UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE GEOFÍSICA

CFD SIMULATIONS OF TURBULENT BUOYANT ATMOSPHERIC FLOWS OVER COMPLEX GEOMETRY. SOLVER DEVELOPMENT IN OPENFOAM AND APPLICATION TO LARGE OPEN PIT MINES

TESIS PARA OPTAR AL GRADO DE DOCTOR EN CIENCIAS DE LA INGENIERÍA, MENCIÓN FLUIDODINÁMICA

FEDERICO ESTEBAN FLORES MENESES

PROFESOR GUÍA:
RENÉ GARREAUD SALAZAR

PROFESOR CO-GUÍA:
RICARDO MUÑOZ MAGNINO

MIEMBROS DE LA COMISIÓN:
ALVARO VALENCIA MUSALEM
RODRIGO ESCOBAR MORAGAS
RAINER SCHMITZ

Este trabajo ha sido parcialmente financiado por Conicyt, a través de la "Beca para Estudios de Doctorado Nacional"

SANTIAGO DE CHILE
JULIO 2013

RESUMEN DE TESIS
PARA OPTAR AL GRADO DE DOCTOR EN
CIENCIAS DE LA INGENIERÍA
MENCIÓN FLUIDODINÁMICA
POR: FEDERICO FLORES MENESES
FECHA: 31 DE JULIO 2013
PROF. GUÍA: RENÉ GARREAUD
PROF. CO-GUÍA: RICARDO MUÑOZ

Resumen

En los últimos años se ha registrado un creciente interés por aplicar Dinámica de Fluidos Computacional (CFD por sus siglas en inglés) a flujos atmosféricos complejos. Esto debido a que el constante aumento de la capacidad computacional hace ahora más accesible la resolución numérica de este tipo de problemas multifísicos que consideran flujos altamente turbulentos, boyancia térmica, estratificación, y la geometría compleja con la que interactúa la circulación atmosférica cerca del suelo.
En esta tesis nos enfocamos en desarrollar y aplicar un solver CFD capaz de simular flujos atmosféricos en condiciones convectivas sobre geometrías complejas, que pueda considerar la compleja topografía que caracteriza la mayoría de los entornos naturales. Como plataforma usamos el software de código abierto OpenFOAM, enfocando su aplicación a la dispersión de contaminantes que ocurre dentro de rajos mineros de grandes dimensiones, la cual está controlada por la interacción de procesos netamente mecánicos producidos por la geometría de la cavidad, y procesos convectivos determinados por el flujo de calor desde la superficie del suelo. Para permitir la incorporación de geometría compleja en las simulaciones, sin perder la capacidad de resolver en detalle la turbulencia, utilizamos un modelo del tipo DES (detached eddy simulation), el cual permite aprovechar las capacidades de la técnica LES (large eddy simulation) en las zonas con turbulencia desarrollada lejos de las paredes, y reducir la demanda computacional en las zonas complejas cerca de las paredes mediante aproximaciones del tipo RANS (Reynolds Averaged Navier Stokes). Para incorporar el efecto de la estratificación usamos una formulación que incluye la densidad como una variable de cálculo en el sistema de ecuaciones, facilitando así el estudio de flujos boyantes. Definimos condiciones de borde e iniciales específicamente desarrolladas para el problema.
La primera parte de este trabajo, que incluye el desarrollo del modelo y su validación mediante diferentes casos experimentales y numéricos bien documentados, permitió comprobar la validez de usar OpenFOAM para estudiar este tipo de problemas multifísicos. Además fue posible corroborar la eficiencia de la técnica DES, capaz de resolver en detalle mediante la técnica LES la turbulencia atmosférica lejos del suelo, donde los flujos convectivos tienen más relevancia, y simular con precisión la interacción del flujo con la geometría superficial de manera económica mediante la técnica RANS. La segunda parte, que considera la aplicación del modelo al caso de Chuquicamata, uno de los rajos mineros más grande del mundo, permitió estudiar el efecto de la boyancia térmica sobre la dispersión de contaminantes tanto dentro como fuera del rajo, en condiciones ideales y reales. En particular, se concluye que las corrientes convectivas producidas por el flujo de calor desde la superficie reducen el efecto de la recirculación mecánica que se genera al barrer el flujo sobre la cavidad, lo que aumenta considerablemente la salida de partículas desde el interior de la cavidad, reduciendo su tiempo de vida dentro del rajo. Este efecto, unido a la importancia que se comprobó tiene la geometría 3D del rajo sobre la circulación, puede ser útil para el eventual desarrollo de sistemas de ventilación que incorporen estos aspectos.

Abstract

In recent years there has been an increasing interest in applying Computational Fluid Dynamics (CFD) to complex atmospheric flows. The steady increment of the computational capacity makes it now more accessible the numerical resolution of these multiphysics problems, that include highly turbulent flows, buoyancy, stratification, and the complex geometry with which the atmospheric flows interact near ground.

In this thesis we focus on the development and application of a CFD solver able to simulate atmospheric flows under buoyant conditions above complex geometries, such as those generated by the complex topography that characterizes many natural environments. As development platform we use the open-software OpenFOAM, focusing its application on the pollutant dispersion inside large scale open pit mines, that is controlled by the interaction of mechanical processes generated by the geometry of the cavity, and buoyant effects defined by the heat flux from the ground. To incorporate complex geometries in the simulations we use a DES (detached eddy simulation) approach, that allows us to use the high resolution of LES (large eddy simulation) technique in the zones of developed turbulence far from walls, but reducing the computational cost near complex geometries by means of RANS (Reynolds Averaged Navier Stokes) models. To incorporate the effect of stratification in the simulations we have chosen a formulation that includes density as a variable in the system of equations, thus facilitating further study of buoyant flows. Specific initial and boundary conditions were defined.

The first part of this work, which includes the development of the solver and its validation against experimental and numerical well documented cases, has demonstrated the validity of using OpenFOAM to study this type of complex multiphysics problems. Also, we probed the benefits of using a DES approach, allowing us to use LES to solve in detail developed turbulence far from ground, where turbulent convective flows are more relevant, and to include RANS models near complex geometry, in order to reduce computational requirements. The second part, which consideres the application of the solver to Chuquicamata, one of the largest open pit mines in the world, allowed us to study the effect of buoyancy on pollutant dispersion inside and outside the pit, under idealized and real conditions. In particular, it has been concluded that convective currents, controlled by the heat flux from ground, reduce the effect of the mechanical recirculation produced by the main flow that sweeps the cavity, increasing the exit of particles from inside the pit, reducing their lifetime inside the cavity. This effect, and the importance that we have proved has the 3D geometry of the pit over circulation, would make possible in the future the development of ventilation systems that consider these aspects.

Dedicada a mis padres y hermanos por su apoyo todo este tiempo.

Agradecimientos

Agradezco a mis profesores guía y co-guía, René Garreaud y Ricardo Muñoz, por los consejos y el tiempo dedicado a este largo proceso. También quisiera agradecer a Roberto Rondanelli y Marcos Díaz por darme la posibilidad de participar en diferentes e interesantes proyectos.

Mi reconocimiento va además para todos aquellos que hacen posible la operación del cluster Levque del Laboratorio Nacional de Computación de Alto Rendimiento (NLHPC por sus siglas en inglés), sin cuya ayuda las simulaciones numéricas de este trabajo no se habrían podido llevar a cabo.

Esta investigación fue financiada por Conicyt mediante la "Beca para Estudios de Doctorado Nacional".

Powered@NLHPC: Esta investigación fue apoyada en parte por la infraestructura de supercómputo del NLHPC (ECM-02) del Centro de Modelamiento Matemático CMM de la Universidad de Chile.

Acknowledgment

I would like to thank my advisor and co-advisor, René Garreaud and Ricardo Muñoz, for their guidance, and time dedicated to this long process. I would also like to thank Roberto Rondanelli and Marcos Díaz, for giving me the opportunity to take part in different and interesting projects.

I would like to extend my acknowledgements to those who made possible the operation of the Levque cluster of the NLHPC (National Laboratory of High Performance Computing), without whose support the numerical simulations of this work could not have been achieved.

This work was supported by Conicyt through the doctoral grant "Beca para Estudios de Doctorado Nacional".

Powered@NLHPC : This research was partially supported by the supercomputing infrastructure of the NLHPC (ECM-02), Center for Mathematical Modeling CMM, Universidad de Chile.

Contents

List of Tables

List of Figures

Nomenclature

α	thermal diffusivity [kg/(ms)]
α_t	effective SGS thermal diffusivity [kg/(ms)]
α_{sgs}	subgrid scale thermal diffusivity [kg/(ms)]
β	thermal expansion coefficient [1/K]
Δ	delta, filter width
δ_{ij}	kronecker delta
$\frac{\partial}{\partial t}$	partial derivative
$\frac{D}{Dt}$	total derivative
κ	von Karman constant
λ	potential temperature lapse rate [K/m]
$\mathbb{L}$	Reference length [m]
$\mathbb{U}$	Reference velocity [m/s]
$\mathbf{T}$	stress tensor [kg/(ms^2))]
$\mathbf{T}_{sgs}$	subgrid scale stress tensor [kg/(ms^2))]
μ	dynamic viscosity [Pa·s]
μ_{sgs}	subgrid turbulent viscosity [Pa·s]
∇p^*	external forcing [Pa/m]
∇	gradient operator [m^{-1}]
$\nabla\cdot$	divergence operator [m^{-1}]
ν	kinematic viscosity [m^2/s]

ν_{T} turbulent, or eddy, kinematic viscosity [m^2/s]

ν_{eff} effective kinematic viscosity [m^2/s]

$\overline{Q}$ mean heat flux [W/m^2]

ρ density [kg/m^3]

σ standard deviation

σ_k $k-\varepsilon$ model constant

σ_ε $k-\varepsilon$ model constant

τ dimensionless time

τ_{ij} Reynolds stress

θ potential temperature [K]

θ_0 initial potential temperature [K]

θ_1 potential temperature at level 1 [K]

θ_f final potential temperature [K]

$\vec{\Omega}$ earth rotation angular velocity [rad/s]

$\vec{f_c}$ Coriolis force [N/m^3]

$\vec{U}$ velocity vector [m/s]

B production/destruction of turbulence by buoyancy

C passive scalar

c_p specific heat of dry air [J/(KgK)]

$C_{1\mathrm{RNG}}$ RNG $k-\varepsilon$ model constant

C_μ $k-\varepsilon$ model constant

$C_{\varepsilon 1}$ $k-\varepsilon$ model constant

$C_{\varepsilon 2}$ $k-\varepsilon$ model constant

d displacement height [m]

e specific internal energy [J/kg]

f variable

f_v source

G function

h specific enthalpy [J/kg]

k turbulent kinetic energy [m^2/s^2]

l length scale

M Mach Number

n exponent of the power law

p pressure [Pa]

p_1 pressure at level 1 [Pa]

P_m modified pressure [P]

p_o reference pressure [Pa]

Pr_t Prandtl turbulent number

q_s kinematic heat flux at surface [K m/s]

R gas constant for dry air [J/(KgK)]

Ra Rayleigh number

Re Reynolds Number

Ri_b Bulk Richardson Number

Ro Rossby Number

t time [s]

T_* convective temperature scale [K]

t_f final time [s]

u velocity component in the x axis [m/s]

$U(z)$ horizontal velocity at height z [m/s]

U_* friction velocity [m/s]

u_h horizontal velocity, $\sqrt{u^2 + v^2}$ [m/s]

U_s horizontal velocity at height z_s [m/s]

$u_{\mathrm{i},j,k}$ velocity components [m/s]

v specific volume [m^3/kg]

v velocity component in the y axis [m/s]

V_o buoyancy velocity [m/s]

w velocity component in the z axis [m/s]

w_* convective velocity scale [m/s]

y_1^+ distance in wall units between the centroid of the first cell and the wall

z height [m]

z_1 height at level 1 [m]

z_i convective boundary layer height [m]

z_o roughness height [m]

z_s reference height [m]

ABL Atmospheric Boundary Layer

ASL Above Sea Level

CAE Computer Aided Engineering

CBL Convective Boundary Layer

CFD Computational Fluid Dynamics

DES Detached Eddy Simulation

DNS Direct Numerical Simulation

FCFM School of Physical and Mathematical Sciences of the University of Chile

g gravity acceleration [m/s^2]

k wavenumber [m^{-1}]

LES Large Eddy Simulation

NLHPC National Laboratory for High Performance Computing

RANS Reynolds Averaged Navier Stokes

RNG Renormalization Group

SGS Subgrid Scale

STL stereolithography

T temperature [K]

U_x component aligned with buildings [m/s]

URANS Unsteady Reynolds Average Navier Stokes

WRF Weather Research and Forecasting

Chapter 1

Introduction

1.1 Overview

In recent years there has been an increasing interest in applying Computational Fluid Dynamics (CFD) to complex micrometeorological problems, such as the wind in urban environments, above complex topography, or that controlled by thermal gradients (Fernando et al. (2010); Kondo et al. (2009)). Many relevant factors account for this interest, mainly due to new fields of application, such as renewable energies, pollutant dispersion, or the design of natural ventilation systems. In general, the problem can be described in terms of the interaction, within the atmospheric boundary layer, between the airflow and the objects that define the surface geometry, which can consider the complex structure of urban canyons, defined by the irregular distribution of buildings, or the varied topography of irregular terrain.

In terms of scale, this problem is an intermediate one, between the mechanical engineering models (aerodynamics), that consider spatial scales generally in centimeters, and meteorological models, whose scales are greater than a kilometer. Up to now the problem of simulating atmospheric flows within the boundary layer has been addressed from two angles. The focus adopted in meteorology has emphasized the developed flow, far from the walls, focusing the attention on the importance of stratification and buoyancy, without taking into account the effect that the obstacles or complex geometry have on the circulation. This vision is contrasted against the perspective taken by the researches related to wind in urban conditions, which have made the effort to develop specific wall functions capable of correctly simulating the effect of obstacles on the flow. There is consensus in the need that both visions converge to achieve more realistic simulations of the flows that characterize the Atmospheric Boundary Layer (ABL), in particular close to the ground, where the majority of human activity takes place (Chen et al. (2011)).

Despite being present in almost all real flows, and being particularly important in the atmospherical flows that interest us, turbulence has resisted mathematical approaches for centuries. Even today, a complete closed theory of turbulence is not available, and nor is there any hope to develope one. Not in vain it has been said that turbulence is *"the cemetery of theories"* (H. W. Liepmann cited by Tsinober (2004)). The numerical approach

to turbulence used today has been developed mainly from an engineering point of view, and is well documented in books like Pope (2008), Lesieur (2008) and Tennekes & Lumley (1972). In many ways it lacks strict mathematical rigor, but it establishes its foundation in experimental studies, the only alternative that has allowed a clear approach to the subject.

The recent interest in applying CFD to complex real flows has increased the need of a more precise numerical modelling of turbulence. This has resulted in the development of different turbulence models, each one with particular features and fields of application. In particular, turbulent flows with different levels of complexity have been recognised (table 1.1). The problem that interests us, wind over complex geometry, is among the most difficult to simulate. Its computational difficulty is increased mainly by the number of directions of its statistical inhomogeneity (3), the different scales involved, and the presence of developed turbulence. This difficulty is increased when we include buoyancy in the problem.

Table 1.1: Example of turbulent flows of various levels of computational difficulty (the difficulty increases downward and to the right) (from Pope (2008), page 338).

Dimensionality *(number of directions of statistical inhomegeneity)*	***Boundary layer*** *(statiscally stationary, boundary-layer approximation apply)*	***Statistically stationary***	***Not statistically stationary***
0			Homogeneous shear flow
1		Fully developed pipe or channel flow; self-similar free shear flows	Temporal mixing layer
2	Flat-plate boundary layer; jet in a co-flow	Flow through a sudden expansion in a two-dimensional duct	Flow over an oscillating cylinder
3	Boundary layer on a wing	Jet in a cross-flow; flow over an aircraft or building	Flow in the cylinder of a reciprocating engine

The dimensionless numbers that define the problem are described bellow.

- The *Rossby Number* is used to estimate the importance of Coriolis forces. It is the ratio of inertial to Coriolis forces:

$$Ro = \frac{\text{inertial forces}}{\text{Coriolis forces}} = \frac{\mathbb{U}}{\mathbb{L}2\Omega \sin\phi} \,. \tag{1.1}$$

  Using the tipical scales of the air flow that interacts with the surface geometry, $\mathbb{U} \sim 10$ m/s, $\mathbb{L} \sim 10$ m, $\Omega = 7.292 \cdot 10^{-5}$ rad/s and $\phi \sim 30^{o}$, we can estimate $Ro \sim$ 1e4. Therefore, in the case of small scale atmospheric flows interacting with obstacles we can assume that the Coriolis effect is relatively unimportant.
- The *Reynolds Number* is used to estimate the importance of turbulence. It is the ratio of inertial to viscous forces:

$$\mathrm{Re} = \frac{\text{inertial forces}}{\text{viscous forces}} = \frac{\rho \mathbb{U}\mathbb{L}}{\mu} \,. \tag{1.2}$$

Using the tipical scales of the air flow that interacts with the surface geometry, $\mathbb{U} \sim 10$ m/s, $\mathbb{L} \sim 10$ m, and air viscosity, $\mu/\rho = \nu = 1.5 \cdot 10^{-5}$ m^2/s, we can estimate $Re \sim$ 1e6. Therefore, inertial forces play a key role in this kind of flows, and turbulence must be considered in detail.

- The *Bulk Richardson Number* is usually used to estimate the importance of thermal convection. It represents the ratio of thermally produced turbulence and turbulence generated by vertical shear:

$$Ri_b = \frac{\text{buoyancy}}{\text{vertical shear}} = g\frac{\Delta\theta}{\theta}\frac{\Delta z}{(\Delta U)^2} , \tag{1.3}$$

  where $\Delta\theta$ is the potential temperature difference across a layer of thickness Δz, and ΔU is the change in horizontal wind across that same layer. This number is highly dependent on temperature. Using the tipical scales of the air flow that interacts with the surface geometry, over a heated topography, $\theta \sim 288$ K, $\Delta\theta \sim 1$ K, $\Delta z \sim 100$ m and $\Delta U \sim 1$ m/s, we can estimate $Ri_b \sim 3.4$. In urban environments this number is in the range 0.3 - 7 (Buccolieri et al. (2008)). Therefore, in cases where the heat flux from ground is important, thermal turbulence must be considered.

- The Mach number is used to estimate compressibility effects:

$$M = \frac{\mathbb{U}}{c} , \tag{1.4}$$

  where c is the velocity of the sound in the medium. Using $\mathbb{U} \sim 10$ m/s and $c \sim 1200$ km/h, we can estimate $M \sim 0.03$. This value is one order of magnitude below the limit usually used to assume incompressibility ($M \sim 0.3$). Therefore, due to the low velocity of most environmental flows, we can assume an incompressible flow.

From the previous analysis it can be concluded that a detailed simulation of turbulence is crucial in the case of small scale atmospheric flows interacting with complex geometries. Therefore, in the next few pages we present an overview of the main turbulence modelling techniques and their relationship with our work.

1.1.1 Turbulence modelling

Direct Numerical Simulation (DNS)

This technique is the most detailed and produces the best results, but it is too computationally demanding to be applied to atmospheric flows in large domains, or with complex geometry. It is not properly a turbulence model, because, when applied, the Navier-Stokes equations are solved numerically without any turbulence model in all the turbulence range, from the small dissipative scales up to those that occupy all the domain (Moin & Mahesh (1998)). As a consequence, the flow needs to be discretized at the resolution of the Kolmogorov microscales in all the domain. This allows to resolve the complete spatial and temporal state of the turbulent flow. Even if it is too demanding to be applied to real turbulent flows, the DNS technique is widely used in turbulence research, since its results make

it possible a detailed study of the flow, generating data that would be very difficult or even impossible to acquire experimentally.

The methods used to directly solve the Navier-Stokes equation depend on the type of turbulence considered. For example, in the case of isotropic homogeneous turbulence semi-spectral methods are widely used, representing the velocity field as a truncated Fourier serie. The cost of applying this type of techniques is that the computational requirements grow rapidly with the Reynolds number, aproximaterly as Re^3, limitating the simulation of high Reynolds flows to supercomputer facilities.

It is important to note, as stated by Pope (2008), that the use of the DNS technique is not only limited by the computational capacity, but also by the way in which it addresses the problem. It is highly likely that in solving most cases we are not interested in solving the flow with such level of accuracy. In DNS, above the 99% of the computational effort is used to solve the dissipative scales, even though in reality the physical phenomena observed are a reflex of the mean flow (movements that contain most of the energy). In fact, a large part of the flow resolved using DNS is not even measurable with the sensors that currently exist.

Large Eddy Simulation (LES)

LES is a natural extension of DNS, designed to preserve many of its advantages but reducing its computational requirements. This technique solves directly the large scales of the flow, modelling the small scales using some kind of turbulence modelling technique. As a consequence, LES solves in detail the dynamic of the large scale circulation, that is affected by the geometry of the domain (and therefore is not universal), while the influence of the small scales (which are to some extent universal in the domain) is represented by means of simple models. In this way the computational requirements of DNS are highly reduced, preserving its precision in the scales that contain much of the energy. This approach to the problem of turbulence modelling rests on the supposition that the most important processes of turbulent transport occur in the eddies of large and medium scale, while the smaller eddies are mainly responsible of the dissipation of turbulent energy.

When we apply LES, we assume that any variable (f) is decomposed into two components, one large-scale ($\overline{f}$) and the other small-scale (f'), i.e.:

$$f = \overline{f} + f' \ . \tag{1.5}$$

A filtering operation is used to extract the large-scale components, consisting of a convolution with a previously defined function:

$$\overline{f} = \oint G(x, x'; \Delta) f(x') \, \mathrm{d}x' \ , \tag{1.6}$$

where Δ, the filter width, depends generally on the mesh. There are different alternatives for the function G, depending on the numerical discretization to be used. The approach employed by LES is to directly solve the large scale flow while modelling the effect of the small scales, which will be in general subgrid. There are different SGS (subgrid scale) models

to do this. For a description of the manner in which LES is applied in *OpenFOAM* and the SGS models available, see for example de Villiers (2006) and chapter 2 of this work.

Because in LES the small scales are not resolved, higher Reynolds numbers can be reached without the computational cost of DNS. However, despite this simplification, the computational requirements when LES is used in large scale atmospheric flows are still high, and parallel computing is needed.

Interestingly, though, this method was originally developed to deal with meteorological applications (Smagorinsky (1963), Lilly (1967) and Deardorff (1974), as cited by Pope (2008)), and nowadays the detailed modelling of the atmospheric boundary layer is one of its main fields of application (Moeng & Sullivan (1994); Khanna & Brasseur (1998); Churchfield et al. (2010)).

Just as LES was developed from DNS in the hope to reduce its computational cost, many techniques have been developed from LES with the same objective. For example, there exist DES (detached eddy simulation) and VLES (very large eddy simulation) techniques to which we will refer later in this chapter.

Reynolds Averaged Navier-Stokes Equations (RANS)

This approach to turbulence modelling considers different models that solve the Reynolds equations of the averaged velocity field. RANS models are by far the most used modelling technique in CFD, in particular in engineering applications. Given that they only consider averaged movements they allow the use of less refined meshes, and drastically reduce the computational requeriments of the simulations.

RANS uses the mean flow equations. This approximation consideres the average of the Navier Stokes equations after the inclusion of perturbations.

Using Navier-Stokes and the continuity equation:

$$\frac{\mathrm{D}\vec{U}}{\mathrm{D}t} = -\frac{1}{\rho}\nabla P_m + \nu\nabla^2\vec{U} \;, \tag{1.7}$$

$$\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\vec{U}) = 0 \;, \tag{1.8}$$

with P_m a modified pressure, $P + \rho g z$, hereafter only P.

Perturbations are introduced in the system:

$$\vec{U}(\vec{x},t) = \langle\vec{U}(\vec{x},t)\rangle + \vec{u}(\vec{x},t) \;. \tag{1.9}$$

Using the continuity equation and averaging the Navier-Stokes equation we have:

$$\frac{\bar{\mathrm{D}}}{\bar{\mathrm{D}}t}\langle U_j\rangle = -\frac{1}{\rho}\frac{\partial\langle P\rangle}{\partial x_j} + \nu\nabla^2\langle U_j\rangle - \frac{\partial\langle u_\mathrm{i}u_j\rangle}{\partial x_\mathrm{i}} \;. \tag{1.10}$$

The preceding equation receives the name of Reynolds-averaged Navier–Stokes equations, and is generally written as:

$$\rho \frac{\bar{\mathrm{D}}}{\bar{\mathrm{D}}t}\langle U_j \rangle = \frac{\partial}{\partial x_{\mathrm{i}}} \left[\underbrace{\mu \left(\frac{\partial \langle U_{\mathrm{i}} \rangle}{\partial x_j} + \frac{\partial \langle U_j \rangle}{\partial x_{\mathrm{i}}} \right)}_{\text{viscous}} - \underbrace{\langle p \rangle \delta_{\mathrm{i}j}}_{\text{isotropic}} - \underbrace{\rho \langle u_{\mathrm{i}} u_j \rangle}_{\text{apparent}} \right] . \tag{1.11}$$

By convention $\langle u_{\mathrm{i}} u_j \rangle$ are defined as the Reynolds Stress, $\tau_{\mathrm{i}j}$.

$$\tau_{\mathrm{i}j} = \langle u_{\mathrm{i}} u_j \rangle = \begin{bmatrix} \langle u_1 u_1 \rangle & \langle u_1 u_2 \rangle & \langle u_1 u_3 \rangle \\ \langle u_2 u_1 \rangle & \langle u_2 u_2 \rangle & \langle u_2 u_3 \rangle \\ \langle u_3 u_1 \rangle & \langle u_3 u_2 \rangle & \langle u_3 u_3 \rangle \end{bmatrix} , \tag{1.12}$$

$\langle u_1 u_1 \rangle, \langle u_2 u_2 \rangle, \langle u_3 u_3 \rangle$: normal stresses.

$\langle u_1 u_2 \rangle$, ... : shear stresses.

The turbulent kinetic energy, k, is also defined:

$$k = \frac{1}{2} \left[\langle u_1 u_1 \rangle + \langle u_2 u_2 \rangle + \langle u_3 u_3 \rangle \right] = \frac{1}{2} \langle u_{\mathrm{i}} u_{\mathrm{i}} \rangle . \tag{1.13}$$

The Reynolds stresses that appear in these equations are modelled by a turbulence model. Basically, the different models available adress the problem from two approach: using the turbulent-viscosity hypothesis, or by means of calculating the effect that each component of the Reynolds stress tensor has over the flow. Hereafter we will review the turbulent-viscosity hypothesis, the most widely used approach in RANS CFD.

The inspiration behind this idea, originally proposed by Boussinesq (Schmitt (2007)), is to suppose that there exists a linear relationship between the Reynolds stresses and the deformation rate. The proportionality constant in this relationship is called turbulent viscosity, in analogy to the viscosity used in Newton's stress law. This relationship can be expressed as:

$$\langle u_{\mathrm{i}} u_j \rangle = \frac{2}{3} k \delta_{\mathrm{i}j} - \nu_{\mathrm{T}} \left(\frac{\partial \langle U_{\mathrm{i}} \rangle}{\partial x_j} + \frac{\partial \langle U_j \rangle}{\partial x_{\mathrm{i}}} \right) , \tag{1.14}$$

where ν_{T} is the turbulent, or eddy, viscosity.

Using this approach the Reynolds equations are expressed as:

$$\frac{\bar{\mathrm{D}}}{\bar{\mathrm{D}}t}\langle U_j \rangle = \frac{\partial}{\partial x_{\mathrm{i}}} \left[(\nu + \nu_{\mathrm{T}}) \left(\frac{\partial \langle U_{\mathrm{i}} \rangle}{\partial x_j} + \frac{\partial \langle U_j \rangle}{\partial x_{\mathrm{i}}} \right) - \frac{1}{\rho} \left(\langle p \rangle + \frac{2}{3} \rho k \right) \delta_{\mathrm{i}j} \right] , \tag{1.15}$$

$$\frac{\bar{\mathrm{D}}}{\bar{\mathrm{D}}t}\langle U_j \rangle = \frac{\partial}{\partial x_{\mathrm{i}}} \left[\nu_{eff} \left(\frac{\partial \langle U_{\mathrm{i}} \rangle}{\partial x_j} + \frac{\partial \langle U_j \rangle}{\partial x_{\mathrm{i}}} \right) - \frac{1}{\rho} \left(\langle p \rangle + \frac{2}{3} \rho k \right) \delta_{\mathrm{i}j} \right] . \tag{1.16}$$

To solve them it is necessary to estimate the value of ν_{T}. Different techniques are available. We describe some of them below.

- Standard $k-\varepsilon$ model

  The $k-\varepsilon$ model calculates the turbulent viscosity using a length scale l and a velocity scale v, both based on k, the turbulent kinetic energy, and ε, the dissipation rate of this energy. We have:

$$\text{length scale: } l = \frac{k^{\frac{3}{2}}}{\varepsilon} , \tag{1.17}$$

$$\text{velocity scale: } v = k^{\frac{1}{2}} , \tag{1.18}$$

  and the turbulent viscosity is computed as:

$$\nu_{\mathrm{T}} = C_\mu \cdot l \cdot v , \tag{1.19}$$

$$\nu_{\mathrm{T}} = C_\mu \frac{k^2}{\varepsilon} . \tag{1.20}$$

  To calculate k and ε, the transport equations for each variable are used:
  turbulent kinetic energy transport equation:

$$\frac{\partial k}{\partial t} + \bar{u}_j \frac{\partial k}{\partial x_j} = -\tau_{\mathrm{i}j} \frac{\partial \bar{u}_{\mathrm{i}}}{\partial x_j} - \varepsilon + \frac{\partial}{\partial x_j} \left[\left(\nu + \frac{\nu_{\mathrm{T}}}{\sigma_k} \right) \frac{\partial k}{\partial x_j} \right] , \tag{1.21}$$

  turbulent kinetic energy dissipation rate transport equation:

$$\frac{\partial \varepsilon}{\partial t} + \bar{u}_j \frac{\partial \varepsilon}{\partial x_j} = -C_{\varepsilon 1} \frac{\varepsilon}{k} \tau_{\mathrm{i}j} \frac{\partial \bar{u}_{\mathrm{i}}}{\partial x_j} - C_{\varepsilon 2} \frac{\varepsilon^2}{k} + \frac{\partial}{\partial x_{\mathrm{i}}} \left[\left(\nu + \frac{\nu_{\mathrm{T}}}{\sigma_\varepsilon} \right) \frac{\partial \varepsilon}{\partial x_j} \right] . \tag{1.22}$$

  The value of each constant is obtained experimentally. The standard values, recommended by Pope (2008), are:

$$C_\mu = 0.09, \quad C_{\varepsilon 1} = 1.44, \quad C_{\varepsilon 2} = 1.92, \quad \sigma_k = 1.0, \quad \sigma_\varepsilon = 1.3 . \tag{1.23}$$

  The buoyancy-modified $k-\varepsilon$ model uses the same eddy-viscosity Boussinesq concept applied before to Reynold Stresses, to define a turbulent heat flux. It introduces in the transport equations a new term that represents production/destruction of turbulence by buoyancy, linked to temperature fluctuations:

$$B = -\beta g \langle u_{\mathrm{i}} \mathrm{T} \rangle = -\beta g \frac{\nu_{\mathrm{T}}}{Pr_t} \frac{\partial \langle \mathrm{T} \rangle}{\partial x_{\mathrm{i}}} . \tag{1.24}$$

- RNG $k-\varepsilon$ model (Renormalisation Group)

  This model is a variation of the original $k-\varepsilon$ model. It was proposed in 1986 by Yakhot and Orzag, who applied the RNG theory (Renormalisation Group) to the Navier Stokes equations in order to derive the $k-\varepsilon$ equations. The unique change is the inclusion of an extra term in the dissipation rate ε equation, and a small modification of the original constants. For ε it is used:

$$\frac{\partial \varepsilon}{\partial t} + \bar{u}_j \frac{\partial \varepsilon}{\partial x_j} = -\left(C_{\varepsilon 1} - C_{1\mathrm{RNG}} \right) \frac{\varepsilon}{k} \tau_{\mathrm{i}j} \frac{\partial \bar{u}_{\mathrm{i}}}{\partial x_j} - C_{\varepsilon 2} \frac{\varepsilon^2}{k} + \frac{\partial}{\partial x_{\mathrm{i}}} \left[\left(\nu + \frac{\nu_{\mathrm{T}}}{\sigma_\varepsilon} \right) \frac{\partial \varepsilon}{\partial x_j} \right] , \tag{1.25}$$

with

$$C_{1\mathrm{RNG}} = \frac{\eta\left(1-\frac{\eta}{\eta_0}\right)}{(1+\beta\eta^3)}\,, \tag{1.26}$$

and

$$\eta = \left(\frac{1}{\rho}\frac{P}{\nu_{\mathrm{T}}}\right)^{\frac{1}{2}}\frac{k}{\varepsilon}\,. \tag{1.27}$$

The new values of the constants are:

$$C_\mu = 0.0845, \quad C_{\varepsilon 1} = 1.42, \quad C_{\varepsilon 2} = 1.68, \quad \sigma_k = \sigma_\varepsilon = 0.72\,. \tag{1.28}$$

In general, in the case of wind simulations, it is recomended to use this model rather than the standard $k-\varepsilon$ (Franke et al. (2004); Palmer et al. (2003)).

As a summary, the main turbulence models available can be grouped as shown in table 1.2, where the cost is given by the computational requirements of applying the model to real life flows.

Table 1.2: Summary of main turbulence models.

Model	Technique	Equations	Cost	Application
DNS	Spectral	Fourier	Extremely high	Turbulence Theory
LES	DNS-RANS	Fourier	Very High	Environmental Flows, Aerodynamics
RANS	Turbulent Viscosity	Length scale	low	Pipe flows
		Standard $k-\varepsilon$	medium	Engineering, wide range
		Standard $k-\omega$	medium	Engineering, wide range
		RNG $k-\varepsilon$	medium	Engineering, wide range
	Nonlinear	nonlinear $k-\varepsilon$	high	Engineering, specific applications
	Second order	Reynolds Stresses	high	Engineering, specific applications
		Algebraic Stresses	high	Engineering, specific applications

Micrometeorological flows as the ones that interest us, where the flow interacts with complex geometry, but is also influenced by intense atmospheric turbulence, requires a detailed treatment of turbulence modelling. To solve this problem in this work we used a DES (detached eddy simulation) approach, using LES to solve in detail the flow, but including RANS techniques near walls in order to reduce the computational cost of the simulations. We describe the DES technique below.

DES

In the case of LES simulations the treatment of the flow close to the walls is especially complex, since, in general, the computational cost of applying this technique in these areas is too high, seriously limiting its application in the case of complex geometries (Wyngaard (2004); Spalart (2009)). Close to the surface the energy-containing turbulence scale may be close to the scale of the spatial filter used in the LES equations, which could lead the

simulation to operate in a range that moves away from the optimum for which LES was designed ("Terra Incognita" Wyngaard (2004)), unless very refined meshes are used in that zone. However, the excellent results obtained when LES is applied to areas of flow separation, where most RANS techniques fail, have led in the past few years to the development of a series of hybrid techniques that are capable of adapting to different scales, combining RANS with LES models. The DES technique (detached-eddy-simulation), originally proposed by Spalart (Spalart et al. (1997)) to deal with highly separated flows such as those existing in the aerospace industry, combines the benefits of LES and RANS, dividing the domain according to different turbulence zones. Near-wall regions are solved in a RANS-like manner, while the rest of the flow is solved in a LES-like manner (Bechmann et al. (2007); Spalart et al. (1997)), as shown in figure 1.1. These characteristics make DES especially useful in the case of atmospheric flows around complex geometries, allowing us to benefit from the excellent LES results in the zones with highly developed turbulence far from the ground, and from the RANS methods widely used in wind engineering near walls.

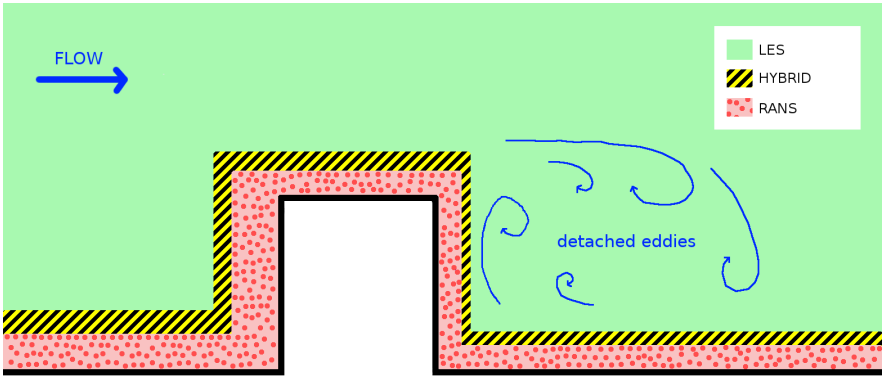


Figure 1.1: Typical turbulence zones in a DES simulation, adapted from de Villiers (2006).

In the DES formulation proposed by Spalart, the transition from near wall RANS simulation to LES treatment of the flow is accomplish by replacing the near wall distance, d, by a modified distance function, $\tilde{\mathrm{d}}$, defined as (Spalart et al. (1997); Squires (2004); de Villiers (2006)):

$$\tilde{\mathrm{d}} = \min(\mathrm{d}, C_{DES}\Delta)\,, \tag{1.29}$$

with Δ the grid size, defined in DES as the largest cell edge length, $\Delta = \max(\Delta x, \Delta y, \Delta z)$, and C_{DES} a constant. The Spalart Allmaras turbulence model (described below) contains a destruction term for its viscosity $\tilde{\nu}$, which is proportional to $(\tilde{\nu}/\mathrm{d})^2$. When balanced with the production term, this term adjusts the eddy viscosity to scale with the local deformation rate S and d: $\tilde{\nu} \propto S\mathrm{d}^2$, while the sub grid scale eddy viscosity scales with S and the grid spacing Δ : $\nu_{SGS} \propto S\Delta^2$. Thus, the idea behind the proposed modified distance is that the Spalart Allmaras turbulence model, with d replaced by a length proportional to Δ, can be a SGS model (Spalart et al. (1997)). As a consequence, if we use $\tilde{\mathrm{d}}$ instead of d the model acts as the Spalart-Allmaras RANS model for $\mathrm{d} \ll \Delta$ and a Subgrid Scale model for $\Delta \ll \mathrm{d}$. Different tests have allowed C_{DES} to be calibrated at 0.65 (de Villiers (2006)).

The Spalart-Allmaras (S-A) one-equation eddy viscosity model used as RANS turbulence

model in the near-wall region use an eddy viscosity ν_t given by:

$$\nu_t = \tilde{\nu} f_{v1}\,, \tag{1.30}$$

$$f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3}\,, \tag{1.31}$$

$$\chi = \frac{\tilde{\nu}}{\nu}\,, \tag{1.32}$$

where ν is the molecular viscosity, and $\tilde{\nu}$ is a working variable that obeys the tranport equation:

$$\frac{D\tilde{\nu}}{Dt} = c_{b1}\tilde{S}\tilde{\nu} + \frac{1}{c_\sigma}\left[\nabla\cdot((\nu+\tilde{\nu})\nabla\tilde{\nu}) + c_{b2}(\nabla\tilde{\nu})^2\right] - c_{w1}f_w\left[\frac{\tilde{\nu}}{\mathrm{d}}\right]^2\,, \tag{1.33}$$

$$\tilde{S} = \omega + \frac{\tilde{\nu}}{\kappa^2\mathrm{d}^2}f_{v2}\,, \tag{1.34}$$

$$f_{v2} = 1 - \frac{\chi}{1+\chi f_{v1}}\,. \tag{1.35}$$

ω is the vorticity, and the function f_w is given by

$$f_w = g\left[\frac{1+c_{w3}^6}{g^6+c_{w3}^6}\right]^{1/6}\,, \tag{1.36}$$

$$g = r + c_{w2}(r^6 - r)\,, \tag{1.37}$$

$$r = \frac{\tilde{\nu}}{\tilde{S}\kappa^2\mathrm{d}^2}\,. \tag{1.38}$$

The wall boundary condition is $\tilde{\nu} = 0$ and the constants are $c_{b1} = 0.1355$, $c_\sigma = 2/3$, $c_{b2} = 0.622$, $\kappa = 0.41$, $c_{w1} = c_{b1}/\kappa^2 + (1+c_{b2})/C_\sigma$, $c_{w2} = 0.3$, $c_{w3} = 2$ and $c_{v1} = 7.1$ (de Villiers (2006)).

The results shown in this work correspond to DES simulations using Spalart Allmaras model for the near-wall treatment, although we also experimented with different standard wall functions using URANS simulations (RNG $k-\varepsilon$ model) to compare results. For a description of the manner in which DES is applied in *OpenFOAM*, see for example de Villiers (2006) and chapter 2 of this work. The Spalart Allmaras DES formulation is currently available in OpenFOAM, but it is also possible to implement different hybrid RANS-LES models using as a base the turbulence models available in the code.

1.1.2 Micrometeorology and dispersion of pollutants inside large open pit mines

The micrometeorology of open pit mines can be defined as the study of meteorological conditions inside the cavity of open pit mines, including air circulation, heat transfer and pollutant dispersion.

As explained by Baklanov (Baklanov & Rigina (1995); Baklanov (1995, 2000)), this meteorological problem has specific features that differentiate it from other problems inside the atmospheric boundary layer:

- existence of very inhomogeneous orographic surfaces, that produce forced air currents inside the cavity, which make it necessary to use a non-hydrostatic approach. Also, the wind that sweeps the pit can produce air flow recirculation inside its atmosphere. This is a well known phenomenon affecting flows over cavities (Bres & Colonius (2008); Kang & Sung (2009); Mesalhy et al. (2009)), being the aspect ratio of the cavity (length/depth) the key parameter defining the number, size and location of the main rolls inside the cavity.
- solar radiation has an important influence in the micrometeorology inside the pit, producing local temperature inversions and buoyant currents. The daily insolation cycle changes these conditions during the day, controlling the forcing produced by the heat flux from the surface. Since our primary objective is to understand pollutant dispersion inside the pit during day, in this work we will focus on daytime conditions.
- the atmosphere inside the pit is more closed than the external atmosphere, which tends to keep the air, and in consequence the pollutants, inside the cavity. It has been shown that this pollutant confinement is enhanced by the turbulent fluxes that occur inside the pit (Rigina & Baklanov (1995); Silvester et al. (2009)).

Although the most part of the air circulation inside the pit is controlled by thermal convection, several other mechanisms are involved, at different scales. In particular, there are local mechanical effects due to complex topography and large scale pressure gradients that control the general atmospheric circulation affecting the pit (Whiteman (1990)).

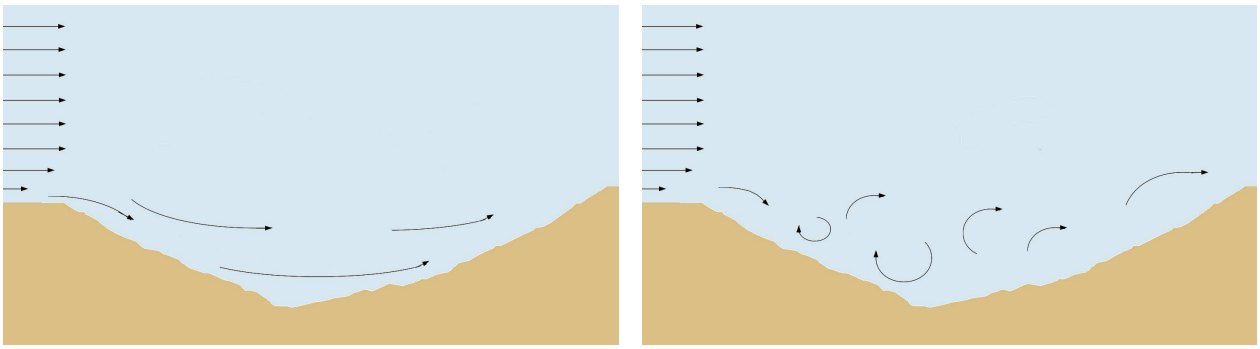
The complex topography that characterizes large scale open pit mines is illustrated by figure 1.2, that corresponds to Chuquicamata, whose vast dimensions make it one of the largest pits in the world, with more than 4 km long, 3.5 km wide and almost 1 km deep.

(a) Real topography **(b)** Lateral view

Figure 1.2: Chuquicamata open pit. a) Topography of the pit (©Codelco, Chilean National Copper Corporation, under Creative Commons). b) lateral view, color scale shows depth.

The particular features described above have made it difficult to numerically simulate the pollutant dispersion inside open pit mines, forcing the use of a more simplified approach to dispersion modelling based on Gaussian plume theory. This is in spite of the fact that it has been proved that the complex turbulent air flow circulation inside the pit controls the exit of pollutants from inside the cavity (Silvester et al. (2009)). As illustrated in figure 1.3, the flow is not limited only to following the shape of the pit, but it interacts with the topography

generating recirculating and turbulent flows, that modify air flow circulation inside the cavity.



(a) Simplified trajectories, Gaussian model **(b)** Real turbulent trajectories

Figure 1.3: Topographic influence over air circulation inside the pit, adapted from Silvester et al. (2009).

Given the many processes included in the problem, with strong interactions among them, there is consensus that the generalization of the problem will be almost impossible (Whiteman (1990); Whiteman et al. (2008)). Although in most cases it is possible to identify the processes participating in the pollutant dispersion, their relative importance and the way they interact depend on the particular topography of the pit, on the hour of day, and even on the location inside the cavity if the pit is of a large scale. Also, the external forcing (regional wind) is a condition unique to each open pit, increasing the difficulty of a generalization of the problem. These conditions explain the interest in developing detailed numerical simulations, able to represent the particular conditions of each case, including the interaction of scales and processes that control the dispersion of pollutants inside large open pit mines.

In general, the air flow affecting the dispersion of pollutants inside and around closed valleys, like the ones that interest us, is composed by two main processes that control its evolution, and must be included in a numerical simulation of the problem:

- Slope flows controlled by buoyancy, which will generally be the flow dominating the circulation inside the pit. There exist several references relative to this type of flows (Sharples et al. (2010); Skyllingstad (2002); Axelsen (2010)).
- Mechanical effects produced by the interaction of the external flow with the geometry of the cavity. The approximately conical features of open pit mines modify the circulation imposed by buoyant slope flows, introducing mechanical effects produced by the slope angle and circular symmetry of the pit. In this regard, the analysis is similar to that developed for valleys (Whiteman (1990); Colette et al. (2003)), although the open condition of the latter allows for exit flows not possible in closed pits. Given that the existence of large scale closed valleys is rare in nature the research about air circulation inside their atmosphere has remained restricted to meteoritical craters (Whiteman et al. (2008); Kiefer & Zhong (2010)), or to the specific case of large open pit mines (Silvester et al. (2009); Whiteman et al. (2008)).

As a consequence, a CFD solver developed to simulate pollutant dispersion inside large open pits must incorporate buoyancy, stratification and a detailed simulation of turbulence in the study of atmospheric flows, resolving the flow around complex geometries.

1.1.3 The *OpenFOAM* framework

OpenFOAM (Open Field Operation and Manipulation, Weller et al. (1998)) is a collection of C++ libraries designed to solve complex problems in fluid mechanics. Developed with the desire of obtaining a more effective numerical platform than Fortran, it has benefitted from the new C++ object-oriented programming functionality. Its completely free distribution and the flexibility it offers, allowing the development of specific solvers by the user, which can be integrated with already existing tools, have made *OpenFOAM* a widely used research tool. As most of the CFD codes currently used in engineering, it uses a finite volume formulation. Within its libraries, *OpenFOAM* integrates turbulence models (RANS and LES), thermophysical models, radiation models and wall functions, which can be accessed when developing a solver. Furthermore, the code offers the possibility of integrating complex geometries in the calculation, through the use of the *snappyHexMesh* tool. For an extensive review of the numerical methods used by the code, as well as of the available tools, we refer to the available documentation OpenCFD (2009).

There are several examples of the application of *OpenFOAM* to the study of atmospheric flows. García & Boulanger (2006) simulated the wind flow over Mount Saint Helens in the United States employing *OpenFOAM* with a standard $k-\varepsilon$ RANS turbulence model. A similar work was done by Hussein & El-Shishiny (2009) studying the wind flow over the Giza Plateau in Egypt. García et al. (2008) simulated the convective winds generated in the Aburra open valley, near Medellín, Colombia, using standard $k-\varepsilon$ RANS turbulence. Pedruelo (2009) used *OpenFOAM* to model the non buoyant wind flow over complex terrain using RANS turbulence models with wall functions, to make accurate predictions of wind power production. Churchfield (2010) developed an *OpenFOAM* solver using LES to simulate buoyant flows under the Boussinesq approximation, focusing on the flow interaction with wind turbines.

Given the great versatility that it provides, allowing us to focus the simulation from a multi-physical perspective, we selected *OpenFOAM* as development platform. The development and validation of our solver will be presented in chapter 2.

1.2 Objectives

The primary target of this work was to identifie the main modes of contaminant transport inside and outside Chuquicamata, one of the largest open pit mines in the world, whose ventilation seems to be dominated by convective effects (Barkan & Alpert (2010)). With this aim, three specific objectives were defined:

- Use *OpenFOAM* as development platform of a CFD solver that incorporates buoyancy, stratification and a detailed simulation of turbulence in the study of atmospheric flows, resolving the flow around complex geometries.
- Aim our application towards the modelling of atmospheric circulation within open pit mines, with the objective of applying it to study the complex circulation and dispersion of contaminants that originates within Chuquicamata.

- Perform numerical simulations of the air flow inside the pit, in order to study the effect of mechanical and buoyant processes on the dispersion of pollutants, and identifie the main modes of contaminant transport.

The development, and validation, of a CFD solver able of simulating turbulent buoyant atmospheric flows, from a multiphysical approach incorporating the different scales involved, proved to be a very demanding task, and represents a large section of this thesis.

1.3 Organization

This thesis is based on two articles. Each article represents one chapter of the thesis. A third article not directly related with numerical simulations but focused on the development of a platform able to collect field data in order to test the numerical results of the simulations is presented in the appendices.

In chapter 2 we explain in detail the development and validation of the CFD solver. Details about the numerical approach, turbulence modelling and initial and boundary conditions considered, are included. Also, we compare our results against several numerical and experimental results, in order to validate our solver. We use documented experimental data (ERCOFTAC and Architectural Institute of Japan (AIJ) libraries), experimental field data (small field campaign), and numerical data well documented in previous works.

In chapter 3 we apply our solver to study the air flow circulation and particle dispersion inside the open pit of Chuquicamata. Both idealized and real topographies were used. Different combinations of boundary conditions (wind and surface heat flux) were applied, in order to study the importance of mechanical and buoyant effects on air circulation inside the pit.

Chapter 4 summarizes the main conclusions of this work.

The appendices include a third paper and details about the use of *OpenFOAM* and the development of our solver.

Chapter 2

Model development and testing

This chapter corresponds to an extended version of the paper *"CFD simulations of turbulent buoyant atmospheric flows over complex geometry: solver development in OpenFOAM"*, authored by Federico Flores, René Garreaud and Ricardo Muñoz, accepted in the form of research article in *Computers and Fluids* (Flores et al. (2013a)).

ABSTRACT

This paper, first of a two-part work, presents an overview of the development of a computational fluid dynamics (CFD) solver in OpenFOAM platform to simulate the internal ventilation regime within an open pit including the effects of developed turbulence, buoyancy and stratification. To incorporate the effect of stratification in the simulations we have chosen a formulation that includes density as a variable in the system of equations, thus facilitating further study of buoyant flows. Given the importance of turbulence in this type of large-scale flows we have used Large Eddy Simulation (LES) to incorporate it in the calculation, using a Detached Eddy Simulation (DES) approach to solve the flow near walls. Specific initial and boundary conditions were defined.

The results presented in this paper, including several tests of the solver where we compared our results with experimental or numerical data, have demonstrated the validity of using OpenFOAM to study this type of complex multiphysics problems. Especially advantageous in this regard are the flexibility provided by the modular structure of the code, the possibility of defining specific boundary and initial conditions for each case, and the ability of generating detailed meshes of complex geometries. Also, we probed the benefits of using a DES approach, allowing us to solve developed turbulence and the interaction of the flow with detailed geometry. A second paper associated to this work will expose the application of the solver to large open pit mines, simulating the particular case of Chuquicamata, one of the largest open pit mines in the world, located in northern Chile.

2.1 Introduction

In recent years, a growing interest has been seen in applying Computational Fluid Dynamics (CFD, for acronyms see nomenclature) to simulate complex micrometeorological processes, like air flow in urban areas, over complex topography, or that controlled by thermal gradients (Fernando et al. (2010); Franke et al. (2007); Kondo et al. (2009); Tominaga et al. (2008)). Among several factors that have contributed to this interest is the application of CFD to new fields, such as renewable energies, the dispersion of contaminants or the study of natural ventilation systems. In general, the problem can be described in terms of the interaction, within the atmospheric boundary layer, between the airflow and the objects that define the complex surface geometry.

A main challenge of the numerical simulation of atmospheric flows arises from the different spatial scales involved, ranging from the small scale typically used in engineering to the large scale used in meteorology (table 2.1). The DNS technique, in which the Navier-Stokes equations are numerically solved without any turbulence model, allows to study the turbulence in detail (Moin & Mahesh (1998)), but its computational requirements restrict it to small domains with reduced geometrical complexity. This makes it incompatible with the study of wind around complex geometries, in which case less expensive techniques using time averaged equations (RANS) are preferred (Franke et al. (2007)) and specialized in the treatment of walls (Franke et al. (2004)). To simulate the circulation in valleys or to study the turbulence within the convective boundary layer, it is preferable to opt for LES techniques (Fritts et al. (2010); Moeng & Sullivan (1994); Brasseur & Wei (2010)), that can resolve large scale vortices more accurately than RANS, although they are difficult to apply to complex geometries (Bechmann et al. (2007)). In general, the techniques used in meteorological large scale problems (mesoscale, numerical weather prediction, climate models (Lee et al. (2011); Han & Pan (2011))) employ numerical methods that make difficult for these models to resolve the interaction of the flow with small scale obstacles, in spite of using refined meshes (Wyngaard (2004)). They typically use standard finite-difference methods, which lack the geometric flexibility offered by finite volume methods used in CFD. These methods usually employ relatively low-order numerical approximations. This, for example, explains the problems when using WRF (Weather Research and Forecasting) in urban areas and the need of coupling it with CFD models that are capable of resolving urban-scale flows (Chen et al. (2011)).

There is a consensus in the need that both modelling approaches explained above (importance of walls and developed turbulence) converge to achieve more realistic simulations of the flows that characterize the Atmospheric Boundary Layer (ABL), in particular close to the ground, where most human activities take place. It is necessary to consider that close to the surface the energy-containing turbulence scale may be close to the scale of the spatial filter used in the LES equations, which could lead the simulation to operate in a range that moves away from the optimum for which LES was designed ("Terra Incognita" Wyngaard (2004)). To solve this problem new subgrid models have been developed for LES, as well as hybrid techniques that are capable of adapting to different scales, combining RANS with LES models. Detached Eddy Simulation (DES) techniques solves near-wall regions in a RANS-like manner, and the rest of the flow in a LES-like manner (Bechmann et al. (2007); Spalart et al. (1997)).

Table 2.1: Spatial scales and turbulence models commonly used in the numerical simulation of atmospheric flows

scale (grid length)	area of interest	technique	references
$\Delta x < 0.1$ m	turbulence research	DNS	Moin & Mahesh (1998)
0.1 m $< \Delta x <$ 10 m	wind in urban areas	RANS,URANS	Franke et al. (2007); Blocken et al. (2008); Zhang et al. (2005); Buccolieri et al. (2008); Tominaga et al. (2008)
	natural ventilation	RANS	Assimakopoulos et al. (2005)
1 m $< \Delta x <$ 10 m	wind energy	LES, DES	Bechmann et al. (2007)
	pollutant dispersion	RANS,LES	Silvester et al. (2009); Chan (2004)
10 m $< \Delta x <$ 100 m	closed valleys	URANS LES	Baklanov (2000); Shi et al. (2000) Colette et al. (2003); Kiefer & Zhong (2010); Fritts et al. (2010)
	katabatic flows	LES	Skyllingstad (2002); Axelsen (2010)
	atmospheric boundary layer	LES	Moeng (1984); Moeng & Sullivan (1994); Khanna & Brasseur (1998); Brasseur & Wei (2010); Churchfield et al. (2010)
100 m $< \Delta x <$ 10 km	mesoscale	RANS	Lee et al. (2011)
$\Delta x >$10 km	numerical weather prediction, climate models	RANS	Han & Pan (2011)

To simulate the air circulation of interest to us, which involves several physical processes interacting between each other and with a complex geometry, it is necessary to use computational tools that are capable of incorporating these processes and complex meshes. In our case, given the great versatility that it provides, we selected the CFD tool *OpenFOAM* (Open Field Operation and Manipulation, Weller et al. (1998)) as development platform, focusing the problem from a multi-physical perspective, incorporating the effects of buoyancy and turbulence. It is within the efforts described in the previous paragraph where we can place this research, focusing on using *OpenFOAM* as a development platform for DES type models that incorporate buoyancy and stratification in the study of atmospheric flows, but resolving the flow around complex geometries.

We aim our application towards the modelling of atmospheric circulation within open pit mines, with the objective of applying it to study the complex circulation and dispersion of contaminants that originate within Chuquicamata, one of the largest open pit mines in the world, whose ventilation seems to be dominated by convective effects (Barkan & Alpert (2010)). The pit, located in northern Chile (22^{o}17'20"S 68^{o}54'W, about 3000m ASL), measures 4 km long, 3 km wide and 1 km deep, and has a complex topography. Particulate matter is almost always present, as well as air currents produced by natural convection, leading to a strong contamination inside and outside the pit.

The structure of this first part of the work, describing the development of the solver, consists of a first section where we briefly expose the numerical treatment used to deal with the physical problem, including the governing equations and the initial and boundary conditions considered. After that, we present several cases in which we applied the solver and compared its results with values measured or documented in other works. This stage follows a logical progression, from relatively simple problems up to more complex ones that incorporate

multiphysics simulation. Upon finishing, we include a review of the main conclusions of the work, discussing the strengths and weaknesses of the solver, its possible applications and improvements that could be incorporated in the future. We leave for a second paper the application of the solver to open pit mines, including the particular case of Chuquicamata.

2.2 Physical problem and model description

2.2.1 Physical problem

In general the type of atmospheric circulation that we study comprises two main processes that dominate its evolution:

- Mechanical effects caused by the topographic or structural obstacles with which the flow interacts, leading to acceleration or recirculation zones.
- Buoyancy effects caused by the heat flux from or towards the surface, which can generate important vertical air flow accelerations. These convective effects depend on the stability of the atmosphere, so that stratification must be taken into account.

Both processes interact with each other, and are directly influenced by the intense turbulence that characterizes many environmental flows. As we saw, this obligates us to widen the range of scales considered from those used in the study of urban wind to those used in the analysis of atmospheric turbulence, making it necessary to use techniques that are able to work on this range of scales.

2.2.2 Solver

As already mentioned, to address the problem we have selected the CFD toolbox *OpenFOAM*. Although we used as a base the solver *buoyantPimpleFoam* available in the 1.7.1 version of *OpenFOAM*, we modified it to better adapt to our problem. Furthermore, we included specific boundary and initial conditions necessary to deal with the problem in question. We detail the development of our solver in the next sections.

2.2.2.1 The *OpenFOAM* framework

OpenFOAM is a collection of C++ libraries designed to solve complex problems in fluid mechanics. Developed with the desire of obtaining a more effective numerical platform than Fortran, it has benefitted from the new C++ object-oriented programming functionality. Its completely free distribution and the flexibility it offers allows the development of specific solvers by the user, which can be integrated with already existing tools. As most of the CFD codes currently used in engineering, it uses a finite volume formulation. Within its libraries, *OpenFOAM* integrates turbulence models (RANS and LES), thermophysical models,

radiation models and wall functions, which can be accessed when developing a solver. Furthermore, the code offers the functions of integrating complex geometries in the calculation, through the use of the *snappyHexMesh* tool. For an extensive review of the numerical methods used by the code, as well as of the available tools, we refer to the available documentation (OpenCFD (2009)).

There are several examples of the application of *OpenFOAM* to the study of atmospheric flows. García & Boulanger (2006) simulated the wind flow over Mount Saint Helens in the United States employing *OpenFOAM* with a standard $k-\varepsilon$ RANS turbulence model. A similar work was done by Hussein & El-Shishiny (2009) studying the wind flow over the Giza Plateau in Egypt. García et al. (2008) simulated the convective winds generated in the Aburra open valley, near Medellín, Colombia, using standard $k-\varepsilon$ RANS turbulence. Pedruelo (2009) used *OpenFOAM* to model the non-buoyant wind flow over complex terrain using RANS turbulence models with wall functions, to make accurate predictions of wind power production. Churchfield (2010) developed an *OpenFOAM* solver using LES to simulate buoyant flows under the Boussinesq approximation, focusing on the flow interaction with wind turbines. There is apparently no record that *OpenFOAM* has been used to study the circulation caused by buoyant flows within closed valleys with complex topography or obstacles, as open pit mines. In particular, there is no evidence that the new compressible solvers that consider density as a variable of calculation (instead of using the Boussinesq approximation), using a DES approach, have been used for these applications. Our interest lies in studying the problem using this new approach, that allows integrating buoyancy, stratification, turbulence and complex geometry in one single model, with the aim of applying it to the particular case of ventilation within large open pit mines. The DES approach allows us to include complex topography in the analysis, while the explicit inclusion of density variations improves the treatment of stratification and buoyancy effects.

2.2.2.2 Governing equations and implementation in *OpenFOAM*

Given the need to incorporate stratification into the model, we use a quasi-compressible approximation, including density as an explicit variable in the calculation. A detailed view of the algorithms used in *OpenFOAM* to couple pressure and velocity in the compressible case (PISO type algorithms, *Pressure-Implicit with Splitting of Operators*), can be found in Issa (1985), Issa (1986), Oliveira & Issa (2001) and Demirdzic et al. (1993).

When we apply LES, we assume that any variable (f) is decomposed into two components, one large-scale ($\overline{f}$) and the other small-scale (f'), i.e.:

$$f = \overline{f} + f' \, . \tag{2.1}$$

A filtering operation is used to extract the large-scale components, consisting of a convolution with a previously defined function:

$$\overline{f} = \oint G(x, x'; \Delta) f(x') \, \mathrm{d}x' \, , \tag{2.2}$$

where Δ, the filter width, generally depends on the mesh. There are different alternatives for the function G, depending on the numerical discretization to be used. The approach

employed by LES is to directly solve the large scale flow while modelling the effect of the small scales, which will be in general subgrid. There are different SGS (subgrid scale) models to do this. For a description of the manner in which LES is applied in *OpenFOAM* and in the SGS models available, see for example de Villiers (2006).

In our case, considering compressibility effects, we must use a filtered version of the general continuity equation (for simplicity we will not include bars above large-scale variables):

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{U}) = 0 \; . \tag{2.3}$$

The momentum equation corresponds to the filtered Navier-Stokes equation:

$$\frac{\partial}{\partial t}(\rho u_{\mathrm{i}}) + \frac{\partial}{\partial x_j}(\rho u_{\mathrm{i}} u_j) - \nabla \cdot (\mathbf{T} + \mathbf{T}_{sgs}) = \\ -\nabla p + \rho \vec{g} - \nabla p^* + \vec{f}_c \; , \tag{2.4}$$

where $\mathbf{T}$ represents the stress tensor associated with molecular viscosity, μ:

$$\mathbf{T} = \mu \left[\left(\frac{\partial u_{\mathrm{i}}}{\partial x_j} + \frac{\partial u_j}{\partial x_{\mathrm{i}}} \right) - \frac{2}{3} \left(\frac{\partial u_k}{\partial x_k} \right) \delta_{\mathrm{ij}} \right] \; , \tag{2.5}$$

while $\mathbf{T}_{sgs}$ is the Subgrid-Scale (SGS) stress tensor, which will be calculated using one of the subgrid models. For the latter, *OpenFOAM* offers different alternatives. In general, a model is used to estimate a subgrid turbulent viscosity μ_{sgs}. In this work we used one equation SGS model (Yoshizawa & Horiuti (1985)).

The term ∇p^* in (2.4) corresponds to an external force used to maintain the flow. When using cyclic boundaries this is done, for example, computing ∇p^* such that the mass flow rate is constant. When using periodic boundaries this term is used to include a body force. Although in (4) we include Coriolis ($\vec{f}_c = -2\rho\,\vec{\Omega} \times \vec{U}$), in many simulations we do not consider it, to focus on the effect of the other physical processes that control the flow and to simplify the simulation configuration. Furthermore, the Coriolis effect would be very weak in the majority of the cases that we are interested in analyzing. The great advantage of *OpenFOAM* is that each one of the above terms can easily be incorporated or eliminated from the solver that is used for each simulation, given that each application can be compiled independently.

The thermal analysis is incorporated through an enthalpy equation (Satoh (2002)),

$$\frac{\partial}{\partial t}(\rho h) + \nabla \cdot (\rho \vec{U} h) - \nabla \cdot (\alpha_t \nabla h) - rad = \frac{Dp}{Dt} \; . \tag{2.6}$$

Among the benefits of using enthalpy ($h = \mathrm{e} + pv = f(T, p)$) as a variable is the relative simplicity with which mixtures of variable composition can be incorporated into the model, which is a useful characteristic if in the future we include humidity (Akmaev & Juang (2008)). In (2.6), the effective SGS thermal diffusivity, α_t, is equal to the sum of the molecular thermal diffusivity, α, and the subgrid scale turbulent thermal diffusivity, α_{sgs}, i.e.:

$$\alpha_t = \alpha + \alpha_{sgs} \; . \tag{2.7}$$

The value of α_{sgs} is estimated from the subgrid turbulent viscosity μ_{sgs} (calculated by the SGS model) through the Prandtl turbulent number (Pr_t):

$$\alpha_{sgs} = \frac{\mu_{sgs}}{Pr_t} \ . \tag{2.8}$$

There is evidence that in the stratified cases the calculation of α_{sgs} should depend on the level of stratification (in general the Prandtl number is modified as a function of the level of stratification (Pino et al. (2000); Porté-Agel et al. (2001))). However, in this case we have used a fixed value of Pr_t (0.7), trusting that the effect of subgrid turbulence on the calculation is minor. The subject of subgrid turbulent heat transfer is still an area of uncertainty in CFD (Huang et al. (2007); Wang et al. (2007)).

In the enthalpy equation (2.6) we have included a term associated with radiative transfer (rad), which considers the use of the radiation models available in *OpenFOAM* (Finite Volume Discrete Ordinates Model and P1 Method of Spherical Harmonics (Vdovin (2009); Modest (2003))). Divergence of radiation is important in the nightime close to the ground, favoring rapid cooling (Savijarvi (2006)). Although we tested this effect when simulating some simple cases, in general we did not activate radiation in the more complex cases, fundamentally due to the high computing cost of incorporating it and because we are mostly interested in daytime conditions.

We used the ideal gas state equation to close the description of the state of the flow:

$$p = \rho R T \ , \tag{2.9}$$

with $R = 287$ J/(KgK). It is interesting to indicate that in *OpenFOAM* this equation is not incorporated in the files that define the model, as the previous equations, but it is defined when the simulation is prepared through the use of the thermophysical libraries provided by the code.

To include passive scalar transport in the model, we use

$$\frac{\partial \rho C}{\partial t} + \nabla \cdot \left(\rho C \vec{U} \right) - \nabla \cdot (\alpha_t \nabla C) = f_v \ , \tag{2.10}$$

where C is the concentration of the scalar and f_v is a source that we wish to include. We use α_t to approximate effective SGS diffusivity, as with the enthalpy equation, since it is usually accepted that this property is part of the flow and applicable to any property that diffuses within it (Schmidt number $\sim$ Prandtl number in gases (Lim et al. (2009a))). Although the molecular diffusivity of the scalar might be very different than that of the enthalpy, the problem is minor because in DES these contributions might be very small.

In summary, the solver includes the following equations: continuity, momentum conservation (Navier-Stokes), enthalpy conservation, ideal gas state and passive scalar transport, using the libraries and tools provided by *OpenFOAM* to solve them (thermophysical, turbulence and finite volume libraries).

The modular and structured design of *OpenFOAM* allows the equations defined previously to be integrated through different codes (in general a different file for each main equation), all

controlled by a central file that establishes the necessary steps to numerically solve the global problem, and other files that define the necessary variables for the calculation and use of the available libraries. The *wmake* tool, a compiler included in the *OpenFOAM* distribution, is used to compile the code developed. The main strength of the code lies in its modular character, since it easily allows components of the model to be included or eliminated, thus allowing multiphysical problems to be addressed in a clear and ordered manner.

2.2.2.3 Geometry and Meshing

Given our interest in working with complex geometries, we used the *snappyHexMesh* tool to solve the problem of mesh generation. *SnappyHexMesh* proved to be very versatile when applied to different domain configurations (OpenCFD (2010)). In particular, it allows STL files to be used as precursors to mesh generation, making it possible to include complex geometrical forms and even topography in the simulations. In the majority of cases we used the free CAE software *Salome* to generate the necessary STL files. The meshes consisted of hexahedra (hex) and split-hexahedra (split-hex) cells. In cases with complex geometry the mesh was refined near the walls to produce cells with wall normal dimensions between $y_1^+ = 10$ and $y_1^+ = 300$ adjacent to the surface (y_1^+ is the distance in wall units between the centroid of the first cell and the wall assuming the y coordinate is normal to the wall). However, it is impossible to satisfy this criteria everywhere when processes of flow separation and attachment occur.

2.2.2.4 Initial and boundary conditions

In CFD the initial and boundary conditions used are of fundamental importance, because they directly affect the evolution of the simulation and must necessarily take into account the physical processes that we are modelling. Although *OpenFOAM* offers a wide variety of pre-defined conditions, in the majority of case, it is necessary to create specific conditions for each simulation. Furthermore, there exist important differences whether the simulation considers buoyant effects or not, or whether LES or URANS models are used for turbulence. As we focused on DES buoyant simulations we will describe in detail these conditions, while also presenting non-buoyant URANS and DES conditions as reference:

a) URANS
There are different alternatives to define the inlet velocity profile, depending on the information available. If we have experimental data, we can use the expression:

$$U(z) = \frac{U_*}{\kappa} \ln\left(\frac{z - \mathrm{d}}{z_o}\right), \tag{2.11}$$

where U_* (friction velocity), d (displacement height) and z_o (roughness height) are known or can be estimated from data.

If detailed data is not available, we can use a simple power law up to 300 meters, such as

that suggested in Shi et al. (2000), and maintain the velocity fixed at higher altitudes:

$$U(z) = U_s \left(\frac{z}{z_s} \right)^n \ , \tag{2.12}$$

where U_s is the velocity we want to have at height z_s, and $n \sim 0.25$. In URANS simulations this profile is maintained fixed as a forcing during the entire simulation. We also need to define the turbulence kinetic energy and dissipation rate at the inflow boundary, using expressions like those developed in Blocken et al. (2007). The outlet profile of velocity is defined as zero gradient, assuming a fully developed flow. Lateral and top boundaries are defined as stress-free wall: normal component and gradients of tangential components of velocity equal to zero.

b) non-buoyant DES
If we use DES it will be necessary to slightly modify the conditions described above. In particular, the inlet profile of velocity must be perturbed to favor turbulence, for which there exist different techniques (Tabor & Baba-Ahmadi (2010); Xie & Castro (2008)). In *OpenFOAM* it is possible to use the mapping technique to perturb the initial profile, in which a preliminary cyclic computation section coupled into the main calculation is used to give origin to the turbulent perturbation (Baba-Ahmadi & Tabor (2009)). This was the technique used in this work. Also, there exist techniques that numerically generate appropriate turbulent inflows and are relatively easy to implement (Xie & Castro (2008)).

c) buoyant DES
In most complex cases, where we incorporate DES as well as buoyancy through changes in density, it is necessary to use cyclic boundary conditions in all the lateral boundaries, in both directions. This is needed to correctly simulate the heat flux from the surface, since in the atmosphere it can be assumed that it is released from an infinite surface. To avoid initial compressibility effects, it is also recommended that the simulation be started setting a variable vertical profile of velocity (as described in equations (2.11) or (2.12)) but mainly uniform throughout the domain (to avoid an inlet flow). Given that when using cyclic boundaries we cannot set the velocity at entrance, it will be necessary to use an external force (as in equation (2.4)) to ensure that the mean velocity aloft is maintained. This force must be incorporated into the code in such a way that it adjusts during the simulation to reduce velocity variations of the mean flow aloft. When defining an initial velocity profile in the entire domain we can favor the development of turbulence close to the ground perturbing the profile there. To do so, we introduce periodic perturbations along the x and y-axes through sine and cosine functions (de Villiers (2006)).

Attention must be paid to configuring the vertical pressure profile in accordance with the potential temperature profile used, to ensure the initial balance of the system. For example, if using an initial potential temperature profile of the following form on a layer of the domain:

$$\theta = \theta_1 + \lambda \left(z - z_1 \right) \ , \tag{2.13}$$

we use the system of equations:

$$\frac{\partial p}{\partial z} = -\rho g \ , \qquad p = \rho R T \ , \qquad \theta = T \left(\frac{p_o}{p} \right)^{R/c_p} \ , \tag{2.14}$$

to obtain:

$$p(z) = \left(p_1^{R/c_p} - \frac{g p_o^{R/c_p}}{\lambda c_p} \ln \left[1 + \frac{\lambda}{\theta_1}(z - z_1) \right] \right)^{c_p/R} , \tag{2.15}$$

which we use to define the pressure profile within the sub-layer in question.

In *OpenFOAM*, when boundary conditions are defined, there are in general two available alternatives: to set the value of the variable in the surface or to set its gradient. Unfortunately, when the heat flux is defined at the walls, neither of the two alternatives is a good choice. The value of the coefficient of effective SGS thermal diffusivity, that controls the subgrid calculations that are dominant close to walls, evolves along with the variations in the flow, and therefore we cannot prescribe the heat flux unless we take into account the possible local variations of this coefficient. To avoid this problem we defined specific boundary conditions that use the value of α_t (equation (2.7)) to estimate the temperature gradient necessary to obtain the required heat flux. This temperature gradient is then employed to compute the temperature imposed as boundary condition. Through simple simulations, where we set a surface flux and calculated the total energy transferred to the domain in a determined amount of time, it was possible to clearly verify the proper functioning of this type of boundary condition.

2.2.2.5 Treatment of walls

OpenFOAM offers a series of specialized libraries to define the boundary conditions on a surface. It has been shown (Franke et al. (2007); Tominaga et al. (2008); Franke et al. (2004)) that in the case of complex configurations that include a set of buildings or obstacles, it is preferable to use a smooth wall condition, given that a rough wall model requires a large grid, which makes it difficult to capture the details of the flow in complex geometries. Because our final motivation was to simulate the flow inside open pits with complex topographies and structures, we focused on using a mesh that includes the topography inside the pit, instead of using surface roughness. Although different works have shown rough walls to be crucial in maintaining the correct ABL profiles along the upwind fetch, in this work any effect of the surface roughness is assumed to be small compared with the effects of the terrain, as suggested in Silvester et al. (2009).

There are many considerations to take into account when using wall functions in RANS simulations, considering the advantages and limitations of each wall model. For an extensive review, see Blocken et al. (2007).

In the case of LES simulations the treatment of the flow close to the walls is especially complex, since, in general, the computational cost of applying this technique in these areas is too high, seriously limiting its application in the case of complex geometries (Wyngaard (2004); Spalart (2009)). However, the excellent results obtained when LES is applied to areas of flow separation, where most RANS techniques fail, have led in the past few years to the development of a series of hybrid techniques. The DES technique (detached-eddy-simulation), originally proposed by Spalart (Spalart et al. (1997)) to deal with highly separated flows

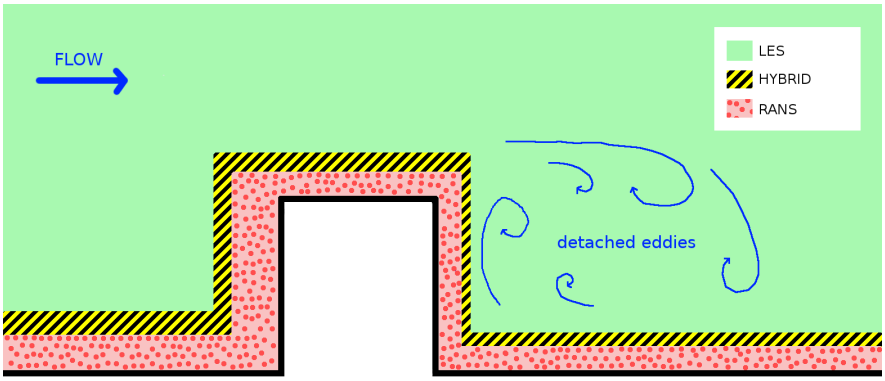


Figure 2.1: Typical turbulence zones in a DES simulation, adapted from de Villiers (2006).

such as those existing in the aerospace industry, combines the benefits of LES and RANS, dividing the domain according to different turbulence zones, as shown in figure 2.1. These characteristics make DES especially useful in the case of atmospheric flows around complex geometries, allowing us to benefit from the excellent LES results in zones with highly developed turbulence far from the ground, and from the RANS methods widely used in wind engineering near walls. For a review of the wall functions available in *OpenFOAM*, as well as the DES techniques used in the code, refer to de Villiers (2006). The results shown in this work correspond to DES simulations using Spalart Allmaras model for the near-wall treatment, although we also experimented with different standard wall functions using URANS simulations to compare results.

2.2.2.6 Computing support

The development of the model, the configuration of the simulations, and the analysis of the results were done on a personal workstation, running *OpenFOAM* and the *Paraview* viewer. Due to the high computing requirements of the simulations, with domains exceeding 5 million cells, their execution was done in parallel in the Levque cluster of the National Laboratory for High Performance Computing (NLHPC) of the Center for Mathematical Modeling of the University of Chile. The Levque cluster is an IBM iDataplex machine with 536 cores, equipped with Intel Nehalem processors, an Infiniband QDR switch and different development tools (Maureira et al. (2011)). Different *OpenFOAM* scaling tests (Jasak (2010)) were carried out in the cluster, one of which is presented in figure 2.2. Since the scaling depends on different factors, such as the complexity of the geometry used in the domain, the type of decomposition used, or the turbulence models selected, the results of figure 2.2 are only referential. Bottlenecks reducing the scalability of *OpenFOAM*, and linked to the linear algebra core libraries of the code, have been reported in the literature (Culpo (2012)).

2.3 Results and discussion

Considering the complexity of the problem to be adressed, with several physical processes interacting in a complex geometry, at different scales, as we moved forward in our work

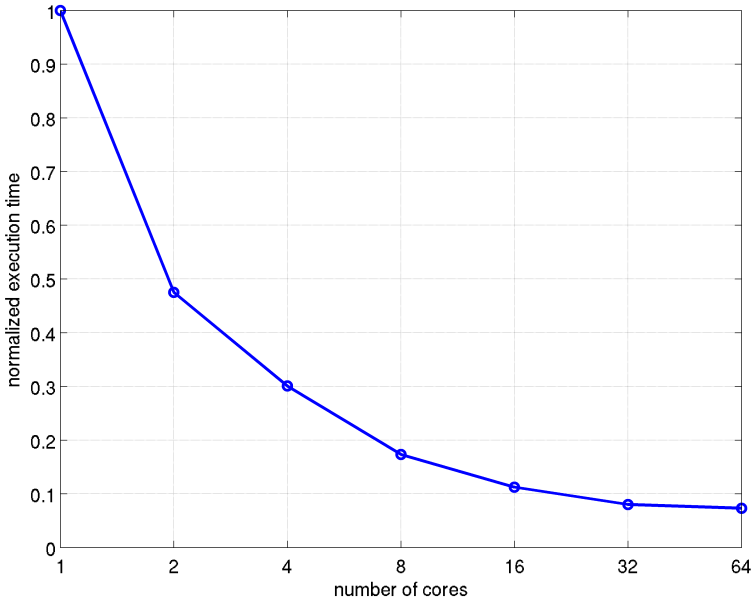


Figure 2.2: *OpenFOAM* scalability in cluster Levque, execution time v/s number of cores used.

we chose to perform different levels of validation at each stage of solver development. The logic followed in this process was to gradually incorporate processes to the simulation. We began with a simple case of atmospheric flow around a single obstacle, then we used complex geometries of small and large-scale, and finally we incorporated buoyancy, first with a simple experimental case and then with a well documented atmospheric case. At each stage we compared our results with available data or other simulation efforts. The results shown in each case correspond to DES simulations with OneEquationEddy as the SGS model and Spalart Allmaras wall treatment, although we also performed several URANS simulations with RNG $k-\varepsilon$ turbulence model in order to compare the perfomance of both techniques. A fully second-order method in space was used for all simulations. A second order implicit method (backward differentiation) was used for the time integration. We used linear and limited linear differencing schemes for convective terms approximation. Two correctors were set for a PISO loop and 0, 1 or 2 correctors (depending on the complexity of the mesh) for non-orthogonal corrections. Preconditioned (bi-) conjugate gradient method with incomplete-Cholesky preconditioner was used for solving the linear systems with a local accuracy of 10^{-6} for all dependent variables at each time step. In cases with simple geometry we used the mesh generation utility *blockMesh* to create meshes of hexahedral cells in 3-D (*polyMesh* in *OpenFOAM* (OpenCFD (2009))), changing the grid size by changing the number of cells in each direction. In cases with complex geometry we used the toolbox *snappyHexMesh* to modify a previously defined *blockMesh* mesh (background hex mesh) and adapt it around complex surfaces defined by means of a STL file, creating hexahedra (hex) and split-hexahedra (split-hex) cells. In these cases the grid resolution was adjusted changing the minimum and maximum refinement level in the *refinementSurfaces* dictionary, and modifying the number of patch smoothing iterations before finding correspondence to a surface with the *snapControls* sub-dictionary. In all cases the meshes generated were tested by means of the *checkMesh* utility. In general, the time step used in each simulation was chosen in order to maintain the Courant number below 0.5.

2.3.1 Non-buoyant cases

In the first stage of model development we were interested in studying the interaction of the flow with complex geometries, so we focused on the use of wall treatment models and the generation of complex meshes. In this early stage we did not consider buoyancy effects, so a simplified version of the solver developed in section 2.2.2 was used, which did not include any thermal effect (enthalpy equation (2.6) was not included). In these non-buoyant cases we used initial and boundary conditions as those described in section 2.2.2.4 (non-buoyant URANS and DES).

2.3.1.1 Simple experimental cases

As a first step in studying the interaction of the flow with obstacles, we used the ERCOFTAC library (European Research Community on Flow, Turbulence and Combustion, http://cfd.mace.manchester.ac.uk/ercoftac/), and compared the results of our simulation of atmospheric flow around a 6-m cube with reported measurements (Case 84 Turbulent Boundary Layer Flow Over a Cube, Lim et al. (2007, 2009b)), with good results. Figure 2.3 shows 20 second averages after 2 minutes of simulation, using three different meshes. The simulation was able to resolve the recirculation over the cube (negative velocity in the figure), and the pressure changes associated to the flow-obstacle interaction. A mesh refined in the vertical direction improved results, while the use of *snappyHexMesh* improved the resolution near the edges of the cube. The flow and pressure patterns observed in our results are also in accordance to simulation results obtained by other authors using similar test cases (Lim et al. (2009b)).

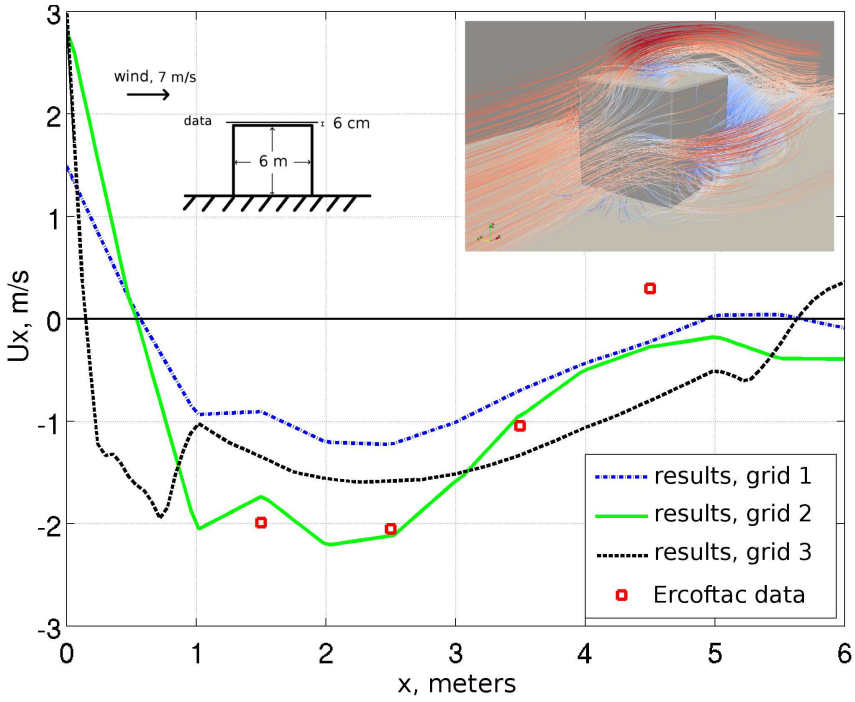


Figure 2.3: Flow recirculation near top of a 6 meters cube, simulation v/s ERCOFTAC field data (red squares). X-velocity component (m/s) along a line 6 cm above the top surface of the cube. Inlet wind speed aloft $\sim$ 7 m/s. Blue line, grid size $h/12 \times h/12 \times h/48$; green line, grid size $h/12 \times h/12 \times h/96$, with h the height of the cube; black line, *snappyHexMesh* mesh.

The next step was to study the effect of more complex geometries, like a non-cubical obstacle or various obstacles, for which we used data available in the database of the Architectural Institute of Japan AIJ (Tominaga et al. (2008)). A summary of the results of this second case, in which we simulated the flow around a rectangular obstacle like that set out by Mochida et al. (2002), is seen in figure 2.4 (5 second averages after 1 minute of simulation). The simulation was able to represent in detail the most important characteristics of the flow, including the recirculation that occurs behind the obstacle and the perturbation that occurs over it. Our DES results are similar to those presented in Mochida et al. (2002) and Yoshie et al. (2007) using different $\kappa-\varepsilon$ RANS models, in particular the velocity distribution behind the obstacle and the turbulent energy over the roof.

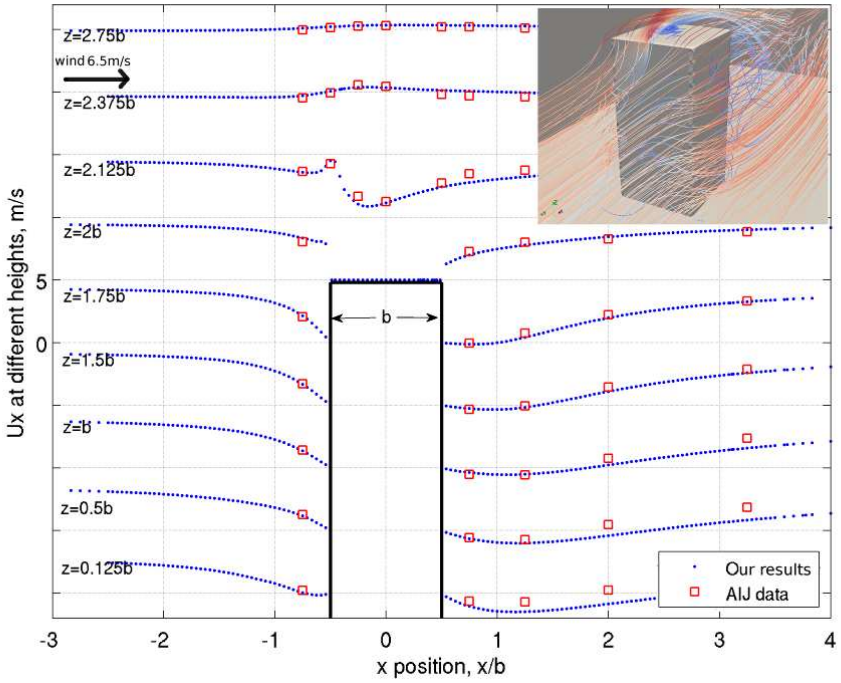
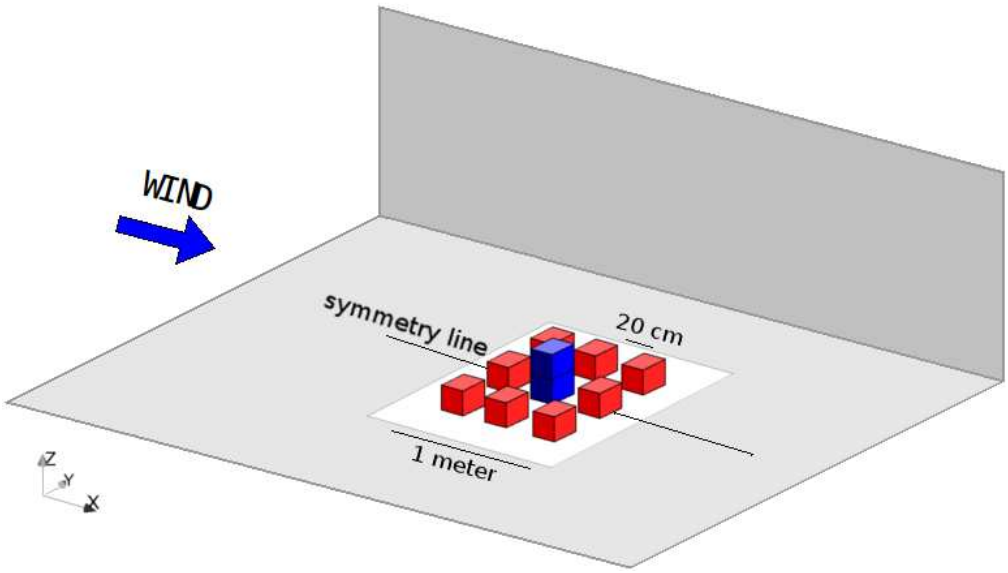


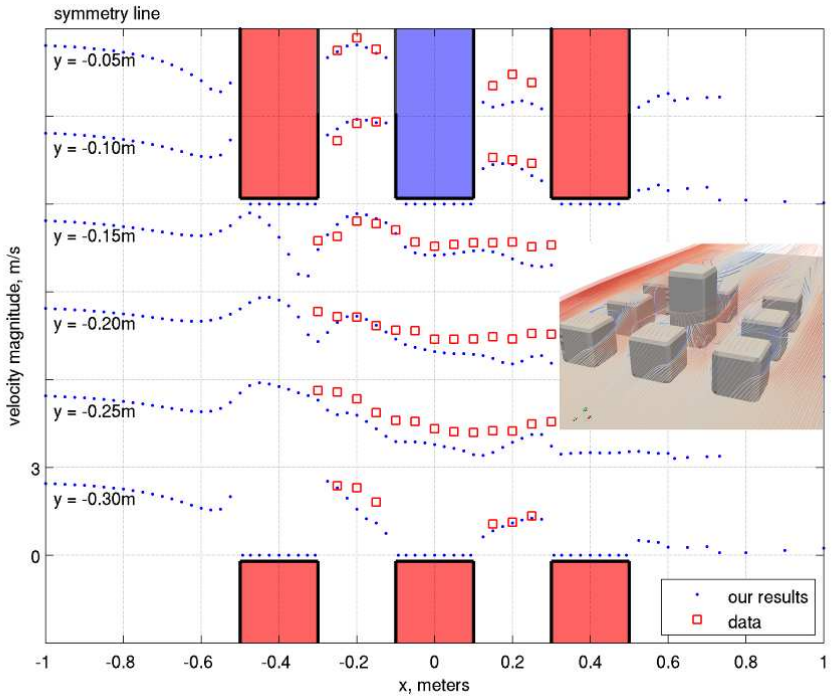
Figure 2.4: Flow around a $b \times b \times 2b$ box ($b = 8$ cm), simulation (blue dots) v/s AIJ experimental data (red squares). X-velocity component (m/s) in lines at different heights on a vertical plane y=0. Geometrical scales were modified to improve data representation. Inlet wind speed aloft ~ 6.5 m/s.

The third experimental test case corresponds to a complex geometry consisting of a set of 9 obstacles (Takayoshi (2003) and Yoshie et al. (2005), figure 2.5a). Given the complexity of the geometry of this case, with different-sized obstacles distributed over a large domain, we used the *snappyHexMesh* tool to generate the mesh, refining the most complex zones by the use of *snappyHexMesh* controls, to produce cells with wall normal dimensions between $y_1^+ = 10$ and $y_1^+ = 100$ adjacent to the surface. Different meshes were used until obtaining grid independent results. The inlet profile of velocity was adjusted using the experimental data provided by the library ($\sim$ 6 m/s wind speed aloft), using the mapping technique available in *OpenFOAM* to initiate turbulence. Part of the results obtained are seen in figure 2.5b, corresponding to a horizontal plane close to the ground (10 second averages after 2 minutes of simulation, although different combinations of final and averaging times have been tested to ensure that the final converged time-averaged results were attained). The simulation results are consistent with experimental data. In particular, the simulation was able to solve the flow established between the obstacles, and in the wake generated by them. Also, our DES

results are in accordance with the distribution of turbulence energy around a simple city block reported by Takayoshi (2003) using RANS simulations.



(a)



(b)

Figure 2.5: Simple city blocks (side 20 cm). a) experimental test case, AIJ, adapted from Tominaga et al. (2008). b) simulation (blue dots) v/s AIJ experimental data (red squares). Velocity magnitude (m/s) in lines at different positions with respect to y-axis, on a horizontal plane near the ground (z = 0.02m) (bilateral symmetry across symmetry line). Geometrical scales were modified to improve data representation. Inlet wind speed aloft $\sim$ 6 m/s.

2.3.1.2 Complex experimental cases

In order to test the versatility of the *snappyHexMesh* tool in generating complex large-sized meshes, such as those we are interested in working with, and to analyze how these meshes interact with the numerical methods available in *OpenFOAM*, we used more complex configurations as study cases.

The first complex case corresponds to the geometry of the MUST project (Mock Urban Setting Test, Biltoft (2001)), consisting of an arrangement of 12 by 10 containers, each 12.2 m long, 2.4 m wide and 2.5 m high, with the total size of the arrangement being 193 by 171 meters, whose configuration in *OpenFOAM* is seen in figure 2.6. In this case the capacity of *snappyHexMesh* to refine the mesh in critical zones, such as the edges of the containers, was made evident, achieving a mesh completely integrated with the numerical methods. Different levels of refinement near the containers were used until reaching a grid independent result.

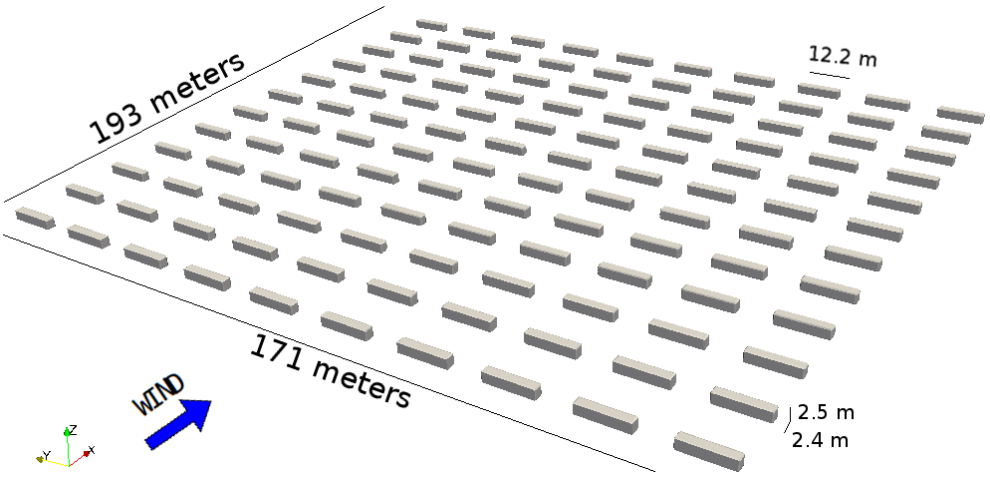
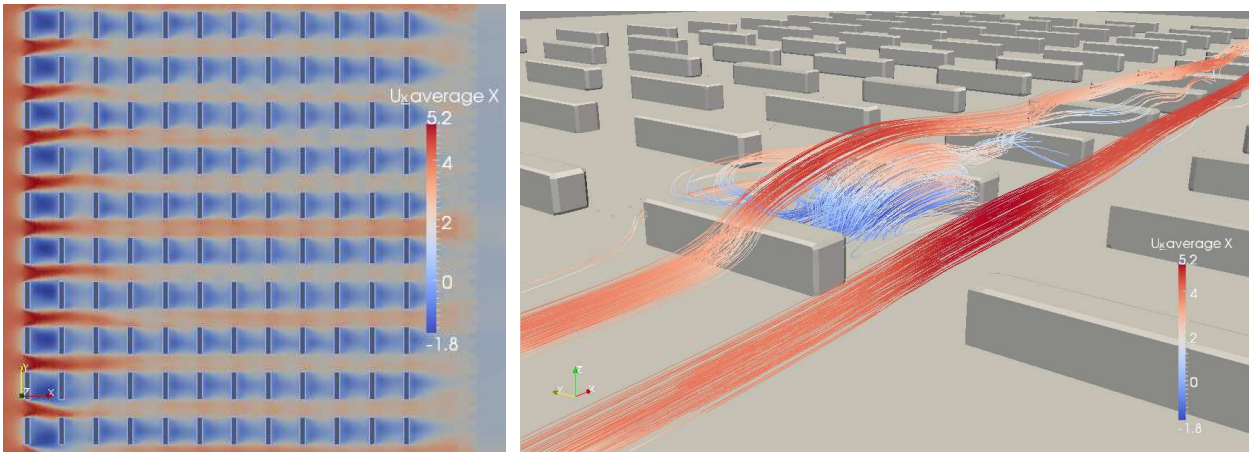


Figure 2.6: *OpenFOAM* model of the 120 containers of the project MUST.

Although in this case we do not have access to a complete library with which to validate our results, they are comparable to those documented in similar works (Dejoan et al. (2008)), showing recirculation zones in the back of each container and acceleration zones between them (figure 2.7).

The second complex large-sized mesh used corresponds to a simplified version of the buildings of the School of Physical and Mathematical Sciences of the University of Chile (FCFM), figure 2.8. We used a mesh generated by *snappyHexMesh* and we performed different simulations changing the incident wind direction (southwest, west and northwest wind). The grid exceeded 3 million cells, refined close to the buildings ($\Delta < 1$ m), and was chosen after several mesh tests until obtaining solutions independent of domain size and mesh. The mesh was refined near walls, by the use of *snappyHexMesh* controls, to produce cells with wall normal dimensions between $y_1^+ = 10$ and $y_1^+ = 300$ adjacent to the surface. We simulated 3 hours (until reaching a statistically stationary flow) with a time step of 0.05 s, using inlet conditions as those described in the non-buoyant cases of section 2.2.2.4. The inlet profile of horizontal velocity was assumed to be distributed as a power law in the vertical direction



(a) **(b)**

Figure 2.7: Averaged results of the MUST case. a) Horizontal plane in the middle of the containers. Colors represent the value of the X-velocity component. b) Streamlines over one container (recirculation), and in the free zone between two (acceleration). Colors represent the value of the X-velocity component.

with the index $n = 0.25$ (equation (2.12)), with a representative velocity of 2 m/s at a reference height of 3 meters. The initial inlet boundary for subgrid turbulent kinetic energy (k) is not simple and is unknown. We fixed an inlet value of 0.001 m^2/s^2, trusting that it would be adjusted quickly once the simulation starts. The values of velocity and k were perturbed using the mapping technique described previously and available in *OpenFOAM*.

The last hour average results for a southwest wind, the most common condition in this location, are shown in figure 2.9, which corresponds to a plane 3 meters above the ground. As is seen through the color scale, the simulated flow records a local maximum within the building complex (location 2), which coincides with the subjective appreciations of those who transit there frequently.

There are strong recirculation zones inside the domain, linked to complex geometrical patterns, a well-documented feature in urban wind (Fernando et al. (2010); Zhang et al. (2005)), driven by pressure gradients produced as the flow interacts with obstacles. Also, the simulation displays strong turbulence inside the building complex (large perturbations in wind speed, standard deviation $\sigma(U) \sim 1$ m/s).

In order to have a partial validation of these results we carried out a small field campaign in the building complex, installing 4 sonic anemometers (R.M. Young Model 8100) and 2 conventional anemometers (locations shown in figure 2.9), aimed at identifying the existence of a local maximum in the flow velocity. Anemometer B was located on the roof of one of the buildings, and was used to verify the main wind conditions, while the others were located 3 meters above the ground.

The data generated by anemometers 1, 2 and 3 during an afternoon that fulfills the southwest wind condition are shown in figure 2.10. The magnitude of the wind measured by anemometer 2 exceeded, at all times, the records of anemometers 1 and 3, consistent with

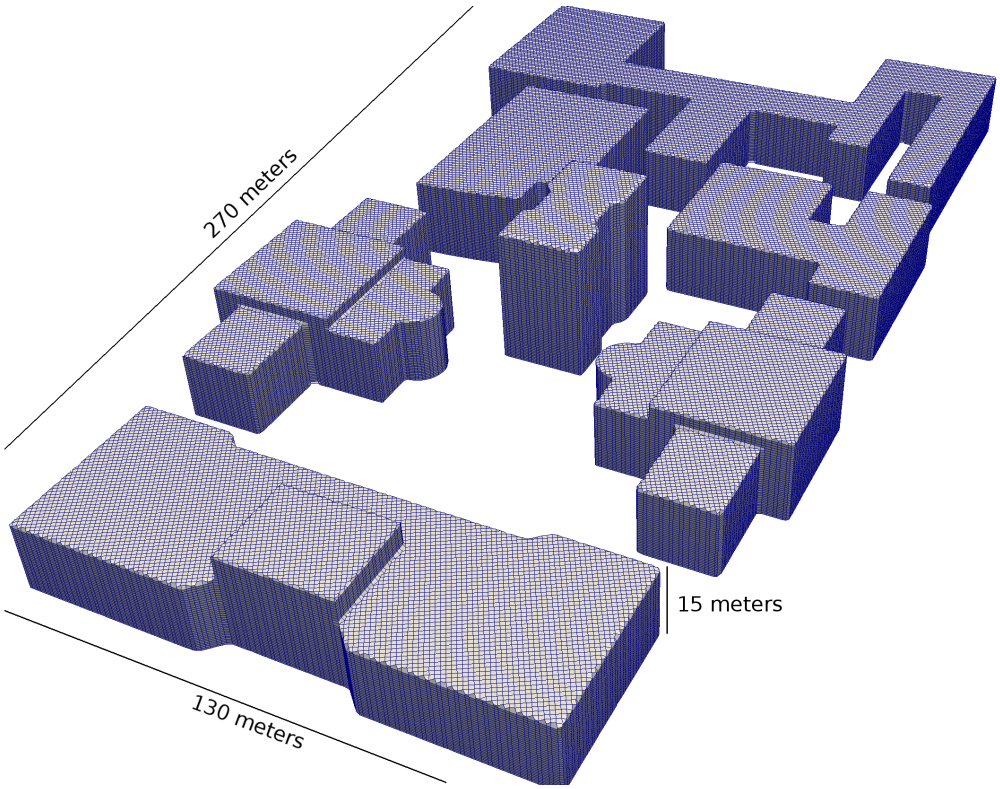


Figure 2.8: FCFM building complex generated by using the *snappyHexMesh* tool. Mesh refined close to the buildings.

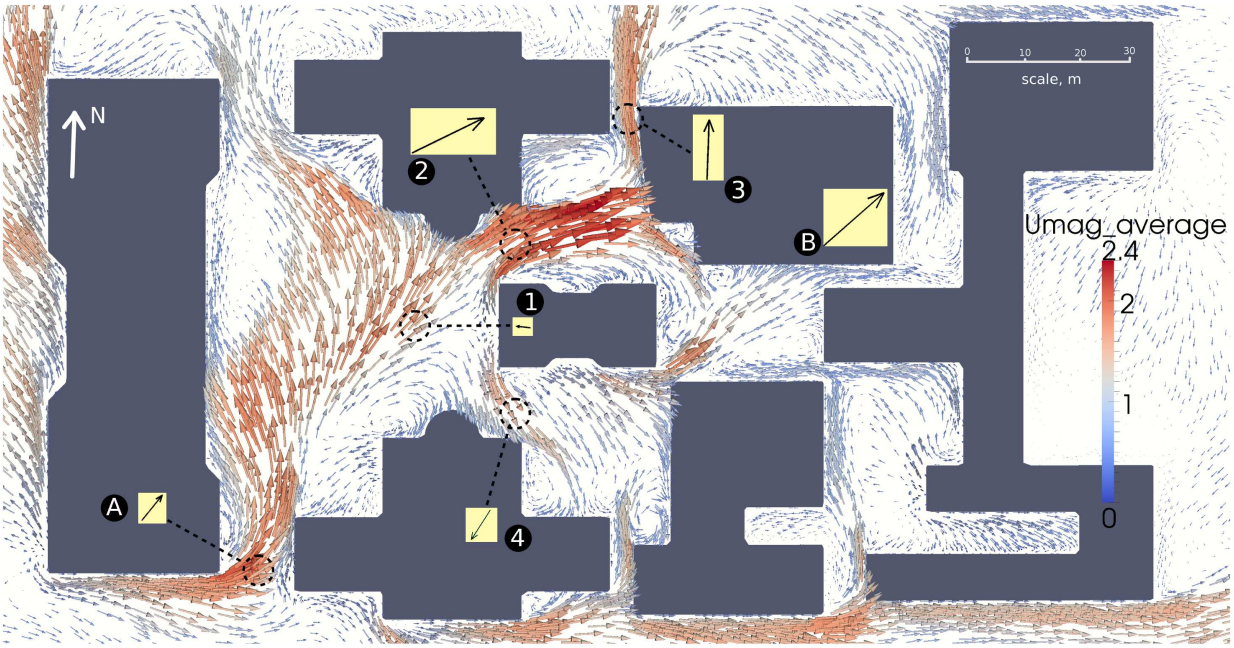


Figure 2.9: Simulation results inside the FCFM buildings complex (mean velocity vectors 3 meters above ground, m/s). Mean velocity vectors recorded by anemometers are shown as displaced vectors. Segmented circles indicate the location of sonic anemometers 1, 2, 3 and 4, and conventional anemometers A and B.

the averaged results of the simulation. This feature is also present in other afternoons that fulfill the southwest wind condition (not shown), and when comparing the values recorded by sonic anemometer 2 and sonic anemometer 4 in another day different than that used for the simulation (figure 2.11, different day than that shown in figure 2.10).

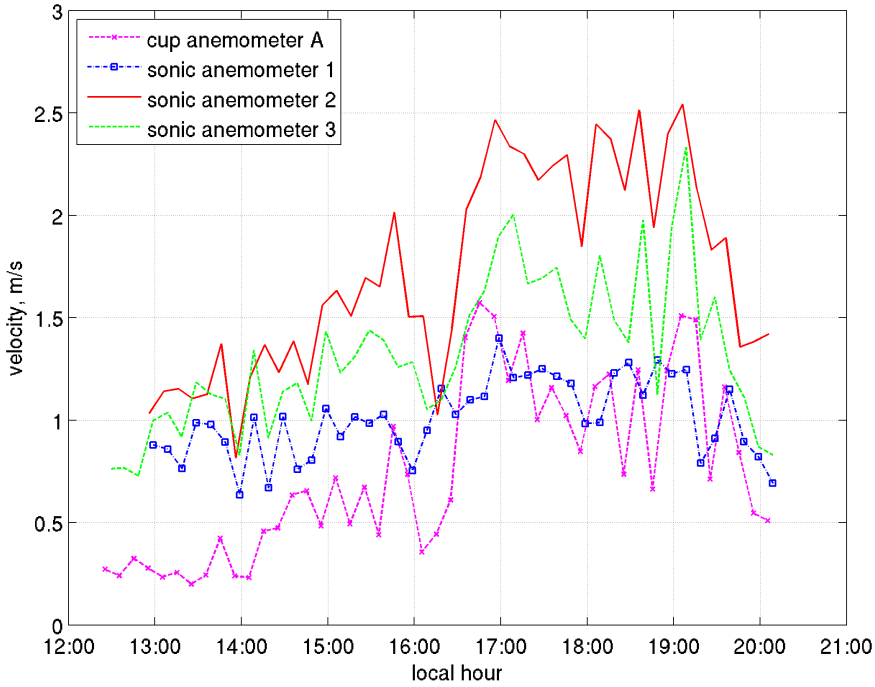


Figure 2.10: Anemometer data recorded during one of the days sampled, 10 minutes mean horizontal velocity magnitude (m/s) versus local time.

The average velocity vectors recorded by the anemometers are shown in figure 2.9 as vectors displaced from the location of measurement. In the flow acceleration zones (locations 2 and 3), the simulated values correctly approximate the measured values in magnitude ($\sim$2 and $\sim$1.5 m/s respectively), and in direction. On the contrary, the simulated values at point 1 differ from measurements, that record low-intensity counter flow winds. These differences may be attributed to obstacles that were not included in the simulation, in particular a 3 m terrace located in front of the central building that appears to be responsible for the flow perturbations in this area. The larger variability at point 1 is evident when comparing the frequency distribution of the wind directions measured by the anemometers at different locations (figure 2.12). A clear prevailing wind direction is not seen at location 1, in contrast to what happens at locations B, 2 and 3. U_x (velocity component aligned with the main distribution of buildings) statistics from the simulation coincide with those from data sampled by sonic anemometers, confirming the existence of intense turbulence inside the building complex (table 2.2).

Even if the evidence of this particular test case is qualitative at best it supports the validity of using DES type simulations to solve the atmospheric flow around complex geometries, and demonstrates the ability of the *snappyHexMesh* tool to generate fully operational complex meshes, which was our primary objective for this section of the work.

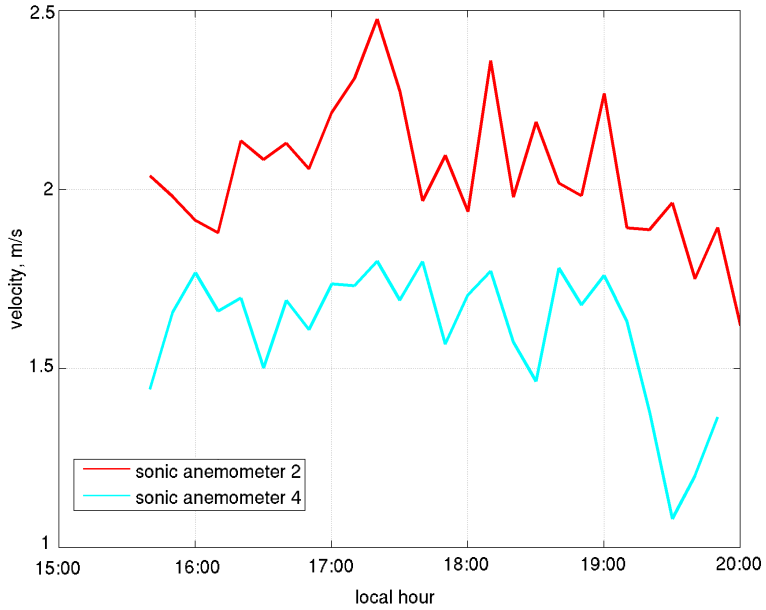


Figure 2.11: Anemometer data recorded during one of the days sampled (different day than that shown in figure 2.10), 10 minutes mean horizontal velocity magnitude (m/s) versus local time.

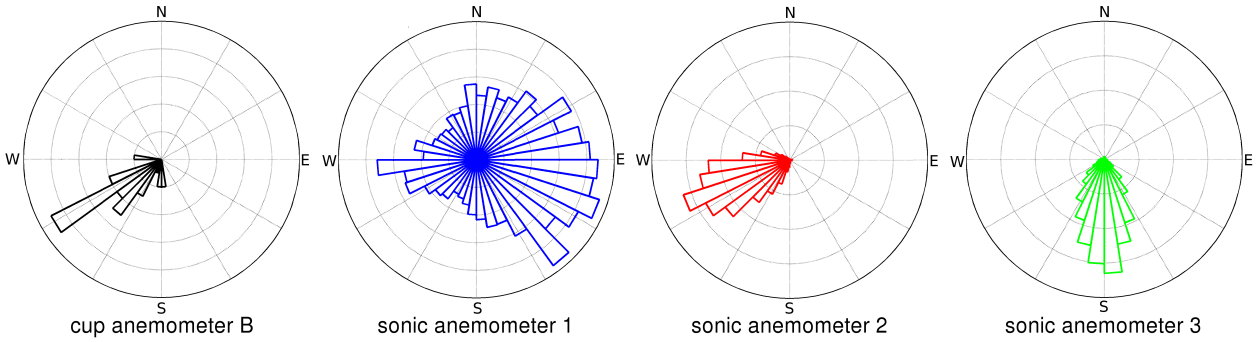


Figure 2.12: Wind roses of recorded data, location B (over the roof) and sonic anemometers 1, 2 and 3 respectively (locations shown in figure 2.9).

Table 2.2: Standard deviation (σ, m/s) of U_x (velocity component aligned with the main distribution of buildings) in selected points inside the domain (locations shown in figure 2.9). Field data (sonic anemometers sampled each 0.25 s) and simulation data (time step 0.05 s, resampled each 0.25 s).

point	**standard deviation $\sigma(U_x)$, m/s**	
	measured	**simulated**
1	0.7	0.6
2	0.6	0.5

2.3.2 Buoyant Cases

None of the cases above does consider thermal effects on the flow, as they focus on the interaction of the flow with obstacles and complex geometries. The next step, therefore, was to validate the results of buoyant cases. We selected two cases, first a simple experimental one with detailed data available, and then a complex atmospheric case well documented in the literature. We will specifically focus our attention in the second case, because its conditions are similar to those affecting large open pit mines under intense solar insolation, as Chuquicamata.

2.3.2.1 Simple experimental case

We used experimental data from the ERCOFTAC library, with case 79, Turbulent Natural Convection in an Enclosed Tall Cavity (Betts & Bokhari (2000)). The case consists of a closed rectangular cavity 2.18m high by 0.076m wide by 0.52m in depth, whose lateral walls are subjected to 20 ^{o}C differential heating (ΔT), with a Rayleigh Number, based on the width of the cavity (L), of Ra=0.86x10^{6} (figure 2.13).

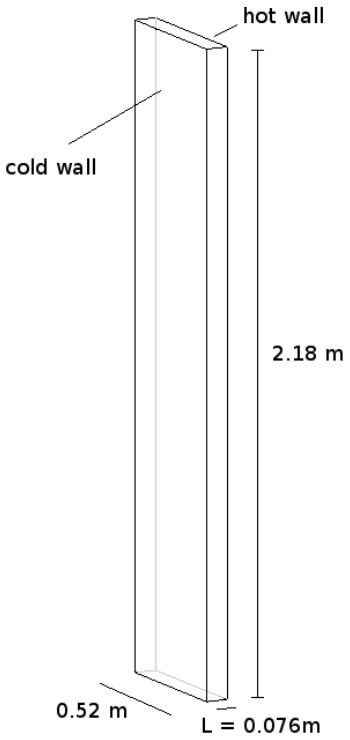


Figure 2.13: ERCOFTAC library, case 79, Turbulent Natural Convection in an Enclosed Tall Cavity.

The boundary conditions used in this case are particularly simple, since we deal with an enclosed flow: zero velocity at walls, fixed temperature at vertical walls, and adiabatic upper and lower horizontal walls. We used a mesh of 64x52x218 grid points and a time step of 0.001 seconds for a full simulation time of 20 minutes (reaching a statistically stationary flow). Similar results were obtained refining the mesh in each axis. Wall normal dimensions

between $y_1^+ = 5$ and $y_1^+ = 20$ adjacent to the surface were used. The buoyancy velocity (V_o) was calculated based on the width of the cavity (L):

$$V_o = \sqrt{g\beta L\,(\Delta T)} = 0.2 \text{ m/s}\,, \tag{2.16}$$

with β the thermal expansion coefficient. V_o scales as the vertical mean velocity near the walls. We can define a reference time as the total height of the cavity divided by the calculated buoyancy velocity: $t_{ref} = H/V_o \sim 10$s, and introduce a dimensionless time $\tau = t/t_{ref}$.

Figure 2.14 shows the instantaneous distribution of temperature at final time. Turbulence linked to buoyant currents is present.

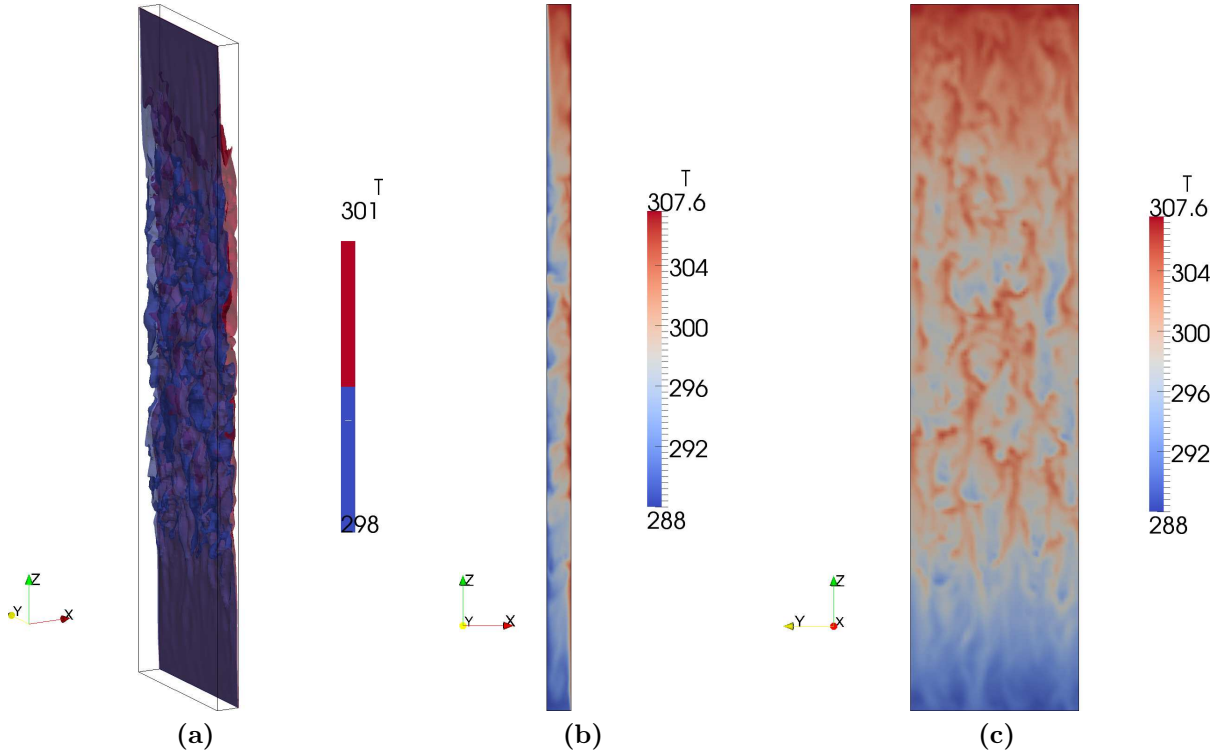
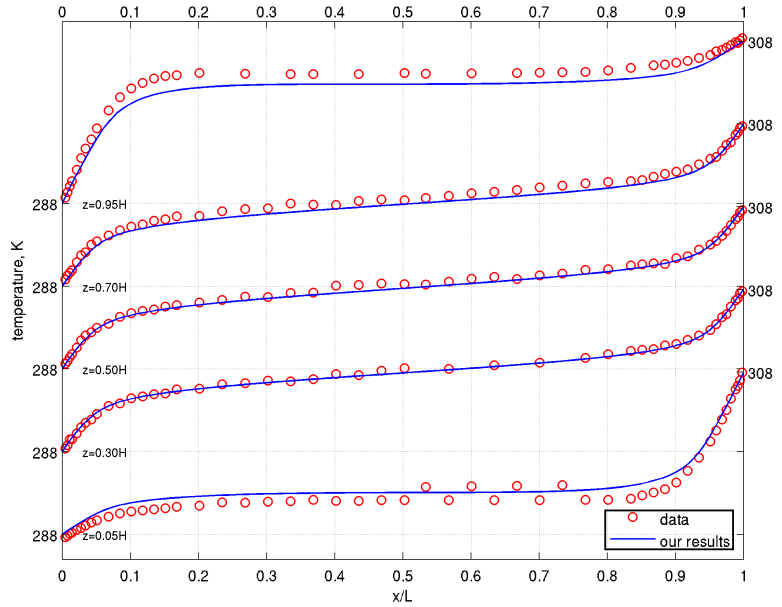


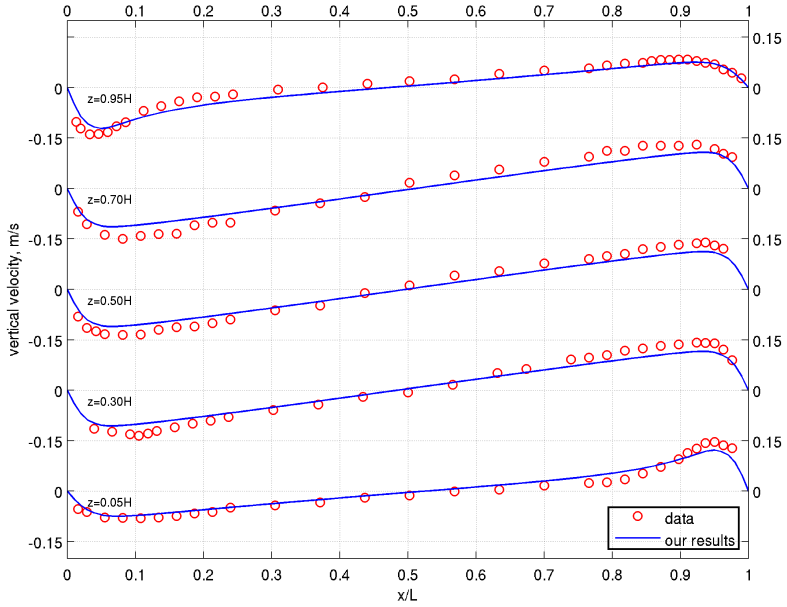
Figure 2.14: Instantaneous distribution of temperature (Kelvin) at final time. a) 3D Isoterms. Only two isoterms are shown in order to improve visualization. b) Vertical XZ plane in the middle of the domain. c) Vertical YZ plane in the middle of the domain.

In figure 2.15 we compare our results (100 second averages ($\tau = 10$) at final time ($\tau = 120$)) with temperature and velocity experimental data for horizontal sections between both walls at different heights above the bottom. Different combinations of final and averaging times have been tested to ensure that the final convergent time-averaged results were attained. The model is capable of correctly simulating the distribution of the temperature (hotter wall to the right and colder to the left), as well as the circulation caused by the changes in density (rising along the right wall, falling along the left wall). The asymmetry in the distribution of velocity and temperature generated by the upper and bottom boundaries can be distinguished in the results (lower velocity of the flow that impacts against the boundary). In contrast, at the center of the cavity, far from the upper and lower boundaries, the profiles

are rotationally symmetric. Our DES results are similar to those obtained by Hsieh & Lien (2004) using unsteady RANS with a low-Re $k-\varepsilon$ model (figures 3 and 4 of that work).



(a)



(b)

Figure 2.15: a) Temperature (K) and b) vertical velocity (m/s), in a line linking cold and hot walls at different heights, simulation (blue line) v/s ERCOFTAC experimental data (red circles). H is the total height of the closed cavity and L is the separation between the walls at a different temperature.

2.3.2.2 Turbulence in the atmospheric boundary layer

Finally, we studied the efficiency of the solver in simulating buoyant flows on a large-scale, following the work by Moeng & Sullivan (1994) and Churchfield et al. (2010) (hereafter referred to as MS94 and CH10, respectively). Both works used LES (pseudo-spectral solver in MS94, finite volume solver in CH10), to accurately simulate the atmospheric boundary layer under different combinations of wind shear and buoyancy forces.

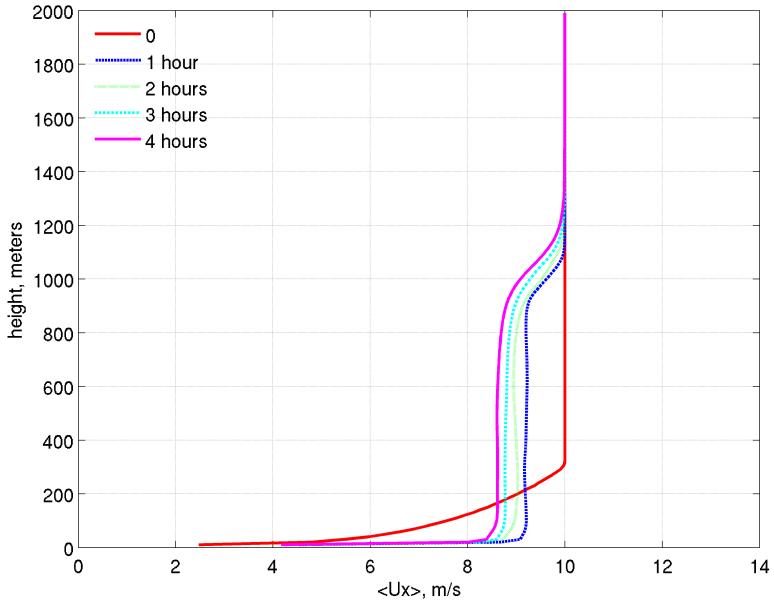
The study case corresponds to a horizontal domain of 5 x 5 km and 2 km height. The results shown correspond to a mesh of 160x160x128 grid points, like that used in CH10, although similar results were obtained with a 96x96x96 mesh, like that used in MS94. A fixed surface heat flux of 240 W/m^2 and a 10 m/s wind speed above 300 meters are prescribed. u_*, the friction velocity, has a value of 0.56 m/s, while the convective temperature scale ($T_* = q_s/w_*$) is 0.12 K. The initial potential temperature profile has a constant value of 300 K up to 937 m, followed by an 8 K increase in the next 167 m, and a constant 0.003 K/m gradient up to the top of the domain. We employ cyclic lateral boundary conditions in both horizontal directions. The upper boundary conditions are free-slip and maintain a constant potential temperature gradient, specified to be that of the initial capping inversion profile (0.003 K/m). To facilitate the initial development of turbulence the initial velocity field was perturbed as suggested by de Villiers (2006), introducing periodic perturbations along the x and y-axes through sine and cosine functions. The time step used was 0.25 seconds, for a full simulation time of 4 hours. In this case we included Coriolis in the simulation (equation (2.4)) and kept a 10 m/s geostrophic wind above the boundary layer.

The time evolution of velocity and potential temperature profiles are shown in figure 2.16 (horizontal averages over the entire computational domain). The convective turbulent mixing generated by buoyancy occurs below the first inversion, generating a clearly identifiable convective boundary layer (CBL). In the case of velocity profiles our results compare well with those reported in CH10, showing the development of a boundary layer with less wind than the layer above it, due to the turbulent momentum transfer to the surface. The profiles of potential temperature show the heating of the air below the first inversion, the increase of the boundary layer height (after 9000 seconds $z_\mathrm{i} \sim 980$ meters), and the entrainment at the top of this layer. Also evident is a weak stabilization of the temperature profile in the upper half of the CBL. These features are similar to those documented by Moeng (1984), Stull (1988) and Wyngaard (2010). We integrate the net heating of the temperature profile to estimate the average heat flux received by the entire volume:

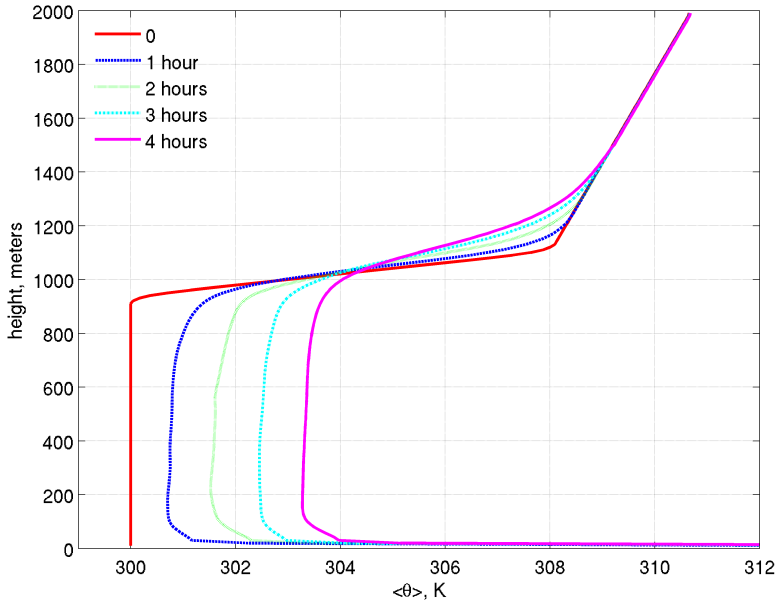
$$\overline{Q} = \frac{1}{t_f} \int \rho c_p \left(\theta_f - \theta_0\right) \, \mathrm{d}z, \tag{2.17}$$

where t_f is the final time, c_p the specific heat of dry air, ρ the air density profile, and θ_f and θ_0 the final and initial vertical profiles of potencial temperature (horizontal averages over the entire computational domain). We obtain $\overline{Q} \sim 243$ W/m^2, very close to the imposed surface heat flux (240 W/m^2).

The existence of powerful thermals, which favor the mixing and occur over the entire domain, is clearly shown in the instantaneous concentration of a passive tracer released from the surface (figures 2.17 and 2.18).



(a)



(b)

Figure 2.16: Time evolution of vertical mean profiles (horizontal average over the entire computational domain). a) horizontal velocity, m/s. b) potential temperature, K.

The full model time series of vertical velocity and potential temperature in a point at 500 meters above the ground in the center of the domain, are shown in figures 2.19a and 2.19b. The presence of thermals is evident in figure 2.19a, by maximum vertical velocities that reach up to 4 m/s. The root mean square value of these perturbations ($\sigma_w \sim 1.6$ m/s) is of the

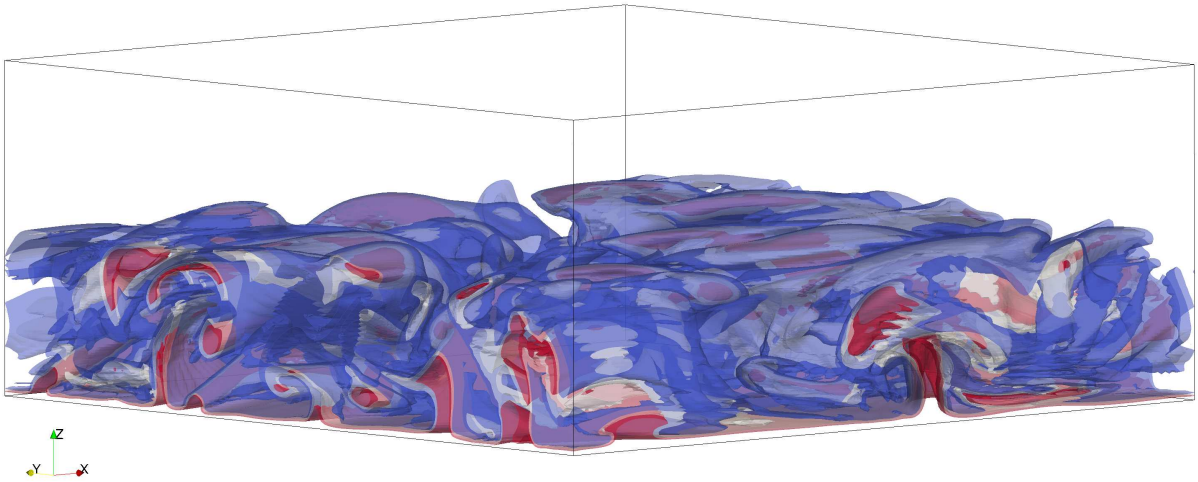


Figure 2.17: Instantaneous concentration of a passive tracer released from the surface.

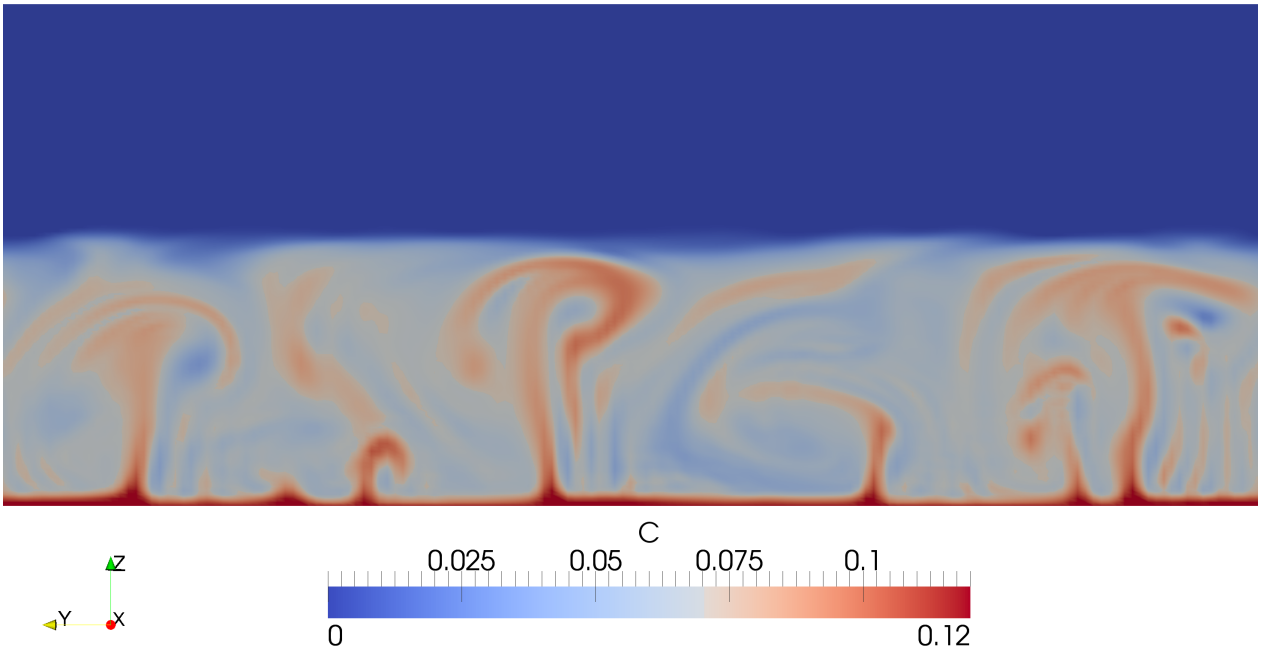
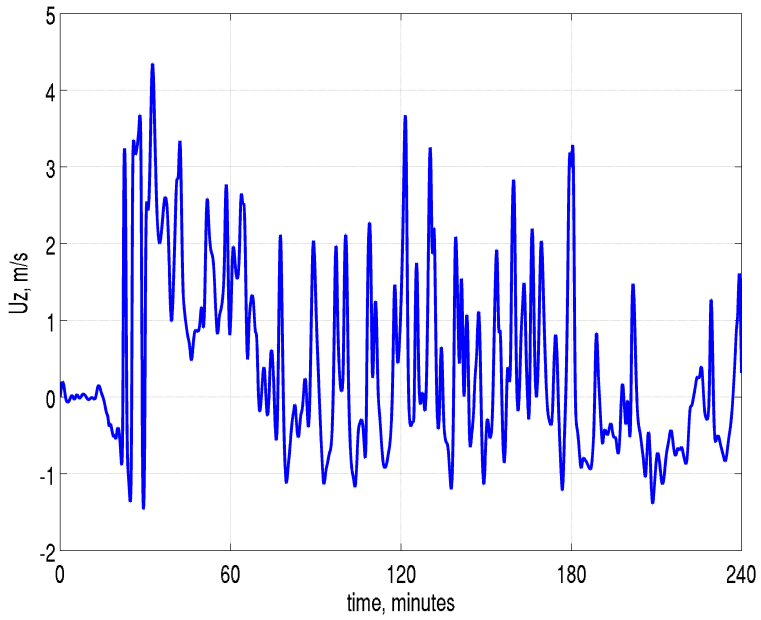


Figure 2.18: Instantaneous concentration of a passive tracer released from the surface for a y-z plane in the middle of the domain.

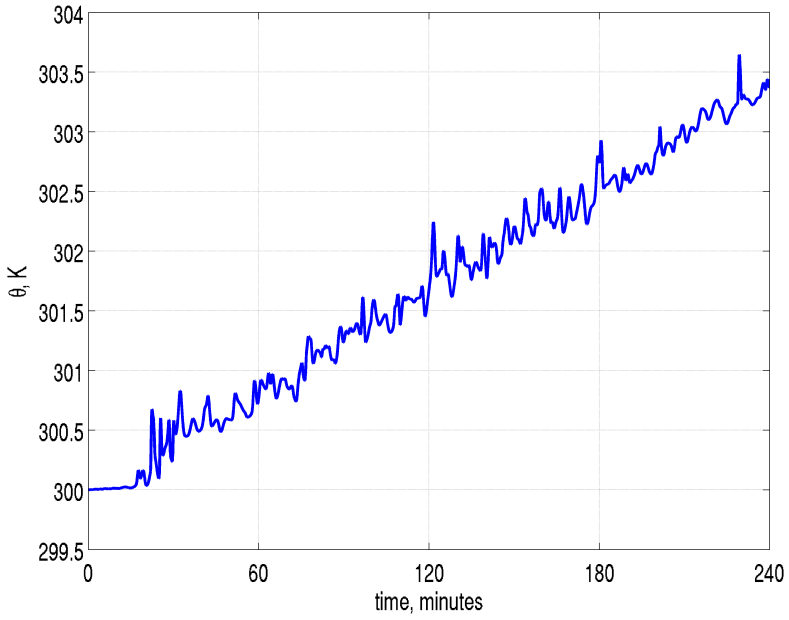
same scale as the convective velocity scale,

$$w_* = \left(\frac{g}{\theta_0} q_s z_\mathrm{i}\right)^{1/3} \sim 2 \text{ m/s}, \tag{2.18}$$

in accordance to Deardorff (1970). The heating of the air below the inversion is reflected in the gradual increase of potential temperature shown in figure 2.19b. In the evolution of vertical velocity and temperature an initial delay is observed, attributable to the time taken by the perturbation induced by the surface heat flux in reaching the level where the data was taken.



(a)



(b)

Figure 2.19: Time evolution of selected variables at 500 meters above ground at the center of the domain, data sampled at each time step (0.25 s). a) vertical velocity, m/s. b) potential temperature, K.

Figure 2.20 shows the spectra of energy contained within the horizontal velocity fluctuations, u_h, the vertical velocity, w, and the potential temperature, θ, at height $z/z_\mathrm{i} = 0.5$ (Churchfield et al. (2010); Khanna & Brasseur (1998)). To create the energy spectra we used a 2D fast Fourier transform to calculate the 2D energy spectrum and then we computed the spectral energy contained within shells of constant wavenumber magnitude ($\mathrm{k}_h = \sqrt{\mathrm{k}_1^2 + \mathrm{k}_2^2}$)

(Wyngaard (2010)). In red we plot the slope of the Kolmogorov spectrum ($k_h^{-5/3}$), which is partially followed by our results in the inertial range. However, they do not follow the -5/3 slope as well as the spectra of Khanna & Brasseur (1998), probably because we are using a finite volume formulation, more numerically diffusive than a pseudo-spectral solver (CH10).

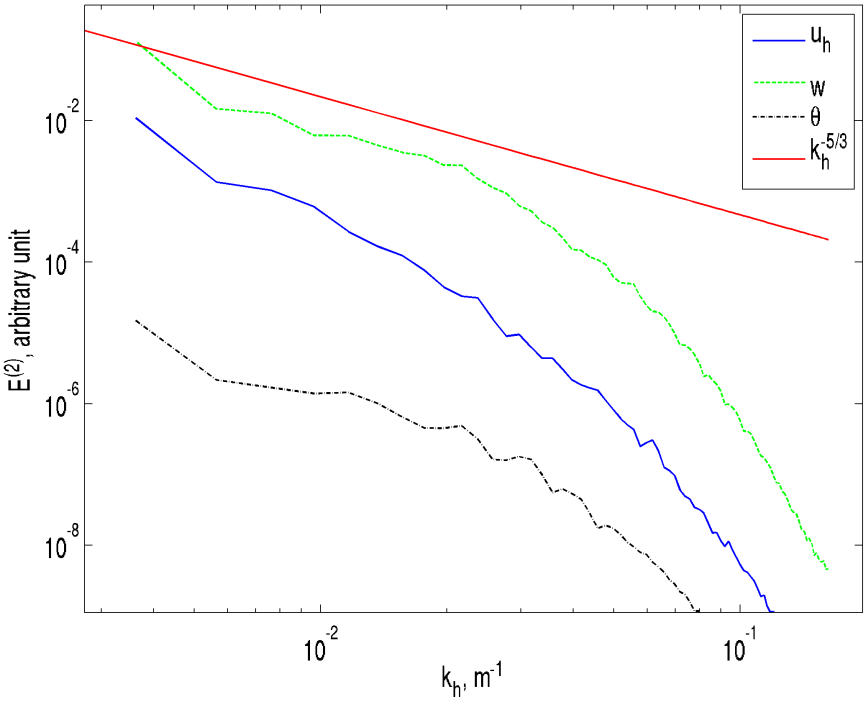


Figure 2.20: Two-dimensional variance spectra of $u_h = \sqrt{u^2 + v^2}$, w, and θ over an horizontal plane at $z/z_i = 0.5$.

Important turbulent statistics of our DES simulation results (velocity variances, momentum fluxes and velocity variance fluxes), comparable with those of MS94 and CH10, are shown in figure 2.21. Our statistics show good agreement with these previous studies, especially those of vertical velocity, controlled by the buoyancy produced by the heat flux. In particular, as described in CH10, vertical velocity variance, $\langle w'w' \rangle$, peaks around $z/z_i = 0.4$, vertical velocity variance flux, $\langle w'w'w' \rangle$, is greater than the other variance fluxes, and horizontal velocity variances, $\langle u'u' \rangle$ and $\langle v'v' \rangle$, peak near the base of the capping inversion and near the surface. The differences observed in horizontal velocity statistics close to the ground are attributable in part to our use of specific wall functions different than those used in the other works, which were needed in order to incorporate complex geometries in our simulations. Furthermore, MS94 data include the sub-grid-scale contribution, while our results consider only resolved variances. Also, our finite-volume solver is inherently more dissipative than the pseudo-spectral solver used in MS94. Spectral methods are not, however, easy to apply for complex domains, as the one we are interested in simulating, so at the beginning of this work we opted for a finite-volume approach, as that used by *OpenFOAM*. Perhaps the selection of higher order numerical schemes could avoid this problem, but could introduce stability problems when used in complex geometries. There are also two more sources for the differences between our results and the references. (1) The DES solver with RANS in the near wall regions damps the turbulence fluctuations. A few published papers (e.g. Piomelli

et al. (2003)) report some ways to enhance such fluctuations at the interface of the RANS and LES regions. (2) In MS94 a modified wall model is used to take into account the rough wall effects.

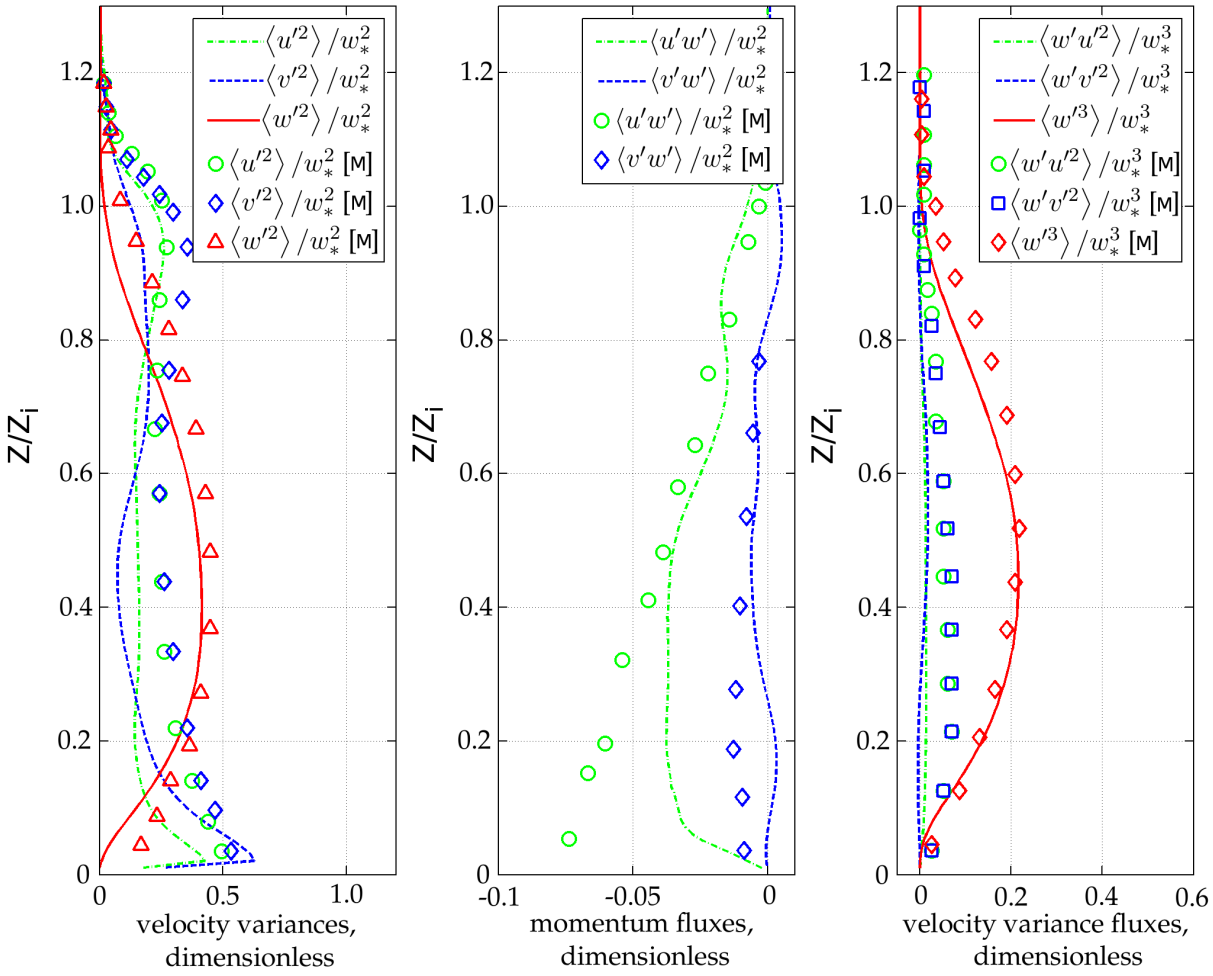


Figure 2.21: Nondimensional vertical profiles of horizontally-averaged resolved velocity variance, momentum fluxes and velocity variance fluxes, compared to results of Moeng & Sullivan (1994) ([M] in the figures). We have normalized the results by the convective velocity scale w_*.

The important process of the entrainment at the top of the CBL can be identified by means of the buoyancy flux vertical profile (figure 2.22). The resolved buoyancy flux correctly approximates the reference simulations. In particular, the entraiment is shown by the negative value of the buoyancy flux at the CBL top, with differences from the references originating probably in the subgrid scale values. In particular, these differences may be linked to the fact that we used a fixed value of the turbulent Prandtl number, Pr_t, not modified as a function of the level of stratification. We expect to investigate this idea in future simulations. Since our grid has higher resolution than that used in the references the increase of the subgrid scale contribution occurs closer to the ground.

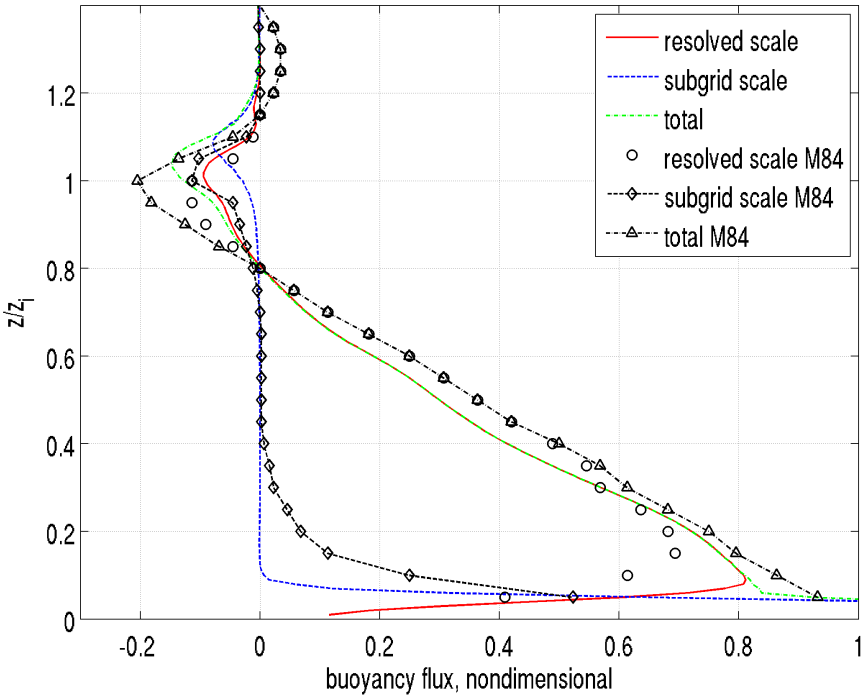


Figure 2.22: Nondimensional resolved, subgrid and total buoyancy flux, compared to results from Moeng (1984) (M84).

2.4 Conclusions

We have described the development of a numerical solver implemented in *OpenFOAM* for investigating complex atmospheric flows that include developed turbulence, stratification, thermal buoyancy and complex geometry. The model includes specific boundary and initial conditions needed to simulate the problem and uses a DES approach, combining LES to correctly simulate developed turbulence with RANS to solve the flow close to walls. Density was included as a variable in the system of equations to improve the treatment of stratification and buoyancy. To incorporate complex geometries we used the *snappyHexMesh* tool, obtaining fully operative meshes. We used different case studies to test our framework at different levels. Simple non-buoyant experimental cases were used to check the simulation of recirculation and the perturbation in the wake of obstacles. More complex large scale non-buoyant cases were used to study the ability of the *snappyHexMesh* tool to generate detailed complex meshes fully integrated to the numerical formulation used by *OpenFOAM*. Finally, buoyancy was introduced using first a simple experimental case, and then studying in detail a well documented atmospheric case. Both buoyancy and stratification demonstrated being well simulated, in agreement with previous observational and modelling results. While DES and URANS techniques showed similar results in simple non-buoyant cases, in strongly buoyant cases the ability of DES to solve developed turbulence was not replicated by URANS techniques. Although there are still many points to improve, such as the efficiency of the subgrid model or the treatment close to the walls, it was possible to show that the model and the particular boundary and initial conditions implemented in *OpenFOAM* for this work can deal with the complex multiphysical problem that implies modelling atmospheric flows close

to a complex ground.

Despite its limitations and it still being a technique under development, DES is a good alternative for making use of the advantages offered by LES when modelling turbulent atmospheric flows, without having to pay the high computing cost that would be implied by using this technique in complex geometries or topographies. Much research is still required to define the most efficient ways of integrating the RANS models that are so well-known in wind engineering to the LES simulations, in particular when considering convective atmospheric cases, but tools such as those allowed by the *OpenFOAM* platform will be of great help in this development. The modular characteristic of *OpenFOAM* allows equations and submodels to be added to a principal model as the simulation becomes more complex. This makes it advisable to address the modelling of complex atmospheric flows through a methodology like that used in this work, incorporating physical processes to the simulation progressively.

Given the good results obtained when simulating the convective case, in the second part of this work we will apply this solver to the study of the internal circulation developing in large open pit mines. We will simulate the particular case of Chuquicamata, one of the largest open pit mines in the world, located in northern Chile. In this mine the dispersion and exit of contaminants are controlled by the convective conditions inside the cavity (Barkan & Alpert (2010)).

Chapter 3

Model application to large open pit mines

This chapter corresponds to an extended version of the paper *"OpenFOAM applied to the CFD simulation of turbulent buoyant atmospheric flows and pollutant dispersion inside large open pit mines under intense insolation"*, authored by Federico Flores, René Garreaud and Ricardo Muñoz, submitted in the form of research article to *Computers and Fluids* (Flores et al. (2013b)).

ABSTRACT

The particular conditions of air circulation inside large open pit mines under intense insolation, dominated by mechanical and buoyant effects, are crucial when studying the dispersion of pollutants inside and outside the pit. Considering this, we study this problem using CFD tools able to include the complex geometry characterizing it and the different processes affecting circulation: flow interaction with obstacles, buoyancy, stratification and turbulence. We performed simulations using a previously developed *OpenFOAM* solver, focusing in the particular case of Chuquicamata, a large open pit mine ($\sim$ 1 km deep) located in northern Chile. Both idealized and real topographies were used. Given the importance of turbulence in this type of large-scale flows we have used LES to incorporate it in the calculation, using a DES approach to solve the flow near walls.

The results from the idealized cases support the idea that buoyant currents foster the exit of particles from the pit and increase the turbulence inside its atmosphere, modifying the purely mechanical recirculatory flow inside the cavity. Differences in the air circulation and dispersion of particles between idealized and non-idealized cases are reported. In particular, there are changes in the intensity and location of the recirculation inside the pit due to variations in the aspect ratio (length/depth) of the cavity along the axis perpendicular to the main flow. Also, the topography surrounding the mine affects the main flow that sweeps the cavity, channelling it along the main axis of the pit and forcing it to enter the cavity through the lower level of the top edge. As a consequence, the patterns of pollutant transport observed in the idealized cases, dominated by near-wall upward currents, are different than those observed in the cases with complex topography, where the dispersion is dominated

by internal buoyant upward currents. Anyhow, whether by internal or near wall upward currents, in all buoyant cases considered a large percentage (> 60 %) of the particles injected inside the pit leaves the cavity.

Further numerical experiments studying the effect of 3D aspect ratio over the mechanically forced internal flow are needed to fully understand the effect of the internal geometry of the pit over the flow.

3.1 Introduction

Extraction and transport of minerals from open pit mines can produce significant emissions of fugitive dust, severely affecting the operations both inside and outside the pit. Throughout the years different techniques have been developed in order to minimize health and environmental issues, with the objective of maximizing the operations. This has increased the interest in understanding the different patterns of pollutant transport inside the atmosphere of open pit mines.

Due to the special meteorological conditions of closed valleys (natural analog of open pit mines), they have been the focus of vast research since the beginning of twenty century (Schmidt (1930); Sauberer & Dirmhirn (1956); Geiger (1965)). More recently, in the 80's, several observational field campaigns were carried out in Japan (Magono et al. (1982); Yoshino (1984); Kondo et al. (1989)), and USA (Whiteman (1982, 1986, 1990)). Some of the most important observational campaigns since 1990 are summarized in table 3.1. The most recent of these was the METCRAX experiment (Whiteman et al. (2008)), that took place in October 2006 in a near-ideal basin-shaped meteor crater near Winslow, Arizona. The analysis of field campaign data has shown a well defined diurnal cycle inside the atmosphere of closed valleys, controlled by surface heat flux induced by solar radiation during daytime, and the formation of a cold pool at the bottom of the valley during the night. Because the exit flows that dominate the circulation in open valleys are not allowed in a closed configuration, air circulation in this latter case shows unique patterns. In particular, air flow recirculation induced mechanically inside the pit reduces the exit flows, while slope currents controlled by heat flux from the surface are critical in the definition of currents leaving or entering the pit (subsidence in nocturnal cases is a well documented phenomenon (Whiteman et al. (2008))).

experiment	**location**	**size**	**depth**	**reference**
Sinbad valley	Colorado, USA	∼2 km	∼400 m	Whiteman et al. (1996)
Peter Sinks	Utah, USA	∼1 km	∼80 m	Clements et al. (2003)
Gruenloch doline	Austrian Alps	∼1 km	∼150 m	Steinacker et al. (2007)
METCRAX	Arizona, USA	∼1.2 km	∼165 m	Whiteman et al. (2008)

Table 3.1: Important field campaigns in closed valleys since 1990.

We summarize in table 3.2 the main numerical studies of air circulation inside closed valleys. In the 90's Baklanov was one of the first to propose the need to approach the problem from a multiphysics perspective, highlighting the combination of scales involved, and the importance of topography (Baklanov (1995, 2000)). His work focused on open pit

mines, studying the use of ventilators to facilitate the dispersion of contaminants (Baklanov & Rigina (1995)) and the effect of surface explosions on the atmosphere of the pit (Rigina & Baklanov (1995)), and showed the importance of inclined currents over the ventilation inside the pit as well as the critical role played by topography in the dispersion of contaminants. Since then, several numerical modelling approaches to the problem have been performed. Most of them studied the evolution of the temperature inversion layer inside valleys and the cold pool formation inside the pit (Kiefer & Zhong (2010); Fritts et al. (2010); Fast et al. (1996); Colette et al. (2003)), highlighting the importance of the large scale wind direction over the circulation inside the valley and the key role played by the geometry of the pit. Shi et al. (2000) and Silvester et al. (2009) focused on open pit mines. The former used a high resolution 3D nonhydrostatic model to simulate the air circulation inside a 2 km wide and 100 m deep pit, and were able to reproduce the intense recirculation inside the cavity, consistent with wind tunnel experiments, that was responsible for maintaining high levels of pollution inside the pit. Their results showed that both mechanical and thermal forcings are important mechanisms controlling the evolution of the atmosphere inside the pit, with fairly strong turbulence produced by the interaction of both processes. The latter used the CFD code Fluent to study the mechanically forced circulations (they did not account for buoyancy) developed inside the Old Moor open pit (1 km wide and 650 m deep), showing the importance of mechanical processes in the configuration of the air flow inside the pit, strong mechanical shear being produced by the interaction of the wind that sweeps over the cavity and the internal atmosphere. These numerical experiments highlighted the importance that the geometry of the pit has over the flow (slope angle, aspect ratio, etc), the existence of intense air flow recirculation inside the pit, and the role played by surface heat flux (that determines the intensity of buoyant currents), all factors controlling the exit of pollutants from open pits. The flow recirculation detected inside the pit by these authors is a well known phenomenon affecting flows over cavities (Bres & Colonius (2008); Kang & Sung (2009); Mesalhy et al. (2009)), being the aspect ratio of the cavity (length/depth) the key parameter defining the number, size and location of the main rolls inside the cavity.

reference	**model**	**pit (xyz) km**	**compressibility**	**turbulence**
Baklanov (1995)	own	3 x 1 x 0.5	incomp.	Smagorinsky, $k-\varepsilon$
Fast et al. (1996)	RAMS	2 x 2 x 0.4	incomp.	Mellor and Yamada
Shi et al. (2000)	own	2 x 2 x 0.1	incomp.	$k-\varepsilon$
Colette et al. (2003)	ARPS	2 x 2 x 0.5	comp.	LES
Silvester et al. (2009)	Fluent	1 x 0.65 x 0.12	incomp.	R $k-\varepsilon$
Kiefer & Zhong (2010)	ARPS	1.2 x 1.2 x 0.165	comp.	LES
Fritts et al. (2010)	NEK5000	1.2 x 1.2 x 0.165	incomp.	Baldwin-Lomax

Table 3.2: Summary of some important numerical simulations of air circulation in closed valleys since 1990.

The environmental problem described before is increased in the case of very large open pit mines subject to strong insolation, which are affected by complex patterns of pollutant dispersion inside their atmosphere and around the pit. Despite its large scale, the complex features that characterize this circulation, with developed turbulence, buoyant currents and mechanical effects due to topography, are difficult to adress with standard meteorological models. There exists consensus that the problem demands appropriate inclusion of multi-physics and multi-scale processes into a Computational Fluid Dynamics (CFD) approach

(Wyngaard (2004); Chen et al. (2011)). In particular, intense turbulence and recirculation induced mechanically inside the pit must be correctly simulated (Shi et al. (2000)). Also, buoyancy must be considered.

In this work we focus our attention on understanding the interaction between the two main processes that control the air flow inside the internal atmosphere of large open pit mines under intense insolation: mechanical effects due to the surface geometry and convective effects due to buoyancy. Our primary target is to define the main modes of contaminant transport inside and outside Chuquicamata, a very large open pit copper mine located in northern Chile (22^o17'20"S 68^o54'W). Due to its large scale (width $\sim$4 km, depth $\sim$1 km, see section 3.4) and intense insolation (the mine is located in the Atacama Desert), previous studies can't be directly applied here.

First, we performed simulations aimed at studying the circulation using a simplified geometry, that conserves the main size of the real pit but with circular symmetry. In second place we used the complex real topography of Chuquicamata and its surroundings. The simplified geometry allows a clearer study of the processes interacting in the formation of the air circulation inside the pit, while the more realistic topography includes the effect of the real geometry. Given the great versatility that it provides, we selected the CFD tool *OpenFOAM* as development platform, using a solver previously developed (Flores et al. (2013a)). We used Detached Eddy Simulation (DES), to properly simulate turbulence around complex geometries, and included density as an explicit variable in the system of equations in order to improve the treatment of buoyancy. DES combines Large Eddy Simulation (LES) far from walls, to properly solve large atmospheric eddies, and Reynolds Averaged Navier Stokes (RANS) techniques near walls, to solve complex geometries (Spalart et al. (1997); Spalart (2009)). Even if the circulation inside closed valleys has been simulated in previous works (Silvester et al. (2009); Kiefer & Zhong (2010); Fritts et al. (2010)), there is apparently no record that *OpenFOAM* has been used to study the circulation and pollutant dispersion within very large scale open pit mines subject to intense insolation. In particular, there is no evidence that the new compressible solvers that consider density as a calculation variable (instead of using the Boussinesq approximation) have been used for these applications.

Section 3.2 of this paper briefly describes the physical problem considered and the numerical treatment. The numerical approach used to configure the simulations was described in detail and validated in a previous work (Flores et al. (2013a)). In section 3.3 we present different numerical simulations using an idealized geometry similar to Chuquicamata, considering different boundary conditions in order to investigate the effect of mechanical and buoyant processes over pollutant dispersion separately. In section 3.4 we describe the results of the CFD simulation of air flow inside and around Chuquicamata using its real topography, in order to identify the role played by it over the dispersion of contaminants. Section 3.5 includes the main conclusions taken from the work, and proposes future applications.

3.2 Physical problem and numerical treatment

3.2.1 Air circulation inside a closed valley

The air flow affecting the dispersion of pollutants inside and around closed valleys, like the ones that interest us, is composed by two main processes that control its evolution:

- Slope flows controlled by buoyancy, which will generally be the flow dominating the circulation inside the pit. There exist several references relative to this type of flows (Sharples et al. (2010); Skyllingstad (2002); Axelsen (2010)).
- Mechanical effects produced by the interaction of the external flow with the geometry of the cavity. The approximately conical features of open pit mines modify the circulation imposed by buoyant slope flows, introducing mechanical effects produced by the slope angle and circular symmetry of the pit. In this regard, the analysis is similar to that developed for valleys (Whiteman (1990); Colette et al. (2003)), although the open condition of the latter allows for exit flows not possible in closed pits. Given that the existence of large scale closed valleys is rare in nature the research about air circulation inside their atmosphere has remained restricted to meteoritical craters, or to the specific case of large open pit mines (Silvester et al. (2009); Whiteman et al. (2008)).

3.2.2 Numerical Approach

3.2.2.1 Model

We have already described in Flores et al. (2013a) the numerical approach used in our simulations, as well as its validation against documented data. The solver includes the following equations: continuity, momentum conservation (Navier-Stokes), enthalpy conservation, ideal gas state and passive scalar transport, using the libraries and tools provided by *OpenFOAM* to solve them (thermophysical, turbulence and finite volume libraries). We have chosen a numerical approach that includes density as an explicit variable in the system of equations, instead of using the Boussinesq approximation. We use a DES (detached eddy simulation) approach to solve the flow near walls, in order to be able to include complex topography. LES (large eddy simulation) is used to solve the turbulent eddies far from walls. Results obtained in Flores et al. (2013a) showed that the model and the particular boundary and initial conditions implemented in *OpenFOAM* for this work can deal with the complex multiphysical problem that implies modelling atmospheric buoyant flows close to a complex ground. DES allows us to use the advantages offered by LES when modelling turbulent atmospheric flows, without having to pay the high computing cost that would be implied by using this technique in complex geometries.

3.2.2.2 Geometry and meshing

As in the previous work (Flores et al. (2013a)) we used the *snappyHexMesh* toolbox to generate the mesh, in conjunction with the CAE software *Salome* to create the necessary STL files. *GlobalMapper*, a mapping software package, was used to create STL files from real topography (SRTM data). *Matlab* was used to modify the files generated by *GlobalMapper* to allow their use by *snappyHexMesh*.

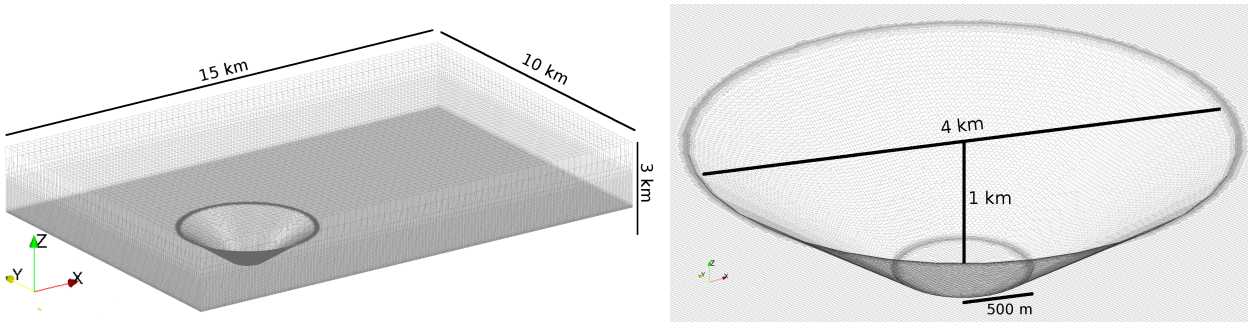
3.2.2.3 Computing support

The configuration of the simulations and the analysis of their results was done on a personal workstation using *Paraview*, while their execution was done in parallel in the Levque cluster of the National Laboratory for High Performance Computing (NLHPC) of the Center for Mathematical Modeling of the University of Chile. The Levque cluster is an IBM iDataplex machine with 536 cores, equipped with Intel Nehalem processors, an Infiniband QDR switch and several development tools (Maureira et al. (2011)).

3.3 Idealized cases

3.3.1 Domain

In this first part we used a simplified topography, that retains the same general dimensions of Chuquicamata (see section 3.4), representing the pit as an inverted truncated cone with a superior diameter of 4 km, an inferior diameter of 1 km and 1 km deep (figure 3.1b). We selected this simplified symmetric geometry since it allows a clearer study of the processes interacting in the formation of the air circulation inside the pit. The simulation domain consisted of a rectangular region 15 km long, 10 km wide and 3 km high (figure 3.1a). Although the domain is small, considering the size of the cone, it was chosen after several mesh tests produced solutions independent of domain size and mesh (figure 3.2). Different domain sizes, changing height and the distance of the pit from boundaries, were used in order to test domain size independence. Due to the high computational cost of this kind of simulations a balance must be reached between domain size and mesh resolution. The *snappyHexMesh* tool was used to generate the mesh based on a STL file created using *Salome*, providing a mesh with nearly 5 million cells. The mesh was refined near walls, by the use of *snappyHexMesh* controls, to produce cells with wall normal dimensions of between $y_1^+ = 100$ and $y_1^+ = 1000$ adjacent to the surface (y_1^+ is the distance in wall units between the centroid of the first cell and the wall assuming the y coordinate is normal to the wall). However, it is impossible to satisfy this criterion everywhere when processes of flow separation and attachment occur.



(a) (b)

Figure 3.1: Mesh used. a) total domain b) detail, inverted cone.

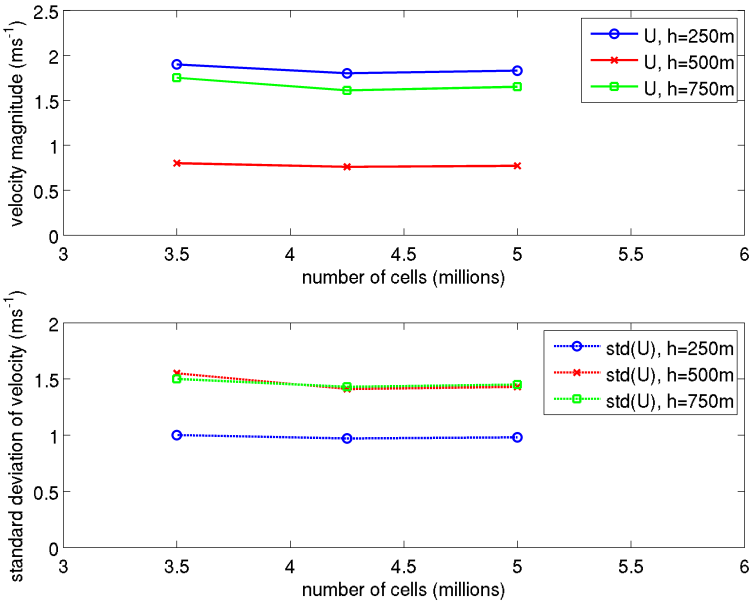


Figure 3.2: Example showing the variation of mean velocity and standard deviation of velocity at three single points, used to study mesh independence. The points are located at the center inside the pit, at 250 m (blue), 500 m (red) and 750 m (green) above the bottom.

3.3.2 Experimental setup

We employed cyclic lateral boundary conditions in both horizontal directions. The upper boundary conditions (2 km above the top edge of the pit) are free-slip and mantain a constant potential temperature gradient. The initial potential temperature profile has a constant value of 300 K up to 937 m above the top edge of the pit (all the internal volume of the cone has constant potential temperature), followed by an 8 K increase in the next 167 m, and a constant 0.003 K/m gradient up to the top of the domain (as used by Moeng & Sullivan (1994) and Churchfield et al. (2010)).

Because of our interest in studying the interaction between buoyancy and mechanical effects, we defined three numerical experiments in which these processes have different relative

importance. A summary of the cases and the designation codes is shown in table 3.3.

Table 3.3: Idealized cases. Wind speed is specified 300 m above the top edge of the pit. The surface heat flux was imposed using the boundary condition defined in section 2.2.2.4.

predominance	name	wind above 300m	surface heat flux
wind	W	10 m/s	0
surface heat flux	B	1 m/s	$\sim$ 240 W/m^2
wind and surface heat flux	WB	10 m/s	$\sim$ 240 W/m^2

The first case, W, is dominated by strong winds aloft (10 m/s at 300 m over the top edge of the pit, vertical profile following a power law as described in Flores et al. (2013a)), and does not consider any thermal effect (no enthalpy equation included, neutral atmospheric boundary layer). Case B is dominated by buoyancy, with a surface heat flux of 240 W/m^2, and only a 1 m/s wind speed 300 m above the top edge of the pit. The third case, WB, combines both previous cases, keeping a 10 m/s wind aloft and a 240 W/m^2 surface heat flux.

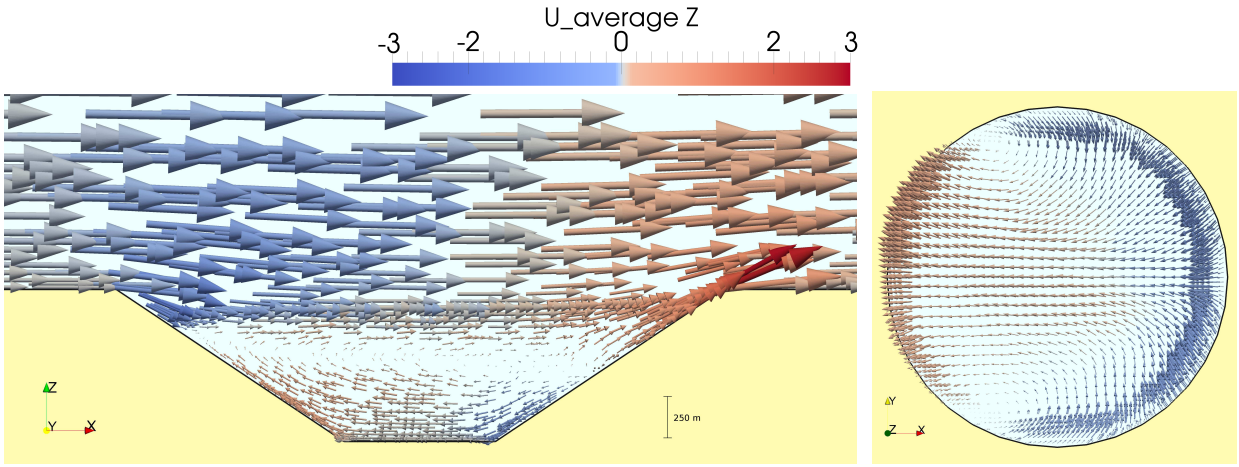
In order to facilitate the initial development of turbulence, the initial velocity field was perturbed as suggested by de Villiers (2006), introducing periodic perturbations along the x and y-axes through sine and cosine functions. The time step used in each simulation was 0.025 seconds, necessary to keep a low Courant number near the surface where the mesh is refined, for a full simulation time of 2 hours (until reaching a statistically stationary flow). Different final times have been tested to ensure that convergent time-averaged results were attained. For further details of the numerical approach used see Flores et al. (2013a).

3.3.3 Results

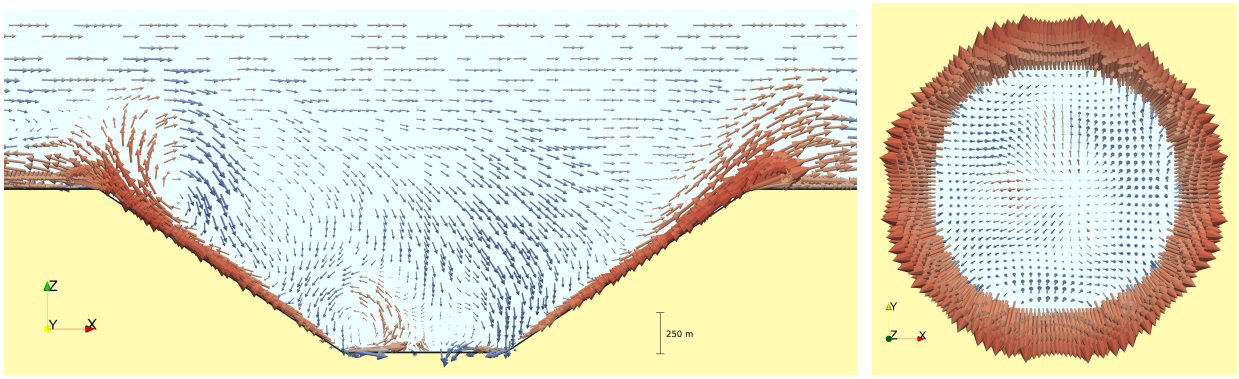
In order to analyze the circulation imposed by each case and its effect on the dispersion of contaminants inside and around the pit, we use two simple techniques to visualize this circulation. First, we use time-averaging over the last hour of simulation to describe the mean patterns of circulation. Then, injecting passive tracers to the velocity field, we study the features that characterize the currents leaving the pit, in such a way that we can explore the dispersion of contaminants emitted from the surface.

3.3.3.1 Mean values

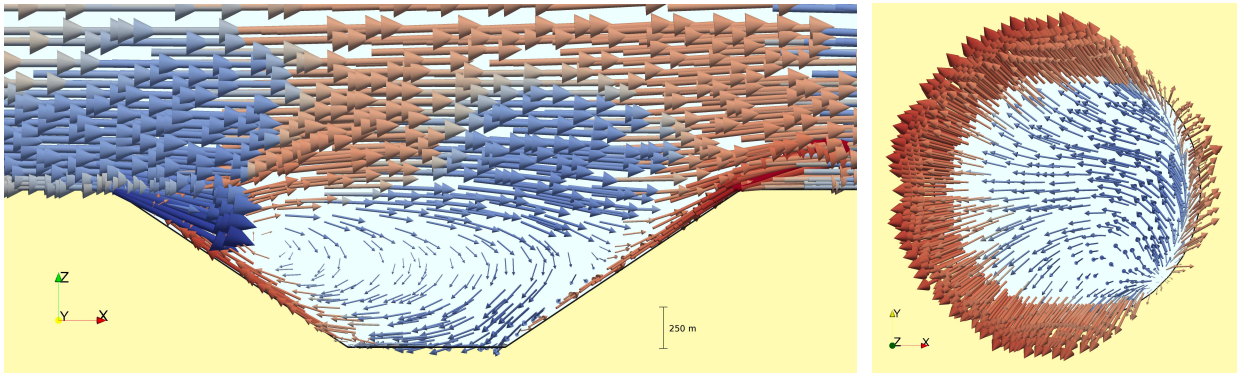
The last hour mean vector velocity field is presented for each case in figure 3.3. Panels 3.3a, 3.3c and 3.3e show a vertical plane that runs through the center of the pit, while panels 3.3b, 3.3d and 3.3f show a horizontal plane 250 meters above the bottom of the pit (height scale indicated in the vertical plane). The existence of distinctive circulation patterns in each case can be observed. In both cases with positive surface heat flux (B, panels 3.3c and 3.3d and WB, panels 3.3e and 3.3f) the air circulation inside the pit is characterized by strong convective currents that ascend close to the walls. In case W (panels 3.3a and 3.3b), in which the wind dominates, there are no clearly defined ascending currents close to the walls. Two



(a) case W **(b)** case W



(c) case B **(d)** case B



(e) case WB **(f)** case WB

Figure 3.3: Mean vector velocity field during the last hour of simulation, for each case. a), c) and e), vertical plane in the middle of the basin (y=0). b), d) and f), horizontal plane 250 meters above the bottom of the pit. Color indicates mean vertical velocity (color bar, m/s). Vector scale is the same for all cases.

flow features can be identified in this case: a circulation that follows the geometric pattern of the pit, with convergent currents near the downwind slope, induced by circular symmetry (the flow near the top edge of the pit follows the circular walls of the cavity, feature not seen in the vertical plane but visible in a horizontal plane near the top edge (not shown) and in figure 3.3b), and a relatively strong counter-direction flow inside the pit (as reported in previous works (Baklanov (1995); Shi et al. (2000))), which induces descending flow in the downstream wall and ascending flow in the upstream wall. Flow separation occurs at

the top of the upstream wall, induced by local changes in pressure (Batchelor (1967)). The vorticity present near ground (no-slip boundary) evolves into a large eddy when the boundary condition changes (Schlichting (1979)). This eddy is not clear in figure 3.3a, mainly due to vector scale and symmetry. Descending currents closing the eddy are very weak (~ 0.5 m/s) compared to the main wind aloft (~ 10 m/s). Also, the main eddy is more evident outside the symmetry plane shown in figure 3.3a, since it developes mainly near the walls outside that plane. The circular geometry of the pit reduces the aspect ratio of the cavity along the y-axis (perpendicular to the main flow) outside the vertical symmetry plane, favoring the formation of a more intense eddy near the lateral walls (Kang & Sung (2009)) (a well defined eddy is visible in a vertical plane 300 meters outside the symmetry plane, figure 3.4). Figures 3.3c and 3.3d (case B) show the recirculation produced by the interaction between upslope flows and the weak incident ambient wind, and the presence of vertical convective currents inside the pit near ground (below 300 meters). In this case, wall updrafts seem to be the more relevant flow, with compensatory subsidence inside. The effect of upslope currents is less important in case WB (figures 3.3e and 3.3f), affected also by strong wind, in which the mean circulation seems to be a combination of both previous cases. In this case the main internal eddy shows a clear structure, and seems reinforced by buoyant currents.

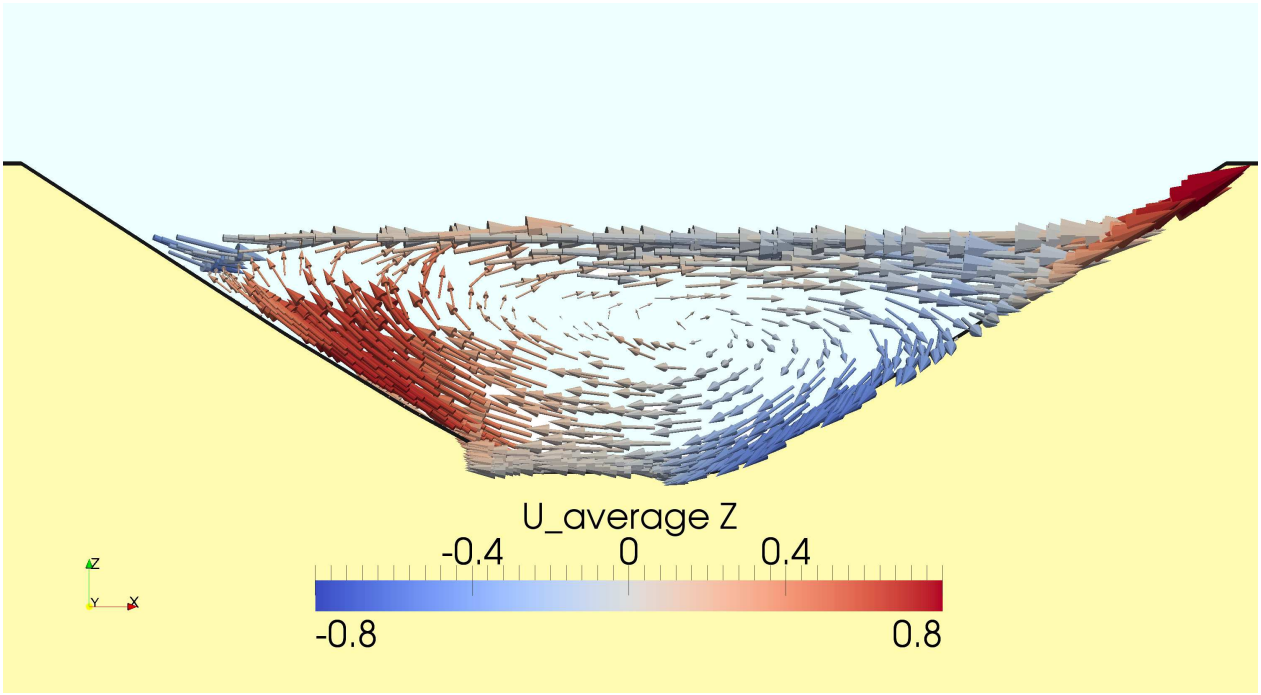
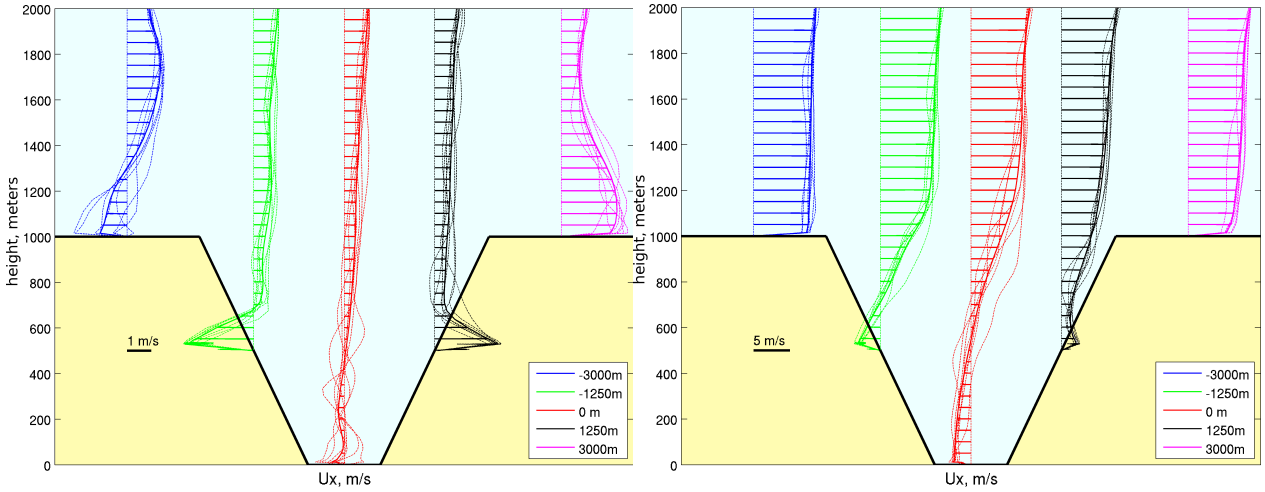


Figure 3.4: mean vector velocity field during the last hour of simulation for case W. Vertical plane 300 meters outside the symmetry plane. Vector scale is not the same of figure 3.3. We have increased it and eliminated the vectors above the pit in order to improve the visualization of the main roll. Color indicates mean vertical velocity (color bar, m/s).

While the above results revealed the presence of a mean circulation clearly defined in each scenario, it is interesting to analyze the importance of temporal perturbations over the mean flow, in particular in convective cases, which are our primary focus. Figure 3.5 shows instantaneous vertical profiles of velocity component U_x, at different locations inside the domain, for cases B and WB. Each dotted line represents the instantaneous profile every 10



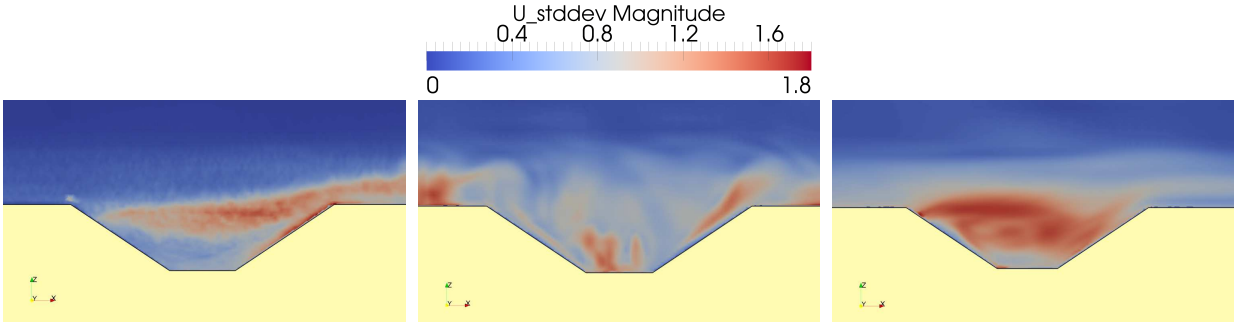
(a) case B **(b)** case WB

Figure 3.5: Vertical profiles of velocity component Ux, at different locations inside the domain. Instantaneous (dotted lines, each 10 minutes) and mean (solid lines) profiles, during the last hour of simulation. a) case B, b) case WB. A different scale was used in cases with stronger winds (W and WB).

minutes during the last hour, while the solid line shows the mean profile during the same time. All locations considered are in symmetry plane y=0 (see figure 3.1), being the first one located 3000 m upwind of the center of the pit (1000 m before its top edge), the second one 1250 m before the center of the pit (in the middle of the slope), and the third one right at its center. Locations 4 and 5 are symmetric to locations 2 and 1, respectively. A different scale was used in each case, due to stronger winds in case WB. This latter case shows an important flow recirculation inside the pit (negative velocity Ux in figure 3.5b). Also, this case shows wide variability in the U_x profile, in particular at levels where the forcing aloft couples with the circulation inside the basin. The effect of convective currents is evident in case B, dominated by positive surface heat flux (figure 3.5a), with strong perturbations in the profile of U_x at the center of the pit (location 3), and with intense currents leaving the pit close to the walls (locations 2 and 4). This effect is also shown by perturbations in the profiles far from the pit, amplifying U_x in location 5 and reducing it in location 1.

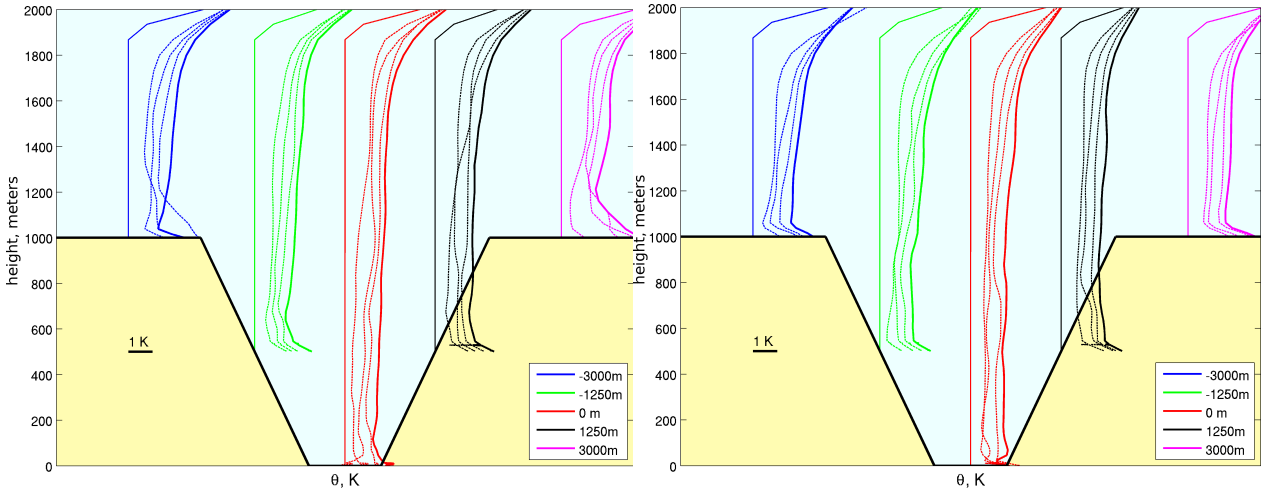
Figure 3.6 shows the standard deviation of velocity magnitude in all cases. In case W (figure 3.6a) intense turbulence occurs near the top of the pit due to the interaction between the flow aloft and the internal atmosphere. In case B (figure 3.6b) the turbulence is linked to thermal updrafts, while in case WB (figure 3.6c) turbulence increases its magnitude and extends inside all the pit. As shown in figure 3.3, buoyancy increases the recirculation induced by the incident wind, increasing the circulation and turbulence inside the pit.

Figure 3.7 shows instantaneous profiles of potential temperature, following the same configuration as we have already presented for U_x in figure 3.5. These profiles show the heating of each escenario. In case B (figure 3.7a) the presence of higher air temperatures near ground at locations 1 and 5, over the top edge of the pit, could be associated to hot air currents leaving the basin close to the walls (especially at location 5, favored by wind direction).



(a) case W **(b)** case B **(c)** case WB

Figure 3.6: Standard deviation of velocity magnitude (m/s). a) case W, b) case B and c) case WB.



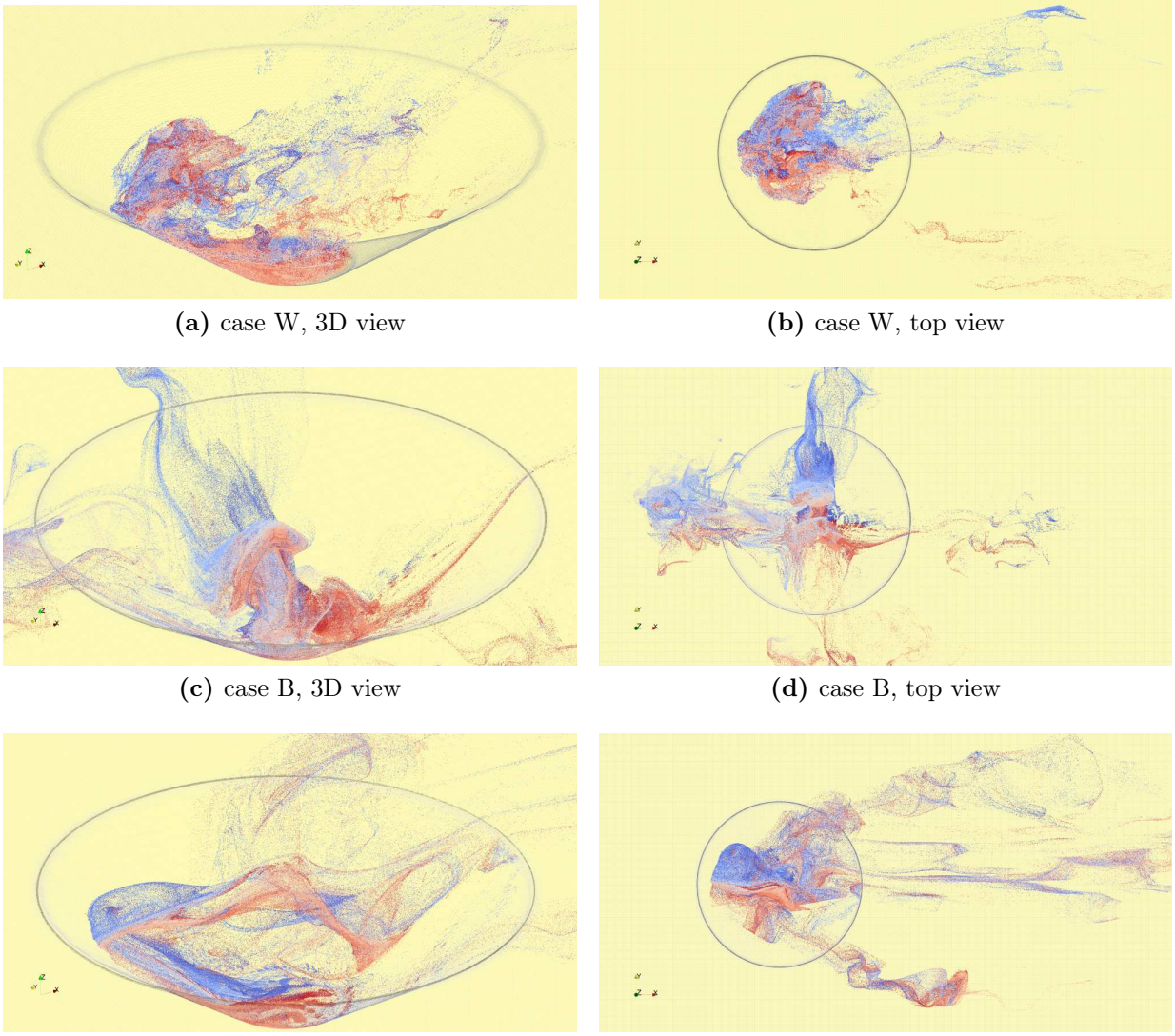
(a) case B **(b)** case WB

Figure 3.7: Instantaneous vertical profiles of potential temperature θ, at different locations inside the domain, each 15 minutes apart during the last hour of simulation. a) case B, b) case WB.

3.3.3.2 Tracking of particles

Given our interest in studying the dispersion of pollutants from the interior of the pit we used tracking of particles to analyze air circulation. Using the *Paraview* software we injected instant puffs of 18000 particles in an imaginary 500 m diameter disc 10 meters above the bottom of the pit (radial resolution: 50 particles, circumferential resolution: 360 particles), every 50 seconds, during the last hour of simulation. The particles are treated as ideal tracers, with no sedimentation velocity. Figure 3.8 shows the final locations of the injected particles (at the end of the second hour), the color indicating the original location of each particle (red -y, blue +y).

In cases B and WB a large number of particles leaves the pit, while in case W only a small fraction of the injected particles leaves the cavity, after a long time since their injection, due to the recirculation effect, as suggested by previous studies (Shi et al. (2000); Silvester et al. (2009)). The role of wind becomes clear comparing scenarios B and WB, especially the top



(a) case W, 3D view

(b) case W, top view

(c) case B, 3D view

(d) case B, top view

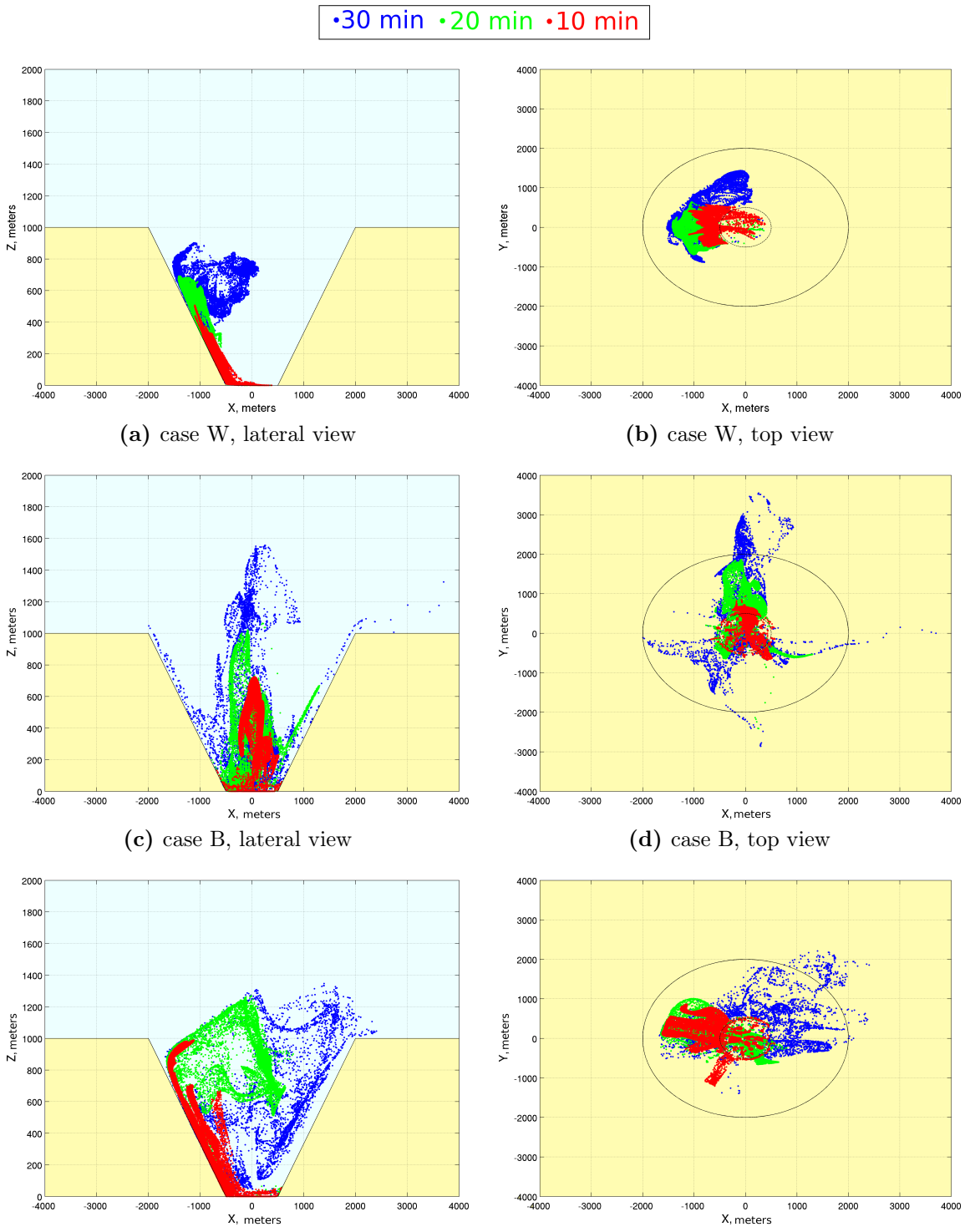
(e) case WB, 3D view

(f) case WB, top view

Figure 3.8: Location of bottom injected particles at final time. Color indicates the original location of each particle (red -y, blue +y).

views 3.8d and 3.8f. In the case with stronger winds (WB) the particles, once they reach the top edge of the pit, are dragged downstream. In case B on the contrary, particles leaving the basin close to the walls accumulate over the top edge, especially upstream, where convective currents interact with the weak incident wind (recirculation at the upstream top edge seen in figure 3.3).

Figure 3.9 shows the temporal evolution of the injected particles. Only the final location of three puff of particles is shown: blue dots mark the final locations of particles released 30 minutes before end time, green dots those released 20 minutes before end time and red dots those released 10 minutes before end time. In case W (figures 3.9a and 3.9b) the intense recirculation inside the pit drags the particles toward the upwind slope, producing their uplift, but not enough to reach the edge. Even the oldest particles stay inside the pit, but are affected by the turbulent flow present at the top of the cavity, where the incident wind



(a) case W, lateral view

(b) case W, top view

(c) case B, lateral view

(d) case B, top view

(e) case WB, lateral view

(f) case WB, top view

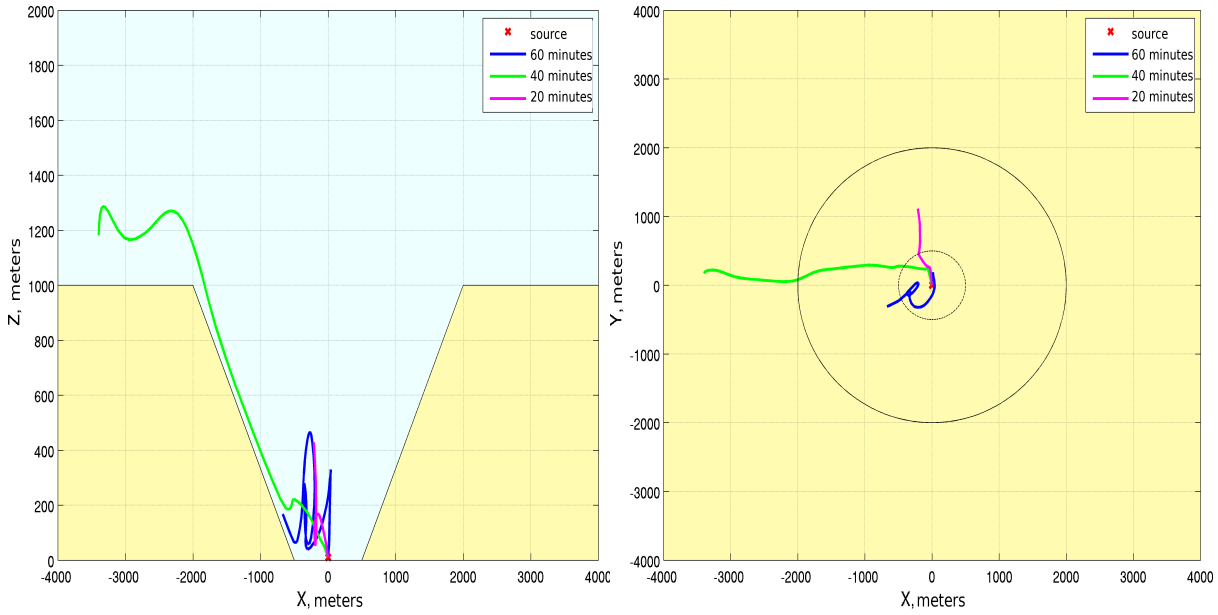
Figure 3.9: Final locations of particles released 30 (blue), 20 (green) and 10 (red) minutes before end time. a) and b) case W. c) and d) case B. e) and f) case WB. These are not section views, all particles in 3D space are included.

aloft interacts with the internal atmosphere (figure 3.6a), extracting some of the particles. In case B (figures 3.9c and 3.9d) the temporal evolution of particles is influenced by convective currents, present both in the center of the basin and near the walls, where they are stronger.

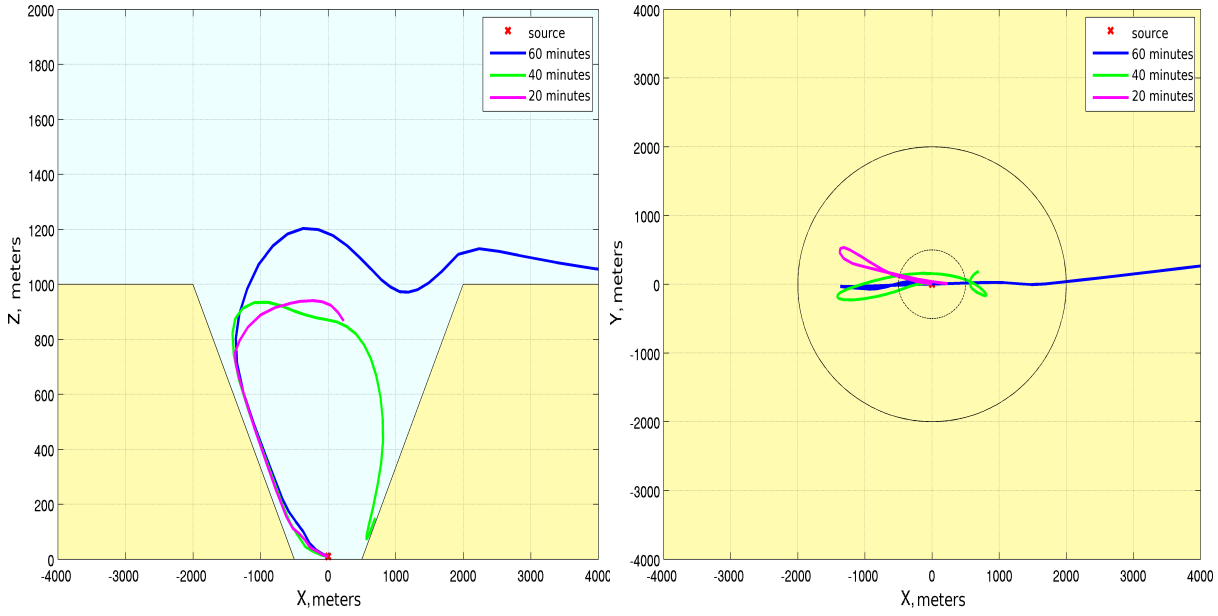
Older particles (blue), that have had the time necessary to reach the walls, have been expelled out of the pit. Particles with shorter lifetime (green and red) are dominated by internal thermal lifts and remain near the center of the basin. In this case upward currents following the lateral walls are preferred by the particles leaving the pit, probably due to descending currents induced by recirculation and compensatory subsidence (figure 3.3c). The strong wind present in case WB (figures 3.9e and 3.9f) changes the patterns described above. In this scenario the thermal lifts at the center of the basin are less important, probably due to the recirculation induced by the external wind, perturbing air circulation inside the pit and approaching released particles to the upwind slope (left wall in figure). This process increases the number of particles that reach the top edge of the pit and reduces their lifetime near ground (red particles). The recirculatory flow induced by the interaction between the wind aloft and the internal atmosphere is shown by green particles that are strongly mixed in that zone (figure 3.9e). In this case the convective currents speed up the evolution of particles, accelerating the processes observed in case W. Air recirculation inside the pit facilitates the approach of particles to the upwind slope, accelerating their exit of the pit. At the same time this recirculatory flow produces that some of the particles that have risen re-enter into the basin, while others go downstream following the winds aloft. Then, even if the wind accelerates the exit of particles, approaching them to convective currents near walls, it also contributes to their re-entry into the pit.

The features analyzed above for cases B and WB are also evident when studying the trajectory of a single particle. Figure 3.10 shows the trajectory followed by a particle released at a point 10 meters above ground in the center of the pit. Three lifetimes were considered: 60, 40 and 20 minutes (blue, green and magenta lines, respectively). The trajectories of case B (figures 3.10a and 3.10b) confirm that exit of particles depends on whether or not their initial circulation, controlled by thermal currents, led them to approach the walls of the pit, where strong upward currents exist. Of the three particle trajectories shown in the figure only one, which was driven close to the upwind slope, leaves the pit. Even an older one (blue line) remains inside the basin, because internal currents did not drive it close to upward currents near the slopes. Trajectories of case WB (figures 3.10c and 3.10d) show the effect of wind over the convective currents described above. Air recirculation inside the pit facilitates the approach of particles to the upwind slope. Also, this recirculatory flow produces that some of the particles that have risen re-enter into the basin (green line), while others go downstream following the winds aloft (blue line).

Figure 3.11 quantifies the percentage of particles remaining inside the pit after a given time since their injection (lifetime). In this case we injected a single puff of 18000 particles at the beginning of the last hour at different locations: a horizontal disc 10 meters above the bottom of the pit as the previous one (black lines in figure 3.11), a 50 m-wide ring at 250 meters above the bottom with its outer perimeter touching the surface of the slope of the pit (blue lines), a similar ring at 500 meters (green lines) and another one at 750 meters (red lines). In case W the percentage of particles inside the pit decreases slowly. Some particles, due to the intense recirculation inside the cavity, are able to reach the top edge of the pit and exit the cavity driven by the wind aloft. In this case, particles injected in a disc near the ground (black line) follow a counterintuitive behavior: their discharge is faster than those of particles injected in a ring at 250 meters. This could be related to recirculation: particles injected at 250 meters near the downstream wall are forced to descend before their exit following the



(a) case B, lateral view **(b)** case B, top view

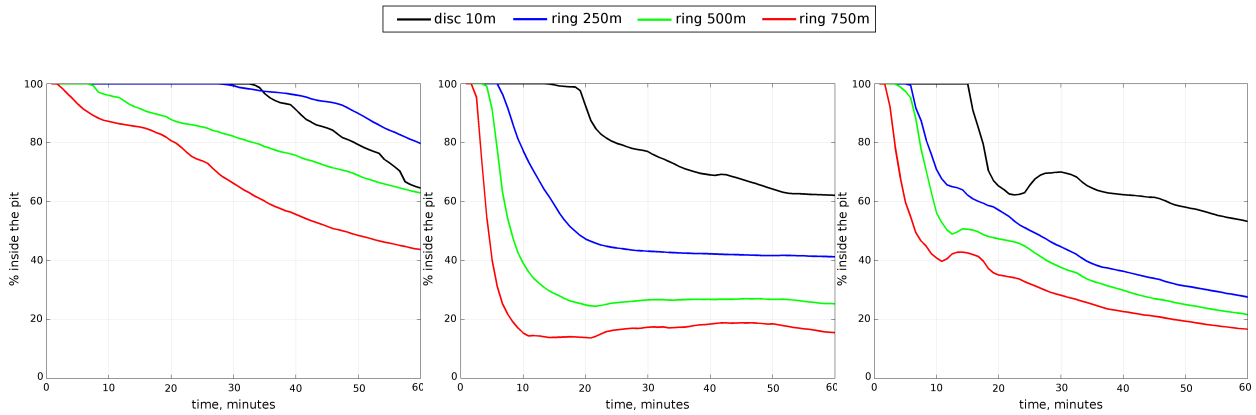


(c) case WB, lateral view **(d)** case WB, top view

Figure 3.10: Trajectory followed by an individual particle released at a point 10 meters above ground in the center of the pit. Three lifetimes: 60 minutes (blue), 40 (green) and 20 (magenta). a) and b) case B. c) and d) case WB.

upstream flow along the upstream wall, while particles injected in a disc near the ground are swept by internal counter-direction flow (figure 3.3b). In case B (figure 3.11b), affected by strong convective currents and weak wind, the percentage of particles injected in a ring that remains inside the pit drops quickly, depending on the altitude of injection. This fast discharge is related to the upward currents close to the walls of the pit. In this same case the discharge of particles injected in a disc close to the bottom of the pit is slower, and takes also a large amount of time to begin. As we already saw, this can be related to the time required

by the particles released at the center of the pit to approach the strong upward currents near the walls. The quasi-steady state reached could be attributed to particles that are not able to exit the pit due to subsidence inside its atmosphere. In case WB (figure 3.11c) the effect of recirculation induced by the stronger wind aloft is responsible for an abrupt change in the rate of discharge, initially similar to that of case B. There is also a reduction in the time needed to initiate the exit of particles injected at the disc near the bottom of the pit. The differences between both cases can be attributed to the recirculation enhanced by the stronger wind in case WB, which favours the entry of particles inside convective currents, accelerating their initial output, but also inducing their re-entry to the pit. In fact, as shown in figures 3.11b and 3.11c, particles injected at the upper ring discharge faster in case B than in case WB, while those injected at the bottom ring discharge faster in case WB than in case B.

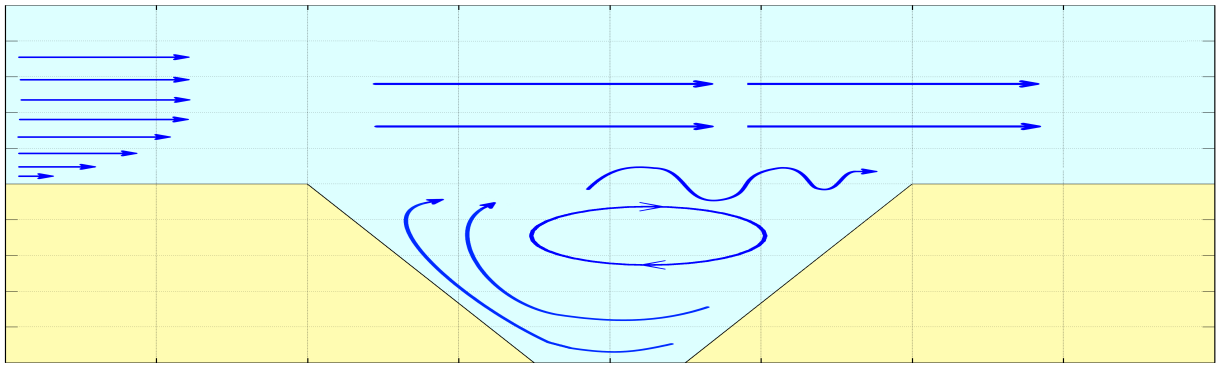


(a) case W **(b)** case B **(c)** case WB

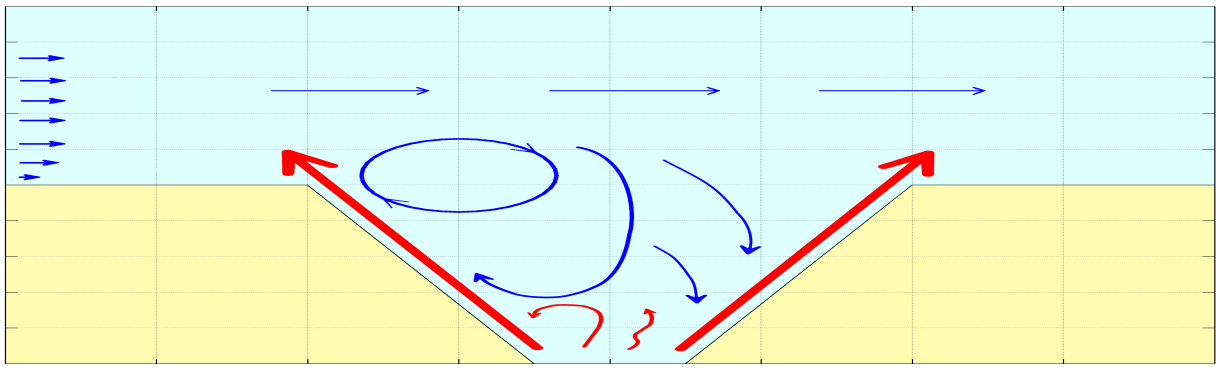
Figure 3.11: Percentage of a puff of particles injected at different levels that remains inside the pit v/s time. a) case W. b) case B. c) case WB. Black line, 500 m diameter disc at 10 meters from the bottom; blue line, 50 m-wide ring at 250 m; green line, ring at 500 m; red line, ring at 750 m.

3.3.4 Conceptual model

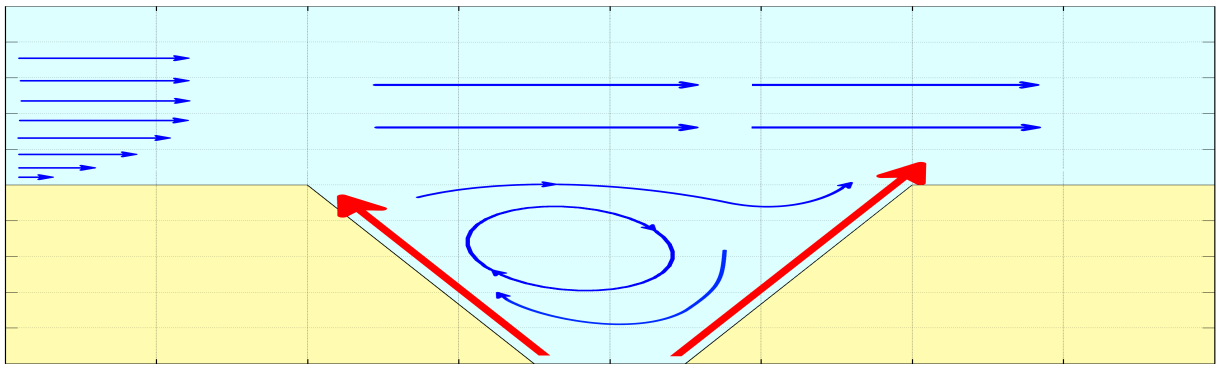
A simplified conceptual scheme is presented in figure 3.12 for each idealized case. In case W (figure 3.12a) the main flow aloft induces recirculation inside the pit. This is a well known pattern of the flow over open cavities (Bres & Colonius (2008); Kang & Sung (2009); Mesalhy et al. (2009)). The magnitude and location of the main roll inside the cavity is a function of the geometry, particularly, the aspect ratio of the cavity (length/depth). The aspect ratio changes the number, location and magnitude of the eddies inside the cavity (see for example figure 2 in Kang & Sung (2009) and 6 in Mesalhy et al. (2009)). In this case the geometry of the pit mainly coincides with a 4:1 aspect ratio, although its conical feature enhances upward and downward currents near the walls, while its circular geometry produces changes in the aspect ratio along the y-axis. Both factors slightly modify the patterns seen in Kang & Sung (2009), where the main eddy inside the pit is located closer to the downstream wall than in this case. Also, as previously described, in our case the structure of the main roll is more clearly defined outside the vertical symmetry plane, since the walls outside the symmetry plane, where the aspect ratio is smaller, highly contribute to the recirculation. Strong turbulence



(a) case W



(b) case B



(c) case WB

Figure 3.12: Simplified conceptual scheme of the flow. a) case W. b) case B. c) case WB. Geometrical scales were modified to improve representation. Vectors are not at scale. In red, convective currents.

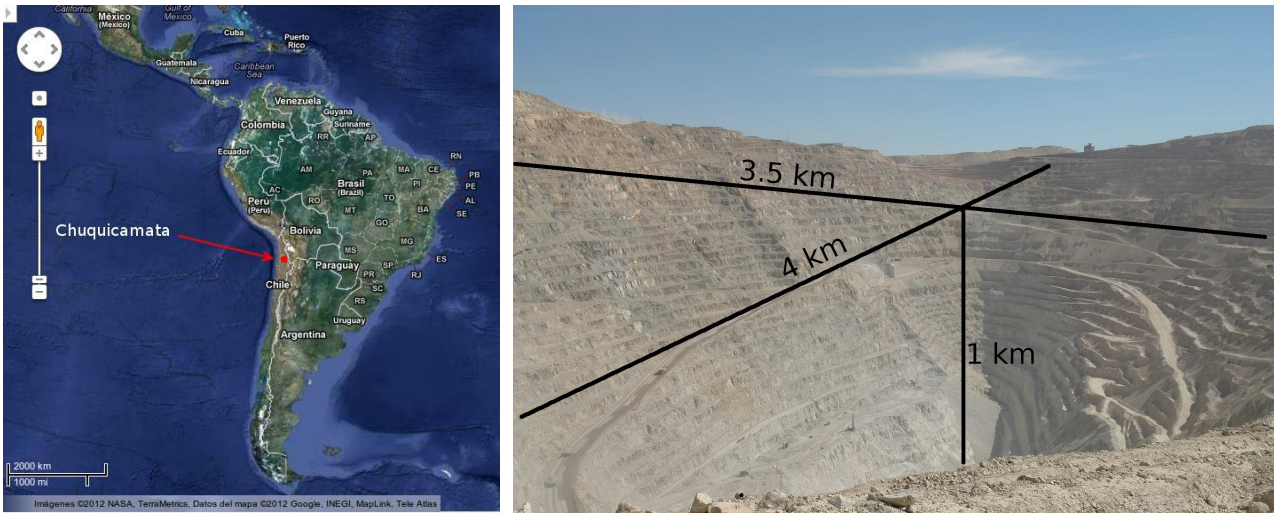
exists at the top of the pit, enhancing the exit of particles that reach that zone carried by ascending currents linked to the main roll. In case B (figure 3.12b) strong convective currents near the walls dominate the flow. The interaction between the low incident wind aloft and the convective updraft leaving the pit upstream generates a roll near the top edge of the pit (similar to that seen over cavities). This roll blocks the influence of the weak main flow over the rest of the atmosphere inside the pit. The strong currents leaving the pit induce, by continuity, descending flow at the center of the cavity. However, due to the intense heat flux at

the base, convective updrafts at the center of the pit remain, overcoming the descending flow near the base. In case WB (figure 3.12c) both schemes described above combine. The strong incident wind is able to overcome the convective current at the upwind wall and generate a roll inside the pit similar to that seen in case W. This roll changes its location and intensity, becoming more intense and approaching the upwind wall. This displacement leaves a wide area of descending flow near the center of the pit, that compensates the intense convective upward currents. The intensity of the main roll sweeps all the convective updrafts near the base at the center of the pit. In this case the effect of buoyancy is similar to changing the aspect ratio of the pit (roll similar to 1:1 aspect ratio cavities (Kang & Sung (2009))). In case W the exit of particles is mainly produced by the intense turbulence at the top levels inside the pit: upslope currents produced mechanically near the upstream wall carry the particles near the top edge, where some of them leave the pit due to the intense turbulence in that zone. In case B the exit of particles is controlled by convective currents near walls, while in case WB the upslope convective current near the upstream wall dominates: recirculation inside the pit sweeps the particles approaching them to the upstream wall, where an intense upslope current carry them to the top edge. There, some of them leave the pit directly and others enter the zone of intense turbulence as in case W.

3.4 Real topography

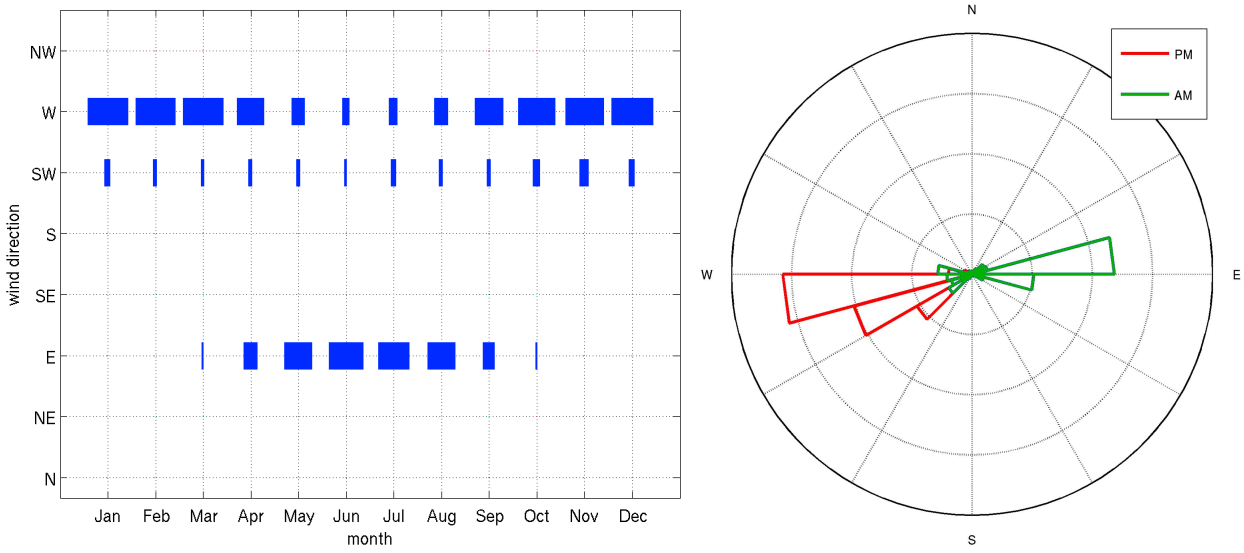
Chuquicamata is an open pit copper mine located in northern Chile (western slope of the Andes, 22°17'20"S 68°54'W, ∼3000 m altitude, figure 3.13a), whose vast dimensions make it one of the largest pits in the world, with more than 4 km long, 3.5 km wide and almost 1 km deep (figure 3.13b). Due to its location in the Chilean desert, Chuquicamata is exposed to high rates of solar radiation all the year (200-350 W/m^2 annual mean surface radiation (Barkan & Alpert (2010))), fostering the generation of strong convective currents inside its atmosphere. Due to operations undertaken inside the pit (movement of machinery, blasting), and the type of soil, fine powder streams are almost always present, generating an environmental problem both inside and outside the pit. The transport of these particles follows a diurnal cycle, influenced by convective currents that promote their dispersion during day (Barkan & Alpert (2010)).

Due to its geographic location, the wind over Chuquicamata is affected by the regional circulation over the Pacific and the Andes cordillera. The upper-level large-scale circulation is characterized by moderate easterly wind at low latitudes (±15° of latitude) and westerly winds at subtropical/extratropical latitudes (Garreaud (2009)). Due to its altitude (∼3000 m a.s.l.) and latitude (22°17'20"S) the mean wind conditions above Chuquicamata are strongly affected by this westerly circulation (Rutllant et al. (2003); Garreaud et al. (2011)). In fact, the data from a weather station in Calama, the nearest city to the pit (located about 13 km southwest of the mine), show a predominance of strong westerly wind during daytime/summer, while weak easterly wind predominates during nightime/winter (figure 3.14). Both patterns are driven by heating/cooling of the Andean slope and the large-scale westerlies. For detailed information see for example the data from Chilean Weather Service (DMC), available at http://www.meteochile.gob.cl/.



(a) location map **(b)** geometry of the pit

Figure 3.13: a) location of Chuquicamata, b) Chuquicamata open pit, most relevant dimensions (©Codelco, Chilean National Copper Corporation, under Creative Commons).



(a) **(b)**

Figure 3.14: Predominant wind direction in Calama. a) Monthly distribution of daily predominant wind direction, 10 years mean (2001-2010). The size of each horizontal bar represents the mean number of days each direction predominates during the month. Data from the archive of the Dirección Meteorológica de Chile (DMC). b) Wind rose of data sampled during day, every 3 hours (08:00, 11:00, 14:00, and 17:00 local hour), 10 years record (2001-2010). AM data (08:00 and 11:00 local hour) in green, PM data (14:00 and 17:00 local hour) in red. Data from NOAA's National Climatic Data Center, Integrated Surface Data.

Because local radiosonde data is not available we used WRF (Weather Research and Forecasting model) simulations to explore the atmosphere over the pit. The model results belong to project "Wind Energy Explorer", developed by the Department of Geophysics at University of Chile under commission from the Ministry of Energy and GIZ (Deutsche Gesellschaft fur Internationale Zusammenarbeit), to explore wind conditions all over the country (detailed data available at http://ernc.dgf.uchile.cl/Explorador/Eolico2/). WRF was applied using a 1 km spatial resolution with 12 vertical levels between 0 and 200 meters simulating an entire year (2010). We selected different locations around Chuquicamata (figure 3.15) to study the wind at 110 meters above ground near the pit. Figure 3.15 shows the complex topography that surrounds Chuquicamata, in particular to the West and NorthWest of the pit. There are several peaks reaching ∼3200 m high, disrupting the surface (Calama ∼2600 m).

Figure 3.16a shows the spatial mean of the daily/annual cycle of wind velocity simulated by WRF during year 2010 in the southwest zone (the zone with simpler topography around the mine, red points with a black cross in figure 3.15). The contours show an increase in wind speed during afternoon throughout the year. Figure 3.16b shows the wind rose associated to the WRF data analysed. West wind (W, NW, SW) appears as the predominant direction. These results are similar for the other zones around the mine (not shown), with small perturbations, probably linked to the complex topography present at these zones.

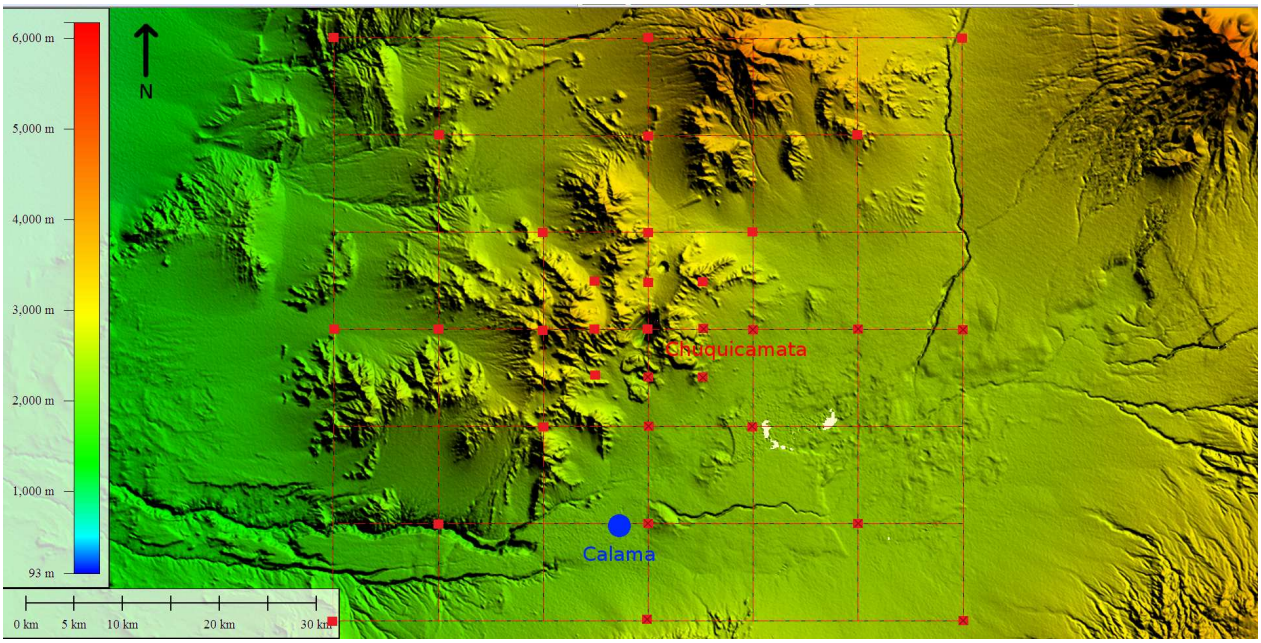
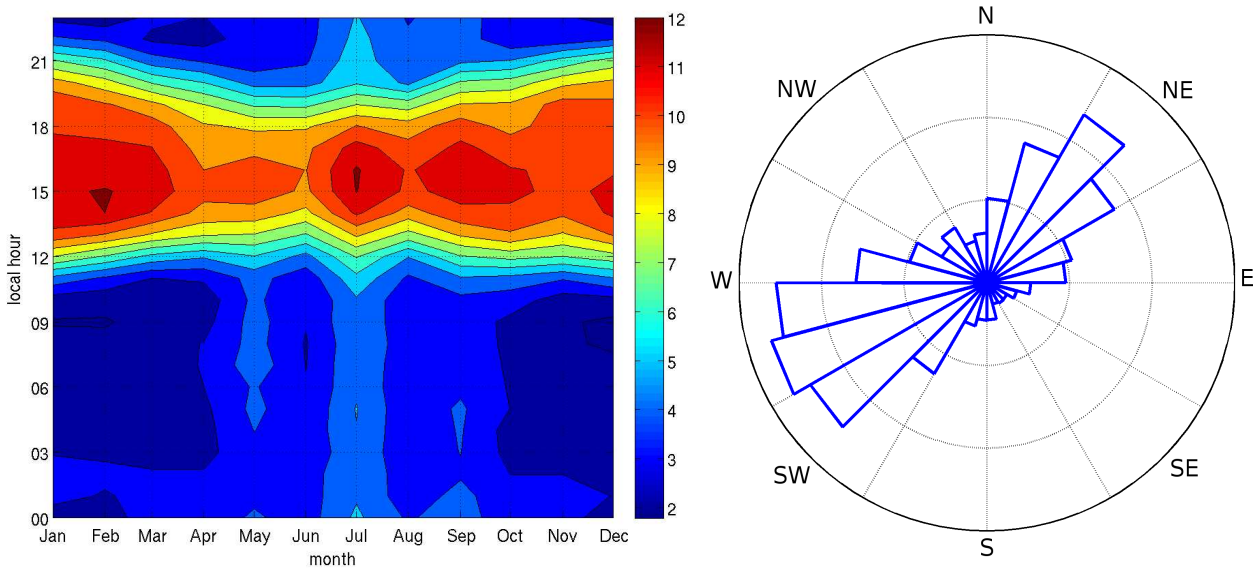


Figure 3.15: Location (red points) of simulation data used to study the wind at 110 meters above the ground near the pit. Chuquicamata at center. A black cross indicates points used to create plots in figure 3.16b.



(a) **(b)**

Figure 3.16: a) Daily/annual cycle of wind velocity (color bar, m/s) at 110 meters of altitude, southeast from Chuquicamata. b) Wind rose at 110 meters of altitude, southeast from Chuquicamata. Mean values from points selected in figure 3.15, year 2010. Data from Eolic Energy Explorer (WRF simulations).

3.4.1 Domain

The topography around Chuquicamata is complex, especially to the west of the pit. Considering that the large scale predominant wind direction during day is westerly, to correctly simulate the flow affecting the pit, the complex topography at its west must be incorporated in the domain. As the configuration of our simulations uses cyclic boundary conditions (to correctly simulate the flow of heat and avoid compressibility problems), the domain and mesh generation of this case is particularly complex. To solve this problem we selected the domain in such a way that both inlet and outlet boundary topographies are similar, and as simple as possible (figure 3.17a), using SRTM data and *GlobalMapper* to create an elevation data file. Then, we modified the file created by the software to include inlet and outlet transition zones at the same level with flat topography (we included different levels until reaching the same level at both boundaries), in order to allow the creation of well defined inlet and outlet cyclic boundaries. Finally, we used again *GlobalMapper* to create the necessary STL file. As expected, this technique highly increases the computational cost of the simulation, increasing the number of cells of the domain, but it allows the inclusion of the complex topography upwind of the pit, in order to run a more realistic simulation. The *snappyHexMesh* tool was used to generate the final mesh based on the STL file, providing a mesh with nearly 7 million cells (figure 3.17b). The figure also shows a detail of the pit included in the mesh. Topographic contour lines are included as reference to highlight the complex topography and the detail required by the mesh, refined close to the terrain. Several mesh tests were performed until obtaining solutions independent of domain size and mesh.

The main differences between the more realistic topography used here and the idealized

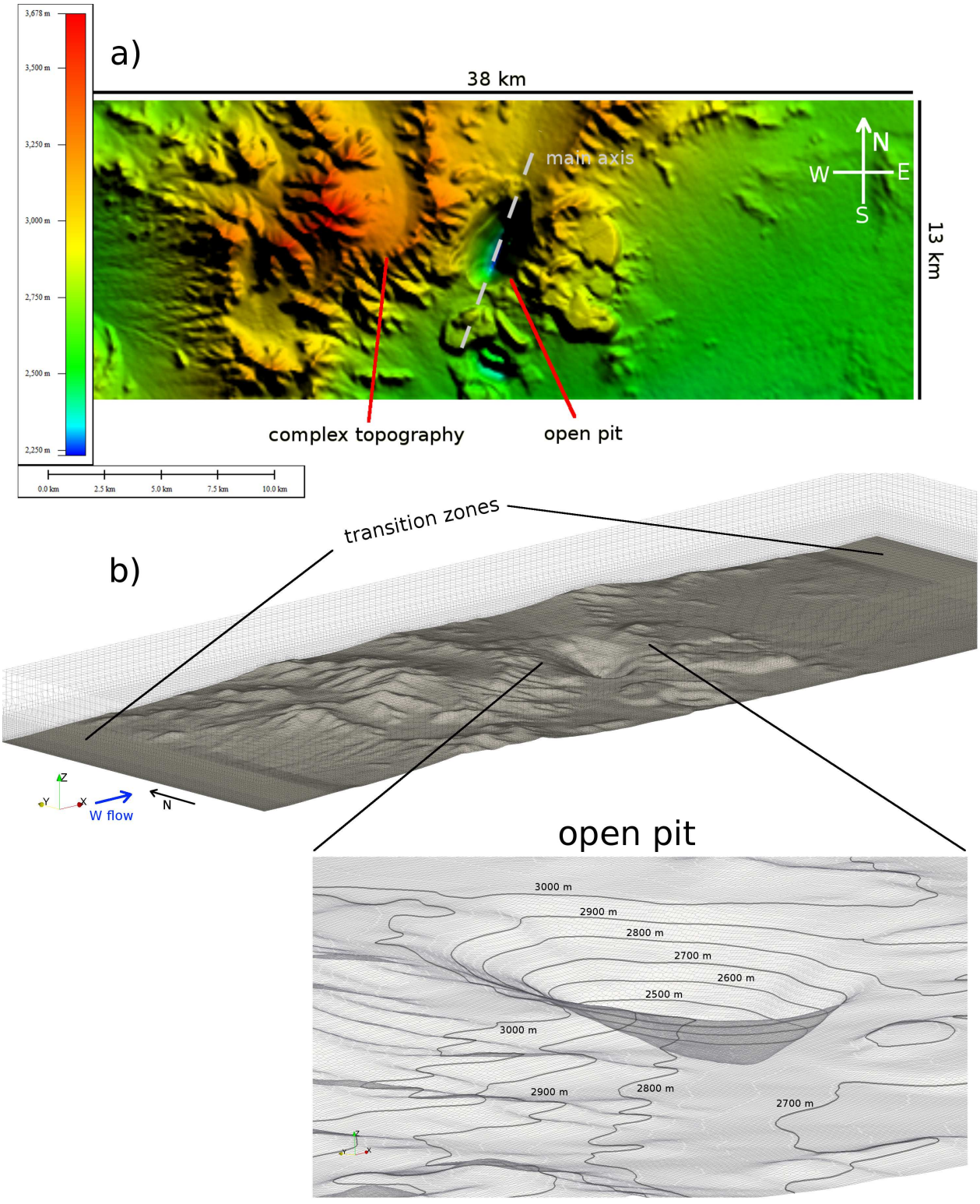


Figure 3.17: a) Topography from *GlobalMapper*. Color bar represents height above sea level. Hills block the inlet flow at the west of the mine. The main axis of the pit is tilted $\sim 15^o$ from north. Size of the domain included. b) *OpenFOAM* mesh, includes inlet and outlet transition zones. Detail of the pit includes topographic contour lines (only contours over 2500 m are visible).

geometry used in the previous section are the following:

- Asymmetry. The real topography shows a clear main axis, especially at the bottom layers inside the pit (figure 3.17), tilted $\sim 15^o$ from north. As a consequence, the base area is much larger than in the idealized case ($\sim$2.5 km against $\sim$1 km), eliminating the circular symmetry near ground.
- Top edge irregularity. The top edge of the real pit is mostly irregular, with different heights depending on location (top edge between 700 and 1000 meters above the bottom of the pit).
- Irregular walls. Real hillsides are perturbed by several irregularities, changing their slope angle along the interior of the pit.
- Surroundings. Hills surround the real pit, particularly at its west (highest peaks reach $\sim$600 meters above the top edge of the mine).

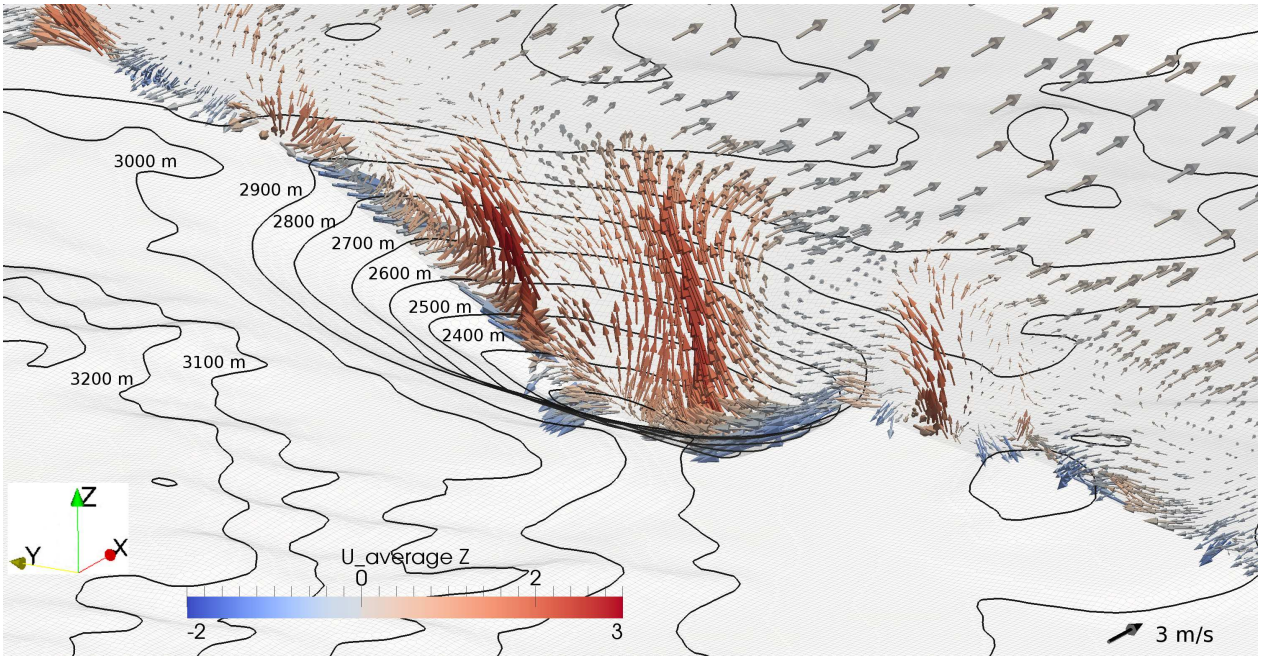
3.4.2 Initial and boundary conditions

We employed cyclic inlet and outlet boundary conditions in the west-east direction. To simplify the mesh generation and limit the size of the domain we did not include cyclic lateral boundaries in the south-north direction, as we imposed a west-east forcing flow (lateral boundaries are defined as inviscid wall: normal component and gradients of tangential components of velocity equal zero). The upper boundary conditions are free-slip and maintain a constant potential temperature gradient. The initial potential temperature profile has a constant value of 300 K up to 937 m above the top edge of the pit (including its internal volume), followed by an 8 K increase in the next 167 m, and a constant 0.003 K/m gradient up to the top of the domain (as used in the idealized cases and in Moeng & Sullivan (1994) and Churchfield et al. (2010)). We simulated two cases, with 3 and 10 m/s wind speed over the top edge of the pit (cases TB and TWB respectively), using a surface heat flux of 240 W/m^2 in both cases (boundary conditions are similar to those of the idealized cases, see section 3.3.2). The time step was 0.01 s, necessary to keep a low Courant number near complex topography, where the mesh is refined, for a full simulation time of 4 hours (reaching a statistically stationary flow). Different combinations of final and averaging times have been tested to ensure that the final converged time-averaged results were attained.

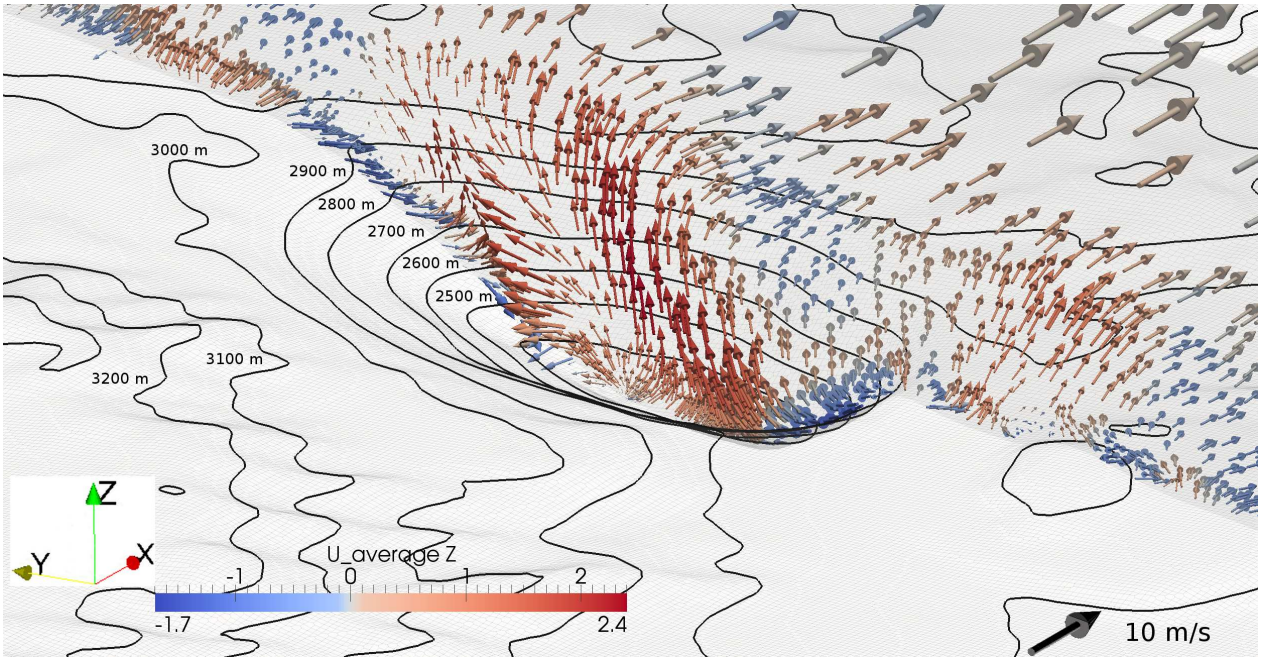
3.4.3 Results

Figure 3.18 shows the last hour mean velocity vector field in a vertical plane aligned with the main axis of the pit (tilted 15^o from north, see figure 3.17), for the two cases simulated. The vector field shows intense convective upward currents inside the pit, that modify the air circulation over and around the mine. Also, the flow near the mine shows a south-north orientation (y-axis, figure 3.17), different from the imposed west-east mean flow (x-axis). The case TB shows stronger uplifts, while case TWB shows a flow more aligned with the imposed main direction.

Figure 3.19, showing the last hour mean velocity vector field in a horizontal plane near



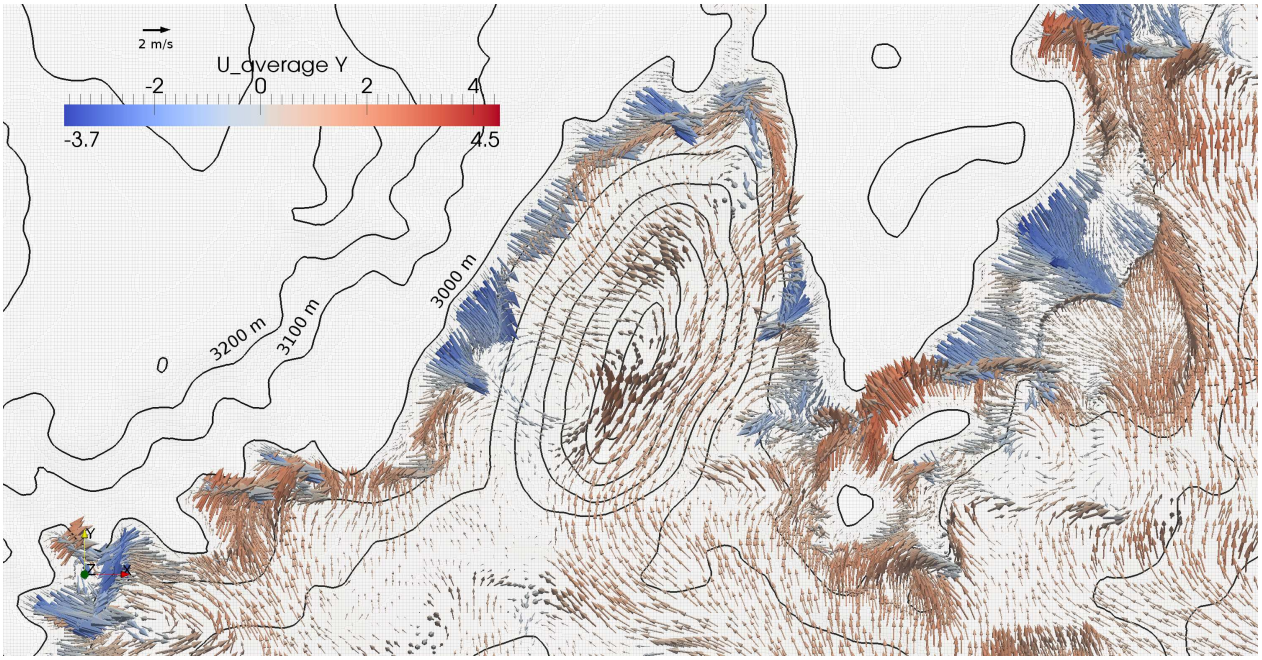
(a) case TB



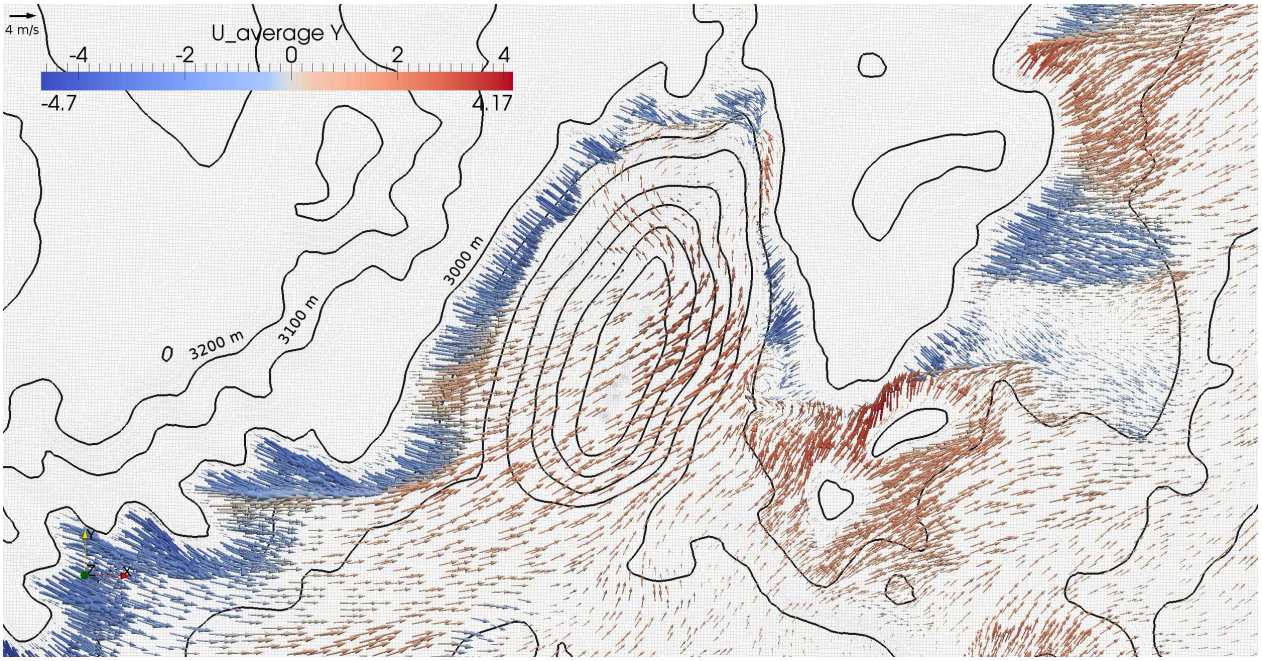
(b) case TWB

Figure 3.18: Last hour mean velocity vector field in a plane aligned with the main axis of the pit (see figure 3.17). a) case TB, wind aloft 3 m/s. b) case TWB, wind aloft 10 m/s. Color bar associated to vertical velocity Uz, m/s. Topographic contour lines included. To improve visualization a different vector scale was used in each case, due to the difference in wind magnitude.

the top edge of the pit, is useful to explain the south-north orientation of the main flow near the mine. Near the pit, positive values of U_y (flow from the south) dominate the circulation, acting as a forcing for the flow entering the cavity, in both cases. This is so because the topography surrounding the pit forms a valley at its south, channelling the mean flow and



(a) case TB



(b) case TWB

Figure 3.19: Last hour mean velocity vector field in a horizontal plane at 3000 meters. a) case TB, wind aloft 3 m/s. b) case TWB, wind aloft 10 m/s. Color bars associated to horizontal velocity component U_y, m/s. Topographic contour lines included. To improve visualization a different vector scale was used in each case, due to the difference in wind magnitude.

forcing it to enter the pit through the lower level of the top edge. The flow configuration seen in figure 3.19 is similar to that seen in figure 3.3, near the top edge of the inverted cone used in the idealized simulations, with mean flow entering from the southwest. Stronger convective updrafts are seen at the center of the pit in the case with real topography. These

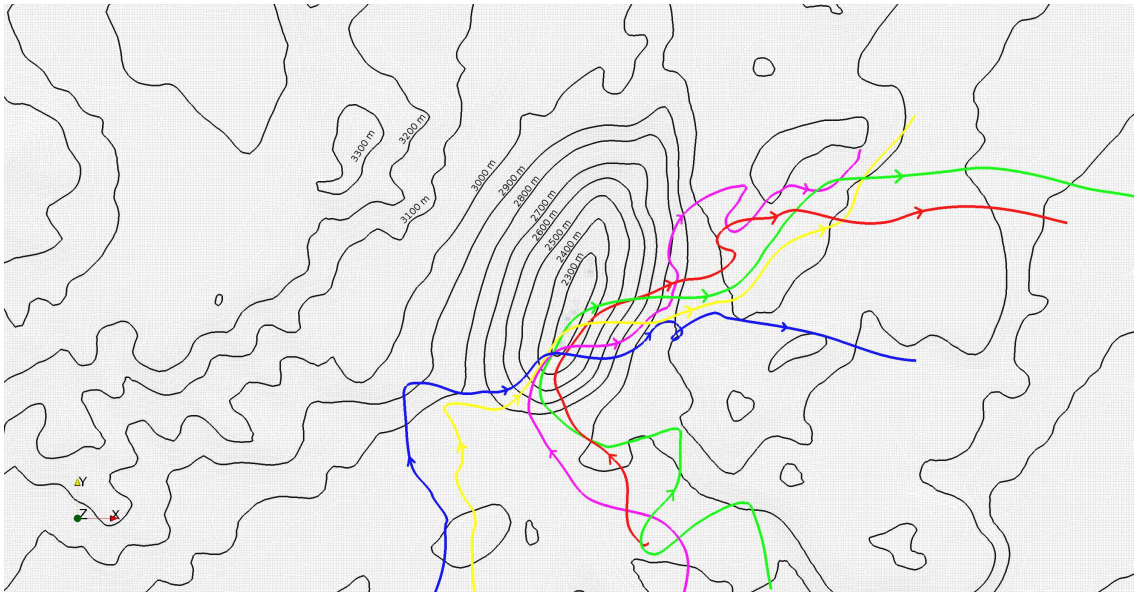
convective updrafts at the center of the cavity, far from walls, are also present in the idealized simulation (figure 3.3c), but with less intensity, probably due to the symmetric configuration of the ground and reduced extension of the main axis of the bottom surface. Both factors increase recirculation at the center of the pit and perturb the development of convective currents at that zone seen in the idealized cases. The high concentration and magnitude of convective updrafts seen at the center of the pit in figure 3.19 originates downward currents that descend over the contour of the mine.

In order to quantify the relative importance of internal and near-wall convective currents in each buoyant case, we computed the mean vertical velocity at 250 meters above the bottom of the pit, dividing the horizontal plane inside the cavity in two zones: an internal region, far from the walls of the pit, and a near-wall region, less than 150 meters from the hillsides. The results are shown in table 3.4. As suggested by the vector fields in figures 3.3 and 3.18, clear differences between buoyant idealized cases (B and WB) and real cases (TB and TWB) exist. In idealized cases the mean value of vertical velocity is positive near the walls of the pit, due to high speed convective currents, and negative in the internal region, due to compensating subsidence. In cases with real topography, however, this configuration reverses, with positive mean vertical velocity far from the hillsides and negative close to them ($\overline{W}$ internal < 0 in cases B and WB and > 0 in cases TB and TWB). These results confirm the effect of the recirculatory flow, induced by the interaction of the wind that sweeps over the cavity and the internal atmosphere, on the vertical circulation inside the pit, fostering descending currents far from walls. In idealized cases stronger wind aloft (WB) increases both the negative value of internal mean velocity and the positive value of near-wall vertical mean velocity. In cases with real topography stronger wind aloft (TWB) decreases both the positive value of internal mean vertical velocity and the negative value of near-surface mean vertical velocity. Also, the air flux ($\overline{F}$) is larger in case WB, where recirculation is more important. This confirms the previous idea of recirculation induced by the wind that sweeps the pit favoring descending currents inside its atmosphere, and evinces a link between wind aloft, geometry, recirculation and internal convective currents. In idealized cases the variation of the aspect ratio along the y-axis, caused by the circular geometry of the pit, induces strong recirculation inside the cavity, that inhibits the formation of upward currents inside the pit. In non-idealized cases the particular geometry of the pit, long and narrow, is aligned with the main direction of the flow, that is also channelled by the surrounding topography entering the pit through the lower levels of the top edge. Both factors increase the aspect ratio, reducing the recirculation, especially that linked to lateral walls in idealized cases. This increases the magnitude of internal updrafts, allowing the main flow to sweep the lateral hillsides of the pit.

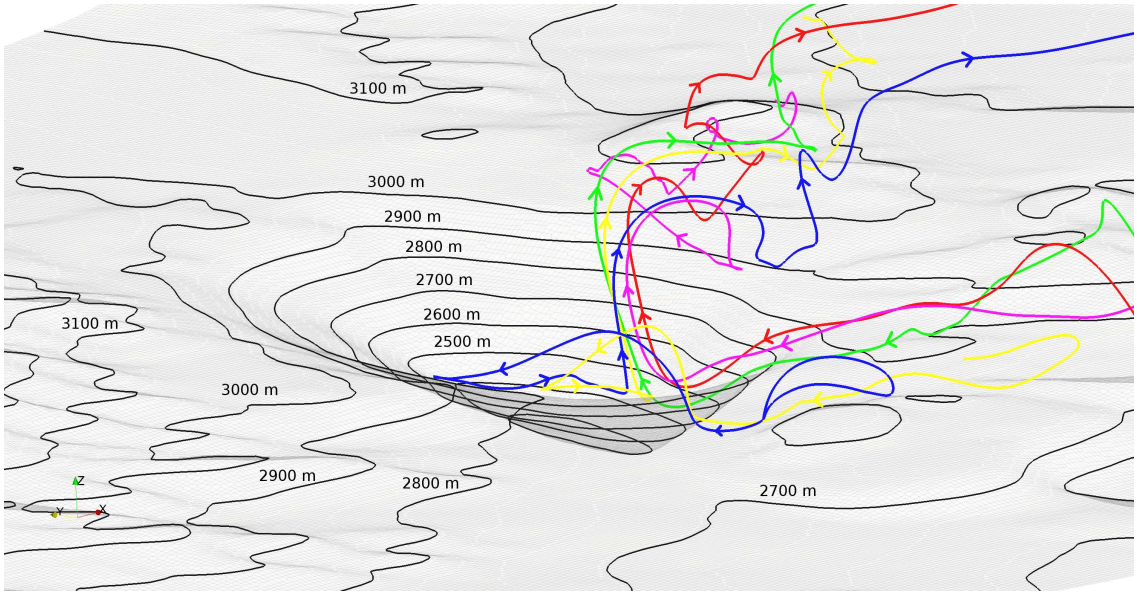
Table 3.4: Mean vertical velocity ($\overline{W}$, m/s) and flux ($\overline{F}$, dam^3/s) at 250 meters above the bottom of the pit, for each buoyant case simulated (B and WB, idealized geometry; TB and TWB, real topography). The horizontal plane inside the cavity was divided in two zones: an internal region, far from the walls of the pit, and a near-wall region, less than 150 meters from the hillsides.

region \ case	**B**		**WB**		**TB**		**TWB**	
	$\overline{W}$	$\overline{F}$	$\overline{W}$	$\overline{F}$	$\overline{W}$	$\overline{F}$	$\overline{W}$	$\overline{F}$
Near-wall	+0.67	+580	+0.85	+740	-0.55	-550	-0.40	-420
Internal	-0.38	-585	-0.51	-780	+1.06	+600	+0.83	+470

Figure 3.20, showing streamlines of the last hour mean flow passing through five points at 10 meters above ground level near the southernmost top edge of the pit, depicts the topographic channelling described above, which is followed by updraft currents near the center of the pit, and downstream flow above, where the imposed west-east flow dominates (similar behavior in both cases simulated, only case TB shown).



(a) top view

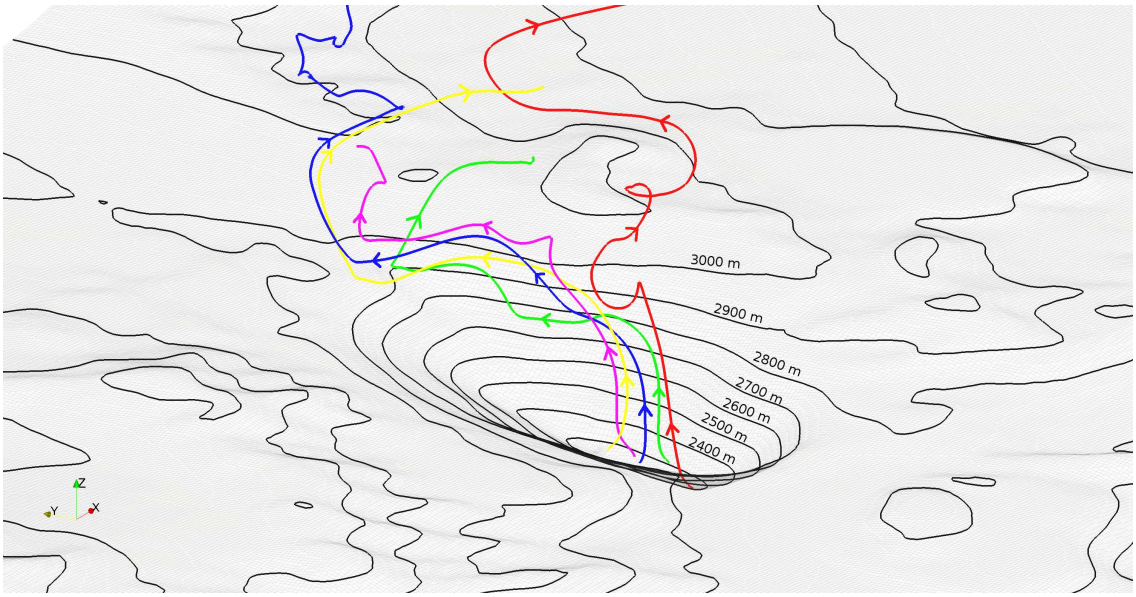


(b) isometric view

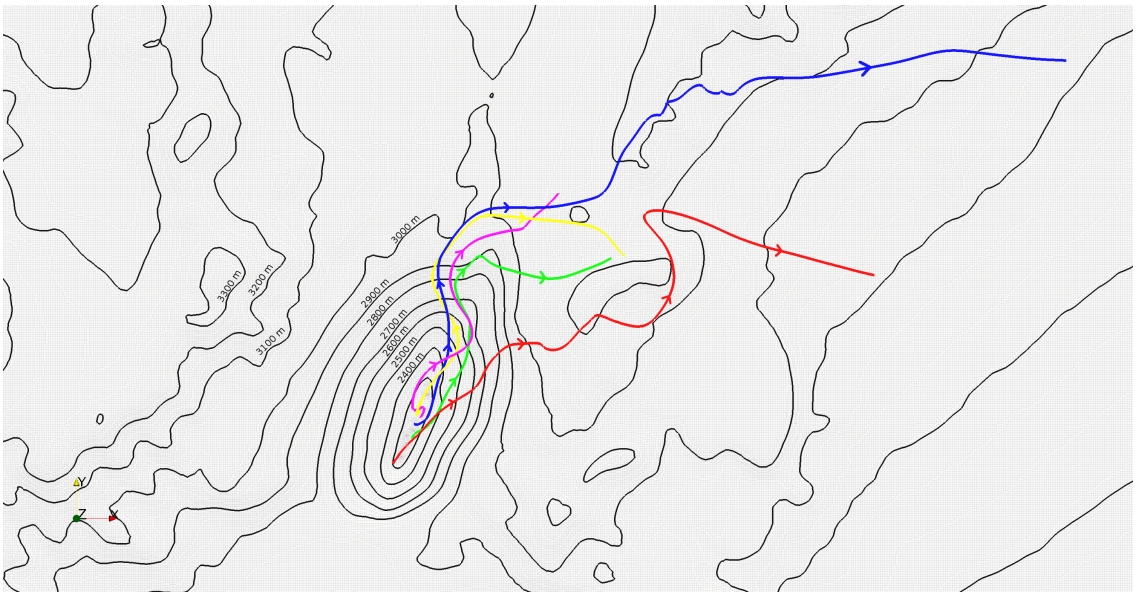
Figure 3.20: Streamlines of the mean flow passing through five points at 10 m above the ground level near the southernmost top edge of the pit, case TB. Similar behavior at case TWB.

Figure 3.21 shows streamlines of the last hour mean flow seeded from five points at 20 meters above ground at the deepest zone inside the pit. Convective currents leaving the basin are well depicted, and confirm the main role played by vertical upward currents inside the pit in the exit of particles from the mine in the cases that include topography. Also,

the streamlines show the south-north flow dominating at low altitude over the pit, and the west-east mean flow controlling the circulation at higher altitudes (similar behavior in both cases simulated, only case TB shown).



(a) 3D view

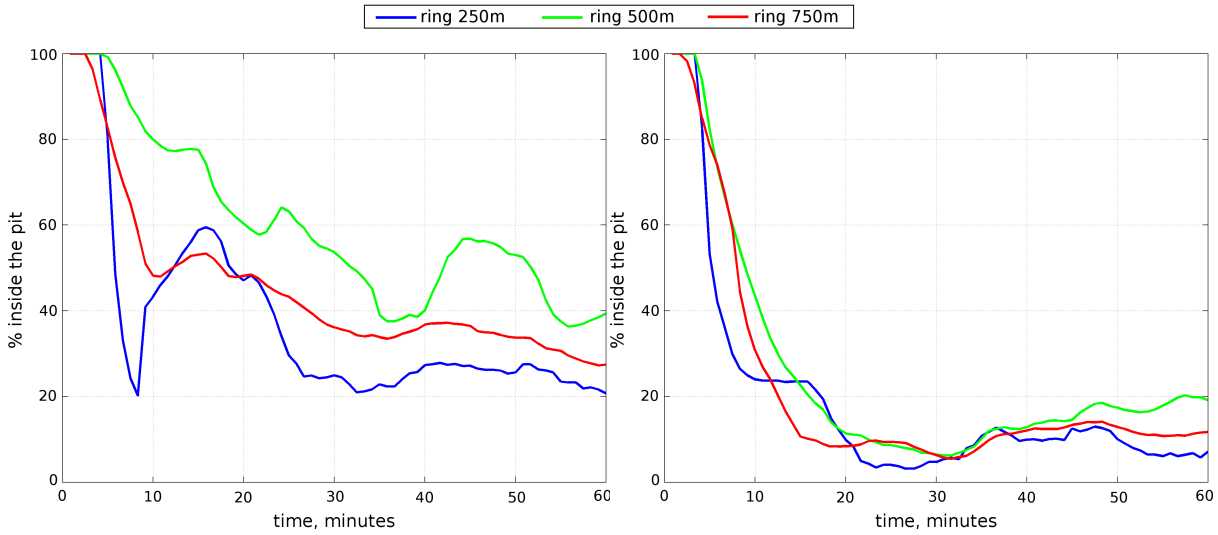


(b) top view

Figure 3.21: Streamlines of the mean flow seeded from five points at 20 m above ground at the deepest zone inside the pit. Case TB, similar behavior at case TWB.

In these complex-topography cases strong upward convective currents at the center of the cavity, induced by the particular topography of the pit, accelerate the exit of particles from the interior of the cavity. As in idealized cases, we injected particles inside the pit and studied their evolution (figure 3.22). The graphs show the percentage of particles remaining inside the pit v/s time. To simulate the generation of fugitive dust near the surface of the pit we injected a single puff of particles (at t = 3 hours), at different locations: a 50 m-wide ring

at 250 meters above the bottom, that follows the surface of the slope of the pit (blue lines), a similar surface at 500 meters (green lines) and another one at 750 meters (red lines). In both cases a quick drop in the percentage of particles remaining inside the pit is seen. Even if the particles are injected near the walls they are quickly affected by internal convective updrafts. Since in this case the exit of particles is not controlled by upward currents near the walls, the level of the injection does not affect directly the percentage remaining inside the pit, as occurred in idealized cases. A quick change in the rate of discharge is seen after 10 minutes, with particles reentering the pit, especially those injected near the bottom of the cavity (blue lines in figure 3.22). This process is probably linked to downward currents associated to convective updrafts (large vertical eddies). The presence of strong interior convective updrafts explains the low percentage of particles remaining inside the pit, that in case TWB is even lower than those seen in idealized cases. Like in previous idealized cases, stronger wind increases the exit of particles.



(a) case TB **(b)** case TWB

Figure 3.22: Percentage of a puff of particles injected at a ring at different levels that remains inside the pit v/s time. a) TB, b) TWB. Blue line, particles injected at a ring 250 m above the bottom; green, ring at 500 m; red, ring at 750 m.

3.4.4 Conceptual model

A simplified scheme of the flow is summarized in figure 3.23. The presence of an inlet valley modifies the mean imposed wind, channelling the flow reaching the pit. As a consequence the flow sweeps the pit along its principal axis, and enters the cavity through the lower level of the top edge. This, together with the particular topography near the lower level of the pit, narrow in one axis and long in other, modifies the 3D aspect ratio of the cavity, reducing recirculation inside the pit, allowing the development of intense internal convective updrafts. Due to continuity, downward currents descend over the walls, at both sides of internal updrafts, where they interact with incident wind, that sweeps the lateral hillsides of the pit. The existence of intense internal convective currents and reduced recirculation inside the pit fosters the exit of particles from the mine and reduces their residence time inside its atmosphere.

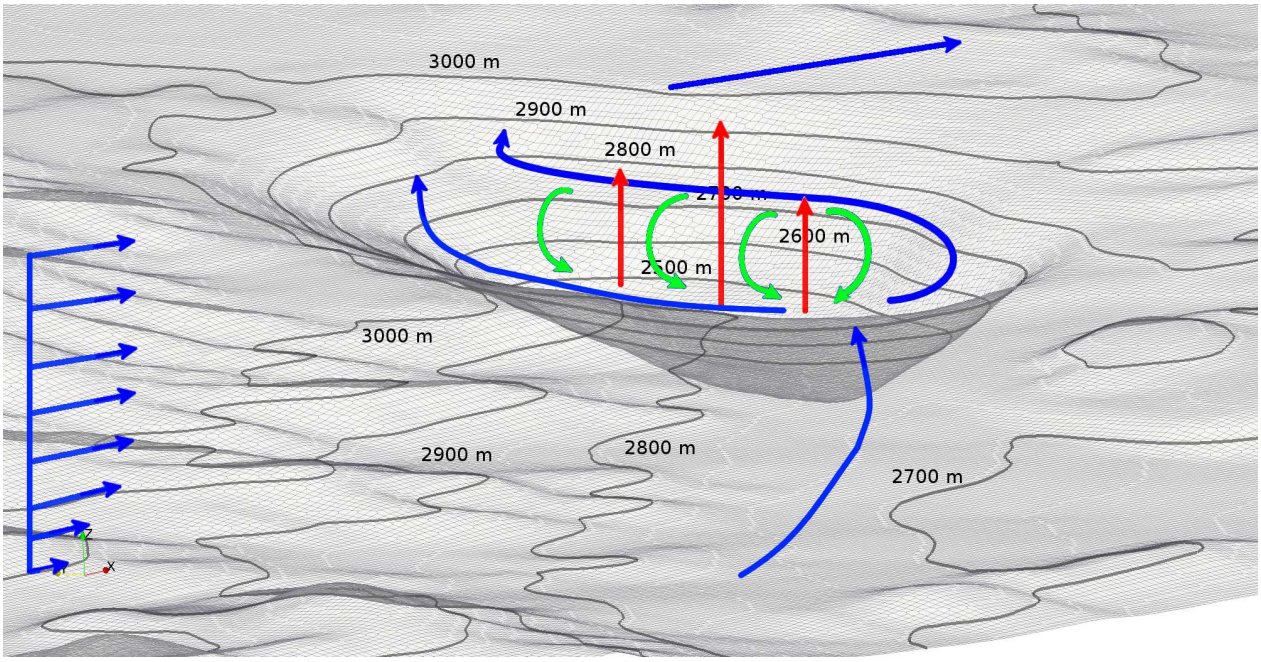


Figure 3.23: Simplified conceptual scheme of flow circulation. Vectors are not at scale.

3.5 Conclusions

We have applied a CFD solver to study the turbulent buoyant atmospheric flow inside large open pit mines under intense insolation, and its effect over pollutant dispersion. Using a DES approach we have incorporated buoyancy, stratification, developed turbulence and complex topography in our analysis. Three idealized cases were studied, and two full scale simulations using the complex topography of Chuquicamata were performed.

The simplified conceptual model of the air circulation seen in each case shows clear differences. In idealized case W, without buoyant currents, the main flow aloft induces recirculation inside the pit, while strong turbulence exists at the top of the pit, enhancing the exit of particles that reach that zone carried by ascending currents linked to the main roll. In idealized case B, dominated by buoyancy, strong convective currents near the walls dominate the flow, enhancing the exit of particles, while a descending flow is induced at the center of the cavity by continuity. Both previous schemes combine in idealized case WB, where intense convective currents near walls enhance the main roll inside the pit. In both cases with complex topography, TB and TWB, the inlet valley modifies the mean imposed wind, channeling the flow, that now sweeps the cavity along its principal axis, and enters the pit through the lower level of the top edge. This, together with the particular topography near the lower level of the pit, narrow in one axis and long in other, modifies the 3D aspect ratio of the cavity, reducing recirculation inside the pit, allowing the development of intense internal convective updrafts.

The results of this work point to the key role played by buoyant currents fostering the dispersion of contaminants inside and outside large open pit mines under intense insolation: despite the large size of the pit considered, in all cases buoyant currents contribute to the exit

of a large percentage of the particles injected inside the pit (> 60 % after 1 hour from the injection). In particular, buoyancy modifies the flow patterns that the purely mechanically-induced recirculation generates inside the pit, reducing the particle residence time seen in the purely mechanical case (non-buoyant case). The different cases studied, with and without a complex topography, also shed light in the role of geometry in the flow patterns, especially in flow recirculation. Given that the aspect ratio of the cavity controls the intensity and location of the main roll inside the pit, idealized cases, with a circular geometry that produces a reduction of the aspect ratio along the axis perpendicular to the main flow, show intense recirculation inside the pit, even in cases with strong buoyancy. Non-idealized cases, on the other hand, have an inlet valley that aligns the main flow with the main axis of the cavity, forcing it to enter the cavity through the lower level of the top edge, and have a non circular geometry that is aligned with the main flow. Both factors highly increase the aspect ratio, reducing the recirculatory flow induced mechanically inside the pit, that is now unable to overcome the intense buoyant currents at the center of the cavity. Anyhow, whether by internal or near wall upward currents, in all cases considered a large percentage of the particles injected inside the pit leaves the cavity after 30 minutes. Idealized cases also showed that recirculation contributes to the re-entry of particles at the top edge, reducing the exit of particles.

Further experiments studying the effect of 3D aspect ratio over the mechanically forced internal flow are needed to understand the precise effect of the internal geometry of the pit over the flow. As reported in this work, this is particularly important considering the variations of the cavity aspect ratio along the axis perpendicular to the main flow seen in large scale open pit mines.

The present work may give more insight into the complex patterns of circulation that affect Chuquicamata, allowing the study of measures to minimize the effect of fugitive dust over the operations. Also, a similar framework as the one presented here can be used to study similar large scale atmospheric flows that include complex topography, developed turbulence, stratification and strong buoyancy, such as dispersion of contaminants in urban environments, or wind energy studies.

Chapter 4

Conclusions

This thesis has addressed the development of a CFD solver in *OpenFOAM* platform for studying complex atmospheric flows that include developed turbulence, stratification, thermal buoyancy and complex geometry, in order to simulate the atmospheric circulation within large open pit mines under intense insolation, and to identifie the main modes of contaminant transport inside their atmosphere. The model includes specific boundary and initial conditions needed to simulate the problem and uses a DES approach, combining LES to correctly simulate developed turbulence with RANS to solve the flow close to walls. Density was included as a variable in the system of equations to improve the treatment of stratification and buoyancy. To incorporate complex geometries we used the *snappyHexMesh* tool, obtaining fully operative meshes.

In the first part of this work we used different case studies to test our framework at different levels. Simple non-buoyant experimental cases were used to check the simulation of recirculation and the perturbation in the wake of obstacles. More complex large scale non-buoyant cases were used to study the ability of the *snappyHexMesh* tool to generate detailed complex meshes fully integrated to the numerical formulation used by *OpenFOAM*. Buoyancy was introduced using first a simple experimental case, and then studying in detail a well documented atmospheric case. Both buoyancy and stratification demonstrated being well simulated, being the results of our test simulations in agreement with previous observational and modelling data. While DES and URANS techniques showed similar results in simple non-buoyant cases, in strongly buoyant cases the ability of DES to solve developed turbulence was not replicated by URANS techniques. Although there are still many points to improve, such as the efficiency of the subgrid model or the treatment close to the walls, it was possible to show that the model and the particular boundary and initial conditions implemented in *OpenFOAM* for this work can deal with the complex multiphysical problem that implies modelling atmospheric flows close to a complex ground.

Despite its limitations and being still a technique under development, DES is a good alternative for making use of the advantages offered by LES when modelling turbulent atmospheric flows, without having to pay the high computing cost that would be implied by using this technique in complex geometries or topographies. Much research is still required to define the most efficient ways of integrating the RANS models that are so well-known in wind

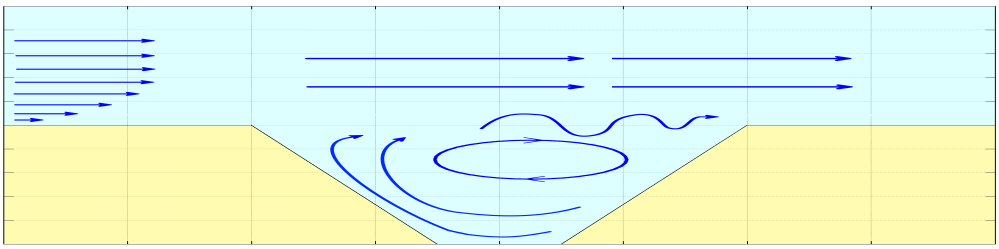
engineering to the LES simulations, in particular when considering convective atmospheric cases, but tools such as those allowed by the *OpenFOAM* platform will be of great help in this development. The modular characteristic of *OpenFOAM* allows equations and submodels to be added to a principal model as the simulation becomes more complex. This makes it advisable to address the modelling of complex atmospheric flows through a methodology like that used in this work, incorporating physical processes to the simulation progressively.

Given the good results obtained when simulating the convective case, in the second part of this work we applied the solver to study the turbulent buoyant atmospheric flow inside large open pit mines under intense insolation, and its effect over pollutant dispersion. Three idealized cases were studied, and two full scale simulations using the complex topography of Chuquicamata, one of the largest open pit mines in the world, were performed. The simplified geometry allowed a clearer study of the processes interacting in the formation of the air circulation inside the pit, while the more realistic topography included the effect of the real geometry.

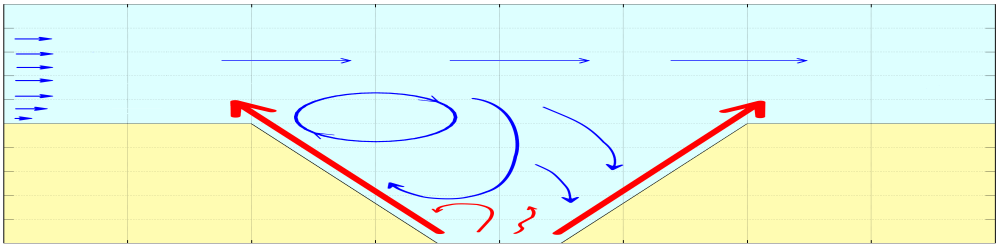
The first idealized case, W, considered strong winds aloft (10 m/s at 300 m over the top edge of the pit), and did not include any thermal effect. Idealized case B was dominated by buoyancy, with a surface heat flux of 240 W/m^2, and only a 1 m/s wind speed 300 m above the top edge of the pit. The third idelized case, WB, combined both previous cases, keeping a 10 m/s wind aloft and a 240 W/m^2 surface heat flux. The two non-idealized cases, that included real topography, considered a surface heat flux of 240 W/m^2, with 3 and 10 m/s wind speed over the top edge of the pit (cases TB and TWB respectively).

The simplified conceptual model of the air circulation seen in each case shows clear differences. In idealized case W (figure 4.1a), without buoyant currents, the main flow aloft induces recirculation inside the pit, while strong turbulence exists at the top of the pit, enhancing the exit of particles that reach that zone carried by ascending currents linked to the main roll. In idealized case B (figure 4.1b), dominated by buoyancy, strong convective currents near the walls dominate the flow, enhancing the exit of particles, while a descending flow is induced at the center of the cavity by continuity. Both previous schemes combines in idealized case WB (figure 4.1c), where intense convective currents near walls enhance the main roll inside the pit. In both cases with complex topography, TW and TWB (figure 4.2), the inlet valley modifies the mean imposed wind, channeling the flow, that now sweeps the cavity along its principal axis, and enters the pit through the lower level of the top edge. This, together with the particular topography near the lower level of the pit, narrow in one axis and long in other, modifies the 3D aspect ratio of the cavity, reducing recirculation inside the pit, allowing the development of intense internal convective updrafts.

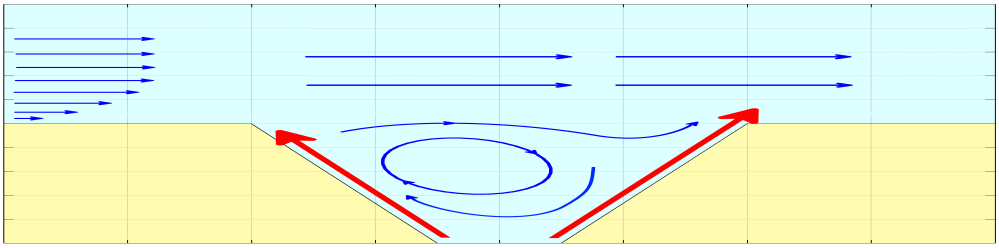
The results of this work point to the key role played by buoyant currents fostering the dispersion of contaminants inside and outside large open pit mines under intense insolation: despite the large size of the pit considered, in all cases buoyant currents contribute to the exit of a large percentage of the particles injected inside the pit. In particular, buoyancy modifies the flow patterns that the purely mechanically-induced recirculation generates inside the pit, reducing the particle residence time seen in the purely mechanical case (non-buoyant case). The different cases studied, with and without a complex topography, also shed light in the role of geometry in the flow patterns, especially in flow recirculation. Given that the aspect



(a) case W



(b) case B



(c) case WB

Figure 4.1: Simplified conceptual scheme of the flow. a) case W. b) case B. c) case WB. Geometrical scales were modified to improve representation. Vectors are not at scale. In red, convective currents.

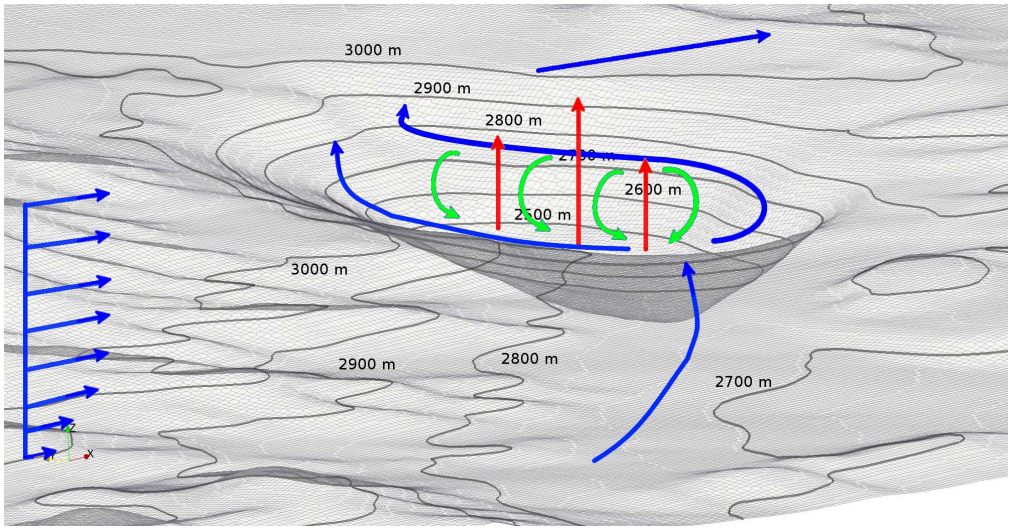


Figure 4.2: Simplified conceptual scheme of flow circulation in non-idealized cases. Vectors are not at scale.

ratio (length/depth) of the cavity controls the intensity and location of the main roll inside the pit, idealized cases with a circular geometry (that produces a reduction of the aspect ratio along the axis perpendicular to the main flow) show intense recirculation inside the pit, even in cases with strong buoyancy. Idealized cases also showed that recirculation contributes to the re-entry of particles at the top edge, reducing the exit of particles. Non-idealized cases, on the other hand, have an inlet valley that aligns the main flow with the main axis of the cavity, forcing it to enter the cavity through the lower level of the top edge, and have a non circular geometry that is aligned with the main flow. Both factors increase the aspect ratio, in particular in the zones outside the simmetry plane aligned with the main axis of the pit, where the circular geometry of previous idealized cases greatly reduced it (in this zones de non circular real topography aligned with the main flow maintains a $\sim$4:1 aspect ratio, while the idealized geometry reduced it to $\sim$1:1). This effect contributes to reduce the recirculatory flow induced mechanically inside the pit, that is now unable to overcome the intense buoyant currents at the center of the cavity. Anyhow, whether by internal or near wall upward currents, in all cases considered a large percentage of the particles injected inside the pit leaves the cavity after 30 minutes.

Further numerical experiments studying the effect of 3D aspect ratio over the mechanically forced internal flow are needed to understand the precise effect of the internal geometry of the pit over the flow. As reported in this work, this is particularly important considering the variations of the cavity aspect ratio along the axis perpendicular to the main flow seen in large scale open pit mines.

The present work may give more insight into the complex patterns of circulation that affect Chuquicamata, allowing the study of measures to minimize the effect of fugitive dust over the operations. Also, a similar framework as the one presented here can be used to study similar large scale atmospheric flows that include complex topography, developed turbulence, stratification and strong buoyancy, such as dispersion of contaminants in urban environments, or wind energy studies.

Bibliography

Akmaev, R., & Juang, H.-M. H. (2008). Using enthalpy as a prognostic variable in atmospheric modelling with variable composition. *Quarterly Journal of the Royal Meteorological Society*, *134*, 2193–2197.

Assimakopoulos, V., Georgakis, C., & Santamouris, M. (2005). Comparison of computed and measured wind fields within street canyons. International Conference 'Passive and Low Energy Cooling for the Built Environment', Santorini, Greece.

Axelsen, S. L. (2010). *Large-eddy simulation and analytical modelling of katabatic winds*. Ph.D. thesis Institute for Marine and Atmospheric Research, Faculty of Science, Department of Physics and Astronomy, Utrecht University, The Netherlands.

Baba-Ahmadi, M., & Tabor, G. (2009). Inlet conditions for LES using mapping and feedback control. *Computers & Fluids*, *38*, 1299–1311.

Baklanov, A. (1995). Numerical modelling of atmosphere processes in mountain cirques and open pits. *Air Pollution III: Air Pollution Theory and Simulation*, *1*.

Baklanov, A. (2000). Application of CFD methods for modelling in air pollution problems: possibilities and gaps. *Journal Environmental Monitoring and Assessment*, *65*, 181–190.

Baklanov, A., & Rigina, O. Y. (1995). Research of local zones atmosphere normalization efficiency by artificial currents. *Transactions on Ecology and the Environment*, *3*.

Barkan, J., & Alpert, P. (2010). The linkage between solar insolation and dust in the major world deserts. *The Open Atmospheric Science Journal*, *4*, 101–113.

Batchelor, G. K. (1967). *An Introduction to Fluid Dynamics*. Cambridge University Press.

Bechmann, A., Sorensen, N., Johansen, J., Vinther, S., Nielsen, B., & Botha, P. (2007). Hybrid RANS/LES Method for High Reynolds Numbers, Applied to Atmospheric Flow over Complex Terrain. *Journal of Physics*, *Conference Series 75*, 012054–012054.

Betts, P., & Bokhari, I. (2000). Experiments on turbulent natural convection in an enclosed tall cavity. *International Journal of Heat and Fluid Flow*, *21*, 675–683.

Biltoft, C. A. (2001). Customer report for Mock Urban Setting Test (MUST). DPG Document WDTC-TP-01-028, West Desert Test Center, U.S. Army.

Blocken, B., Stathopoulos, T., & Carmeliet, J. (2007). CFD simulation of the atmospheric boundary layer: wall function problems. *Atmospheric Environment*, *41*, 238–252.

Blocken, B., Stathopoulos, T., Saathoff, P., & Wang, X. (2008). Numerical evaluation of pollutant dispersion in the built environment: Comparisons between models and experiments. *Journal of Wind Engineering and Industrial Aerodynamics*, *96*, 1817–1831.

Brasseur, J. G., & Wei, T. (2010). Designing large-eddy simulation of the turbulent boundary layer to capture law-of-the-wall scaling. *Physics of Fluids*, *22*, 021303–021303.

Bres, G., & Colonius, T. (2008). Three-dimensional instabilities in compressible flow over open cavities. *J. Fluid Mech.*, *599*, 309–339.

Buccolieri, R., Pulvirenti, B., di Sabatino, S., & Britter, R. (2008). The influence of buoyancy on flow and pollutant dispersion in street canyons. 12th International Conference on Harmonisation within Atmospheric Dispersion Modelling for Regulatory Purposes, Cavtat (Croatia).

Chan, S. T. (2004). Large eddy simulation of turbulent flow and dispersion in urban areas and forest canopies. Workshop on Mesoscale and CFD Modeling for Military Applications, Jackson State University.

Chen, F., Kusaka, H., Bornstein, R., Ching, J., Grimmond, C., Grossman-Clarke, S., Loridan, T., Manning, K. W., Martilli, A., Miao, S., Sailor, D., Salamanca, F. P., Taha, H., Tewari, M., Wang, X., Wyszogrodzki, A. A., & Zhang, C. (2011). The integrated WRF/urban modeling system: development, evaluation, and applications to urban environmental problems. *International Journal of Climatology*, *31*, 273–288.

Churchfield, M. J. (2010). A Description of the OpenFOAM Solver buoyantboussinesqpisofoam. National Renewable Energy Laboratory, National Wind Technology Center.

Churchfield, M. J., Moriarty, P., Vijayakumar, G., & Brasseur, J. (2010). Wind energy-related atmospheric boundary layer Large-Eddy Simulation using OpenFOAM. 19$^{\text{th}}$ Symposium on Boundary Layers and Turbulence, Keystone, Colorado, August 2-6, 2010.

Clements, C. B., Whiteman, C. D., & Horel, J. D. (2003). Cold-air-pool structure and evolution in a mountain basin: Peter sinks, utah. *Journal of Applied Meteorology*, *42*, 752–768.

Colette, A., Chow, F. K., & Street, R. L. (2003). A numerical study of inversion-layer breakup and the effects of topographic shading in idealized valleys. *Journal of Applied Meteorology*, *42*, 1255–1272.

Culpo, M. (2012). Current Bottlenecks in the Scalability of OpenFOAM on Massively Parallel Clusters. PRACE Whitepapers, Partnership for Advanced Computing in Europe.

Deardorff, J. W. (1970). Convective Velocity and Temperature Scales for the Unstable Planetary Boundary Layer and for Rayleigh Convection. *Journal of the Atmospheric Sciences*, *27*, 1211–1213.

Dejoan, A., Santiago, J. L., Pinelli, A., & Martilli, A. (2008). Comparison between LES and RANS computations for the study of contaminant dispersion in the M.U.S.T field experiment. American Meteorological Society.

Demirdzic, I., Lilek, Z., & Peric, M. (1993). A Collocated Finite Volume Method for Predicting Flows at All Speeds. *International Journal For Numerical Methods in Fluids*, *16*, 1029–1050.

Fast, J. D., Zhong, S., & Whiteman, C. D. (1996). Boundary layer evolution within a canyonland basin. part II: numerical simulations of nocturnal flows and heat budgets. *Journal of Applied Meteorology*, *35*, 2162–2178.

Fernando, H. J. S., Zajic, D., Sabatino, S. D., Dimitrova, R., Hedquist, B., & Dallman, A. (2010). Flow, turbulence, and pollutant dispersion in urban atmospheres. *Physics of Fluids*, *22*, 051301–051301.

Ferziger, J., & Peric, M. (2002). *Computational Methods for Fluid Dynamics*. Germany: Springer.

Flores, F., Garreaud, R., & Muñoz, R. (2013a). CFD simulations of turbulent buoyant atmospheric flows over complex geometry: solver development in OpenFOAM. *Accepted, Computers & Fluids*, .

Flores, F., Garreaud, R., & Muñoz, R. (2013b). OpenFOAM applied to the CFD simulation of turbulent buoyant atmospheric flows and pollutant dispersion inside large open pit mines under intense insolation. *Submitted to Computers & Fluids*, .

Flores, F., Rondanelli, R., Díaz, M., Querel, R., Mundnich, K., Herrera, L. A., Pola, D., & Carricajo, T. (2013c). The Life Cycle of a Radiosonde. *Bulletin of the American Meteorological Society*, *94*, 187–198.

Franke, J., Hellsten, A., Schlünzen, H., & Carissimo, B. (2007). Best practice guideline for the CFD simulation of flows in the urban environment. COST Action 732 Quality Assurance and Improvement of Microscale Meteorological Models.

Franke, J., Hirsch, C., Jensen, A., Krüs, H., Schatzmann, M., Westbury, P., Miles, S., Wisse, J., & Wright, N. (2004). Recommendations on the use of CFD in wind engineering. WG2 dissemination.

Fritts, D. C., Goldstein, D., & Lund, T. (2010). High-resolution numerical studies of stable boundary layer flows in a closed basin: Evolution of steady and oscillatory flows in an axisymmetric Arizona meteor crater. *Journal of Geophysical Research*, *115*, D18109–D18109.

García, M., & Boulanger, P. (2006). Low altitude wind simulation over Mount Saint Helens using NASA SRTM digital terrain model. Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06).

García, M., Boulanger, P., Duque, J., & Giraldo, S. (2008). CFD analysis of the effect on

buoyancy due to terrain temperature based on an integrated DEM and Landsat infrared imagery. *Ingeniería y Ciencia*, *4*, 65–84.

Garreaud, R. (2009). The Andes climate and weather. *Advances in Geosciences*, *7*, 1–9.

Garreaud, R. D., Rutllant, J. A., Munoz, R. C., Rahn, D. A., Ramos, M., & Figueroa, D. (2011). VOCALS-CUpEx: the Chilean Upwelling Experiment. *Atmospheric Chemistry and Physics*, *11*, 2015–2029.

Geiger, R. (1965). *The Climate near the Ground*. Harvard University Press.

Han, J., & Pan, H.-L. (2011). Revision of Convection and Vertical Diffusion Schemes in the NCEP Global Forecast System. *Weather and Forecasting*, *26*, 520–533.

Hsieh, K., & Lien, F. (2004). Numerical modeling of buoyancy-driven turbulent flows in enclosures. *International Journal of Heat and Fluid Flow*, *25*, 659–670.

Huang, H.-Y., Stevens, B., & Margulis, S. A. (2007). Application of Dynamic Subgrid-scale Models for Large-eddy Simulation of the Daytime Convective Boundary Layer over Heterogeneous Surfaces. *Boundary-Layer Meteorology*, *126*, 327–348.

Hussein, A. S., & El-Shishiny, H. (2009). Influences of wind flow over heritage sites: A case study of the wind environment over the Giza Plateau in Egypt. *Environmental Modelling & Software*, *24*, 389–410.

Issa, R. (1985). Solution of the Implicitly Discretised Fluid Flow Equations by Operator-Splitting. *Journal of Computational Physics*, *62*, 40–65.

Issa, R. (1986). The Computation of Compressible and Incompressible Recirculating Flows by a Non-iterative Implicit Scheme. *Journal of Computational Physics*, *62*, 66–82.

Jasak, H. (2010). Parallelisation and Scalability in OpenFOAM. International Workshop on Scalable Engineering Software, National Science Foundation, 2 June 2010.

Kang, W., & Sung, H. J. (2009). Large-scale structures of turbulent flows over an open cavity. *Journal of Fluids and Structures*, *25*, 1318–1333.

Khanna, S., & Brasseur, J. (1998). Three-Dimensional Buoyancy- and Shear-Induced Local Structure of the Atmospheric Boundary Layer. *Journal of the Atmospheric Sciences*, *55*, 710–743.

Kiefer, M. T., & Zhong, S. (2010). A numerical modeling study of the nocturnal boundary layer inside Arizona's Meteor Crater. 19$^{\text{th}}$ Symposium on Boundary Layers and Turbulence, American Meteorological Society, 12 april 2010.

Kondo, H., Horiuchi, K., Hirano, Y., Maeyama, N., Ogata, K., Iizuka, S., & Mizuno, T. (2009). Attempt to make guideline to use CFD model for atmospheric environmental assessment in urban area in Japan. The seventh International Conference on Urban Climate, 29 June - 3 July 2009, Yokohama, Japan.

Kondo, J., Kuwagata, T., & Haginoya, S. (1989). Heat budget analysis of nocturnal cooling and daytime heating in a basin. *J. Atmos. Sci.*, *46*, 2917–2933.

Lee, S.-H., Kim, S.-W., Angevine, W. M., Bianco, L., McKeen, S. A., Senff, C. J., Trainer, M., Tucker, S. C., & Zamora, R. J. (2011). Evaluation of urban surface parameterizations in the WRF model using measurements during the Texas Air Quality Study 2006 field campaign. *Atmospheric Chemistry and Physics*, *11*, 2127–2143.

Lesieur, M. (2008). *Turbulence in Fluids*. 3300 AA Dordrecht, The Netherlands: Springer.

Lim, H., Castro, I., & Hoxey, R. (2007). Bluff bodies in deep turbulent boundary layers: Reynolds number issues. *Journal of Fluid Mechanics*, *571*, 97–118.

Lim, H., Yu, T., Glimm, J., Li, X., & Sharp, D. H. (2009a). Subgrid models for mass and thermal diffusion in turbulent mixing. Report, Los Alamos National Laboratory.

Lim, H. C., Thomas, T., & Castro, I. P. (2009b). Flow arround a cube in a turbulent boundary layer: LES and experiment. *Journal of Wind Engineering and Industrial Aerodynamics*, *97*, 96–109.

Magono, C., Nakamura, C., & Yoshida, Y. (1982). Nocturnal cooling of the Moshiri Basin, Hokkaido in midwinter. *J. Meteor. Soc. Japan*, *60*, 1106–1116.

Maureira, J.-C., Baeza, C., & Pérez, T. (2011). *Levque Cluster User Manual*. Universidad de Chile. Major scientific and technological equipment for user facility centers.

Mesalhy, O., Aziz, S. S. A., & El-Sayed, M. M. (2009). Flow and heat transfer over shallow cavities. *International Journal of Thermal Sciences*, *49*, 514–521.

Mochida, A., Tominaga, Y., Murakami, S., Yoshie, R., Ishihara, T., & Ooka, R. (2002). Comparison of various k-ε models and DSM applied to flow around a high-rise building. Report on AIJ cooperative project for CFD prediction of wind environment. *Wind and Structures*, *5*, 227–244.

Modest, M. F. (2003). *Radiative Heat Transfer*. 525 B Street. Suite 1900, San Diego, California 92101-4495, USA: Academic Press.

Moeng, C.-H. (1984). A large-eddy-simulation model for the study of planetary boundary-layer turbulence. *Journal of Atmospheric Sciences*, *41*, 2052–2062.

Moeng, C.-H., & Sullivan, P. (1994). A comparison of shear -and buoyancy- driven planetary boundary layer flows. *Journal of Atmospheric Sciences*, *vol. 51, Issue 7*, 999–1022.

Moin, P., & Mahesh, K. (1998). Direct Numerical Simulation: A Tool in Turbulence Research. *Annual Review of Fluid Mechanics*, *30*, 539–578.

Oliveira, P., & Issa, R. (2001). An Improved PISO Algorithm for the Computation of Buoyancy-Driven Flows. *Numerical Heat Transfer*, *Part B, 40*, 473–793.

OpenCFD (2009). *User and Programmer's Guide*. OpenFOAM. Version 1.6.

OpenCFD (2010). OpenFoam Advanced Training, Version 1.7.x.

Palmer, G., Vazquez, B., Knapp, G., & Wright, N. (2003). The practical application of CFD to wind engineering problems. Eighth International IBPSA Conference, Eindhoven, Netherlands.

Pedruelo, X. (2009). *Modelling of wind flow over complex terrain using OpenFoam*. Master's thesis Department of Technology and Built Environment, University of Gavle.

Pino, M., Piomelli, U., & Candler, G. (2000). Subgrid-scale models for compressible large-eddy simulations. *Theoretical and Computational Fluid Dynamics*, *13*, 361–376.

Piomelli, U., Balaras, E., Pasinato, H., Squires, K., & Spalart, P. (2003). The inner-outer layer interface in large-eddy simulations with wall-layer models. *International Journal of Heat and Fluid Flow*, *24*, 538–550.

Pope, S. B. (2008). *Turbulent Flows*. Cambridge, UK: Cambridge University Press.

Porté-Agel, F., Pahlow, M., Meneveau, C., & Parlange, M. B. (2001). Atmospheric stability effect on subgrid-scale physics for large-eddy simulation. *Advances in Water Resources*, *24*, 1085–1102.

Rigina, O., & Baklanov, A. (1995). Numerical modelling of dispersion of dust-gas plume in the atmosphere after mass explosions. *Transactions on Ecology and the Environment*, *6*.

Rutllant, J., Fuenzalida, H., & Aceituno, P. (2003). Climate dynamics along the arid northern coast of Chile: the 1997–1998 Dinámica del Clima de la Región de Antofagasta (DICLIMA) experiment. *Journal of Geophysical Research*, *108*, 4538–4538.

Satoh, M. (2002). Conservative scheme for the compressible nonhydrostatic models with the horizontally explicit and vertically implicit time integration scheme. *Monthly Weather Review*, *130*, 1227–1245.

Sauberer, F., & Dirmhirn, I. (1956). Weitere Untersuchungen über die kaltluftan-sammungen in der Doline Gstettner-Alm bei Lunz im Niede-rösterreich (Further investigations of the cold air buildup in the Gstettner-Alm doline near Lunz in lower Austria). *Wetter Leben*, *8*, 187–196.

Savijarvi, H. (2006). Radiative and turbulent heating rates in the clear-air boundary layer. *Quarterly Journal of the Royal Meteorological Society*, *132*, 147–161.

Schlichting, H. (1979). *Boundary-Layer theory*. McGraw Hill.

Schmidt, W. (1930). Die tiefsten Minimuntemperaturen in Mitteleuropa (The lowest minimum temperatures in Central Europe). *Die Naturwissenschaften*, *18*, 367–369.

Schmitt, F. G. (2007). About Boussinesq's turbulent viscosity hypothesis: historical remarks

and a direct evaluation of its validity. *Comptes Rendus Mécanique*, *335*, 617–627.

Sharples, J., McRae, R., & Weber, R. (2010). Wind characteristics over complex terrain with implications for bushfire risk management. *Environmental Modelling & Software*, *25*, 1099–1120.

Shi, Y., Feng, X., & Wei, F. (2000). Three-dimensional nonhydrostatic numerical simulation for the PBL of an open-pit mine. *Boundary-Layer Meteorology*, *94*, 197–224.

Silvester, S., Lowndes, I., & Hargreaves, D. (2009). A computational study of particulate emissions from an open pit quarry under neutral atmospheric conditions. *Atmospheric Environment*, *43*, 6415–6424.

Skyllingstad, E. D. (2002). Large-eddy simulation of katabatic flows. *Boundary-Layer Meteorology*, *106*, 217–243.

Spalart, P. R. (2009). Detached-eddy simulation. *Annual Review of Fluid Mechanics*, *41*, 181–202.

Spalart, P. R., Jou, W., Strelets, M., & Allmaras, S. R. (1997). Comments on the feasibility of LES for wings, and on a hybrid RANS/LES approach. Proceedings of 1st AFOSR International Conference on DNS/LES, pp. 137–147.

Squires, K. D. (2004). Detached-Eddy Simulation: Current Status and Perspectives. chapter of book "Direct and Large-Eddy Simulation V, ERCOFTAC Series".

Steinacker, R., Whiteman, C. D., Doringer, M., Pospichal, B., Eisenbach, S., Holzer, A. M., Weihs, P., Adlgruber, E. M.-R., & Baumann, K. (2007). A Sinkhole Field Experiment in the Eastern Alps. *BAMS, American Meteorological Society*, .

Stull, R. B. (1988). *An Introduction to Boundary Layer Meteorology*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Tabor, G., & Baba-Ahmadi, M. (2010). Inlet conditions for large eddy simulation: A review. *Computers & Fluids*, *39*, 553–567.

Takayoshi, T. (2003). The cross comparison of CFD results for flowfield around building models (part3), comparison of prediction accuracy for flowfield around building blocks. Architectural Institute of Japan, NII-Electronic Library Service.

Tennekes, H., & Lumley, J. (1972). *A First Course in Turbulence*. Massachusetts, USA: The MIT Press.

Tominaga, Y., Mochida, A., Yoshie, R., Kataoka, H., Nozu, T., Yoshikawa, M., & Shirasawa, T. (2008). AIJ guidelines for practical applications of CFD to pedestrian wind environment around buildings. *Journal of Wind Engineering and Industrial Aerodynamics*, *96*, 1749–1761.

Tsinober, A. (2004). *An Informal Introduction to Turbulence*. Dordrecht, The Netherlands:

Kluwer Academic.

Vdovin, A. (2009). Radiation heat transfer in Openfoam. Final Assignment for the Course CFD with OpenSource Software.

de Villiers, E. (2006). *The Potential of Large Eddy Simulation for the Modeling of Wall Bounded Flows*. Ph.D. thesis Thermofluids Section, Department of Mechanical Engineering, Imperial College of Science, Technology and Medicine.

Wang, B.-C., Yee, E., Yin, J., & Bergstrom, D. J. (2007). A General Dynamic Linear Tensor-Diffusivity Subgrid-Scale Heat Flux Model for Large-Eddy Simulation of Turbulent Thermal Flows. *Numerical Heat Transfer, Part B*, *51*, 205–227.

Weller, H., Tabor, G., Jasak, H., , & Fureby, C. (1998). A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in Physics*, *12(6)*, 620–631.

Whiteman, C. (1982). Breakup of temperature inversions in deep mountain valleys: Part I. Observations. *J. Appl. Meteor.*, *21*, 270–289.

Whiteman, C. (1986). Temperature inversion buildup in Colorado's Eagle Valley. *Meteor. Atmos. Phys.*, *35*, 220–226.

Whiteman, C. D. (1990). Observations of thermally developed wind systems in mountainous terrain. Atmospheric processes over complex terrain, Meteorological Monographs.

Whiteman, C. D., McKee, T. B., & Doran, J. (1996). Boundary Layer Evolution within a canyonland basin. Part I: mass, heat, and moisture budgets from observations. *Journal of Applied Meteorology*, *35*, 2145–2161.

Whiteman, C. D., Muschinski, A., Zhong, S., Fritts, D., Hoch, S. W., Hahnenberger, M., Yao, W., Hohreiter, V., Behn, M., Cheon, Y., Clements, C., Horst, T., Brown, W., & Oncley, S. (2008). METCRAX 2006 Meteorological Experiments in Arizona's Meteor Crater. *BAMS, American Meteorological Society*, .

Wyngaard, J. C. (2004). Toward Numerical Modeling in the "Terra Incognita". *Journal of the Atmospheric Sciences*, *61*, 1816–1826.

Wyngaard, J. C. (2010). *Turbulence in the Atmosphere*. The Edinburgh Building, Cambridge CB2 8RU, UK: Cambridge University Press.

Xie, Z.-T., & Castro, I. P. (2008). Efficient generation of inflow conditions for large eddy simulation of street-scale flows. *Flow, Turbulence and Combustion*, *81*, 449–470.

Yoshie, R., Mochida, A., Tominaga, Y., Kataoka, H., Harimoto, K., Nozu, T., & Shirasawa, T. (2007). Cooperative project for CFD prediction of pedestrian wind environment in the Architectural Institute of Japan. *Journal of Wind Engineering and Industrial Aerodynamics*, *95*, 1551–1578.

Yoshie, R., Mochida, A., Tominaga, Y., Kataoka, H., & Yoshikawa, M. (2005). Cross Comparisons of CFD Prediction for Wind Environment at Pedestrian Level around Buildings. The Sixth Asia-Pacific Conference on Wind Engineering(APCWE-VI), Seoul, Korea.

Yoshino, M. (1984). Thermal belt and cold air drainage on the mountain slope and cold air lake in the basin at quiet, clear night. *Geo J.*, *8.3*, 235–250.

Yoshizawa, A., & Horiuti, K. (1985). A statistically-derived subgrid-scale kinetic energy model for the large-eddy simulation of turbulent flows. *Physical Society of Japan, Journal*, *54*, 2834–2839.

Zhang, A., Gao, C., & Zhang, L. (2005). Numerical simulation of the wind field around different building arrangements. *Journal of Wind Engineering and Industrial Aerodynamics*, *93*, 891–904.

Appendices

Appendix A

Paper: The Lifecycle of a Radiosonde

Despite great advances in electronic sensors, the accurate measurement of atmospheric variables still remains a complex and expensive task. In particular, to achieve the accurate measurement of vertical profiles, considering spatial and time evolutions of the parameters, is very restrictive. These limitations are particularly important when detailed numerical models, as the one developed in this thesis, are to be validated. This because to study the interaction of the flow with complex geometries or under intense buoyancy, it is required spatial and temporal sampling of distributed data.

Unfortunately, the cost and limitations of traditional radiosondes make it difficult to apply this well-proven technique to detailed studies of vertical profiles near ground. As a way of studying alternatives to remediate these limitations we endeavored to develope a low-cost sonde adaptable to our requirements. In order to achieve these objectives we used open-hardware and software in the prototype, including a recovery approach: one can attempt the recovery of the radiosonde with the help of a precalculated trajectory and a release system.

The result of this work is exposed below, by means of the paper "The Lifecycle of a Radiosonde", authored by Federico Flores, Roberto Rondanelli, Marcos Díaz, Richard Querel, Karel Mundnich, Luis Alberto Herrera, Daniel Pola and Tomás Carricajo, and published in *BAMS (Bulletin of the American Meteorological Society)*, 2013 (Flores et al. (2013c)).

ABSTRACT

The development of scientific instruments was, only a few years ago, confined to universities and electronic companies having highly specialized human and/or technical resources. With the advent of open hardware initiatives, engineers, scientists, hobbyists, and even people with limited electronic skills have been able to tinker with complex electronic systems. Taking advantage of these inexpensive and widely available tools and in the context of an engineering project class for undergraduates, the authors set about building a working radiosonde prototype from the ground up, based on an open hardware platform and easily accessible components. As a result, a fully functional radiosonde has been built that measures, records, and transmits pressure, temperature, humidity, and wind, plus a small camera that stores images on a flash card. A release system was also developed so that the radiosonde can be detached from a balloon upon reaching a certain height, pressure level, or flight time. Once it is released, one can attempt the recovery of the radiosonde with the help of a precalculated trajectory using a numerical mesoscale forecasting model and visualization software. The performance of the sonde was compared with two commercial radiosondes using climate chambers and two field launches. This paper also discusses some of the more interesting capabilities foreseeable for such a platform: 1) collaboration between meteorology and engineering departments in both education and research and 2) development of a flexible hardware platform that allows for an effective way to compare different commercially available sensors and to easily integrate new prototype sensors.

A.1 Introduction

The importance of the radiosonde as a tool for gathering meteorological data is enormous. The radiosonde allows, in a matter of minutes and at relatively low cost, measurement and transmission of data from regions mostly inaccessible to humans. Key advantages of the radiosonde over other observing systems are its relatively low cost, high accuracy, and high vertical resolution in situ vertical profiles of the atmosphere from the surface up to tens of kilometers. The development of the first radiosondes around 1924-31 [see historical discussions on priority and first publication of radiosonde data in DuBois et al. (2002) and Pettifer (2009)] paved the way for numerical weather forecasting and allowed researchers to test the theoretical foundations of dynamical meteorology. Much of what we know about the atmosphere comes from data originally measured and transmitted by one of these instruments, so it is no exaggeration to call the radiosonde the fundamental instrument of meteorological research. Although satellites have filled much of the gaps in our knowledge of the upper atmosphere, there is still the need for radiosondes to provide high-quality and high-resolution data not only for calibration of other observing systems, but also for basic meteorological research, for creating the analyses needed to initialize weather forecasting models, and for detecting climate trends.

Current radiosondes have evolved into a relatively standard and well-developed set of instruments that measure temperature, pressure, and humidity. Wind is currently inferred from the GPS position of the radiosonde, assuming that the carrying balloon acts as a passive tracer with respect to horizontal winds and filtering the pendular motion of the radiosonde

package. Several technological improvements of the radiosonde have been led by the Finnish company Vaisala (especially since the 1980s), which produced about 70% of the radiosondes used globally by 2002 (Dabberdt et al. 2002). The World Meteorological Organization has established requirements for the accuracy of the data gathered by a radiosonde (see, e.g., Nash et al. 2011). For instance, operational standards allow for errors of 1 K in the tropospheric temperature and less than 7.5% in the relative humidity (RH). Operational standards are still too relaxed when compared, for instance, to the magnitude of the climate trends that we expect to detect from the observational record over a period of decades (observed trends in midtropospheric temperature during the last 30 years range from about -0.1 to 0.2 K decade^{-1}; Thorne et al. 2011). Therefore, more stringent requirements are being proposed for climate monitoring that allow for errors of less than 0.3 K in temperature and less than 3% relative humidity in the troposphere (Immler et al. 2010; Nash et al. 2011). Humidity measurements also suffer from low accuracy in the upper troposphere and stratosphere (Miloshevich et al. 2009), regions that provide only a small fraction of the total water vapor in the atmosphere but that have a significant impact on the radiative transfer of longwave radiation (Turner et al. 2012).

Only about 1,000 synoptic radiosonde stations exist around the globe. This is explained by the relatively large operational cost of the ground stations, including the cost of the receiving station, the gas supply, the radiosondes, and the personnel in charge (see, e.g., Douglas 2010). Receiving stations are usually up to $100,000 (U.S. dollars) or at, the lower end, $5,000 for InterMet boundary layer systems that consist mostly of a radio receiver, a modem interface, and the software. Each individual radiosonde costs about $200. The cost of establishing a radiosonde ground station makes it difficult to increase the spatial extent of the radiosonde network, which is particularly deficient in vast regions of the Southern Hemisphere.

Given the relevance of this instrument for research as well as day-to-day operations in forecast centers (weather and air pollution), airports, and other applications, and the availability of inexpensive microcontrollers and sensors, we challenged ourselves to build a radiosonde with commercial off-the-shelf parts and materials within a semester and within reasonable budget constraints. It is fair to ask the question, Why bother reproducing an already well-developed meteorological instrument? The main purpose of our exercise was pedagogical, so even if the radiosonde turned out to be more expensive than a commercial one, there would still be a value in developing it as an engineering project. In addition to the purely pedagogical value of the exercise, building an "open hardware" radiosonde allows for accelerated development by a community of users. Also, we envisioned the prototype of our sonde as a standard platform that could be easily adapted and used by the global community. The possibility of adding and comparing different sensors can potentially lead to improvements in the accuracy, price, weight, and durability of sensors, batteries, actuators, and other components.

We are aware of the existence of projects that bear some similarities to the one we present here. On one side of the spectrum, there are several hobby-oriented projects whose main objective is usually to get images and recordings from the upper atmosphere and usually attempt the recovery of the payload. On the other side, there are several research-oriented projects usually carried out by large labs; they involve the design of specific soundings focusing on a particular research objective. Our project differs from purely hobby-oriented projects in that we attempt to obtain high-quality data; however, our project differs from

purely research-oriented projects in that we have a strong pedagogical component. Our project also differs from a commercial radiosonde in that we attempt to have our project to be completely "open" in both hardware and software platforms, so it can be easily replicated and improved.

In the next section we will briefly describe the logistics of the classwork as well as the hardware and software used for building the radiosonde and programming the microcontrollers as well as the graphical interface of the sounding ground station. In section 3 we describe some of the tests carried out using our instrument. In section 4 we ask the question of whether the engineering students learn any meteorology during the building and operation of the radiosonde. Finally in section 5 we discuss some possible uses and improvements of the radiosonde platform.

A.2 Building the radiosonde

A.2.1 Research and development teams

In the context of a second-year engineering class called Engineering Project Workshop at the University of Chile, the students are asked to "conceive, design, implement and operate an engineering project that gives an innovative solution to a problem in a specific area" (inspired by the socalled Conceive, Design, Implement and Operate (CDIO) approach; Crawley et al. 2007). Like many other meteorology groups around the world, our Geophysics Department is embedded within an engineering school. Many of the faculty in the meteorology group had a background in engineering before moving into the atmospheric sciences and are usually asked to teach sciences to the engineering students. As an applied science, problems in meteorology usually offer a variety of challenges for almost all specialties within a classical engineering school. Therefore, meteorology offers the potential for providing design problems that are interesting for engineering faculty as well as for a large number of students with a breadth of different interests and backgrounds. We believe that the building of a radiosonde is one such problem.

We will illustrate this by enumerating some of the challenges the students faced during the process of building a radiosonde. Some of these issues are usually covered in the typical curricula of engineering as purely theoretical problems; for instance, when studying viscous dissipation in a fluid, students may be asked to calculate the terminal velocity of a balloon filled with a gas lighter than air, or in chemistry students may be asked to find the equilibrium water vapor pressure of a given solution at a certain temperature and pressure. These are usually unrelated problems given in the context of a set of sequential classes in the basic and applied sciences. There is a more meaningful and deep learning when these problems do not appear in isolation and, perhaps even more important, when the resolution of the individual problems is critical for the successful completion of the project. In Table 1, we show examples of some of these tasks and loosely identify them with some engineering specialty. Most of the activities shown in the table were effectively carried out during the semester.

At the beginning of the 15-week semester, we proposed the problem to the 15 students taking the class and initially motivated them about the possible applications of such an instrument. We had the help of six teaching assistants (four students from electrical engineering and two students from meteorology). The class was divided into three groups, each composed of five students and two research assistants, and we gave each group a different task: temperature-humidity measurement, pressure-wind measurement, and radio communications. As the semester advanced, the three groups of students specialized into their designated tasks and were required to interact with the other groups in anticipation of the final integration of the radiosonde components. The student groups suggested two additional features that would distinguish their unit from a commercial radiosonde: 1) a camera to record images during the flight and 2) a release system so that the sonde could be detached from the balloon during the flight and potentially be recovered.

Table 1. Some individual tasks carried out during construction, calibration, and operation of the radiosonde.

Engineering specialty	Radiosonde building and calibration	Sounding
Computer science	• Programming of the microcontroller • Development of the communication protocols for the radio transmission	• Estimation of the wind from the GPS position • Development of a GUI
Chemistry/chemical engineering	• Design and construction of the insulation for the electronics to prevent malfunction of the components at very low temperatures of high humidity • Calibration of RH using air in equilibrium with chemical solutions	• The resistance of the balloon material and the thermodynamics of the sounding determine the burst height and the final diameter of the balloon
Electrical/electronic	• Integration of the sensors with the microcontroller • Estimation of the power consumption of the elements in the radiosonde • Development of the communication protocols for the radio transmission	• Design of antennas to allow for a larger range of data reception • Layout of electronics to avoid interference of the data stream and the GPS signal
Mechanical/aerospace	• Design of an automatic release system for recovery of the payload	• Calculation of the balloon aerodynamics that determines the rate of ascent and initial filling of the balloons
Industrial	• Estimation of the costs and management of the project	• Development of launching mission protocols

Figure A.1: Some individual tasks carried out during construction, calibration, and operation of the radiosonde.

A.2.2 Hardware

The computational core of the radiosonde is an Arduino board, an open-source electronics prototyping platform (http://arduino.cc/). Arduino was selected because it provides a complete, flexible, easy-to-use hardware and software platform that is widely used not only by engineers, but also by artists, designers, and hobbyists (e.g., Sarik and Kymissis 2010). The sensors, GPS, transceiver, release linear motor, memory card, and camera in the radiosonde are connected to the Arduino via a customized daughter board, or "shield" in Arduino nomen-

clature. The ground receiver consists of an Arduino and a transceiver module that are easily connected to a computer where data are stored. Pictures taken by the radiosonde camera are stored internally on the memory card and not transmitted because of bandwidth limitations. Figure A.2 shows a view of the components as they were laid out in our final prototype.

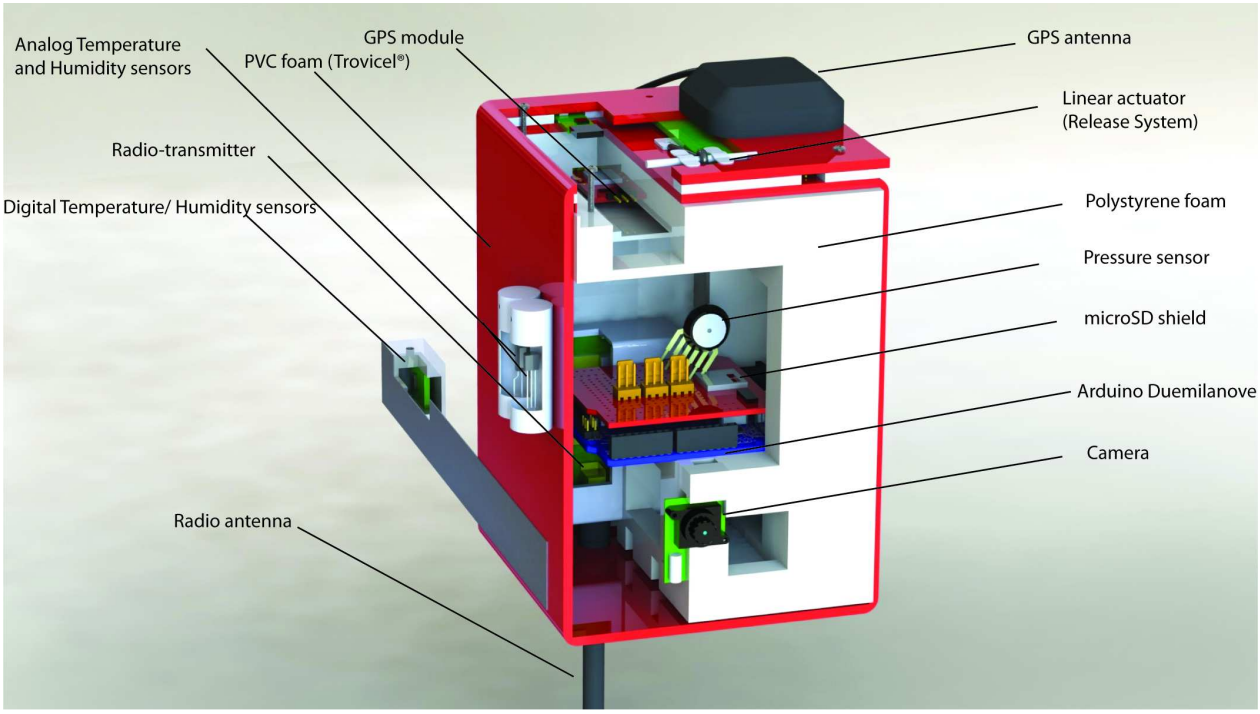


Figure A.2: Cut-away view of the radiosonde, showing the parts and sensors.

The radiosonde and receiver hardware used in our tests were based on the version of the Arduino board called "Arduino Duemilanove". Recently, the source code has been updated to also work with newer versions of the Arduino board: the "Arduino Uno" and "Arduino Mini Pro". The choice of original sensors was made by students in consultation with the teaching assistants and professors. When deciding on sensors we attempted to obtain those that were inexpensive but that could also provide high-quality meteorological data. Table 2 shows a description of the sensors used in building the radiosonde. More detailed specifications of the sensors, parts, and their connections to the Arduino board, as well as circuit diagrams, are available online (at `http://www.dgf.uchile.cl/radiosonde`).

A.2.3 Software

A summary of the information flow from the radiosonde is as follows: Raw data (voltages) are measured by the sensor/transmitter unit and communicated to the receiver unit which is connected to a computer where the voltages are converted to actual meteorological values. The radiosonde software is divided into three distinct programs: Arduino instructions for the transmitter/sensor unit, Arduino instructions for the receiver, and a computer-based graphical user interface (GUI) to process and display the received data.

The Arduino instruction sets are compiled and uploaded to the Arduino boards using a

Table 2. Parts, pieces, and sensors used in building the radiosonde. Prices are in U. S. dollars without taxes or shipping.								
Name	**Description**	**Part number**	**Vendor/ manufacturer**	**Variable/ function**	**Measuring/ operating range**	**Error**	**Response time/sampling frequency**	**Value ($)**
Analogue temperature sensor	10-K thermistor	2322 640	Olimex/Vishay BC-components	Temperature	−40° to 125°C	~1°C	15 s	2.0
Analogue temperature sensor	Two-terminal zener diode with voltage proportional to temperature	LM235Z	Digikey/National Semiconductor Corporation	Temperature	−40° to 125°C	1°C	3 s (at 5 m s^{-1})	1.5
Analogue humidity sensor	Thermoset polymer capacitive sensing element	HIH-4010–001	Digikey/ Honeywell	RH	0%–100% RH	±3.5%	5 s (slow-moving air)	20
Analogue pressure sensor	Integrated silicon piezoresistive transducer with analogue output	MPX5100D	Digikey/Motorola	Pressure	0–1,000 hPa	±25 hPa	1 ms	16
Humidity and temperature digital sensor	Capacitive sensor (RH) band gap sensor (temperature)	SHT10	Sensirion	Humidity, temperature	0%–100 % RH −40° to 124°C	±4.5% ±0.5°C	8 s 5 to 30 s	15
GPS module/ current	GPS Sirf Star III module with internal patch antenna	MOD-GPS	Olimex	Wind, position	−40° to 85°C (<18-km height)	1–3 m s^{-1} (at 10 m s^{-1})	<1 s	50
GPS module/ discontinued	SparkFun GPS micro-mini	MN5010HS	SparkFun	Wind, position	−20° to 85°C (<18-km height)	≤3 m	<1 s	
Communication transceiver	Data radio module	HAC-LM12	Shenzen HAC Technology	Radio transmission	402–405 MHz			50
Camera/ discontinued	Video graphics array (VGA) module	C328–7640	COMedia Ltd	Image capture			5 min	39
Linear actuator	MigaOne−12 linear actuator		Miga Motor Company	Release system				40
Switch	Miga analog driver V5		Miga Motor Company	Release system				14
Microcontroller	Arduino Uno	DEV-10356	SparkFun	Radiosonde brain				30
Board	Micro secure digital (microSD) Arduino Shield printed circuit board (PCB)	DEV-09802	SparkFun	Radiosonde sensors board				15
Battery	Polymer lithium-ion battery 1000 mAh 7.4 V	PRT-10472	SparkFun/ King Max					7

Figure A.3: Parts, pieces and sensors used in building the radiosonde. Prices are in U.S. dollars without taxes or shipping.

command-line interface and makefile. The rather simplistic Arduino integrated development environment was not used since some compilation errors would occur when certain external C++ libraries were included.

The computer-based GUI, called BaseCamp, has been compiled and run on Windows, Linux, and OS X systems (see screenshot in Fig. A.4). All programs and source code will be available for download from the radiosonde project website.

A.3 Calibration and cross comparison

A.3.1 Preliminary experiments

We performed several test experiments in preparation for the main field launch. A tethered balloon campaign was conducted as the end of the semester activity where the radiosonde was first used to transmit meteorological data from an altitude of about 1,000 m above ground level. We also tested the data link from the transmitter to the receiver by communicating between the top of Cerro San Cristóbal (850 MSL) and the top of the Geophysics Department

building in Santiago, Chile (520 MSL), about a 4-km line of sight with an unobstructed view of the hill. A successful transmission of meteorological data was completed including GPS data as shown by the trajectory of the team carrying the sonde as they were climbing up the hill in Fig. A.4. We also performed a second communication test where successful transmission was established from a park near the Andes piedmont and the top of the geophysics building, about a 13-km line of sight.

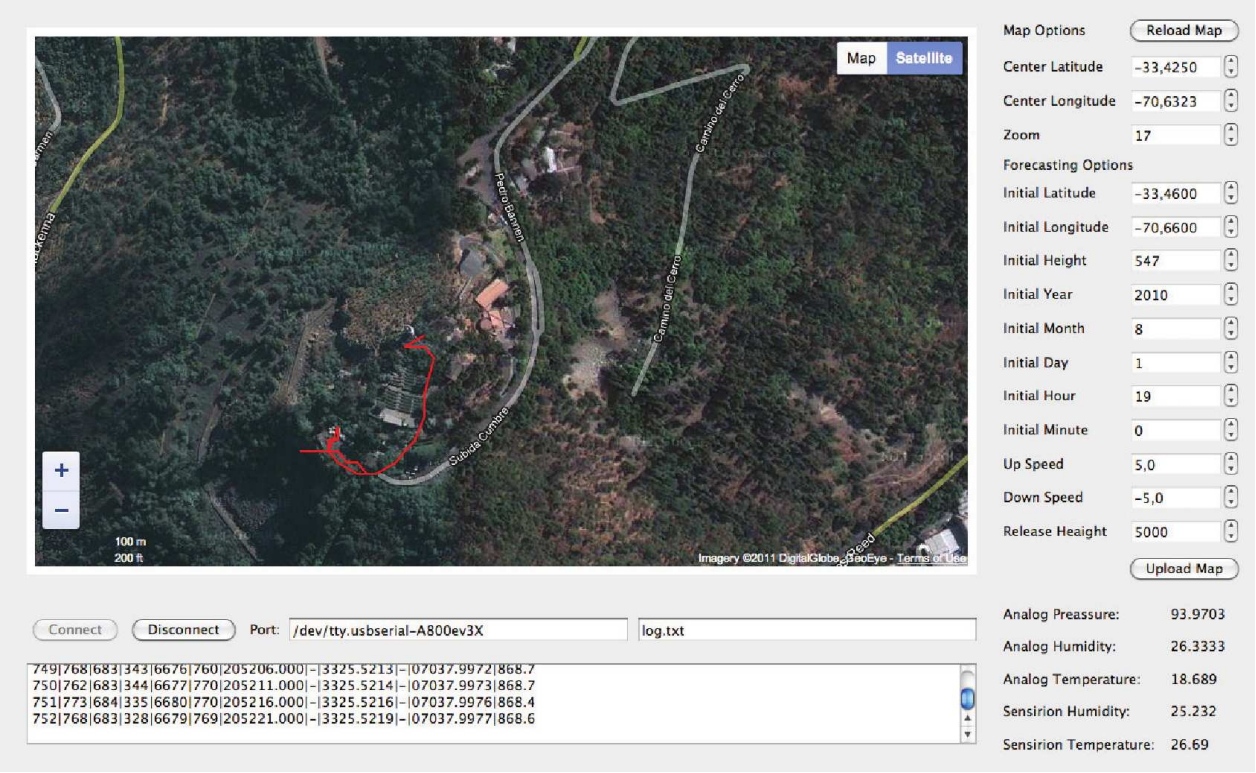


Figure A.4: Screenshot of the BaseCamp software showing the trajectory followed by the students around top of the San Cristobal hill in downtown Santiago. The data were received by students located on the roof of the geophysics building about a 4-km line of sight from the top of the hill.

A.3.2 Methodology for release and recovery

In addition to comparing data measured with our sonde with those obtained with the research-grade radiosondes, we tested the possibility of recovering the equipment. While the notion of recovery was at first a curiosity, it quickly became an important objective of the campaign as the team grew attached to the product of several months of work. Recovery of balloonborne equipment was an essential part of the early attempts to characterize the upper atmosphere. Those balloonsondes only carried registering apparatus and therefore needed to be recovered in order to retrieve the data. For instance, the discovery of the tropopause in 1902 was made using these balloonsondes by Teisserenc de Bort (1902) and Assman (1902) (see, e.g., DuBois et al. 2002).

In order to recover the radiosonde, we took advantage of three features of our sonde, two of them not available to the early balloonists. First, the release system was able to detach

a balloon from the sonde at any given pressure level, upon reaching a certain temperature or elapsed flight time, allowing the sonde to travel a predetermined distance in the vertical. Second, the GPS signals from our sonde and one of the commercial sondes allowed us to accurately know their positions. Third, we made use of a Weather Research Forecasting model (WRF; Skamarock et al. 2008) forecast simulation with a resolution of ∼6 km in the horizontal and 50 eta levels in the vertical, with 10 eta levels located lower than 0.9 with a top pressure of 50 hPa. The model forecast winds were used to determine the sonde trajectory, allowing a recovery team to be ready near the projected landing zone. Reviewing the literature, we learned that the recovery system devised by the students was not new and was originally called the "multiple-balloon technique" reported by Hugo Hergesell in 1904 (Hergesell 1906; DuBois et al. 2002). Our technique and the original one were effectively the same: two balloons would produce the lift necessary for an ascent of ∼5 m s^{-1} of the payload (three sondes weighting in total 1,250 g), and in our case at a given pressure level the linear actuator would release one of the balloons so that the remaining one would carry the payload back to the ground at ∼2 m s^{-1} of vertical descent velocity. In the accounts given by the early balloonsonde researchers, they report high rates of recovery mostly based on the willingness of countryside people to return the equipment in exchange for a reward (see, e.g., Clayton and Fergusson 1909). In our case, evidence from comparing WRF forecast and real sonde trajectories showed that the trajectories were sufficiently close to the real trajectories (especially for the region below 10-km height and within the bounds of the city) as to allow a recovery team to be within the range of the radiosonde signals at ground level (much smaller than the range of the reception of the sonde in flight) and perhaps even observe the descent of the sonde (see Fig. A.5). The multiple-balloon technique has an additional advantage over the most common approach/technique of using a parachute: if the line attaching the payload to the descent balloons is given sufficient length, it will serve as a marker for the position of the radiosonde package. This was successfully tested by Hergesell (1906) over the ocean.

A.3.3 Boundary layer sounding

Our first field experiment was designed to test all sensors in a real flight that included ascent and descent. The flight was launched at 1400 local time 1 October 2011 from Quinta Normal (33.44^{o}S, 70.68^{o}W; 530 MSL) within the city of Santiago. Together with our sonde (which we call FCFM, which is the Spanish acronym for the Faculty of Physical and Mathematical Sciences) we launched two commercially available research-grade radiosondes: a Vaisala RS92 and an InterMet (iMet)-1 radiosonde. The three sondes were attached to a Styrofoam structure. The campaign did not proceed as originally planned because several of the sensors malfunctioned (from the commercial sondes as well as from our own sonde). Three main problems affected the development of the campaign and compromised the recovery of the sonde. First, ascent and descent velocities were computed using a payload weight of 1,250 g. This weight changed during the campaign because of extra batteries and tape being added to attach the sondes together. Therefore, the ascent velocity was smaller than calculated (∼3 instead of 5 m s^{-1}) and the descent velocity was larger than calculated (∼4 instead of 2 m s^{-1}). Second, the release did not occur at 500 hPa (∼5.7 km) because the time condition for release was not adjusted for the small ascent rate of the balloon and because several minutes were lost during the launch procedure. Instead, the radiosonde was released at about 3.5 km

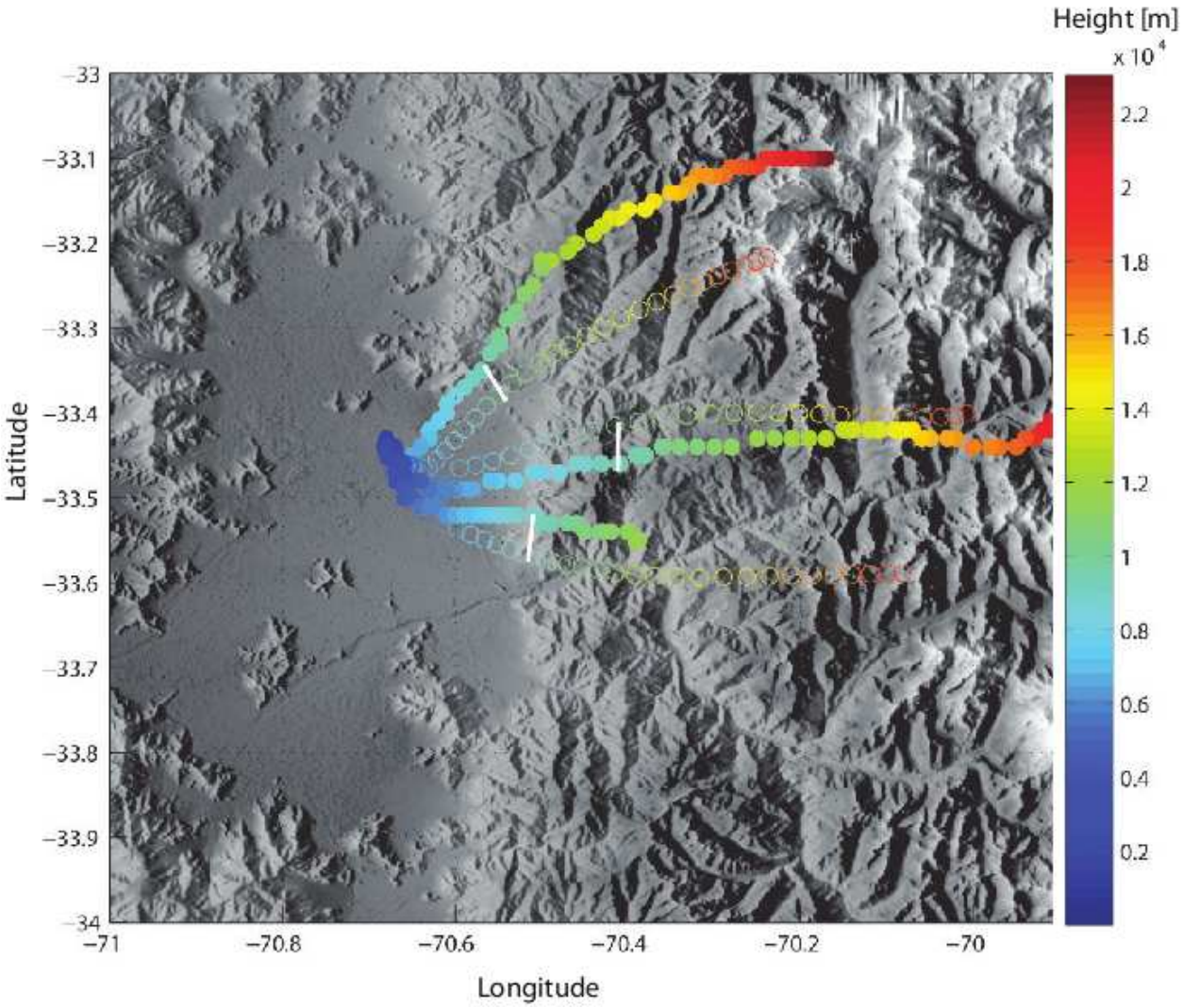


Figure A.5: WRF-calculated trajectories (open circles) and actual observed trajectories (closed circles) for three consecutive days (16, 17, and 18 May 2011 at 1400 local time). The observed trajectories are the trajectories followed by the radiosondes launched from Quinta Normal station in Santiago. The color bar indicates height above Santiago (in meters). Taking as a reference the Andes piedmont, which nearly coincides with the bounds of the city of Santiago, the trajectories show good agreement (within 5 km to the real ones) for the first 10 km in height, even though they show a separation of about 30 km from one day to the next for this particular case. White segments joining the calculated and real sonde trajectories are about 5 km in length. Several trajectories were calculated from WRF output in a similar fashion for dates for which soundings starting at the main office of the Chilean National Weather Service in Santiago were available, and they all showed a similar agreement.

above sea level. Third, we were unable to obtain a signal lock with GPS satellites with the iMet and with our sonde, likely due to interference from a nearby cell phone antenna (the Vaisala sounding system from the Chilean National Weather Service had a dedicated GPS antenna, which may have made the difference). In summary, from the three sondes, we only had a GPS signal during the ascent from the Vaisala sonde. Using the information of the actual release point, we were able to calculate a possible landing point from the WRF wind field. Since the descent velocity was incorrect because of the larger weight of the payload,

the recovery team was given a possible landing point approximately 4 km south of the actual landing point. At that point, the batteries were still powering the sondes, and the team realized that the descent velocity was much larger than estimated. Therefore, they decided to turn on the sonde receivers and attempt to triangulate their signals. By iterating along the path calculated by WRF and observing the strength of the signal from the receptors, they were able to determine a small section of about four blocks (within a populated part of the city) where the sonde should have landed. With no visual evidence of the sonde on the street, the team started asking people about the balloon and on the first of these attempts they found the family that recovered the sonde. Figure A.6 shows the predicted and the actual trajectory followed by the sonde, as well a picture taken by the sonde near the top of the trajectory.

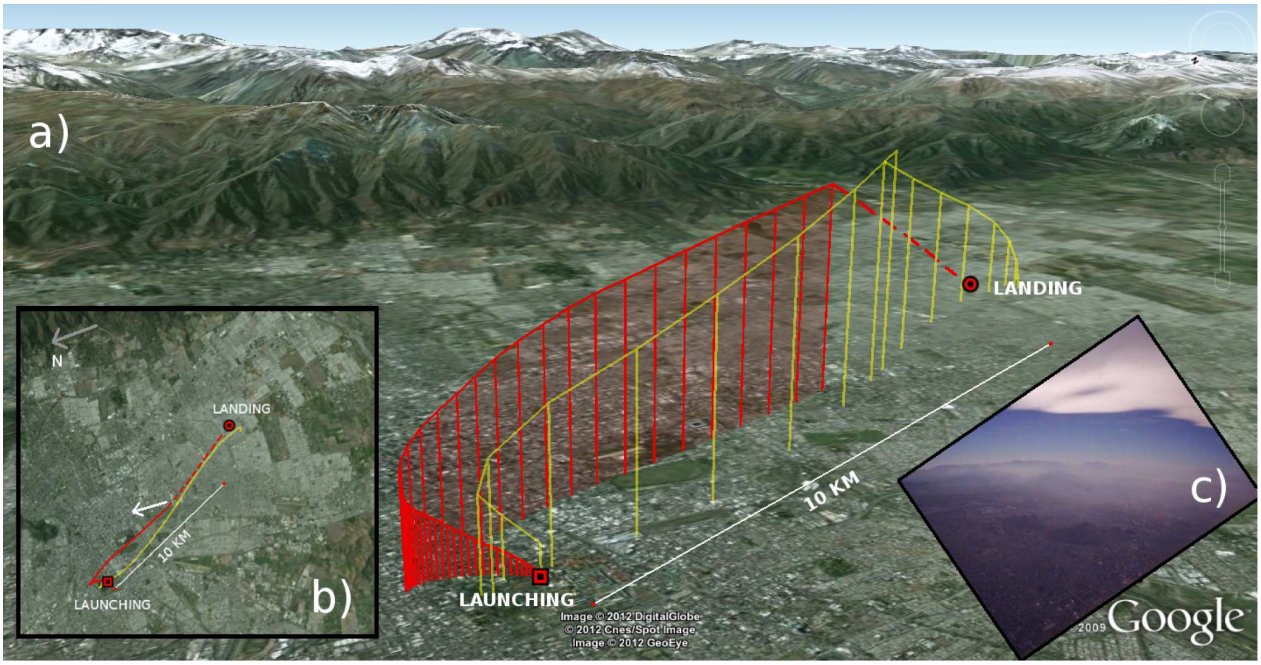


Figure A.6: (a),(b) Aerial view showing the predicted (yellow) and actual trajectory (red) of the radiosonde during the ascent and descent for the Santiago campaign. The missing GPS data of the descent are represented with dashed lines. The forecast trajectory was made a posteriori, knowing the exact ascent and descent velocities. The forecast was made 24 h before the launch. The separation between the end of the forecast trajectory and the actual landing point was about 700 m. These images were modified from Google Earth. (c) A picture taken by the radiosonde near the top of the sounding and looking in the direction marked by the white arrow in (b).

Figure A.7 shows a portion of the data retrieved by the array of sondes. The Vaisala RS92 stopped transmitting data during descent by the default configuration of the Vaisala receiving station. However, the iMet radiosonde was being inadvertently interfered by another iMet radiosonde that was not the one in the array (we realized this at ∼800 hPa into the sounding). The iMet-1 and the Vaisala temperatures show an almost perfect agreement (Fig. A.7a). The temperature from the digital sensor in our sonde suffered from a clear time lag. The time in which the sensors reached the minimum temperature is about 70-80 s longer for our digital sensor than for the research-grade radiosondes. Also, the temperature measured by our digital sensor is clearly overestimated during the ascent and underestimated during descent. This can be in part explained by the relatively large range of response time of the

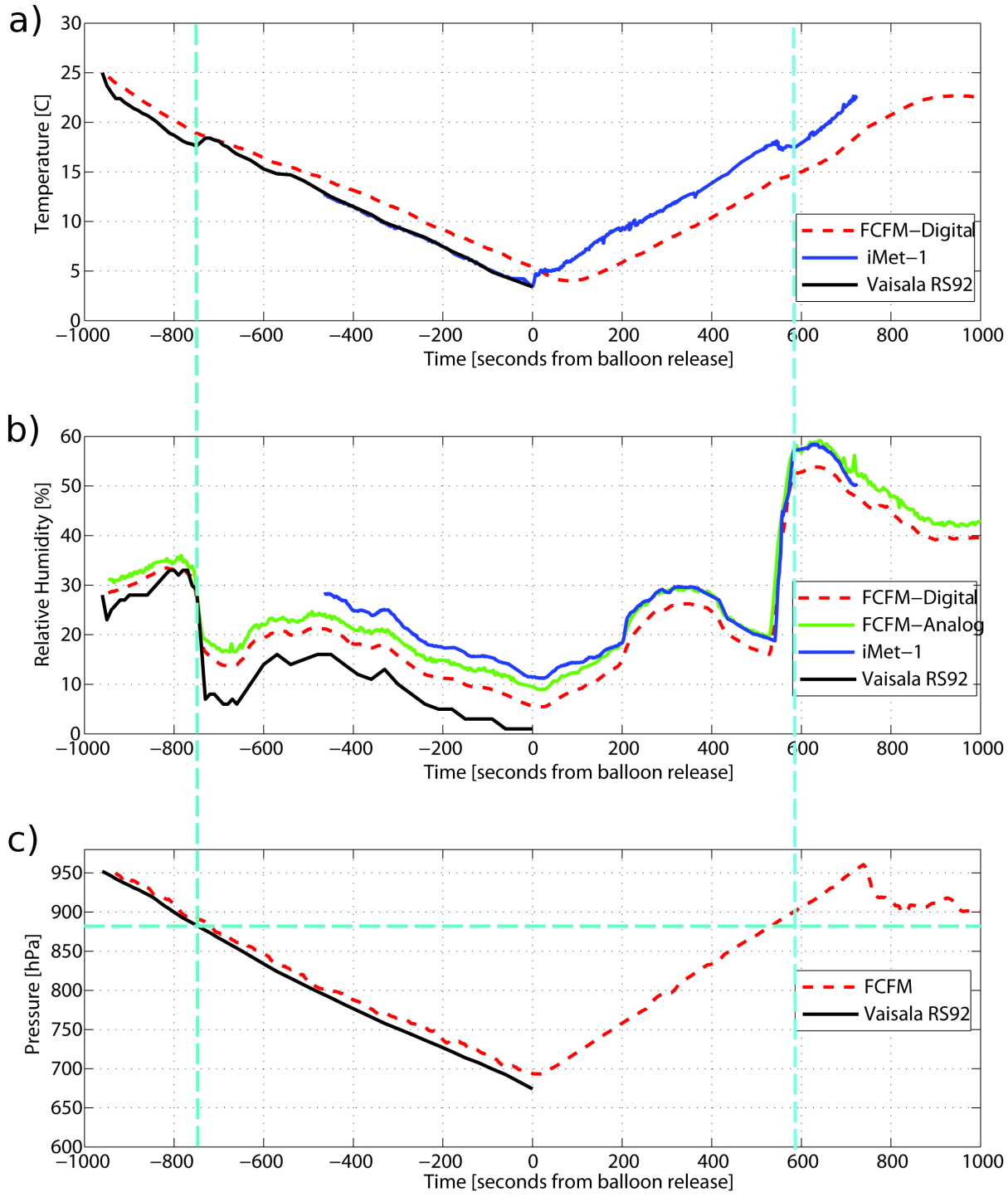


Figure A.7: Time series data of the different meteorological sensors from the three different sondes during the boundary layer intercomparison campaign for (a) temperature, (b) RH, and (c) pressure. The three sondes are referred to as FCFM, Vaisala RS92, and iMet-1. The light blue dashed lines show the approximate time at which the sonde crossed the top of the boundary layer during the ascent and descent. The horizontal light blue line shows the estimated pressure of the top of the boundary layer.

sensor (see Table 1) from 8-30 s compared to the much faster bed thermistors used in most of the research-grade radiosondes (which have a time constant of less than 1 s; Nash et al. 2011). However, most of the effect comes from a deficient design of the casing of the digital temperature/humidity sensors (see Fig. A.2). Although the main purpose of the casing was to protect the sensor from direct sunlight, we inadvertently increased the stagnation of the flow around the sensor, further increasing the effective response time in temperature. This points to the critical role of the casing design in the total error of the measurement (in this case, the casing accounted for several degrees of error in temperature).

The humidity sensors, however, worked reasonably well and took measurements that were in between those of the research-grade radiosonde systems (Fig. A.7b). Vaisala humidity measurements during the ascent from the boundary layer to the free troposphere showed humidity values sharply decreasing from about 32% to less than about 8% at the top of the inversion layer. At the top of the sounding, RH dropped to ∼1% according to the Vaisala measurement, whereas the iMet sonde registered a minimum of nearly 11% and our digital and analogue RH sensors showed ∼6% and ∼9%, respectively. Both our digital and analogue humidity sensors showed the sharp decrease of relative humidity at the top of the boundary layer in both the ascent and the descent. Therefore, even in the absence of the research-grade radiosonde temperature, an independent estimation of the height of the boundary layer was obtained by our sonde. Again, the large response time of our temperature sensor filtered out all signal of the boundary layer top in our temperature profile, which is clearly visible in both the Vaisala and iMet temperature data (Fig. A.7a). There seems to be a difference in the performance of the RH sensors between the ascent and descent, which might be explained by the different velocity of the sonde relative to the air, and therefore different ventilation of the payload. Also, since the digital temperature had a time lag, the RH measured by FCFM-digital, which has a temperature correction, was also affected by the stagnation of the casing.

The pressure data shown in Fig. A.7c are a smoothed version of the raw pressure from the sensor using a moving average window of 10 s. Although a calibration of the pressure was made using equipment provided by the Chilean Meteorological Service, a small bias in pressure is observed, apparently increasing with height. There is a small change in the pressure of the top of the mixed layer between ascent and descent as observed from the horizontal line in Fig. A.7c. This may in be in part due to the pressure bias.

A.3.4 Free tropospheric sounding

A second campaign was carried out at 1400 local time 4 May 2012 to test the radiosonde electronics subject to conditions in the upper troposphere as well as to test communication over a longer range. The sonde was released at Santo Domingo (33.63^{o}S, 71.65^{o}W; 75 MSL), which is one of the four official sounding stations located in Chile, and the closest to Santiago. Figure A.8 shows the comparison among a simultaneous iMet, Vaisala, and FCFM sounding. During this sounding we completely removed the casing covering the digital temperature and relative humidity sensor (FCFM-Sensirion). For this campaign we also added a new analog temperature sensor, which is a thermistor that comes with the most popular Arduino kit distribution, the "Arduino Starter Kit" (see Table 2).

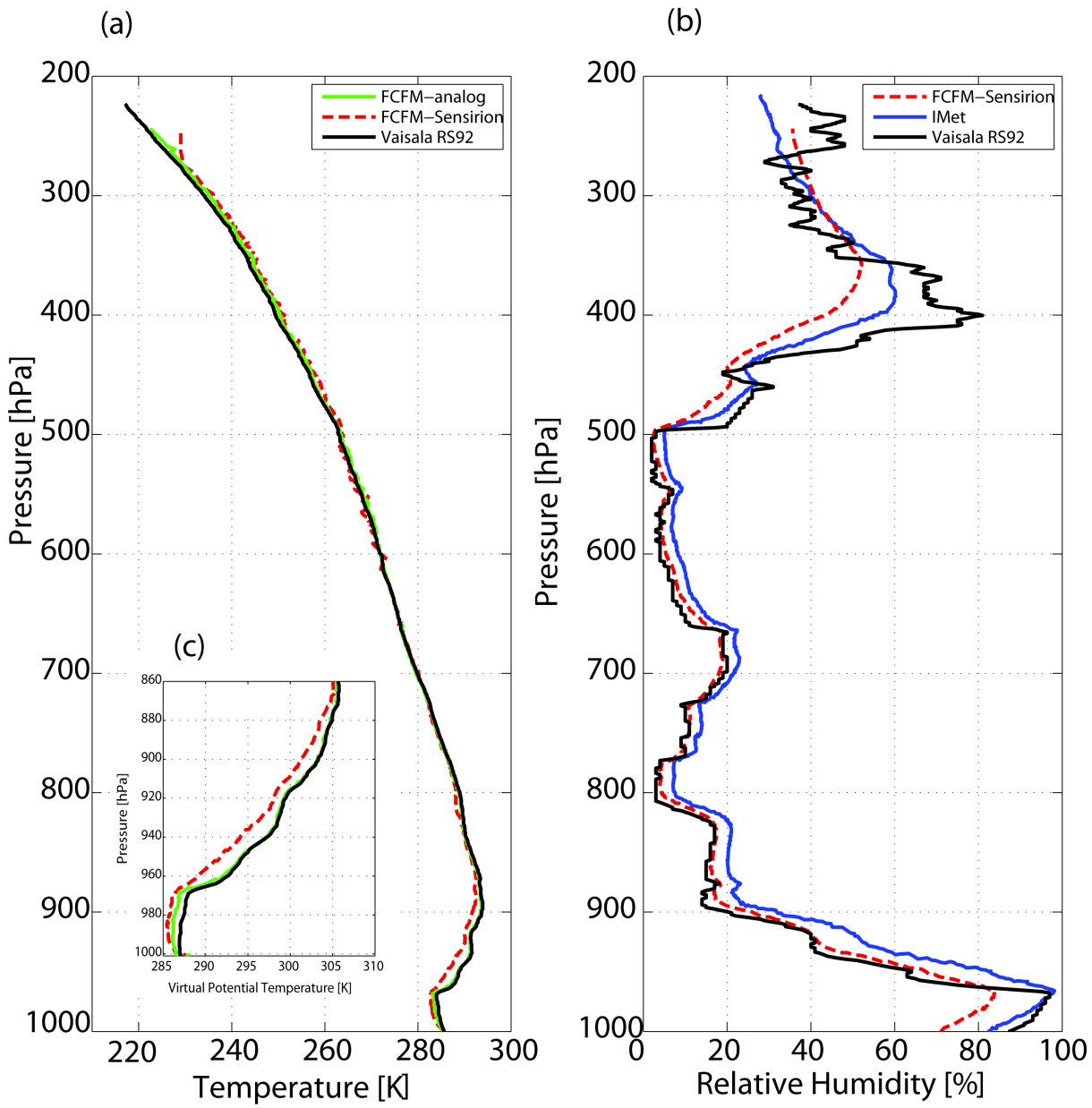


Figure A.8: Ascent vertical profiles for the free troposphere intercomparison campaign of 4 May 2012: (a) temperature in Kelvins as a function of pressure, (b) RH, and (c) a view of the boundary layer virtual potential temperature.

Unfortunately, because of a bad design of the payload, we had interference between the transmitters and the GPS receivers, so no signal from GPS was available from any of the sondes. Individually, the sondes each obtained a GPS signal lock and were transmitting geolocation data to their respective receivers. The lack of geolocation data during flight further resulted in the loss of payload after descending from about 12 km (∼220 hPa).

Despite the loss of the payload, the radiosonde transmitted good data even during the descent. Here, we report observations obtained from the ascent part of the sounding. There is a well-mixed boundary layer between the surface and 970 hPa (almost constant virtual

potential temperature in the layer; Fig. A.8c). Stratocumulus was present at the top of the mixed layer. There is a relatively deep temperature inversion layer from 970 to about 900 hPa. Even without the casing, the FCFM-Sensirion time lag is long enough to smooth out the rapid change at the top of the mixed boundary layer. Figure A.8c shows more clearly the lag in temperature by the FCFM-Sensirion; nevertheless, the pressure level of the top of the mixed boundary late could be clearly identified from our sensors. The performance of the analogue temperature sensor was surprisingly good, following very closely the behavior of the Vaisala sounding. The mean absolute error of the temperature was about 0.85 K for the FCFM-Sensirion (mostly due to the large error in temperature within the inversion layer) and only 0.4 K for the FCMF-analog. The FCFM-Sensirion exceeded its operating range and transmitted a fixed temperature after reaching 229 K.

Both the iMet-1 and Vaisala RS92 show relative humidities near 100% at the top of the mixed layer. Our sensor, however, shows an ∼15% dry bias within the mixed boundary layer (Fig. A.8b). The inversion at the top of the well-mixed boundary layer coincides with a rapid drying from near saturation at the top of the mixed layer to about 20% RH at the top of the inversion layer (about 900 hPa). The performance of our sensor in RH improves for drier conditions compared to Vaisala, consistent with the performance during the first campaign. Near 400 hPa, there was an ice cloud that is apparently detected by the Vaisala sounding as a thin layer of ∼100% RH with respect to ice. In this case, both iMet and in particular our sonde have trouble responding to the presence of the cloud and therefore show a much smoother curve (the Vaisala sounding shows a rich structure in humidity), giving also a lower value for the pressure level of the cloud. The slow time response of both iMet and FCFM-Sensirion is clearly seen in the increase of RH above 500 hPa and also in the increase of RH from about 450 to 400 hPa which is the level of the cloud (Fig. A.8).

Although we did not recover this first prototype of our radiosonde, the campaign was successful in that it provided us with a wider range of atmospheric conditions to compare and to test the performance of the sensors and the electronics. Also, we were able to receive the signal of the radiosonde over a much longer path than previously tested (about 30 km in a direct line of sight, as calculated from the balloon trajectory inferred from a WRF simulation).

A.4 Did engineering students learn meteorology by building and testing a radiosonde?

The main purpose of our original class was to allow students to conceive, design, implement, and operate a radiosonde. The project had an explicit engineering purpose that we believe was largely fulfilled by most of the students. However, we also expected that as an unintended consequence of building a radiosonde, the students would naturally familiarize themselves with atmospheric properties and processes. From conversations with the students at the end of the semester and from the presentations and technical reports that they handed in, we can categorize the students broadly into three equally populated groups that also reflect the students' interests and depth of learning.

- *Students did not learn any meteorology.* By the design of the class work, students needed to specialize in one of three tasks. The group in charge of the radiosonde communication was only barely in touch with the actual atmosphere we were planning to observe. Some of the best students in the communication group, however, were not in this category.
- *Students that took meteorological observations only as a design constraint on the instruments.* At this level of learning, students were able to take into consideration the range of variables observed in the real atmosphere for the selection of the sensors but did not go much further.
- *Students that took meteorological phenomena into account for the choice of sensors and the design of the radiosonde.* These students revised the design several times in order to comply with the need to observe a particular process. For instance, in order to distinguish rapid changes in the atmospheric variables with height, some students realized that there is a trade-off between the ascent rate of the balloons (ν_{asc}) and the time response of the sensors; by simply reducing the ascent rate of the balloons, one is also reducing the ventilation of the sensors and therefore increasing the time response. This is an important issue in the local Santiago climate where the height of the mixed boundary layer is a critical parameter for the dispersion of pollutants (e.g., Muñoz and Undurraga 2010). If the depth of the inversion layer is Δh, the response time of the sensor in temperature needs to be at least $\Delta h/2\nu_{asc}$ 2 in order to resolve the inversion layer (this can be similarly applied to the detection of a cloud layer from a RH sounding). Solving this type of problem requires interaction between meteorologists and engineers. Other students that fall into this category were interested in knowing the height of the balloon during the trajectory. The height of the balloon was critical to our exercise as it allows for determining whether the ascent and descent rates during the campaign are the ones estimated before the launching. This permits in situ corrections of the radiosonde trajectory calculated from WRF. The need for the height of the balloons led students to study the concept of geopotential height and the hydrostatic and hypsometric equations. The idea of recovering the payload was also a motivation for some students to become interested in numerical weather forecasting models, the calculation of trajectories, and the local meteorology. Several of these students have continued by enrolling in introductory classes in meteorology.

Reflecting on the motivation and performance of the students during the semester, perhaps instead of dividing the groups into particular variables to measure parts of the radiosonde to build, it might be better to assign them to particular phenomena to observe. This would put the students in to a closer relationship with the actual purpose of the instrument and motivate them to come back to the design of the instrument if the data are not satisfactory, as opposed to students thinking that the class is over when the first prototype is ready.

A.5 Other sensors and further development

The lessons learned from the campaigns have led to the development of many improvements to our radiosonde system. Our first two full campaigns were successful in accomplishing some of

the main goals that were initially proposed to the students, namely, the transmission of data during flight from most of the depth of the troposphere, the ability to recover the radiosonde, the functioning of the electronics subject to very cold temperatures, and the ability to record and transmit data that compared reasonably well with the operational standards.

Since the end of the semester, some students and most of the teaching assistants have continued working on different aspects of the FCFM radiosonde project. In order to improve the measurement of humidity, we are currently developing a chilled-mirror hygrometer using Peltier cooling elements in part based on the one described by Vömel et al. (2003). Control routines to sequentially cool and warm the Peltier elements can be easily programmed to the Arduino. Also, we are attempting the integration of our system with an ozonesonde, which, being an expensive instrument to begin with ($\sim$ \$1,000 per ozonesonde), makes the possibility of the recovery of the ozonesonde payload much more attractive. Companies that produce electrochemical ozonesondes have designed their instruments to communicate with research-grade radiosondes. Ozonesondes can be reused provided that the reactants are replaced and the sonde is recalibrated.

Equally important for the continuation of the project is to improve the quality of the data obtained from the commercially available sensors, attempting to get as close as possible to the research-grade accuracy and response time in all meteorological variables. Although our comparisons are relative to Vaisala RS92 and iMet-1 radiosondes, these instruments also suffer from biases and random errors [see, e.g., Miloshevich et al. (2009) for a discussion of the errors in the relative humidity measurements made by the Vaisala RS92 sonde]. Improvements in the data come not only from a better instrumentation but also from software corrections due to the time lag of the sensors and possibly improvements in the layout of the instruments in the payload. Attempting to solve some of these challenges makes for interesting ideas for future instances of the workshop class.

If one can prove that radiosondes have errors that are smaller than operational for a significantly large number of soundings, then the idea of adaptive sounding as proposed by Douglas (2010) becomes feasible using our system, given the relatively low cost of implementation of a ground receiving station (the limiting issue then becoming the availability of gas supply). Although the cost of our radiosonde turned out to be on the order of the current commercial radiosondes, the receiver (ground station) is several orders of magnitude cheaper than commercial solutions. In addition to this, the cost of the radiosonde can be of less relevance if we notice two extra advantages of developing a reusable radiosonde, especially as an open software/hardware initiative: 1) open initiatives tend to be developed faster than closed ones because of the community cooperation and 2) specific needs (e.g., radio observatories needing very accurate humidity measurements in the upper troposphere to perform interferometry observations) can be addressed by adding new custom-made sensors in a much easier and more flexible manner.

Other improvements to the system involve the visualization and trajectory software to allow for the real-time correction of the forecast trajectory of the radiosonde to speed up the recovery of the radiosonde package. Also, the Arduino board can be programmed so that in the case of encountering interference at the ground or during the trajectory, the GPS module can be reinitialized and the GPS signal lock can be recovered. Nevertheless, the successful

recovery of the payload requires not only having the technology in place, but also a strict set of protocols to be followed by the people participating in the campaign (essential for mission planning engineers).

From a pedagogical point of view, we believe that this exercise is beneficial for both meteorologists and engineers. Meteorologists get a first-hand sense of the data they analyze and study; they recognize the source of errors by looking at amplified versions of the errors relative to what they will encounter in the commercial radiosondes. Engineers, however, learn meteorology (even sometimes inadvertently) by attempting to solve seemingly practical problems. In retrospect, the greatest change we would make to future instances of the class would be to focus the main objective of the class from the beginning on measuring particular meteorological phenomena rather on the construction and integration of the parts and pieces of the equipment.

We have developed a website (http://www.dgf.uchile.cl/radiosonde) that provides the details of the construction, parts, and programming involved in the radiosonde, so we hope others will attempt the construction of the radiosonde either as a part of the regular curriculum of engineering and meteorology or simply due to enthusiasm. If a community of users and developers flourishes, we hope that many unexpected improvements of the platform that we present here will ensue.

A.6 Acknowledgments

We appreciate the help of the Chilean National Weather Service [Dirección Meteorológica de Chile (DMC)] for providing access to an environmental chamber to make the calibration of our sensors, and also for providing the ground stations for retrieving the Vaisala measurements. We appreciate the comments and suggestions by two anonymous reviewers and by Lodovica Illari that significantly improved the original manuscript. We also appreciate the comments by Dr. Ricardo Muñoz and Dr. Angel Otárola. R. Rondanelli and R. Querel acknowledge funding from Conicyt through Fondecyt grants 1110393 and 3120150. F. Flores' efforts were supported by the Conicyt doctoral program. M. Diaz acknowledges support from the Department of Electrical Engineering.

REFERENCES

Assman, R., 1902: Uber die Existenz eines wärmeren luftstromes in der Höbe von 10 bis 15km. Sitzber. Konigl. Preuss. Akad. Wiss. Berlin, 24, 495-504.

Clayton, H. H. and S. P. Fergusson, 1909: Exploration of the upper air by means of balloonssondes, at St. Louis, in the years 1904 - 07. Annals of Harvard College Observatory, 68, 1-84

Crawley, E., J. Malmqvist, S. Östlund, and D. Brodeur, 2007: Rethinking Engineering Education: The CDIO Approach, Vol. 133. Springer Verlag.

Dabberdt, W., H. Cole, A. Paukkunen, J. Horhammer, V. Antikainen, and R. Shellhorn, 2002: Radiosondes, Vol. 6, 1900-1913. Academic Press, San Diego.

Douglas, M., 2010: Adaptive sounding arrays for tropical regions. 29th Conference on Hurricanes and Tropical Meteorology, American Meteorological Society.

DuBois, J., R. Multhauf, and C. Ziegler, 2002: The Invention and Development of the Radiosonde with a Catalog of Upper-Atmospheric Telemetering Probes in the National Museum of American History, Smithsonian Institution. No. 53 in Smithsonian studies in history and technology, Smithsonian Institution Press, Washington, D.C.

Hergesell, H., 1906: Ballon-Aufstiege über dem freien Meere zur Erfassung der Temperatur- und Feuchtigkeitsverhältnisse sowie der Luftströmungen bis zu sehr grossen Höhen der Atmosphäre. Beitr. Phys. Atmos., 1, 200-204.

Immler, F., J. Dykema, T. Gardiner, D. Whiteman, P. Thorne, and H. Vömel, 2010: Reference Quality Upper-Air Measurements: Guidance for developing GRUAN data products. Atmospheric Measurement Techniques, 3, 1217-1231.

Muñoz, R. and A. Undurraga, 2010: Daytime mixed layer over the Santiago basin: Description of two years of observations with a lidar ceilometer. Journal of Applied Meteorology and Climatology, 49, 1728-1741.

Nash, J., T. Oakley, H. Vömel, and L. Wei, 2011: WMO intercomparison of high quality radiosonde systems, Yangjian China, 12th July to 3rd August. Tech. rep., World Meteorological Organization.

Pettifer, R., 2009: From observations to forecasts. Part 2. The development of in situ upper air measurements. Weather, 64 (11), 302-308.

Sarik, J. and I. Kymissis, 2010: Lab kits using the Arduino prototyping platform. 40th ASEE/IEEE Frontiers in Education Conference, IEEE. T3C-1.

Skamarock, W., J. Klemp, J. Dudhia, D. Gill, D. Barker, W. Wang, and J. Powers, 2008: A Description of the Advanced Research WRF Version 3. Tech. Rep., NCAR. NCAR Tech Notes-475+ STR.

Teisserenc de Bort, L., 1902: Variations de la température de l'air libre dans la zone comprise entre 8km et 13km d'altitude. Comptes Rendus de l'Acad. Sci. Paris, 134 (987.989).

Thorne, P., J. Lanzante, T. Peterson, D. Seidel, and K. Shine, 2011: Tropospheric temperature trends: History of an ongoing controversy. Wiley Interdisciplinary Reviews: Climate Change, 2 (1), 66-88.

Turner, D. D., E. J. Mlawer, G. Bianchini, M. P. Cadeddu, S. Crewell, J. S. Delamere, R. O. Knuteson, G. Maschwitz, M. Mlynzcak, S. Paine, L. Palchetti, and D. C. Tobin, 2012: Ground-based high spectral resolution observations of the entire terrestrial spectrum under extremely dry conditions. Geophys. Res. Lett., 39, L10801, 1-5.

Vömel, H., M. Fujiwara, M. Shiotani, F. Hasebe, S. Oltmans, and J. Barnes, 2003: The behavior of the snow white chilled-mirror hygrometer in extremely dry conditions. Journal of Atmospheric and Oceanic Technology, 20 (11), 1560-1567.

Appendix B

Desarrollo del Modelo

Si bien *OpenFOAM* provee una serie de solvers disponibles, también brinda la posibilidad de desarrollar códigos que respondan a las características específicas de cada problema. En este caso, dada nuestra necesidad de incorporar diferentes procesos físicos, condiciones de borde e iniciales en el modelo, optamos por desarrollar un solver apropiado para el problema. El código desarrollado parte de la base de un solver ya existente en *OpenFOAM* 1.7, *buoyantPimpleFoam*, como es recomendable en la mayoría de los casos en que necesitemos configurar un modelo a medida. Su desarrollo permitió conocer en detalle cómo funciona la estructura del código, cómo incorporar ecuaciones gobernantes y qué simplificaciones se realizan durante el cálculo.

A continuación realizamos una extensa descripción del modelo, los pasos necesarios para su desarrollo, y la forma en cómo se estructuran los códigos que ejecuta *OpenFOAM*. Esta descripción es válida para cualquier código que se pretenda desarrollar en *OpenFOAM* y es además una buena guía para futuras modificaciones que se pretendan hacer al modelo, ya que da luces sobre las poderosas herramientas que brinda el esquema estructurado de programación con el que trabaja esta herramienta de simulación CFD.

Hemos divido este apéndice en dos secciones. La primera, *Programando en OpenFOAM*, contiene las herramientas básicas que se utilizan al generar el código de un solver específico en la plataforma. La segunda, *Solver*, contiene un resumen de las ecuaciones y librerías utilizadas en el desarrollo del modelo usado en este trabajo. En ambos casos hemos incluido líneas de código a manera de referencia, señalándolas mediante recuadros de color gris.

B.1 Programando en *OpenFOAM*

Dado que *OpenFOAM* está diseñado para resolver ecuaciones diferenciales en medios continuos, posee una serie de herramientas disponibles que es necesario conocer antes de utilizarlo en un problema particular. Si bien el conjunto de funciones y herramientas disponibles es demasiado extenso para presentarlo en este trabajo, en las próximas páginas realizaremos una breve descripción de algunas de las funciones que tendremos que utilizar para el desarrollo del modelo. Para una descripción más detallada, referirse al manual del programador (OpenCFD (2009)) o a parte de la extensa descripción disponible en Internet (tanto en el foro de usuarios como en el sitio del código).

B.1.1 Entorno de cálculo

Si bien *OpenFOAM* usa como base de programación el lenguaje C++, y es posible usar directamente las funciones que provee este lenguaje o desarrollar funciones más complejas en base a ellas, *OpenFOAM* es también en sí mismo un lenguaje de alto nivel, proveyendo funciones específicas para el tratamiento de ecuaciones diferenciales. Estas funciones especiales se podrán utilizar en cualquier solver que sea compilado mediante las herramientas provistas por *OpenFOAM*, y se integrarán directamente con el cálculo y solución de las ecuaciones mediante volúmenes finitos. Una breve reseña de algunas de las funciones más importantes que usaremos a lo largo del desarrollo del modelo están expuestas en la tabla B.1.

Dado que *OpenFOAM* es capaz de trabajar con tensores de diferentes rangos (hasta rango 3), cada una de estas funciones se podrá aplicar, según corresponda, a escalares (tensor de rango 0), vectores (rango 1), tensores típicos (rango 2) o tensores de rango 3.

En particular, en el código es posible definir un tensor a través del comando

```
tensor T(1, 2, 3, 4, 5, 6, 7, 8, 9);
```

que equivaldrá a

$$T = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} , \tag{B.1}$$

siendo lo mismo aplicable para escalares y vectores.

Para acceder a la información contenida en los tensores se usan las funciones de acceso descritas en la tabla B.2. Por ejemplo, para acceder a la componente T_{13} usaremos el comando `T.xz()`.

La resolución de sistemas de ecuaciones diferenciales se ve simplificada en *OpenFOAM* dado que el software incorpora una serie de operadores diferenciales disponibles para ser usados en los códigos desarrollados. Algunos de estos operadores se presentan en la tabla B.3. Estas herramientas confirman la condición de "lenguaje de alto nivel" del código, en el sentido de que cada una de estas funciones se puede integrar en el desarrollo de un solver e interac-

Operación	Matemática	*OpenFOAM*
Suma	$a + b$	`a + b`
Resta	$a - b$	`a - b`
Multiplicación escalar	sa	`s * a`
División escalar	a/s	`a / s`
Producto externo	ab	`a * b`
Producto interno	$a \cdot b$	`a & b`
Producto cruz	$a \times b$	`a ^ b`
Cuadrado	a^2	`sqr(a)`
Magnitud al cuadrado	$\|a\|^2$	`magSqr(a)`
Magnitud	$\|a\|$	`mag(a)`
Potencia	a^n	`pow(a,n)`
Promedio de componentes	$\overline{a_\mathrm{i}}$	`cmptAv(a)`
Máximo de los componentes	$max(a_\mathrm{i})$	`max(a)`
Mínimo de los componentes	$min(a_\mathrm{i})$	`min(a)`
Transpuesta	$\mathbf{T}^T$	`T.T()`
Diagonal	$diag\mathbf{T}$	`diag(T)`
Traza	$tr\mathbf{T}$	`tr(T)`
Componente deviatórico	$dev\mathbf{T}$	`dev(T)`
Determinante	$det\mathbf{T}$	`det(T)`
Inverso	$inv\mathbf{T}$	`inv(T)`
Raíz	$\sqrt{(s)}$	`sqrt(s)`
Exponencial	e^s	`exp(s)`
Logaritmo natural	$ln(s)$	`log(s)`
Logaritmo base 10	$log_{10}(s)$	`log10(s)`
Componente trigonométrica	$sin(s)$	`sin(s)`

Table B.1: Algunos de los operadores matemáticos disponibles en *OpenFOAM* (OpenCFD (2009)).

Rango	Nombre común	Clase	Función de acceso
0	escalar	`scalar`	
1	vector	`vector`	`x(), y(), z()`
2	tensor	`tensor`	`xx(), xy(), xz() ...`
3	tensor	`tensor`	`xxx(), xxy(), xxz() ...`

Table B.2: Tensores disponibles en *OpenFOAM* (OpenCFD (2009)).

tuará con el resto de funciones provistas por *OpenFOAM*, usando además el tratamiento en volúmenes finitos definido por la estructura del código.

De esta forma es posible lograr una representación sencilla de las ecuaciones al desarrollar un solver. Por ejemplo, la ecuación

$$\frac{\partial \rho U}{\partial t} + \nabla \cdot \phi U - \nabla \cdot \mu \nabla U = -\nabla p \,, \tag{B.2}$$

se puede representar mediante el código

Operador	**Implícito/Explícito**	**Expresión**	**Función** (`fvm::/fvc::`)
Laplaciano	Imp/Exp	$\nabla^2\phi$ $\nabla \cdot \Gamma\nabla\phi$	`laplacian(phi)` `laplacian(Gamma, phi)`
Derivada temporal	Imp/Exp	$\frac{\partial\phi}{\partial t}$ $\frac{\partial\rho\phi}{\partial t}$	`ddt(phi)` `ddt(rho,phi)`
Segunda derivada temp.	Imp/Exp	$\frac{\partial}{\partial t}\left(\rho\frac{\partial\phi}{\partial t}\right)$	`d2dt2(rho, phi)`

Convección	Imp/Exp	$\nabla \cdot (\psi)$
		$\nabla \cdot (\psi\phi)$
			`div(psi,esquema)`
			`div(psi,phi,word)`
			`div(psi,phi)`

Divergencia	Exp	$\nabla \cdot \chi$	`div(chi)`

Gradiente	Exp	$\nabla\chi$
		$\nabla\phi$
			`grad(chi)`
			`gGrad(phi)`
			`lsGrad(phi)`
			`snGrad(phi)`
			`snGradCorrection(phi)`

Grad-grad al cuadrado	Exp	$\lvert\nabla\nabla\phi\rvert^2$	`sqrGradGrad(phi)`
Rotor	Exp	$\nabla \times \phi$	`curl(phi)`
Fuente	Imp	$\rho\phi$	`Sp(rho,phi)`

Table B.3: Algunos operadores diferenciales disponibles en *OpenFOAM* (OpenCFD (2009)).

```
solve
(
fvm::ddt(rho, U)
+ fvm::div(phi, U)
- fvm::laplacian(mu, U)
==
- fvc::grad(p)
);
```

donde los términos `fvm::` y `fvc::` serán discutidos más adelante.

Para acceder a la geometría del modelo existen funciones disponibles que hacen referencia específica a alguna característica geométrica. En particular, al programar usando *OpenFOAM* será necesario, cada vez que definamos una variable, definir cuál es su contenido espacial, es decir si se trata de un campo vectorial, escalar, o está asociado a la superficie entre las celdas. Para acceder a la información que el programa almacena asociada a la malla de cada problema existen funciones específicas, algunas de las cuales se detallan en la tabla B.4. Por medio de estas funciones se puede acceder a la información específica contenida en cada celda del dominio.

Otro punto a tener en cuenta durante la programación es el hecho de que *OpenFOAM* calcula manteniendo un estricto balance de unidades en las ecuaciones, por lo que es necesario definir las unidades correctas de cada variable considerada. Si bien esto puede resultar en muchas ocasiones engorroso, en particular al incluir constantes en el cálculo, hace más

Descripción	Clase	Símbolo	Función de acceso
volumen de cada *Celda*	`volScalarField`	V	`V()`
vector de área de una *Cara*	`surfaceVectorField`	S_f	`Sf()`
área de una *Cara*	`surfaceScalarField`	$\|S_f\|$	`magSf()`
centro de una *Celda*	`volVectorField`	C	`C()`
centro de una *Cara*	`surfaceVectorField`	C_f	`Cf()`
flujos en una *Cara*	`surfaceScalarField`	ϕ_g	`phi()`

Table B.4: Acceso a la información en los campos usados en *OpenFOAM* (OpenCFD (2009)).

rigurosa la implementación de las ecuaciones, y permite además asegurar la integridad de éstas. Por ejemplo, para definir una variable escalar dimensional, como la contante de Stefan-Boltzman, usaremos la estructura:

```
dimensionedScalar sigma
(
"sigma",
dimensionSet(1, 0, -3, -4, 0, 0, 0),
scalar(5.67E-8),
);
```

en donde `dimensionSet(1, 0, -3, -4, 0, 0, 0)`, define las unidades kg s^{-3} K^{-4}, equivalentes a $W/(m^2K^4)$, usando la notación descrita en la tabla B.5.

Número	Propiedad	Unidad	Símbolo
1	masa	kilógramo	k
2	distancia	metro	m
3	tiempo	segundo	s
4	temperatura	Kelvin	K
5	cantidad	moles	mol
6	corriente	ampere	A
7	intensidad luminosa	candela	cd

Table B.5: Unidades S.I. usadas en *OpenFOAM* (OpenCFD (2009)).

B.1.2 Discretización y métodos numéricos

Tal como lo expresa el manual de programación de *OpenFOAM* (OpenCFD (2009)), el término discretización se refiere a *aproximar un problema mediante cantidades discretas*. Para hacerlo, tanto en *OpenFOAM* como en la mayoría de los códigos numéricos se recurre a tres tipos de discretizaciones:

- Discretización espacial, dividiendo el dominio espacial en un conjunto de puntos de cálculo.
- Discretización temporal, dividiendo el tiempo en un número finito de intervalos.

- Discretización de las ecuaciones, generando representaciones algebraicas de las ecuaciones diferenciales que definen el problema.

Dado que la discretización espacial es realizada en forma casi automática por el código, usando técnicas clásicas del método de volúmenes finitos, y está asociada fundamentalmente a la generación de la malla, nos dedicaremos a continuación a explicar únicamente cómo se realiza la discretización temporal y de las ecuaciones en *OpenFOAM*. Comprender esta discretización es fundamental para el desarrollo de modelos, pues permitirá entender cómo trabaja el código integrando las funciones diferenciales definidas anteriormente, y qué métodos numéricos son aplicables al cálculo. Por motivos de claridad iniciaremos la descripción refiriéndonos a la discretización de las ecuaciones.

B.1.2.1 Discretización de las ecuaciones

El objetivo final de la discretización de ecuaciones es representar el sistema de ecuaciones diferenciales a través de un conjunto de relaciones algebraicas representadas matricialmente como

$$[A]\,[x] = [b]\ , \tag{B.3}$$

donde $[A]$ es una matriz cuadrada, $[x]$ es el vector columna que contiene las variables dependientes, y $[b]$ representa los términos fuente existentes en el sistema. Si bien los vectores $[x]$ y $[b]$ son sólo un conjunto de valores asociados a cada punto de cálculo dentro de la malla, y pueden ser descritos por la clase `volField` que describe un campo de vectores, la definición de la matriz $[A]$ es un poco más compleja, y está asociada directamente a las relaciones matemáticas establecidas por la ecuaciones que dominan el problema. Por eso en *OpenFOAM* se le asigna a ella una clase especial llamada `fvMatrix`.

Esta distinción entre las clases `volField` y `fvMatrix` es importante en *OpenFOAM*, y determinará la aplicación de los operadores definidos en la tabla B.3, diferenciando entre dos tipos de cálculo, definidos por las clases `finiteVolumeCalculus` y `finiteVolumeMethod`. Cuando apliquemos un operador a un campo vectorial, como $[x]$ o $[b]$, usaremos el indicador `fvc::` antecediendo el operador para indicar que se trata de una operación del tipo `finiteVolumeCalculus`, que calcula explícitamente y genera un campo vectorial de salida. Si aplicamos el mismo operador a una matriz que no puede ser descrita únicamente como un campo vectorial, como la matriz $[A]$, usaremos el indicador `fvm::` para señalar que el cálculo será del tipo `finiteVolumeMethod`, es decir implícito, y generará como salida otra matriz del tipo `fvMatrix`. Esta distinción es a la que hacen referencia los términos explícito e implícito de la tabla B.3.

Una vez aplicado uno de los operadores de dicha tabla, y hecha la distinción entre `fvm::` y `fvc::`, la discretización de cada término se realizará integrando primero el término en un volumen de control V. A continuación la mayoría de los términos derivativos espaciales son convertidos en integrales sobre la superficie de la celda de cálculo mediante el teorema de Gauss:

$$\int_V \nabla \star \phi \,\mathrm{d}V = \int_S \mathrm{d}S \star \phi\ , \tag{B.4}$$

donde S es el vector de área superfical, ϕ representa a algún tensor, y la $\star$ se refiere a algún operador tensorial. Las integrales son a continuación linealizadas usando alguno de los esquemas disponibles para este efecto. Los esquemas disponibles van a depender del tipo de término considerado, y su elección puede ser hecha por el usuario a través del archivo `fvSchemes`, contenido en la carpeta `system`. Es importante considerar que cada vez que incluyamos una ecuación al modelo deberemos necesariamente incluir en el archivo `fvSchemes` el esquema de discretización que usaremos para los operadores que incluyamos en la ecuación.

Una breve descripción de la linealización de algunos de los términos se describe a continuación, a manera de ilustración. Para una descripción más extensa referirse al manual del programador (OpenCFD (2009)).

La **primera derivada temporal** se integra en el volumen de control usando:

$$\frac{\partial}{\partial t}\int_V \rho\phi \,\mathrm{d}V \ , \tag{B.5}$$

y luego se discretiza usando uno de los dos esquemas a continuación:

- *Euler implícito* (`EulerImplicit` en `fvSchemes`), que tiene una precisión de primer orden:

$$\frac{\partial}{\partial t}\int_V \rho\phi \,\mathrm{d}V = \frac{(\rho_P\phi_P V)^n - (\rho_P\phi_P V)^o}{\Delta t} \ , \tag{B.6}$$

- *Paso atrás* (`BackwardDifferencing` en `fvSchemes`), que tiene una precisión de segundo orden y considera más tiempos en el cálculo

$$\frac{\partial}{\partial t}\int_V \rho\phi \,\mathrm{d}V = \frac{3\,(\rho_P\phi_P V)^n - 4\,(\rho_P\phi_P V)^o + (\rho_P\phi_P V)^{oo}}{2\Delta t} \ , \tag{B.7}$$

donde en ambos casos el subíndice $_P$ indica un punto de cálculo, y los superíndices ^{n}, ^{o} y ^{oo} el tiempo considerado (valores actuales (new), anteriores (old) o anteriores al anterior (old-old), respectivamente).

El **término convectivo** se integra sobre un volumen de control y se linealiza usando el teorema de Gauss para transformar la integral en una de superficie:

$$\int_V \nabla\cdot(\rho U\phi)\ \mathrm{d}V = \int_S \mathrm{d}S\cdot(\rho U\phi) = \sum_f S_f\cdot(\rho U)_f\,\phi_f = \sum_f F\phi_f \ , \tag{B.8}$$

en donde el campo ϕ_f se puede evaluar usando alguno de los esquemas disponibles: diferencia central (`CD` en `fvSchemes`), upwind (`UD`) o diferencia blended (`BD`).

Con el **término laplaciano** se hace un trabajo similar al anterior:

$$\int_V \nabla\cdot(\Gamma\nabla\phi)\ \mathrm{d}V = \int_S \mathrm{d}S\cdot(\Gamma\nabla\phi) = \sum_f \Gamma_f S_f\cdot(\nabla\phi)_f \ , \tag{B.9}$$

aproximando el término $S_f \cdot (\nabla\phi)_f$ como

$$S_f \cdot (\nabla\phi)_f = |S_f| \frac{\phi_N - \phi_P}{|\mathrm{d}|} \,, \tag{B.10}$$

donde el subíndice $_N$ se refiere a una celda vecina y d a la distancia entre el centro de las celdas.

B.1.2.2 Discretización temporal

En *OpenFOAM* una integral temporal puede ser discretizada de tres maneras:

- Euler implícita, considerando valores actuales de la variable (ϕ^n):

$$\int_t^{t+\Delta t} A\,\phi\,\mathrm{d}t = A\,\phi^n\,\Delta t \,, \tag{B.11}$$

  que es de primer orden en el tiempo, garantiza acotamiento (boundedness) y es incondicionalmente estable.
- Explícita, que considera valores antiguos de la variable (ϕ^o)

$$\int_t^{t+\Delta t} A\,\phi\,\mathrm{d}t = A\,\phi^o\,\Delta t \,, \tag{B.12}$$

  y es también de primer orden en el tiempo pero es inestable si el número de Courant es mayor que uno.
- Crank Nicholson, que usa un promedio entre los valores actuales y los antiguos

$$\int_t^{t+\Delta t} A\,\phi\,\mathrm{d}t = A\left(\frac{\phi^n + \phi^o}{2}\right)\Delta t \,, \tag{B.13}$$

  y es de segundo orden en el tiempo, incondicionalmente estable, pero no garantiza acotamiento (boundedness).

Es importante tener presente al programar el tratamiento que se realiza en el código de la discretización temporal, pues la discretización utilizada dependerá de la forma en que definamos las ecuaciones. Por ejemplo si queremos resolver la ecuación

$$\frac{\partial\phi}{\partial t} = \kappa\nabla^2\phi \,, \tag{B.14}$$

una implementación Euler implícita sería utilizando el indicador `fvm::`

```
solve(fvm::ddt(phi) == kappa*fvm::laplacian(phi))
```

mientras que una explícita ocuparía el indicador `fvc::`

```
solve(fvm::ddt(phi) == kappa*fvc::laplacian(phi))
```

Para usar una discretización estilo Crank Nicholson tendríamos que combinar ambas:

```
solve
   (
   fvm::ddt(phi)
   ==
   kappa*0.5*(fvm::laplacian(phi) + fvc::laplacian(phi))
   )
```

Para una descripción más detallada de los métodos numéricos usados por *OpenFOAM*, que requeriría mucho más espacio de el que disponemos aquí, por favor referirse a ambos manuales del código, del usuario y del programador (OpenCFD (2009)), y a la información disponible en Internet.

B.1.3 Modelos termofísicos, radiativos y de turbulencia

Dada su importancia en el problema que nos interesa simular es importante conocer con precisión cómo se integran en *OpenFOAM* los modelos termofísicos, radiativos y de turbulencia. Si bien su desarrollo es demasiado extenso para presentarlo aquí en detalle, con cientos de líneas de código empleadas en las distintas librerías, realizaremos a continuación una breve descripción de las herramientas disponibles y de cómo éstas se integran al momento de programar usando *OpenFOAM*.

B.1.3.1 Termofísica

Al resolver un problema que involucre relaciones termofísicas, *OpenFOAM* pone a nuestra disposición una serie de librerías capaces, en su conjunto, de integrarse con el resto del modelo y resolver el cálculo de las variables térmicas necesarias. Las librerías más importantes se resumen en la tabla B.6.

Como se observa en la tabla, que no incluye aquellas librerías aplicables a combustión, las librerías están agrupadas en categorías bien definidas que se aplican a características especiales del problema termofísico. Para definir un modelo necesitaremos usar una librería de cada categoría. El significado de cada categoría es el siguiente:

- Modelo termofísico: define el modelo utilizado en la simulación. Variará dependiendo si usamos un modelo compresible (asociado a las alternativas del tipo `Rho`), o incompresible. Cada una de las alternativas de esta categoría está asociada a un código en particular que contiene las ecuaciones que resolverán la termodinámica de la simulación.
- Propiedades de mezcla: la elección depende de si el fluido a considerar es puro o corresponde a una mezcla.
- Propiedades de transporte: esta librería definirá el tratamiento que hará el modelo de las propiedades de transporte.

Modelo termofísico (`thermoModel`)	
`hPsiThermo`	general, basado en la entalpía h y la compresibilidad ϕ
`hsPsiThermo`	general, basado en la entalpía sensible h_s y en ϕ
`ePsiThermo`	general, basado en la energía interna e y en ϕ
`hRhoThermo`	general, basado en la entalpía h
`hsRhoThermo`	general, basado en la entalpía sensible h_s
Propiedades de mezcla (`mixture`)	
`pureMixture`	general, para mezclas gaseosas pasivas
`basicMultiComponentMixture`	mezcla básica basada en múltiples componentes
`MultiComponentMixture`	mezcla derivada basada en múltiples componentes
Propiedades de transporte (`transport`)	
`constTransport`	constantes
`polynomialTransport`	basadas en polinomios
`sutherlandTransport`	basadas en la ecuación de Sutherland
Propiedades termofísicas derivadas (`specieThermo`)	
`specieThermo`	derivadas de C_p, h y/o s
Propiedades termofísicas básicas (`thermo`)	
`hConstThermo`	C_p constante con evaluación de entalpía h y entropía s
`eConstThermo`	C_p constante con evaluación de energía interna e y s
`hPolynomialThermo`	C_p evaluado con un polinomio, cálculo de h y s
`janafThermo`	C_p evaluado de tablas termodinámicas JANAF
Ecuación de estado (`equationOfState`)	
`icoPolynomial`	polinomial incompresible, para líquidos
`perfectGas`	gas ideal

Table B.6: Librerías termofísicas disponibles en *OpenFOAM*, en donde no hemos considerado aquellas aplicables a combustión. Para una referencia completa referirse al manual (OpenCFD (2009)).

- Propiedades termofísicas derivadas: contiene relaciones para derivar propiedades a partir de las propiedades básicas.
- Propiedades termofísicas básicas: en esencia establece si supondremos el calor específico constante o no.
- Ecuación de estado: define la ecuación de estado usada para modelar el fluido.

Es interesante que, si bien la elección de las librerías termofísicas modifica el modo en el que el modelo realiza el cálculo del problema, en *OpenFOAM* esta elección se realiza en los archivos que definen la simulación, y no en el código que define el solver. Esta característica nos permite ejecutar un mismo caso cambiando las librerías termodinámicas usadas, lo que si bien puede ser una ventaja en algunos casos, complica un poco la simulación pues es necesario tener cuidado de usar librerías termodinámicas que se ajusten al tipo de problema que se pretende resolver, y en particular al modelo utilizado para el cálculo general. Por ejemplo, de tratarse de una simulación que considere un fluido compresible, las librerías termodinámicas que seleccionemos deben considerar esto.

Para definir las propiedades termofísicas, tanto las librerías usadas como las propiedades

del fluido, usamos el archivo de texto `thermophysicalProperties`, contenido en la carpeta `constant/` dentro de la carpeta que contiene la simulación. El contenido típico de uno de estos archivos podría ser el siguiente:

```
thermoType      hRhoThermo<pureMixture<constTransport<specieThermo
                <hConstThermo<perfectGas»»>;

mixture         air 1 28.9 1000 0 1.8e-05 0.7;

pRef            100000;
```

La línea asociada a `thermoType` es la encargada de definir las librerías termodinámicas usadas, separadas por el delimitador `<`. De acuerdo a lo expresado anteriormente en este caso tendríamos:

- `hRhoThermo`: Modelo general de cálculo termofísico basado en la entalpía, considera compresibilidad.
- `pureMixture`: Modelo para mezclas gaseosas pasivas.
- `constTransport`: Propiedades de transporte constantes.
- `specieThermo`: Propiedades termofísicas de las especies se derivan de C_p, h y s.
- `hConstThermo`: Modelo de calor específico C_p constante, con cálculo de entalpía h y entropía s.
- `perfectGas`: Ecuación de estado de gas ideal.

La línea de código que se inicia con `mixture` establece las propiedades termofísicas para cada especie. En este caso hay sólo una, el aire (`air`). Las propiedades que establece son las siguientes:

- $n_{\text{moles}} = 1$: número de moles de la especie
- $W = 28.9$: peso molecular, en kg/kmol
- $C_p = 1000$: calor específico a presión constante
- $H_f = 0$: calor de fusión (no considerado)
- $\mu = 1.8 \cdot 10^{-5}$: viscosidad dinámica
- $Pr = 0.7$: número de Prandtl

Las propiedades que es necesario definir en este archivo dependerán de las librerías termodinámicas seleccionadas.

Finalmente `pRef` establece la presión de referencia, que en este caso es 1 atmósfera o 100000 Pa.

Acceso de las variables termodinámicas en el modelo

Para acceder desde el código a las variables termodinámicas generadas por las librerías descritas anteriormente usaremos el comando `thermo`. Así, por ejemplo, la densidad calculada por el modelo termodinámico será `thermo.rho()`, la presión `thermo.p()`, la compresibilidad `thermo.psi()` y la entalpía `thermo.h()`. Además, en el código principal deberemos incluir

una llamada al código que controla la definición del modelo termodinámico, a través del comando

```
#include "basicRhoThermo.H"
```

Para actualizar el valor de las variables termodinámicas usando las librerías definidas anteriormente, en particular después de el cálculo de alguna ecuación dentro del modelo general, usamos la orden

```
thermo.correct();
```

que forzará el cálculo termodinámico corrigiendo las variables asociadas.

B.1.3.2 Radiación

En la actualidad *OpenFOAM* ofrece únicamente dos modelos de radiación, los que se resumen en la tabla B.7, aunque no se descarta que a futuro se incluyan nuevos modelos.

Modelo de radiación `radiation`	
`P1`	modelo P1
`fvDOM`	método 'finite volume discrete ordinate'

Table B.7: Modelos radiativos disponibles en *OpenFOAM*.

Para acceder a las herramientas que ofrecen estos modelos es necesario incluir, dentro del código base de la aplicación que deseamos desarrollar, un llamado al código de control de estos modelos:

```
#include "radiationModel.H"
```

y al código que permite la creación del modelo:

```
#include "createRadiationModel.H"
```

Una vez incluidas estas órdenes para utilizar la herramienta usamos el comando `radiation`. En particular, como veremos más adelante, por lo general el único llamado a los modelos radiativos ocurre al calcular la ecuación de energía. Si resolvemos la ecuación para la entalpía h, incluiremos el término radiativo usando el comando `radiation->Sh(thermo)`.

Esta función permitirá incluir el término radiativo como una fuente en la ecuación de energía asociada a la entalpía. Para entender su significado revisemos el código que la define:

`src/thermophysicalModels/radiation/radiationModel/radiationModel.C`:

```
00137 Foam::tmp<Foam::fvScalarMatrix> Foam::radiation::radiationModel::Sh
00138 (
00139     basicThermo& thermo
00140 ) const
00141 {
00142     volScalarField& h = thermo.h();
00143     const volScalarField cp = thermo.Cp();
00144     const volScalarField T3 = pow3(T_);
00145
00146     return
00147     (
00148         Ru()
00149       - fvm::Sp(4.0*Rp()*T3/cp, h)
00150       - Rp()*T3*(T_ - 4.0*h/cp)
00151     );
00152 }
```

en donde `fvm::Sp` es sólo una función para reescribir matricialmente el contenido. El código anterior se traduce entonces en

$$Sh() = Ru() - 4Rp()\frac{T^3 h}{C_p} - Rp()T^3\left(T - 4\frac{h}{C_p}\right) \ , \tag{B.15}$$

lo que simplificando términos se reduce a (la forma de la ecuación (B.15) mejora la estabilidad numérica):

$$Sh() = Ru() - Rp()T^4 \ . \tag{B.16}$$

Las funciones $Ru()$ y $Rp()$ dependerán del modelo radiativo seleccionado. Por ejemplo, en el caso del modelo P1 se tiene para $Rp()$:

`src/thermophysicalModels/radiation/radiationModel/P1/P1.C`:

```
00188 Foam::tmp<Foam::volScalarField> Foam::radiation::P1::Rp() const
00189 {
00190     return tmp<volScalarField>
00191     (
00192         new volScalarField
00193         (
00194             IOobject
00195            (
00196                 "Rp",
00197                 mesh_.time().timeName(),
00198                 mesh_,
00199                 IOobject::NO_READ,
00200                 IOobject::NO_WRITE,
00201                 false
00202             ),
00203             4.0*absorptionEmission_->eCont()*radiation::sigmaSB
00204        )
00205     );
00206 }
```

lo que representa

$$Rp() = 4\,\varepsilon\,\sigma_{SB}\ , \tag{B.17}$$

con ε el coeficiente de emisión calculado mediante el modelo y σ_{SB} la constante de Stefan-Boltzmann.

En el mismo código $Ru()$ se define como

```
00209 Foam::tmp<Foam::DimensionedField<Foam::scalar, Foam::volMesh> >
00210 Foam::radiation::P1::Ru() const
00211 {
00212     const DimensionedField<scalar, volMesh>& G =
00213         G_.dimensionedInternalField();
00214     const DimensionedField<scalar, volMesh> E =
00215         absorptionEmission_->ECont()().dimensionedInternalField();
00216     const DimensionedField<scalar, volMesh> a =
00217         absorptionEmission_->aCont()().dimensionedInternalField();
00218
00219     return  a*G - 4.0*E;
00220 }
```

es decir

$$Ru() = aG - 4E\ , \tag{B.18}$$

donde a es el coeficiente de absorción y E un coeficiente asociado a la contribución a la emisión determinada por el modelo de absorción-emisión usado por el modelo radiativo. G es la intensidad de radiación incidente calculada por el modelo.

Reemplazando las relaciones (B.17) y (B.18) en la ecuación para $Sh()$ (B.16), se tiene finalmente:

$$Sh() = aG - 4\left(\varepsilon\sigma_{SB}T^4 + E\right)\ . \tag{B.19}$$

Por lo que en definitiva el aporte radiativo a la ecuación de entalpía se resume en un aporte dado por la absorción de radiación incidente, y una reducción generada por la radiación emitiva. Los distintos modelos radiativos calcularán los factores de esta relación de diferente manera, pero en esencia seguirá siendo la misma ecuación. Por lo tanto, cada vez que incluyamos en una ecuación de energía el operador `radiation->Sh(thermo)` estaremos incluyendo el aporte energético calculado por los modelos radiativos, dejando que *OpenFOAM* establezca las relaciones necesarias para calcular el valor de este aporte dependiendo de los modelos radiativos que hayamos definido para la simulación.

Para forzar el cálculo radiativo, y actualizar las variables involucradas usamos

```
radiation.correct();
```

dentro del código.

Para más información con respecto a los modelos radiativos en *OpenFOAM*, referirse al excelente trabajo de Alexey Vdovin (Vdovin (2009)).

B.1.3.3 Turbulencia

La forma correcta de incluir los procesos turbulentos en los modelos desarrollados en *OpenFOAM* es utilizar alguna de las muchas librerías que el código ofrece para esto. Al igual que en los casos anteriores, si bien el cálculo de la turbulencia afectará directamente el funcionamiento del modelo, la definición de las librerías utilizadas se realiza mediante un archivo de texto incluido en la carpeta `constant`, que está dentro de la carpeta que contiene la simulación. En este caso particular existen dos archivos. El primero de los cuales es `turbulenceProperties`, que usaremos para definir el tipo de modelo turbulento que usaremos. Existen tres alternativas:

- `laminar`: no se aplica ningún modelo de turbulencia
- `RASModel`: usa modelos del tipo RAS (Reynolds Averaged Stress)
- `LESModel`: usa modelos del tipo LES (Large Eddy Simulation)

y la elección la realizaremos dentro del archivo mediante la orden

```
simulationType  LESModel;
```

ejemplificando éste un caso en que usaremos modelos LES, aunque podría ser cualquiera de los anteriores.

Una vez definido el tipo de modelo usamos un segundo archivo, el que dependiendo del tipo de modelo que hayamos definido anteriormente se llamará `RASProperties` o `LESProperties`, en el que precisaremos qué modelo RAS o LES usaremos en la simulación, y podremos configurar algunas de las variables de control del modelo. Para conocer la extensa lista de modelos RAS y LES disponibles, tanto para fluidos incompresibles como compresibles, referirse a la tabla 3.9 del manual de usuario (OpenCFD (2009)).

Acceso de las variables turbulentas en el modelo

Para incluir el cálculo de turbulencia en el modelo deberemos incluir en el código base un llamado al código que controla la ejecución de las librerías necesarias. Para ello usamos la línea de código

```
#include "turbulenceModel.H"
```

Ahora bien, dentro de los códigos que usemos para resolver las ecuaciones de control del modelo, que expondremos más adelante, podremos acceder a las variables turbulentas mediante el comando `turbulence`. En particular hay dos comandos que debemos tener presentes:

- `turbulence->divDevRhoReff(U)`: como veremos más adelante este término estará incluido dentro de la ecuación de momentum para representar los flujos turbulentos. Para entender su significado vale la pena revisar su definición en uno de los códigos asociados a los distintos modelos de turbulencia. Por ejemplo, en el caso del modelo $\kappa - \varepsilon$,

tenemos, en el archivo

`src/turbulenceModels/compressible/RAS/kEpsilon/kEpsilon.C`:

```
00223 tmp<fvVectorMatrix> kEpsilon::divDevRhoReff(volVectorField& U) const
00224 {
00225    return
00226    (
00227      - fvm::laplacian(muEff(), U)
00228      - fvc::div(muEff()*dev2(fvc::grad(U)().T()))
00229    );
00229 }
```

En notación vectorial esto se refiere a:

$$\text{divDevRhoReff}\left(\overline{u}\right) = -\nabla^2\left(\mu_{eff}\,\overline{u}\right) - \nabla\cdot\left\{\mu_{eff}\,\text{dev2}\left[\left(\nabla\overline{u}\right)^T\right]\right\}\ , \tag{B.20}$$

en donde el operador dev2 genera la parte deviatórica del tensor, tal como está definido en el código del archivo

`src/OpenFOAM/primitives/Tensor/TensorI.H`:

```
00442 //- Return the deviatoric part of a tensor
00443 template <class Cmpt>
00444 inline Tensor<Cmpt> dev2(const Tensor<Cmpt>& t)
00445 {
00446     return t - SphericalTensor<Cmpt>::twoThirdsI*tr(t);
00447 }
```

lo que representa que, siendo I la matriz identidad, se cumple:

$$\text{dev2}\left(A\right) = A - \frac{2}{3}\text{traza}(A)\text{I}\ . \tag{B.21}$$

Reemplazando la relación anterior en la ecuación (B.20) tenemos:

$$\text{divDevRhoReff}\left(\overline{u}\right) = -\nabla^2\left(\mu_{eff}\,\overline{u}\right) - \nabla\cdot\left\{\mu_{eff}\,\left[\left(\nabla\overline{u}\right)^T - \frac{2}{3}\text{traza}\left[\left(\nabla\overline{u}\right)^T\right]\text{I}\right]\right\}\ . \tag{B.22}$$

Lo que se puede reordenar y escribir como:

$$\text{divDevRhoReff}\left(\overline{u}\right) = -\frac{\partial}{\partial x_j}\left\{\mu_{eff}\left[\left(\frac{\partial\overline{u}_{\text{i}}}{\partial x_j}\right) - \frac{2}{3}\left(\frac{\partial\overline{u}_k}{\partial x_k}\right)\delta_{\text{i}j}\right]\right\}\ , \tag{B.23}$$

que como veremos más adelante representa directamente el término asociado a los esfuerzos de Reynolds en la ecuación de momentum.

De esta forma, al usar el operador `turbulence->divDevRhoReff(U)`, disponible en *OpenFOAM*, estamos haciendo referencia directa al término turbulento, dejándole al modelo la libertad de calcular μ_{eff} según el modelo turbulento que hayamos definido para la simulación. De esta forma se simplifica notablemente la inclusión del cálculo turbulento en los códigos generados en *OpenFOAM*, y es posible usar directamente las librerías disponibles en el código. Si bien hemos desarrollado el caso particular del modelo RAS $\kappa - \varepsilon$, el desarrollo es similar para los otros modelos, tanto RAS como LES, estructurándose siempre el código de tal manera que el operador `turbulence->divDevRhoReff(U)` haga referencia a la descripción que el modelo turbulento seleccionado haga de los esfuerzos turbulentos.

- `turbulence->alphaEff()`: este operador turbulento será usado para representar la difusividad térmica turbulenta efectiva. Al igual que en el caso anterior, para entender cómo opera revisemos el caso particular del modelo de turbulencia $\kappa - \varepsilon$. En este caso la definición del operador viene dada en el archivo

  `src/turbulenceModels/compressible/RAS/kEpsilon/kEpsilon.H`:

```
00140        //- Return the effective turbulent thermal diffusivity
00141        virtual tmp<volScalarField> alphaEff() const
00142        {
00143            return tmp<volScalarField>
00144            (
00145                new volScalarField("alphaEff", alphat_ + alpha())
00146            );
00147        }
```

  lo que significa que la difusividad térmica turbulenta efectiva viene dada por la suma de la difusividad térmica turbulenta y la molucular:

$$\alpha_{eff} = \alpha_t + \alpha \ . \tag{B.24}$$

  Ahora bien, la difusividad turbulenta será calculada a partir de la viscosidad turbulenta μ_t en el código que define el modelo de turbulencia respectivo. En el caso del modelo $\kappa - \varepsilon$ tenemos:

  `src/turbulenceModels/compressible/RAS/kEpsilon/kEpsilon.C`:

```
00323    // Re-calculate viscosity
00324    mut_ = rho_*Cmu_*sqr(k_)/epsilon_;
00325    mut_.correctBoundaryConditions();
00326
00327    // Re-calculate thermal diffusivity
00328    alphat_ = mut_/Prt_;
00329    alphat_.correctBoundaryConditions();
```

  lo que representa la relación típica de este modelo turbulento:

$$\mu_t = \rho \, C_\mu \frac{k^2}{\varepsilon} \ . \tag{B.25}$$

  Y el cálculo de la difusividad turbulenta mediante

$$\alpha_t = \frac{\mu_t}{Pr_t} \ , \tag{B.26}$$

  donde Pr_t será la constante asignada en el modelo al número de Prandtl turbulento.

  Lo mismo aplicará para los diferentes modelos de turbulencia, incluidos los modelos LES, por lo que usando el operador `turbulence->alphaEff()` podremos hacer referencia directa al valor de la difusividad térmica efectiva y dejar que el código establezca las relaciones necesarias para calcularla dependiendo del modelo turbulento que hayamos definido para la simulación.

B.1.4 Desarrollo de aplicaciones

En *OpenFOAM* la forma correcta de integrar las herramientas descritas anteriormente es el desarrollo de aplicaciones. Estas aplicaciones pueden ser solvers para resolver algún tipo de problema en ecuaciones diferenciales, como el que nos proponemos desarrollar, o herramientas de manipulación (utilidades) que nos permitan interactuar con el dominio de un problema particular, definiendo por ejemplo condiciones de borde especiales, o extrayendo información desde el resultado de una simulación.

Cualquier aplicación desarrollada por el usuario debe ser creada dentro de la carpeta *applications* y estar contenida en una carpeta con su nombre, la que contendrá un código central que llevará necesariamente el nombre de la aplicación desarrollada con extensión .C, y diferentes archivos auxiliares con la extensión .H, que serán invocados por el código principal. Además la carpeta debe contener una subcarpeta *Make*, que definirá las condiciones de compilación a través de los archivos de texto `files` y `options`. El archivo `files` contendrá la ruta del ejecutable que se creará, mientras que el archivo `options` establecerá las rutas a las librerías de *OpenFOAM* que el código necesita para ejecutarse. Un ejemplo de ambos se presentará más adelante cuando hagamos referencia al código desarrollado para este trabajo.

Una vez definidos los códigos .C y .H necesarios para la aplicación, y establecidas las rutas a través de los códigos contenidos en la carpeta *Make*, es posible compilar el código mediante la herramienta `wmake`, compilador específico provisto en la distribución de *OpenFOAM*.

B.2 Solver

El solver usado en este trabajo es una modificación del solver *buoyantPimpleFoam* y está diseñado para resolver flujos boyantes no estacionarios con transporte de un escalar pasivo usando modelos LES o RANS de turbulencia e incorporando en el cálculo modelos radiativos. Si bien no se hace uso de la aproximación de incompresibilidad el solver no es totalmente compresible (no puede resolver por ejemplo ondas de choque). Es compresible en el sentido de que no usa la aproximación de Boussinesq para calcular la boyancia, sino que como veremos más adelante hace una simplificación para incluir los cambios de densidad en el cálculo. Algunas de las variables usadas en los distintos módulos del solver se resumen a continuación en la tabla B.8.

símbolo	**en el código**	**variable**	**archivos**
p	`p`	presión	rhoEqn
ρ	`rho`	densidad	rhoEqn UEqn hEqn traza
h	`h`	entalpía	hEqn
ϕ	`phi`	flujo a través de las caras $= \rho\vec{U}.\vec{S_f}$	UEqn hEqn rhoEqn traza
α_{eff}	`alphaEff`	difusividad térmica turbulenta efectiva	hEqn traza
ψ	`psi`	compresibilidad $= \rho/p$	UEqn
μ	`mu`	viscosidad laminar molecular	UEqn
μ_t	`mut`	viscosidad turbulenta	UEqn
μ_{eff}	`muEff`	viscosidad efectiva $= \mu + \mu_t$	UEqn
C	`traza`	trazador pasivo	traza
$\frac{\mathrm{d}p}{\mathrm{d}t}$	`DpDt`	derivada temporal de la presión	hEqn rhoEqn
$p_{\rho gh}$	`p_rgh`	presión corregida, $p - \rho gh$	UEqn

Table B.8: Algunas de las variables usadas en el solver.

El solver trabaja en función a distintos módulos (archivos .H), que se integran mediante un código central (archivo .C), el que hace llamados a estos módulos y a diferentes librerías disponibles en *OpenFOAM* a medida que son necesarios en el cálculo (figura B.1). A continuación expondremos el detalle de cada uno de estos módulos y terminaremos integrando el código central. Diferentes variantes del código base que se describe a continuación fueron utilizadas durante el desarrollo de este trabajo.

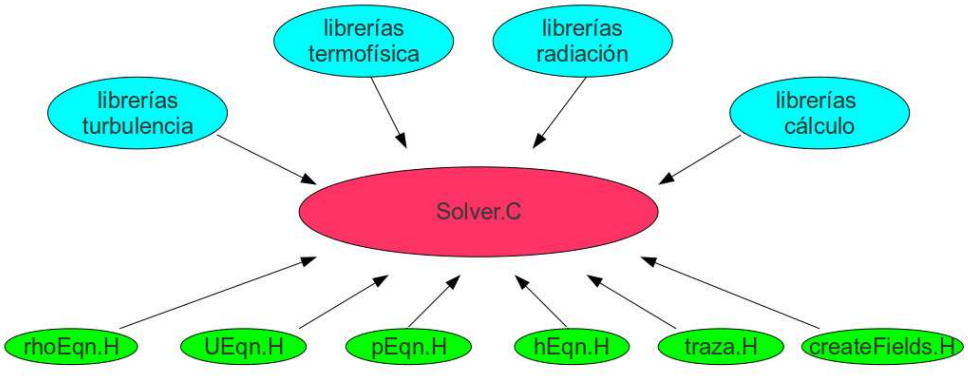


Figure B.1: Componentes del código

B.2.1 Ecuación de Continuidad

En este caso al considerar compresibilidad deberemos resolver la ecuación de continuidad general:

$$\frac{D\rho}{Dt} + \rho(\nabla \cdot \vec{U}) = 0 \,, \tag{B.27}$$

$$\frac{\partial \rho}{\partial t} + \vec{U} \cdot \nabla\rho + \nabla \cdot (\rho\vec{U}) = 0 \,, \tag{B.28}$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho\vec{U}) = 0 \,. \tag{B.29}$$

Si definimos el flujo a través de las caras `phi`, $\phi = \rho\vec{U}.\vec{S}_f$, obtenemos finalmente:

$$\boxed{\frac{\partial \rho}{\partial t} + \nabla \cdot \phi = 0} \tag{B.30}$$

Esta ecuación está incluida en el archivo `rhoEqn.H`:

```
/*-----------------------------------*/
{
    solve(fvm::ddt(rho) + fvc::div(phi));
}
/ ************************************************************************ //
```

y será invocada por el solver durante el cálculo.

B.2.2 Ecuación de Momentum

En una masa de control M_c podemos aplicar la primera ley de Newton:

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{M_c} \rho\vec{U} \mathrm{d}M_c = \sum \vec{F} \,, \tag{B.31}$$

y si consideramos un volumen de control V_c:

$$\frac{\partial}{\partial t} \int_{V_c} \rho\vec{U} \mathrm{d}V_c + \int_{SV_c} \rho\vec{U}\vec{U} \mathrm{d}\vec{S} = \sum \vec{F} \,. \tag{B.32}$$

Las fuerzas externas pueden ser fuerzas de superficie (presión, esfuerzos de corte y normales, etc), o fuerzas de cuerpo, que en este caso asociaremos únicamente a la gravedad. Por lo tanto podemos escribir:

$$\sum \vec{F} = \int_S T \mathrm{d}\vec{S} + \int_{V_c} \rho\vec{g} \, \mathrm{d}V_c \,. \tag{B.33}$$

Por lo tanto tenemos:

$$\frac{\partial}{\partial t}\int_{V_c}\rho\vec{U}\ \mathrm{d}V_c+\int_{SV_c}\rho\vec{U}\vec{U}\ \mathrm{d}\vec{S} = \int_S T\ \mathrm{d}\vec{S}+\int_{V_c}\rho\vec{g}\ \mathrm{d}V_c\ , \tag{B.34}$$

$$\frac{\partial}{\partial t}\int_{V_c}\rho\vec{U}\ \mathrm{d}V_c+\int_{SV_c}\rho\vec{U}\vec{U}\ \mathrm{d}\vec{S} = \int_{V_c}\nabla\cdot T\ \mathrm{d}V_c+\int_{V_c}\rho\vec{g}\ \mathrm{d}V_c\ , \tag{B.35}$$

$$\frac{\partial}{\partial t}(\rho\vec{U})+\nabla\cdot(\rho\vec{U}\vec{U}) = \nabla\cdot T+\rho\vec{g}\ . \tag{B.36}$$

Y para la componente i:

$$\frac{\partial}{\partial t}(\rho u_{\mathrm{i}})+\nabla\cdot(\rho u_{\mathrm{i}}\vec{U})=\nabla\cdot t_{\mathrm{i}}+\rho g_{\mathrm{i}}\ . \tag{B.37}$$

Si suponemos $t_{\mathrm{i}}=\tau_{\mathrm{i}j}\cdot\hat{\mathbf{i}}_j-p\hat{\mathbf{i}}_{\mathrm{i}}$

$$\frac{\partial}{\partial t}(\rho u_{\mathrm{i}})+\nabla\cdot(\rho u_{\mathrm{i}}\vec{U})=\nabla\cdot(\tau_{\mathrm{i}j}\cdot\hat{\mathbf{i}}_j)-\nabla\cdot(p\hat{\mathbf{i}}_{\mathrm{i}})+\rho g_{\mathrm{i}}\ , \tag{B.38}$$

pero,

$$\nabla\cdot(p\hat{\mathbf{i}}_{\mathrm{i}}) = (\nabla p)\cdot\hat{\mathbf{i}}_{\mathrm{i}}=\frac{\partial p}{\partial x_{\mathrm{i}}}\ , \tag{B.39}$$

$$\nabla\cdot(\tau_{\mathrm{i}j}\cdot\hat{\mathbf{i}}_j) = \frac{\partial\tau_{\mathrm{i}j}}{\partial x_j}\ , \tag{B.40}$$

$$\nabla\cdot(\rho u_{\mathrm{i}}\vec{U}) = \frac{\partial}{\partial x_j}(\rho u_j u_{\mathrm{i}})\ , \tag{B.41}$$

por lo que obtenemos finalmente:

$$\frac{\partial}{\partial t}(\rho u_{\mathrm{i}})+\frac{\partial}{\partial x_j}(\rho u_j u_{\mathrm{i}})=\frac{\partial\tau_{\mathrm{i}j}}{\partial x_j}-\frac{\partial p}{\partial x_{\mathrm{i}}}+\rho g_{\mathrm{i}}\ , \tag{B.42}$$

que se puede escribir como:

$$\frac{\partial}{\partial t}(\rho\vec{U})+\nabla\cdot(\rho\vec{U}\vec{U})+\nabla\cdot(\rho R)=-\nabla p+\rho\vec{g}\ . \tag{B.43}$$

Si definimos el flujo a través de las caras `phi`, $\phi=\rho\vec{U}.\vec{S_f}$, obtenemos finalmente:

$$\boxed{\frac{\partial}{\partial t}(\rho\vec{U})+\nabla\cdot(\phi\vec{U})+\nabla\cdot(\rho R)=-\nabla p+\rho\vec{g}} \tag{B.44}$$

En donde dentro del término $\nabla\cdot(\rho R)$ hemos incluido la turbulencia. Esta ecuación está implementada en el archivo UEqn.H, expuesto a continuación:

```
/*-----------------------------------*/
    fvVectorMatrix UEqn
    (
        fvm::ddt(rho, U)
      + fvm::div(phi, U)
      + turbulence->divDevRhoReff(U)
    );

    UEqn.relax();

    if (momentumPredictor)
    {
        solve
        (
            UEqn
         ==
            fvc::reconstruct
            (
                (
                  - ghf*fvc::snGrad(rho)
                  - fvc::snGrad(p_rgh)
                )*mesh.magSf()
            ),
            mesh.solver(U.select(finalIter))
        );
    }
// ************************************************************************* //
```

Este código está diseñado para predecir implícitamente la velocidad (resuelve la forma matricial $Ax = b$).

Primero se define el lado izquierdo de la ecuación de momentum (B.44):

$$\texttt{Ueqn} = \frac{\partial}{\partial t}(\rho\vec{U}) + \nabla \cdot (\phi\vec{U}) + \nabla \cdot (\rho R) \ , \tag{B.45}$$

incluido en el código como

```
    fvVectorMatrix UEqn
    (
        fvm::ddt(rho, U)
      + fvm::div(phi, U)
      + turbulence->divDevRhoReff(U)
    );
```

`fvm` señala que se trata de una "finite volume matrix", y es usado cuando las operaciones van a ser implícitas y se forma una matriz izquierda. Como ya hemos señalado al analizar la forma en que *OpenFOAM* trabaja con la turbulencia el operador `turbulence->divDevRhoReff(U)` contiene los términos turbulentos.

Luego se define la ecuación a resolver, equivalente a la ecuación de momentum completa (B.44):

$$\texttt{Ueqn} = -\nabla p + \rho\vec{g}\ , \tag{B.46}$$

incluida en el código mediante la función `solve`:

```
solve
(
    UEqn
 ==
    fvc::reconstruct
    (
        (
          - ghf*fvc::snGrad(rho)
          - fvc::snGrad(p_rgh)
        )*mesh.magSf()
    ),
    mesh.solver(U.select(finalIter))
);
```

en donde la clase `fvc` señala "finite volume calculus" y se usa para operaciones explícitas.

Dado que el término `ghf` es equivalente a gz y `p_rgh` $= p - \rho gz$, vemos que el término de la derecha en la ecuación (B.44) ha sido escrito en el código como:

$$-\,gz\nabla\rho - \nabla\left(p - \rho gz\right)\ , \tag{B.47}$$

lo que desarrollando y simplificando es equivalente al original:

$$\begin{aligned} -\,gz\nabla\rho - \nabla\left(p - \rho gz\right) &= -gz\nabla\rho - \nabla p + gz\nabla\rho + \rho g\nabla z\ , & \text{(B.48)}\\ &= -\nabla p + \rho\vec{g}\ , & \text{(B.49)} \end{aligned}$$

dado que el gradiente de z es sólo un indicador de dirección. La versión expandida del lado derecho de la ecuación es utilizada pues mejora la aplicación de los métodos numéricos sobre la ecuación, en particular el tratamiento matricial.

En el caso de ser necesario incluir términos extra a la ecuación de momentum, como la fuerza de Coriolis, o un forzante para mantener el flujo en el caso de usar condiciones de borde cíclicas, ésto se puede realizar fácilmente incluyendo la expresión correspondiente en el desarrollo expuesto anteriormente. En el caso de Coriolis, usamos `rho` y `U` para definir $\vec{f_c} = -2\rho\,\vec{\Omega}\times\vec{U}$, mientras que en el caso del forzante podemos implementarlo usando como guía el código disponible en el solver `channelFoam.C`, predefinido en *OpenFOAM*, y que sirve de referencia en el caso de usar condiciones de borde cíclicas.

B.2.3 Ecuación de Energía

La ecuación de energía para un fluido compresible toma la forma:

$$\frac{\partial}{\partial t}(\rho \mathrm{e}) + \frac{\partial}{\partial x_j}(\rho \mathrm{e} u_j) = -p\frac{\partial u_k}{\partial x_k} + \tau_{\mathrm{i}j}\frac{\partial u_{\mathrm{i}}}{\partial x_j} - \frac{\partial q_k}{\partial x_k} \ , \tag{B.50}$$

con e la energía interna y q_k el flujo de calor por conducción que deja el volumen de control. Si bien en el caso de un fluido incompresible podríamos despreciar el primer término del lado derecho (asociado a la divergencia de velocidad), en este caso no podemos hacerlo, pues la ecuación de continuidad para fluidos compresibles no se reduce a divergencia igual a cero. Por el contrario, el segundo término del lado derecho, asociado a $\tau_{\mathrm{i}j}$ sí se puede despreciar, al igual que en el caso incompresible, pues su orden de magnitud es $O(u^2)$, y en este caso a pesar de suponer compresibilidad consideraremos únicamente flujos de baja velocidad, por lo que este término tendrá un efecto menor sobre la energía interna comparado con la conducción de calor (véase Ferziger & Peric (2002); Churchfield (2010)).
Tenemos entonces que la ecuación (B.50) se reduce a:

$$\frac{\partial}{\partial t}(\rho \mathrm{e}) + \frac{\partial}{\partial x_j}(\rho \mathrm{e} u_j) = -p\frac{\partial u_k}{\partial x_k} - \frac{\partial q_k}{\partial x_k} \ . \tag{B.51}$$

Por razones que veremos más adelante, y que tienen que ver con la forma de cálculo que utiliza *OpenFOAM*, nos interesa expresar esta ecuación en términos de la entalpía h. Para ello es necesario recordar las siguientes relaciones:

$$h = c_p T \ , \qquad \mathrm{e} = c_v T \ , \qquad R = c_p - c_v \ , \tag{B.52}$$

$$\Rightarrow \qquad \mathrm{e} = h - RT \ . \tag{B.53}$$

Luego, reemplazando (B.53) en (B.51) se obtiene:

$$\frac{\partial}{\partial t}(\rho(h - RT)) + \frac{\partial}{\partial x_j}(\rho(h - RT)u_j) = -p\frac{\partial u_k}{\partial x_k} - \frac{\partial q_k}{\partial x_k} \ , \tag{B.54}$$

$$\frac{\partial}{\partial t}(\rho h) - \frac{\partial}{\partial t}(\rho RT) + \frac{\partial}{\partial x_j}(\rho u_j h) - \frac{\partial}{\partial x_j}(u_j \rho RT) = -p\frac{\partial u_k}{\partial x_k} - \frac{\partial q_k}{\partial x_k} \ , \tag{B.55}$$

y dado que se tiene que cumplir la ecuación de estado $p = \rho RT$, se tiene:

$$\frac{\partial}{\partial t}(\rho h) - \frac{\partial p}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_j h) - \frac{\partial}{\partial x_j}(u_j p) = -p\frac{\partial u_k}{\partial x_k} - \frac{\partial q_k}{\partial x_k} \ . \tag{B.56}$$

Combinando y expandiendo los términos asociados a la presión, y eliminando los términos repetidos:

$$\frac{\partial}{\partial t}(\rho h) + \frac{\partial}{\partial x_j}(\rho u_j h) = \frac{\partial p}{\partial t} + u_j\frac{\partial p}{\partial x_j} - \frac{\partial q_k}{\partial x_k} \ , \tag{B.57}$$

$$\frac{\partial}{\partial t}(\rho h) + \frac{\partial}{\partial x_j}(\rho u_j h) = \frac{Dp}{Dt} - \frac{\partial q_k}{\partial x_k} \ . \tag{B.58}$$

Si al igual que en los casos anteriores definimos el flujo a través de las caras `phi`, $\phi = \rho\vec{U}.\vec{S}_f$ podemos escribir:

$$\frac{\partial}{\partial t}(\rho h) + \nabla \cdot (\phi h) = \frac{Dp}{Dt} - \frac{\partial q_k}{\partial x_k} \ . \tag{B.59}$$

Ahora bien, resta encontrar una expresión para el término asociado a la conducción de calor (último término de la derecha). Este término debe considerar tanto el flujo de calor por conducción como el turbulento (de un desarrollo más detallado de la ecuación anterior en el que se incluyen los términos turbulentos a través de perturbaciones de las variables originales, y que no haremos aquí por razones de espacio, se puede comprobar que es posible incluir de forma aproximada los efectos turbulentos dentro del término asociado al flujo de calor (véase por ejemplo Churchfield (2010)). Luego, definiendo un coeficiente de transferencia de calor efectivo κ_{eff} podemos escribir:

$$q_k = -\kappa_{eff} \frac{\partial T}{\partial x_k} \, . \tag{B.60}$$

Recordando que se cumple $h = c_p T$, tenemos:

$$q_k = -\frac{\kappa_{eff}}{c_p} \frac{\partial h}{\partial x_k} \, , \tag{B.61}$$

$$q_k = -\alpha_{eff} \frac{\partial h}{\partial x_k} \, , \tag{B.62}$$

en donde hemos definido el término α_{eff}, difusividad turbulenta térmica efectiva. Tenemos entonces para el último término de la ecuación (B.59):

$$\frac{\partial q_k}{\partial x_k} = -\frac{\partial}{\partial x_k} \left(\alpha_{eff} \frac{\partial h}{\partial x_k} \right) \, , \tag{B.63}$$

$$\frac{\partial q_k}{\partial x_k} = -\nabla \cdot (\alpha_{eff} \nabla h) \, . \tag{B.64}$$

Incorporando esta expresión a la ecuación principal (B.59), tenemos:

$$\frac{\partial}{\partial t}(\rho h) + \nabla \cdot (\phi h) - \nabla \cdot (\alpha_{eff} \nabla h) = \frac{Dp}{Dt} \, . \tag{B.65}$$

Ahora bien, dado que en el desarrollo anterior no hemos considerado los aportes radiativos, deberemos incluirlos en esta ecuación como un término de fuente. Si llamamos a esta fuente rad y la incluimos en el lado izquierdo de la ecuación tenemos finalmente:

$$\boxed{\frac{\partial}{\partial t}(\rho h) + \nabla \cdot (\phi h) - \nabla \cdot (\alpha_{eff} \nabla h) - rad = \frac{Dp}{Dt}} \tag{B.66}$$

Esta será la ecuación que resolverá el modelo, y está incorporada en el archivo hEqn.H, expuesto a continuación:

```
{
    fvScalarMatrix hEqn
    (
        fvm::ddt(rho, h)
      + fvm::div(phi, h)
      - fvm::laplacian(turbulence->alphaEff(), h)
      - radiation->Sh(thermo)
     ==
        DpDt
    );

    hEqn.relax();
    hEqn.solve(mesh.solver(h.select(finalIter)));

    thermo.correct();

    radiation->correct();
}
```

En donde, tal como se describió al estudiar los modelos radiativos disponibles en *OpenFOAM*, la fuente radiativa ha sido incluida en la ecuación de la entalpía usando el operador `radiation->Sh(thermo)`.

Para forzar el cálculo termodinámico y de radiación en cada una de las iteraciones incluimos los operadores `thermo.correct()` y `radiation->correct()`.

B.2.4 Transporte Escalar Pasivo

La tasa de cambio de una propiedad cualquiera M en el sistema será igual a la tasa de cambio de M en el volumen de control más el valor neto del flujo saliente de M a través de la superficie de éste:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_V M\,\mathrm{d}V = \int_V \frac{\partial M}{\partial t}\,\mathrm{d}V + \oint_S M\vec{U}\cdot\vec{n}\,\mathrm{d}S\ . \tag{B.67}$$

Dado el teorema de Gauss, podemos escribir:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_V M\,\mathrm{d}V = \int_V \left[\frac{\partial M}{\partial t} + \nabla\cdot\left(M\vec{U}\right)\right]\,\mathrm{d}V\ . \tag{B.68}$$

Pueden existir también fuentes locales de M, como fuentes volumétricas distribuidas a lo largo del volumen y que aquí llamaremos f_v, y fuentes superficiales, como la difusión que actúa a través de las superficies del volumen de control, y que aquí llamaremos $\vec{f}_s$.

Tenemos entonces:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_V M\,\mathrm{d}V = \int_V f_v\,\mathrm{d}V - \oint_S \vec{f}_s\cdot\vec{n}\,\mathrm{d}S\ . \tag{B.69}$$

Recuperando la expresión anterior, ecuación (B.68), y usando nuevamente el teorema de Gauss:

$$\int_V \left[\frac{\partial M}{\partial t} + \nabla\cdot\left(M\vec{U}\right)\right]\,\mathrm{d}V = \int_V f_v\,\mathrm{d}V - \oint_S \vec{f}_s\cdot\vec{n}\,\mathrm{d}S\ , \tag{B.70}$$

$$\frac{\partial M}{\partial t} + \nabla\cdot\left(M\vec{U}\right) = f_v - \nabla\cdot\vec{f}_s\ . \tag{B.71}$$

Dado que en este caso nos interesa la concentración de un escalar pasivo, que llamaremos C, podemos suponer que M es la masa de este escalar, por lo que se cumple $M = \rho C$ y tenemos:

$$\frac{\partial \rho C}{\partial t} + \nabla\cdot\left(\rho C\vec{U}\right) = f_v - \nabla\cdot\vec{f}_s\ . \tag{B.72}$$

Ahora bien, para aproximar el término difusivo $\vec{f}_s$ deberemos incluir los efectos turbulentos. Para ello, y dado que suponemos que se trata de un escalar pasivo, usamos el término de difusividad turbulenta efectiva usado anteriormente α_{eff}, y calculado por los modelos turbulentos de *OpenFOAM*, para definir:

$$\vec{f}_s = -\alpha_{eff}\nabla\frac{M}{\rho} = -\alpha_{eff}\nabla C\ . \tag{B.73}$$

Incorporando esta expresión en la ecuación (B.72) tenemos entonces:

$$\frac{\partial \rho C}{\partial t} + \nabla\cdot\left(\rho C\vec{U}\right) - \nabla\cdot\left(\alpha_{eff}\nabla C\right) = f_v\ . \tag{B.74}$$

Si al igual que en los casos anteriores definimos el flujo a través de las caras `phi` ($\phi = \rho\vec{U}.\vec{S_f}$), podemos escribir:

$$\boxed{\frac{\partial \rho C}{\partial t} + \nabla \cdot (\phi C) - \nabla \cdot (\alpha_{eff} \nabla C) = f_v} \tag{B.75}$$

Esta será la ecuación que resolverá el modelo, y está incorporada en el archivo traza.H, expuesto a continuación:

```
/*---------------------------------*/\\
for (int nonOrth=0; nonOrth<=nNonOrthCorr; nonOrth++)
{
    solve
    (
        fvm::ddt(rho,traza)
      + fvm::div(phi,traza)
      - fvm::laplacian(turbulence->alphaEff(),traza)
    );
}
// ************************************************************************ //
```

En donde hemos supuesto $f_v = 0$, pues en la mayoría de los casos podemos incluir las fuentes como condición de borde mediante las herramientas ya definidas en *OpenFOAM* para este fin.

B.2.5 Corrección de la Presión

La corrección de presión se realiza a través del archivo `pEqn.H`, que es estándar en *OpenFOAM*, pues implementa los métodos numéricos dispuestos por el código. Para una revisión más completa de los algoritmos del tipo PISO (*Pressure-Implicit with Splitting of Operators*), referirse a Issa (1985, 1986); Oliveira & Issa (2001) y Demirdzic et al. (1993)

La ecuación de momentum es discretizada para obtener:

$$UEqn = AU - H \,, \tag{B.76}$$

$$UEqn = \begin{pmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,n} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m,1} & A_{m,2} & \cdots & A_{m,n} \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \\ \vdots \\ U_n \end{pmatrix} - \begin{pmatrix} H_1 \\ H_2 \\ \vdots \\ H_n \end{pmatrix} \,, \tag{B.77}$$

donde $UEqn$ es el lado izquierdo de la ecuación de momentum, es decir sin los términos de presión ni gravitatorios.

Las matrices A y H se extraen en *OpenFOAM* mediante el comando `UEqn.A` y `UEqn.H` implementados en la librería `src/finiteVolume/fvMatrices/fvMatrix/fvMatrix.C` de tal manera de hacer de A una matriz diagonal.

Se define a continuación la matriz `rUa`, que es el inverso de `A` ($rUA = A^{-1}$), y es un campo vectorial (línea 5 del código). En la línea siguiente se multiplica la matriz `rUA` por la densidad `rho` y se interpola en las caras de cada celda para formar el campo escalar de superficie `rhorUAf`.

Para obtener la primera aproximación del campo de velocidades basta con multiplicar el inverso de `A` por `H`, realizando la aproximación:

$$AU - H = 0 \,, \tag{B.78}$$

$$U = \frac{H}{A} \,, \tag{B.79}$$

que es la contribución a la velocidad corregida sin incluir ni la presión ni la gravedad en el cálculo (línea 5).

Para obtener el flujo `phi` se multiplica una versión interpolada de la densidad por una de la velocidad, a la que previamente se le ha realizado producto punto con los vectores normales a cada cara (en *OpenFOAM* `&` es el operador que representa el producto interno o punto). En este cálculo, el término `fvc::ddtPhiCorr(rUA, rho, U, phi)`, que ocupa las líneas 6 a 10 del código, es sólo un corrector que da cuenta de la divergencia del campo de velocidad en las caras tomando la diferencia entre la velocidad interpolada y el flujo.

Para incluir los efectos boyantes se define un flujo boyante, que considera las variaciones de densidad:

$$buoyancyPhi = -\frac{\rho}{A} g h \nabla\rho \cdot \text{área} \,, \tag{B.80}$$

donde A es la matriz A definida anteriormente y se usan las definiciones `rhorUAf` $= -\rho/A$, `ghf` $= gh$, con h la altura del centro de la cara (ver definición de campos), y el gradiente de ρ es el gradiente normal (`SnGrad`). Todo es multiplicado por el área de cada cara (`mesh.magSf()`) para obtener un flujo.

A continuación se inicia un loop en donde la presión será corregida, con el objetivo de aplicar una corrección no ortogonal. La ecuación para la presión corregida se define en las líneas 15 a 20, expresada en términos de la presión estática p_{rgh}. Esta ecuación tiene como objetivo forzar a la presión a satisfacer continuidad, es decir el flujo neto en cada volumen de control debe ser cero.

$$\frac{\partial \rho}{\partial t} + \frac{\rho}{P}\frac{\partial p_{rgh}}{\partial t} + \nabla\phi - \nabla\left(\frac{\rho}{A}\nabla p_{rgh}\right) = 0\ . \tag{B.81}$$

Luego se corrigen los valores de `U` y de la densidad termodinámica.

Archivo `pEqn.H`:

```
{
1    rho = thermo.rho();

    // Thermodynamic density needs to be updated by psi*d(p) after the
    // pressure solution - done in 2 parts. Part 1:
2    thermo.rho() -= psi*p_rgh;

3    volScalarField rUA = 1.0/UEqn.A();
4    surfaceScalarField rhorUAf("(rho*(1|A(U)))", fvc::interpolate(rho*rUA));

5    U = rUA*UEqn.H();

6    phi = fvc::interpolate(rho)*
7    (
8        (fvc::interpolate(U) & mesh.Sf())
9      + fvc::ddtPhiCorr(rUA, rho, U, phi)
10   );

11   surfaceScalarField buoyancyPhi = -rhorUAf*ghf*fvc::snGrad(rho)*mesh.magSf();
12   phi += buoyancyPhi;

13   for (int nonOrth=0; nonOrth<=nNonOrthCorr; nonOrth++)
14   {
15       fvScalarMatrix p_rghEqn
16       (
17           fvc::ddt(rho) + psi*correction(fvm::ddt(p_rgh))
18         + fvc::div(phi)
19         - fvm::laplacian(rhorUAf, p_rgh)
20       );
```

```
21        p_rghEqn.solve
22        (
23            mesh.solver
24            (
25                p_rgh.select
26                (
27                    (
28                        finalIter
29                     && corr == nCorr-1
30                     && nonOrth == nNonOrthCorr
31                    )
32                )
33            )
34        );

35        if (nonOrth == nNonOrthCorr)
36        {
            // Calculate the conservative fluxes
37            phi += p_rghEqn.flux();

            // Explicitly relax pressure for momentum corrector
38            p_rgh.relax();

            // Correct the momentum source with the pressure gradient flux
            // calculated from the relaxed pressure
39            U += rUA*fvc::reconstruct((buoyancyPhi + p_rghEqn.flux())/rhorUAf);
40            U.correctBoundaryConditions();
41        }
42    }

43    p = p_rgh + rho*gh;

    // Second part of thermodynamic density update
44    thermo.rho() += psi*p_rgh;

45    DpDt = fvc::DDt(surfaceScalarField("phiU", phi/fvc::interpolate(rho)), p);

46    #include "rhoEqn.H"
46    #include "compressibleContinuityErrs.H"
47}
```

B.2.6 Creación de Campos

El archivo *createFields.H* es el encargado de crear las variables que se van a usar durante la ejecución del solver, e invocar desde otras librerías los parámetros necesarios. Este archivo cambiará dependiendo de qué información queremos extraer del solver, pero en términos generales tiene la siguiente forma:

```
    Info« "Reading thermophysical properties\n" « endl;
    autoPtr<basicRhoThermo> pThermo
    (
        basicRhoThermo::New(mesh)
    );
    basicRhoThermo& thermo = pThermo();

    volScalarField rho
    (
        IOobject
        (
            "rho",
            runTime.timeName(),
            mesh,
            IOobject::NO_READ,
            IOobject::NO_WRITE
        ),
        thermo.rho()
    );

    volScalarField& p = thermo.p();
    volScalarField& h = thermo.h();
    const volScalarField& psi = thermo.psi();

    Info« "Reading field U\n" « endl;
    volVectorField U
    (
        IOobject
        (
            "U",
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh
    );
    #include "compressibleCreatePhi.H"

    Info« "Creating turbulence model\n" « endl;
    autoPtr<compressible::turbulenceModel> turbulence
    (
```

```
        compressible::turbulenceModel::New
        (
            rho,
            U,
            phi,
            thermo
        )
    );

    Info<< "Calculating field g.h\n" << endl;
    volScalarField gh("gh", g & mesh.C());
    surfaceScalarField ghf("ghf", g & mesh.Cf());

    Info<< "Reading field p_rgh\n" << endl;
    volScalarField p_rgh
    (
        IOobject
        (
            "p_rgh",
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh
    );

    // Force p_rgh to be consistent with p
    p_rgh = p - rho*gh;

    Info<< "Creating field DpDt\n" << endl;
    volScalarField DpDt
    (
        "DpDt",
        fvc::DDt(surfaceScalarField("phiU", phi/fvc::interpolate(rho)), p)
    );

//traza
    Info<< "Reading field traza\n" << endl;
    volScalarField traza
    (
        IOobject
        (
            "traza",
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh
```

```
    );

//potential temperature
    Info« "Reading field theta\n" « endl;
    volScalarField theta
    (
        IOobject
        (
            "theta",
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh
    );
dimensionedScalar coefp
("coefp", dimensionSet(0, 2, -2, -1, 0, 0, 0),1004);
```

B.2.7 Código Principal

El código principal, llamado aquí `Solver.C`, establece el loop principal y hace llamados a los diferentes subcódigos necesarios para el cálculo. En términos generales toma la forma incluida a continuación:

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

#include "fvCFD.H"
#include "basicRhoThermo.H"
#include "turbulenceModel.H"
#include "fixedGradientFvPatchFields.H"
#include "radiationModel.H"

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

int main(int argc, char *argv[])
{
    #include "setRootCase.H"
    #include "createTime.H"
    #include "createMesh.H"
    #include "readGravitationalAcceleration.H"
    #include "createFields.H"
    #include "initContinuityErrs.H"
    #include "readTimeControls.H"
    #include "compressibleCourantNo.H"
    #include "createRadiationModel.H"
    #include "setInitialDeltaT.H"

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

    Info« "\nStarting time loop\n" « endl;

    while (runTime.run())
    {
        #include "readTimeControls.H"
        #include "readPIMPLEControls.H"
        #include "compressibleCourantNo.H"
        #include "setDeltaT.H"

        runTime++;

        Info« "Time = " « runTime.timeName() « nl « endl;

        #include "rhoEqn.H"

        // -- Pressure-velocity PIMPLE corrector loop
        for (int oCorr=0; oCorr<nOuterCorr; oCorr++)
```

```
        {
            bool finalIter = oCorr == nOuterCorr-1;

            if (nOuterCorr != 1)
            {
                p_rgh.storePrevIter();
            }

            #include "UEqn.H"
            #include "hEqn.H"

            // -- PISO loop
            for (int corr=0; corr<nCorr; corr++)
            {
                #include "pEqn.H"
            }

            turbulence->correct();

            rho = thermo.rho();
        }

#       include "traza.H"

dimensionedScalar po
("po", dimensionSet(1, -1, -2, 0, 0, 0, 0),100000);

theta = thermo.T() * pow(po/p,0.285856) ;

        runTime.write();

        Info« "ExecutionTime = " « runTime.elapsedCpuTime() « " s"
            « "  ClockTime = " « runTime.elapsedClockTime() « " s"
            « nl « endl;
    }

    Info« "End\n" « endl;

    return 0;
}
```

B.2.8 Compilación e Integración del Código

Para compilar los códigos descritos anteriormente es necesario incluir dos archivos de referencia en la subcarpeta *Make* de la carpeta que los contiene.

El primero de ellos lleva por nombre `files`, y es el encargado de hacer del código una aplicación, disponible para ser ejecutada desde la línea de comandos. Su contenido es el siguiente:

Archivo `files`:

```
Solver.C

EXE = $(FOAM_USER_APPBIN)/Solver
```

El segundo archivo necesario lleva por nombre `options`, e indica las rutas de acceso a los diferentes archivos de cada una de las librerías requeridas por el código principal.

Archivo `options`:

```
EXE_INC = \
    -I$(LIB_SRC)/thermophysicalModels/basic/lnInclude \
    -I$(LIB_SRC)/thermophysicalModels/radiation/lnInclude \
    -I$(LIB_SRC)/turbulenceModels/compressible/turbulenceModel \
    -I$(LIB_SRC)/finiteVolume/lnInclude

EXE_LIBS = \
    -lmeshTools \
    -lbasicThermophysicalModels \
    -lspecie \
    -lradiation \
    -lcompressibleTurbulenceModel \
    -lcompressibleRASModels \
    -lcompressibleLESModels \
    -lfiniteVolume
```

Una vez definidos estos archivos el solver puede ser compilado usando las herramientas `wclean` y `wmake`.

Appendix C

Generación de Mallas

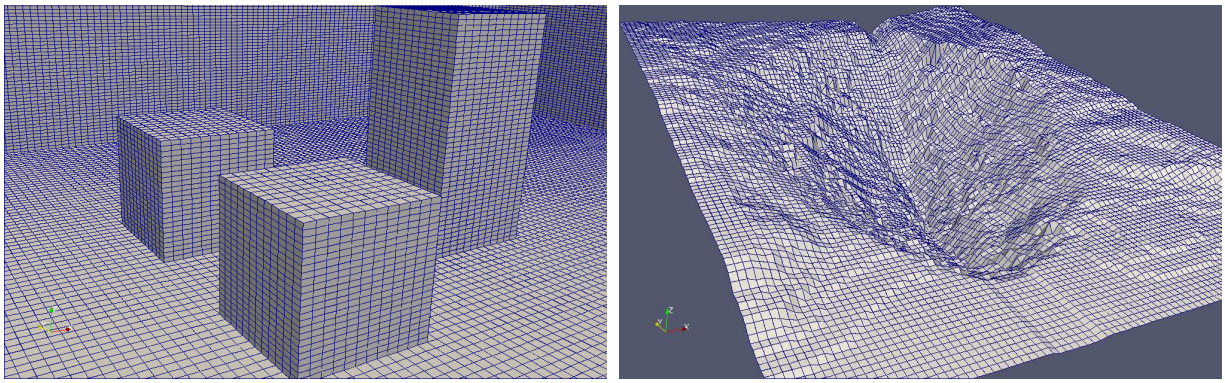
Una de las principales características de *OpenFOAM* es la rigurosidad con la que trata el tema del mallado. Conscientes de la importancia que tiene en la calidad de la resolución numérica, los desarrolladores del código colocaron un especial énfasis en asegurarse de contar con una malla de calidad antes de iniciar el trabajo de los solvers, y de hecho recomiendan el uso de la herramienta *checkMesh* antes de ejecutar cualquier simulación. Es en buena medida por esto que la generación de mallas sigue siendo un tema abierto en el desarrollo del software. Si bien la distribución actual del código ofrece un generador de malla propio, *blockMesh*, y permite importar mallas desde otros formatos, generar mallas de calidad compatibles con *OpenFOAM*, especialmente para geometrías complejas, sigue siendo un desafío.

En este trabajo abordamos el tema desde tres perspectivas: primero, para geometrías sencillas, y en los casos en los que nos interesaba contar con mallas de cuya calidad estuviéramos seguros, usamos *blockMesh* en conjunto con *Matlab* para generar el mallado. En segundo lugar, para geometrías complejas, utilizamos las herramientas provistas por el programa para la importación de mallas, desarrollando las mallas completas en softwares específicos e importándolas directamente a *OpenFOAM*. En tercer lugar, usamos una técnica que se puede definir como una combinación de las dos anteriores, pues consiste en generar un modelo 3D de la geometría que nos interesa, y utilizar el programa *snappyHexMesh*, incluido en la distribución de *OpenFOAM*, para generar a partir de esa geometría, y de una malla base generada con *blockMesh*, la malla final. Describimos a continuación los detalles de cada una de estas técnicas, y los resultados que generan.

C.1 *blockMesh* y *Matlab*

En esencia el código *blockMesh* genera mallas hexaédricas a partir de la información contenida en un archivo de texto, en el cual es necesario definir las coordenadas de los vértices de cada bloque en el que se divide el dominio, agrupar estos vértices dependiendo del bloque al que pertenecen, definir el tipo de mallado de cada bloque, y describir en función de los vértices las caras que usaremos como superficies de borde para la simulación.

Este sistema de entrada, mediante líneas de texto en donde se definen las coordenadas de los bloques que definen la malla, las características de la grilla, y las distintas zonas de borde, hacen posible que al generar una malla se tenga absoluto control de todos los parámetros que definen el mallado (tamaño de la malla, tratamiento de los bordes, zonas de refinamiento, etc). Esto permite conocer con absoluta seguridad la calidad de la malla con la que se realiza una determinada simulación, pero hace muy engorroso el proceso de creación de la misma, aún para el caso de geometrías sencillas. Sin embargo, dado que el proceso de generación de mallas usado por *blockMesh* sigue un protocolo estándar (descrito en detalle en el manual de usuario de *OpenFOAM* (OpenCFD (2009))), es posible, al comprender cómo funciona, automatizar el proceso y hacer que *Matlab* genere rápidamente el archivo de texto necesario para definir una malla en formato *blockMesh*. De esta forma se pueden generar mallas como las de la figura C.1, con absoluto control de la resolución. Sin embargo, este procedimiento es sólo aplicable a geometrías sencillas, como bloques o superficies extendidas.



(a) ejemplo malla bloques **(b)** ejemplo malla superficie

Figure C.1: Resultados del mallado usando una combinación de *blockMesh* y *Matlab* para automatizar el proceso.

C.2 Importación Desde Otros Softwares

Dado el interés que ha despertado *OpenFOAM*, en los últimos años se han desarrollado, y se están desarrollando, varias aplicaciones para generar mallas en su formato. En particular, existen dos alternativas, de código libre, que permiten exportar directamente la malla:

- *Netgen* (http://www.hpfem.jku.at/netgen/)
- *Discretizer* (http://www.discretizer.org/)

Si bien ambas alternativas cumplen con lo ofrecido, el proceso no es tan sencillo como se podría esperar. En particular, *Netgen* es un generador de mallas tetrahédricas, a diferencia de *blockMesh*, el estándar de *OpenFOAM*, que genera mallas hexaédricas, por lo que los solvers no están optimizados para trabajar con el tipo de mallas generadas directamente por *Netgen* y podrían generarse problemas de convergencia. Por su parte *Discretizer* está en fase de desarrollo, lo que lo hace todavía difícil de usar y de instalar, pero según su creador debería ser capaz de generar mallas complejas directamente en el formato *OpenFOAM*. Por lo general el problema de este tipo de códigos libres es que su desarrollo es muy lento y se hacen difíciles de instalar y usar. Por eso, como primera aproximación a la generación de mallas desde otros softwares usaremos *Salome* (http://www.salome-platform.org/), un software de ingeniería que si bien es distribuido gratuitamente, no corresponde a una iniciativa personal sino que fue desarrollado por EDF R&D (Electricité de France).

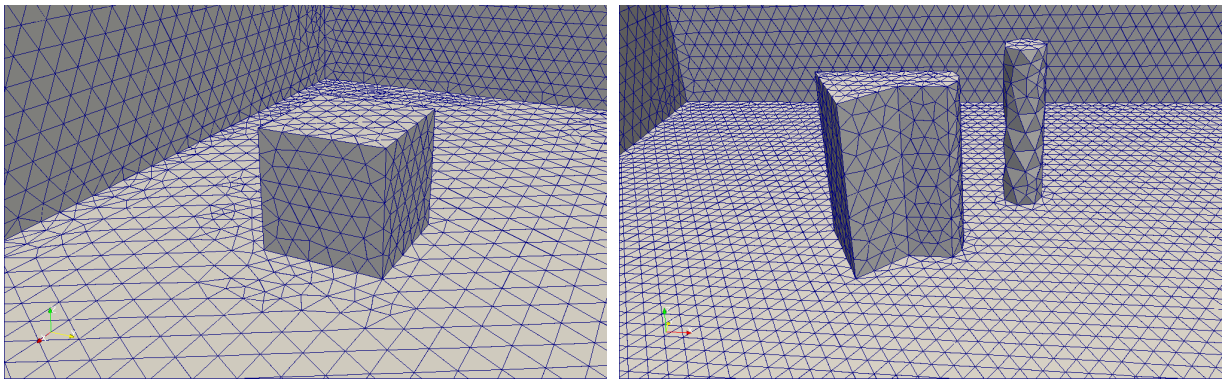
El uso de *Salome*, en particular en lo que a diseño 3D se refiere, es muy similar al de cualquier software de ingeniería mecánica, por lo que nos referiremos únicamente al procedimiento necesario para exportar la malla a *OpenFOAM*:

- crear una geometría 3D en *Salome*
- usar el comando *New Entity:Explode:faces* para generar las caras que definen el volumen (serán usadas posteriormente para definir las condiciones de borde en *OpenFOAM*). Eventualmente se le puede asignar a cada cara un nombre con el que identificarla posteriormente en *OpenFOAM* (ej: entrada, salida, techo, etc.)
- generar la malla. Por lo general en *Salome* los mejores resultados se obtienen usando mallado tetrahédrico. Configurar la resolución según lo deseado.
- una vez generada la malla, seleccionarla en el *Object Browser* y seleccionar *Create Groups from Geometry*. Este paso es fundamental para que la malla resultante identifique las diferentes superficies que vamos a considerar como condiciones de borde. Dentro de esta opción asignar en el recuadro *Elements: Geometry* cada una de las caras definidas anteriormente. No usar el recuadro *Nodes*.
- Realizado el paso anterior la malla está lista para ser exportada. Seleccionarla en el *Object Browser* y seleccionar *Export to UNV file*. El formato UNV es un formato universal de malla que *OpenFOAM* puede importar directamente mediante la herramienta *ideasUnvToFoam*.
- Para usar la herramienta *ideasUnvToFoam* debe existir en la misma carpeta el archivo *system* que caracteriza a las simulaciones *OpenFOAM*. Bastará entonces con ejecutar la orden *ideasUnvToFoam 'nombredelamalla.unv'* para que la herramienta cree la carpeta *constant* con la carpeta *polyMesh* conteniendo la malla. Si revisamos dentro de la

carpeta el archivo *boundary* observaremos los bordes que se han importado, los que por norma están siempre definidos como *patch*, pero podemos modificar el archivo para asignarles el tipo *wall* a los que corresponda hacerlo.

- el último paso consiste en completar los archivos que faltan para configurar la simulación (los que completan la carpeta *constant* y las condiciones iniciales). Al definir las condiciones de borde es importante asegurarse que el nombre que usemos para las superficies coincida con las importadas.

Como se observa en las figuras C.2a y C.2b con este procedimiento es posible generar rápidamente mallas para distintas geometrías. La gran ventaja que ofrece radica en que es posible trabajar fácilmente con geometrías complejas, integrar varios obstáculos, y controlar la calidad de la malla. Sin embargo, dado que *Salome* está optimizado para la generación de mallas tetrahédricas, serán éstas las que se generarán casi por defecto con este procedimiento (observar el mallado triangular de las figuras, en comparación con el cuadriculado del observado en las mallas de *blockMesh* (figuras C.1a y C.1b)).



(a) ejemplo malla geometría sencilla **(b)** ejemplo malla geometría compleja

Figure C.2: Resultados del mallado usando combinación *Salome-ideasUnvToFoam*

C.3 *snappyHexMesh*

snappyHexMesh es una herramienta incluida en las distribuciones de *OpenFOAM* desde la versión 1.5. Si bien no existe mucha documentación al respecto, se desprende de la información disponible que la herramienta utiliza un sistema de descomposición iterativa en bloques para aproximar geometrías complejas mediante mallas hexahédricas de gran precisión. Tal como la mayoría de las herramientas de *OpenFOAM*, *snappyHexMesh* se ejecuta a través de una serie de archivos de texto que definen las condiciones de trabajo. Si bien esto permite tener un control absoluto sobre los resultados del mallado, resulta poco intuitivo y difícil de utilizar, en especial considerando que prácticamente no existe documentación al respecto y el aprendizaje necesariamente debe realizarse sobre la marcha. Todas estas dificultades resultan sin embargo menores al comprobar los excelentes resultados que se obtienen al usar esta herramienta, pues permite realizar eficientemente el mallado de geometrías complejas, con absoluto control de las zonas de refinamiento, y completamente integradas a los solvers utilizados en *OpenFOAM*.

Dada la complejidad en su utilización presentamos a continuación un resumen del procedimiento necesario para usar esta herramienta, a manera de guía resumida.

snappyHexMesh utiliza como base un archivo con extensión .stl (estereolitografía), conteniendo la información geométrica del modelo que se quiere mallar. En este caso debe corresponder a la geometría del obstáculo que se pretende "extraer" del dominio (mallado inverso en dinámica de fluidos). Para crearlo es posible utilizar cualquiera de los softwares de diseño mecánico existentes hoy en día, pues la mayoría dispone de la alternativa de guardar en formato .stl. En este caso describiremos los pasos necesarios para crear un obstáculo usando *Salome*, y luego lo extenderemos al caso de varios obstáculos.

- El primer paso consiste en crear un volumen 3D de la geometría del obstáculo. Para ello la manera más sencilla es crear una superficie en el plano con el perfil del volumen, transformarla en una cara (seleccionar los trazos y aplicar *New Entity* → *Build* → *Face*) y luego extruirla (seleccionar la cara y aplicar *New Entity* → *Generation* → *Extrusion*). En el caso de geometrías complejas se pueden superponer varios volúmenes creados con el método anterior para generar una única entidad 3D usando *Operations* → *Boolean* → *Fuse* o → *Cut*.
- Por regla general las unidades en *Salome* son metros, por lo que al crear la geometría se debe tener en cuenta el dominio que se pretende utilizar. Además, es útil usar como referencia el origen *(0,0,0)*, de tal forma de hacerlo coincidir más adelante con el origen del dominio.
- Creado el obstáculo 3D que nos va a interesar extraer del mallado procedemos a exportarlo en formato stl. Para ello seleccionamos el volumen en el *ObjectBrowser*, y usamos *File* → *Export*, usando como tipo de archivo STL ASCII Files (.stl).
- Si trabajamos con geometrías que contemplen más de un obstáculo, por ejemplo varios edificios, existen dos alternativas. Exportar directamente el conjunto a formato .stl, y perder la posibilidad de individualizar cada geometría, al no poder aplicar condiciones de borde o de refinamiento de malla individuales a cada objeto, o crear un archivo .stl para cada volumen, y conservar la individualidad de cada obstáculo. En este caso

podemos seguir usando un único archivo *Salome*, pero deberemos seleccionar cada volumen y exportarlo individualmente a su archivo .stl correspondiente. Luego, a partir de estos archivos individuales creamos un solo archivo que los contenga a todos, copiando y pegando el contenido de cada archivo consecutivamente en un solo archivo .stl. Cada archivo individual tiene como formato: *solid ..... ensolid*. Al formar el archivo conjunto pegando el contenido de cada archivo es importante añadir a este formato el nombre de cada obstáculo. Por ejemplo: *solid edificio1.....endsolid edificio1*. De esta formar al crear el mallado *OpenFOAM* asignará a cada objeto un borde individual.

Definido el archivo .stl con el o los obstáculos que se pretenden introducir en el dominio de la simulación es necesario incorporarlo a la carpeta de archivos usada por *OpenFOAM* y crear el mallado. Este es el paso más importante del proceso, y el que en definitiva permite controlar la calidad del mallado final. Los pasos necesarios para realizarlo son los siguientes:

- Por lo general es recomendable usar como carpeta base una carpeta conteniendo los archivos generales de cualquier simulación (carpetas *constant*, *system*, *0*).
- El archivo .stl debe copiarse dentro de la carpeta *constant* en una nueva carpeta llamada *triSurface*.
- Dado que la herramienta *snappyHexMesh* usa como base una malla creada usando *blockMesh*, cuyo dominio contenga el obstáculo que se pretende insertar, antes de crear la malla final usando *snappyHexMesh* debemos usar *blockMesh* de la manera tradicional para crear una malla hexaédrica cuyas dimensiones definan el dominio total de la simulación. Es importante que el mallado de esta primera malla no sea muy refinado, pues el refinamiento se realizará con *snappyHexMesh*, pero es recomendable que el espaciamiento de las grillas *xyz* sea de tamaño similar.
- Ahora bien, un vez creada la malla base del dominio, usamos *snappyHexMesh* para insertar el obstáculo y refinar la malla. Para ello es necesario agregar un archivo llamado *snappyHexMeshDict* en la carpeta *system*. Dado que este es un archivo extenso, que contiene una serie de comandos, lo recomendable es usar una copia de alguno de los archivos que aparecen en los archivos del tutorial y modificarlo de forma que se ajuste a nuestros requisitos. Detallamos a continuación los aspectos más importantes de este archivo, aunque se recomienda remitirse al manual de usuario de *OpenFOAM* para más detalles.
- Lo primero, en el apartado *geometry*, es indicarle a *snappyHexMesh* el archivo .stl que vamos a usar como referencia geométrica, y las coordenadas del subdominio del dominio total que vamos a refinar (*refinementBox*). Se asigna además un nombre a la superficie completa, que idealmente debe corresponder con el nombre del archivo, para evitar confusiones. Aún cuando sean varios obstáculos, ese nombre es único.
- A continuación, en el apartado *castellatedMeshControls*, definiremos el control del mallado. Este apartado contempla muchas opciones, por lo que indicaremos únicamente las más importantes. En el apartado *refinementSurfaces* debemos indicar el nombre de la superficie usada para refinar la malla, que corresponde al nombre que hemos definido al principio. También ahí debemos indicar el nivel mínimo y máximo de refinación (el máximo se va a usar en los bordes y en las geometrías más complejas). En *refinementRegions* debemos indicarle al software que usaremos la región definida en

refinementBox como zona de refinación. En *locationInMesh* indicamos el punto usado como referencia en el archivo .stl. Idealmente este punto debe ser *(0,0,0)*, por lo que al definir la geometría en *Salome* es importante considerar que la referencia debe ser el punto *(0,0,0)* del dominio original.

- Dentro del apartado *snapControls* es importante definir *nSmoothPatch*, que va a controlar el número de iteraciones antes de encontrar una correspondencia con la geometría del obstáculo.
- Finalmente en el apartado *addLayersControls* debemos identificar cada uno de los bordes que hemos introducido con el archivo .stl. Es decir, aquí nombramos los diferentes obstáculos que hemos considerado, usando el nombre que les hemos asignado en el archivo .stl y anteponiéndoles el nombre de la superficie completa que hemos definido anteriormente, de tal forma que el nombre compuesto es de la forma *superficie_ obstáculo*.

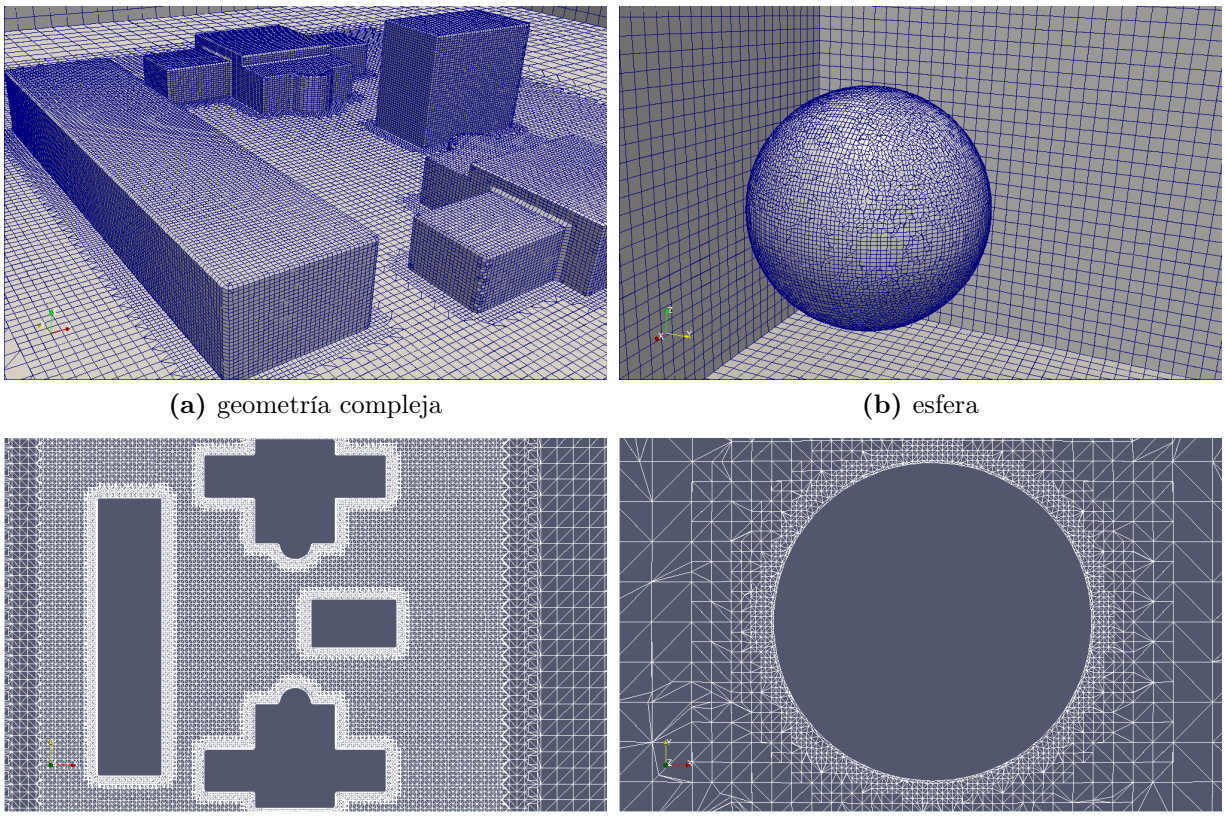
Son muchas más las opciones que considera este archivo de control, en particular en cuanto a la verificación y refinamiento de la malla, pero la mayoría aparecen comentadas en el mismo archivo, o están referidas en el manual.

Una vez creado y configurado el archivo *snappyHexMeshDict* ejecutamos la herramienta tal como si se tratara de un solver. Dependiendo de la complejidad de la malla el proceso puede demorar varios minutos, indicándosenos en todo momento la evolución del proceso y los pasos que va siguiendo el mallado.

El resultado del proceso será la creación de tres carpetas en la carpeta base del modelo, cada una con el nombre de un intervalo de tiempo según esté definido en el archivo de control de la simulación. En la última estarán los archivos correspondientes a la malla (la carpeta *polyMesh* y los archivos *cellLevel* y *pointLevel*). Será necesario copiar estos archivos a la carpeta *constant* y reemplazar los archivos allí existentes (antes es recomendable realizar una copia de la carpeta *polyMesh* original ahí existente, de manera que sirva de respaldo para futuros cambios). Como último paso para configurar la simulación es necesario revisar el archivo *boundary* dentro de la carpeta *polyMesh* que se ha creado, y verificar si las definiciones de los bordes creados corresponden con nuestros requerimientos, y sino modificarlos (bordes tipo wall, patch, etc). Además, será necesario finalmente definir los archivos de la carpeta *0*, de condiciones iniciales, de acuerdo a las nuevas superficies que se han creado.

Como se puede comprobar en las figuras C.3a, C.3b, C.3c y C.3d, los resultados que se obtienen al generar mallas usando este sistema son muy buenos. En particular el sistema permite refinar el mallado en zonas específicas del dominio, con especial atención en los bordes, en especial en geometrías con bordes rectos, como los que se aprecian en la figura C.3a. Tal como se observa en la figura C.3c, la malla se configura de tal forma de formar zonas de refinamiento en torno a los bordes, lo que facilita el cálculo numérico en zonas complejas y reduce la exigencia computacional en el resto del dominio. Además, *snappyHexMesh* permite integrar fácilmente en las geometrías zonas curvas, como la que se observa en la figura C.3b, que representa la malla generada en torno a una esfera. Tal como refleja la figura C.3d el contorno curvo se aproxima generando una sucesión de mallas hexaédricas en torno a él (los triángulos que se observan corresponden únicamente al efecto de visualización que se genera dado que la imagen es un corte del dominio 3D).

De esta forma, al poder integrar fácilmente geometrías con bordes rectos y curvos, y con distinto grado de detalle, la herramienta *snappyHexMesh* nos ofrece completa libertad en cuanto a la geometría de trabajo. Si a esto le agregamos que la malla generada está completamente optimizada para ser utilizada por los solvers de *OpenFOAM*, lo que comprobamos al ejecutar simulaciones usando las mallas generadas, estamos en condiciones de afirmar que la combinación de *Salome*, creando archivos .stl, y la herramienta *snappyHexMesh* para generar la malla, representa la mejor opción para trabajar con geometrías complejas en *OpenFOAM*. De hecho, dados los resultados obtenidos usando estas herramientas, y considerando que ambas son de uso libre, podría decirse que esta combinación representa una solución definitiva al problema de la generación de mallas para *OpenFOAM*.



(a) geometría compleja **(b)** esfera

(c) detalle geometría compleja **(d)** detalle esfera

Figure C.3: Resultados del mallado usando combinación *Salome-snappyHexMesh*

C.4 Mallas Topográficas

Por lo general la superficie del terreno que interesa incluir en la simulación presenta un gran grado de complejidad. Su mallado debe ser abordado con especial cuidado, dado que al representar la condición de borde inferior del modelo tendrá un efecto directo sobre los resultados. En particular en el caso de los rajos mineros interesa estudiar cómo afecta la geometría del terreno la evolución del flujo, por lo que la característica irregular de éste adquiere una especial importancia. Por lo general se cuenta con datos de elevación, es decir coordenadas XYZ, en donde la tercera columna indica la elevación del terreno sobre una particular ubicación. Por lo tanto en el caso del terreno no es posible crear directamente un modelo virtual usando algún software CAD, dado que la geometría es muy variable, y no se puede representar de manera sencilla mediante el dimensionado. Será necesario entonces transformar directamente los datos de coordenadas en una malla 3D. Afortunadamente, es posible abordar este problema usando *snappyHexMesh*, introduciendo algunas modificaciones al procedimiento revisado anteriormente. En particular se hace necesario introducir un paso intermedio, necesario para crear a partir de las coordenadas XYZ un archivo en el formato utilizado por *snappyHexMesh* (.stl, estereolitografía). Si bien existen varios softwares capaces de hacer esta conversión, en particular en este caso se optó por *GlobalMapper*, software topográfico de amplia difusión, que abre además la posibilidad de importar terrenos en diferentes tipos de formatos. El terreno a modelar se carga en *GlobalMapper*, y luego se selecciona la opción de exportar en formato stl (para más referencias revisar por ejemplo Pedruelo (2009)). Existen diferentes opciones configurables, siendo la más importante el espaciamiento de la malla, que va a controlar la calidad de la malla que generemos y el peso del archivo final. Una vez generado el archivo stl procedemos con él tal como se describió en la sección destinada a la generación de mallas usando *snappyHexMesh*. El único aspecto importante a tener en cuenta es que dado que en este caso la superficie stl va a reemplazar el borde inferior de nuestro dominio es necesario que ésta intersecte el volumen del dominio inicial (malla generada originalmente con *blockMesh*), y lo divida en dos. Sólo de esta forma nos aseguramos que la generación de la nueva malla elimine el borde inferior y lo reemplace por el terreno. Por lo general los resultados obtenidos mediante este procedimiento de generación de mallas topográficas son muy buenos, lográndose de manera económica y rápida una malla de buena calidad completamene integrada a los requerimientos de *OpenFOAM*.

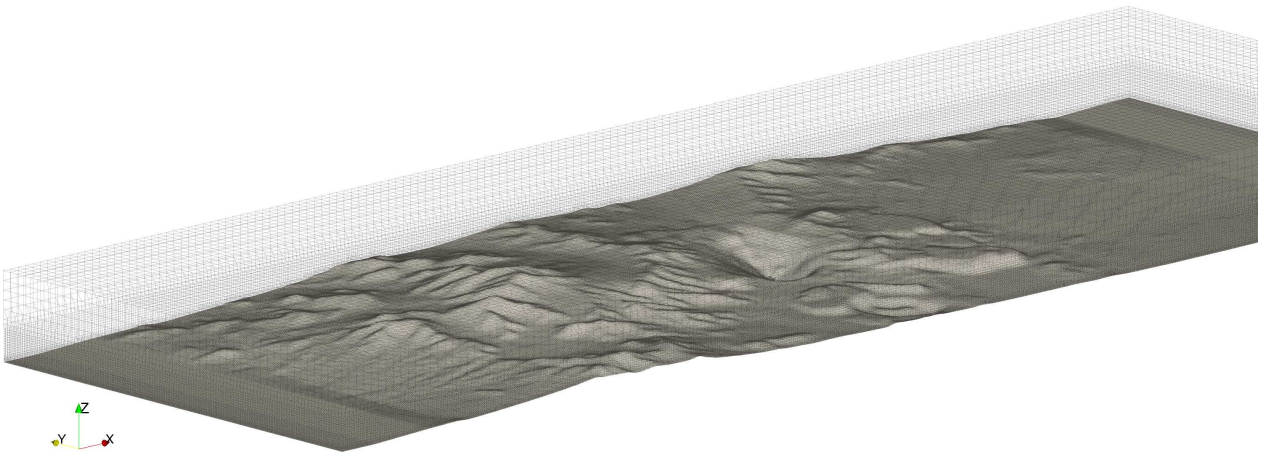


Figure C.4: Ejemplo de topografía compleja mallada usando *snappyHexMesh*