Challenges in applying the PSPG/SUPG Finite element method to the atmosphéric boundary layer

Yoshiyuki Nishio

VON KARMAN INSTITUTE FOR FLUID DYNAMICS
ENVIRONMENTAL & APPLIED FLUID DYNAMICS DEPARTMENT

UNIVERSITÉ DE LA ROCHELLE - UFR SCIENCE ET TECHNOLOGIE
ECOLE DOCTORALE SCIENCES ET INGÉNIERIE EN MATÉRIAU, MÉCANIQUE, ÉNERGÉTIQUE ET AÉRONAUTIQUE
LABORATOIRE DES SCIENCES DE L'INGÉNIEUR POUR L'ENVIRONNEMENT

ROYAL MILITARY ACADEMY
DEPARTMENT OF MECHANICAL ENGINEERING

Challenges in applying the PSPG/SUPG Finite Element Method to the Atmospheric Boundary Layer

YOSHIYUKI NISHIO

Cover art: Instantaneous turbulent velocity contour for a neutral atmospheric boundary layer, using a Smagorinsky-Lilly model and a Schumann wall model implemented in the finite element method.

Thesis presented by Yoshiyuki Nishio in order to obtain the degree of "Doctor of Philosophy in Applied Sciences - Mechanics", Université de la Rochelle, France, and the Royal Military Academy, Belgium, 20th of July 2021.
Thesis defended on the 23th of September 2021.

Promoters:
Prof. em. dr. Walter Bosschaerts (Royal Military Academy, Belgium)
Prof. dr. Jeroen van Beeck (von Karman Institute for Fluid Dynamics, Belgium)
Maj assoc. dr. prof. Bart Janssens (Royal Military Academy, Belgium)
Assoc. prof. HDR Karim Limam (La Rochelle University, France)

Doctoral Committee:
Col prof. dr. Bart Scheers (**president**, Royal Military Academy, Belgium)
Prof. dr. Emmanuel Antczak (**rapporteur**, University of Artois, France)
Assoc. prof. HDR Cristiana Croitoru (**rapporteur**, Technical University of Civil Engineering Bucharest, Romania)
LtCol assoc. prof. dr. Kristof Harri (Royal Military Academy, Belgium)
Prof. em. dr. Walter Bosschaerts (Royal Military Academy, Belgium)
Prof. dr. Jeroen van Beeck (von Karman Institute for Fluid Dynamics, Belgium)
Assoc. prof. HDR Karim Limam (La Rochelle University, France)
Maj assoc. prof. dr. Bart Janssens (Royal Military Academy, Belgium)

D/2021/0238/747, T. Magin, Editor-in-Chief
Published by the von Karman Institute for Fluid Dynamics with permission.

ISBN 978-2-87516-177-2

Contents

Abstract

English version

In the context of a Chemical, Biological, Radiological, and Nuclear (CBRN) application for the Belgian Defense, the original objective of the work was to simulate a realistic open-air CBRN case (e.g. dispersion after an explosion of particles in a city) by applying the Streamline-Upwind Petrov-Galerkin (SUPG) stabilization on a finite element method (FEM), together with a second phase (i.e. particles). This would be done through the code *Coolfluid 3*, a Domain Specific Language (DSL) written in `C++`.

However, open-air applications require to describe the atmospheric boundary layer (ABL) correctly. This has never been done using stabilized FEM. Consequently, the challenge of this work is to answer the simple question: How to model an ABL taking advantage of the SUPG stabilization method.

To reduce the number of elements produced by a wall-resolved simulation, the ABL was implemented with a wall model and verified in 2D. At the same time, a few corrections (e.g. grid scalability, stable velocity profile) could also be addressed.

However, the 3D implementation revealed spurious oscillations, suggesting a numerical origin. Although SUPG does provide dissipation, it seemed not sufficient enough for such a high Reynolds flow. Consequently, two directions were followed to add numerical dissipation: Firstly, the implementation of an extended version of the SUPG, the Variational MultiScale method (VMS), was initiated. The latter provides a combined framework for stabilization and turbulence modeling. Secondly, two LES formulations, known for their dissipative behavior, were integrated.

Having solved the spurious oscillations, the velocity profile was analyzed. Eventually, the viscous Reynolds number for the ABL domain was reduced to enable the comparison with an available DNS result. Fortunately, relative to the standard no-slip wall condition and the friction velocity condition, the wall model implementation provided the best result, although not matching.

In conclusion, we ascertained two methodologies (LES and SUPG / VMS) that have the potential to approach the ABL flow. The stabilized FEM using SUPG revealed that it is currently not sufficient to avoid spurious oscillations in an ABL flow. In contrast, LES provided encouraging results for reduced Reynolds number, supporting that some kind of turbulence model is indispensable. This emphasizes that the implementation of VMS should be promising, although challenging.

Version française

Suite à une requête de la Défense belge dans le cadre des applications CBRN (chimique, biologique, radiologique et nucléaire), l'objectif initial du travail était de simuler un cas CBRN réaliste à l'air libre (i.e. dispersion de particules après une explosion dans une ville), en appliquant la stabilisation Streamline-Upwind Petrov-Galerkin (SUPG) sur une méthode d'éléments finis (FEM), incluant une deuxième phase (i.e. particules). Pour cette simulation, les développement se font dans le code *Coolfluid 3*, un langage spécifique au domaine (DSL) écrit en `C++`.

Cependant, les applications à l'air libre nécessitent de décrire correctement la couche limite atmosphérique (ABL). Cela n'a jamais été fait en utilisant des éléments finis stabilisés. Par conséquent, le défi de ce travail est de répondre à la question simple : Comment modéliser une ABL en profitant de la méthode de stabilisation SUPG.

Afin de réduire le nombre d'éléments nécessaires pour une simulation résolvant toutes les échelles de turbulences jusqu'aux parois, l'ABL a été implémentée avec un modèle de paroi et vérifié en 2D, tandis que quelques corrections (e.g. la résolution du maillage, la stabilité du profil de vitesse) ont également pu être adressées.

Néanmoins, l'implémentation 3D a révélé des oscillations parasites, laissant supposer à une origine numérique. Bien que SUPG produise de la dissipation, cette dernière ne semble pas suffisante pour un écoulement à nomber de Reynolds aussi élevé. Par conséquent, pour ajouter de la dissipation, deux directions ont été suivies : Premièrement, une implémentation de l'évolution de la SUPG, la méthode Variational MultiScale (VMS), a été initiée. Cette dernière fournit un cadre combiné pour la stabilisation et la modélisation de la turbulence. Deuxièmement, deux formulations LES, connues pour leur comportement dissipatif, ont été intégrées.

Après avoir réduit les oscillations parasites, le profil de vitesse a été analysé. Finalement, pour permettre la comparaison avec un résultat DNS disponible, le nombre de Reynolds visqueux du domaine ABL a été réduit. Favorablement, relativement à deux autres conditions, l'implémentation du modèle ABL a fourni le résultat se rapprochant le plus de la courbe DNS.

En conclusion, nous avons déterminé deux méthodologies (LES et SUPG / VMS) qui ont le potentiel d'approcher l'écoulement ABL. La FEM stabilisée utilisant SUPG a révélé qu'elle n'est actuellement pas suffisante pour éviter les oscillations parasites dans le cas d'un écoulement ABL. En revanche, LES a fourni des résultats encourageants, ce qui prouve qu'un certain type de modèle de turbulence est indispensable. Cela souligne l'intérêt pour la méthode VMS, bien que difficile à implémenter.

Acknowledgments

The successful completion of this work would not have been possible without the help of many people, so I would like to express my gratitude here.

First of all, I thank my promotors, Prof. Janssens, Prof. Limam, Prof. van Beeck, and Prof. Bosschaerts, for offering the opportunity to do this work and for their valuable support throughout the research. I am grateful for the tremendous amount of freedom they allowed, without which it would have been impossible to delve into the C++ programming aspects of the work. I also thank the reporters and other jury members for their questions and comments on the manuscript.

Of course, my colleagues at the RMA were always ready to lend a hand, and I thank all of them. Prof. Bosschaerts, thank you for the opportunity you gave me and all the valuable comments on my work. Benoît, many thanks for all the interesting discussions we had.

Finally, I thank my family and especially my lovely girlfriend. Daphne, thank you for your love and support during all these years; I know it was difficult at times.

Nomenclature

A	[m^2]	Area
C_S	[$-$]	Smagorinsky - Lilly constant
F	[N]	Force
Kn	[$-$]	Knudsen Number
N	[$-$]	Number of steps
N_i	[$-$]	Shape function
P	[N/m^2]	Pressure
Re	[$-$]	Reynolds number
S_{ij}	[s^{-1}]	Rate-of-strain tensor
St	[$-$]	Stokes Number
V	[m^3]	Volume
Δ	[m]	Filter width
Δt	[s]	Timestep ($= t_{n+1} - t_n$)
Γ	[m^2]	Contour of the domain
Ω	[m^3]	Space domain
$\tilde{C}_S$	[$-$]	Smagorinsky - Lilly coefficient
α_X	[$-$]	α-method parameter
β	[$-$]	Robin parameter
ϵ	[m^2/s^3]	Energy dissipation rate
ϵ_I	[m^2/s^3]	Energy production
γ	[$-$]	α-method parameter
κ	[$-$]	von Karman constant (≈ 0.4)
λ_p	[m]	Mean free path of a particle's carrier phase
μ	[$N \cdot s/m^2$]	Dynamic viscosity
ν	[m^2/s]	Kinematic viscosity
$\overline{U}$	[m/s]	Mean bulk velocity
ρ	[kg/m^3]	Density
τ_w	[Pa]	Shear stress at the wall
τ_X	[$-$]	Stabilization coefficient for X
$\tau_{f,p}$	[s^{-1}]	Fluid, particle response time
$\tilde{\epsilon}$	[m^2/s^3]	Energy flux
$\tilde{u}_{adv}$	[m/s^2]	Reference advection velocity
f	[N/kg]	Body force (external force per unit mass)
h	[m]	Height of the model
n	[m]	Normal vector
p	[$kg/(m \cdot s^2)$]	Pressure
t	[s]	Time
u, v	[m/s]	Velocity components
u_τ	[m/s]	Friction velocity
x_s	[m]	Surface points
y	[m]	Vertical distance from the wall

Acronyms

$\mathcal{R}^X$	Residual vector for X
^{137}Cs	Radioactive isotope Cesium-137
A_e	Assembly matrix for the elements
ABL	Atmospheric Boundary Layer
AEGL	Acute Exposure Guideline

	Levels
CBRN	Chemical, Biological, Radiological or Nuclear environment
CFD	Computational Fluid Dynamics
CFL	Courant–Friedrichs–Lewy condition := $\frac{u\Delta t}{\Delta x}$
DNS	Direct Numerical Simulation
DOVO	Dienst voor Opruiming en Vernietiging van Ontploffingstuigen - Explosive Equipment Removal and Destruction Service
DQMOM	Direct Quadrature Method Of Moments
DSL	Domain Specific Language
DSM	Direct Stiffness Method
EDQNM	Eddy-Damped QuasiNormal Markovian
FDM	Finite Difference Method
FEM	Finite Element Method
FVM	Finite Volume Method
IAEA	International Atomic Energy Agency
LBB	Ladyzhenskaya-Babuška-Brezzi Condition
LES	Large Eddy Simulation
LHS	Left Hand Side
LSS	Linear Solver System
MOM	Method Of Moments
NATO	North Atlantic Treaty Organization
PBE	Population Balance Equation
PBL	Planetary Boundary Layer
PDE	Partial Differential Equations
PG	Petrov-Galerkin formulation
PMSS	Parallel-Micro-Swift-Spray
PSD	Particle Size Distribution
PSPG	Pressure-Stabilizing Petrov-Galerkin method
QMOM	Quadrature Method Of Moments
RANS	Reynolds Averaged Navier-Stokes
RHS	Right Hand Side
SL	Smagorinsky-Lilly turbulence model
SU	Streamline-Upwind scheme
SUPG	Streamline-Upwind Petrov-Galerkin method
TFM	Two-Fluid Model
UK	United Kingdom
US	United States of America
VMS	Variational Multi-Scale method
WALE	Wall-Adapting Local Eddy-viscosity

Superscripts and Subscripts

adv	Advection
e	Per element
F, M	α-method parameters
l	Multi-corrector iteration number
M, C	Momentum, continuity
m, c	Momentum, continuity
n	Timestep
PS, SU, BU	Parameter of PSPG/-SUPG/BULK stabilization
T	Transpose of

Chapter 1

Introduction

1.1 Background

In April 2010, a considerable part of the European airspace was locked down due to the sudden eruption of the Icelandic volcano Eyjafjallaj “okull.

It resulted in the most extensive air-traffic shutdown in Europe after the Second World War. From the 15th of April 2010 till the 17th of May 2010 for some countries, the airspace was partially or totally interrupted due to ashes suspended in the air, leaving the airplanes on the ground to avoid casualties.

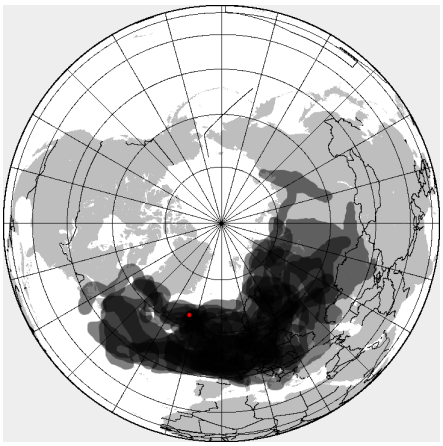


Figure 1.1: Volcanic ash dispersion from Eyjafjallajökull (From Wikimedia, 2010)

Such a considerable amount of gas and particle dispersion in the atmosphere influenced aerial transport but also all living beings that need to breathe air (Bukowiecki et al., 2011).

In 2016, a massive citizen's action concerning air quality was initiated in Europe (by the OK Lab Stuttgart from the Open Knowledge Foundation Germany (Blon, 2017; Luftdaten, 2017)) and, in parallel, in the Northern part of Belgium (by the University of Antwerpen, the newspaper De Standaard and the Flemish Environment Society (CurieuzeNeuzen, 2018)).

Each participant received a measurement kit and had to report the NO_2 gas concentration in the air. NO_2 gas is easy to measure and mainly correlates with emissions from vehicles or industries using combustion engines. This gas concentration is related to the distribution of $PM2.5$ and $PM10$ solid fine particles, where PM stands for atmospheric Particulate Matter and is followed by a number that represents the largest size of the considered particles, in μm (Priemus and Schutte-Postma, 2009).

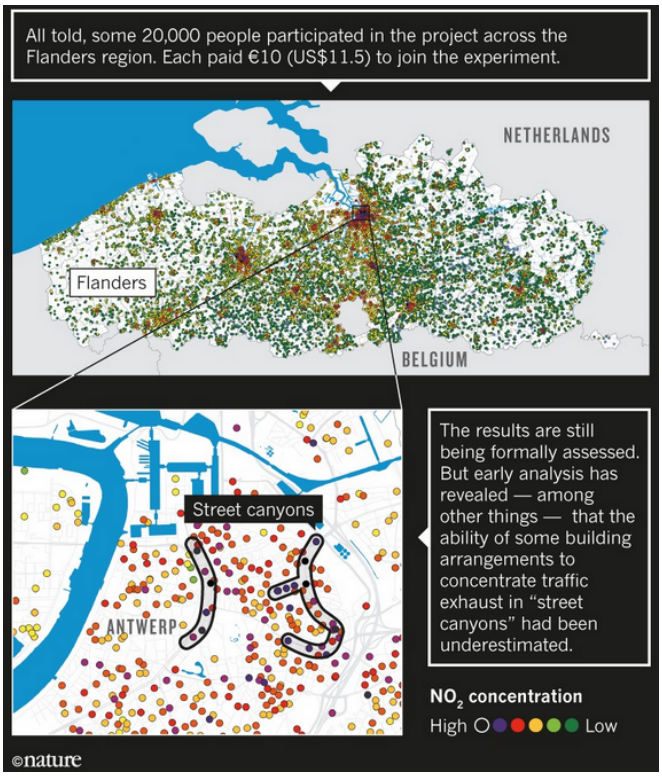


Figure 1.2: Citizen Science: Air Quality in Flanders, Belgium (From CurieuzeNeuzen, 2018)

Note that measuring $PM2.5$ and $PM10$ requires devices that can analyze the size and concentration of particles. The particles are generated by the exhaust of combustion engines, but also by sand particles from dry regions, wood-burning (Lefebvre et al., 2016), or even agricultural regions producing dust during cereals extraction. Thus, knowledge of the considered regions' geography is required to reach the correct conclusions.

Moreover, the human breathing system can not filter $PM2.5$ particles nor reject them. Consequently, these particles are kept in the lungs and therefore pose a high health risk factor. However, $PM10$ particles can still have

irritating effect (Oberdörster, Oberdörster, and Oberdörster, 2005; Pope and Dockery, 2006).

Thus, returning to the massive citizen's action of 2016, the participants were asked to register NO_2 concentration at a fixed location on their home's facade, schools, and several industrial areas.

This action aimed to objectify the impact of air quality on health and quality of life and increase awareness of the problem with citizens and politicians (Meysman and Craemer, 2018).

As the last example, at the beginning of 2020, a world pandemic occurred, leading to more than two months of complete lockdown in several countries and social and economic restrictions for the following months.

At first instance, only the propagation by physical contact was considered, but a few epidemiologists rapidly raised questions concerning propagation through the air (Lednicky et al., 2020). The citizens were asked to reduce contact by avoiding crowds, keeping their distance, cleaning their hands, and eventually wearing masks to reduce the spreading.

The pandemic has not yet vanished, and until the vaccine is effectively operating, only social distancing measures and moderate tracing methodologies are proposed to contain the spreading. Its economic and sociological consequences are considerable.

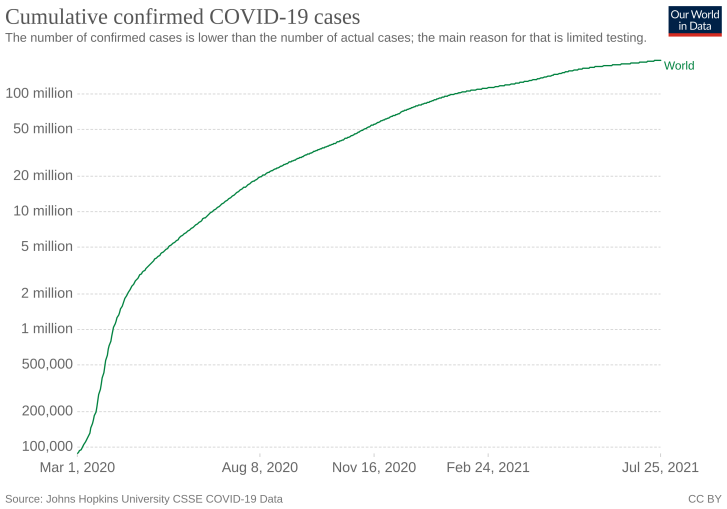


Figure 1.3: COVID-19: total number of confirmed cases in the world (log scale) (From Ritchie et al., 2020)

These three examples emphasize that the air surrounding the earth can transport various types of particles that can seriously impact the daily life of one another. The Belgian Defense, in charge of the external security of the Belgian Kingdom and all its citizens, is concerned with the threat posed

by aerosols. Some of these particles are liquid; others are solid; some particles are more dangerous because of their biochemical interactions with its surrounding, others because of their obstructing effect. In a military context, the accurate prediction of an explosion's dispersion is both a defensive and an offensive asset that could be taken into account for all chemical, biological, radiological or nuclear (CBRN) applications. This point will be developed further in the motivation section (§ 2).

In this study, the aim is to extend the development of the fluid simulation software *Coolfluid 3*, a cutting-edge `C++` framework (Quintino et al., 2012) presented in the work of Janssens, 2014 by applying a multiphase flow model to dispersion cases in the atmosphere.

The model would allow the simulation of flow in an open-field context, taking the atmospheric dimensions into account, and would study the dispersion of fine solid particles in the air.

Referring to the three proposed examples, such a model would allow a better prediction by handling the dynamics of the particles' dispersion in the air, considering complex geometries and domains with sizes up to a few kilometers.

1.2 Thesis outline

While the introduction tends to smoothly bring the reader to the notion of what the subject is, this section presents the document's structure.

The document is divided into four parts: context, modeling, validation, and conclusions.

In Part I, we present both the motivation of this work and the physical aspects related to the current subject. This part is required to understand the constraints that have to be taken into account in conjunction with their associated theory.

Next, in Part II, the numerical methods will be addressed. Since the work is first and foremost a numerical implementation effort, the fluid dynamics theory has to be interpreted and translated into a discrete numerical model suitable for scalable, parallel solutions on a computer.

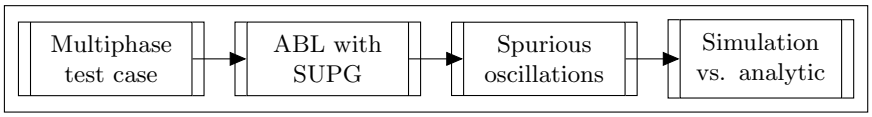
After defining the modeling, its implementation has to be validated. Part III compares simulation results to several study cases presented in the literature. This enables the reader to gain confidence in the proposed implementation and investigate the limitations and potential errors.

To conclude, Part IV summarises the aim of the presented work, its achievement, but also its limitations, and how further studies could reduce these limitations.

1.3 Thesis flow chart diagram

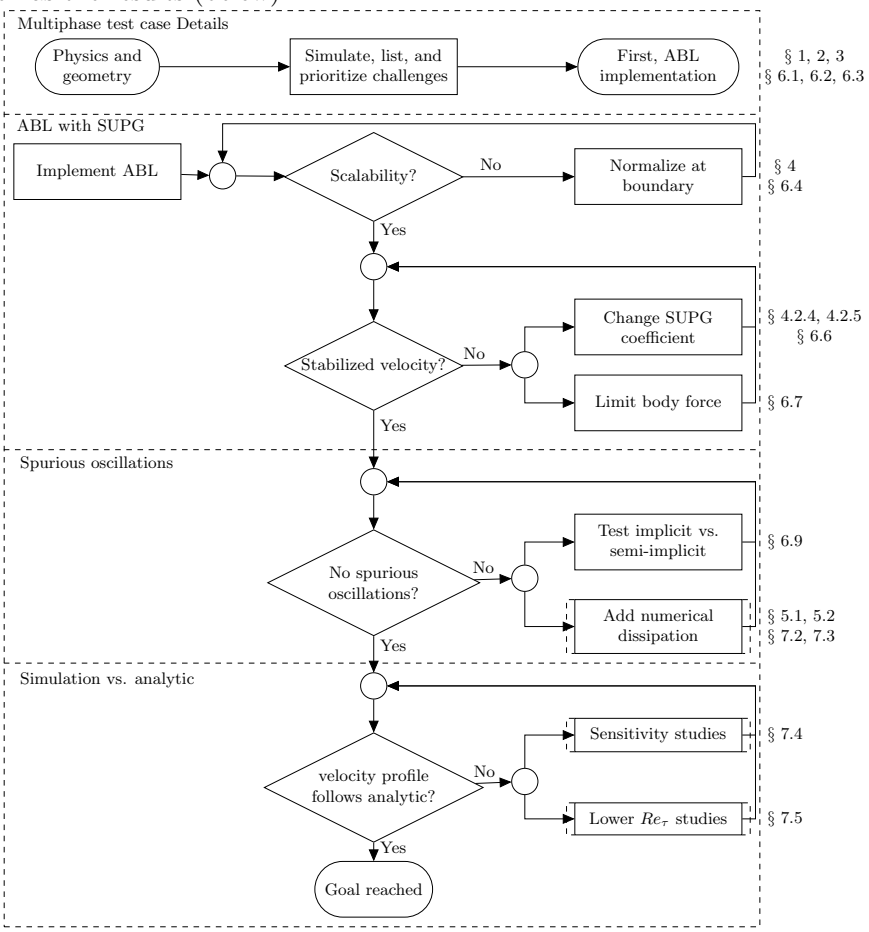
A flow chart diagram is proposed to the reader to provide a schematic view of the structure of this manuscript. Firstly, a global view is proposed.

Global view



Details per process

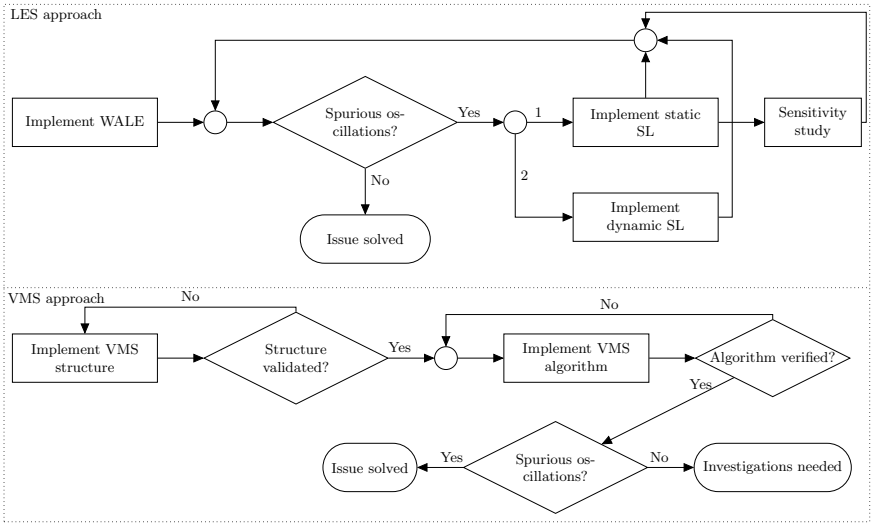
Secondly, each part is developed in a detailed flow chart, where, on the right side, one will find a reference to the related theoretical sections (above) as well as the results (below).



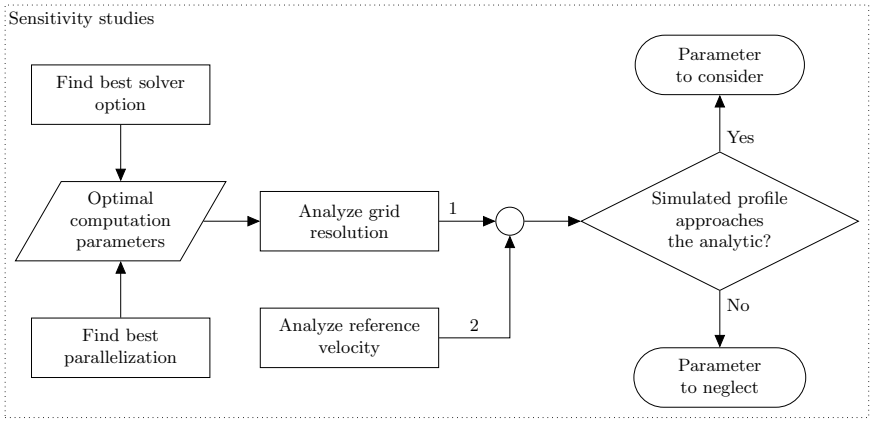
Details per subprocess

For three subprocesses, although the references are already available in the previous chart, more details is provided here.

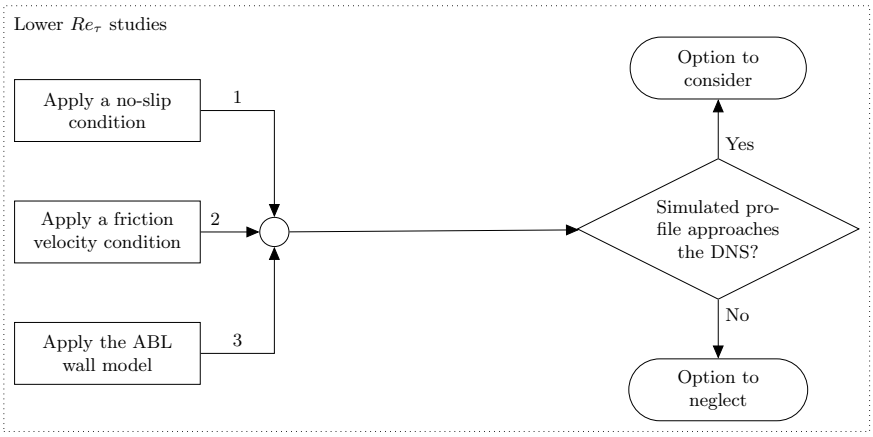
Firstly, the *Add numerical dissipation* subprocess contains two directions, the LES approach and the VMS approach:



Secondly, the *Sensitivity studies* subprocess is presented here after.



Finally, the chart concerning the reduction to the lower viscous Reynolds number is displayed here.



The idea of these diagrams is to offer the reader an aerial view of the holograph to smooth the immersion.

Having introduced the work, its flow, and equipped with the instrument hereabove, we can enter the contextual part in serenity.

Part I

Context

Chapter 2

Motivation

In the introduction, the subject was initiated to the reader through rather "civilian and environmental" concerns. In this section, the extra motivation for the Belgian Defense to explore this field will further be explained.

In the protection and safety activities performed by armies at an international level, there is a field called CBRN (Chemical, Biological, Radiological, and Nuclear) defense that has snowballed for the last decades. It is concerned with the protection against different kinds of contaminating agents and particles, ranging from chemical and biological weapons to radioactive particles or fallout. Apart from their composition, these agents can be characterized by their ability to contaminate by dispersion or spreading, yielding to challenging prediction and protection measures. The interest in CBRN protection measures emerged from three main streams:

- classical chemical, biological, and nuclear warfare,
- industrial accidents, and
- terrorism.

CBRN warfare and industrial accidents

Although their origins are distinct, we will address the two first streams through one shared topic related to dispersion: nuclear plant disasters. Indeed, a nuclear plant destroyed during a war will primarily affect the inhabitants and the military troops in the vicinity before propagating farther away. Thus, we could start with what is considered the most severe disaster in the nuclear industry: the explosion of the Tchernobyl power plant on the 26th of April 1986.

This accident happened during a maintenance procedure on reactor number 4. After a succession of unforeseen, unplanned, and incorrect manipulations, the control of the reactor's kernel was lost. In a few seconds, the nominal power of the reactor was more than 100 times exceeded and followed by two explosions breaking the reactor's structure, blasting radioactive particles at $1000m$ of altitude into the atmosphere. Two days later, a Swedish nuclear plant raised the European alarm due to a high radioactivity concentration unrelated to their installation. The Soviet ministers' council confirms the existence of a severe nuclear hazard on the site of Tchernobyl that could

have reached the other European countries. In April, an international conference is organized by the International Atomic Energy Agency (IAEA) to gather experts (IAEA, 1992; Marshall, 1986). By December 1986, a protective sarcophagus was built around the nuclear plant to reduce radioactive contamination.

For this disaster, one should notice that the Soviet government did not mention the use of a dispersion modeling methodology. However, some models were available in the US and the UK (ApSimon, Macdonald, and Wilson, 1986; ApSimon and Wilson, 1987).

25 years later, in Japan, after a significant earthquake happening on the 11th of March 2011, the nuclear plants' sites of Fukushima, *Dai-ichi* and *Dai-ni*, were both weakened (Holt, Campbell, and Nikitin, 2012). Concerning the site that was the most severely impacted, *Dai-ichi*, from the six nuclear reactors available, three were working and automatically stopped, following the emergency procedure. While the city's electrical net was out-of-order, electro-generators powered the mandatory water cooling system until, 41 minutes later, a tsunami, consequence of the earthquake, flooded all reactors, except nr 6, leaving them without electricity, hence without their cooling system. Consequently, reactors nr $1, 2, 3$ all melted, producing radioactive plumes in the atmosphere and the pacific ocean (see figure 2.1).

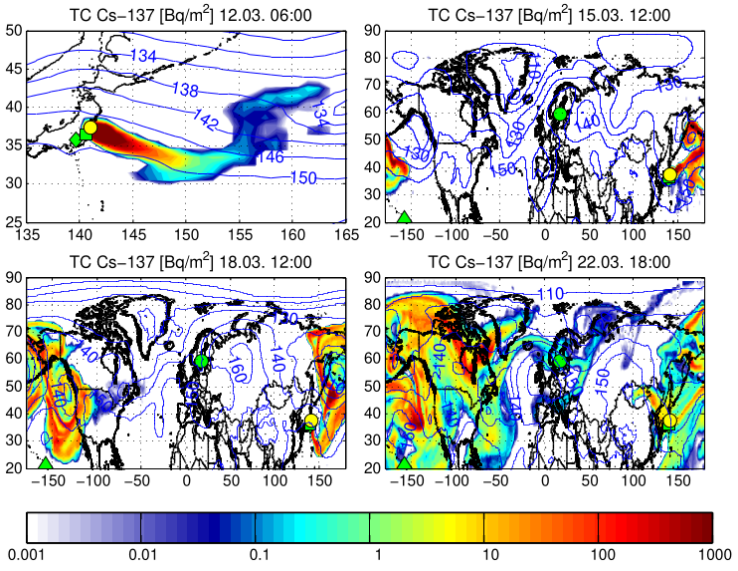


Figure 2.1: Simulated total atmospheric columns of radioactive ^{137}Cs (contours) and geopotential at 850 hPa from GFS (blue isolines). (yellow circle: Nuclear reactor leak ; green shapes: air sampling site) (From Stohl et al., 2011)

In Japan, nearly 200.000 inhabitants (18.000 losses, 6.000 injured and 170.000 relocated) were directly impacted (in the short term, as a result of the initial tsunami; in the long term, as a consequence of the radioactivity).

For this second nuclear disaster, although it was not prevented, the safety measures and the dispersion modeling were available and applied (Chino, Ishikawa, and Yamazawa, 1993; Srinivas et al., 2012).

Terrorism

The third stream came, concurrently, starting beginning of the twenty-first century, with a new wave of terrorism that amplified the interest in CBRN applications.

It amplified on the 11th of September 2001 with the hijack and crash of four airliners (Wikipedia, 2001), affecting the United States (US) strongly, fragilizing their symbolic almightiness (and, in a broader sense, of the North Atlantic Treaty Organization (NATO)). The subsequent attacks, mainly in Europe (Reshetin and Regens, 2003; Wikipedia, 2014), were predominantly dispersed and isolated, but, like in the US, they had consequences on a wide range of civilians. This new type of terrorism, difficult to prevent due to its dispersion, raised fear in society and led to unique protection measurements, new investments, and new studies by the governments and their national defenses.

CBRN dispersion

As a result of these three streams, some countries decided to study the spreading of CBRN through the air, water, or any other fluid, considering the environment:

- France: A few studies (Armand, Duchenne, and Bouquot, 2014; Benamrane, 2015; Stohl et al., 2011) were initiated to evaluate and mitigate the impact of a CBRN attack on the most populated cities.

  Figure 2.2 provides a simulated dispersion of an ammonia release in Paris, for different release locations, with various wind directions. The contamination intensity is colored according to the Acute Exposure Guideline Levels (AEGL).

  An equivalent study was proposed for the city of Marseille (figure 2.3), where a RANS (cf. § 4.1.4) simulation was performed using CERES® code, together with the simplified flow and dispersion solver PMSS, for a statistical average wind coming from the south-west. The aim was to provide a fast response on the possible dispersion of the yperite gas, located in a downtown warehouse (Armand, Duchenne, and Bouquot, 2014).

- Germany: Similarly, in the work of Duchenne et al., 2013, the spreading of an ammonia seeding was studied in the second-largest city in

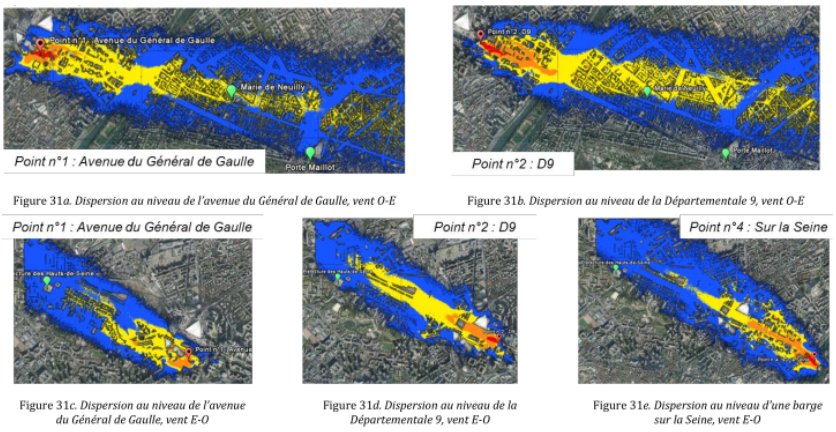


Figure 31a. *Dispersion au niveau de l'avenue du Général de Gaulle, vent O-E*

Figure 31b. *Dispersion au niveau de la Départementale 9, vent O-E*

Figure 31c. *Dispersion au niveau de l'avenue du Général de Gaulle, vent E-O*

Figure 31d. *Dispersion au niveau de la Départementale 9, vent E-O*

Figure 31e. *Dispersion au niveau d'une barge sur la Seine, vent E-O*

Figure 2.2: Simulation of an ammonia release (10 tons) in Paris (From Stohl et al., 2011). Intensities: blue (> 1% AEGL-1) to red (AEGL-3) limits.

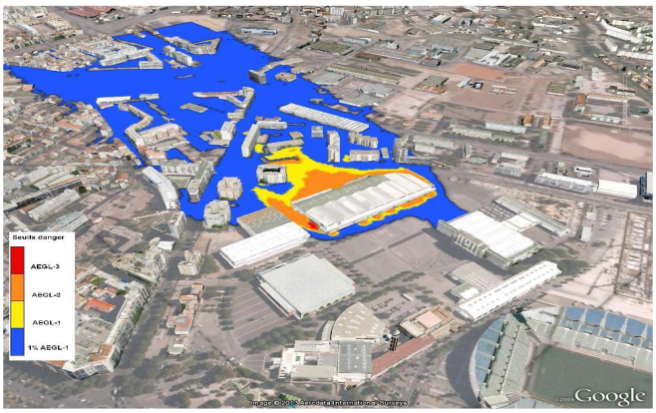


Figure 2.3: AEGL zones for an Yperite dispersion (From Armand, Duchenne, and Bouquot, 2014)

Germany, Hamburg, using both experimental setups and simulations.

Figure 2.4 presents a view of a Hamburg downtown area where the plume of a tracer, located on a boat and emitting continuously for 45 minutes at a constant rate of $0.02kg/s$, was measured. A second experimental test was performed in a wind tunnel (right picture) with a 1 : 350 scaled mockup, using the similarity laws to translate the measurements to realistic values.

Eventually, in the pictures below, three simulations were performed to compute ammonia concentration near the ground. For these simu-

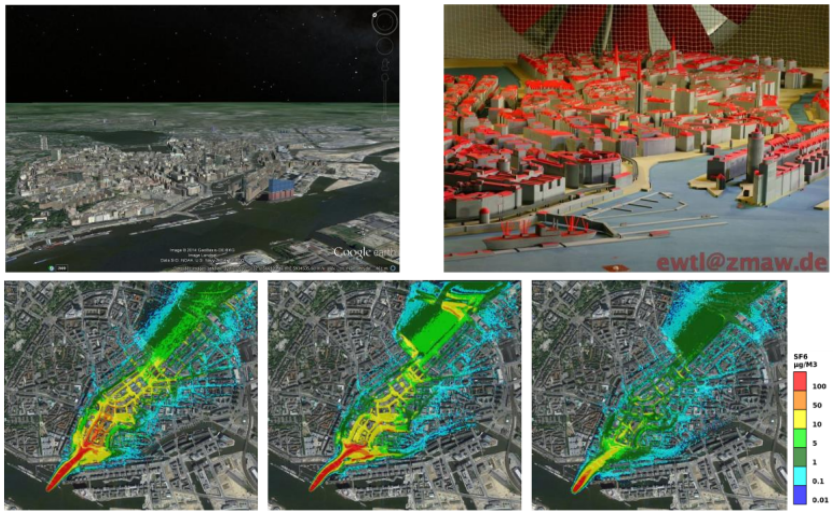


Figure 2.4: Left-top to right-bottom: Hamburg city, wind tunnel mokup, Field experiment with continuous release, Concentration field, Cross-section near the ground (From Duchenne et al., 2013)

lations, two were performed with variations of the PMSS solver used in fig. 2.3, while for the third, the SATURNE® code (that provides a RANS model, coupled to a k-ϵ turbulence model) was used. Note that the PMSS solver uses a Lagrangian approach (§ 3.1.3) with 2000 particles per second for the modeling of the particle dispersion.

Following an analogous reflection, the Belgian Defense was interested in developing a simulation software that could accurately define the dispersion of solid particles in the atmosphere due to an explosion in the open field while being able to handle any complex terrain.

The two main reasons for their interest are:

- During the first and second world wars, the Belgian territory was deeply impacted by bombshells and mines. As a consequence, a branch (DOVO) of the Belgian Defense is dedicated to explosive ordnance disposal activities. They ensure that any explosive is safely removed and destroyed at DOVO's facilities. Since DOVO has to destroy each year around 200 tons of munitions, they have set a storage space (cf. fig 2.5).

  Both spaces (i.e. the original location of the explosive and the storage space) can represent a hazard (e.g. explosions but also dispersions of potential contaminant). Although DOVO is applying strong regulations to each of their processes, they remain interested in pursuing the development of a simulation tool that could refine the prediction

associated with the dispersion of solid particles in the atmosphere.

Figure 2.5: Explosive storage space at Poelkapelle, Belgium (From Defensie, 2019)

- After the resurgence of terrorist attacks in Europe (2014–2017), the Belgian government requested the Belgian Defense to assign detachments of soldiers in the main cities. The explosion of a subway inside the Belgian capital, Brussels (near the European Parliament), on the 22nd of March 2016 increased even further the measures taken after the Paris attacks on the 13th of November 2015.

  As a consequence, the interest of the Belgian Defense in CBRN applications grew. Notably, they were keen to enhance their possibility of predicting how and where an isolated explosion or contaminant could propagate through any fluid in complex topographies.

For these reasons, this work aims to develop further the thesis of Janssens, 2014, that created a CFD software, *Coolfluid 3*, enabling the simulation and study of flows over complex geometries, thanks to its inherent use of Finite Elements, together with its integrated stabilization method (§ 4.2.5).

Chapter 3

Related physical aspects

Having defined the context and the motivation that brought us to study the propagation of a multiphase flow in an open field or city, this chapter will introduce two required concepts before laying the focus on the modeling:

- Multiphase flow
- Atmospheric Boundary Layer (ABL)

3.1 Multiphase flow

A multiphase flow is a broad term covering a wide spectrum of flows, having single or multiple interactions between at least two physical phases (cf. solid, liquid, or vapor phase).

In this work, we focus on a two-phase flow involving a gaseous fluid (viz. the air) and solid particles (viz. $PM2.5$). The former is in a continuous phase (i.e. where elements can move from one position to another while remaining in the same medium); the latter is in a dispersed phase (as opposed to a continuous phase).

The reason is double: first, the description of all types of multiphase flows involving a dispersed and a continuous phase is a domain that, although they all do have a common element (i.e. the possibility to distinguish the continuous from the dispersed phases), is far too extensive for this work. (The reader is referred to Crowe et al., 2011; Fox, 2003; Marchisio and Fox, 2013 for detailed explanations) ; second, the motivation chapter (§ 2) promoted the study of how solid particles emanating from an explosion would propagate.

It should be pointed out that the air is a single-phase (here, gas) multicomponent (mixture of chemical species) flow. It can be simplified to a single component flow for practical matters, with specific viscosity and thermal conductivity. This simplification is possible when molecular weights are alike and there is no high temperature (that could facilitate molecular disassociation).

Eventually, in the atmosphere, the air is considered a continuum, namely, a continuous matter acting as a bulk (where all constituents aim towards the same direction). Since the Navier-Stokes equations (§ 4.1.2) can describe

a continuum, they can be applied to study the motion of the air in the atmosphere.

3.1.1 Dimensionless numbers

To introduce the theory behind a two-phase flow, two dimensionless numbers have to be defined:

- The Knudsen Number: provides an information on the rarefaction of the flow. It is defined as the ratio between the mean free path λ_f of the particle's carrier phase by its diameter d_p:

$$Kn = \frac{\lambda_f}{d_p} \tag{3.1}$$

  The Knudsen number should be smaller than 10^{-3} to validate the continuum assumption (Crowe et al., 2011). At standard conditions, in the case of air (containing 78% of nitrogen) with particles of diameter $d_p = 2.5\mu m$ and a $\lambda_f = 6.8e{-}8m$, Kn would be equal to 0.027, making the continuum assumption questionnable in our scope. However, practically, this assumption was used in similar studies (Bechmann, 2006; Goit, 2015), resulting in acceptable results, and is therefore considered in this study as well.

- The Stokes Number: provides information on the behavior of large particles compared to smaller particles (here the surrounding fluid). It is defined as the ratio between the large particle response time τ_p and the fluid time scale τ_f, that both have dimensions of $[s]$:

$$St = \frac{\tau_p}{\tau_f} \tag{3.2}$$

  where τ_p can be derived from Stokes flow motion equation for a small sphere (Crowe et al., 2011):

$$\frac{dv}{dt} = \frac{18\mu}{\rho_p d_p^2}(u_p - v_p) \quad with \quad \frac{18\mu}{\rho_p d_p^2} \equiv \tau_p^{-1} \tag{3.3}$$

  and τ_f can be determined by means of two expressions (depending on the available inputs):

$$\tau_f = \frac{d_f}{|v_f - v_p|} \quad \vee \quad \tau_f = \frac{\nu}{u_\tau^2} \tag{3.4}$$

  with d_f being the diameter of the small fluid particles and $|v_f - v_p|$ the relative velocity between the large particles and the small fluid particles (Crowe et al., 2011) ; and ν being the fluid viscosity, while

u_τ is the fluid friction velocity at the surface (when considering a channel flow (Kuerten, 2006)).

Note that the particle response time τ_p can be interpreted as the time necessary for the particle to reach 63% of the fluid velocity, starting from a standstill.

Thus, a Stokes number significantly smaller than one ($St \ll 1$) will correspond to particles with their velocity considerably influenced by the fluid velocity. On the contrary, a Stokes number that is much higher than one ($St \gg 1$) will indicate that the particle velocity is largely independent of the fluid velocity.

The distinction given by the Stokes number has an impact on the model chosen. In § 3.1.3 we will describe four techniques.

Nevertheless, before describing these approaches, another notion will greatly influence the complexity and approach that will be used further on. This notion is related to the particle response time τ_p defined in this section (cf. eq. (3.3)) and is therefore developed in the next section.

3.1.2 Particles in the flow

In a two-phase dispersed flow, if the fluid forces prevail on the interactions of the particles (collision against walls and/or other particles) to stir the motion of particles, then the flow is considered diluted.

To fulfill this condition, one will refer to the ratio between the particle response time τ_p (eq. (3.3)) and the average collision time τ_c. For a dilute flow, the ratio should be smaller than unity, signifying that the particle will spend more time propagating than colliding.

One can refer to the kinetic theory (Feynman, Leighton, and Sands, 1965) to express the particle collision frequency, that is, the inverse of τ_c. Its expression, for a monodisperse flow, is:

$$\tau_c = (n\pi d_p^2 v_r)^{-1} \tag{3.5}$$

where v_r is the relative velocity between particles and n is the particle concentration number, defined as:

$$n \equiv \lim_{\delta V \to \delta V_0} \frac{\delta N}{\delta V} \tag{3.6}$$

with δN the number of particles and δV the considered volume. The expression for a polydisperse flow follows a more complex derivation, detailed in Janssens, 2014.

If the ratio τ_p/τ_c is greater than unity, the interactions between particles will represent a major motion agent, leading to the notion of interaction between phases. We will outline three cases:

- the one-way coupling: is a coupling where only the fluid phase will affect the particles.
- two-way coupling: will enable the interaction in both directions (from fluid to particles and reverse).
- four-way coupling: integrates the interactions between particles (Vreman et al., 2008)

In this study, a dilute flow, thus with no particle interaction, will be considered to avoid further complications.

3.1.3 Modeling approaches

From previous § 3.1.1, the motion of a continuum can be determined by the Navier-Stokes equations. Ideally, in the case of a two-phase flow with particles, the motion of the flow and each particle would accurately and separately be resolved[1].

However, practically, the resolution of these systems will depend significantly on the number of particles, the complexity of the flow, and, obviously, on the computational resources available.

For this reason, four approaches have been taken into consideration and are illustrated in figure 3.1.

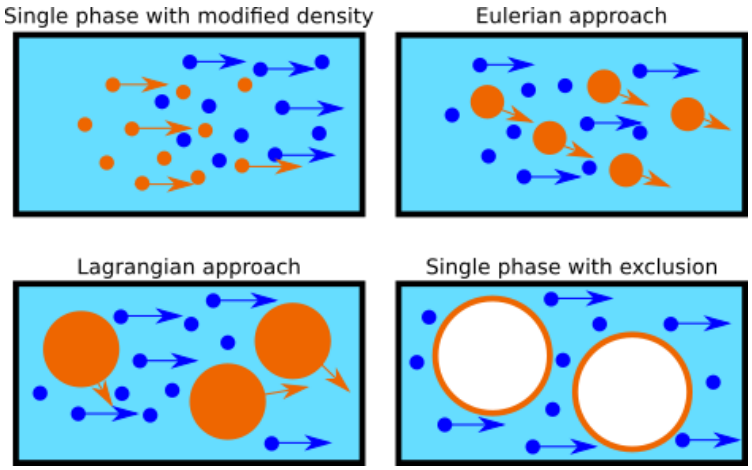


Figure 3.1: Two-phase flow modeling approaches

[1]Note that the same reasoning for dynamics is applicable for thermal properties, although this will not be detailed.

Single phase with modified density

The first approach (top left on figure 3.1) is the easiest to implement since the complete two-phase flow is studied as one phase with combined inherent properties. In other words, considering the fluid's viscosity as an example, one viscosity will be determined, created from the combination of both phases' viscosity.

Although this approach does not isolate each phase's motion (for there is only one mixed-phase), it still provides an idea of the mixture's flow behavior for a computational cost equivalent to a single-phase flow.

This approach is only feasible in the limit of vanishing Stokes numbers.

Eulerian approach

For higher Stokes numbers, the fluid and the solid particles can no more respond similarly to any excitation. Intuitively, the drag induced by the form factor of the larger particles will impact them more, and the particles with higher density will require more momentum to change their trajectories. Any existing external forces (e.g. gravity) will affect them more.

Nevertheless, if, in their respective phase, the particles can be considered statistically homogeneous and isotropic, the two phases will only require two distinct sets of equations to govern their motions. Without diving into the details that will be developed in § 4.1.2, the Navier-Stokes equations make use of 4 equations (one for continuity and three for momentum), while a few additional transport equations will be introduced for the dispersed phase. As a consequence, the computational resources' needs will be increased reasonably.

On the other hand, it would provide a more accurate solution than the first approach by providing the motion for each phase.

This modeling approach is feasible with a $St \ll 1$, simplifying the modeling to two interacting continuum fluids.

In this case, the solid particles phase is also called a pseudo-fluid, and although this approach is relatively fast, it necessitates extra constitutive relations to close the flow equations. After having finished listing the different approaches, section 3.1.4 will develop this point more in detail.

Lagrangian approach

When the Stokes number increases even further, one can no longer study the second phase as a continuum, and each particle's motion has to be studied separately. This approach is named the Lagrangian approach. The top right side of figure 3.1 illustrates it.

Thus, one will study the fluid phase as a continuum. In contrast, the solid particle phase is a dispersed phase, where calculating the flow field (Elsayed and Lacor, 2010) allows for the tracking of each particle's trajectory, tak-

ing both the Newtonian equations of motions and any external forces into account (Niemi, 2012).

This type of modeling approach can be used with $St \gg 1$, and with solid particles that differ in type, size, shape, and velocities but will drastically increase the size of the system that has to be solved for each time step.

Moreover, another critical aspect of this approach is the particle-particle and particle-wall interactions and collisions. As outlined in § 3.1.2, these interactions imply very complex calculations that are not always explicitly studied.

For these reasons, Snider, 2001, reports the process will often restrict the modeling to two dimensions, without taking the continuous phase into account and with the dispersed phase simulated with an order of 2×10^5 particles.

Several techniques exist from which the most popular creates a reduced amount of parcels that group particles by properties to circumvent these limitations. Only these parcels are being tracked, reducing the computational cost.

Intermezzo: Correct denomination

In the literature, each phase can adopt an Eulerian or a Lagrangian approach. Stricto sensu, in a two-phase flow, each phase can be either Eulerian or Lagrangian.

Consequently, the correct denominations for the respective two previous sections are an Eulerian-Eulerian approach and an Eulerian-Lagrangian approach (fluid particles follow an Eulerian approach while the solid particles follow a Lagrangian approach).

A nice representation of the different techniques can be found in figure 3.2, detailed in Ariyaratne et al., 2018.

Single phase with exclusion

For even greater solid particle size, if the study permits it, the second phase can be omitted and replaced by geometrical obstructions (in other words, walls).

The bottom right side of figure 3.1 presents this approach. It has the clear advantage of simplifying the model to one phase. Furthermore, as a matter of fact, this is the most accurate method but unfeasible for large numbers of particles. The particles are "excluded" from the fluid mesh, but that does not mean that no information is available. Indeed, using deforming meshes, sliding meshes, or immersed boundaries particle motion can prove to accurately model the effect of particles on the fluid.

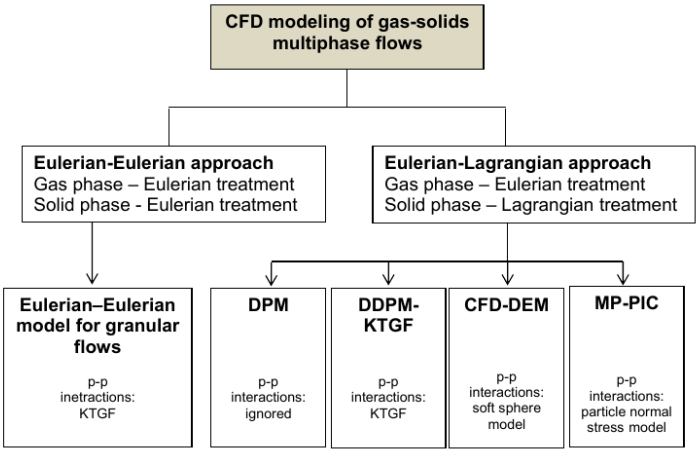


Figure 3.2: Summary of model approaches for gas-solids multiphase flow modeling (From Ariyaratne et al., 2018)

Considered approach

To conclude on the modeling approach, this work will consider a domain with significant dimensions (see § 6.2), where the limited computational resources will constrain the simulation's objective. Effectively, the latter will determine the dispersion pattern of the particles in the atmosphere rather than the precise location of each particle.

For this reason, the favored approach is the Eulerian-Eulerian approach, where both the fluid phase and the solid particles phase will be studied using continuous transport equations. This approach will provide relevant information per phase while still reducing considerably the computational resources that will, even so, remain substantial.

To ensure this approach is suitable, one could start from an academic CFD case, namely, the channel flow, and assume the ABL domain to be a halve channel flow (the top of the ABL domain being half the height of the channel flow). Notice there is a significant difference between these two cases. § 3.2 will address it.

Thus, suppose a channel flow of half-height $1200m$, with air ($\mu = 1.8 \times 10^{-5} kg/ms$ and $\rho_f = 1.2 kg/m^3$ at standard conditions) flowing together with solid particles ($\rho_p = 2620 kg/m^3$ for carbon), the fluid time scale τ_f is given by eq. (3.4), where the friction velocity u_τ can be derived from the skin friction coefficient for turbulent regime in a duct channel flow (cf. correlation of Dean, 1978):

$$C_f \equiv \frac{\tau_w}{\frac{1}{2}\rho \overline{U}^2} = \frac{2u_\tau^2}{\overline{U}^2} = 0.073 Re^{-1/4} \tag{3.7}$$

with $\overline{U}$ the channel mean bulk velocity and Re the associated Reynolds number (§ 4.1.1). Notice that the correlation of Dean was proposed for $6.0 \times 10^3 < Re < 6.0 \times 10^5$.

With these relations (eqs. (3.2 - 3.4), (3.7)), both τ_p and $\overline{U}$ can be found for $St = 1$. The values are presented in Table 3.1.

$d_p(\mu m)$	$\tau_p(\mu s)$	$\overline{U}(m/s)$	Re
10	697.5309	10.98	1.76×10^9
2.5	43.5957	53.55	8.57×10^9
0.1	0.0697	2120.49	3.39×10^{11}

Table 3.1: Flow properties and particle response time for different particle size, at $St = 1$

Table 3.1 shows that already for the large particles $PM10$, the high bulk velocity value together with the considerable domain height will produce a Re that is substantial. Notice that smaller particles amplify the phenomenon. The flow velocity needs to rise even more drastically to conform with the decreasing particle response time τ_p. Since we are primarily interested in particles around the $PM2.5$ limit, it is clear that the Stokes number will not easily exceed 1, and the Eulerian-Eulerian approach is valid.

3.1.4 Details on the solid phase

Before closing this section on multiphase flow, a few precisions should be given concerning the second phase, namely the solid particle phase.

When particles are modeled as a continuum phase, considering an average diameter is convenient. The basic Eulerian approach offers this behavior. However, it does not sufficiently represent physics. To illustrate this, Niemi, 2012 considered a somewhat different case, that is, a circulating fluidized bed, where he observed that large particles remained in suspension near the bottom. At the same time, smaller particles traveled quickly above them[2]. A simulation with only an average diameter taken into account would lead to an erroneous vertical distribution and, consequently, an incorrect motion.

A better hypothesis is to substitute the average diameter by a Particle Size Distribution (PSD). The generalized Navier-Stokes equations require additional equations to achieve this. The following section will introduce the theory behind this.

Population balance equation

When studying a collection of comparable elements, one can use a theory called the population balance approach. This theory delineates the evolution of these elements, here solid particles, that have a distribution of properties

[2]Notice these tiny particles also filled the holes to minimize the space inside the bed.

that can change, both in time and space. The equations associated with this approach are named the Population Balance Equations (PBE) (Ramkrishna, 2000).

In this theory, the particles are evolving in a continuous phase, where external and internal coordinates define each particle's state space. The former depicts the spatial location, while the second is bound to the intrinsic properties of the particles (e.g. size, material, concentration, etc.). Additionally, the PBE supposes the existence of a density function f (e.g. the PSD) that indicates the concentration of particles spatially. Notice that, if integrated, this density function results in the total quantity of particles, N, in the domain Ω (eq. (3.8)). Resolving the PBE generates this function.

$$\int_{\Omega} f d\Omega = N \tag{3.8}$$

The PBE can be derived in a similar way to other balance equations (e.g. continuity, momentum, etc.). Starting from an arbitrary control volume in the considered state space and supposing there is no creation neither destruction of particles, the total amount of particle will remain constant in time (eq. (3.9)).

$$\frac{d}{dt} N = \frac{d}{dt} \int_{\Omega} f d\Omega = 0 \tag{3.9}$$

Supposing a smooth density function, the Reynold's transport theorem (Gonzalez and Stuart, 2015) can be applied to move the time derivative in the integral, implying eq. (3.10):

$$\int_{\Omega} \frac{\partial f}{\partial t} + \nabla \cdot (v\, f) d\Omega = 0 \tag{3.10}$$

with v the particle velocity.

Because the considered control volume Ω was chosen arbitrarily, the integrand has to vanish. By including a source term following Ramkrishna, 2000, eq. (3.11) represents the general differential form for the PBE (Yeoh, 2010). Notice that the source term could include influences such as evaporation, coagulation, fragmentation, aggregation, breakage, and nucleation (Janssens, 2014; Niemi, 2012). The curious reader will find a multitude of models following this general PBE expression in Ferry and Balachandar, 2001; Simonin, 2005; Zaichik et al., 2010.

$$\frac{\partial f(x, \xi, t)}{\partial t} + \nabla \cdot (v(x, \xi, t)\, f(x, \xi, t)) = S(x, \xi, t) \tag{3.11}$$

where both the density function f, the velocity v, and the added source term S are depending on the external coordinates x, the internal coordinates ξ, and the time t.

The PBEs are thus PDEs. Numerical methods such as Monte Carlo methods, class methods (Yeoh, 2010), or Methods Of Moments (MOM) can solve PBEs. The first method, a statistical ensemble approach, is very flexible and accurate but also highly demanding in terms of computation. The second method is a direct method that is very popular although still computationally demanding. The third method is much less demanding while still very accurate. Additionally, this last method is implemented in our code and, as such, will further be explained in the next section.

Method Of Moments (MOM)

Hulburt and Katz proposed the method of moments in 1964 (Hulburt and Katz, 1964). In this method, supposing the concentration distribution is the scalar that we are looking for, the concept is to convert the PBE into a problem of finding the moments of the concentration distribution. As such, one defines the integer moments of a distribution by eq. (3.12).

$$m^k(x,t) = \int_\Omega f(x,\xi,t)\,\xi^k d\xi \tag{3.12}$$

For example, the first moment (i.e. $k = 0$) would lead to expression (3.8).

Thus, when looking at eq. (3.12), the PBE can be converted to its moment form by integrating it after having multiplied it with a power of its internal coordinate. The direct consequence of this integration is a closure problem that will prove to be problematic for any numerical implementation.

To circumvent this, McGraw, 1997 introduced an evolution of the method, called the quadrature method of moments (QMOM). Practically, analogously to numerical integrations, the integral terms from MOM are approached by numerical quadrature (i.e. discretized) following eq. (3.13).

$$m^k(x,t) \approx \sum_{i=0}^{N} w_i\,\xi^k \tag{3.13}$$

with w_i the weights and ξ_i the abscissas of the quadrature.

Various algorithms such as the product-difference algorithm from Gordon, 1968 are available to obtain the values of the moments. From these values, QMOM permits the calculation of the weights and abscissas.

QMOM was successfully implemented in CFD by Marchisio et al., 2003; Marchisio, Vigil, and Fox, 2003a,b. However, it suffered a significant limitation: it could only provide moments that propagate with the same velocity. In other words, it was limited to a single particle velocity field.

Fortunately, the QMOM method, proposed initially by Marchisio and Fox, 2005, continued to be developed, and a modified version called the direct quadrature method of moments emerged. The next section of this multiphase flow segment will detail the latter.

Direct Quadrature Method Of Moments (DQMOM)

Reverting to previous section, the main limitation of QMOM was its inability to handle moments propagating at different velocities. The Direct Quadrature Method Of Moments (DQMOM) addresses it at a lower level. The transport equations are no more originating from tracking the moments themselves. Instead, they are written directly for the weights and abscissas available in the QMOM approximation.

As a consequence, the number of scalar equations is equal to QMOM. Moreover, they are equivalent to monodispersed flows.

In contrast, DQMOM lifted the limitation by allowing each transported weight and abscissa to use different velocities, making it convenient when considering particles of different sizes. Another advantage, compared to QMOM, is its facility to integrate polydisperse capabilities.

Practically, eq. (3.14) defines the PSD through Dirac-functions.

$$f(x, L, t) = \sum_{q=0}^{N} w_q(x, t)\, \delta\left[L - L_q\right] \tag{3.14}$$

with w_q the weights and L_q the abscissas for the quadrature approximation.

Eq. (3.14) can then be substituted into the PBE (eq. (3.11)) to generate, after a few operations, the transport equations for each weight and abscissa following eq. (3.15).

$$\begin{aligned} \frac{\partial w_q}{\partial t} + \nabla \cdot (v_q\, w_q) &= a_q \\ \frac{\partial \mathcal{L}_q}{\partial t} + \nabla \cdot (v_q\, \mathcal{L}_q) &= b_q \end{aligned} \tag{3.15}$$

where $\mathcal{L}_{\text{II}} = L_q w_q$ is the weighted abscissa, a_q and b_q are associated with the source terms and can be found by solving a linear system at each iteration and for each control volume. And where v_q are the velocities, that are now, specific to each weight and abscissa.

These multiphase continuity equations can now be coupled and solved with standard equations for the fluid phase.

Eventually, because:

- the DQMOM was validated in several applications (Fan, Marchisio, and Fox, 2004; Selma, Bannari, and Proulx, 2010; Silva et al., 2010),
- only a few quadrature points (i.e. two to four) are necessary to obtain good results,
- of its multivariate capacity,

this method is considered the most promising method for multiphase flows, both in terms of results and computations. This method was implemented by Janssens, 2014 in our *Coolfluid 3* code, and therefore, it is the method that will be applied in § 6.3.

Despite these excellent characteristics, one should mention two main drawbacks associated with the (DQ)MOM:

- it is sensitive to ill-conditioning (Grosch et al., 2006). Therefore, increasing the accuracy by heavily increasing the number of quadratures may lead to worsening the solution.

- it is possible to locally reach sets of moments that do not match any real PSD (Wright, 2007). The discretization of the advection term in the moment transport equations is at the origin of this issue.

3.2 Atmospheric boundary layer

Having exposed the multiphase properties of the flow, we can introduce the central part of the work, namely, the creation of a system characterized by an atmospheric boundary layer.

3.2.1 Boundary layer meteorology (or micrometeorology)

The first notion is the boundary layer, a layer of fluid directly surrounding a physical surface where an interchange of mass, momentum, and energy occurs. This exchange can induce fluctuations in the flow properties (velocity, density, and temperature).

Secondly, one can divide the Earth's atmosphere into six layers named, from the furthest to the nearest, the exo-, iono-, thermo-, meso-, strato-, and troposphere. Figure 3.3 presents them.

The first six layers are related to ionization, aurora, spatial protections (burning of meteors), and ultraviolet protection (due to the ozone). Nonetheless, their density is negligible compared to the last layer, the troposphere. Because of its higher concentration and the direct contact with the Earth's surface, the troposphere is where most weather phenomena happen. It will therefore be the primary region of interest in meteorology and, consequently, in this work.

Figure 3.4 suggests a cross-section view of the troposphere layer that starts at the Earth's surface and expands to $\sim 9km$ of height at the poles and $\sim 17km$ at the equator. In this layer, the temperature decreases with the altitude by $6.5°C/km$ on average, and its upper limit is defined by the point where the decrease ceases, namely, the tropopause.

Furthermore, the troposphere reveals motions that contain a wide range of scales, from millimeters to thousands of kilometers in the horizontal direction

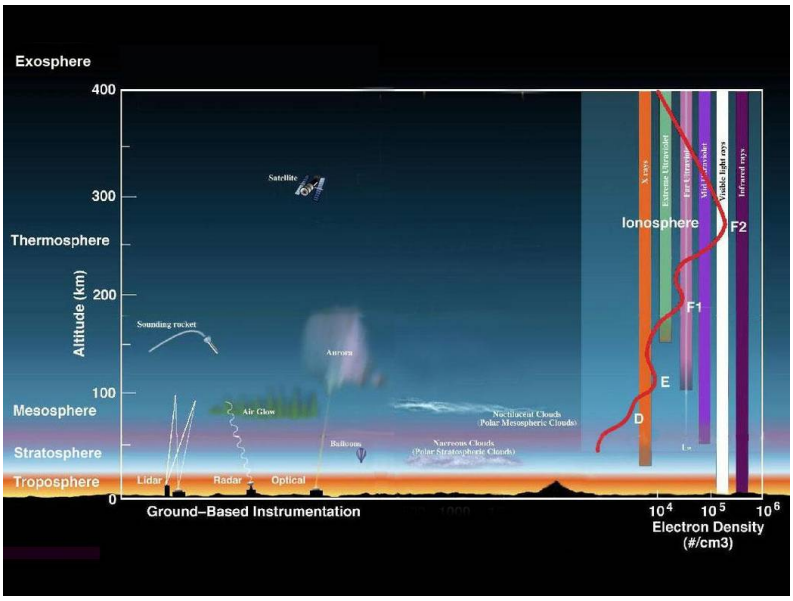


Figure 3.3: Diagram of the atmosphere's layers (From Zell, 2017)

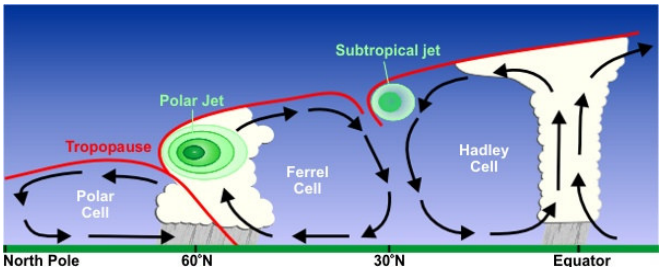


Figure 3.4: Troposphere cross section (From Service, 2008)

and kilometers in the vertical direction. These length scales are associated with time scales that have a span from seconds to years.

In meteorology, one can divide these scales into three branches: the micro-, meso-, and macro-scales (also designated as local-, regional-, and global scales). The micro-scale branch is the branch of interest in this work because it is constrained to phenomena predominated by frictions appearing at the surface of the Earth, also named atmospheric boundary layer (ABL) or more generally planetary boundary layer (PBL).

Thus, the ABL is a boundary layer mainly affected by exchanges between the Earth's surface and the surrounding atmosphere. Figure 3.5 proposes a schematic representation subdividing the ABL into a surface layer and an outer layer:

- The former corresponds to one-tenth of the ABL and contains the roughness layer (or canopy layer) specific to the considered surface of the Earth. The characteristic of this layer will be further developed in § 3.2.2 and taken into account in § 4.3.
- The latter represents the major volume of the ABL. Because of its dimension, the perturbations generated in the surface layer will further develop in the outer layer and create a great amount of mixing. This mixing will be fed by a specificity of the ABL, namely, the variation of the wind direction in function of the height. This effect will be clarified in section 3.2.2.

  Nevertheless, although the complete domain will be considered, only the effect of the surface layer will be taken into account in the first instance.

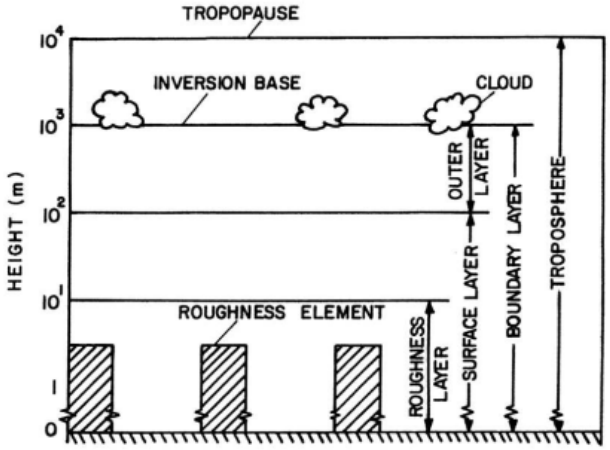


Figure 3.5: Atmospheric Boundary Layer illustration (From Arya, 2001)

Note that figure 3.5 represents a nearly neutral ABL, that is, where each layer has relatively high stability, generating consequently stratified layers (as opposed to mixing layers).

Ordinarily, the ABL thickness varies enormously, depending on the warming, the wind velocity, the topology, and other factors. As an order of magnitude, these daily fluctuations could imply an ABL thickness stretching from $100m$, early in the morning to kilometers in the evening (García Sánchez, 2017; Gorlé et al., 2009).

Because of these wide variations, any air pollutant release at the Earth's surface would eventually contaminate the whole volume, defining a clear and sharp separation between the dirty turbulent air (in the ABL) and the cleaner and streamlined air (above the ABL, in the free-atmosphere).

The split or ABL height is usually referred to as the mixing depth and is commonly associated with an arbitrary cutoff delimited by the cloud base (Arya, 2001).

In this work, one will restrain the complexity to a purely convective phenomenon, producing a nearly neutral ABL condition following fig. 3.5. In reality, such a condition rarely appears, but they do occur during overcast skies or strong geostrophic winds.

3.2.2 ABL specificities

Before diving into the modeling part, one should clarify two important specificities of the ABL:

1. the velocity profiles associated to the roughness layer, and

2. the wind direction in function of the height.

Notice that other factors influence the ABL wind distribution (e.g. pressure and temperature gradients, thermal stratification due to the diurnal cycle, the momentum/heat/moisture exchange between the ABL and the layers above, the presence of clouds). Still, since we considered a purely convective and neutral ABL, these influences will not be discussed.

Roughness layer

The first specificity is associated with the lower part of the ABL, the surface layer, that is, directly connected to the Earth's surface. This surface layer will encounter the largest fluctuations in physical quantities depending on the height. One of the reasons is the topology of the ground, or, more precisely, what is named the roughness layer.

The roughness layer (or canopy layer) is the lowest layer near the surface where rough elements will potentially disturb the flow. The roughness layer's thickness is characterized by a specific roughness height z_0 that can vary from nearly insignificant, for stripped land, to the height of the most outstanding buildings in suburban or urban terrain. In between, a rural or a vegetated region will produce a roughness layer with moderate elevations. In the literature, a roughness parameter (or height or even length) z_0 was introduced as a geometric average roughness that will be used later, in eq. (3.20). Typical values for z_0 are $0.23m$, $0.33m$, and $2.47m$ respectively associated with a rural, suburban, and urban terrain (Vasaturo et al., 2018).

Figure 3.6 presents an illustration of different roughness layers.

As sketched in figure 3.6, the variation in roughness suggests a modification in the considered wind velocity profiles. The three following sections will develop the possible profiles in more detail and briefly compare them.

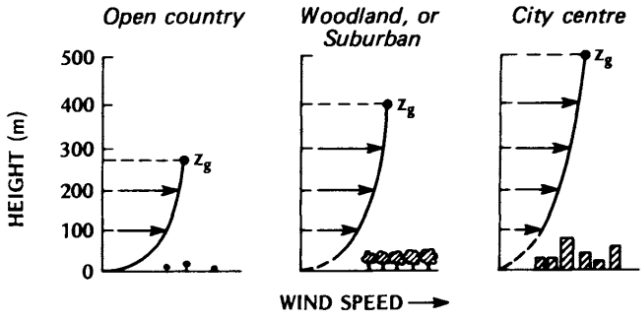


Figure 3.6: Roughness illustrations (From Oke, 1987)

Roughness layer: Power-law profile

A common methodology to model an ABL is to assimilate it to an academic case, that is, the study of a half channel flow (i.e. flow between two infinitely long parallel plates, or plane Poiseuille flow). In other words, one could consider a flat-plate boundary layer on the ground and the middle of a channel flow at the top of the domain to simulate an ABL.

In this case, the velocity distribution would follow approximatively a power-law expression given by:

$$\frac{U}{U_h} = \left(\frac{z}{h}\right)^m \tag{3.16}$$

for different z height, with h, the half-channel depth (or boundary layer thickness), U_h the velocity at that height, and the stability coefficient $m = 1/7$ for a smooth surface, as reported by Prandtl. Because, for an ABL, the relation is valid until a reference height (smaller than the half-channel depth h), U_h and h are often replaced by the velocity U_r given at a reference height r.

In accordance with Prandtl‘s discovery, Izumi and Caughey verified the expression for the various surface's roughness and the various ABL stability condition presented in figure 3.7.

Figure 3.7 reveals that, independently of the considered roughness and the flow condition, the observed wind velocity follows a power-law profile. The only difference resides in the value of the exponent m, varying between ~ 0.1 for smooth surfaces and ~ 0.4 for urban regions, under near-neutral flow conditions; and approaching 1.0 for conditions increasing in stability.

Unfortunately, these observations, validating the power-law velocity profile in the surface layer, do not provide a result that is satisfactory when considering the turbulent momentum flux near the surface.

Effectively, near the surface (in the constant stress layer), the momentum flux is assumed constant with the height. As a consequence, by applying similarity hypothesis, the power-law velocity profile can be used to express a

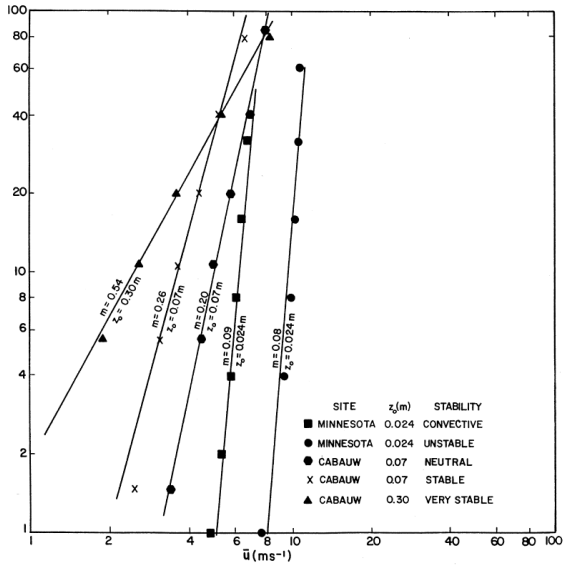


Figure 3.7: Wind velocity power-Law profiles for various roughness z_0 and ABL stability conditions, in function of the height. (From Izumi and Caughey, 1976)

power-law eddy viscosity ν_m distribution in function of the height, following expression (3.17):

$$\frac{\nu_m}{\nu_{mr}} = \left(\frac{z}{z_r}\right)^{(1-m)} \tag{3.17}$$

Eqs. (3.16) and (3.17) are often referred to as the conjugate power laws in the atmospheric theories. When these two equations are used outside the constant stress layer, the coefficient $(1 - m)$ becomes too restrictive and impacts negatively the profile (Arya, 2001).

Roughness layer: Log-law profile

Instead of using the presented power-law profile, another slightly different approach exists, which is more grounded theoretically and physically, especially for a neutral ABL, and follows a logarithmic profile law or log-law profile.

To facilitate the explanation, consider a neutral surface layer, horizontally homogeneous, with no additional complexity brought by a viscous layer or a roughness layer and with no external forces. The velocity distribution can be reduced to an expression (3.18) depending solely on the height z (not necessarily on the ground but rather in its vicinity), and the ratio between

the wall shear stress τ_0 and the fluid density ρ_f (also known as the kinematic momentum flux):

$$\frac{\partial U}{\partial z} = f(z, \tau_0/\rho_f) \tag{3.18}$$

In expression (3.18), tacitly, it is assumed that the wall shear stress τ_0 that establishes the velocity gradients in the surface layer contains the influence of both the surface roughness and the horizontal pressure gradients (induced by the geostrophic winds).

Thus, eq. (3.18) is function of two characteristic scales, namely, the velocity scale $u_* \equiv (\tau_0/\rho_f)^{1/2}$ and the length scale z. These can lead, after dimensional analysis, to the dimensionless wind shear similarity relation:

$$\left(\frac{z}{u_*}\right)\left(\frac{\partial U}{\partial z}\right) = constant = \frac{1}{\kappa} \tag{3.19}$$

with κ, the von Karman's constant. The latter was found empirically and is considered a constant equal to 0.401 for all wall surfaces, although its exact value could not be reached at a better precision than 5% (Högström, 1985).

From integrating eq. (3.19) in relation with z, and after defining the mixing length as $l = \kappa z$ and the eddy viscosity $\nu_m = \kappa z u_*$, one will find the established log-law profile (Oke, 1987):

$$\frac{U}{u_*} = \left(\frac{1}{\kappa}\right) \ln\left(\frac{z}{z_0}\right) \tag{3.20}$$

where z_0 is the integration constant that was presented in the beginning of section 3.2.2.

Figure 3.8 shows a perfect correlation between the observations performed on the plains of Wangara, located in the Hay region of Australia, where a neutral surface layer could be measured, and the log-law wind profiles. This correlation is supported by the linear variations of ν_m and l with the height that is consistent with both theory and physics (in contrast with the non linear behavior for the power-law profile).

Power-law vs log-law

To further enrich the discussion, the two profiles can be compared by plotting the reference height z_r version of eq. (3.16) together with an adapted version of eq. (3.20) given by:

$$\frac{U}{U_r} = 1 + \frac{\ln(z/z_r)}{\ln(z_r/z_0)} \tag{3.21}$$

Figure 3.9 presents both profiles, where the power-law profiles are given for varying values of m, while the variation is in z_r/z_0 for the log-law profiles.

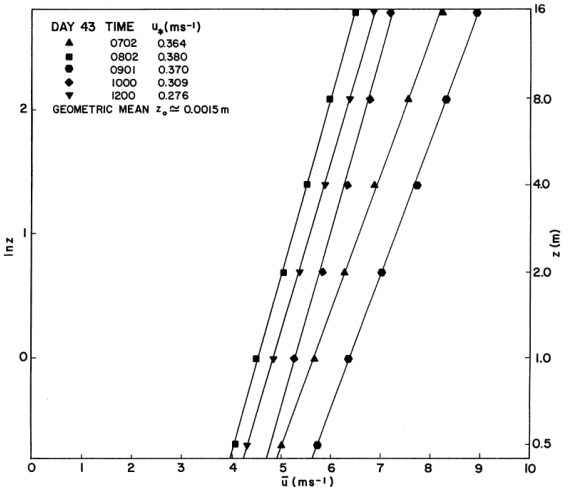


Figure 3.8: Observations of wind profiles in the neutral surface layer in Wangara compared to the corresponding log-law profiles (From Clarke et al., 1971)

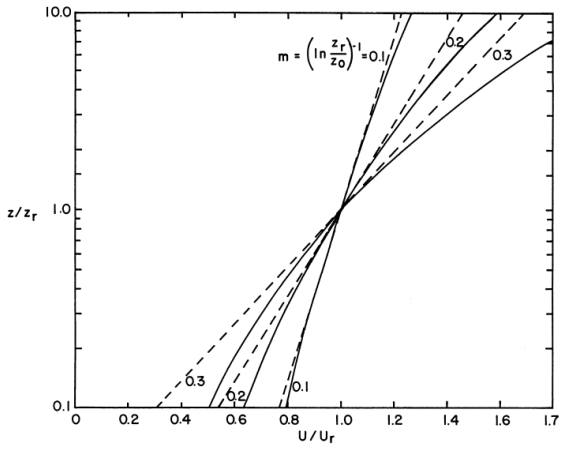


Figure 3.9: Power-law (–) and log-law (– – –) comparison for hypothetical wind profiles (From Arya, 2001)

Since these two profiles are different, the equivalence between these two parameters can be approached by equalling the velocity gradients, at the reference height z_r following eq. (3.22):

$$m = \frac{d(\ln U)}{d(\ln z)} = \frac{z}{U}\left.\frac{\partial U}{\partial z}\right|_{z=z_r} = \left(\frac{z_r}{z_0}\right)^{-1} \tag{3.22}$$

By observing fig. 3.9, one can clearly see the deviation taken by the power-law profiles relative to the log-law profiles, the further z travels from the reference height z_r.

This concludes that, although both profiles perform reasonably in the surface layer, the log-law profile follows the observations better, especially in neutral conditions, and is more consistent with the physics, making it a suitable choice. As a matter of fact, this approach is commonly used nowadays and will further be developed in the *Modeling* Part (§ 4.3)

Wind direction

An additional specificity of ABL is related to the change in wind direction with the height.

In the classical channel flow case (introduced in section 3.2.2), a unidirectional flow is considered. This approach is perfectly acceptable in the surface layer but becomes unrealistic when considering the entire ABL.

In the ABL, because of the significant scales, together with the variations of pressure (and temperature), the rotation of the Earth (i.e. Coriolis effect) will influence how the direction of the geostrophic wind evolves with the altitude. Note that the geostrophic winds are defined as the winds that would occur only with these two influences (hence, with local acceleration, but with no advection nor friction).

To elaborate, the geostrophic winds components U_g and V_g, respectively in the longitudinal (x) and lateral (y) directions, can be defined in terms of pressure gradients through eq. (3.23):

$$U_g = -\frac{1}{\rho_f f}\frac{\partial P}{\partial y} \quad ; \quad V_g = \frac{1}{\rho_f f}\frac{\partial P}{\partial x} \tag{3.23}$$

where f is the Coriolis factor defined in function of the rotational velocity of the earth Ω and the specific latitude ϕ, in eq. (3.24):

$$f = 2\Omega \sin\phi \tag{3.24}$$

Thus, in the outer layer, the Coriolis effects will induce deviations of the geostrophic wind direction, that can be expressed by the velocity-defect law (Monin and Yaglom, 1971) given in eq. (3.25):

$$\begin{aligned} (U - U_g)/u_* &= F_u(fz/u_*) \\ (V - V_g)/u_* &= F_v(fz/u_*) \end{aligned} \tag{3.25}$$

where $F_{u,v}()$ are functions based on the similarity hypothesis for barotropic neutral ABL, originally proposed by Kazanski and Monin, 1961.

Accordingly, the influence of the Coriolis effect was measured and validated experimentally by Kaimal in 1976 while conducting an experiment over flat terrain in Minnesota for a convective overland (figure 3.10).

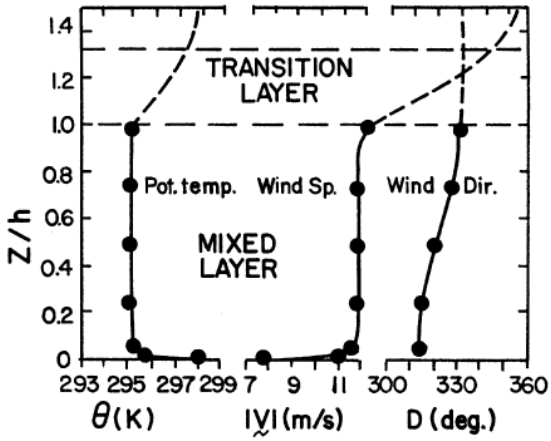


Figure 3.10: Temperature, velocity and direction of the wind in function of the height, for a convective overland condition. (From Kaimal et al., 1976)

Nevertheless, in this work, as previously stated, the change in direction will not be considered to reduce the complexity.

As a special remark, in 1968, during a World Health Organization symposium on urban climates and building climatology held at Brussels, Oke presented an interesting observation where, in unstable conditions, the wind direction from the surface layer progressively shifts cyclonically (to the left in the northern hemisphere, right in the southern hemisphere) when the roughness layer changes from smooth to rough. These shifts can reach 10 to 20° for a passage from rural to urban areas (Oke, 1974).

This final remark shows that these two essential characteristics of ABL, namely the roughness layer and the Coriolis effect, can significantly influence the wind distribution.

3.3 Synthesis on related physical aspects

Chapter 3 presented two physical aspects influencing the dispersion of particles in the atmosphere.

To define the interaction between the solid particles and the air, dimensionless numbers were defined which allowed to find what model would be appropriate, taking both the physics and the resources into account. As such, we decided to develop a model that would use an Eulerian-Eulerian approach, applying the DQMOM to the solid particle phase. This would ease the numerical development and distinguish the two phases while still allowing the computation with limited computational resources. Eventually,

this approach also allows to focus on one phase and add the second phase independently.

In the second part of this chapter, the atmospheric boundary layer was introduced through a meteorological approach to understand physics better. Two main specificities, namely, the roughness layer and the variation in wind direction with the height, were further detailed to introduce the study that needs to be performed to bring us closer to the final goal.

From this last part, one needs to keep three notions in mind:

- we studied two wind profiles in the literature: the power-law and the log-law profiles. The log-law is the most commonly used thanks to its deeper connection to theory and physics, near the surface but also until the outer layer, in contrast with the power-law profile. It is also the log-law profile that we will exploit in this work.

- The average roughness height (z_0) is a parameter used to express the roughness of the terrain. Since it influences the turbulences occurring near the ground and, therefore, on the potential dispersion of particles (or more globally on all quantum), this parameter is intrinsic to any ABL representation. By way of illustration, three values of z_0, for three different terrains, are displayed: $0.23m$ (rural), $0.33m$ (suburban), and $2.47m$ (urban).

- The Coriolis effect and the associated wind deviation with the height is a peculiar characteristic of ABL, but in first instance, although necessary in general, this effect will not be implemented.

Having defined what is to model and what profile will be studied, the main goal of this work remains to develop code inside the software *Coolfluid 3* (Quintino et al., 2012) that will enable us to:

1. Simulate an airflow in an open field, taking the size of the atmospheric boundary layer into consideration.

2. Ascertain the validity of the modeling and its stabilizations without jeopardizing the computational resources.

Nevertheless, before diving into the different particularities of the code, the first sections of the next part will propose a description of the various computational fluid dynamics (CFD) simulation types.

Part II

Modeling

Chapter 4

Numerical modeling

After having defined this work's motivation and physical aspects, this chapter will focus on the primary modeling techniques.

First, an overview of the principal modeling directions will be outlined (§ 4.1).

Then, the methodology used in the *Coolfluid 3* code will be detailed, together with its advantages but also weaknesses. For the latter, the available solutions (e.g. stabilization method) will be specified (§ 4.2).

The third section will extend the prior ABL knowledge (§ 3.2) by introducing the modeling strategy (§ 4.3) that will be further developed in Part III.

Eventually, a few words will be written concerning the influence of the domain's dimension (§ 4.4).

All these elements are essential to the proper comprehension and analysis of the future results but not sufficient. As a consequence, this chapter will be followed by a chapter deepening the turbulence modeling (§ 5), concluding the *Modeling* Part II.

4.1 From accurate but slow, to fast but simplified, to a realistic compromise

Modeling physical behaviors can be defined as trying to explain and reproduce, mathematically, physical phenomena.

To help with the modeling, first, a non-dimensional parameter, the Reynolds number, will be defined. This parameter is derived from the dimensional analysis and will help in characterizing some fluid properties.

The second subsection will establish the needed governing or state equations. They represent the mathematical model that fits the physical processes and will be solved in their discrete forms.

Lastly, three approaches (§ 4.1.3, § 4.1.4, § 4.1.5) used to solve them will be described.

4.1.1 Reynolds number

The *Reynolds number*, symbolized by Re, is a non-dimensional parameter that was introduced by Sir Osborne Reynolds in 1883. It is defined as the

ratio between the inertial and the viscous forces occurring in the considered fluid (Bottin, 2015).

It plays a fundamental role by indicating the nature of the flow, namely laminar or turbulent, in a broader sense, by identifying the transition between these two regimes.

$$\begin{aligned} F_{inertia} &:= mU\frac{dU}{dx} = m\frac{U^2}{L} = \rho V\frac{U^2}{L} = \rho L^2U^2 \quad with \quad V = L^3 \\ F_{viscous} &:= \tau S = \mu\frac{U}{L}L^2 = \mu UL \qquad\qquad\qquad with \quad S = L^2 \\ &\rightarrow Re := \frac{F_{inertia}}{F_{viscous}} = \frac{\rho UL}{\mu} \end{aligned} \tag{4.1}$$

where U, L, ρ, μ are respectively the characteristic velocity, the length, the density and the dynamic viscosity of the flow. τ is the shear stress related to the dynamic viscosity by:

$$\tau := \mu\frac{\partial U}{\partial y} \tag{4.2}$$

where x is aligned with the flow direction while y is normal to the boundary.

Thus, if the inertial forces prevail on the viscous forces (typically when Re is more significant than 10^5 in an open stream), the flow will become turbulent.

4.1.2 Navier-Stokes equations

The Navier-Stokes equations provide a complete description of the flow (both, its motion and all the turbulent structures contained in the fluid). If the density and the viscosity of the fluid remain constant (i.e., incompressible and homogeneous), the equations are:

$$\begin{aligned} \partial_i u &= 0 \\ \partial_t u_j + u_i\partial_i u_j &= 2\nu\partial_i S_{ij} - \frac{1}{\rho}\partial_j P + f_i \quad with \quad j = 1,2,3 \end{aligned} \tag{4.3}$$

With for the intrinsic properties of the fluid: ν the kinematic viscosity and ρ the fluid's density. The other variables are depending on the considered system: the velocity field $u(x,t)$ expressed both in space and time, the rate-of-strain tensor S_{ij} corresponding to the friction between the particles (John, 2014), the pressure P and, eventually, any external force per unit mass f acting on the fluid.

Note that the expression for the rate-of-strain tensor is:

$$S_{ij} = \frac{1}{2}\left(\partial_i u_j + \partial_j u_i\right) \tag{4.4}$$

Thus, these eqs. (4.3) are respectively based on the conservation of mass and the conservation of momentum.

By looking at these Navier-Stokes equations, it should be stressed that the numerical resolution will present three difficulties: the amount of information in the velocity field, the non-linearity of the convective term $u_i\partial_i u_j$ and the coupling between the pressure and the velocity (John, 2014).

To facilitate the interpretation of the momentum equation (from eq. (4.3)), one can reformulate it with the dimensionless Reynolds number (eq. (4.1)):

$$\partial_t u_j + u_i\partial_i u_j = 2\frac{1}{Re}\partial_i S_{ij} - \frac{1}{\rho}\partial_j P + f_i \tag{4.5}$$

From expression (4.5), a solution that is predominantly viscous ($Re \to 0$) will monotonously reach a stationary solution. For increased Re ($\to 10^2 \sim 10^3$), the influence of the non-linear convection term ($u_i\partial_i u_j$) will increase, causing a flow that will head toward a periodic solution. For even greater Re ($\to \infty$), where inertia prevails on viscous forces, even more periodicities aggregate to eventually produce a flow in a turbulent state, where disorder is predominant.

An elegant manner to illustrate this progression is to look at the generation of vortices behind a cylinder, in function of Re. Figure 4.1 displays the Q-criterion, which is an indicator of the relative rotational fluctuation in the flow compared to its translational fluctuation. It is defined by:

$$Q_{ij} = \frac{1}{2}\left(|\Omega_{ij}|^2 - |S_{ij}|^2\right) \tag{4.6}$$

where S_{ij} and Ω_{ij} are respectively the strain (eq. (4.4)) and vorticity tensors that will further be detailed in section 5.2.2.

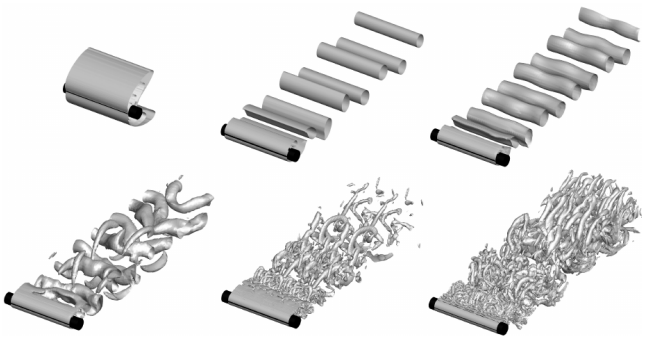


Figure 4.1: Q contours for increasing Re. (From Banyai, 2016)

Referring to the introduction of this section, although the Navier-Stokes equations provide a complete description of the flow, the complexity brought by turbulence clarifies why only a few accessible cases have an analytical

solution. To solve the vast majority of flows, the only currently available choice is to discretize and use numerical models.

The following sections will present three approaches that are primarily acting on the numerical size of the system to be solved.

4.1.3 Direct Numerical Simulation (DNS)

The direct numerical simulation (DNS) takes all turbulent scales into account. To enable an accurate representation of all pertinent scales, one has to divide the studied domain into cells (i.e. the grid resolution) that are fine enough to capture the smallest eddies (e.g. $\sim 10^{-3}m$). In contrast, the dimension of the considered geometry has to be wide enough to contain the largest structures. Note that the latter is met if the correlation between two points placed in the stream- and spanwise directions vanishes in the first half of the computational domain (Trofimova et al., 2009).

As a consequence, the number of cells increases drastically. In fact, according to Kolmogorov, 1991, supposing isotropic turbulence, the number of mesh points necessary to capture the smallest eddies is proportional to $Re^{9/4}$. Pope, 2001 presented an analogous comparison where the increase is a polynomial function of the Reynolds number that can, for a turbulent flow (e.g. $Re \approx 10^6$), lead to a grid with 10^{18} ($= Re^3$) mesh points in space-time (Hoffman and Johnson, 2006). Spalart estimated that half a century will still be needed for the computational resources to be able to solve a typical engineering case in turbulence (Spalart, 2000).

In the case of an ABL (with a domain's unit dimension of $\sim 10^3m$), due to the wide range of turbulent scales, resolving all time and length scales to obtain an actual turbulent flow is numerically impossible. Indeed, the large amount of computational cells required to detect the small eddies, together with the short numerical time steps essential to capture the fastest turbulent structures, results in a substantial computational cost.

Because the computational cost of this type of simulation is significantly high, because these simulations are only reachable for small Reynolds numbers and eventually because, in our work, an open-field simulation is considered (in other words, both the Reynolds number and the spatial dimension are large), this type of simulation is not affordable.

To circumvent this limitation, an option is to transfer the complexity to the modeling (Tennekes and Lumley, 1976). This can be carried out by reducing the number of information to handle through averaging. Three techniques are widely used:

- Ensemble averaging in time: all physical quantities are averaged in time.
- Spatial filtering: in this case, a low-pass filter, preserving continuity, is applied to the flow, reducing the number of mesh points involved in the computation.

- A blending of the two previous techniques.

The following two subsections will introduce the two first approaches. The third one, the blending, although widely studied in the literature, will not be expanded[1].

4.1.4 Reynolds Averaged Navier-Stokes (RANS) equations

The first technique uses the so-called Reynolds Averaged Navier-Stokes (RANS) equations. In this approach, physical values are averaged in time (also called ensemble averaging in time):

$$\overline{\phi} = \lim_{T\to\infty} \frac{1}{T - T_0} \int_{T_0}^{T} \phi(T_0 + t)dt \tag{4.7}$$

By averaging in time, the fluctuations disappear, leaving a flow field that is less sensitive to the spatial configuration. As a consequence, the mesh resolution needed to capture these averaged quantities is also less restrictive. As a result, this method could rapidly be applied to turbulent models.

In concrete terms, each physical quantity ($\phi(t)$) can be expressed by eq. (4.8), as the sum of a time averaged ($\overline{\phi}$) that follows eq. (4.7), and a fluctuating ($\phi'(t)$) component that is illustrated by figure 4.2:

$$\phi(t) = \overline{\phi} + \phi'(t) \tag{4.8}$$

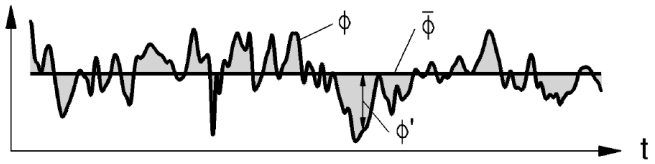


Figure 4.2: Illustration of a signal ϕ composed of an average $\overline{\phi}$ and a fluctuating $\phi'(t)$ part. (From Banyai, 2016)

By averaging in time the Navier-Stokes eqs. (4.3) and applying identities (4.9),

$$\overline{\nabla\phi(t)} = \nabla\overline{\phi}, \quad \overline{\phi'(t)} = 0, \quad \overline{\nabla\phi'(t)} = 0, \quad \overline{\frac{\partial\phi'(t)}{\partial t}} = 0 \tag{4.9}$$

one can express the RANS equations (4.10). Note that, here, to ease the reading, the vectorial notation is used.

[1]Nonetheless, by seeking the benefits of the two first approaches while reducing their drawbacks, there is undoubtedly much interest in this last direction (Chaouat, 2017; Hoarau et al., 2019; Sagaut, 2006).

$$\nabla \overline{\vec{u}} = 0$$
$$\partial_t \overline{\vec{u}} + \left(\overline{\vec{u}} \cdot \nabla\right) \overline{\vec{u}} = \nu \nabla^2 \overline{\vec{u}} - \frac{1}{\rho} \nabla \overline{P} - \nabla \overline{\overline{\tau'}} \quad with \quad \overline{\overline{\tau'}} = \overline{u_i' u_j'} \tag{4.10}$$

When looking at eqs. (4.10), all terms are averaged in time except $\overline{\overline{\tau'}}$ that expresses the fluctuations produced by the convection. $\overline{\overline{\tau'}}$, named the *Reynolds stress tensor*, is a tensor with six degrees of freedom (in 3D) that requires six more equations to solve the system. To solve the system, a *closure* needs to be found. The latter should not only solve the mathematical equations but also consider the physics of the considered flow. The most widely used formulation is the *Boussinesq approximation* (4.11), proposed in 1877 (Boussinesq, 1877):

$$\overline{\overline{\tau'}} = -2\nu_t \overline{\overline{S}} + \frac{2}{3} \overline{\overline{\delta_{ij}}} k \quad with \quad k = \frac{1}{2} \overline{u_i' u_i'} \tag{4.11}$$

where $\overline{\overline{S}}$ is defined by eq. (4.4), k is defined as the turbulent kinetic energy, and where ν_t, the turbulent eddy viscosity, is the only remaining parameter.

Concerning the last unknown, ν_t, several models exist to define it. Since they will not be used in this work, only a brief outline will be provided. The linear models can be distributed in three categories:

- the algebraic models: are not based on the solution of any other equation (exception for the Johnson-King model) but directly calculated from flow variables. These are robust models for high-speed flows but do not take the history of the flow into account. Examples are the Baldwin-Lomax (Baldwin and Lomax, 1978), Cebeci-Smith (Smith and Cebeci, 1967), and Johnson-King (Johnson and King, 1985) models.

- the one-equation models: resolve an extra transport equation, typically the turbulent kinetic energy equation. The initial version is from Prandtl (Wilcox, 2006) while the most common is the Spalart-Allmaras version (Spalart and Allmaras, 1992). Other more modern versions are the Baldwin-Barth (Baldwin and Barth, 1990), Rahman-Argawal-Siikonen (Rahman, Siikonen, and Agarwal, 2011), WA (Wray-Argawal) (Han, Wray, and Agarwal, 2017), and the Shuai-Argawal (Shuai and Agarwal, 2020) turbulence models.

- the two-equations models: describe the turbulences in the flow by two transport equations (one equation associated with the turbulent kinetic energy k, and another for its dissipation ϵ or rate of dissipation ω). Its main advantages are connected to its ability to provide a flow that preserves its "memory" (e.g. the convection and diffusivity of the

turbulent energy) at a reasonable cost (i.e. resources vs. flow resolution). As such, these models are the most popular in the industry. The most used model for nearly a half-century was the $k-\epsilon$ (Jones and Launder, 1972) and its variants. A conceptually different model, the $k-\omega$ model, was proposed in 1988, and its optimized version was popularized by Menter (Menter, 1993) in 1993 under the name SST $k-\omega$ (with SST standing for Shear Stress Transport).

Besides linear turbulence models, non-linear models are also developed, associating the turbulent eddy viscosity to the velocity, via a non-linear relation. It supports some anisotropy near the wall. The author refers to the v^2-f and the cubic $k-\epsilon$ models for more details (Durbin, 1995; Popovac and Hanjalic, 2007).

To synthesize, the RANS approach uses time-averaged flow properties that do not require extra-fine meshes. As a result, they produce a solution with limited resources both in time and computation. The drawback is related to the loss in resolution due to the averaging. This also makes these models less suitable for particle dispersion since turbulent scales can influence particle behavior, which would require additional modeling.

4.1.5 Large Eddy Simulation (LES)

In between these two approaches, there is a statistical method that does not use time-averaged values and mitigates the constraints of a pure DNS simulation. This is achieved by limiting the computation to the dynamics of the large-scale motions while representing the smaller scales through simple models (Pope, 2001). This approach is categorized in the so-called Large Eddy Simulations (LES) and will be developed more in detail in § 5.2. Thanks to the increasing available computational resources, the industry increased its use widely during the last decades.

In the same region of interest, there is another approach that is substantially distinct from the LES approach. This different approach is called the Variational Multi-Scale method (VMS). The latter will be developed in § 5.1 and is mainly different from the LES in the reasoning and the technique used to solve the closure problem inherent to the filtering used in these types of simulation.

Although small, this section contains the methodologies that will be used in this work and, for this reason, a specific chapter will be dedicated to them (§ 5) after having defined first the method used to compute the flow properties inside the domain, namely the Finite Element Method (FEM) (§ 4.2), and second, the implementation of the model that will be used near the wall surface (§ 4.3).

4.2 Discretization with the finite element method

4.2.1 Path to discretization

To solve a fluids problem using CFD, two fundamental facets are required:

- the physical modeling
- the numerics

The physical modeling will require finding mathematical equations that can close the governing equations. These will need to agree, as much as possible, with the observed physics. In this matter, the first stones have been laid in chapter 3 and will further be investigated in chapter 4.3.

The second facet, numerics, is dedicated to finding algebraic relations that will faithfully represent the physics of the governing equations. In other words, since solving Partial Differential Equations (PDE) (e.g the Navier-Stokes equations) is not always analytically possible, making them correspond with algebraic relations will allow finding a solution.

Although simply enounced, this process is neither unique nor exact. The PDE often describes a conservation principle, where no "artificial" loss is expected. Discretizing these principles by means of a numerical algorithm will induce approximations. Moreover, each differential term of these governing equations contains a physical meaning that could slightly be altered by the chosen discretization scheme. For this reason, but also because of stability and convergence motivations, several schemes were developed (upwind, central, quick, etc.). However, it remains a challenge to derive a consistent mathematical discretization.

This section aims to present and detail the Finite Element Method (FEM) choice for the discretization of the Navier-Stokes equations.

Since it is not the most prominent choice in CFD, it was decided to introduce it by describing briefly two other available methods (§ 4.2.2): The Finite Difference Method (FDM) and the Finite Volume Method (FVM).

Note that although other methods are available (e.g. the spectral methods, the weighted residual method, or even the time discretization methods), they will not be described since the purpose of this section is not to propose a full CFD discretization course in the context of flow fields but to situate and focus on the FEM in the CFD environment.

After the contextualization, FEM will further be detailed (§ 4.2.3 - 4.2.5) by elaborating on the stabilization method used. As will be seen in chapter 6, numerical stabilization should not be neglected when considering the simulation of ABL.

4.2.2 Finite difference and finite volume methods

As implied but not expressed in the previous section, the governing equations in a physical domain describe a continuous phenomenon in a continuous

domain. Since, for most situations, the governing equations can not be analytically resolved, they have to be approached by algebraic relations. These relations are bound to discrete quantities. Therefore, the domain also needs to be divided into a grid (also called mesh) with discrete points (i.e grid points, mesh points, or even nodes).

Finite Difference Method (FDM)

Historically, the oldest method is the FDM (Sadrehaghighi, 2021). It was designed by Thom (Thom and Apelt, 1961) in 1961, under the name "the method of square", to resolve nonlinear hydrodynamic equations.

The FDM is a mathematical tool that can be used to approximate PDEs by replacing them, algebraically, through finite difference equations. The computational domain is typically sliced in quads for a 2D domain (hexahedras in 3D) and the finite difference equations are linked to its grid points (see figure 4.3).

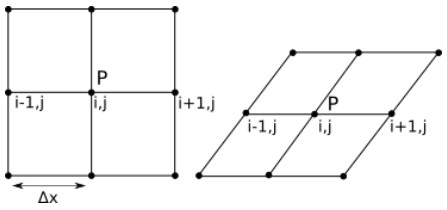


Figure 4.3: FDM grid patterns.

For this purpose, Taylor's series expansions are applied. This is exemplified, in eqs. (4.12), by three distinct expressions for the first derivative in the x-direction of the physical quantity (here, the velocity) and one for the second derivative.

$$\partial_x u_i \approx \underbrace{\frac{u_{i+1} - u_i}{\Delta x}}_{\text{Forward}} \approx \underbrace{\frac{u_i - u_{i-1}}{\Delta x}}_{\text{Backward}} \approx \underbrace{\frac{u_{i+1} - u_{i-1}}{2\Delta x}}_{\text{Central}}$$
$$\partial_{xx} u_i \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2} \tag{4.12}$$

where Δx is the distance between two grid points (e.g. i and $i+1$). As noticeable, it takes its origin from the definition of the derivative (eq. (4.13)):

$$\frac{\partial u(x,y)}{\partial x} = \lim_{\Delta x \to 0} \frac{u(x + \Delta x, y) - u(x,y)}{\Delta x} \tag{4.13}$$

Although old, this method is still often used thanks to its straightforward implementation and light cost for a simple case. The drawbacks are associated with truncation error (cf. Taylor's series expansion), consistency, stability, convergence, and ultimately conservation. In-depth studies are provided by Anderson, Tannehill, and Pletcher, 1984; Anderson, 2003.

Finite Volume Method (FVM)

Because of the conservation limitation of the FDM and due to the increasing demand for complex flows study, an incentive to develop other discretization methods in the early 1980s resulted in the massive growth and use of the FVM in fluid mechanics, where both the advective and the diffusive terms inside a PDE could be handled.

In the FVM, analogously to the FDM (and other discretization methods), the objective is to replace the PDE with a set of algebraic equations. It can be described in three steps:

- The computational domain is divided into finite volumes (also called cells).
- The PDEs are integrated over each given cell, producing balance equations. In these balance equations, specific volume integrals can be converted into surface integrals (following the divergence theorem), implying the conservation of flux through the surfaces (cf. fig. 4.4). Subsequently, the surface (and volume) integrals are substituted by discrete algebraic relations, following specific integration quadratures.

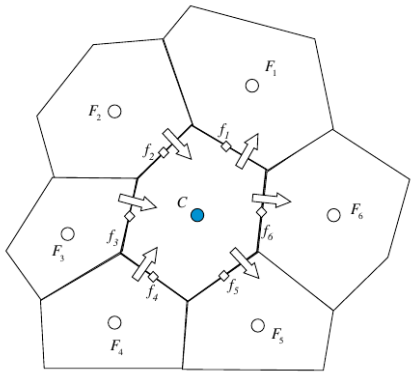


Figure 4.4: FVM flux illustrations. (From Moukalled, Mangani, and Darwish, 2015)

- Inside each cell, but also at no-storage locations (e.g. grid points), a chosen interpolation profile will offer an approximated value for each quantity, between the discrete quantities produced in the first step.

Certainly, the choice of the integration quadrature's order as well as the interpolation profile will have an impact on the accuracy of the produced solution.

Similar to FDM, the solutions are discrete, but, in contrast with FDM, the field variables are usually stored in the center of each cell and not on the grid points (also called nodes).

The main advantages of FVM is that:

- it satisfies the conservation laws (even in coarse mesh), enabling it to compute and evaluate the flux at the boundaries of each cell (also possible for nonlinear issues or discontinuous solutions in highly compressible flows), and
- it does not require a structured grid (allowing studies of more complex geometries while avoiding any internal mesh conversion). Figure 4.5 illustrates the latter.

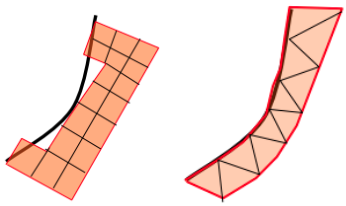


Figure 4.5: Geometry resolution for FDM (left) and FVM (right) (From Ebrahimi, 2010)

It should also be noted that the FVM resolution in regions of interest can be increased by refining the mesh. However, the interpolation profiles are limited to a low-order accuracy (relative to FEM in the next section). Its drawbacks are associated with increased complexity in code structure, together with a loss in computational efficiency (relative to FDM).

4.2.3 Finite Element Method (FEM)

Brief history

In previous section on FDM, it was stated that FDM was the oldest discretization method of the three presented. Technically, it depends on what is considered the starting date for the FEM (Felippa, 2001; Mohite, 2001).

FEM originates from the analysis of mechanical structures. Precisely, its ancestor, the Matrix Structure Analysis (MSA), was developed in 1934 to enable humans (in the pre-computer era) to solve discrete aeroelastic models. The motivation and development were linked to the aircraft industry associated with the two World Wars.

In 1959, Turner, working for the aircraft builder Boeing, proposed an evolution of MSA, named Direct Stiffness Method (DSM). Thanks to its computer implementation, integrated into structural and continuum models,

the DSM supplanted other methods (e.g. Force Methods) in the aerospace industry.

From 1962, the DSM was connected to an early idea of subdivisions in interconnected elements, where a structural component could be idealized by finite elements. From that moment on, DSM gave birth to FEM. Subsequently, FEM integrated new variational approximations schemes that demonstrated that such type of conforming DSM is actually a form of the Rayleigh-Ritz method (i.e. direct numerical method to approximate eigenvalue) based on the minimum potential energy principle. This last discovery strengthens the mathematical foundations of the young FEM.

Thanks to the increasing interest, in 1965, FEM was applied to non-structural problems, higher-order elements were developed to increase the resolution performance ratio, and the mathematical foundations of FEM were consolidated in the monograph of Strang and Fix, 1975.

As of today, although FEM remains unrivaled in the mechanical, vibratory, and structural analysis, most industrial CFD codes mainly compose with FVM by default due to both the FVM handling of the advection term and to the complexity associated with FEM implementation. Nevertheless, FEM remains present in CFD, especially in multiphysics cases where the mathematical foundations of FEM play a substantial role and in cases with complex geometries where high-order elements can save computational resources (Hughes and Tezduyar, 1984).

Strengths and weaknesses

After the brief history and the comparison with two popular spatial discretization methodologies used in CFD, having a better understanding of how FEM works will provide a better insight on what are its forces but also on how to circumvent its weaknesses.

Returning to section 3.1.3, two kinetic descriptions of the flow field were defined to simulate the flow: the Lagrangian approach, where each particle was traced; and the Eulerian approach, where the accent was set on the flow direction as a continuum, from a fixed referential.

FEM, albeit introduced in the previous section, was intensively used in mechanical structure analysis, where deformations of structures are progressive and can be discretized spatially by nodes that follow the structure that is being deformed. In other words, the material points stick to their respective grid points during the complete motion (fig. 4.6). (Mathematically, it means the material derivative does not contain any convective term, reducing it to an ordinary time derivative). As a result, there is no convective effect in the Lagrangian approach.

In figure 4.6, R_X is the domain associated with the original referential X, R_x is the deformed domain after motion, ϕ is the transformation, considering the velocity v, applied to the original referential.

These kinds of problems are typically governed by elliptic or parabolic

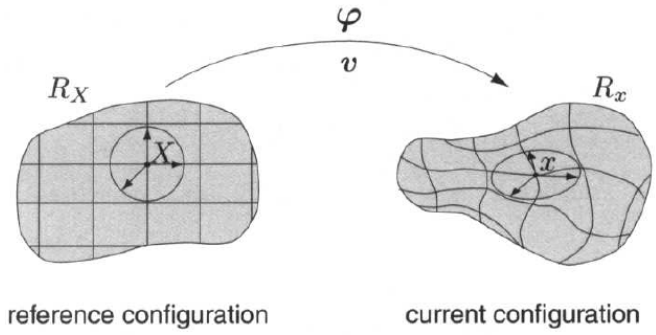


Figure 4.6: Lagrangian referential motion. (From Donea and Huerta, 2003)

PDE and present a symmetric stiffness matrix (Donea and Huerta, 2003). In fact, thanks to both the Lagrangian context and the symmetric stiffness matrix, FEM can efficiently be applied using what is called the Galerkin FEM. In the Galerkin FEM, the difference between the FEM approximation and the exact solution is minimized (following a weighted residuals method) according to the energy norm (Strang and Fix, 1975).

These types of algorithms are advantageous when tracking free surfaces or interfaces between materials and are therefore optimal to describe mechanical structures (specifically elasto- and visco-plastic deformations).

However, when the computational domain undergoes significant distortions without frequent repositioning of the different nodes (i.e. remeshing), Galerkin FEM will not succeed in following the deformations.

In fluid mechanics, fluid motion does encounter relatively large displacement, making the Lagrangian formulation much less attractive. In contrast, the Eulerian approach[2], where the computational mesh does not evolve (or get transformed) with the flow motion, can easily capture the fluid motion: It is the flow that moves, relative to the fixed grid.

The corollary is that the particles moving relative to the computational grid will create a convective effect that leads to an unsymmetric stiffness matrix. Unfortunately, Galerkin FEM was created to solve symmetric matrices.

Fortunately, a solution was developed (§ 4.2.4) precisely to solve this issue. Before presenting it, three steps still need to be addressed:

- The first step is to define necessary mathematical tools and terminol-

[2]Another formulation exists, the ALE approach, that was developed to combine the advantages of both the Lagrangian and the Eulerian approaches. Depending on the specificity of the investigated case, the mesh displacement will be activated (Lagrangian) or not (Eulerian). This approach is elegant but requires an extra mesh displacement algorithm as well as an error estimator activating the switch. This approach exists in FEM but was not used in this work. Great details are provided by Giuliani, 1982; Huerta et al., 1999.

ogy.

- The second step is to address the original Galerkin FEM more in detail with the new tools.
- Then, the third step will be to demonstrate the weakness with a simple example but also to glimpse at possible mitigations.

Eventually, we will have all the information to present the solution. Note that although each step will need its section, the author decided to skip some details that can be found in any complete course on FEM (Donea and Huerta, 2003; Johnson, 1987; Trofimova et al., 2009; Zienkiewicz, 2000; Zienkiewicz, Taylor, and Zhu, 2013).

Mathematical terminology and tools

In FEM, considering a spatial domain $\Omega \subset \mathbb{R}^{n_{sd}}$ ($n_{sd} = 1, 2, 3$ the space dimension) with a piecewise continuous boundary Γ ($= d\Omega$), the spatial discretization is based on discretizing the weak integral form[3] of the PDE of interest (e.g. the Navier-Stokes equations, the Poisson equation, etc.). Because of these integral formulations, the required functions should be square integrable over the considered domain Ω, denoted by $\mathcal{L}_2(\Omega)$. This space is a Sobolev space characterized by a standard inner product and a norm given by eqs. (4.14).

$$(u, v) = \int_\Omega uv \, d\Omega, \quad \& \quad ||v||_0 = (v, v)^{1/2} \tag{4.14}$$

In fact, these capabilities should be extended to the derivatives of all these functions, up to order k. This new space can be defined as a Hilbert space $\mathcal{H}^k(\Omega)$ (eq. 4.15)[4].

$$\mathcal{H}^k(\Omega) = \left\{ u \in \mathcal{L}_2(\Omega) \,\middle|\, \frac{\partial^{|\alpha|} u}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \cdots \partial x_{n_{sd}}^{\alpha_{n_{sd}}}} \in \mathcal{L}_2(\Omega) \quad \forall |\alpha| \leq k \right\} \tag{4.15}$$

where k is a positive integer, and $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_{n_{sd}}) \in \mathbb{N}^{n_{sd}}$ is a n-tuple of integers.

For the sake of clarity, in FEM, three spaces are of interest, $\mathcal{L}_2$ (already defined), $\mathcal{H}^1$, and $\mathcal{H}_0^1$ (subspace of $\mathcal{H}^1$ associated with the boundary Γ) that can be derived from eq. (4.15):

$$\begin{aligned} \mathcal{H}^1(\Omega) &= \left\{ v \in \mathcal{L}_2(\Omega) \,\middle|\, \frac{\partial v}{\partial x_i} \in \mathcal{L}_2(\Omega) \quad with\ i = 1, \cdots, n_{sd} \right\} \\ \mathcal{H}_0^1(\Omega) &= \left\{ v \in \mathcal{H}^1(\Omega) \,|\, v = 0 \quad on\ \Gamma \right\} \end{aligned} \tag{4.16}$$

[3] also named variational form.

[4] A Hilbert space is a linear space that includes an inner product definition characterized by convergent Cauchy sequences.

where the inner product and norm are given by eqs.(4.17).

$$(u,v)_1 = \int_\Omega \left(uv + \sum_{i=1}^{n_{sd}} \frac{\partial u}{\partial x_i}\frac{\partial v}{\partial x_i} \right) d\Omega \quad \& \quad ||u||_1 = (u,u)_1^{1/2} \tag{4.17}$$

Remark 4.1. *$\mathcal{L}_2(\Omega)$ is in fact a $\mathcal{H}^0(\Omega)$ space.*

Remark 4.2. *These spaces are valid for both scalar- and vector-valued functions. In the case of vector-valued functions, $\mathcal{H}^k(\Omega)$ will be rewritten $[\mathcal{H}^k(\Omega)]^m$, with m the number of vector-components.*

These space definitions enables us to define two classes of functions required for FEM:

- the *test* (also called *weighting*) functions, and
- the *trial* (also called *admissible*) solutions.

The first class, designated by $\mathcal{V}$, is a $\mathcal{H}^1_{\Gamma_D}$ space (i.e. $\mathcal{H}^1$ vanishing on the boundary where Dirichlet condition applies):

$$\mathcal{V} = \left\{ w \in \mathcal{H}^1(\Omega) | w = 0 \quad on\ \Gamma_D \right\} \equiv \mathcal{H}^1_{\Gamma_D}(\Omega) \tag{4.18}$$

The second class, designated by $\mathcal{S}$, is similar to $\mathcal{V}$ except that it does fulfill the Dirichlet conditions on Γ_D:

$$\mathcal{S} = \left\{ u \in \mathcal{H}^1(\Omega) | u = u_D \quad on\ \Gamma_D \right\} \equiv \mathcal{V} + \{\bar{u}_D\} \tag{4.19}$$

Remark 4.3. *If the boundary conditions are homogeneous ($u_D = 0$), the two classes concur: $\mathcal{V} = \mathcal{S} = \mathcal{H}^1_0(\Omega)$.*

Remark 4.4. *After subdividing[5] the domain into subdomains (i.e. finite element spaces) following eq. (4.20), the $\mathcal{S}$ and $\mathcal{V}$ containing an infinite amount of solutions, are respectively approximated by $\mathcal{S}^h$ and $\mathcal{V}^h$, where h is a characteristic mesh size in order that $diam(\Omega^e) \leq h$ for all elements.*

$$\Omega = \cup_{e=1}^{n_{el}} \Omega^e \quad with \quad \Omega^e \cap \Omega^f = \phi \quad for\ e \neq f \tag{4.20}$$

Subsequent to this space terminology, two tools can be defined: the bilinear and the trilinear forms. These will enable to write the FEM weak form in its concise integral form.

- the bilinear forms:

$$\begin{aligned} a(u,v) &= \int_\Omega \nabla u : \nabla v \, d\Omega \quad \forall u, v \in \mathcal{H}^1(\Omega), \\ b(v,q) &= -\int_\Omega q \nabla \cdot v \, d\Omega \quad \forall v \in \mathcal{H}^1(\Omega) \quad and \quad q \in \mathcal{L}_2(\Omega) \end{aligned} \tag{4.21}$$

  with $a(u,v)$ and $b(v,q)$ respectively for two $\mathcal{H}^1$ functions, or for a more restrictive Hilbert case (with $\mathcal{H}^1$ and $\mathcal{L}_2$ functions).

[5] also called triangulation

- the trilinear form:

$$c(v;w,u) = \int_\Omega w \cdot (v \cdot \nabla) u \, d\Omega \quad \forall u, v, w \in \mathcal{H}^1(\Omega) \tag{4.22}$$

containing subsequent symbols:

$$[\nabla u]_{ij} = \frac{\partial u_i}{\partial x_j}, \qquad for \quad i = 1, \cdots, m \quad and \quad j = 1, \cdots, n_{sd}$$
$$\nabla u : \nabla v = \sum_{i=1}^{m} \sum_{j=1}^{n_{sd}} \frac{\partial u_i}{\partial x_j} \frac{\partial v_i}{\partial x_j}, \qquad \& \qquad w \cdot (v \cdot \nabla) u = \sum_{i=1}^{m} \sum_{j=1}^{n_{sd}} w_i v_j \frac{\partial u_i}{\partial x_j} \tag{4.23}$$

Galerkin FEM formulation

Having defined all needed mathematical features, the original Galerkin FEM formulation will be applied to the simplest (not Navier-Stokes) governing equation that contains a convection term (important for the oscillation phenomenon). It will ease the manipulation.

We chose the steady convective-diffusive governing equation of a scalar quantity $u = u(x)$ to illustrate its principle. Eq. (4.24) offers the strong form:

$$\begin{aligned} a \cdot \nabla u - \nabla \cdot (\nu \nabla u) &= s \quad in\ \Omega, \\ u &= u_D \quad on\ \Gamma_D, \\ n \cdot \nu \nabla u = \nu \frac{\partial u}{\partial n} &= h \quad on\ \Gamma_N \end{aligned} \tag{4.24}$$

where the considered domain $\Omega \in \mathbb{R}^{n_{sd}}$ is enclosed: partially by an essential (Dirichlet) boundary, Γ_D, with a determined u_D value ; and partially by a natural (Neuman) boundary, Γ_N, with an established normal diffusive flux h. The remaining parameters are the convective velocity $a(x)$, the diffusivity coefficient $\nu > 0$, and the volumetric source $s(x)$.

The following phase is to find the variational (i.e. weak) form of the equation. This is made possible by defining a $\mathcal{S}$ space where all u are fulfilling the Dirichlet condition u_D, while the $\mathcal{V}$ space provides weighting functions w vanishing on the boundary Γ_D (cf. eqs (4.18), (4.19)). The weak form is then given by:

$$u \in \mathcal{S} \,| \quad \int_\Omega w(a \cdot \nabla u) d\Omega - \int_\Omega w \nabla \cdot (\nu \nabla u) d\Omega = \int_\Omega w\, s \, d\Omega \quad \forall w \in \mathcal{V} \tag{4.25}$$

Then, applying the Gauss-Ostrogradsky divergence theorem to the diffusion term together with eq. (4.19) transform eq. (4.25) to:

$$\int_\Omega w(a \cdot \nabla u) d\Omega + \int_\Omega \nabla w \cdot (\nu \nabla u) d\Omega = \int_\Omega w\, s \, d\Omega + \int_{\Gamma_N} w\, h \, d\Gamma \quad \forall w \in \mathcal{V} \tag{4.26}$$

where the natural boundary condition is inherently applied.

Finally, the bilinear and trilinear forms (eqs. (4.21), (4.22)) can be rewritten for this convective-diffusion problem as:

$$\begin{aligned} a(w,u) &= \int_\Omega \nabla w \cdot (\nu \nabla u) d\Omega, \qquad (w,s) = \int_\Omega ws\, d\Omega, \\ c(a;w,u) &= \int_\Omega w(a \cdot \nabla u) d\Omega, \qquad (w,h)_{\Gamma_N} = \int_{\Gamma_N} wh\, d\Gamma \end{aligned} \tag{4.27}$$

By applying them, the concise weak form becomes:

$$a(w,u) + c(a;w,u) = (w,s) + (w,h)_{\Gamma_N} \quad \forall w \in \mathcal{V} \tag{4.28}$$

Or even better, after dividing the domain into a finite number of sub-domains, and creating the sub-spaces $\mathcal{S}^h$ and $\mathcal{V}^h$, we obtain:

$$a(w^h,u^h) + c(a;w^h,u^h) = (w^h,s) + (w^h,h)_{\Gamma_N} \quad \forall w^h \in \mathcal{V} \tag{4.29}$$

that will yield to the Galerkin FEM approximation of u^h (eq. (4.30)).

However, before expressing it, a small remark has to be stressed.

Remark 4.5. *One should understand that the Dirichlet condition will imply a difference between the number of nodal points n_{np} of the finite element mesh and the number of unknowns (or equations) n_{eq} of the system. If we define $\eta = \{1, 2, \cdots, n_{np}\}$ as the global node number in the mesh, and η_D as the nodes, included in η, where Dirichlet condition applies, then $\eta \backslash \eta_D = n_{eq}$.*

With this comprehension, the trial solution u^h can now be expressed:

$$u^h(x) = \sum_{A \in \eta \backslash \eta_D} N_A(x) u_A + \sum_{A \in \eta_D} N_A(x) u_D(x_A) \tag{4.30}$$

where u_D is the Dirichlet value, u_A is the nodal unknown, and the shape function N_A, bound to node A, is intimately linked to the the arbitrary weighting function w^h by:

$$w^h \in \mathcal{V}^h := \underset{A \in \eta \backslash \eta_D}{span} \{N_A\} \tag{4.31}$$

Eventually, inserting eq. (4.30) into eq. (4.29) will generate the discrete weak expression:

$$\begin{aligned} \sum_{B \in \eta \backslash \eta_D} [a(N_A,N_B) + c(a;N_A,N_B)] u_B = & \qquad \forall A \in \eta \backslash \eta_D \\ (N_A,s) + (N_A,h)_{\Gamma_N} - \sum_{B \in \eta_D} & [a(N_A,N_B) + c(a;N_A,N_B)] u_D(x_B) \end{aligned} \tag{4.32}$$

that can be reduced to the matrix form:

$$(D + C)u = f \tag{4.33}$$

where the convection matrix C, the diffusion matrix D, and the source term vector s (on the right hand side (RHS)) are achieved by assembling, topologically, the contribution of each element ($a, b, \cdots$ with $1 \leq \{a, b\} \leq n_{en}$) locally, by means of the A^e local[6] assembly operator (Hughes, Mazzei, and Jansen, 2000). Their respective share are given by eq. (4.34):

$$\begin{aligned} D = A^e D^e \quad with \quad D^e_{ab} &= \int_{\Omega^e} \nabla N_a \cdot \nu \nabla N_b d\Omega \\ C = A^e C^e \quad with \quad C^e_{ab} &= \int_{\Omega^e} N_a (a \cdot \nabla N_b) d\Omega \\ f = A^e f^e \quad with \quad f^e_{ab} &= (N_a, s)_{\Omega^e} + (N_a, h)_{\partial\Omega^e \cap \Gamma_N} \\ &\quad - \sum_{b=1}^{n_{en}} [a(N_a, N_b)_{\Omega^e} + c(a; N_a, N_b)_{\Omega^e}]\, u^e_{D_b} \end{aligned} \tag{4.34}$$

with $u^e_{D_b} = u_D(x^e_b)$ following Dirichlet condition.

Remark 4.6. *Accordingly, an excellent characteristic of the FEM is that unstructured meshes are inherently handled*[7]*.*

Illustration of the Galerkin FEM oscillations

Having defined the Galerkin FEM formulation for the steady convective-diffusive governing equation, it can be used to:

- demonstrate the oscillation behavior that takes place,
- why it happens and perhaps most important,
- how to counter it.

To simplify even further the case, the advection a and diffusion ν coefficients will be assumed constant ; a homogeneous Dirichlet boundary condition is imposed on both sides ; the source term s depends solely on x:

$$\begin{aligned} a u_x - \nu u_{xx} = s(x) \qquad & in\ ]0, L[\\ u = 0 \qquad & at\ x = 0\ and\ x = L \end{aligned} \tag{4.35}$$

After integration by parts, the weak form and its compact form are given by:

$$\begin{aligned} \int_0^L (w\, a\, u_x + w_x\, \nu\, u_x) dx &= \int_0^L w\, s\, dx \\ a(w, u) + c(a; w, u) &= (w, s) \end{aligned} \tag{4.36}$$

[6] local: element referential ; as opposed to global: domain referential

[7] A mesh is unstructured if the number of elements connected to each node can differ.

For the discretization, consider a $1D$ uniform mesh composed of linear elements of size h, where the node numbering is successive (for the sake of simplicity). n_{eq} is the amount of interior nodes of the spatial discretization $\eta = \{1, \cdots, n_{eq}, n_{eq}+1, n_{np}\}$ containing $\eta_D = \{1, n_{np}\}$.

By incorporating the trial (eq. (4.30)) and weighting (eq. (4.31)) functions, eq. (4.36) can be rewritten, for interior node A (with $A = 2, \cdots, n_{eq}+1$):

$$\int_0^L \sum_{b=2}^{n_{eq}+1} \left(aN_A \frac{\partial N_B}{\partial x} + \nu \frac{\partial N_A}{\partial x} \frac{\partial N_B}{\partial x} \right) u_B \, dx = \int_0^L N_A s \, dx \tag{4.37}$$

To solve eq. 4.37, a few more details can be added:

- Within a $1D$ linear element (figure 4.7), composed of 2 nodes ($n_{en} = 2$), with local number 1 and 2, the shape functions have following expressions:

$$N_1(\xi) = \frac{1}{2}(1-\xi) \qquad N_2(\xi) = \frac{1}{2}(1+\xi) \tag{4.38}$$

  with ξ the normalized coordinate (contained between -1 and 1).

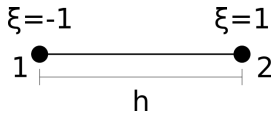


Figure 4.7: Linear element in 1D.

- In the local referential, any point inside the element will have an expression for $x(\xi)$ and for $u(\xi)$ that will be a linear combination of its respective values in 1 and 2.

- Furthermore, the global distance of this linear element of size h can be expressed in the local referential by:

$$dx = \frac{1}{2}(x_2 - x_1)d\xi = \frac{h}{2}d\xi \tag{4.39}$$

- The latter implies that:

$$\frac{\partial N_b}{\partial x} = \frac{\partial N_b}{\partial \xi}\frac{\partial \xi}{\partial x} = \frac{2}{h}\frac{\partial N_b}{\partial \xi} \quad \textit{with } b = 1, 2 \tag{4.40}$$

Remark 4.7. *The case proposed here is applied to a 1D linear element for convenience. However, it was generalized in the literature (Zienkiewicz, Taylor, and Zhu, 2013).*

All these details contribute to evaluating locally the convection C^e and diffusion D^e matrices:

$$C^e = a\int_{\Omega^e}\begin{pmatrix} N_1\frac{\partial N_1}{\partial x} & N_1\frac{\partial N_2}{\partial x} \\ N_2\frac{\partial N_1}{\partial x} & N_2\frac{\partial N_2}{\partial x}\end{pmatrix}dx = \frac{a}{2}\begin{pmatrix}-1 & +1\\ -1 & +1\end{pmatrix}$$
$$D^e = \nu\int_{\Omega^e}\begin{pmatrix} \frac{\partial N_1}{\partial x}\frac{\partial N_1}{\partial x} & \frac{\partial N_1}{\partial x}\frac{\partial N_2}{\partial x} \\ \frac{\partial N_2}{\partial x}\frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial x}\frac{\partial N_2}{\partial x}\end{pmatrix}dx = \frac{\nu}{h}\begin{pmatrix}+1 & -1\\ -1 & +1\end{pmatrix} \tag{4.41}$$

with $\Omega^e = [x_e, x_{e+1}]$ for $e = 1, \cdots, n_{el}$[8].

The same process can be performed for the source term s, assuming the linear combination $s(\xi) = N_1(\xi)s_1 + N_2(\xi)s_2$, and applying eq. (4.34).

$$f^e = \int_{\Omega^e}\{N_1(N_1s_1 + N_2s_2), N_2(N_1s_1 + N_2s_2)\}^T\,dx \tag{4.42}$$

Eventually, after having assembled the contribution of each element connected to one interior node j, the Galerkin FEM in node j will generate following discrete expression:

$$a\left(\frac{u_{j+1} - u_{j-1}}{2h}\right) - \nu\left(\frac{u_{j+1} - 2u_j + u_{j-1}}{h^2}\right) = \frac{1}{6}(s_{j-1} + 4s_j + s_{j+1}) \tag{4.43}$$

Remark 4.8. *The informed reader will have observed an apparent similarity between the Galerkin method based on linear elements and the central difference method (developed in the FDM). However, they also differ substantially in the way they handle the source term. Briefly, the former takes advantage of a weighted average (cf. RHS of eq. (4.43)), while the latter utilizes a punctual and local source value.*

Eq. (4.43) is an important equation in the examination of the oscillatory behavior of the Galerkin FEM. But first, let us define a relation showing the influence of the convective effect relative to the diffusive part. This relation is named the Péclet number:

$$P_e = \frac{a\,h}{2\nu} \tag{4.44}$$

Inserting eq. (4.44) into eq.(4.43) results in eq. (4.45):

$$\frac{a}{2h}\left(\frac{P_e - 1}{P_e}u_{j+1} + \frac{2}{P_e}u_j - \frac{P_e + 1}{P_e}u_{j-1}\right) = \frac{1}{6}(s_{j-1} + 4s_j + s_{j+1}) \tag{4.45}$$

The next step before analyzing is to simplify even more the equation by:

- setting the source term, the domain dimension, and the advection to unity ($s = L = a = 1$), and

[8] here, $n_{el} = n_{eq} + 1 = n_{np} - 1$

- selecting three values for P_e ($P_e = 0.24, 0.9, 5$) to evaluate the influence of the convection, relative to the diffusion.

Remark 4.9. *Forcing a uniform source term has the advantageous side effect to avoid any potential truncation errors caused by discretizing the source (cf. RHS of eq. (4.45)). This enables us to precisely being able to impute any truncation error to the discretization of the left-hand side (LHS), namely the convection-diffusion influence.*

Since there exists an analytical and exact solution (Donea and Huerta, 2003) given by eq. (4.46) for this equation, both the analytical and the approximated solutions for u can be drawn. The latter is estimated on a uniform mesh with 10 elements.

$$u(x) = \frac{1}{a}\left(x - \frac{1-\exp(\gamma x)}{1-\exp\gamma}\right) \quad with\ \gamma = \frac{a}{\nu} \tag{4.46}$$

The resulting profiles are given in figure 4.8.

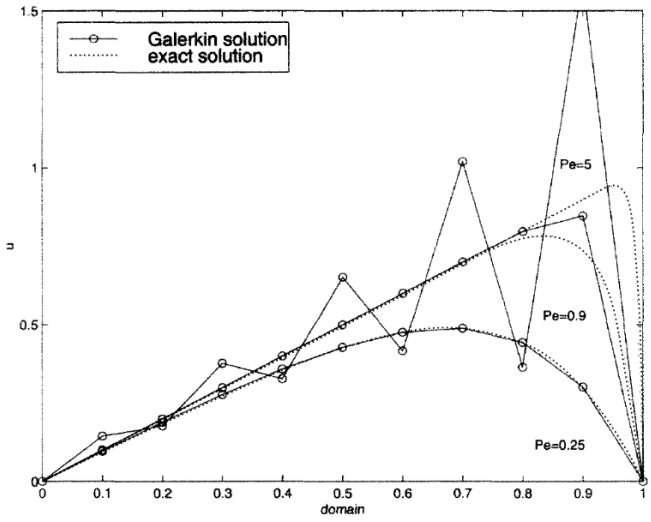


Figure 4.8: Exact $(-)$ and Galerkin $(\cdots)$ solutions for the convection-diffusion equation for 10 linear elements on a uniform mesh, with $L = s = 1$. (From Donea and Huerta, 2003)

The unambiguous observation is that the higher P_e is (i.e. the higher the domination of the non-symmetric convection relative to the diffusive effect), the greater the oscillation encountered by the Galerkin approximation will be. In other words, the more convective the flow is, the less correct the Galerkin FEM will be, and the more these numerical (and non-physical) oscillations will pollute the expected solution.

This observation is critical.

Now that the oscillations for convective flow using Galerkin FEM were demonstrated, the next step is to understand why it happens.

In this respect, eq. (4.45) can be remodeled in a homogeneous form so that the left and right side of node j are isolated:

$$(1 + P_e)(u_j - u_{j-1}) = (1 - P_e)(u_{j+1} - u_j) \tag{4.47}$$

Eq. (4.47) reveals that, for high Péclet number (i.e. $P_e > 1$), the slope on the left and right sides of node j are in opposition.

This first analysis can be reinforced by the exact solution of this linear difference equation (Isaacson and Keller, 1994) that has the following characteristic equation:

$$(1 - P_e)\lambda^2 - 2\lambda + (1 + P_e) = 0 \tag{4.48}$$

that is solved by $\lambda_1 = 1$ and $\lambda_2 = (1 + P_e)/(1 - P_e)$. As a consequence, the solution to eq. (4.47) is:

$$u_j = C_1 + C_2 \left(\frac{1 + P_e}{1 - P_e}\right)^j \tag{4.49}$$

where C_1 and C_2 are provided by the boundary conditions.

Eq. (4.49) confirms that a high Péclet number affects the Galerkin FEM in such a manner that its solution has no choice but to be oscillating.

Eventually, to find out how to remedy these oscillations, the LHS structure of eq. (4.45) will be mimicked to produce a scheme generating an exact solution at each node x_j, for any P_e. Then, comparing these two expressions will enable us to determine how to modify the Galerkin scheme in order to reduce these oscillations.

Thus, the mimicked scheme should have following structure:

$$\alpha_1 u_{j-1} + \alpha_2 u_j + \alpha_3 u_{j+1} = 1 \tag{4.50}$$

where each α_i must be replaced by a value according to the exact solution (eq. (4.46)).

Eq. (4.46) provides an expression for u_{j-1}, u_j, *and* u_{j+1}:

$$\begin{cases} u_{j-1} & = \frac{1}{a}\left(x_j - h - \frac{1-\exp(\gamma x_j)\exp(-2P_e)}{1-\exp\gamma}\right), \\ u_j & = \frac{1}{a}\left(x_j - \frac{1-\exp(\gamma x_j)}{1-\exp\gamma}\right), \\ u_{j+1} & = \frac{1}{a}\left(x_j + h - \frac{1-\exp(\gamma x_j)\exp(2P_e)}{1-\exp\gamma}\right) \end{cases} \tag{4.51}$$

These expressions can be inserted into eq. (4.50) to produce a system of three equations with three unknowns:

$$\begin{cases} \alpha_1 + \alpha_2 + \alpha_3 = 0 \\ -\alpha_1 + \alpha_3 = a/h \\ \alpha_1 \exp(-2P_e) + \alpha_2 + \alpha_3 \exp(2P_e) = 0 \end{cases} \tag{4.52}$$

that will deliver eqs. (4.53):

$$\begin{cases} \alpha_1 = -a(1+\coth P_e)/(2h) \\ \alpha_2 = a(\coth P_e)/h \\ \alpha_3 = a(1-\coth P_e)/(2h) \end{cases} \tag{4.53}$$

By inserting these values into eq. (4.50), we obtain the intended exact scheme:

$$\frac{a}{2h}\left[(1-\coth P_e)u_{j+1} + (2\coth P_e)u_j - (1+\coth P_e)u_{j-1}\right] = 1 \tag{4.54}$$

that is similar but not equal to the original Galerkin scheme (eq. (4.45)).

Reorganizing eq. (4.54) in two specific ways will enable us to better understand how to reduce these spurious numerical oscillations:

- Firstly, by rearranging it analogously to the original scheme (eq. (4.45)):

$$a\frac{u_{j+1}-u_{j-1}}{2h} - (\nu+\bar{\nu})\frac{u_{j+1}-2u_j+u_{j-1}}{h^2} = 1 \tag{4.55}$$

  where $\bar{\nu}$ is an artificial numerical diffusion that depends solely on the element size h and the characteristics of the transport equation. It is given by:

$$\bar{\nu} = \beta\frac{ah}{2} = \beta\nu P_e \quad with\ \beta = \coth P_e - \frac{1}{P_e} \tag{4.56}$$

- Secondly, by incorporating β:

$$\frac{1-\beta}{2}\left(a\frac{u_{j+1}-u_j}{h}\right) + \frac{1+\beta}{2}\left(a\frac{u_j-u_{j-1}}{h}\right) - \nu\frac{u_{j+1}-2u_j+u_{j-1}}{h^2} = 1 \tag{4.57}$$

  In this version, the discretized convection term corresponds to a weighted average of the fluxes of the solution that occurs on the right and left sides of node j. As a matter of fact, the centered scheme is not appearing in this case.

Returning to the subject of our work, the considered case is highly convective; the classical Galerkin FEM can thus not be used as-is. A sort of stabilization has to be applied to reduce these spurious numerical oscillations. To summarize, according to the previous paragraph, two potential directions for stabilization methods are available[9]:

- replacing the centered discretization scheme by an upwind scheme, or
- insert an additional numerical diffusive term.

This brings us to the next section dedicated to stabilization methods.

[9] order inversed because the first that will be developed is the upwind scheme.

4.2.4 From early stabilization to Streamline-Upwind Petrov-Galerkin (SUPG)

As introduced in the previous section, replacing the centered scheme with an upwind scheme should counteract the numerical oscillations introduced by the classical Galerkin FEM. Although it is not what was performed in our work, it will still be explained since, eventually, the reader will find out both directions to help understand one another. Furthermore, more important, it will help clarify the main complication of this work: the spurious numerical oscillations (cf. results part (§ 6.8)). In a few chapters, all the foundations will be set to identify and try to counter them.

From centered to upwind scheme

Returning to the modest convection-diffusion case expressed by eq. (4.43), that is, the Galerkin form in node j, the source term will be neglected[10], and the convection term central scheme will be replaced by an upwind scheme. The new expression is given by eq. (4.58):

$$a\left(\frac{u_j - u_{j-1}}{h}\right) - \nu\left(\frac{u_{j+1} - 2u_j + u_{j-1}}{h^2}\right) = 0 \tag{4.58}$$

Notice that the additional numerical diffusion (of magnitude $ah/2$) introduced by a first-order upwind scheme on the convective term can be observed by developing the scheme via Taylor expansion around x_j:

$$a\left(\frac{u_j - u_{j-1}}{h}\right) = au_x(x_j) - \frac{ah}{2}u_{xx}(x_j) + \mathcal{O}(h^2) \tag{4.59}$$

Actually, eq. (4.58) could also be derived from the central difference approximation of the equation given by eq. (4.60):

$$au_x - (\nu + \frac{ah}{2})u_{xx} = 0 \tag{4.60}$$

where the diffusive coefficient is clearly the sum of the physical diffusion ν and the artificial diffusion $ah/2$.

However, as criticized and explained by Gresho and Lee, 1981; Leonard, 1979; Vahl Davis and Mallinson, 1976, the addition of both the physical and the additional upwind diffusions will over-rate the expected value. This over diffusive phenomenon is visible on figure 4.9, for low P_e. Conversely, for higher P_e, the upwind solution is much more stable and convincing than the standard Galerkin solution.

[10] to ease the calculations

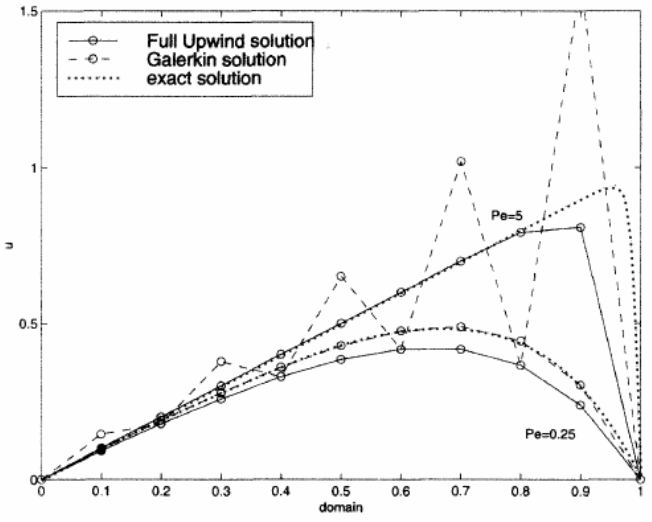


Figure 4.9: Upwind relative to Galerkin and exact solutions. (From Donea and Huerta, 2003)

The rise of Petrov-Galerkin

Nevertheless, the stabilizing effect from the upwind scheme was further studied and gave rise to an alternative to the classical Galerkin formulation: the Petrov-Galerkin (or PG). This new formulation is a weighted residual formulation where, by opposition to the Galerkin formulation, the test functions are not of the same class as the trial solutions.

In the seventies, the first PG formulations were proposed (Christie et al., 1976). Their weighting (or test) functions were build in such a manner that, on each node j, more weight would be set for the upstream element, and less for the downstream element. An illustration is given in figure 4.10.

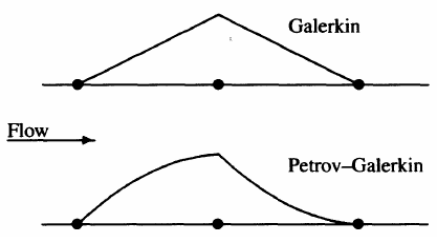


Figure 4.10: Upwind-type weighting function. (From Donea and Huerta, 2003)

Reverting to the $1D$ example of convection-diffusion with linear elements

(eq. 4.35), the shape function of internal node j (pertaining to element e and subsequent element $e+1$) can be computed by considering the contribution of both surrounding elements:

$$N_j = \begin{cases} N_2 = \frac{1}{2}(1+\xi) & from\ element\ e \\ N_1 = \frac{1}{2}(1-\xi) & from\ element\ e+1 \end{cases} \qquad for\ \xi \in [-1,1] \tag{4.61}$$

As expressed by the PG formulation, the upwind weighting function w_j will be composed of the contribution of upstream element e, reinforced, and downstream element $e+1$, reduced. The reinforcement (or upwinding) will be set through a parameter β and the functions are given by eq. (4.62)

$$w_j = \begin{cases} w_2 = \frac{1}{2}(1+\xi) + \frac{3}{4}\beta(1-\xi^2) & from\ element\ e \\ w_1 = \frac{1}{2}(1-\xi) - \frac{3}{4}\beta(1-\xi^2) & from\ element\ e+1 \end{cases} \qquad for \xi \in [-1,1] \tag{4.62}$$

Applying these PG weighting functions to the convection-diffusion weak formulation (eq. (4.36))[11] reformulated below:

$$\int_0^L (w\, a\, u_x + w_x\, \nu\, u_x) dx = 0 \tag{4.63}$$

and discretizing it in node j will transform eq. (4.47) into:

$$[1 + (1+\beta)P_e](u_j - u_{j-1}) - [1 - (1-\beta)P_e](u_{j+1} - u_j) = 0 \tag{4.64}$$

Remark 4.10. *The exact solution is given for $\beta = \coth P_e - 1/P_e$ (cf. eq. 4.56), and if $\beta = 1$, the full upwind differencing is effective.*

Unfortunately, although stable, the solution provided by this PG formulation encountered the same disproportionate dissipation as the ordinary upwind differences. Besides this, their higher-order test functions induced a more complex implementation and a higher cost in computational resources.

Streamline-Upwind (or the balancing diffusion)

Referring to the beginning of this subsection (§ 4.2.4), although the first direction (i.e. replacing the convection term centered scheme by an upstream scheme) led to the just discussed excessive dissipation, a second direction could still be analyzed: The addition of an artificial diffusion, also called the balancing diffusion (Kelly et al., 1980) for it counterbalances the negative diffusion engendered by the classical Galerkin method.

Once more, consider the steady $1D$ linear convection-diffusion case, starting from eq. (4.55), with no source term. Adding the artificial diffusion $\bar{\nu}$ to

[11] With no source term to ease the development

the weak form (eq. (4.36)), as proposed by Hughes, Liu, and Brooks, 1979, will generate following equation:

$$\int_0^L (w\, a\, u_x + w_x\, (\nu + \bar{\nu})\, u_x) dx = 0 \tag{4.65}$$

where the additional diffusion is defined in eq. (4.56) by $\bar{\nu} = \beta a h/2$.

By isolating the convective contributions from the diffusive's, eq. (4.65) can be rearranged as:

$$\int_0^L \left[(w + \beta \frac{h}{2} w_x) a\, u_x + w_x\, \nu\, u_x \right] dx = 0 \tag{4.66}$$

In this equation, one can observe a modified weighting function $\bar{w} = w + \beta(h/2)w_x$ that only affects the convection (i.e. no modification of the diffusive term), which makes it a slightly inconsistent PG formulation.

Remark 4.11. *A second observation is that this modified test function is no more continuous between two consecutive elements, as illustrated by figure 4.11.*

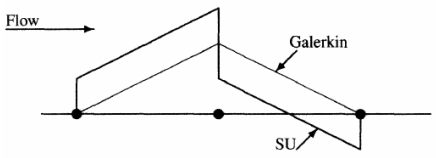


Figure 4.11: Streamline-Upwind weighting function. (From Donea and Huerta, 2003)

Coming back to the modified weighting function, because it influences the convection term only and because convection follows the streamlines, the balancing diffusion should only be inserted in the flow direction. If the latter were to be added in transversal directions, the resulting diffusion would de novo be overly diffusive.

This reflection becomes even more significant when considering a multi-dimensional domain.

To comply with this constraint, Hughes and Brooks proposed a tensorial diffusion operator where diffusion would only act along streamlines. The Streamline-Upwind (SU) schemes were born (Brooks and Hughes, 1982; Hughes, Liu, and Brooks, 1979).

In concrete terms, Hughes and Brooks substituted the scalar diffusion $\bar{\nu}$ from eq. (4.56) by the tensor diffusion $\bar{\bar{\nu}} = \bar{\nu}_{ij}$:

$$\bar{\nu}_{ij} = \bar{\nu} a_i a_j / ||a||^2 \tag{4.67}$$

for a flow velocity a, with i components.

Consecutively, to find the corresponding SU weighting function for n_{sd} spatial dimensions, the variational form is:

$$u \in \mathcal{S} \,|\, \forall w \in \mathcal{V} : \int_{\Omega} \left[w(a \cdot \nabla u) + \nabla w \cdot (\nu I_{n_{sd}} + \bar{\bar{\nu}}) \cdot \nabla u \right] d\Omega = 0 \qquad (4.68)$$

with $I_{n_{sd}}$ a n_{sd}-dimensions identity matrix.

Inserting eq. (4.67) into eq. (4.68) and isolating convective from diffusive effect yields to:

$$\int_{\Omega} \left\{ \left[w + \frac{\bar{\nu}}{||a||^2}(a \cdot \nabla w) \right] (a \cdot \nabla u) + \nu \nabla w \cdot \nabla u \right\} d\Omega = 0 \qquad (4.69)$$

From eq. (4.69), one can find the modified weighting function, the SU weighting function, expressed as:

$$\bar{w} = w + \frac{\bar{\nu}}{||a||^2}(a \cdot \nabla w) \qquad (4.70)$$

Another elegant manner to define the SU method is as a combination of the classical Galerkin method with an additional SU term:

$$\underbrace{\int_{\Omega} \left[w(a \cdot \nabla u) + \nu \nabla w \cdot \nabla u \right] d\Omega}_{\text{Classical Galerkin}} + \underbrace{\int_{\Omega} \frac{\bar{\nu}}{||a||^2}(a \cdot \nabla w)(a \cdot \nabla u) d\Omega}_{\text{Additional SU term}} = 0 \qquad (4.71)$$

where the extra SU term will be effective inside each element but not on its boundaries, due to the discontinuouty illustrated in figure 4.11.

Eventually, the compact SU weak form for the convection-diffusion case can be expressed by:

$$a(w,u) + c(a;w,u) + \underbrace{\sum_{e} \int_{\Omega^e} \frac{\bar{\nu}}{||a||^2}(a \cdot \nabla w)(a \cdot \nabla u) d\Omega}_{\text{SU stabilization term}} = (w,s) + (w,h)_{\Gamma_N} \qquad (4.72)$$

Returning to our $1D$ linear element example, the resulting profile given by figure 4.12 is stable and matching the exact solution for all P_e (remember the method is acting on the interior nodes only), what makes it an ideal candidate for future development.

Nevertheless, accuracy issues remain for more complex cases (e.g. mobile source term, variable convectivity, or time-dependency). The main reason is that the SU weighting is applied in a slightly non-consistent way, only to the convective term, inducing a non-residual formulation.

Consequently, an even more significant manner to follow the SU path will be detailed in the following subsection. This method is called the Streamline-Upwind Petrov-Galerkin (SUPG) and is the fundamental stabilization implemented and used in our work.

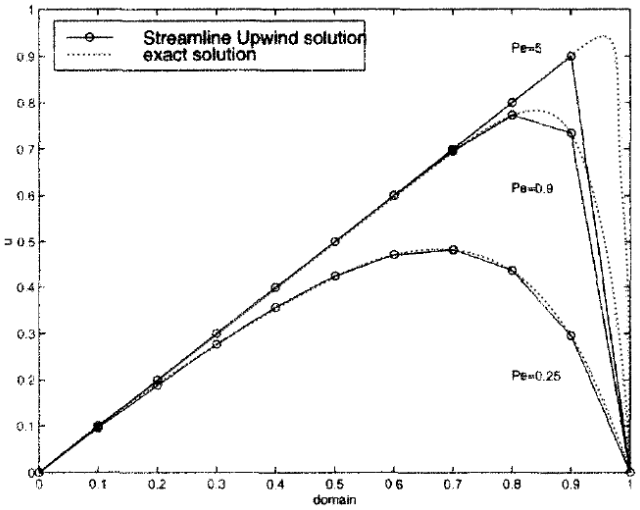


Figure 4.12: Streamline-upwind and exact solutions for the $1D$ convection-diffusion case with linear elements. (From Donea and Huerta, 2003)

The completeness of the Streamline-Upwind Petrov-Galerkin (SUPG)

For this last subsection on stabilization of the convective term, the two last concepts (i.e. PG and SU) are brought together to reach a consistent stabilization method, namely, a method that ensures that both the governing equations and its weak form have the same solution.

Starting from the SU formulation, Hughes et al. continued to evolve with this concept and proposed a residual formulation to ensure its consistency. As previously indicated, this is not the case for eq. (4.72), the SU version.

Thus, to express the residual formulation, let us first resume by rewriting the steady convection-diffusion equation from eq. (4.24) together with its essential and natural boundary conditions, for convenience:

$$\begin{aligned} a \cdot \nabla u - \nabla \cdot (\nu \nabla u) = s \quad & in\ \Omega, \\ u = u_D \quad & on\ \Gamma_D, \\ n \cdot \nu \nabla u = \nu \frac{\partial u}{\partial n} = h \quad & on\ \Gamma_N \end{aligned}$$

Its residual formulation is given by:

$$\mathcal{R}(u) = \underbrace{a \cdot \nabla u - \nabla \cdot (\nu \nabla u)}_{\mathcal{L}(u)} - s = \mathcal{L}(u) - s \tag{4.73}$$

where $\mathcal{L}$ is the PDE operator.

By constraining it to the finite dimensional spaces, the residual $\mathcal{R}(u)$ is evaluated on each element interior Ω^e, inducing following compact weak form:

$$a(w,u) + c(a;w,u) + \sum_e \int_{\Omega^e} \mathcal{P}(w)\tau\mathcal{R}(u)d\Omega = (w,s) + (w,h)_{\Gamma_N} \quad (4.74)$$

where the conventional weighting function is replaced by a generalized operator $\mathcal{P}(w)$, and where a new parameter is defined: the stabilization parameter τ.

By applying the SU weighting function consistently to all terms (and not only to the convective term), one can express it through the operator $\mathcal{P}(w)$ from eq. (4.75):

$$\mathcal{P}(w) = a \cdot \nabla w \quad (4.75)$$

involving that the test function space $\mathcal{V}$ is no more corresponding with the trial solution space $\mathcal{S}$[12]. Therefore, this consistent stabilization technique is named SUPG.

Eventually, injecting the operator from eq. (4.75) and the residual expression from eq. (4.73) into eq. (4.74), one will find the desired discrete case to find $u \in \mathcal{S}^h|$

$$\begin{aligned} a(w^h,u^h) + c(a;w^h,u^h) \\ + \sum_e \int_{\Omega^e} (a\cdot\nabla w^h)\tau\left[a\cdot\nabla u^h - \nabla\cdot(\nu\nabla u^h) - s\right]d\Omega \\ = (w^h,s) + (w^h,h)_{\Gamma_N} \quad \forall w^h \in \mathcal{V}^h \end{aligned} \quad (4.76)$$

with $\tau = \nu/||a||^2$.

To illustrate the performance of the SUPG formulation, the $1D$ linear elements case from the previous sections will be reconsidered but this time, with a source term added, to complexify the expression:

$$\begin{cases} a\,u_x - \nu\,u_{xx} = 5e^{-100(x-\frac{1}{8})^2} - 5e^{-100(x-\frac{1}{4})^2} & in\ \ ]0,1[\\ u = 0 & at\ x = 0\ and\ x = 1 \end{cases} \quad (4.77)$$

The resulting plot is given by figure 4.13 where the matching between the exact solution and the SUPG solution is excellent.

Remark 4.12. *The informed reader will have noticed no detail was provided concerning the stabilization parameter τ, neither concerning the unsymmetric behavior brought by the SUPG stabilization term from eq. (4.76). These two issues are not trivial and are discussed further in Codina, 2000; Tezduyar and Osawa, 2000.*

[12] Which was the definition of a PG formulation.

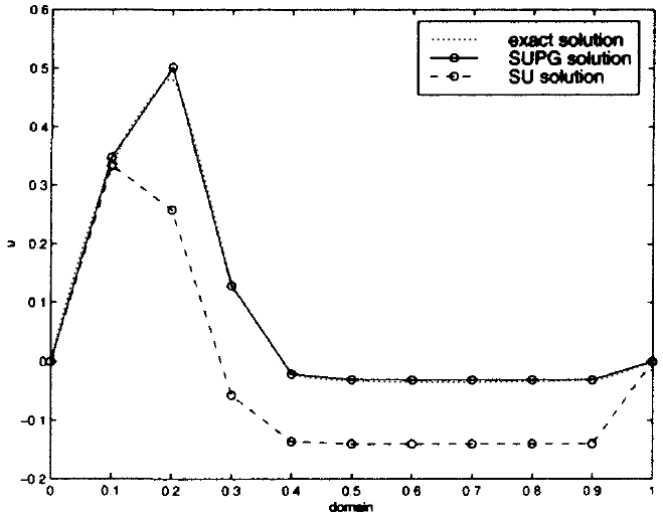


Figure 4.13: Comparison between the exact, SU and SUPG solution for eq. (4.77). (From Donea and Huerta, 2003)

Eventually, while the SUPG method reduces oscillations caused by strongly advective flows, there is another type of spurious numerical oscillation. The latter is visible when discretizing PDEs containing a pressure term (e.g. the Navier-Stokes equations). In this case, having an identical finite element order for the velocity and the pressure will result in instabilities on the pressure side. Briefly, these instabilities are due to the Ladyzhenskaya-Babuška-Brezzi condition (LBB) not being satisfied.

This issue is analogously solved by the stabilizing method named the Pressure-Stabilizing Petrov-Galerkin method (PSPG). Together with SUPG, they form the so-called PSPG/SUPG stabilizing method.

Although essential and used in this work, the author elaborated primarily on the SUPG design to facilitate the comprehension of the main constraint of this work, which will be covered in § 6.8 when presenting the results. The curious reader will find enlightening clarifications concerning PSPG in Bányai, Vanden Abeele, and Deconinck, 2006; Tezduyar, 1991.

4.2.5 PSPG/SUPG applied to Navier-Stokes

Having introduced the (PSPG/)SUPG methodology on a simple PDE, this theory can be incorporated into our model, where the two first Navier-Stokes transport equations of eqs. (4.3) are considered.

The finite element formulation can be achieved by multiplying these equations with weighting functions. The unknown variables can be interpolated between the discrete nodes by using shape functions, and eventually, these

equations can be integrated over the whole domain. The expressions can be simplified by choosing weighting functions equal to shape functions, which leads to a Galerkin formulation. Its implementation (eqs. (4.78)) follows the details provided by Janssens, 2014.

$$\int_\Omega (N_i + \tau_{SU}(\vec{u}\cdot\nabla N_i))\vec{\mathcal{R}}_m d\Omega + \int_\Omega \tau_{BU}\nabla N_i \mathcal{R}_c d\Omega = 0$$
$$\int_\Omega N_i \mathcal{R}_c d\Omega + \int_\Omega \tau_{PS}\nabla N_i \cdot \vec{\mathcal{R}}_m d\Omega = 0 \tag{4.78}$$

with the time scale (or stabilization coefficients) τ_{SU} coupled to the SUPG method, and $\tau_{PS,BU}$ to the PSPG method. These time scales are further described in Braack et al., 2007. The indices c, m for the residual $\mathcal{R}$ refer respectively to the continuity and momentum equations of the governing equations.

After applying a θ-method (Aslefallah, Rostamy, and Hosseinkhani, 2014) to find the time derivative and because the shape functions are non-zero only on their respective node and surrounding element, the integrals for all the elements can be replaced by a sum of the integral on each element. At last, the pressure and the velocity for the next timestep can be found by solving this discrete system:

$$\sum_{e=1}^{N}\left(\frac{1}{\Delta t}T_e + \theta A_e\right)\left(x_e^{n+1} - x_e^n\right) = -\sum_{e=1}^{N} A_e x_e^n \tag{4.79}$$

where θ is set to 1 for a (fully implicit) forward Euler scheme or 0.5 for a Crank-Nicolson scheme. By default, it is set to 0.5 in Janssens, 2014.

The unknown x_e^n for each element, grouped by nodal values, have following format for a 3D element with $m+1$ nodes:

$$x_e^n = [p_0^n \cdots p_m^n (u_0^n)_0 \cdots (u_0^n)_m \cdots (u_2^n)_m] \tag{4.80}$$

The matrices A_e and T_e have the following structure:

$$A_e = \begin{bmatrix} A_{pp} & A_{pu} \\ A_{up} & A_{uu} \end{bmatrix} = \begin{bmatrix} A_{pp} & A_{pu_0} & A_{pu_1} & A_{pu_2} \\ A_{u_0p} & A_{u_0u_0} & A_{u_0u_1} & A_{u_0u_2} \\ A_{u_1p} & A_{u_1u_0} & A_{u_1u_1} & A_{u_1u_2} \\ A_{u_2p} & A_{u_2u_0} & A_{u_2u_1} & A_{u_2u_2} \end{bmatrix} \tag{4.81}$$

Each block of A_e can be formulated following eqs. (4.82-4.88):

$$A_{pp} = \int_{\Omega_e} \tau_{PS}\nabla N_p^T \nabla N_p d\Omega_e \tag{4.82}$$

$$A_{pu_i} = \int_{\Omega_e}\left(\left(N_p + \frac{\tau_{PS}\tilde{u}_{adv}\nabla N_p}{2}\right)^T (\nabla N_u)_i + \tau_{PS}\left(\nabla N_p\right)_i^T \tilde{u}_{adv}\nabla N_u\right) d\Omega_e \tag{4.83}$$

$$A_{u_i p} = \int_{\Omega_e} \left(N_u + \tau_{SU} \tilde{u}_{adv} \nabla N_u\right)^T \nabla N_p d\Omega_e \tag{4.84}$$

$$A_{u_i u_i} = \int_{\Omega_e} \Bigg(\nu \nabla N_u^T \nabla N_u + \left(N_u + \tau_{SU} \tilde{u}_{adv} \nabla N_u\right)^T \tilde{u}_{adv} \nabla N_u \Bigg) d\Omega_e + A_{u_i u_j} \tag{4.85}$$

$$A_{u_i u_j} = \int_{\Omega_e} \Bigg(\tau_{BU} \left(\nabla N_u\right)_i + \frac{1}{2} \left(\tilde{u}_{adv}\right)_i \left(N_u + \tau_{SU} u \nabla N_u\right) \Bigg)^T \left(\nabla N_u\right)_j d\Omega_e \tag{4.86}$$

$$T_{p u_i} = \int_{\Omega_e} \tau_{PS} \left(\nabla N_p\right)_i^T N_u d\Omega_e \tag{4.87}$$

$$T_{u_i u_i} = \int_{\Omega_e} \left(N_u + \tau_{SU} \tilde{u}_{adv} \nabla N_u\right)^T N_u d\Omega_e \tag{4.88}$$

Notice that $\tilde{u}_{adv}$ is the advection velocity, obtained, in Janssens, 2014, by a Taylor expansion of the previous velocities. And where the stabilization terms, multiplied by their respective stabilization coefficients ($\tau_{PS,SU,BU}$), are presented in Braack et al., 2007. The thorough development is proposed by Banyai (Bányai, Vanden Abeele, and Deconinck, 2006).

Remark 4.13. *When observing the previous equations, one will notice that:*

1. τ_{PS} *has an influence both on the* A_{pp} *and* A_{pu_i} *blocks. The most impacted is the former, the laplacian of the pressure, which is non-zero when* $\tau_{PS} \neq 0$.

2. τ_{SU} *increases the weight of the upstream nodes relative to the stream direction.*

3. τ_{BU} *is influencing advection flows.*

Consequently, these coefficients have to be chosen in such a manner that the numerical values are properly stabilized but not over-dissipated.

Hereafter, eqs. (4.89-4.94) reveal the definition proposed by Trofimova et al., 2009, based on the element metric tensor G_{ij} (i.e. the element shape function inverse Jacobian matrix multiplied with its transpose).

$$\tau_{SU1}^2 = \frac{1}{u_i G_{ij} u_j} \tag{4.89}$$

$$\tau_{SU2} = \frac{\Delta t}{2} \tag{4.90}$$

$$\tau_{SU3}^2 = \frac{1}{\nu^2 G_{ij} G_{ij}} \tag{4.91}$$

$$\tau_{SU} = \left(\frac{1}{\tau_{SU1}^2} + \frac{c_1^2}{\tau_{SU2}^2} + \frac{c_2}{\tau_{SU3}^2} \right)^{-1/2} \tag{4.92}$$

$$\tau_{PS} = \tau_{SU} \tag{4.93}$$

$$\tau_{BU} = \frac{1}{\tau_{SU} G_{ij}} \tag{4.94}$$

where c_1 and c_2 are tuning parameters set to $4 \leq c_1 \leq 16$ and $c_2 = 36$ by the author in Trofimova et al., 2009, for an advection flow.

This stabilization method was implemented by Janssens, 2014 and successfully helped to solve oscillations for reasonable Reynolds simulation (e.g. the channel flow test case).

Eqs. (4.79) constitute the genuine implementation in the *Coolfluid 3* code. It will be used in § 5.1, § 6, and § 7.

Now that the foundation of our modeling has been set, the following section will provide a few more details concerning the ABL modeling.

4.3 ABL modeling

Section 3.2 described the composition of an ABL physically and presented its complexity in great detail. In Part III, the complete representation will be developed, and its resulting velocity profile will be analyzed.

To bind these two sections, we still need to introduce the required modeling. In this respect, we recall two substantial ABL specificities:

- the size of the domain (and the Reynolds number associated).
- the roughness of the ground

The first feature is constrained by the physical time and the computational resources available. However, as previously commented, reducing the number of elements in the domain to study can decrease the computation at each simulation time step in a non-negligible manner.

The second ABL particularity is associated with the irregularities and the properties of the studied terrain. One choice is to meticulously reproduce the topography and apply specific physical properties to the interface (e.g. porosity). This methodology requires a significant amount of elements.

Hence, it works against the first feature. An alternative would be to substitute the resolution in the near-wall region, where the complex geometries and the significant velocity gradients will require smaller (read more) elements, with a model representing the dynamics of that zone. Such a model is named a wall model.

This wall model solution would thus operate on these two ABL specificities beneficially. As such, the following section will further develop this direction.

4.3.1 Wall models: Preamble

To have a better idea on where these wall models are operating, figure 4.14 presents the near-wall region (also named inner layer) in the reduced coordinate system (also called wall units).

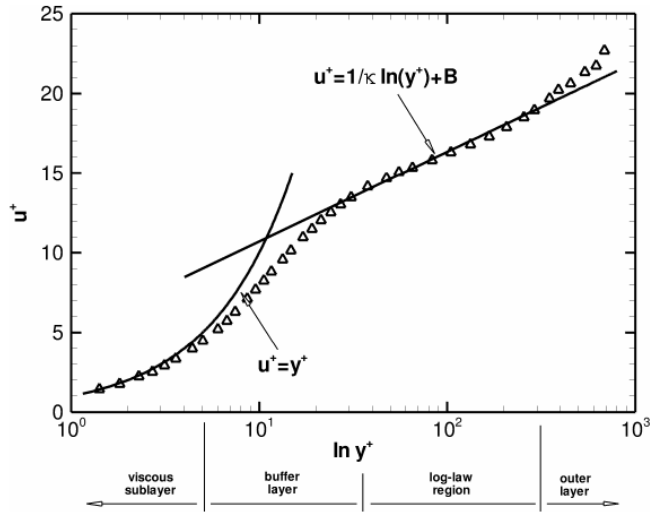


Figure 4.14: Boundary layer regions expressed non-dimensionally in a semi-log scale. (From Banyai, 2016)

The first region, the viscous sublayer, is characterized by its linear behavior. Including this scale for the resolution of the governing equations corresponds to a full-scale resolution (in other words, DNS resolution (§ 4.1.3)) and implies a considerable amount of tiny elements to capture the smallest turbulent scales and the high-velocity gradient near the wall. Notice that results for unsophisticated flow cases like a channel flow exist until a viscous Reynolds number of $Re_\tau = 5200$ (Lee and Moser, 2015; Moser, Kim, and Mansour, 1999). These cases will be of interest in § 7.5.

Following this first sublayer, there is a transition region named the buffer zone, where initiating any flow resolution is unadvised since no deterministic profile can be defined.

Eventually, after gradually shifting from the linear profile, the last sublayer of the inner region named the inertial sublayer (or log-law region) presents an explicit logarithmic behavior.

The region of interest is located in what is called the viscous and buffer sub-layers. By modeling these two sub-layer regions and starting resolving the governing equations from the inertial region, the quantity of elements is significantly reduced, rendering such a simulation accessible.

4.3.2 Wall models: Choice

Referring to Sagaut, 2006, when considering a single layer simulation, several wall model implementations are available:

- The Schumann model: requires the skin friction as input and relates the wall stress to the wall parallel velocity components at the first grid points.
- The Grötzbach model: is an evolution of the Schumann model, deducing the skin friction from the log-law profile inversion associated with the streamwise velocity.
- The various extensions of the Grötzbach model: the shifted correlation model considers the coherent structures; the ejection model includes the sweep and ejection effects; the Werner and Wengle model substitutes the log-law profile for the power-law (§ 3.2.2); and lastly, the simplified Mukarami model.
- The Mason and Callen model: designed for rough walls.
- and finally the suboptimal-control-based wall models: designed to improve the results with high Reynolds number and unrefined grid.

In this work, although the hereabove listed models are more sophisticated than the Schumann's, we decided to follow the path proposed by recent ABL studies (Bechmann, 2006; Goit, 2015) to provide a reference. Thus, the chosen approach is the Schumann model.

As will be presented in the section dedicated to its implementation (§ 6.4.1), it will be augmented with a zero vertical velocity on the wall (i.e. impermeability), while the neutral rough boundary layer will be induced by the Monin-Obukhov similarity theory (Monin and Obukhov, 1954).

4.3.3 Main FEM contribution

To emphasize the importance of what the author intends to achieve, figure 6.6 provides three sketches. The two upper sketches present the current situation in the CFD scope, when using FVM:

- The upper left sketch represents the full resolution of all scales, using a large amount of elements near the wall, drastically constraining the domain size of the case studies.

- The upper right sketch substitutes all elements near the wall by a single layer of cells that will follow a specific wall model. This second approach only exists within the FVM theory.

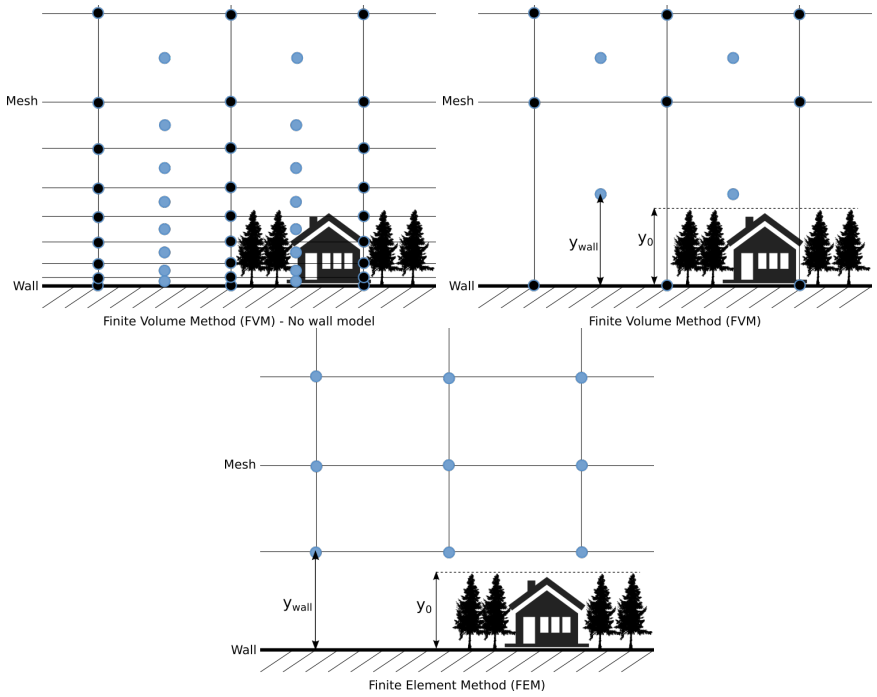


Figure 4.15: Wall model integration to FVM and FEM.

The specificity of the path followed by the author is suggested in the lowest sketch of figure 6.6, namely, to introduce the wall models into the FEM theory.

The latter has not yet been developed, to the author's knowledge, and represents a main contribution.

In this peculiar case, the computational domain is completely separated from the physical wall, by a distance y_{wall} and the physical properties of the first nodes above the wall will include the result of the newly implemented wall model.

By enabling the use of wall models in the FEM theory, the author allies the robustness of the FEM to the reduced domain size achieved through the wall models.

Its implementation will be developped in § 6.4.1, where both the distance to the physical wall, y_{wall}, and the terrain roughness, y_0, will be integrated.

Before closing this chapter, a brief section concerning the dimensionality of the present work is required to emphasize the challenges that are to be expected.

4.4 Dimension of the system

Although not directly related to the previous sections, this segment is necessary to emphasize the complexity associated with the system's dimensionality that is to be studied.

To keep the writing concise, the first paragraph will express the requirements. They will be followed by a paragraph regarding the correlated constraints and concluded by the adopted approach.

Thus, as explained in chapter 3, the atmospheric boundary layer is occurring at the surface of the Earth. The latter is revolving, inducing a Coriolis effect that will eventually be added to the modeling to represent physical behavior properly. Moreover, the roughness of the ground, together with the temperature gradient happening during a diurnal cycle, influences the amount of turbulence that is to be observed near the surface. Lastly, turbulences are not only propagating in the vertical plane, but they do also spread and develop laterally, making them a 3D phenomenon per se. As a consequence, a three-dimensional simulation is a prerequisite to allowing an accurate representation of an ABL.

On the other side, adding a third dimension multiplies the number of elements by the chosen number of subdivisions. By way of example, a computation performed on a domain subdivided into 256×256 (i.e. 65536) elements in 2D will explode to 16 millions of elements if the same number of subdivisions is taken in the third dimension. Since for each timestep of the simulation, the unknowns have to be computed for each element, the velocity at which the simulation can evolve in time is dependent on two factors:

- the computation time inside each element,
- the interaction time between these elements.

Of course, the computational resources available are increasing quickly (cf. fig. 4.16). Indeed, it could be possible to take advantage of computational tricks like parallelism and multi-processing if the code allows it. However, the resources are ultimately limited, and the simulation still needs to provide a sufficiently accurate result. In this respect, there is an extra non-negligible advantage associated with FEM code: it only requires the definition of a new type of element (e.g. 3D) to reuse all the already developed algorithms.

In conclusion, the typical approach in FEM is to implement the required algorithm and then proceed to primary assessments on a small 2D grid.

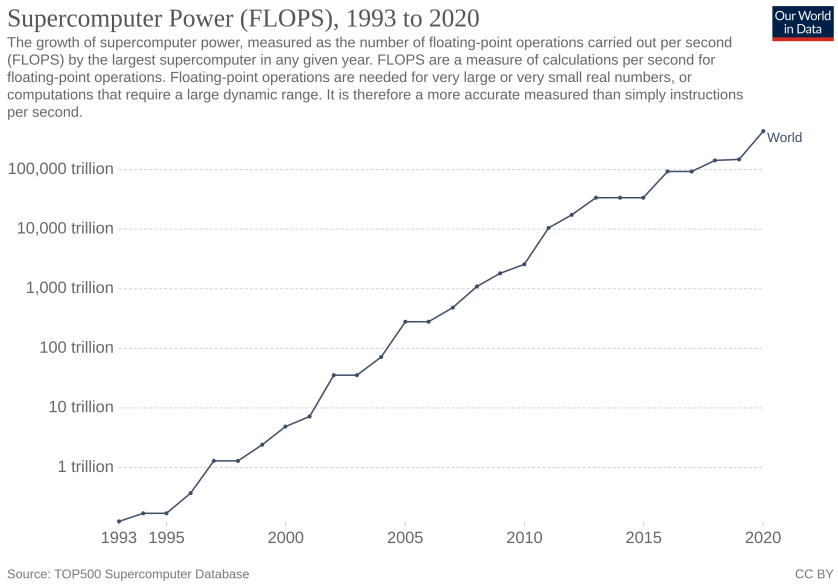


Figure 4.16: Supercomputer Power(FLOPS) following Moore's Law. (From Ritchie et al., 2020)

When physical behavior demands the third dimension, the analysis is done in 3D. This work followed this methodology (except in the first section of chapter 6 used to identify the challenges).

Chapter 5

Turbulence models

In the last chapter, we have evoked the main challenge faced in this work (i.e. spurious numerical oscillations that will be presented in Part III). These oscillations justified, on the one hand, the previous study on the PSPG/-SUPG stabilization method (§ 4.2.4), and, on the other hand, the two fundamental directions we have considered to investigate further.

These two directions are:

- the Variational MultiScale method (VMS), and
- the Large Eddy Simulations (LES)

5.1 VMS

The PSPG/SUPG stabilization methods gave FEM the ability to handle more complex problems better and, in particular, convective flows. Supposing that these stabilizations would not be sufficient for our case, there is an evolution, or to be specific, a generalized version of the SUPG stabilization method, called the Variational Multiscale method (VMS). Hughes introduced it (Hughes, Mazzei, and Jansen, 2000), while Bazilevs proposed an algorithm (Bazilevs et al., 2007).

The essence of the VMS is that it applies the scale separation (coarse vs. fine) primarily and only then approximates the fines scales. In these small scales, the stabilization terms appear "naturally".

The next section will present an implementation of the latter, following the convention proposed by Y. Bazilevs, adjusted to our notation.

5.1.1 Structure

The proposed VMS method derives from Bazilevs et al., 2007 and follows a Newton structure, also called predictor / multi-corrector structure, schematized in figure 5.1 and detailed in the following subsections. This method has more parameters than the standard SUPG method, allowing it to better respond to numerical oscillations.

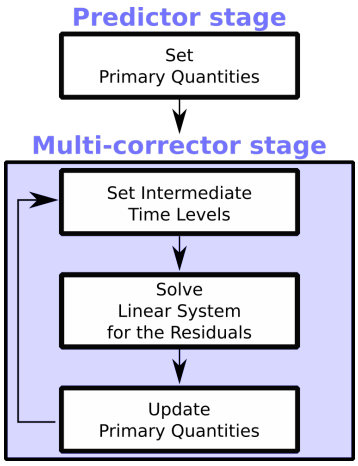


Figure 5.1: Predictor / Multi-corrector's structure

5.1.2 Stages

Predictor stage

For each timestep, during the first stage, the predictor stage, the three variables (velocity U, derivative of the velocity $\dot{U}$ and pressure P) are set according to previous timestep.

$$U_{n+1,(0)} = U_n \tag{5.1}$$

$$\dot{U}_{n+1,(0)} = \frac{\gamma - 1}{\gamma} \dot{U}_n \tag{5.2}$$

$$P_{n+1,(0)} = P_n \tag{5.3}$$

γ (as well as α_f and α_m that are presented in § 5.1.2) is a real-valued parameter defining the α-method. It is developped in Chung and Hulbert, 1993 and especially for fluid dynamics in Jansen, Whiting, and Hulbert, 2000.

Multi-corrector stage

Then, during the second stage (that is, the nonlinear multi-corrector stage), a number of iterations ($l = 1, 2, ..., l_{max}$) is performed (typically between 2 and 4) to converge to steady values, or in other words, the residuals for, both, the continuity and momentum equations of eq. (4.3) are minimized.

To be more explicit, during these iterations, the following steps are performed:

(1) The first step is to set the intermediate time levels:

$$\dot{U}_{n+\alpha_m,(l)} = \dot{U}_n + \alpha_m \left(\dot{U}_{n+1,(l-1)} - \dot{U}_n \right) \tag{5.4}$$

$$U_{n+\alpha_f,(l)} = U_n + \alpha_f \left(U_{n+1,(l-1)} - U_n \right) \tag{5.5}$$

$$P_{n+1,(l)} = P_{n+1,(l-1)} \tag{5.6}$$

α_f and α_m are intermediate time level parameters. They (as well as previously mentionned γ) are selected, taking accuracy and stability considerations into account. According to Jansen, Whiting, and Hulbert, 2000, obtaining second-order accuracy in time is possible if:

$$\gamma = 1/2 + \alpha_m - \alpha_f \tag{5.7}$$

and the method is unconditionally stable if:

$$\alpha_m \geqslant \alpha_f \geqslant 1/2 \tag{5.8}$$

For the latter, α_f and α_m can be expressed as a function of one single parameter: ρ_∞. That is, the spectral radius of the amplification matrix when $\Delta t \to \infty$.

$$\alpha_m = \frac{1}{2}\left(\frac{3-\rho_\infty}{1+\rho_\infty}\right) \qquad \textit{and} \qquad \alpha_f = \frac{1}{1+\rho_\infty} \tag{5.9}$$

Further details on how this single parameter controls the dissipation for high-frequencies are extensively described in Hughes, 2000. Therefore, by selecting a γ agreeing with eq. (5.7) and by defining the proper ρ_∞, one can ensure the time integration scheme is unconditionally stable.

(2) These intermediate values are used to assemble the residuals of the continuity and momentum equations and solve the following linear system:

$$A_{\dot{u}\dot{u}} \Delta \dot{U}_{n+1,(l)} + A_{\dot{u}p} \Delta P_{n+1,(l)} = -\mathcal{R}^M_{(l)} \tag{5.10}$$

$$A_{p\dot{u}} \Delta \dot{U}_{n+1,(l)} + A_{pp} \Delta P_{n+1,(l)} = -\mathcal{R}^C_{(l)} \tag{5.11}$$

that can be rearranged as below:

$$\begin{bmatrix} A_{\dot{u}\dot{u}} & A_{\dot{u}p} \\ A_{p\dot{u}} & A_{pp} \end{bmatrix} \begin{bmatrix} \Delta \dot{U}_{n+1,(l)} \\ \Delta P_{n+1,(l)} \end{bmatrix} = \begin{bmatrix} -\mathcal{R}^M_{(l)} \\ -\mathcal{R}^C_{(l)} \end{bmatrix} \tag{5.12}$$

(3) This will provide values for the derivative of the velocity and the pressure that can be updated in the last step of the multi-stage corrector:

$$\dot{U}_{n+1,(l)} = \dot{U}_{n+1,(l-1)} + \Delta \dot{U}_{n+1,(l)} \tag{5.13}$$

$$U_{n+1,(l)} = U_{n+1,(l-1)} + \gamma \Delta t \Delta \dot{U}_{n+1,(l)} \tag{5.14}$$

$$P_{n+1,(l)} = P_{n+1,(l-1)} + \Delta P_{n+1,(l)} \tag{5.15}$$

In terms of computation, the second step of the multi-corrector stage, namely, assembling and solving the linear system, is the most costly.

5.1.3 VMS assembly matrix Ae

Remembering the SUPG Assembly matrix detailed in eqs. (4.82-4.88), a slightly different assembly matrix A_e is provided:

$$A_e = \begin{bmatrix} A_{\dot{u}\dot{u}} & A_{\dot{u}p} \\ A_{p\dot{u}} & A_{pp} \end{bmatrix} = \begin{bmatrix} A_{\dot{u}_0\dot{u}_0} & A_{\dot{u}_0\dot{u}_1} & A_{\dot{u}_0\dot{u}_2} & A_{\dot{u}_0 p} \\ A_{\dot{u}_1\dot{u}_0} & A_{\dot{u}_1\dot{u}_1} & A_{\dot{u}_1\dot{u}_2} & A_{\dot{u}_1 p} \\ A_{\dot{u}_2\dot{u}_0} & A_{\dot{u}_2\dot{u}_1} & A_{\dot{u}_2\dot{u}_2} & A_{\dot{u}_2 p} \\ A_{p\dot{u}_0} & A_{p\dot{u}_1} & A_{p\dot{u}_2} & A_{pp} \end{bmatrix} \tag{5.16}$$

Indeed, in this case, the first block depends on the derivative of the velocity $\dot{u}$ and the last block on the pressure p. This structure was chosen to match the proposed implementation of Bazilevs et al., 2007 but it is equivalent to the one presente in the SUPG section (§ 4.2.5).

For each block, the expression is detailed hereafter:

$$\begin{aligned} A_{\dot{u}_i\dot{u}_i} = \int_{\Omega_e} \Big(& \alpha_m N_{\dot{u}}^T N_{\dot{u}} \\ & + \alpha_m \left(u\tau_M \nabla N_{\dot{u}}\right)^T N_{\dot{u}} \\ & + \alpha_f \gamma \Delta t N_{\dot{u}}^T u \nabla N_{\dot{u}} \\ & + \alpha_f \gamma \Delta t \left(\nabla N_{\dot{u}} \nu\right)^T \nabla N_{\dot{u}} \\ & + \alpha_f \gamma \Delta t \left(u \nabla N_{\dot{u}} \tau_M\right)^T \left(u \nabla N_{\dot{u}}\right) \Big) d\Omega_e \end{aligned} \tag{5.17}$$

$$\begin{aligned} A_{\dot{u}_i\dot{u}_j} = \int_{\Omega_e} \Big(& \alpha_f \gamma \Delta t \left(\nabla N_{\dot{u}}\right)_j^T \nu \left(\nabla N_{\dot{u}}\right)_i \\ & + \alpha_f \gamma \Delta t \left(\nabla N_{\dot{u}}\right)_i^T \tau_C \left(\nabla N_{\dot{u}}\right)_j \Big) d\Omega_e \end{aligned} \tag{5.18}$$

$$\begin{aligned} A_{\dot{u}_i p} = \int_{\Omega_e} & - \left(\nabla N_{\dot{u}}\right)_i^T N_p \\ & + \left(u^T \tau_M \left(\nabla N_{\dot{u}}\right)_i\right)^T \nabla N_p d\Omega_e \end{aligned} \tag{5.19}$$

$$\begin{aligned} A_{p\dot{u}_i} = \int_{\Omega_e} \Big(& \alpha_f \gamma \Delta t N_p^T \left(\nabla N_{\dot{u}}\right)_i \\ & + \alpha_f \gamma \Delta t \left(\nabla N_p\right)_i^T \tau_M u \nabla N_{\dot{u}} \\ & + \alpha_m \left(\nabla N_p\right)_i^T \tau_M N_{\dot{u}} \Big) d\Omega_e \end{aligned} \tag{5.20}$$

$$A_{pp} = \int_{\Omega_e} (\nabla N_p)^T \tau_M \nabla N_p d\Omega_e \tag{5.21}$$

As previously described, the $N_{[\dot{u},p]}$ represent the shape functions, the $\alpha_{[f,m]}$ and γ represent the intermediate time level parameters, Δt represents the difference between timestep t_{n+1} and t_n, u is the advection velocity (different of $\tilde{u}_{adv}$ because computed during the simulation and not constructed from a Taylor serie), ν is the kinematic viscosity, and at last, $\tau_{[C,M]}$ are the stabilizing coefficients.

All these components have been defined except the two stabilizing coefficients. The following subsection will describe them thoroughly.

5.1.4 Defining the stabilizing coefficients

Two stabilizing coefficients are defined: τ_m and τ_c. The first one is the stabilization coefficient associated with the momentum equation, and the second, with the continuity equation.

Below, their respective expressions:

$$\tau_m = \left(\frac{4}{\Delta t^2} + u \cdot Gu + C_1 \nu^2 G : G \right)^{-1/2} \tag{5.22}$$

$$\tau_c = (\tau_m g \cdot g)^{-1} \tag{5.23}$$

where C_1 is a positive constant, derived from an element-wise inverse estimate described in Johnson, 1987 and set to 1 in our work. And where three specific operations have to be introduced:

1) the tensor product $G : G$

2) the advection term $u \cdot Gu$

3) the vector product $g \cdot g$

To compute all these items, first, the element's shape function inverse Jacobian matrix has to be computed (eq. 5.24).

$$J^{-1} = \left(\frac{\partial x_i}{\partial \xi_k} \right)^{-1} \tag{5.24}$$

Before computing items 1) and 2), we first need to define the element metric tensor (eq. (5.25)).

$$G_{ij} = \sum_{k=1}^{3} J^{-1}_{k,i} \left(J^{-1}_{k,j} \right)^T \tag{5.25}$$

Then, item 1) can be reached by computing the tensor product of G_{ij} (eq. (5.26)).

$$G : G = \sum_{i,j=1}^{3} G_{ij} G_{ij} \tag{5.26}$$

Item 2) can be obtained by determining the scalar product of the velocity by the product of the inverse jacobian tensor and the velocity (eq. (5.27)).

$$u^h \cdot G u^h = \sum_{i,j=1}^{3} u_i^h G_{ij} (u_j^h)^T \tag{5.27}$$

For the last item, 3), first the sum of the inverse of the jacobian is formed and then, its product is column-wise summed (eq. (5.28)).

$$g_i = \sum_{j=1}^{3} (J^{-1})_{j,i} \qquad \Rightarrow \qquad g \cdot g = \sum_{i=1}^{3} g_i (g_i)^T \tag{5.28}$$

When looking to the SUPG method, τ_c (eq. (5.23)) is equivalent to τ_{BU} (eq. (4.94)). For the relation between τ_m and τ_{SU} (and thus τ_{PS}), when looking to eq. (4.92) and eq. (5.22), except for the tuning parameters, these coefficients are equivalent.

5.1.5 SUPG vs VMS assembly matrix comparison

Now that both SUPG (§ 4.2.4) and VMS (§ 5.1) are described, it is convenient to show that, indeed, the VMS method can be considered as a generalized version of the older and more used SUPG method.

To ease the comprehension, the viscosity term will be dash-underlined, the stabilization terms dot-underlined. Moreover, to simplify the expressions, the intermediate time level parameters will be set to unity: $\alpha_f = \alpha_m = \gamma = 1$. Furthermore, because the SUPG assembly matrices have the velocity and the pressure as unknowns whereas the VMS assembly matrix has the derivative of the velocity and the pressure as unknowns, to be able to compare them, Δt is set equal to 1 making the value of $\dot{u}$ and u equal.

In the next equations, each block of the assembly matrices will be rewritten for comparison.

1. PP block:

$$SUPG\ (eq.\ (4.82)) : A_{pp} = \int_{\Omega_e} \underdot{\tau_{PS} \nabla N_p^T \nabla N_p} d\Omega_e$$

$$VMS\ (eq.\ (5.21)) : A_{pp} = \int_{\Omega_e} \underdot{\nabla N_p^T \tau_M \nabla N_p} d\Omega_e$$

Both equations are expressing the shape functions related to the pressure and can be considered equivalent, assuming $\tau_{PS} \equiv \tau_M$

2. PU_i block:

$$
\begin{aligned}
&SUPG\ (eq.\ (4.83)):\\
&A_{pu_i} = \int_{\Omega_e} \left(\left(N_p + \frac{\tau_{PS} \tilde{u}_{adv} \nabla N_p}{2} \right)^T (\nabla N_u)_i \right.\\
&\qquad \left. + \tau_{PS} \left(\nabla N_p\right)_i^T \tilde{u}_{adv} \nabla N_u \right) d\Omega_e\\
&SUPG\ (eq.\ (4.87)):\\
&T_{pu_i} = \int_{\Omega_e} \tau_{PS} \left(\nabla N_p\right)_i^T N_u d\Omega_e
\end{aligned}
$$

$$
\begin{aligned}
&VMS\ (eq.\ (5.20)):\\
&A_{pu_i} = \int_{\Omega_e} \left(\alpha_f \gamma \Delta t N_p^T \left(\nabla N_{\dot{u}}\right)_i \right.\\
&\qquad + \alpha_f \gamma \Delta t \left(\nabla N_p\right)_i^T \tau_M u \nabla N_{\dot{u}}\\
&\qquad \left. + \alpha_m \left(\nabla N_p\right)_i^T \tau_M N_{\dot{u}} \right) d\Omega_e
\end{aligned}
$$

First observation is that both the assembly matrix A_e and T_e for the SUPG method are gathered into the assembly matrix A_e in the VMS method.

Second observation is that eq. (4.83) is proportional to the two first terms of eq. (5.20), while the third term in eq. (5.20) is equivalent to eq. (4.87).

Note that the second term of the SUPG formulation is written in a skew-symmetric form to improve its stability and accuracy properties.

3. U_iP block:

$$
\begin{aligned}
&SUPG\ (eq.\ (4.84)):\\
&A_{u_ip} = \int_{\Omega_e} \left(N_u + \tau_{SU} \tilde{u}_{adv} \nabla N_u \right)^T (\nabla N_p)_i d\Omega_e\\
&VMS\ (eq.\ (5.19)):\\
&A_{u_ip} = \int_{\Omega_e} - \left(\nabla N_{\dot{u}}\right)_i^T N_p\\
&\qquad + \left(u \tau_M \nabla N_{\dot{u}}\right)^T (\nabla N_p)_i d\Omega_e
\end{aligned}
$$

After developing the integration by parts, the first term of eq. (4.84) can be expressed as the first term of eq. (5.19). The second terms are directly equivalent.

This also implies that nothing has to be done to set the pressure to 0 on a boundary (e.g. at the outlet). This type of boundary condition is named the "do-nothing" boundary condition (Braack and Mucha, 2014).

4. U_iU_i block:

$$
\begin{aligned}
&SUPG\ (eq.\ (4.85)):\\
&A_{u_iu_i} = \int_{\Omega_e} \Bigg(\underline{\nu \nabla N_u^T \nabla N_u} \\
&\qquad + \left(N_u + \underset{\cdots}{\tau_{SU}} \tilde{u}_{adv} \nabla N_u \right)^T \tilde{u}_{adv} \nabla N_u \Bigg) d\Omega_e + A_{u_iu_j}\\
&SUPG\ (eq.\ (4.88)):\\
&T_{u_iu_i} = \int_{\Omega_e} \left(N_u + \underset{\cdots}{\tau_{SU}} \tilde{u}_{adv} \nabla N_u \right)^T N_u d\Omega_e\\
&VMS\ (eq.\ (5.17)):\\
&A_{u_iu_i} = \int_{\Omega_e} \Bigg(\alpha_m N_{\dot{u}}^T N_{\dot{u}} \\
&\qquad + \underset{\cdots}{\alpha_m} \left(u \tau_M \nabla N_{\dot{u}} \right)^T N_{\dot{u}} \\
&\qquad + \alpha_f \gamma \Delta t N_{\dot{u}}^T u \nabla N_{\dot{u}} \\
&\qquad + \underline{\alpha_f \gamma \Delta t} \left(\nabla N_{\dot{u}} \nu \right)^T \nabla N_{\dot{u}} \\
&\qquad + \underset{\cdots}{\alpha_f \gamma \Delta t} \left(u \nabla N_{\dot{u}} \tau_M \right)^T \left(u \nabla N_{\dot{u}} \right) \Bigg) d\Omega_e + A_{u_iu_j}
\end{aligned}
$$

For U_iU_i block, when comparing the two SUPG blocks to the VMS block, all the terms are identical: there is one term (dash-underlined) for viscosity, two terms (dot-underlined) for stabilization and two terms for the shape functions time derivative and the advection.

5. U_iU_j block:

$$
\begin{aligned}
&SUPG\ (eq.\ (4.86)):\\
&A_{u_iu_j} = \int_{\Omega_e} \left(\underset{\cdots}{\tau_{BU}} \left(\nabla N_u \right)_i + \frac{1}{2} \left(\tilde{u}_{adv} \right)_i \left(N_u + \underset{\cdots}{\tau_{SU}} u \nabla N_u \right) \right)^T \left(\nabla N_u \right)_j d\Omega_e
\end{aligned}
$$

VMS ($eq.$ (5.18)) :

$$A_{u_i u_j} = \int_{\Omega_e} \left(\underbrace{\alpha_f \gamma \Delta t \left(\nabla N_{\dot{u}}\right)_j^T \nu \left(\nabla N_{\dot{u}}\right)_i}_{\text{- - - - - - - - - - - - - - - - - - -}} + \underbrace{\alpha_f \gamma \Delta t \left(\nabla N_{\dot{u}}\right)_i^T \tau_C \left(\nabla N_{\dot{u}}\right)_j}_{\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots} \right) d\Omega_e$$

The last block, $A_{u_i u_j}$ is interesting since there is a clear distinction between the SUPG and the VMS formulation. For the first term of eq. (4.86), supposing $\tau_{BU} \equiv \tau_C$, it can be considered equivalent to the second term of eq. (5.18). The second term of eq. (4.86) is the skew-symmetric form in the SUPG matrix while the viscous tensor in VMS is treated differently.

Eventually, although the comparison above reflects the implementation proposed by Bazilevs et al., 2007, the most important distinction is not visible and lies between the theory presented by Bazilevs and his implementation. Indeed, one cross term (eq. 5.29) in the variational form of his eq. (71) is no more visible in the implemented form (eqs. (101-106) in his article or eqs. (5.17-5.21) in current work):

$$-(\nabla w^h, \tau_M r_M(u^h, p^h) \otimes \tau_M r_M(u^h, p^h))_\Omega \tag{5.29}$$

which is a bilinear form (eq. 4.21) containing the gradient of the weight function, the stabilization coefficient for the momentum equation (eq. (5.22)), and the residual for the momentum equation, for each element.

This concludes the description of the VMS theory. Before moving to the *Results and Validation* (Part III), the theory related to the second direction, LES, will also be described.

5.2 LES

Implementing the additional cross terms for VMS proved challenging in our current framework. Because of that, it was decided to also work on the implementation of various LES formulations. Indeed, the LES formulations, known to be dissipative, could help when dealing with spurious artifacts like oscillations. The idea behind this is to reach results that could be compared to the literature and give a better insight into the limitation of LES.

The LES simulations are characterized by the direct representations of the large-scale motions and the small-scales' models. To do this, Pope, 2001 proposed a four steps description:

1. The velocity u is composed of a filtered component $\bar{u}$ characterizing the motion of the large eddies and a perturbation velocity u' corresponding to the small eddies. Thus, the velocity can be expressed as $u = \bar{u} + u'$

2. To compute the filtered velocity $\bar{u}$, a filtered version of the Navier-Stokes equations (eq. (4.3)) is introduced. The filtered equations are composed of the unfiltered version of a residual stress tensor derived from the residual motions.

3. The closure of these filtered equations implies the modeling of this residual stress tensor.

4. The filtered velocity is obtained by solving these filtered equations.

The filtering operation is represented by an overline on a variable. It is defined as the multiplication of the desired quantity (e.g., the velocity) by a filter function G_Δ, integrated over the flow domain:

$$\bar{u}(x,t) := \int G_\Delta(r,x)u(x-r,t)dr \tag{5.30}$$

The filtered function G_Δ depends on the filter width Δ (often proportional to the length of the considered element size) and has to satisfy the normalization condition:

$$\int G_\Delta(r,x)dr = 1 \tag{5.31}$$

A typical filter choice is the Gaussian filter:

$$\int G_\Delta(r)dr = \left(\frac{6}{\pi\Delta^2}\right)^{\frac{1}{2}} exp\left(-\frac{6|r|^2}{\Delta^2}\right) \tag{5.32}$$

This filter has the properties of being uniform because not depending on x, and it is isotropic for it is built on the norm of r (Pope, 2001). Moreover, it preserves linearity and allows commutativity for differentiation with respect to time and space.

Thanks to these properties, the filtered Navier-Stokes equations can be expressed as:

$$\begin{aligned} \partial_i\bar{u} &= 0 \\ \partial_t\bar{u}_j + \overline{u_i\partial_i u_j} &= 2\nu\partial_i\bar{S}_{ij} - \frac{1}{\rho}\partial_j\bar{P} + \bar{f}_j \quad with \quad j = 1,2,3 \end{aligned} \tag{5.33}$$

Reducing the filtered momentum equation to an expression analogous to the unfiltered version is possible by further developing the convection term $\overline{u_i\partial_i u_j}$. For this purpose, two groups of tools are introduced:

- Firstly, three new variables are defined:

$$\tau_{ij}^R := \overline{u_iu_j} - \bar{u}_i\bar{u}_j \tag{5.34}$$

$$\tau_{ij}^r := \tau_{ij}^R - \frac{1}{3}\tau_{kk}^R\delta_{ij} \tag{5.35}$$

$$\bar{p} := \frac{1}{\rho}\bar{P} + \frac{1}{3}\tau_{kk}^R \tag{5.36}$$

  Respectively the residual-stress tensor, the anisotropic residual-stress tensor and the modified filtered pressure.

- Secondly, an identity of the product rule, filtered and unfiltered, is suggested, simplified by virtue of the continuity equation:

$$\partial_i(u_i u_j) := (\partial_i u_i)u_j + u_i \partial_i u_j = u_i \partial_i u_j \tag{5.37}$$
$$\partial_i(\bar{u}_i \bar{u}_j) := (\partial_i \bar{u}_i)\bar{u}_j + \bar{u}_i \partial_i \bar{u}_j = \bar{u}_i \partial_i \bar{u}_j \tag{5.38}$$

By first applying identity (eq. (5.37)) to the LHS of the filtered momentum equation (eq. (5.33)), consequently applying a derivative property, substituting the residual-stress tensor (eq. (5.34)) and finally invoking the filtered identity (eq. (5.38)), one can express the LHS as:

$$\begin{aligned}
\partial_t \bar{u}_j + \overline{u_i \partial_i u}_j &= \partial_t \bar{u}_j + \overline{\partial_i \left(u_i u_j\right)} \\
&= \partial_t \bar{u}_j + \partial_i \left(\overline{u_i u_j}\right) \\
&= \partial_t \bar{u}_j + \partial_i \left(\bar{u}_i \bar{u}_j\right) + \partial_i \tau_{ij}^R \\
&= \partial_t \bar{u}_j + \bar{u}_i \partial_i \bar{u}_j + \partial_i \tau_{ij}^R
\end{aligned} \tag{5.39}$$

where the first two terms are analogous to the unfiltered LHS equation.

When bringing the last term of eq. (5.39) to RHS and considering it together with the pressure term, both terms can be reworked by injecting the expression for the anisotropic stress-tensor (eq. (5.35)), applying derivative properties and finally embedding the modified filtered pressure (eq. (5.36)):

$$\begin{aligned}
-\partial_i \tau_{ij}^R - \frac{1}{\rho}\partial_j \bar{P} &= -\partial_i \tau_{ij}^r - \partial_i \frac{1}{3}\tau_{kk}^R \delta_{ij} - \frac{1}{\rho}\partial_j \bar{P} \\
&= -\partial_i \tau_{ij}^r - \partial_j \frac{1}{3}\tau_{kk}^R - \frac{1}{\rho}\partial_j \bar{P} \\
&= -\partial_i \tau_{ij}^r - \partial_j \left(\frac{1}{3}\tau_{kk}^R + \frac{1}{\rho}\bar{P}\right) \\
&= -\partial_i \tau_{ij}^r - \partial_j \bar{p}
\end{aligned} \tag{5.40}$$

allowing the filtered momentum equation to be expressed analogously to the unfiltered equation.

Together with the continuity equation, the filtered Navier-Stokes equations, or in other words, the LES version of the Navier-Stokes equations are thus:

$$\begin{aligned}
\partial_i \bar{u} &= 0 \\
\partial_t \bar{u}_j + \bar{u}_i \partial_i \bar{u}_j &= \partial_i \left(2\nu \bar{S}_{ij} - \tau_{ij}^r\right) - \partial_j \bar{p} + \bar{f}_i \quad with \quad j = 1,2,3
\end{aligned} \tag{5.41}$$

Solving this filtered version of the Navier-Stokes equations induces a closure problem due to increased unknowns inherent to the (anisotropic) residual stress-tensors.

To close the equations, we propose two specific turbulence models:

- the Smagorinsky-Lilly model and

- the Wall-Adapting Local Eddy-viscosity (WALE) model

Historically, the Smagorinsky-Lilly model was developed before the WALE model. Practically, in *coolfluid 3*, the WALE model was implemented first because of its simplicity and promising properties.

To ease the description, the Smagorinsky-Lilly model will be presented first.

5.2.1 Smagorinsky-Lilly

The Smagorinsky-Lilly (SL) model was proposed by Joseph Smagorinsky in 1963 (Smagorinsky, 1963), after discovering, together with his colleague Douglas Lilly, one of the first but also simplest LES model to perform well (Pope, 2001).

Following previous section, the idea is to find an expression for the anisotropic residual-stress tensor $\tau^r_{ij}(x,t)$ that will enable to find the filtered velocity field $\bar{u}(x,t)$ and filtered pressure $\bar{p}(x,t)$, starting from the filtered Navier-Stokes equations (eq. (5.41)).

By supposing the mean turbulent fluctuations as dissipative, the Boussinesq hypothesis (eq. 4.11) (Berselli, Iliescu, and Layton, 2005; John, 2014; Pope, 2001) can provide a mathematical expression for the tensor:

$$\tau^r_{ij} := -2\nu_r \bar{S}_{ij} \tag{5.42}$$

where ν_r can be considered as an artificial viscosity (John, 2014) that represents the viscosity of the residual motions. Therefore, ν_r is named the residual subgrid-scale eddy viscosity.

Its generic form, for classical approaches like the SL model, is:

$$\nu_r = C_m \Delta^2 \overline{OP}(\boldsymbol{x}, t) \tag{5.43}$$

where, partially quoting Nicoud and Ducros, 1999, "C_m is a constant of the model, Δ is the [...]" filter width or "[...] subgrid characteristic length scale (in practice the size of the mesh) and OP is a [...]" chosen "[...] operator both in space and time, homogeneous to a frequency, and defined from the resolved fields".

Smagorinsky - Lilly

Explicitly, for the SL model, ν_r is function of:

- the SL lengthscale l_S, composed of the SL constant C_S and the filter width Δ:

$$l_S = C_S \Delta \tag{5.44}$$

- the local strain rate $|\bar{S}|$ as operator OP (or characteristic turbulent velocity):

$$|\bar{S}| = \sqrt{2\bar{S}_{lk}\bar{S}_{lk}} \tag{5.45}$$

It aims to replicate the transfer of energy from the resolved to the subgrid scales.

As such, the SL model expression (Pope, 2001; Sagaut, 2006) is:

$$\nu_r := l_S^2|\bar{S}| = (C_S\Delta)^2\sqrt{2\bar{S}_{lk}\bar{S}_{lk}} \tag{5.46}$$

According to Ferziger, Perić, and Street, 2002, the SL model "can be derived in many ways, including heuristic methods, for example, by equating production and dissipation of subgrid-scale turbulent kinetic energy, or via turbulence theories." The SL model will be obtained, taking under advisement this exact methodology.

1. The first step is to start from the statement of Kolmogorov, 1991 concerning the universal form of the energy spectrum function (or energy density per unit wave number k):

$$E(k) = K\langle\epsilon\rangle^{\frac{2}{3}}k^{-\frac{5}{3}} \quad with \quad K \approx 1.4 \tag{5.47}$$

where K is a constant and

$$\epsilon(t) := \frac{\nu}{|\Omega|}\int_\Omega |\nabla u|^2(x,t)dx \tag{5.48}$$

is the rate of energy dissipation per unit volume Ω, in this case, averaged statistically. Notice, for the subsequent development, that ϵ is defined as the energy dissipation rate, $\tilde{\epsilon}$ as the energy flux, and ϵ_I as the energy production rate.

Eq. (5.47) suggests that the energy contained in turbulent scales cascades from large to small scales.

2. The second step is to find an expression for the residual subgrid-scale eddy viscosity ν_r, related to the kinetic energy transfer rate $\tilde{\epsilon}[m^2/s^3]$ and to the filter width $\Delta[m]$, starting from the dimensional analysis:

$$\tau_{ij}^r = -2\nu_r\bar{S}_{ij}\left[\frac{m^2}{s^2}\right] \Leftrightarrow \nu_r\left[\frac{m^2}{s}\right] \equiv \tilde{\epsilon}^{\frac{1}{3}}\Delta^{\frac{4}{3}} \tag{5.49}$$

In Sagaut, 2006, two theories propose a relation: the Two-Fluid Model (TFM) and the Eddy-Damped QuasiNormal Markovian (EDQNM).

$$\langle\nu_r\rangle = \frac{A}{\pi^{4/3}K}\langle\tilde{\epsilon}\rangle^{\frac{1}{3}}\Delta^{\frac{4}{3}} \tag{5.50}$$

where A is a constant equal to 0.438 for TFM and to 0.441 for EDQNM (Aupoix and Cousteix, 1982).

3. The third step is to find an expression for the energy dissipation rate, function of the rate-of-strain tensor S_{ij}. This can be achieved by supposing an isotropic homogeneous case (Sagaut, 2006), where:

$$\langle 2\bar{S}_{lk}\bar{S}_{lk}\rangle = \int_0^{\frac{\pi}{\Delta}} 2k^2 E(k)dk \tag{5.51}$$

, then substituing eq. (5.47), and isolating $\langle\epsilon\rangle$ so that

$$\begin{aligned}
\langle 2\bar{S}_{lk}\bar{S}_{lk}\rangle &= \int_0^{\frac{\pi}{\Delta}} 2k^2 K\langle\epsilon\rangle^{\frac{2}{3}} k^{-\frac{5}{3}} dk \\
&= 2K\langle\epsilon\rangle^{\frac{2}{3}} \int_0^{\frac{\pi}{\Delta}} k^{\frac{1}{3}} dk \\
&= \frac{3}{2} K\langle\epsilon\rangle^{\frac{2}{3}} \pi^{\frac{4}{3}} \Delta^{-\frac{4}{3}} \\
\Leftrightarrow \left(\frac{3K}{2}\right)^{\frac{3}{2}} \langle\epsilon\rangle \pi^2 \Delta^{-2} &= \langle 2\bar{S}_{lk}\bar{S}_{lk}\rangle^{\frac{3}{2}} \\
\Leftrightarrow \langle\epsilon\rangle = \frac{1}{\pi^2}\left(\frac{3K}{2}\right)^{-\frac{3}{2}} \Delta^2 \langle 2\bar{S}_{lk}\bar{S}_{lk}\rangle^{\frac{3}{2}}
\end{aligned} \tag{5.52}$$

4. In the last step, because there is constant spectral equilibrium (following the local equilibrium hypothesis), there is no energy accumulation at any frequency. As a consequence, there is no time dependence of the energy spectrum, and therefore, there is equality between the production $\langle\epsilon_I\rangle$, dissipation $\langle\tilde{\epsilon}\rangle$ and energy $\langle\epsilon\rangle$ rates through the cutoff (Sagaut, 2006):

$$\langle\epsilon_I\rangle = \langle\tilde{\epsilon}\rangle = \langle\epsilon\rangle \tag{5.53}$$

This equality allows to substitute eq. (5.52) in eq. (5.50):

$$\begin{aligned}
\langle\nu_r\rangle &= \frac{A}{\pi^{4/3}K}\left(\frac{1}{\pi^2}\frac{3K}{2}^{-\frac{3}{2}} \Delta^2 \langle 2\bar{S}_{lk}\bar{S}_{lk}\rangle^{\frac{3}{2}}\right)^{\frac{1}{3}} \Delta^{\frac{4}{3}} \\
&= \frac{A}{\pi^2 K}\left(\frac{3K}{2}\right)^{-\frac{1}{2}} \Delta^2 \langle 2\bar{S}_{lk}\bar{S}_{lk}\rangle^{\frac{1}{2}}
\end{aligned} \tag{5.54}$$

where the SL constant can be defined as:

$$C_S := \left(\frac{A}{\pi^2 K}\left(\frac{3K}{2}\right)^{-\frac{1}{2}}\right)^{\frac{1}{2}} \approx 0.148 \tag{5.55}$$

justifying the definition for the SL model, introduced in eq. (5.46) and expressed without averaging here:

$$\nu_r(x,t) = (C_S\Delta)^2 \left(2\bar{S}_{ij}(x,t)\bar{S}_{ij}(x,t)\right)^{\frac{1}{2}} \tag{5.56}$$

This model is admitted (Sagaut, 2006) without further justification except it is verified on average and that its performances are satisfactory.

Note that the squared value of the SL constant C_S is defined as the SL coefficient and will be denoted $\tilde{C}_S$ in the next subsections: $\tilde{C}_S = C_S^2$

Value for the Smagorinsky - Lilly constant

While eq. (5.55) provides a defined value for C_S, several works produced valid simulations with values between 0.1 and 0.23 (0.15 in Pope, 2001, 0.17 in Berselli, Iliescu, and Layton, 2005, 0.18 in Sagaut, 2006).

In 1991, understanding that the C_S constant should be a function of space and time, Germano et al., 1991 proposed a version that would dynamically be adapted. This proposal was even further developed by Lilly, 1991 and gave birth to what is commonly known as the *dynamic Smagorinsky-Lilly* model.

Dynamical Smagorinsky-Lilly model

The concept, behind the dynamical Smagorinsky-Lilly model, is to filter a second time the filtered Navier-Stokes equations (eq. (5.41)), defining a second filter called the "test" filter $\widehat{\Delta}$ that is greater than the first "grid" filter Δ:

$$\begin{aligned} \partial_i \hat{\bar{u}} &= 0 \\ \partial_t \hat{\bar{u}}_j + \partial_i \widehat{\bar{u}_i \bar{u}_j} &= \partial_i \left(2\nu \hat{\bar{S}}_{ij} - \widehat{\tau^r_{ij}} \right) - \partial_j \hat{\bar{p}} + \hat{\bar{f}}_i \quad with \quad j = 1,2,3 \end{aligned} \tag{5.57}$$

where the second filtering is indicated by hats and where eq. (5.38) was applied to the LHS second term of the momentum equation.

As a result, the filtered anisotropic residual-stress tensor can be approximated by:

$$\widehat{\tau^r_{ij}} = \left[\tau^R_{ij} - \widehat{\frac{1}{3}\tau^R_{kk}\delta_{ij}} \right] \approx -2\tilde{C}_S(x,t)\Delta^2 \left[\widehat{(2\bar{S}_{lk}\bar{S}_{lk})^{\frac{1}{2}}} \bar{S}_{ij} \right] \tag{5.58}$$

Note that the exact result will be reached only if $\tilde{C}_S$ is not depending on x (John, 2014).

To find $\tilde{C}_S$, one can define the subtest-scale stress tensor, in an analogous manner to eq. (5.34):

$$K_{ij} := \widehat{\bar{u}_i \bar{u}_j} - \hat{\bar{u}}_i \hat{\bar{u}}_j \tag{5.59}$$

as well as its anisotropic version:

$$K_{ij} - \frac{1}{3} K_{kk}\delta_{ij} = -2\tilde{C}_S(x,t)\widehat{\Delta}^2 \left(2\hat{\bar{S}}_{lk}\hat{\bar{S}}_{lk} \right)^{\frac{1}{2}} \hat{\bar{S}}_{ij} \tag{5.60}$$

and finally, the Germano identity (Breuer, 1998):

$$L_{ij} := K_{ij} - \widehat{\tau_{ij}^R} = \widehat{\bar{u}_i \bar{u}_j} - \hat{\bar{u}}_i \hat{\bar{u}}_j \tag{5.61}$$

With these tools defined, the anisotropic version of the Germano identity can be expressed as:

$$\begin{aligned} L_{ij} - \frac{1}{3} L_{kk}\delta_{ij} =& \left(K_{ij} - \widehat{\tau_{ij}^R}\right) - \left(\frac{1}{3}K_{kk}\delta_{ij} - \frac{1}{3}\widehat{\tau_{kk}^R}\delta_{ij}\right) \\ \approx& -2\tilde{C}_S \left(\widehat{\Delta}^2 \left(2\widehat{\bar{S}}_{lk}\widehat{\bar{S}}_{lk}\right)^{\frac{1}{2}} \widehat{\bar{S}}_{ij} - \Delta^2 \widehat{\left(2\bar{S}_{lk}\bar{S}_{lk}\right)^{\frac{1}{2}}\bar{S}_{ij}}\right) \\ =& -2\tilde{C}_S M_{ij} \end{aligned} \tag{5.62}$$

with

$$M_{ij} := \widehat{\Delta}^2 \left(2\widehat{\bar{S}}_{lk}\widehat{\bar{S}}_{lk}\right)^{\frac{1}{2}} \widehat{\bar{S}}_{ij} - \Delta^2 \widehat{\left(2\bar{S}_{lk}\bar{S}_{lk}\right)^{\frac{1}{2}}\bar{S}_{ij}} \tag{5.63}$$

Unfortunately, eq. (5.62) provides five independent equations for one unknown making it impossible to find a value for $\tilde{C}_S$.

The trick of Lilly, 1991 is to minimize the square of its error by means of a least-square method. Defining Q as the square of the error:

$$Q = \left(L_{ij} - \frac{1}{3}L_{kk}\delta_{ij} + 2\tilde{C}_S M_{ij}\right)^2 \tag{5.64}$$

One can find $\partial Q/\partial \tilde{C}_S$:

$$\begin{aligned} \frac{\partial Q}{\partial \tilde{C}_S} &= 2\left(L_{ij} - \frac{1}{3}L_{kk}\delta_{ij} + 2\tilde{C}_S M_{ij}\right) 2M_{ij} \\ &= 4L_{ij}M_{ij} - \frac{4}{3}L_{kk}M_{ll} + 8\tilde{C}_S M_{ij}M_{ij} \\ &= 4L_{ij}M_{ij} + 8\tilde{C}_S M_{ij}M_{ij} \end{aligned} \tag{5.65}$$

with $M_{ll} = 0$ (due to the filtered continuity equation in eq. (5.41) making $\widehat{\bar{S}}_{ll} = \bar{S}_{ll} = 0$).

$\tilde{C}_S$ is found by $\partial Q/\partial \tilde{C}_S = 0$:

$$\tilde{C}_S(x,t) = -\frac{L_{ij}M_{ij}}{2M_{ij}M_{ij}}(x,t) \tag{5.66}$$

Note that the second derivative is positive, confirming it is a minimum:

$$\frac{\partial^2 Q}{\partial \tilde{C}_S^2} = 8M_{ij}M_{ij} > 0 \tag{5.67}$$

Backscattering and numerical instabilities

A nice side effect of the double filtering is that the SL coefficient $\tilde{C}_S(x,t)$ can have negative values (due to not being squared like the SL constant $C_S(x,t)$). By this, it diverges from the classical SL model and allows energy to flow from small to big scales: backscatter is allowed (John, 2004).

However, due to possible substantial variations in space or high negative values of $\tilde{C}_S$, the numerical solution can be very unstable. To limit these instabilities, both the nominator and denominator of eq. (5.66) can be averaged or limited in time (Breuer, 1998; Lesieur, 2007; Sagaut, 2006).

5.2.2 WALE

Another LES model that was considered is the Wall-Adapting Local Eddy-viscosity model (WALE). This model was proposed by Nicoud and Ducros (Ducros, Nicoud, and Poinsot, 1998; Nicoud and Ducros, 1999) in 1999 to circumvent three limitations classical LES approaches, like the Smagorinsky-Lilly model (§ 5.2.1), do have:

1. The first limitation, is related to the eddy-viscosity ν_r, or more specifically, to the chosen operator OP (eq. (5.43)).

   Choosing the local strain rate (eq. (5.45)) implies that the subgrid dissipation is directly connected to the strain rate of the smallest resolved scales while Wray and Hunt (Wray and Hunt, 1990) could demonstrate that the energy dissipation concentrates in eddies where vorticity dominates (in place of irrotational strain), and in convergences zones where irrotational strain prevails.

   Therefore, the operator OP should take both the local strain rate $\bar{S}$ and the rotational rate $\bar{\Omega}$ into account.

   Note: $\bar{\Omega}$ is also named the anti-symmetric part of the velocity gradient tensor $\bar{g}$ (to oppose to the symmetric part $\bar{S}$ from eq. (4.4)) and is defined as:

   $$\bar{\Omega}_{ij} = \frac{1}{2}\left(\partial_j \bar{u}_i - \partial_i \bar{u}_j\right) \tag{5.68}$$

2. The second constraint, in the classical method, comes from test filtering that can be difficult to implement with complex geometries (Jansen, 1994). The WALE model proposes a solution that depends neither on translation nor on rotation, thus, invariant for both.

3. The third issue, for the classical LES approach, is that the eddy-viscosity ν_r does not vanish near the wall, although all turbulent fluctuations should be damped. To solve this issue, damping functions (e.g., Van Driest, 1956, Piomelli, 2008; Piomelli and Balaras, 2002, or the mixing length from Prandtl, 1927) were proposed for classical

methods. However, these functions provided a ν_r that is proportional to $O(y^2)$ while the natural diminishing at walls is $O(y^3)$.

Moreover, as a result of the chosen operator OP, the WALE model is numerically well-conditioned (here: $\nu_r \geq 0$). This implies an unambiguous and efficient implementation (Banyai, 2016; Nicoud and Ducros, 1999).

Eventually, to briefly review the modifications proposed by Nicoud and Ducros and to propose the WALE model:

- the two first issues are solved by defining a modified strain rate tensor that takes both rotation and irrotational rate into account while being invariant on any translation or rotation. Thus, the retained version is based on the traceless symmetric part of the square velocity gradient tensor $\bar{\bar{g}}$:

$$S^d = \frac{1}{2}\left[\bar{g}^2 + (\bar{g}^2)^T\right] - \frac{1}{3} I \bar{g}^2 \tag{5.69}$$

  with I the identity matrix.

- the third is tackled by chosing the third power of the new operator OP and then, normalize it:

$$\nu_{sgs} = \left(C_w \Delta^2\right) \sqrt{\frac{(S^d_{ij} S^d_{ij})^3}{(S_{ij} S_{ij})^5}} \tag{5.70}$$

  However, due to its denominator capable of reaching zero (e.g. in vortex centres), this new eddy-viscosity is not well conditionned, numerically. To easily get around this matter, a stabilization term was added to the final Wall-Adapting Local Eddy-viscosity (WALE) model:

$$\nu_{sgs} = \left(C_w \Delta^2\right) \sqrt{\frac{(S^d_{ij} S^d_{ij})^3}{(S_{ij} S_{ij})^5 (S^d_{ij} S^d_{ij})^{\frac{5}{2}}}} \tag{5.71}$$

  with the WALE coefficient commonly fixed to $C_w = 0.325$, corresponding, for an isotropic turbulence, to a SL coefficient $C_s = 0.1$ (Banyai, 2016), via the relation:

$$C_w = \sqrt{10.6 C_s^2} \tag{5.72}$$

This concludes the theory related to the last LES turbulence model implemented in this work, and, at the same time, it is also the last chapter of the *Modeling* Part.

It is now time to move to the *Results and Validation* Part.

Part III

Results and Validation

Chapter 6

From broad view to ABL implementation

The first part of this monograph, *Context* (Part I), introduced the circumstances in which this work was initiated, but also the specificities related to the environment and, in a broader sense, to the type of simulation that had to be performed.

The second part, *Modeling* (Part II), provided the theory that is required to handle these simulations, taking into account the existing software, *Coolfluid 3*, its advantages but also its constraints. It also elaborated on the valuable tools to circumvent the complications specific to ABL modeling with FEM.

In this third part, *Results and Validation*, the tests that were performed, the encouraging results that were achieved, but also the disappointing results that brought us to investigate other paths; are all described for the reader to have a deeper insight on what are the challenges that were encountered and how these were apprehended.

Notice that since the specifications of the different cases will be provided, the results will mainly be presented in local units (i.e. dimensionalized) to favor the physical sense. In chapter 7, some results will be non-dimensionalized to enable the comparison with the literature.

6.1 Preamble on Coolfluid 3

Before diving into the results, it is necessary to briefly outline how a typical simulation test case in *Coolfluid 3* works.

The *Coolfluid 3* package is written in the programming language `C++`, which enables:

- The creation of a code fully dedicated to, in this case, the simulation of CFD problems. In other words, it means a Domain Specific Language (DSL) can be built.
- The generation of a computationally efficient code, due to both its low hierarchical level (i.e. straightforward conversion into assembler code, that is, the computer core language) and its standard library implementation, known for its cleanliness and efficiency. With these two assets, the code can then be compiled into a binary form that improves the delivered software's performance even more.

Thus, *Coolfluid 3*, compiled from `C++`, represents the core code. The latter is then queried by an input file written in a language named Python that is less performant but more accessible. Consequently, it has a wider community. This python input file will contain all the test case parameters to be simulated (e.g. the physical properties of the fluid, the governing equations to choose, the boundary and initial conditions, the data to output, etc.). A typical example of such a python file is available in appendix A.3.

In this work, the theoretical implementations are performed in the core code, in `C++`, but the test case will be defined through a python input file.

In the following sections, some `C++` implementations will be described, and sometimes, a python input file will also need to be introduced because the latter, often with specific considerations, activates the former.

This being written, we can introduce the first simulations.

6.2 Geometry and physical properties

Geometry

In this work, the complete domain, illustrated in figure 6.1, has a height of $H = 1200m$ (with the height adopted as the distance unit), a length of nearly six times the height ($L = 2\pi H \approx 7540m$), and a width approximated by three times the height ($W = \pi H \approx 3770m$). These dimensions are commonly selected to represent an ABL domain (Bechmann, 2006; Goit, 2015).

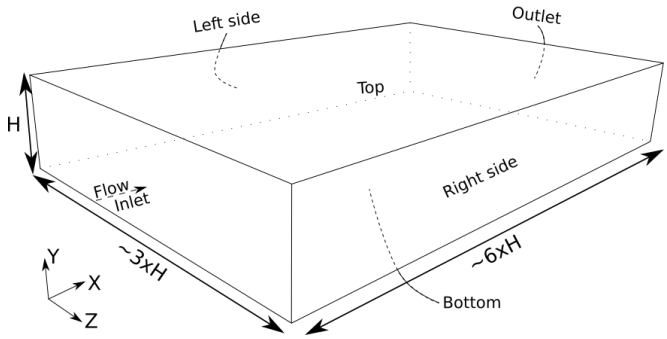


Figure 6.1: Dimensions of the domain.

Physical properties

The considered fluid is the air following standard NTP conditions, with a density $\rho = 1.2kg/m^3$ and a dynamic viscosity $\mu = 1.8e{-}5Pa\,s$. For the second phase, although only relevant for the next section, a density of $\rho_p = 50\mu g/m^3$ (cf. PM10 from chapter 2) was chosen.

6.3 The broad view

The author decided to initiate the work by creating a domain in 3D, including a source emitting a second phase in a developed airflow. The objective of this simulation was to identify potential problems when applying our code to real-life dispersion problems. It would exhibit the potential complications that would need investigation.

The primary purpose of this section is not to attach importance to the quantities but to identify the points of interest.

6.3.1 Setup

Thus, for the first domain, the dimensions are reduced (i.e. $H = 1.2m$) to facilitate the simulation, and a source is positioned longitudinally (i.e. in the $x-$direction, streamlined with the flow), at a two heights distance from the inlet. Because the idea is to simulate a source originating downstream of a building without having to account for the upstream turbulences, the shape of the source is extended until the inlet boundary. In the $y-$direction, it is placed as close as possible from the bottom while ensuring an acceptable meshing quality[1]. Laterally, in the $z-$direction, positioning the source in the middle of the domain allows a spherical propagation.

Boundary conditions

To give the numerical model its physical meaning, boundary conditions have to be set consistently. In this model, periodicity is applied to the inlet, outlet, and sides of the model. In other words, the input is connected to the output, the left side to the right side. A symmetry condition is applied to the top (i.e. the topside should be far enough, not to influence the domain of interest). For the bottom, a no-slip condition (i.e. nul velocity on the ground) is set.

Initial condition

To initiate the simulation, a flow condition is fixed, equal to a rudimentary uniform velocity ($u = 4m/s$).

Mesh creation

The first mesh is generated with hexahedral elements to enable a structured mesh to facilitate the computation and reduce the simulation time. The smallest length is set to $1mm$ and progressively increased with a scale factor of 1.4. Unfortunately, in a structured mesh, each interior node has the same amount of elements in its surroundings, forcing the refined hexahedra's

[1] The quality of the mesh structure is measured through analyzing the skewness of each element

local resolution to extend in the three dimensions. As a consequence, when considering a refinement on the wall locations, after several adjustments, the first grid is reduced from 25 to 15 million of elements (figure 6.2), even though such a grid is impracticable for analysis. Moreover, the solution is hardly computable, even with the available computational resources.

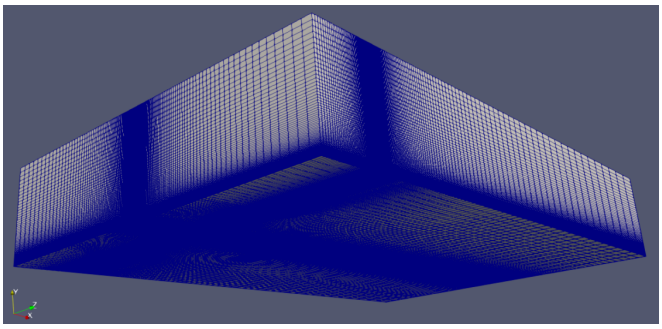


Figure 6.2: Hexahedral mesh including a source.

To reduce the number of elements, an option is to replace the hexahedral elements with tetrahedral elements at the cost of replacing the structured mesh with an unstructured mesh. The latter requires a remapping of each element in the system to be solved. Thus, using tetrahedral elements enable to refine in regions of interest (i.e. high gradient, separation, etc.) and progressively unscale in regions of dissipation (i.e. where interactions with the interfaces are reduced). In our case, a scaling factor of 1.4 is chosen (although 1.1 is preconized) to enable both the insertion of a cylindrical source and the ability to rapidly increase the size of the elements when moving away from the source, its support, and the bottom. The resulting mesh[2] is given in figure 6.3 and contains 5 millions of elements.

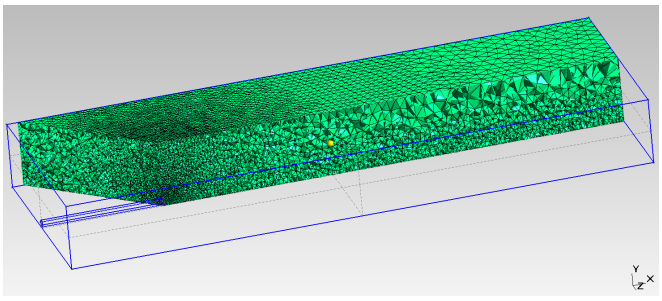


Figure 6.3: Tetrahedral mesh including a source.

[2]Note that for this specific case, the dimensions were modified (i.e. $\sim 9H$ x H x $\sim 3H$) to correlate with another study (Bechmann, 2006), but the thought remains equivalent.

Note that this mesh is by no means relevant to describe an ABL properly, for three obvious reasons:

- Both the height ($H = 1.2m$ vs. $H_{real} = 1200m$) and the uniform velocity profile ($u_{inlet} = 5m/s$) are not realistic.
- Observing the size of the elements at the interfaces, in the $y-$ and $z-$directions, the pressure gradient will not have the span to decrease sufficiently, implying potentially spurious reflections at the interface.
- The dimension of the elements on the ground does not reflect the velocity gradient to be expected, normal to the ground.

Although critical, these observations are helpful to precise the preliminary complexity of the case.

6.3.2 Results

Four additional pieces of information are brought by inspecting qualitatively the first timesteps of the converged simulation. The two firsts are visible in figure 6.4.

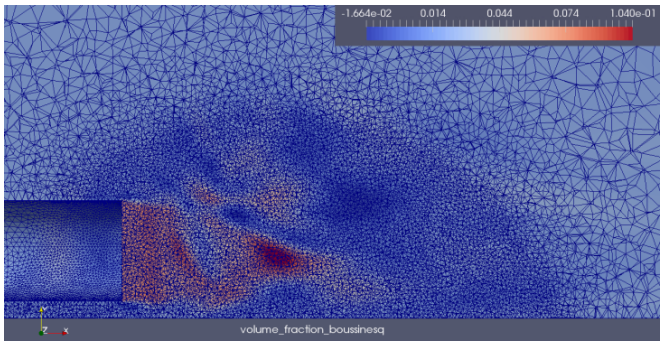


Figure 6.4: Vertical cross-section of the boussinesq volume fraction for the second phase, $0.4s$ after release.

- First, the geometry of the source outlet presents some similarities with a well-known academic case, namely, the backstep flow, which is already subject to numerous studies due to the complexity brought by both the separation and the backscatter flow.
- Second, a significant amount of elements is required because of the space left below the cylindrical shape of the source.

These two observations raise questioning on the necessity to represent the source in such a manner.

The third and fourth details are discernible in figure 6.5.

- The third observation is related to the dispersion that looks symmetrical relative to the horizontal plane. Since both the boundary conditions and the source geometry are symmetrical relative to the horizontal plane, there is no reason to believe this observation would not be accurate. Consequently, slicing the model in two, following the symmetry plane, would reduce the number of elements of the simulation by two. However, two issues will then arise:
  - turbulences is a 3D phenomenon that needs three dimensions to develop;
  - unsymmetrical boundary conditions (that can occur in an ABL, implied by a Coriolis effect) will not be appropriately captured.

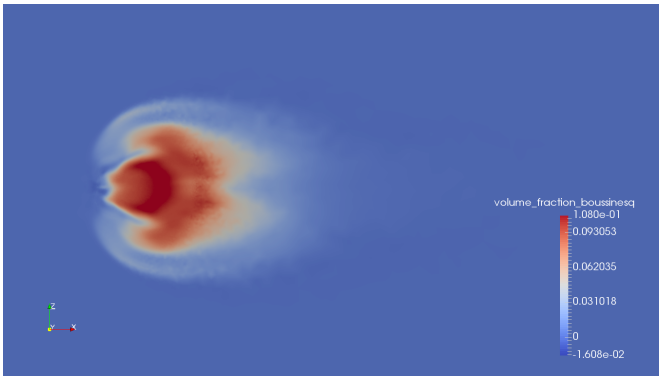


Figure 6.5: Horizontal cross-section of the volume fraction for the second phase, $1s$ after release.

- The fourth observation is noticeable near the source outlet (i.e. pipe extremity). One can perceive a circular propagation pattern that is embracing the outlet's contour. Certainly, the geometry upwind of the source will influence the dispersion pattern. Since the primary goal is to study the influence of the atmospheric boundary layer on the dispersion of a second phase, it would be advantageous to avoid any superfluous obstruction.

Area of focus

Eventually, although incomplete, these qualitative examinations exposed lots of subjects of investigations that are not all directly related to our subject. Therefore, we decided to focus on the fundamental stage: create a proper flow, taking the size and the specificities of the ABL into account, while leaving the geometries and second phase behaviors for further investigations.

The following section will thus present a less sophisticated domain that will allow us to develop the ABL theory.

6.4 The ABL simulation

6.4.1 Setup

Geometry and mesh

As suggested in the previous section, for this section, the geometry is reduced to figure 6.1 containing only the fluid phase (no source, neither support included), and the results will be provided in 2D. Moreover, while the more sophisticated mesh of the previous section was prepared with an external mesher (i.e. *Gmesh*), the more accessible mesh of the following sections will be prepared via the *Coolfluid 3* internal mesher to discard a source of error (namely, the surface creation and node matching).

Moreover, the generated mesh will have a $64 \times 64 \times 64$ and a $64 \times 128 \times 64$ elements distribution (i.e. size) with a reduced scale factor (i.e. 1.1) to smooth the progression.

ABL expression

In this domain, to avoid requiring many elements (cf. § 4.3) and because in FEM, we use a grid where all quantities are computed on the nodes, the concept of a first cell layer is ill-suited to our problem. For this reason, we started the mesh at a height y_{wall} from the wall, where y_{wall} plays the role of first cell height. Figure 6.6 illustrates it.

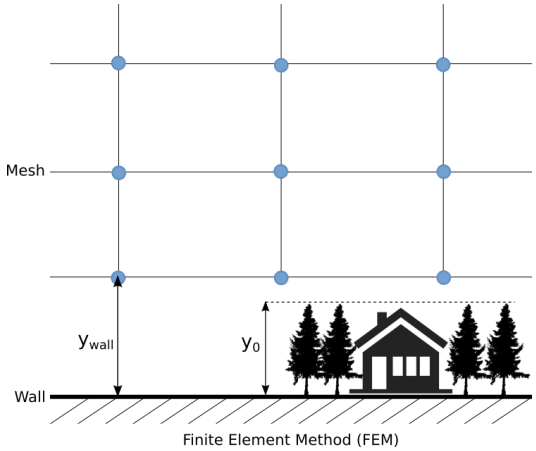


Figure 6.6: Illustration of the 1^{st} node above a wall.

Since most of the literature and code in CFD is written for FDM and FVM (cf. § 4.2.2), our development has its origin in a FDM implementation of the wall model, proposed by Goit, 2015. The latter proposes a LES modeling where the boundary condition incorporates a Schumann's wall model. This expression, derived from Bou-Zeid, Meneveau, and Parlange, 2005, is an improved local law-of-the-wall formulation:

$$\tau_w = -\left[\frac{\kappa}{\ln(y_{wall}/y_0)}\right]^2 \left(\bar{u}_1^2 + \bar{u}_2^2\right) \tag{6.1}$$

with τ_w the shear stress at the wall, y_{wall} is the height of the first node, and y_0 is the roughness of the considered terrain. Thus, this formulation is used in LES of a high-Reynolds number boundary layer, where the viscous sublayer is not resolved.

From the general definition of the shear stress (eq. (6.2)):

$$\tau_{w_i} = \nu \frac{\partial u_i}{\partial x_j} n_j \quad \rightarrow \quad \frac{\partial u_i}{\partial x_j} n_j = \frac{\tau_{w_i}}{\nu} \tag{6.2}$$

and from the definition of a natural Robin boundary condition (eq. (6.3)):

$$\beta u + \frac{\partial u}{\partial n} = 0 \quad \rightarrow \quad \frac{\partial u}{\partial n} = -\beta u \tag{6.3}$$

Because eq. (6.1) is a function of the velocity (i.e. $\tau_w = f(u)$), applying eq. (6.3)) leads to an expression where the wall model appears in the system matrix, represented using the FEM shape functions N as follows:

$$\int_\Gamma \frac{1}{\nu} \left(\frac{\kappa}{\log \frac{y_{wall}}{y_0}}\right)^2 |u| N^T(u) N(u) |\bar{n}| d\Gamma = 0 \tag{6.4}$$

Boundary conditions

Similar to the previous section, the current model will also have a symmetry condition applied to the top and periodicity on the inlet, outlet, and sides.

In contrast, for the bottom, the no-slip condition is replaced by a non-penetration condition (i.e. impermeability, that is, the vertical component of the velocity set to $0 m/s$) coupled to the just defined Schumann's wall-stress boundary condition from eq. (6.4).

Initial condition

For the initial condition, to accelerate the convergence, the previous uniform velocity is replaced by a logarithmic velocity profile, more representative for an ABL profile:

$$u_x^{inlet} = \left(\frac{u_\tau}{\kappa}\right) \log\left(\frac{y + y_{wall}}{y_0}\right) \tag{6.5}$$

The reference velocity is 4 m/s at a height of 200 m. A timestep of 0.1 s was taken (0.05 s sec in 3D) to ensure stability and convergence of the simulation (i. e. a CFL [3] below 0.2 in this case).

Remark 6.1. *In the following sections, the reference velocity can slightly vary around $4m/s$. The grounds behind it are associated with the literature that was taken into account during the examination. Although any reference velocity from this order of magnitude could have been considered to feign a highly convective flow, its value will be indicated each time there is any adjustment to avoid any confusion.*

Body force

Eventually, the model and its boundaries being defined, a body force f is set to keep the ABL at a constant speed. This body force can be found by requiring equilibrium between the force F_f and the force F_w induced by the wall shear stress:

$$\begin{aligned} F_f = fV\rho \quad \& \quad F_w = A_w\tau_w \\ if \quad F_f = F_w \quad \rightarrow \quad f = \frac{\tau_w}{\rho h} \end{aligned} \tag{6.6}$$

and

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}} \quad \rightarrow \quad f = \frac{u_\tau^2}{h} \tag{6.7}$$

with h the height of the model.

Expressed in terms of speed on the first node, derived from eq. (6.1):

$$u_\tau = \frac{\kappa u_{ref}}{\ln((y_{ref} + y_{wall})/y_0)} \tag{6.8}$$

6.4.2 Results

Flow pattern

The logarithmic profile of the flow can qualitatively be seen on figure 6.7. The velocity range varies between 0.58 m/s (on the first node) near the wall and 4.9 m/s at the -top of the model.

The reference speed was 4 m/s at a height of 200 m. Hence, at 1200 m (height of the model), the speed should reach around 4.86 m/s to be consistent with the logarithmic profile.

Velocity profile

By looking closer at the velocity profile in the center of the domain (fig. 6.8), one can confirm the supposition quantitatively.

[3] i.e. Courant-Friedrichs-Lewy condition, that is, a common stability criterion for hyperbolic equations.

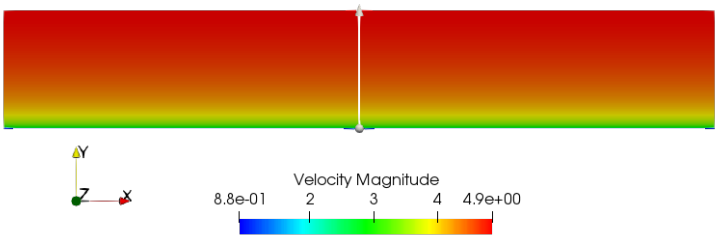


Figure 6.7: Velocity countour map with the new ABL implementation.

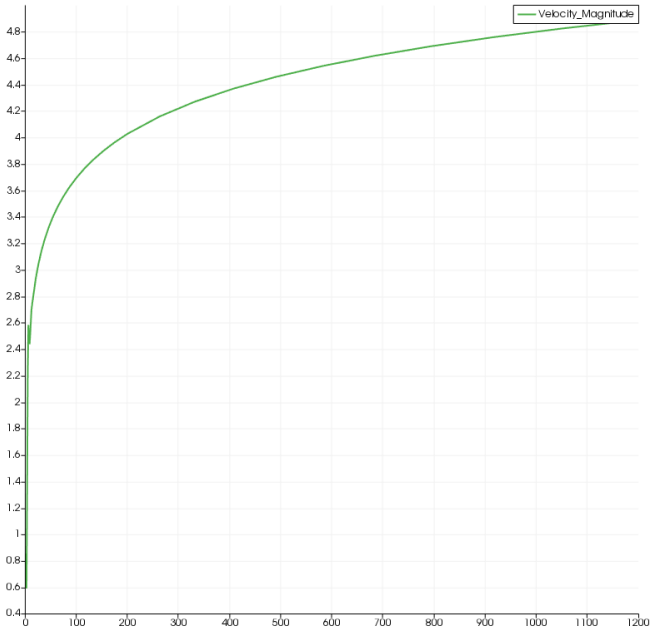


Figure 6.8: Velocity magnitude $[m/s]$ vs y-coordinate [m]

As can be seen in figure 6.8, the velocity plot starts above 0.4 m/s, pass through 4 m/s at the reference height (200 m) and show a logarithmic profile as suspected by previous map.

Sensitivity to the mesh-wall distance

As mentioned in § 6.4.1, to reduce the required number of elements, Schumann's wall model permits an offset between the physical wall and the mesh. If the boundary condition is implemented correctly, the velocity at the first node as a function of this offset should follow the theoretical logarithmic profile.

We performed seven nearly identical simulations to validate this approach. The only difference between each of these simulations was the position of the first node relative to the wall.

By plotting the velocity computed in the first node of each of these simulations and compare it to the theoretical ABL profile, the correctness of the implementation could be validated.

Figure 6.9 presents the results of these seven simulations (seven colored crosses) and compares them to the analytical profile (dash line).

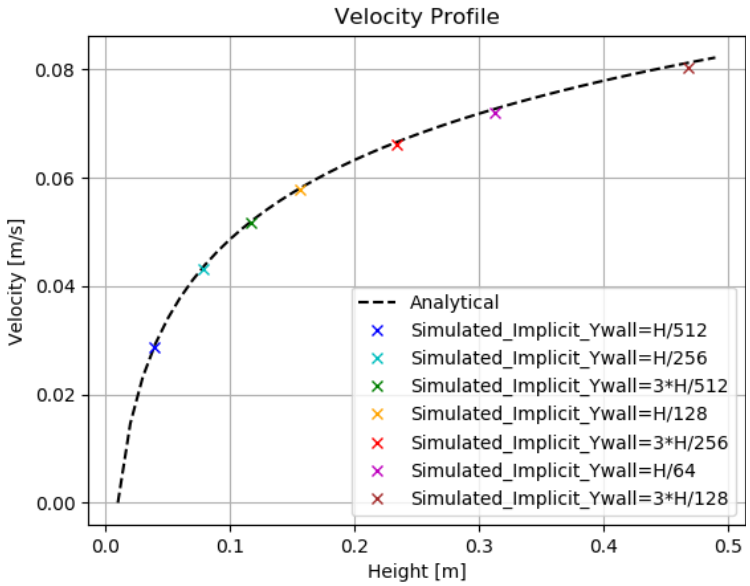


Figure 6.9: 1^{st} node velocity for 7 simulations compared to theoretical value.

One can see the analytic value from eq. (6.8) is nearly equal to the value computed by the simulation, on the 1^{st} node (the most significant deviation is 0.4% at $0.47m$ from the wall). This confirms that our translation of Schumann's model to FEM is implemented correctly.

This brings confidence in both the approach and its implementation, but of course, it is limited to a 2D verification.

The next step is to ensure the boundary condition is stable both in function of the number of elements and in function of the simulation time. This brings us to the following two sections.

6.5 Boundary condition normalization

With the newly implemented wall model boundary condition, it is important to ascertain that the number of elements defining the domain will only

influence the resolution of the solution and not alter the absolute physical values.

We can verify this with two smaller grids (i.e. 4-elements and 16-elements grids on a $20m$ height domain) without waiting for the final converged result. In figure 6.10, the coarse grid has an orange color; the refined grid has a purple color.

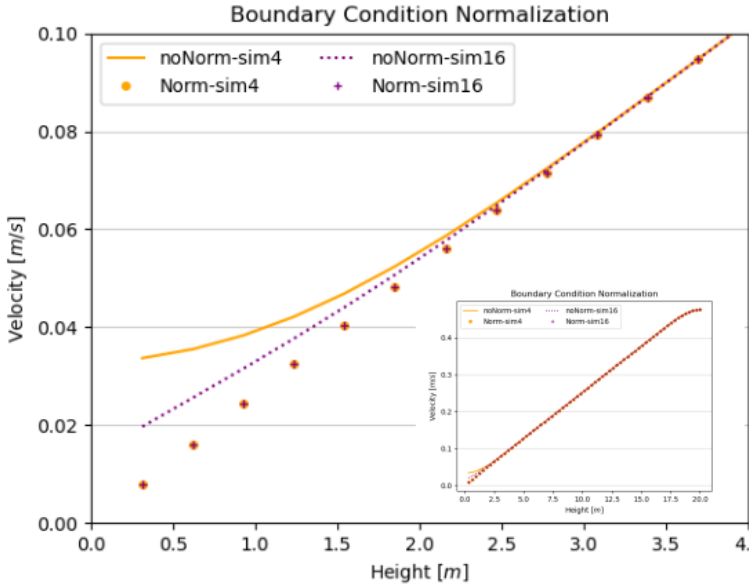


Figure 6.10: Boundary condition normalization. (Inside graph is a full profiles' view)

After a few iterations, although the solution is not converged, one can already observe a slight deviation between the coarse and the refined grids on the first nodes (near the wall).

When normalizing the solution by its first node value, the two grids produce identical values (i.e. dotted values). This optimization is the first helpful step toward a robust spatial solution. The following sections will present a second step heading toward a stable solution in time.

6.6 SUPG coefficients

When applying the body force introduced in eq. (6.7), the system should be excited until the body force achieves a certain equilibrium with the shear stress velocity u_τ. Practically, it is the case in 2D.

However, when moving to 3D, with an unsteady viscous flow, it will be shown that the body force causes the velocity profile to increase continu-

ously, exceeding the reference velocity. As a consequence, no convergence can be reached.

The first idea is to adjust the stabilization method coefficients (α_{SU}, α_{BU}, and α_{PS}) and analyze their influence on the resulting velocity in a position located in the center of the domain, at mid-height, and compare them to the original velocity.

The coefficients α_{SU}, α_{BU}, and α_{PS} are constant values that can manually be adjusted to give more weight respectively to the streamline upwind (SU), the bulk (BU), and pressure (PS) stabilization schemes. In our specific implementation, the pressure is not directly adjusted. Therefore, α_{PS} will not be analyzed.

In the conventional implementation, these coefficients, or gain, are all set to 1.0. The resulting velocity profile, on the last figure 6.13 will be displayed in dotted black. However, before comparing the different results to the regular profile, the influence of the gain coefficients will be examined.

For this purpose, figure 6.11 presents the evolution of the instantaneous velocities for three different simulations for respectively three increasing values of α_{BU}: 0.0 (i.e. deactivation of the BU stabilization), 0.1, and 0.5.

The chosen location for the velocity is the center (i.e. height of $600m$) of the domain, for a reference velocity of $5m/s$ (at $200m$).

Figure 6.12 presents the same evolution, for two increasing values of α_{SU}: 0.1 and 0.5. (no deactivation is necessary to draw the conclusions).

For these two figures, the velocities for the first iterations (from 0 to 32000), slightly more transparent, were part of a first simulation. They are only shown to confirm the velocity started below $5m/s$ and continued its progression through and over the reference value.

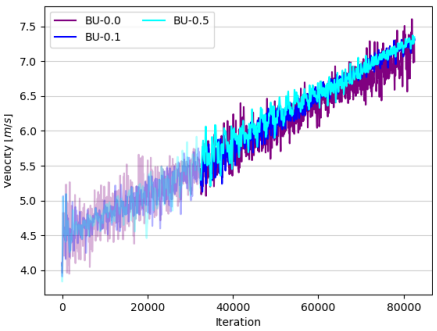


Figure 6.11: BU effects.

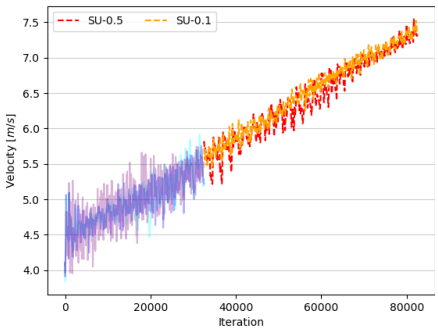


Figure 6.12: SU effects.

Figure 6.11 brings three observations upfront:

- All three velocities follow the same linear and increasing progression. In other words, no velocity tends to a stabilized constant velocity.

- The velocity from the deactivated BU scheme shows the largest oscillation's amplitude, the one with the greatest α_{BU}, the narrowest.
- The mean values are different.

Figure 6.12 is placed next to figure 6.11 because the observations are analoguous:

- a linear progression;
- this time the smallest α_{SU} corresponds to the narrowest oscillation amplitude;
- α_{SU} has less influence on the oscillation's amplitude then α_{BU}.

From these two figures, one could deduce that these gain factors will unfortunately not bring a solution to the over-rated velocities arising with an increase in simulation time (i.e. number of iterations).

Nevertheless, it is still interesting to analyze how these various velocities compare to the original velocity profile. They are all given through their respective linear regression lines in figure 6.13. (The raw signals were too noisy to draw conclusions, hence the filtered representation.)

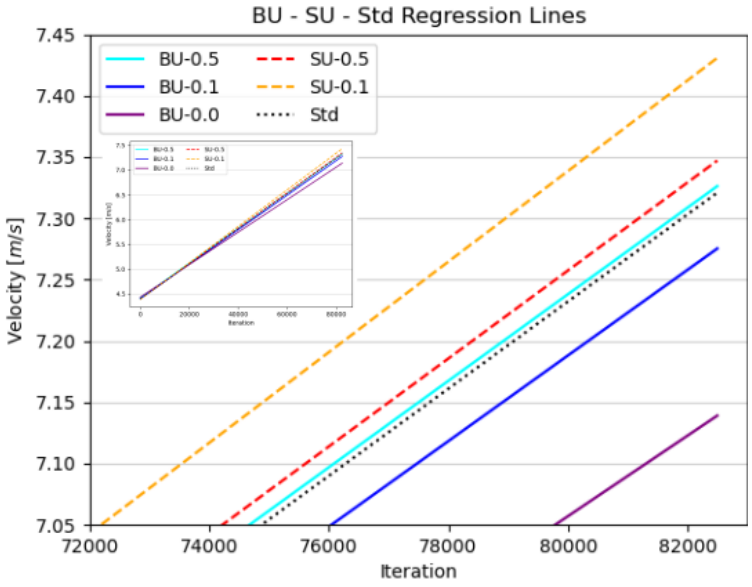


Figure 6.13: Mean SU - BU effects compared to standard case. (Inside graph is a full profiles' view.)

Figure 6.13 presents the trendlines for each considered case, compared to the original case (i.e. dotted black). One can observe that an increasing

weight on the SU scheme tends to decrease the velocity in the direction of the standard case. An advisable option to continue in the decreasing trend is to reduce the BU scheme effect.

Unfortunately, the decrease keeps the angular coefficient of the increasing velocity positive. No improvement is to be expected with these coefficients. Therefore, we decided to remain with the original case.

Remark 6.2. *The confidence interval for each of these lines was computed to confirm the proper linear assumption for these trendlines and, because of the equal distribution of the sampled velocities (i.e. one per iteration), without surprise, the confidence interval is above* 0.95*.*

6.7 Body force limiter

A solution to counter this non-expected behavior is implementing a limiter that would act on the body force according to the maximum velocity expected at the reference height, namely the reference velocity.

Ordinary limiter

An ordinary first implementation of a body force limiter is visible in figure 6.14 where a reference velocity of $3.75m/s$ was applied. In this simple limiter, the body force is first computed thanks to the requested velocity and then recalculated at each timestep, with a correction factor *corr* given by eq. (6.9).

$$corr = \left(\frac{u_{ref} - u_{mean}}{\|u_{ref}\|} \right) \tag{6.9}$$

For the sake of illustration, we set a probe at the reference height and monitored its velocity during the simulation. Figure 6.14 presents the mentioned velocity in function of the time (actually, the iterations). The dotted line corresponds to the trendline.

As the simulation progresses, the velocity should reach its reference value, namely $3.75m/s$. When the velocity computed at the reference height exceeds the requested value, the body force counterbalances it, not taking the previous timesteps into account (no feedback loop). Consequently, the computed velocity tries to return to the requested value but with an opposite derivative, without damping. Figure 6.14 shows the saw teeth behavior of this first implementation.

The saw teeth behavior of the body force limiter has two non-negligible drawbacks:

- The time needed to reach convergence is far from optimal. In reality, it could even be theoretically infinite since oscillation could occur

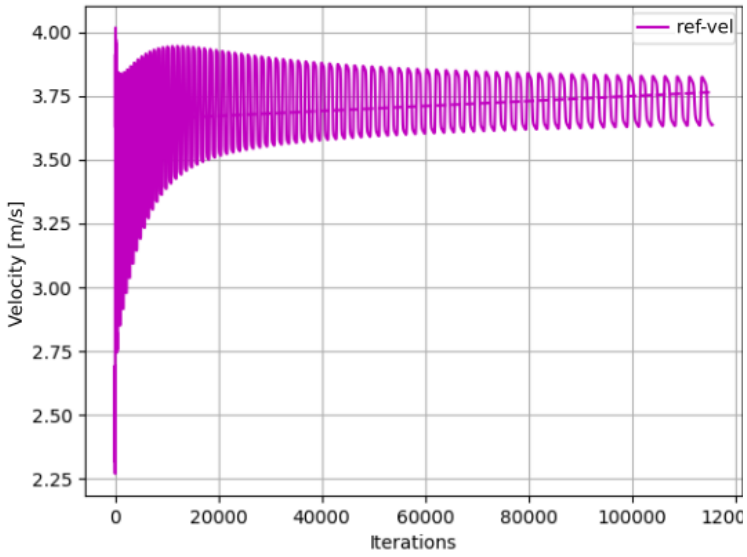


Figure 6.14: Ordinary implementation of a limiter

indefinitely the smaller the deviation becomes. This is noticeable between iterations 40.000 and 120.000, where the progression given by the trendline (i.e. dotted line) follows a slow path to the targeted value. Moreover, the oscillation amplitude does not seem to diminish anymore.

- These oscillations produce discontinuities between each iteration that will increase potential numerical instabilities (or trigger them if not yet present) and deteriorate the quality of the resulting flow.

Thus, this first proposal is not an acceptable solution. In contrast, the same idea, without the oscillatory behavior, could still contribute to improving the simulation. Consequently, the following subsection will present a more advanced limiter.

Dynamic limiter

In this second implementation, two thoughts are taken into account:

- to use the information of the last iteration to control the direction of the future body force.
- to smooth the response by not limiting the information to the very last iteration but consider time-averaged information of multiple iterations.

Accordingly, the idea is to consider the history of the previous iterations before updating the required body force. Put differently, the body force is

once more first computed thanks to the requested velocity and subsequently iteratively updated using a feedback scheme that compares the velocity from the computed flow to the requested velocity.

For this purpose, the proposal of Goldstein, Handler, and Sirovich, 1993 was implemented. The adapted body force has following pattern:

$$f(x_s, t) = \alpha \int_0^t u(x_s, t')dt' + \beta u(x_s, t) \tag{6.10}$$

where a large negative α coefficient will result in a fast response and a large negative β coefficient in a greater damping. The coefficients of the feedback loop expression are therefore named the gain, and the damping. For our case, the optimal coefficient are found to be respectively $-1.$ and -0.2.

Eventually, to include the history of multiple previous iterations and not only the last one, the time integral can be approximated by a Riemann sum:

$$\int_0^t u(x_s, t')dt' \approx \sum_{j=1}^{N} u(x_s, j)\Delta t \tag{6.11}$$

where N is the number of steps and Δt is the size of the timestep.

The next step is to verify the appropriate response of the new limiter. We can ascertain this by inserting three probes set at different heights (bottom, center, and top of the domain), following their behavior in function of the time (i.e. iterations), and simultaneously inspecting the flow velocity. The body force is constant in the entire domain.

Figure 6.15 displays the velocity at the three probes, with a reference velocity set at $5m/s$. In this plot, the three velocities are rapidly reacting to the influence of the body force. Moreover, the damping is clearly observable on the probe located at mid-height.

At first instance, it has met the two criteria expressed at the beginning of this section. The resulting method does facilitate the flow convergence and stabilizes the velocity profile more quickly.

To confirm this behavior is correlated to the action of the body force, figure 6.16 presents the body force intensity for the same simulation.

As expected, the intensity of the body force is high in the first steps, allowing a fast alteration of the physical properties of the model. Then, it decreases rapidly with the number of iterations, a consequence of the limiter, enforcing a reduction and damping of the body force intensity.

Remark 6.3. *The curious reader can find the exact implementation in appendix A.1.*

However, when computing the body force analytically, the obtained value is equal to $3.584e{-}3N/kg$ while the converged numerical value is $3.077e{-}4N/kg$.

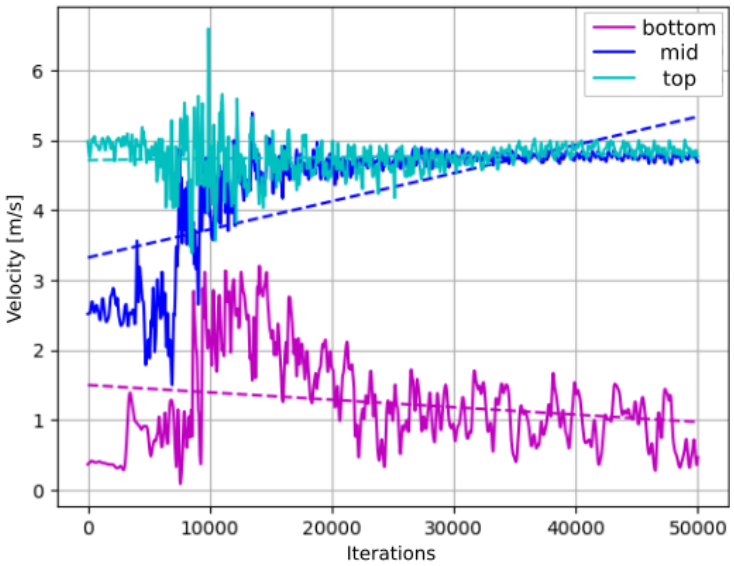


Figure 6.15: Limiter with feedback scheme: Velocities at 3 different heights (-); trendlines (- -)

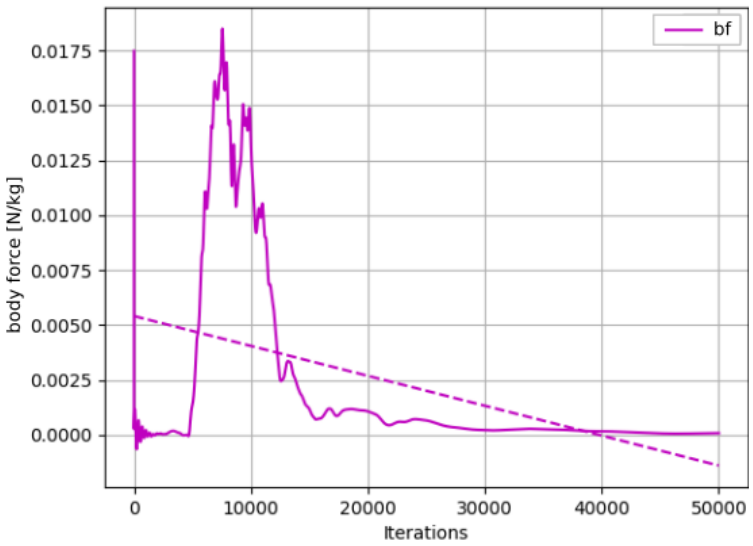


Figure 6.16: Limiter implementation with feedback scheme: body force value in the whole domain

The factor 10 is due to the limiter's use (which is doing what he was re-

quested to do). This difference primarily means that the shear stress is much lower than expected, indicating a severe problem with the solution.

6.8 Spurious oscillations

To better understand and observe the phenomenon, the simulation was run further on both a simple and a full-scale model, until statistical convergence could be obtained. Figure 6.17 illustrates visually the phenomenon that is occurring near the wall.

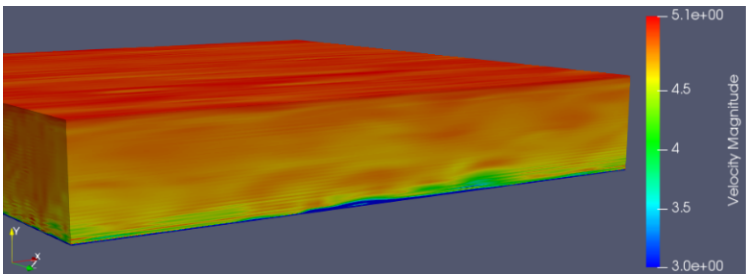


Figure 6.17: Velocity contour.

One can observe horizontal lines near the wall that are undoubtedly not physical.

To quantify and confirm this artifact, figure 6.18 provides two types of graphs:

- outside graph: two velocity profiles in the center of the domain (statistical in purple; instantaneous in red).
- inside graph: the velocity in function of the time (or iterations) for three locations (i.e. bottom, mid, and top of the domain).

The outside graph exhibits an instantaneous velocity profile where the high oscillation near the wall (left side of the curve) is apparent.

Moreover, to support even more these two first inspections, on the inner graph, the velocity capture for each timestep at the bottom of the domain confirms this spurious and significant oscillation.

Remark 6.4. *Notice that the values on the nodes were verified, and the presented oscillation is, unfortunately, no postprocessing artifact; the computed values are effectively affected by spurious numerical oscillations.*

These spurious oscillations that are not physical remind us of § 4.2.3. The latter explained how FEM implementations are sensitive to convective flows and how, basically, the PSPG/SUPG stabilization method was developed to counter these.

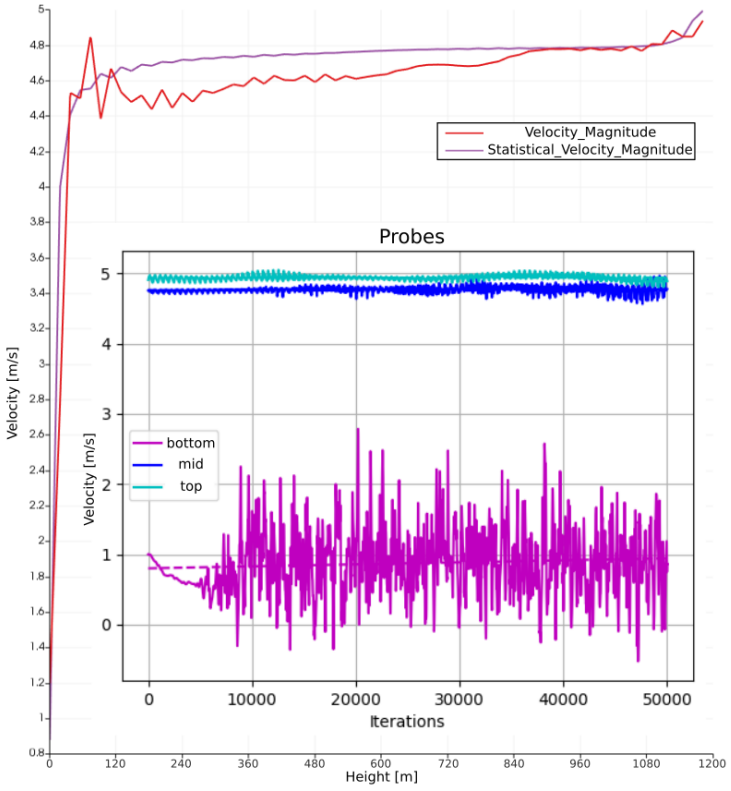


Figure 6.18: Instantaneous and statitical velocity profiles (outside) ;
Velocities at 3 heights in function of the iterations (inside).

Potentially, in the case of an ABL modeling, the PSPG/SUPG method can not be applied without considering both the numerical implementation and perhaps some extra turbulences modeling.

Remark 6.5. *Another interesting observation is that the velocity at the reference height (i.e.* $200m$*) is not the reference velocity (i.e.* $4m/s$*) but a velocity close to the velocity at the top of the domain. As a matter of fact, the profile is not logarithmic.*

This over-estimated velocity can be seen on the statistical velocity (purple line) at the reference height and through the mid-position velocity history (dark blue line), which is very close to the top-position velocity history (cyan line).

Although essential, this observation will be left aside while the investigation will focus on the spurious oscillations.

6.9 Implicit vs semi-implicit

Following the previous section, before proposing any turbulence modeling, the first idea is to ensure that the ABL implementation has no side effect on the numerical resolution of the system. In this respect, two versions of the ABL implementation will be developed. It will provide two parallel paths to solve the simulation's computation and hopefully reach the same result.

If the oscillations remained, these two alternatives would confirm there is no correlation between the oscillations and one of these paths. If the oscillation were to disappear, it could bring a solution.

Regardless of the oscillative behavior, this first investigation will provide more insight and better tools to understand the phenomenon.

Thus, as described in chapter 4, the introduction of convective terms (e.g. in the momentum equation) within a FEM approach results in the use of stabilization methods. As a matter of fact, the existence of convective terms has another important implication:

Combined with the need to solve the governing equations on a large number of elements, for a significant amount of timesteps, a few simulation codes propose an explicit handling of the convective terms because the latter is fast and boosts the scalability. Indeed, in the explicit approach, the term is handled on the RHS of the governing equation as a source term, avoiding inner iterations to find the solution for each timestep. However, more severe timestep restrictions (cf. CFL condition) also constrain the explicit approach. In this regard, such a type of handling will require smaller timesteps to obtain a converged result, but in a more significant amount.

Another option is to propose an implicit approach, where the pressure and the velocities are coupled. The latter considers the convective terms inside the assembly matrix, leading to a significant increase in computational cost per timesteps (because of the inner iteration required for the coupling). However, on the other side, the restriction on the timesteps is no more relevant, enabling to increase the size of these steps, and therefore reduce the number of timesteps. In the case of FEM, the poor conditioning of the linear system leads to poor convergence of the linear system solution.

For non-linear PDEs, an even more advantageous technique, implemented in *Coolfluid 3* by Janssens, 2014 makes use of an implicit scheme for linear terms and an explicit scheme for non-linear terms (González-Calderón, Vivas-Cruz, and Herrera-Hernández, 2018). This method combines a less-restrictive use of the timestep to a more efficient (read less computationally costly) resolution per timestep. This last version is a hybrid named the semi-implicit version. In the context of LES and DNS, this can be justified because resolving turbulence also requires small time steps.

In this section, we will present two adaptations of our case (following the implicit and the semi-implicit approaches). For this purpose, the typical main structure of the wall model implementation first needs to be described. Its implementation is partially visible in appendix A.2 to keep the section

lightweight.

Description of the wall model implementation

Appendix A.2 provides two files that define the ABL component that is included in the FEM resolution, inside the *Coolfluid 3* code:

- The first file is the header file.

  In short: It contains the declarations for all variables and functions that are part of the newly defined ABL component.

  In long: It starts with the necessary libraries (#1-4) and accessible namespaces (#6-9). Then, it defines the new ABL class (#11). This class contains a public section including :

  - (#16) the conventional constructor and,
  - (#19) destructor, but also
  - (#22) the name identification of the class and
  - (#25) one function to enable the execution of the component, when activated.

  The class also contains a private section (i.e. only callable inside the ABL component) that holds the most important information, namely:

  - (#29) one function that associates the component to a specific boundary region (in our case, the region will be the bottom of the domain to study);
  - (#32) one function that contains the core of the wall function implementation. It will be further detailed when describing the second file;
  - (#35-36) the objects containing the assembly matrix (also named the system matrix) and the RHS source term;
  - (#37-43) all the numerical and physical parameters required for the ABL implementation.

- The second file presents the fragment of the primary definition file that contains the ABL implementation itself.

  Longer version:

  - (#3) It starts with some calls to the the actual *WallLaw* component,
  - (#5-9) and the physical quantities of the latest timestep.
  - (#11-15) Then, a small function will compute the ABL coefficient (eq. (6.1)) to avoid having to recompute the same value for each element.

- (#18-35) Finally, the core process is carried out in the *wall_law* → *set_expression* function.
  - (#20) In this task, first the type of FEM element that can be considered will be defined. These could be piecewise linear triangular elements, piecewise linear hexahedral elements, or other available elements.
  - (#21-34) Then, the assembly matrix as well as its RHS have to be composed. These are needed to solve the linear system that will deliver the physical quantities (e.g. velocity and pressure) for each element, at each timestep.

It is precisely the last item (#21-34) from the second file that will diverge in the two approaches. This block will be named the *expression block* to ease the reading.

Implicit approach

For the implicit version, as introduced at the beginning of this section, the idea is to include the effect of the ABL inside the assembly matrix that is part of the linear solver system (LSS) associated with the velocity and coupled to the pressure.

To achieve this, two modifications needs to be applied:

1. The *expression block* displayed in listing 1 needs to be properly implemented.
2. The python inputfile (§ 6.1) for the *Coolfluid 3* simulation test case needs to activate the implicit implementation.

Thus, first, we describe the specific *expression block* implementation:

- (#3) At each timestep, the first stage is to initialize the assembly matrix for both the velocity and the pressure.
- (#4-8) Then, each element is examined and attributed two relations:
  - (#6) The first relation imposes a no-penetration condition by deducting the contribution of the pressure on the velocity.
  - (#7) The second relation enforces the ABL wall model condition by appending the ABL coefficient to the velocity following the exact expression (6.1).
- (#9) After having looped on each element, the system matrix can be completed by appending the assembly matrix, adjusted with a coefficient θ (from the θ-method detailed in Janssens, 2014).
- (#10) Equivalently, the RHS will deduct the effect of the assembly matrix to the unknown vector.

```
1  group
2  (
3    _A(u) = _0, _A(p) = _0,
4    element_quadrature
5    (
6      _A(p, [u_i]) += -transpose(N(p)) * N(u) * normal[_i],
7      _A(u[_i], u[_i]) += _norm(u) * transpose(N(u)) * N(u)
       ↪  * _norm(normal) * lit(dt()) * ABL_factor()
8    ),
9    system_matrix += m_theta * _A,
10   rhs += -_A * _x
11 )
```

Listing 1: Implicit expression block.

Second, the python input file needs to select an implicit implementation of the Navier-Stokes equations and choose an LSS optimized for such a system. Listing 2 illustrates these portions.

In listing 2, line (#2) selects the general version of the Navier-Stokes equations. This version is the only possible for the implicit execution. Then, line (#5-23) specify the type and parameters needed for the LSS associated to the implicit configuration:

- (#6-15) Select the preconditionner and enter its parameters (e.g. type of matrix resolution).
- (#17-23) Select the LSS type and the parameters required to restrain the computation (i.e. convergence criterium, number of iterations and allocated memory).

The LSS options are part of an external package called *Trilinos* that is incorporated to *Coolfluid 3*. More details on this package can be found in Sala et al., 2010.

Semi-implicit approach

By comparison, the approach for the semi-implicit direction is fundamentally different since the ABL effect needs to be converted into a source term. We can then transfer the new source term to the RHS, together with the contribution of other potential source terms (e.g. heat source).

Once more, the first step is to rewrite the *expression block* following listing 3. The steps are equivalent to the implicit version, except the expressions are modified:

```
### Add the Navier-Stokes solver as an unsteady solver
ns_solver =
↪  solver.add_unsteady_solver('cf3.UFEM.NavierStokes')

# Implicit solver setup for NS
for lss in [ns_solver.LSS]:
  lssParam = lss.SolutionStrategy.Parameters
  lssParam.preconditioner_type = 'ML'

  lssML = lssParam.PreconditionerTypes.ML.MLSettings
  lssML.add_parameter(name='ML output', value=0)
  lssML.default_values = 'NSSA'
  lssML.aggregation_type = 'Uncoupled'
  lssML.smoother_type = 'symmetric block Gauss-Seidel'
  ↪   # 'Chebyshev'
  lssML.smoother_sweeps = 2
  lssML.smoother_pre_or_post = 'post'

  lssBelos = lssParam.LinearSolverTypes.Belos
  lssBelos.solver_type = 'Block GMRES'

  lssGMRES = lssBelos.SolverTypes.BlockGMRES
  lssGMRES.convergence_tolerance = 1e-5
  lssGMRES.maximum_iterations = 2000
  lssGMRES.num_blocks = 1000
```

Listing 2: Implicit portion of a typical case.

- (#3) The initialization is no more done on the assembly matrix for both the velocity and the pressure, but on the pressure assembly matrix and the velocity component of the RHS vector.
- (#7) For each element, the non-penetration condition remains unchanged. However, the ABL wall model condition is this time appended to the RHS velocity vector. Moreover, we slightly adapted its formulation by multiplying it by the velocity components while respecting the matricial product rules.
- (#9) The system matrix operation is identical to the implicit version.
- (#10) The RHS, in contrast, will only deduct the effect of the whole RHS vector.

The second step is to adjust the python inputfile to the semi-implicit

```
group
(
  _A(p) = _0, _a[u] = _0,
  element_quadrature
  (
    _A(p, [u_i]) += -transpose(N(p)) * N(u) * normal[_i],
    _a[u[_i]] += ABL_factor() * _norm(u) * transpose(N(u))
     ↪  * u[_i] * _norm(normal) * lit(dt())
  ),
  system_matrix += m_theta * _A,
  rhs += -_a
)
```

Listing 3: Semi-implicit expression block.

formulation, following listing 4.

Briefly:

- (#2) The semi-implicit formulation of Navier-Stokes is selected.
- (#3-6) This new implementation requires to precise a few options (e.g. the coefficient for the θ-method from Janssens, 2014, the velocity-pressure interaction, the use of a body force).
- (#9-27) Analogously to the implicit version, the linear systems need a solver to be defined.
  - (#9-13) First a solver for the pressure part is defined (here, *Amesos-KLU*) and its parameters are filled.
  - (#15-27) Secondly, the velocity part is handled, by defining the preconditionner, the LSS solver, and their respective parameters.

Once more, all these solvers derive from the *Trilinos* algebra package. The curious reader will find all the available solvers with their options detailed in Sala et al., 2010.

These descriptions aim to illustrate the importance of adapting the test case to the chosen implementation.

Outcome

To conclude this section, we have implemented two versions of the wall model. Theoretically, these two versions are fundamentally different in their resolution:

```
### Add the Navier-Stokes solver as an unsteady solver
ns_solver = solver.add_unsteady_solver('cf3.UFEM.NavierStokesSemiImplicit')
ns_solver.options.theta = 0.5
ns_solver.options.nb_iterations = 2 # 1 if isolate velocity computation from pressure
ns_solver.options.enable_body_force = True
ns_solver.options.pressure_rcg_solve = True # True with Amesos ; False with Amesos_Klu

### solver setup for SI
for strat in [ns_solver.children.FirstPressureStrategy, ns_solver.children.SecondPressureStrategy]:
  strat.MLParameters.aggregation_type = 'Uncoupled'
  strat.MLParameters.max_levels = 4
  strat.MLParameters.smoother_sweeps = 2
  strat.MLParameters.coarse_type = 'Amesos-KLU'

lss = ns_solver.VelocityLSS.LSS.SolutionStrategy
lss.preconditioner_reset = 1#20000000

lssParam = lss.Parameters
lssParam.preconditioner_type = 'Ifpack'
lssParam.PreconditionerTypes.Ifpack.overlap = 0

lssBelos = lssParam.LinearSolverTypes.Belos
lssBelos.solver_type = 'Block CG'

lssCG = lssBelos.SolverTypes.BlockCG
lssCG.convergence_tolerance = 1e-6
lssCG.maximum_iterations = 300
```

Listing 4: Semi-implicit portion of a typical case.

- One considers the ABL wall model part of the assembly matrix (implicit version). This implies an alteration of its shape. As a reminder, the sparser the matrix is, the more complex and computationally demanding its resolution will be. On the other hand, this implicit version enables increasing the timesteps' size, facilitating the time progression in the simulation.
- In contrast, the semi-implicit version simplifies the preliminary work

by relaying the ABL wall model to a source term on the RHS, making its implementation more accessible, but constraining the simulation to the CFL condition.

However, the resulting simulation should provide the same outcome.

After letting our simulation test case run with the two implementations, no difference was observable in the results, reducing the two implementations to a prerogative left to the user's preference.

This also supports the fact that the spurious numerical oscillations exposed in § 6.8 are not impacted by the method used to solve the linear system.

This direction of investigation can therefore be closed.

Chapter 7

Turbulent model investigations

By successfully developing Schumann's wall model, the previous chapter delivered a potential implementation of the ABL, supporting a substantial reduction of the number of elements necessary to define the domain. As a matter of fact, it was ascertained that the first nodes of the grid do not need to correspond with the physical wall and that spacing them from the wall will imply a proper adjustment of the velocity that complies with the expected analytical value.

Moreover, a body force limiter was implemented to ensure the requested reference velocity would always bound the computed velocities.

However, because spurious numerical oscillations occur near the wall, because these take place in a region where turbulence is non-negligible, and because turbulence models tend to have a dissipative behavior, it was decided to investigate the effect of turbulence modeling on the ABL velocity profile.

In this chapter, the first sections will examine two LES turbulence models that were introduced in § 5.2.1 and § 5.2.2.

Subsequently, the most appropriate model will be selected to initiate several sensitivity studies on various criteria.

To allow a solid foundation for the future works, the complete model will be adapted to enable a comparison with a DNS resolution.

Eventually, the premises to work on the VMS implementation will be introduced.

In consideration of the preceding, before diving into the turbulence investigations, a unique tool needs to be introduced: the randomizer.

Remark 7.1. *In § 6.4.2, the reference height was set to $h_{ref} = 200m$ for a reference velocity $v_{ref} = 4m/s$, corresponding to a velocity at the top of the domain reaching $\sim 5m/s$.*

To simplify the indicator for the following results, the reference height was set at the top of the domain (i.e. $h_{ref} = 1200m$), and accordingly, the reference velocity was set to $5m/s$.

7.1 Preamble to 3D turbulent modeling: The randomizer

When simulating an ABL with a realistic dimension, even with the help of a wall model, the number of elements necessary to fill the domain is considerable. When starting from a perfectly homogeneous domain (and a priori not the final solution), with the help of the initial and boundary conditions, the transition to a developed flow can require a significant amount of timesteps for the perturbations to propagate in every single element and even more timesteps to get some turbulence to build up.

To accelerate the process, a tool named the randomizer was implemented in *Coolfluid 3*. By initiating a random, directed, and bounded excitation, the randomizer is designed to produce an inhomogeneous domain that will facilitate the creation of turbulences.

Its activation is given in appendix A.2 (#208-215). The steps are:

- to create the randomizer component,
- to map it to the grid,
- to associate the randomizer to a physical quantity (in our case, the velocity), and finally
- to provide its intensity and maximum variation in each direction.

The resulting velocity contour when initiating the simulation is illustrated in figure 7.1.

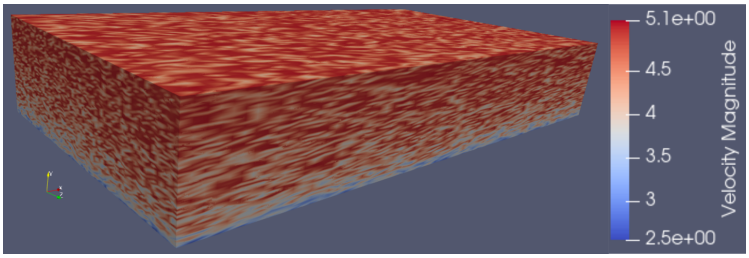


Figure 7.1: Randomizer effect at initialization.

At first instance, this section could seem futile. However, in reality, the variation on the reference velocity, the direction of activation, and foremost, the randomizer implementation (i.e. independent of the quantity to analyze) can substantially impact the simulation convergence time. This will be further commented in § 7.5.

In the meantime, the randomizer will be used to speed up the results provided in the following sections.

7.2 LES WALE

Thus, the first turbulence model that is being tested is the Wall-Adapting Local Eddy-viscosity (WALE) model. As emphasized in § 5.2.2, it was designed to circumvent the weaknesses of more traditional LES models.

Because its implementation was partially present in the existing code, it was decided to finalize it and test it on our case.

Implementation

The important part of the implementation, the computation of the WALE turbulent viscosity, is transcribed in appendix A.4. Briefly, it is composed of:

- (#7-10) the definition of tools (operators and matrix types).
- (#12-24) the computation of velocity gradient, Fröbenius norms, and broadly, factors needed for the WALE viscosity expression below.
- (#26) this is the important expression of the implementation, where the artificial WALE viscosity is assembled. It represents the more conventional expression (5.71).
- (#27-28) a small safety condition to remove negative values.
- (#30) the integration of the WALE viscosity into each element's node.
- (#33) the last constant is user-defined and commonly fixed to 0.325, following eq. (5.72).

Results

To analyze the influence of the artificial dissipation delivered by the WALE model, it was thus included in our ABL system, and to better understand the consequences, figure 7.2 presents the velocity profile[1] in the center of the domain, at different timesteps (equivalent to iterations).

As explained in the previous section, the simulation starts with a randomized velocity field, producing inhomogeneity. The effect is still observable on the dotted profile given after three iterations. Obviously, the profile will converge with the number of iterations. This is shown by the dashed curves (i.e. 20 to 500 iterations).

After 1300 iterations, the profile is not yet converged, but already, tangible observations can be made:

- no oscillation is visible near the wall. The turbulence model did positively influenced our case.

[1] Notice for this test case, the logarithmic profile was not activated. However, the observations are equivalent.

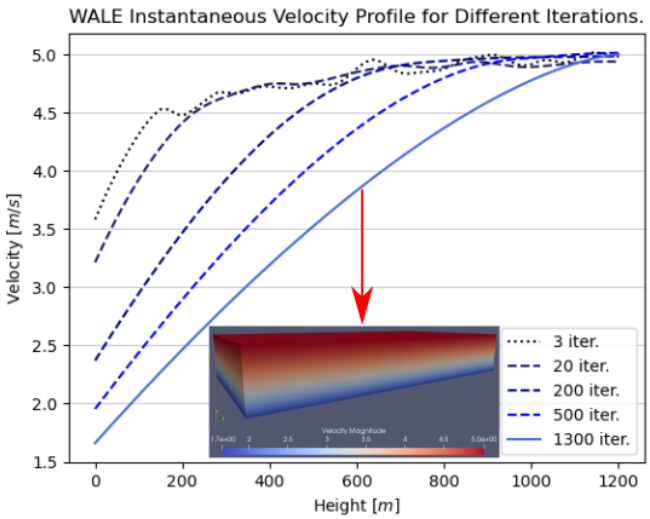


Figure 7.2: WALE: Instantaneous velocity profiles per iteration; laminarised velocity contour for last iteration (inside).

- the profile is dramatically laminar. The velocity contour for this last timestep (inside figure 7.2) pictures it even better.

The latest observation is certainly not desirable and is the consequence of too much dissipation.

As explained in § 5.2.2, the model was optimized from a more traditional LES turbulence model and is therefore also less easy to adjust manually.

The logical next step is thus to revert to a different LES turbulence model: the Smagorinsky - Lilly model.

7.3 LES Smagorinsky - Lilly

In § 5.2.1, the Smagorinsky - Lilly (SL) model was presented as one of the oldest LES turbulence models. Nevertheless, it is still commonly used due to its simplicity and yet satisfying results. Consistently with all LES models, it proposes an artificial viscosity expression that will contribute to close the governing equations for a small dissipation cost. Once more, the question will be how advantageous is this dissipative effect for our ABL implementation.

Two versions of this model were presented in the theoretical part. Their implementation and the resulting velocity profiles for our case will be the subject of the following subsections.

7.3.1 Static version (or Cs constant)

Alike in WALE's section, we will first describe the implementation given in apppendix A.5. Then, we will look at the resulting velocity profiles to review the effect of the model on the spurious oscillations and, most importantly, on the resulting ABL velocity profile.

Implementation

As previously, only the meaningfull part of the implementation will briefly be described. The idea is to display the correlation between the theoretical part (§ 5.2.1) and the FEM implementation.

Thus, succinctly:

- (#7-8) Alike in WALE, a factor is foreseen for the anisotropy of each element (but not detailed here).
- (#10-12) Not used here. It will be explained in § 7.3.3.
- (#18-20) The definition of tools (operators and matrix types).
- (#22) To define the velocity in the element's center, it has to be computed from the surrounding nodes.
- (#24-26) The static SL method requires the grid filtered volume as well as the characteristic filtered width Δ_{gF} to be computed. It is done relative to the considered element's volume.
- (#27-29) The computation of the velocity gradient and Fröbenius norm of the strain tensor are needed for the static SL expression below.
- (#31-94) Not used here. It will be explained in § 7.3.3.
- (#95) This is the first important expression. The subgrid-scale length scale is computed from the manually inserted C_S coefficient and multiplied to a function of the grid filter width. It follows eq. (5.46). Notice that by default, the C_S value is set to 0.148 (§ 5.55).
- (#97-100) One of the SL drawbacks is related to its incapacity to reach the no-slip condition on the wall. We implemented the Mason wall damping approach (Mason and Thomson, 1992) here to circumvent this.
- (#102-104) This is the important expression of the implementation, where the artificial SL viscosity is assembled. It represents the more conventional expression (5.46).
- (#106-107) A safety condition to remove negative values.

```
smag =
↪  solver.add_unsteady_solver('cf3.UFEM.les.Smagorinsky')
smag.options.use_dynamic_smagorinsky = False
smag.options.cs = 0.148
```

Listing 5: Static SL activation in *Coolfluid 3* python input file.

- (#109-114) The integration of the SL viscosity into each element's node.
- (#117-125) The last lines define the used parameters but also the boolean used to de/activate the different options like the static or dynamic SL (that will be analyzed in the next subsection).

After having implemented the static SL model, it was integrated into our ABL system by adding following lines (listing 5) to the python test case.

Hopefully, the simulation will modify the resulting velocity profile positively.

Results

Analogously to the WALE's section, the resulting velocity profile is presented in figure 7.3. Two profiles are displayed: the instantaneous velocity in the center of the domain, at the last simulated time step (dashed line), and the statistical velocity (plain line) computed on the last $3e5$ iterations.

Fortunately, both the instantaneous and the statistical velocity are following a reassuring path. The former displays a turbulent behavior starting at a non-zero velocity (as expected from the wall model implementation) and moving towards the reference velocity. The latter presents a smooth curvature that recalls the expected ABL profile.

To better emphasize the turbulent response present in the instantaneous velocity profile, the velocity contour for the last time step is given in figure 7.4

Qualitatively, the velocity contour exhibits turbulences that do not present the spurious horizontal oscillations near the wall (neither elsewhere). Moreover, the velocity remains in the expected range: between the supposed non-zero velocity at the first node above the wall (bottom) and around the reference velocity at the top of the domain.

Remark 7.2. *This simulation was performed on a* $64\times64\times64$ *grid, with the first nodes' vertical position located at* $y_{wall} = 12.5m$ *(that is, the total height divided by the number of elements) ; a considered roughness of* $y_0 = 0.01m$ *(corresponding to a very smooth rural terrain) ; a timestep of* $1s$ *; and a SL coefficient set to the constant value* $C_S = 0.148$.

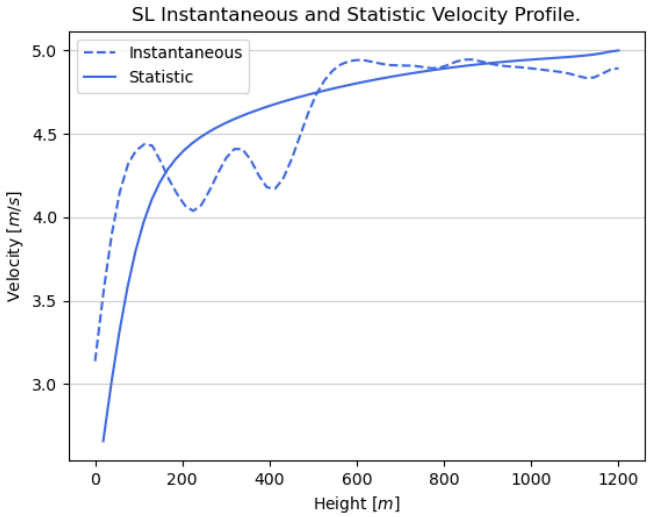


Figure 7.3: SL static model: Velocity profiles after $6e5$ iterations.

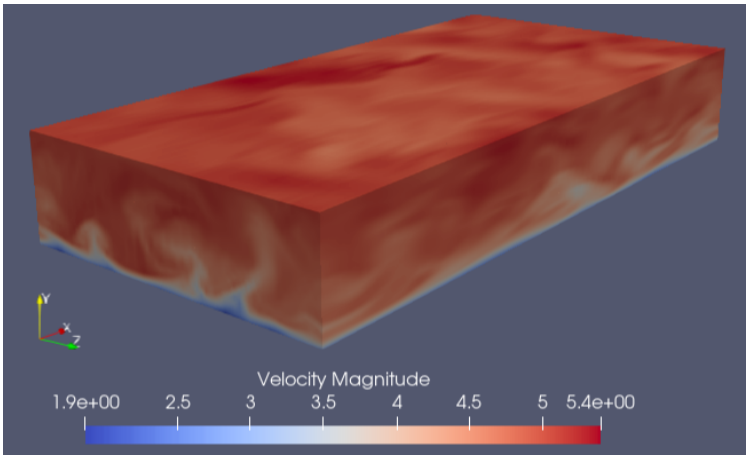


Figure 7.4: SL static model: Instantaneous turbulent velocity contour after $6e5$ iterations.

Having presumably found a solution to both the spurious oscillations near the wall and the convergence issue, the profile is numerically satisfactory but has not yet proven to be physical.

The next section will study the influence of the SL coefficient and compare the result to the expected analytical (i.e. physical) logarithmic profile.

7.3.2 Cs constant: Sensitivity study

In previous section, the C_S value was set to 0.148 following eq. (5.55). However, the literature (§ 5.2.1) also demonstrated that this constant value is actually not performing properly for all cases. Practically, the value is not constant, nor in time, nor spatially. Vasaturo et al., 2018 even stated: "Concerning the Smagorinsky constant, C_S, there is neither a consensus nor a clear guideline about the value to adopt for the simulation of the ABL." He was referring to the C_S values of 0.1, 0.12, 0.16 for the respective works of Thomas and Williams, 1999, Hu, Ohba, and Yoshie, 2008, and Tseng, Meneveau, and Parlange, 2006.

Therefore, in the first step, the idea is to study the influence of C_S for values varying between 0.00 and 0.20 by steps of 0.02. Figure 7.5 presents the resulting velocity profiles, after reaching the statistical convergence (here, $6e5$ iterations). Additionally, the expected analytical ABL velocity profile is included to situate the computed profiles better.

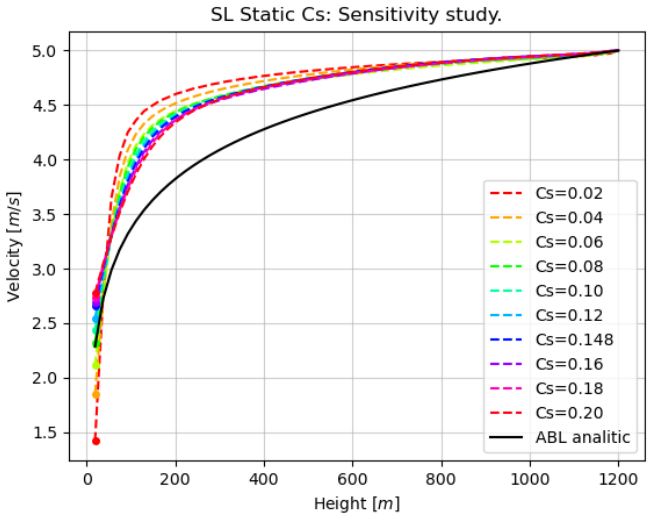


Figure 7.5: SL static model: Statistical velocity profiles for varying C_S.

From figure 7.5, three observations are straightforward:

- All profiles do reach the reference velocity at the reference height.
- Each profile starts at a different velocity on the first node's height (y_{wall}).
- All computed profiles drastically differ from the analytical ABL profile.

The first observation is positive and expected. The second is linked to the C_S value and requires more explanation. The third remark is critical because it can not directly be linked to the implementation.

Reverting to the second observation, the C_S coefficient defines the artificial turbulent viscosity generated by the turbulence model. The artificial turbulent viscosity is added to the kinematic viscosity and, together, they influence the whole domain and, specifically, the shear stress on the wall. By extension, the shear stress applied to the wall (in our case, the first layer above the wall) is a function of C_S. Since the shear stress is directly related to the applied body force (eq. (6.7)), this explains the correspondence between the velocity variation on the first node and the different C_S values.

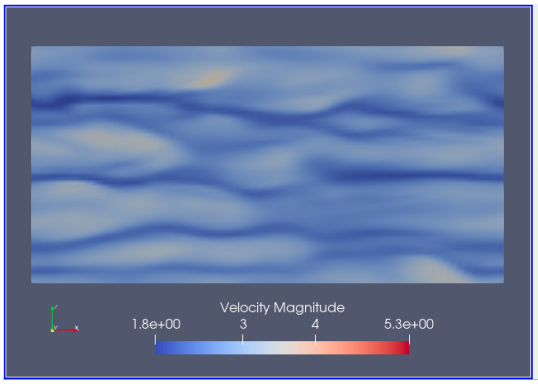
As a matter of verification, we shall demonstrate the shear stress equivalence between the one originating from the simulated body force and the one from the simulated wall velocity.

Thus, starting from the case $C_S = 0.148$, at a specific time step (here, $(6e5 + 264e3)$ iterations[2]):

- On the one hand, we shall consider the computed velocity on the first node's layer, averaged on the whole layer (figure 7.6), to deduce the theoretical shear stress τ_w^{th}.
- On the other hand, we will compute another shear stress τ_w^{sim} from the body force acting on the domain for the same timestep.

Both shear stresses should be equal.

Figure 7.6 displays a slice of the domain, representing the velocity contour at the first nodes' layer only (on the left side), together with the body force and the velocity components averaged on the displayed layer.



	Maximum	Mean	Minimum	Variable
0	1.71761e-05	1.71761e-05	1.71761e-05	Force_0
1	0	0	0	Force_1
2	4.60323e-06	4.60323e-06	4.60323e-06	Force_2
3	3.63169	2.64231	1.76941	Velocity_0
4	0.00113934	-6.07416e-07	-0.00126641	Velocity_1
5	0.387115	-0.00872747	-0.491363	Velocity_2

Figure 7.6: SL static model: Body Force vs. wall velocity. (Magnified version in appendix B.1)

[2]To reduce the convergence time, the new simulation are started from a previously converged (and developed) simulation

Starting from the domain parameters ($\kappa = 0.401$, $h = 1200m$, $y_{wall} = (1200/64)m = 18.75m$, $y_0 = 0.01m$), together with the simulated wall velocity: $u_{wall} = 2.64231m/s$, they produce the theoretical shear stress, from § 6.4.1 and especially eq. (6.1):

$$\tau_w^{th} = \left(\frac{\kappa}{\log(y_{wall}/y_0)}\right)^2 u_{wall}^2 = 0.0197m^2/s^2$$

On the other side, from the simulated body force: $f = 1.71761e{-}5m/s^2$, we find the simulated shear stress via:

$$\tau_w^{sim} = f \times h = 0.0206m^2/s^2 \tag{7.1}$$

Although the two values τ_w^{th} and τ_w^{sim} are not equal, they are incredibly close, which is reassuring. The difference between these two values is nevertheless not adequate and could have been explained by the previously implemented body force limiter but unfortunately, deactivating it in a converged simulation did not suffice to reach equality. Another attempt to reach the balance will be made in § 7.5. For the time being, the solution is sufficient to confirm the second observation made at the beginning of this subsection.

The last observation, the disagreement between the simulated profiles and the analytical ABL profile, still needs to be explored. A first step will be to further develop the SL turbulence model, namely, by considering its evolution: the dynamic Smagorinsky - Lilly turbulence model.

7.3.3 Dynamic version (or Cs dynamic)

Analogously to § 7.3.1, first we will describe the dynamic C_S implementation (given in apppendix A.5). It will allow for a proper identification of the theory presented in § 5.2.1.

Implementation

The dynamic SL was developed as an evolution of the static version. Indeed, the only difference between the two versions resides in the determination of the C_S coefficient. The other steps remain equivalent. As a consequence, the implementation of the dynamic version is also integrated into the static version. As such, all shared lines will not be reviewed.

Below one can find a brief focus on the dynamic version:

- (#7-8) Explained in § 7.3.1.
- (#10-12) The dynamic C_S value will be computed for each element. This value will nevertheless be depending on the neighboring elements. As a consequence, a list of all neighboring elements needs to be obtained. The function *get_neighbElts()* is non trivial and was also developed in this scope. However, it will not be described for conciseness.

- (#18-29) Explained in § 7.3.1.
- (#31-94) As a reminder, the main objective is to generate a dynamic C_S value and to insert it in the regular static SL model. This part contains all the necessary steps to generate the dynamic C_S:
  - (#32-45) For the dynamic version, a second filter, the test filter, needs to be created. All the necessary components for the expressions (5.61) and (5.63) first need to be initialized.
  - (#47-66) This is a loop on each neighboring element. For each of them, a set of elementary computations (e.g. velocity gradient, strain rate tensor, norm, etc.) needs to be created and summed over all these neighboring elements. Later on, they will be used to compute the C_S for the dedicated (center) element.
  - (#68-74) The focus is back on the main center element. The elementary components are estimated.
  - (#76-82) All the computed and summed-up components from the neighboring elements (#47-66) are normalized by the total test filter volume. Also, the strain rate norm for the grid filter is calculated.
  - (#85) The square of the characteristic test filtered width Δ_{tF} is computed.
  - (#86-88) These are the core expressions for the dynamic version, where the matrix M_{ij}, L_{ij}, and finally the square of C_S, named $\tilde{C}_S$ not to confuse it (cf. under eq. (5.56)), is computed.
  - (#90-93) These are two limiters to constrain the value of the real C_S coefficient between 0 and 0.23 (Germano et al., 1991). In addition, the $\tilde{C}_S$ value is square rooted to reduce it to the real C_S value.
- (#95-125) Explained in § 7.3.1.

Without surprise, the implementation follows all steps presented in theory.

Results

Turning now to the results produced by this new implementation, three aspects will be analyzed:

- Does the C_S coefficient truly vary inside the domain?
- How does the resulting viscosity react to this?
- What velocity profile is resulting from this new implementation?

Once more, we will start from the same domain parameter and activate the dynamic SL implementation through setting line #2 to *True* in listing 5. Line #3 is no more active.

To answer the first question, a way to proceed is to look at the C_S contour and plot the values on a vertical line in the center of the domain. Figure 7.7 depicts these information.

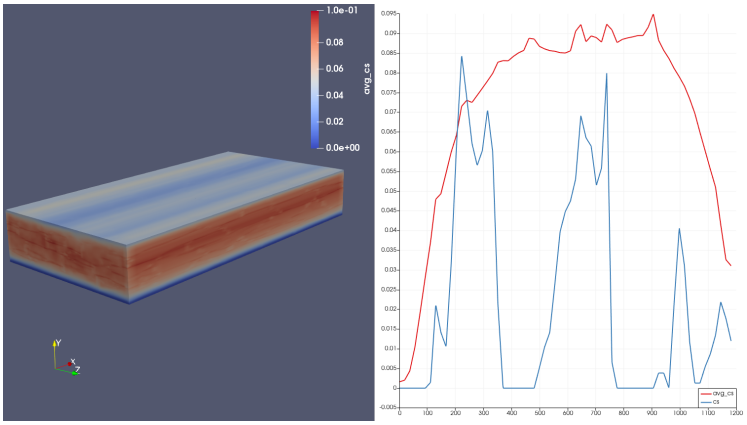


Figure 7.7: SL dynamic model: Cs values on the domain. (Magnified version in appendix B.2)

On the contour (left side), the average C_S value is clearly varying spatially. It is not an accident that the visualization nearly mimics the velocity contour that could be expected from a typical channel flow case. Indeed, the test case, as is, is surrounded by a wall at the bottom and a constant velocity at the top. In the center, the viscosity produced by the SL model has more liberties to amplify.

On the right side, the average C_S value (red-colored) for a vertical profile located in the domain center corroborates with the contour, and the instantaneous C_S value (in blue) exhibits the inhomogeneous and drastic activation on a single vertical line.

The answer to the first question is therefore positive. The C_S value is effectively not constant on the whole domain.

For the second question, similarly, the viscosity can be displayed on the domain. Figure 7.8 presents it.

From figure 7.8, it is not possible to derive the independent influence of the artificial viscosity (Indeed, the implementation directly sums the artificial viscosity to the kinematic viscosity). Nevertheless, it is made clear that the total effective viscosity is not homogeneous in the domain. The pattern is not apparent in any case.

Finally, the third question will be answered by analyzing the dynamic SL result compared to the existing static SL and the analytical velocity profile.

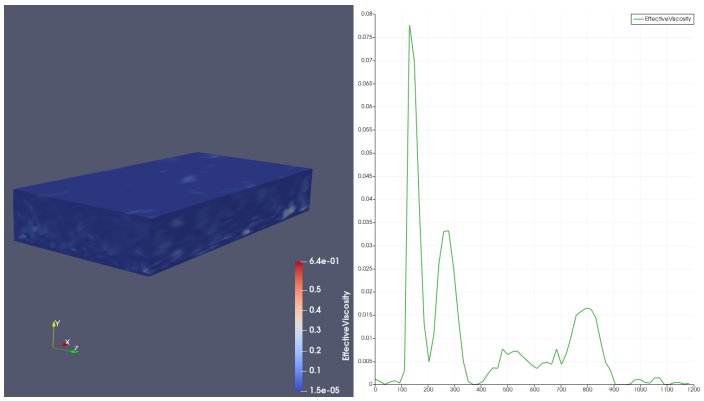


Figure 7.8: SL dynamic model: Viscosity on the domain. (Magnified version in appendix B.3)

Figure 7.9 displays the three statistical profiles for statistically converged simulation. Moreover, the velocity contour for the dynamic case is integrated into the plot.

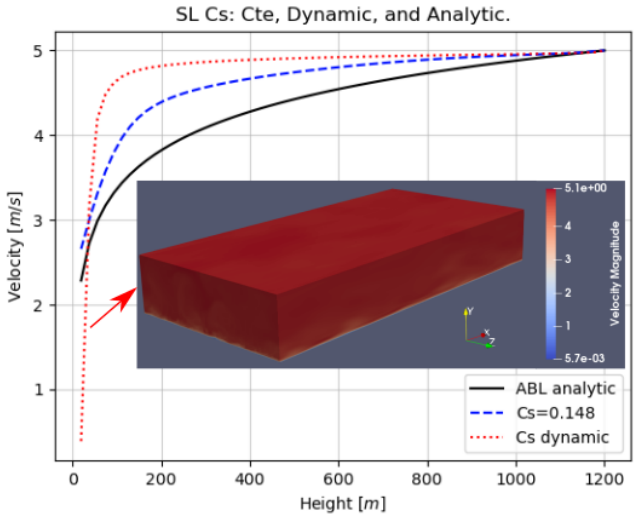


Figure 7.9: SL dynamic model: Statistical velocity profile.

Regrettably, the velocity profile provided by the dynamic version is steering to the other extreme, even further than the most minor C_S case from the static version. By looking at the velocity contour, the whole domain is effectively filled with the reference velocity, except for an extremely narrow

region near the wall, where a thin turbulent layer can be observed.

As a matter of fact, this dynamic version will not help us further. In contrast, a few parameters could still have a beneficial effect. These will be the subjects of the next section, dedicated to sensitivity studies.

7.4 Other sensitivities

From the previous section, it was observed that the constant SL turbulence model does reduce elegantly the spurious numerical oscillations encountered with the ABL wall model FEM implementation. The remaining issue is the great divergence from the expected analytical ABL profile.

In this section, we would like to study how two chosen parameters could affect the profile. Accordingly, a sensitivity study will be performed for each of them. The chosen parameters are:

- the grid resolution,
- the reference velocity,

However, before initiating these studies, because there is no specific section related to performance on our ABL case, and because, except in chapter 3, the significant computation times were never emphasized, we will start the section with a preliminary study on the computation efficiency. Note that this study is also relevant for the grid resolution sensitivity since the latter will make intensive use of this preliminary study.

7.4.1 Computation scalability

Introduction

Following the geometry described in § 6.2 and the mesh proposed in § 6.3, a $64 \times 64 \times 64$ grid would have 262144 elements, what very reasonable is, nowadays. At least if no inner loop is required during the computation of each time step, as, for example, for the computation of the dynamic C_S version seen in the previous section. In that case, the time per iteration can significantly increase.

Suppose now that the grid resolution is increased by 2 in each direction, the resulting grid will reach just above the two million elements, representing today an acceptable but yet large grid to compute. Once more, any inner loop that would derive from an interaction between the computation of the velocity quantities iterating with the pressure matrix (i.e. coupled system) would substantially increase computation time.

To improve our simulations' time, we do have two workhorses:

- find an optimal linear solver, and
- make use of multi-processing.

These items will be analyzed separately and together.

Apart from these two available optimizations, although still related to computational performance, we recently and temporarily received the opportunity to compare the performance of our computation cluster with a new cluster installed in another Belgian university. The last subsection will present the performances for our case, both as a benchmark and to understand where optimization in our computation facility can be envisioned.

MUMPS vs. MueLu solvers

Fundamentally, solving the governing equations (in our case, the Navier-Stokes equations) numerically on a considered domain can be reduced to:

- discretize the domain in a certain amount of elements,
- discretize the PDE's representing these governing equations and transform them to a linear sub-system,
- apply this discretization to each element to constitute the linear system to solve, and eventually,
- solve this linear system.

Although each item of this outline is a field in itself, only the two last tasks will be discussed.

To perform them, *Coolfluid 3* integrates one package called *Trilinos* (Sala et al., 2010) that is specialized in tools and solvers for large matricial systems. It can interface with two libraries, specialized in this respect:

- The first library, *MUMPS*, is a MUltifrontal Massively Parallel Solver. It is an external library, written in Fortran90 (a low-level language), is opensource, and is based on a direct Gaussian elimination approach. The latter is named the multifrontal method (Amestoy, Duff, and L'Excellent, 2000; Amestoy et al., 2019).

  This method solves large linear systems with the $Ax = b$ shape, where the system matrix, A, is a square sparse matrix with no limitation on the "symmetricity"[3]. For this solver, in the semi-implicit method, the factorization process for the pressure system needs to be executed for each iteration, inducing a computational cost.

  MUMPS is a direct solver, and, usually, it is far too expensive for CFD, but it is admissible for the pressure system because the matrix does not change with time.

  Lastly, although not directly integrated into the solver, *MUMPS* can easily be parallelized by using it, through the *MPI* (Message Passing Interface) protocol (Sur, Koop, and Panda, 2006).

[3] x is the unknown vector and b is the RHS vector

- The second library, *MueLu* is a multigrid preconditioner library (smoothing the shape of the system matrix A to fasten the following processes) that is directly integrated to *Trilinos* and can fully be adjusted to the considered application. It uses a Smooth Aggregation (SA) algorithm, which is written in `C++` (also a low-level language), is opensource (Prokopenko et al., 2014; Wiesner et al., n.d.).

  In contrast with the previous library, *MueLu* is not used as a solver but as a preconditioner for the conjugate gradient method to solve the pressure system. It enables it to avoid factorization at each iteration. However, for accuracy, during the sensitivity analysis, both the mean and minimum computation times will be provided to compare the solver process primarily.

  Note that the *MueLu* library allows to fully adjust the system matrix, which means several options need to be selected. The curious reader will find the chosen options for our case in the third listing (i.e. "MueLu specs XML file") in appendix A.6. Their definitions are available in Wiesner et al., n.d. Nevertheless, they will not be further explained for brevity, although background tests were performed.

The activation of each library is made inside the Python input file and is available in appendix A.6.

To emphasize this sensitivity study in the case of our ABL test case (with 64x64x64 elements), it takes around $3e6$ timesteps, timesteps of $5e{-}4$ seconds, to reach statistical convergence, and an equivalent amount is necessary to capture the statistics of the flow. If the CFL remains reasonable within this configuration, it takes ten days to obtain usable results on 80 cores performing each part of the computation on a slice of the domain. If the CFL increased dangerously, the timestep would need to be reduced, increasing the number of iterations proportionally and, therefore, the required time needed to obtain the usable results. If the computational resources are not available, the situation is even worse.

Clearly, reducing the processing time per iteration will have a tremendously beneficial effect on the resolution time. Assembling the system matrix, optimize its shape for faster resolution, and select a solver dedicated to the type of matrix that corresponds to our specific PDE are all factors that are non-negligible when considering a large linear system.

Therefore, four configurations for our case are proposed:

- our classic case with 64x64x64 elements, solved with MUMPS, on a node containing 80 cores.
- our classic case with 64x64x64 elements, solved with MueLu, on a node containing 80 cores.
- a refined case with 128x128x128 elements, solved with MUMPS, on two nodes containing each 80 cores.

- a refined case with 128x128x128 elements, solved with MueLu, on two nodes containing each 80 cores.

For each case, only ten timesteps were performed to reduce the weight of the setup while still limiting the computational time.

Remark 7.3. *Three required notions: Cluster, Nodes, and Cores.*

The available computational facility at the Royal Military Academy is constituted of a server governing a network of computers where the whole group is called a cluster, where each computer is named a node. These nodes contain a certain number of processors that each contain a certain amount of cores (e.g. in our case, it can be 20 cores per processor. See in the clusters' study § 7.4.1). To simplify, the processors will not be considered after that, only the nodes and the cores.

The communication between cores of the same node is optimal (thanks to the vicinity but also the intrinsic construction of a node).

The communication between nodes is less optimal and can present greater latencies than between cores. These latencies can negatively affect computational time. Thus, for a test case that requires more than one node to work, a latency cost is to consider.

Coming back to our four test cases, the two traditional 64x64x64 cases would have sufficed to observe the behavior of the two solvers. However, would these behaviors be confirmed or vanished when the linear system is so large that parallelization between nodes is mandatory? Would the solvers take advantage of the parallelization or, conversely, be heavily affected? This is the reason for the two additional test cases, with the larger and refined 128x128x128 grid.

Table 7.1 presents the mean timing[4] for these different configurations. The total timing is not given because the focus is set on the time differences between the two solver types. As a consequence, the loop timing for the Navier-Stokes solving process is considered as the total timing. This loop timing contains several steps like the initial condition setup, the resolution of considered governing equations, the boundary conditions setup, and other steps. In fact, the most relevant step in our case is the resolution of the governing equations, that is, the Navier-Stokes in its semi-implicit version.

Therefore, one will find a summary of the processing time for each step inside this semi-implicit resolution. Notice that each of these steps is performed ten times.

Eventually, the last section of table 7.1 provides details concerning one of these sections, namely the inner loop section. Notice that each of these steps is performed twenty times.

[4]Notice that the minimum timings are provided in appendix C.1. Although the mean timings are biased by the factorization that occurs in MUMPS before each iteration, the conclusions provided by both the mean and the minimum timings are very similar. Looking at the mean values facilitates the additions.

		mumps64	muelu64	mumps128	muelu128
Grid		64x64x64	64x64x64	128x128x128	128x128x128
Solver		MUMPS	MueLu	MUMPS	MueLu
NS semi-implicit		3.683	1.995	75.581	10.548
10x	LinearizeU	1.699e-2	1.849e-2	1.755e-1	9.751e-2
10x	PressureLSS	3.757e-5	3.751e-5	4.741e-5	4.628e-5
10x	VelocityLSS	3.376e-5	3.284e-5	3.448e-5	3.321e-5
10x	SetSolution	4.608e-3	5.098e-3	6.683e-3	1.422e-2
10x	InnerLoop	3.612	1.930	75.058	10.228
10x	Update	2.046e-2	1.690e-2	2.801e-1	1.703e-1
10x	CFL	1.035e-2	3.250e-3	2.069e-2	1.049e-2
In InnerLoop					
20x	URHSAss	4.369e-2	4.579e-2	1.482e-1	1.542e-1
20x	PRHSAss	3.916e-2	3.919e-2	1.218e-1	1.157e-1
20x	ApplyAup	2.476e-2	2.329e-2	7.507e-2	6.938e-2
20x	SolveUSyst	2.205e-1	2.751e-1	8.014e-1	6.855e-1
20x	SolvePSyst	1.351	4.566e-1	35.876	3.645

Table 7.1: MUMPS vs. MueLu solver timings [s] (mean values).

When looking at the first two cases (i.e. two first columns) from table 7.1, one will find out the total (loop) timing are not equal. The MueLu case processes the information in nearly half the time. According to the data, the difference resides in the Navier-Stokes resolution, and precisely, one timing is diverging between the two cases: the InnerLoop step (dashed-underlined). Again, the timing difference reflects the halving. By diving even further into the InnerLoop details, one will find out the InnerLoop operation contains all the processes related to RHS and system matrix assembly, as well as the resolution, for both the velocity and the pressure system. It is precisely in the resolution of the pressure system that the disparity is observable.

When looking at the two following columns, the 128x128x128 refined cases, the observations are identical, with one exception: the contrast between the MUMPS and the MueLu resolution is even greater (a reduction of more than 15% for the MueLu case). These two right columns demonstrate that the parallelization between two nodes benefits the MueLu solver considerably.

Practically, with no modification of the cluster's setup, the MueLu solver will be chosen for our case.

Multi-processing

After having optimized the solver's choice, the second ingredient that can help to reduce the total simulation time is the parallelization between several nodes to reduce the load and accelerate the processing time for each of them.

For this purpose, our ABL case will be even further refined to reach 256x256x256 (i.e. nearly 17 million) elements. This test case will then be simulated in parallel, using 1 to 4 nodes of 80 cores each.

Remark 7.4. *Cores execute simple instructions, also called threads. Each processor will distribute threads to its cores (cf. Remark 7.3). This low-level operation is called multi-threading.*

Accordingly, the processor uses the results of these small instructions to compose more complex expressions that can then be a part of the final result. For this purpose, the processors can work together with other processors through what is called multi-processing.

Thanks to the MPI protocol, both the multi-threading and multi-processing operations are handled as processes, hence the title of this subsection.

Reverting to the scaling study, the main study will consider only the computation time. Aside, a secondary study will observe the impact of saving intermediate data (named *snap*, for snapshot) on the total simulation time. One could imagine that reading/writing a large amount of data to the disk between computation steps could have an impact on the simulation time. The question will then be on what scale.

Table 7.2 presents 3 simulations with respectively 10, 20, and 100 timesteps (called *Iters* in the table and in Figure 7.10) and only 1 snapshot. Two additional cases are given with respectively 2 and 10 snapshots, for the secondary study.

For each of these cases, the first subtable (in the center) exposes the total simulation time for each case in minutes:seconds. The second subtable (on the right) displays the same information, in percentage, when considering the one node case as the reference (i.e. 100%).

		Timing [mm:ss]				Ratio [%]			
	Nodes	1	2	3	4	1	2	3	4
	Cores	80	160	240	320	80	160	240	320
Iters	Snap								
10	1	17:10	17:16	21:08	21:19	100.0	100.5	123.1	124.2
20	1	22:29	19:43	25:10	24:53	100.0	87.7	112.0	110.7
	2	22:27	25:09	20:09	24:24	100.0	112.0	89.8	108.7
100	1	59:21	42:15	43:52	42:13	100.0	71.2	73.9	71.1
	10	61:31	58:36	36:50	43:51	100.0	95.3	59.9	71.3

Table 7.2: Multi-processing (256x256x256 grid): total simulation times.

For the main study, the observations can be divided into three:

- Vertical analysis: when looking at a single node number (single column), the increase in total processing time is not following a linear path in function of the number of iterations. Actually, it is even decreasing, meaning that the fixed time-cost reduces with an increasing number of iterations. Indeed, at the beginning of the simulation, setting up the domain only happens once. No surprise here.

- Horizontal analysis: When looking at the ratio subtable (right subtable), one will find out there are two trends: on the one hand, the 10 and 20 iterations' cases that tend to increase the total simulation time with increasing parallelization; on the other hand, the 100 iterations' case that follows a more suitable path, reducing the total simulation time when doubling the parallelization nodes. This observation is essential. Predictably, the cost due to the parallelization (and its associated communication latency) needs to be absorbed by the number of iterations to be worthwhile.

- Horizontal and vertical analysis: The optimal simulation time (under dashed) for this case is reached when shifting from 1 to 2 nodes and with 100 iterations.

Concerning the secondary study (i.e. the influence of the snapshots), it is perceptible that some time is necessary for them to occur and, when looking at the most extensive case, 100 iterations and 10 snapshots, the performance penalty is significant (i.e. from 71.2% to 95.3% of the one node timing). In such a case, the three nodes configurations will be more advantageous. Consequently, the communication time needed to write the snapshots should not be neglected when considering the computational setup.

The current analysis is based on the total simulation time. Although interesting if one suppose infinite computational resources, it is also essential to evaluate the efficiency of processing time per core.

Figure 7.10 provides this information. The total times were divided by the number of cores, for all one snap configuration and the ten snaps case, in function of the number of nodes used.

The first observation is that all cases follow power laws (dashed lines) that tend to converge to the same minimal timing when an infinite number of nodes is used. Note that this minimal timing is non-zero. The second observation shows that the most significant reduction occurs when passing from one to two nodes, and specifically for the high number of iterations.

Eventually, our computational resources are limited, bringing the optimal (read simulation time vs. node use) choice to two nodes (or 160 cores) for the ultra-refined case (or even three nodes if multiple snapshots are required).

With these two adjustments, we have a good setup to start the sensitivity study related to the grid resolution (i.e. mesh). However, prior to this new study, the last performance study related to computational resources will be presented.

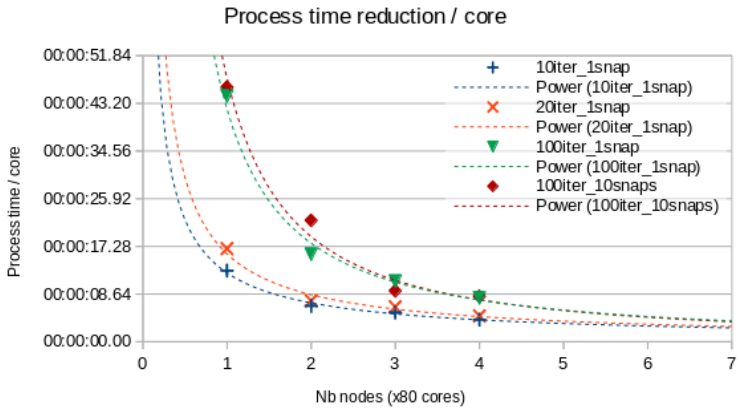


Figure 7.10: Multi-Processing optimum.

Cluster performance

As stated in the introduction, because we received the opportunity to benchmark our cluster to a new cluster facility based in another Belgian university, we used the ABL case to simulate a few time steps. From these simulations, we reiterated the two previous studies in a condensed way and eventually compared the results for the largest grid (i.e. 256x256x256) to simulations ran on our cluster. Following the indicator used in the solver's choice study, table 7.3 displays the Navier-Stokes semi-implicit minimum process time for comparison.

Before analyzing the data's, the characteristics of the two clusters will briefly be listed to have a better insight:

- The new cluster named VSC here is provided by the *Vlaams Supercomputing Center* (VSC, 2019). It is composed of:
  - 408 Skylake[5] nodes (incl. 2 CPUs with 14 cores each), 192GB RAM (with latency checker).
  - 436 Broadwell nodes (incl. 2 CPUs with 14 cores each), 128GB RAM (with latency checker).
  - 144 Broadwell nodes (incl. 2 CPUs with 14 cores each), 256GB RAM.

  The nodes are connected using an Infiniband EDR network (25.8Gbit).
- The RMA cluster is composed of:
  - 12 Skylake nodes (incl. 4 CPUs with 20 cores each), 384GB RAM.

[5]The node's name is associated with the CPU micro-architecture, hence, production date and performance. For instance, Haswell, Broadwell, Skylake were respectively built in 2013, 2014, and 2015.

- 5 Broadwell nodes (incl. 2 CPUs with 20 cores each), 128GB RAM.
- 10 Haswell nodes (incl. 2 CPUs with 16 cores each), 128GB RAM.

The nodes are connected using a 10Gbit network (older technology).

From these characteristics, one can notice that the CPU architectures are equivalent. In contrast, the communication between nodes is substantially different, and the clusters' sizes are not comparable (but the latter has no direct impact on this study). Notice that all simulations were performed on the Skylake architectures.

#	Cluster	Cores (-Nodes)	Grid MUMPS	64 [s]	128 [s]	256 [s]
1	VSC	28	x	1.814	x	x
2		84	x	0.919	x	x
3		168		x	3.847	x
4		336		x	2.313	x
5		336	x	x	6.624	x
6		224		x	x	11.434
7		896		x	x	6.081
8		1792		x	x	4.203
9	RMA	80-1		x	x	25.070
10		160-2		x	x	14.952
11		240-3		x	x	10.458
12		320-4		x	x	8.687
13		400-5		x	x	7.532
14		480-6		x	x	9.994

Table 7.3: VSC and RMA clusters's benchmark: Solver processing time.

Moving on to the analysis of table 7.3:

- The first two rows correspond to a regular 64x64x64 grid case solved using the *MUMPS* solver on the VSC cluster. These two lines imply there is no linear scalability. Indeed, the reduction in processing time is not proportional to the number of cores. However, the gain is fair.
- Lines 3-5 correspond to a slightly larger grid (128x128x128). Analogously, the processing time reduction in function of the number of nodes is observable on lines 3-4, where the preconditioner *MueLu* is applied. Lines 4-5 are the same case except for the solver type, confirming the advantage provided by the *MueLu* solution.
- Lines 6-8 presents the largest grid (256x256x256), using *MueLu*, where increasing the number of nodes still induces an improvement in processing time.

- Lines 9-14 represents the same 256x256x256 grid, on the RMA cluster, for 80 to 560 cores. Firstly, notice that the 240 cores RMA case (line 11) has slightly better timing than the 224 cores VSC case (line 6). However, it also has more cores to perform. Secondly, an optimum is reached at 400 cores (on 5 nodes), and this minimal processing time is clearly greater than the best timings obtained with the VSC cluster (line 8). One could argue that the number of cores is not comparable, and it is true. However, line 14 showed that an increase in cores on the RMA cluster would not lead to further time reduction.

The last point is interesting and can be associated with remark 7.3, where the communication speed between nodes can be responsible for computation latencies. Effectively, when looking at the cluster's characteristics, the communication protocol (e.g. infiniband vs. 10Gbit ethernet) are significantly different, providing a nice hint on the possible improvement to bring to the RMA cluster.

This concludes the subsection on computation sensitivity study. With this deeper insight on the computation hardware and with the adjustments brought to the solver's type and to the multi-processing setup, the next subsections will initiate the sensitivity studies directly connected to the test case's result.

7.4.2 Mesh sensitivity

As expressed in the introduction of this section 7.4, the idea is to play on the parameters to seek a manner to bring the simulated velocity profile to the analytical curve. In this respect, our case will be simulated with 3 degrees of refinement:

- 64x64x64 elements
- 64x128x64 elements
- 64x256x64 elements

Notice that the refinement takes place only in the vertical y-direction to avoid an explosion of elements in less relevant directions and, consequently, an explosion of computational resources. As a matter of completeness, the 128x128x128 case was started for comparison with the 64x128x64 case, but the convergence time was not affordable (i.e. extrapolated, four months of computation time is required).

The resulting velocity profiles for the three grid resolutions are proposed, together with the analytical profile, in figure 7.11.

Happily, the three profiles do end at the same reference velocity and follow a similar profile. Understandably, they all start at different positions and velocities. This is normal since the position of their first node varies with the varying resolution. Accordingly, their respective curve tends to get closer to

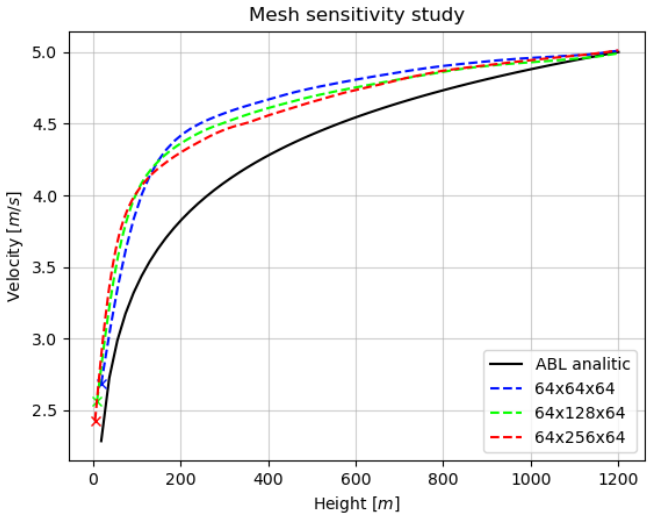


Figure 7.11: Grid resolution.

the analytical profile for increasing resolution, which was expected since it approaches a full resolution.

Unfortunately, the relative progress is small compared to the increase in grid resolution. Moreover, the concavity remains slightly different. However, this parameter does influence the simulated profile positively.

7.4.3 Velocity sensitivity

Another parameter that will be studied is the reference velocity. The latter obviously has an impact on the Reynolds number and therefore modifies the inherent type of flow that is to be expected, although in our case, the flow was already highly turbulent.

The consequence of not having the same Reynolds number, if we keep the same case and only modify the reference velocity, is that the simulations results can not be compared non-dimensionally. It would correspond to comparing apples with pears.

Nevertheless, since our objective is to analyze the influence of the reference velocity on the resulting velocity profile, an option is to normalize these profiles by their respective reference velocity. Doing this will allow having normalized profiles that can easily be set side by side.

Practically, we will compare the analytical values to three velocities:

- $5m/s$
- $10m/s$

- $20m/s$

Figure 7.12 presents these normalized curves. One can observe that none

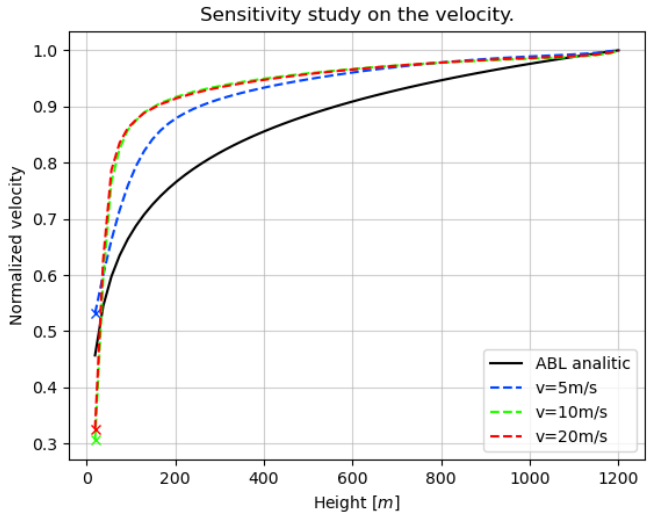


Figure 7.12: Sensitivity study on reference velocity.

of the configurations generates a velocity profile that approaches the analytical profile.

Notwithstanding, the $10m/s$ and the $20m/s$ match each other perfectly, implying that, at a very high Reynolds number, no change in profile is expected. What could we say concerning a smaller reference velocity?

When looking at our reference velocity profile (blue dashed line), the latter matches neither the two higher velocities nor the analytical. It lays in between. Could this mean that the lower the Reynolds number, the better the matching will turn out? By analogy, when considering a similar flow case, the channel flow, DNS simulations could be performed (in 2015) until a viscous Reynolds number (i.e. with respect to u_τ) of 5200, but not higher.

This brings a nice transition to the next section, which attempts to answer this question by reverting to a case with a lower viscous Reynolds number.

7.5 Analogy to the channel flow

In this penultimate section, following the above observations, the idea is to reduce our ABL test case to a more universal case. This reduced case with less severe constraints can also be used for comparison in future works.

The chosen reference case is a DNS simulation performed by Lee and Moser, 2015, for a channel flow, at viscous Reynolds numbers until $Re_\tau =$

5200. As a reminder, one defines the viscous Reynolds number as a non-dimensional number dividing the channel half-height h by the viscous length scale δ_v. We have chosen their $Re_\tau = 1000$ case as a reference.

Indubitably, our ABL case is not a channel flow, but to a certain extent, when considering half the height of the channel flow, one could consider it to be similar to our case. However, this is not the only modification that has to be applied. Our model reaches $Re_\tau = 5e7$, that is, significantly higher than the proposed reference case. Accordingly, we will adapt our case. The modified parameters are proposed in eqs. (7.2).

$$\begin{aligned} Re_\tau = 1000, \quad h = 1m, \quad u_{ref} = 22.593m/s, \\ \rho_f = 1kg/m^3, \quad \mu = 1/Re_\tau, \quad \nu = \mu/\rho_f, \\ z_{wall} = 0m, \quad z_0 = 1.015e{-}4m, \quad u_\tau = 1m/s, \quad f = u_\tau^2/h \end{aligned} \tag{7.2}$$

where all domain parameters are adjusted to reach the imposed Re_τ.

For this basic test case, three variations are proposed:

- The elementary no-slip type: the velocity on the first node is set to zero, assuming the first node position is the wall.
- Ustar version: the velocity on the first node is set equal to the manually computed friction velocity. The latter will be explained in its respective subsection.
- ABL wall model variant: the velocity on the first node follows the wall model implementation.

The resulting velocity profile will be expressed non-dimensionally (cf. previous figure 4.14). We have envisaged different grid resolutions (in the vertical y-direction) for each of these cases, and their results will be compared to the DNS profile.

Each simulation will be performed in two steps:

- a first simulation with the body force limiter activated,
- a second step with the body force limiter deactivated.

The idea behind this is to use the body force limiter to rapidly reach the converged state and then let the equilibrium between the body force and the shear stresses govern the flow.

Lastly, due to the different grid resolutions and lower boundary conditions, the convergence could not be reached at the same pace for each case. Concretely, some simulations needed their timesteps to be adjusted, increasing the computation time dramatically. Therefore, not all simulations have reached convergence at the same rate. Nevertheless, the trend given by the curves can still be used for interpretation. To ease the comprehension, table 7.4 presents the total simulation time for every simulation.

resolution	No Slip	$u_w = u^*$	Wall model
64x64x64	167	220	ϕ
64x128x64	39.6	18.82	75.66
64x256x64	35.4	45.8	22
64x512x64	29	ϕ	ϕ

Table 7.4: Simulation time [s] per grid resolution and wall type.

Notice the wall model case for the 64 grid (i.e. short name for 64x64x64) directly converged to a homogeneous reference velocity in the whole domain, making the results non-usable. For the 512 grid, only the no-slip condition was intended because of the computational cost. The idea was to check the trend.

From table 7.4, one will see that the small 64 grids obtain considerable simulation time. This is due to a timestep of $5e{-}4s$ that could be applied continuously. In contrast, the other grid resolutions suffered very high instabilities at the beginning of their respective simulations, generating peak velocities near the wall as high as $340m/s$ (for the 128 grids). Consequently, they needed a considerable timestep reduction to meet the CFL condition. However, as a matter of example, with a reference velocity of $22.593m/s$ inside a domain with a length of $6.283m$, and after $18.82s$, a particle traveled the domain more than 67 times. In comparison, according to Goit, 2015, at least 8 crossings are needed to develop the flow.

Before analyzing the resulting curve, one remark concerning the 128 grids: although all the other parameters were identical (except the grid resolution thus), they all started with a substantial velocity near the wall. One will remember § 7.1 concerning the ordinary implementation of the randomizer used to stimulate the turbulences. Because of the non-adjusting behavior of the randomizer, the latter could be the reason for the 128 grids to behave radically differently, at least in the first timesteps.

7.5.1 No-slip sensitivity

The no-slip condition case was simulated for four grid resolutions (i.e. 64, 128, 256, and 512 elements in the y-direction) and is displayed in figure 7.13.

One will see that all curves do not match the DNS profile, although they follow similar trends. Moreover, the difference between 256 and 512 cases is not relevant, while, in contrast, it is relevant with the two other resolutions. One would expect an increasing resolution near the wall to have a positive influence on the resulting profile. This is the case in the buffer (and partially inertial) region. However, the 128 grid's result seems to lack some convergence far from the wall.

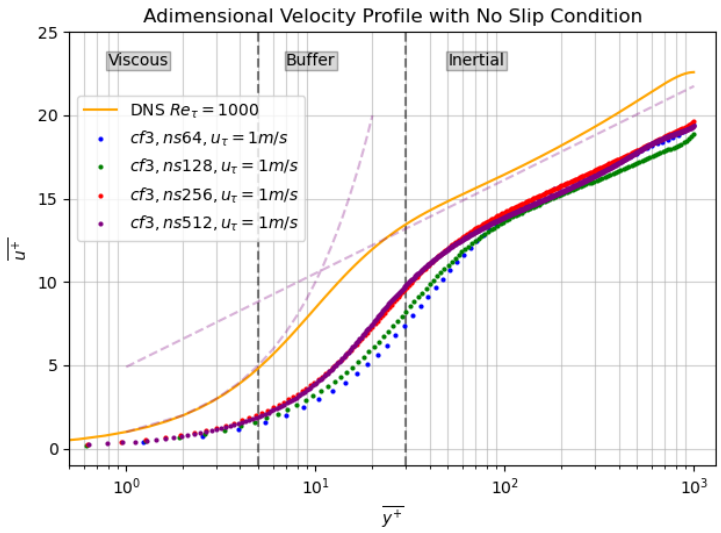


Figure 7.13: Non-dimensional velocity profile with no slip condition.

7.5.2 Ustar sensitivity

Suppose the no-slip condition at the wall is substituted so that the velocity at the wall is set equal to the expected friction velocity u^*.

Figure 7.14 provides the resulting profiles.

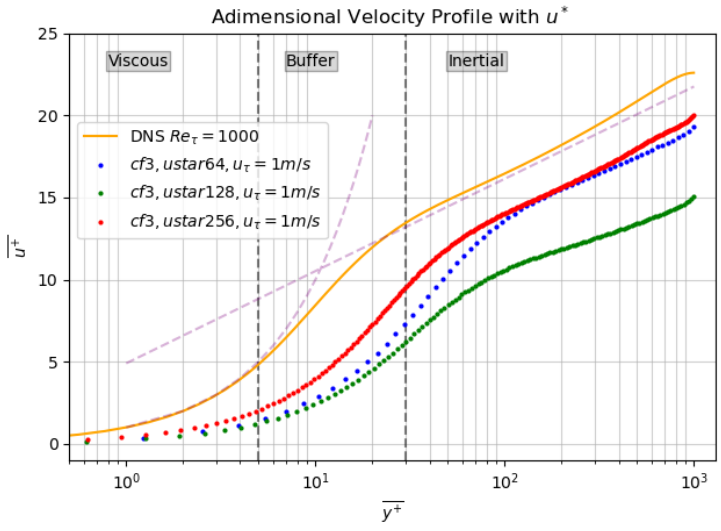


Figure 7.14: Non-dimensional velocity profile with friction velocity at wall.

Once more, all curves do not match the DNS profile, while the trends for the 64 and 256 cases are similar. The 128 grid is clearly not yet converged, corroborating with the high-velocity issue related to the randomizer (stated in the introduction for the three test case variations).

The two other resolutions (64 and 128) follow the same observations as for the no-slip condition: the higher the resolution, the nearer to the DNS line.

7.5.3 Wall model sensitivity

Eventually, applying our wall model implementation results in figure 7.15.

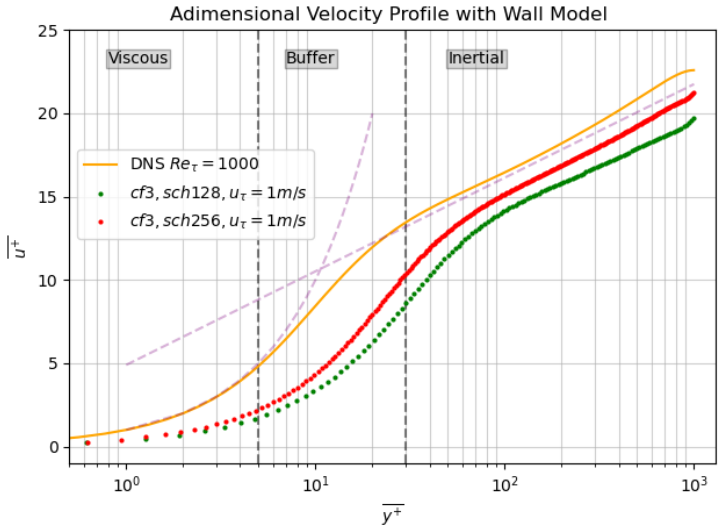


Figure 7.15: Non-dimensional velocity profile with ABL wall model condition.

These are opportunely the best profiles in terms of agreement with the DNS profile:

- positive improvement due to the resolution, and
- close matching with the DNS curve in the inertial region. The latter represents the region of interest for the computation.

To conclude, the first two cases presented a departure from the DNS result, although following a similar slope. The last case, the ABL wall model implementation, presented a reasonable matching with the DNS curve, underlining the benefit of reducing the number of elements by applying a wall model.

Nevertheless, analogously to the conclusion of previous sections of this chapter, the gap still needs further development to reach a perfect matching.

7.6 VMS

During this chapter, the focus was solely on implementing LES turbulence models and their effects on the velocity profile. It was demonstrated that they could significantly help to reduce the spurious numerical oscillations near the wall. Nonetheless, the resulting profiles were not matching with the analytical ones.

In a parallel path, a completely different approach was initiated. The foundation of this approach relies on the evolution of the SUPG stabilization method, named the VMS (§ 5.1). As explained in the modeling part, the VMS was developed precisely to generalize and increase the stabilization effects on a broader range of flows. In this context, it was essential to implement this methodology to determine its effectiveness in our ABL case. The first results will be presented in this section. Notice that all tests are performed in 2D to reduce the computational cost and to facilitate troubleshooting. As explained in the FEM section (§ 4.2), moving from a 2D simulation to a 3D simulation only requires the use of different types of element; the algorithm implementation remains identical.

The first section is focused on results displaying the algorithm's structure, whereas the second section concentrates more on the implementation of the algorithm itself.

Mesh properties

Preliminary to the results, the mesh properties are detailed:

- For the tests related to the structure, a mesh of 16x16 segments was used, with a reference velocity of $1m/s$ in x-direction (horizontal) for the Poisson equation and in the x and y-directions for the Taylor-Green equation.
- For the test related to the algorithm, the number of segments was reduced to 2x2, and the θ-value appearing in the finite element formulation from eq. (4.79) was set to 1 (in other words, the Euler form) to be able to compare the combination of the SUPG assembly matrices, A_e and T_e, to the VMS assembly matrix, A_e.

7.6.1 Structure

The VMS structure described in section 5.1.1 is a predictor / multi-corrector structure. To confirm the correctness of our implementation, it was first tested with classical governing equations.

Poisson equation

The first test, the Poisson equation, defined by $\Delta\varphi = f$, was used because of its simple implementation, its analytical resolution, and no use of periodical boundary conditions.

The correctness of the predictor / multi-corrector structure was verified by comparing this first test case qualitatively to the simulation result, using a standard steady Poisson implementation. No quantitative analyses were performed.

Figure 7.16 proposes a velocity contour of the simulation, where we set the inlet velocity on the left side to $1m/s$.

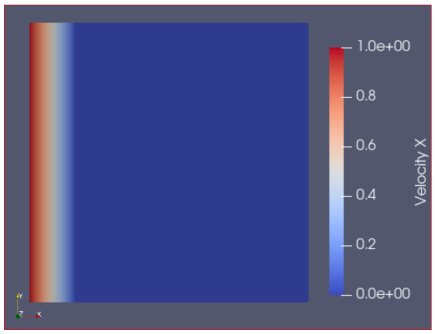


Figure 7.16: Poisson test case

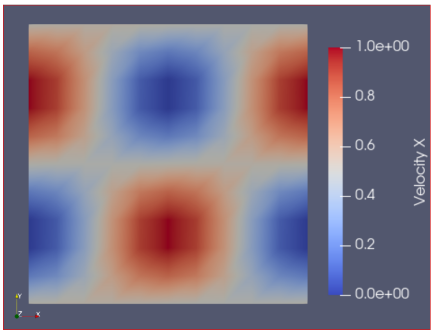


Figure 7.17: Taylor-Green test case

Taylor-Green equation

After having gained some confidence in the structure's implementation, it was tested on a test case with periodical boundary conditions: the Taylor-Green test case.

Once more, we chose this test case because it has an analytical solution and because its standard Navier-Stokes' implementation is available in *Coolfluid 3*.

The objective of this test was to ensure the output generated by the VMS's structure would agree with the one produced with a standard Navier-Stokes structure.

Figure 7.17 presents a velocity contour that reflects the recognizable sinks and sources from the Taylor-Green equation. Moreover, no spurious phenomenon is visible at the boundaries.

7.6.2 Algorithm

Having acquired confidence in the structure, it is time to test the algorithm's implementation. The latter is difficult because no analytical comparison is possible. The only analysis that can be performed is to compare the

assembly matrix A_e between the already existing SUPG implementation and the new VMS algorithm implementation. To do this, both the viscosity ν and the stabilization coefficients τ_x have to be reduced to zero (to reduce the VMS algorithm to an equivalent SUPG's).

To better analyse how each equation described in section 5.1.3 is influencing A_e, they are schemed with blocks in Table 7.5. Notice that each block is actually a 3x3 matrix.

$A_{u_i u_i}$	$A_{u_i u_j}$	$A_{u_i p}$
$A_{u_i u_j}$	$A_{u_i u_i}$	$A_{u_i p}$
$A_{p u_i}$	$A_{p u_i}$	A_{pp}

Table 7.5: A_e structure

The A_e matrix will be printed with three decimals (sufficient to see the differences) to ease the reading.

Results with $\nu = \tau_x = 0$

A simulation was performed, for an identical mesh, with both the SUPG implementation and the new VMS implementation. Table 7.6 displays A_e for the SUPG implementation. This is the reference result.

-0.182	0	0.182	0	0	0	-0.083	0.083	0
-0.182	0	0.182	0	0	0	-0.083	0.083	0
-0.182	0	0.182	0	0	0	-0.083	0.083	0
0	0	0	-0.182	0	0.182	0	-0.083	0.083
0	0	0	-0.182	0	0.182	0	-0.083	0.083
0	0	0	-0.182	0	0.182	0	-0.083	0.083
-0.083	0.083	0	0	-0.083	0.083	0	0	0
-0.083	0.083	0	0	-0.083	0.083	0	0	0
-0.083	0.083	0	0	-0.083	0.083	0	0	0

Table 7.6: A_e with $\nu = 0$ for SUPG

In table 7.6, one can see the matrix is symmetrical, considering the blocks. Another observation is that $A_{u_i u_j}$ and A_{pp} blocks are null value. Eventually, all blocks have a column structure (values varying only column-wise).

The next table 7.7 proposes the result for the VMS implementation (cf. eqs. (5.17 - 5.21)).

When analyzing table 7.7 for the similarities, the observation is that the null valued blocks are identical. For the differences, the first remark is that A_e is column-wise except for $A_{u_i p}$ that is row-wise and has the opposite

-0.083	0	0.083	0	0	0	0.083	0.083	0.083
-0.083	0	0.083	0	0	0	-0.083	-0.083	-0.083
-0.083	0	0.083	0	0	0	0	0	0
0	0	0	-0.083	0	0.083	0	0	0
0	0	0	-0.083	0	0.083	0.083	0.083	0.083
0	0	0	-0.083	0	0.083	-0.083	-0.083	-0.083
-0.083	0.083	0	0	-0.083	0.083	0	0	0
-0.083	0.083	0	0	-0.083	0.083	0	0	0
-0.083	0.083	0	0	-0.083	0.083	0	0	0

Table 7.7: A_e with $u = 0$ for VMS

sign, compared to SUPG values. Second, the $A_{u_i u_i}$ values are not equal to SUPG ones.

After modifying $A_{u_i p}$ by transposing it and taking its opposite value, table 7.7 becomes table 7.8.

-0.083	0	0.083	0	0	0	-0.083	0.083	0
-0.083	0	0.083	0	0	0	-0.083	0.083	0
-0.083	0	0.083	0	0	0	-0.083	0.083	0
0	0	0	-0.083	0	0.083	0	-0.083	0.083
0	0	0	-0.083	0	0.083	0	-0.083	0.083
0	0	0	-0.083	0	0.083	0	-0.083	0.083
-0.083	0.083	0	0	-0.083	0.083	0	0	0
-0.083	0.083	0	0	-0.083	0.083	0	0	0
-0.083	0.083	0	0	-0.083	0.083	0	0	0

Table 7.8: Modified A_e with $u = 0$ for VMS

In the latter, a difference remains when looking at $A_{u_i u_i}$. Unfortunately, the investigation was interrupted at this point due to a lack of details in the literature. Nevertheless, when this issue is solved, the next step will be to reactivate the stabilization coefficients and the viscosity to study their behavior.

To summarize these preliminary results, it was demonstrated that the implementation of the predictor / multi-corrector stages structure behaves correctly. It was also shown that the implementation of the algorithm itself still lacks correctness, although evolving in the right direction. Eventually, when the VMS implementation reflects perfectly the SUPG (for the non-viscous case), it will offer a robust foundation to explore the VMS capacities.

This concludes both this chapter, where the positive effects of LES turbulence models on our ABL case were demonstrated and the preliminary investigations on the new VMS implementation. As introduced in the theoretical part, the VMS approach is still young and lacks validation cases. However, according to the literature, it should turn out to be a major can-

didate to resolve issues like the one we are confronted with.

Part IV

Conclusions, Further Works and Appendices

Chapter 8

Conclusion and further work

8.1 Conclusion

The original objective of this work was to integrate functionalities into the simulation software *Coolfluid 3* in such a manner that the SUPG stabilization could be applied to a FEM, together with a DQMOM model to include a second phase (i.e. particles), on realistic open-air CBRN cases (e.g. dispersion, after an explosion of particles in a city).

Such a utility is of great interest for the Belgian Defense because of its dual aspect: safety for existing sensitive locations and prevention for potential threads (§ 2).

Although the Defense already uses commercial packages that help with these concerns, these solutions are based on a statistical approach that provides fast answers while being limited in complexity. As such, the topographical details, the physical environment, or even the thermal effects are all obstacles that the existing solution can not handle.

In this respect, we presented a preliminary multiphase flow simulation (§ 6.3) that helped us identify and prioritize the expected challenges for an open-air application.

This initial step resulted in establishing the primary requirement when considering open-air applications: describing an ABL correctly. However, this has never been done using stabilized FEM, so this was the focus of the current work, especially by answering this simple question: How to model an ABL taking advantage of the SUPG stabilization method?

The following step was thus to have a better understanding of how FEM (§ 4.2) is implemented inside the *Coolfluid 3* code and to identify the strengths and weaknesses of the SUPG stabilization method (§ 4.2.4). This would help us discern potential complications during the ABL implementation.

While proposing an implementation, based on the Schumann wall model (§ 4.3.2, 6.4), we first tested the correct adjustment of the velocity, depending on the position of the first node, above the physical wall (§ 6.4.2).

Then, we isolated the issues and solved them sequentially:

- We started with a boundary condition normalization to ensure scalability between equivalent domains with varying resolutions (§ 6.5).

- Subsequently, we observed an unstable velocity profile, reaching velocities above and beyond the reference velocity, which could not be physical. We first tried to tune the SUPG stabilization coefficients (§ 6.6). These effectively had a positive influence, but unfortunately, the effect was insufficient compared to the velocity variations. The second attempt (§ 6.7) was more artificial, but effective. For this second solution, we implemented a body force limiter, ensuring control of the body force driving the simulation, hence, the velocity in the domain.

- The third issue that we encountered was related to numerical oscillations occurring near the wall and indicating the SUPG method was insufficient to dissipate the perturbations in this region. The first reaction (§ 6.9) was to ensure that the issue was not coming from the implementation by proposing two different schemes for the implementation: implicit and semi-implicit schemes. Having ascertained the wall model implementation was not the issue (by generating the same results with both schemes), we decided to investigate methods to add numerical dissipation:

  – The first proposed method can be considered as an evolution of the SUPG method. It is called the VMS (§ 5.1) and, although promising, this modern approach lacks validation cases and details that caused the development to be laborious. Nevertheless, its predictor / multi-corrector structure, as well as part of its algorithm, could be implemented and validated (§ 7.6).

  – The second approach is a more conventional CFD method and is based on LES turbulence models. These models are widely known for producing dissipation.

    Within the LES options, we choose first to apply the modern WALE method (§ 5.2.2). The latter regrettably produced too much dissipation (§ 7.2) and, as a consequence, induced a complete laminar flow. Because of its automatic adjusting behavior, it could not be adapted to better approximate the ABL.

    As a consequence, the second chosen method was less sophisticated yet known to generate outstanding results. This method is the Smagorinsky-Lilly method (§ 5.2.1). Two versions exist, the static and the dynamic. The former requires setting a coefficient C_s to a constant value. The latter adapts this coefficient spatially and temporally. We implemented both methods (§ 7.3), and both generated velocity profiles where the spurious numerical oscillations completely disappeared while the turbulence remained. The issue was solved. However, while comparing the resulting velocity profile to the expected analytical profile, a clear difference

appeared. Notice that, although the dynamic version is the evolution of the static, the latter gave the best results.

- Having solved the spurious numerical oscillation issue, the following quest was now directed towards shifting the simulated velocity profile in the direction of the analytical profile.

  For this purpose, we first initiated sensitivity studies to evaluate the influence of chosen parameters: the grid resolution (§ 7.4.2) and the reference velocity (§ 7.4.3). To facilitate these studies, a preliminary study was necessary to optimize the computational setup (§ 7.4.1). The outcome of these studies was that each of these parameters does have a positive impact on the result. Nevertheless, the analytical profile could not be reproduced. However, we could observe a clear improvement when reducing the reference velocity and, in extenso, reducing the corresponding Reynolds number of the flow.

  As a consequence, we decided to reduce the system's complexity by decreasing its viscous Reynolds number to 1000, enabling us to compare our results to an available non-dimensional DNS result (§ 7.5). In this last attempt, we compared three options:

  – to apply a non-slip condition on the wall,

  – to apply a velocity on the wall equal to the friction velocity,

  – to apply our wall model implementation.

  The resulting profiles still deviated from the DNS reference, but the result generated by the wall model was clearly a better match than the two other options.

To conclude, although we were not yet able to accurately simulate the ABL, solutions were discovered for numerous difficulties, bringing us closer to the expected behavior and enabling us to answer the original question. That is, the stabilization provided by the SUPG method is insufficient for the ABL. Nevertheless, adding LES turbulence models does stabilize the solution. The VMS is also an interesting possible solution, but we need to elaborate our implementation of this method further.

In the next section, we will propose a few directions to approach the ideal ABL representation further.

8.2 Further work

In the continuity of this work, the idea would be to extend the investigation at $Re_\tau = 1000$ and reach the matching before increasing over again towards the real ABL case.

Future with low Reynolds

To reach this matching, we propose four directions:

- Applying the body force limiter is an artificial control. Theoretically, at equilibrium, no limiter should be necessary. Practically, the limiter is not only necessary, it is improving the solution. The disagreement between theory and practice necessitates further investigations.

- One could investigate the influence of the two other velocity components (e.g. lateral and vertical) inside the ABL implementation while applying the Smagorinsky-Lilly turbulence model to keep the profile stable.

- Another interesting direction is to study the variety of available wall models (§ 4.3.2). In this work, predecessors mainly inspired our choice, but other models dedicated to high Reynolds number and coarse mesh might perform better.

- Last but not least, in the SUPG direction, one could further develop the VMS implementation. In this respect, interesting resources are the reference book of Donea and Huerta, 2003, and the article of Bazilevs et al., 2007.

From real scale to real case

When the match is in sight, one must carry out a sensitivity study on the surface roughness parameter to stress-test the ABL implementation (Goit, 2015). Consequently, one will need to include the Coriolis effect (Bechmann, 2006) to match the neutral ABL properties perfectly.

From this point, the second phase can be integrated (Janssens, 2014) to extend the applicability and produce the primary CBRN test case.

More capabilities

Then, to improve the capabilities of the implemented tool, various directions can be followed:

- Physical complexity: Integrate the energy equation to account for thermal effects. This will generalize the neutral ABL expression to an ABL that is more realistic. Moreover, it will better represent the whole day cycle.

- Turbulence excitation: To accelerate turbulence creation, we use a basic randomizer. One could incorporate a more dynamic adjustment of this randomizer, spatially and in function of the expected profile.

- Computation performance: One could perform studies to improve even more the computation time. Effectively, the MueLu preconditioner contains many options that permit adjustment and optimize the assembly matrix for the ABL case. Such a study would require further deepening into computer science rather than CFD. However, at a certain point, a numeriscist in CFD is already confronted with computer science.

Interests

Eventually, following the motivation chapter (§ 2), a considerable amount of applications could benefit from this implementation, starting with the Defense, internally, on the site of Poelkapelle, where open-air explosions are performed, but also the University of La Rochelle that is currently carrying an experimental air quality investigation in urban areas. In the same direction, in September 2021, Brussels will start an experimental survey on atmospheric pollution in an urban area (CurieuzenAirBxl, 2021). In fact, it has been a hot topic for the last five years and is only at its prelude, according to the late *fit for 55* European legislation on greenhouse gases reduction, presented in July 2021 (Europarl, 2021).

Appendix A

Implementation details

A.1 The body force Limiter.

```
1  void BodyForceControl::execute()
2  {
3    using boost::proto::lit;
4    const std::string tag = options().option("velocity_field
     ↪  _tag").value<std::string>();
5    Eigen::Map<RealVector> uRef(&m_uRef[0],m_uRef.size());
6    FieldVariable<0, VectorField> u("Velocity", tag);
7
8    /// uMean computation
9    RealVector uMean(physical_model().ndim());
10   uMean.setZero();
11   surface_integral(uMean, m_surface_regions, u);
12   Real surface = 0.0;
13   surface_integral(surface, m_surface_regions, lit(1.0));
14   uMean /= surface;
15
16   /// uInteg computation
17   if(uInteg.size() == 0)                    /// modify
     ↪  uInteg only at 1st iter
18   {
19     uInteg.resize(physical_model().ndim()); /// Resize to
       ↪  speed dim(x, y, z)
20     uInteg.setZero();                       /// Initialize
       ↪  to zero
21   }
22   uInteg += (uMean - uRef) * m_time->dt();
23
24   /// Correction factor
25   m_correction.resize(physical_model().ndim());
26   m_correction = aCoef * uInteg + bCoef * (uMean - uRef);
     ↪  /// Goldstein p.356
27
```

```
28    CFinfo << "Computed body_force: " <<
      ↪  m_correction.transpose() << CFendl;
29
30    ProtoAction::execute();
31  }
```

A.2 The ABL implementation.

Header file

```
1   #include "UnsteadyAction.hpp"
2   #include <solver/actions/Proto/BlockAccumulator.hpp>
3   #include "LibUFEM.hpp"
4   #include "LSSAction.hpp"
5
6   namespace cf3 {
7     namespace math { namespace LSS { class System; } }
8     namespace mesh { class Region; }
9   namespace UFEM {
10
11  class UFEM_API BCWallFunctionABLGoitSI : public
    ↪  UnsteadyAction
12  {
13  public:
14    /// Contructor
15    /// @param name of the component
16    BCWallFunctionABLGoitSI ( const std::string& name );
17
18    /// Destructor
19    virtual ~BCWallFunctionABLGoitSI();
20
21    /// Get the class name
22    static std::string type_name () { return
      ↪  "BCWallFunctionABLGoitSI"; }
23
24    /// Execute the control of the ABL
25    virtual void execute();
26
27  private:
28    /// Called when the boundary regions are set
29    virtual void on_regions_set();
30
31    /// Called when the "lss" options is changed
```

```
32    void trigger_setup();
33
34    /// variables
35    cf3::solver::actions::Proto::SystemRHS rhs;
36    cf3::solver::actions::Proto::SystemMatrix system_matrix;
37    Real m_theta = 0.5;
38    /// Default from KEpsilonPhysics.cpp. Else, from the
      ↪  model
39    Real m_c_mu = 0;
40    Real m_kappa = 0;
41    Real m_zwall = 0;
42    Real m_z0 = 0;
43    Real m_uplus = 0;
44  };
```

Main function in the ABL implementation file

```
1   void BCWallFunctionABLGoitSI::trigger_setup()
2   {
3     Handle<ProtoAction> wall_law(get_child("WallLaw"));
4
5     FieldVariable<0, VectorField> u("Velocity",
      ↪  "navier_stokes_u_solution");
6     FieldVariable<1, ScalarField> p("Pressure",
      ↪  "navier_stokes_p_solution");
7     FieldVariable<2, ScalarField> k("k", "ke_solution");
8     FieldVariable<3, ScalarField>
      ↪  nu_eff("EffectiveViscosity",
      ↪  "navier_stokes_viscosity");
9     FieldVariable<4, VectorField> u_adv("AdvectionVelocity",
      ↪  "linearized_velocity");
10
11    const auto ABL_factor = make_lambda([&]()
12    {
13      Real factor = ::pow(m_kappa/::log(m_zwall/m_z0),2);
14      return factor;
15    });
16
17    // Set normal component to zero and tangential component
      ↪  to the wall-law value
18    wall_law->set_expression(elements_expression
19    (
```

```
    boost::mpl::vector3<mesh::LagrangeP1::Line2D,
    ↪ mesh::LagrangeP1::Triag3D,
    ↪ mesh::LagrangeP1::Quad3D>(),
    group
    (
      _A(u) = _0, //_A(p) = _0, // Assembly version
                  // _a[u] = _0, // rhs version
      element_quadrature
      (
        // _A(p, [u_i]) += -transpose(N(p)) * N(u) *
        ↪ normal[_i], // no-penetration condition
        // _a[u[_i]] += ABL_factor() * _norm(u) *
        ↪ transpose(N(u)) * u[_i] * _norm(normal) *
        ↪ lit(dt()) // rhs version
        _A(u[_i], u[_i]) += _norm(u) * transpose(N(u)) *
        ↪ N(u) * _norm(normal) * lit(dt()) *
        ↪ ABL_factor() // Goit p. 19  // Assembly version
      ),
      system_matrix += m_theta * _A,
      // rhs += -_a // rhs version
      rhs += -_A * _x // Assembly version
    )
  ));
}
```

A.3 Typical Coolfluid 3 ABL case: python inputfile.

```
import os, math
import coolfluid as cf

def simLoop(outputName, interval, meshis='empty',
↪ inputName='out1', restartTstep='0'):
    """
    meshis: 'empty', 'gmesh' or 'restart'
    interval: variable to move to increase the total_time
    restartTstep: timestep (file) from where to restart
    """

    ### Parameters:
    pressureSolve = True

```

```
14      ### Model dimensions: cf. Bechmann p.41
15      h_wt = 20.#1200. # Height windtunnel : Y
16      l_wt = 2*math.pi* h_wt #6 * h_wt  # Length windtunnel
        ↪  : X
17      w_wt = 2*math.pi* h_wt #4 * h_wt  # Width windtunnel :
        ↪  Z
18
19      ### Flow properties
20      rho_fluid = 1.2
21      mu = 1.8e-5 #* 1e4 # Dynamic viscosity of the air
        ↪  ([Pa.s]) ; nu = (mu / rho) = kinematic viscosity
22      nu = mu/rho_fluid
23      kappa = .407
24
25      href = h_wt
26      uref = 5.
27      z0 = h_wt * 5e-4
28      # zwall = 100 * mu / (utau * rho_fluid)   # zwall
        ↪  chosen so that z+=100=Re+=:zwall*utau/nu
29      zwall = h_wt/64 # 12.5
30
31      ### Laminar case
32      #tau_w = -2*nu*uref/h_wt
33      #bf = -tau_w/h_wt
34
35      ### Turbulent case
36      utau = kappa * uref / math.log(href / z0)
37      bf = utau*utau/h_wt
38
39      ### Time parameters
40      tstep = .1 #.002 #2.5e-4 #60*2.5e-4*h_wt/utau
41      #interval = 500#1300#3600 # variable to move to
        ↪  increase the total_time
42      total_time = 10 * interval * tstep # Nb of captures *
        ↪  interval * tstep
43
44      ### Some shortcuts
45      root = cf.Core.root()
46      env = cf.Core.environment()
47
48      ### Global configuration
49      env.assertion_throws = False
50      env.assertion_backtrace = False
51      env.exception_backtrace = False
52      env.regist_signal_handlers = False
```

```
53      env.log_level = 4
54      env.only_cpu0_writes = True
55
56      ### setup a model
57      model = root.create_component('NavierStokes',
        ↪  'cf3.solver.ModelUnsteady')
58      domain = model.create_domain()
59      physics =
        ↪  model.create_physics('cf3.UFEM.KEpsilonPhysics')
60      solver = model.create_solver('cf3.UFEM.Solver')
61
62      ### Add the Navier-Stokes solver as an unsteady solver
63      ns_solver =
        ↪  solver.add_unsteady_solver('cf3.UFEM.NavierStokes')
64
65
66      ### Mesh:
67      if meshis == "empty":
68          blocks = domain.create_component('blocks',
            ↪  'cf3.mesh.BlockMesh.BlockArrays')
69          points = blocks.create_points(dimensions=2,
            ↪  nb_points=4)
70          points[0] = [0.0,  zwall]
71          points[1] = [l_wt, zwall]
72          points[2] = [0.0,  h_wt]
73          points[3] = [l_wt, h_wt]
74          # Block, counter-clockwise ordering
75          block_nodes = blocks.create_blocks(1)
76          block_nodes[0] = [0, 1, 3, 2]
77          # Number of subdivisions in each direction
78          block_subdivs = blocks.create_block_subdivisions()
79          # block_subdivs[0] = [64, 64]
80          block_subdivs[0] = [4, 64]
81          # No grading
82          gradings = blocks.create_block_gradings()
83          gradings[0] = [1.0, 1.0, 1.0, 1.0]
84          # Create the boundaries
85          blocks.create_patch_nb_faces(name='inlet',
            ↪  nb_faces=1)[0] = [2, 0]
86          blocks.create_patch_nb_faces(name='outlet',
            ↪  nb_faces=1)[0] = [1, 3]
87          blocks.create_patch_nb_faces(name='bottom',
            ↪  nb_faces=1)[0] = [0, 1]
88          blocks.create_patch_nb_faces(name='top',
            ↪  nb_faces=1)[0] = [3, 2]
```

```
89          blocks.periodic_x = ['inlet', 'outlet']
90
91          blocks.extrude_blocks(positions=[w_wt],
            ↪  nb_segments=[4], gradings=[1.])
92          blocks.periodic_z = ['front', 'back']
93
94          # Create Mesh
95          mesh = domain.create_component('Mesh',
            ↪  'cf3.mesh.Mesh')
96          blocks.create_mesh(mesh.uri())
97      elif meshis == "restart":
98          mesh =
            ↪  domain.create_component('Mesh','cf3.mesh.Mesh')
99          mesh_reader =
            ↪  domain.create_component('CF3MeshReader',
            ↪  'cf3.mesh.cf3mesh.Reader')
100         mesh_reader.mesh = mesh
101         mesh_reader.file = cf.URI(outputName[:-1] +
            ↪  '-mesh.cf3mesh')
102         mesh_reader.execute()
103         #mesh = domain.load_mesh(file =
            ↪  cf.URI('out-restart-init_P0.msh'), name =
            ↪  'Mesh') # for Gmsh data en mesh file.
104
105     if meshis != "restart":
106         # Write the initial mesh (no fields) -!- Must be
            ↪  before create_point
107         domain.write_mesh(cf.URI(outputName +
            ↪  '-mesh.cf3mesh'))
108
109     ### Pressure set in one center point:
110     create_point_region =
        ↪  domain.create_component('CreatePointRegion',
        ↪  'cf3.mesh.actions.AddPointRegion')
111     create_point_region.coordinates = [l_wt/2., (h_wt -
        ↪  zwall)/64+zwall, w_wt/2.]
112     create_point_region.region_name = 'center'
113     create_point_region.mesh = mesh
114     create_point_region.execute()
115
116
117     # Because of multi-region support, solvers do not
        ↪  automatically have a region assigned,
118     # so we must manually set the solvers to work on the
        ↪  whole mesh
```

```
119     ns_solver.regions = [mesh.topology.uri()]
120
121     # solver setup for NS
122     for lss in [ns_solver.LSS]:
123         lss.SolutionStrategy.Parameters.preconditioner_typ
            ↪ e =
            ↪ 'ML'
124         lss.SolutionStrategy.Parameters.PreconditionerType
            ↪ s.ML.MLSettings.add_parameter(name='ML
            ↪ output', value=0)
125         lss.SolutionStrategy.Parameters.PreconditionerType
            ↪ s.ML.MLSettings.default_values =
            ↪ 'NSSA'
126         lss.SolutionStrategy.Parameters.PreconditionerType
            ↪ s.ML.MLSettings.aggregation_type =
            ↪ 'Uncoupled'
127         lss.SolutionStrategy.Parameters.PreconditionerType
            ↪ s.ML.MLSettings.smoother_type = 'symmetric
            ↪ block Gauss-Seidel'  # 'Chebyshev'
128         lss.SolutionStrategy.Parameters.PreconditionerType
            ↪ s.ML.MLSettings.smoother_sweeps =
            ↪ 2
129         lss.SolutionStrategy.Parameters.PreconditionerType
            ↪ s.ML.MLSettings.smoother_pre_or_post =
            ↪ 'post'
130         lss.SolutionStrategy.Parameters.LinearSolverTypes.
            ↪ Belos.solver_type = 'Block
            ↪ GMRES'
131         lss.SolutionStrategy.Parameters.LinearSolverTypes.
            ↪ Belos.SolverTypes.BlockGMRES.convergence_toler
            ↪ ance =
            ↪ 1e-5
132         lss.SolutionStrategy.Parameters.LinearSolverTypes.
            ↪ Belos.SolverTypes.BlockGMRES.maximum_iteration
            ↪ s =
            ↪ 2000
133         lss.SolutionStrategy.Parameters.LinearSolverTypes.
            ↪ Belos.SolverTypes.BlockGMRES.num_blocks =
            ↪ 1000
134
135     ### Properties for Navier-Stokes
136     physics.density = rho_fluid
137     physics.dynamic_viscosity = mu
138     ### Properties for KEpsilonPhysics
139     physics.href = href
```

```
140     physics.uref = uref
141     physics.z0 = z0
142     physics.zwall = zwall
143     physics.kappa = kappa
144
145     ### Body force control
146     bfControl = solver.add_unsteady_solver('cf3.UFEM.BodyF
        ↪ orceControl')
147     bfControl.velocity_field_tag = 'navier_stokes_solution'
148     bfControl.surface_regions = [mesh.topology.top]
149     bfControl.uRef = [uref, 0., 0.]
150     bfControl.aCoef = -1.
151     bfControl.bCoef = -.2
152     bfControl.regions = [mesh.topology.interior.uri()]
153
154     ### Initial conditions
155     if meshis != "restart":
156         ic_u = solver.InitialConditions.create_initial_con
            ↪ dition(
157             builder_name='cf3.UFEM.InitialConditionFunctio
                ↪ n',
                ↪ field_tag='navier_stokes_solution')
158         ic_u.variable_name = 'Velocity'
159         ic_u.regions = [mesh.topology.uri()]
160         ic_u.value = ['y*{u}/{h}'.format(u=uref,h=h_wt),
            ↪ '0', '0']
161
162     ### Body Force to set u_abl
163     # solver.InitialConditions.navier_stokes_solution.Velo
        ↪ city = [uref,
        ↪ h_wt]
164     solver.InitialConditions.density_ratio.density_ratio =
        ↪ 1.
165     ic_g =
        ↪ solver.InitialConditions.create_initial_condition(
166         builder_name='cf3.UFEM.InitialConditionFunction',
            ↪ field_tag='body_force')
167     ic_g.variable_name = 'Force'
168     ic_g.regions = [mesh.topology.uri()]
169     ic_g.value = ['{f}'.format(f=bf), '0']
170     print 'yop: bodyForce = {f}'.format(f=bf)
171
172     # ### Check value of u1:
173     # u1 = (1/kappa) * math.log(zwall/z0) *
        ↪ math.sqrt((utau*utau/h_wt) * h_wt)
```

```
174     # print 'yop: u1 = {u1}'.format(u1=u1)
175     # print 'yop: CFL = u*dt/dx:
        ↪  {0}'.format(uref*tstep/(h_wt/64))
176
177     ### Boundary conditions
178     bc = ns_solver.get_child('BoundaryConditions')
179     bc.add_constant_bc(region_name='center',
        ↪  variable_name='Pressure').value = 0.
180     bc.add_constant_component_bc(region_name='top',
        ↪  variable_name='Velocity', component=1).value = 0.
181     bc.add_constant_component_bc(region_name='bottom',
        ↪  variable_name='Velocity', component=1).value = 0.
182     bc.create_bc_action(region_name = 'bottom',
        ↪  builder_name = 'cf3.UFEM.BCWallFunctionABLGoit')
183     # bc.add_constant_bc(region_name='bottom',
        ↪  variable_name='Velocity').value = [0., 0.]
184
185     ### Datas needed to restart (better to always produce
        ↪  them in case we need a restart):
186     restart_writer = solver.add_restart_writer()
187     restart_writer.Writer.file = cf.URI(outputName +
        ↪  '-{iteration}.cf3restart')
188     restart_writer.interval = int(interval)
189
190     ### Files writing
191     write_manager = solver.add_unsteady_solver('cf3.solver
        ↪  .actions.TimeSeriesWriter')
192     write_manager.interval = int(interval)
193     writer = write_manager.create_component('VTKWriter',
        ↪  'cf3.mesh.VTKXML.Writer')
194     writer.mesh = mesh
195     writer.fields = [
196         cf.URI('/NavierStokes/Domain/Mesh/geometry/navier_
            ↪  stokes_solution'),
197         cf.URI('/NavierStokes/Domain/Mesh/geometry/body_fo
            ↪  rce'),
198         cf.URI('/NavierStokes/Domain/Mesh/geometry/turbule
            ↪  nce_statistics')]
199     writer.file = cf.URI(outputName + '-{iteration}.pvtu')
200
201     ### Time setup
202     time = model.create_time()
203     time.time_step = tstep
204     time.end_time = total_time
205
```

```
206     ### Randomize the initial condition: (to put after the
        ↪  time setup)
207     if meshis != "restart":
208         solver.create_fields()
209         randomizer =
            ↪  solver.InitialConditions.create_component(
210             'Randomizer',
                ↪  'cf3.solver.actions.RandomizeField')
211         randomizer.field =
            ↪  mesh.geometry.navier_stokes_solution
212         randomizer.variable_name = 'Velocity'
213         randomizer.maximum_variations = [0.2, 0.05, 0.2]
214         randomizer.maximum_values = [uref, uref/3.0,
            ↪  uref/3.0]
215         randomizer.minimum_values = [-uref, -uref/3.0,
            ↪  -uref/3.0]
216
217     ### Read restart file
218     if meshis == "restart":
219         reader = solver.InitialConditions.Restarts.create_
            ↪  component('Reader',
            ↪  'cf3.solver.actions.ReadRestartFile')
220         reader.mesh = mesh
221         reader.time = time
222         reader.file = cf.URI(inputName + '-'+
            ↪  str(restartTstep) +'.cf3restart')
223         reader.read_time_settings = False # to initialize
            ↪  the current time to 0
224         solver.create_fields()
225         solver.InitialConditions.execute()
226         #domain.write_mesh(cf.URI('out-restart-init.pvtu'))
227         #domain.write_mesh(cf.URI('out-restart-init.cf3mes
            ↪  h')) # create mesh w/ infos for 1
            ↪  cpu
228
229     ### Statistics
230     # stats = domain.create_component('Statistics',
        ↪  'cf3.solver.actions.TurbulenceStatistics')
231     stats = solver.add_unsteady_solver('cf3.solver.actions
        ↪  .TurbulenceStatistics')
232     stats.region = mesh.topology
233     stats.file = cf.URI(outputName +
        ↪  '-turbulence-statistics.txt')
234     stats.rolling_window = 1000
```

```
235     stats.add_probe([l_wt/2., (h_wt - zwall)/64+zwall,
        ↪  w_wt/2.]) # ! it has to be on an existing node !
236     #stats.options.count = 18000
237     stats.setup()
238     ### Averaging
239     dir_avg = solver.TimeLoop.children.WriteRestartManager
        ↪  .create_component('DirectionalAverage',
        ↪  'cf3.solver.actions.DirectionalAverage')
240     dir_avg.direction = 1
241     dir_avg.field = mesh.geometry.turbulence_statistics
242     dir_avg.file = cf.URI(outputName +
        ↪  '-turbulence-statistics-profile-{iteration}.txt')
243
244     ### Probes setting:
245     def setProbe(x1, y1, z1, typ="v"):
246         probe = solver.add_probe(name =
            ↪  'Probe-x{x}y{y}'.format(x=int(x1),
247             y=int(y1)), parent = ns_solver, dict =
                ↪  mesh.geometry)
248         if typ == "f":
249             probe.Log.variables = ['Force[0]']
250         else:
251             probe.Log.variables = ['Velocity[0]']
252         probe.coordinate = [x1, y1, z1]
253         probe.History.file = cf.URI(outputName +
254                 '-probe-{typ}-x{x}y{y}.tsv'.format(typ=typ
                    ↪  , x=int(x1),
                    ↪  y=int(y1)))
255
256     setProbe(l_wt/2, (h_wt-zwall), w_wt/2) # Top
257     setProbe(l_wt/2, (h_wt-zwall)*2/3, w_wt/2) # 2/3
258     setProbe(l_wt/2, (h_wt-zwall)/2, w_wt/2) # 1/2
259     setProbe(l_wt/2, (h_wt-zwall)/3, w_wt/2) # 1/3
260     setProbe(l_wt/2, zwall, w_wt/2) # Bottom
261     setProbe(l_wt/2, (h_wt-zwall)/2, w_wt/2, "f") # 1/2
262
263     ### Run the simulation
264     model.simulate()
265     #domain.write_mesh(cf.URI('out-final.pvtu'))
266     #model.print_timing_tree()
267     #mesh.print_tree()
268
269 ### Run the simulation
270 simLoop('out', 1) # dt=.1 #0.01
```

A.4 WALE core part of the implementation.

```
/// Proto functor to compute the turbulent viscosity
struct ComputeNuWALE
{
  template<typename UT, typename NUT, typename ValenceT>
  void operator()(const UT& u, NUT& nu, const ValenceT&
  ↪  valence, const Real nu_visc) const
  {
    typedef typename UT::EtypeT ElementT;
    static const Uint dim = ElementT::dimension;
    typedef mesh::Integrators::GaussMappedCoords<1,
    ↪  ElementT::shape> GaussT;
    typedef Eigen::Matrix<Real, dim, dim> SMatT;

    const SMatT grad_u = u.nabla(GaussT::instance().coords
    ↪  .col(0))*u.value();
    const SMatT S = 0.5*(grad_u + grad_u.transpose());
    const Real S_norm2 = S.squaredNorm();
    const SMatT grad_u2 = grad_u*grad_u.transpose();

    SMatT Sd = 0.5*(grad_u2 + grad_u2.transpose());
    Sd.diagonal().array() -= grad_u2.trace()/3.;
    const Real Sd_norm2 = Sd.squaredNorm();

    Real f = 1.; // anisotropic coefficient
    // [...] Anisotropic cell size adjustment hidden.

    const Real delta_iso = ::pow(u.support().volume(),
    ↪  2./3.); // Isotropic length scale

    Real nu_t = cw*cw*f*f*delta_iso * ::pow(Sd_norm2, 1.5)
    ↪  / (::pow(S_norm2, 2.5) + ::pow(Sd_norm2, 1.25));
    if(nu_t < 0. || !std::isfinite(nu_t))
      nu_t = 0.;

    const Eigen::Matrix<Real, ElementT::nb_nodes, 1>
    ↪  nodal_vals = (nu_t + nu_visc) *
    ↪  valence.value().array().inverse();
    nu.add_nodal_values(nodal_vals);
  }
  Real cw; // Model constant
};
```

A.5 Smagorinsky - Lilly core part of the implementation.

```
1  /// Proto functor to compute the turbulent viscosity
2  struct ComputeNuSmagorinsky
3  {
4    template<typename UT, typename NUT, typename ValenceT,
     ↪  typename CsT, typename SgsT>
5    void operator()(UT& u, NUT& nu, const ValenceT& valence,
     ↪  const Real nu_visc, Real& cs, CsT& cs_elts, SgsT&
     ↪  sgsLambda_elts) const
6    {
7      Real f = 1.; // anisotropic coefficient
8      // [...] Anisotropic cell size adjustment hidden.
9
10     // List all neighbouring elements (center elt incl.)
11     std::set<Uint> neighbElts {};
12     get_neighbElts(neighbElts, u, m_node_connectivity);
13
14     // Compute the dyn Smag parameters
15     typedef typename UT::EtypeT ElementT;
16     static const Uint dim = ElementT::dimension;
17
18     typedef mesh::Integrators::GaussMappedCoords<1,
       ↪  ElementT::shape> GaussT;
19     typedef Eigen::Matrix<Real, dim, dim> SMatT;
20     typedef Eigen::Matrix<Real, 1, dim> SVecT;
21
22     u.compute_values(GaussT::instance().coords.col(0));
       ↪  // Compute u.eval()
23
24     // Initialisation for static smagorinsky:
25     const Real vol_gF = u.support().volume(); // "g"rid
       ↪  "F"ilter = volume of center elt
26     const Real delta2_gF = ::pow(vol_gF, 2./3.); // square
       ↪  of isotropic length scale for grid filter
27     SMatT grad_u = u.nabla(GaussT::instance().coords.col(0
       ↪  ))*u.value();
28     SMatT S = 0.5*(grad_u + grad_u.transpose());
29     Real S_norm = S.squaredNorm(); // SquaredNorm is the
       ↪  sum of the square of all the matrix entries
       ↪  (Frobenius norm).
30
31     if(use_dynamic_smagorinsky) {
```

```
32        // Initialisation for dynamic smagorinsky (TEST
          ↪  filter > GRID filter):
33        Real smagCoefC = 0; // "C" cte equal to cs^2 ;
          ↪  visible in Germano paper eq(5)
34        Real vol_tF = 0.; // "t"est "F"ilter = sum of volume
          ↪  on all neighbouring elts + center elt (e.g. 27
          ↪  for a unit hexa3D in the center of a 3x3x3 cubus)
35        Uint storedElt_idx = u.support().element_idx(); //
          ↪  store the id of the centered element
36        SVecT u_gtF {}; // grid + test filtered velocity
37        u_gtF.setZero();
38        SMatT S_gF {}; // grid filtered strain-tensor
39        S_gF.setZero();
40        SMatT uu_gtF {}; // grid + test filtered velocities
          ↪  product
41        uu_gtF.setZero();
42        Real S_gF_norm = 0.; // grid filtered strain-tensor
          ↪  magnitude
43        Real SS_norm = 0.; // strain tensor product magnitude
44        SMatT SSxS {}; // grid + test filtered strain tensor
          ↪  product magnitude times grid filtered strain
          ↪  tensor
45        SSxS.setZero();
46
47        for (const auto i : neighbElts)
48        {
49          // set the neighbouring element's data to object u:
50          u.set_element(i);
51          removeConstRef(u.support()).set_element(i); //
            ↪  needed to switch to other elt
52
53          // u.value() gives velocity components for all
            ↪  nodes, u.eval() computed for the elt
54          u.compute_values(GaussT::instance().coords.col(0))
            ↪  ; // Compute
            ↪  u.eval()
55
56          vol_tF += u.support().volume(); // test filter
            ↪  volume
57          u_gtF += u.eval() * u.support().volume(); // test
            ↪  filter velocity
58
59          grad_u = u.nabla(GaussT::instance().coords.col(0))
            ↪  *u.value();
60          S = 0.5*(grad_u + grad_u.transpose());
```

```
61
62          S_gF += S * u.support().volume(); // grid filter strain
63          uu_gtF += u.eval().transpose() * u.eval() * u.support().volume(); // test filter uu
64          SS_norm = std::sqrt(2 * S.squaredNorm()); // SSij
65          SSxS += SS_norm * S * u.support().volume(); // vect product SSxS
66        }
67
68        // Restore and re-initialise the center element's data to object u and recompute its values:
69        u.set_element(storedElt_idx);
70        removeConstRef(u.support()).set_element(storedElt_id, x);
71        u.compute_values(GaussT::instance().coords.col(0));
72        grad_u = u.nabla(GaussT::instance().coords.col(0))*u.value();
73        S = 0.5*(grad_u + grad_u.transpose());
74        S_norm = S.squaredNorm();
75
76        // Normalize the neighbElts stencil sums
77        const Real invVol_tF = 1. / vol_tF;
78        u_gtF *= invVol_tF;
79        S_gF *= invVol_tF;
80        S_gF_norm = std::sqrt(2 * S_gF.squaredNorm()); // norm of grid filter Strain
81        uu_gtF *= invVol_tF;
82        SSxS *= invVol_tF;
83
84        // Find the dynamic Smagorinsky parameter (depending on space and time)
85        const Real delta2_tF = ::pow(vol_tF, 2./3.); // square of isotropic length scale for TEST filter
86        const SMatT M = delta2_gF * SSxS - delta2_tF * S_gF_norm * S_gF;
87        const SMatT L = uu_gtF - u_gtF.transpose() * u_gtF;
88        smagCoefC = L.cwiseProduct(M).sum() / (2. * M.cwiseProduct(M).sum());
89
90        // Limiters
91        smagCoefC = std::max(smagCoefC,0.); // limiter to avoid NaN
92        cs = std::sqrt(smagCoefC); // Come back to the expression of cs:
```

```
93        cs = std::min(cs, 0.23); // limiter cf. Lilly and
          ↪ Germano's paper
94      }
95      Real sgsLambda = cs * std::sqrt(delta2_gF); // Subgrid
        ↪ Scale length scale
96
97      if(use_mason_wallDamping) {
98        Real zcoord = u.support().coordinates(GaussT::instan
          ↪ ce().coords.col(0))[1]; // In fact y-coordinate
          ↪ (...[1]) but it is seen as the z-coordinate for
          ↪ us...
99        sgsLambda = std::pow(std::pow(sgsLambda,
          ↪ (-m_masonCoef)) + std::pow(m_kappa*(zcoord +
          ↪ m_z0), (-m_masonCoef)), (-1./m_masonCoef));
100     }
101
102     // Compute the viscosity
103     // Real nu_t = cs*cs*delta2_gF *f*f*
        ↪ std::sqrt(2*S_norm);
104     Real nu_t = std::pow(sgsLambda, 2) * f*f *
        ↪ std::sqrt(2*S_norm);
105
106     if(nu_t < 0. || !std::isfinite(nu_t))
107       nu_t = 0.;
108
109     const Eigen::Matrix<Real, ElementT::nb_nodes, 1>
        ↪ nodal_vals = (nu_t +
        ↪ nu_visc)*valence.value().array().inverse();
110     const Eigen::Matrix<Real, ElementT::nb_nodes, 1>
        ↪ cs_vals = cs*valence.value().array().inverse();
111     const Eigen::Matrix<Real, ElementT::nb_nodes, 1>
        ↪ sgsLambda_vals =
        ↪ sgsLambda*valence.value().array().inverse();
112     nu.add_nodal_values(nodal_vals);
113     cs_elts.add_nodal_values(cs_vals);
114     sgsLambda_elts.add_nodal_values(sgsLambda_vals);
115   }
116
117   // Model constant
118   Real m_cs;
119   int m_masonCoef; // Mason coefficient. Set to 3
      ↪ according to Goit.
120   bool use_anisotropic_correction;
121   bool use_dynamic_smagorinsky;
122   bool use_mason_wallDamping;
```

```
123   mesh::NodeConnectivity* m_node_connectivity = nullptr;
124   Real m_kappa = 0;
125   Real m_z0 = 0;
126 };
127 }
```

A.6 MUMPS and MueLu activation in python input file.

MUMPS activation.

```
1  ns_solver.PressureLSS.solution_strategy =
   ↪  'cf3.math.LSS.DirectStrategy'
2
3  lssP = ns_solver.PressureLSS.LSS
4  lssP.SolutionStrategy.solver_type = 'Amesos_Mumps'
```

MueLu activation.

```
1  ns_solver.PressureLSS.solution_strategy =
   ↪  'cf3.math.LSS.TrilinosStratimikosStrategy'
2
3  lssP = ns_solver.PressureLSS.LSS
4  lssP.SolutionStrategy.Parameters.LinearSolverTypes.Belos.s
   ↪  olver_type = 'Block
   ↪  CG'
5  lssP.SolutionStrategy.Parameters.LinearSolverTypes.Belos.S
   ↪  olverTypes.BlockCG.convergence_tolerance =
   ↪  1e-6
6  lssP.SolutionStrategy.Parameters.LinearSolverTypes.Belos.S
   ↪  olverTypes.BlockCG.maximum_iterations =
   ↪  300
7  lssP.SolutionStrategy.Parameters.LinearSolverTypes.Belos.S
   ↪  olverTypes.BlockCG.assert_positive_definiteness =
   ↪  False
8  lssP.SolutionStrategy.preconditioner_reset = 20000000
9  lssP.SolutionStrategy.Parameters.preconditioner_type =
   ↪  'MueLu'
10 lssP.SolutionStrategy.Parameters.PreconditionerTypes.MueLu
   ↪  .read_parameter_list("muelu.xml")
```

MueLu specs xml file.

```
<ParameterList name="MueLu">
<!-- ============  GENERAL ================ -->
  <Parameter name="verbosity" type="string" value="high"/>
  <Parameter name="problem: type" type="string" value="Poisson-3D"/>
  <Parameter name="coarse: max size" type="int" value="160"/>
  <Parameter name="multigrid algorithm" type="string" value="sa"/>

  <!-- reduces setup cost for symmetric problems -->
  <Parameter name="transpose: use implicit" type="bool" value="true"/>

  <!-- start of default values for general options (can be omitted) -->
  <Parameter name="max levels" type="int" value="10"/>
  <Parameter name="number of equations" type="int" value="1"/>
  <Parameter name="sa: use filtered matrix" type="bool" value="true"/>
  <!-- end of default values -->

<!-- ============  AGGREGATION  ============ -->
  <Parameter name="aggregation: type" type="string" value="uncoupled"/>
  <Parameter name="aggregation: drop scheme" type="string" value="classical"/>
  <!-- Uncomment the next line to enable dropping of weak connections, which can help AMG convergence for anisotropic problems. The exact value is problem dependent. -->
  <!-- <Parameter name="aggregation: drop tol" type="double" value="0.02"/> -->

<!-- ============  SMOOTHING  ============ -->
  <Parameter name="smoother: type" type="string" value="CHEBYSHEV"/>
  <ParameterList name="smoother: params">
    <Parameter name="chebyshev: degree" type="int" value="4"/>
```

```
27        <!-- <Parameter name="chebyshev: ratio eigenvalue"
          ↪  type="double" value="7"/> -->
28        <!-- <Parameter name="chebyshev: min eigenvalue"
          ↪  type="double" value="1.0"/> -->
29        <Parameter name="chebyshev: zero starting solution"
          ↪  type="bool" value="false"/>
30      </ParameterList>
31      <Parameter name="repartition: enable" type="bool"
        ↪  value="false"/>
32      <Parameter name="repartition: partitioner"
        ↪  type="string" value="zoltan"/>
33      <Parameter name="repartition: start level" type="int"
        ↪  value="2"/>
34      <Parameter name="repartition: min rows per proc"
        ↪  type="int" value="400"/>
35      <Parameter name="repartition: max imbalance"
        ↪  type="double" value="1.1"/>
36      <Parameter name="repartition: remap parts" type="bool"
        ↪  value="false"/>
37      <Parameter name="repartition: rebalance P and R"
        ↪  type="bool" value="true"/>
38  </ParameterList>
```

Appendix B

Magnified figures

B.1 Shear stress equivalence between simulated body force and simulated wall velocity.

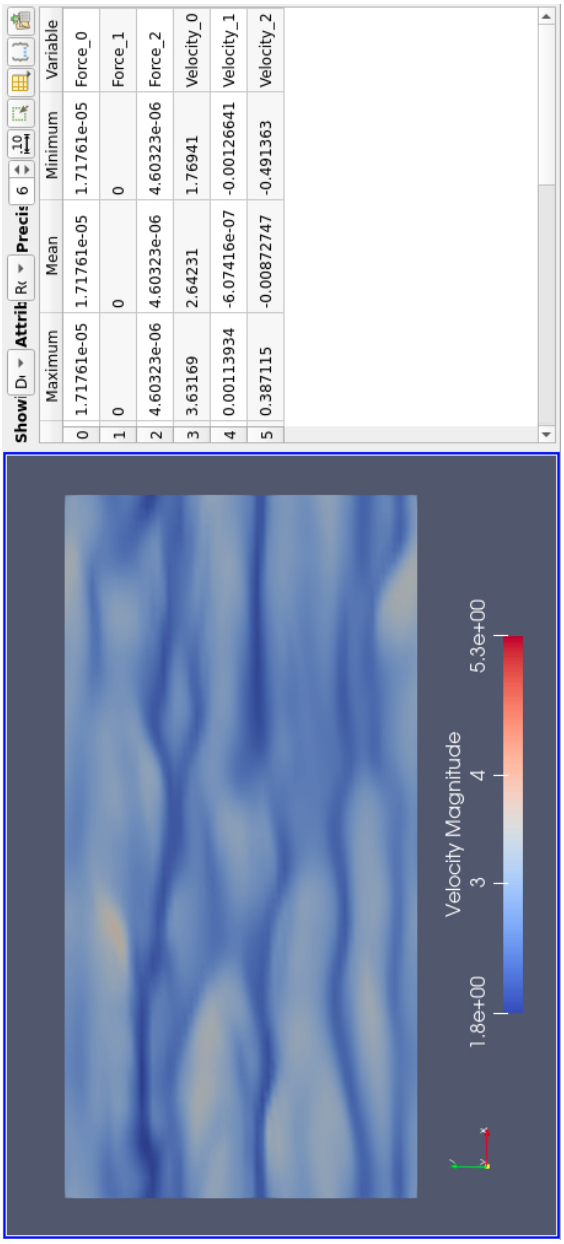


Figure B.1: SL static model: Body Force vs. wall velocity.

B.2 Cs variation for the dynamic Smagorinsky - Lilly implementation.

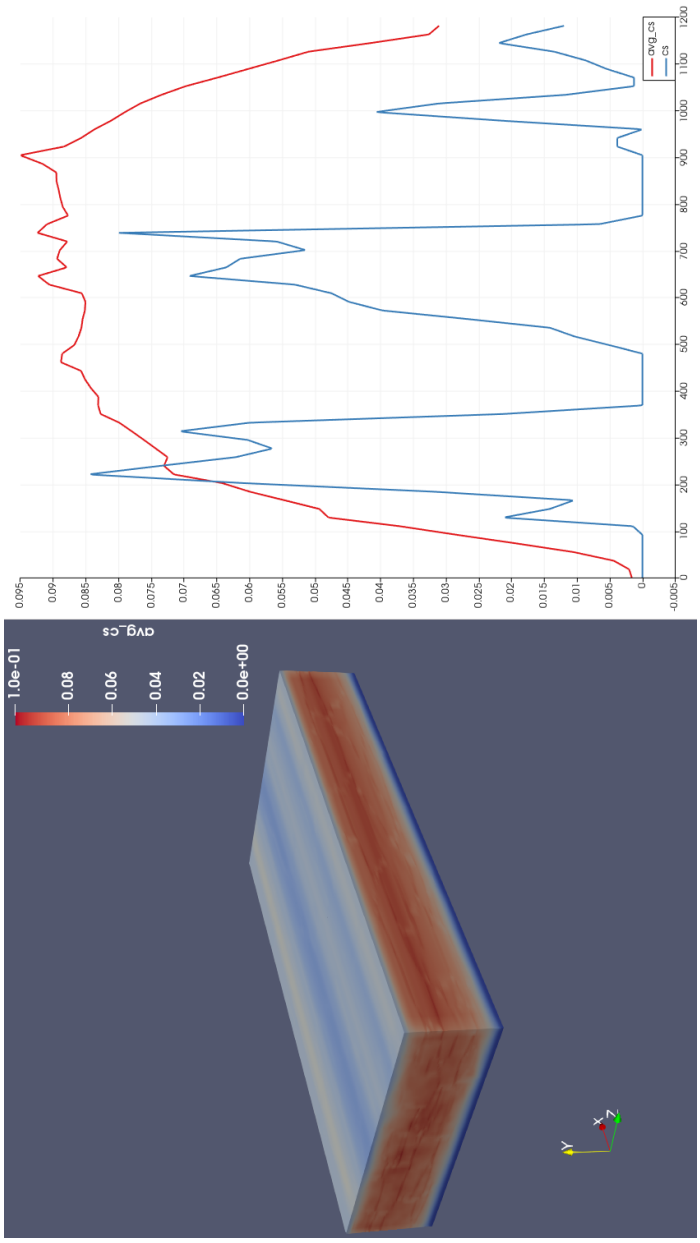


Figure B.2: SL dynamic model: Cs values on the domain.

B.3 Viscosity variation for the dynamic Smagorinsky - Lilly implementation.

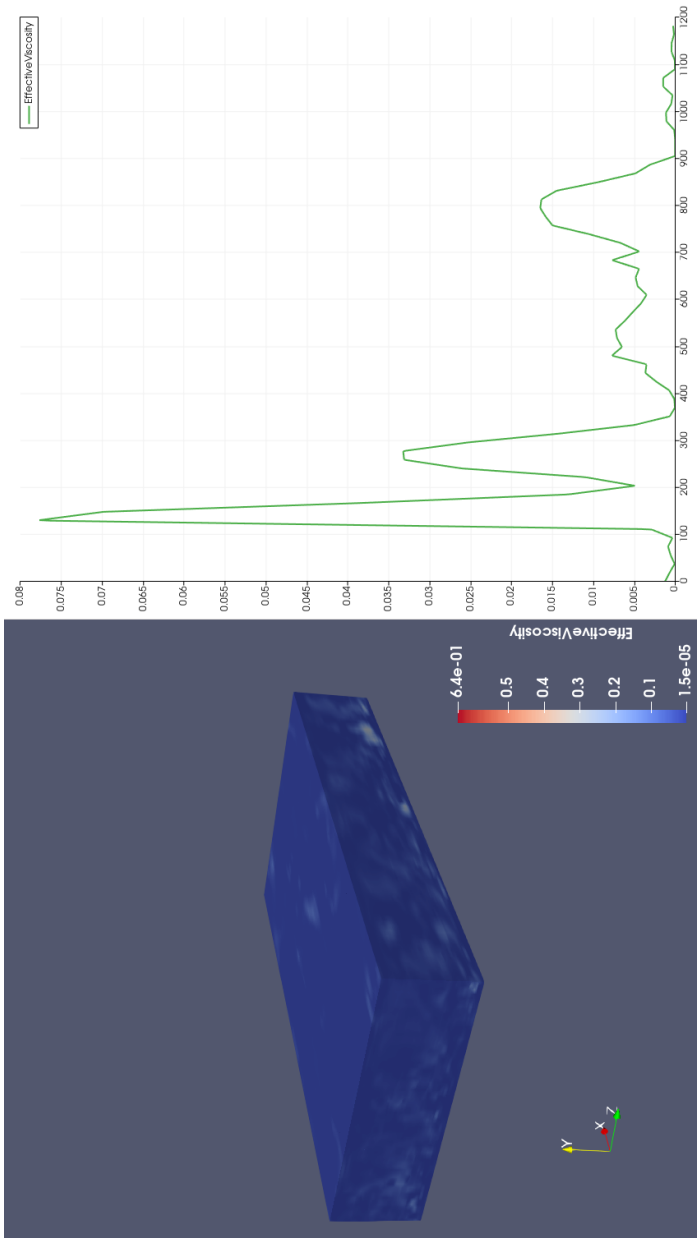


Figure B.3: SL dynamic model: Viscosity on the domain.

Appendix C

Additional values

C.1 MUMPS vs. MueLu solver timings (minimal values).

	mumps64	muelu64	mumps128	muelu128
Grid	64x64x64	64x64x64	128x128x128	128x128x128
Solver	MUMPS	MueLu	MUMPS	MueLu
NS semi-implicit	0.780	0.824	8.688	7.040
10x LinearizeU	1.445e-3	1.436e-3	1.988e-2	2.026e-2
10x PressureLSS	2.979e-5	2.990e-5	3.210e-5	3.254e-5
10x VelocityLSS	2.913e-5	2.896e-5	2.850e-5	2.854e-5
10x SetSolution	6.212e-4	5.790e-4	2.028e-3	2.359e-3
10x InnerLoop	7.650e-1	8.008e-1	8.333	6.850
10x Update	2.392e-3	2.299e-3	6.312e-2	3.815e-2
10x ComputeCFL	7.764e-4	7.267e-4	2.359e-3	2.245e-3
In InnerLoop				
20x URHSAss	1.944e-2	2.137e-2	7.574e-2	7.379e-2
20x PRHSAss	1.482e-2	1.602e-2	5.631e-2	5.438e-2
20x ApplyAup	8.761e-3	8.889e-3	3.228e-2	3.042e-2
20x SolveUSyst	6.035e-2	6.228e-2	2.717e-1	2.738e-1
20x SolvePSyst	1.058e-1	8.425e-2	2.880	1.986

Table C.1: MUMPS vs. MueLu solver timings [s] (min values).

Chapter 9

Bibliography

[1] P. R. Amestoy, I. S. Duff, and J.-Y. L'Excellent. "Multifrontal parallel distributed symmetric and unsymmetric solvers". In: *Computer Methods in Applied Mechanics and Engineering* 184.2-4 (2000), pp. 501–520. DOI: 10.1016/s0045-7825(99)00242-x.

[2] P. R. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. "Performance and Scalability of the Block Low-Rank Multifrontal Factorization on Multicore Architectures". In: *ACM Transactions on Mathematical Software* 45.1 (2019), pp. 1–26. DOI: 10.1145/3242094.

[3] D. A. Anderson, J. C. Tannehill, and R. H. Pletcher. *Computational fluid mechanics and heat transfer*. Washington New York: Hemisphere Pub. Corp. McGraw-Hill, 1984.

[4] J. D. Anderson. *Modern Compressible Flow With Historical Perspective*. Tata McGraw-Hill Education, 2003.

[5] H. M. ApSimon, H. F. Macdonald, and J. J. N. Wilson. "An initial assessment of the Chernobyl-4 reactor accident release source". In: *Journal of the Society for Radiological Protection* 6.3 (1986), pp. 109–119. DOI: 10.1088/0260-2814/6/3/002.

[6] H. M. ApSimon and J. J. N. Wilson. "Modelling Atmospheric Dispersal of the Chernobyl Release Across Europe". In: *Interactions between Energy Transformations and Atmospheric Phenomena. A Survey of Recent Research*. Springer Netherlands, 1987, pp. 123–133. DOI: 10.1007/978-94-017-1911-7_9.

[7] W. K. H. Ariyaratne, E. V. P. J. Manjula, C. Ratnayake, and M. C. Melaaen. "CFD Approaches for Modeling Gas-Solids Multiphase Flows - A Review". In: *Proceedings of The 9th EUROSIM Congress on Modelling and Simulation, EUROSIM 2016, The 57th SIMS Conference on Simulation and Modelling SIMS 2016*. Linköping University Electronic Press, 2018. DOI: 10.3384/ecp17142680.

[8] P. Armand, C. Duchenne, and E. Bouquot. "Atmospheric Dispersion Modelling and Health Impact Assessment in the Framework of a CBRN-E Training Exercise in a Complex Urban Configuration". In: 2014.

[9] P. S. Arya. *Introduction to Micrometeorology*. Elsevier Science and Techn, Apr. 25, 2001. 420 pp.

[10] M. Aslefallah, D. Rostamy, and K. Hosseinkhani. "Solving time-fractional differential diffusion equation by theta-method". In: *Int. J. of Adv. in Aply. Math. and Mech* 2 (Sept. 2014), pp. 1–8.

[11] B. Aupoix and J. Cousteix. "Simple subgrid scale stresses models for homogeneous isotropic turbulence". In: *ReAer* 4 (1982), pp. 1–10.

[12] B. Baldwin and H. Lomax. "Thin-layer approximation and algebraic model for separated turbulentflows". In: *16th Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics, 1978. DOI: 10.2514/6.1978-257.

[13] B. S. Baldwin and T. J. Barth. "A One-Equation Turbulence Transport Model for High Reynolds Number Wall-Bounded Flows". In: *National Aeronautics and Space Administratbn* (1990).

[14] T. Banyai. "Development of Stabilized Finite Element Method for Numerical Simulation of Turbulent Incompressible Single and Eulerian-Eulerian Two-Phase Flows". PhD thesis. Universite libre de Bruxelles, 2016.

[15] T. Bányai, D. Vanden Abeele, and H. Deconinck. "A fast fully-coupled solution algorithm for the unsteady incompressible Navier-Stokes equations". In: *Conference on Modelling Fluid Flow (CMFF'06). Budapest, Hungary*. 2006.

[16] Y. Bazilevs, V. M. Calo, J. A. Cottrell, T. J. R. Hughes, A. Reali, and G. Scovazzi. "Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows". In: *Computer Methods in Applied Mechanics and Engineering* 197.1-4 (2007), pp. 173–201. DOI: 10.1016/j.cma.2007.07.016.

[17] A. Bechmann. "Large-eddy simulation of atmospheric flow over complex terrain". PhD thesis. Risø National Laboratory, 2006.

[18] Y. Benamrane. "La gestion des situations d'urgence à l'interface entre expertise et décision: quelle place pour les outils de modélisation des dispersions NRBC-E et de leurs conséquences?" PhD thesis. Ecole Nationale Supérieure des Mines de Paris, 2015.

[19] L. C. Berselli, T. Iliescu, and W. J. Layton. *Mathematics of Large Eddy Simulation of Turbulent Flows*. Springer-Verlag GmbH, Oct. 20, 2005.

[20] M. Blon. "Untersuchungen zur Messung von FeinstaubDas Citizen Science Projekt luftdaten.info". MA thesis. Ein Modell der Hochschulen Esslingen, Nürtingen, Reutlingen und Stuttgart, 2017.

[21] Bottin. *Mecanique des Fluides*. 2015.

[22] E. Bou-Zeid, C. Meneveau, and M. Parlange. "A scale-dependent Lagrangian dynamic model for large eddy simulation of complex turbulent flows". In: *Physics of Fluids* 17.2 (2005), p. 025105. DOI: 10.1063/1.1839152.

[23] J. Boussinesq. “Essai sur la théorie des eaux courantes.” In: *Mémoires présentés par divers savants á lAcadémie des Sciences* XXIII, 1 (1877), pp. 1–680.

[24] M. Braack, E. Burman, V. John, and G. Lube. “Stabilized finite element methods for the generalized Oseen problem”. In: *Computer Methods in Applied Mechanics and Engineering* 196.4-6 (2007), pp. 853–866. DOI: 10.1016/j.cma.2006.07.011.

[25] M. Braack and P. B. Mucha. “Directional Do-Nothing Condition for the Navier-Stokes Equations”. In: *Journal of Computational Mathematics* 32.5 (2014), pp. 507–521. DOI: 10.4208/jcm.1405-m4347.

[26] M. Breuer. “eddy simulation of the subcritical flow past a circular cylinder: Numerical and modeling aspects”. In: *Internat. J. Numer. Methods Fluids* (1998), pp. 1281–1302. DOI: 10.1.1.465.4731.

[27] A. N. Brooks and T. J. R. Hughes. “Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations”. In: *Computer Methods in Applied Mechanics and Engineering* 32.1-3 (1982), pp. 199–259. DOI: 10.1016/0045-7825(82)90071-8.

[28] N. Bukowiecki, P. Zieger, E. Weingartner, Z. Jurányi, M. Gysel, B. Neininger, B. Schneider, C. Hueglin, A. Ulrich, A. Wichser, S. Henne, D. Brunner, R. Kaegi, M. Schwikowski, L. Tobler, F. G. Wienhold, I. Engel, B. Buchmann, T. Peter, and U. Baltensperger. “Ground-based and airborne in-situ measurements of the Eyjafjallajökull volcanic aerosol plume in Switzerland in spring 2010”. In: *Atmospheric Chemistry and Physics* 11.19 (2011), pp. 10011–10030. DOI: 10.5194/acp-11-10011-2011.

[29] B. Chaouat. “The State of the Art of Hybrid RANS/LES Modeling for the Simulation of Turbulent Flows”. In: *Flow, Turbulence and Combustion* 99.2 (2017), pp. 279–327. DOI: 10.1007/s10494-017-9828-8.

[30] M. Chino, H. Ishikawa, and H. Yamazawa. “SPEEDI and WSPEEDI: Japanese Emergency Response Systems to Predict Radiological Impacts in Local and Workplace Areas due to a Nuclear Accident”. In: *Radiation Protection Dosimetry* 50.2-4 (1993), pp. 145–152. DOI: 10.1093/rpd/50.2-4.145.

[31] I. Christie, D. F. Griffiths, A. R. Mitchell, and O. C. Zienkiewicz. “Finite element methods for second order differential equations with significant first derivatives”. In: *International Journal for Numerical Methods in Engineering* 10.6 (1976), pp. 1389–1396. DOI: 10.1002/nme.1620100617.

[32] J. Chung and G. M. Hulbert. "A Time Integration Algorithm for Structural Dynamics With Improved Numerical Dissipation: The Generalized alpha Method". In: *Journal of Applied Mechanics* 60.2 (1993), pp. 371–375. DOI: 10.1115/1.2900803.

[33] R. H. Clarke, A. J. Dyer, R. R. Brook, D. G. Reid, and A. J. Troup. *The Wangara experiment : boundary layer data*. Melbourne: C.S.I.R.O, 1971.

[34] R. Codina. "On stabilized finite element methods for linear systems of convection–diffusion-reaction equations". In: *Computer Methods in Applied Mechanics and Engineering* 188.1-3 (2000), pp. 61–82. DOI: 10.1016/s0045-7825(00)00177-8.

[35] C. T. Crowe, J. D. Schwarzkopf, M. Sommerfeld, and Y. Tsuji. *Multiphase Flows with Droplets and Particles*. Taylor and Francis Ltd, Aug. 26, 2011. 509 pp.

[36] CurieuzenAirBxl. *The largest citizen science project on air quality ever carried out in Brussels*. https://curieuzenair.brussels. 2021.

[37] CurieuzeNeuzen. *Air Quality in Flanders, Belgium*. https://www.nature.com/articles/d41586-018-07106-5. 2018.

[38] R. B. Dean. "Reynolds Number Dependence of Skin Friction and Other Bulk Flow Variables in Two-Dimensional Rectangular Duct Flow". In: *Journal of Fluids Engineering* 100.2 (1978), pp. 215–223. DOI: 10.1115/1.3448633.

[39] Defensie. *Dienst voor Opruiming en Vernietiging van Ontploffingstuigen (DOVO) in Belgie*. https://www.mil.be. https://youtu.be/RpIYFLkQhnY?t=793. 2019.

[40] J. Donea and A. Huerta. *Finite Element Methods for Flow Problems*. John Wiley and Sons, May 2003. 364 pp.

[41] C Duchenne, P Armand, M Nibart, and V Hergault. "Validation of a LPDM against the CUTE experiments of the COST ES1006 Action–Comparison of the results obtained with the diagnostic and RANS versions of the flow model". In: *Proceedings of the 15th Harmo Conference*. 2013.

[42] F. Ducros, F. Nicoud, and T. Poinsot. "Wall-adapting local eddy-viscosity models for simulations in complex geometries". In: *Numerical Methods for Fluid Dynamics VI* (1998), pp. 293–299.

[43] P. A. Durbin. "Separated flow computations with the k-epsilon-v-squared model". In: *AIAA Journal* 33.4 (1995), pp. 659–664. DOI: 10.2514/3.12628.

[44] S. Ebrahimi. *Finite Difference Method (FDM)*. Tech. rep. Faculty of Sciences University of Kurdistan, 2010.

[45] K. Elsayed and C. Lacor. "The effect of vortex finder diameter on cyclone separator performance and flow field". In: *ECCOMAS CFD Conf., JCF Pereira and A. Sequeira (Eds.) Lisbon*. 2010.

[46] Europarl. *Package Fit for 55*. https://www.europarl.europa.eu/legislative-train/theme-a-european-green-deal/package-fit-for-55. 2021.

[47] R. Fan, D. L. Marchisio, and R. O. Fox. "Application of the direct quadrature method of moments to polydisperse gas–solid fluidized beds". In: *Powder Technology* 139.1 (2004), pp. 7–20. DOI: 10.1016/j.powtec.2003.10.005.

[48] C. A. Felippa. "A historical outline of matrix structural analysis: a play in three acts". In: *Computers and Structures* 79.14 (2001), pp. 1313–1324. DOI: 10.1016/s0045-7949(01)00025-6.

[49] J. Ferry and S. Balachandar. "A fast Eulerian method for disperse two-phase flow". In: *International Journal of Multiphase Flow* 27.7 (2001), pp. 1199–1226. DOI: 10.1016/s0301-9322(00)00069-0.

[50] J. H. Ferziger, M. Perić, and R. L. Street. *Computational methods for fluid dynamics*. Vol. 3. Springer, 2002. DOI: 10.1007/978-3-319-99693-6.

[51] R. P. Feynman, R. B. Leighton, and M. Sands. "The feynman lectures on physics; vol. i". In: *American Journal of Physics* 33.9 (1965), pp. 750–752.

[52] R. O. Fox. *Computational models for turbulent reacting flows*. Cambridge university press, 2003.

[53] C. García Sánchez. "Quantifying inflow uncertainties for CFD simulations of dispersion in the atmospheric boundary layer". PhD thesis. University of Antwerp, 2017.

[54] M. Germano, U. Piomelli, P. Moin, and W. H. Cabot. "A dynamic subgrid-scale eddy viscosity model". In: *American Institute of Physics* 3 (1991), pp. 1760–1765.

[55] S. Giuliani. "An algorithm for continuous rezoning of the hydrodynamic grid in Arbitrary Lagrangian-Eulerian computer codes". In: *Nuclear Engineering and Design* 72.2 (1982), pp. 205–212. DOI: 10.1016/0029-5493(82)90216-3.

[56] J. P. Goit. "Optimal control of energy extraction in large-eddy simulation of windFarms". PhD thesis. 2015.

[57] D Goldstein, R Handler, and L Sirovich. "Modeling a no-slip flow boundary with an external force field". In: *Journal of Computational Physics* 105.2 (1993), pp. 354–366.

[58] O. Gonzalez and A. M. Stuart. *A First Course in Continuum Mechanics*. Cambridge University Press, Feb. 2015. 414 pp.

[59] A. González-Calderón, L. X. Vivas-Cruz, and E. C. Herrera-Hernández. "Application of the theta-method to a telegraphic model of fluid flow in a dual-porosity medium". In: *Journal of Computational Physics* 352 (2018), pp. 426–444. DOI: 10.1016/j.jcp.2017.09.014.

[60] R. G. Gordon. "Error Bounds in Equilibrium Statistical Mechanics". In: *Journal of Mathematical Physics* 9.5 (1968), pp. 655–663. DOI: 10.1063/1.1664624.

[61] C. Gorlé, J. van Beeck, P. Rambaud, and G. V. Tendeloo. "CFD modelling of small particle dispersion: The influence of the turbulence kineticenergy in the atmospheric boundary layer". In: *Atmospheric Environment* 43 (2009), pp. 673–681.

[62] P. M. Gresho and R. L. Lee. "Don't suppress the wiggles—They're telling you something!" In: *Computers & Fluids* 9.2 (1981), pp. 223–253. DOI: 10.1016/0045-7930(81)90026-8.

[63] R. Grosch, H. Briesen, W. Marquardt, and M. Wulkow. "Generalization and numerical investigation of QMOM". In: *AIChE Journal* 53.1 (2006), pp. 207–227. DOI: 10.1002/aic.11041.

[64] X. Han, T. J. Wray, and R. K. Agarwal. "Application of a New DES Model Based on Wray-Agarwal Turbulence Model for Simulation of Wall-Bounded Flows with Separation". In: *AIAA* (2017).

[65] Y. Hoarau, S.-H. Peng, D. Schwamborn, A. Revell, and C. Mockett. *Progress in Hybrid RANS-LES Modelling*. Springer-Verlag GmbH, Nov. 2019. 412 pp.

[66] J. Hoffman and C. Johnson. "A new approach to computational turbulence modeling". In: *Computer Methods in Applied Mechanics and Engineering* 195.23-24 (2006), pp. 2865–2880. DOI: 10.1016/j.cma.2004.09.015.

[67] M. Holt, R. J. Campbell, and M. B. Nikitin. *Fukushima nuclear disaster*. Tech. rep. Congressional Research Service, 2012.

[68] C.-H. Hu, M. Ohba, and R. Yoshie. "CFD modelling of unsteady cross ventilation flows using LES". In: *Journal of Wind Engineering and Industrial Aerodynamics* 96.10-11 (2008), pp. 1692–1706. DOI: 10.1016/j.jweia.2008.02.031.

[69] A. Huerta, A. Rodriguez-Ferran, P. Diez, and J. Sarrate. "Adaptive finite element strategies based on error assessment". In: *International Journal for Numerical Methods in Engineering* 46.10 (1999), pp. 1803–1818. DOI: 10.1002/(sici)1097-0207(19991210)46:10<1803::aid-nme725>3.0.co;2-3.

[70] T. J. R. Hughes and T. E. Tezduyar. "Finite element methods for first-order hyperbolic systems with particular emphasis on the compressible euler equations". In: *Computer Methods in Applied Mechanics and Engineering* 45.1-3 (1984), pp. 217–284. DOI: 10.1016/0045-7825(84)90157-9.

[71] T. J. R. Hughes. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. DOVER PUBN INC, Aug. 2000. 672 pp.

[72] T. J. R. Hughes, W. K. Liu, and A. Brooks. "Finite element analysis of incompressible viscous flows by the penalty function formulation". In: *Journal of Computational Physics* 30.1 (1979), pp. 1–60. DOI: 10.1016/0021-9991(79)90086-x.

[73] T. J. R. Hughes, L. Mazzei, and K. E. Jansen. "Large Eddy Simulation and the variational multiscale method". In: *Computing and Visualization in Science* 3 (2000), pp. 47–59.

[74] H. M. Hulburt and S. Katz. "Some problems in particle technology". In: *Chemical Engineering Science* 19.8 (1964), pp. 555–574. DOI: 10.1016/0009-2509(64)85047-8.

[75] U. Högström. "Von Kármán's Constant in Atmospheric Boundary Layer Flow: Reevaluated". In: *Journal of the Atmospheric Sciences* 42.3 (1985), pp. 263–270. DOI: 10.1175/1520-0469(1985)042<0263:vkciab>2.0.co;2.

[76] IAEA. *INSAG-7. The Chernobyl Accident: Updating of INSAG-1*. Tech. rep. IAEA Safety Series, 1992, p. 148.

[77] E. Isaacson and H. B. Keller. *Analysis of Numerical Methods*. Dover Publications. Corrected reprint of the 1966 original [John Wiley and Sons,New York], June 1994. 576 pp.

[78] Y. Izumi and J. S. Caughey. *Minnesota 1973 atmospheric boundary layer experiment data report*. Vol. 76. 38. Air Force Cambridge Research Laboratories, Air Force Systems Command, United . . ., 1976.

[79] K. Jansen. "Unstructured-grid large-eddy simulation of flow over an airfoil". In: *Center forTurbulence Research, Annual Research Briefs* (1994), pp. 161–173.

[80] K. E. Jansen, C. H. Whiting, and G. M. Hulbert. "A generalized-alpha method for integrating the filtered Navier-Stokes equations with a stabilized finite element method". In: *Computer Methods in Applied Mechanics and Engineering* 190 (2000), pp. 305–319.

[81] B. Janssens. "Numerical modeling and experimental investigation of fine particle coagulation and dispersion in dilute flows". PhD thesis. Université de La Rochelle, 2014.

[82] V. John. *Large Eddy Simulation of Turbulent Incompressible Flows*. Springer Berlin Heidelberg, 2004. DOI: 10.1007/978-3-642-18682-0.

[83] V. John. "Numerical methods for incompressible flow problems I". In: *University Lecture, Freie Universität Berlin* (2014). from https://www.wias-berlin.de/people/john/LEHRE/lehrealt.html.

[84] C. Johnson. *Numerical solutions of partial differential equations by the finite element method*. Ed. by C. U. PRESS. 1987.

[85] D. A. Johnson and L. S. King. "A mathematically simple turbulence closure model for attached and separated turbulent boundary layers". In: *AIAA Journal* 23.11 (1985), pp. 1684–1692. DOI: 10.2514/3.9152.

[86] W. P. Jones and B. E. Launder. "The prediction of laminarization with a two-equation model of turbulence". In: *International Journal of Heat and Mass Transfer* 15.2 (1972), pp. 301–314. DOI: 10.1016/0017-9310(72)90076-2.

[87] J. C. Kaimal, J. C. Wyngaard, D. A. Haugen, O. R. Coté, Y. Izumi, S. J. Caughey, and C. J. Readings. "Turbulence Structure in the Convective Boundary Layer". In: *Journal of the Atmospheric Sciences* 33.11 (1976), pp. 2152–2169. DOI: 10.1175/1520-0469(1976)033<2152:tsitcb>2.0.co;2.

[88] A. B. Kazanski and A. S. Monin. "On the dynamic interaction between the atmosphere and the Earth's surface". In: *Izv. Acad. Sci. USSR, Geophys. Series* 5 (1961), pp. 514–515.

[89] D. W. Kelly, S. Nakazawa, O. C. Zienkiewicz, and J. C. Heinrich. "A note on upwinding and anisotropic balancing dissipation in finite element approximations to convective diffusion problems". In: *International Journal for Numerical Methods in Engineering* 15.11 (1980), pp. 1705–1711. DOI: 10.1002/nme.1620151111.

[90] A. N. Kolmogorov. "The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers". In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 434.1890 (1991), pp. 9–13. DOI: 10.1098/rspa.1991.0075.

[91] J. G. M. Kuerten. "Subgrid modeling in particle-laden channel flow". In: *Physics of Fluids* 18.2 (2006), p. 025108. DOI: 10.1063/1.2176589.

[92] J. A. Lednicky, M. Lauzardo, Z. H. Fan, A. Jutla, T. B. Tilly, M. Gangwar, M. Usmani, S. N. Shankar, K. Mohamed, A. Eiguren-Fernandez, C. J. Stephenson, M. M. Alam, M. A. Elbadry, J. C. Loeb, K. Subramaniam, T. B. Waltzek, K. Cherabuddi, J. G. Morris, and C.-Y. Wu. "Viable SARS-CoV-2 in the air of a hospital room with

COVID-19 patients". In: *International Journal of Infectious Diseases* 100 (2020), pp. 476–482. DOI: 10.1016/j.ijid.2020.09.025.

[93] M. Lee and R. D. Moser. "Direct numerical simulation of turbulent channel flow up to ReTau of 5200". In: *Journal of Fluid Mechanics* 774 (2015), pp. 395–415.

[94] W. Lefebvre, F. Fierens, C. Vanpoucke, N. Renders, K. Jespers, J. Vercauteren, F. Deutsch, and S. Janssen. "The Effect of Wood Burning on Particulate Matter Concentrations in Flanders, Belgium". In: *Springer Proceedings in Complexity*. Springer International Publishing, 2016, pp. 459–464. DOI: 10.1007/978-3-319-24478-5_73.

[95] B. P. Leonard. "A survey of finite differences of opinion on numerical muddling of the incomprehensible defective confusion equation". In: ed. by T. J. R. Hughes. Finite element methods for convection dominated flows. Vol. 34. AMD, 1979, pp. 1–17.

[96] M. Lesieur. *Turbulence in Fluids*. Springer-Verlag GmbH, Dec. 18, 2007.

[97] D. K. Lilly. "A proposed modification of the Germano subgrid-scale closure method". In: (1991).

[98] Luftdaten. *Luftdaten from Open Knowledge Foundation Deutschland*. https://luftdaten.info/. 2017.

[99] D. L. Marchisio and R. O. Fox. "Solution of population balance equations using the direct quadrature method of moments". In: *Journal of Aerosol Science* 36.1 (2005), pp. 43–73. DOI: 10.1016/j.jaerosci.2004.07.009.

[100] D. L. Marchisio and R. O. Fox. *Computational models for polydisperse particulate and multiphase systems*. Cambridge University Press, 2013.

[101] D. L. Marchisio, J. T. Pikturna, R. O. Fox, R. D. Vigil, and A. A. Barresi. "Quadrature method of moments for population-balance equations". In: *AIChE Journal* 49.5 (2003), pp. 1266–1276. DOI: 10.1002/aic.690490517.

[102] D. L. Marchisio, R. D. Vigil, and R. O. Fox. "Implementation of the quadrature method of moments in CFD codes for aggregation–breakage problems". In: *Chemical Engineering Science* 58.15 (2003), pp. 3337–3351. DOI: 10.1016/s0009-2509(03)00211-2.

[103] D. L. Marchisio, R. D. Vigil, and R. O. Fox. "Quadrature method of moments for aggregation–breakage processes". In: *Journal of Colloid and Interface Science* 258.2 (2003), pp. 322–334. DOI: 10.1016/s0021-9797(02)00054-1.

[104] E. Marshall. "The lessons of Chernobyl". In: *Science* 233 (1986), pp. 1375–1377.

[105] P. J. Mason and D. J. Thomson. "Stochastic backscatter in large-eddy simulations of boundary layers". In: *Journal of Fluid Mechanics* 242 (1992), pp. 51–78. DOI: 10.1017/s0022112092002271.

[106] R. McGraw. "Description of Aerosol Dynamics by the Quadrature Method of Moments". In: *Aerosol Science and Technology* 27.2 (1997), pp. 255–265. DOI: 10.1080/02786829708965471.

[107] F. Menter. "Zonal Two Equation k-w Turbulence Models For Aerodynamic Flows". In: *23rd Fluid Dynamics, Plasmadynamics, and Lasers Conference*. American Institute of Aeronautics and Astronautics, 1993. DOI: 10.2514/6.1993-2906.

[108] F. Meysman and S. de Craemer. *CurieuzeNeuzen Vlaanderen: Het cijferrapport*. Tech. rep. 56 p. Universiteit Antwerpen, 2018.

[109] P. M. Mohite. "History of FEM". In: *Indian Institute of Technology Kanpur* (2001).

[110] A. S. Monin and A. M. Yaglom. *Statistical Fluid Mechanics: Mechanics ofTurbulence, Vol. 1*. MIT Press Cambridge, 1971.

[111] A. S. Monin and A. M. Obukhov. "Basic laws of turbulent mixing in the surface layer of the atmosphere". In: *Contrib. Geophys. Inst. Acad. Sci. USSR* 151.163 (1954), e187.

[112] R. D. Moser, J. Kim, and N. N. Mansour. "Direct numerical simulation of turbulent channel flow up to Re τ=590". In: *Physics of Fluids* 11.4 (1999), pp. 943–945. DOI: 10.1063/1.869966.

[113] F. Moukalled, L. Mangani, and M. Darwish. *The Finite Volume Method in Computational Fluid Dynamics*. Springer-Verlag GmbH, Aug. 2015. 791 pp.

[114] F. Nicoud and F. Ducros. "Subgrid-scale stress modelling based on the square of the velocity gradient tensor". In: *Flow, turbulence and Combustion* 62.3 (1999), pp. 183–200. DOI: 10.1023/A:1009995426001.

[115] T. J. Niemi. "Particle Size Distribution in CFD Simulation of Gas-Particle Flows; Partikkelikokojakauman huomioiminen kaasupartikkelivirtausten simuloinnissa". en. G2 Pro gradu, diplomityö. Aalto University - School of Science, 2012, [8] + 84 s.

[116] G. Oberdörster, E. Oberdörster, and J. Oberdörster. "Nanotoxicology: An Emerging Discipline Evolving from Studies of Ultrafine Particles". In: *Environmental Health Perspectives* 113.7 (2005), pp. 823–839. DOI: 10.1289/ehp.7339.

[117] T. R. Oke. *Review of urban climatology 1968-1973, WMO Technical Note No.134*. Geneva: World Meteorological Organization, 1974.

[118] T. R. Oke. *Boundary Layer Climates (2d Ed.)* Routledge, London, 1987.

[119] U. Piomelli. "Wall-layer models for large-eddy simulations". In: *Progress in Aerospace Sciences* 44.6 (2008), pp. 437–446. DOI: 10.1016/j.paerosci.2008.06.001.

[120] U. Piomelli and E. Balaras. "Wall-Layer Models for Large-Eddy Simulations". In: *Annual Review of Fluid Mechanics* 34.1 (2002), pp. 349–374. DOI: 10.1146/annurev.fluid.34.082901.144919.

[121] C. A. Pope and D. W. Dockery. "Health Effects of Fine Particulate Air Pollution: Lines that Connect". In: *Journal of the Air & Waste Management Association* 56.6 (2006), pp. 709–742. DOI: 10.1080/10473289.2006.10464485.

[122] S. B. Pope. *Turbulent flows*. Ed. by Cambridge. IOP Publishing, 2001.

[123] M. Popovac and K. Hanjalic. "Compound Wall Treatment for RANS Computation of Complex Turbulent Flows and Heat Transfer". In: *Flow, Turbulence and Combustion* 78.2 (2007), pp. 177–202. DOI: 10.1007/s10494-006-9067-x.

[124] L. Prandtl. "The Generation of Vortices in Fluids of Small Viscosity". In: *The Journal of the Royal Aeronautical Society* 31.200 (1927), pp. 718–741. DOI: 10.1017/s0368393100139872.

[125] H. Priemus and E. Schutte-Postma. "Notes on the Particulate Matter Standards in the European Union and the Netherlands". In: *International Journal of Environmental Research and Public Health* 6.3 (2009), pp. 1155–1173. DOI: 10.3390/ijerph6031155.

[126] A. Prokopenko, J. J. Hu, T. A. Wiesner, C. M. Siefert, and R. S. Tuminaro. *MueLu User's Guide 1.0 (Trilinos version 11.12)*. SAND2014-18874. Oct. 2014.

[127] T. Quintino, W. Deconinck, B. Janssens, T. Bányai, and Q. Gasper. *Coolfluid 3*. 2012.

[128] M. M. Rahman, T. Siikonen, and R. K. Agarwal. "Improved Low-Reynolds-Number One-Equation Turbulence Model". In: *AIAA Journal* 49.4 (2011), pp. 735–747. DOI: 10.2514/1.j050651.

[129] D. Ramkrishna. *Population Balances: Theory and Applications to Particulate Systems in Engineering*. Academic PR Inc, July 2000. 355 pp.

[130] V. P. Reshetin and J. L. Regens. "Simulation Modeling of Anthrax Spore Dispersion in a Bioterrorism Incident". In: *Risk Analysis* 23.6 (2003), pp. 1135–1145. DOI: 10.1111/j.0272-4332.2003.00387.x.

[131] H. Ritchie, E. Ortiz-Ospina, D. Beltekian, E. Mathieu, J. Hasell, B. Macdonald, C. Giattino, and M. Roser. *Coronavirus Pandemic (COVID-19)*. https://ourworldindata.org/covid-cases. Accessed: 2020-06-19. 2020.

[132] I. Sadrehaghighi. "Essentials of CFD". In: *CFD Open Series* (2021).

[133] P. Sagaut. *Large Eddy Simulationfor Incompressible Flows*. Ed. by Springer. 2006.

[134] M. Sala, M. A. Heroux, D. M. Day, and J. M. Willenbring. *Trilinos Tutorial*. For Trilinos Release 10.12. 2010.

[135] B. Selma, R. Bannari, and P. Proulx. "Simulation of bubbly flows: Comparison between direct quadrature method of moments (DQ-MOM) and method of classes (CM)". In: *Chemical Engineering Science* 65.6 (2010), pp. 1925–1941. DOI: 10.1016/j.ces.2009.11.018.

[136] N. W. Service. *Cross section of the two main jet streams, by latitude*. 2008.

[137] S. Shuai and R. K. Agarwal. "A New Improved One-Equation Turbulence Model Based on k-kL Closure". In: *AIAA Scitech 2020 Forum*. American Institute of Aeronautics and Astronautics, 2020. DOI: 10.2514/6.2020-1075.

[138] L. F. L. R. Silva, R. C. Rodrigues, J. F. Mitre, and P. L. C. Lage. "Comparison of the accuracy and performance of quadrature-based methods for population balance problems with simultaneous breakage and aggregation". In: *Computers & Chemical Engineering* 34.3 (2010), pp. 286–297. DOI: 10.1016/j.compchemeng.2009.11.005.

[139] O. Simonin. "Statistical and continuum Modelling of turbulent reactive particulate Flows - Part I: Theoretical Derivation of dispersed Phase Eulerian Modelling from Probability Density Function Kinetic Equation". In: *LectureSeriesVKI*. 2005.

[140] J. Smagorinsky. "General Circulation Experiments with the Primitive Equations". In: *Monthly Weather Review* 91.3 (1963), pp. 99–164. DOI: 10.1175/1520-0493(1963)091<0099:gcewtp>2.3.co;2.

[141] A. M. Smith and T. Cebeci. *Numerical Solution of the Turbulent-Boundary-Layer Equations*. Tech. rep. 1967. DOI: 10.21236/ad0656430.

[142] D. M. Snider. "An Incompressible Three-Dimensional Multiphase Particle-in-Cell Model for Dense Particle Flows". In: *Journal of Computational Physics* 170.2 (2001), pp. 523–549. DOI: 10.1006/jcph.2001.6747.

[143] P. Spalart and S. Allmaras. "A one-equation turbulence model for aerodynamic flows". In: *30th Aerospace Sciences Meeting and Exhibit*. AIAA, Paper 92-0439. American Institute of Aeronautics and Astronautics, 1992. DOI: 10.2514/6.1992-439.

[144] P. R. Spalart. "Strategies for turbulence modelling and simulations". In: *International Journal of Heat and Fluid Flow* 21.3 (2000), pp. 252–263. DOI: 10.1016/s0142-727x(00)00007-2.

[145] C. V. Srinivas, R. Venkatesan, R. Baskaran, V. Rajagopal, and B. Venkatraman. "Regional scale atmospheric dispersion simulation of accidental releases of radionuclides from Fukushima Dai-ichi reactor". In: *Atmospheric Environment* 61 (2012), pp. 66–84. DOI: 10.1016/j.atmosenv.2012.06.082.

[146] A. Stohl, P. Seibert, G. Wotawa, D. Arnold, J. F. Burkhart, S. Eckhardt, C. Tapia, A. Vargas, and T. J. Yasunari. "Xenon-133 and caesium-137 releases into the atmosphere from the Fukushima Dai-ichi nuclear power plant: determination of the source term, atmospheric dispersion, and deposition". In: *Atmospheric Chemistry and Physics Discussions* 11.10 (2011), pp. 28319–28394. DOI: 10.5194/acpd-11-28319-2011.

[147] G. Strang and G. J. Fix. *An Analysis of the Finite Element Method.* Vol. 55. 11. N. J. 1973. Prentice-Hall, Inc., 1975, pp. 696–697. DOI: 10.1002/zamm.19750551121.

[148] S. Sur, M. J. Koop, and D. K. Panda. "MPI and communication—High-performance and scalable MPI over InfiniBand with reduced memory usage". In: *Proceedings of the 2006 ACM/IEEE conference on Supercomputing - SC '06.* ACM Press, 2006. DOI: 10.1145/1188455.1188565.

[149] H. Tennekes and J. L. Lumley. *A first course in turbulence.* Cambridge, Massachusetts: The MIT Press, 1976.

[150] T. E. Tezduyar. "Stabilized Finite Element Formulations for Incompressible Flow Computations". In: *Advances in Applied Mechanics.* Elsevier, 1991, pp. 1–44. DOI: 10.1016/s0065-2156(08)70153-4.

[151] T. E. Tezduyar and Y. Osawa. "Finite element stabilization parameters computed from elementmatrices and vectors". In: *Computer Methods in Applied Mechanics and Engineering* 190 (2000), pp. 411–430.

[152] A. S. Thom and C. J. Apelt. "Field Computations in Engineering and Physics". In: *Geophysical Journal International* 5.3 (1961), pp. 264–264. DOI: 10.1111/j.1365-246x.1961.tb00437.x.

[153] T. G. Thomas and J. J. R. Williams. "Generating a wind environment for large eddy simulation of bluff body flows". In: *Journal of Wind Engineering and Industrial Aerodynamics* 82.1-3 (1999), pp. 189–208. DOI: 10.1016/s0167-6105(99)00037-9.

[154] A. V. Trofimova, A. E. Tejada-Martínez, K. E. Jansen, and R. T. Lahey. "Direct numerical simulation of turbulent channel flows using a stabilized finite element method". In: *Computers and Fluids* 38.4 (2009), pp. 924–938. DOI: 10.1016/j.compfluid.2008.10.003.

[155] Y.-H. Tseng, C. Meneveau, and M. B. Parlange. "Modeling Flow around Bluff Bodies and Predicting Urban Dispersion Using Large Eddy Simulation". In: *Environmental Science & Technology* 40.8 (2006), pp. 2653–2662. DOI: 10.1021/es051708m.

[156] G. de Vahl Davis and G. D. Mallinson. "An evaluation of upwind and central difference approximations by a study of recirculating flow". In: *Computers & Fluids* 4.1 (1976), pp. 29–43. DOI: 10.1016/0045-7930(76)90010-4.

[157] E. R. Van Driest. "On Turbulent Flow Near a Wall". In: *Journal of the Aeronautical Sciences* 23.11 (1956), pp. 1007–1011. DOI: 10.2514/8.3713.

[158] R. Vasaturo, I. Kalkman, B. Blocken, and P. van Wesemael. "Large eddy simulation of the neutral atmospheric boundary layer: performance evaluation of three inflow methods for terrains with different roughness". In: *Journal of Wind Engineering and Industrial Aerodynamics* 173 (2018), pp. 241–261. DOI: 10.1016/j.jweia.2017.11.025.

[159] B. Vreman, B. J. Geurts, N. G. Deen, J. A. M. Kuipers, and J. G. M. Kuerten. "Two- and Four-Way Coupled Euler–Lagrangian Large-Eddy Simulation of Turbulent Particle-Laden Channel Flow". In: *Flow, Turbulence and Combustion* 82.1 (2008), pp. 47–71. DOI: 10.1007/s10494-008-9173-z.

[160] VSC. *Breniac, Vlaams Supercomputing Center Tier-1 cluster*. https://vlaams-supercomputing-centrum-vscdocumentation.readthedocs-hosted.com/en/latest/leuven/tier1_hardware/breniac_hardware.html. Vlaams Supercomputing Center. 2019.

[161] T. Wiesner, M. Gee, A. Prokopenko, and J. Hu. *The MUELU tutorial*. SAND2014-18624 R.

[162] Wikimedia. *Composite map (14-25/04/2010) of Icelandinc volcanic ash cloud*. http://www.metoffice.gov.uk/aviation/vaac/vaacuk_vag.html. Accessed: 2020-06-19. 2010.

[163] Wikipedia. *Attacks of Nine September 2001*. https://en.wikipedia.org/wiki/September_11_attacks. 2001.

[164] Wikipedia. *Terrorism in Europe*. https://en.wikipedia.org/wiki/Islamic_terrorism_in_Europe. 2014.

[165] D. Wilcox. *Turbulence modeling for CFD*. DCW Industries, 2006.

[166] A. Wray and J. Hunt. "Algorithms for Classification of Turbulent". In: *Topological Fluid Mechanics: Proceedings of the IUTAM Symposium, Cambridge, UK, 13-18 August, 1989*. 0-521-38145-2. Cambridge University Press. 1990, pp. 95–104.

[167] D. L. Wright. "Numerical advection of moments of the particle size distribution in Eulerian models". In: *Journal of Aerosol Science* 38.3 (2007), pp. 352–369. DOI: 10.1016/j.jaerosci.2006.11.011.

[168] G. Yeoh. *Computational techniques for multiphase flows*. Oxford Burlington, MA: Butterworth-Heinemann, 2010.

[169] L. I. Zaichik, N. I. Drobyshevsky, A. S. Filippov, R. V. Mukin, and V. F. Strizhov. "A diffusion-inertia model for predicting dispersion and deposition of low-inertia particles in turbulent flows". In: *International Journal of Heat and Mass Transfer* 53.1-3 (2010), pp. 154–162. DOI: 10.1016/j.ijheatmasstransfer.2009.09.044.

[170] H. Zell. *Diagram of the layers within Earth's atmosphere*. 2017.

[171] O. C. Zienkiewicz. *The finite element method - Volume 1 - The Basis*. Oxford Boston: Butterworth-Heinemann, 2000.

[172] O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu. *The Finite Element Method: Its Basis and Fundamentals*. Elsevier LTD, Oxford, Oct. 1, 2013. 768 pp.