ASSESSMENT OF THE FIRE MODEL FIREFOAM AND DEVELOPMENT OF A USER'S GUIDE

SERGIO VARGAS-CÓRDOBA

.

International Master of Science in Fire Safety Engineering (M.Sc.)
The Department of Fire Protection Engineering
A. James Clark School of Engineering
University of Maryland

May 2018 – version 1.0

Sergio Vargas-Córdoba:
Assessment of the fire model FireFoam and development of a User's Guide
International Master of Science in Fire Safety Engineering (M.Sc.)
© May 2018

SUPERVISOR:
Prof. Arnaud Trouvé

PROMOTER:
Prof. Bart Merci

COMMITTEE:
Prof. José Luis Torero
dr. ir. Georgios Maragkos

LOCATION:
College Park, Maryland

ABSTRACT

FireFoam is a fire model developed by FM Global, based on the open-source code OpenFoam. There is a need to create a User's Guide for teaching purposes and spreading the use of the FireFoam. This thesis aims to be a benchmark for a FireFoam User's Guide that also provides a verification and validation framework for FireFoam. Three completed Cases and one in progress are presented in this manuscript.

The Lamb-Oseen vortex case tests the accuracy of *backward* and *euler* time integration schemes, the *backward* scheme (second order) exhibits better result than the *euler* scheme (first order). Furthermore, it shows that at least 20 grid cells span the diameter of Lamb-Oseen vortex core are required for getting a accurate representation of its variations.

The Taylor-Green Vortex reproduces the eddy break-up process as described on the Richardson *energy cascade* where kinetic energy is transfered from the large scales to the small ones, numerical dissipation arises due to the residual scales which allows to compare different time and spacial schemes through the evolution of the flow kinetic energy and enstrophy.

The Comte-Bellot and Corrsin isotropic turbulent decay flow reproduces the kinetic energy transfer from large to small scales, characteristic of a turbulent flow. From the evolution of the resolved turbulent kinetic energy, one can evaluate the influence of the models used for the residual (eddy viscosity models). The CBC shows to be a relevant test when comparing different eddy viscosity models.

The buoyant plumes cases are meaningful tests for mass conservation. These configurations allow to compare the influence of the number of outer corrector loops of the PIMPLE algorithm in mass conservation. Two configurations are proposed: a helium plume case for an inert scenario and a methane plume for a combustion configuration.

RESUMEN DOCUMENTAL

FireFoam es un modelo para incendios desarrollado por FM Global, basado en el código abierto OpenFoam. Existe una necesidad de crear una guía de usuario para propósitos de enseñanza y para extender el uso de FireFoam. Esta tesis pretende ser un punto de partida para una guía de usuario y a la vez proveer un marco para verificación y validación del modelo FireFoam. Tres casos terminados y uno en progreso son presentados en el presente manuscrito.

El vórtice de Lamb-Oseen evalúa la exactitud de los esquemas temporales *backward* y *euler*, el esquema *backward*(segundo orden) exhibe mejores resultados que el esquema *euler* (primer orden). Además, éste muestra que son necesarias 20 celdas a largo del vórtice de Lamb-Oseen para obtener una representación precisa de sus variaciones.

El vórtice de Taylor-Green reproduce el proceso de ruptura del remolino (eddies) como se describe en la *cascada de energía* de Richardson donde la energía cinética se transfiere de las escalas grandes a las pequeñas, la disipación numérica surge debido a las escalas residuales lo que permite comparar diferentes esquemas espaciales y de temporales a través de la evolución de la energía cinética del flujo y la enstrofía.

La caída de turbulencia isotrópica de Comte-Bellot y Corrsin reproduce la transferencia de energía cinética desde escalas grandes a escalas pequeñas, característica de un flujo turbulento. A partir de la evolución de la energía cinética turbulenta resuelta, se puede evaluar la influencia de los modelos utilizados para el término residual (modelos de viscosidad turbulenta). El CBC muestra ser una prueba relevante al comparar diferentes modelos de viscosidad turbulenta.

Los casos de penachos (plumas flotantes) son pruebas significativas para la conservación de masa. Estas configuraciones permiten comparar la influencia del número de bucles correctores externos del algoritmo PIMPLE en la conservación de masa. Se proponen dos configuraciones, un pluma de helio para un escenario inerte y una pluma de metano para una configuración con combustión.

"I am an old man now, and when I die and go to heaven there are twomatters on which I hope for enlightenment.One is quantum electrodynamics, and the other is the turbulent motion of fluids. And about the former I am rather optimistic."

— Horace Lamb

ACKNOWLEDGMENTS

Firstly, I would like to thank my advisor Prof. Arnaud Trouvé for his dedication, guidance and patience. I deeply appreciate Prof. Trouvé's exhaustive review and contributions in Chapter 2 to Chapter 4 (which at this moment we considered final version for the User's Guide). During our Monday weekly meetings Prof. Trouvé always found a way to teach me something related to either CFD, turbulence or fire modelling, something I really appreciate. I also want to thank Prof. Trouvé for organizing our Friday Group Meetings where some of the topics related to my thesis such as *temporal discretization* and *discretization of the convection term* where discussed.

Thanks also to all the CFD group members: : Le Van Minh, Cong Zhang, Rui Xu, Shiyun Wu, Rafiziana Binti MD Kasmani, Alexis Lhuillier-Marchand (Docteur Marchand) and Salman Verman for their support, patience, their effort in organizing small team-building activities and their dedication preparing the presentations on our Group Meetings.

I want to extend my gratitude to Minh for his endless patience and help during my first month in our group when I started learning about OpenFoam/FireFoam. To Alexis (PB) for his enormous help every time I met difficulties, for his enthusiasm about Linux, Python, and programming, but especially for all the lame french jokes. To Salman, this thesis would have never been done without all the *good talks*, *high-quality* advises, *smooth* guidance and *good luck!*.

Finally, I would like to thank my family and friends to whom I owe everything.

CONTENTS

LIST OF TABLES

LISTINGS

NOMENCLATURE

Abbreviations

rms	root-mean square
CBC	Comte–Bellot & Corsin
CFD	computational fluid dynamics
CFL	Courant–Friedrichs–Lewy
DFT	discrete Fourier transform
DNS	direct numeric simulation
FFT	fast Fourier transform
GS	resolved, Grid-Scale
LES	large eddy simulation
LOV	Lamb–Oseen Vortex
SGS	residual, subgrid scale
SI	International System of Units, *Système international (d'unités)*
TGV	Taylor–Green Vortex
TKE	turbulent kinetic energy

Greek Symbols

α_t	turbulent thermal diffusivity
Δx	grid resolution in x-direction
Δ	change of any changeable quantity
Δ	filter width
ϵ	dissipation rate
Γ	molecular diffusivity
γ	isentropic expansion factor
κ	wavenumber
κ_0	lowest wavenumber
κ_c	wavenumber at the Nyquist limit, $2\pi/\Delta$

μ	dynamic stress viscosity
ν	momentum diffusivity
ν_{SGS}	turbulent viscosity / eddy viscosity
ω	vorticity
ω_x	vorticity in x-direction
ω_y	vorticity in y-direction
ω_z	vorticity in z-direction
ϕ	mass flow
ϕ	passive scalar
$\Phi_{ij}(\kappa)$	velocity spectrum tensor
π	Archimedes' constant
ρ	volumetric mass density
ρ_∞	ambient mass density
τ_{TGV}	characteristic time for Taylor–Green vortex

Roman Symbols

A_s	Sutherland transport coefficient
c_∞	speed of sound in the medium
C_{max}	maximum Courant–Friedrichs–Lewy number for the advection criterion
D	molecular mass diffusivity
D_t	turbulent mass diffusivity
e	Euler–Mascheroni constant
$E(\kappa)$	Energy spectrum function/ Spectral-Energy function
g	gravitational constant
k	sub-grid turbulent kinetic energy dictionary in FireFoam
k_{GS}	resolved turbulent kinetic energy
k_{SGS}	sub-grid turbulent kinetic energy
k_{total}	total turbulent kinetic energy
L	characteristic lengthscale of the flow
L	circulation

L	length of side of cube in physical space
M	characteristic mesh spacing
p	pressure
p_∞	atmospheric pressure
p_{gauge}	positive pressure
p_{max}	maximum pressure
p_{min}	minimum pressure
Pr	Prandtl number
Pr_t	turbulent Prandtl number
Q	Q-criterion
r	radius
r_c	core radius of the Lamb-Oseen Vortex
$r_{c,0}$	core radius of the Lamb-Oseen Vortex at initial time
S_ϕ	Source term for the passive scalar ϕ
T	temperature
t	time
t_i	initial time
T_s	Sutherland temperature coefficient
u	velocity
u_0	air velocity in the CBC wind tunnel
u_x	velocity in x-direction
u_y	velocity in y-direction
u_z	velocity in z-direction
$u_{\theta,max,0}$	maximum velocity in angular coordinate at initial time
$u_{\theta,max}$	maximum velocity in angular coordinate
u_θ	velocity in angular coordinate
u_{coflow}	co-flow velocity
u_{max}	maximum velocity
V	Volume

x	absolute position in x-direction
x_c	absolute position of the core, LOV
y	absolute position in y-direction
Y_{He}	helium mass fraction
z	absolute position in z-direction
z_c	absolute position of the core, LOV

Additional Symbols

$\bar{E}(\kappa)$	energy-spectrum function of filtered velocity
$\dot{m}_{He}$	helium mass flow rate
$\hat{u}_i(\kappa, t)$	Fourier coefficient of velocity
$\mathcal{E}$	enstrophy $\omega^2/2$
$\mathcal{L}$	length of side of cube in physical space
$\mathcal{O}$	error's Order
$\mathcal{S}(\kappa)$	sphere in wavenumber space of radius κ
$\tilde{\Omega}_{ij}$	filtered rate-of-rotation tensor
$\tilde{S}_{ij}$	filtered rate-of-strain tensor
$\widehat{G}(\kappa)$	atenuation factor in the RJM filter

Part I

INTRODUCTION TO THE USER’S GUIDE

1 INTRODUCTION

1.1 OUTLINE

This thesis has been written in two sections this first part provides: a general overview for each case, their objectives and a brief description of the methodology. A complete description of the cases and their numerical set up will be found in section II. This structure allows to present the User's Guide as part of this thesis rather than a Appendix or an additional document.

1.2 INTRODUCTION AND OBJECTIVES

The FireFoam project was launched in 2008 when FM Global decided to create a fire model from an existing open-source toolbox, called OpenFoam, with the goal of provide a solver which includes physical models related to fluid mechanics, heat transfer and combustion.[2] FM Global sponsors FireFoam-related research activities, organizes the annual Open-Source Fire Modeling Workshop and also coordinates teamwork activities with different universities and research institutions around the world.[2][3] Although considerable effort has been done to promote this fire model, little educational material oriented to the newly FireFoam users exist. The OpenFOAM Foundation Ltd., OpenCFD Ltd and the OpenFoam community have created educational resources such as User Guide[4] and the OpenFoam wiki[5], both of which provide a set of handful tutorials that illustrates how to use tools available on OpenFoam and in some cases even the physics behind them, however there is no particular emphasis on FireFoam. This manuscript includes a set of cases that can be used for assessing different capabilities of the FireFoam, its main goal is to provide a starting point for a FireFoam user's guide.

1.2.1 *CASE 1*

The Lamb-Oseen Vortex is a canonical 2D flow for which an analytical solution of the Navier-Stokes equations is known.[1, 6]. Hence, it can be quite convenient while evaluating the performance of different schemes that discretize the Navier-Stokes equations in CFD packages, specially the errors that arise due to numerical dissipation since this vortex will decay faster than when only molecular viscosity is present.

The effects of the numerical dissipation will be evaluated mainly through the velocity and the vorticity, the latter provides condensed information that helps to analyze the flow, given that it represents the curl of the velocity.

1.2.1.1 *Time Schemes available on FireFoam*

OpenFoam provides different options for the discretization of the transient term: Euler (first order implicit), Backward (second order implicit) and Crank-Nicolson (second order implicit)[7].

The finite volume method is a powerful tool that allows a discrete description of the continuum. Under this method, the equations that model the transport phenomena (mass transfer, momentum, heat, etc..) are discretized and different schemes are used to model the discrete form of each term (transient, advection, laplacian and source term), see Equation 1.1. Since LES technique does take into account the transient features of the flow, the accuracy of the time schemes has a considerable influence on the solution, therefore, it has to be assessed.

$$\frac{\partial(\rho\phi)}{\partial t} + \nabla \cdot (\rho\phi u) = \nabla \cdot (\Gamma\nabla\phi) + S_\phi \tag{1.1}$$

There are different numerical methods for solving ordinary differential equations, the Euler difference methods are the most basic and popular. In the Euler Forward method also called just Euler a field is approximated from the current field state, the discretization of the derivative can be expressed as:

$$\frac{f(t+\Delta t) - f(t)}{\Delta t} = \frac{df(t)}{dt} + \frac{\Delta t}{2}\frac{d^2 f(t)}{dt^2} + \dots \tag{1.2}$$

$$\frac{f(t+\Delta t) - f(t)}{\Delta t} = \frac{df(t)}{dt} + \mathcal{O}(\Delta t) \tag{1.3}$$

On the other hand, the Euler backward method expresses a previous state in terms of the current state, the discretization of the derivative is shown on Equation 1.4

$$\frac{f(t) - f(t-\Delta t)}{\Delta t} = \frac{df(t)}{dt} - \frac{\Delta t}{2}\frac{d^2 f(t)}{dt^2} + \dots \tag{1.4}$$

$$\frac{f(t) - f(t-\Delta t)}{\Delta t} = \frac{df(t)}{dt} + \mathcal{O}(\Delta t) \tag{1.5}$$

As can be seen on Equation 1.2 and Equation 1.4 both schemes provides first order accuracy, however, OpenFoam tends to use implicit schemes. So the scheme named Euler on OpenFoam is actually the Euler Backward scheme, see Equation 1.6 [8].

$$\frac{\partial}{\partial t}(\phi) = \frac{\phi - \phi^0}{\Delta t} \tag{1.6}$$

Another classic method is the central difference, which is a two-steps scheme combination of the Euler forward and the Euler backward scheme, for this scheme the derivative is express in terms of $t + \Delta t$ and $t - \Delta t$ Equation 1.7.

$$\frac{f(t+\Delta t) - f(t-\Delta t)}{2\Delta t} = \frac{df(t)}{dt} - \frac{\Delta t^2}{3!}\frac{d^3 f(t)}{dt^3} + \dots \tag{1.7}$$

$$\frac{f(t+\Delta t)-f(t-\Delta t)}{2\Delta t}=\frac{df(t)}{dt}+\mathcal{O}(\Delta t^2) \tag{1.8}$$

This method provides second order accuracy, on OpenFoam is presented as the Crank-Nicolson scheme [8]. When using uniform time steps it takes to following form:

$$\frac{\partial}{\partial t}(\phi)=\frac{\phi-\phi^{00}}{2\Delta t} \tag{1.9}$$

Finally, the Adams-Moulton method in which the derivative is calculated in in terms of $t-2\Delta t$ and $t-\Delta t$, the functions are combined in a such way that the second derivative is eliminated Equation 1.10, so the method becomes second order accurate.

$$\frac{3f(t)-4f(t-\Delta t)+f(t-2\Delta t)}{2\Delta t}=\frac{df(t)}{dt}-\frac{2\Delta t^2}{3!}\frac{d^3f(t)}{dt^3}+\dots \tag{1.10}$$

$$\frac{3f(t)-4f(t-\Delta t)+f(t-2\Delta t)}{2\Delta t}=\frac{df(t)}{dt}+\mathcal{O}(\Delta t^2) \tag{1.11}$$

The OpenFoam documentation presents its backward scheme as Equation 1.12 which corresponds to the two-steps Adams-Moulton scheme [8].

$$\frac{\partial}{\partial t}(\phi)=\frac{1}{\Delta t}\left(\frac{3}{2}\phi-2\phi^{0}+\frac{1}{2}\phi^{00}\right) \tag{1.12}$$

Case 1 aims:

- To identify the minimum resolution necessary to accurately solve the Lamb-Oseen Vortex with the fire model FireFoam.
- To assess through the LOV the error of the euler and backward schemes available in FireFoam/OpenFoam libraries.

1.2.2 *CASE 2*

The Taylor–Green Vortex, as the Lamb–Oseen Vortex, can be treated as a canonical flow. It has been considered a high level 3D benchmark case on DNS simulations for the first five editions of the International Workshop on High Order CFD Methods.[9] [10] [11] Furthermore, it has been used in several studies for evaluating and simulating the convection of turbulent structures in the LES approach.[12][13][14] This flow follows the compressible (at low Mach number) and incompresible 3D Navier-Stokes equations with constant physical properties and it is widely used when evaluating temporal and spatial discretization schemes. CASE 3 focuses on the inviscid version of the Taylor–Green vortex corresponding to the configuration employed by Brachet *et al*[15] and Johnsen *et al*[12]. A detailed explanation of the chosen configuration is provided on Chapter 3.

In the CFD approach the partial differential equations i.e. Equation 1.1 are solved for finite volumes. Therefore, equations like Equation 1.1, which represents a general transport equation for the passive scalar are integrated over time and space.

$$\int_t^{t+\Delta t}\int_V \frac{\partial(\rho\phi)}{\partial t} dVdt + \int_t^{t+\Delta t}\int_V \nabla\cdot(\rho\phi u)dVdt = \int_t^{t+\Delta t}\int_V \nabla\cdot(\Gamma\nabla\phi)dVdt + \ldots \int_t^{t+\Delta t}\int_V S_\phi dVdt \tag{1.13}$$

As can be noticed from Equation 1.13 the advection term contains a volume integral of the divergence. Thus, by using the Gauss's theorem, the volume integral can be expressed in terms of a surface integral, see Equation 1.14

$$\int_V \nabla\cdot(\rho\phi u)dV = \oint_S dS\cdot(\rho\phi u) \tag{1.14}$$

The surface integral on Equation 1.14 is numerically approximated by summation of the fluxes over the faces, see Equation 1.15 [7]. Since the values of the different fields are known at the cell center, a procedure to obtain the values at the surfaces has to be done; the selected method is referred as *divergence scheme* or *advection scheme*. [4] [7] [16]

$$\oint_S dS\cdot(\rho\phi u) = \sum_f S\cdot(\rho u)_f\phi_f \tag{1.15}$$

It is difficult to find references about all the OpenFoam/FireFoam *divergence schemes* since some of them where developed specifically for OpenFoam; the best references are the code itself, the book by Moukalled[7] and Rusche's thesis[16]. Some of the *divergence schemes* classic schemes are linear upwind in which only the value of one cell is taken into account (assuming a 1D problem) based on your velocity direction, see Equation 1.16. Another scheme is the linear difference scheme which consists in a simple linear interpolation of the cell values to the face, see Equation 1.17. Being begin C and D adjacent cells in a 1D problem.

$$\phi_f = \begin{cases} \phi_C & \text{for} \quad (u\cdot n) > 0 \\ \phi_D & \text{for} \quad (u\cdot n) < 0 \end{cases} \tag{1.16}$$

$$\phi_f = f_x\phi_C + (1-f_x)\phi_D \quad \text{for} \quad f_x = \frac{|x_f - x_D|}{|x_f - x_D| + |x_f - x_C|} \tag{1.17}$$

Case 2 aims to:

- Develop a case, based on the Taylor-Green configuration, capable of providing a benchmark for evaluating time and spacial (advection) schemes in FireFoam.

1.2.3 *CASE 3*

In 1971 at the Johns Hopkins University, Genevieve Comte-Bellot and Stanley Corrsin managed to generate *isotropic turbulence*.[17] The isotropic turbulence consists in a flow of vortical motions decaying over the time, where the mean shear and the mean velocity are equal to zero, so the production of turbulent kinetic energy is equal to zero ($\mathcal{P} = 0$). [17] Besides, the aforementioned characteristics of this flow allow to get exhaustive statistical information by employing few measurements. The CBC *grid-generated 'isotropic' turbulence* paper provides a complete experimental data and description of the methodology employed both of which have made it a reference in the turbulence field. In DNS all the scales are solved: from the *integral* scales (where most of the energy is contain) to the *kolmogorov* scale (where dissipation takes place). On the other hand, in RANS the scales are not solved but rather modeled, since one is solving for the mean velocity field, where the fluctuating nature of the turbulent flows is no explicitly shown. The LES can be considered a middle point, given that large scales are solved but the the subgrid scales are modeled. Basically, the velocity field is decomposed in the filtered velocity field (resolved velocity field) and the subgrid scales (whose effects is modeled), then a set of equations is solved for filtered velocity field in which a *residual stress tensor* that models the effects of the subgrid scales is introduced in the momentum equations. Finally, the closure is done through a turbulent viscosity model.[18]

This isotropic turbulence decay case is considered a benchmark case for testing whether the turbulence model works properly or not.[19] Moreover, this case is quite susceptible to errors in the advective and diffusive terms[20] which implies a verification of the spacial discretization schemes. On the one hand, this case can be considered a verification since it test if the eddy viscosity model has been coded properly, on the other, it is considered a validation tests given that a comparison with experimental result is done.[19][20][21]

As mentioned before, in the finite-volume LES approach some of the energy is not captured by the grid which does not allow a direct comparison of the resolved turbulent kinetic energy with the full spectrum CBC experimental results, one alternative is comparing the simulations results with a filtered CBC spectrum. McDermortt says that up to the Nyquist limit $\bar{E}(\kappa) \sim \kappa^2$, between $\kappa_c < \kappa \leq \sqrt{2}\kappa_c$ some of the surface of the sphere is inside the Cartesian box and it is for the spheres of $\sqrt{2}\kappa_c < \kappa$ that $\bar{E}(\kappa) = 0$[21]. Thus, the relation between the spectrum and the filter spectrum is:

$$\bar{E}(\kappa) = \widehat{G}(\kappa)^2 E(\kappa) \tag{1.18}$$

Where the *atenuation factor* $\widehat{G}(\kappa)$ is equal to :

$$\widehat{G}(\kappa) = \begin{cases} 1 & \text{for} \quad \kappa \leq \kappa_c \\ \sqrt{3\frac{\kappa_c}{\kappa} - 2} & \text{for} \quad \kappa_c < \kappa \leq \sqrt{2}\kappa_c \\ 0 & \text{for} \quad \sqrt{2}\kappa_c < \kappa \end{cases} \tag{1.19}$$

Another method is performed by initializing the sub-grid kinetic energy field, so one the simulation data can be compared with the CBC full spectrum. Both approaches are employed in CASE 3.

Finally, the CBC data provides an energy spectrum for 3 non-dimensional times, the first one corresponds to the initialization of the velocity field. Thus, the other two can be used for assessing the effects of the subgrid model in the energy distribution of Energy-Spectrum function. The *Energy-Spectrum function* can be obtained through a Fourier analysis of the velocities field. Pope [18] provides a good explanation of the procedure to follow. Essentially, the energy spectrum function is obtained from a simplification of the velocity spectrum tensor $\Phi_{ij}(\kappa)$, while the latter is a second-order tensor, the former represents only half of its trace ($\frac{1}{2}\Phi_{ii}(\kappa)$). Additionally, information related to the Fourier modes direction is eliminated by integrating over the surfaces of all spheres of radius $\mathcal{S}(\kappa)$[18]. In other words, the energy-spectrum is the density of spectral energy in surfaces $\mathcal{S}(\kappa)$[19], see Equation 1.20 :

$$E(\kappa) = \oint \frac{1}{2}\Phi_{ii}(\kappa) d\mathcal{S}(\kappa) \tag{1.20}$$

McDermott [19] mentions that in practice this integral is solved numerically. Firstly, by using a 3D Fast Fourier transform that generates $\hat{u}_i(\kappa, t)$, then calculating it as Equation 1.21:

$$E(\kappa, t) = \frac{\mathcal{L}}{2\pi} \sum_{\kappa=\sqrt{\kappa_i \kappa_i}} \frac{1}{2} \hat{u}_i^*(\kappa, t) \cdot \hat{u}_i(\kappa, t) \tag{1.21}$$

For this case two *eddy viscosity* models will be employed. Both of them solve a transport equation for the subgrid turbulent kinetic energy, see Equation 1.22[8]. In the *kEqn* model based on Yoshizawa model [22] the coefficients C_e and C_k are constant, meanwhile the *dynamicKEqn* follows Kim *et al* formulation [23] thus, C_e and C_k are dynamic calculated from the velocity field[8].

$$\frac{D}{Dt}(\rho k_{sgs}) = \nabla \cdot (\rho D_{k_{sgs}} \nabla k_{sgs}) + \rho G - \frac{2}{3}\rho k_{sgs} \cdot u - \frac{C_e \rho k_{sgs}{}^{1.5}}{\Delta} + S_{k_{sgs}} \tag{1.22}$$

A detailed description of the numerical set up as well as the initial velocity field for this case is given in Chapter 4

Case 3 aims to:

Provide a test case for assessing and comparing the performance of eddy viscosity models.

1.2.4 CASE 4

Finally, CASE 4 consists in 2 different configurations that evaluate the conservation of mass. In OpenFoam/FireFoam one is solving the Navier-Stoke equations, thus an iterative algorithm for solving the coupled momentum-pressure equations is needed.

OpenFoam/FireFoam provides three algorithms: Pressure-implicit split operator (PISO): generally used for transient problems. Semi-implicit method for pressure-linked equations (SIMPLE): generally used for steady-state problems. PIMPLE: a combination of PISO and SIMPLE.

Broadly speaking, the main difference between the algorithms is how they loop over the equations, since all of them aim to enforce mass conservation, by solving a pressure equation. [4] A more detailed explanation of the algorithms is provided in Moukalled *et al*[7] and by Holzmann.[24]

- nCorrectors: the number of times the algorithm solves the pressure equation and momentum corrector in each step (number of inner loops, pressure correction)[4]
- nOuterCorrectors: number of times the algorithm loops over all the equations(pressure-momentum correction).[4]

Case 4 focuses on PIMPLE, an example of PIMPLE algorithm set with 1 nCorrectors and 2 nOuterCorrectors will perform the following operations.[24][25]

- Build the momentum matrix
- Calculate the pressure matrix from the momentum $\nabla^2 p = f(u, \nabla p)$
- Calculate the p_{new1}
- Correct the velocity field with $p_{new1} \rightarrow u_1$
- Start second outer-Loop with p_{new1} and u_1
- Build the momentum matrix with p_{new1} and u_1
- Calculate the pressure matrix from the momentum $\nabla^2 p_{new1} = f(u, \nabla p_{new1})$
- Calculate the p_{new2}
- Correct the velocity field $p_{new2} \rightarrow u_2$

Case 4 consists in two different configurations: an inert flow and a combustion one. In the former the conservation of mass is tested through the time integration of the mass flow rate of the inert specie (by using the mass fraction flow of the inert specie), in the latter by analyzing the mass flow rate of the mixture (mixture fraction), a more detailed explanation is provided in Chapter 5.

This case aims to:

- Develop the basis for a case that tests the mass conservation under different settings of the PIMPLE algorithm.

1.3 GOVERNING EQUATIONS

FireFoam solves the instantaneous continuity for density, (*rhoEqn.H*), momentum equation (*UEqn.H*) and Energy equation (*YEEqn.H*), the way these equations are structured and written varies from version to version. The FireFoam version used for this thesis is firefoam-dev available at the Github repository fireFoam-dev `https://github.com/fireFoam-dev/fireFoam-dev`.

1.4 CASES' STRUCTURE IN FIREFOAM

The general FireFoam/OpenFoam structure consists in a set of three directories:

- 0: corresponds to the time equal to zero, it is considered as a *Time* directory; it is in the files contained in this folder where the user specifies all the initial and boundary conditions for the different fields.
- constant: provides the constants that can be considered as physical properties i.e. gravity (g dictionary), the geometry and the mesh(polyMesh directory), combustion properties, reactions, turbulence properties, etc...
- system: the system directory contains all the necessary parameters related to the solution procedure

A better understanding of the OpenFoam/FireFoam structure is obtained by reading the Part ii (Cases–User's Guide)

1.5 COMPUTERS' DESCRIPTION

All the simulations were run in the The Deepthought2 High-Performance Computing (HPC) cluster of the University of Maryland at College Park. Most of the nodes are dual Intel Ivy Bridge E5-2680v2, memory is DDR3 at 1866 MHz.[26]

Table 1.1: Computational Power on the Deepthought2 cluster[26]

DESCRIPTION	PROCESSOR	NUMBER OF NODES	CORES/NODE
C8220	2.8 GHz	444	20
C8220X	2.8 GHz	40	20
Poweredge R820	2.2 GHz	4	40

Part II

USER'S GUIDE

2 CASE 1: LAMB-OSEEN VORTEX

2.1 CASE 1 DESCRIPTION

The Lamb-Oseen vortex case is a simple two-dimensional laminar flow configuration that features convective motion and viscous decay and that is often used in Computational Fluid Dynamics (CFD) to evaluate the ability of a particular software to correctly solve the Navier-Stokes equations. Because a turbulent flow can be viewed as a population of vortices of different sizes and rotational velocities, the configuration is also relevant to the general problem of simulating turbulent flows. In addition, the Lamb-Oseen vortex problem has a well-known analytical solution [1, 6, 27, 28] and the case thereby provides a valuable quantification of the error associated with a particular numerical solution. The error is typically studied as a function of choices made in the spatial and/or temporal discretization of the problem, or as a function of the numerical boundary schemes adopted at the edges of the computational domain.

The solution of the Lamb-Oseen vortex is typically first presented in a polar coordinate system:

$$u_\theta = u_{\theta,max}\left(\frac{r}{r_c}\right)\exp\left(\frac{1}{2}\left(1-\frac{r^2}{{r_c}^2}\right)\right) \tag{2.1}$$

where u_θ is the circumferential flow velocity, r the radial distance from the center of the vortex, and where r_c and $u_{\theta,max}$ are the characteristic size and velocity of the vortex at a particular time. The characteristic size r_c is defined as the radial distance (from the center of the vortex) where u_θ takes its maximum value and the characteristic velocity $u_{\theta,max}$ is that particular maximum value. Both r_c and u_θ depend on time t as follows:

$$\begin{aligned} r_c &= (r_{c,0} + 2\nu t)^{(1/2)} \\ u_{\theta,max} &= \frac{u_{\theta,max,0}}{\left(1+\frac{2\nu t}{{r_{c,0}}^2}\right)^{(3/2)}} \end{aligned} \tag{2.2}$$

where $r_{c,0}$ and $u_{\theta,max,0}$ are the prescribed values of r_c and $u_{\theta,max}$ at initial time $t = 0$, and where ν is the kinematic viscosity of the fluid. Equation (2.2) describes the increase in size and decrease in velocity that characterize the flow structure as the vortex experiences viscous decay. Note that in the Lamb-Oseen vortex solution, the radial velocity u_r is 0.

The previous solution can also be presented in a rectangular Cartesian coordinate system:

$$\begin{aligned} u_x &= u_{\theta,max}\left(\frac{z-z_c}{r_c}\right)\exp\left(\frac{1}{2}\left(1-\frac{(x-x_c)^2+(z-z_c)^2}{{r_c}^2}\right)\right) \\ u_z &= -u_{\theta,max}\left(\frac{x-x_c}{r_c}\right)\exp\left(\frac{1}{2}\left(1-\frac{(x-x_c)^2+(z-z_c)^2}{{r_c}^2}\right)\right) \end{aligned} \tag{2.3}$$

where u_x and u_z are the components of velocity in the x- and z-direction, respectively, and where (x_c, z_c) are the coordinates of the center of the vortex. We have assumed in Eq. (2.3) that these coordinates are constant, $(x_c, z_c) = (x_{c,0}, z_{c,0})$, where $(x_{c,0}, z_{c,0})$ are the values of (x_c, z_c) at $t = 0$. Note also that: if $u_{\theta,max,0} > 0$, the solution above corresponds to a clockwise-rotating vortex; and if $u_{\theta,max,0} < 0$, the solution corresponds to a counterclockwise-rotating vortex.

The previous solution can easily be extended to the case in which the vortex is convected at constant and uniform external velocity. For instance, for an external flow at constant and uniform velocity u_{coflow} in the x-direction, we have:

$$\begin{aligned} u_x &= u_{\theta,max}\left(\frac{z-z_c}{r_c}\right)\exp\left(\frac{1}{2}\left(1-\frac{(x-x_c)^2+(z-z_c)^2}{{r_c}^2}\right)\right)+u_{coflow} \\ u_z &= -u_{\theta,max}\left(\frac{x-x_c}{r_c}\right)\exp\left(\frac{1}{2}\left(1-\frac{(x-x_c)^2+(z-z_c)^2}{{r_c}^2}\right)\right) \end{aligned} \tag{2.4}$$

where the center of the vortex is now convected by the external flow and (x_c, z_c) are time-dependent functions: $(x_c, z_c) = ((x_{c,0} + u_{coflow} \times t), z_{c,0})$.

The corresponding expressions for the y-component of vorticity and for the pressure are:

$$\begin{aligned} \omega_y &= \frac{\partial u_x}{\partial z} - \frac{\partial u_z}{\partial x} \\ &= \frac{2u_{\theta,max}}{r_c}\left(1-\frac{(x-x_c)^2+(z-z_c)^2}{2{r_c}^2}\right)\exp\left(\frac{1}{2}\left(1-\frac{(x-x_c)^2+(z-z_c)^2}{{r_c}^2}\right)\right) \\ p &= p_\infty - \frac{\rho_\infty u_{\theta,max}^2}{2}\exp\left(1-\frac{(x-x_c)^2+(z-z_c)^2}{{r_c}^2}\right) \end{aligned} \tag{2.5}$$

where p_∞ is the ambient pressure and ρ_∞ the ambient mass density (both assumed constant and uniform). Note that ω_y takes a maximum value at the center of the vortex and that this value is time-dependent and gives a measure of the strength of the vortex. Similarly, p takes a minimum value at the center of the vortex and this value gives another measure of the strength of the vortex.

2.2 DIRECTORY STRUCTURE

FireFOAM adopts the basic structure of an OpenFOAM working directory that consists of three folders: the *constant/* directory that contains files specifying physical parameters of the simulation; the *system/* directory that contains files specifying data management and numerical parameters of the simulation; and the *0/* directory that contains files specifying the initial and boundary conditions of the simulation. In addition, the *constant/* directory has a subdirectory *constant/polyMesh* that contains the *blockMeshDict* dictionary that specifies the computational domain and the computational mesh (note that an acceptable alternative in OpenFOAM is to install the *blockMeshDict* dictionary in the *system/* directory).

For the Lamb-Oseen vortex case, the main relevant features of the *constant/* directory are as follows. The *combustionProperties* dictionary specifies that there is no combustion; the *g* dictionary specifies that there is no gravity; the *pyrolysisZones* dictionary specifies that there is no pyrolysis; the *thermophysicalProperties* dictionary specifies that molecular transport is described using Sutherland's law; the *turbulenceProperties* dictionary specifies that the flow is laminar (the governing equations are solved without a turbulence model). In addition, the *thermo.compressibleGas* dictionary is a file that (among other things) gives, for each chemical species, the values of the model coefficients *As* and *Ts* required in Sutherland's law.[29][4] Note that this file has been artificially modified to impose the same values of *As* and *Ts* for O_2 and N_2 (the only species used in the present case), which means that the viscosities of O_2 and N_2 are identical and that the kinematic viscosity of air (treated as a mixture of O_2 and N_2) can be simply calculated as follows:

$$\nu = \frac{1}{\rho} \cdot \frac{A_s\sqrt{T}}{1 + (T_s/T)} \tag{2.6}$$

where ρ and T are the mass density and temperature of ambient air, $\rho = \rho_\infty$ and $T = T_\infty$. This expression is used later in comparisons between numerical results and the analytical solution presented in Section 2.1.

Furthermore, for the Lamb-Oseen vortex case, the main relevant features of the *system/* directory are as follows. The *decomposeParDict* dictionary specifies the domain decomposition scheme adopted in parallel computing mode (in the present case, we use 4 or 16 processors); the *fvSchemes* dictionary specifies the schemes used for temporal and spatial discretization (in the present case, we consider two temporal schemes: *backward* and *Euler*); the *fvSolution* dictionary specifies the equation solvers and the tolerances used for convergence of the iterative schemes. In addition, the *controlDict* dictionary specifies that: the duration of the simulation is 3.0 s (*endTime 3.0*); the simulation runs with an adjustable time step (*adjustTimeStep yes*) controlled by a Courant-Friedrichs-Lewy (CFL) number of 0.5 (*maxCo 0.5*); the output files are written every 0.05 s (*writeInterval 0.05* in the *writeObjects1* section) and in compressed format (*writeCompression compressed*; this instruction generates files with a *.gz* extension); and the output files store the velocity and pressure fields (*U* and *p* in the *writeObjects1* section). The *initLOVDict* dictionary specifies the initial conditions of the velocity field: a Lamb-Oseen vortex located at $(x_c, z_c) = (0, 0)$ with $r_{c,0} = 0.005$ m, $u_{\theta,max,0} = 0.5$ m/s and $u_{coflow} = 0.1$ m/s. The *sampleDict* dictionary specifies the names of the variables to be later analyzed and plotted (*Uz* and *vorticityy* in the *fields* section, where *Uz* designates the z-component of flow velocity, u_z, and *vorticityy* designates the y-component of vorticity, ω_y); it also specifies the coordinates of the line along which profiles will be plotted (*start (-0.05 0 0)* and *end (0.05 0 0)* in the *sets* section).

Finally, for the Lamb-Oseen vortex case, the main relevant features of the *0/* directory are as follows. The *O2* and *N2* files specify the initial (and constant) composition of air (*internalField uniform 0.23301* and *internalField uniform 0.76699*). The *p* and *p_rgh* files specify the initial ambient pressure (*internalField uniform 101325*); the pressure will vary in the simulations as described in Section 2.1. The *T* file specifies the initial (and constant) ambient air temperature (internalField uniform 236.315); note that this value has been selected so that the kinematic viscosity is $\nu = 10^{-5}$ m^2/s (see Eq. (2.6)). The *U* file specifies a uniform initial value of the flow velocity vector (*internalField* *uniform (0 0 0)*) and will be updated after calling the *initLOV* pre-processor (see Section 2.3.2); the velocity will then vary in the simulations as described in Section 2.1. Furthermore, the files in the *0/* directory also contain information on boundary conditions. In the present case, the simulation are two-dimensional and only require boundary conditions at the west, east, south and north boundaries: the west and east boundaries along the x-direction are periodic (called *cyclic* in OpenFOAM); the south and north boundaries along the z-direction correspond to zero-gradient conditions (*type zeroGradient*) for the mass, velocity and temperature variables (*O2*, *N2*, *T* and *U* files) and correspond to a fixed total pressure (*type totalPressure* and *p0 uniform 101325*) for pressure variables (*p* and *p_rgh* files). Numerical tests have shown that for this particular case, the total pressure boundary conditions allow accurate simulations of the pressure field and that other choices (for instance *type zeroGradient*) lead to incorrect results.

2.3 PRE-PROCESSING

2.3.1 *Definition of the Computational Domain and Mesh Generation*

The first step in preparing a simulation of the Lamb-Oseen vortex case consists in generating the computational mesh. The parameters of the mesh are specified in the *blockMeshDict* dictionary (located in the *constant/polyMesh* subdirectory). The main features of the *blockMeshDict* file are as follows. The computational domain is a simple rectangular cuboid of size $(0.1 \times 0.001 \times 0.1)$ m^3 in an (x, y, z) rectangular Cartesian system (see lines 22-29 in Listing 2.1). The mesh is uniform and uses 101 cells in the x- and z-directions and 1 cell in the y-direction (line 45 in Listing 2.1); the spatial resolution in the x- and z-directions is equal to $(0.1/101) \approx 1$ mm; the specification of 1 cell in the y-direction makes the simulation two-dimensional. In the following, the spatial resolution is varied and uses (101×101), (201×201) or (401×401) cells, which corresponds to $(r_{c,0}/\Delta x) \approx 5$, 10 or 20.

Listing 2.1: BlockMeshDict for the Lamb-Oseen vortex case

```
1  /*--------------------------------*- C++ -*----------------------------------*\
2  | =========                 |                                                 |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4  |  \\    /   O peration     | Version:  2.2.0                                 |
5  |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6  |    \\/     M anipulation  |                                                 |
7  \*---------------------------------------------------------------------------*/
8  FoamFile
9  {
10     version     2.0;
11     format      ascii;
12     class       dictionary;
13     object      blockMeshDict;
14 }
15 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
16
17 convertToMeters 1;
18
19 vertices
```

```
(

    (-0.05     -0.0005    -0.05)  //0
    ( 0.05     -0.0005    -0.05)  //1
    ( 0.05      0.0005    -0.05)  //2
    (-0.05      0.0005    -0.05)  //3
    (-0.05     -0.0005     0.05)  //4
    ( 0.05     -0.0005     0.05)  //5
    ( 0.05      0.0005     0.05)  //6
    (-0.05      0.0005     0.05)  //7

);

//          7 ---------- 6
//         /|           /|
//        / |          / |    z
//       4 ---------- 5  |    ^
//       |  |         |  |    |    y
//       |  3 --------|-- 2   |   /
//       | /          | /     |  /
//       |/           |/      | /
//       0 ---------- 1       ------> x

blocks
(
    hex (0 1 2 3 4 5 6 7) (101 1 101) simpleGrading (1 1 1)

);

edges
(
);

boundary
(
    north
    {
        type wall;
        faces
        (
                (4 5 6 7)
        );
    }

    south
    {
        type wall;
        faces
        (
                (0 3 2 1)
        );
    }

    west
    {
        type cyclic;
        neighbourPatch east;
        faces
        (
                (0 4 7 3)
        );
    }

    east
    {
        type cyclic;
        neighbourPatch west;
        faces
        (
                (1 2 6 5)
        );
    }
    frontBack
    {
        type empty;
        faces
        (
                (0 1 5 4)
                (2 3 7 6)
        );
    }
);

// ************************************************************************* //
```

We call *CASE1* the working directory for the Lamb-Oseen vortex case. Generate the mesh by going to this directory and by typing the following command in the terminal window:

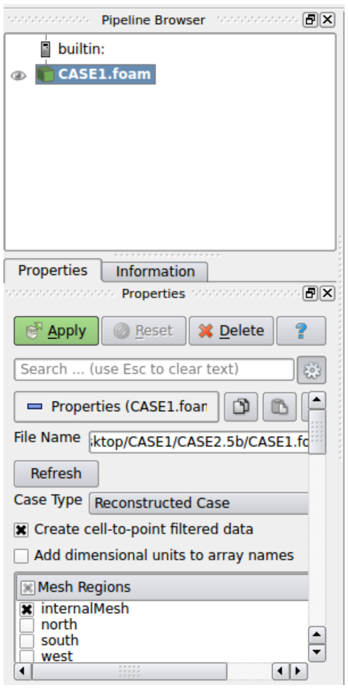
```
user@hostname/CASE1: blockMesh
```

Check the mesh before running the simulation by creating a dummy file in the case directory with the extension .foam (here called *case1.foam*) and by calling ParaView or paraFoam in the terminal window:

```
user@hostname/CASE1: touch case1.foam
```

```
user@hostname/CASE1: paraview
```

In ParaView, click *Apply* in the Pipeline Browser (see Figure 2.1) and select *Solid Color* and *Wireframe* (and also the $x - z$ view) in the Toolbar (see Figure 2.2). One can now visualize the mesh (Figure 2.3).



File Name ;ktop/CASE1/CASE2.5b/CASE1.fo

Case Type Reconstructed Case

- [x] Create cell-to-point filtered data
- [ ] Add dimensional units to array names

Mesh Regions

- [x] internalMesh
- [ ] north
- [ ] south
- [ ] west

Figure 2.1: Pipeline Browser in ParaView

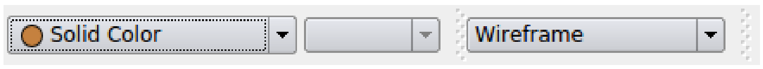


Figure 2.2: Toolbar in ParaView

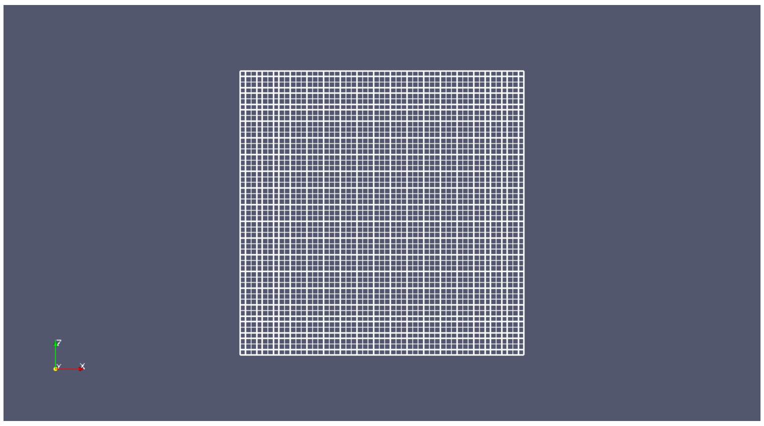


Figure 2.3: Computational mesh as seen in ParaView in the Lamb-Oseen vortex case (51×51 resolution)

2.3.2 *Initial conditions*

As mentioned previously, the *initLOVDict* dictionary specifies the initial conditions of the velocity field. First, one needs to install the *initLOV* pre-processing utility: go to any directory of choice (for instance the *OpenFOAM* installation directory); copy the archive file *initLOV.tar* to that directory; extract the archive file by typing *"tar -xvf initLOV.tar"*; go to the newly created *initLOV* directory by typing *"cd initLOV"*; and then compile the utility by typing *"wmake"*. Generate the initial velocity field by going to the *CASE1* directory and by typing the following command in the terminal window:

```
user@hostname/CASE1: initLOV
```

The *U* file in the *0/* directory has now been updated with an initial field that corresponds to a Lamb-Oseen vortex flow (see Eq. (2.4) and lines 89-90 in Listing 2.2). One can now visualize this initial field in ParaView (Figure 2.4).

Listing 2.2: initLOV program for the Lamb-Oseen vortex case

```
1  /*---------------------------------------------------------------------------*\
2    =========                 |
3    \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
4     \\    /   O peration     |
5      \\  /    A nd           | Copyright (C) 1991-2005 OpenCFD Ltd.
6       \\/     M anipulation  |
7  -------------------------------------------------------------------------------
8  License
9      This file is part of OpenFOAM.
10
11     OpenFOAM is free software; you can redistribute it and/or modify it
12     under the terms of the GNU General Public License as published by the
13     Free Software Foundation; either version 2 of the License, or (at your
14     option) any later version.
15
16     OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
17     ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
18     FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
19     for more details.
20
21     You should have received a copy of the GNU General Public License
22     along with OpenFOAM; if not, write to the Free Software Foundation,
23     Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
24
25 Application
26     initLOV -- written by Salman Verma
27
28 Description
29     initialize the Lamb-Oseen vortex.
30
31     Reads in initLOVDict.
32
33 \*---------------------------------------------------------------------------*/
34
35
36 #include "fvCFD.H"
37
38 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
39
40 int main(int argc, char *argv[])
41 {
42 #   include "setRootCase.H"
43
#   include "createTime.H"
#   include "createMesh.H"

    IOdictionary initLOVDict
    (
        IOobject
        (
            "initLOVDict",
            runTime.system(),
            mesh,
            IOobject::MUST_READ,
            IOobject::NO_WRITE
        )
    );

    Info<< "Reading the parameters from the initLOVDict file" << endl;
    const scalar Xo(readScalar(initLOVDict.lookup("Xo")));
    const scalar Zo(readScalar(initLOVDict.lookup("Zo")));
    const scalar Umaxo(readScalar(initLOVDict.lookup("Umaxo")));
    const scalar Ro(readScalar(initLOVDict.lookup("Ro")));
    const scalar Ucoflow(readScalar(initLOVDict.lookup("Ucoflow")));

    Info<< "Vortex center at t=0 [s]          = (" << Xo << ",0," << Zo << ") [
        m]" << nl
        << "Peak vortex velocity at t=0 [s]   = " << Umaxo << " [m/s]" << nl
        << "Vortex radius at t=0 [s]          = " << Ro << " [m]" << nl
        << "Co-flow velocity                  = " << Ucoflow << " [m/s]" << nl
        << endl;

    IOobject Uheader
    (
        "U",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ
    );
    Info<< "Reading U" << endl;
    volVectorField U(Uheader, mesh);

    scalar xx;
    scalar zz;

    forAll(mesh.C(), celli)
    {
            xx = mesh.C()[celli].x();
            zz = mesh.C()[celli].z();
            U[celli].component(0) = Ucoflow+(Umaxo*(zz-Zo)/Ro)*Foam::exp(0.5-(
                sqr(xx-Xo)+sqr(zz-Zo))/(2*sqr(Ro)));
            U[celli].component(2) = -(Umaxo*(xx-Xo)/Ro)*Foam::exp(0.5-(sqr(xx-
                Xo)+sqr(zz-Zo))/(2*sqr(Ro)));
    }

    Info<< "Writing modified U field with the Lamb-Oseen vortex to " << runTime
         .timeName() << endl;
   U.write();

    Info<< endl;

    return(0);
}


// ************************************************************************* //
```

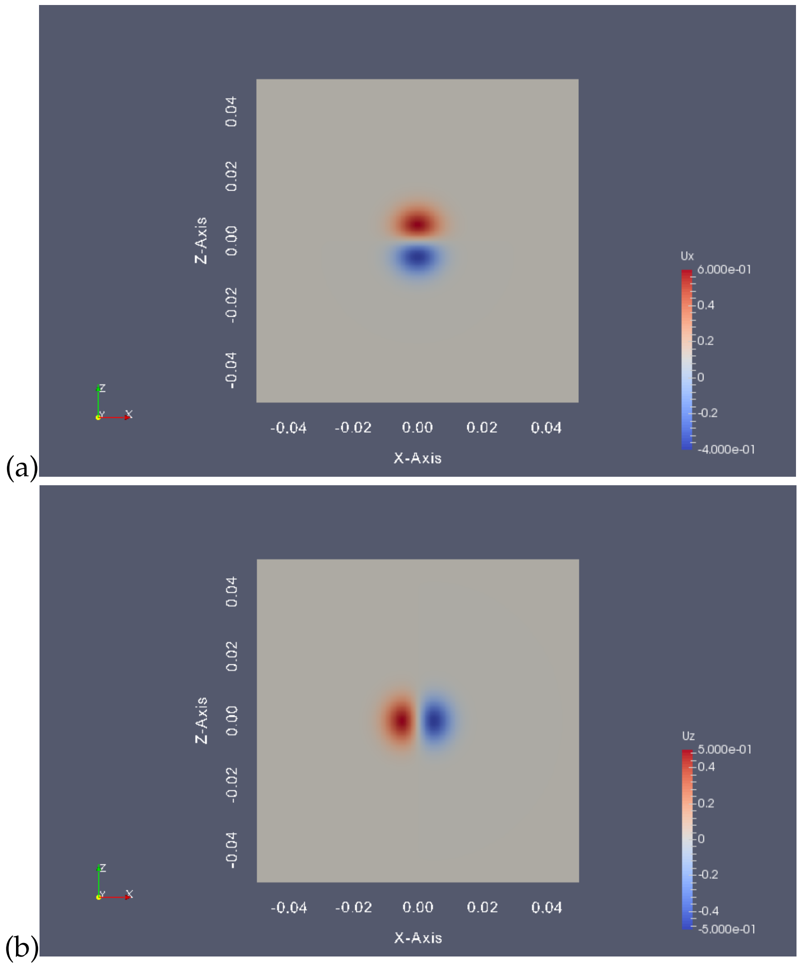


Figure 2.4: Initial conditions for flow velocity in the Lamb-Oseen vortex case: (a) u_x; (b) u_z. The case corresponds to a clockwise-rotating vortex with an initial size of $(2 \times r_{c,0}) = 0.01$ m.

2.4 RUNNING THE SIMULATION

The Lamb-Oseen vortex case is run in parallel computing mode. The *decomposePar-Dict* dictionary in the *system/* directory specifies the number of processors to be used (*numberOfSubdomains 4* or *numberOfSubdomains 16*) and the structure of the domain decomposition (*n (2 1 2)* for a decomposition of the *x*- and *z*-directions into 2 blocks each; or *n (4 1 4)* for a decomposition of the *x*- and *z*-directions into 4 blocks each). Decompose the computational domain and assign a portion of the mesh to each processor by going to the *CASE1* directory and by typing the following command in the terminal window:

```
user@hostname/CASE1: decomposePar
```

Then run the simulation by typing:

```
user@hostname/CASE1: mpirun -np 4 fireFoam -parallel > log.CASE1
```

where we have assumed a run with 4 processors and where the file named *log.CASE1* contains the messages generated by FireFOAM during the simulation. FireFOAM creates a separate output directory for each processor (here 4 directories called *processor0*, *processor1*, *processor2* and *processor3*) and in each of those, generates time-stamped folders every 0.05 s. These folders contain *U* and *p* files that store the portion of the velocity and pressure fields managed by the processor and at the time indicated by the name of the folder.

When the simulation is completed, merge the output directories, folders and files for analysis by typing:

```
user@hostname/CASE1: reconstructPar
```

The *reconstructPar* functionality creates new time-stamped folders in the *CASE1* folder (written every 0.05 s) that now contain *U* and *p* files that store the full velocity and pressure fields.

2.5 POST-PROCESSING

2.5.1 *Gauge Pressure*

The pressure variable in FireFOAM is the absolute pressure and it contains information on the thermodynamic, hydrostatic and dynamic variations that control the pressure field. It is often more insightful to plot the gauge pressure defined as the deviation of pressure from its ambient (thermodynamic) value (here $p_\infty = 101325$ Pa), $p_{gauge} = (p - p_\infty)$. For this purpose, the Lamb-Oseen vortex case uses the *pGauge* functionality. First, one needs to install the *pGauge* post-processing utility: go to any directory of choice (for instance the *OpenFOAM* installation directory); copy the archive file *pGauge.tar* to that directory; extract the archive file by typing *"tar -xvf pGauge.tar"*; go to the newly created *pGauge* directory by typing *"cd pGauge"*; and then compile the utility by typing *"wmake"*. Finally, generate the gauge pressure fields by going to the *CASE1* directory and by typing the following command in the terminal window:

```
user@hostname/CASE1: pGauge
```

The *pGauge* functionality generates new *pGauge* files inside the time-stamped folders previously created by *reconstructPar*.

2.5.2 *Time Evolution of Minimum and Maximum Values*

The Lamb-Oseen vortex case uses the *postProcess* functionality of OpenFOAM to analyze the time evolution of the minimum and maximum values of different variables of interest. First, generate the x-, y and z components of flow velocity by going to the *CASE1* directory and by typing the following command in the terminal window:

```
user@hostname/CASE1: postProcess -func "components(U)"
```

This command generates new *Ux*, *Uy* and *Uz* files (corresponding to u_x, u_y and u_z, respectively; note that $u_y = 0$ here) inside the time-stamped folders previously created by *reconstructPar*. Next, calculate the vorticity (vector) fields and then generate the x-, y- and z-components of vorticity by typing:

```
user@hostname/CASE1: postProcess -func vorticity
```

```
user@hostname/CASE1: postProcess -func "components(vorticity)"
```

This commands generate new *vorticityx*, *vorticityy* and *vorticityz* files (corresponding to ω_x, ω_y and ω_z, respectively; note that $\omega_x = \omega_z = 0$ here) inside the time-stamped folders previously created by *reconstructPar*. And finally generate the minimum and maximum values of *Ux*, *Uz*, *pGauge* and *vorticityy* by typing:

```
user@hostname/CASE1: postProcess -func ’cellMin(Ux)’
```

```
user@hostname/CASE1: postProcess -func ’cellMax(Ux)’
```

```
user@hostname/CASE1: postProcess -func ’cellMin(Uz)’
```

```
user@hostname/CASE1: postProcess -func ’cellMax(Uz)’
```

```
user@hostname/CASE1: postProcess -func ’cellMin(pGauge)’
```

```
user@hostname/CASE1: postProcess -func ’cellMax(pGauge)’
```

```
user@hostname/CASE1: postProcess -func ’cellMin(vorticityy)’
```

```
user@hostname/CASE1: postProcess -func ’cellMax(vorticityy)’
```

These commands create a new directory called *postProcessing* with 8 folders (called *cellMin(Ux)*, *cellMax(Ux)*, *etc*) and 8 files called *volFieldValue.dat* that store the minimum or maximum value of u_x, u_z, p_{gauge} and ω_y as a function of time.

2.5.3 *Spatial Profiles*

The Lamb-Oseen vortex case also uses the *postProcess* functionality of OpenFOAM to generate spatial profiles of different variables of interest at discrete times. As mentioned previously, the *sampleDict* dictionary specifies the names of the variables to be analyzed and plotted (*Uz* and *vorticityy*, *i.e.*, u_z and ω_y) and the coordinates of the line along which profiles will be plotted ($z = 0$). Generate the spatial profiles of u_z and ω_y by going to the *CASE1* directory and by typing the following command in the terminal window:

```
user@hostname/CASE1: postProcess -func sampleDict
```

This command creates a new folder called *postProcessing/sampleDict* with time-stamped folders that each contain a file called *lineX1_Uz_vorticityy.xy*; this file stores the spatial variations of u_z and ω_y along $z = 0$ and at the time indicated by the name of the folder. Note that the post-processing provides values of the variables of interest at the center of each computational cell (*interpolationScheme cell* in the *sampleDict* dictionary). For instance, in the case of a simulation with 101 cells in the x- and z-directions, the coordinates of the first point are $(x,z) = (-0.0495049504950495, 0)$ where the x-coordinate is given by $x = -0.05 + (\Delta x/2)$ with $\Delta x = (0.1/101)$; similarly, the coordinates of the second point are $(x,z) = (-0.0485148514851485, 0)$ where the x-coordinate is given by $x = -0.05 + (3 \times \Delta x/2)$. Because variables in FireFOAM are colocated and calculated at cell centers, the *sampleDict* post-processing gives here direct access to the simulated values of u_z and ω_y without interpolation (interpolation should be avoided as interpolation errors may preclude the analysis of numerical solution errors).

2.6 RESULTS

2.6.1 *Visualization with ParaView*

The general features of the vortex flow solution can be conveniently examined in ParaView. Plot the time variations of the u_x, u_z, p_{gauge} and ω_y fields by creating a dummy file in the *CASE1* directory with the extension .foam (*case1.foam*) and by calling ParaView or paraFoam in the terminal window:

```
user@hostname/CASE1: touch case1.foam
```

```
user@hostname/CASE1: paraview
```

For instance, Figure 2.5 presents the spatial variations of vorticity ω_y at time $t = 3$ s. Recall that in addition to its own rotational motion, the vortex structure is convected along the x-direction by the external flow at velocity $u_{coflow} = 0.1$ m/s: at time $t = 3$ s, the vortex has covered a distance of $(u_{coflow} \times t) = 0.3$ m (*i.e.*, three times the size of the computational domain). While this problem could be simulated using a computational domain large enough to capture the entire displacement of the vortex structure, it is more effective to simulate the problem using periodic boundary conditions. The periodic boundary conditions applied at the west and east boundaries of the domain essentially simulate an infinite train of vortices along the x-direction separated by a 0.1-m distance: when the vortex leaves the domain at the east boundary, the same vortex re-enters at the west boundary. Thus, at time $t = 3$ s, the center of the vortex structure is back to its original position at $(x_c, z_c) = (0, 0)$.

Figure 2.5 illustrates the effects of grid resolution on the numerical solution: when the resolution is fine, *i.e.* for $(r_{c,0}/\Delta x) = 10$ and $(r_{c,0}/\Delta x) = 20$, the simulated vorticity field features the expected circular symmetry (see Section 2.1); in contrast, when the resolution is coarse, *i.e.* for $(r_{c,0}/\Delta x) = 5$, the simulated vorticity field displays an incorrect lack of circular symmetry and also an unsteady structure (not shown).

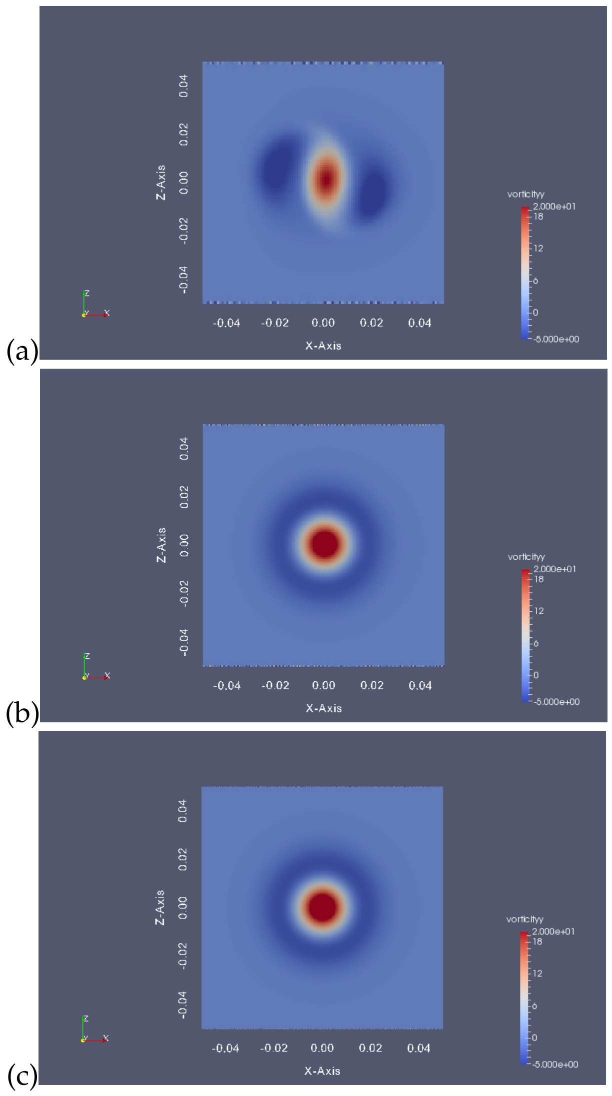


Figure 2.5: Spatial variations of ω_y at time $t = 3$ s. Comparison between numerical solutions obtained with three levels of grid resolution: (a) $(r_{c,0}/\Delta x) = 5$; (b) $(r_{c,0}/\Delta x) = 10$; (c) $(r_{c,0}/\Delta x) = 20$. Simulations performed using the *backward* time integration scheme and an adjustable time step controlled by a CFL number of 0.5.

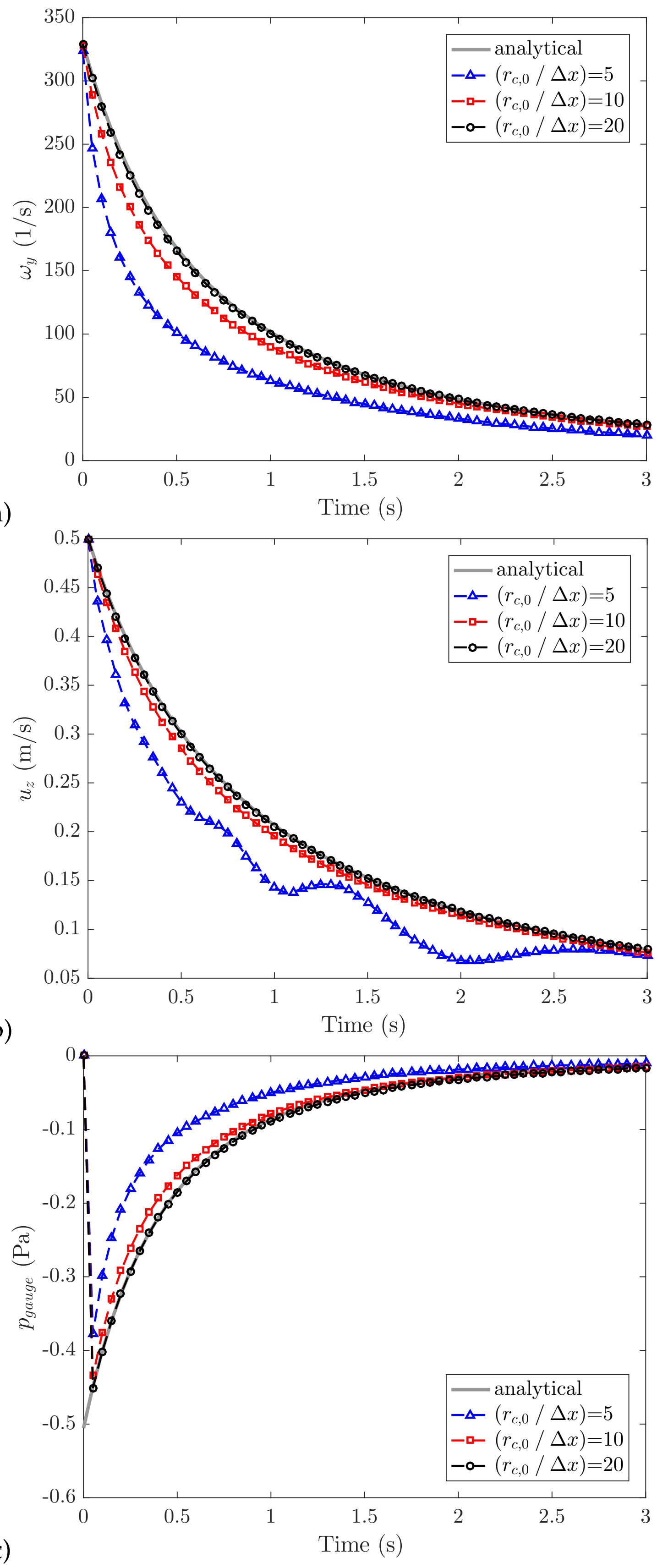


Figure 2.6: Temporal variations of: (a) the maximum value of ω_y; (b) the maximum value of u_z; (c) the minimum value of p_{gauge}. Comparison between the analytical solution and the numerical solutions obtained with three levels of grid resolution, $(r_{c,0}/\Delta x) = 5$, 10 and 20. Simulations performed using the *backward* time integration scheme and an adjustable time step controlled by a CFL number of 0.5.

2.6.2 *Time Evolution of Vortex Strength*

The time evolution of the vortex strength is now examined in Figure 2.6 by plotting the maximum values of ω_y and u_z as well as the minimum value of p_{gauge} as a function of time (note that the maximum value of ω_y and the minimum value of p_{gauge} occur at the vortex center whereas the maximum value of u_z occurs at a distance r_c from the vortex center, see Section 2.1). These values are extracted from the *volFieldValue.dat* files created by the *postProcess* functionality (see Section 2.5.2). The numerical results in Figure 2.6 are presented for three levels of grid resolution and are compared to the analytical solution presented in Section 2.1.

Figure 2.6 illustrates the temporal decay of the vortex strength due to viscosity through the progressive deceleration of the vortex and the return to uniform pressure. In addition, Figure 2.6 illustrates the effects of grid resolution on the numerical solution: when the resolution is fine, *i.e.* for $(r_{c,0}/\Delta x) = 10$ or 20, the simulated values are very close to the analytical values; in contrast, when the resolution is coarse, *i.e.* for $(r_{c,0}/\Delta x) = 5$, the viscous decay is overestimated (a classical problem known as numerical diffusion) and the solution displays unrealistic oscillations. The numerical errors increase at even lower resolution levels (not shown).

2.6.3 *Spatial Profiles of Vorticity and Velocity*

The spatial variations of y-vorticity and z-velocity are now examined in Figure 2.7 by plotting the instantaneous profiles of ω_y and u_z along $z = 0$ at time $t = 3$ s. The profiles are extracted from the *lineX1_Uz_vorticityy.xy* files created by the *postProcess* functionality (see Section 2.5.3). The numerical results in Figure 2.7 are presented for three levels of grid resolution and are compared to the analytical solution presented in Section 2.1.

Figure 2.7 shows the classical structure of a (clockwise-rotating) Lamb-Oseen vortex: the structure is characterized by a central core of positive vorticity surrounded by a ring region of negative vorticity (at large distances from the vortex center, the Lamb-Oseen vortex has zero circulation). Figure 2.7 also illustrates the effects of grid resolution and confirms that for $(r_{c,0}/\Delta x) \geq 10$, the solution is accurate while for $(r_{c,0}/\Delta x) \leq 5$, the solution has significant numerical error.

2.6.4 *Scaling of the Error with Grid Resolution*

The numerical error in the profiles presented in Figure 2.7 can be quantified using the standard deviation of the differences between values obtained with the simulated and analytical solutions (*i.e.* using the root-mean-square or *rms* error). While the analysis is simple, it is important to recognize that the scaling of the numerical error, noted ϵ, is far from trivial since ϵ depends on both the spatial discretization through the grid spacing Δx and the temporal discretization through the time increment Δt. In order to isolate the effect of Δx, new simulations are performed using a very small value of Δt: $\Delta t = 10^{-5}$ s (to this end, the parameters called *adjustTimeStep* and *deltaT* in the *controlDict* dictionary are modified as follows: *adjustTimeStep no* and *deltaT 0.00001*). The intent here is to minimize the temporal discretization error and create conditions in which ϵ is dominated by Δx. Figure 2.8 presents the resulting variations of the *rms* error with Δx and shows that the numerical error is second-order in space. In addition, Figure 2.8 compares results obtained with the *backward* and *Euler* time integration schemes. As may have been expected, in simulations with $\Delta t = 10^{-5}$ s, there is little difference between the *backward* and *Euler* schemes. Note, however, that this result does not hold in baseline simulations, *i.e.* in simulations using an adjustable time step controlled by a CFL number of 0.5. For these baseline conditions, Figure 2.9 shows that the *backward* scheme is significantly more accurate than the *Euler* scheme.

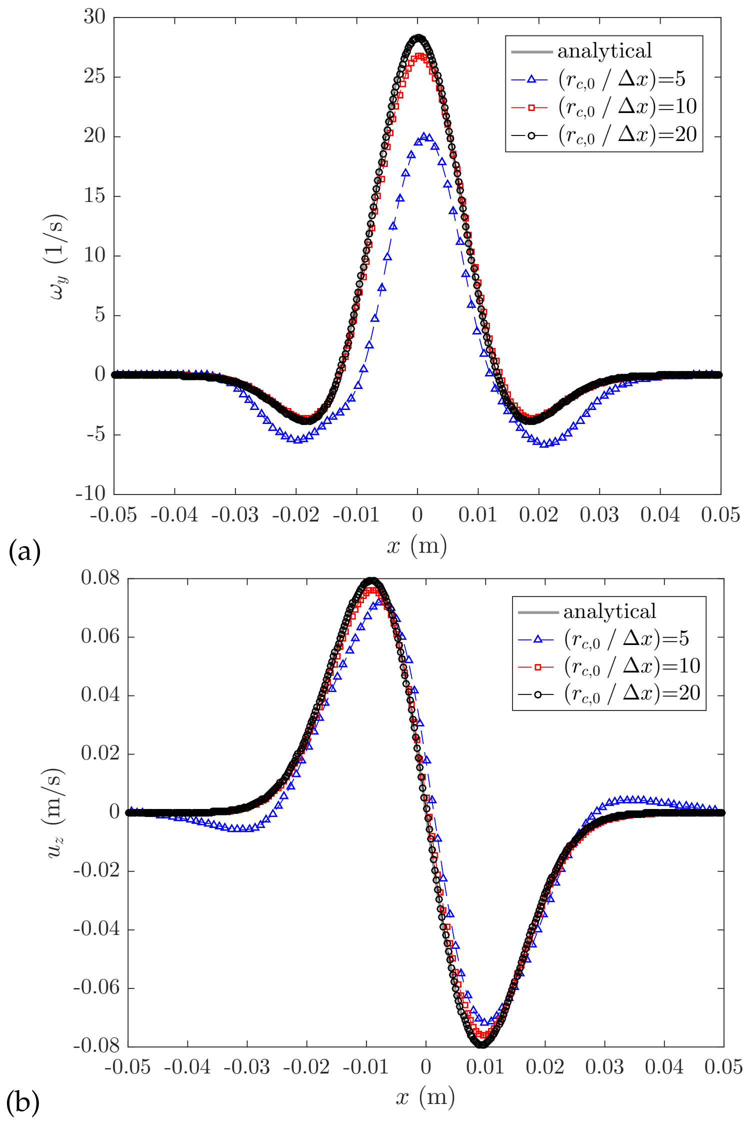


Figure 2.7: Spatial variations of: (a) ω_y; and (b) u_z, along $z = 0$ at time $t = 3$ s. Comparison between the analytical solution and the numerical solutions obtained with three levels of grid resolution, $(r_{c,0}/\Delta x) = 5$, 10 and 20. Simulations performed using the *backward* time integration scheme and an adjustable time step controlled by a CFL number of 0.5.

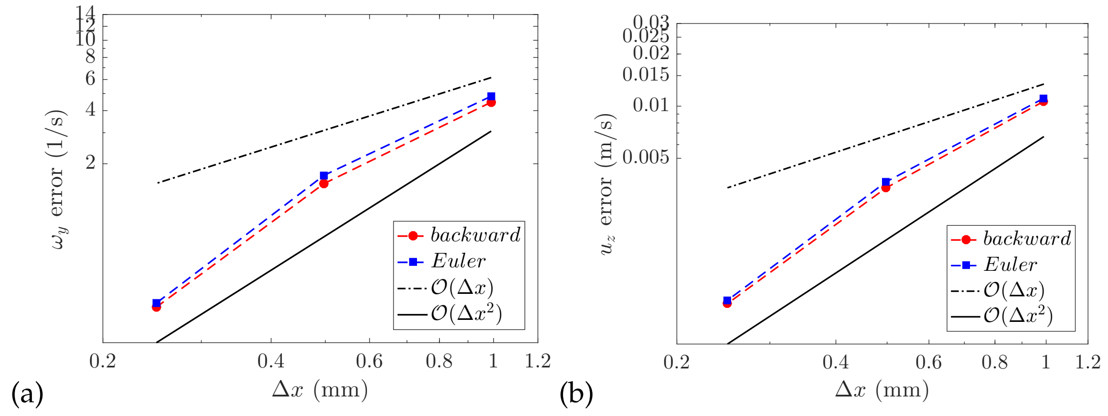


Figure 2.8: Root-mean-square error versus grid spacing Δx (log-log plots); error in: (a) ω_y; and (b) u_z, calculated along the line $z = 0$ and at time $t = 1$ s. Comparison between the *backward* and *Euler* time integration schemes. The black dashed line indicates first order behavior; the black solid line indicates second order behavior. Simulations performed using a constant time step $\Delta t = 10^{-5}$ s.

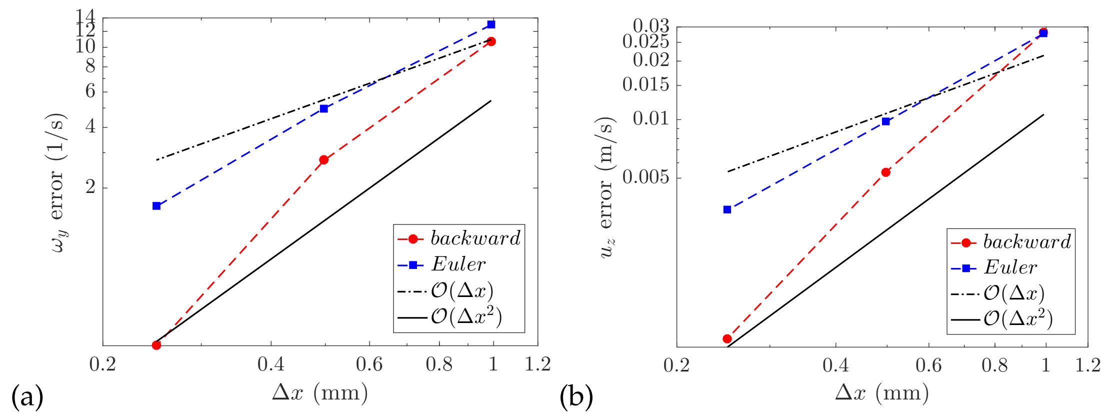


Figure 2.9: Root-mean-square error versus grid spacing Δx (log-log plots); error in: (a) ω_y; and (b) u_z, calculated along the line $z = 0$ and at time $t = 1$ s. Comparison between the *backward* and *Euler* time integration schemes. The black dashed line indicates first order behavior; the black solid line indicates second order behavior. Simulations performed using an adjustable time step controlled by a CFL number of 0.5.

2.7 CONCLUSION

The Lamb-Oseen vortex case is a simple flow configuration that allows meaningful tests of the effects of spatial or temporal resolution. The results above confirm the usual rule of thumb that suggests that for a given feature of size L, 10 grid cells are required across L to provide an accurate representation of the variations associated with that particular feature. In the present case, L may be interpreted as the diameter of the vortex structure, $L = (2 \times r_c)$, and accurate results are obtained provided that at least 10-20 grid cells span the diameter of the vortex.

3 CASE 2: TAYLOR-GREEN VORTEX

"Big whirls have little whirls,
That feed on their velocity;
And little whirls have lesser whirls,
And so on to viscosity."

— Lewis F. Richardson [30]

3.1 CASE DESCRIPTION

The inviscid Taylor-Green vortex case is a three-dimensional flow configuration that features analytical initial conditions (corresponding to a complex but single-scale laminar flow), strong unsteady dynamics, generation of small-scales by vortex stretching, and transition to a turbulent-like flow [15, 31]. At initial time, the flow motions are well-resolved but because there is no lower bound on the small scales produced by vortex stretching, at a certain point during the simulations, the smallest scales of the flow will become first under-resolved, and later fully unresolved. In addition, because the simulations are run in direct numerical simulation (DNS) mode (*i.e.*, without subgrid-scale models), the continuous production of under-resolved scales will result in growing numerical errors. The Taylor-Green vortex case is often used in CFD to evaluate the ability of a particular software to treat under-resolved scales [12]. Furthermore, because the generation of small-scales in the Taylor-Green vortex case is similar to the cascade of turbulent kinetic energy observed in turbulent flows [30], the configuration is also relevant to the general problem of simulating turbulent flows.

In the Taylor-Green vortex problem, the flow dynamics are usually characterized in terms of the temporal evolution of the (volume-averaged) flow kinetic energy and enstrophy (the enstrophy is defined as one half times the square of the magnitude of the vorticity vector, $(\omega^2/2) = (\omega_x^2 + \omega_y^2 + \omega_z^2)/2$, and is a quantity that characterizes the energy contained in the smallest scales). Theoretically, because the flow is inviscid, the kinetic energy remains constant and equal to its initial value; numerically, however, because the solution is gradually affected by dissipative errors due to under-resolved scales, the simulated kinetic energy decreases in time and deviations from the initial value give a measure of the amplitude of numerical errors. In addition, while the evolution of enstrophy is more difficult to interpret, a semi-analytical solution for the early growth of enstrophy can be found in [15] and can also be used to evaluate the weight of numerical errors; this solution is reported in Table 3.1.

Thus, in the simulation of the inviscid Taylor-Green vortex case in DNS mode, the question is not whether but when numerical errors become significant: simulation results will stay close to the theoretical predictions (constant kinetic energy; variations of enstrophy close to the results of [15]) up to a certain characteristic time τ_{TGV}

and will deviate from theory after that time. The characteristic time τ_{TGV} is long for high-quality solvers and/or high levels of grid resolution; in contrast, τ_{TGV} is short for low-quality solvers and/or low levels of grid resolution.

The initial conditions for the Taylor-Green vortex case are [12]:

$$\begin{aligned} u_x &= \sin(x)\cos(y)\cos(z) \\ u_y &= -\cos(x)\sin(y)\cos(z) \\ u_z &= 0 \end{aligned} \tag{3.1}$$

The velocity field is divergence-free and is associated with the following choices for mass density and pressure:

$$\begin{aligned} \rho &= 1 \\ p &= p_\infty + \frac{1}{16}\left(\cos(2x) + cos(2y)\right)\left(2 + \cos(2z)\right) \end{aligned} \tag{3.2}$$

where p_∞ is the ambient pressure and where the expression for p comes from integration of the classical Poisson equation for pressure. The ambient pressure can be chosen arbitrarily (as long as the speed of sound, $c_\infty = (\gamma p_\infty/\rho)^{0.5}$, is large enough to justify the implicit assumption of an incompressible flow): we choose $p_\infty = 101325$ Pa. Finally, given these choices, the temperature is initialized using the ideal gas law.

TIME (S)	$\overline{(\omega^2/2)}$ (1/s^2)	$\overline{\omega^2}/\overline{\omega^2}(t=0)$
0	0.37500	1.00000
0.25	0.37746	1.00656
0.5	0.38495	1.02653
0.75	0.39789	1.06104
1	0.41691	1.11176
1.25	0.44288	1.18101
1.5	0.47686	1.27163
1.75	0.52016	1.38709
2	0.57435	1.53160
2.25	0.64130	1.71013
2.5	0.72333	1.92888
2.75	0.82347	2.19592
3	0.94608	2.52288
3.25	1.09875	2.93000
3.5	1.29652	3.45739

Table 3.1: Temporal evolution of volume-mean enstrophy $\overline{(\omega^2/2)}$ in the Taylor-Green vortex case for $t \leq 3.5$ s [15]. The second column of the table gives the values of the volume-mean of enstrophy in SI units; the third column gives the same values made non-dimensional by the value at initial time $t = 0$.

3.2 DIRECTORY STRUCTURE

For the Taylor-Green vortex case, the main relevant features of the *constant/* directory are as follows. The *turbulenceProperties* dictionary specifies that the simulation is performed in DNS mode, *i.e.*, without subgrid-scale turbulence models (*simulationType laminar*, which does not mean that the flow is necessarily laminar but rather means that subgrid-scale models are not used). In addition, the *thermo.compressibleGas* dictionary has been artificially modified to impose zero viscosity ($As = 0$) for O_2 and N_2 (the only species used in the present case).

Furthermore, for the Taylor-Green vortex case, the main relevant features of the *system/* directory are as follows. The *decomposeParDict* dictionary specifies the domain decomposition scheme adopted in parallel computing mode (in the present case, we use 8 processors); the *controlDict* dictionary specifies that: the duration of the simulation is 10.0 s (*endTime 10*); the simulation runs with an adjustable time step (*adjustTimeStep yes*) controlled by a Courant-Friedrichs-Lewy (CFL) number of 0.5 (*maxCo 0.5*); the output files are written every 0.5 s (*writeInterval 0.5* in the *writeObjects1* section) and in compressed format (*writeCompression compressed*; this instruction generates files with a *.gz* extension); and the output files store the velocity, mass density, pressure and temperature fields (U, ρ, p and T in the *writeObjects1* section). In addition, the *fvSchemes* dictionary specifies the schemes used for temporal and spatial discretization; in the present case, we consider four spatial schemes for the convective term that appears in the momentum equation: *div(phi,U) Gauss filteredLinear2V 0.1 0.05*; *div(phi,U) Gauss filteredLinear2V 1 0.05*; *div(phi,U) Gauss linear*; and *div(phi,U) Gauss LUST grad(U)*. In these instructions, *Gauss* refers to the finite volume method of calculating changes inside a cell volume based on summing flux contributions on cell faces. The *linear* option refers to a classical central differencing scheme; the *filteredLinear2V* option refers to a filtered version of the central differencing scheme; the *LUST* option refers to a scheme that blends the upwind and central differencing schemes. The first coefficient in the *filteredLinear2V* statement takes values between 0 and 1 and indicates the level of filtering: a value of 0 corresponds to no filtering; a value of 1 corresponds to maximum filtering. See the OpenFOAM documentation for details.

Finally, for the Taylor-Green vortex case, the main relevant features of the *0/* directory are as follows. All boundaries of the computational domain correspond to periodic boundary conditions (called *cyclic* in OpenFOAM). The *T*, *U*, *p* and *p_rgh* files specify uniform initial values of the temperature, flow velocity vector, pressure and modified pressure, and will be updated after calling the *initTGV* pre-processor (see Section 3.3.2).

3.3 PRE-PROCESSING

3.3.1 *Definition of the Computational Domain and Mesh Generation*

For the Taylor-Green vortex case, the main features of the *blockMeshDict* file (located in the *constant/polyMesh* subdirectory) are as follows. The computational domain is a simple rectangular cube of size $(2 \times \pi)^3$ m^3 in an (x, y, z) rectangular Cartesian system. The baseline mesh is uniform and uses 64 cells in each direction; the spatial resolution is equal to $((2 \times \pi)/64) \approx 0.098$ m. In the following, the spatial resolution is varied and we consider simulations with 32, 64 (baseline choice) and 128 cells in each direction.

We call *CASE2* the working directory for the Taylor-Green vortex case. Generate the mesh by going to this directory and by typing the following command in the terminal window:

```
user@hostname/CASE2: blockMesh
```

3.3.2 *Initial conditions*

The *initTGV* pre-processing utility specifies the initial conditions of the temperature, flow velocity vector, pressure and modified pressure fields. First, one needs to install the *initTGV* utility: go to any directory of choice (for instance the *OpenFOAM* installation directory); copy the archive file *initTGV.tar* to that directory; extract the archive file by typing *"tar -xvf initTGV.tar"*; go to the newly created *initTGV* directory by typing *"cd initTGV"*; and then compile the utility by typing *"wmake"*. Generate the initial fields by going to the *CASE2* directory and by typing the following command in the terminal window:

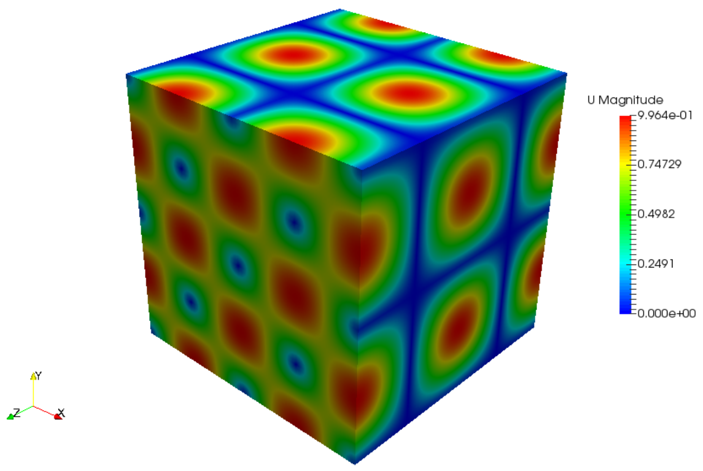
```
user@hostname/CASE2: initTGV
```

The *T*, *U*, *p* and *p_rgh* files in the *0/* directory have now been updated with initial fields that correspond to a Taylor-Green vortex problem (see Eqs. (3.1) and (3.2) and lines 101-109 in Listing 3.1). One can now visualize the initial fields in ParaView (Figure 3.1).

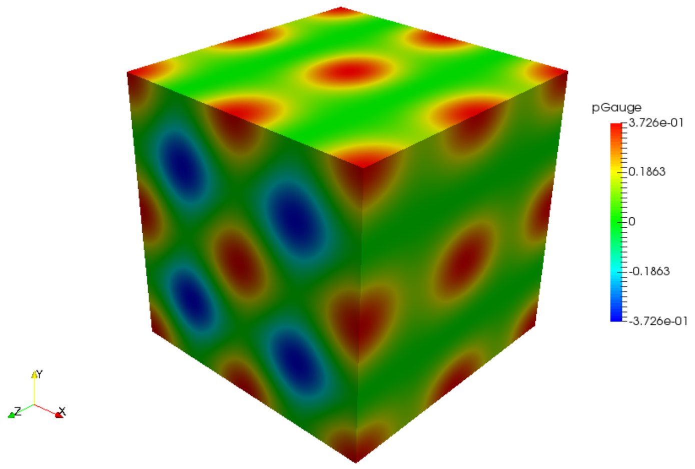
Listing 3.1: initTGV program for the Taylor-Green vortex case

```
1  /*---------------------------------------------------------------------------*\
2    =========                 |
3    \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
4     \\    /   O peration     |
5      \\  /    A nd           | Copyright (C) 1991-2005 OpenCFD Ltd.
6       \\/     M anipulation  |
7  -------------------------------------------------------------------------------
8  License
9      This file is part of OpenFOAM.
10
11     OpenFOAM is free software; you can redistribute it and/or modify it
12     under the terms of the GNU General Public License as published by the
13     Free Software Foundation; either version 2 of the License, or (at your
14     option) any later version.
15
16     OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
17     ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
18     FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
19     for more details.
20
```

```
    You should have received a copy of the GNU General Public License
    along with OpenFOAM; if not, write to the Free Software Foundation,
    Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Application
    initTGV -- written by Salman Verma

Description
    initializes the Taylor-Green vortex.

\*---------------------------------------------------------------------------*/

#include "fvCFD.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

int main(int argc, char *argv[])
{
#   include "setRootCase.H"

#   include "createTime.H"
#   include "createMesh.H"

#   include "physicoChemicalConstants.H"

    IOobject Uheader
    (
        "U",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ
    );
    Info<< "Reading U" << endl;
    volVectorField U(Uheader, mesh);

    IOobject pheader
    (
        "p",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ
    );
    Info<< "Reading p" << endl;
    volScalarField p(pheader, mesh);


    IOobject p_rghheader
    (
        "p_rgh",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ
    );
    Info<< "Reading p_rgh" << endl;
    volScalarField p_rgh(p_rghheader, mesh);

    IOobject Theader
    (
        "T",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ
    );
    Info<< "Reading T" << endl;
    volScalarField T(Theader, mesh);

    scalar rho=1;
    scalar myR = 288.189780685556;

    scalar x1;
    scalar x2;
    scalar x3;

    forAll(mesh.C(), celli)
    {

            x1 = mesh.C()[celli].x();
            x2 = mesh.C()[celli].y();
            x3 = mesh.C()[celli].z();

            U[celli].component(0) = Foam::sin(x1)*Foam::cos(x2)*Foam::cos(x3);
            U[celli].component(1) = -Foam::cos(x1)*Foam::sin(x2)*Foam::cos(x3);
            U[celli].component(2) = 0;

            p[celli] = 101325+((Foam::cos(2*x3)+2)*(Foam::cos(2*x1)+Foam::cos
                (2*x2))/16);

            p_rgh[celli] = p[celli];

            T[celli] = p[celli]/(rho*myR);

    }

    Info << nl << "Writing modified U, p, p_rgh and T fields with the Taylor-
        Green vortex to folder " << runTime.timeName() << endl;
    U.write();
    p.write();
    p_rgh.write();
    T.write();

    Info<< endl;

    return(0);
}
// ************************************************************************* //
```



(a)



(b)

Figure 3.1: Initial conditions for flow velocity and pressure in the Taylor-Green vortex case: (a) magnitude of the velocity vector; (b) gauge pressure. The gauge pressure field was generated using the *pGauge* functionality previously described in section Section 2.5.1.

3.4 RUNNING THE SIMULATION

The Taylor-Green vortex case is run in parallel computing mode. The *decomposeParDict* dictionary in the *system/* directory specifies the number of processors to be used (*numberOfSubdomains 8*) and the structure of the domain decomposition (*n (2 2 2)* for a decomposition of the *x*-, *y*- and *z*-directions into 2 blocks each). Decompose the computational domain and assign a portion of the mesh to each processor by going to the *CASE2* directory and by typing the following command in the terminal window:

```
user@hostname/CASE2: decomposePar
```

Then run the simulation by typing:

```
user@hostname/CASE2: mpirun -np 8 fireFoam -parallel > log.CASE2
```

where the file named *log.CASE2* contains the messages generated by FireFOAM during the simulation. FireFOAM creates 8 output directories (called *processor0*, ..., *processor7*) and in each of those, generates time-stamped folders every 0.5 s. These folders contain U, ρ, p and T files that store the portion of the velocity, mass density, pressure and temperature fields managed by the processor and at the time indicated by the name of the folder.

When the simulation is completed, merge the output directories, folders and files for analysis by typing:

```
user@hostname/CASE2: reconstructPar
```

The *reconstructPar* functionality creates new time-stamped folders in the *CASE2* folder (written every 0.5 s) that now contain U, ρ, p and T files that store the full fields.

3.5 POST-PROCESSING

3.5.1 *Flow Visualization*

A classical approach to visualize the complex structure of a three-dimensional vortical flow field consists in plotting iso-contours of the second invariant of the velocity gradient tensor, also called Q, $Q = (1/2)(\tilde{\Omega}_{ij}\tilde{\Omega}_{ij} - \tilde{S}_{ij}\tilde{S}_{ij})$, where $\tilde{\Omega}_{ij}$ is the rotation rate tensor and $\tilde{S}_{ij}$ is the strain rate tensor [32, 33]. First calculate the Q field by typing:

```
user@hostname/CASE2: postProcess -func 'Q'
```

This command generates new Q files inside the time-stamped folders previously created by *reconstructPar*. These files are now available for visualization in ParaView.

3.5.2 *Time Evolution of Volume-Averaged Quantities*

The Taylor-Green vortex case uses the *postProcess* functionality of OpenFOAM to analyze the time evolution of the volume-averaged flow kinetic energy and enstrophy. First generate the relevant temporally- and spatially-resolved information by going to the *CASE2* directory and by typing the following commands in the terminal window:

```
user@hostname/CASE2: postProcess -func "magSqr(U)"
```

```
user@hostname/CASE2: postProcess -func enstrophy
```

The first command generates new *magSqr(U)* files containing the instantaneous local values of the square of the velocity vector, $u^2 = (u_x^2 + u_y^2 + u_z^2)$ (*i.e.*, twice the kinetic energy); the second command generates new *enstrophy* files containing the instantaneous local values of enstrophy, $(\omega^2/2) = (\omega_x^2 + \omega_y^2 + \omega_z^2)/2$. These files are created inside the time-stamped folders previously created by *reconstructPar*.

The data in the *magSqr(U)* and *enstrophy* files now need to be volume-averaged for analysis. For this purpose, the Taylor-Green vortex case uses the *volAverageEnstrophyKE* functionality. First, one needs to install the *volAverageEnstrophyKE* post-processing utility: go to any directory of choice (for instance the *OpenFOAM* installation directory); copy the archive file *volAverageEnstrophyKE.tar* to that directory; extract the archive file by typing *"tar -xvf volAverageEnstrophyKE.tar"*; go to the newly created *volAverageEnstrophyKE* directory by typing *"cd volAverageEnstrophyKE"*; and then compile the utility by typing *"wmake"*. Finally, generate a file that contains the volume-mean values of kinetic energy and enstrophy by going to the *CASE2* directory and by typing the following command in the terminal window:

```
user@hostname/CASE2: volAverageEnstrophyKE
```

This command generates a text file called *volAveragedEnstrophyKE* that contains three columns: the first column gives the discrete output times (in units of s and with a temporal resolution of 0.5 s); the second column gives the corresponding values of the global (*i.e.*, volume-mean) enstrophy (in units of $1/s^2$), $\overline{(\omega^2/2)} = (\int_V (\omega^2/2) dV)/V$, where V is the volume of the computational domain; and the third column gives the corresponding values of the global (*i.e.*, volume-mean) kinetic energy (in units of $kg/m/s^2$), $\overline{(\rho u^2/2)} = (\int_V (\rho u^2/2) dV)/V$. Note that in the present case, the mass density remains very close to its initial value of 1 and the introduction of ρ in the definition of the global kinetic energy could be simply ignored.

3.6 RESULTS

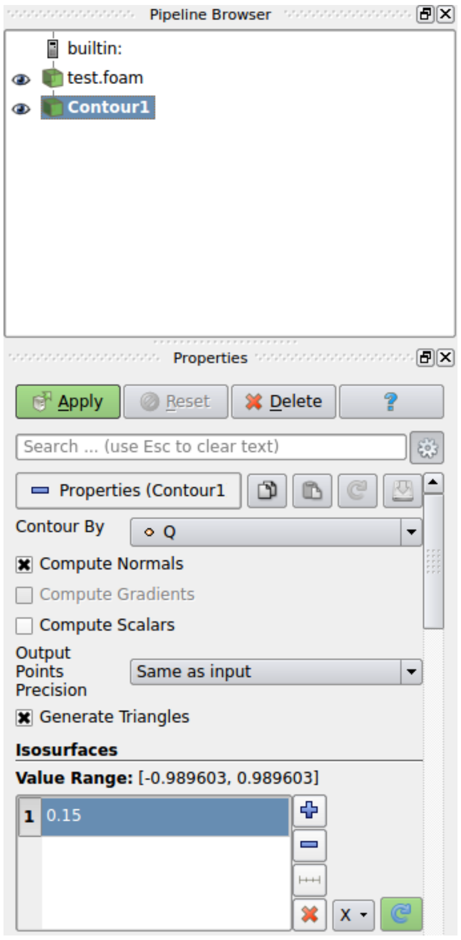
3.6.1 *Visualization with ParaView*

The general features of the flow solution can be conveniently examined in ParaView. Plot the time variations of the u_x, u_y, u_z or Q fields by creating a dummy file in the *CASE2* directory with the extension .foam (*case2.foam*) and by calling ParaView or paraFoam in the terminal window:

```
user@hostname/CASE2: touch case2.foam
```

```
user@hostname/CASE2: paraview
```

In order to visualize an iso-contour of Q, select the *Contour* filter which can be found on the *Common* tool bar or in the *Common* sub-menu of the *Filters* top menu. A new item is added to the *Pipeline Browser* (see Figure 3.2) called *Contour1*. In the *Properties* panel (see Figure 3.2), change the *Contour By* parameter to Q; click the *Value Range* in the *Isosurfaces* section and write a new value equal to 0.15; and then click *Apply*. In the *Properties* panel, change the option in the *Coloring* section to U. ParaView will now display the iso-surface $Q = 0.15\ s^{-2}$ colored by the values of the magnitude of the velocity vector.



Contour By: Q

[✓] Compute Normals
[ ] Compute Gradients
[ ] Compute Scalars

Output Points Precision: Same as input

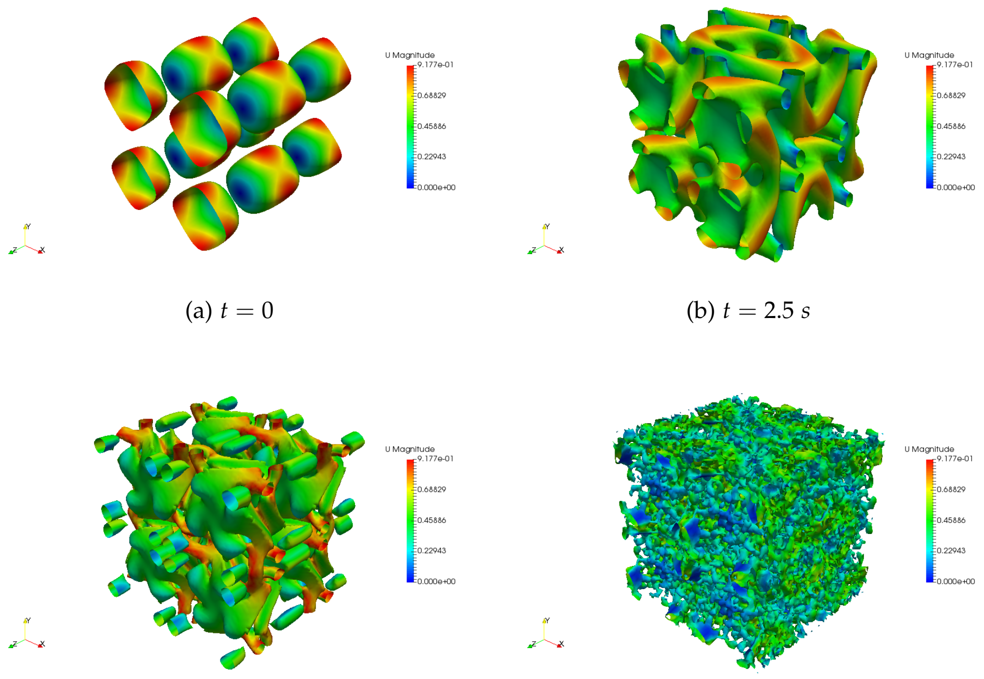
[✓] Generate Triangles

Value Range: [-0.989603, 0.989603]

1 0.15

Figure 3.2: *Pipeline Browser* and *Properties* in ParaView

Figure 3.3 presents instantaneous snaphots of the iso-surface $Q = 0.15\ s^{-2}$ taken at different times during a simulation corresponding to the *div(phi,U) Gauss filteredLinear2V 0.1 0.05* scheme and a 64^3 computational grid. The snapshots show that the simulated flow starts as a single-scale laminar flow ($t = 0$), then produces a discrete number of smaller scales ($t = 2.5$ and $5\ s$), and finally transitions to a turbulent-like flow characterized by a broadband range of scales ($t = 10\ s$).



(a) $t = 0$ (b) $t = 2.5\ s$

(c) $t = 5\ s$ (d) $t = 10\ s$

Figure 3.3: Instantaneous snapshots of the iso-surface $Q = 0.15\ s^{-2}$ colored by the values of the magnitude of the velocity vector. Simulation performed with the *div(phi,U) Gauss filteredLinear2V 0.1 0.05* scheme and a grid resolution corresponding to 64^3 cells.

3.6.2 *Time Evolution of Kinetic Energy and Enstrophy*

The time evolution of global kinetic energy and enstrophy are now examined in Figure 3.4. The data are extracted from the *volAveragedEnstrophyKE* files created by the *volAverageEnstrophyKE* utility (see Section 3.5.2). Figure 3.4 compares numerical results obtained with four different spatial schemes proposed to treat convection and with theoretical predictions corresponding to constant kinetic energy and to the variations of enstrophy presented in Table 3.1. Figure 3.4 illustrates the growing effect of dissipative errors due to under-resolved scales: for the baseline grid resolution (64^3 cells), the simulations under-estimate the global flow kinetic energy for $t \geq 1.5$ s and also under-estimate the global enstrophy for $t \geq 2.5$ s. The simulated variations of global kinetic energy do not allow a differentiation between the four spatial schemes that are considered; in contrast, the simulated variations of global enstrophy suggest that the best schemes are the *linear* and *filteredLinear2V 0.1* options (these schemes provide higher values for the global enstrophy, which suggests that the energy contained in the smallest scales is more accurately described).

Next we consider the effect of grid resolution on the Taylor-Green vortex case. Figure 3.5 presents the simulated variations of $\overline{(\rho u^2/2)}$ and $\overline{(\omega^2/2)}$ obtained for grid resolutions corresponding to 128^3 (finest grid), 64^3 (baseline grid) and 32^3 (coarsest grid) cells. As may have been expected, Figure 3.5 shows that results obtained with

DIVERGENCE SCHEME	KINETIC ENERGY ($t = 5\ s$)	ENSTROPHY ($t = 3.5\ s$)
filteredLinear2V 0.1	0.9234	2.8010
filteredLinear2V 1	0.9226	2.8010
linear	0.9235	2.8010
LUST grad(U)	0.9320	2.8369
[15]	1.0	3.4574

Table 3.2: Accuracy metrics for the Taylor-Green vortex case, as proposed in [12]

the coarsest grid deviate from theoretical predictions at early times whereas results obtained with the finest grid remain close to the theoretical predictions for a long time.

Following [12], we present in Table 3.2 a quantification of the accuracy of the FireFOAM simulations performed with a baseline computational grid (64^3) by reporting the value of global kinetic energy at time $t = 5$ s and the value of global enstrophy at time $t = 3.5$ s (the last time for which semi-analytical results are available in [15]). In Table 3.2, the values of $\overline{(\rho u^2/2)}$ and $\overline{(\omega^2/2)}$ are made non-dimensional by their values at initial time $t = 0$. The values in Table 3.2 can be compared to those obtained with the higher-order schemes discussed in [12]: they are found to be lower, which suggests that the FireFOAM schemes (which are second-order accurate in space and in time) are more dissipative than the higher-order schemes considered in [12]. However, note that the performance of the FireFOAM schemes can be readily improved by increasing spatial resolution: for instance, for the *div(phi,U) Gauss filteredLinear2V 0.1 0.05* scheme with a 64^3 computational grid, the test value of normalized $\overline{(\rho u^2/2)}$ is 0.9234 and the test value of normalized $\overline{(\omega^2/2)}$ is 2.8010; with a 128^3 computational grid, the test value of normalized $\overline{(\rho u^2/2)}$ becomes 0.9659 and the test value of normalized $\overline{(\omega^2/2)}$ becomes 3.1879. These higher values indicate more accurate simulations and are similar to those obtained with the higher-order schemes considered in [12].

3.7 CONCLUSION

The Taylor-Green vortex case is a complex flow configuration that reproduces the kinetic energy transfer from large to small scales observed in turbulent flows and that allows meaningful tests of the effects of dissipative errors due to under-resolved scales in simulations without subgrid-scale models. The evolution of flow kinetic energy and enstrophy provide valuable metrics to compare different numerical schemes and/or different solvers.

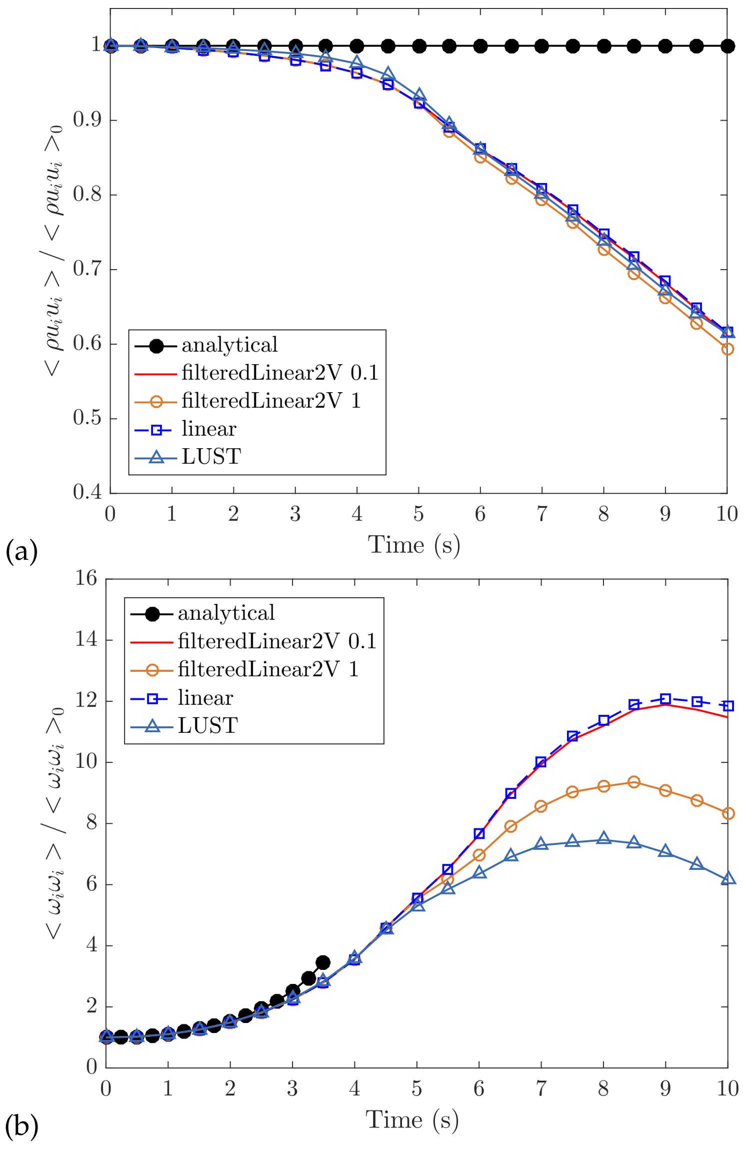


Figure 3.4: Temporal variations of: (a) global kinetic energy $\overline{(\rho u^2/2)}$; (b) global enstrophy $\overline{(\omega^2/2)}$. Quantities are made non-dimensional by their value at initial time $t = 0$. Comparison between the theoretical solution of [15] (black dots) and the numerical solutions obtained with four different spatial schemes for the convective term in the momentum equation (see section Section 3.2). Simulations performed with the baseline grid resolution corresponding to 64^3 cells.

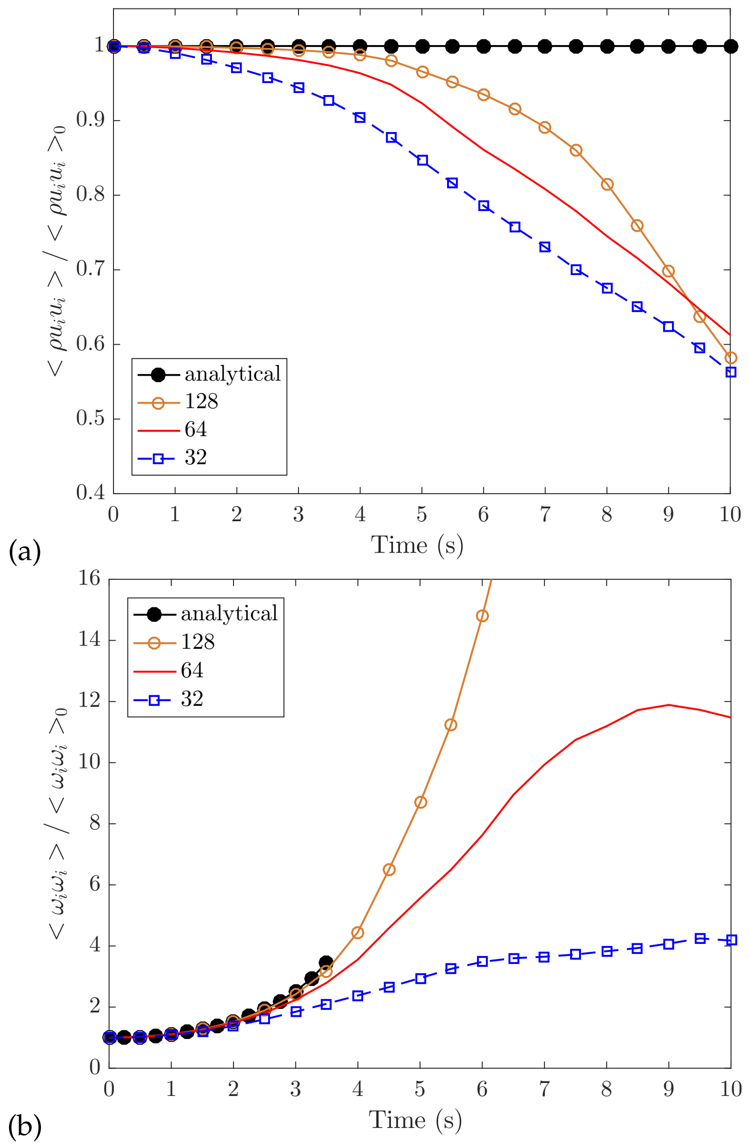


Figure 3.5: Temporal variations of: (a) global kinetic energy $\overline{(\rho u^2/2)}$; (b) global enstrophy $\overline{(\omega^2/2)}$. Quantities are made non-dimensional by their value at initial time $t = 0$. Comparison between the theoretical solution of [15] (black dots) and the numerical solutions obtained with the *filteredLinear2V 0.1* scheme and with three levels of grid resolution corresponding to 128^3, 64^3 and 32^3 cells.

4 CASE 3: DECAY OF HOMOGENEOUS ISOTROPIC TURBULENCE

4.1 CASE DESCRIPTION

This case is a three-dimensional turbulent flow configuration that simulates the classical wind tunnel experiment of von Kármán [34] and Comte-Bellot & Corrsin (CBC) [17]. The case features approximate initial conditions (corresponding to a turbulent flow field with prescribed statistical properties), cascading transport of turbulent kinetic energy from large to small scales, and viscous decay at small scales. At initial time, the flow velocity field is specified according to a measured distribution of turbulent kinetic energy as a function of size r (or more precisely, as a function of wavenumber, $\kappa = (2\pi/r)$); the field is assumed to be homogeneous and isotropic. While the CBC wind tunnel experiment corresponds to a stationary flow configuration with statistical properties that are constant in time and variable in space (properties vary with downstream distance along the flow path), the present simulations correspond to an analog homogeneous flow configuration with statistical properties that are variable in time and uniform in space. Comparisons between experimental data and simulation results therefore require a space-time transformation based on Taylor's hypothesis and a measured mean flow velocity inside the wind tunnel. Using this transformation, experimental measurements are provided at three specific times equal to 0, 0.28 and 0.66 s. Experimental measurements give both the mean value of turbulent kinetic energy and the corresponding spectral distribution (*i.e.*, the variations of kinetic energy with wavenumber, $E(\kappa)$). The measured spectral variations $E(\kappa)$ are reported in Table 4.1.

Note that in the simulations, only a fraction of the flow motions is grid-resolved (the fraction that corresponds to flow structures with a size r larger than the Nyquist wavelength, $\lambda_c = 2\Delta$, where Δ is the grid cell size, or equivalently, to a wavenumber κ smaller than the cut-off value $\kappa_c = (2\pi/\lambda_c) = (\pi/\Delta)$). Because the simulations are run in large eddy simulation (LES) mode, the dynamic effect of the unresolved flow motions is represented by a subgrid-scale model. Thus, the CBC case may be used to evaluate the ability of a LES software to accurately describe unresolved phenomena and in particular the rate of change of turbulent kinetic energy in a configuration that features the usual mechanism for turbulence dissipation (viscosity) but no mechanism for production.

Table 4.1: Measured variations of turbulent kinetic energy with wavenumber, $E(\kappa)$, as obtained in the CBC experiment [17]. The first column of the table gives κ in units of 1/cm; the other columns give $E(\kappa)$ in units of cm^3/s^2.

κ	$t = 0$	$t = 0.28\ s$	$t = 0.66\ s$
0.15	———	———	4.97E+01
0.2	1.29E+02	1.06E+02	9.20E+01
0.25	2.30E+02	1.96E+02	1.20E+02
0.3	3.22E+02	1.95E+02	1.25E+02
0.4	4.35E+02	2.02E+02	9.80E+01
0.5	4.57E+02	1.68E+02	8.15E+01
0.7	3.80E+02	1.27E+02	6.02E+01
1	2.70E+02	7.92E+01	3.94E+01
1.5	1.68E+02	4.78E+01	2.41E+01
2	1.20E+02	3.46E+01	1.65E+01
2.5	8.90E+01	2.86E+01	1.25E+01
3	7.03E+01	2.31E+01	9.12E+00
4	4.70E+01	1.43E+01	5.62E+00
6	2.47E+01	5.95E+00	1.69E+00
8	1.26E+01	2.23E+00	5.20E-01
10	7.42E+00	9.00E-01	1.61E-01
12.5	3.96E+00	3.63E-01	5.20E-02
15	2.33E+00	1.62E-01	1.41E-02
17.5	1.34E+00	6.60E-02	———
20	8.00E-01	3.30E-02	———

4.2 DIRECTORY STRUCTURE

For the CBC case, the main relevant features of the *constant/* directory are as follows. The *turbulenceProperties* dictionary specifies that the simulation is performed in LES mode, *i.e.*, with a subgrid-scale (SGS) turbulence model (*simulationType LES*). In the present case, we consider two SGS models: *LESModel kEqn* and *LESModel dynamicKEqn*. The *kEqn* model corresponds to a description of the subgrid-scale component of turbulent kinetic energy, k_{SGS}, through a transport equation with constant model coefficients (see [8] for details; the values of the model coefficients C_e and C_k used in the expressions of the SGS turbulent viscosity and the rate of dissipation of SGS turbulent kinetic energy are specified in the *kEqnCoeffs* section). The *dynamicKEqn* model corresponds to a description of k_{SGS} through a transport equation with dynamic model coefficients (see [8] for details).

Furthermore, for the CBC case, the main relevant features of the *system/* directory are as follows. The *decomposeParDict* dictionary specifies the domain decomposition scheme adopted in parallel computing mode (in the present case, we use 8 processors); the *controlDict* dictionary specifies that: the duration of the simulation is 0.66 s (*endTime 0.66*); the simulation runs with an adjustable time step (*adjustTimeStep yes*) controlled by a Courant-Friedrichs-Lewy (CFL) number of 0.5 (*maxCo 0.5*); the output files are written every 0.02 s (*writeInterval 0.02* in the *writeObjects1* section) and in compressed format (*writeCompression compressed*; this instruction generates files with a *.gz* extension); and the output files store the velocity and SGS turbulent kinetic energy fields (*U* and *k* in the *writeObjects1* section).

Finally, for the CBC case, the main relevant features of the *0/* directory are as follows. All boundaries of the computational domain correspond to periodic boundary conditions (called *cyclic* in OpenFOAM). The *U* file contains the initial velocity field and is taken from the Github repository of the Fire Dynamics Simulator [35]. This velocity field is constructed from a series of complex steps involving a starting guess based on a superposition of Fourier modes with random phases (this field satisfies mass conservation), a short simulation (this field now satisfies both mass and momentum conservation), and a re-normalization of the velocity field (this field satisfies both mass and momentum conservation and matches the measured turbulence intensity); the *U* field matches the spectral distribution $E(\kappa)$ obtained at the first CBC measurement station (time $t = 0$ in Table 4.1). Furthermore, compared to previous DNS cases, three additional files are required when running in LES mode: the *k*, *nut* and *alphat* files that respectively correspond to dictionaries for the SGS turbulent kinetic energy, $k = k_{SGS}$, the SGS (kinematic) turbulent viscosity, $nut = \nu_{SGS}$ (in units of m^2/s), and the product of mass density times the turbulent diffusivity, $alphat = (\bar{\rho} \times \nu_{SGS}/Pr_t)$ (in units of kg/m/s), where Pr_t is the turbulent Prandtl number. The *k*, *nut* and *alphat* dictionaries specify uniform initial values: the initial values of *nut* and $alphat$ are equal to 0 and are not used (in FireFOAM, ν_{SGS} and related quantities are calculated through their expressions based on k_{SGS}); in contrast, the initial value of $k = k_{SGS}$ is important and is specified as explained in Section 4.3.2.

4.3 PRE-PROCESSING

4.3.1 *Definition of the Computational Domain and Mesh Generation*

For the CBC case, the main features of the *blockMeshDict* file (located in the *constant/polyMesh* subdirectory) are as follows. The computational domain is a simple rectangular cube of size $L^3 = (0.18 \times \pi)^3 \approx (0.565)^3$ m^3 in an (x, y, z) rectangular Cartesian system (the size L follows a recommendation taken from the literature [18][36]). The baseline mesh is uniform and uses 64 cells in each direction; the spatial resolution is equal to $\Delta = (L/64) \approx 0.00884$ m. In terms of wavenumbers, the lowest resolved wavenumber is $\kappa_1 = (2\pi/L) \approx 11.11$ m^{-1}; the highest resolved wavenumber is $\kappa_c = (\pi/\Delta) \approx 355.55$ m^{-1}.

We call *CASE3* the working directory for the CBC case. Generate the mesh by going to this directory and by typing the following command in the terminal window:

```
user@hostname/CASE3: blockMesh
```

4.3.2 *Initial conditions*

As mentioned previously, the initial velocity field stored in the *U* file matches the spectral distribution $E(\kappa)$ obtained at the first CBC measurement station. At this station, the value of the global (*i.e.*, volume-mean) turbulent kinetic energy is $\int_0^\infty E(\kappa)d\kappa \approx 0.0777\ \mathrm{m}^2/\mathrm{s}^2$. However, in LES mode, the *U* file only contains the (resolved) grid-scale (GS) component of the flow field (this component corresponds to large length scales characterized by a wavenumber κ smaller than the cut-off value $\kappa_c \approx 355.55\ \mathrm{m}^{-1}$). Following a filtering procedure proposed in [21], we estimate that the GS component of the global turbulent kinetic energy is approximately equal to $0.0642\ \mathrm{m}^2/\mathrm{s}^2$ (this component is obtained after applying a low-pass filter to the CBC energy spectrum) whereas the SGS component is equal to $0.0135\ \mathrm{m}^2/\mathrm{s}^2$. Therefore, we prescribe the initial value of k_{SGS} in the k file as uniform (an approximation) and equal to $0.0135\ \mathrm{m}^2/\mathrm{s}^2$ (*internalField uniform 0.0135*).

One can visualize the initial fields in ParaView (Figure 4.1).

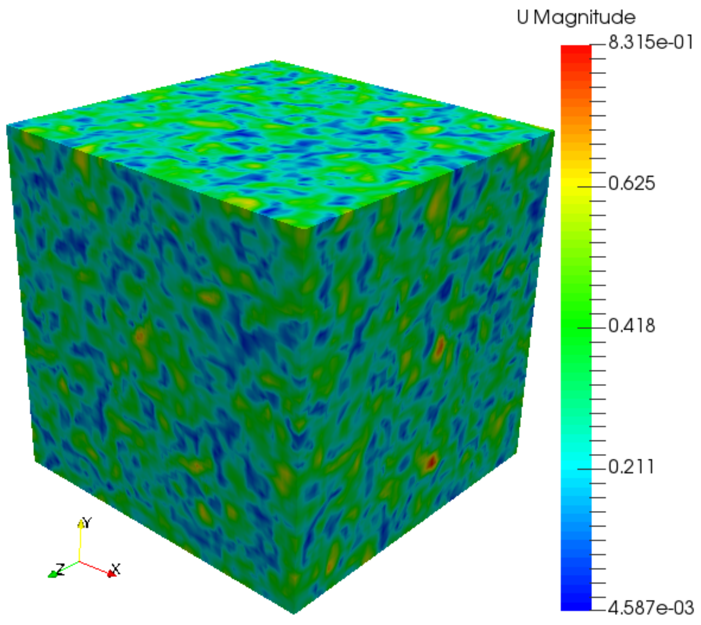


Figure 4.1: Initial conditions for the magnitude of the flow velocity vector in the CBC case

4.4 RUNNING THE SIMULATION

The CBC case is run in parallel computing mode. The *decomposeParDict* dictionary in the *system/* directory specifies the number of processors to be used (*numberOfSubdomains 8*) and the structure of the domain decomposition (*n (2 2 2)* for a decomposition of the *x*-, *y*- and *z*-directions into 2 blocks each). Decompose the computational domain and assign a portion of the mesh to each processor by going to the *CASE3* directory and by typing the following command in the terminal window:

```
user@hostname/CASE3: decomposePar
```

Then run the simulation by typing:

```
user@hostname/CASE3: mpirun -np 8 fireFoam -parallel > log.CASE3
```

where the file named *log.CASE3* contains the messages generated by FireFOAM during the simulation. FireFOAM creates 8 output directories (called *processor0*, ..., *processor7*) and in each of those, generates time-stamped folders every 0.02 s. These folders contain *U* and *k* files that store the portion of the velocity and SGS turbulent kinetic energy fields managed by the processor and at the time indicated by the name of the folder.

When the simulation is completed, merge the output directories, folders and files for analysis by typing:

```
user@hostname/CASE3: reconstructPar
```

The *reconstructPar* functionality creates new time-stamped folders in the *CASE3* folder (written every 0.02 s) that now contain *U* and *k* files that store the full fields.

4.5 POST-PROCESSING

The CBC case uses the *postProcess* functionality of OpenFOAM to analyze the time evolution of the volume-averaged GS and SGS components of turbulent kinetic energy. First generate the relevant temporally- and spatially-resolved information by going to the *CASE3* directory and by typing the following command in the terminal window:

```
user@hostname/CASE3: postProcess -func "magSqr(U)"
```

This command generates new *magSqr(U)* files containing the instantaneous local values of the square of the GS velocity vector, $\widetilde{u}^2 = (\widetilde{u}_x^2 + \widetilde{u}_y^2 + \widetilde{u}_z^2)$ (*i.e.*, twice the kinetic energy), where the tilde symbol denotes a density-weighted LES-filtered (*i.e.*, GS) quantity. These files are created inside the time-stamped folders previously created by *reconstructPar*.

The data in the *magSqr(U)* files now need to be volume-averaged for analysis. For this purpose, the CBC case uses the *kineticEnergyCBC* functionality. First, one needs to install the *kineticEnergyCBC* post-processing utility: go to any directory of choice (for instance the *OpenFOAM* installation directory); copy the archive file *kineticEnergyCBC.tar* to that directory; extract the archive file by typing *"tar -xvf kineticEnergyCBC.tar"*; go to the newly created *kineticEnergyCBC* directory by typing *"cd kineticEnergyCBC"*; and then compile the utility by typing *"wmake"*. Finally, generate a file that contains the volume-mean values of GS and SGS kinetic energy by going to the *CASE3* directory and by typing the following command in the terminal window:

```
user@hostname/CASE3: kineticEnergyCBC
```

This command generates a text file called *kineticEnergy* that contains four columns: the first column gives the discrete output times (in units of s and with a temporal resolution of 0.02 s); the second column gives the corresponding values of the global (*i.e.*, volume-mean) SGS turbulent kinetic energy (in units of m^2/s^2), $\overline{k_{SGS}} = (\int_V k_{SGS} dV)/V$; the third column gives the corresponding values of the global GS kinetic energy (in units of m^2/s^2), $\overline{(\widetilde{u}^2/2)} = (\int_V (\widetilde{u}^2/2) dV)/V$; and the fourth column gives the corresponding values of the total kinetic energy, defined as the sum of the GS and SGS components presented in the second and third columns.

4.6 RESULTS

4.6.1 *Time Evolution of GS and SGS Turbulent Kinetic Energy*

The time evolution of global GS and SGS turbulent kinetic energy is now examined in Figure 4.2 and Figure 4.3. The data are extracted from the *kineticEnergy* files created by the *kineticEnergyCBC* utility (see Section 4.5). Figure 4.2 focuses on the GS component of turbulent kinetic energy and presents a comparison between the CBC experimental data and numerical results obtained with two different SGS models. As reported in Table 4.1, the CBC experimental data ara available at times $t = 0, 0.28$ and 0.66 ms. Using the filtering procedure proposed in [21], the estimated values of the GS component of global turbulent kinetic energy are: 0.0642 $\mathrm{m}^2/\mathrm{s}^2$ at time $t = 0$; 0.0223 $\mathrm{m}^2/\mathrm{s}^2$ at $t = 0.28$ ms; and 0.0113 $\mathrm{m}^2/\mathrm{s}^2$ at $t = 0.66$ ms. Figure 4.3 presents a similar but more complete perspective by considering both the GS and SGS components of global turbulent kinetic energy. In this plot, the CBC experimental data are unfiltered and correspond to the following estimated values of the total turbulent kinetic energy: $\int_0^\infty E(\kappa) d\kappa \approx 0.0777$ $\mathrm{m}^2/\mathrm{s}^2$ at time $t = 0$; 0.0250 $\mathrm{m}^2/\mathrm{s}^2$ at $t = 0.28$ ms; and 0.0121 $\mathrm{m}^2/\mathrm{s}^2$ at $t = 0.66$ ms.

Figure 4.2 and Figure 4.3 illustrate the viscous decay of turbulent kinetic energy. As seen in Figure 4.2, both LES simulations remain close to the measured values of global GS turbulent kinetic energy and the level of agreement with the CBC data is satisfactory. However, as seen in Figure 4.3, the results obtained for the SGS component of turbulent kinetic energy are less satisfactory: while the simulation with the *kEqn* model appears to provide accurate results, the simulation with the *dynamicKEqn* model significantly over-estimates the measured values of global total turbulent kinetic energy. Note that these inaccuracies do not necessarily indicate a lower performance of the *dynamicKEqn* model; they may indicate instead the uncontrolled effect of errors introduced during the initialization of the k_{SGS} field (for instance, errors introduced by the incorrect assumption of a uniform distribution).

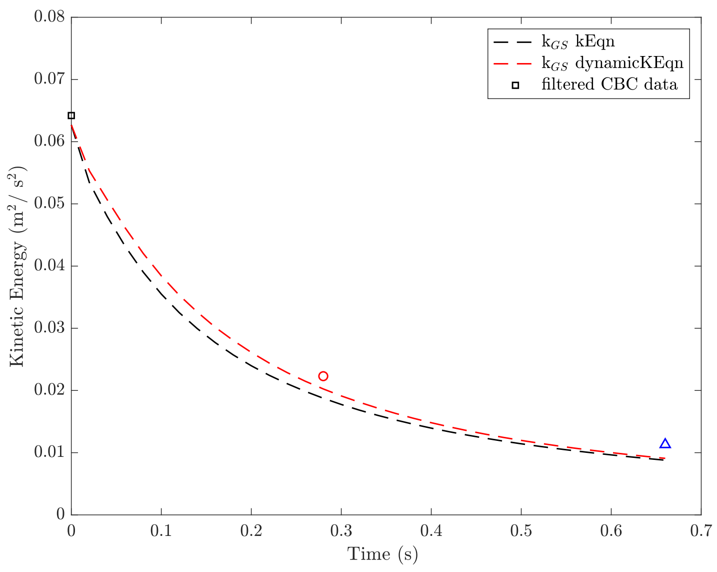


Figure 4.2: Temporal variations of global GS turbulent kinetic energy. Comparison between the filtered CBC data (symbols) and the numerical solutions obtained with the *kEqn* (black dashed line) and *dynamicKEqn* (red dashed line) models.

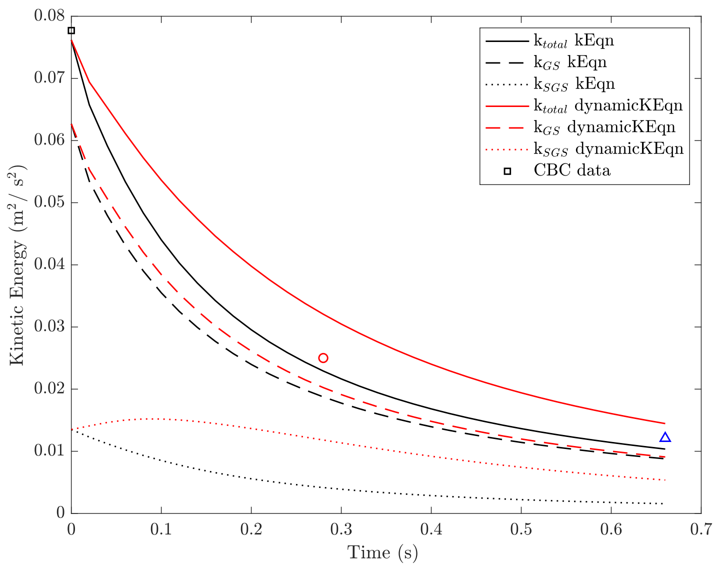


Figure 4.3: Temporal variations of global GS and SGS turbulent kinetic energy. Comparison between the (unfiltered) CBC data (symbols) and the numerical solutions obtained with the *kEqn* (black lines) and *dynamicKEqn* (red lines) models. For the LES curves: the dashed line corresponds to GS kinetic energy, $\overline{(\widetilde{u}^2/2)}$; the dotted line corresponds to SGS kinetic energy, $\overline{k_{SGS}}$; and the solid line corresponds to total kinetic energy, $(\overline{(\widetilde{u}^2/2)} + \overline{k_{SGS}})$.

4.6.2 *Power Spectra*

A more detailed comparison between the CBC experimental data and the numerical results is presented in Figure 4.4 which compares the measured and simulated spectral distribution $E(\kappa)$ at times $t = 0$, 0.28 and 0.66 ms. The measured variations $E(\kappa)$

are taken from Table 4.1; the simulated variations are obtained through a MATLAB-based discrete Fourier transform algorithm applied to the instantaneous LES velocity field. Figure 4.4 suggests that the errors accumulate over time in the κ-range corresponding to the highest resolved wavenumbers, (*i.e.*, for κ lower than, but close to κ_c).

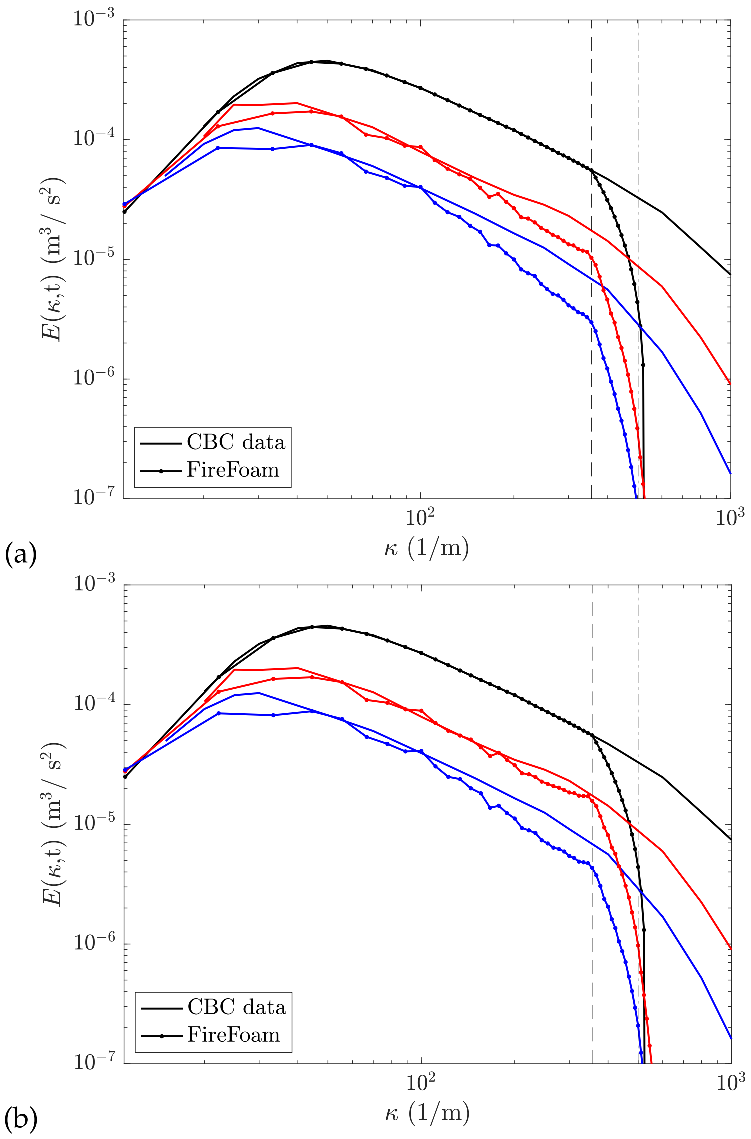


Figure 4.4: Spectral variations of the turbulent kinetic energy. Comparison between the CBC data (solid lines) and the numerical solutions (dotted lines) obtained with: (a) the *kEqn* model; (b) the *dynamicKEqn*. The comparisons are made at time $t = 0$ (black lines), 0.28 ms (red lines) and 0.66 ms (blue lines). The dashed vertical lines in the plots indicate the Nyquist wavenumber κ_c and the modified cut-off value ($\sqrt{2} \times \kappa_c$) proposed in [21].

4.7 CONCLUSION

The CBC case is a classical flow configuration that reproduces the kinetic energy transfer from large to small scales observed in turbulent flows and that allows meaningful tests of the effects of subgrid-scale models on the overall flow dynamics. The evolution of grid-scale flow kinetic energy provides a valuable metric to compare different subgrid-scale models.

5 CASE 4: BUOYANT PLUMES

5.1 CASE DESCRIPTION

This chapter is considered an in-progress case. At this time we have not been able to reach a final version for the Bouyant Plumes (mass conservation case). On this configuration a fluid (Helium or Methane) is injected through an inlet patch, the flow rate is measured at different height and at the outlet patch. Our goal is to create a simple 2D case that produces consistent metrics for evaluating the role of the number of nOuterCorrectors (PIMPLE algorithm) in enforcing mass conservation.

5.2 PRE PROCESSING

5.2.1 *Geometry: Domain and Mesh generation*

For all previous cases the domain consisted of a cube which was discretized in regular, simple grading hexahedral elements by applying the blockMesh command. However, for this case refinement regions and a geometry for the 'burner' has to be created, thus, new commands are presented. Two different approaches are recommended: one is the presented on the OpenFoam tutorial *Breaking of a dam*[4], the other which will be used on this cases is based on the Multiphase modeling (VOF) tutorial developed by Jozsef Nagy. [37]

The first step to build the mesh is generating the STL surfaces that will represent the different patches. The STL files have been placed on the constant/triSurface directory. By running the command surfaceFeatureExtract the surfaces features will be extracted and written to .eMesh files. On the terminal, run the command on the CASE4 folder

```
user@hostname/CASE4: surfaceFeatureExtract
```

The user can check the files that were created on the triSurface directory. The next step is to create the domain by running the command blockMesh

```
user@hostname/CASE4: blockMesh
```

Finally, the SnappyHexMesh folder will create the refinement regions and remove the unnecessary cells. For further detailed, look the commented dictionary SnappyHexMeshDict. Since this tutorial is not focused on the OpenFoam meshing tool, the user is advised to check the section 5.4 of the OpenFoam user guide [4] and the Multiphase modeling (VOF) tutorial [37]

```
user@hostname/CASE4: snappyHexMesh -overwrite
```

5.2.2 *Inlet: boundary conditions for U and He*

In this case the helium (or methane) flow is injected from a surface (patch called inlet) that has the same area as the burner used by McCaffrey in his experiments about Purely buoyant diffusion flames.[38] The mass flow for this case is declared in the U dictionary in the 0 folder(~/CASE4/0/), the type of boundary condition used is *flowRateInletVelocity* the value is specified by using the *massFlowRate* in kg/s, see Listing 5.1.

Listing 5.1: Initial Conditions for the Velocity Field

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  dev                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [0 1 -1 0 0 0 0];
internalField   uniform (0 0 0);

boundaryField
{
    outlet
    {

        type            inletOutlet;
        value           $internalField;
        inletValue      uniform (0 0 0);
    }
    sides
    {
        type            pressureInletOutletVelocity;
        value           $internalField;
    }
    base
    {
         type            fixedValue;
        value           uniform (0 0 0);
    }
    inlet
    {
        type            flowRateInletVelocity;
        massFlowRate    0.1;
        value           uniform (0 0 0);
    }
}
// ************************************************************************* //
```

The composition of the mixture injected is defined in the species' directories, for this case only one specie is introduced in the system (He), the boundary condition is define in the *boundaryField* section, the type selected is *totalFlowRateAdvectiveDiffusive* and a mass flux fraction of 1. Total advective diffusive means that the mass flux fraction is calculated by taking into account both transport phenomena, see Equation 5.1 and Equation 5.2

$$\frac{totalFlowRateAdvectiveDiffusive}{massFlowRate} = 1 \tag{5.1}$$

Listing 5.2: He mass fraction

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  dev                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      He;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    outlet
    {
        type            inletOutlet;
        inletValue      $internalField;
        value           $internalField;
    }
    sides
    {
        type            inletOutlet;
        inletValue      $internalField;
        value           $internalField;
    }
    base
    {
        type            zeroGradient;
    }
    inlet
    {
        type            totalFlowRateAdvectiveDiffusive;
        massFluxFraction 1;
        phi                 phi;
        rho                 rho;
        value               uniform 1;
    }
}
// ************************************************************************* //
```

5.3 PROCESSING

A default field called phiFt is calculated by FireFoam *on the fly* it corresponds to $\left(\bar{\rho}\tilde{u}_z\tilde{Z} - \bar{\rho}\left(D + D_t\right)\frac{\partial\tilde{Z}}{\partial z}\right)$ the instantaneous mass flow rate through a given face, the total mass flow rate at a specific height can be obtained through numerical integration of *phiFt* over the cells at that height (faceZone). The *faceZone* is created with the command *topoSet* (see topoSet dictionary) and the summation of all the flows is calculated with the *sum* operation, the latter is set in the *controlDict* dictionary. As mentioned before, a expression for *phiFt* exists by default, nevertheless, for the helium plume the field *phiFu* $\left(\bar{\rho}\tilde{u}_z\tilde{Y}_{He} - \bar{\rho}\left(D + D_t\right)\frac{\partial\tilde{Y}_{He}}{\partial z}\right)$ was created in *infoFieldsOutput.H* and in *infoOutput.H* by using the variable *Fu* (Fuel mass fraction) and He is declared as Fuel in the *reactions* dictionary.

This simulation is run in parallel mode as the previous cases. However, the decompose method employed is the *scotch* rather the *simple* method, for the *scotch* method only the number of processors has to be set in the *decomposerPar* dictionary, an algorithm balances the number of cells per processor and the number of neighbouring cells to get maximum efficiency, a rule of thumb is 1 processor per 10 000 to 15 000 cells.

5.4 POST PROCESSING

The mass conservation is evaluated by comparing the time averaged mass flow rate at different heights with the injected mass flow, for the Helium Plume configuration see Equation 5.2 and Equation 5.3 ; for the combustion case see Equation 5.4 to Equation 5.5.

$$\dot{m}_{He}\left(t,z\right) = \int\int\left(\bar{\rho}\tilde{u}_z\tilde{Y}_{He} - \bar{\rho}\left(D + D_t\right)\frac{\partial\tilde{Y}_{He}}{\partial z}\right)dxdy \tag{5.2}$$

$$\overline{\dot{m}}_{He}\left(z\right) = \frac{1}{T-t}\int_t^T \dot{m}_{He}\left(t,z\right)dt \tag{5.3}$$

$$\dot{m}_{Z}\left(t,z\right) = \int\int\left(\bar{\rho}\tilde{u}_z\tilde{Z} - \bar{\rho}\left(D + D_t\right)\frac{\partial\tilde{Z}}{\partial z}\right)dxdy \tag{5.4}$$

$$\overline{\dot{m}}_{Z}\left(z\right) == \frac{1}{T-t}\int_t^T \dot{m}_{Z}\left(t,z\right)dt \tag{5.5}$$

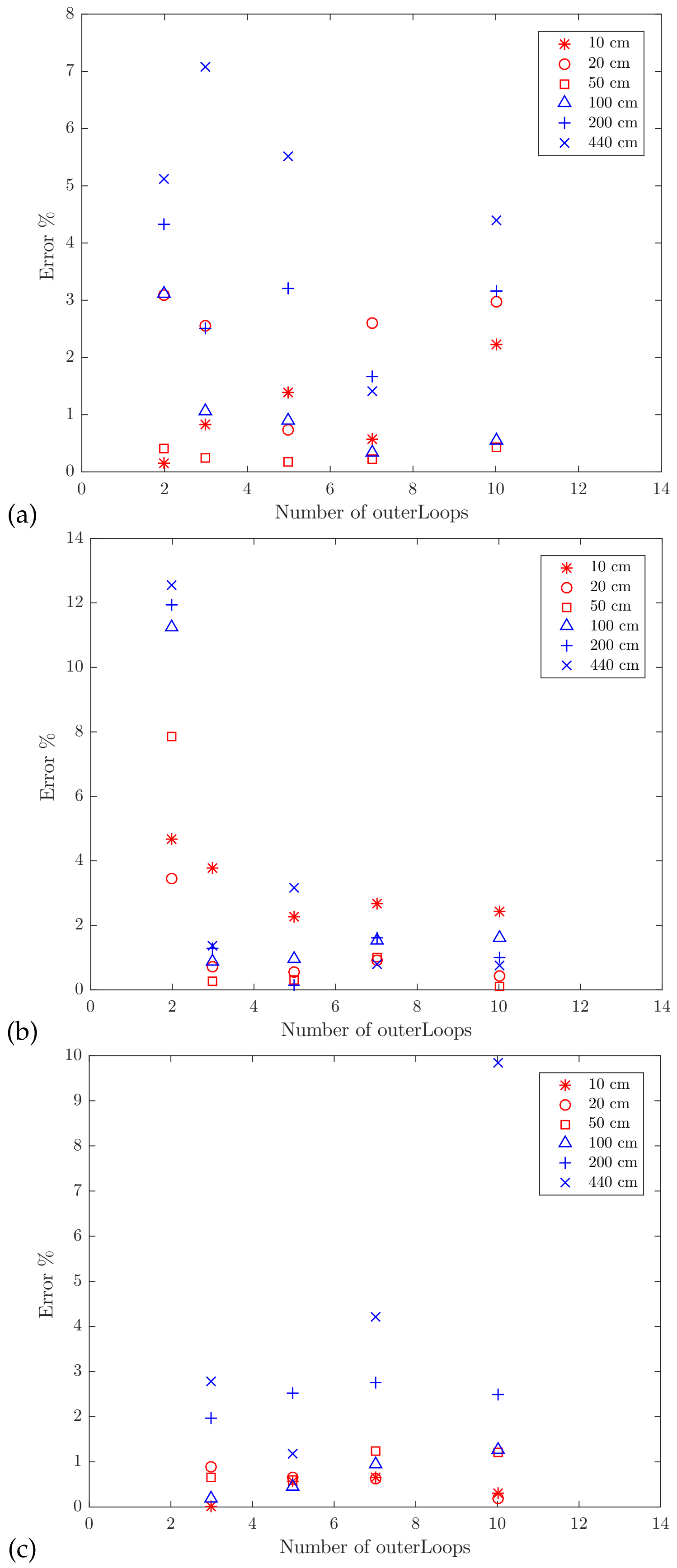
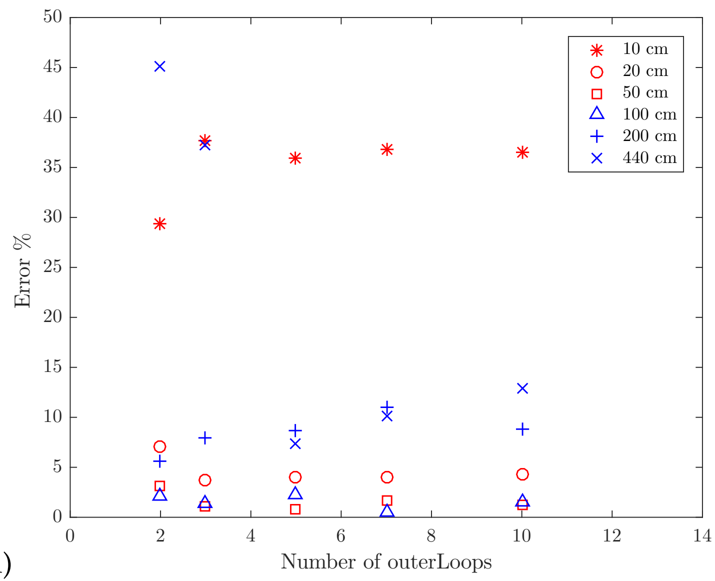
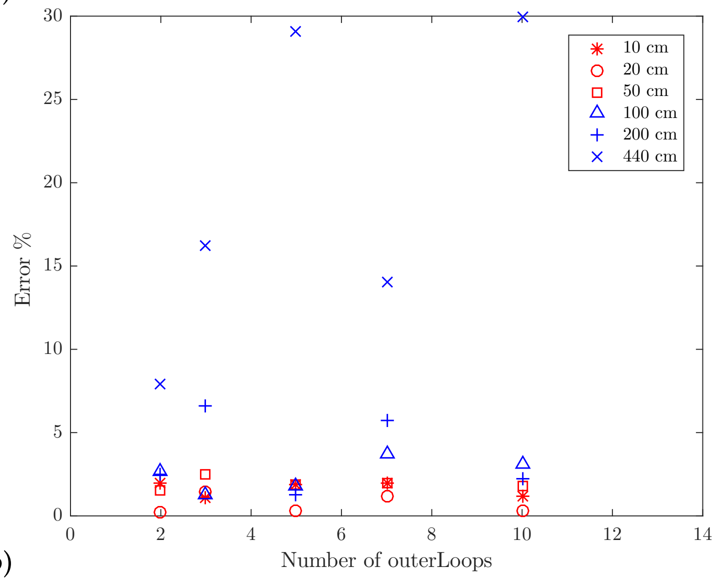


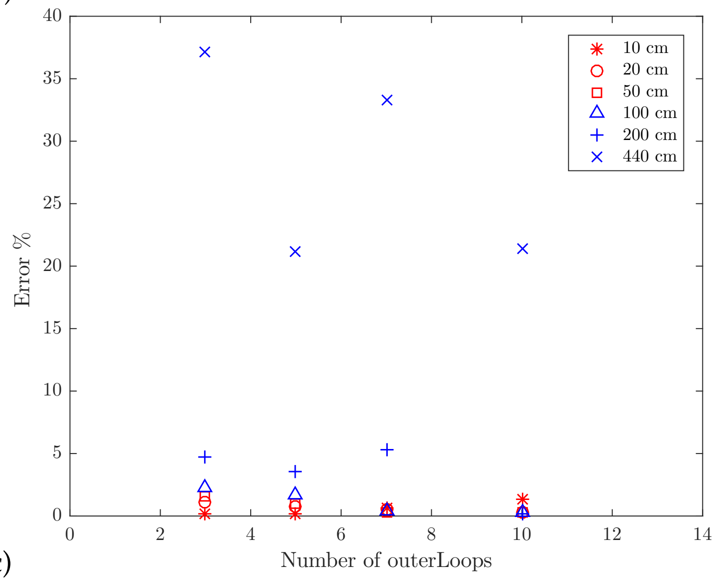
Figure 5.1: Mass flow rate error percentage with different nOuterCorrector (outerLoops) at different heights for inlet for inlet injection velocity $u_z = 10cm/s$ with 3 different resolutions (a) $D/\Delta x = 6$; (b) $D/\Delta x = 12$; (c) $D/\Delta x = 24$.



(a)

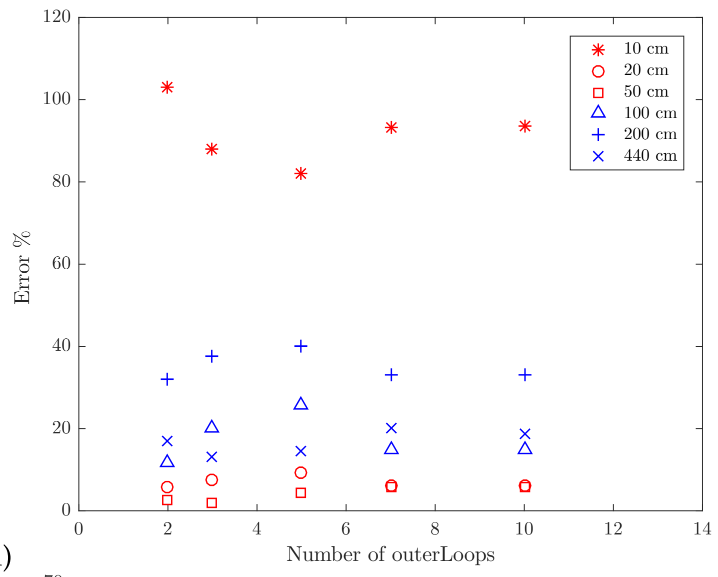


(b)

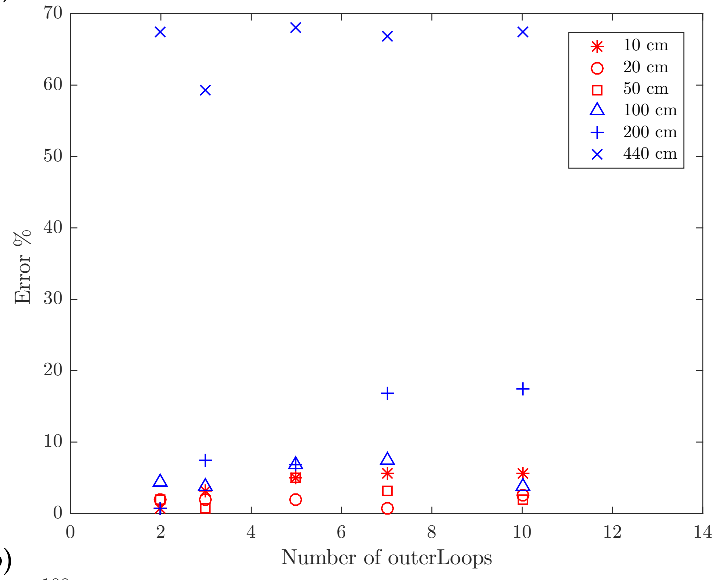


(c)

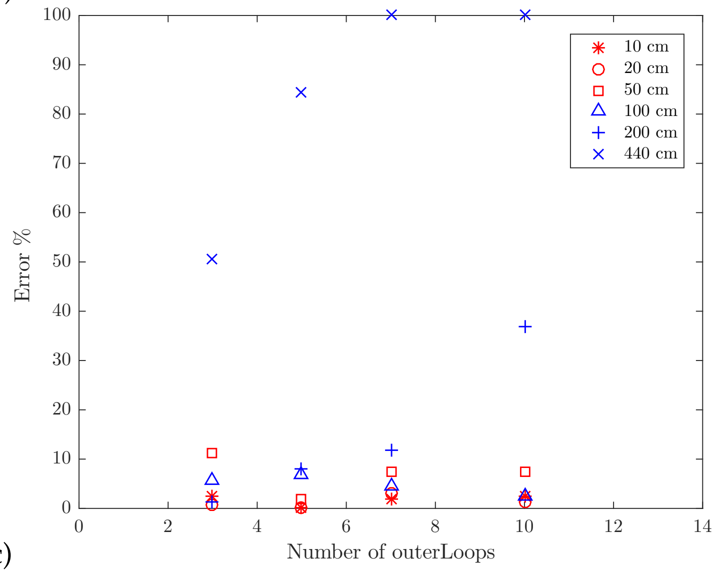
Figure 5.2: Mass flow rate error percentage with different nOuterCorrector (outerLoops) at different heights for inlet injection velocity $u_z = 1cm/s$ with 3 different resolutions (a) $D/\Delta x = 6$; (b) $D/\Delta x = 12$; (c) $D/\Delta x = 24$.



(a)



(b)



(c)

Figure 5.3: Mass flow rate error percentage with different nOuterCorrector (outerLoops) at different heights for inlet injection velocity $u_z = 1mm/s$ with 3 different resolutions (a) $D/\Delta x = 6$; (b) $D/\Delta x = 12$; (c) $D/\Delta x = 24$.

The results in Figure 5.1 to Figure 5.3 show that the error increases considerable over the height, which evidences a problem on the case configuration. Most of the tests have been done for the Helium Plume, as this one is supposed to be a simpler case (given its inert nature) than the combustion one(2D-McCaffrey Plume). Some problems have been detected on the current set up: firstly, the Helium seems to stay in the domain for long periods; in the highest velocity case some instabilities create re-circulation of helium inside the domain (see Figure 5.4), the helium diffuses into air and loses buoyancy, for the lowest velocity the helium loses bouyancy due to air entrainment. Two solutions have been proposed to reduce these effects: the addition of an air-coflow which could drag the helium out of the domain; also a reduction of the domain size in $z - direction$ has been considered. Secondly, some of the helium seems to be escaping from the domain by the sizes; a change of the side boundary conditions to a periodic boundary will prevent mass losses through these patches. In Figure 5.5 the mass flow rate at highest planes is not considered the errors for the $10cm, 20cm, 50cm$ and $100cm$ is averaged and condensed is a single parameter. However, the results do not show the expected trend.

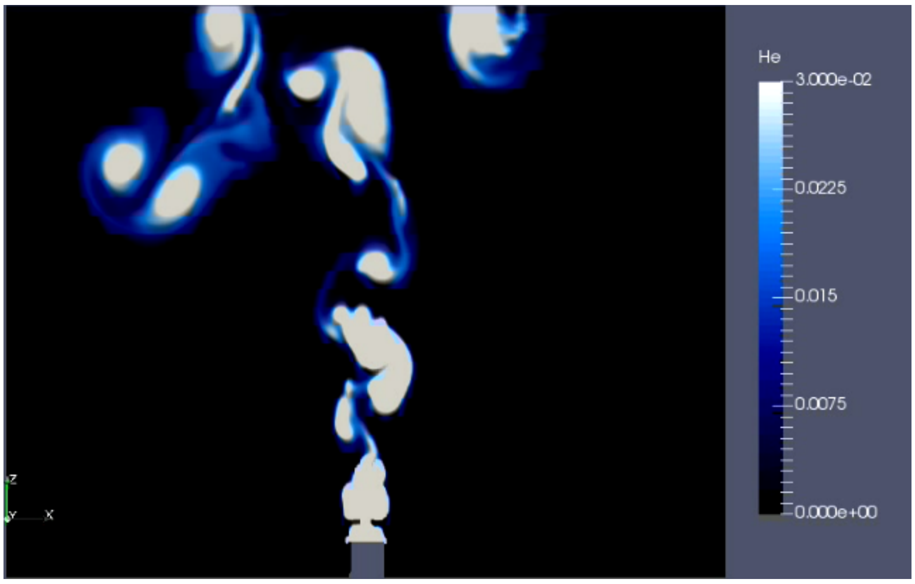
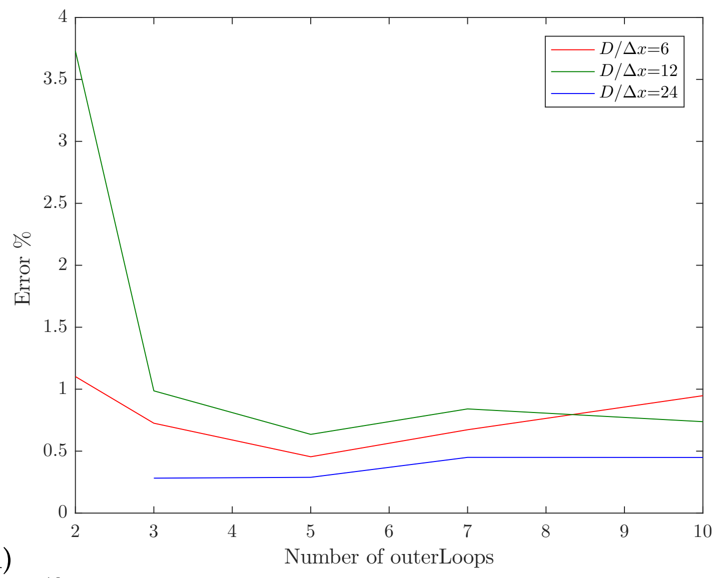


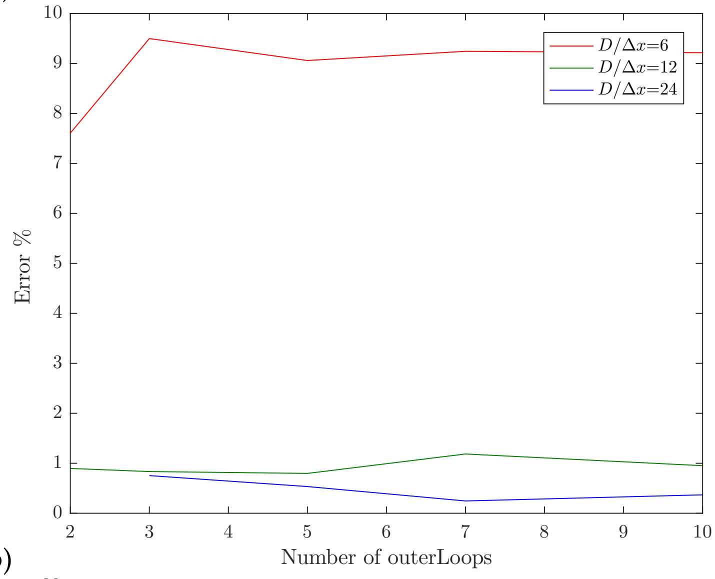
Figure 5.4: Helium Plume colour by YHe instabilities generates vortical structures that moves to the sides, which produces loses throught the sides

5.5 CONCLUSION

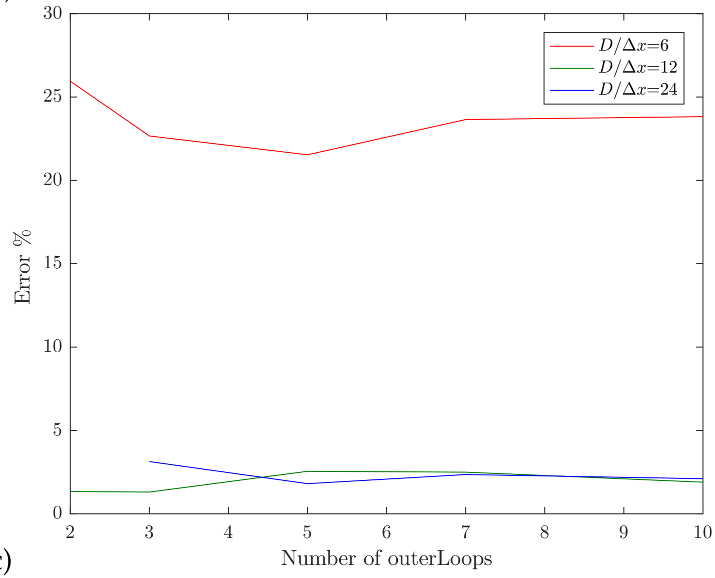
The buoyant plume is a fire related case that aims to produce metrics for evaluating the influence of the PIMPLE algorithm in the conservation of mass. We have set the basis for a configuration, however, at this point we have not been capable to create a mature set up for Case 4.



(a)



(b)



(c)

Figure 5.5: Averaged error percentage calculated with the mass flow rates at 10 cm, 20 cm, 50 cm and 100 cm height from the inlet, for different inlet injection velocities: (a) $u_z = 10cm/s$; (b) $u_z = 1cm/s$; (c) $u_z = 10mm/s$.

5.6 FUTURE WORK

More work needs to be done in the Bouyant Plume configuration which at this moment is considered an in-progress case.

This thesis starts from a simple laminar 2D case (Lamb–Oseen Vortex) and it gradually moves towards more complex cases, by adding different features and continually testing FireFoam capabilities. Nevertheless, these cases only evaluate the gas phases. Hence, it is important to create a set of cases for the radiation and pyrolysis models. Furthermore, it can be considered that the cases presented in this thesis follow a building block approach, thus, it could be extended by adding some of the MaCFP Working Group Cases.

Part III

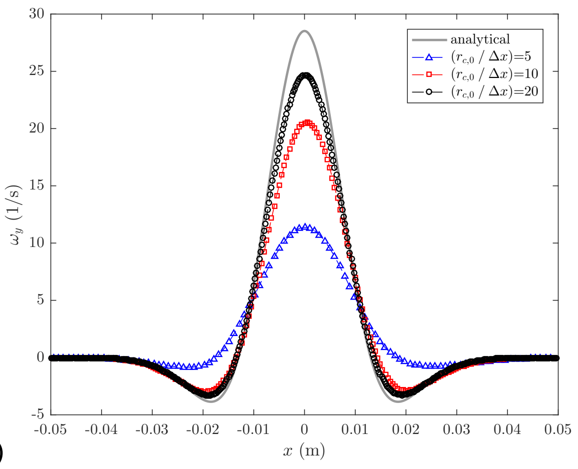
APPENDIX

APPENDIX CASE1

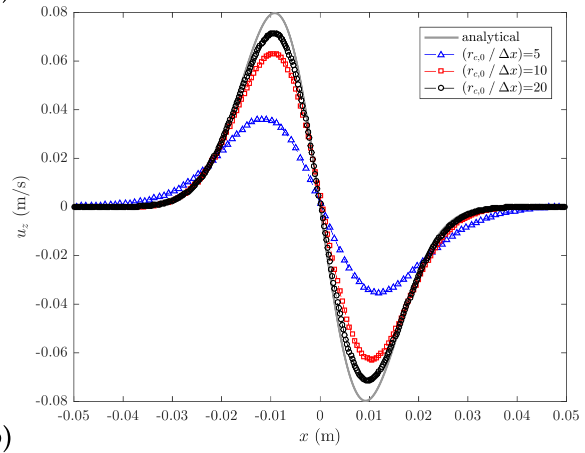
A

A.1 ADDITIONAL RESULTS WITH AN ADJUSTABLE TIME STEP CONTROLLED BY A CFL NUMBER

A.1.1 *Euler time scheme*

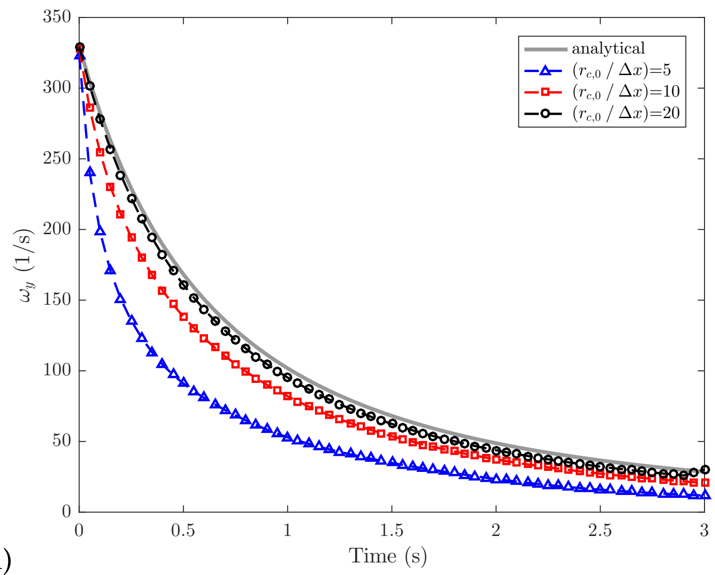


(a)

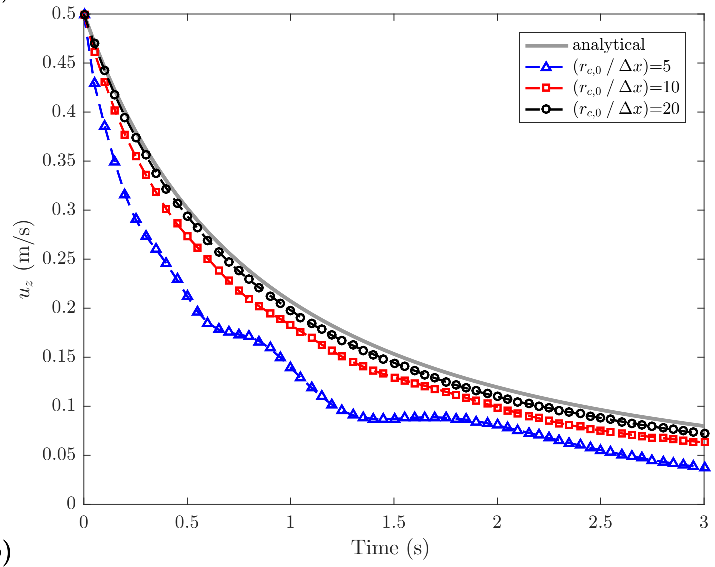


(b)

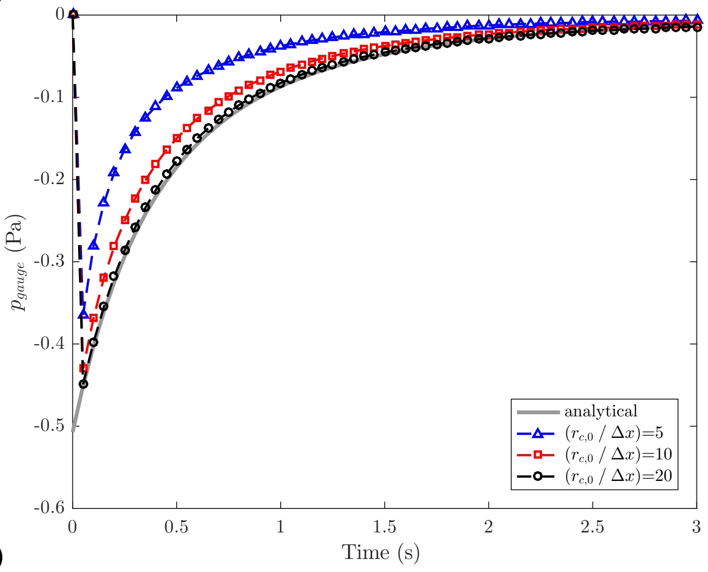
Figure A.1: Spatial variations of: (a) ω_y; and (b) u_z, along $z = 0$ at time $t = 3$ s. Comparison between the analytical solution and the numerical solutions obtained with three levels of grid resolution, $(r_{c,0}/\Delta x) = 5$, 10 and 20. Simulations performed using the *euler* time integration scheme and an adjustable time step controlled by a CFL number of 0.5.



(a)



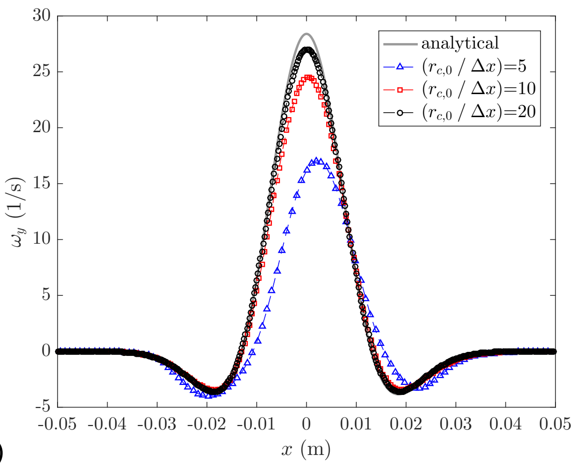
(b)



(c)

Figure A.2: Temporal variations of: (a) the maximum value of ω_y; (b) the maximum value of u_z; (c) the minimum value of p_{gauge}. Comparison between the analytical solution and the numerical solutions obtained with three levels of grid resolution, $(r_{c,0}/\Delta x) = 5$, 10 and 20. Simulations performed using the *euler* time integration scheme and an adjustable time step controlled by a CFL number of 0.5.

A.1.2 *Crank-Nicolson scheme*

(a)


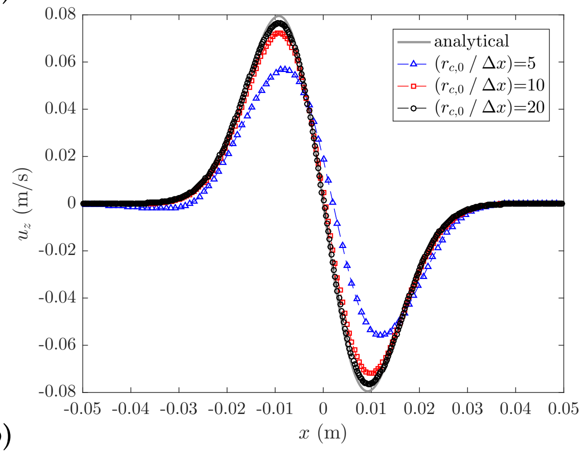
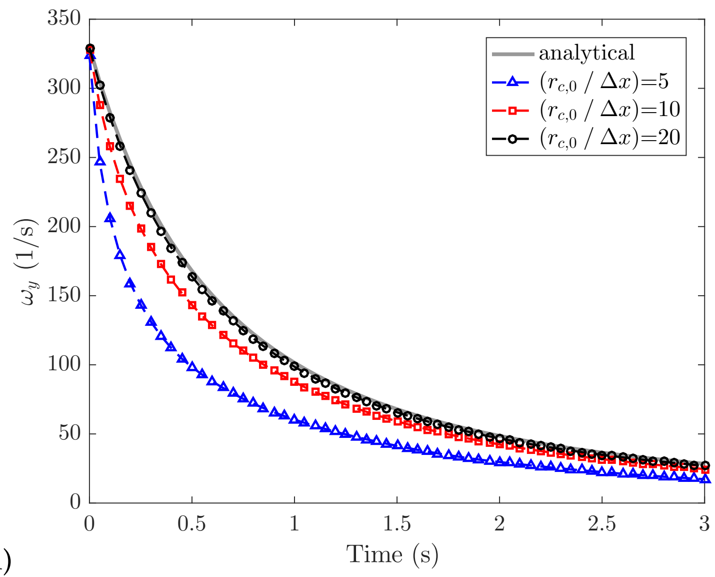
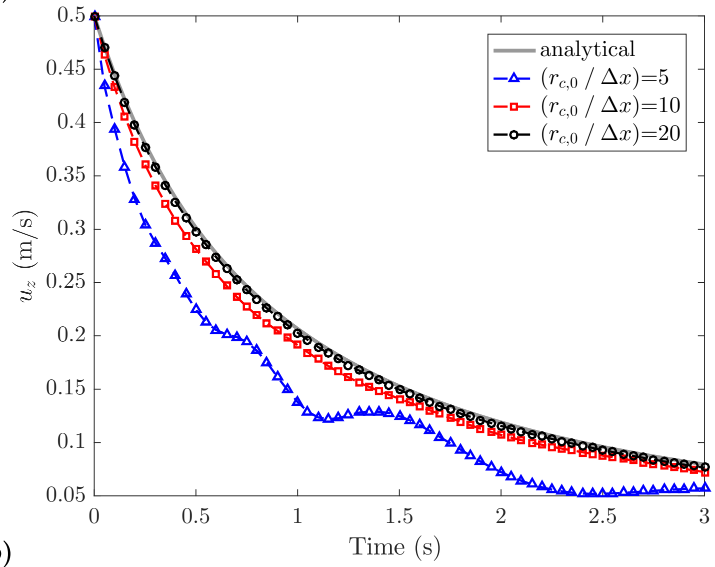
(b)


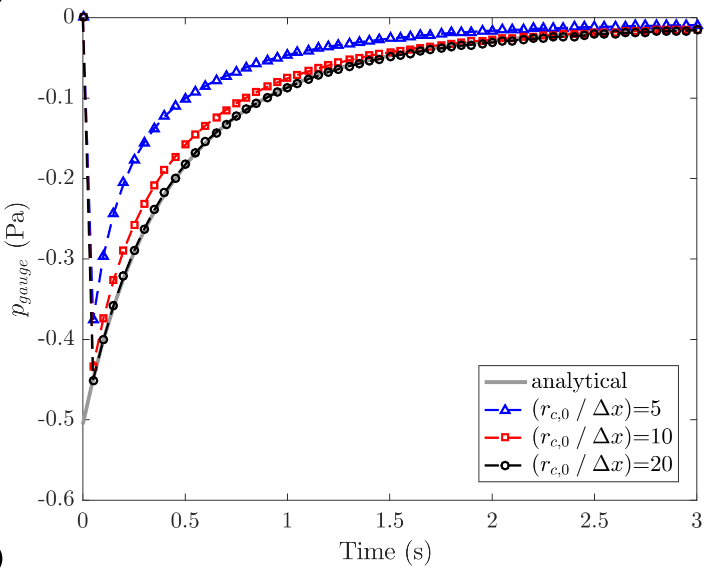
Figure A.3: Spatial variations of: (a) ω_y; and (b) u_z, along $z = 0$ at time $t = 3$ s. Comparison between the analytical solution and the numerical solutions obtained with three levels of grid resolution, $(r_{c,0}/\Delta x) = 5$, 10 and 20. Simulations performed using the *Crank-Nicolson 0.5* time integration scheme and an adjustable time step controlled by a CFL number of 0.5.



(a)



(b)



(c)

Figure A.4: Temporal variations of: (a) the maximum value of ω_y; (b) the maximum value of u_z; (c) the minimum value of p_{gauge}. Comparison between the analytical solution and the numerical solutions obtained with three levels of grid resolution, $(r_{c,0}/\Delta x) = 5$, 10 and 20. Simulations performed using the *Crank-Nicolson 0.5* time integration scheme and an adjustable time step controlled by a CFL number of 0.5.

A.2 ADDITIONAL RESULTS WITH CONSTANT TIME STEP $\Delta t = 10^{-5}$ S

A.2.1 *Backward time scheme*

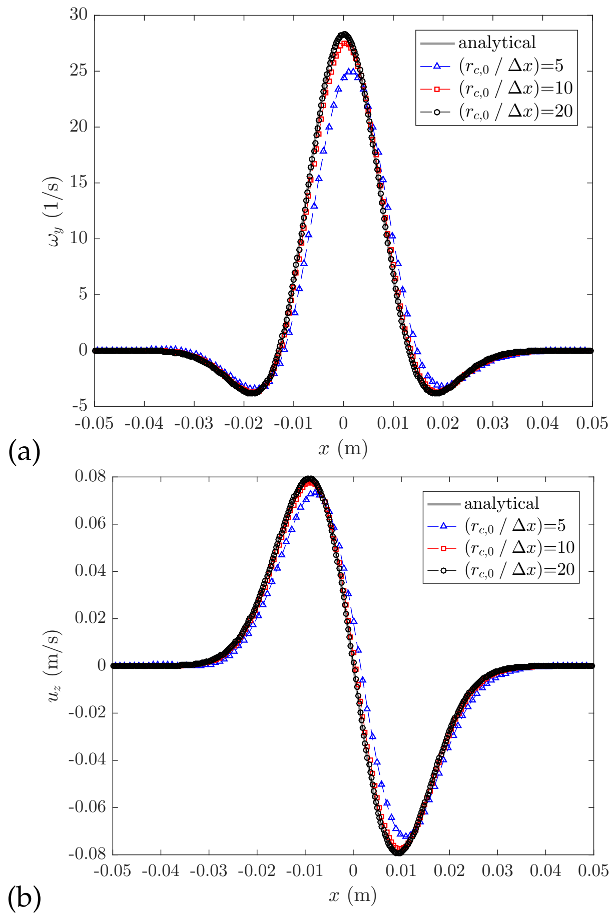
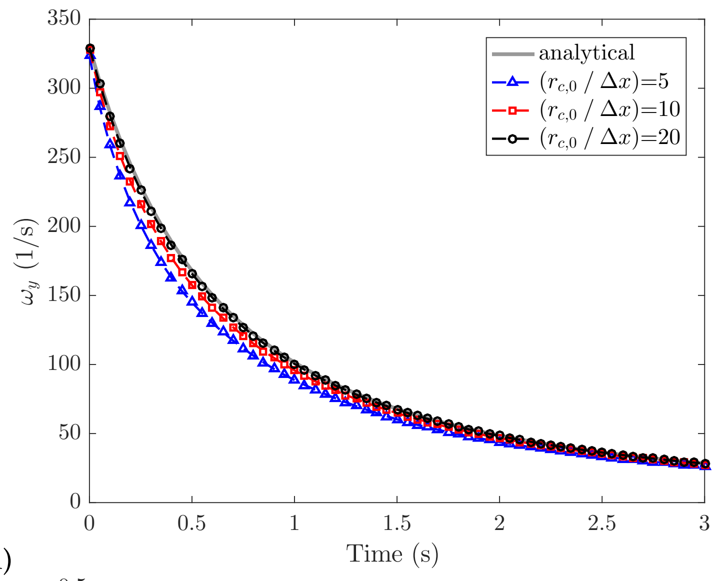
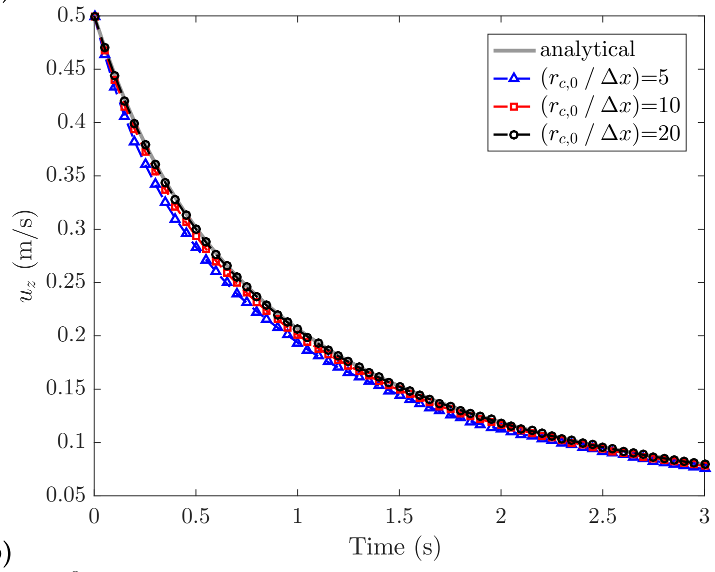


Figure A.5: Spatial variations of: (a) ω_y; and (b) u_z, along $z = 0$ at time $t = 3$ s. Comparison between the analytical solution and the numerical solutions obtained with three levels of grid resolution, $(r_{c,0}/\Delta x) = 5$, 10 and 20. Simulations performed using the *Crank-Nicolson* time integration scheme and an constant time step $\Delta t = 10^{-5}$ s.





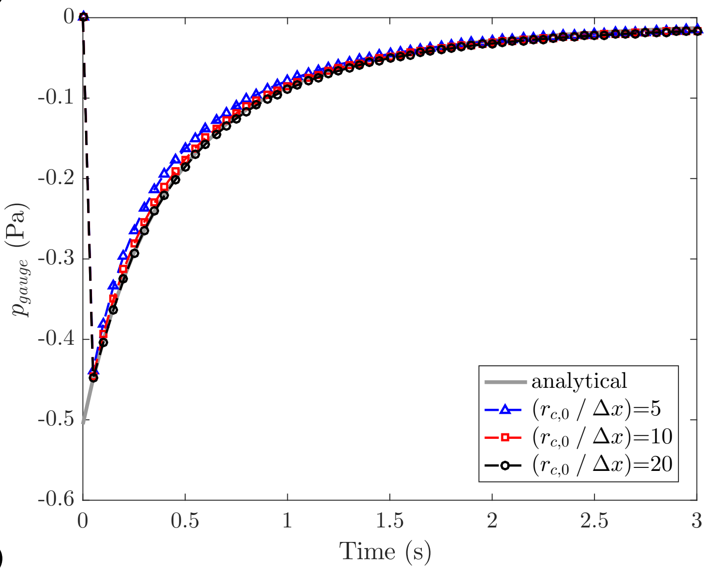
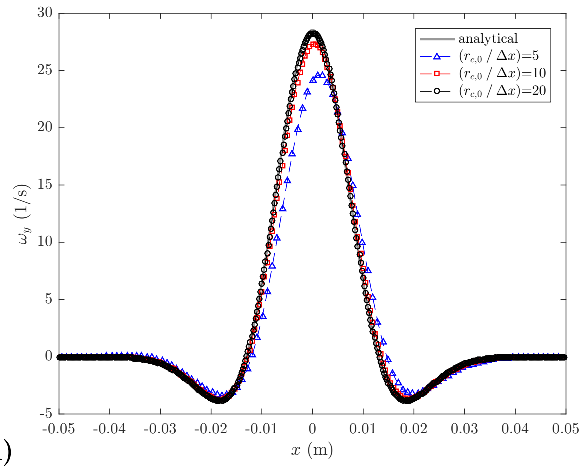


Figure A.6: Temporal variations of: (a) the maximum value of ω_y; (b) the maximum value of u_z; (c) the minimum value of p_{gauge}. Comparison between the analytical solution and the numerical solutions obtained with three levels of grid resolution, $(r_{c,0}/\Delta x) = 5$, 10 and 20. Simulations performed using the *Crank-Nicolson* time integration scheme and an constant time step $\Delta t = 10^{-5}$ s.

A.2.2 *Euler time scheme*

(a)



(b)

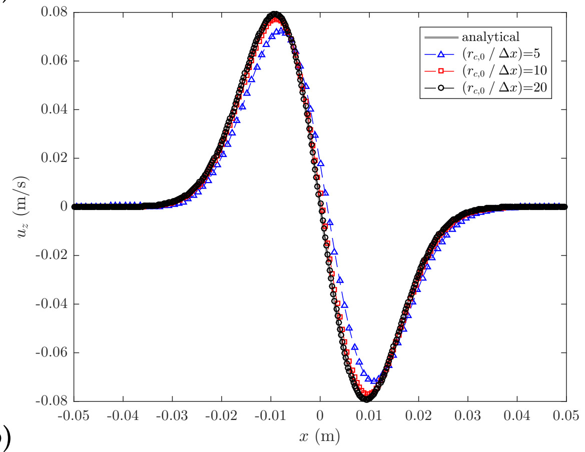
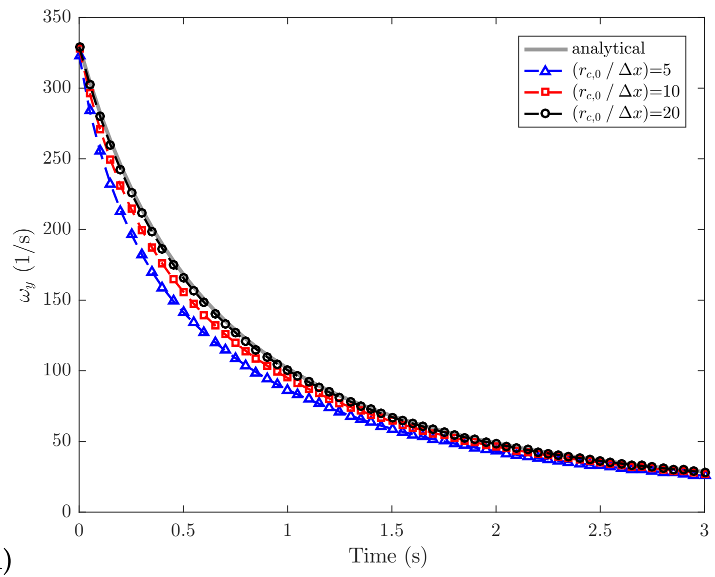
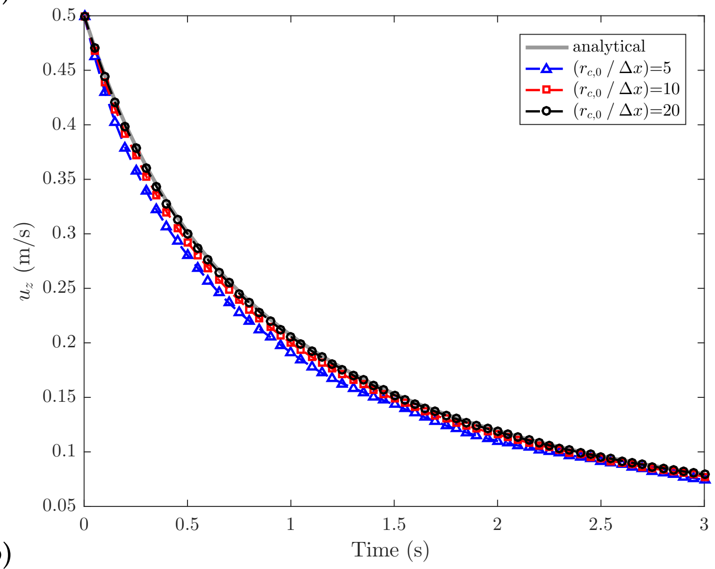


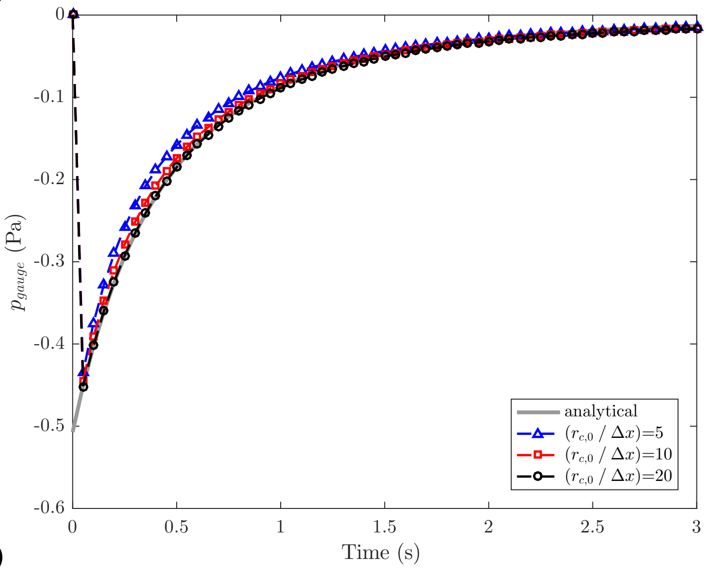
Figure A.7: Spatial variations of: (a) ω_y; and (b) u_z, along $z = 0$ at time $t = 3$ s. Comparison between the analytical solution and the numerical solutions obtained with three levels of grid resolution, $(r_{c,0}/\Delta x) = 5$, 10 and 20. Simulations performed using the *euler* time integration scheme and an constant time step $\Delta t = 10^{-5}$ s.



(a)



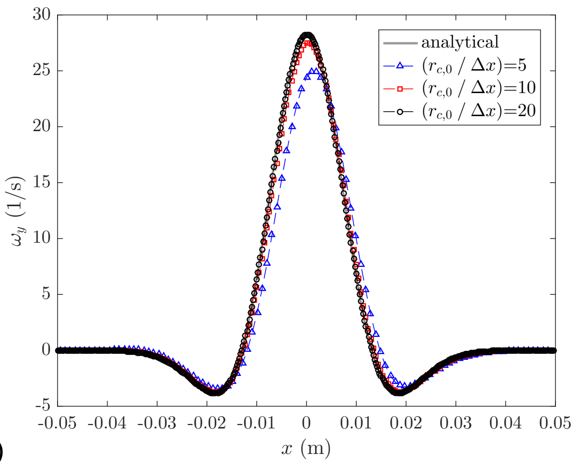
(b)



(c)

Figure A.8: Temporal variations of: (a) the maximum value of ω_y; (b) the maximum value of u_z; (c) the minimum value of p_{gauge}. Comparison between the analytical solution and the numerical solutions obtained with three levels of grid resolution, $(r_{c,0}/\Delta x) = 5$, 10 and 20. Simulations performed using the *euler* time integration scheme and an constant time step $\Delta t = 10^{-5}$ s.

A.2.3 *Crank-Nicolson time scheme*

(a)


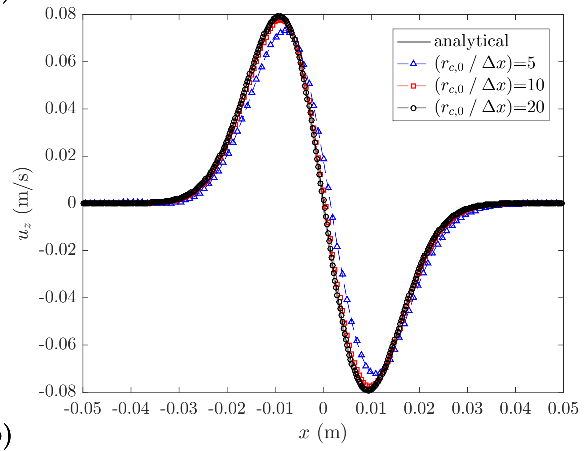
(b)


Figure A.9: Spatial variations of: (a) ω_y; and (b) u_z, along $z = 0$ at time $t = 3$ s. Comparison between the analytical solution and the numerical solutions obtained with three levels of grid resolution, $(r_{c,0}/\Delta x) = 5$, 10 and 20. Simulations performed using the *Crank-Nicolson* time integration scheme and an constant time step $\Delta t = 10^{-5}$ s.

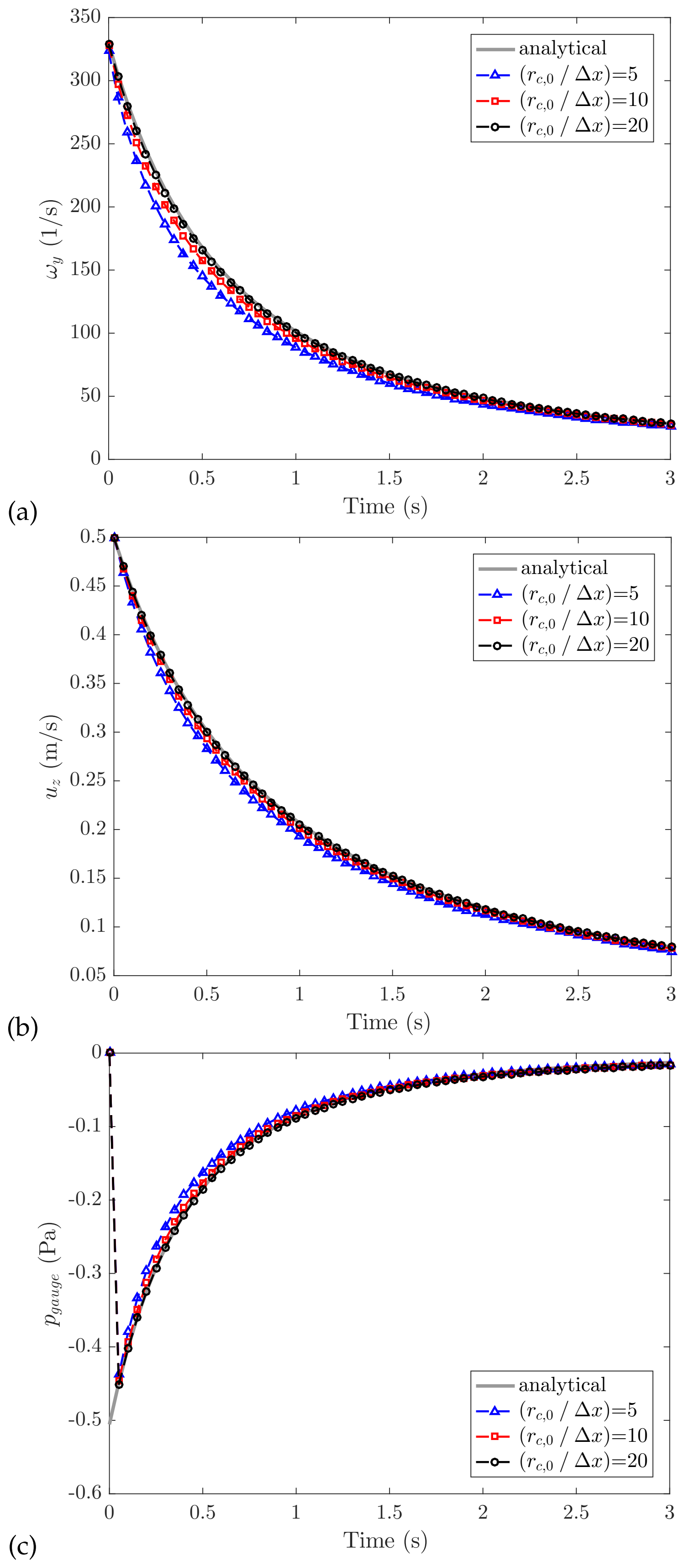


Figure A.10: Temporal variations of: (a) the maximum value of ω_y; (b) the maximum value of u_z; (c) the minimum value of p_{gauge}. Comparison between the analytical solution and the numerical solutions obtained with three levels of grid resolution, $(r_{c,0}/\Delta x) = 5$, 10 and 20. Simulations performed using the *Crank-Nicolson* time integration scheme and an constant time step $\Delta t = 10^{-5}$ s.

A.3 SIMULATION FILES

The CASES' folders with all necessary files for running the simulations are provided in the UserGuide.zip file.

BIBLIOGRAPHY

1. Lamb, H. *Hydrodynamics* (Cambridge University Press, Cambridge, 1932).
2. FM Global. *Open Source Fire Modeling* [Online; accessed 10-April-2018]. `https://www.fmglobal.com/research-and-resources/research-and-testing/theoretical-computational-and-experimental-research/open-source-fire-modeling`.
3. FM Global. *FM Global Open Source CFD Fire Modeling Workshop* [Online; accessed 10-April-2018]. `https://sites.google.com/site/firemodelingworkshop/`.
4. Greenshields, C. J. *OpenFOAM User Guide* version v5. [Online; accessed 2-February-2018]. CFD Direct Ltd. `https://cfd.direct/openfoam/user-guide/` (2018).
5. OpenFOAM-Wiki. *Tutorials — OpenFOAM Wiki,* [Online; accessed 20-January-2018]. 2018. `https://wiki.openfoam.com/index.php?title=Tutorials&oldid=2164`.
6. Oseen, C. W. Über wirbelbewegung in einer reiben den flüssigkeit. *Arkiv för matematik, astronomi och fysik* **7,** 14–21 (1912).
7. Moukalled, F, Mangani, L, Darwish, M, *et al.* The finite volume method in computational fluid dynamics (2016).
8. OpenCFD Ltd. *Extended Code Guide* version v1712. [Online; accessed 2-February-2018]. `https://www.openfoam.com/documentation/cpp-guide/html/openfoam-guide.html` (2018).
9. Wang, Z. J., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H. T., *et al.* High-order CFD methods: current status and perspective. *International Journal for Numerical Methods in Fluids* **72,** 811–845 (2013).
10. Diosady, L & Murman, S. in *Tech. Rep.* (NASA Ames Research Center, CA, USA, 2015).
11. 5th International Workshop on High Order CFD Methods. [Online; accessed 2-June-2018]. `https://how5.cenaero.be/`.
12. Johnsen, E. *et al.* Assessment of high-resolution methods for numerical simulations of compressible turbulence with shock waves. *Journal of Computational Physics* **229,** 1213 –1237. ISSN: 0021-9991 (2010).
13. DeBonis, J. *Solutions of the Taylor-Green vortex problem using high-resolution explicit finite difference methods* in *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition* (2013), 382.
14. Fauconnier, D., De Langhe, C. & Dick, E. *Quality assessment of Dynamic Finite Difference schemes on the Taylor-Green Vortex* in *Quality and Reliability of Large-Eddy Simulations II* (eds Salvetti, M. V., Geurts, B., Meyers, J. & Sagaut, P.) (Springer Netherlands, Dordrecht, 2011), 287–296. ISBN: 978-94-007-0231-8.

15. Brachet, M. E., Meiron, D. I., Orszag, S. A., Nickel, B. G., Morf, R. H. & Frisch, U. Small-scale structure of the Taylor–Green vortex. *Journal of Fluid Mechanics* **130,** 411–452 (1983).

16. Rusche, H. *Computational fluid dynamics of dispersed two-phase flows at high phase fractions* PhD thesis (Imperial College London (University of London), 2003).

17. Comte-Bellot, G. & Corrsin, S. Simple Eulerian time correlation of full-and narrow-band velocity signals in grid-generated,'isotropic'turbulence. *Journal of Fluid Mechanics* **48,** 273–337 (1971).

18. Pope, S. B. *Turbulent Flows* doi:10.1017/CBO9780511840531 (Cambridge University Press, 2000).

19. McDermott, R. J. *Toward one-dimensional turbulence subgrid closure for large-eddy simulation* (Department of Chemical Engineering, University of Utah, 2005).

20. McGrattan, K. B., McDermott, R. J., Weinschenk, C. G. & Forney, G. P. *Fire dynamics simulator: Technical Reference Guide Volume 2: Verification* tech. rep. (2013).

21. McDermott, R. J. Private Communication. Randy explained the relationship between the filtered energy spectrum and the unfiltered spectrum, using the attenuation factor he presented on his talk in the 58th Annual Meeting of the Division of Fluid Dynamics: Characteristics of 1d spectra in finite-volume large-eddy simulations with 'one-dimensional turbulence' subgrid closure. College Park, MD, USA, 2018.

22. Yoshizawa, A. Statistical theory for compressible turbulent shear flows, with the application to subgrid modeling. *The Physics of fluids* **29,** 2152–2164 (1986).

23. Kim, W.-W. & Menon, S. *A new dynamic one-equation subgrid-scale model for large eddy simulations* in *33rd Aerospace Sciences Meeting and Exhibit* (1995), 356.

24. Holzmann, T. *Mathematics, Numerics, Derivations and OpenFOAM®* https://holzmann-cfd.de (Holzmann CFD, 2016).

25. OpenFOAM-Wiki. *OpenFoam Guide — The PIMPLE algorithm in OpenFOAM,* [Online; accessed 20-June-2018]. 2018. https://openfoamwiki.net/index.php/OpenFOAM_guide/The_PIMPLE_algorithm_in_OpenFOAM.

26. Division of Information Technology. *The Deepthought2 HPC cluster* [Online; accessed 12-April-2018]. http://hpcc.umd.edu/hpcc/dt2.html.

27. Saffman, P. G. *Vortex Dynamics* 192–209, 253–270. doi:10.1017/CBO9780511624063 (Cambridge University Press, 1993).

28. Tryggeson, H. *Analytical Vortex Solutions to the Navier-Stokes Equation* PhD thesis (Växjö University, Sweden, 2007).

29. Sutherland, W. LII. The viscosity of gases and molecular force. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* **36,** 507–531 (1893).

30. Richardson, L. F. & Lynch, P. *Weather Prediction by Numerical Process* 2nd ed. doi:10.1017/CBO9780511618291 (Cambridge University Press, 2007).

31. Taylor, G. & Green, A. Mechanism of the production of small eddies from large ones. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* **158,** 499–521. ISSN: 0080-4630 (1937).

32. Hunt, J. C., Wray, A. A. & Moin, P. *Eddies, streams, and convergence zones in turbulent flows* in *Proceedings of the 1988 Summer Program* (1988).

33. Haller, G. An objective definition of a vortex. *Journal of Fluid Mechanics* **525,** 1–26 (2005).

34. Von Kármán, T. Progress in the Statistical Theory of Turbulence. *Proceedings of the National Academy of Sciences* **34,** 530–539. ISSN: 0027-8424 (1948).

35. Forney, G. P. *plotspec_uvw.m* [Online; accessed 3-March-2018]. Fire Models, July 2016. `https://github.com/firemodels/fds/blob/master/Verification/Turbulence/cbc64_uvw.csv` (2018).

36. De Bruyn Kops, S. M. & Riley, J. J. Direct numerical simulation of laboratory experiments in isotropic turbulence. *Physics of Fluids* **10,** 2125–2127 (1998).

37. Nagy, J. *Multiphase modeling (VOF)* [Accessed on 04/04/2018]. Institute of Polymer Injection Molding and Process Automation, Johannes Kepler University Linz, Austria, Jan. 2018. `https://wiki.openfoam.com/Multiphase_modeling_(VOF)_by_Jozsef_Nagy` (2018).

38. McCaffrey, B. J. *Purely buoyant diffusion flames: Some experimental results. Final Report* tech. rep. NBSIR 79-1910 (Center for Fire Research, National Engineering Laboratory, National Bureau of Standards, Department of Commerce, Washington, D.C., Oct. 1979), 49.

DECLARATION

This thesis is submitted in partial fulfillment of the requirements for the degree of The International Master of Science in Fire Safety Engineering (IMFSE). This thesis has never been submitted for any degree or examination to any other University/programme. The author(s) declare(s) that this thesis is original work except where stated. This declaration constitutes an assertion that full and accurate references and citations have been included for all material, directly included and indirectly contributing to the thesis. The author(s) gives (give) permission to make this master thesis available for consultation and to copy parts of this master thesis for personal use. In the case of any other use, the limitations of the copyright have to be respected, in particular with regard to the obligation to state expressly the source when quoting results from this master thesis. The thesis supervisor must be informed when data or results are used.

Read and approved

College Park, Maryland, May 2018

[illegible]

Sergio Vargas-Córdoba

COLOPHON

This document was written and developed at the Department of Fire Protection Engineering at the University of Maryland, College Park. It was typeset using a slightly modified version of the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić.