountered. Browsing on web there are many examples for altering an existing OpenFOAM solver to add, for example, a thermal transport equation [172]. Despite of the tutorials, extending this solver into more a complex one, such as `MPPICFoam`, it turned out to be a very laborious and tiring operation. In fact, after recompiling the source code, it was very difficult to navigate within the source code, which additionally was completely absent from all comments. The OpenFOAM community also has a forum, where users may help each other, but specific problems of a certain technical difficulty as this one does not seem to have been addressed.

In addition, there are also specific courses to learn using the software at an advanced level, but the payment fee of more than 1200 Euros for 12 hours of course, alters the open source philosophy. Because of these complications, it was decided to simulate only the flow of particles that interacts with the velocity field of the fluid, without including a thermal coupling. Section 5.2 reports the formulation and implementation of the problem with OpenFOAM.

FEniCS To overcome the problem, given the previous experience, it was essential to choose the right open source code to rely on for the implementation of the problem. Initially the choice fell on FEniCS, a popular open-source computing platform for solving partial differential equations [173], based on the C++ and Python [174] finite element library DOLFIN, that works as the main user interface of FEniCS. In particular, it provides a problem solving environment for models based on PDEs and implements core parts of the functionality of FEniCS, including data structures and algorithms for computational meshes and finite element assembly [168]. The main advantage of this software is the possibility to write the weak form of the equations in a very easy way, using the Unified Form Language (UFL) [175]. In particular, variational forms expressed in the UFL form language are passed to a specific compiler to generate Unified Form-assembly (UFC) code, which can then be used by DOLFIN to assemble linear systems. Despite these positive aspects, this software has many dependencies on external libraries, which has led to problems related to the installation of a stable version. In fact some users refused to update the software to a more recent version, choosing to work with a previously installed version, which was stable and functioning in their machines. This is due to poorly updated documentation, to an underactive community and to the limited use of the software itself. Lately the developers of FEniCS provide a version of the software on Docker, a Linux virtual machine container platform, but, however, it turns out an uncomfortable and limiting choice especially when debugging the code. All these problems have led to discard the use of FEniCS and to look for an alternative software that could meet our needs.

deal.II In the end the choice fell on deal.II [55], which is C++ software library supporting the creation of finite element codes and an open community of users and developers. The mission of the project is to provide a well-documented tool to build finite element codes for a broad variety of PDEs, from laptops to supercomputers. Moreover, in 2007 the authors won the J. H. Wilkinson Prize for Numerical Software for deal.II.

The documentation, very comprehensive and detailed, provides a wide range of clear tutorials, explaining step by step all the implementation details for the numerical resolution of the PDEs. The participation of its members is crucial for the growth of the project, which subsequently publishes and promotes softwares developed by the members themselves. This aspect has created a very active community on the forum dedicated to the development of deal.II, which allows continuous support to both the inexperienced and advanced users. For all these reasons it was decided to undertake the numerical simulation of the LMD process with deal.II; in Section 5.3 all details referred to the problem formulation and implementation are explained.

5.2 OpenFOAM: Problem formulation and implementation

As mentioned in Section 3.1.1, different CFD approaches can be adopted to model the physical problem of powder and gas mixture flow during LMD process. From a computational viewpoint, the crucial issue in terms of efficiency and accuracy is the simulation of the particles dynamics [176].

Two approaches can be followed to this purpose, both based on an Eulerian formulation. The former is a pure Eulerian approach, namely the Eulerian-Eulerian (EE) method, where powder and gas are both treated as continuous fluids, and coupled each other with the Navier-Stokes system of equations. The latter is a mixed approach, namely a Lagrangian-Eulerian (LE) method, working with a Lagrangian description of powders as particles that are tracked inside the problem domain, where gas dynamics is simultaneously described as an Eulerian incompressible flow, described by the Navier-Stokes equations.

In particular, in the LE approach, the powder diffusion is simulated with a Discrete Parcel Method (DPM) [102], that, instead of solving each individual particle as in the Discrete Element Method (DEM) [177, 178], identifies and tracks a parcel of particles which moves through the flow field. In other words, the parcel is a representative elementary volume of particles, so as to homogenize in a unique macro-particle all powder properties of such a volume (size, velocity, rotational rate, etc.).

This section aims at investigating the LMD printing process through both Eulerian-Eulerian and Lagrangian-Eulerian approaches, and compare them in terms of computational efficiency, implementation, and physical applicability. In particular, performance and accuracy are studied in order to reproduce the shape of the powder cone outside the deposition head, such as its minimum diameter and its positioning, which are the key factors commonly evaluated in the experiments. Both the approaches are implemented in a in-house code using OpenFOAM, a C++ toolbox whose language follows an object-oriented paradigm, allowing to develop customized numerical solvers for the solution of continuum mechanics problems, including CFD [179].

5.2.1 Eulerian-Eulerian (EE) approach

In the Eulerian-Eulerian (EE) approach, the powder-particle phase is treated as an additional continuous phase that interacts with the primary gas-fluid continuous phase. We indicate with $\boldsymbol{u}$ and $\boldsymbol{g}$ the velocity and gravity vector-fields, respectively, and with p the pressure scalar-field. Introducing an index i to indicate the general phase and assuming $i = \phi$ and $i = \sigma$ for the continuous fluid phase and the particle solid phase, respectively, the governing equations for the Eulerian-Eulerian model are the continuity equations for mass conservation (5.139a), and

the momentum balances for both continuous phases (5.139b), *i.e.*:

$$\frac{\partial \epsilon_i}{\partial t} + \nabla \cdot (\epsilon_i \, \boldsymbol{u}_i) = 0 \tag{5.139a}$$

$$\rho_i \frac{\partial (\epsilon_i \, \boldsymbol{u}_i)}{\partial t} + \rho_i \nabla \cdot (\epsilon_i \, \boldsymbol{u}_i \boldsymbol{u}_i) - \nabla \cdot \epsilon_i \boldsymbol{\tau}_i = -\nabla (\epsilon_i p_i) + \rho_i \epsilon_i \, \boldsymbol{g} - \boldsymbol{F}_{i,j} \tag{5.139b}$$

where ϵ_i describes the volumetric fraction of each phase:

$$0 \leq \epsilon_i \leq 1 \,, \quad \text{with} \quad \epsilon_\phi + \epsilon_\sigma = 1 \,; \tag{5.140}$$

the phase densities are indicated with ρ_i, while $\boldsymbol{\tau}_i$ is the stress tensor field of each phase:

$$\boldsymbol{\tau}_i = \nu_i \left(\nabla \boldsymbol{u}_i + (\nabla \boldsymbol{u}_i)^T \right) + \left(\lambda_i - \frac{2}{3} \nu_i \right) (\nabla \cdot \boldsymbol{u}_i) \, \boldsymbol{\delta} \tag{5.141}$$

$\nu_i = \mu_i / \rho_i$ being the i-th phase kinematic viscosity, while μ_i the corresponding dynamic viscosity, and λ_i is the bulk viscosity, with $\lambda_i = \lambda_\sigma$ if $i = \sigma$ and $\lambda_i = 0$ if $i = \phi$; $\boldsymbol{\delta}$ is the second-order identity tensor.

The term $\boldsymbol{F}_{i,j}$ is the momentum transfer term between the phases i and j; in particular, $\boldsymbol{F}_{\phi,\sigma}$ represents the forces acting on the gas-fluid phase caused by the particle-phase, and conversely, $\boldsymbol{F}_{\sigma,\phi}$ represents the forces acting on the particle-solid phase due to the gas-fluid phase. These forces of momentum transfer are related each other according to the Newton third law:

$$\boldsymbol{F}_{\phi,\sigma} + \boldsymbol{F}_{\sigma,\phi} = 0 \tag{5.142}$$

Referring to [180], such forces represent just the drag forces contributions, which are usually considered as the most meaningful ones (see also [181]). We then assume:

$$\boldsymbol{F}_{\phi,\sigma} = \beta \, (\boldsymbol{u}_\phi - \boldsymbol{u}_\sigma) \tag{5.143}$$

$$\boldsymbol{F}_{\sigma,\phi} = \beta \, (\boldsymbol{u}_\sigma - \boldsymbol{u}_\phi) \tag{5.144}$$

where β is a momentum exchange coefficient. Several expressions are proposed in the literature to tune such a coefficient. We adopt the drag correlation coefficient proposed by Gidaspow [182], where the estimated values that Ergun found out for gas volume fractions lower than 0.8, and the ones found out by Wen and Yu for gas volume fractions greater than 0.8, are combined each other (see again [180] for more details). The momentum exchange coefficient turns out to obey to the following expressions:

$$\beta = \begin{cases} 150 \dfrac{\epsilon_\sigma^2 \mu_\phi}{\epsilon_\phi d_\sigma^2} + 1.75 \epsilon_\sigma \dfrac{\rho_\phi}{d_\sigma} |\boldsymbol{u}_\phi - \boldsymbol{u}_\sigma|, & \epsilon_\phi < 0.8 \\ \dfrac{3}{4} C_D \dfrac{\epsilon_\phi \epsilon_\sigma}{d_\sigma} \rho_\phi |\boldsymbol{u}_\phi - \boldsymbol{u}_\sigma| \epsilon_\phi^{-2.65}, & \epsilon_\phi \geq 0.8 \end{cases} \tag{5.145}$$

where C_D is the drag coefficient

$$C_D = \begin{cases} \dfrac{24}{Re_\sigma} (1.0 + 0.15 \, Re_\sigma^{0.687}), & Re_\sigma \leq 1000 \\ 0.44, & Re_\sigma > 1000 \end{cases} , \tag{5.146}$$

depending on the particle Reynolds number Re_σ for the solid phase, which is averaged on the cell, and expressed in terms of the average particle diameter d_σ, *i.e.*,

$$Re_\sigma = \frac{\epsilon_\phi \, d_\sigma \, |\boldsymbol{u}_\phi - \boldsymbol{u}_\sigma| \, \rho_\phi}{\mu_\phi} \,. \tag{5.147}$$

5.2.2 Lagrangian-Eulerian (LE) approach

In the Lagrangian-Eulerian (LE) approach, the particle-solid phase is treated through a Lagrangian representation that models and computes the particle evolution. The resulting Lagrangian description is then coupled with the gas-fluid phase, described exactly as in the above mentioned Eulerian model.

The particle evolution is determined on the basis of a particle distribution function $\xi(\boldsymbol{x}_\sigma, \boldsymbol{u}_\sigma, \rho_\sigma, V_\sigma, t)$, varying on time t and depending on position $\boldsymbol{x}_\sigma$, velocity $\boldsymbol{u}_\sigma$, particle density ρ_σ and volume of the particle V_σ, respectively. From now on, we will adopt the term *parcel*, as the computational entity consisting of N_σ particles – kept constant in time – together with an homogenized set of value $(\boldsymbol{x}_\sigma, \boldsymbol{u}_\sigma, \rho_\sigma, V_\sigma)$.

The distribution function ξ obeys a phase continuity conditions and, in particular, satisfies the following transport equation in the phase space:

$$\frac{\partial \xi}{\partial t} + \nabla \cdot (\xi\, \boldsymbol{u}_\sigma) + \nabla_v \cdot (\xi\, \boldsymbol{a}_\sigma) = 0 \tag{5.148}$$

where ∇_v is the divergence operator respect with to the velocity field, while $\boldsymbol{a}_\sigma$ is the parcel acceleration expressed as:

$$\boldsymbol{a}_\sigma = \beta\,(\boldsymbol{u}_\phi - \boldsymbol{u}_\sigma) - \frac{1}{\rho_\sigma}\nabla p + \boldsymbol{g} - \frac{1}{\epsilon_\sigma \rho_\sigma}\nabla \boldsymbol{\tau}_\sigma \tag{5.149}$$

β being the momentum exchange coefficient already defined in (5.145), and $\boldsymbol{\tau}_\sigma$ the parcel normal stress. According to Harris and Crighton model [183], $\boldsymbol{\tau}_\sigma$ is obtained on the basis of experimental evidences, and its expression turns out to be:

$$\boldsymbol{\tau}_\sigma = \frac{p_\sigma\, \epsilon_\sigma^\alpha}{\max(\overline{\epsilon}_\sigma - \epsilon_\sigma, \zeta(1-\epsilon_\sigma))} \tag{5.150}$$

where p_σ is a constant value of pressure (namely, 100 Pa), α is a dimensionless empirical constant (ranging between 2 and 5), ζ is an empirical small number of the order to 10^{-7}, while $\overline{\epsilon}_\sigma$ is a limit value of parcel volume fraction, meaning the maximum value of particles in cubic close-packing mode [183]. The solid volume fraction ϵ_σ is defined by integrating the parcels distribution function as:

$$\epsilon_\sigma = \iiint \xi\, V_\sigma\, dV_\sigma\, d\rho_\sigma\, d\boldsymbol{u}_\sigma \tag{5.151}$$

Once the distribution function ξ is solved by time-integration of (5.148), the parcels velocity and position are updated in the n-th time step as:

$$\begin{aligned} \boldsymbol{u}_\sigma^{n+1} &= \boldsymbol{u}_\sigma^n + \Delta t\, \boldsymbol{a}_\sigma^n \\ \boldsymbol{x}_\sigma^{n+1} &= \boldsymbol{x}_\sigma^n + \Delta t\, \boldsymbol{u}_\sigma^{n+1} \end{aligned} \tag{5.152}$$

Note that the collision between the parcels and the wall is taken into consideration for assuming the following conditions

$$\begin{aligned} \boldsymbol{u}_{\sigma,n}^{n+1} &= -e\, \boldsymbol{u}_{\sigma,n}^n \\ \boldsymbol{u}_{\sigma,t}^{n+1} &= (1-f_r)\boldsymbol{u}_{\sigma,n}^n \end{aligned} \tag{5.153}$$

where $\boldsymbol{u}_{\sigma,n}$ and $\boldsymbol{u}_{\sigma,t}$ are the normal and tangential velocities, respectively, and e the coefficient of restitution, whereas f_r the coefficient of kinetic friction. Such coefficients are evaluated on the basis of the specific test to solve, and they will be specified later in the conducted numerical campaign.

Note finally that the momentum transfer term $\boldsymbol{F}_{\sigma,\phi}$ can be obtained from the expression of the parcel acceleration (5.149), and in line with (5.151) it can be written as:

$$\boldsymbol{F}_{\sigma,\phi} = \iiint \xi \, V_\sigma \, \rho_\sigma \left[\beta \left(\boldsymbol{u}_\phi - \boldsymbol{u}_\sigma \right) - \frac{1}{\rho_\sigma} \nabla p \right] \, dV_\sigma \, d\rho_\sigma \, d\boldsymbol{u}_\sigma \tag{5.154}$$

5.2.3 Implementation details

In the Eulerian-Eulerian model, two separate continuous problems for each phase are solved, whence they interact each with other through the interphase transfer equations (5.142) computing the interphase terms $F_{i,j}$. In the Eulerian-Lagrangian model, the domain of solution is solved only for continuous phase. The parcel path is recovered according to eqs. (5.148)-(5.154) within a Lagrangian framework. The Lagrangian simulations are carried out by a particular DPM (Discrete Parcel Method) solver of OpenFOAM, `MPPICFoam` (MultiPhase Particle-In-Cell method [184, 185, 186, 187, 188, 189, 190, 191]), that implements a Lagrangian approach for dense flows. On the other hand, the `twoPhaseEulerFoam` [191] solver is set up to solve the problem with the Eulerian-Eulerian approach. These approaches are described in the OpenFOAM documentation [54].
In this section some numerical details will be provided for the Eulerian computations, while the interested reader can find all the details for the Lagrangian ones in several here mentioned papers, *e.g.*, in [187, 192, 191].

Note that in the Lagrangian-Eulerian approach several quantities computed in Lagrangian form have to be evaluated in Eulerian form. This point is solved by coarse averaging procedures, that interpolates the Lagrangian properties (i.e., particle volume fraction, particle velocity, and fluid-particle interaction force) from their discrete values to the corresponding Eulerian properties defined on the Eulerian grid. For more details, see [193].

Discretization of the Navier-Stokes equations Within the Navier-Stokes equations for incompressible flows as adopted in (5.139), the convection term $\nabla \cdot (\epsilon_i \, \boldsymbol{u}_i \boldsymbol{u}_i)$ represent the nonlinear contribution to the problem. Several nonlinear solvers can be implemented to handle such a term in the equations but these are in general quite expensive and impractical [194, 195, 196, 197]. Therefore, we linearize the convection term: for a control cell-volume V_σ, with faces f centered around a cell-point P, and all variables defined at the cell center, we use the discretized version of the Gauss theorem to express the discretized version of the convection term as a linear combination of the velocity variables:

$$\int_{V_\sigma} \nabla \cdot (\epsilon_i \, \boldsymbol{u}_i \, \boldsymbol{u}_i) dV = \sum_f \Phi \, \boldsymbol{u}_f \approx a_p \, \boldsymbol{u}_p + \sum_n a_n \, \boldsymbol{u}_n \tag{5.155}$$

All the coefficients a_n and a_p collect the known fluxes $\Phi = \boldsymbol{S} \cdot (\epsilon_i \, \boldsymbol{u}_i)_f$ across the f-th face of the control volume, and the velocity values at the f-th face are interpolated from the control volume and neighboring volumes by means of the operator $\boldsymbol{S}$, so that $\boldsymbol{u}_p$ represents the velocity for the control cell-volume, $\boldsymbol{u}_n$ the one of neighboring volumes. Note that within this procedure we implicitly assume the phase fraction ϵ_i to be known when computing $\boldsymbol{u}_p$. Moreover, the continuity equation (5.139a) can be written as:

$$\frac{\partial \epsilon_i}{\partial t} + \sum_f \boldsymbol{S} \cdot (\epsilon_i \, \boldsymbol{u}_i)_f = 0 \tag{5.156}$$

when considering some explicit discretization in time.

Derivation of the pressure equation Once the convection term is linearized, at a given time-step the entire equation (5.139b) in its discrete form can be represented as follows:

$$\begin{aligned} \left(a_p' - \frac{f_p}{\rho_i} \right) \boldsymbol{u}_p &= - \sum_n a_n \, \boldsymbol{u}_n + a_0 \, \boldsymbol{u}_0 + \boldsymbol{g} + \frac{\boldsymbol{F}_{i,j}}{\rho_i} - \nabla p \\ a_p \, \boldsymbol{u}_p &= \boldsymbol{H}(\boldsymbol{u}_i) - \nabla p \\ \boldsymbol{u}_p &= \frac{\boldsymbol{H}(\boldsymbol{u}_i)}{a_p} - \frac{\nabla p}{a_p} \end{aligned} \tag{5.157}$$

where all quantities having subscript "$_0$" refer to the known solution at the previous time-step. In the above equation all the diagonal coefficients of the linearized discrete problem are assembled in one coefficient a_p, while $\boldsymbol{H}(\boldsymbol{u}_i)$ collects all off-diagonal contributions of the same linear system, including source terms.
As concerning continuity conditions (5.139a) and then (5.156), the velocity of the f-th cell face can be computed as follows:

$$\boldsymbol{u}_f = \left(\frac{\boldsymbol{H}(\boldsymbol{u}_i)}{a_p} \right)_f - \left(\frac{\nabla p}{a_p} \right)_f \tag{5.158}$$

intending here all quantities interpolated at the f-th face itself. By combining equation (5.156) with (5.158) we obtain the pressure equation:

$$\sum_f \boldsymbol{S} \cdot \epsilon_f \left(\frac{\nabla p}{a_p} \right)_f = \frac{\partial \epsilon_i}{\partial t} + \sum_f \boldsymbol{S} \cdot \epsilon_f \left(\frac{\boldsymbol{H}(\boldsymbol{u}_i))}{a_p} \right)_f \tag{5.159}$$

so that the flux Φ appearing in (5.155) can be computed as:

$$\Phi = \boldsymbol{S} \cdot (\epsilon_i \boldsymbol{u}_i)_f = \boldsymbol{S} \cdot \epsilon_f \left[\left(\frac{\boldsymbol{H}(\boldsymbol{u}_i))}{a_p} \right)_f - \left(\frac{\nabla p}{a_p} \right)_f \right] \tag{5.160}$$

`PIMPLE` **solver** The solution procedures adopted for both Eulerian and Lagrangian approaches rely on the `PIMPLE` algorithm [198], that merges the PISO [199] algorithm, addressed to recover a pressure correction for the problem, and the SIMPLE [200] algorithm, working as relaxation step of the problem variables. The `PIMPLE` algorithm, that provides at each time-step the

velocity vectors $\boldsymbol{u}_i$, proved to have faster convergence than algorithms using such numerical schemes separately (see [198] for more details).

The following steps are iteratively performed in `PIMPLE`:

1. assemble the discretized equation for $\boldsymbol{u}_i$ on the grid without any source terms: a coefficient matrix is set up according to $(5.157)_1$, source terms not included;
2. relax the equation;
3. solve the equation (5.157) with a trial value of the pressure field, namely p_0, that corresponds to the pressure field in the previous time-step: such a solution allows us to obtain the momentum predictor $\boldsymbol{u}^\star$ at the cell centers;
4. using the predictor $\boldsymbol{u}^\star$, assemble the operator $\boldsymbol{H}(\boldsymbol{u}^\star)$ and interpolate both a_p and $\boldsymbol{H}(\boldsymbol{u}^\star)$ to the cell faces;
5. solve the pressure equation (5.159) at the cell faces;
6. update by (5.160) the flux field at the cell faces;
7. relax the pressure;
8. update by (5.157) the velocity at the cell centres;
9. update the boundary conditions for consistency.

5.3 Deal.II: Problem formulation and implementation

The coupling between the fluid and temperature fields characterizes a variety of industrial applications, like heat exchangers, spent fuel storage of nuclear power plants, solar collectors, crude oil storage tanks, energy storage devices, and modern crop dryers [201]. Due to such an heterogeneous span of applications, two different classes of fluid-thermal coupling problems can be identified: (i) in presence of low fluid velocities Stokes equations can be used to properly capture the phenomenon, otherwise (ii) when fluid velocities are high, hence when the coupling between the fluid behavior and the heat transfer gives rise to non-negligible advection-diffusion effects, Navier-Stokes equations must be employed. In this work we focus on this latter class of problems.

It is well known that, due to the intrinsic nature of advection-diffusion governing equations, numerical solutions of the momentum equation of the Navier-Stokes system and the heat equation, either as separate or as coupled equations, may be affected by spurious oscillations, typically associated with the rate of the velocity fields. In this field, literature is really extensive and covers a broad variety of general numerical methods [59], but also dedicated numerical technologies for specific applications [202, 203].

The present Section introduces a numerical approach, developed as modified Newton-Raphson scheme and embedded in a time-marching algorithm, to accurately and efficiently solve coupled problems involving incompressible fluids and temperature heat exchange [204, 57]. Such an approach tackles the fully nonlinear formulation of the Navier-Stokes equation, without introducing any linearization nor stabilization contribution in the ensuing discrete equations. The accuracy of the proposal approach will be assessed by controlling first the incompressibility constraint of the Navier-Stokes problem, and then the thermal convection of the heat transfer equation, which are considered to be reliable measures for the solution of the coupled problem.

Implemented in an in-house `C++` code based on the open-source library of `deal.II` [170], the proposed approach will be validated by comparison with several results available in the literature, both for fluid motion and for coupled fluid and thermal convections. Besides the numerical approach the present Section also discusses the development of two additional novel schemes, obtained by extending two projection methods successfully employed for Navier-Stokes problems [59, 110] to coupled thermal-fluid problems. Such two alternatives, namely the Projection-Correction scheme [110] and the iterated Projection-Correction scheme [59], adapted to the coupled problem, allow us to argue on the efficiency of our proposal in terms of CPU times.

Then, the following Section introduces our modified Newton-Raphson algorithm aimed at solving the coupled Navier-Stokes and heat transfer equations introduced in the previous chapter. Additionally, in order to test and benchmark our proposal, we present two alternative schemes, coming from linearized approaches to incompressible flow problems, a one-step pressure-correction scheme and an iterated pressure-correction scheme.

5.3.1 Navier-Stokes equations: Newton-Raphson Method

By employing a standard Forward Euler Method (see Section 4.3) for time discretization, *i.e.*, $\partial_t \mathbf{u}\,\Delta t = \mathbf{u}_{n+1} - \mathbf{u}_n$, where $\mathbf{u}_{n+1}$ is the velocity field at the time step $n+1$, and $\mathbf{u}_n$ is the velocity field at the previous time step n, let us rearrange the Navier-Stokes equations (4.4), along with the continuity ones (4.5), in the following form:

$$\mathbf{r}(\mathbf{u}_{n+1}, p_{n+1}) = \begin{pmatrix} \frac{1}{\Delta t}\left(\mathbf{u}_{n+1} - \mathbf{u}_n\right) - \nu\Delta\mathbf{u}_{n+1} + (\mathbf{u}_{n+1}\cdot\nabla)\mathbf{u}_{n+1} + \nabla p_{n+1} - \mathbf{f}_{n+1} \\ -\nabla\cdot\mathbf{u}_{n+1} \end{pmatrix}. \tag{5.161}$$

By collecting in a discrete vector $\mathbf{x}^0 = \{\mathbf{u}_n,\, p_n\}$, serving as initial guess for the iterative scheme, *i.e.*, the vector collecting both velocity and pressure discrete vectors at the time step n, a classical Newton-Raphson approach applied to such a problem (see for instance [111]) consists of the following system of equation:

$$\mathbf{J}(\mathbf{x}^k)\,\delta\mathbf{x}^k = -\mathbf{r}(\mathbf{x}^k)\,, \tag{5.162}$$

where $\delta\mathbf{x} = \mathbf{x}^{k+1} - \mathbf{x}^k$ is the update term, $\mathbf{x}^{k+1}$ is the approximate solution of the Newton iteration $k+1$, $\mathbf{x}^k$ represents the solution from the previous Newton iteration k, and $\mathbf{J}(\mathbf{x}^k) = \nabla\mathbf{r}(\mathbf{x}^k)$ is the *Jacobian matrix* evaluated at $\mathbf{x}^k$, for $k = 0,\, 1,\, \ldots,\, \bar{k}$, being $\bar{k}$ the last iteration where the convergence is reached at the solution $\mathbf{x}^{\bar{k}} = \{\mathbf{u}_{n+1},\, \mathbf{p}_{n+1}\}$. The left hand side of (5.162) represents the *directional gradient* of $\mathbf{r}(\mathbf{x})$ along $\delta\mathbf{x}^k$ at $\mathbf{x}^k$. By definition, the directional gradient is given by the following:

$$\begin{aligned}
&\nabla\mathbf{r}(\mathbf{u}^k, p^k)\,(\delta\mathbf{u}^k, \delta p^k) \\
&= \lim_{\epsilon\to 0}\frac{1}{\epsilon}\left(\mathbf{r}\left(\mathbf{u}^k + \epsilon\delta\mathbf{u}^k, p^k + \epsilon\nabla\delta p^k\right) - \mathbf{r}(\mathbf{u}^k, p^k)\right) \\
&= \lim_{\epsilon\to 0}\frac{1}{\epsilon}\begin{pmatrix} \delta\mathbf{u}^k + \Delta t\left(-\epsilon\nu\Delta\delta\mathbf{u}^k + \epsilon\mathbf{u}^k\cdot\nabla\delta\mathbf{u}^k + \epsilon\delta\mathbf{u}^k\cdot\nabla\mathbf{u}^k + \epsilon^2\delta\mathbf{u}^k\cdot\nabla\delta\mathbf{u}^k + \epsilon\nabla\delta p^k\right) \\ \Delta t(-\epsilon\nabla\cdot\delta\mathbf{u}^k) \end{pmatrix} \\
&= \begin{pmatrix} \delta\mathbf{u}^k + \Delta t\left(-\nu\Delta\delta\mathbf{u}^k + \mathbf{u}^k\cdot\nabla\delta\mathbf{u}^k + \delta\mathbf{u}^k\cdot\nabla\mathbf{u}^k + \nabla\delta p^k\right) \\ \Delta t(-\nabla\cdot\delta\mathbf{u}^k) \end{pmatrix},
\end{aligned} \tag{5.163}$$

leading to the following linearized system:

$$\begin{aligned}
\delta\mathbf{u}^k + \Delta t\left(-\nu\Delta\delta\mathbf{u}^k + \mathbf{u}^k\cdot\nabla\delta\mathbf{u}^k + \delta\mathbf{u}^k\cdot\nabla\mathbf{u}^k + \nabla\delta p^k\right) &= -\mathbf{r}(\mathbf{x}^k)\,, \\
\Delta t(-\nabla\cdot\delta\mathbf{u}^k) &= \nabla\cdot\mathbf{u}^k\,,
\end{aligned} \tag{5.164}$$

where $\mathbf{u}^k$ and p^k are the velocity and pressure solutions from the previous iteration. Furthermore, the right hand side of the second equation is not zero since the discrete solution is not exactly divergence free (divergence free for the continuous solution). The right hand side here acts as a correction which leads the discrete solution of the velocity to be divergence free along Newton's iteration.

Modified Newton-Raphson Algorithm

In a classical Newton method, described in the previous paragraph, the most expensive routine is the computation of the inverse of the Jacobian matrix $\mathbf{J}(\mathbf{x}^k)$. To avoid its assembly for every Newton iteration, we may "freeze" the Jacobian within the same time-step, hence considering it constant in each Newton iteration.

This approach uses the inverse of the Jacobian matrix, computed at the beginning of the iteration loop, to find the solution at every further iteration. In particular, we will implement a LU decomposition of the Jacobian matrix, that, as well known, acts in two steps, the factorization step done out of the iteration loop, and the subsequent solution at each iteration, allowing to save noticeably CPU time [205] for both parallel and sequential computations.

We may rewrite the classical Newton method as follows:

$$\delta\mathbf{x}^k = -\left(\mathbf{J}^0\right)^{-1}\mathbf{r}(\mathbf{x}^k)\,, \tag{5.165}$$

where $\mathbf{J}^0$ is the Jacobian matrix assembled at $\mathbf{x}^0$, the unknown vector at the first Newton iteration, and hence it needs to be calculated once for every time-step. In this case, the system of equations to be solved at each k-th iteration is therefore $\mathbf{J}^0\ \delta\mathbf{x}^k = -\mathbf{r}(\mathbf{x}^k)$. The complete algorithm outline is presented in Algorithm 1, whereas for a graphical comparison between the classical and the modified Newton-Raphson method we can refer to Figure 5.5.

At every iteration, a multifrontal sparse direct solver (see Section 4.4.1) provided by the

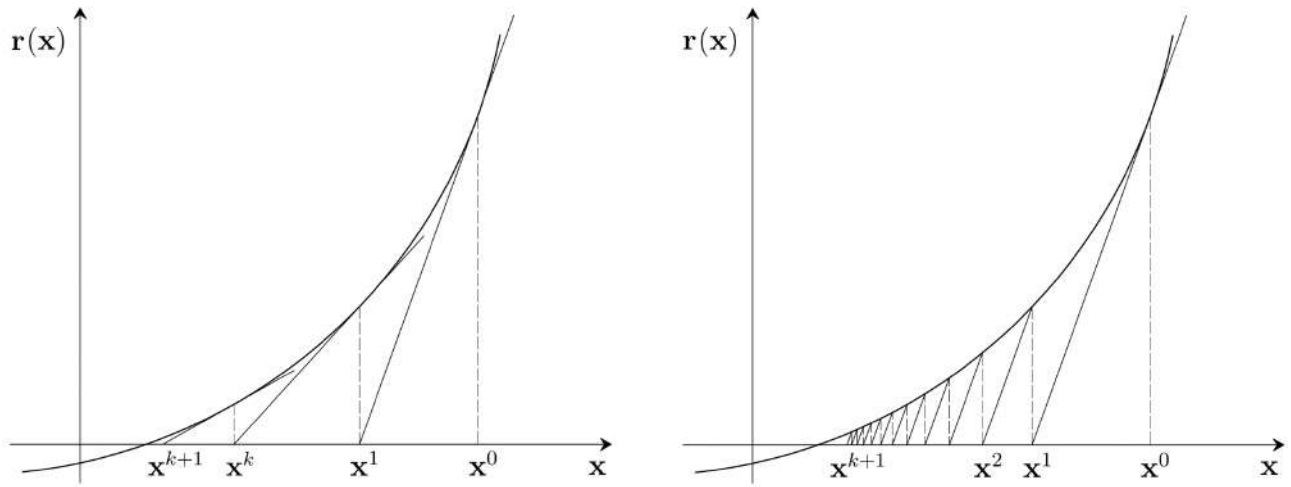

Figure 5.5: Comparison between the classical, on the left, and the modified, on the right, Newton-Raphson method.

deal.II library have been used. In particular, the SparseDirectUMFPACK Class is the interface implemented to deal with sparse direct solver UMFPACK, which is part of the SuiteSparse library [206]. UMFPACK gathers a set of routines for solving non-symmetric sparse linear systems, such as Ax=b, using the Unsymmetric-pattern MultiFrontal method and direct sparse

LU factorization. Furthermore, matrices may have symmetric or unsymmetric sparsity patterns, and may have unsymmetric entries.

Algorithm 1 Modified Newton-Raphson scheme.

function MNR($\Delta t, \mathbf{u}_n, \mathbf{p}_n$)
  $\mathbf{x}^0 = \{\mathbf{u}_n,\ \mathbf{p}_n\}$ ▷ initialize from last solution in step n
  $\mathbf{J}^0 = \mathbf{J}\left(\mathbf{x}^0\right)$ ▷ assembly and LU decompose the Jacobian matrix

  for $k = 1, \ldots, k_{\max}$ **do** ▷ Newton-Raphson loop

    $\mathbf{r}^k = \mathbf{r}(\mathbf{x}^k)$ ▷ assembly the residue
    $\mathbf{J}^0\,\delta\mathbf{x} = -\mathbf{r}^k$ ▷ linear solve by back-substitution for $\delta\mathbf{x}$
    $\mathbf{x}^{k+1} = \mathbf{x}^k + \delta\mathbf{x}$ ▷ update the solution
    err $= \left|\,(\mathbf{r}^k)^\top \delta\mathbf{x}\,\right|$ ▷ check residual norm
    if err $\leq$ toll. **then**
      break
    end if

  end for

  return $(\mathbf{u}_{n+1}, p_{n+1})$
end function

5.3.2 The stabilized heat equation

The system (3.1) represents the temperature scalar field T coupled with the velocity vector field $\boldsymbol{u}$. As we can see, this coupling takes place combining the classical Navier-Stokes equations for the velocity field, with the heat (or energy) equation for the temperature field, through an advection field $\boldsymbol{u}$, that is the velocity itself, in this case.

As widely discussed is Section 5.3.2, this variable particularly influences the Péclet number in the heat equation, and when it becomes more than one, it starts to produce some oscillations in the solution of the temperature field. This means that the heat equation has to stabilized to avoid the appearance of wiggles, and the Streamline Upwind Petrov-Galerkin (SU/PG) method [60, 130, 131, 207] is implemented for this purpose. Therefore, testing the heat equation with functions $\tau\boldsymbol{u}\cdot\nabla w$ instead of τ (cf. equations (4.61) and (4.85)), we can implement the following algorithm (see Algorithm (2)) based on the stabilized version of the semi-discrete advection-diffusion equation (4.81).

Finally, the fully discrete form of the heat equation is written using the Multistep Backward Difference Formula (see Section 4.3.2) of order 2 (BDF2), whereas, exploiting the solvers available in the deal.ii library, the SolverGMRES Class (refer to Section 4.4.2 for more details regarding the GMRES method) is used to solve the nonsymmetric linear system of heat equations (5.166), which, written in time discretization form leads to

$$\frac{1}{2\Delta t}\bar{\mathbf{M}}\left(\mathbf{u}\right)\left(3\mathbf{T}_{n+1} - 4\mathbf{T}_n + \mathbf{T}_{n-1}\right) + \left[\bar{\mathbf{K}}\left(\mathbf{u}\right) + \bar{\mathbf{C}}\left(\mathbf{u}\right)\right]\mathbf{T}_{n+1} = \bar{\mathbf{f}}\left(\mathbf{u}\right) \tag{5.166}$$

As concerns assumptions for time discretization, we here considered a backward differentiation of second order to integrate in time the equations (3.1)-(3). Indeed, the bilinearity of the coupling term requires to discretize heat equations in time of an order greater than the (linear) one introduced for the velocity field in (5.161), so as to limit further time-discretization errors. Algorithm 2 outlines the temperature numerical solution working with such a second order approximation in time, and it is intended to be called after Algorithm 1, *i.e.* at each time step the temperature values are updated once the velocity values have achieved convergence in that time step.

Algorithm 2 Stabilized Temperature scheme.

function $\mathrm{T}(\Delta t, \mathbf{u}_{n+1}, \mathbf{T}_{n-1}, \mathbf{T}_n)$

$\overline{\mathbf{M}}(\mathbf{u}_{n+1}) = \mathbf{M} + \mathbf{M}^{\mathrm{SUPG}}(\mathbf{u}_{n+1})$ ▷ update stabilization terms
$\overline{\mathbf{K}}(\mathbf{u}_{n+1}) = \mathbf{K} + \mathbf{K}^{\mathrm{SUPG}}(\mathbf{u}_{n+1})$
$\overline{\mathbf{C}}(\mathbf{u}_{n+1}) = \mathbf{C}(\mathbf{u}_{n+1}) + \mathbf{C}^{\mathrm{SUPG}}(\mathbf{u}_{n+1})$
$\overline{\mathbf{F}}(\mathbf{u}_{n+1}) = \mathbf{F} + \mathbf{F}^{\mathrm{SUPG}}(\mathbf{u}_{n+1})$

$\bar{\mathbf{f}}\left(\mathbf{u}_{n+1}\right) = \frac{1}{2\Delta t}\bar{\mathbf{M}}\left(\mathbf{u}_{n+1}\right)\left(3\mathbf{T}_{n+1} - 4\mathbf{T}_n + \mathbf{T}_{n-1}\right) + \left[\bar{\mathbf{K}}\left(\mathbf{u}_{n+1}\right) + \bar{\mathbf{C}}\left(\mathbf{u}_{n+1}\right)\right]\mathbf{T}_{n+1}$ ▷ solve for $\mathbf{T}_{n+1}$

return $\mathbf{T}_{n+1}$
end function

5.3.3 Navier-Stokes equations: Projection Methods

In order to compute in an efficient way the solutions of incompressible Navier-Stokes equations, primarily Chorin [208] and the Temam [209], introduced the *Projection method*, that is one of the most popular numerical strategy used in applications that treat viscous incompressible flows.

The philosophy behind projection methods for incompressible Navier-Stokes equations is quite simple and it is based on the Lagrange multiplayer role of the pressure, which ensures the incompressibility constraint without carrying any thermodynamic meaning. In fact, thanks to this observation a time-splitting discretization scheme which decouples the computation of velocity and pressure is possible [210]. Precisely, the key feature of the projection method is the computational splitting in two main step: in the first one the incompressibility constraint is ignored and an intermediate velocity field $\boldsymbol{u}^*$ is computed using the momentum equation, whereas, in the second step, the intermediate velocity is projected back to the space of incompressible vector fields to obtain $\boldsymbol{u}_{n+1}$ and p_{n+1}. This procedure results very efficient but a small price has to be paid. In particular it introduces a numerical boundary layer on the pressure approximations and the intermediate velocity fields. Therefore, the critical phase in the implementation of projection methods is the treatment of boundary conditions.

The Helmholtz–Hodge decomposition

The Helmholtz-Hodge decomposition is the base of projection method's algorithms, in which a vector filed $\boldsymbol{u}$ is decomposed into a solenoidal (divergence-free) part $\boldsymbol{u}_{sol}$ and an irrotational part $\boldsymbol{u}_{irrot}$. In particular,

$$\boldsymbol{u} = \boldsymbol{u}_{sol} + \boldsymbol{u}_{irrot} = \boldsymbol{u}_{sol} + \nabla\phi \tag{5.167}$$

where $\nabla \times \nabla\phi = 0$ for some scalar function ϕ. Then, calculating the divergence of equation (5.167), we can write

$$\nabla \cdot \boldsymbol{u} = \nabla^2\phi \qquad (\text{since, } \nabla \cdot \boldsymbol{u}_{sol} = 0) \tag{5.168}$$

that is the Poisson equation for a scalar function ϕ. Therefore, if both the vector field $\boldsymbol{u}$ and the scalar function ϕ are known, the divergence-free part of $\boldsymbol{u}$, can be extract simply using the following relation

$$\boldsymbol{u}_{sol} = \boldsymbol{u} - \nabla\phi \tag{5.169}$$

This is the theoretical mathematical background behind solenoidal projection methods for solving incompressible Navier–Stokes equations.

The Chorin's projection method

In the original version of the projection method implemented by Chorin [208], the intermediate velocity $\boldsymbol{u}^*$ is computed using the momentum equation (4.4) by ignoring the pressure gradient term

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}_n}{\Delta t} = -\left(\boldsymbol{u}_n \cdot \nabla\right)\boldsymbol{u}_n + \nu\nabla^2\boldsymbol{u}_n \tag{5.170}$$

where $\boldsymbol{u}_n$ is the velocity field at n^{th} time step. After calculating $\boldsymbol{u}^*$ the projection step takes place. In the second part of the algorithm the intermediate velocity is corrected in order to obtain the final solution $\boldsymbol{u}_{n+1}$ solving the following equation at the $n+1$ time step,

$$\frac{\boldsymbol{u}_{n+1} - \boldsymbol{u}^*}{\Delta t} = -\frac{1}{\rho}\nabla p_{n+1} \tag{5.171}$$

This shows that the algorithm is based on an operator splitting approach, where the equation of the first step (5.170) account for viscous forces, whereas the equation of the second step (5.171) considers pressure forces. In order to compute the final velocity vector field through equation (5.171), the knowledge of the pressure p_{n+1} is required. This is computed taking the divergence of equation (5.171) and imposing that $\nabla \cdot \boldsymbol{u}_{n+1} = 0$, that is the continuity condition, deriving the above mentioned Poisson equation for p_{n+1}

$$\nabla^2 p_{n+1} = \frac{\rho}{\Delta t}\nabla \cdot \boldsymbol{u}^* \tag{5.172}$$

The boundary conditions for $\boldsymbol{u}^*$ in the equation (5.170) do not lead to specific problems. Whereas, the specification of boundary condition for equation (5.171) are a tricky decision [210]. If the space of divergence-free vector fields is imposed to be orthogonal to the space of irrotational vector fields, it must satisfy the boundary condition:

$$\boldsymbol{u} \cdot \boldsymbol{n} = 0 \qquad \text{on } \partial\Omega \tag{5.173}$$

Then, for equation (5.171) one has

$$\frac{\partial p_{n+1}}{\partial \boldsymbol{n}} = 0 \qquad \text{on } \partial\Omega \tag{5.174}$$

that ensure the standard Helmholtz–Hodge decomposition. This kind of boundary condition is strongly favored, but is responsible for errors that appear close to the boundary of the domain since the real pressure does not satisfy such boundary conditions.

Pressure Correction Method

Alternative approaches belonging to the class of "projection methods" can be followed to solve faster the Navier-Stokes equations paying a loss in terms of accuracy of the solution [58]. Projection methods, as in the early works [208, 209], deal with the incompressibility constraint of the Navier-Stokes equations by decoupling pressure and velocity variables and developing a fractional-step time-marching algorithm, where only viscous effects are first accounted for, and a correction is subsequently done in the divergence-free space through auxiliary velocity variables suitable for a pressure adjustment. In particular, we adopt here the procedure employed in [58] to handle the coupled problem of advection-diffusion.

The advection term $\mathbf{u}\cdot\nabla\mathbf{u}$ is first replaced by its skew symmetric form

$$\mathbf{u}\cdot\nabla\mathbf{u} + \frac{1}{2}(\nabla\cdot\mathbf{u})\mathbf{u}\,. \tag{5.175}$$

This is consistent with the continuity equation $\nabla\cdot\mathbf{u}=0$—though this is not true pointwise for the discrete solution—and it is needed to guarantee unconditional stability of the time-stepping scheme. Moreover, in order to linearize the term above following a second-order BDF2 time-stepping scheme, a second order extrapolation $\mathbf{u}_\star$ of $\mathbf{u}_{n+1}$, *i.e.*, $\mathbf{u}_\star = 2\mathbf{u}_n - \mathbf{u}_{n-1}$, is used to rewrite the term in (5.175) as:

$$\mathbf{u}_\star\cdot\nabla\mathbf{u}_{n+1} + \frac{1}{2}(\nabla\cdot\mathbf{u}_\star)\mathbf{u}_{n+1}\,. \tag{5.176}$$

Similarly, a second order extrapolation is used to approximate the pressure field as

$$p_\sharp = p_n + \frac{4}{3}\phi_n - \frac{1}{3}\phi_{n-1}\,, \tag{5.177}$$

ϕ being the auxiliary variable solution. Therefore, the two linear equations to solve read as

$$\frac{1}{2\Delta t}\mathbf{M}(3\mathbf{u}_{n+1} - 4\mathbf{u}_n + \mathbf{u}_{n-1}) + \mathbf{K}\mathbf{u}_{n+1} + \mathbf{C}_1(\mathbf{u}_\star)\mathbf{u}_{n+1} + \mathbf{C}_2(\mathbf{u}_\star)\mathbf{u}_{n+1} + \mathbf{G}\mathbf{p}_\sharp = \mathbf{f}_{n+1} \tag{5.178}$$

and

$$\mathbf{K}_p\boldsymbol{\phi}_{n+1} = \frac{3}{2\Delta t}\mathbf{G}^\top\mathbf{u}_{n+1} \tag{5.179}$$

where we defined the following linearized terms

$$\begin{aligned}
\mathbf{C}_1(\mathbf{u}_\star)\,\mathbf{u} &= \mathop{\mathbb{A}}_{e}\int_\Omega [\boldsymbol{u}_\star\cdot\nabla\boldsymbol{u}]\cdot\boldsymbol{v}\,d\Omega \\
\mathbf{C}_2(\mathbf{u}_\star)\,\mathbf{u} &= \mathop{\mathbb{A}}_{e}\int_\Omega \left[\frac{1}{2}(\nabla\cdot\boldsymbol{u}_\star)\,\boldsymbol{u}\right]\cdot\boldsymbol{v}\,d\Omega \\
\mathbf{K}_p\boldsymbol{\phi} &= \mathop{\mathbb{A}}_{e}\int_\Omega \nabla q\cdot\nabla\phi\,d\Omega
\end{aligned} \tag{5.180}$$

The resulting routine is summarized in Algorithm 3.

Algorithm 3 Pressure-Correction scheme.

function PC($\Delta t, \mathbf{u}_{n-1}, \mathbf{u}_n, \boldsymbol{\phi}_{n-1}, \boldsymbol{\phi}_n, \mathbf{p}_n$)

$\mathbf{u}_\star = 2\mathbf{u}_n - \mathbf{u}_{n-1}$ ▷ extrapolate velocity and
$\mathbf{p}_\sharp = \mathbf{p}_n + \frac{4}{3}\boldsymbol{\phi}_n - \frac{1}{3}\boldsymbol{\phi}_{n-1}$ pressure values

$\mathbf{f}_{n+1} = \frac{1}{2\Delta t}\mathbf{M}(3\mathbf{u}_{n+1} - 4\mathbf{u}_n + \mathbf{u}_{n-1}) + \mathbf{K}\mathbf{u}_{n+1}$ ▷ solve for $\mathbf{u}_{n+1}$ the diffusion
$+\mathbf{C}_1(\mathbf{u}_\star)\mathbf{u}_{n+1} + \mathbf{C}_2(\mathbf{u}_\star)\mathbf{u}_{n+1} + \mathbf{G}\mathbf{p}_\sharp$ (linear) equations

$\mathbf{K}_p\boldsymbol{\phi}_{n+1} = \frac{3}{2\Delta t}\mathbf{G}^\top\mathbf{u}_{n+1}$ ▷ project velocities

$\mathbf{p}_{n+1} = \mathbf{p}_n + \boldsymbol{\phi}_{n+1} - \nu\,\mathbf{G}^\top\,\mathbf{u}_{n+1}$ ▷ correct pressure

return $(\mathbf{u}_{n+1}, \boldsymbol{\phi}_{n+1}, \mathbf{p}_{n+1})$
end function

Iterated Pressure Correction Method

The previously exemplified pressure-correction algorithm performs just a single-step of correction per time step. A second possible approach to the scheme is to perform a cycle of such pressure-correction routines in order to improve convergence, computational efficiency, and numerical accuracy [59].

Disregarding an explicit extrapolation (5.177) of the pressure field, the correction step in pressure is repeated several times, according to a suitable stopping criterion on the pressure correction itself. The previous Algorihtm 3 is therefore rearranged in to Algorithm 4.

For both Pressure-Correction (PC) ed Iterated Pressure Correction (iPC) algorithms a Multistep Backward Difference Formula of order 2 (BDF2) is used for the time discretization (see Section 4.3.2). The solution for the momentum equation is computed employing an iterative solver (see Section 4.4.2) of the GMRES type, whereas for the Poisson equation a CG solver is used. The choice to use a GMRES type solver for the equation of moment is due to the non-symmetric nature of the system of linear equations, while for the Poisson equation, characterized by a symmetric matrix form, a Conjugate Gradients solver is sufficient.

Algorithm 4 Iterated Pressure-Correction scheme.

function IPC($\Delta t, \mathbf{u}_{n-1}, \mathbf{u}_n, \boldsymbol{\phi}_{n-1}, \boldsymbol{\phi}_n, \mathbf{p}_n$)

$\mathbf{u}_\star = 2\mathbf{u}_n - \mathbf{u}_{n-1}$ ▷ extrapolate velocity and
$\mathbf{p}_{n+1}^0 = \mathbf{p}_n$ pressure values

for $k = 1, \dots, k_{\max}$ **do**

▷ pressure correction loop

$\mathbf{f}_{n+1} = \frac{1}{2\Delta t}\mathbf{M}(3\mathbf{u}_{n+1}^k - 4\mathbf{u}_n + \mathbf{u}_{n-1}) + \mathbf{K}\mathbf{u}_{n+1}^k$ ▷ solve for $\mathbf{u}_{n+1}$ the diffusion
$+\mathbf{C}_1(\mathbf{u}_\star)\mathbf{u}_{n+1}^k + \mathbf{C}_2(\mathbf{u}_\star)\mathbf{u}_{n+1}^k + \mathbf{G}\mathbf{p}_{n+1}^{k-1}$ (linear) equations

$\mathbf{K}_p\boldsymbol{\phi}_{n+1}^k = \frac{3}{2\Delta t}\mathbf{G}^\top\mathbf{u}_{n+1}^k$ ▷ project velocities

$\mathbf{p}_{n+1}^k = \mathbf{p}_{n+1}^{k-1} + \boldsymbol{\phi}_{n+1}^k - \nu\,\mathbf{G}^\top\mathbf{u}_{n+1}^k$ ▷ correct pressure

$\text{err} = \left\|\mathbf{p}_{n+1}^{k-1} - \mathbf{p}_{n+1}^k\right\| \;/\; \left\|\mathbf{p}_{n+1}^k\right\|$ ▷ check error on pressure

if err $\leq$ toll. **then**
break
end if
end for

return $(\mathbf{u}_{n+1}, \mathbf{p}_{n+1})$
end function

5.4 Deal.II: Benchmark tests

We now report five numerical investigations aimed to explore the capabilities of the proposed modified Newton-Raphson strategy (here abbreviated as "mNR"). In particular, for coupled advection-diffusion problems mNR scheme consists of a sequence of Algorithms 1 and 2, in orderly succession.

From Section 5.4.1 to Section 5.4.4, we perform key test-cases for comparison with well validated results present in the literature. In Section 5.4.5 we also illustrate some tests able to highlight the performance and efficiency of the proposed mNR schemes, when comparing it with the two alternative approaches presented in Section 5.3.3, based on projection methods: the Projection-Correction method (labeled as "PC", Algorithm 3 and Algorithm 2) and the iterated Projection-Correction method ("iPC", Algorithm 4 and Algorithm 2).

Our implementation utilizes `deal.II` library and employs quadrilateral and hexahedral Lagrange finite elements Q2 and Q1, that yield piecewise polynomials of degree 2 for velocity, and degree 1 for temperature and pressure, respectively. All tests are conducted on an off-the-shelf desktop computer with eight-cores Intel Core i7-6700 running at 3.40 GHz, with 24 GB of RAM, running 64-bit Ubuntu Linux 18.04.4 LTS.

5.4.1 Benchmark 1: Expansion channel

The numerical experiment is intended to validate the capability of the proposed mNR approach to capture nonlinear behaviour of standard Navier-Stokes problems. In particular, the geometry of the test is depicted in Figure 5.6, representing a case of an expansion channel of ratio $\lambda = W/w = 15.4$, where $W = 4$ is the width of both inlet and outlet channels, while $w = 0.26$ is the width of the contraction channel; the length of the contraction channel is $L_c = 2$.

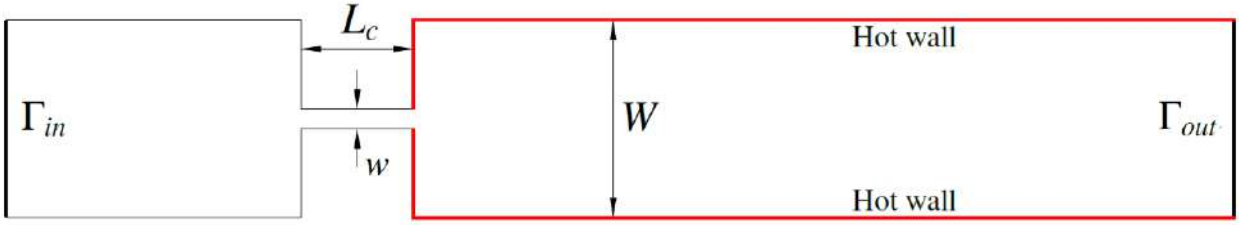

Figure 5.6: Geometry considered for the expansion channel test [211].

Different Reynolds numbers $Re \in [0.01, 71.3]$ are considered as detailed in [211], and defined according to the following expression

$$Re = 2\frac{\rho U w}{\nu}, \tag{5.181}$$

where ρ is the density of the fluid, ν is the kinematic velocity and U is the average velocity in the contraction channel. Note that the contraction channel is designed to be long enough to fully develop a parabolic velocity profile: assuming U_{max} to be the maximum velocity reached in such a channel, the average velocity can be derived as $U = 2U_{max}/3$.

Concerning the boundary conditions, a parabolic profile is prescribed as inlet velocity condition at the inflow boundary Γ_{in}, while stress-free conditions are set at the outflow boundary Γ_{out}, together with no-slip assumptions in the remaining parts of the boundary.

The test is detailed in [211] whose outcomes are here reported in comparison with ours.

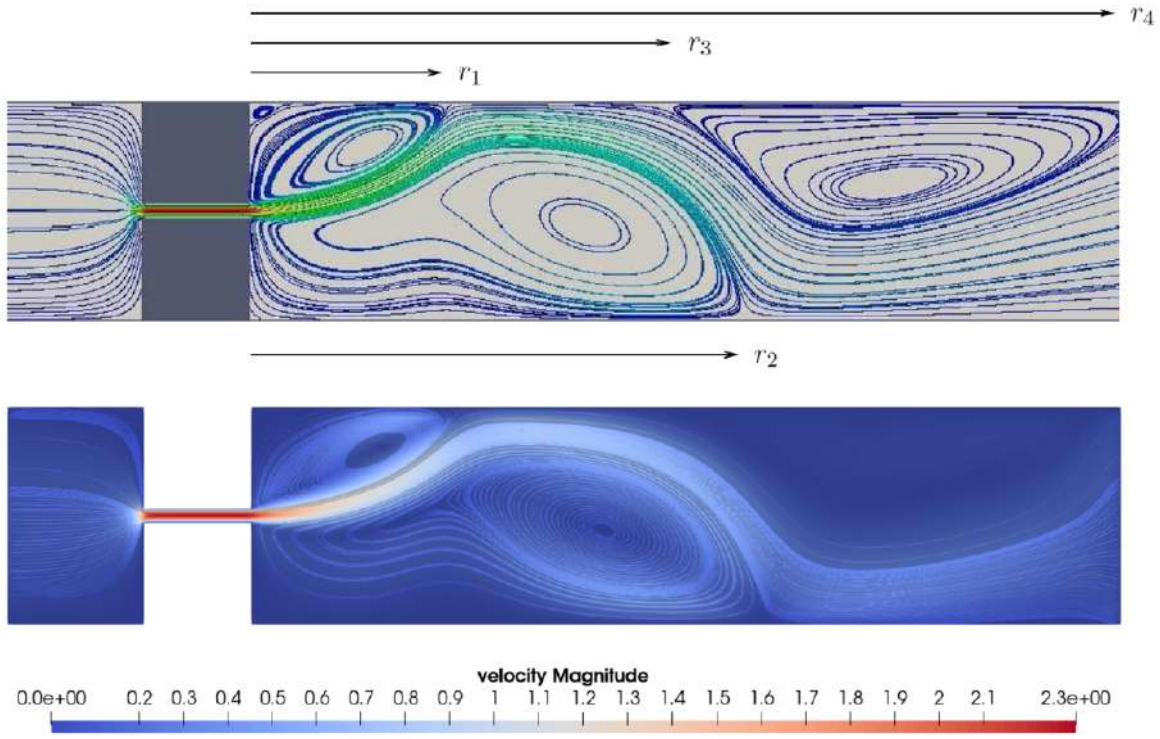

Figure 5.7: Comparison between velocity distributions: reference solution given in [211] (top) and our solution obtained through the proposed mNR scheme (bottom).

Is worth mentioning that the value of the Reynolds number is tuned to highlight instabilities in the velocity field, producing a symmetry breaking of the flow across the contraction channel, and forming vortices with different lengths. The lengths of the vortices, namely r_1, r_2, r_3, and r_4, are measured starting from the beginning of the downstream channel, and normalized with respect to the downstream channel width W (as indicated in Figure 5.7). Their values are plotted versus Re in Figure 5.8. Figure 5.7 shows the velocity distributions for $Re = 71.3$, proving how our simulation is in good agreement with that reported in [211] in terms of velocity streamlines at the time stage when complex vortices appear in the downstream channel. Furthermore, Figure 5.8 proves how the proposed mNR numerical strategy is able to capture the expected behaviour as reproduced in [211].

We then propose to couple the Navier-Stokes problem with a heat advection-diffusion problem, where a constant temperature $T = 0$ is imposed at the inflow Γ_{in}, whereas at the boundary of the downstream channel the temperature is set to $T = 1$, except on Γ_{out}. Adiabatic conditions are assumed on the other boundaries.

The resulting convection behaviour is depicted in Figure 5.9, where heat is transported by the larger-scale fluid motion. Such a phenomenon is also determined by the fluid properties, in

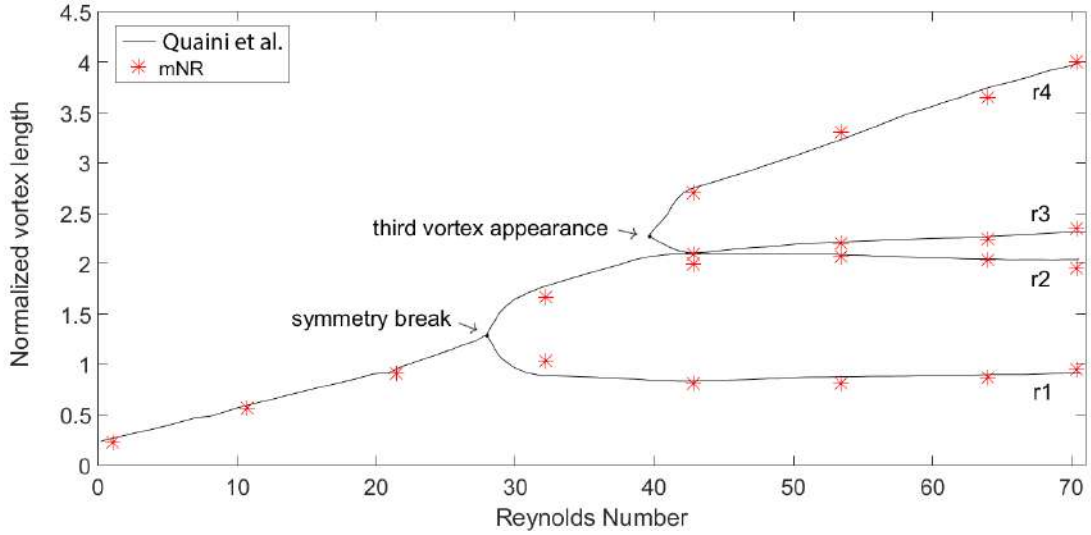

Figure 5.8: Bifurcation diagram: comparison between results in Quaini et al. [211] and ours in terms of normalized vortex length varying with Reynolds number in the test of Figure 5.6.

particular the Prandtl number, defined as the ratio between the kinematic viscosity ν and the thermal diffusivity κ ($Pr = \nu/\kappa$). Two types of fluid, characterized by $Pr = 7.56$ (as water) and by $Pr = 0.71$ (as air), respectively, are simulated and compared each with other. As highlighted by the temperature colour map in Figure 5.9, the air higher thermal diffusivity with respect to that of the water leads to a temperature distribution spreading out in the downstream channel.

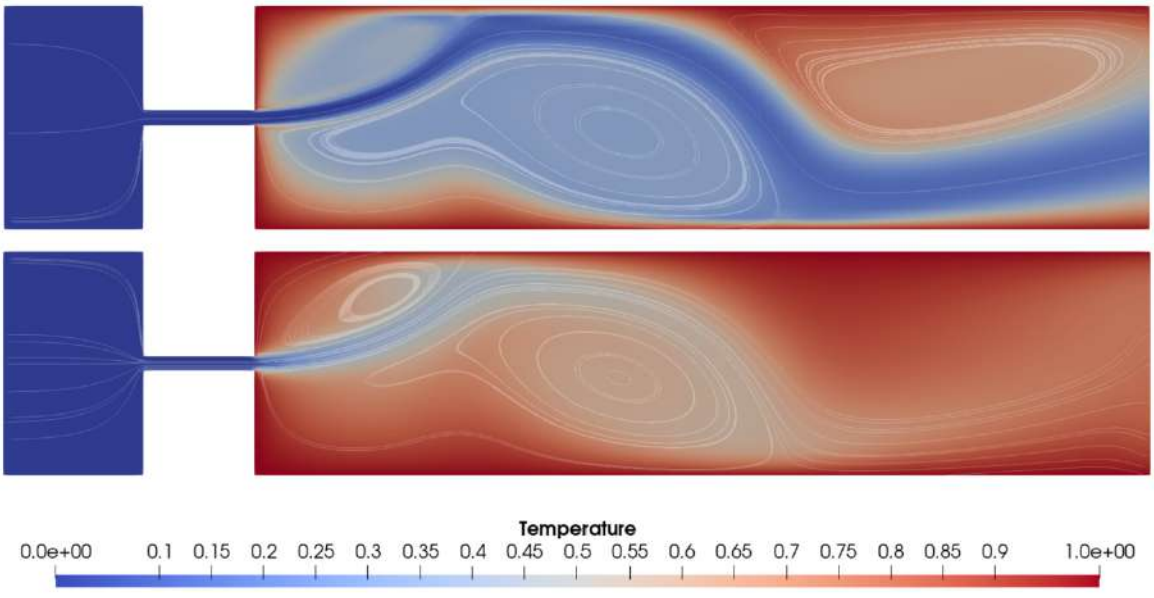

Figure 5.9: Expansion channel test of Figure 5.6: comparison of temperature field and velocity streamlines between two fluids distinguished in terms of Prandtl number, $Pr = 7.56$ (water, on top) and $Pr = 0.71$ (air, on bottom).

5.4.2 Benchmark 2: Transient advection-diffusion

The following test is proposed with the aim to compare our SUPG implementation for the transient heat equation with the one treated in [131]. In particular Bochev et al. [131] observed that the SUPG method cannot be destabilized for small Courant numbers and, through several numerical test varying the time step size, they prove the stability of such method.

Then, to test our algorithm we compare the numerical results of [131] in the pure advection limit, i.e., for $\kappa = 0$, and several different time steps are used to provide a representative range of correctness.

The geometry is just a unit square. The advection filed is imposed equal to $\mathbf{u} = [1.0;\ 0.7002075]$ and the initial temperature field is set as follow

$$T_0(\mathbf{x}) = \begin{cases} 1 & \text{if } |\mathbf{x} - \mathbf{x}_C| \leq 0.2 \\ 0 & \text{otherwise} \end{cases} \tag{5.182}$$

where $\mathbf{x}_C = [0.25;\ 0.25]$. Note that the choice of $\mathbf{u}$ and the initial and boundary data corresponds to an advection of a cylinder of unit height, radius equal to 0.2, and positioned at x_C. On the inflow portion of the domain, that is at $\mathbf{x}_1 = 0$ and $\mathbf{x}_2 = 0$, homogeneous boundary condition are specified.

Results from Figure 5.10 to Figure 5.13 show the contours of the solutions at $t = 0.5$ computed using the four different time steps. A good agreement between the solution provided by our algorithm and the numerical results of [131] is proven by the fact that the pure advection of the temperature filed is correctly reproduced.

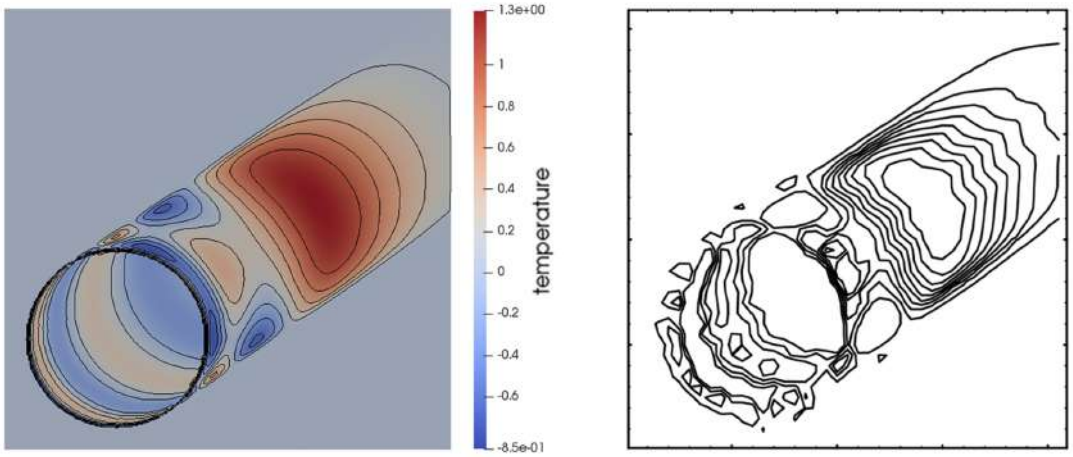

Figure 5.10: Our algorithm results (colored) and Bochev et al. [131] results (black and white) at $t = 0.5$ computed with $\Delta t = 0.1$.

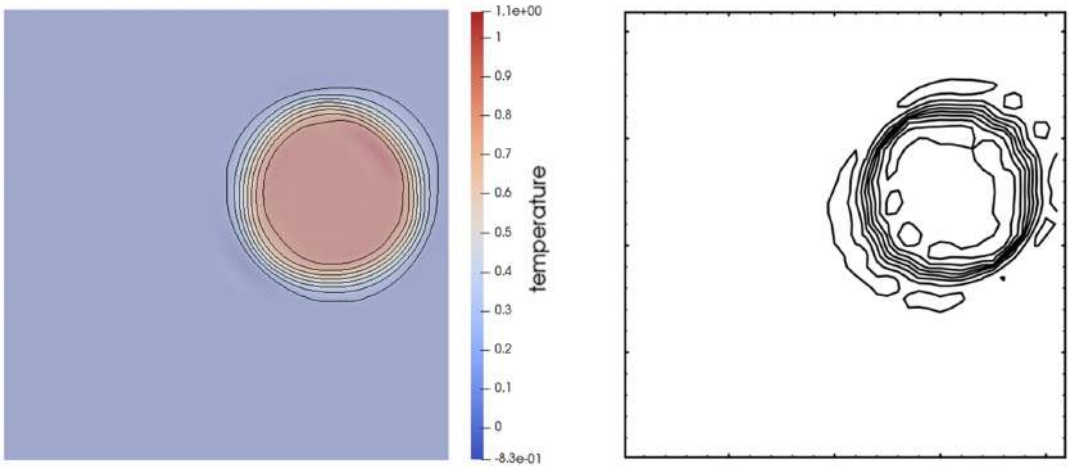

Figure 5.11: Our algorithm results (colored) and Bochev et al. [131] results (black and white) at $t = 0.5$ computed with $\Delta t = 0.01$.

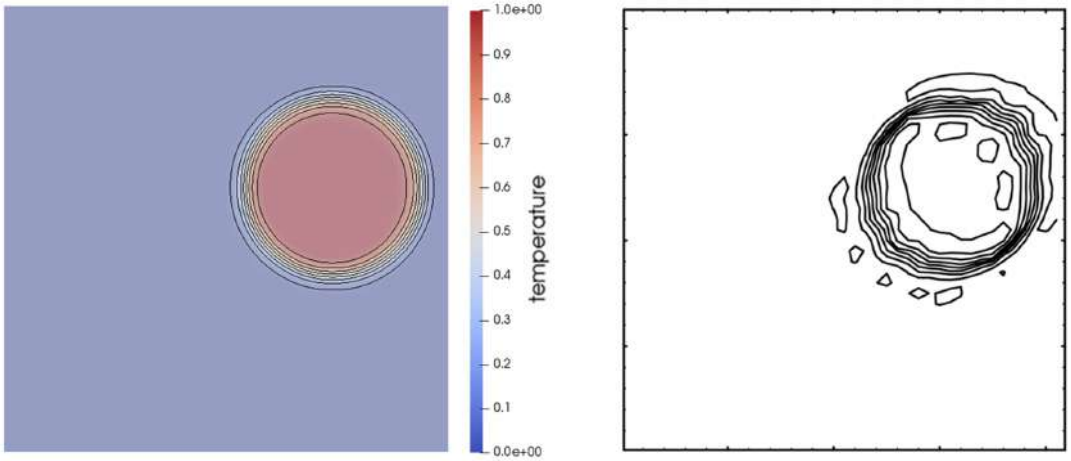

Figure 5.12: Our algorithm results (colored) and Bochev et al. [131] results (black and white) at $t = 0.5$ computed with $\Delta t = 0.001$.

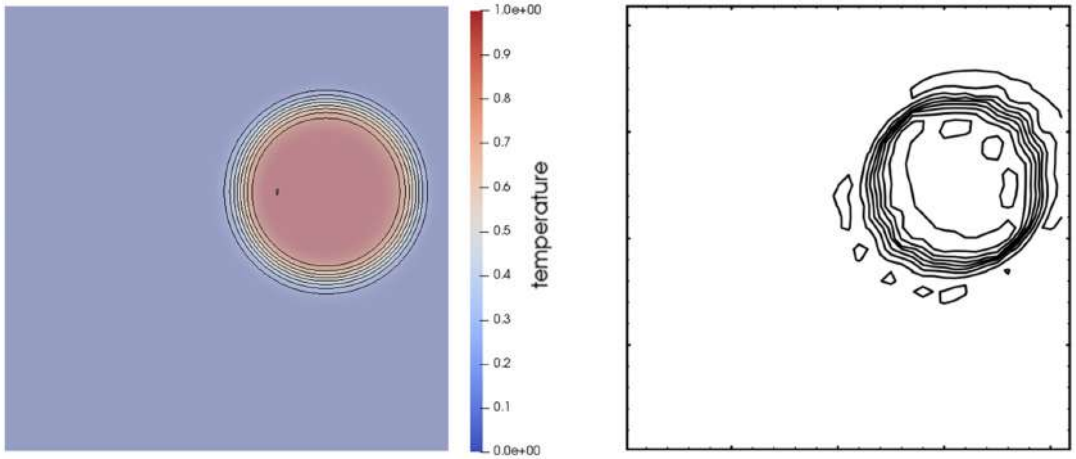

Figure 5.13: Our algorithm results (colored) and Bochev et al. [131] results (black and white) at $t = 0.5$ computed with $\Delta t = 0.0005$.

5.4.3 Benchmark 3: Backward-facing step flow

This test has been proposed in [202], which we refer to for validating the implemented coupled problem of heat advection-diffusion problem embracing Navier-Stokes flow. In Figure 5.14 geometry and boundary conditions for the backward-facing step flow are reported. The channel Expansion Ratio (ER $= H/(H - S)$) is fixed to 2 for all computations, and the length of the channel is equal to 35 S. At the inflow boundary Γ_{in} a parabolic velocity field is prescribed, according to the following law $U = 12y - 24y^2$, where y is the spatial coordinate in the second dimension, and a uniform temperature of $T = 0$ is also set. Stress-free conditions are imposed at the outflow boundary Γ_{out}, together with no-slip assumptions in all the other parts of the boundary. Finally, a fixed value of temperature equal to 1 is prescribed at the bottom of the channel, and adiabatic conditions elsewhere on the boundary.

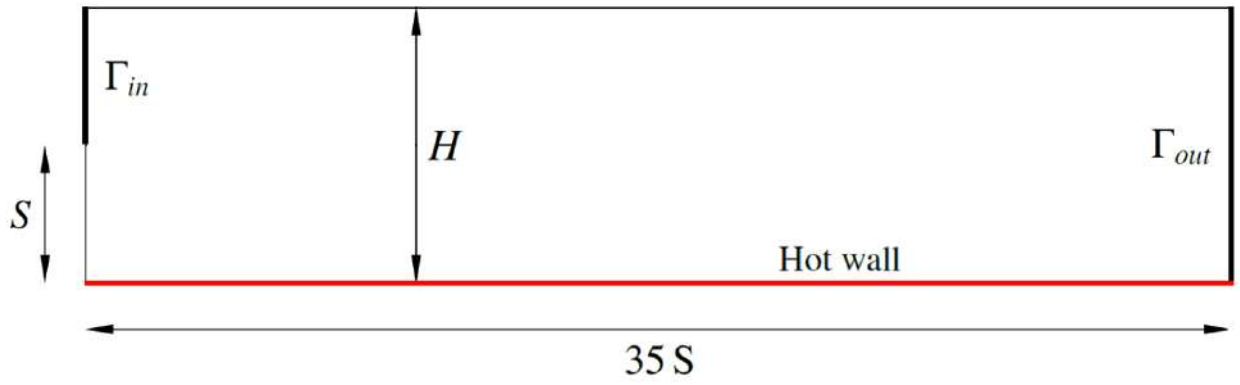

Figure 5.14: Schematic diagram of the backward-facing problem.

Three simulations, for increasing Reynolds numbers $Re \in \{50, 100, 150\}$, are performed fixing the Prandtl number at $Pr = 0.71$, so as to validate the implemented coupled problem through the reference solutions offered in [202]. Each simulation runs until a steady-state configuration is achieved: Figure 5.15, Figure 5.16, and Figure 5.17 report a detailed comparison between our mNR strategy with the reference one with $Re = 50$, 100, and 150, respectively. Again, the proposed method is in good agreement with all temperature distributions carried out by [202].

We additionally compare the resulting Nusselt number Nu, locally computed along the bottom heated wall, and defined as follows:

$$\mathrm{Nu} = -\frac{\partial T}{\partial y}\,. \tag{5.183}$$

Figure 5.18 shows that, except for a limited portion of the boundary close to $x = 0$, the mNR scheme reproduces accurately the entire Nusselt number distribution along the bottom heated wall in comparison with the results carried out in [202], for all considered Re numbers.

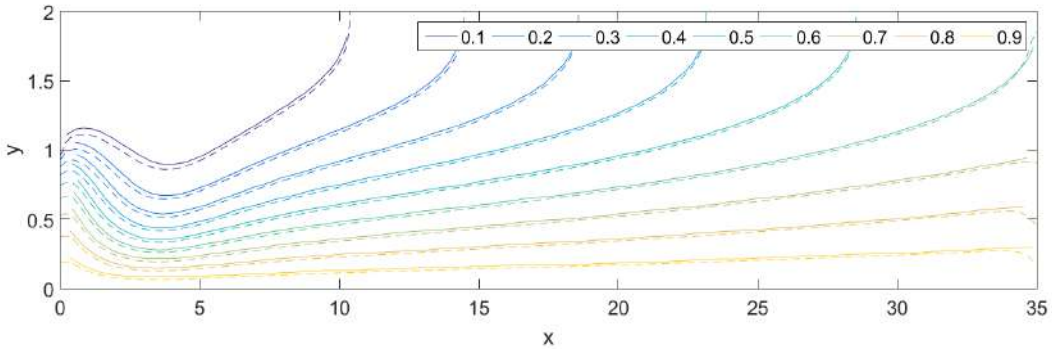

Figure 5.15: Temperature contour plots for the test in Figure 5.14 using $Re = 50$. Comparison between the results reported in Kumar et al. [202] (continuous line) and ours (dashed line) obtained by mNR algorithm.

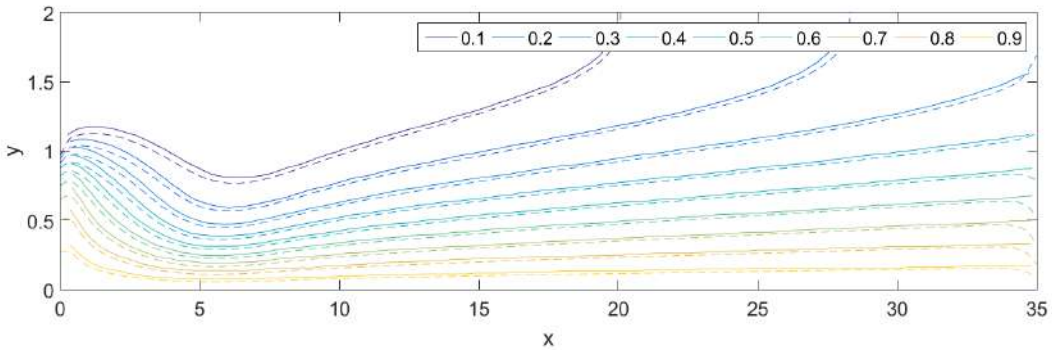

Figure 5.16: Temperature contour plots for the test in Figure 5.14 using $Re = 100$. Comparison between the results reported in Kumar et al. [202] (continuous line) and ours (dashed line) obtained by mNR algorithm.

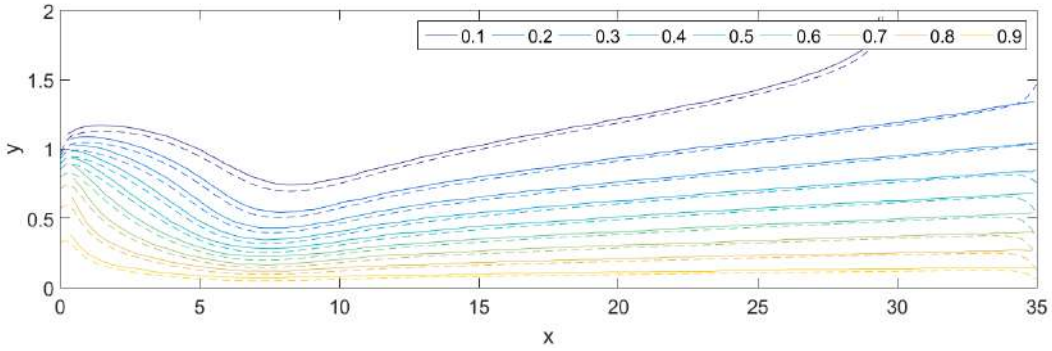

Figure 5.17: Temperature contour plots for the test in Figure 5.14 using $Re = 150$. Comparison between the results reported in Kumar et al. [202] (continuous line) and ours (dashed line) obtained by mNR algorithm.

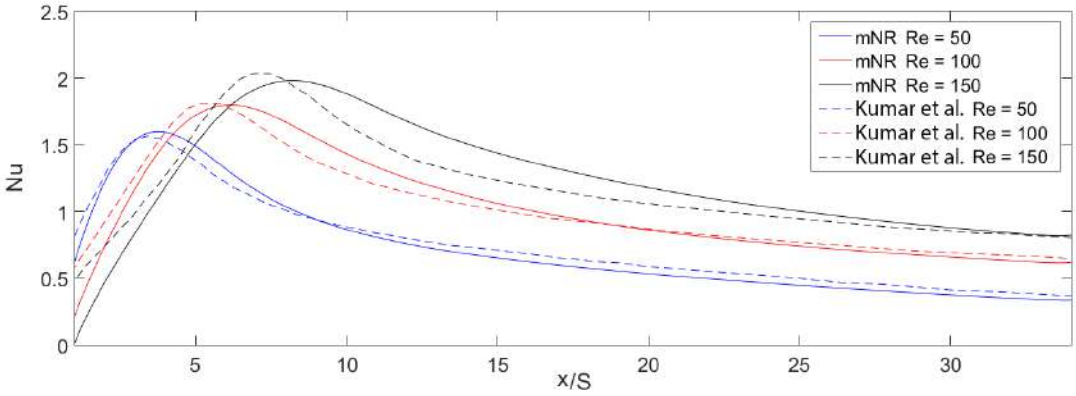

Figure 5.18: Nusselt number Nu varying along the bottom heated wall of Figure 5.14. Comparison between the results reported in Kumar et al. [202] (solid lines) with those obtained by mNR method (dashed lines), corresponding to Reynolds numbers $Re = \{50, 100, 150\}$.

5.4.4 Benchmark 4: Flow over square obstacle through different pipes.

This test is devoted to furthermore validate our fluid-temperature coupling strategy taking as reference the test reported in [212]. Such a problem is depicted schematically in Figure 5.19, where also boundary conditions are expressed. In particular, it consists of a backward facing

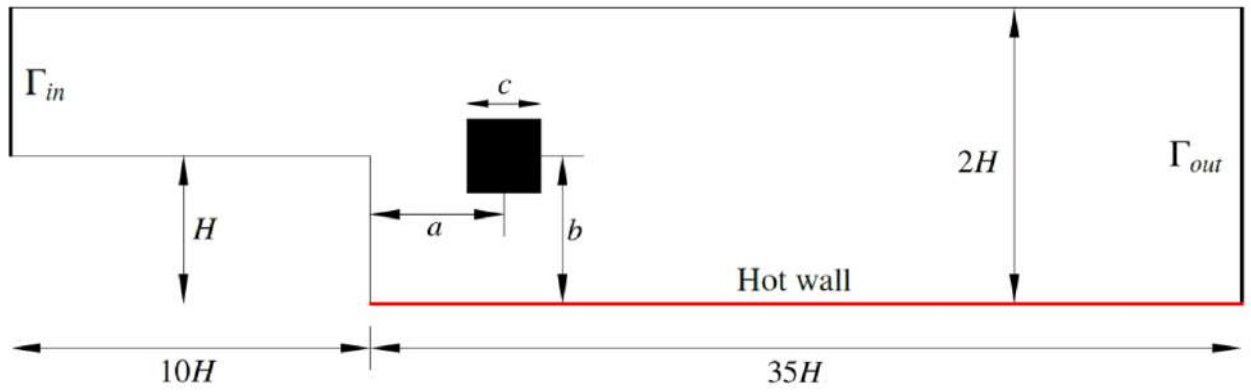

Figure 5.19: Schematic diagram of flow over square obstacle problem.

step of size H, and of a channel of height $2H$; the length of the inlet channel is set to $10H$ in order to fully develop a parabolic velocity profile, while the downstream channel length is $35H$, so as to make the recirculation length (downstream of the step) independent on the computational domain. Additionally, a square obstacle (edge $c = 0.5H$) is inserted at the end of the inflow channel, a distance $a = 0.75H$ and $b = H$ (see again Figure 5.19). Boundary conditions at the inflow Γ_{in} consist of a uniform horizontal velocity with a sinusoidal time dependent part, according to the expression:

$$U = U_0 + 0.9 \sin(2\pi \, \mathrm{St} \, \tau) \, ,$$

where St is the Strouhal number and $U_0 = 1$ the mean velocity; the vertical velocity is set to zero, as well as the temperature. At the bottom surface of the downstream channel a constant temperature equal to $T = 1$ is imposed, while the other boundaries are assumed to be adiabatic. Regarding the velocity field, stress-free conditions are set at the outflow boundary Γ_{out} and no-slip condition on the other walls. We first compare our results with those presented in [212] in terms of different Reynolds number $Re \in \{10, 100, 200\}$, setting Prandtl and Strouhal number to $Pr = 0.71$, and $\mathrm{St} = 2$, respectively. A time step size $dt = \overline{\tau}/100$, where $\overline{\tau}$ is the period of the pulsating flow, is adopted for time integration.

The proposed mNR method is validated against simulations reported in [212], by monitoring the spatially averaged Nusselt number ($\overline{Nu}$), divided by the steady state value (Nu_s), varying in time and defined as

$$\overline{Nu} = \frac{1}{L} \int_0^L Nu \, dx = \frac{1}{L} \int_0^L -\frac{\partial T}{\partial y} dx \, ,$$

where $L = 35H$ is the length of the bottom wall. Figure 5.20, Figure 5.21, and Figure 5.22, associated with $Re \in \{10, 100, 200\}$, respectively, prove that our simulations are in good

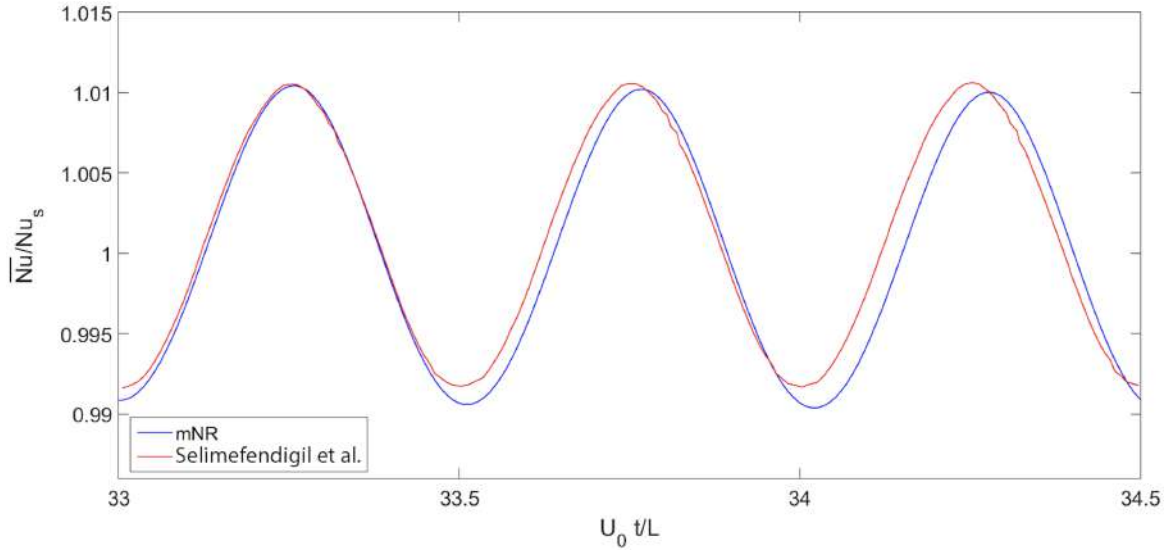

Figure 5.20: mNR outcomes (blue line) compared with simulations in Selimefendigil et al. [212] (red line) in terms of spatially averaged Nusselt number varying in time, when $Re = 10$ is considered and steady-state periodic oscillations are reached for the test depicted in Figure 5.19.

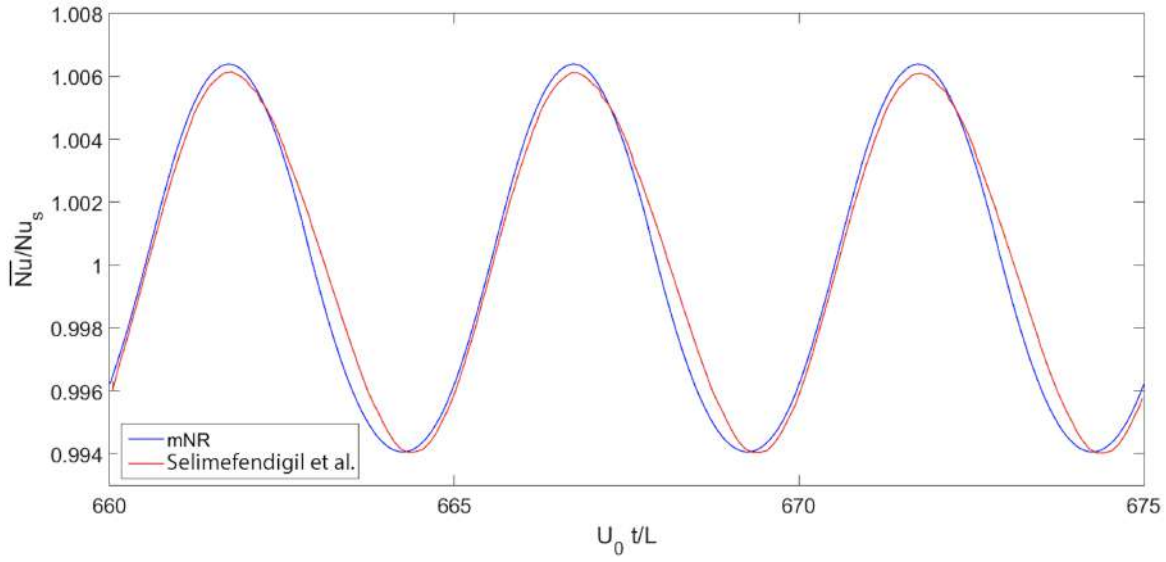

Figure 5.21: mNR outcomes (blue line) compared with simulations in Selimefendigil et al. [212] (red line) in terms of spatially averaged Nusselt number varying in time, when $Re = 100$ is considered and steady-state periodic oscillations are reached for the test depicted in Figure 5.19.

agreement with reference ones. Finally, we report in Figure 5.23 the distributions of both velocity magnitude and temperature obtained by our mNR scheme when steady-state conditions are reached. Such distributions show how the above discussed validation is done in a context where the coupled problem exhibits a highly nonlinear behaviour as vortex formations clearly depict.

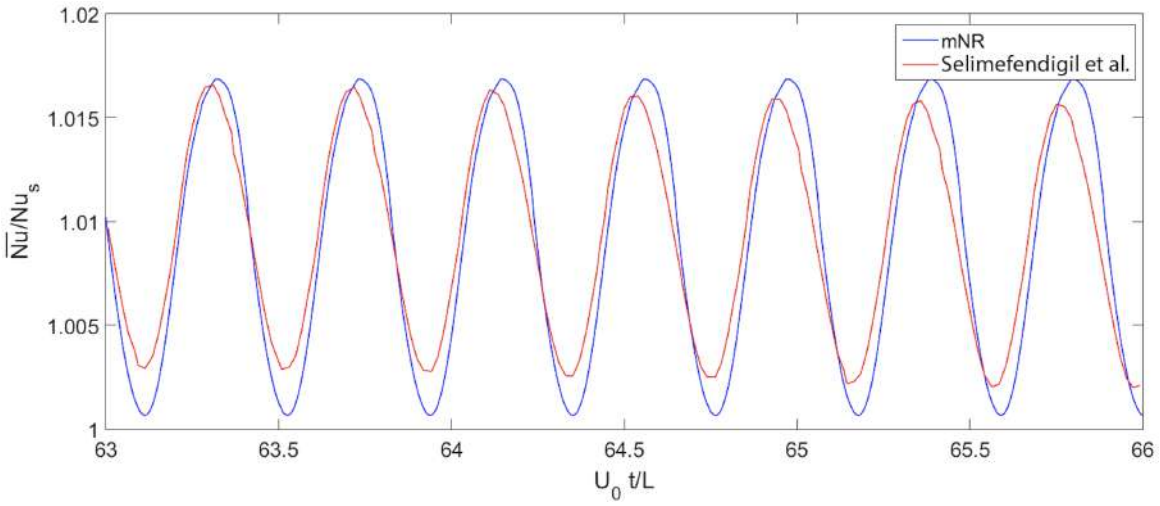

Figure 5.22: mNR outcomes (blue line) compared with simulations in Selimefendigil et al. [212] (red line) in terms of spatially averaged Nusselt number varying in time, when $Re = 200$ is considered and steady-state periodic oscillations are reached for the test depicted in Figure 5.19.

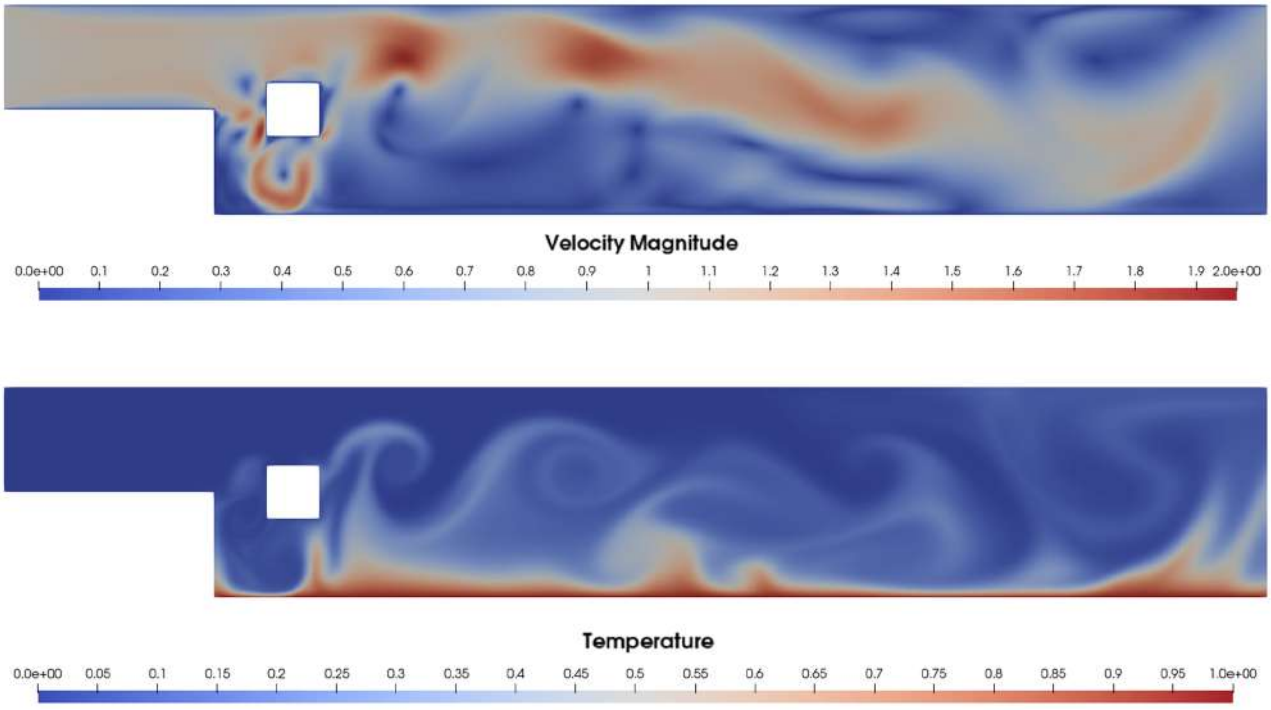

Figure 5.23: Velocity magnitude (top) and temperature distributions (bottom), reproduced by our mNR scheme when steady-state conditions are reached for the test depicted in Figure 5.19 using $Re = 100$ at $U_0\, t/L = 668$.

5.4.5 Benchmark 5: Flow over square obstacle in a pipe

The test-case here investigated consists of a flow around a square obstacle according to the geometry depicted in Figure 5.24. It is a benchmark reported in the "step-35" `deal.II` tutorial[1]. Such a test is depicted in Figure 5.24, where the channel height is set to $H = 4.1$, thus making the entire test geometry slightly non-symmetric, the channel length is set to $L = 25$, and distances $a = b = 1.5$ and $c = 1$. Additionally, no-slip boundary conditions are imposed on

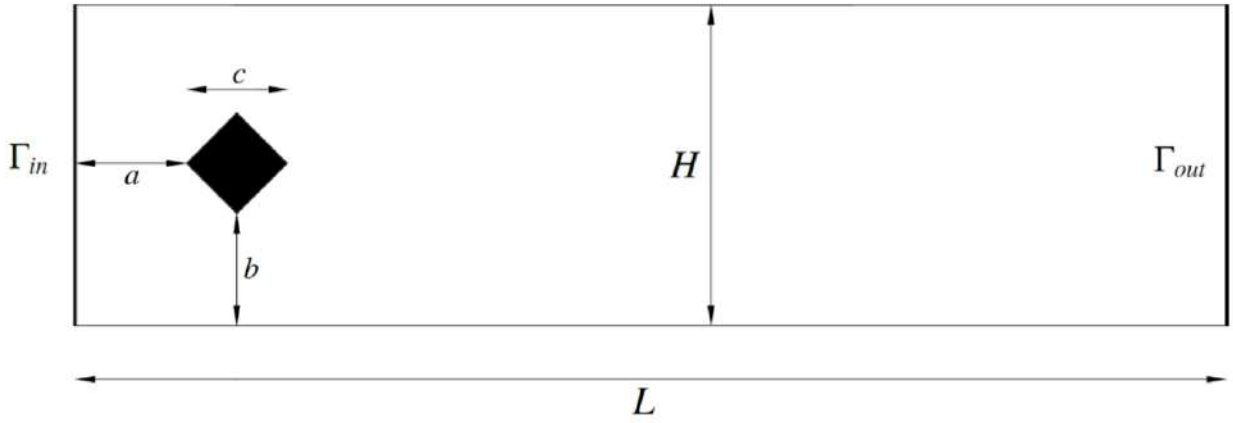

Figure 5.24: Geometry of the `deal.II` test-case.

both the top and bottom walls and on the obstacle. The left side of the channel is characterized by inflow conditions on boundary Γ_{in} corresponding to a Poiseuille flow problem:

$$U = 4U_m\, y(H - y)/H^2\,, \tag{5.184}$$

with $U_m = 1.5$; finally, the boundary conditions at the outflow side Γ_{out} prescribe both vertical component of the velocity and pressure set to be zero.

Several tests are conducted in order to compare the proposed nonlinear scheme with other approaches, fully validated in various papers present in the literature. The numerical approaches we compare with are the pressure-correction method illustrated in the "step-35" `deal.II` tutorial (see again footnote 1), and the iterated pressure-correction scheme developed in [59]; for the sake of simplicity, such methods are named from now on respectively with "PC" and "iPC".

As detailed in Section 5.3.3 and 5.3.3, both PC and iPC methods, in contrast with mNR, not only work with some numerical approximations, but they admit also different models of approximation of the Navier-Stokes field equations. We therefore opt to use the continuity condition as general criterion of measuring the accuracy of the numerical results. In particular, we make comparisons between the three schemes by calculating the averaged divergence of the velocities, i.e.:

$$\text{err} := \left\| \nabla \cdot \mathbf{u} \right\|_{L^2}\,, \tag{5.185}$$

which is an error expected to be equal to zero, satisfying in an average sense the continuity condition in each point of the problem domain.

[1] See `https://www.dealii.org/current/doxygen/deal.II/step_35.html`

For all the computations here reported, and first limited to a pure Navier-Stokes problem, we consider $Re = 100$ as the Reynolds number able to emphasize significant differences between the three implemented approaches. The total time is $t = 20$ and the velocity field reached at this time is depicted in Figure 5.25.

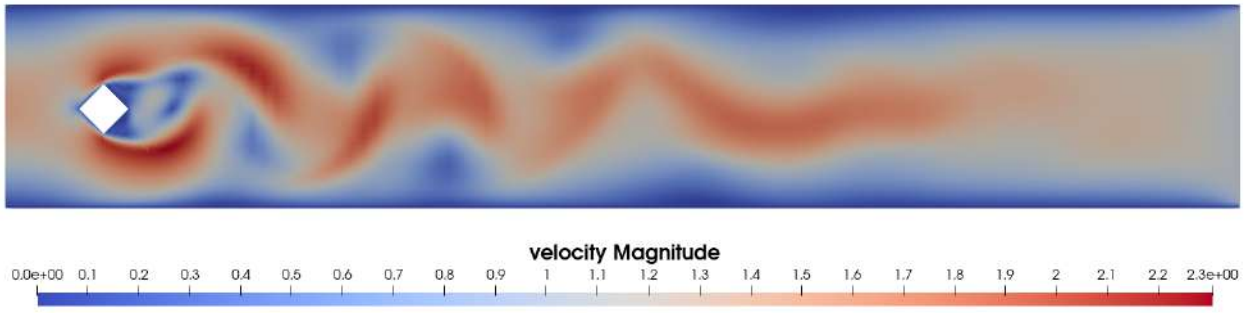

Figure 5.25: Velocity magnitude distribution at the final time $t = 20$ for a mNR simulation conducted with $Re = 100$.

The mNR scheme proves to be noticeably more accurate than both the PC and iPC ones. By considering a common time-step size $dt = 5 \cdot 10^{-3}$, the error in the flow continuity, measured following (5.185), turns out to be of about 10^{-14} for all time steps performed by the mNR method, while is around 10^{-4} and 10^{-5} for PC's and iPC's, respectively (see Figure 5.26). Such a high level of accuracy of the mNR method is very expensive in terms of computational CPU time, which results more than 650% of those employed by PC, with iPC doubling the simulation time with respect to PC.

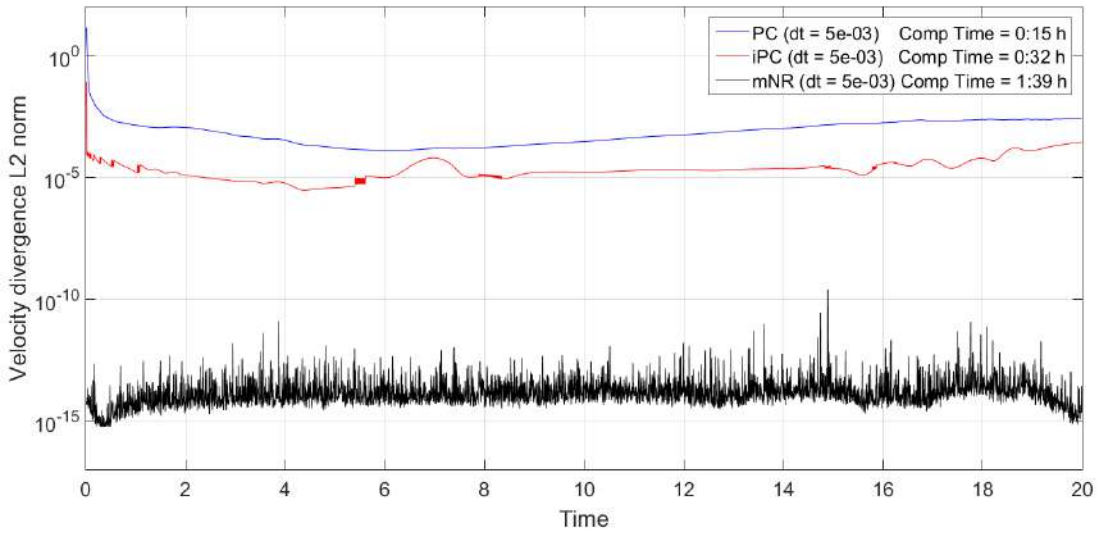

Figure 5.26: Velocity-divergence norm error (5.185) varying in time, carried out by the three implemented schemes, PC (blue line), iPC (red line), and mNR (black line), using the same time-step size for solving the problem in Figure 5.24.

However, in several applications both PC and iPC methods, although working with lin-

earized equations, are considered to be suitable enough to simulate even complex Navier-Stokes problems, and to reproduce real experiments in a sufficiently accurate way (see [59] for a deepen discussion on this point). We can therefore relax the accuracy obtained with the mNR scheme by taking a larger time step, greater than those considered as acceptable in both PC and iPC methods.

In this perspective, a significant outcome is reported in Figure 5.27, where the before described analysis is repeated with a time-step size for mNR progressively increased as $dt \in 10^{-3} \cdot \{5,\ 10,\ 50,\ 100\}$. Figure 5.27 shows how mNR achieves slightly more accurate results than both implemented projection methods, drastically reducing the computational costs, yielding 250% and 533% speedup with respect to PC and iPC numerical schemes, respectively. In order

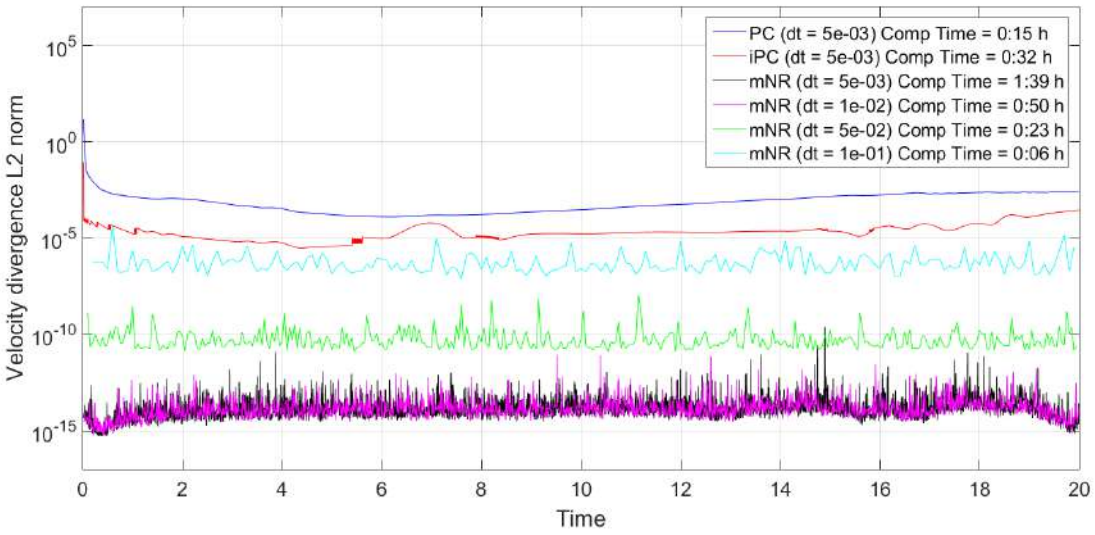

Figure 5.27: Velocity-divergence norm error (5.185) varying in time for the problem in Figure 5.24: PC (blue line) and iPC (red line) schemes compared with mNR scheme employing different time-step size $dt = \{5,\ 10,\ 50,\ 100\} \cdot 10^{-3}$ (black up to cyan lines, respectively).

to emphasize the performance obtained by the proposed mNR method, we also compare the three numerical approaches for the test described in Section 5.4.1, extending it to the fluid-temperature coupled problem. Remember that for such a test, mNR proved to be very accurate if compared with the reference solution reported in [211]. In this view, Figure 5.29, showing how the three schemes rise to a similar level of resulting temperature at the outlet boundary Γ_{out} (see Figure 5.6), also demonstrates both PC and iPC to be accurate enough compared to mNR, even working with a linearization of the Navier-Stokes equations. Such an implicit validation done also for both PC and iPC schemes is furthermore evident in Figure 5.28, showing how all the implemented methods provide practically undistinguished temperature distributions for the test in Figure 5.6. However, the mNR scheme turns out to be more efficient than other two schemes costing about 80% less of CPU times, in reasons of time-step size equal to $dt = 0.2$ for mNR and $dt = 0.01$ for both PC and iPC. Finally, mNR shows higher accuracy in terms of velocity-divergence norm error than both PC and iPC methods. Figure 5.30 shows how PC

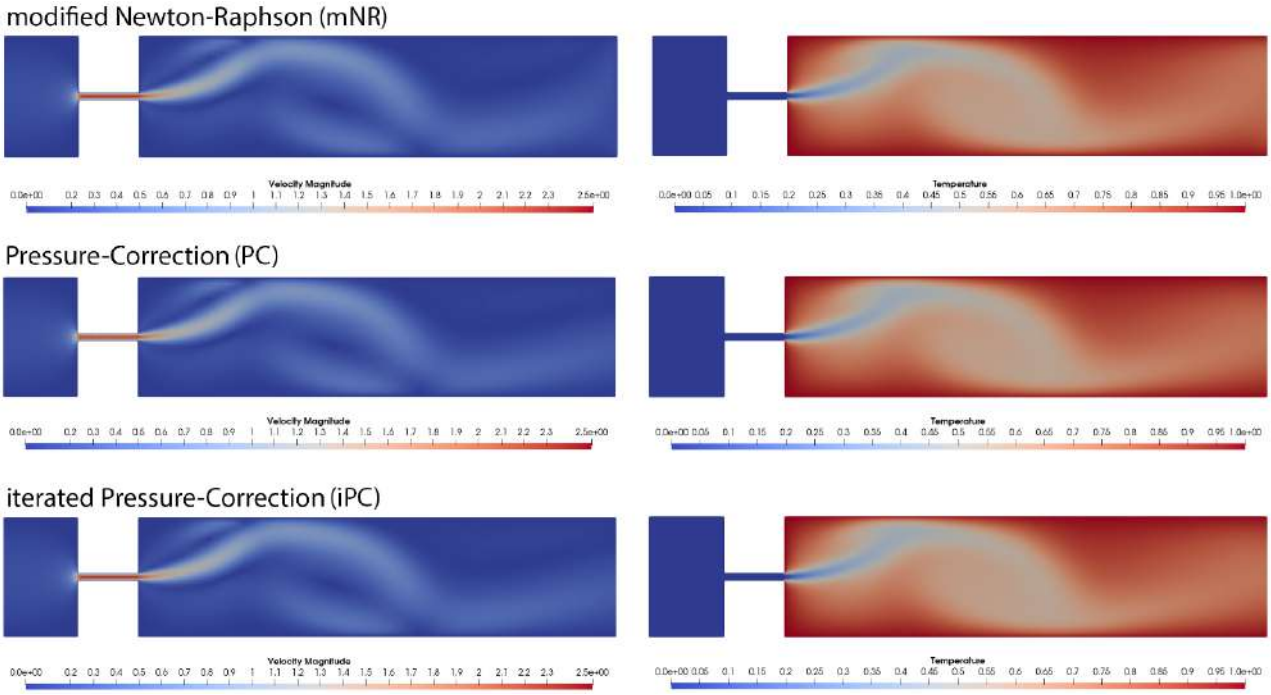

Figure 5.28: Expansion channel test of Figure 5.6: mNR scheme compared with PC's and iPC's in terms of both velocity magnitude (left column) and temperature (right column) distributions.

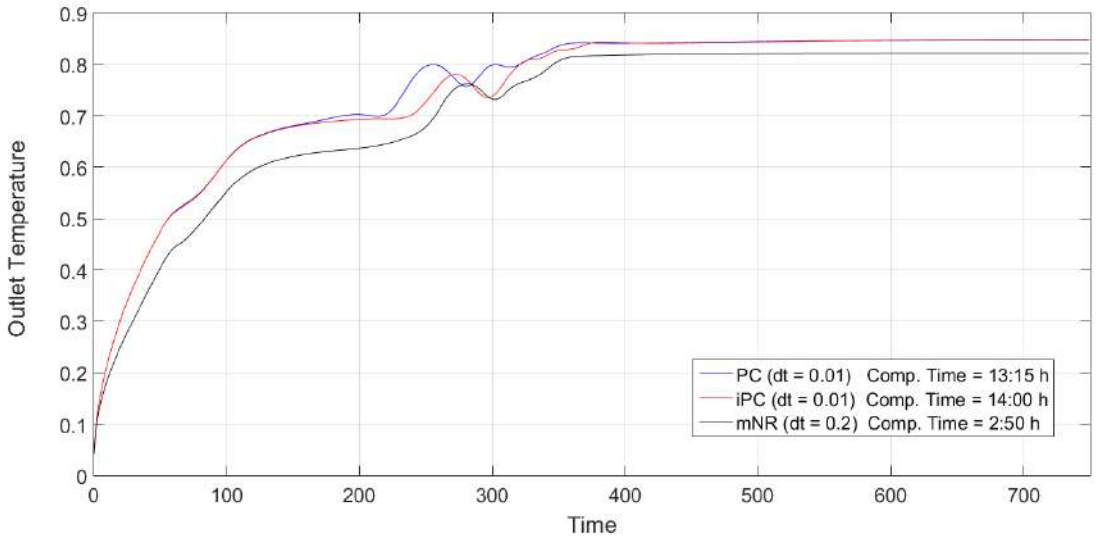

Figure 5.29: The PC (blue line), iPC (red line) and mNR (black line) schemes compared each with other in terms of resulting temperature at the outlet boundary Γ_{out} of the test depicted in Figure 5.6, simulated for $Pr = 0.71$ (air) in the fluid-temperature coupled problem.

leads to an unacceptable error with respect to iPC whose error is about 10^{-3}; instead, mNR drops down to 10^{-14}, resulting the most accurate and efficient numerical scheme.

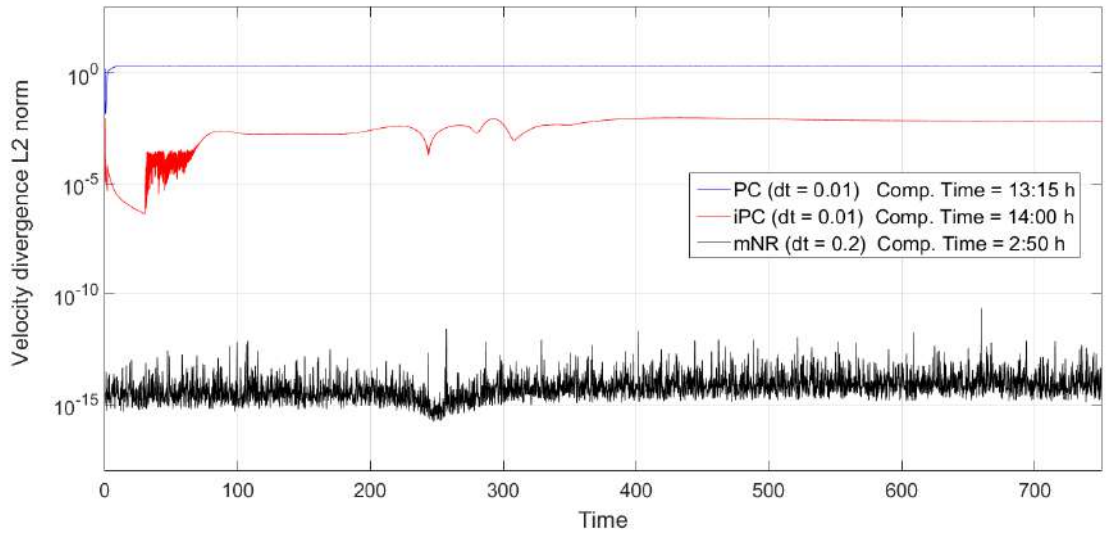

Figure 5.30: The PC (blue line), iPC (red line) and mNR (black line) schemes compared each with other in terms of velocity-divergence norm error (5.185) varying in time for the test depicted in Figure 5.6, simulated for $Pr = 0.71$ (air) in the fluid-temperature coupled problem.

5.5 Deal.II: Particles dynamic equations

The dynamic of the powder particles is simulated by solving the equations for mass, momentum and energy using a Lagrangian approach [61], where every particles are treated as discrete entities moving in the Eulerian flow field. This method is particularly advantageous since the location of the discrete-phase can be calculated everywhere in the domain, regardless of the size of the cell in which the particle resides. Then, rearranging equations (4.129) - (4.131) using a Backward Euler method (see Section 4.3.1) to integrate with respect to time, the particle thermal evolution is described by the following equation,

$$T_p^{n+1} = T_p^n + \Delta t \left(\frac{I_T \eta_p A_{p,p} - h A_p \left(T - T_p\right)}{m_p \left(c_p + \frac{L_f}{T_{liq} - T_{sol}}\right)} \right) \tag{5.186}$$

whereas, the particle velocity can be found solving the momentum equation that reads as,

$$\boldsymbol{u}_p^{n+1} = \boldsymbol{u}_p^n + \frac{\Delta t}{m_p} \left(0.5 A_p C_D \rho_p \left(\boldsymbol{u} - \boldsymbol{u}_p\right) |\boldsymbol{u} - \boldsymbol{u}_p| + m_p \boldsymbol{g}\right) \tag{5.187}$$

and finally the particle position is updated according to

$$\boldsymbol{x}_p^{n+1} = \boldsymbol{x}_p^n + \Delta t \, \boldsymbol{u}_p^{n+1} \tag{5.188}$$

The drag coefficient C_D in (5.187) is calculated in function of the particle Reynolds number Re_p as follows [213]

$$C_D = \frac{24}{Re_p} \qquad \text{for } \mathrm{Re}_p \leq 1 \tag{5.189}$$

$$C_D = \frac{24}{Re_p} \left(1 + 0.15\, Re_p^{0.687}\right) \qquad \text{for } 1 \leq \mathrm{Re}_p \leq 1000 \tag{5.190}$$

$$C_D = 0.44 \qquad \text{for } \mathrm{Re}_p \geq 1000 \tag{5.191}$$

the particle Reynolds number being defined as:

$$Re_p = \frac{\rho \, d_p |\boldsymbol{u} - \boldsymbol{u}_p|}{\nu} \tag{5.192}$$

where ν and ρ are the kinematic viscosity and the density of the fluid, respectively, and d_p is the particle diameter. All other parameters in the above equations have been previously defined in Section 4.5.

5.5.1 Particle-wall interaction

The equations shown above allow the single particle to move freely within the fluid domain, exchanging information about the motion and energetic properties that characterize the fluid itself. However, other conditions must be added to these equations which take into account the interactions between the single particle and the boundary conditions of the domain. Apart the

trivial inlet and outlet conditions of the particles, the particle-wall interaction problem, that is the rebound conditions, has to be addressed when analyzing fluid-particle flows contained within walls. In particular, two models are used to deal with particle-wall interaction, the *hard sphere model* and the *soft sphere model* [102].
The hard sphere model is based on the impulsive force defined by the integration of the equations of motion, keeping implicitly the instantaneous deformation of the particle in the formulation. On the other hand, the soft sphere model works explicitly with the instantaneous motion during the whole collision process. In fact, in the soft sphere model, the whole process of collision is solved by numerical integration of the equations of motion, and consequently the computation time required to take in account this physical interaction is much longer than in the hard sphere model. Considering the small size and the rigid material of the powder particles in the LMD process, the soft sphere model would be excessively accurate; therefore, for this type of problem, the hard sphere model is used.

In the hard sphere model the difference in momentum between two instants of time is equal to the impulsive force acting on the particle during that time period, and it is defined as

$$m_p \left(\boldsymbol{u}_p^{n+1} - \boldsymbol{u}_p^n \right) = \boldsymbol{J}_p^{n+1} \tag{5.193}$$

where $\boldsymbol{u}_p^{n+1}$ and $\boldsymbol{u}_p^n$ are the particle velocities at time t_{n+1} and time t_n, respectively, and $\boldsymbol{J}_p$ is the impulsive force acting on the particle during the time step $t_{n+1} - t_n$. $\boldsymbol{J}_p$ is defined as the time integral of the force acting on the body, corresponding to the second term of the right hand side of (5.187) multiply for the particle mass.
The difference in angular momentum is expressed

$$I_p \left(\boldsymbol{\omega}_p^{n+1} - \boldsymbol{\omega}_p^n \right) = -r_p \times \boldsymbol{J}_p^{n+1} \tag{5.194}$$

where r_p is the radius of the particle, $\boldsymbol{\omega}_p^{n+1}$ and $\boldsymbol{\omega}_p^n$ are angular velocities at t_{n+1} and t_n, respectively, and I_p is the moment of inertia of the particle. In general, (5.193) and (5.194) are not sufficient to evaluate the rebound velocities, and some auxiliary equations, employing both the restitution and friction coefficients, are needed to determine the relationship between the pre- and post-collisional velocities.
Assuming the particle as a sphere, the velocity vector after rebound can be expressed by the following relation

$$e = \frac{\boldsymbol{u}_{p,n}^{n+1}}{\boldsymbol{u}_{p,n}^n} \tag{5.195}$$

where $\boldsymbol{u}_{p,n}$ is the particle velocity component normal to the wall. Notice that the normal impulse ratio $\boldsymbol{J}_{p,n}^{n+1}/\boldsymbol{J}_{p,n}^n$ is equal to the velocity ratio $-\boldsymbol{u}_{p,n}^{n+1}/\boldsymbol{u}_{p,n}^n$. Moreover, defining the longitudinal component as $\boldsymbol{u}_{p,t}$ (see Figure 5.31) the particle velocity vector components can be calculated as follow

$$\boldsymbol{u}_{p,n} = \left(\frac{\boldsymbol{u}_p \cdot \boldsymbol{n}}{\boldsymbol{n} \cdot \boldsymbol{n}} \right) \boldsymbol{n} \tag{5.196}$$

$$\boldsymbol{u}_{p,t} = \boldsymbol{u}_p - \boldsymbol{u}_{p,n} \tag{5.197}$$

where $\boldsymbol{n}$ is the surface normal to the wall at the collision point. However, both rebound

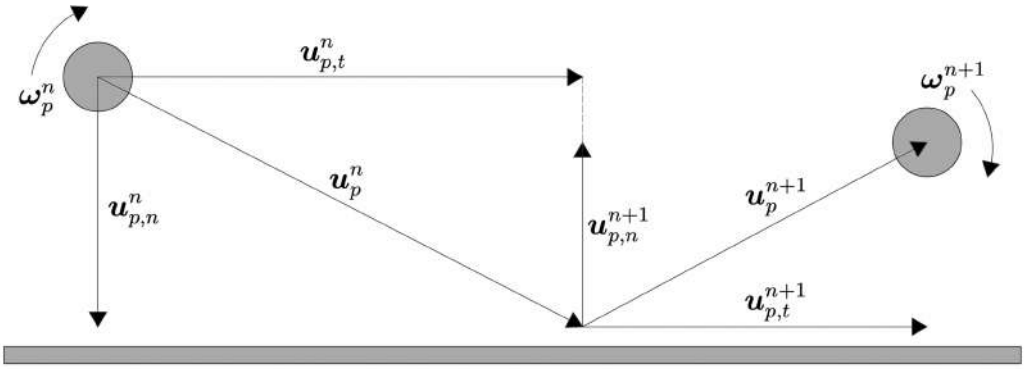

Figure 5.31: Particle colliding with a wall.

velocity components $\boldsymbol{u}_{p,n}^{n+1}$ and $\boldsymbol{u}_{p,t}^{n+1}$ can be obtained if the restitution coefficient e and the kinetic (sliding) friction coefficient f are known. The following assumptions are made: (i) the particle deformation is neglected, and then during the rebound the distance between che collision point and the particle center of mass is constant end equal to the particle radius; (ii) the Coulomb friction law is applied to the particle sliding along a wall; and (iii) once a particle stops sliding, there is no further sliding.
Therefore, the only known variables are the initial velocities, $\boldsymbol{u}_p^n$ and $\boldsymbol{\omega}_p^n$, and introducing the variable $\boldsymbol{u}_p^{n+1/2}$ that represent the particle velocity at the collision point, we can write the impulse equations as

$$\begin{aligned} m_p\left(\boldsymbol{u}_p^{n+1/2}-\boldsymbol{u}_p^n\right) &= \boldsymbol{J}_{p,n}^{n+1/2} \\ m_p\left(\boldsymbol{u}_p^{n+1}-\boldsymbol{u}_p^{n+1/2}\right) &= \boldsymbol{J}_{p,n}^{n+1} \\ I_p\left(\boldsymbol{\omega}_p^{n+1/2}-\boldsymbol{\omega}_p^n\right) &= -r\times\boldsymbol{J}_{p,n}^{n+1/2} \\ I_p\left(\boldsymbol{\omega}_p^{n+1}-\boldsymbol{\omega}_p^{n+1/2}\right) &= -r\times\boldsymbol{J}_{p,n}^{n+1} \end{aligned} \tag{5.198}$$

the boundary condition is

$$\boldsymbol{u}_{p,n}^{n+1/2}=0 \tag{5.199}$$

Given the value of the restitution coefficient e, the normal component of the impulsive force can be found as

$$\boldsymbol{J}_{p,n}^{n+1}=e\boldsymbol{J}_{p,n}^{n+1/2} \tag{5.200}$$

and, finally, from Coulomb's friction law

$$\boldsymbol{J}_{p,t}^{n+1/2}=(1-f)\boldsymbol{J}_{p,n}^{n+1/2} \tag{5.201}$$

$$\boldsymbol{J}_{p,t}^{n+1}=(1-f)\boldsymbol{J}_{p,n}^{n+1} \tag{5.202}$$

substituting the values of the impulsive forces in the (5.198) the post-collisional velocities $\boldsymbol{u}_p^{n+1}$ and $\boldsymbol{\omega}_p^{n+1}$ can be finally calculated.

5.5.2 Particles algorithm

In the present section the Lagrangian particle tracking algorithm is presented and discussed. Such an algorithm has to faithfully represent the powder flow that exit from the nozzle, transported by the fluid, and subsequently invested by the laser beam. The absorption of the energy generated by the laser source results in an increase of the particle temperature leading to phase changing phenomena of melting in cases of high laser power employed.
In order to perform a correct implementation of the solution algorithm it is necessary to know how to handle several computational aspects: (i) appropriate data structures; (ii) particles generation; (iii) particles advection by integrating their trajectory and properties; (iv) treatment of particles that cross cells and boundaries assigned to different processors in parallel computations; (v) definition of properties associated with each particle; and (vi) methods to interpolate or project particle properties onto the mesh, and viceversa. The set of classes and functions already present in the deal.ii library have been crucial to handle computational aspects listed above in a simplified and effective way. These data structures will be shown in detail below and in particular their use within the numerical code will be highlighted.

Data structure

First of all, it is essential to create a data structure able to manage in a clear and a simple way the storage of all the properties of the particles that will be inserted in the computational domain. In particular, it must easily and directly access the information of the single particle, independently from the number of existing particles. Therefore, for the development of an efficient algorithm, as part of a mesh-based CFD code, the following basic requirements have to be satisfied:

- Each particle must be associated with a cell E within which it is located at that time (the cell E represents a part Ω_e of the global model domain Ω).
- The number of particles $N_{par,E}$ contained in the single cell E can change from cell to cell and in the various time steps due to particles being advected from one cell to another during the simulation.
- Each processor P manages a number of particles $N_{par,P}$ that may change from time step to time step due to the number of new particles generated.
- Each particle stores its current location, a global index p_{id}, and a fixed number of scalar properties $N_{properties}$.
- At each particle location the field-based variables, such as fluid velocity, temperature and laser energy, could be evaluated.

These requirements are successfully satisfied using the `ParticleHandler` class present in the deal.II library. This class manages the storage and handling of particles and it provides the

data structures necessary to store particles efficiently, accessor functions to iterate over particles and find particles in the domain. Its initialization proceeds as follows:

```
Particles::ParticleHandler<dim,dim> particle_handler;
particle_handler.initialize(const Triangulation<dim,spacedim> & tria,
                            const Mapping<dim,spacedim> & mapping,
                            const unsigned int n_properties);
```

The `initialize` function does not clear the internal data structures, it just sets the triangulation, the mapping to be used, and the number of properties that each particle owns.

Generation

Once the `particle_handler` object of the `ParticleHandler` class has been initialized, the next step is to generate and insert the particles into it. In this specific case it was chosen to insert the particles with a random position on the inlet boundary and with a constant mass flow rate. Therefore, at each time step, a number of particles to be generated are defined, depending on both the mass flow rate and the inlet time duration imposed. Each individual particle will be assigned a position, an index and certain properties, such as: initial velocity and temperature of the particle, velocity and temperature of the fluid and laser intensity at the location of the particle, the liquid mass fraction, and the diameter of the particle, which will be calculated following a Gaussian distribution with mean and variance previously imposed. The simplified function `ParticlesDynamic::generate_particles` implemented for the generation of the particles is shown below.

```
void ParticlesDynamic::generate_particles(unsigned int n_particles_per_time_step)
{
 for (int i = 0; i < n_particles_per_time_step; i++)
 {
  // Evaluate initial particle location, velocity and diameter
  Point<dim> particle_location = random_particle_location();
  Point<dim> particle_velocity = initial_particle_velocity();
  double particle_diameter = gaussian_distribution();

  // Update particle properties
  std::vector<double>  initial_properties(num_properties);
  update_properties(initial_properties);

  // Mapping particle location and insert in the particle_handler
  auto reference_cell =
  GridTools::find_active_cell_around_point(dof_handler, particle_location);

  Point<dim> particle_reference_location =
  StaticMappingQ1<dim>::mapping.transform_real_to_unit_cell(reference_cell,
```

```
                                                    particle_location);

  auto it = particle_handler.insert_particle(Particles::Particle<dim>
  (location, particle_reference_location, next_particle_index),reference_cell);

  // Set particle properties
  it->set_properties(initial_properties);
  location.clear();

  // Update particle index
  ++next_particle_index;
 }
}
```

Here, the function `insert_particle` of the `ParticleHandler` class is able to insert a particle into the collection of particles `particle_handler` previously defined. In particular, it returns an iterator to the new position of the particle, together with a copy of the particle and its properties, including the particle index p_{id} that can be called with the function

```
it->get_id();
```

Advection

After generating the particles they will begin to move and exchange energy with the surrounding fluid. Their dynamics is influenced both by the interaction with the fluid itself and by the initial conditions imposed in the function `ParticlesDynamic::generate_particles`. For this purpose an ad hoc function has been implemented, `ParticlesDynamic::advect_particle`, in which the Eulerian values of the solutions found for the fluid velocity and temperature are first interpolated at the particle location, and subsequently, depending on the position of the particle once it has evolved in the domain, it is checked whether this remains within the boundaries, or tends to exit through them. If the particle is out of the domain, it is necessary to determine if it has crossed an outlet boundary, and therefore cancel it, or a wall type buondary, and then calculate its rebound. The commented code that handles this type of operations is shown below.

```
void ParticlesDynamic::advect_particle
(Particles::ParticleHandler<dim>::particle_iterator &p)
{
  // Evaluate Eulerian fields at particle location
  double temperature_interpolated_field = interpolate(fluid_temperature, p);
  double laser_interpolated_field = interpolate(laser_intensity, p);
  vector<double> velocity_interpolated_field = interpolate(fluid_velocity, p);

  // Update particle temperature and liquid mass fraction
```

```
double new_particle_temperature =
update_particle_temperature(temperature_interpolated_field,
                            laser_interpolated_field, p)

liquid_fraction = ((new_particle_temperature - settings.solid_temperature)/
                   (settings.liquid_temperature - settings.solid_temperature));

// Update particle velocity
std::vector<double> new_particle_velocity =
update_particle_velocity(velocity_interpolated_field, p);

// Evaluate new particle location
Point<dim> new_particle_location = old_location + dt * new_particle_velocity;

// Check if the particle is inside the domain
try
{
     auto coming_cell = GridTools::find_active_cell_around_point
          (triangulation, new_particle_location);

     // Update particle properties
     p->set_location(newLoc);
     std::vector<double>  updated_properties(num_properties);
     update_properties(updated_properties);
     p->set_properties(updated_properties);
}
// Exception: particle is not in the domain
catch (...)
{
    // Check which kind of boundary is crossed by the particle
    auto rebound_boundary_id = check_crossed_boundary(p);

    if (rebound_boundary_id == settings.walls_manifold)
    {
       // Calculate the rebound conditions
       std::vector<double> rebound_velocity = calculate_rebound_velocity(p);
       Point<dim> rebound_location = old_location + dt * rebound_velocity;

       // Update particle properties
       p->set_location(rebound_location);
       std::vector<double>  updated_properties(num_properties);
       update_properties(updated_properties);
       p->set_properties(updated_properties);
    }
    else if (rebound_boundary_id == settings.outlet_manifold)
    {
       // Outlet boundary is crossed, the particle is deleted
       delete_particle(p);
```

```
        }
    }
}
```

In this function all the properties of the particle are updated, according to its interactions with the mesh-based CFD fields, in order to keep track of all its evolution over time. Of particular use for writing this procedure were the functions `ParticlesDynamic::interpolate` and `GridTools::find_active_cell_around_point`. The first has been rewritten taking inspiration from the one already present in the library of dealii, that is `Particles::Utilities::interpolate_field_on_particles`, which allows to know the Eulerian field in correspondence with the particle location. The second, on the other hand, returns us directly to the cell E within which the particle is located. If the particle is not present in any cell, this function goes into exception, indicating that we are out of domain, and therefore an outlet or rebound condition has occurred.
The more careful observer would have noticed that in this procedure only the dynamics of the single particle are managed. The loop on all particles is managed by a second function `ParticlesDynamic::advect_particles()` which has been implemented by parallelizing the calculation on several processors, and it is presented in the following section.

Multithreading

To effectively exploit all hardware resources, the heaviest part of code from the computational point of view, that is the advecting of the particles, is implemented in parallel. This is possible as the particle evolution is independent from particle to particle and the fluid-particle interaction occurs only once velocity and temperature fields has been found. For this purpose, all the particles that are present in each time step are subdivided into all the processors available. Each processor calculates the evolution of the particles of its competence, and when all the processors run out the algorithm moves on to the next operation.

```
void ParticlesDynamic::advect_particles()
{
    // Mutex used by threads
    std::mutex m;

    // Condition variable used to see when all threads finish
    std::condition_variable cv;

    // Number of threads still active
    std::atomic<int> counter(0);

    // Vector of thread ids and dof ranges
    std::vector<std::thread::id> threadids;
    std::vector<std::pair<std::size_t, std::size_t>> threadranges;
```

```
auto loop = [&](std::size_t begin, std::size_t end)
{
    auto p = particle_handler.begin();
    int i = 0;
    while (i != begin)
    {
        i++; p++;
    }
    for(i = begin; i<end; i++, p++)
    {
        advect_particle(p);
    }
};

// Get number of threads to run
std::size_t nthreads = std::thread::hardware_concurrency();

// Get number of eqs per thread and the remainder
int qq = next_particle_index / nthreads;
int rr = next_particle_index % nthreads;

std::size_t s, e;

// Build start/stop ranges per each thread
for (std::size_t i = 0; i < nthreads; i++) {
    // Start
    s = (i == 0) ? 0 : threadranges.back().second;
    // End
    e = (i < rr) ? (s + qq + 1) : (s + qq);
    // Store
    threadranges.push_back(std::make_pair(s, e));
}

// Create threads
for (std::size_t i = 0; i < nthreads; i++)
{
    // Increase the number of running threads
    counter++;

    // Worker thread
    std::thread worker([&]()
    {
        // Index of thread in the vector of ids
        std::size_t idx;

        // Critical start
        {
```

```
                std::lock_guard<std::mutex> lk(m);

                // Push the thread id
                threadids.push_back(std::this_thread::get_id());

                // Index is the last entry in the vector
                idx = threadids.size() - 1;
            }

            // Critical end
            int start = threadranges[idx].first;
            int stop = threadranges[idx].second;

            // Loop over the given equations
            loop(start, stop);

            // Critical start
            {
                std::lock_guard<std::mutex> lk(m);
                counter--;
                cv.notify_all();
            }
            // Critical end
        });

        // The single threads starts
        worker.detach();
    }

    // Wait for all threads
    std::unique_lock<std::mutex> lock(m);
    cv.wait(lock, [&]() { return counter == 0; });
}
```

The C++ Thread support library, which includes built-in support for threads, mutual exclusion, condition variables, is used for the implementation. In particular, `std::thread` is the thread class that represents a single thread of execution in C++, and when a `thread` object is created a new thread starts executing the code passed into its constructor. In order to prevent that different threads execute concurrently and access the same memory locations a `mutex` lockable object, of the `std::mutex` class, has to be defined. It is designed to signal when critical sections of code need exclusive access and provides exclusive ownership. An other useful class is `std::condition_variable` whose object is able to block a thread, or multiple threads at the same time, until another thread notifies to resume (`notify_all`). It uses a `unique_lock` (over a mutex) to lock the thread when one of its wait functions is called. The thread remains blocked until woken up by another thread that calls a notification function

on the same `condition_variable` object. Furthermore the `std::atomic` template is used to define an `atomic` type. This ensure a well-defined behaviour if one thread writes to an atomic object while another thread reads from it. All the classes just defined are used in the code implementation and for more details see [214].

5.5.3 The fully coupled Particles-CFD algorithm

After showing the algorithms to calculate velocity, pressure and temperature of the fluid, see Section 5.3, and those to manage particle dynamics, see Section 5.5, the global implementation that couples the Eulerian field variables with a Lagrangian particle tracking method is presented. Then, Algorithm 5 shows the complete management of the fully coupled problem presented in Section 3.2.
The first step is to create, or import, the mesh that defines the computational domain and then mark its boundary to easily define the boundary conditions of the equations. Then all the variables, both those of the fluid, such as velocity, pressure and temperature, and those relating to the particles, are initialized with the initial conditions of the problem. At this point it is possible to start the iterative procedure over the time, which allow us to find the solution to each time increment dt. We proceed, first of all, in solving the Navier Stokes system of equations finding velocity and pressure solutions at time $t+1$, depending on the numerical scheme chosen. As widely explained in Section 5.3, three solution methods are possible: the mNR (see Algorithm 1), the PC (see Algorithm 3) and the iPC (see Algorithm 4). Then the temperature T^{t+1}, a function of the velocity $\boldsymbol{u}^{t+1}$ just found, is calculated following the procedure written in the Algorithm 2. At this point, known all the field variables $\boldsymbol{u}$, p, and T at time $t+1$, we move on to the generation and subsequent advection of the particles, respectively managed by the `generate_particles` function and by the `advect_particles` function, described in Section 5.5. The former deals with creating, at each time step, a collection of particles in the inlet portion of the domain, according to an imposed constant mass flow rate, in which each single particle has its own diameter, density, and initial values of both velocity and temperature. The latter, on the other hand, manages the actual dynamics of the particle, using the equations (5.186), (5.187) and (5.188), to describe both the motion and the thermal evolution of the particle irradiates by the laser beam. Finally, before updating the time variable for the calculation to the next time step, the results at the current time step are plotted.

Algorithm 5 Coupled Lagrangian Particles - CFD scheme.

$\Omega = \bigcup_{e=1}^{n_{el}} \Omega^e,\ \ \Gamma^e = \partial\Omega^e$ ▷ create mesh and mark boundaries
$\boldsymbol{u} = \boldsymbol{u}^0,\ p = p^0,\ T = T^0$ ▷ initialize Eulerian fields: velocity, pressure and temperature
$\boldsymbol{u}_p = \boldsymbol{u}_p^0,\ T_p = T_p^0,\ n_p$ ▷ initialize particles: velocity, temperature and number per time step
$t = 0$ ▷ initialize time variable

while $t < t_{tot}$ **do** ▷ time loop
  if `(NS_scheme == "non_linear")` **then**
    $\boldsymbol{u}^{t+1}, p^{t+1} = \text{mNR}\ (t, \boldsymbol{u}^t, p^t)$ ▷ solve Navier-Stokes with Algorithm 1
  else if `(NS_scheme == "linear")` **then**
    if `(linear_scheme == "PC")` **then**
      $\boldsymbol{u}^{t+1}, p^{t+1} = \text{PC}\ (t, \boldsymbol{u}^t, p^t)$ ▷ solve Navier-Stokes with Algorithm 3
    else
      $\boldsymbol{u}^{t+1}, p^{t+1} = \text{iPC}\ (t, \boldsymbol{u}^t, p^t)$ ▷ solve Navier-Stokes with Algorithm 4
    end if
  end if
  $T^{t+1} = \text{T}\ (t, \boldsymbol{u}^{t+1}, T^t)$ ▷ solve heat equation with Algorithm 2
  generate_particles $(n_p, \boldsymbol{u}_p^t, T_p^t)$ ▷ generate particles
  advect_particles $(\boldsymbol{u}^{t+1}, T^{t+1})$ ▷ evolve particles
  output_results ▷ generate output files at current time step
  $t = t + dt$ ▷ update time variable
end while

5.5.4 Benchmark 6: OpenFOAM particle tracking method comparison

This test is proposed to validate our coupled Particles-CFD implementation taking as reference the model proposed in the OpenFOAM platform. The geometry of the problem, together with boundary conditions imposed, are depicted in Figure 5.32. It consist in a 2D channel of height $H = 2\,m$ and length $L = 10\,m$, so that a parabolic velocity profile can be achieved. For the fluid phase, a constant velocity, equal to $\boldsymbol{u} = (1.0, 0.0)$, is used as boundary condition on the inflow portion Γ_{in}, stress-free conditions are set at the outflow boundary Γ_{out}, whereas on both the upper and the bottom walls a no-slip assumption is made. Regarding the disperse phase, several particles are created next to the inflow boundary Γ_{in} with a velocity set to zero, so that the fluid-particle interaction is particularly enhanced. Their diameter and density are assumed to be fixed and equal to $d_p = 50\,\mu m$ and $\rho_p = 1000\,kg/m^3$, respectively.

Even if the problem does not present particular irregularities, it was decided to use the

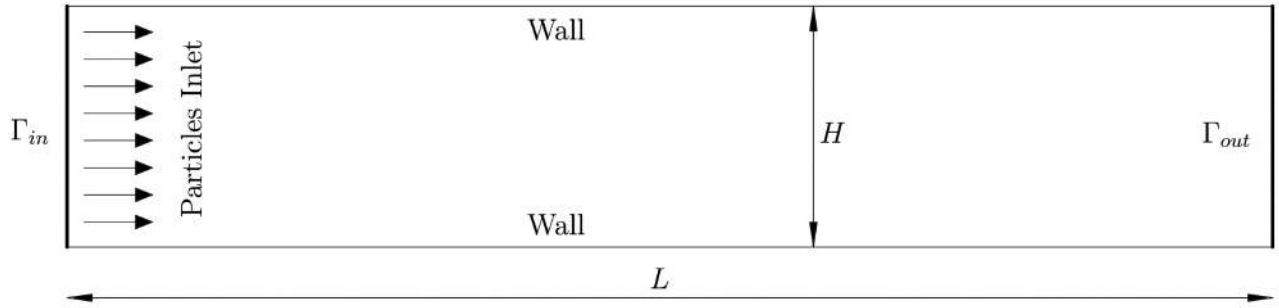

Figure 5.32: Geometry considered for the comparison with OpenFOAM.

mNR scheme proposed (see Algorithm 1) to solve the fluid flow, while the disperse phase evolution is calculated as explained in Section 5.5. The time step is set equal to $dt = 0.001\,s$. A preliminary comparison of the development of the velocity profile in the two performed numerical simulations, together with particles distribution at $\boldsymbol{x} = 5\,m$, are shown in Figure 5.33, on the left. The distribution of the particles, transported by the fluid, shows an excellent agreement between the results obtained with our code and that calculated using the OpenFOAM software. In order to be sure that the particles assume a similar velocity value, a comparison of the velocity profiles, at a fixed $\boldsymbol{x} = 5\,m$, has been plotted in Figure 5.33, on the right. It can be noted that the difference is minimal, and therefore we can conclude that our implementation does not present considerable inaccuracies and fit well both particle velocities and positions predicted by OpenFOAM.

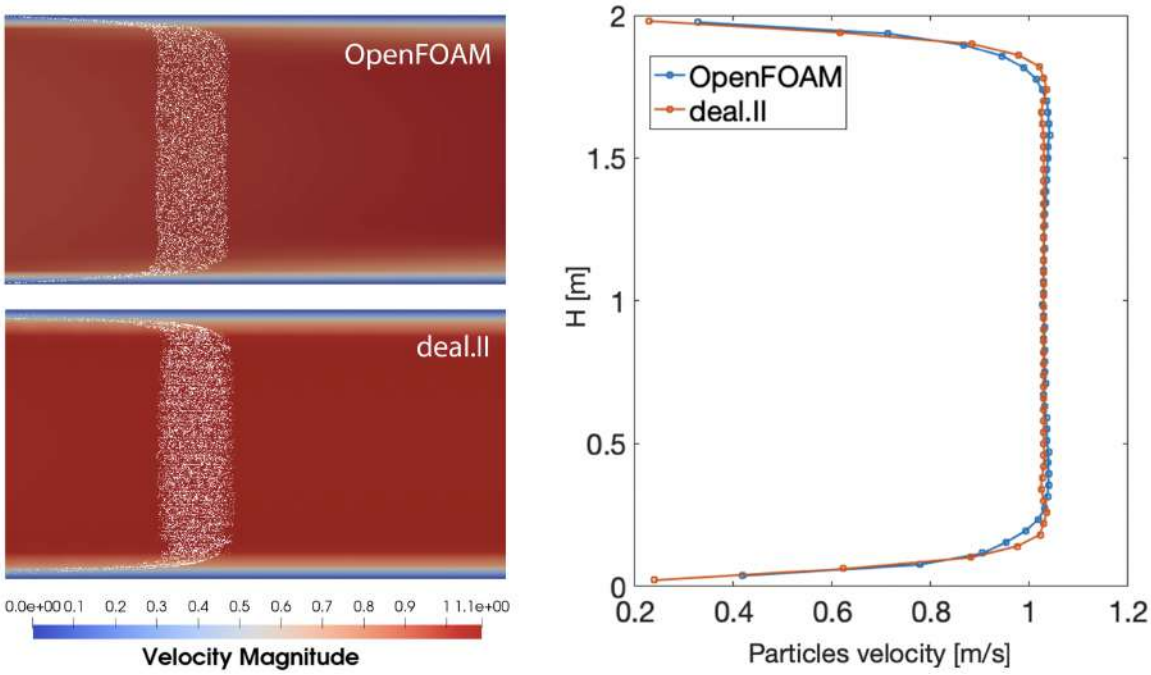

Figure 5.33: Comparison in computing particles dynamic between OpenFOAM and our deal.II algorithm.

Numerical results

6.1 LMD simulations with OpenFOAM

6.1.1 Simulation set up

In this Section we provide several settings of the numerical campaign, addressed also to have a preliminary comparison between numerical simulations and experimental evidences. In particular, we refer to the pilot experimental study conducted by PoliMi [215], where a custom fabricated LMD printer, using a triaxial nozzle has been employed.

Geometry and physical settings The tested geometry of the LMD printer consists of a nozzle with an height of 42 mm, the powder channels inclined by 21^o, with a diameter of 2.5 mm, and the laser channel with diameter of 6 mm (see Figure 6.34).

Since we are dealing with a metallic powder, we describe it with the following assumptions: i) the particles are considered to be spherical; ii) all the particles have the same radius, *i.e.*, the mean radius of a normal distribution; iii) the size of the particles does not change in time. The density of the particles is larger that the gas density ($\rho_\sigma \gg \rho_\phi$) and we assume the gas density ρ_ϕ to be equal to 1.145 kg/m^3, and the metal powder density ρ_σ to be equal to 8000 kg/m^3. This allows to consider the drag force as the main cause of the particle motion. Moreover, the nitrogen kinematic viscosity ν_ϕ is set equal to $15 \cdot 10^{-6}$ m^2/s, whereas, since in the Eulerian-Eulerian approach we describe metal powder as an equivalent fluid, we compute the kinematic viscosity associated to the solid phase, ν_σ, using the kinetic theory of granular flow implemented in OpenFOAM [216].

Velocities inlets of both gas and particles are directly taken from the experiments (see Table 6.1) and are used as boundary conditions of the problem. In the experiments (see Sec. 6.1.5) particle velocities proved to be strongly affected by nitrogen flow rate inserted in the system, specifically set to the values $\{5, 7.5, 10\}\, l/min$. Hence, we impose inlet velocities equal to $\{1.4, 1.6, 1.8\}\, m/s$, for both particles and gas phases, disregarding to accurately reproduce the flow developing along the nozzle channels. The values of powder and gas flow rate allowed us to identify the number of particles and the particle volume fraction to impose at the inlet boundaries.

The Reynolds number is estimated under the hypothesis of a fluid velocity equivalent to the maximum value imposed at the inlet, that is 1.8 m/s. The resulting Reynolds number,

about 300, is consistent with the assumption of a flow modeled in laminar regime. Moreover, the particle size ensures that fluid-solid interaction does not cause turbulence phenomena, as showed by Kussin and Sommerfeld in [217], where the turbulence intensity decreases significantly for particles with diameter ranging between 60 μm and 190 μm.

For the gas phase, no-slip conditions are considered along the nozzle head channel walls; for the particle phase, in the Eulerian formulation slip condition at the walls are assumed, whereas, in the Lagrangian framework the non-dimensional coefficients of restitution and kinetic friction are set equal to $e = 0.97$ and $f_r = 0.09$, respectively, to model the particle collision with the walls as expressed in eq. (5.153). Such coefficients are determined by considering spherical steel particles which exhibit a low loss of kinetic energy when impacting with walls [218].

As concerning the boundary conditions imposed for the Navier-Stokes problem, the pressure at the outlet is imposed to be to equal to the atmospheric pressure, while at the inlet the velocity is fixed to the values described above. For more details regarding how OpenFOAM numerically handles such a type of Navier-Stokes boundary conditions see [54].

Discretization and numerical settings The computational domain consists of a cylinder in which the nozzle is included, thus defining the whole computational domain as depicted in Figure 6.34.

The deposition head is formed by three channels, in which gas and particles pass through, and an additional channel, bigger than others and positioned at the center of the nozzle, accounted for laser beam irradiation. The domain is discretized using tetrahedral elements. The mesh has to be fine enough to reach small Courant number and make the simulation stable, but a severe limitation of the solver occurs in the case of the Eulerian-Lagrangian approach. In fact, the size of the particles must be sufficiently small in comparison with the computational grid so as to allow the coarse averaging procedures (see Sec. 5.2.3) to provide accurate interpolations of the Lagrangian properties, i.e., particle volume fraction, particle velocity, and fluid-particle interaction force [193].

The smallness of the Courant number influences also the time-step size which is set to $1 \cdot 10^{-5}\, s$ in our computations. The time discretization methods of explicit Euler scheme and Runge-Kutta scheme are employed for solving in time the Navier-Stokes momentum equation (5.139b) and the transport equation (5.148), respectively. Gauss-Seidel iterative solvers with an imposed tolerance equal to $1e-06$ are used. As concerns the spatial discretization, we adopted a standard finite volume approach using Gaussian integration with linear interpolation (central difference method), so as to compute gradient, laplacian, and divergence operators. See [54] for more details.

6.1.2 On the comparability of the two approaches

In order compare the Eulerian-Eulerian (EE) and the Lagrangian-Eulerian (LE) approaches, we first performed a preliminary investigation at the inlet to have the two models to be comparable

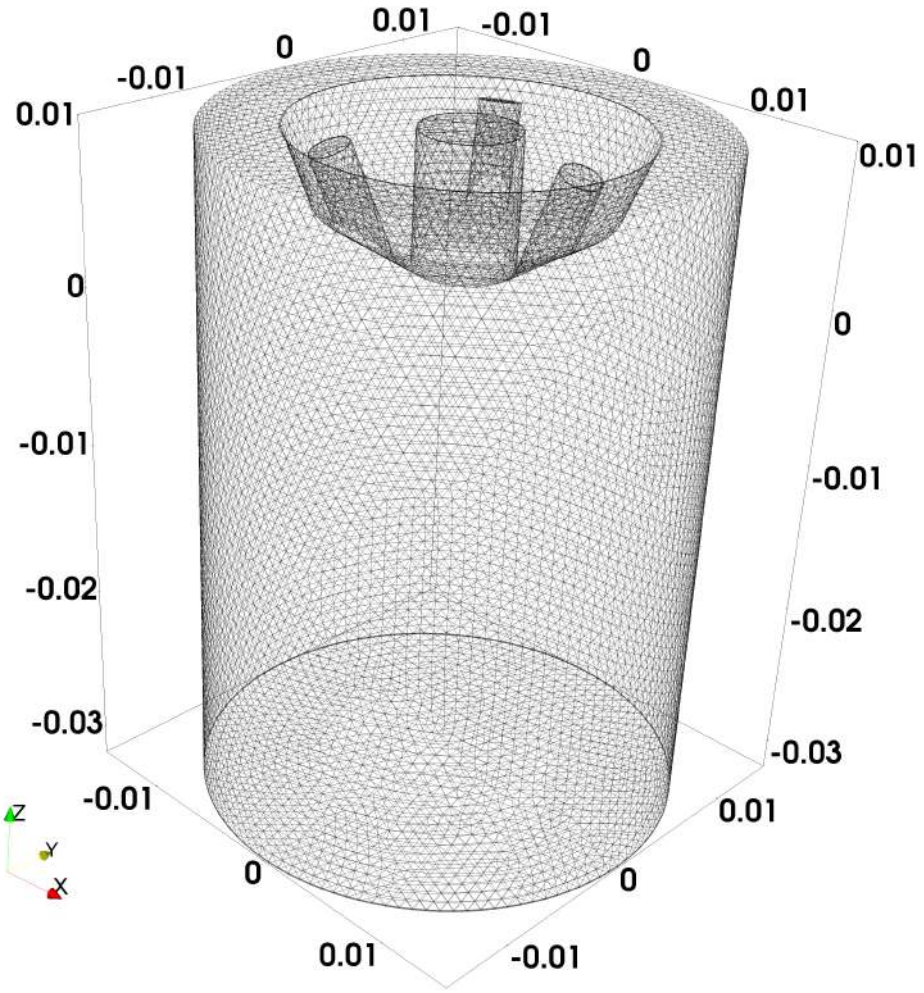

Figure 6.34: Computational domain used for both Eulerian-Eulerian and Eulerian-Lagrangian approaches

from a physical viewpoint. Then, volume fraction and velocity fields are carefully compared in terms of both gas and solid phases.

The numerical campaign is also driven in order to determine the properties of the powder cone, specifically: (i) the minimum diameter of powder cone [mm]; (ii) the distance of the minimum diameter from the exit of the nozzle [mm]; (iii) the particle velocity [m/s]. Such properties defining the powder shape cone outside the deposition head are conceived as testbeds for future works of identification and validation with experiments. A preliminary, qualitative, comparison is introduced at the end of the present Section, to show the capability of the developed numerical schemes on predicting experimented trends.

The goal of the first test is to compare the amount of solid fraction, i.e., the mass, entering in the domain for both the approaches. For the Eulerian-Eulerian approach the solid volume fraction is directly imposed, whereas, for the Eulerian-Lagrangian approach, the number of particles has to be assigned at the boundary inlets. These two different input data, that represent the same mass, must return the same conditions at the inlet.

Within the EL approach, the number of particles to impose at the inlet of a single channel

is estimated as follows. Let be V_ϕ the known volume of a single particle and ρ_ϕ its relative density. Then, the mass of a particle, m_ϕ can be simply calculated as the product between V_ϕ and ρ_ϕ. Known the mass m_ϕ, the number of particles per second to be introduced in each single channel can be evaluated as:

$$n_\phi = \frac{\dot{m}_\phi}{m_\phi} \frac{T_{sim}}{n_{ch}} \tag{6.203}$$

where $\dot{m}_\phi$ is the total powder flow rate, n_{ch} the number of nozzle channels and T_{sim} the simulation time.

Regarding the EE approach, the mass relative to the particles to introduce in the system, as boundary condition at the inlet, has to be computed in terms of solid volume fraction, which is defined as:

$$\alpha_p = \frac{\sum_i V_{\phi,i}}{V_{tot}} \tag{6.204}$$

where $V_{tot} = \sum_i V_{\phi,i} + V_\sigma$ is the total volume introduced in the system, defined as the sum of the particles volume plus the carrier gas volume. The volume occupied by the particles $\sum_i V_{\phi,i}$ is directly derived from eq. (6.203) as the product between the volume of a sigle particle V_ϕ and the number of particles introduced at the inlet n_ϕ. On the other hand, the volume occupied by the gas phase is evaluated assuming a uniform velocity field of the gas phase $\boldsymbol{u}_\sigma$ at the inlet. Then the carrier gas volume injected through the channels is obtained as:

$$V_\sigma = \boldsymbol{u}_\sigma \sum_i A_{ch,i}\, T_{sim} \tag{6.205}$$

where $\sum_i A_{ch,i}$ is the sum of all channels area. This procedure is expected to ensure the same boundary conditions for both simulations.

Figure 6.35 depicts a color map of the volume fraction distributions at the inlet of the nozzle channels, for both the approaches. Such a figure prove that the inlet amount of density of the solid fraction entering in the domain is definitely equal for both the approaches, except for a small difference due to the speckled behaviour of the EL approach. Such a behaviour is due to the non-uniformity of the particle distribution in the computational domain, that is characterized by a random positioning at the inlet, and governed by a flow rate according with boundary conditions.

The comparability of the two approaches is finally validated by considering the mass distribution at regime. As specified in the conservation equations, the mass is expected to be preserved. In Figure 6.36 we then report the flow of the volume fraction, for both the solid and the gas phases, either inside and outside the nozzle, where the deposition process occurs. Note that the volume fraction flow is obtained by numerically integrating the pointwise values of the volume fraction. Such values are given for each cell of the computational domain (see Figure 6.34): then, fixing a certain value of z, for all the cells in the range $[z - \epsilon,\, z + \epsilon]$, with $\epsilon = 1\,mm$, we integrate through the midpoint rule the discrete volume fraction distribution to evaluate its flow value associated with z.

Figure 6.36(a) represents the powder outline formed below the nozzle. As explained in Section 3.1.2, such a result emphasizes the main difference between the two approaches. With

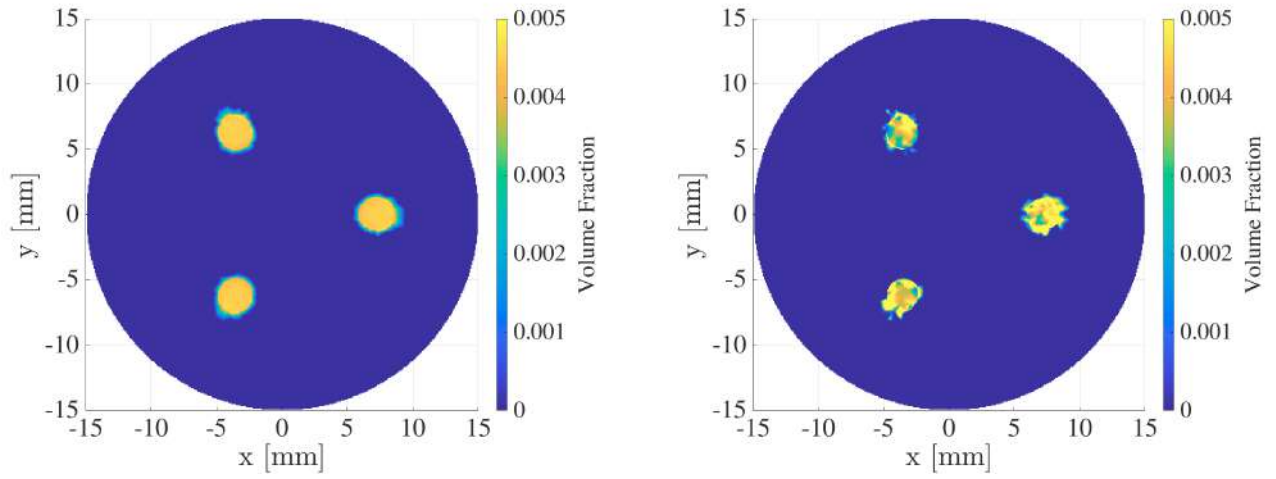

Figure 6.35: Color maps representing volume fraction distribution at the inlet. Eulerian-Eulerian approach (left) and Eulerian-Lagrangian approach (right). In the Eulerian-Lagrangian case, the volume fraction representation in non uniform due to the presence of solid particles.

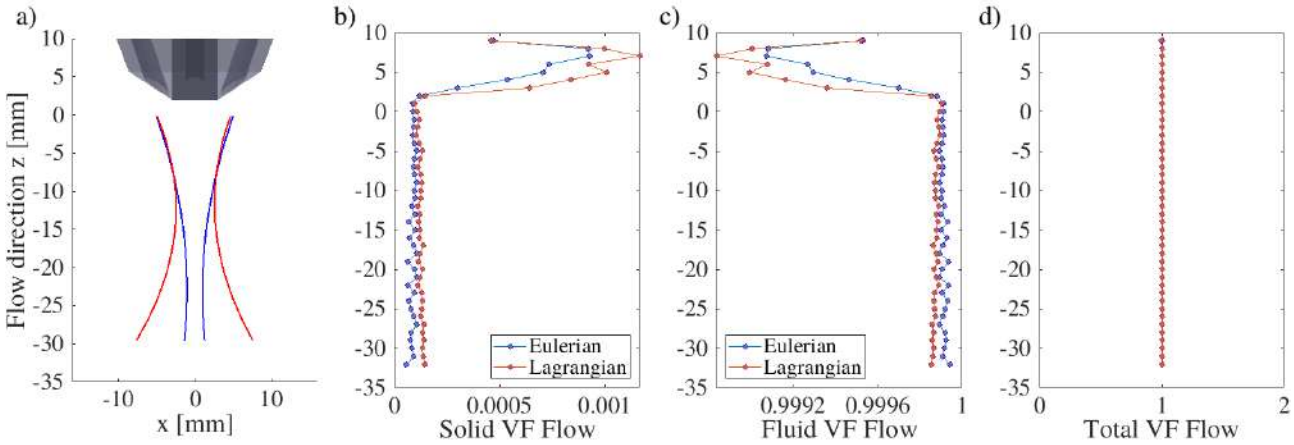

Figure 6.36: Comparison between Eulerian-Eulerian simulation (blue lines) and Lagrangian-Eulerian's (red lines). Figure a) reports the outline of the nozzle and the powder cone shape. Figures b) and c) show the solid and fluid volume fraction (VF) flow rate along the z-direction, respectively. The total volume fraction flow rate, that is the sum of the two previous quantities, is depicted in Figure d).

the Eulerian-Lagrangian approach, the powder ejected from the nozzle forms ah hourglass-like shape, whereas the powder outline obtained with the Eulerian-Eulerian approach remains constant after convergence of the powder streams. This behavior is due to the fact that Eulerian-Eulerian methods do not represent more than one velocity value for each phase in one computational cell, and then it can not capture crossing trajectories. This point will be further investigated later. As expected, the total mass is preserved and there are no appreciable differences between the approaches (Figure 6.36(d)). Figure 6.36(b-c) prove that the volume fraction distributions obtained with the two different approaches are very similar, for both solid and gas phases. The two simulations slightly differ in the upper zones inside the nozzle, where the

flows of volume fraction of both phases grow up as densities increase in the channels.

6.1.3 Velocity and mass fields

In this Section a comparison between the two approaches in terms of velocity and mass flows fields, measured in different ways, is reported. Figures 6.37 and 6.38 depict the velocity-magnitude 3D fields associated with the two phases respectively. For the solid phase (see Figures 6.37), the velocity fields of the two approaches appear quite similar in terms of magnitude. However, as mentioned in the previous section, the approaches significantly differ: the EE approach cannot reproduce the crossing trajectories predicted by the EL approach, which directly represents the velocity of each particle.

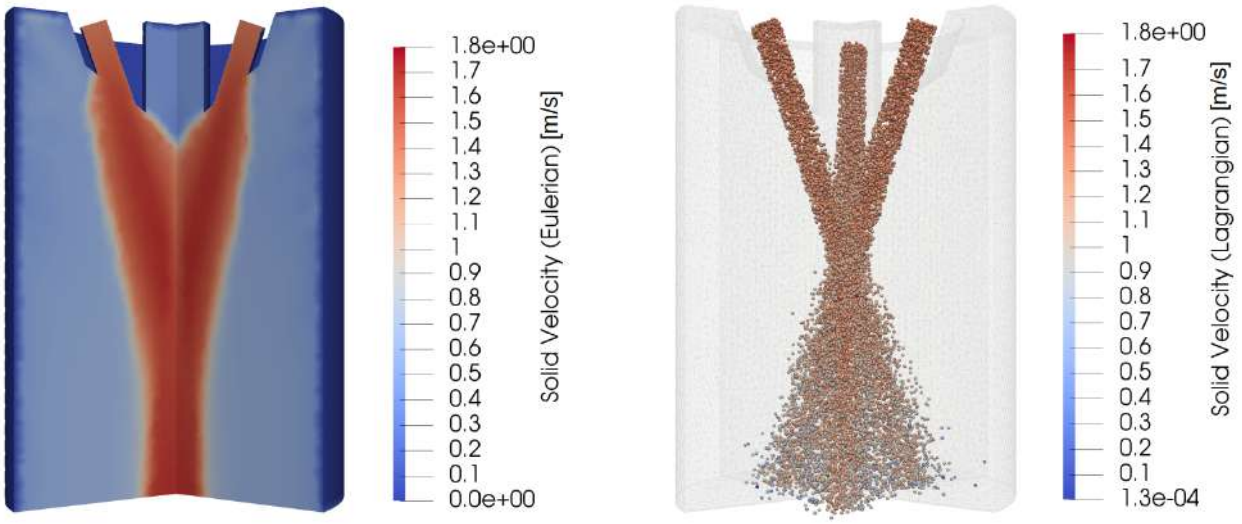

Figure 6.37: Magnitude of particles velocity field. Eulerian-Eulerian approach (left) and Eulerian-Lagrangian approach (right).

On the other hand, for the gas phase, the magnitude of the velocity field estimated by the EE approach differs from the one predicted by the LE approach (see Figures 6.38).

Such a result is furthermore emphasized by Figure 6.39, where the mass flow rates are reported along the z-direction (flow quantities are computed as done for Figure 6.36). Figure 6.39(b-c) confirm the results obtained in Figures 6.37 and 6.38: except for the part inside the nozzle, the solid mass flow rate calculated with the EL approach is slightly higher than the same quantity computed by the EE (Figure 6.39(b)), while the EE approach overestimates the mass flow rate associated with the gas phase (Figure 6.39(c)).

As concerning the solid phase, although small, the gap of mass flow rates between the two approaches is due to the different way of computing velocities: in the EE approach the velocity of the particles represents an average value calculated in each computational cell (see Figure 6.39(b)) and this can lead to an underestimation of the actual velocity of each particle,

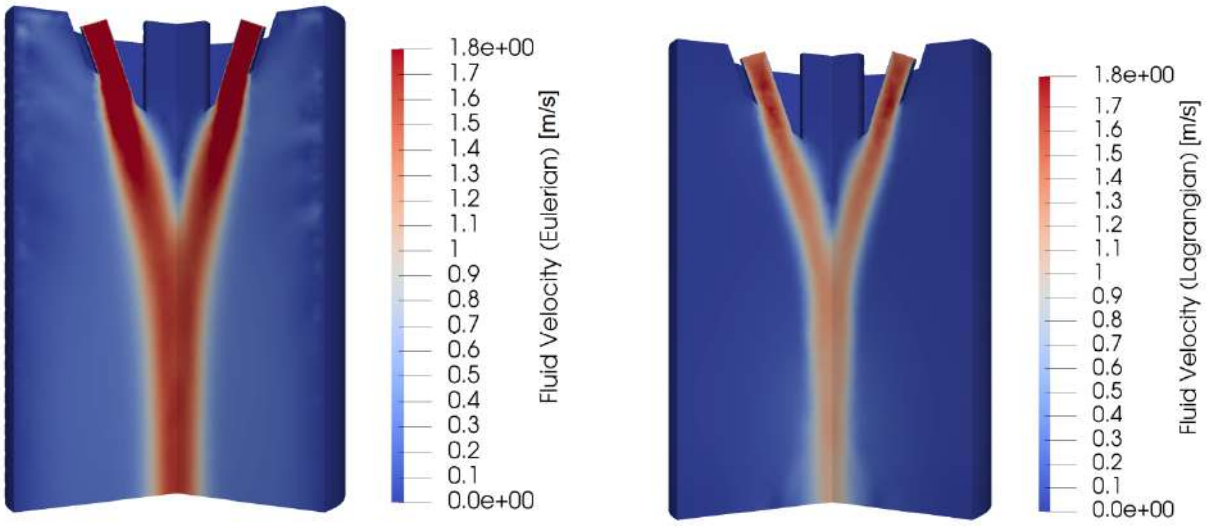

Figure 6.38: Magnitude of gas velocity field. Eulerian-Eulerian approach (left) and Eulerian-Lagrangian approach (right).

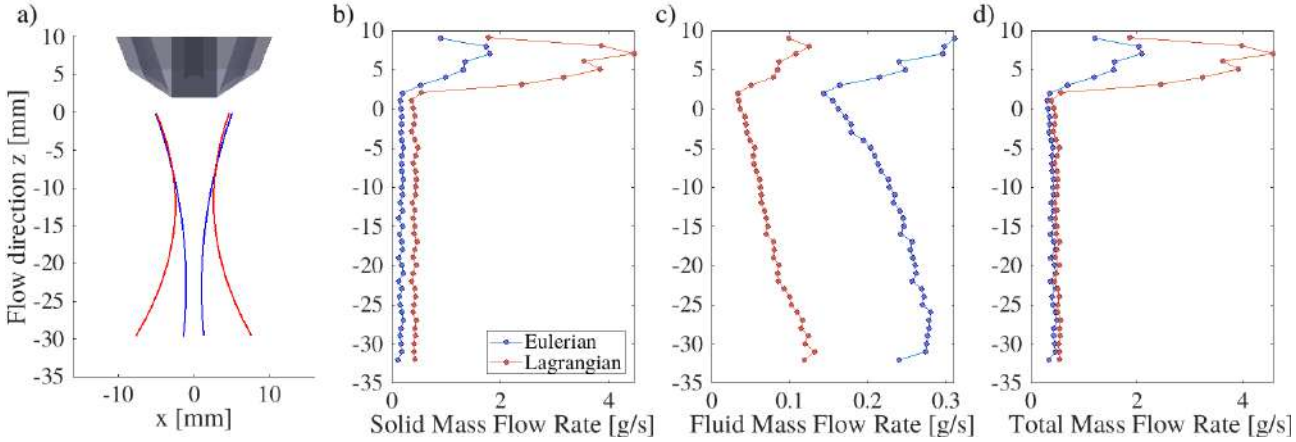

Figure 6.39: Mass flow rates along the z-direction. Comparison between Eulerian-Eulerian (blue) and Eulerian-lagrangian (red) approach.

which on the contrary are explicitly computed in the LE approach as pointwise value, in the spirit of the Lagrangian description.

The evaluation of the mass flow rate associated with the gas phase shows an opposite trend (see Figure 6.39(c)). Such a rate computed by the EL approach is lower than that predicted in the EE approach. This result can be seen as a consequence of the momentum balance (see eq. (5.139b)), that is numerically confirmed by Figure 6.39(d) showing how the total amount of mass flow rate is predicted very similarly in both the approaches.

Note that all such comments are valid only for the behavior outside the deposition head, whereas inside the nozzle channels discrepancies between the two approaches are evident. Especially in the evaluation of the solid phase, and in agreement with the results depicted in Figure 6.36,

a relevant increase of the mass flow rate is predicted by the EL approach in contrast with the EE approach. In fact, inside the nozzle channels the solid mass is greater than the fluid mass (see again Figure 6.36(b-c)): while the solid volume fraction reaches inside the channels a maximum value which is about 8 times the value outside the channels, the values of the gas volume fraction vary less than 1%.

The results regarding the velocity and mass field obtained with the two approaches are finally summarized in terms of mass flow ratio, mass ratio, and velocity ratio, respectively (see Figures 6.40). All such quantities are computed for a given time at each cross section. The

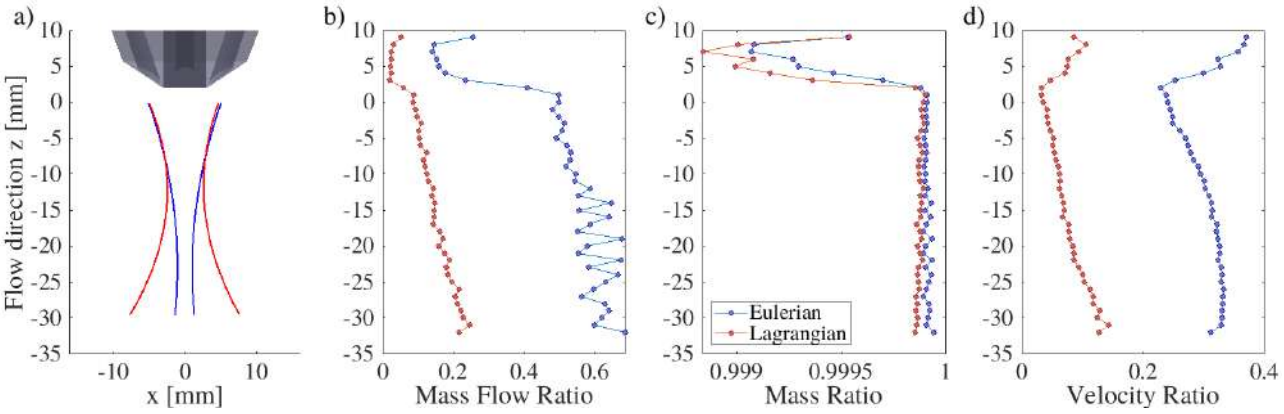

Figure 6.40: Mass flow ratio, mass ratio, and velocity ratio along the z-direction. Comparison between Eulerian-Eulerian (blue) and Eulerian-lagrangian (red) approach.

mass flow ratio is computed as the ratio between the mass flow rate of the fluid phase and the total mass flow rate; the mass ratio is computed as the ratio between the fluid volume fraction and the total volume fraction; the velocity ratio is computed as the ratio between z-direction velocity component of the fluid phase and the one of the solid phase.
The different predictions of the two approaches discussed above are here emphasized by the fact that both mass flow and velocity ratios are overestimated by the EE approach with respect to the EL approach, although the estimations of the mass ratio are very close in the two approaches.

6.1.4 Powder shape cone

In this Section, we report the predictions offered by the two approaches for what concerns the most significant quantities from an engineering viewpoint, i.e., the powder stream structure formed by the triaxial nozzle and its cone shape. More specifically, such quantities allow us to estimate the minimum diameter size below the nozzle, as we will present in the next Section.

Figure 6.41 shows the powder flux simulated by the EE approach in terms of volume fraction distribution. On the left a 3D representation of the ejection process is depicted, with a color scale mapping the values of particles volume fraction (red color corresponding to high values). On the right, the volume fraction distributions are represented on a X-Z plane cutting the central axis of the domain, that is where maximum particles concentration is reached. This figure shows that, as expected, the highest amount of volume fraction rise at the focal point, *i.e.* where fluxes cross each other.

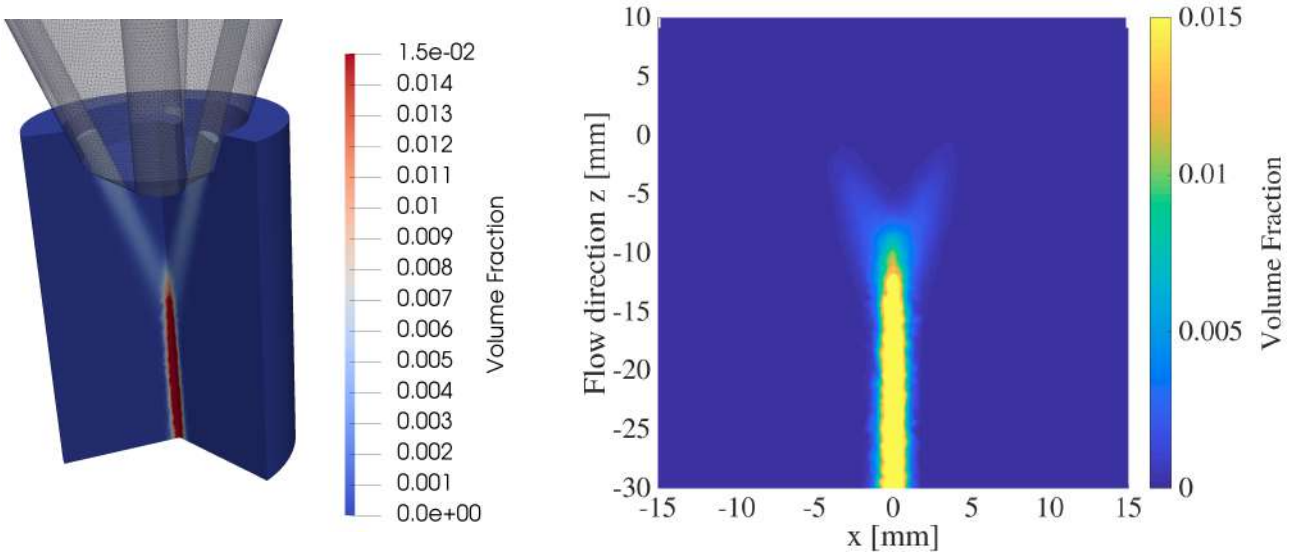

Figure 6.41: Eulerian 3D Simulation via EE approach: screenshot of the simulation (left), particle volume fraction (right).

The EL simulations are reported in Figure 6.42, where the same representations as above are depicted. In this approach, in contrast with the EE approach, the volume fraction of the solid phase is post-processed by using the information regarding the volume fraction of the fluid phase, which is computed in the Eulerian mesh: in each cell of the mesh, we then compute the particles volume fraction as $\epsilon_\sigma = 1 - \epsilon_\phi$.

Figure 6.43 reports a comparison between the Eulerian-Eulerian and the Lagrangian-Eulerian simulations in terms of powder cone shape. In particular, these trends are determined using a 3rd-order polynomial function that interpolates the most external particles, starting from the central axis, in order to capture the shape of the power flux.

As already mentioned, Figure 6.43 shows the key difference between the two approaches: the Eulerian-Eulerian method (blue line on the left) can not capture crossing trajectories and the three powder fluxes exiting from the nozzle merge each other in a single stream. Such a

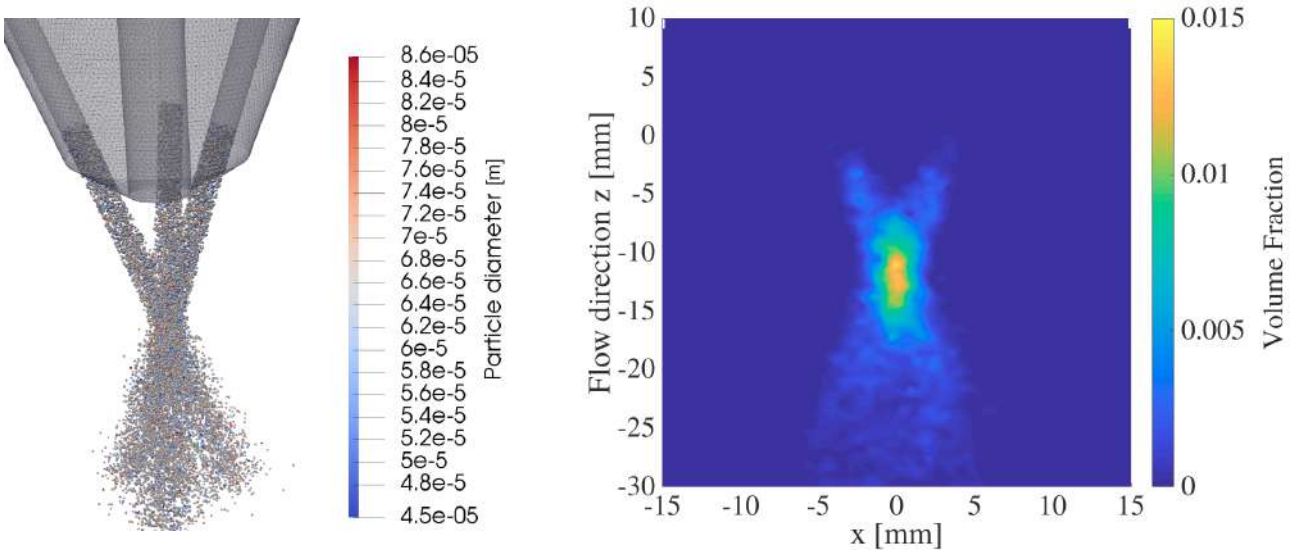

Figure 6.42: Lagrangian 3D Simulation via LE approach: screenshot of the simulation (left), particle volume fraction (right).

result is a consequence of the different formulations employed for the numerical approaches. In the Eulerian formulation the kinematics of the solid phase is represented only in terms of nodal values in the Finite Element mesh, while in the Lagrangian formulation the solid phase consist of particle-points that can move independently in the domain. When the flows of the three solid phases cross each other at a certain point of the domain (the focal point), the resulting velocity in the nodes is averaged in the Eulerian approach (see again Figure 6.38), thus configuring a single flow of increased density (see again Figure 6.41). On the contrary, the Lagrangian-Eulerian method (red line on the right of the same Figure) can reproduce the experimented behavior, that is the enlargement of total flux after particles have crossed each other in the focal point (see again Figure 6.42).

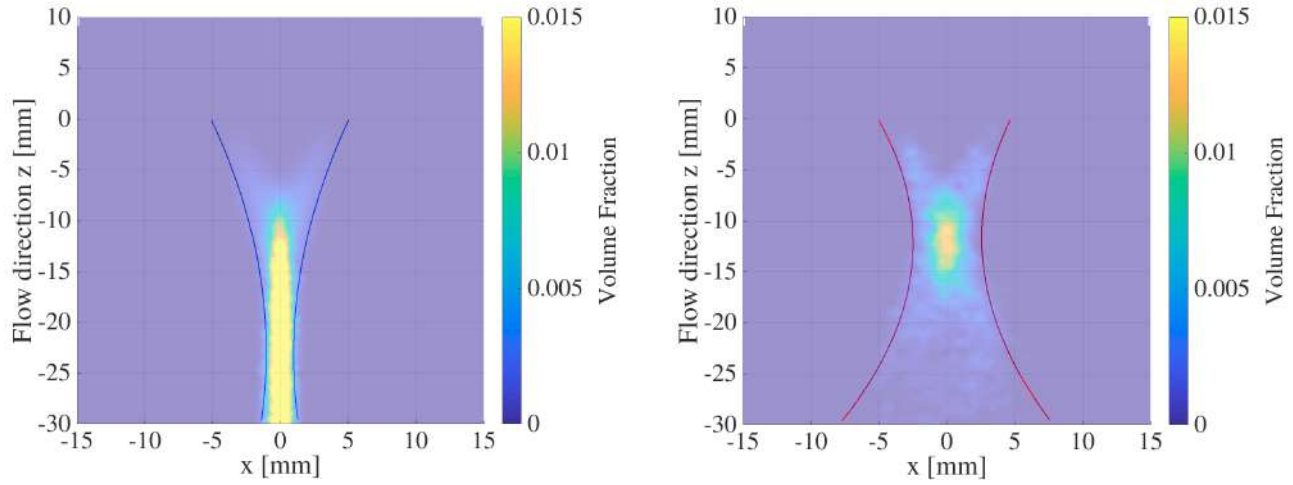

Figure 6.43: Comparison between Eulerian-Eulerian (left) and Lagrangian-Eulerian (right) simulations in terms of volume fraction and powder cone shape.

Finally, the two approaches are compared in terms of CPU times. All tests are conducted

on an off-the-shelf desktop computer with eight-cores Intel Core i7-6700 running at 3.40 GHz, with 24 GB of RAM, on 64-bit Ubuntu Linux 18.04.4 LTS. Note that the computational time of the Eulerian-Eulerian approach is independent from the mass flow rate considered at the inlet, whereas the Lagrangian-Eulerian one increases as the number of particles introduced in the process grows. In particular, as depicted in Figure 6.44, the LE approach has better performances than the EE approach when the number of particles is less than 10^5. For values greater than 10^5, the former tends to become more and more slow than the latter, costing up to six times more in the case of 10^6 particles.

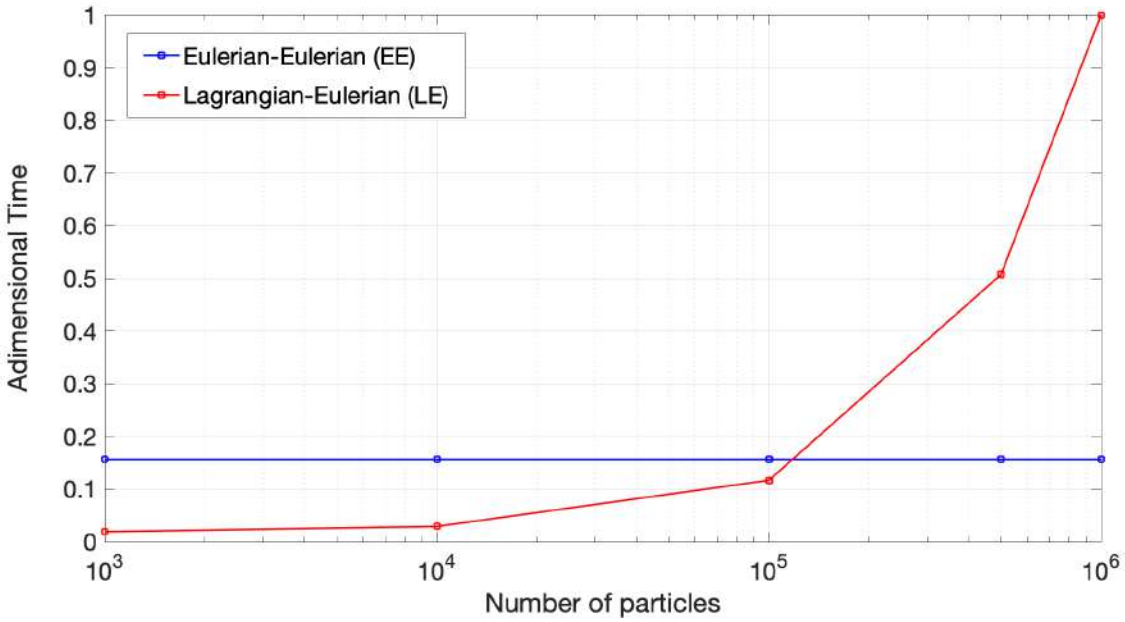

Figure 6.44: Comparison between Eulerian-Eulerian approach and Lagrangian-Eulerian approach in terms of CPU times nondimensionalized with respect to the maximum value 5540 secs. of CPU time required by LE running with 10^6 particles.

6.1.5 Towards experimental validation

Thanks to the collaboration of the Mechanical and Production Technologies laboratory of the Department of Mechanics at the Milan Polytechnic, it was possible to carry out an experimental campaign to measure the most significant quantities that occur in the printing process, in order to compare them with the numerical simulations. A brief description of the machinery and materials used, together with a description of the measurement methods and controlled parameters, is reported below.

Materials and methods

In the employed LMD machine, built by Milan Polytechnic (see Figure 2.2), the powder is delivered throughout a powder feeder (GTV Twin PF 2/2-MF) used to manage the powder

flow rate. The powder is stored in a hopper and it is gradually spread on a disc able to rotate. The rotation per minutes, RPM, can be changed as such as the flow rate of carrier and shielding gases. The powder flows are directed in the three-way nozzle by means of a powder splitter, which splits the carrier gas in three equal flows. In Figure 6.45 is reported a schematic representation of the employed set-up.

The three-way nozzle is also connected to a laser head (KUKA REIS MWO-I), mounted on

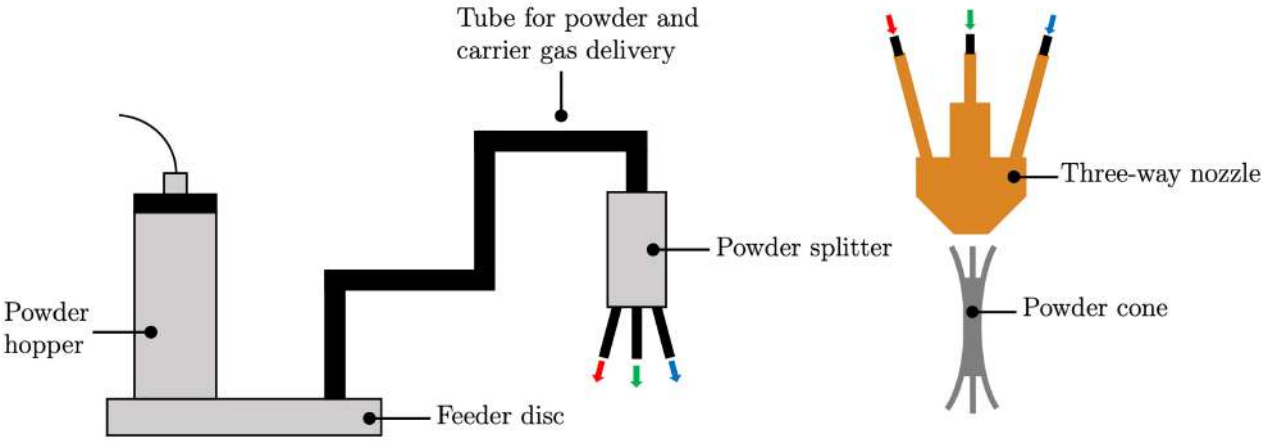

Figure 6.45: Schematic representation of the employed set-up

a 6-axis anthropomorphic robot (ABB IRB 4600-45), and equipped with a 200 mm focal lens and a 129 mm collimation lens with variable position. The experimental campaign is performed only on the factors affecting the powder cone without laser beam interaction.
The employed material is a stainless steel AISI 316 L powder from Carpenter Additive. The powder is characterized by a grain size distribution between $D_{10} = 45\,\mu m$ and $D_{90} = 90\,\mu m$. In Figure 6.46 the SEM images of employed powder are reported.

Measurements principle

The powder stream properties are measured with different approach. The geometrical aspects of powder cone are investigated by camera acquisition approach. The three-way nozzle is set with a dark background, facing a high speed camera (Photron, Mini AX_{200}), and a LED lamp is used in order to outline the powder cone shape. The high speed camera is fitted with 1 megapixel CMOS sensor capable to capture as much as 540000 frames per second. The camera is equipped by a macro lens (Tokina, AT-M_{100} AF PRO D) to have good resolution during the acquisition. The frames per second are set at 6400 with a shutter speed set at the minimum (1/6400 s). The measurement of particle velocity involves a different set-up. In fact, the LED lamp is replaced by laser sheet which ensures that only the particles travelling in its plane are taken into account. This method known as Continuous Particle Image Velocimetry (CPIV) is commonly used for particle traces in order to avoid errors from perspective. In order to realize the laser sheet, a laser diode with a round shape beam of 3.5 mm is used (Thorlabs, CPS532-C2). The beam spot is delivered and managed by a system of lenses. Firstly, the beam

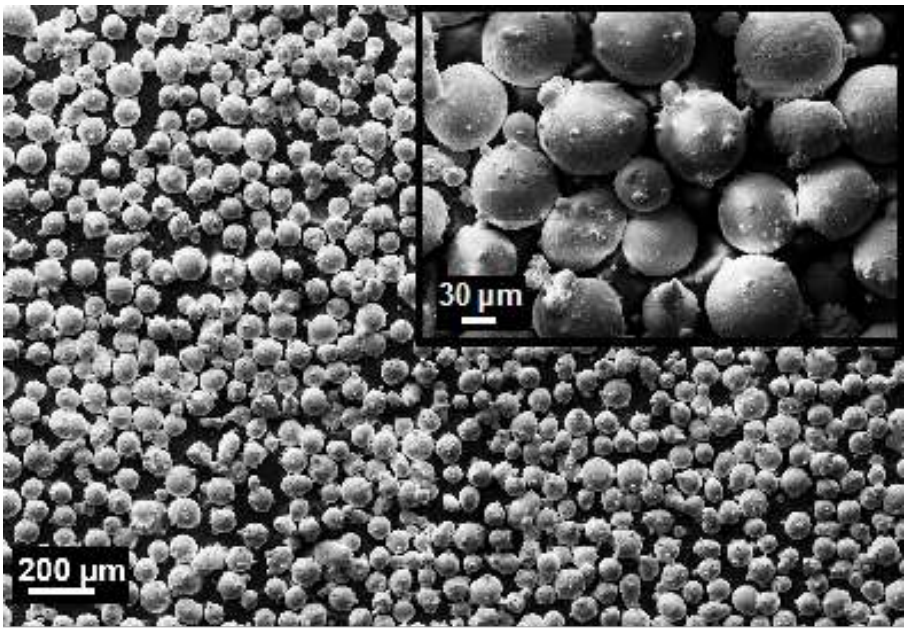

Figure 6.46: AISI 316L powder

spot is expanded passing through a plano-convex lens of 50 mm length. Another plano-convex lens of 200 mm length, is used in order to obtain a bigger collimated beam. A cylindrical lens of 200 mm length, transforms the round shape of the collimated beam in a laser sheet. In Figure 6.47 a sketch of employed set-up is reported. The acquisitions are performed with the same high speed camera. Nevertheless, the frames per second are set to 750. Also the shutter speed is changed to 1/1000 s.

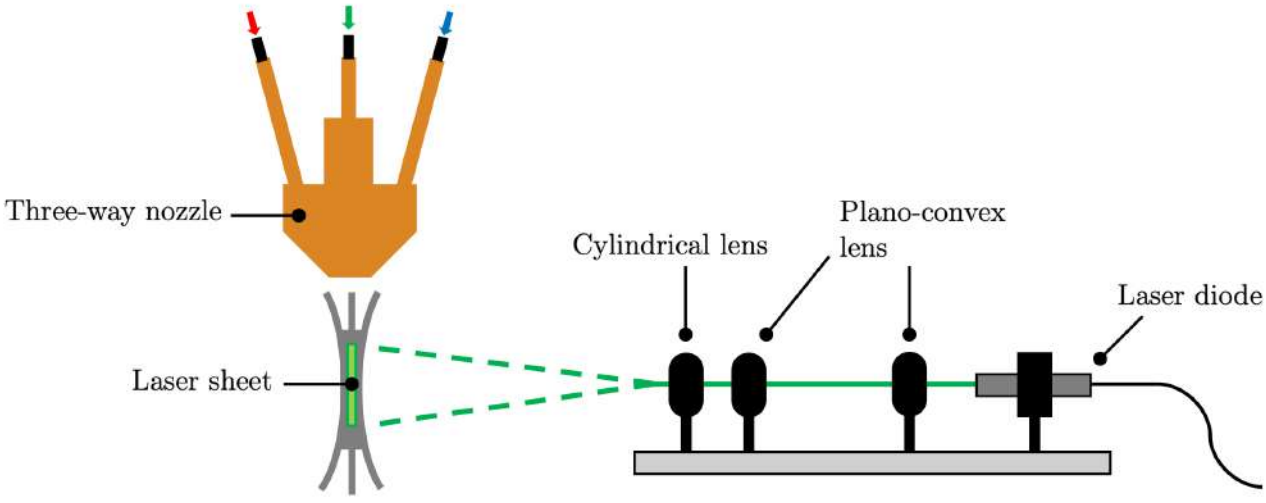

Figure 6.47: Employed set-up for particles tracing

Controlled factors

The experimental measurements are performed only on the factors which affect the powder cone without laser beam interaction. Three factors are varied on three levels for each one, and three replica of each condition are performed. In Table 6.1, the varied and fixed factors are reported. The experimental campaign is realized to determine the properties of powder cone, particularly: (i) minimum diameter of powder cone [mm]; (ii) distance of minimum diameter from the exit of the nozzle [mm]; and (iii) particle velocity [m/s].

Varied Factors			
Rotation per minutes, RPM	2	4	6
Carrier gas flow rate [l/min]	5	7.5	10
Shielding gas flow rate [l/min]	10	17.5	25
Fixed Factors			
Carrier gas	Nitrogen		
Shielding gas	Nitrogen		

Table 6.1: Varied and fixed factors of the experimental campaign

Comparison with numerical simulations

By varying the amount of powder, as well as the carrier and shielding gas flow rate, several configurations of particle cone shape and particle velocity can occur. Side-view images of the powder flow shape formed outside the working nozzle are carried out for these purposes. Pictures are captured with a high speed camera converted into binary format and then post-processed by means of the MATLAB Image Processing Toolbox. For each 2D picture, such a toolbox stores the first white pixel on every row of pixels, starting from left and right sides at the same time. This procedure provides an outline of the cone shape. On the other hand, the measurement of the powder velocity is performed tracking the particles distribution exiting from the nozzle. Thanks to a binarization image process, the maximum pixel distance of a particle trace can be obtained, to calculate the local particle velocities. An example of a binarizated picture used to compute the LMD flow features is shown in Figure 6.48.

The results of experimental data produced by PoliMi are then compared with Eulerian-Lagrangian numerical simulations. Figure 6.49 shows that simulations are able to capture the trend of the minimum diameter, changing the carrier gas flow rate, but not the absolute values of minimum diameter. This aspect is mainly due to the uncertainty regarding the geometric dimensions of the nozzle that significantly affects the results of the ejection process. In fact, the inclination and the dimensions of the channels, where both powder and carrier gas stream out, are not known a priori and they can only be estimated with appropriate measurements, not provided by the manufacturer.

Figure 6.48: Images displaying the binarization process. The image on the left is the original frame withdrawn from the imaging experiment. The one on the right shows the binary image after post-processing

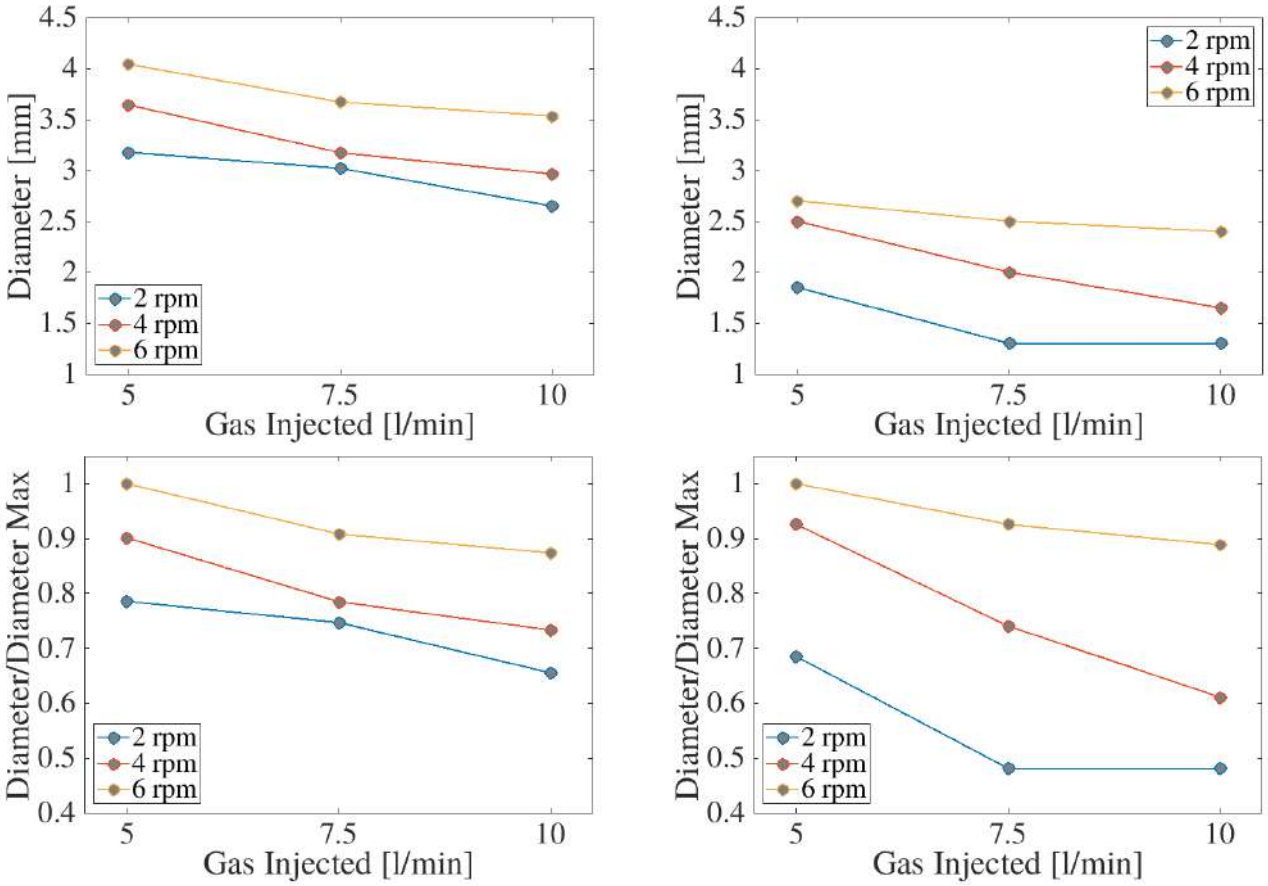

Figure 6.49: Comparison between our simulations (left) and PoliMi experimental data (right), in terms of both dimensional (top) and nondimensional diameter (bottom).

6.2 LMD simulations with Deal.II

This section presents in detail both set up and results of the simulations conducted with the numerical strategy discussed in Section 5.3, and in particular the Algorithm 5, to simulate the LMD manufacturing process. As widely explained in Section 2.1.1, the coupling of the powder with the temperature field is fundamental to describe the multiphysics dynamics that occur during the printing process, and in particular the interaction between particles and the laser beam [135, 219, 136]. For implementation reasons, it was not possible to implement this interaction with OpenFOAM, because adding a temperature field into the source code is not an east task [220]. On the other hand, the LMD process simulated through our own finite element C++ code allows us to properly handle a rich sensitivity analysis testing different printing conditions.

6.2.1 Simulations set up

Before describing geometry, physical aspects and boundary conditions it is necessary to specify that, in order to carry out a parametric study, several numerical simulations have been performed varying the parameters that most influence the LMD process (see for instance [221, 222, 223]). Such parameters, although not being exhaustive, are of macroscopic nature and directly employable from an engineering viewpoint. The ranges of values to be used have been obtained according to some evidences present in the literarture and are summarized in Table 6.2.
In particular, the following quantities are taken in account:

- Nozzle geometry, i.e. the inclination of the channel through which particles and gas flow is varied;
- Carrier gas flow rate imposed in the inlet, this translates into different velocity conditions, due to the concept that, for a fixed area, an increase in flow rate generates a velocity increase;
- Powder mass flow rate entering the nozzle;
- Laser power used to melt the particles.

In Section 6.2.2 will be shown the outcomes when these conditions vary. Results obtained for different combinations of parameters will be compared, together with the influence that they have on the simulated multiphysics process. Therefore, having to change the value of each of the 4 selected parameters 5 times, exactly $5^4 = 625$ simulations have been performed.

Geometry and physical settings

The 3D CAD model of the nozzle was created with the open-source software Salome, which allows the parametric generation of the geometry by writing a Python code. Taking advantage of this peculiarity it was possible to generate different geometries, by varying the angle of

Parameter	Values	Unit
Nozzle inclination	$\theta = \{15°, 20°, 25°, 30°, 35°\}$	Grad
Carrier gas flow rate	$\dot{m}_g = \{4, 6, 8, 10, 12\}$	[l/s]
Powder flow rate	$\dot{m}_p = \{0.15, 0.20, 0.25, 0.30, 0.35\}$	[g/s]
Laser power	$P = \{200, 700, 1200, 1700, 2200\}$	[W]

Table 6.2: Analysis parameters employed in the numerical simulations. For the values here reported, we refer the reader to [219, 136, 224, 225, 44].

inclination of the coaxial channel, changing only the aforementioned parameter in the Python script. In particular, 5 different geometries have been generated in which the inclination of the channel is equal to $\theta = \{15°, 20°, 25°, 30°, 35°\}$. In Figure 6.50, on the left, the geometry for an inclination angle equal to 25°, along with boundary type for each face, is shown. The coaxial

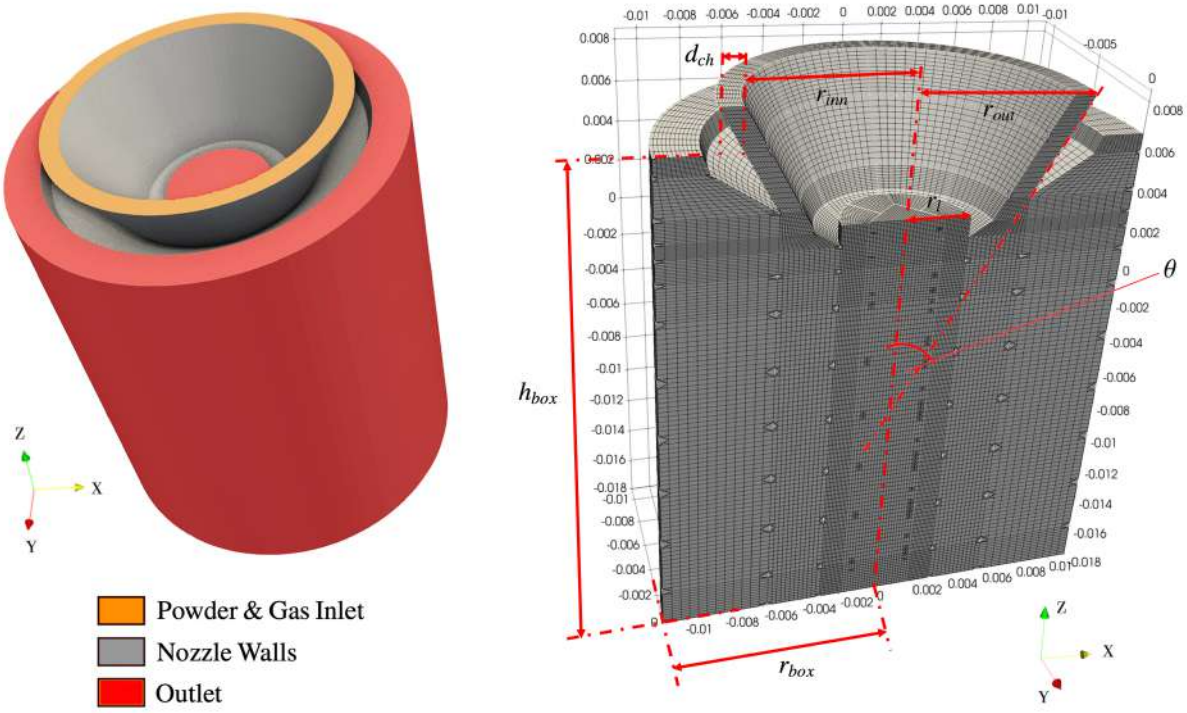

Figure 6.50: Geometry of the nozzle with boundaries (on the left) and the discretized computational domain (on the right).

nozzle has an inlet face, for both particles and carrier gas, positioned in the upper part of the geometry and having the shape of a ring. The outer radius of this ring measures $r_{out} = 8.55\,mm$ while the inner one measures $r_{inn} = 7.30\,mm$, which means that the distance that divides the inner wall from the outer one, that is the channel that allows fluid and powder passage, is equal to $d_{ch} = 1.25\,mm$. In the middle of nozzle bottom, there is an additional channel with a diameter of $d_l = 3.00\,mm$, through which the laser beam will pass. Finally, to ensure that both

the powder stream and the fluid are not affected by the boundaries of computational domain, the cylinder is set to be large enough. This part, located below the nozzle, represents the whole portion of the outlet whose height and radius are equal to $h_{box} = 24.5\,mm$ and $r_{box} = 11.0\,mm$, respectively.
The corresponding boundary conditions are summarized as follows. We fixed an inlet gas velocity, constant over time, and equal to 5 different values chosen to carry out the parametric analysis: $\boldsymbol{u} = \{1.0, 1.5, 2.0, 2.5, 3.0\}\,m/s$. Note that the dimensions of inlet cross-section area are fixed, therefore we work with the following inlet gas flow rate $\dot{g} = \{0.0031,\ 0.0046,\ 0.0062,\ 0.0081,\ 0.0093\ \}\ m^3/s$. The same velocities are also set for the initial velocity of the particles, thus assuming that they have already been transported by the carrier fluid for a previous stretch. The value of the powder flow rate, that is one of the four parameters chosen to perform the parametric analysis, is assumed to be $\dot{m}_p = \{0.15, 0.20, 0.25, 0.30, 0.35\}\,g/s$. The inlet temperature is set to $T_{in} = 350\,K$ assuming that the fluid at this stage has not yet interacted with any heat source. On the walls along the nozzle channel, a no-slip condition is imposed for the fluid, whereas the involved particles have to perform a rebound, whit the coefficients of restitution and kinetic friction equal to $e = 0.97$ and $f_r = 0.09$, respectively. The normal flux of temperature and pressure are prescribed to be zero at the wall boundaries. Finally, in the outlet contours both velocity and temperature have a zero gradient, while a value of zero is set for the pressure. Instead the particles are allowed to cross this type of boundary and exit the domain.
At the initial time the value of velocity is set to zero, the pressure has a small gradient between the inlet and the outlet, while the temperature is set equal to $T_0 = 350\,K$, as ambient temperature in the printing process [136]. Regarding the laser source, this has been modeled as an independent Eulerian field fixed in time according to Equation (4.132), thus following a Gaussian distribution in the transverse plane. The central axis of laser beam is set to be coincident with the Z-axis suggesting that the laser beam travels through the middle of the nozzle perpendicularly and then interacts with powder stream. The power of laser beam is the last parameter involved in the sensitivity analysis, equal to $P = \{200, 700, 1200, 1700, 2200\}\,W$, with an effective diameter of $D = 5.0\,mm$. Starting from the middle channel, the laser beam converges to the focal plane at $\boldsymbol{z} = -7.0\,mm$ with a half angle of 3.7°.
The adopted gas in this model is Argon, meanwhile, Stellite 6 is used as powder material . The used material properties of gas and powder are shown in Table 6.3. The distribution of the size of powder particles diameter is Gaussian, with mean and variance equal to $\mu = 50\,\mu m$ and $\sigma^2 = 0.0002$, respectively.

Discretization and numerical settings

The computational domain consist in a 3D structured quadrilateral mesh generated by Salome with 518400 cells. This is shown on the right of Figure 6.50, which includes the geometry of the nozzle. The mesh is fine enough to correctly represent the behaviour of the fluid, and there are no problems related to the particles interpolation, as happens in OpenFOAM, and therefore

Argon property	**Value**	**Unit**	**Stellite 6 property**	**Value**	**Unit**
Density ρ_g	1.603	$[kg/m^3]$	Density ρ_p	8380	$[kg/m^3]$
Specific heat $c_{p,g}$	520.6	$[J/(kg \cdot K)]$	Specific heat $c_{p,p}$	421.0	$[J/(kg \cdot K)]$
Thermal conductivity h_g	0.01580	$[W/(m \cdot K)]$	Thermal conductivity h_p	14.82	$[W/(m \cdot K)]$
Kinematic viscosity ν	1.403e-05	$[m^2/s]$	Laser absorption coefficient η_p	0.35	
Prandtl number Pr	0.66865		Latent heat L_f	2.920e+05	$[J/kg]$
Thermal diffusivity κ	2.0053e-05	$[m^2/s]$	Liquidus temperature T_{liq}	1630	$[K]$
Thermal expansion α	3.4112e-03	$[1/K]$	Solidus temperature T_{sol}	1533	$[K]$

Table 6.3: Properties for Argon and Stellite 6, simulated as materials for carrier gas and powder, respectively. [135, 136]

their behaviour is independent of the mesh size[2] .
As regards the temporal discretization, the same step size for all simulations was set equal to $dt = 0.00005\,s$. As extensively discussed in Section 5.3, using a Newton-like scheme, there is a very low error on continuity constraint, and therefore great accuracy, even considering time steps smaller than the one imposed.
The Navier Stokes equations are then solved with the modified Newton-Raphson scheme (see Algorithm 1) using a direct solver with an imposed tolerance equal to $DS_{tol} = 1e-10$, and $DS_{max,iter} = 50$ maximum iterations. The heat equation is solved with a CG method with tolerance equal to $CG_{tol} = 1e-08$, and max iterations equal to $CG_{max,iter} = 20$, the particle dynamic time discretization concerns in a classical Euler scheme.
The model employs quadrilateral and hexahedral Lagrange finite elements Q2 and Q1, yielding to polynomials of degree 2 for velocity, and degree 1 for temperature and pressure, respectively. Simulations are conducted on an off-the-shelf desktop computer with eight-cores Intel(R) Xeon(R) W-2125 running at 4.00 GHz, with 256 GB of RAM, running 64-bit Ubuntu Linux 20.04.1 LTS.

Note finally that all the operations concerning matrix assemblage matrices are done in parallel, as well as particle generation and solutions (see again Section 5.5.2). As well known, all the numerical procedures concerning particle simulation make to leaven the total computational costs as the number of particle increases. In Figure 6.51 the non-dimensional CPU times are reported against the inlet powder mass flow rates. As expected, computational times increase for increasing number of simulated particles, in particular with a rate of about 80% with respect to both powder flow rate and corresponding number of particles.

[2]In order to save computational costs, when a particle moves from a cell to an other during the time step, OpenFOAM tries to find this particle in cells adjacent to the previous one (the cell where the particle is located at the previous time step). If it cannot find the particle, since such a particle moves crossing more than one cell in the time step, the software crashes. To prevent this problem a proper ratio between mesh size and time step size must be set.

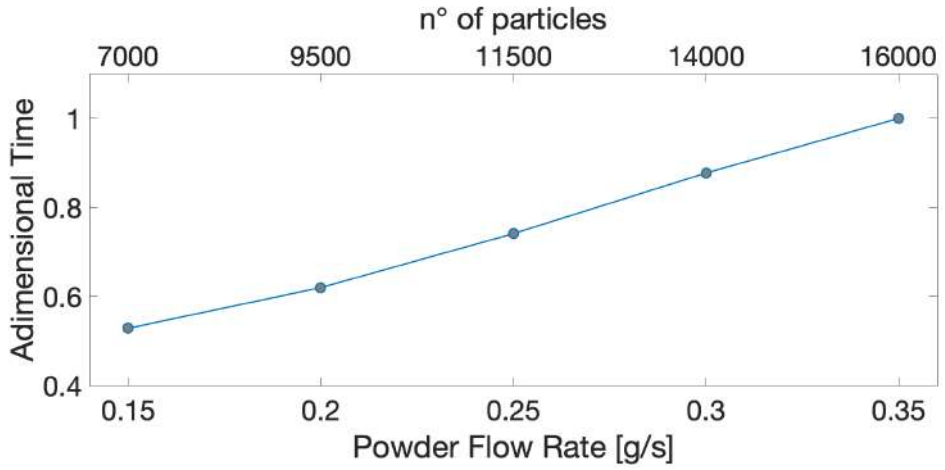

Figure 6.51: CPU times plotted versus the number of simulated particles, associated with the inlet powder mass flow rates $\dot{m}_p = \{0.15, 0.20, 0.25, 0.30, 0.35\}$ l/s. CPU times are nondimensionalized with respect to the maximum value of 9906 secs.

6.2.2 Results and discussions

The obtained results are presented and discussed in the following paragraphs. Initially, the results related to the particles distribution in the computational domain will be shown, describing the shape of the powder stream and the different configurations that are created along the vertical axis. The study of the variation of the powder concentration at the focal plane, varying the flow rate of gas and powder, respectively, concludes this first study. In the last part of this section, the interaction with the thermal field is shown and in particular the evolution of the phase change of the particles that form the powder stream is analyzed.

Distribution of powder stream

The shape of the powder stream has great impacts on its thermal profile and it significantly affects the melting and solidification processes that characterize the quality of built parts. The distribution of the particles in the computational domain is shown in Figure 6.52. In particular, the plot represents the velocity magnitude of each individual particle passing through a nozzle with a 25° inclined coaxial channel, on the left, together with particles concentration projected on the x/y plane cutting the central axis of the domain, on the right. It can be seen that the usage of a coaxial nozzle produce a powder stream with an annular pattern at the beginning of its path. Then, primarily due to the drag of gas flow and inertia, particles start to merge into a main stream to form a waist, producing a concentrated region of powder below the nozzle. Below such a region, the powder stream diverges. In order to investigate in-depth the powder stream structure, the distribution of particles concentration on various transversal X-Y planes are reported in Figure 6.53 - 6.55. In particular, according to the structure of the powder stream below the nozzle, particles concentration can be approximately categorized into three distinct stages: (i) pre-waist, (ii) waist, and (iii) post-waist, which are shown in

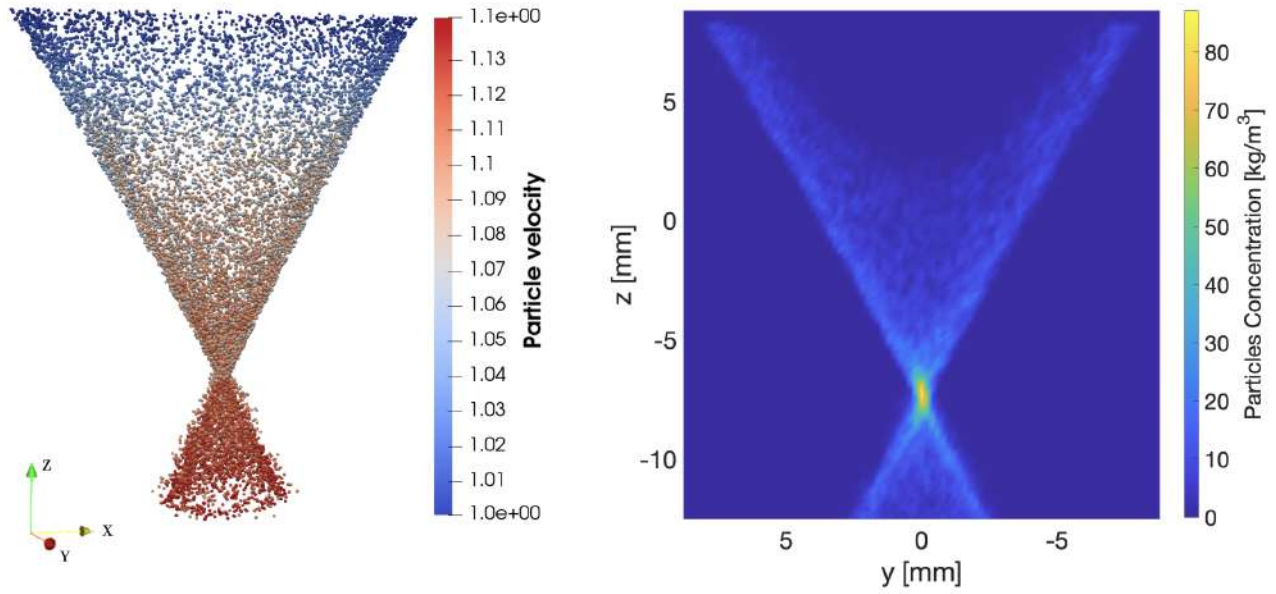

Figure 6.52: Particles velocity map for nozzle with a 25° angle of inclination of the coaxial channel, powder mass and carrier gas flow rate equal to $\dot{m}_p = 0.35\,\mathrm{g/s}$ and $\dot{m}_g = 4\,\mathrm{l/s}$, respectively. Velocity magnitude of each individual particle passing through the nozzle (on the left), particles concentration projected on the x/y plane cutting the central axis of the domain (on the right).

Figure 6.53, 6.54 and 6.55, respectively. In addition, for every stage, the values of the particles concentration at $y = 0\,mm$ are interpolated with a spline, to continuously display the trend of the powder distribution along the vertical axis. These are plotted next to the corresponding stages considered. The pre-waist profile located at $z = -4\,mm$ is shown in Figure 6.53 and

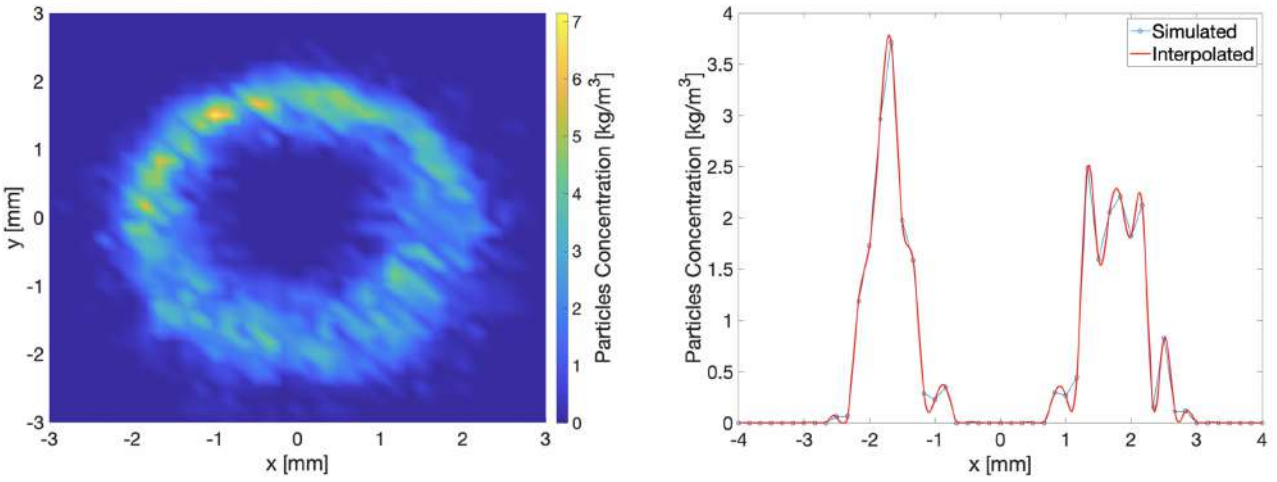

Figure 6.53: Particles concentration at pre-waist transversal X-Y plane, located at $z = -4\,\mathrm{mm}$ (on the left), and relative spline interpolation at $y = 0$ (on the right). The nozzle inclination angle is equal to $\theta = 25°$, the powder mass and carrier gas flow rate considered $\dot{m}_p = 0.35\,\mathrm{g/s}$ and $\dot{m}_g = 4\,\mathrm{l/s}$, respectively.

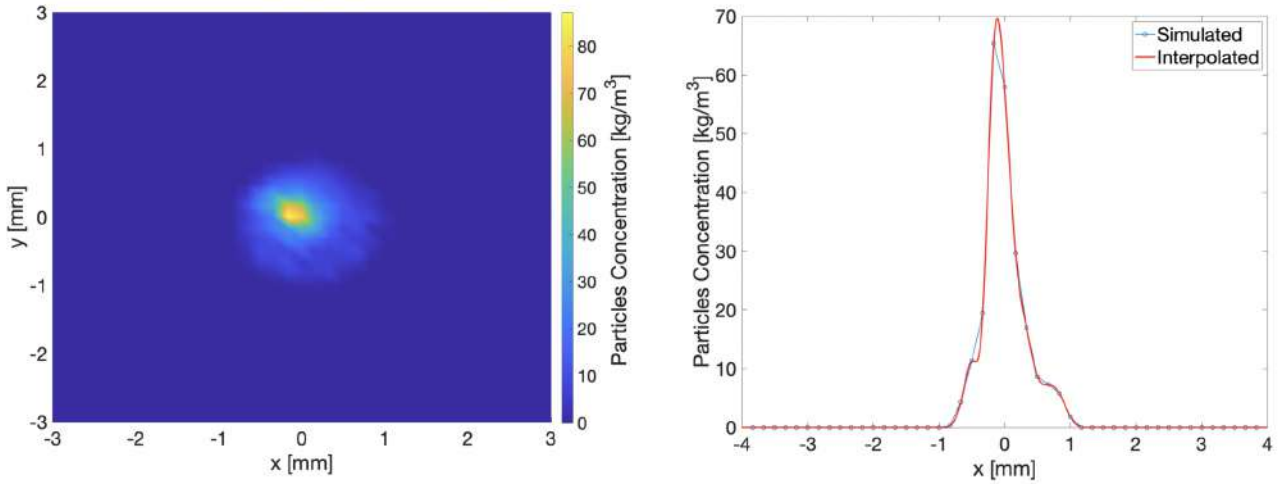

Figure 6.54: Particles concentration at waist transversal X-Y plane located at $z = -7\,\text{mm}$ (on the left), and relative spline interpolation at $y = 0$ (on the right). The nozzle inclination angle is equal to $\theta = 25°$, the powder mass and carrier gas flow rate considered $\dot{m}_p = 0.35\,\text{g/s}$ and $\dot{m}_g = 4\,\text{l/s}$, respectively.

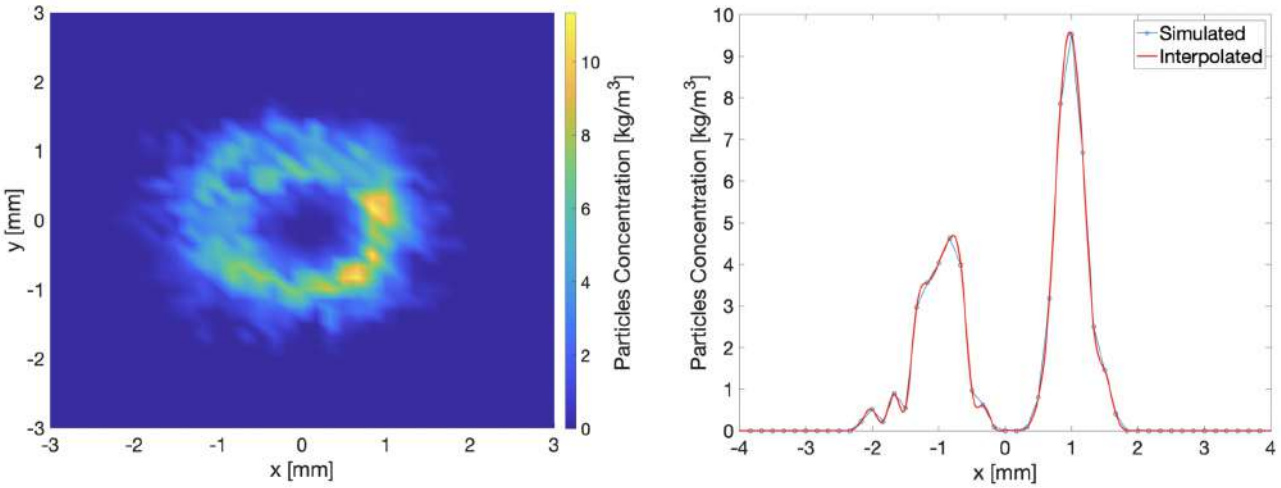

Figure 6.55: Particles concentration at post-waist transversal X-Y plane located at $z = -10\,\text{mm}$ (on the left), and relative spline interpolation at $y = 0$ (on the right). The nozzle inclination angle is equal to $\theta = 25°$, the powder mass and carrier gas flow rate considered $\dot{m}_p = 0.35\,\text{g/s}$ and $\dot{m}_g = 4\,\text{l/s}$, respectively.

it represents the typical annular powder stream structure formed at the nozzle exit, keeping such a shape up to the focal plane. Furthermore, we can observe, that the powder flux distribution has a bimodal shape at that location. Then, the powder stream converges to form the waist stage at $z = -7\,mm$, and the maximum concentration is reached with a peak values of $81.7\,kg/m^3$ and showing a typical Gaussian profile (see Figure 6.54). At $z = -10\,mm$, the waist stage disappears and the powder stream goes through the post-waist stage. It can be seen from Figure 6.55 that the maximum value of powder concentration drops really fast to around $10.6\,kg/m^3$ and the powder stream diverges.

All these considerations have been done for fixed values of channel inclination angle, and powder and carrier gas flow rate. In order to evaluate particles concentration for different configurations, in Figure 6.56 are shown the outcomes of several set up simulated varying the above mentioned parameters (see Section 6.2). In particular, the plot represents particles concentrations at focal planes for different carrier gas inlet velocities and varying the inlet of powder mass flow rate. The amount of particle concentration decreases, obviously as the powder mass

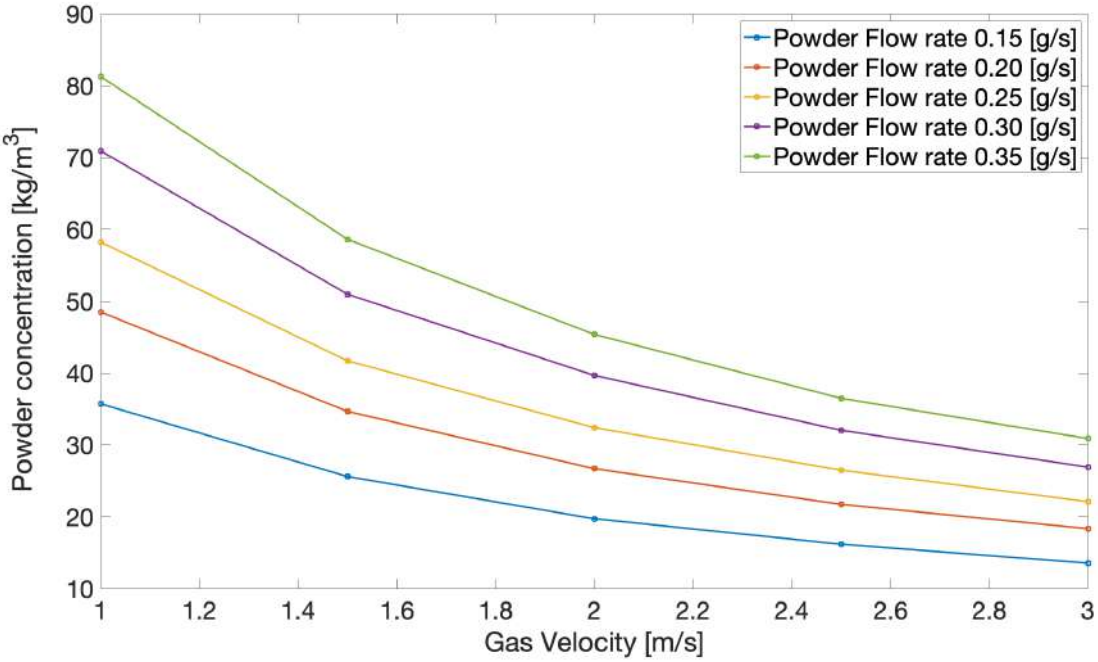

Figure 6.56: Particles concentrations values at focal planes for a nozzle with 25° inclination angle of the coaxial channel and for different carrier gas inlet velocities and powder mass flow rate.

flow rate decreases, but also by increasing the inlet velocity of the carrier gas, because a more diluted flow takes place. In fact, considering a powder flow rate of $0.35\,g/s$ the concentration peaks tend to decrease starting from a maximum value of $81.7\,kg/m^3$ for a carrier gas velocity of $\boldsymbol{u}_g = 1\,m/s$ up to a value of $30.8\,kg/m^3$ when the carrier gas velocity is equal to $\boldsymbol{u}_g = 3\,m/s$. Note finally that, in the perspective of designing optimal conditions for the LMD process, high values of particle concentration can be attained employing low amounts of both powder and carrier gas: a powder concentration of $35.7\,kg/m^3$ is reached in the simulations with powder and carrier gas flow rates $\dot{m}_p = 0.15\,g/s$ and to $\boldsymbol{u}_g = 1\,m/s$, whereas we obtain $30.8\,kg/m^3$ for $\dot{m}_p = 0.35\,g/s$ and $\boldsymbol{u}_g = 3\,m/s$. This first outcomes could be of great help in the set up phase of the printing process as it helps to understand which parameters handles to reach the desired powder configuration.

Thermal profile of powder stream

On the basis of the intensity profile of the laser beam, we predict the temperature profile of powder stream as depicted in Figure 6.57, where both laser intensity and particles temperature are reported. Starting from the nozzle exit, powder particles keep their initial temperature,

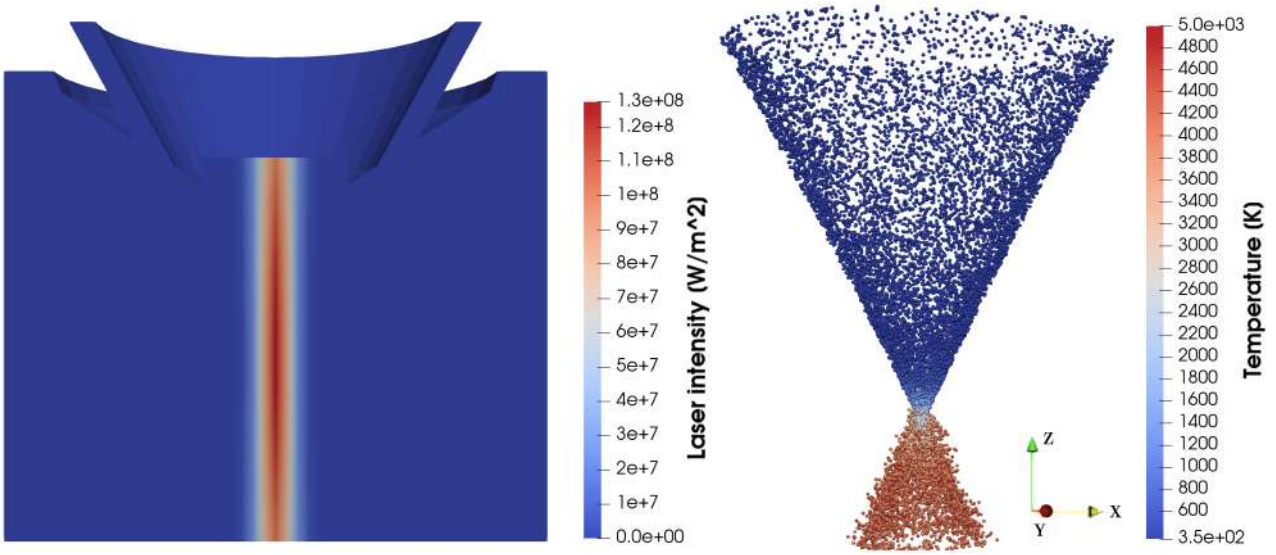

Figure 6.57: Intensity profile of a 2200 W laser beam (on the left), and particles temperature map (on the right) for a nozzle with inclination angle equal to 25°. Powder mass and carrier gas flow rate are equal to $\dot{m}_p = 0.35\,\mathrm{g/s}$ and $\dot{m}_g = 4\,\mathrm{l/s}$, respectively.

around 350 K prescribed as initial value, until they enter in the laser interaction zone. Then, approaching the focal plane location, particles are quickly heated up from laser beam and their temperature increases suddenly. Figure 6.57 shows the high gradient values that characterize the band in which the particles interact with the energy irradiated by the laser, leading to large temperature over 3000 K. Finally, after passing the focal zone, particles continue to exchange thermal energy exclusively with the external ambient.

Another relevant contribution to the heat exchange is given by the temperature field generated by the substrate. Figure 6.58 show both temperature mappings, the one on the fluid, heated through the substrate, and the one present on the particles irradiated by the laser beam (specifically, with a power of $P = 2200\,\mathrm{W}$). In order to have a more realistic view of the LMD process, the substrate is placed in correspondence with the focal plane of the particles, as it is usually set during the printing stage. The powder flux receiving energy from the laser beam, is also affected by the temperature field of the substrate. In this case the particles having a higher temperature than the melting point interact with the substrate when they are already melted. This is an optimal condition for the LMD process as all the metallic material that contributes to the formation of the track is in the liquid state. In fact, if the particles enter the substrate in a non-fused state they can remain unmelted and this does not allow the formation of a homogeneous track and the presence of non-fused particles can be found in the deposited layers [18].

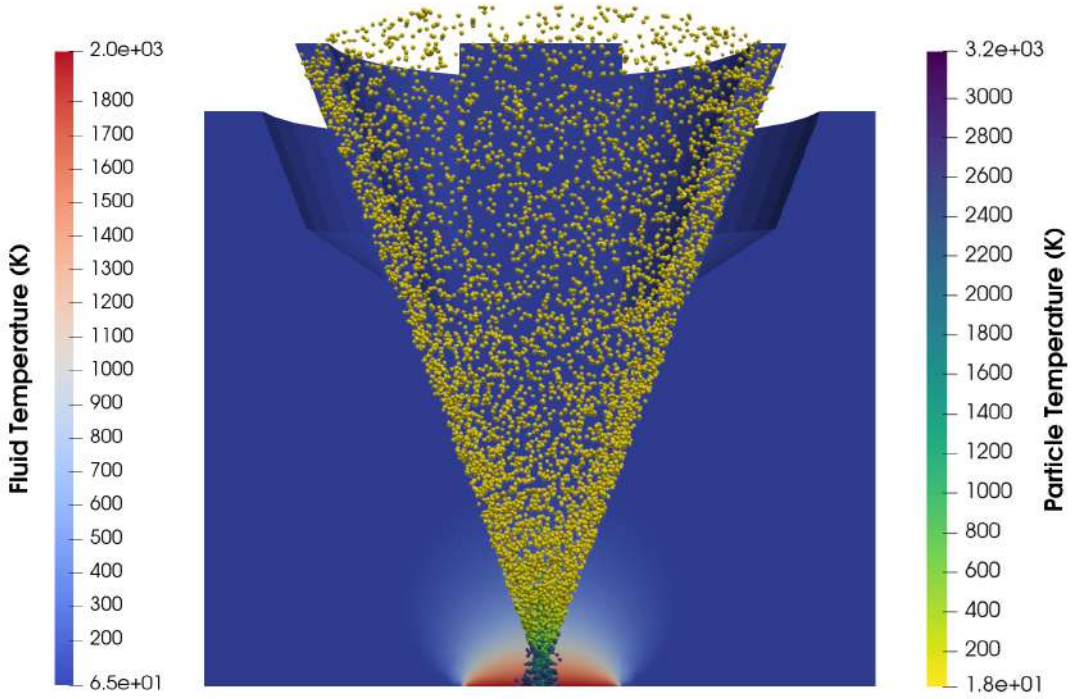

Figure 6.58: Temperature field generated by the substrate together with the temperature of the particles considering a laser power equal to $P = 2200\,\mathrm{W}$. The nozzle inclination angle and the powder mass and carrier gas flow rate considered are respectively equal to $\theta = 25°$, $\dot{m}_p = 0.35\,\mathrm{g/s}$ and $\dot{m}_g = 4\,\mathrm{l/s}$.

With the purpose of better highlighting where the particle flow may achieve the melted state, Figure 6.58 gives us reasons to neglect in the following simulations the heated substrate [226, 227]. In particular, the phase transition from the solid to the liquid state of the single particle is tracked in terms of evolution of fraction of liquid mass m_l/m_p, according to the simplified model of the liquid fraction evolution law (4.136).

Figure 6.59 reports the evolution of the m_l/m_p ratio, computed by averaging the liquid mass fraction of each particle occupying the x/y plane at a given z coordinate of the computational domain. As expected, the particles tend to melt at a slower rate as the laser power decreases, and therefore the distance from the nozzle at which the particles are completely melted increases.

The laser heating effects are also affected by the time spent by the particles in the region with high laser intensity. As Figure 6.59 shows, the value of $m_l/m_p = 1$ is reached faster according to the laser power. This behaviour is furthermore highlighted in Figure 6.60, where the liquid mass fraction of each particle is mapped at the same time step of the analysis (in steady-state regime) by comparison between the process with 200 W laser power and that with 2200 W. In the first case, the complete fusion of the particles takes place after the focal plane, while, in the second case, it happens before.

Such a behaviour is only qualitatively similar when changing the nozzle geometry. Figures 6.61 depicts several curves of the liquid mass fraction varying along the vertical axis for different inclination angles θ of the nozzle channel, and for different laser powers P, according

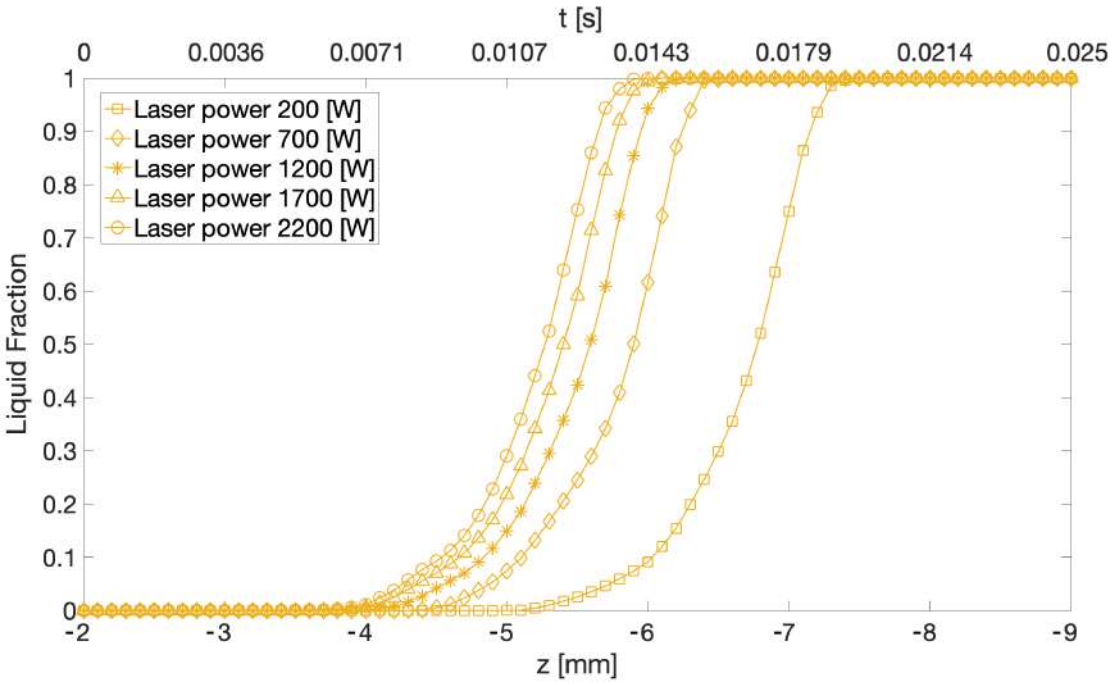

Figure 6.59: Liquid mass fraction plotted against the vertical position of the particles and increasing laser power $P = \{200, 700, 1200, 1700, 2200\}$ W. The nozzle inclination angle and the powder mass and carrier gas flow rate considered are respectively equal to $\theta = 25°$, $\dot{m}_p = 0.35\, g/s$ and $\dot{m}_g = 4\, l/s$.

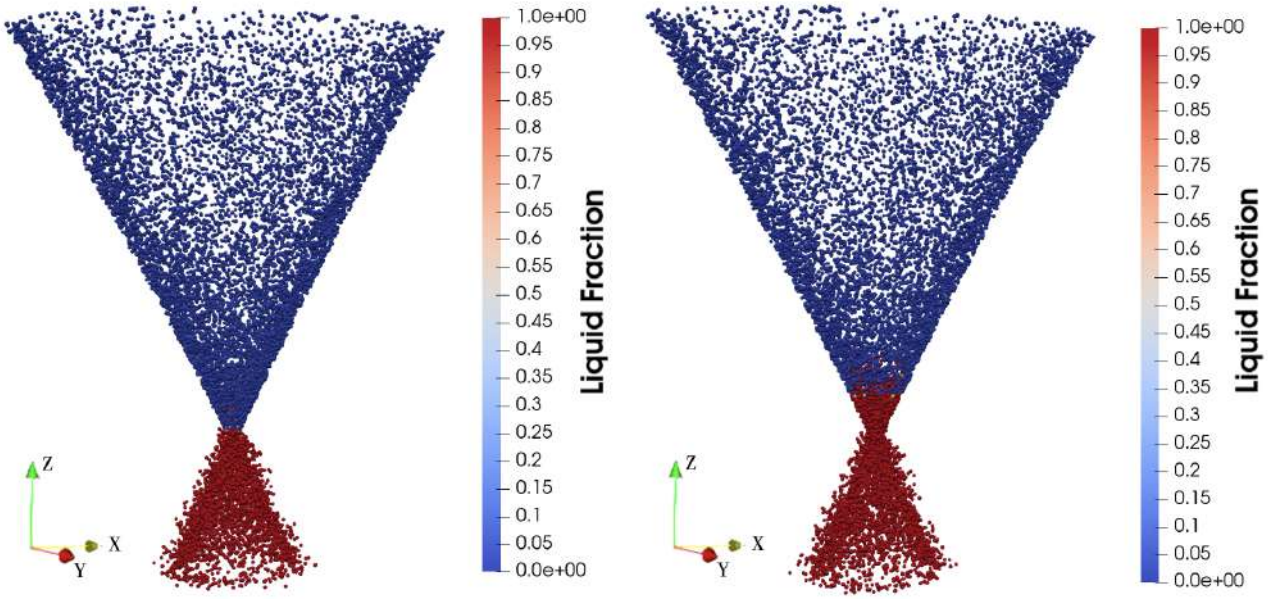

Figure 6.60: Liquid fraction distribution along the powder flow considering a laser power of 200 W (a) and 2200 W (b). The nozzle inclination angle and the powder mass and carrier gas flow rate considered are respectively equal to $\theta = 25°$, $\dot{m}_p = 0.35\, g/s$ and $\dot{m}_g = 4\, l/s$.

to the values reported in Table 6.2. For each level of P, the state of complete fusion is reached at distances from the nozzle as θ decreases, in reason of the distance between the focal plane

and the nozzle, increasing as the inclination of the coaxial channel becomes more collinear with respect to the vertical axis, *i.e.*, with decreasing θ.

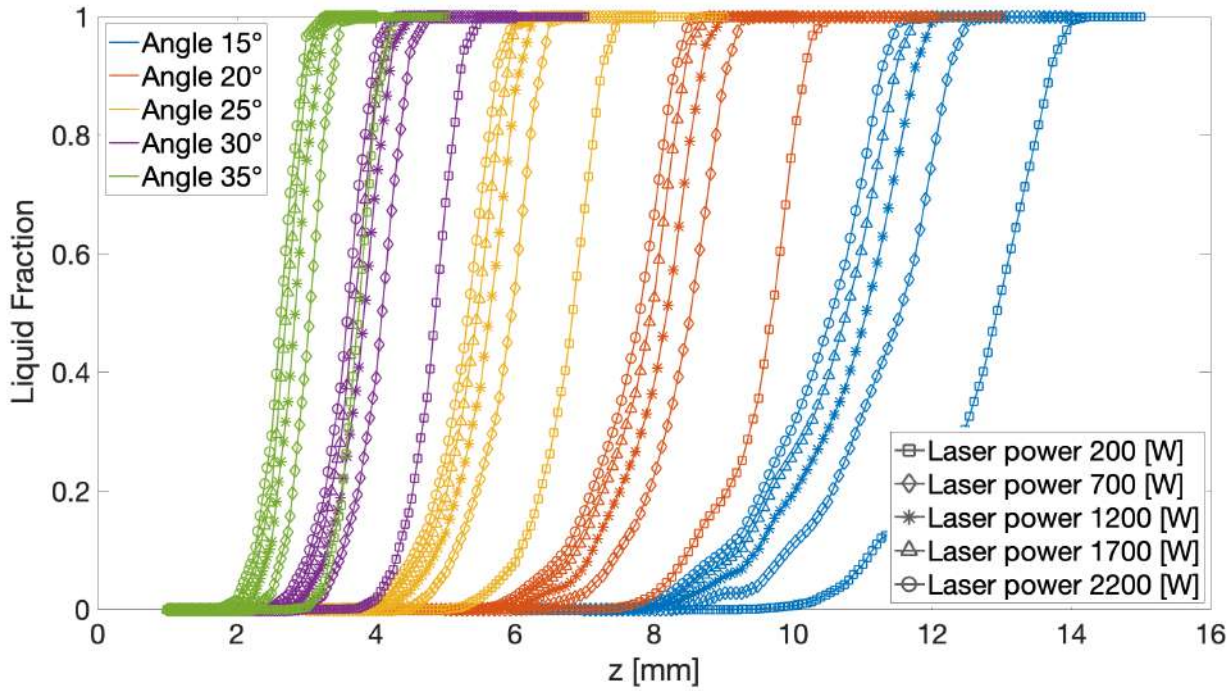

Figure 6.61: Liquid mass fraction plotted against the vertical position of the particles considering different values of inclination angle, and increasing laser power $P = \{200, 700, 1200, 1700, 2200\}$ W. Powder mass and carrier gas flow rate are equal to $\dot{m}_p = 0.35\,\mathrm{g/s}$ and $\dot{m}_g = 4\,\mathrm{l/s}$, respectively.

Moreover, increasing the laser power leads to melting particles over the focal plane, with a flattening of the liquid mass curves, evident as the nozzle inclination angle increases. For $\theta = 15°$ the point of complete melting is obtained at $z = -14.0\,mm$ when $P = 200\,\mathrm{W}$, and at $z = -11.8\,mm$ when $P = 2200\,\mathrm{W}$. On the other hand, for $\theta = 35°$ a unit liquid mass fraction is reached at $z = -4.1\,mm$ for a laser power of 200 W, at a $z = -3.0\,mm$ for 2200 W. Such a trend indicates also that a similar melting process can be obtained with inclination angles from $\theta = 30°$ to $\theta = 35°$, significantly saving costs on the laser power, reduced from 1200–1700 W to 200 W.

The previous considerations are finally emphasized through Figure 6.62, where the percentage of liquid mass fraction reached at the focal plane, associated with different inclination angle, is plotted versus the laser power. Such a percentage increases non linearly with the laser power, highlighting a sort of plateau for high laser powers, reached sooner as the inclination angle decreases. Considering high laser powers (1200, 1700, and 2200 W), values at least of 85% of the liquid mass fraction are reached at the focal plane for any inclination angles $\theta \in [15, 35]$ degrees. On the other hand, by increasing θ, a reduction of liquid mass fraction is observed for any value of laser power, three percentage points for $P = 2200\,\mathrm{W}$, and about seven points for the other values of P. However, by accepting suboptimal percentages of molten material at the

focal point, the laser power can be significantly diminished by suitably decreasing the inclination angle: about 97% of liquid fraction can be obtained either by laser powers $P = 2200\,\mathrm{W}$ with $\theta = 35°$, or by laser powers $P = 1200\,\mathrm{W}$ with $\theta = 15°$.

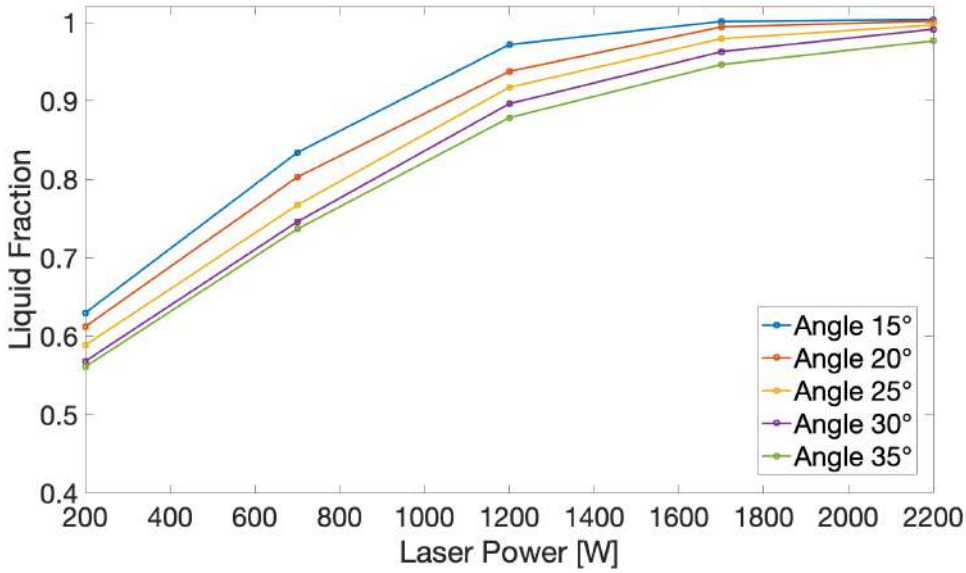

Figure 6.62: Liquid fraction values at the focal points considering different nozzle inclination and varying the laser power employed. Powder mass and carrier gas flow rate are equal to $\dot{m}_p = 0.35\,g/s$ and $\dot{m}_g = 4\,l/s$, respectively.

Conclusions

In this thesis, several 3D numerical strategies that model the powder stream exiting from a coaxial nozzle has been developed and analyzed.
The dynamic of the particles flow in the LMD process was first investigated using the open source software OpenFOAM: two different approaches, namely Eulerian-Eulerian (EE) and Lagrangian-Eulerian (LE), was developed and then implemented.

The numerical campaign, all referred to a three-dimensional problem, was addressed to highlight performance of the two approaches, measuring the geometrical key features of the powder shape cone, outside the deposition head. Among the obtained results, the EE method proved to be not able to capture crossing trajectories, carrying out a merged powder flux exiting from the nozzle as a single high-density stream. On the contrary, the LE method was able to predict the experimented behavior, where the total flux divaricates in three streams after particles cross each other in the focal point. However, the EE approach, as well as LE's, reproduced the solid volume fraction amount around the focal point, which is definitely one of the key information in the set up of the LMD printing process. Additionally, the efficiency of the two approaches was investigated in terms of CPU times, showing that the LE simulation turns out to be more expensive than EE's, as the number of simulated particles increases.

A preliminary validation with experiments conducted by PoliMi showed, although in a qualitative sense, the capability of the developed numerical schemes on predicting measured decreasing trends of the cone diameter for increasing injected gas velocity.
As widely explained, the script code developed in OpenFOAM allowed us to simulate only the flow of particles that interacts with the velocity field of the fluid, without including a thermal coupling. In order to model the energy problem that describes the interaction between powder flow and the laser beam, the open source Finite Element library deal.II was then used to implement an in-house C++ code: commercial software are less versatile performing numerical campaigns, especially if referred to parameters sensitivity analysis in order to optimize the process. Moreover, as to the author's knowledge, commercial codes able to simulate the entire physical LMD process have not been implemented yet, as this AM technology is being developed mainly as a research tool, and not a commercial product. In recent years many software companies have been implementing ad-hoc codes for the additive world, but still nothing has been released for the LMD process.
For this purpose, an exhaustive in-house code capable of representing the LMD manufacturing process in reasonable times is implemented, with the possibility to control both physical and

numerical parameters.

In particular, the heat equations are coupled with the Navier-Stokes equations in an Eulerian description, whereas the thermal evolution of the particle is also included in the Lagrangian description. In particular, to handle the Eulerian coupled problem we proposed a modified Newton-Raphson (mNR) scheme, embedded in a time-marching algorithm, that solves both accurately and efficiently advection-diffusion problems, coupling incompressible fluid motions with temperature heat exchange. Such an approach handles the fully nonlinear formulation of the Navier-Stokes equations, without introducing any linearization as usually done in stabilized methods. As a consequence, for each time step of the analysis, iteration loops are needed in the numerical strategy to get convergence; the ensuing computational cost, in a single time step higher than those working with linearized equations, can be drastically reduced by increasing the size of the time step. After a successful validation through available numerical results present in the literature, the proposed algorithm has been compared with two linearization approaches coming from well consolidated numerical formulations, namely the projection-correction scheme (PC) and the iterated projection-scheme (iPC). Such approaches, recently improved only for solving Navier-Stokes equations, were ad-hoc implemented for the present coupled problem. While the efficiency has been examined in terms of CPU times, saving in some cases an 80% of overall costs, the accuracy has been controlled in terms of divergence-velocity norm and of Nusselt number. Especially the former, which ensures in average the continuity condition of Navier-Stokes fluids, showed that the mNR scheme is able to gain several orders of magnitude of accuracy, depending on the cost level accepted for the whole computation. Some of the reported results highlighted that mNR fares almost five orders of magnitude of accuracy better than other schemes, while paying almost the same order of CPU time employed by PC and iPC.

As regard the characterization of the particles that form the powder flow exiting form the nozzle, a Lagrangian method is chosen to describe the dynamics of the single particle. The equation of momentum balance was first implemented for solving the particle-fluid interaction, so as to capture the powder flux configuration at each time step. The resulting code, fully parallelized in order to furthermore save computational costs, was validated in comparison with an OpenFOAM simulation of the same problem, showing a good agreement in terms of particle velocity profile. Then, the equations that describe the balance of the forces exerted by the fluid on the single particle were coupled with the heat exchange contributions. Such equations model the laser energy absorbed by the particle, the heat loss by convection, and an additional condition describing in a simplified way the phase change process.

The algorithm was able to trace, apart from the position and velocity of the particles, also the thermal evolution of the powder flow and the liquid mass fraction accounting for the phase change. A sensitivity analysis of the LMD process was performed, in terms of the most meaningful physical parameters. Preliminarily, the outcomes of the distribution of the particles in the domain were discussed, analysing the configuration of powder and carrier gas flow rate and the correlation between each with other.

Then, by simulating the fully coupled problem, the obtained results showed that there is a strong influence of the geometry of the nozzle and the power of the laser beam on the amount of the molten powder, While the inflow rates do not affect the resulting liquid mass fraction, the nozzle inclination, and obviously the intensity of the laser source, remarkably influences both the distribution and the magnitude of the powder melting process.

In particular, the outcomes showed that increasing the laser power particles tend to melt before reaching the focal plane, leading upward the complete melting point. This shift is more pronounced for nozzles with small inclination angles, rather than using nozzles with large inclination. It was also noted that, strong inclinations lead to low amount of molten material in the focal plane, as lower as higher laser powers are considered. In fact, the presented numerical outcomes allowed us to compute how not only the local laser intensity but also the traveling time spent in the region with high laser intensity can determine the heating effects on the particles in terms of liquid mass percentage. The limit state of completed melted particles is reached in shorter times for higher levels of laser power. Moreover, the evolution of the particle liquid mass fraction along the vertical axis of the LMD chamber shows also that LMD systems designed with a sufficiently high nozzle inclination angle and low laser power can lead to speed up the melting process with the same efficiency of those systems with lower inclination angles for which however require higher laser powers. In the reported simulations channel inclinations increasing from 30° to 35° can admit decreasing laser powers of about 7 times.

This type of outcomes is therefore intended to pave the way to improve the whole set up of the printing process, by predicting the stand-off distance, *i.e.*, the nozzle base vs. substrate distance, necessary to reach the complete fusion of the particles. As recently pointed in recent research works, *e.g.* [103], such an information can lead to considerably decrease the thermal gradients that characterize the cooling of the molten pool, and also to accelerate the deposition process of the metallic material, so as to definitely optimize the whole printing process in terms of times and costs.

Moreover, it must be specified that such completeness of the results, relative to the physical description of the LMD process, was not possible to obtain with OpenFOAM, as an Eulerian thermal field coupled with a Lagrangian method is not implemented in this software. Both OpenFOAM and other commercial software are not designed to describe the process under investigation, and using them in this context could be limiting; in fact, the liquid mass fraction evolution of a particles flow irradiated by a laser beam has never been presented in the literature. In addition, commercial software are "black boxes", with limited flexibility, and even with OpenFOAM, where source code changes are allowed, it becomes very complicated when editing entire solvers.

For this reason, our requirements regarding the physical description of LMD printing process are extensive compared with what has been found in the literature. The creation of a in-house source code allows a freedom not comparable to that allowed by commercial software, at the cost of an effort in both implementation and validation stages of the algorithm.

Future improvements

The developed algorithm has been used to study the deposition process of a powder flux interacting principally with an energy source generated by a laser beam. The tool is able to predict both the thermal and the dynamic evolution of every particle that forms the powder flux, but no interactions with the melt pool have been considered.

The entrainment of particles into the molten substrate plays a key role in the LMD printing process, and therefore an area of improvement would be the simulation of the particle-melt pool interaction, using the developed model as an input. Few but interesting works on this kind of interaction are already present in the literature [88], in which the Lagrangian description of the particles loses its meaning. In fact, multiphysics aspects regarding the free surface forces, such as the recoil pressure generated from evaporation in conjunction with thermo-capillary forces, cannot be captured by a typical Lagrangian tracking method.

Hence, it would be necessary to implement a novel fully Eulerian formulation able to represent a real fusion of two materials, and the consequent solidification. In this context our tool could accurately provide both boundary and initial conditions for this new implementation.

Source code availability

The deal.ii source code implemented to perform the numerical simulations of the Laser Metal Deposition process is available online at the following link `https://doi.org/10.5281/zenodo.5888174`

Acknowledgements

This work was supported by Regione Lombardia through the project "MADE4LO – Metal ADditivE for LOmbardy" (No. 240963) within the POR FESR 2014-2020 program.

Bibliography

[1] B. Rylands, T. Böhme, R. Gorkin, J. Fan, and T. Birtchnell, "The adoption process and impact of additive manufacturing on manufacturing systems," *Journal of manufacturing technology management*, 2016.

[2] N. Gupta, C. Weber, and S. Newsome, "Additive manufacturing: status and opportunities," *Science and Technology Policy Institute, Washington*, 2012.

[3] J. Li, B. Wu, and C. Myant, "The current landscape for additive manufacturing research," *Imperial College Additive Manufacturing Network*, 2016.

[4] "Regione Lombardia." `https://www.regione.lombardia.it`.

[5] "MADE4LO Project." `https://www.openinnovation.regione.lombardia.it/it/b/7189/progetto-made4lo`.

[6] "AFIL (Associazione Fabbrica Intelligente Lombarda)." `https://www.afil.it/`.

[7] T. Stock and G. Seliger, "Opportunities of sustainable manufacturing in industry 4.0," *Procedia Cirp*, vol. 40, pp. 536–541, 2016.

[8] E. Oztemel and S. Gursev, "Literature review of Industry 4.0 and related technologies," *Journal of Intelligent Manufacturing*, vol. 31, no. 1, pp. 127–182, 2020.

[9] M. J. O'Brien, "Development and qualification of additively manufactured parts for space," *Optical Engineering*, vol. 58, no. 1, p. 010801, 2019.

[10] F. Auricchio, E. Bonetti, M. Carraturo, D. Hömberg, A. Reali, and E. Rocca, "Structural multiscale topology optimization with stress constraint for additive manufacturing," *arXiv preprint arXiv:1907.06355*, 2019.

[11] A. Cattenone, S. Morganti, G. Alaimo, and F. Auricchio, "Finite element analysis of additive manufacturing based on fused deposition modeling: distortions prediction and comparison with experimental data," *Journal of Manufacturing Science and Engineering*, vol. 141, no. 1, 2019.

[12] G. Alaimo, S. Marconi, L. Costato, and F. Auricchio, "Influence of meso-structure and chemical composition on FDM 3D-printed parts," *Composites Part B: Engineering*, vol. 113, pp. 371–380, 2017.

[13] "3D@UniPv project." http://www-4.unipv.it/3d/.

[14] F. Auricchio and S. Marconi, "3D printing: clinical applications in orthopaedics and traumatology," *EFORT open reviews*, vol. 1, no. 5, pp. 121–127, 2016.

[15] A. Pietrabissa, S. Marconi, E. Negrello, V. Mauri, A. Peri, L. Pugliese, E. M. Marone, and F. Auricchio, "An overview on 3D printing for abdominal surgery," *Surgical endoscopy*, pp. 1–13, 2019.

[16] N. Inverardi, S. Pandini, F. Bignotti, G. Scalet, S. Marconi, and F. Auricchio, "Temperature-memory effect in 3D printed photopolymers with broad glass transition," in *AIP Conference Proceedings*, p. 020146, AIP Publishing LLC, 2018.

[17] "The Second International Conference on Simulation for Additive Manufacturing (Sim-AM 2019)." https://congress.cimne.com/sim-am2019/frontal/default.asp.

[18] B. S. I. Gibson, D.W. Rosen, *Additive Manufacturing Technologies*. Springer, 2010.

[19] T. DebRoy, H. Wei, J. Zuback, T. Mukherjee, J. Elmer, J. Milewski, A. M. Beese, A. Wilson-Heid, A. De, and W. Zhang, "Additive manufacturing of metallic components–process, structure and properties," *Progress in Materials Science*, vol. 92, pp. 112–224, 2018.

[20] S. Ford and M. Despeisse, "Additive manufacturing and sustainability: an exploratory study of the advantages and challenges," *Journal of Cleaner Production*, pp. 1573–1587, 2016.

[21] K. S. Prakash, T. Nancharaih, and V. S. Rao, "Additive Manufacturing Techniques in Manufacturing -An Overview," *Materials Today: Proceedings*, pp. 3873–3882, 2018.

[22] J. Holmström, M. Holweg, S. H. Khajavi, and J. Partanen, "The direct digital manufacturing (r) evolution: definition of a research agenda," *Operations Management Research*, pp. 1–10, 2016.

[23] M. Brettel, M. Klein, and N. Friederichsen, "The relevance of manufacturing flexibility in the context of Industrie 4.0," *Procedia Cirp*, vol. 41, pp. 105–110, 2016.

[24] J. Gardan, "Additive manufacturing technologies: state of the art and trends," *International Journal of Production Research*, pp. 3118–3132, 2016.

[25] D. Thomas, "Costs, benefits, and adoption of additive manufacturing: a supply chain perspective," *The International Journal of Advanced Manufacturing Technology*, vol. 85, no. 5-8, pp. 1857–1876, 2016.

[26] G. Costabile, M. Fera, F. Fruggiero, A. Lambiase, and D. Pham, "Cost models of additive manufacturing: A literature review," *International Journal of Industrial Engineering Computations*, vol. 8, no. 2, pp. 263–283, 2017.

[27] U. M. Dilberoglu, B. Gharehpapagh, U. Yaman, and M. Dolen, "The Role of Additive Manufacturing in the Era of Industry 4.0," *Procedia Manufacturing*, pp. 545–554, 2017.

[28] M. Baumers, P. Dickens, C. Tuck, and R. Hague, "The cost of additive manufacturing: machine productivity, economies of scale and technology-push," *Technological forecasting and social change*, vol. 102, pp. 193–201, 2016.

[29] M. Attaran, "The rise of 3-D printing: The advantages of additive manufacturing over traditional manufacturing," *Business Horizons*, vol. 60, no. 5, pp. 677–688, 2017.

[30] A. Emelogu, M. Marufuzzaman, S. M. Thompson, N. Shamsaei, and L. Bian, "Additive manufacturing of biomedical implants: A feasibility assessment via supply-chain cost analysis," *Additive Manufacturing*, vol. 11, pp. 97–113, 2016.

[31] R. Russell, D. Wells, J. Waller, B. Poorganji, E. Ott, T. Nakagawa, H. Sandoval, N. Shamsaei, and M. Seifi, *Qualification and certification of metal additive manufactured hardware for aerospace applications*. Elsevier, 2019.

[32] R. Galante, C. G. Figueiredo-Pina, and A. P. Serro, "Additive manufacturing of ceramics for dental applications: A review," *Dental Materials*, 2019.

[33] M. K. Thompson, G. Moroni, T. Vaneker, G. Fadel, R. I. Campbell, I. Gibson, A. Bernard, J. Schulz, P. Graf, B. Ahuja, *et al.*, "Design for Additive Manufacturing: Trends, opportunities, considerations, and constraints," *CIRP annals*, vol. 65, no. 2, pp. 737–760, 2016.

[34] D. Böckin and A.-M. Tillman, "Environmental assessment of additive manufacturing in the automotive industry," *Journal of cleaner production*, vol. 226, pp. 977–987, 2019.

[35] A. Bandyopadhyay and B. Heer, "Additive manufacturing of multi-material structures," *Materials Science and Engineering: R: Reports*, vol. 129, pp. 1–16, 2018.

[36] R. Singh, R. Kumar, I. Farina, F. Colangelo, L. Feo, and F. Fraternali, "Multi-material additive manufacturing of sustainable innovative materials and structures," *Polymers*, vol. 11, no. 1, p. 62, 2019.

[37] C. L. A. Leung, S. Marussi, R. C. Atwood, M. Towrie, P. J. Withers, and P. D. Lee, "In situ X-ray imaging of defect and molten pool dynamics in laser additive manufacturing," *Nature communications*, vol. 9, no. 1, pp. 1–9, 2018.

[38] P. Stavropoulos, P. Foteinopoulos, A. Papacharalampopoulos, and H. Bikas, "Addressing the challenges for the industrial application of additive manufacturing: Towards a hybrid solution," *International Journal of Lightweight Materials and Manufacture*, vol. 1, no. 3, pp. 157–168, 2018.

[39] A. Dass and A. Moridi, "State of the art in directed energy deposition: From additive manufacturing to materials design," *Coatings*, vol. 9, no. 7, p. 418, 2019.

[40] T. Schopphoven, A. Gasser, and G. Backes, "EHLA: Extreme High-Speed Laser Material Deposition: Economical and effective protection against corrosion and wear," *Laser Technik Journal*, vol. 14, no. 4, pp. 26–29, 2017.

[41] A. Gökhan Demir, V. Furlan, N. Lecis, and B. Previtali, "Laser surface structuring of AZ31 Mg alloy for controlled wettability," *Biointerphases*, vol. 9, no. 2, p. 029009, 2014.

[42] E. Ferreira, M. Dal, C. Colin, G. Marion, C. Gorny, D. Courapied, J. Guy, and P. Peyre, "Experimental and Numerical Analysis of Gas/Powder Flow for Different LMD Nozzles," *Metals*, vol. 10, no. 5, p. 667, 2020.

[43] I. Tabernero, A. Lamikiz, E. Ukar, L. L. de Lacalle, C. Angulo, and G. Urbikain, "Numerical simulation and experimental validation of powder flux distribution in coaxial laser cladding," *Journal of Materials Processing Technology*, vol. 210, no. 15, pp. 2125–2134, 2010.

[44] B. Zhang and C. Coddet, "Numerical study on the effect of pressure and nozzle dimension on particle distribution and velocity in laser cladding under vacuum base on CFD," *Journal of Manufacturing Processes*, vol. 23, pp. 54–60, 2016.

[45] R. K. S. Liu, Y. Zhang, "Numerical Simulation and Experimental Study of Powder Flow Distribution in High Power Direct Diode Laser Cladding Process," *Lasers in Manufacturing and Materials Processing*, vol. 2, no. 4, pp. 199–218, 2015.

[46] Q. Zeng, Y. Tian, Z. XUa, and Y. QINa, "Numerical modelling of the gas-powder flow during the laser metal deposition for additive manufacturing," in *Advances in Manufacturing Technology XXXI: Proceedings of the 15th International Conference on Manufacturing Research, Incorporating the 32nd National Conference on Manufacturing Research, September 5–7, 2017, University of Greenwich, UK*, vol. 6, p. 154, IOS Press, 2017.

[47] J. Arrizubieta, I. Tabernero, J. E. Ruiz, A. Lamikiz, S. Martinez, and E. Ukar, "Continuous Coaxial Nozzle Design for LMD based on Numerical Simulation," *Physics Procedia*, vol. 56, pp. 429–438, 2014.

[48] P. Balu, P. Leggett, and R. Kovacevic, "Parametric study on a coaxial multi-material powder flow in laser-based powder deposition process," *Journal of Materials Processing Technology*, vol. 212, no. 7, pp. 1598–1610, 2012.

[49] A. Grigoryants, R. Tretyakov, I. Shiganov, and A. Stavertiy, "Optimization of the shape of nozzles for coaxial laser cladding," *Welding International*, vol. 29, no. 8, pp. 639–642, 2015.

[50] H. Pan and F. Liou, "Numerical simulation of metallic powder flow in a coaxial nozzle for the laser aided deposition process," *Journal of Materials Processing Technology*, vol. 168, no. 2, pp. 230–244, 2005.

[51] Y. D. T. Heng Pan, Todd Sparks and F. Liou, "The Investigation of Gravity-Driven Metal Powder Flow in Coaxial Nozzle for Laser-Aided Direct Metal Deposition Process," *Journal of Manufacturing Science and Engineering*, vol. 128, pp. 541–553, 2006.

[52] A. J. Pinkerton and L. Li, "Modelling powder concentration distribution from a coaxial deposition nozzle for laser-based rapid tooling," *J. Manuf. Sci. Eng.*, vol. 126, no. 1, pp. 33–41, 2004.

[53] J. Ibarra-Medina, M. Vogel, and A. J. Pinkerton, "A CFD model of laser cladding: from deposition head to melt pool dynamics," *Proc. ICALEO'2011*, pp. 23–27, 2011.

[54] C. J. Greenshields, *OpenFOAM User Guide - Version 5.0*, 2017.

[55] "deal.II — An open source finite element library." `https://www.dealii.org/`.

[56] M. Chiesa, V. Mathiesen, J. A. Melheim, and B. Halvorsen, "Numerical simulation of particulate flow by the Eulerian–Lagrangian and the Eulerian–Eulerian approach with application to a fluidized bed," *Computers & chemical engineering*, vol. 29, no. 2, pp. 291–304, 2005.

[57] P. M. Gresho, R. L. Lee, S. T. Chan, and R. L. Sani, "Solution of the time-dependent incompressible Navier-Stokes and Boussinesq equations using the Galerkin finite element method," in *Approximation Methods for Navier-Stokes Problems*, pp. 203–222, Springer, 1980.

[58] J. L. Guermond, P. Minev, and J. Shen, "An overview of projection methods for incompressible flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 195, no. 44, pp. 6011–6045, 2006.

[59] J. Aoussou, J. Lin, and P. F. Lermusiaux, "Iterated pressure-correction projection methods for the unsteady incompressible Navier–Stokes equations," *Journal of Computational Physics*, vol. 373, pp. 940–974, 2018.

[60] A. N. Brooks and T. J. Hughes, "Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations," *Computer methods in applied mechanics and engineering*, vol. 32, no. 1-3, pp. 199–259, 1982.

[61] C.-Y. Liu and J. Lin, "Thermal processes of a powder particle in coaxial laser cladding," *Optics & Laser Technology*, vol. 35, no. 2, pp. 81–86, 2003.

[62] Q. Zeng, Z. Xu, Y. Tian, and Y. Qin, "Advancement in additive manufacturing & numerical modelling considerations of direct energy deposition process," in *Proceeding of the 14th International Conference on Manufacturing Research* (Y. M. Goy and K. Case, eds.), ICMR, pp. 104–109, Amsterdam: IOS Press, September 2016.

[63] D. Ravoori, C. Lowery, H. Prajapati, and A. Jain, "Experimental and theoretical investigation of heat transfer in platform bed during polymer extrusion based additive manufacturing," *Polymer Testing*, vol. 73, pp. 439–446, 2019.

[64] M. Revilla-León, J. L. Sánchez-Rubio, J. Oteo-Calatayud, and M. Özcan, "Impression technique for a complete-arch prosthesis with multiple implants using additive manufacturing technologies," *The Journal of prosthetic dentistry*, vol. 117, no. 6, pp. 714–720, 2017.

[65] D.-S. Shim, G.-Y. Baek, and E.-M. Lee, "Effect of substrate preheating by induction heater on direct energy deposition of AISI M4 powder," *Materials Science and Engineering: A*, vol. 682, pp. 550–562, 2017.

[66] W. J. Oh, W. J. Lee, M. S. Kim, J. B. Jeon, and D. S. Shim, "Repairing additive-manufactured 316L stainless steel using direct energy deposition," *Optics & Laser Technology*, vol. 117, pp. 6–17, 2019.

[67] T. G. Spears and S. A. Gold, "In-process sensing in selective laser melting (SLM) additive manufacturing," *Integrating Materials and Manufacturing Innovation*, vol. 5, no. 1, pp. 16–40, 2016.

[68] A. Cattenone, S. Morganti, and F. Auricchio, "Basis of the Lattice Boltzmann Method for Additive Manufacturing," *Archives of Computational Methods in Engineering*, pp. 1–25, 2019.

[69] J. Ibarra-Medina, "Development and application of a CFD model of laser metal deposition," *Ph.D. thesis*, 2013.

[70] D.-S. Shim, G.-Y. Baek, J.-S. Seo, G.-Y. Shin, K.-P. Kim, and K.-Y. Lee, "Effect of layer thickness setting on deposition characteristics in direct energy deposition (DED) process," *Optics & Laser Technology*, vol. 86, pp. 69–78, 2016.

[71] J. Arrizubieta, J. Ruiz, S. Martinez, E. Ukar, and A. Lamikiz, "Intelligent nozzle design for the Laser Metal Deposition process in the Industry 4.0," *Procedia Manufacturing*, vol. 13, pp. 1237–1244, 2017.

[72] F. Brueckner, M. Riede, M. Müller, F. Marquardt, R. Willner, A. Seidel, E. Lopéz, C. Leyens, and E. Beyer, "Enhanced manufacturing possibilities using multi-materials in laser metal deposition," *Journal of Laser Applications*, 2018.

[73] R. M. Mahamood, *Laser metal deposition process of metals, alloys, and composite materials*. Springer, 2018.

[74] R. M. Mahamood and E. T. Akinlabi, "Effect of scanning speed and gas flow rate on surface roughness of LMD titanium-alloy," in *Proceedings of the World Congress on Engineering and Computer Science*, vol. 2, 2016.

[75] R. Mahamood and E. T. Akinlabi, "Laser metal deposition of functionally graded Ti6Al4V/TiC," *Materials & Design*, vol. 84, pp. 402–410, 2015.

[76] T. Petrat, B. Graf, A. Gumenyuk, and M. Rethmeier, "Laser metal deposition as repair technology for a gas turbine burner made of inconel 718," *Physics Procedia*, vol. 83, pp. 761–768, 2016.

[77] T. Yamada, S. Hasegawa, Y. Kinoshita, S. Yamada, M. Inoue, C. Rosebrock, and S. Bracke, "Process integration concept for waste reduction among manufacturing planning, modularization and validation," *Procedia manufacturing*, pp. 337–344, 2018.

[78] J. Ibarra-Medina and A. J. Pinkerton, "Numerical investigation of powder heating in coaxial laser metal deposition," *Surface engineering*, vol. 27, no. 10, pp. 754–761, 2011.

[79] A. Pinkerton, "16 - Laser direct metal deposition: theory and applications in manufacturing and maintenance," in *Advances in Laser Materials Processing* (J. Lawrence, J. Pou, D. Low, and E. Toyserkani, eds.), Woodhead Publishing Series in Welding and Other Joining Technologies, pp. 461–491, Woodhead Publishing, 2010.

[80] J. I. A. Jose Exequiel Ruiz, Magdalena Cortina and A. Lamikiz, "Study of the Influence of Shielding Gases on Laser Metal Deposition of Inconel 718 Superalloy," *Materials*, vol. 11, no. 8, p. 1388, 2018.

[81] A. L. M. Fera, F. Fruggiero and R. Macchiaroli, "State of the art of additive manufacturing: Review for tolerances, mechanical resistance and production costs," *Cogent Engineering*, pp. 1–16, 2016.

[82] F. Mazzucato, S. Tusacciu, M. Lai, S. Biamino, M. Lombardi, and A. Valente, "Monitoring approach to evaluate the performances of a new deposition nozzle solution for DED systems," *Technologies*, vol. 5, no. 2, p. 29, 2017.

[83] N. Tamanna, R. Crouch, and S. Naher, "Progress in numerical simulation of the laser cladding process," *Optics and Lasers in Engineering*, vol. 122, pp. 151–163, 2019.

[84] J. J. Nijdam, B. Guo, D. F. Fletcher, and T. A. Langrish, "Lagrangian and Eulerian models for simulating turbulent dispersion and coalescence of droplets within a spray," *Applied Mathematical Modelling*, vol. 30, pp. 1196–1211, 2006.

[85] R. Z. A. Vié, H. Pouransari and A. Mani, "Comparison between Lagrangian and Eulerian methods for the simulation of particle-laden flows subject to radiative heating," *Center for Turbulence Research. Annual Research Briefs 2014*, 2014.

[86] J. L. Lee and E. W. C. Lim, "Comparisons of Eulerian-Eulerian and CFD-DEM simulations of mixing behaviors in bubbling fluidized beds," *Powder Technology*, vol. 318, pp. 193–205, 2017.

[87] C. W. Hirt and B. D. Nichols, "Volume of fluid (VOF) method for the dynamics of free boundaries," *Journal of computational physics*, vol. 39, no. 1, pp. 201–225, 1981.

[88] S. Lin, Z. Gan, J. Yan, and G. J. Wagner, "A conservative level set method on unstructured meshes for modeling multiphase thermo-fluid flow in additive manufacturing processes," *Computer Methods in Applied Mechanics and Engineering*, 2020.

[89] S. Osher, R. Fedkiw, and K. Piechor, "Level set methods and dynamic implicit surfaces," *Appl. Mech. Rev.*, vol. 57, no. 3, pp. B15–B15, 2004.

[90] L. Han and F. Liou, "Numerical investigation of the influence of laser beam mode on melt pool," *International journal of heat and mass transfer*, vol. 47, no. 19-20, pp. 4385–4402, 2004.

[91] E. Attar and C. Körner, "Lattice Boltzmann model for thermal free surface flows with liquid–solid phase transition," *International Journal of Heat and Fluid Flow*, vol. 32, no. 1, pp. 156–163, 2011.

[92] C. Körner, A. Bauereiß, and E. Attar, "Fundamental consolidation mechanisms during selective beam melting of powders," *Modelling and Simulation in Materials Science and Engineering*, vol. 21, no. 8, p. 085011, 2013.

[93] S. A. Khairallah and A. Anderson, "Mesoscopic simulation model of selective laser melting of stainless steel powder," *Journal of Materials Processing Technology*, vol. 214, no. 11, pp. 2627–2636, 2014.

[94] W. Yan, W. Ge, Y. Qian, S. Lin, B. Zhou, W. K. Liu, F. Lin, and G. J. Wagner, "Multi-physics modeling of single/multiple-track defect mechanisms in electron beam selective melting," *Acta Materialia*, vol. 134, pp. 324–333, 2017.

[95] Z. Sun, W. Guo, and L. Li, "Numerical modelling of heat transfer, mass transport and microstructure formation in a high deposition rate laser directed energy deposition process," *Additive Manufacturing*, p. 101175, 2020.

[96] D. Roychowdhury, S. K. Das, and T. Sundararajan, "Numerical simulation of natural convective heat transfer and fluid flow around a heated cylinder inside an enclosure," *Heat and mass transfer*, vol. 38, no. 7-8, pp. 565–576, 2002.

[97] E. Salimipour, "A numerical study on the fluid flow and heat transfer from a horizontal circular cylinder under mixed convection," *International Journal of Heat and Mass Transfer*, vol. 131, pp. 365–374, 2019.

[98] J. N. Reddy, *An Introduction to Nonlinear Finite Element Analysis: with applications to heat transfer, fluid mechanics, and solid mechanics*. OUP Oxford, 2014.

[99] R. Temam, *Navier-Stokes equations: theory and numerical analysis*, vol. 343. American Mathematical Soc., 2001.

[100] J. Donea and A. Huerta, *Finite element methods for flow problems*. John Wiley & Sons, 2003.

[101] A. Sochinskii, D. Colombet, M. M. Muñoz, F. Ayela, and N. Luchier, "Flow and heat transfer around a diamond-shaped cylinder at moderate Reynolds number," *International Journal of Heat and Mass Transfer*, vol. 142, p. 118435, 2019.

[102] C. T. Crowe, J. D. Schwarzkopf, M. Sommerfeld, and Y. Tsuji, *Multiphase flows with droplets and particles*. CRC press, 2011.

[103] T. Li, L. Zhang, G. G. P. Bultel, T. Schopphoven, A. Gasser, J. H. Schleifenbaum, and R. Poprawe, "Extreme High-Speed Laser Material Deposition (EHLA) of AISI 4340 Steel," *Coatings*, vol. 9, no. 12, p. 778, 2019.

[104] G. Strang and G. J. Fix, "An analysis of the finite element method," 1973.

[105] K.-J. Bathe, *Finite element procedures*. Klaus-Jurgen Bathe, 2006.

[106] O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu, *The finite element method: its basis and fundamentals*. Elsevier, 2005.

[107] L. Corradi Dell'Acqua, *Meccanica delle strutture, Vol. 2 - Le teorie strutturali e il metodo degli elementi finiti*. McGraw-Hill, Milano, 2010.

[108] T. E. Tezduyar and Y. Osawa, "Finite element stabilization parameters computed from element matrices and vectors," *Computer Methods in Applied Mechanics and Engineering*, vol. 190, no. 3-4, pp. 411–430, 2000.

[109] R. Codina, H. Coppola-Owen, P. Nithiarasu, and C.-B. Liu, "Numerical comparison of CBS and SGS as stabilization techniques for the incompressible Navier–Stokes equations," *International Journal for Numerical Methods in Engineering*, vol. 66, no. 10, pp. 1672–1689, 2006.

[110] J. Guermond, A. Marra, and L. Quartapelle, "Subgrid stabilized projection method for 2D unsteady flows at high Reynolds numbers," *Computer Methods in Applied Mechanics and Engineering*, pp. 5857–5876, 2006.

[111] J. N. Reddy and D. K. Gartling, *The finite element method in heat transfer and fluid dynamics*. CRC press, 2010.

[112] J. Shen, "On error estimates of the penalty method for unsteady Navier–Stokes equations," *SIAM Journal on Numerical Analysis*, vol. 32, no. 2, pp. 386–403, 1995.

[113] P.-A. Raviart and J.-M. Thomas, "A mixed finite element method for 2-nd order elliptic problems," in *Mathematical aspects of finite element methods*, pp. 292–315, Springer, 1977.

[114] D. N. Arnold, F. Brezzi, and M. Fortin, "A stable finite element for the Stokes equations," *Calcolo*, vol. 21, no. 4, pp. 337–344, 1984.

[115] P. M. Gresho and R. L. Sani, "Incompressible flow and the finite element method. Volume 1: Advection-diffusion and isothermal laminar flow," 1998.

[116] D. Boffi, F. Brezzi, M. Fortin, *et al.*, *Mixed finite element methods and applications*, vol. 44. Springer, 2013.

[117] J. Guzmán, A. J. Salgado, and F.-J. Sayas, "A note on the Ladyženskaja-Babuška-Brezzi condition," *Journal of Scientific Computing*, vol. 56, no. 2, pp. 219–229, 2013.

[118] C. Taylor and P. Hood, "A numerical solution of the Navier-Stokes equations using the finite element technique," *Computers & Fluids*, vol. 1, no. 1, pp. 73–100, 1973.

[119] T. J. Hughes, L. P. Franca, and M. Balestra, "A new finite element formulation for computational fluid dynamics: V. Circumventing the Babuška-Brezzi condition: A stable Petrov-Galerkin formulation of the Stokes problem accommodating equal-order interpolations," *Computer Methods in Applied Mechanics and Engineering*, vol. 59, no. 1, pp. 85–99, 1986.

[120] T. E. Tezduyar, S. Mittal, S. Ray, and R. Shih, "Incompressible flow computations with stabilized bilinear and linear equal-order-interpolation velocity-pressure elements," *Computer Methods in Applied Mechanics and Engineering*, vol. 95, no. 2, pp. 221–242, 1992.

[121] L. P. Franca, T. J. Hughes, and R. Stenberg, *Stabilized finite element methods for the Stokes problem.* Teknillinen korkeakoulu, 1991.

[122] R. D. Cook *et al.*, *Concepts and applications of finite element analysis.* John Wiley & sons, 2007.

[123] O. C. Zienkiewicz and P. Morice, *The finite element method in engineering science*, vol. 1977. McGraw-Hill London, 1971.

[124] T. J. Hughes, *The finite element method: linear static and dynamic finite element analysis.* Courier Corporation, 2012.

[125] T. J. Hughes, "A simple scheme for developing 'upwind'finite elements," *International journal for numerical methods in engineering*, vol. 12, no. 9, pp. 1359–1365, 1978.

[126] I. Christie, D. F. Griffiths, A. R. Mitchell, and O. C. Zienkiewicz, "Finite element methods for second order differential equations with significant first derivatives," *International Journal for Numerical Methods in Engineering*, vol. 10, no. 6, pp. 1389–1396, 1976.

[127] J. Heinrich, P. Huyakorn, O. Zienkiewicz, and A. Mitchell, "An'upwind'finite element scheme for two-dimensional convective transport equation," *IJNME*, vol. 11, no. 1, pp. 131–143, 1977.

[128] T. Hughes and J. Atkinson, "A variational basis for "upwind" finite elements," in *Variational methods in the mechanics of solids*, pp. 387–391, Elsevier, 1980.

[129] K. Morton and J. Barrett, "Optimal finite element methods for diffusion-convection problems," *bilc*, pp. 134–148, 1980.

[130] T. J. Hughes, G. R. Feijóo, L. Mazzei, and J.-B. Quincy, "The variational multiscale method—a paradigm for computational mechanics," *Computer methods in applied mechanics and engineering*, vol. 166, no. 1-2, pp. 3–24, 1998.

[131] P. B. Bochev, M. D. Gunzburger, and J. N. Shadid, "Stability of the SUPG finite element method for transient advection–diffusion problems," *Computer methods in applied mechanics and engineering*, vol. 193, no. 23-26, pp. 2301–2323, 2004.

[132] V. John and E. Schmeyer, "Finite element methods for time-dependent convection–diffusion–reaction equations with small diffusion," *Computer methods in applied mechanics and engineering*, vol. 198, no. 3-4, pp. 475–494, 2008.

[133] G. Jothiprasad, D. J. Mavriplis, and D. A. Caughey, "Higher-order time integration schemes for the unsteady Navier–Stokes equations on unstructured meshes," *Journal of Computational Physics*, vol. 191, no. 2, pp. 542–566, 2003.

[134] A. Quarteroni, *Modellistica numerica per problemi differenziali*, vol. 97. Springer, 2016.

[135] S. Wen, Y. Shin, J. Murthy, and P. Sojka, "Modeling of coaxial powder flow for the laser direct deposition process," *International Journal of Heat and Mass Transfer*, vol. 52, no. 25-26, pp. 5867–5877, 2009.

[136] X. Guan and Y. F. Zhao, "Numerical modeling of coaxial powder stream in laser-powder-based Directed Energy Deposition process," *Additive Manufacturing*, p. 101226, 2020.

[137] J. Blazek, *Computational fluid dynamics: principles and applications*. Butterworth-Heinemann, 2015.

[138] H. H. Hu, "Computational fluid dynamics," in *Fluid mechanics*, pp. 421–472, Elsevier, 2012.

[139] B. E. Rapp, *Microfluidics: modeling, mechanics and mathematics*. William Andrew, 2016.

[140] J. H. Ferziger, M. Perić, and R. L. Street, *Computational methods for fluid dynamics*, vol. 3. Springer, 2002.

[141] H. William *et al.*, *Numerical recipes in C++: the art of scientific computing*. Cambridge University Press, 2002.

[142] J. H. Ferziger, *Numerical methods for engineering application*, vol. 1. Wiley New York, 1981.

[143] D. Neupane, "Comparison of some FEM codes in static analysis," Master's thesis, HAMK - University of Applied Science, Mechanical Engineering and Production Technology, 2014.

[144] F. Ansys, "12.0 Theory Guide," *Ansys Inc*, 2019.

[145] G. Hager and G. Wellein, *Introduction to high performance computing for scientists and engineers*. CRC Press, 2010.

[146] S. CD-adapco, "STAR CCM+ User Guide Version 12.04," *CD-Adapco: New York, NY, USA*, 2017.

[147] "FLOW-3D." `https://www.flow3d.com/`.

[148] R. W. Pryor, *Multiphysics modeling using COMSOL®: a first principles approach*. Jones & Bartlett Publishers, 2009.

[149] D. M. Holman, R. M. Brionnaud, and Z. Abiza, "Solution to industry benchmark problems with the lattice-Boltzmann code XFlow," in *Proceeding in the European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS)*, 2012.

[150] H. K. Versteeg and W. Malalasekera, *An introduction to computational fluid dynamics: the finite volume method*. Pearson education, 2007.

[151] "GNU General Public License." `https://www.gnu.org/licenses/gpl-3.0.html`.

[152] A. Der Kiureghian, T. Haukaas, and K. Fujimura, "Structural reliability software at the University of California, Berkeley," *structural safety*, vol. 28, no. 1-2, pp. 44–67, 2006.

[153] F. Palacios, J. Alonso, K. Duraisamy, M. Colonno, J. Hicken, A. Aranake, A. Campos, S. Copeland, T. Economon, A. Lonkar, *et al.*, "Stanford University unstructured (SU 2): An open-source integrated computational environment for multi-physics simulation and design," in *51st AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition*, p. 287, 2013.

[154] J. Kozicki and F. V. Donze, "A new open-source software developed for numerical simulations using discrete modeling methods," *Computer Methods in Applied Mechanics and Engineering*, vol. 197, no. 49-50, pp. 4429–4443, 2008.

[155] "SimFlow." `https://sim-flow.com/`.

[156] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. Gropp, *et al.*, "PETSc users manual," *Mathematics and Computer Science Division*, 2019.

[157] M. A. Heroux and J. M. Willenbring, "A new overview of the Trilinos project," *Scientific Programming*, vol. 20, no. 2, pp. 83–88, 2012.

[158] J. E. Roman, C. Campos, E. Romero, and A. Tomás, "SLEPc users manual," *D. Sistemes Informàtics i Computació Universitat Politècnica de València, Valencia, Spain, Report No. DSIC-II/24/02*, 2015.

[159] G. Guennebaud, B. Jacob, *et al.*, "Eigen," *URl: http://eigen. tuxfamily. org*, 2010.

[160] R. Chandra, L. Dagum, D. Kohr, R. Menon, D. Maydan, and J. McDonald, *Parallel programming in OpenMP*. Morgan kaufmann, 2001.

[161] J. Reinders, *Intel threading building blocks: outfitting C++ for multi-core processor parallelism*. " O'Reilly Media, Inc.", 2007.

[162] W. Gropp, W. D. Gropp, E. Lusk, A. Skjellum, and A. D. F. E. E. Lusk, *Using MPI: portable parallel programming with the message-passing interface*, vol. 1. MIT press, 1999.

[163] U. Ayachit, *The paraview guide: a parallel visualization application*. Kitware, Inc., 2015.

[164] J. Walters, "Application of finite element method in forging: An industry perspective," *Journal of Materials Processing Technology*, vol. 27, no. 1-3, pp. 43–51, 1991.

[165] S. S. Rao, *The finite element method in engineering*. Butterworth-heinemann, 2017.

[166] O. Comas, Z. A. Taylor, J. Allard, S. Ourselin, S. Cotin, and J. Passenger, "Efficient nonlinear FEM for soft tissue modelling and its GPU implementation within the open source framework SOFA," in *International Symposium on Biomedical Simulation*, pp. 28–39, Springer, 2008.

[167] N. G. Jacobsen, D. R. Fuhrman, and J. Fredsøe, "A wave generation toolbox for the open-source CFD library: OpenFoam®," *International Journal for numerical methods in fluids*, vol. 70, no. 9, pp. 1073–1088, 2012.

[168] A. Logg, K.-A. Mardal, and G. Wells, *Automated solution of differential equations by the finite element method: The FEniCS book*, vol. 84. Springer Science & Business Media, 2012.

[169] O. Pironneau, F. Hecht, A. Hyaric, and K. Ohtsuka, "FreeFEM." `http://www.freefem.org`.

[170] W. Bangerth, R. Hartmann, and G. Kanschat, "deal.II—a General Purpose Object Oriented Finite Element Library," *ACM Trans. Math. Softw.*, vol. 33, no. 4, pp. 24/1–24/27, 2007.

[171] B. Stroustrup, *The C++ programming language*. Pearson Education India, 2000.

[172] "How to add temperature to icoFoam." `https://openfoamwiki.net/index.php/How_to_add_temperature_to_icoFoam`.

[173] "The FEniCS Project." `https://fenicsproject.org/`.

[174] J. Guttag, *Introduction to computation and programming using Python: With application to understanding data*. MIT Press, 2016.

[175] M. S. Alnæs, A. Logg, K. B. Ølgaard, M. E. Rognes, and G. N. Wells, "Unified form language: A domain-specific language for weak formulations of partial differential equations," *ACM Transactions on Mathematical Software (TOMS)*, vol. 40, no. 2, pp. 1–37, 2014.

[176] K. Cheng, Y. Wang, and Q. Yang, "A semi-resolved CFD-DEM model for seepage-induced fine particle migration in gap-graded soils," *Computers and Geotechnics*, vol. 100, pp. 30–51, 2018.

[177] S. Schmelzle, E. Asylbekov, B. Radel, and H. Nirschl, "Modelling of partially wet particles in DEM simulations of a solid mixing process," *Powder Technology*, vol. 338, pp. 354–364, 2018.

[178] V. Vivacqua, A. López, R. Hammond, and M. Ghadiri, "DEM analysis of the effect of particle shape, cohesion and strain rate on powder rheometry," *Powder Technology*, vol. 342, pp. 653–663, 2019.

[179] H. Jasak, "OpenFOAM: open source CFD in research and industry," *International Journal of Naval Architecture and Ocean Engineering*, vol. 1, no. 2, pp. 89–94, 2009.

[180] D. Hirche, F. Birkholz, and O. Hinrichsen, "A hybrid Eulerian-Eulerian-Lagrangian model for gas-solid simulations," *Chemical Engineering Journal*, vol. In Press, Corrected Proof. Available online 23 August 2018., 2018.

[181] Y. Wang, L. Zhou, and Q. Yang, "Hydro-mechanical analysis of calcareous sand with a new shape-dependent fluid-particle drag model integrated into CFD-DEM coupling program," *Powder Technology*, vol. 344, pp. 108–120, 2019.

[182] D. Gidaspow, *Multiphase flow and fluidization: continuum and kinetic theory descriptions*. Academic press, 1994.

[183] S. Harris and D. Crighton, "Solitons, solitary waves, and voidage disturbances in gas-fluidized beds," *Journal of Fluid Mechanics*, vol. 266, pp. 243–276, 1994.

[184] M. Andrews and P. O'Rourke, "The multiphase particle-in-cell (MP-PIC) method for dense particulate flows," *International Journal of Multiphase Flow*, vol. 22, no. 2, pp. 379–402, 1996.

[185] M. Feng, F. Li, W. Wang, and J. Li, "Parametric study for MP-PIC simulation of bubbling fluidized beds with Geldart A particles," *Powder Technology*, vol. 328, pp. 215–226, 2018.

[186] S. Yang, H. Wu, W. Lin, H. Li, and Q. Zhu, "An exploratory study of three-dimensional MP-PIC-based simulation of bubbling fluidized beds with and without baffles," *Particuology*, vol. 39, pp. 68–77, 2018.

[187] F. Li, F. Song, S. Benyahia, W. Wang, and J. Li, "MP-PIC simulation of CFB riser with EMMS-based drag model," *Chemical Engineering Science*, vol. 82, pp. 104–113, 2012.

[188] H. Razmi, A. S. Goharrizi, and A. Mohebbi, "CFD simulation of an industrial hydrocyclone based on multiphase particle in cell (MPPIC) method," *Separation and Purification Technology*, vol. 209, pp. 851–862, 2019.

[189] Q. Ma, F. Lei, X. Xu, and Y. Xiao, "Three-dimensional full-loop simulation of a high-density CFB with standpipe aeration experiments," *Powder Technology*, vol. 320, pp. 574–585, 2017.

[190] S. Karimipour and T. Pugsley, "Application of the particle in cell approach for the simulation of bubbling fluidized beds of Geldart A particles," *Powder Technology*, vol. 220, pp. 63–69, 2012.

[191] A. Stroh, F. Alobaid, M. T. Hasenzahl, J. Hilz, J. Ströhle, and B. Epple, "Comparison of three different CFD methods for dense fluidized beds and validation by a cold flow experiment," *Particuology*, vol. 29, pp. 34–47, 2016.

[192] K. Luo, F. Wu, S. Yang, and J. Fan, "CFD-DEM study of mixing and dispersion behaviors of solid phase in a bubbling fluidized bed," *Powder Technology*, vol. 274, pp. 482–493, 2015.

[193] D. Fantin, *Towards Fluid-Particle Simulation. CFD-DEM Coupling.* PhD thesis, Delft University of Technology, 2018.

[194] P. Minev and P. N. Vabishchevich, "Splitting schemes for the stress formulation of the incompressible Navier-Stokes equations," *Journal of Computational and Applied Mathematics*, vol. 344, pp. 807–818, 2018.

[195] J. Guermond and P. Minev, "A new class of massively parallel direction splitting for the incompressible Navier–Stokes equations," *Computer Methods in Applied Mechanics and Engineering*, vol. 200, no. 23, pp. 2083–2093, 2011.

[196] P. Birken and A. Jameson, "Nonlinear iterative solvers for unsteady Navier-Stokes equations," *Proceedings of Symposia in Applied Mathematics*, pp. 429–437, 2009.

[197] S. S. Clift and P. A. Forsyth, "Linear and non-linear iterative methods for the incompressible Navier-Stokes equations," *International Journal for Numerical Methods in Fluids*, vol. 18, no. 3, pp. 229–256, 1994.

[198] A. Passalacqua and R. Fox, "Implementation of an iterative solution procedure for multi-fluid gas-particle flow models on unstructured grids," *Powder Technology*, vol. 213, no. 1, pp. 174–187, 2011.

[199] C. M. Venier, C. I. Pairetti, S. M. Damian, and N. M. Nigro, "On the stability analysis of the PISO algorithm on collocated grids," *Computers & Fluids*, vol. 147, pp. 25–40, 2017.

[200] H. Xiao, J. Wang, Z. Liu, and W. Liu, "A consistent SIMPLE algorithm with extra explicit prediction — SIMPLEPC," *International Journal of Heat and Mass Transfer*, vol. 120, pp. 1255–1265, 2018.

[201] S. K. D. D.G. Roychowdhury and T. Sundararajan, "Numerical simulation of natural convective heat transfer and fluid flow around a heated cylinder inside an enclosure," *Heat and Mass Transfer*, pp. 565–576, 2001.

[202] A. Kumar and A. Dhiman, "Forced Convection Heat Transfer of Newtonian Fluids from a Backward-Facing Step: Effects of Expansion Ratio, Reynolds, and Prandtl Numbers," *Heat Transfer-Asian Research*, vol. 45, no. 4, pp. 313–341, 2016.

[203] E. Salimipour, "A numerical study on the fluid flow and heat transfer from a horizontal circular cylinder under mixed convection," *International Journal of Heat and Mass Transfer*, pp. 365–374, 2019.

[204] P. M. Gresho, R. L. Lee, R. L. Sani, and T. Stullich, "Time-dependent FEM solution of the incompressible Navier–Stokes equations in two-and three-dimensions," tech. rep., California Univ., 1978.

[205] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou, "Communication-optimal Parallel and Sequential QR and LU Factorizations," *SIAM Journal on Scientific Computing*, vol. 34, no. 1, pp. A206–A239, 2012.

[206] "Suite Sparse." `https://people.engr.tamu.edu/davis/suitesparse.html`.

[207] E. Burman, "Consistent SUPG-method for transient transport problems: Stability and convergence," *Computer Methods in Applied Mechanics and Engineering*, pp. 1114–1123, 2010.

[208] A. J. Chorin, "Numerical solution of the Navier-Stokes equations," *Mathematics of Computation*, vol. 22, no. 104, pp. 745–745, 1968.

[209] R. Temam, "Une méthode d'approximation de la solution des équations de Navier-Stokes," *Bulletin de la Société Mathématique de France*, vol. 96, pp. 115–152, 1968.

[210] E. Weinan and J.-G. Liu, "Projection method I: convergence and numerical boundary layers," *SIAM journal on numerical analysis*, pp. 1017–1057, 1995.

[211] A. Quaini, R. Glowinski, and S. Čanić, "Symmetry breaking and preliminary results about a Hopf bifurcation for incompressible viscous flow in an expansion channel," *International Journal of Computational Fluid Dynamics*, vol. 30, no. 1, pp. 7–19, 2016.

[212] F. Selimefendigil and H. Öztop, "Control of Laminar Pulsating Flow and Heat Transfer in Backward-Facing Step by Using a Square Obstacle," *Journal of Heat Transfer*, vol. 136, no. 8, p. 081701, 2014.

[213] H. Yang, M. Fan, A. Liu, and L. Dong, "General formulas for drag coefficient and settling velocity of sphere based on theoretical law," *International Journal of Mining Science and Technology*, vol. 25, no. 2, pp. 219–223, 2015.

[214] "C++ reference." `https://en.cppreference.com/w/cpp/thread`.

[215] I. Erhardt, *Investigation of the Powder Cone in Laser Metal Deposition*. Master of Science in Engineering Management, Department of Management, Economics and Industrial Engineering, Politecnico di Milano, 2017.

[216] T. Engen, "CFD analysis of gas-particle flow in a scaled circulating fluidized bed," 2016.

[217] J. Kussin and M. Sommerfeld, "Experimental studies on particle behaviour and turbulence modification in horizontal channel flow with different wall roughness," *Experiments in Fluids*, vol. 33, pp. 143–159, 2002.

[218] C. Güttler, D. Heißelmann, J. Blum, and S. Krijt, "Normal Collisions of Spheres: A Literature Survey on Available Experiments," 2012.

[219] H. Tan, Y. Fang, C. Zhong, Z. Yuan, W. Fan, Z. Li, J. Chen, and X. Lin, "Investigation of heating behavior of laser beam on powder stream in directed energy deposition," *Surface and Coatings Technology*, p. 126061, 2020.

[220] A. Dapelo, "Numerical simulations of jet impingement cooling: OpenFOAM vs. Ansys Fluent comparison," Master's thesis, University of Genova, Mechanical Engineering, 2016.

[221] S. Donadello, V. Furlan, A. G. Demir, and B. Previtali, "Interplay between powder catchment efficiency and layer height in self-stabilized laser metal deposition," *Optics and Lasers in Engineering*, vol. 149, p. 106817, 2022.

[222] I. Gornushkin and G. Pignatelli, "Optical detection of defects during laser metal deposition: Simulations and experiment," *Applied surface science*, vol. 570, p. 151214, 2021.

[223] M. Murer, V. Furlan, G. Formica, S. Morganti, B. Previtali, and F. Auricchio, "Numerical simulation of particles flow in Laser Metal Deposition technology comparing Eulerian-Eulerian and Lagrangian-Eulerian approaches," *Journal of Manufacturing Processes*, vol. 68, pp. 186–197, 2021.

[224] W. Jiazhu, T. Liu, H. Chen, F. Li, H. Wei, and Y. Zhang, "Simulation of laser attenuation and heat transport during direct metal deposition considering beam profile," *Journal of Materials Processing Technology*, vol. 270, pp. 92–105, 2019.

[225] F. Lia, J. Park, J. Tressler, and R. Martukanitz, "Partitioning of laser energy during directed energy deposition," *Additive Manufacturing*, vol. 18, no. Supplement C, pp. 31–39, 2017.

[226] J. Gao, C. Wu, X. Liang, Y. Hao, and K. Zhao, "Numerical simulation and experimental investigation of the influence of process parameters on gas-powder flow in laser metal deposition," *Optics & Laser Technology*, vol. 125, p. 106009, 2020.

[227] Z. Jardon, P. Guillaume, J. Ertveldt, M. Hinderdael, and G. Arroud, "Offline powder-gas nozzle jet characterization for coaxial laser-based directed energy deposition," *Procedia CIRP*, vol. 94, pp. 281–287, 2020.