UNIVERSITA' DEGLI STUDI DI NAPOLI "FEDERICO II"

DETEC

DIPARTIMENTO DI ENERGETICA, TERMOFLUIDODINAMICA APPLICATA E CONDIZIONAMENTI AMBIENTALI

DOTTORATO DI RICERCA IN
INGEGNERIA AEROSPAZIALE - XXI Ciclo

High level languages analysis of quasi-segregated Navier-Stokes solvers and its application to 3D reactive flows

Tutors:

Prof. Carlo Meola

Prof. Giuseppe De Felice

Candidate:

Valerio Grazioso

November 2008

Contents

Chapter 1

Introduction

In order to design a computational fluid dynamics code oriented to the simulation of turbulent and chemically reacting flows, it is convenient to adopt high level algebraical languages (like Python, Scilab, Octave, etc.) and in particular Matlab®. This choice is made considering the opportunities that this kind of programming languages offer in terms of easy user interfacing, fast libraries and packages integration (i.e. NAG, LAPACK, UMFPACK, etc.) and increasing usage of natively multi-threaded functions. In comparison with low level programming languages (like C or Fortran) they result to be easier to use and more portable across platforms because of their strong abstraction from the details of the computer. Moreover high level algebraical languages have also an easy parallelization capability by means of specifically designed toolboxes (MPI based) and packages (like Star-P®).

Following such considerations and with the purpose in mind of designing a flexible code (sometimes referred to as *proto-code*) with turbulent and reactive capabilities and oriented to the study of new mathematical and numerical models and to the development or optimization of new numerical algorithms, the first idea has been to import (translate) in high level algebraical environment a previously developed Fortran 77 code as kernel of the new code. Unfortunately this attempt has turned out to be more tricky than aspected for two reasons. A Fortran 77 code can have a strong not structured form with intensive use of unstructured *goto* statements (not supported in high level languages such Matlab) as it can be argued by the observation of its “main” flow chart’s *spaghetti-like* structure in figure (1.1))

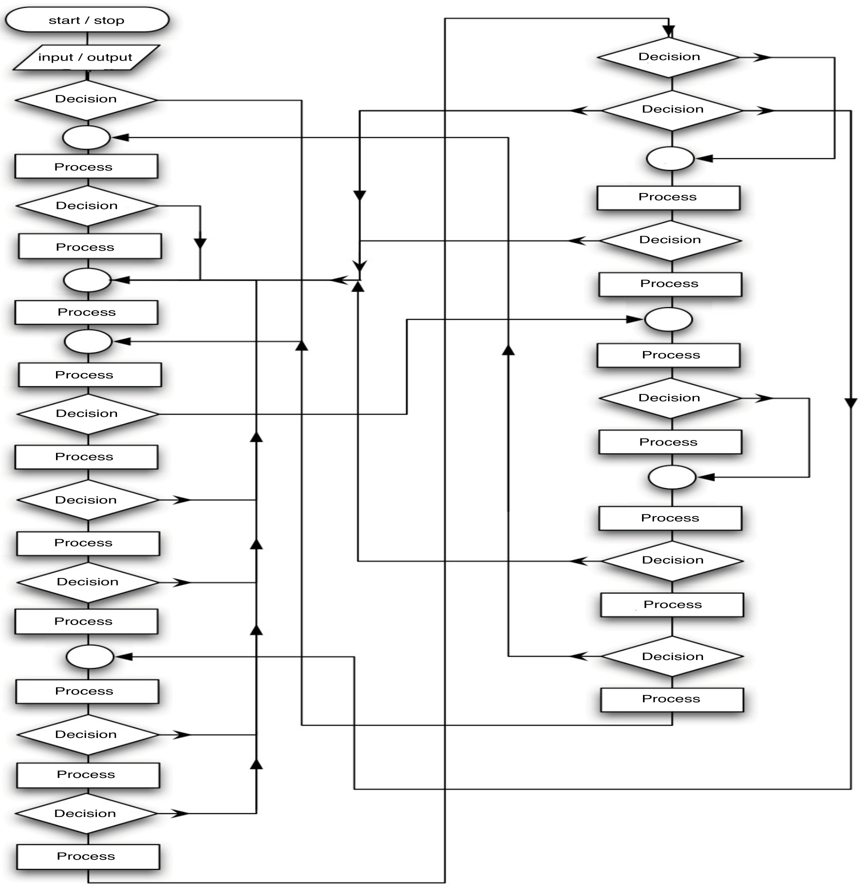


Figure 1.1: **Fortran 77 code flow chart.**

Moreover in Fortran 77 codes there is an extensive use of *common blocks* that, also having a Matlab counterpart in the *global variables*, makes very difficult to use the *mex-files* interfacing procedure between Matlab and pre-compiled subroutines written in other other languages like Fortran 77. This interfacing procedure could be particularly helpful just when it is impossible or not so convenient to rewrite in Matlab the original Fortran subroutines.

This effort of using a pre-existent structure as a model for the kernel design has been abandoned for its not so straight-forward realization and for the not so efficient runtime execution most of all as a consequence of the not efficient translation of linear solvers.

Taking this experience into account the proto-code design has been reoriented on new directions: to write from scratch a new numerical code called PRIN-3D (*PR*oto-code for *I*nternal flows modeled by *N*avier-Stokes equations in *3-D*imensions) tailored to the general structure of the mathematical models that we want to solve (in particular the incompressible and the slightly compressible Navier-Stokes model) but also flexible enough to easy implement new models exploiting the fast built in functions of high level algebraical languages. The focus is also on the introduction of advanced numerical solvers with pressure segregation by means of preconditioning techniques, and to the study of different linear solvers (iterative and direct). Also a new convective numerical scheme is introduced.

Chapter 2

Mathematical models

The main differential model referred to in this work is described by the following equations, every other model will simply be derived from the latter adopting certain hypothesis:

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u_i)}{\partial x_i} = 0 \tag{2.1}$$

$$\frac{\partial(\rho u_j)}{\partial t} + \frac{\partial(\rho u_i u_j)}{\partial x_i} = -\frac{\partial p}{\partial x_j} + \frac{\partial}{\partial x_i}\left[\mu(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j}) - \frac{2}{3}\mu\frac{\partial u_k}{\partial x_k}\delta_{ji}\right] \tag{2.2}$$

$$\frac{\partial(\rho h_0)}{\partial t} + \frac{\partial(\rho u_i h_0)}{\partial x_i} = \dots$$

$$\dots = \frac{\partial p}{\partial t} + \frac{\partial}{\partial x_i}\left[\frac{\lambda}{c_p}\frac{\partial h}{\partial x_i} + \sum_{k=1}^{N}(\rho D_k - \frac{\lambda}{c_p})h_k\frac{\partial Y_k}{\partial x_i}\right] + \Phi \tag{2.3}$$

$$\frac{\partial(\rho Y_m)}{\partial t} + \frac{\partial(\rho u_i Y_m)}{\partial x_i} = \frac{\partial}{\partial x_i}\left(\rho D_m \frac{\partial Y_m}{\partial x_i}\right) + w_m$$

$$(m = 1, 2, \dots, N). \tag{2.4}$$

Here (2.1) is the continuity equation and (2.2) is the momentum equation in which terms, that would be normally neglected due to the incompressibility of the flow, are now taken into account in a Slightly Compressibile Model that will be described later. The equation (2.3) decribes the evolution of the specific enthalpy h_0 governed by advection, diffusion and production terms that depend on the species' mass fractions Y_m, defined as usual. The m equations (2.4) are essentially describing the transport phenomena associated with chemical reactions (modeled by the production term w_m in (2.4)) that occur in every given control volume of the domain and they will be discussed in detail in the slightly compressibile model section.

2.1 Incompressible Flow

The incompressible Navier-Stokes model is already closed if only equations (2.1) and (2.2) are considered, since ρ is a constant and there's no need for the equations (2.3) and (2.4) (that essentially are needed to update such variable).

$$\frac{\partial U_i}{\partial x_i} = 0 \tag{2.5}$$

$$\frac{\partial U_j}{\partial t} + \frac{\partial U_i U_j}{\partial x_i} = -\frac{\partial p}{\partial x_j} + \nu \frac{\partial^2}{\partial x_i \partial x_i} U_j \tag{2.6}$$

with

$$p = P/\rho$$

Where P is the sum of the usual pressure and mass force potentials. Fluid is assumed to be Newtonian (i.e. a linear relation holds between the non isotropic parts of the stress and rate-of-strain tensors and such proportionality is expressed by means of a single, constant μ termed viscosity of the fluid [4]). There are only 4 unknows left to be solved: u, v, w and p. It is recalled here that this model has an important drawback: since temperature and density are preassigned constants, from any given constitutive gas equation, for example

$$P = \rho RT$$

it could be argued that the pressure field ought be a constant. On the contrary, in a typical incompressibile solver, pressure is the essential unknown to be computed. It has a complex physical meaning being just a scalar field whose gradient guarantees that the velocity field respects the indivergence constraint expressed by (2.5). It could be argued from a dimensional analysis that it could be a fairly good approximation to neglect temperature and density gradients. This doesn't apply at all for the pressure gradients which are very effective on the momentum field while having a weak influence in the energy equations.

A better understanding of the role of the pseudo-pressure gradient in equation (2.2), constrained by (2.1), is given by applying the Inverse Theorem of the Vector Field Calculus providing a closure for the vector field $\frac{\partial u_j}{\partial t}$ [13]. The latter has a given divergence (zero), a given curl (known by taking the curl of equation (2.6)) and given boundary conditions on every point of the domain boundary.

2.2 Incompressible Turbulent Flow Modeling

By taking (any) statistical average $\langle \cdot \rangle$ of equations (2.5)-(2.6) it is possible to obtain the mean flow governing equations

$$\frac{\partial \langle U_i \rangle}{\partial x_i} = 0 \tag{2.7}$$

$$\begin{aligned} \frac{\partial \langle U_j \rangle}{\partial t} + \frac{\partial}{\partial x_i} \left(\langle U_i \rangle \langle U_j \rangle \right) &= \dots \\ \dots &= \frac{1}{\rho} \frac{\partial \langle P \rangle}{\partial x_j} + \frac{1}{\rho} \frac{\partial}{\partial x_i} \left[\mu \left(\frac{\partial \langle U_j \rangle}{\partial x_i} + \frac{\partial \langle U_i \rangle}{\partial x_j} \right) - \rho \langle u_i \, u_j \rangle \right]. \end{aligned} \tag{2.8}$$

Equation (2.8) can be rewritten as

$$\begin{aligned} \frac{\partial \langle U_j \rangle}{\partial t} + \frac{\partial}{\partial x_i} \left(\langle U_i \rangle \langle U_j \rangle \right) &= \cdots \\ \cdots = -\frac{1}{\rho} \frac{\partial}{\partial x_i} \left[\mu \left(\frac{\partial \langle U_j \rangle}{\partial x_i} + \frac{\partial \langle U_i \rangle}{\partial x_j} \right) - \rho \langle u_i \, u_j \rangle \right] - \cdots \\ \cdots - \frac{1}{\rho} \frac{\partial}{\partial x_i} \left[\frac{2}{3} \rho k \delta_{ij} - \left(\langle P \rangle + \frac{2}{3} \rho \, k \right) \delta_{ij} \right] \end{aligned} \tag{2.9}$$

and considering that the dynamic viscosity coefficient is constant for incompressible flows ($\mu = cost$) and that the isotropic part of the Reynolds stress tensor can be included in the pressure term, redefining a new pressure-like variable (or divergence corrector) as $\mathcal{P} = \langle P \rangle + \frac{2}{3} \rho k$ and adopting the *turbulent-viscosity model* which is stated as follows

$$-\rho \langle u_i \, u_j \rangle + \frac{2}{3} \rho k \delta_{ij} = \rho \nu_T \left(\frac{\partial \langle U_j \rangle}{\partial x_i} + \frac{\partial \langle U_i \rangle}{\partial x_j} \right) \tag{2.10}$$

(isotropic but non homogeneous linear relation between the deviatoric Reynolds stress and mean rate of strain) this leads to

$$\frac{\partial\langle U_i\rangle}{\partial x_i} = 0 \tag{2.11}$$

$$\frac{\partial\langle U_j\rangle}{\partial t} + \frac{\partial}{\partial x_i}\left(\langle U_i\rangle\langle U_j\rangle\right) = \dots$$
$$\dots = -\frac{1}{\rho}\frac{\partial \mathcal{P}}{\partial x_j} + \nu\frac{\partial^2\langle U_j\rangle}{\partial x_i\partial x_i} + \frac{\partial}{\partial x_i}\left[\nu_T\left(\frac{\partial\langle U_j\rangle}{\partial x_i} + \frac{\partial\langle U_i\rangle}{\partial x_j}\right)\right]. \tag{2.12}$$

Turbulent viscosity ν_T is still unknown and has to be updated. A typical two-equation model, the K-Epsilon model in its low Reynolds form, will be used.

Standard $k - \epsilon$ model

A commonly used two-equation turbulence model is the $k - \epsilon$ model. The partial differential equation are derived for kinetic energy of turbulence (k), and the dissipation of turbulence (ϵ), where

$$k = \frac{1}{2}\langle u_i u_i\rangle$$

and

$$\epsilon = \nu_T\left\langle\frac{\partial u_i}{\partial x_j}\frac{\partial u_i}{\partial x_j}\right\rangle$$

The standard $k - \epsilon$ two equation model is expressed by the turbulent kinetic equation

$$\rho\frac{\partial k}{\partial t} + \rho\langle U_i\rangle\frac{\partial k}{\partial x_i} = \nu\frac{\partial^2}{\partial x_i\partial x_i}k + \frac{\partial}{\partial x_i}\left[\mu_T/\sigma_k\frac{\partial k}{\partial x_i}\right] + P_k - \rho\epsilon \tag{2.13}$$

and the dissipation rate equation

$$\rho \frac{\partial \epsilon}{\partial t} + \rho \langle U_i \rangle \frac{\partial \epsilon}{\partial x_i} = \nu \frac{\partial^2}{\partial x_i \partial x_i} \epsilon + \frac{\partial}{\partial x_i} \left[\mu_T / \sigma_\epsilon \frac{\partial \epsilon}{\partial x_i} \right] + \\ + C_{\epsilon 1} P_k \epsilon / k - C_{\epsilon 2} \rho \epsilon^2 / k \quad (2.14)$$

where P_k is the production of turbulence defined as $P_k = \tau_{ij} \frac{\partial \langle U_i \rangle}{\partial x_j}$ with

$$\tau_{ij} = \mu_T \left(\frac{\partial \langle U_i \rangle}{\partial x_j} + \frac{\partial \langle U_j \rangle}{\partial x_i} - \frac{2}{3} \delta_{ij} \frac{\partial \langle U_i \rangle}{\partial x_i} \right) - \frac{2}{3} \rho k \delta_{ij} \quad (2.15)$$

Low Reynolds number $k - \epsilon$ model

The difficulty with the standard $k-\epsilon$ model introduced in the previous section is that the equations become numerically unstable when integrated to the wall [10]. In order to overcome such problem and improve the capability of the standard $k-\epsilon$ model, several modifications are introduced. The resulting formulation is known as the *low-Reynolds number $k-\epsilon$ model* and the first one was developed by Jones and Launder and subsequently it has been modified by several inverstigators. The primary modifications introduced by Jones and Launder were to include turbulence Reynolds number dependent dumping functions f_1, f_2 and f_μ within the standard $k - \epsilon$ model. Furthermore, additional terms L_k and L_ϵ were added to the equations to account for the dissipation processes which may not be isotropic. Thus the low-Reynolds number $k - \epsilon$ equation is written as

$$\rho \frac{\partial k}{\partial t} + \rho \langle U_i \rangle \frac{\partial k}{\partial x_i} = \nu \frac{\partial^2}{\partial x_i \partial x_i} k + \frac{\partial}{\partial x_i} \left[\mu_T / \sigma_k \frac{\partial k}{\partial x_i} \right] + P_k - \rho \epsilon + L_k \quad (2.16)$$

$$\rho \frac{\partial \epsilon}{\partial t} + \rho \langle U_i \rangle \frac{\partial \epsilon}{\partial x_i} = \nu \frac{\partial^2}{\partial x_i \partial x_i} \epsilon + \frac{\partial}{\partial x_i} \left[\mu_T / \sigma_\epsilon \frac{\partial \epsilon}{\partial x_i} \right] + \\ + C_{\epsilon 1} f_1 P_k \epsilon / k - C_{\epsilon 2} f_2 \rho \epsilon^2 / k + L_\epsilon \quad (2.17)$$

where the tubulent viscosity is now computed according to

$$\mu_T = \rho f_\mu \, c_\mu \frac{k^2}{\epsilon} \tag{2.18}$$

A number of selected $k - \epsilon$ models are provided in tables (2.1)-(2.2)

Model	f_1	f_2	f_μ
Standard	1.0	1.0	1.0
Jones-Launder	1.0	$1 - .3exp(-Re_T^2))$	$exp\left[\frac{-2.5}{1+0.02Re_T}\right]$
Hoffman	1.0	$1 - .3exp(-Re_T^2))$	$exp\left[\frac{-1.75}{1+0.02Re_T}\right]$
Nagano-Hishida	1.0	$1 - .3exp(-Re_T^2))$	$\left[1 - exp\left(-Re_T/26.5\right)\right]^2$

Table 2.1: **low Reynolds $k - \epsilon$ models. (source [10])**

Model	L_k	L_ϵ	C_μ	$C_\epsilon 1$	$C_\epsilon 2$	σ_k	σ_ϵ
Standard	0	0	0.09	1.44	1.92	1.0	1.30
Jones-Launder	$-2\mu\left(\frac{\partial\sqrt{k}}{\partial x_j}\right)^2$	$2\mu\nu_T\left(\frac{\partial^2 u_i}{\partial x_j^2}\right)^2$	0.09	1.44	1.92	1.0	1.30
Hoffman	$-\frac{\nu}{y_w}\frac{\partial k}{\partial y_w}$	0	0.09	1.81	2.0	2.0	3.0
Nagano-Hishida	$-2\nu\left(\frac{\partial\sqrt{k}}{\partial y_w}\right)$	$\nu\nu_T\left(1 - f_\mu\right)\left(\frac{\partial^2 u}{\partial y_w^2}\right)^2$	0.09	1.45	1.90	1.0	1.30

Table 2.2: **low Reynolds $k - \epsilon$ models. (source [10])**

2.3 Slightly Compressibile Flow

The Slightly Compressibile model is particulary effective in simulating variable density flows but its usage is restricted to low Mach flows (i.e. $(0.7 - 0.8)$). No shocks can be captured by this model and pressure is still lacking thermodynamic relevance, and, just like in the incompressible case, it simply has the role of the velocity field's divergence corrector. The equations for such model are (2.24), (2.25), (2.26), and (2.27). The simulation of low Mach, laminar or turbulent reacting flows will be discussed in this section and, nonetheless, the incompressible Navier Stokes solver structure will be preserved [1]. The logical steps leading to the Flamelet model which is used to solve equations (2.26) and (2.27) separately from (2.24) and (2.25) will be also explained.

2.3.1 Complete Mathematical Model

Density is now a fully time and space variable quantity and it is given by the following expression in the hypothesis of ideal gas mixtures

$$\rho = \frac{pW}{R^0 T} \tag{2.19}$$

For low Mach flows high changes in pressure barely determine a sensibile change in the density field and this is the main reason why in incompressible flows pressure looses its thermodynamic role. In *low speed* reacting flows density changes are still not determined by changes in pressure but rather by changes in temperature and in the chemical composition of the gas mixture. Changes in pressure will still be significant in the momentum equation (2.25) but, as far as every other equation is concerned, they can be neglected and it will be considered thermochemically constant.

[1]See the section 3.4, dedicated to the numerical solving techique of Slightly Compressibile Flows

The molecular weight of the gas mixture W can be calculated with given molar fractions of all the single species

$$W = \sum_{m=1}^{N} (X_m W_m) \tag{2.20}$$

where N is the total number of chemical species and X_m and W_m are molar fraction and molecular weight of the m^{th} species. Alternatively, it is possible to rewrite equation (2.20) in terms of mass fraction Y_m and molar fraction X_m considering that

$$Y_m = \frac{X_m W_m}{W} \tag{2.21}$$

and

$$\sum_{m=1}^{N} X_m = 1$$

obtaining

$$W = \left[\sum_{m=1}^{N} \frac{Y_m}{W_m} \right]^{-1}. \tag{2.22}$$

In this way equation (2.19) can be restated as

$$\rho = \frac{p}{R^0 T \sum_{m=1}^{N} \frac{Y_m}{W_m}}. \tag{2.23}$$

As previously explained, the complete set of equations (here rewritten) is comprehensive of continuity and momentum equations, including energy and N species transport equations [20]:

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u_i)}{\partial x_i} = 0 \tag{2.24}$$

$$\frac{\partial\left(\rho u_j\right)}{\partial t}+\frac{\partial\left(\rho u_i u_j\right)}{\partial x_i}=\cdots$$
$$\cdots=-\frac{\partial p}{\partial x_j}+\frac{\partial}{\partial x_i}\left[\mu\left(\frac{\partial u_j}{\partial x_i}+\frac{\partial u_i}{\partial x_j}\right)-\frac{2}{3}\mu\frac{\partial u_k}{\partial x_k}\delta_{ji}\right] \quad (2.25)$$

$$\frac{\partial\left(\rho h_0\right)}{\partial t}+\frac{\partial\left(\rho u_i h_0\right)}{\partial x_i}=\cdots$$
$$\cdots=\frac{\partial p}{\partial t}+\frac{\partial}{\partial x_i}\left[\frac{\lambda}{c_p}\frac{\partial h}{\partial x_i}+\sum_{k=1}^{N}\left(\rho D_k-\frac{\lambda}{c_p}\right)h_k\frac{\partial Y_k}{\partial x_i}\right]+\Phi \quad (2.26)$$

$$\frac{\partial(\rho Y_m)}{\partial t}+\frac{\partial(\rho u_i Y_m)}{\partial x_i}=\frac{\partial}{\partial x_i}\left(\rho D_m\frac{\partial Y_m}{\partial x_i}\right)+w_m \quad (m=1,\ldots,N). \quad (2.27)$$

In order to have these equations in this form, a set of assumptions are taken into account, apart from the ordinary ones being newtonian viscosity, Fourier's and Fick's diffusion laws and vibrational equilibrium (that allows to consider the specific heat capacity of every species just as a function of temperature). In the momentum equation terms regarding the bulk viscosity and mass forces have been dropped out. Moreover, in the energy equation some terms have been neglected like volume forces work, radiative term, and Dufour (diffusive-thermometric) term. In the species' equations Soret's thermo-diffusive terms and the pressure gradient contribution to diffusion have been dropped out; also binary diffusion coefficients D_i of species i in a background prevalent gas (like nitrogen in the air) are assumed instead of multicomponent diffusion coefficients D_{ij} (of species i in species$j$). It is also important to know that in the energy equation h and h_0 are respectively static enthalpy and stagnation enthalpy and that the energy diffusive flux is the sum of two contributions

$$\lambda\frac{\partial T}{\partial x_i}+\sum_{k=1}^{N}\rho D_k h_k\frac{\partial Y_k}{\partial x_i}.$$

The first is given by the conduction heat transfer according to the Fourier law, and can be rewritten (considering the specific enthalpy dependancy by the mass fractions and by every single species' specific enthalpy) as

$$\lambda \frac{\partial T}{\partial x_i} = \frac{\lambda}{c_p}\frac{\partial h}{\partial x_i} - \frac{\lambda}{c_p}\sum_{k=1}^{N} h_k \frac{\partial Y_k}{\partial x_i}$$

whereas the second one is a consequence of the Fick's mass diffusion transport law that brings an energy contribution proportional to $\rho D_k h_k \frac{\partial Y_k}{\partial x_i}$.

In the momentum equation, Φ is the viscous dissipation function that models the irreversible energy transformation from kinetic to internal energy and has the expression

$$\Phi = \frac{\partial}{\partial x_i}\left\{\mu\left[u_k\left(\frac{\partial u_k}{\partial x_i} + \frac{\partial u_i}{\partial x_k}\right) - \frac{2}{3}u_i\frac{\partial u_k}{\partial x_k}\right]\right\}.$$

In the species equation (2.27) w_m is the source term

$$w_m = W_m \sum_{k=1}^{N} \Delta\nu_{mk}\omega_k$$

where ω_k is the reaction rate of the k^{th} reaction ($k = 1, 2, \ldots, M$ and M is the number of reactions). The symbolic reaction expression is

$$\sum_{m=1}^{N} \nu'_{m,k} M_m \rightleftharpoons \sum_{m=1}^{N} \nu''_{m,k} M_m \qquad (2.28)$$

and it takes into account the direct and inverse reaction steps so that to be referred to as *reversible reactions.* It is possible to adopt another symbolism when both reaction steps are separately considered

$$\sum_{m=1}^{N} \nu'_{m,k} M_m \rightarrow \sum_{m=1}^{N} \nu''_{m,k} M_m \qquad (2.29)$$

and in this case they are called *elementary reactions steps*, and $2M$ reactions must be considered. In (2.28) e (2.29) $\nu'_{m,k}$ e $\nu''_{m,k}$ represent the stoichiometric coefficients of species m as reactant and product in the k^{th} reaction, and $\Delta\nu_{m,k} = \nu''_{m,k} - \nu'_{m,k}$. There is also the definition of *molecularity* of direct and inverse reaction steps (it is the number of molecules needed in order for the reaction to take place)

$$m_k = \sum_{m=1}^{N} \nu'_{m,k} \qquad\qquad n_k = \sum_{m=1}^{N} \nu''_{m,k} \tag{2.30}$$

Considering equation (2.29), the reaction rate of k^{th} elementary reaction step can be expressed by Arrhenius law as

$$\omega_k = B_k T^{\alpha_k} \exp(-\frac{E_k}{R^0 T})\rho^{m_k} \prod_{m=1}^{N} (\frac{Y_m}{W_m})^{\nu'_{m,k}} \qquad (k = 1, 2, .., M) \tag{2.31}$$

where B_k, α_k e E_k can be assumed as constants (actually they should be considered as piecewise constants in different temperature intervals).

If form (2.28) is considered, the k^{th} reaction rate is the difference between reaction rates of direct and inverse reaction steps, that are linked together (when vibrational equilibrium hypothesis is assumed) by means of the equilibrium constant

$$K_{p,k} = \exp(-\frac{1}{R^0 T} \sum_{m=1}^{N} \Delta\nu_{m,k} W_m \mu_m^0) \tag{2.32}$$

where μ_m^0 is the chemical potential per unit mass of m species at standard pressure p^0. So the reaction rate of a reversible reaction is

$$\omega_k = B_k T_k^\alpha \exp(-\frac{E_k}{R^0 T})\rho^{m_k} \prod_{m=1}^{N} (\frac{Y_m}{W_m})^{\nu'_{m,k}}$$
$$\left[1 - \frac{(pW/p^0)^{n_k - m_k}}{K_{p,k}} \prod_{m=1}^{N} \frac{Y_m}{W_m}^{\Delta\nu_{m,k}}\right] \qquad (k = 1, 2, .., M) \tag{2.33}$$

Moreover, when considering simple systems, like combustion of methane in air, the number of reactions M range from 18 to 128 and the number of species N can vary from 15 to 39 according to the desired level of accuracy.

For the application of equations (2.24) (2.25) (2.26) (2.27) to reacting flows it is convenient to consider other approximations. It is possible to assume diffusion coefficients all equals to a single one

$$D_m = D \qquad (m = 1, 2, ..., N) \tag{2.34}$$

this assumption is particularly wrong when hydrogen is present as one of the species because of its high mobility, but is not so crucial when Reynolds numbers become so higher that molecular transports becomes less dominant with respect to convective ones. Introducing the Prandtl, Schmidt and Lewis adimensional numbers

$$Pr = \frac{c_p \mu}{\lambda}, \qquad Sc = \frac{\mu}{\rho D}, \qquad Le = \frac{Sc}{Pr} \tag{2.35}$$

it is possible to rewrite the diffusive term part that involves concentration gradients as follows

$$\frac{\partial}{\partial x_i}\left[\sum_{k=1}^{N}\left(\rho D_k - \frac{\lambda}{c_p}\right) h_k \frac{\partial Y_k}{\partial x_i}\right] = \frac{\partial}{\partial x_i}\left[\mu(\frac{1}{Sc} - \frac{1}{Pr})\sum_{k=1}^{N} h_k \frac{\partial Y_k}{\partial x_i}\right] \tag{2.36}$$

and for most of the gas species, with the exception of hydrogen, it results that $Le \approx 1$ that is to say $Pr \approx Sc$. This further assumption makes it possible to restate equations (2.27) (2.26) obtaining

$$\frac{\partial\left(\rho h_0\right)}{\partial t}+\frac{\partial\left(\rho u_i h_0\right)}{\partial x_i}=\frac{\partial p}{\partial t}+\frac{\partial}{\partial x_i}\left[\frac{\mu}{\sigma}\frac{\partial h}{\partial x_i}\right]+\Phi \tag{2.37}$$

$$\frac{\partial(\rho Y_m)}{\partial t}+\frac{\partial(\rho u_i Y_m)}{\partial x_i}=\frac{\partial}{\partial x_i}\left(\frac{\mu}{\sigma}\frac{\partial Y_m}{\partial x_i}\right)+w_m \quad (m=1,2,\ldots,N). \tag{2.38}$$

The final system of equations is now composed by equations (2.24) (2.25) (2.37) (2.38) that is an $N+5$ system in the $N+5$ variables u_j $(j=1,2,3), p, h_0, Y_m$ $(m=1,2,\ldots,N)$. The static enthalpy of the mixture is $h=h_0-u_k u_k/2$ and density can be calculated from thermal state equation (2.23). Temperature is needed in such equation and in the evaluation of reaction rates and diffusion coefficients. This system is theoretically closed.

To take into account fluctuations of the thermo-fluid-dynamc field due to turbulent behavior it is convenient to use a Favre's mean averaged form of these equations, which is shown below

$$\frac{\partial\overline{\rho}}{\partial t}+\frac{\partial\left(\overline{\rho}\tilde{u_i}\right)}{\partial x_i}=0 \tag{2.39}$$

$$\begin{aligned}\frac{\partial\left(\overline{\rho}\tilde{u_j}\right)}{\partial t}+\frac{\partial\left(\overline{\rho}\tilde{u_i}\tilde{u_j}\right)}{\partial x_i}&=-\frac{\partial\overline{p}}{\partial x_j}+\\&+\frac{\partial}{\partial x_i}\left[(\mu+\mu_t)\left(\frac{\partial\tilde{u_j}}{\partial x_i}+\frac{\partial\tilde{u_i}}{\partial x_j}\right)-\frac{2}{3}(\mu+\mu_t)\frac{\partial\tilde{u_k}}{\partial x_k}\delta_{ji}-\frac{2}{3}\overline{\rho}\tilde{k}\delta_{ji}\right]\end{aligned} \tag{2.40}$$

$$\frac{\partial\left(\overline{\rho}\tilde{h_0}\right)}{\partial t}+\frac{\partial\left(\overline{\rho}\tilde{u_i}\tilde{h_0}\right)}{\partial x_i}=\frac{\partial\overline{p}}{\partial t}+\frac{\partial}{\partial x_i}\left[(\frac{\mu}{\sigma}+\frac{\mu}{\sigma_0})\frac{\partial h}{\partial x_i}\right]+\overline{\Phi} \tag{2.41}$$

$$\begin{aligned}\frac{\partial\left(\overline{\rho}\tilde{Y_m}\right)}{\partial t}+\frac{\partial\left(\overline{\rho}\tilde{u_i}\tilde{Y_m}\right)}{\partial x_i}&=\ldots\\ \ldots&=\frac{\partial}{\partial x_i}\left[(\frac{\mu}{\sigma}+\frac{\mu}{\sigma_m})\frac{\partial\tilde{Y_m}}{\partial x_i}\right]+\overline{w_m} \qquad (m=1,2,\ldots,N)\end{aligned} \tag{2.42}$$

but these equations (solved with models like $k - \epsilon$) brings critical issues of thermochemical closure regarding the determination $\overline{w_m}$ and $\overline{\rho}$. First of all, it is convenient to note that instantaneous density and production rates are strongly nonlinearly dependent from all of the problem's variables so that, if the usual technique is followed, that is to split every variable in its mean value and its fluctuation, and then taking the evolution equation only of the mean quantity, a lot of undetermined double correlations will be present and a strong modeling will be needed. *Turbulent combustion models*, with some assumptions, provide the thermochemical closure and reduce the problem to a numerical manageable one. Every model has a restricted field of application and different models have different suitable utilization. It is also fundamental to distinguish two different kind of models: *non-premixed* and *premixed* combustion models. In the first case fuel and oxidizer enter with separate fluxes in the combustion chamber and in the second one they enter in a completely mixed state. There is also an intermediate case when fuel and oxidizer are considered partially premixed. Only non-premixed models will be here taken into account.

2.3.2 Passive scalar approach

The idea is to introduce new hypothesis so that it is possible to consider the instantaneous density, that in general is a function of $N + 2$ variables as can be argued by (2.23), as a function of just one variable and then to assume some other hipotesys on the statistical behavior of this single variable so that is possible to obtain mean values (variances, etc.) of the state variables that are considered like the density itself. This hipotesys are:

- Low velocity flux. In this case we can neglect the dissipation function Φ in the energy equation and use the sensible enthalpy h instead of stagnation enthalpy h_0. Pressure is considered *thermochemically constant* that is (as already mentioned) constant in the density expression (2.23) but variable (as divergence corrector) in the momentum equation. It can be observed that in the (2.23) temperature can vary of one order of magnitude (e.g. it varies from $300K$ to $2200K$ in methane-air stoichiometric combustion) and mix molecular weight can vary at most of one order of magnitude (e.g. in hydrogen air combustion it varies from $W \approx 2$ (fuel) to $W \approx 29$ (air)) but pressure, if the Mach is assumed to be low ($M \approx 0.2$) varies within a small range with respect to its stagnation value.

- Molecular diffusion coefficients are all equal to D and Lewis's number $Le = \frac{Sc}{Pr}$ is assumed unitary. Therefore Prandtl and Schmidt turbulent numbers are equal and their value is a constant $\sigma_Z = 0.7$.

- Adiabatic flux. There is no convective or radiative heat exchange with solid walls.

- Chemical equilibrium. This implies that the the Damköhler number $Da = t_f/t_c$, with t_f characteristic flux time and t_c characteristic chemical time, has to be very much greater than one. Low velocity flux, and high pressure and temperature hypothesis have a good agreement with this assumption (as it can be implied by Arrhenius expression).

For the chemical equilibrium hypothesis every variable can be expressed as a function of two other state variables for a particular initial value of the equivalence ratio φ of reactants defined as

$$\varphi = \frac{(F/O)}{(F/O)_{st}} = \frac{\frac{Z}{1-Z}}{(\frac{Z}{1-Z})_{st}} \tag{2.43}$$

where there is also Z, called *mixture fraction* or *passive scalar*. It is the fuel mass fraction whether it is burned, unburned or partially burned, and it varies between 0 and 1. The oxidizer's mixture fraction will be $1 - Z$. For its definition it can be argued that the mixture fraction's governing equation is a classical convection-diffusion, that is to say

$$\frac{\partial(\rho Z)}{\partial t} + \frac{\partial(\rho u_i Z)}{\partial x_i} = \frac{\partial}{\partial x_i}\left[\frac{\mu}{\sigma}\frac{\partial Z}{\partial x_i}\right] \tag{2.44}$$

For the thermo-chemically constant pressure hypothesis, the ρ functional expression, that in general (for the chemical equilibrium) is $\rho = \rho(p, h; \phi)$, will be dependent only from the specific enthalpy and from the mixture fraction, and so

$$\rho = \rho(h; Z) \tag{2.45}$$

It is also useful to point out that with all the previously stated hypothesis, the energy equations is

$$\frac{\partial\left(\rho h_0\right)}{\partial t} + \frac{\partial\left(\rho u_i h_0\right)}{\partial x_i} = \frac{\partial p}{\partial t} + \frac{\partial}{\partial x_i}\left[\frac{\mu}{\sigma}\frac{\partial h}{\partial x_i}\right] \tag{2.46}$$

and in its steady state formulation it is identical to (2.44). Both equations, though, differ just for boundary conditions but if enthalpy in (2.46) is scaled as follows

$$\frac{h - h_0}{h_f - h_0} \tag{2.47}$$

where h_0 and h_f are respectively the oxidizer and fuel enthalpy, boundary conditions will coincide. So they both vary from 0 (pure oxidizer) to 1 (pure fuel), and an homogeneous Neumann condition can be imposed (solid (Z) and adiabatic (h) wall) on solid walls. Because enthalpy is function of Z,

$$h = h_0 + Z(h_f - h0) \tag{2.48}$$

equation (2.45) becomes

$$\rho = \rho\left[h(Z);Z\right] = \rho(Z). \tag{2.49}$$

It is clear that this functional relation can be numerically determined because, for a given Z (point value fluid-dynamically evaluated), it is possible to obtain enthalpy and temperature and with the thermo-chemically constant pressure, it is possible to evaluate the M non linear equations that link the N species molar fractions in M reactions in chemical equilibrium

$$\prod_{m=1}^{N} X_m^{\Delta\nu_{m,k}} = K_{X,k}(T,p) \qquad (k = 1, 2, ..., M) \tag{2.50}$$

where $K_{X,k}$ is the equilibrium constant, in terms of molar fraction, of the k^{th} reaction. Density can be evaluated from (2.23), and so then all other state variables can be evaluated.

In the turbulent case, knowing the mixture fraction's statistical distribution dependancy from spatial coordinates (i.e. $P(Z; x_1, x_2, x_3, t)$, probability distribution function (pdf)), from (2.49) it could be possible to evaluate the mean density (and the mean value for all the others state variables) by

$$\overline{\rho} = \int_0^1 \rho(Z)P(Z)dZ. \tag{2.51}$$

Because of the smooth variation of $\rho(Z)$ (not taking into account its strong variation around Z_{st}), it is possible to argue that the integral (2.51) will be computed correctly even if a presumed form of the Z pdf is adopted. It is possible to locally determine the shape of the pdf transporting its moments $\tilde{Z}$,$\widetilde{Z''^2}$,$\widetilde{Z''^3}$,... In general just the first two moments are used because of the superior order moments' equations modeling difficulty, without substantial improvement (considering the model roughness). The steady state equations for mean and variance of Z are

$$\frac{\partial}{\partial x_i}(\overline{\rho}\tilde{u}_i\tilde{Z}) = \frac{\partial}{\partial x_i}\left[(\frac{\mu}{\sigma} + \frac{\mu_t}{\sigma_Z})\frac{\partial \tilde{Z}}{\partial x_i}\right] \tag{2.52}$$

$$\frac{\partial}{\partial x_i}(\overline{\rho}\tilde{u}_i\widetilde{Z''^2}) = \frac{\partial}{\partial x_i}\left[(\frac{\mu}{\sigma} + \frac{\mu_t}{\sigma_{Z''^2}})\frac{\partial \widetilde{Z''^2}}{\partial x_i}\right] + 2\frac{\mu_t}{\sigma_Z}\frac{\partial \tilde{Z}}{\partial x_i}\frac{\partial \tilde{Z}}{\partial x_i} - \overline{\rho}\tilde{\chi}. \tag{2.53}$$

On the right hand side of (2.53) it is possible to identify, as in the turbulent kinetic energy equation, respectively, a diffusion term (molecular and turbulent), a production and a dissipation (known as *scalar dissipation*) one. The latter one can be modeled according to Kolmogorov, assuming direct proportionality with $\widetilde{Z''^2}$ that is due to the increasing of fluctuations so that

$$\tilde{\chi} = C_\chi \frac{\tilde{\epsilon}\widetilde{Z''^2}}{\tilde{k}} \tag{2.54}$$

where to the modeling constant C_χ the value 2.0 is given.

The functional form of $P(Z)$ can't be a Gaussian one because random variable Z can assume values just between 0 and 1. Additionally the *intermittence* phenomenon, where Z can assume its extremity values (0 and 1) for a limited amount of time, must be taken into account. This can be achieved with a divergent Z pdf function in 0 and 1 that depends from two parameters

$$P(Z; \tilde{Z}(x), \widetilde{Z''^2}(x)) \tag{2.55}$$

the chosen form of the function is

$$P(Z) = C\ Z^{a-1}(1-Z)^{b-1} \tag{2.56}$$

where C is a normalizing factor meant to normalize to one the pdf integral, and a and b are functions of mean and variance of Z

$$a = \frac{\tilde{Z}^2(1-\tilde{Z})}{\widetilde{Z''^2}} - \tilde{Z} \tag{2.57}$$

$$b = \frac{\tilde{Z}(1-\tilde{Z})^2}{\widetilde{Z''^2}} - 1 + \tilde{Z}. \tag{2.58}$$

The intermittence phenomenon is taken into account when $\tilde{Z}$ and $\widetilde{Z''^2}$ assume values so that a and b are negative.

As mentioned above, mean density can be locally evaluated with a numerical integration of equation (2.51) where $P(Z)$ is known from the $\tilde{Z}$ and $\widetilde{Z''^2}$ transport. For this integral calculation a lot of points in the Z space could be necessary and this could represent a severe computational issue. In the same way all the mean state quantities can be evaluated including mean species mass fractions and their variances. This presumed pdf technique is possible because of the sufficiently smooth behavior of the $\rho(Z)$ law, but this is absolutely not true for the mean species production rate that can vary of ten orders of magnitude within a short range of Z. It is here useful to remember that the chemical equilibrium hypothesis is the reason why the transport equation of species and energy are not taken into account but replaced with the passive scalar equation that by definition has no source term. On the contrary, the chemical species transport equation source term $\overline{w}_m$ is not zero.

In conclusion, in the passive scalar approach combustion is assumed as controlled by the mixing (or molecular and turbulent diffusion) of fuel and oxidizer fluxes rather than by chemical reaction rates. In this way it is possible to evaluate all of the instantaneous state variables as functions of just one variable, the passive scalar Z. Assuming an approach based on the pdf reconstruction from (2.23) would have required a $N+2$ variables p, T, Y_m $(m = 1, 2, ..., N)$ joined pdf prevision that needs at least the evaluation of all variables first moments together with second ones and the latter calculation includes all the possible double correlations. And yet there would the evaluation of the joint pdf of the species production rate problem that is difficult to calculate for all the already described reasons including the stiff dependancy of $\overline{w}_m$ with the pdf shape for its high variability.

2.3.3 Flamelet Model

The key hypothesis of this model is that chemical reactions have a very small time scale (but not zero) so that flames have one dimension (thickness) much smaller than other two. With this hypothesis it is possible to write all the transport equations with functions of just two variables Z and χ. The first one is the *mixture fraction* and, as already stated, it represents the fluid mass fraction composed by fuel originating atoms, not taking into account if this atoms are linked to other species as a consequence of combustion. The second variable χ is the *scalar dissipation rate* and takes into account slow-chemistry temporal effects that is still considered *fast* (compared to flow velocity) but not in equilibrium. As it will be seen later on , among other things χ governs the flame *quenching* effects.

Considering for the transport equations the following variables transformation

$$\begin{aligned}
&\tau = t \\
&Z = Z\left(x_1, x_2, x_3, t\right) \\
&Z_2 = x_2 \\
&Z_3 = x_3
\end{aligned}$$

and with the definition

$$\chi = 2D\frac{\partial Z}{\partial x_k}\frac{\partial Z}{\partial x_k} \tag{2.59}$$

it is possible to show that, for the thin flames hypothesis, species transport equations can be rewritten as

$$\rho\frac{\partial Y_j}{\partial \tau} = \rho\frac{\chi}{2}\frac{\partial^2 Y_j}{\partial Z^2} + w_j \qquad j = 1, 2, \ldots, N \tag{2.60}$$

and in the steady state

$$\rho\frac{\chi}{2}\frac{\partial^2 Y_j}{\partial Z^2} + w_j\left(\rho, T, Y_1, \ldots, Y_N\right) = 0 \tag{2.61}$$

where w_j is the j^{th} species production rate. It is also possible to show that enthalpy transport equation becomes

$$\rho\frac{\chi}{2}\frac{\partial^2 h}{\partial Z^2} = 0. \tag{2.62}$$

If χ was known, it would be possible to solve (2.61) and (2.62), together with state equations

$$\rho = \frac{p}{R^0 T \sum_{i=1}^{N} \frac{Y_i}{W_i}} \tag{2.63}$$

$$h = \sum_{i=1}^{N} Y_i h_i\left(T\right) \tag{2.64}$$

obtaining state variables equations as

$$\rho = \rho\left(Z, \chi\right) \tag{2.65}$$

$$h = h\left(Z, \chi\right) \tag{2.66}$$

$$T = T\left(Z, \chi\right) \tag{2.67}$$

$$Y_i = Y_i\left(Z, \chi\right) \tag{2.68}$$

. . .

Flamelet approach makes it possible to consider chemical kinetics effects by means of w_i that are present in (2.61) and introduces (with respect to the passive scalar approach) new variable χ with definition (2.59). In this definition of χ, due to the presence of a spatial derivative, there is a spatial scale dependancy: in thin flames χ will be big, on the other hand, it will be small for thick ones. In the first case, finite velocity chemical effects are important, in the latter there is a quasi-equilibrium situation.

In this approach flames will have an internal distribution of state variables given by (2.63)-(2.68). For example temperature will be a function of Z and with the right choice of a reference frame and once χ is known, Z is a function of the spatial coordinate x normal to the flame, according to

$$dx = \sqrt{2D/\chi}\, dZ.$$

As a function of Z temperature has high gradients when χ is big and this situation increases the heat transfer from flame's inner zones (where conditions are nearer to stoichiometric ones and temperature is high) to outer zones with a subsequent peak temperature decrease. When χ reaches values greater than a given quenching value χ_q the flame stops burning.

Because of the non linear relation between Z and x, according to its definition χ is not a constant but a Z function. This problem is solved with a presumed functional form for the scalar dissipation rate that corresponds to an idealized configuration and this is a counterflow diffusion flame configuration ([16] and [15]). This functional form can be written as

$$\chi\left(Z\right) = \chi_{st}\frac{f\left(Z\right)}{f\left(Z_{st}\right)}.$$

where χ_{st} is the scalar dissipation rate at a reference value of the mixture fraction Z_{st} that is typically chosen in stoichiometric conditions. For the sake of exactness it must be specified that state variables are not functions of two independent variables Z and χ, but of a variable Z and of a parameter χ_{st}.

In the turbulent flows case, the flamelet model can be applied for the instantaneous values of state variables when the laminar flames thickness is smaller then the smallest spatial turbulent scale that is the Kolmogorov scale. In this case the turbulent flame is a composition of small laminar flames (flamelets) transported and stretched by the turbulent flow. The instantaneous value of a generic state variable Φ can be still evaluated with a law of the form $\Phi = \Phi\left(Z, \chi_{st}\right)$ but for the evaluation of mean values a particular presumed probability distribution function $P\left(Z, \chi_{st}\right)$ is needed so that is possible to evaluate mean values for every state variable Φ,

$$\bar{\Phi} = \int_0^{\infty} \int_0^1 \Phi\left(Z, \chi_{st}\right) P\left(Z, \chi_{st}\right) dZ d\chi_{st}.$$

with the statistical independence hypothesis it is possible to factorize the joint pdf in two probability distributions, one for Z and another one for χ_{st}

$$P\left(Z, \chi_{st}\right) = P\left(Z\right) P\left(\chi_{st}\right).$$

For the first one a beta pdf is assumed

$$P\left(Z\right) = C Z^{a-1} \left(1 - Z\right)^{b-1}$$

where C is a normalizing factor needed to set equal to one the pdf integral and is evaluated as

$$C = \left[\int_0^1 Z^{a-1} \left(1 - Z\right)^{b-1} dZ\right]^{-1}$$

and a and b exponentials have a Z mean and variance dependancy

$$a = \frac{\tilde{Z}^2\left(1-\tilde{Z}\right)}{\widetilde{Z''^2}} - \tilde{Z}$$
$$b = \frac{\tilde{Z}\left(1-\tilde{Z}\right)^2}{\widetilde{Z''^2}} - 1 + \tilde{Z}.$$

For the scalar dissipation rate probability distribution a log-normal law is assumed

$$P\left(\chi_{st}\right) = \frac{1}{\sqrt{2\pi}}\frac{1}{\hat{\sigma}\chi_{st}}\exp\left[-\frac{1}{2\hat{\sigma}^2}\left(\log\frac{\chi_{st}}{\chi_{ref}} - \hat{\mu}\right)^2\right]$$

where

$$\frac{\tilde{\chi}}{\chi_{ref}} = \exp\left(\hat{\mu} + \frac{\hat{\sigma}^2}{2}\right)$$
$$\tilde{\chi} = C_\chi \frac{\tilde{\epsilon}\widetilde{Z''^2}}{\tilde{k}}$$

and $C_\chi = \hat{\sigma} = 2.0$ is assumed.

For the $\chi\left(Z\right)$ law determination, needed to solve equations (2.63)-(2.68), a self similarity hypothesis is assumed and this leads to the law

$$\chi = \frac{a_s}{\pi}\exp\{-2\left[\operatorname{erfc}^{-1}\left(2Z\right)\right]^2\}$$

where a_s represents the strain rate in a counter-flow diffusion flame stagnation point, so that it is a velocity gradient with the dimension of the inverse of a convective time. The χ_{st} value ranges form zero (chemical equilibrium) to χ_q (flame quenching) and for greater values than the latter an inert mix of fuel and oxidizer is supposed.

With the described procedure a flamelet library is obtained.

Chapter 3

Numerical Models

In this chapter the structure of the PRIN-3D solver of the linear system of equations yielded by the discretization of the incompressible laminar Navier Stokes partial differential equations:

$$\frac{\partial v_j}{\partial t} + \frac{\partial v_i v_j}{\partial x_i} = -\frac{1}{\rho}\frac{\partial P}{\partial x_j} + \nu \frac{\partial^2 v_j}{\partial x_i \partial x_i}$$
$$\frac{\partial v_i}{\partial x_i} = 0$$

will be analyzed.

3.1 Time discretization

An implicit time discretization for the diffusive part, weighing with θ the unknowns at time $n+1$ and $(1-\theta)$ the known values at time n, will be adopted[1] whereas for the non linear part, several numerical schemes are available in the code. In general an asterisk (*) will be used to indicate the presence

[1]The most common choice for the value of θ may be $1/2$ and this leads to a truncation error $e = O(\Delta t^2) + O(\Delta x^2)$, i.e. the Cranck-Nicholson scheme, but there are specific values for θ that increase the order of accuracy of the scheme such as $\theta = \nu \Delta t / \Delta x^2$ that lead to a truncation error $e = O(\Delta t^2) + O(\Delta x^4)$

of a generic numerical evaluation strategy for such terms[2]. All non linear terms are left on the right side of the discrete equations, that is to say the evaluation of such terms is only done on the basis of values at time t^n. This time discretization scheme leads to the following set of equations:

$$v_j^{n+1} - \nu\theta\Delta t \frac{\partial^2 v_j^{n+1}}{\partial x_i \partial x_i} + \frac{\Delta t}{\rho}\frac{\partial P^{n+1}}{\partial x_j} = {v_j}^n - \Delta t \frac{\partial v_i v_j}{\partial x_i}^{n^*} + \\ + \nu\Delta t\,(1-\theta)\frac{\partial^2 v_j^n}{\partial x_i \partial x_i} \quad (3.1)$$

$$\frac{\partial {v_i}^{n+1}}{\partial x_i} = 0. \quad (3.2)$$

The pressure gradient has been collocated directly at the time $t^{(n+1)}$ and this is to stress the fact that the vector field $\frac{\partial P^{n+1}}{\partial x_j}$ has to correct the divergence of the finally computed field at time $t^{(n+1)}$ and that in incompressible flows pressure has barely no physical meaning and so does any attempt to assign a time collocation to it. Equations (3.2) and (3.1) can be rewritten as follows

$$(1 - \nu\theta\Delta t \frac{\partial^2}{\partial x_i \partial x_i})\, v_j^{n+1} + \frac{\partial}{\partial x_j} p^{\,n+1} = q_j^n \qquad (j = 1,2,3) \quad (3.3)$$

$$\frac{\partial}{\partial x_i} v_i^{n+1} = 0 \quad (3.4)$$

with

$$p = \Delta t\,\frac{P}{\rho}$$

$$q_j^n = {v_j}^n - \Delta t \frac{\partial v_i v_j}{\partial x_i}^{n^*} + \nu\Delta t\,(1-\theta)\frac{\partial^2 v_j^n}{\partial x_i \partial x_i}.$$

The solver of such equations will be the *solving kernel* for every incompressible or even slightly compressible model being resolved.

[2]see the Chapter 4 dedicated to the 1D Advection Testing

3.2 Full Pressure Segregation

Introducing a spatial discretization in the system of equations (3.3)-(3.4), the whole problem can be restated in terms of algebraic operators

$$\begin{aligned} &F\, v_j^{n+1} + G_j\, p^{\,n+1} = q_j \qquad (j = 1, 2, 3) \\ &D_i\, v_i^{\,n+1} = g. \end{aligned}$$

The symbology can be further more condensed in the following manner

$$\begin{pmatrix} \mathcal{F} & \mathcal{G} \\ \mathcal{D} & 0 \end{pmatrix} \begin{pmatrix} v \\ p \end{pmatrix}^{n+1} = \begin{pmatrix} q \\ g \end{pmatrix} \tag{3.5}$$

where $\mathcal{F}$ is a block-diagonal elliptic operator $\left(\begin{smallmatrix} F & 0 & 0 \\ 0 & F & 0 \\ 0 & 0 & F \end{smallmatrix}\right)$, $\mathcal{G}$ is a column operator $\left(\begin{smallmatrix} G_1 \\ G_2 \\ G_3 \end{smallmatrix}\right)$, $\mathcal{D}$ is a row operator $\left(\begin{smallmatrix} D_1 & D_2 & D_3 \end{smallmatrix}\right)$ and $v = \left(\begin{smallmatrix} v_1 \\ v_2 \\ v_3 \end{smallmatrix}\right)$.

In order to solve system (3.5), one choice is to consider Richardson iteration with the following left preconditioner

$$\mathcal{P} = \begin{pmatrix} \mathcal{F} & 0 \\ \mathcal{D} & -I \end{pmatrix} \tag{3.6}$$

that can be interpreted in terms of a Chorin-like projection method. In fact, for every iterative step

$$\begin{pmatrix} \mathcal{F} & 0 \\ \mathcal{D} & -I \end{pmatrix} \begin{pmatrix} v \\ p \end{pmatrix}^{n+1,iter+1} = - \begin{pmatrix} 0 & -\mathcal{G} \\ 0 & I \end{pmatrix} \begin{pmatrix} v \\ p \end{pmatrix}^{n+1,iter} + \begin{pmatrix} q \\ g \end{pmatrix} \tag{3.7}$$

an intermediate velocity field is computed (momentum equations solved without the pressure-gradient term) and then pressure is, with virtually no cost, retrieved from the computation of the divergence of such field. Supposing such method to be convergent, this will lead the iterated solution to satify to the following preconditioned system (see [3], [9] and [14])

$$\begin{pmatrix} \mathcal{F}^{-1} & 0 \\ \mathcal{D}\mathcal{F}^{-1} & -I \end{pmatrix} \begin{pmatrix} \mathcal{F} & \mathcal{G} \\ \mathcal{D} & 0 \end{pmatrix} \begin{pmatrix} v \\ p \end{pmatrix}^{n+1} = \begin{pmatrix} q' \\ g' \end{pmatrix} \tag{3.8}$$

that with simple algebraic manipulation can be rewritten as

$$\begin{pmatrix} I & \mathcal{F}^{-1}\mathcal{G} \\ 0 & \mathcal{D}\mathcal{F}^{-1}\mathcal{G} \end{pmatrix} \begin{pmatrix} v \\ p \end{pmatrix}^{n+1} = \begin{pmatrix} \mathcal{F}^{-1}q \\ \mathcal{D}\mathcal{F}^{-1}q - g \end{pmatrix}. \tag{3.9}$$

This is the typical structure of a segregated system where velocity and pressure variable computations have been uncoupled. As a matter of fact, the segregation procedure can also be suggested by the Inverse Theorem of the Vector Field Calculus [13] which gives a fair interpretation of the pressure field [3] as a scalar function that provides the divergence correction for the velocity field and, thus, has to satisfy the classic pressure *segregated* Poisson's equation

$$\mathcal{D}\mathcal{F}^{-1}\mathcal{G}p^{n+1} = \mathcal{D}\mathcal{F}^{-1}q - g. \tag{3.10}$$

This process involves the computation of the inverse $\mathcal{F}^{-1}$ or of the group $\mathcal{F}^{-1}\mathcal{G}$. Since the coefficient matrix of the linear system of equations (3.5) remains unchanged throughout the whole computation (advection term is estimated with explicit techniques such as *predictor-corrector*), such inverse could be precomputed and stored in memory. This option is clearly not recommendable since if the matrices $\mathcal{F}^{-1}$ and $\mathcal{F}^{-1}\mathcal{G}$ were directly computed,

[3] even though would be exactly the case only if an explicit time stepping scheme had been adopted

both would turn out to be completely full and their storage would be dramatically demanding on memory resources. Nevertheless, solving exactly (3.10) will guarantee that the new velocity field is divergence-free. Of course also matrix $\mathcal{D}\mathcal{F}^{-1}\mathcal{G}$ is completely full.

3.3 Sparse Approximate Inverse: Quasi-segregated discrete equations

Since, as stated above, performing this first segregation process, by means of direct solving, is too expensive especially in the 3D case ([11], [6]), a smarter choice would be to adopt the following polynomial expansions of the inverse of F (and so of the whole block diagonal matrix $\mathcal{F}$):

$$F^{-1} = (I - \nu\theta\Delta t\ \nabla^2)^{-1} = I + \sum_{n=1}^{\infty}(\nu\theta\Delta t\ \nabla^2)^n.$$

which converges only if $\nu\theta\Delta t\rho(\nabla^2) < 1$. It can be truncated at the N-th order so that to obtain the following approximate inverse operator

$$\tilde{F}_N^{-1} = I + (\nu\theta\Delta t\ \nabla^2) + (\nu\theta\Delta t\ \nabla^2)^2 + \cdots + (\nu\theta\Delta t\ \nabla^2)^N \tag{3.11}$$

with an obvious reduced sparsity if N increases. Instead of using the full inverse F^{-1}, the approximate N-th order inverse (3.11) for the three diagonal blocks of $\mathcal{F}^{-1}$ will be now adopted, yielding the following sparse approximate block diagonal matrix

$$\tilde{\mathcal{F}}_N^{-1} = \begin{pmatrix} \tilde{F}_N^{-1} & 0 & 0 \\ 0 & \tilde{F}_N^{-1} & 0 \\ 0 & 0 & \tilde{F}_N^{-1} \end{pmatrix}. \tag{3.12}$$

Before proceeding, it is important to (numerically) analyze and eventually control the error of adopting the sparse approximate inverse (3.11) instead of the full inverse $F^{-1} = \left[I - (\nu\theta\Delta t\ \nabla^2)\right]^{-1}$. Let D be an n-by-n matrix so that $\frac{1}{h^2}D$ is the numerical approximation on uniform mesh with $\Delta x = \Delta y = \Delta z = h$ of the Laplacian operator with homogeneous Dirichlet boundary conditions. The numerical approximation of the operator F will then be $(I - \beta D)$ with $\beta = \frac{\nu\theta\Delta t}{h^2}$. In Figure (3.1), it is shown how the error of a first order (i.e. $N = 1$) sparse approximate inverse defined as

$$\frac{||(I + \beta\nabla^2) - (I - \beta\nabla^2)^{-1}||}{||(I - \beta\nabla^2)^{-1}||}$$

increases with n but for large matrix sizes it is only function of β. To control such error, a value of β (e.g $\beta = 0.02$) can be chosen and this, for a given mesh size, viscosity and θ will be an upper bound for the time step interval Δt, apart from other constraints given, for example, by the CFL condition.

Here, the approximate inverse (3.12) instead of $\mathcal{F}^{-1}$ is used, and the segregation process (3.8) is repeated. Bearing in mind that

$$\tilde{F}_N^{-1} F = I - B_N = I - (\nu\theta\Delta t\ \nabla^2)^{N+1} \tag{3.13}$$

this yields

$$\begin{pmatrix} I - \mathcal{B}_N & \tilde{\mathcal{F}}_N^{-1}\mathcal{G} \\ -\mathcal{D}\mathcal{B}_N & \mathcal{D}\tilde{\mathcal{F}}_N^{-1}\mathcal{G} \end{pmatrix} \begin{pmatrix} v \\ p \end{pmatrix}^{n+1} = \begin{pmatrix} \tilde{\mathcal{F}}_N^{-1} q \\ \mathcal{D}\tilde{\mathcal{F}}_N^{-1} q - g \end{pmatrix}. \tag{3.14}$$

The pressure variables in this system are quasi-segregated (not fully segregated as in the (3.9) system) since $\mathcal{B}_N$ is different from zero but of the $(N+1)^{th}$ order, every term with $\mathcal{B}_N$ can be somehow neglected or used to

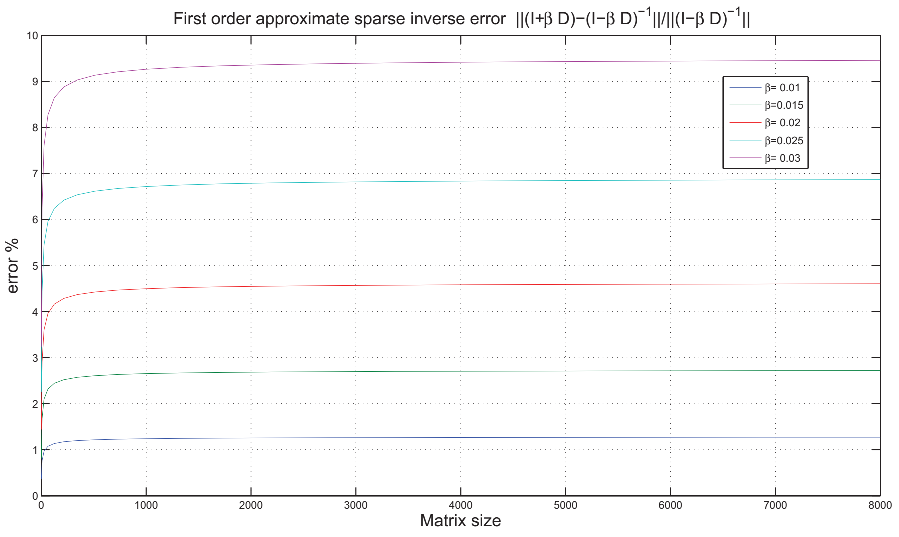


Figure 3.1: **First order approximate sparse inverse error versus natrix size for increasing values of β**

start the following iterative method which will be denoted with the i iterative counter:

$$\begin{pmatrix} I & \tilde{\mathcal{F}}_N^{-1}\mathcal{G} \\ 0 & \mathcal{D}\tilde{\mathcal{F}}_N^{-1}\mathcal{G} \end{pmatrix} \begin{pmatrix} v \\ p \end{pmatrix}^{n+1,i+1} = \begin{pmatrix} \mathcal{B}_N & 0 \\ \mathcal{D}\mathcal{B}_N & 0 \end{pmatrix} \begin{pmatrix} v \\ p \end{pmatrix}^{n+1,i} + \begin{pmatrix} \tilde{\mathcal{F}}_N^{-1}\, q \\ \mathcal{D}\tilde{\mathcal{F}}_N^{-1} q - g \end{pmatrix}. \tag{3.15}$$

The system of equations (3.15) is clearly block-triangular and when solving for the $v^{n+1,i+1}$ unkowns, the following pressure segregated equation has to be solved first:

$$\mathcal{D}\tilde{\mathcal{F}}_N^{-1}\mathcal{G}p^{n+1,i+1} = \mathcal{D}\mathcal{B}_N v^{n+1,i} + \mathcal{D}\tilde{\mathcal{F}}_N^{-1} q - g. \tag{3.16}$$

It is possible to split the approximate inverse operator $\tilde{F}_N^{-1}$ into two parts $I + F_1$ and F_2, as follows

$$\tilde{F}_N^{-1} = I + \sum_{n=1}^{N_{lhs}} (\nu\theta\Delta t\ \nabla^2)^n + \sum_{n=N_{lhs}+1}^{N} (\nu\theta\Delta t\ \nabla^2)^n = I + F_1 + F_2.$$

With such splitting, equation (3.16) is ready to be solved with an iterative method. The laplacian's power to be left in the left-hand side of equation (i.e. in part $I + F_1$) is indicated with N_{lhs}, and this leads to the following "nested" iterative scheme indicated with the counter k:

$$\mathcal{D}(I+\mathcal{F}_1)\mathcal{G}p^{n+1,i+1,k+1} = -\mathcal{D}\mathcal{F}_2\mathcal{G}p^{n+1,i+1,k} + \mathcal{D}\mathcal{B}_N v^{n+1,i} + \mathcal{D}\tilde{\mathcal{F}}_N^{-1}q - g. \quad (3.17)$$

If $N_{lhs} = 0$ (i.e. $\mathcal{F}_1 = 0$), the equation (3.17) has the same coefficient matrix of the pressure equation in an explicit time discretization. The higher the value of N_{lhs} the less sparse will the coefficient matrix $\mathcal{D}(I + \mathcal{F}_1)\mathcal{G}$ be. Chosing $N_{lhs} \geq 1$ gives a bi-harmonic nature to such equation [19], reducing the pressure checkerboard effect, while, on the other hand, increasing the memory and computational demands for solving such system (see Figures (3.4) and (3.5)).

The Galerkin matrix $\left(\begin{smallmatrix} \mathcal{F} & \mathcal{G} \\ \mathcal{D} & 0 \end{smallmatrix}\right)$, being here attempted to be solved, is illustrated in Figure (3.3). In this case a sequential variable ordering (shown in Figure (3.2)) has been adopted and, for expository purposes, a coarse mesh (384 pressure nodes) is used.

Performing the previously described solving process with such variable ordering, the pattern of the final pressure equation's coefficient matrix for $N_{lhs} = 0$ and $N_{lhs} = 1$ (with $N \geq 1$) will respectively be as in Figure (3.4) and Figure (3.5).

Apparently there is no further possible nested iterative cycle that could be exploited to solve equation (3.17). Let's analyze the stencil of an internal

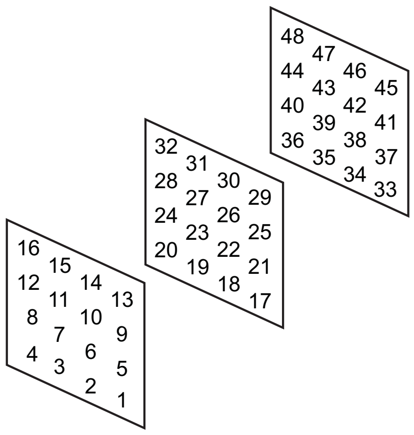


Figure 3.2: **Sequential (lexicographic) variable ordering**

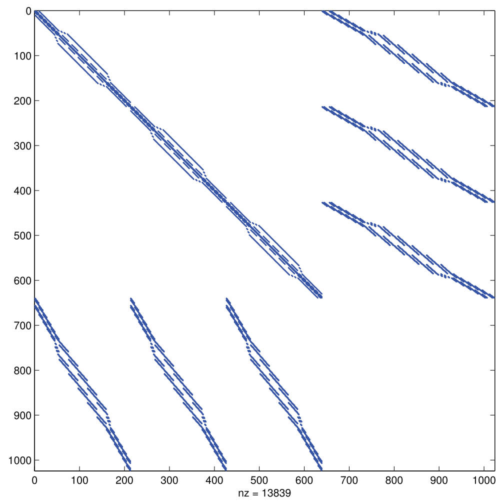


Figure 3.3: **Matrix pattern visualization of the Galerkin matrix**

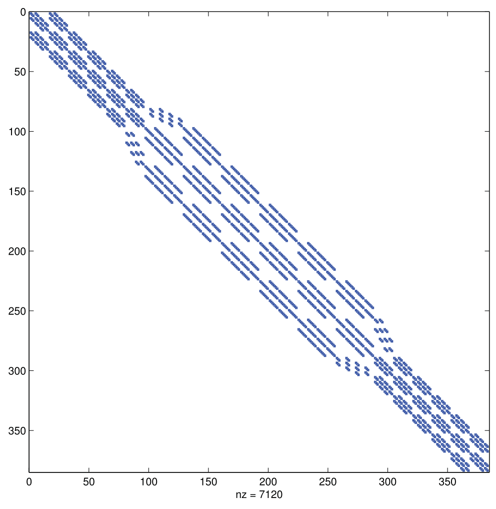


Figure 3.4: **Matrix pattern visualization of $\mathcal{D}(I+\mathcal{F}_1)\mathcal{G}$ with $N_{lhs}=0$**

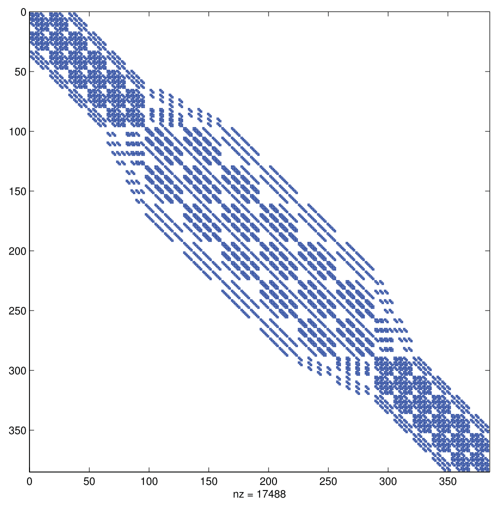


Figure 3.5: **Matrix pattern visualization of $\mathcal{D}(I+\mathcal{F}_1)\mathcal{G}$ with $N_{lhs}=1$**

equation given by the matrix $\mathcal{D}(I+\mathcal{F}_1)\mathcal{G}$ (that, for the sake of simplicity will be chosen with $N_{lhs}=0$) which is shown in Figure (3.6.)

An idea is to find here alternative types of pressure variable ordering so that the elliptic operator $\mathcal{D}(I+\mathcal{F}_1)\mathcal{G}$ assumes a block-structured pattern. This will make it possible to introduce another nested iterative cycle on equation (3.17) apart from allowing a parallel resolution for such equation. The other pressure variable ordering that are being examinated are the following

2. classic 3D red-black ordering, Figure (3.7), yielding patterns shown in Figure (3.10) and Figure (3.11)

3. three pressure "colors", Figure (3.8), yielding patterns Figure (3.12) and Figure (3.13)

4. four pressure "colors", Figure (3.9), yielding patterns Figure (3.14) and Figure (3.15)

All of the pressure orderings from 2 to 4 make it possible to set up a further iterative nested cycle (with counter l). Two strategies have been pursued: a block Gauss-Siedel method for a non-parallel solving technique and a block Gauss-Jacobi method which allows parallel computing on clusters or on multi-threaded single machine.

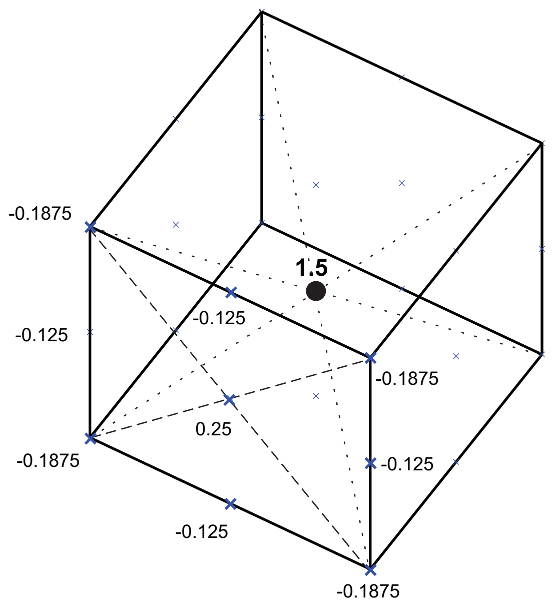


Figure 3.6: **Stencil of numerical operator $\mathcal{D}(I+\mathcal{F}_1)\mathcal{G}$ with $N_{lhs}=0$ - The weights are repeated identically for every cube face (isotropic elliptic operator), the central weight is $+1.5$**

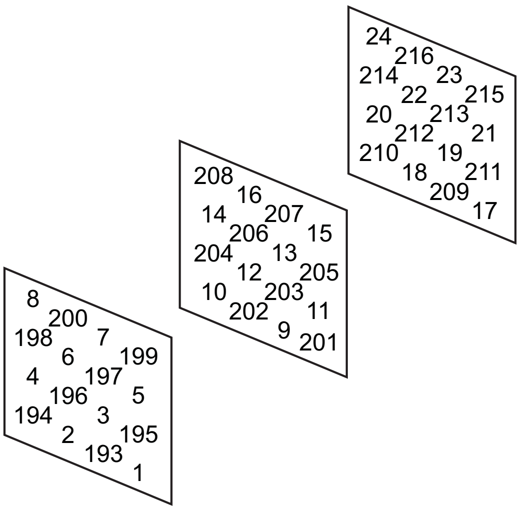


Figure 3.7: **Two colors pressure variable ordering (red-black)**

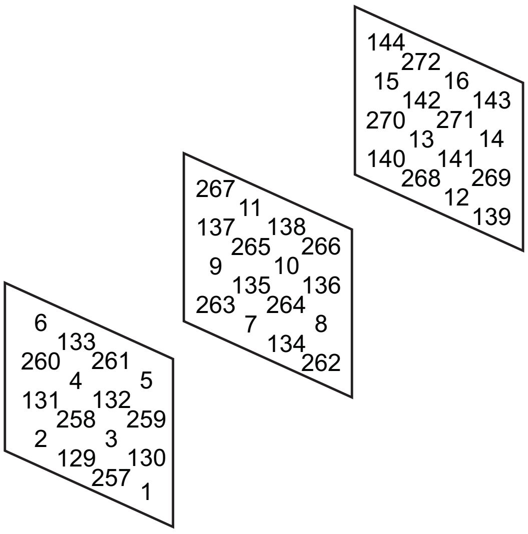


Figure 3.8: **Three colors pressure variable ordering**

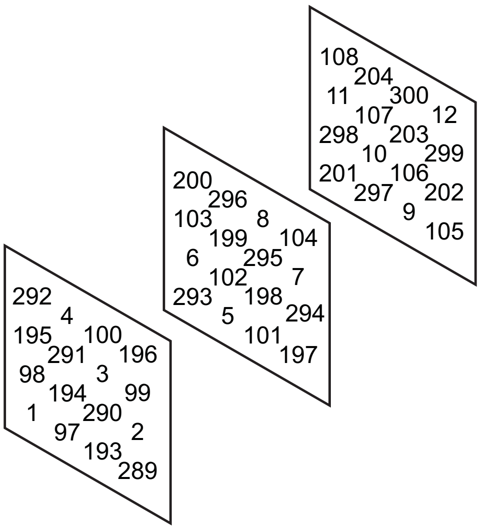


Figure 3.9: **Four colors pressure variable ordering**

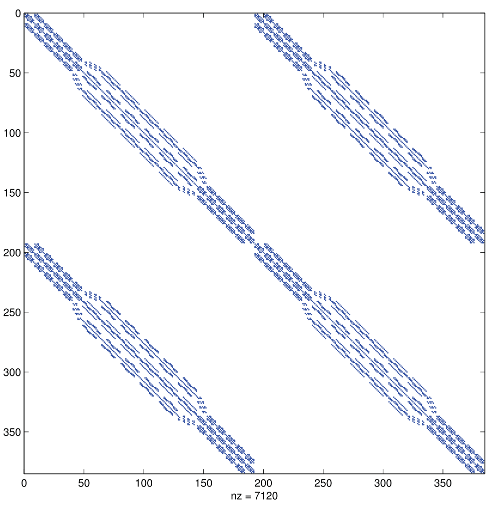


Figure 3.10: **Matrix pattern visualization of $\mathcal{D}(I + \mathcal{F}_1)\mathcal{G}$ with $N_{lhs} = 0$**

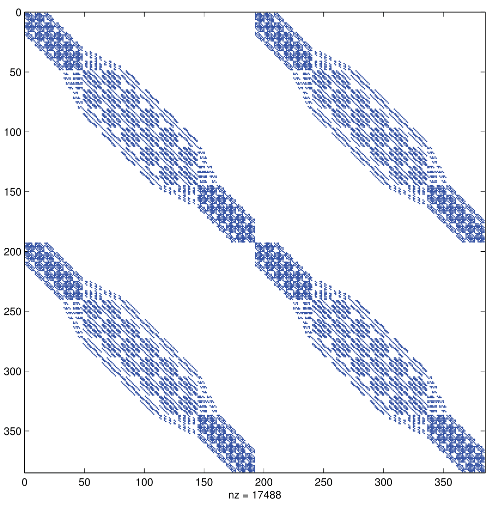


Figure 3.11: **Matrix pattern visualization of $\mathcal{D}(I + \mathcal{F}_1)\mathcal{G}$ with $N_{lhs} = 1$**

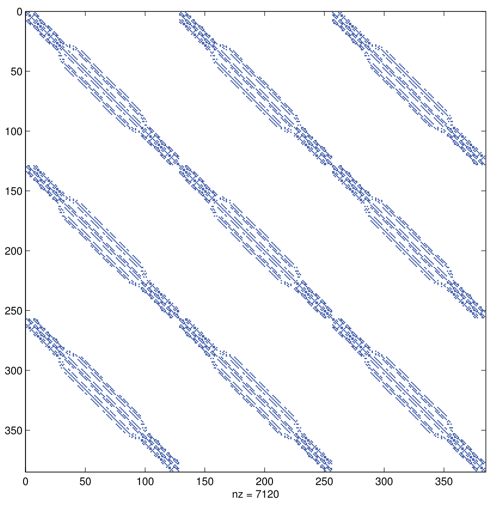


Figure 3.12: **Matrix pattern visualization of $\mathcal{D}(I + \mathcal{F}_1)\mathcal{G}$ with $N_{lhs} = 0$**

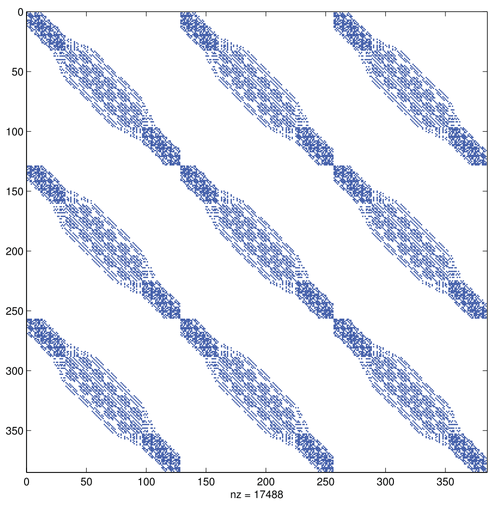


Figure 3.13: **Matrix pattern visualization of $\mathcal{D}(I + \mathcal{F}_1)\mathcal{G}$ with $N_{lhs} = 1$**

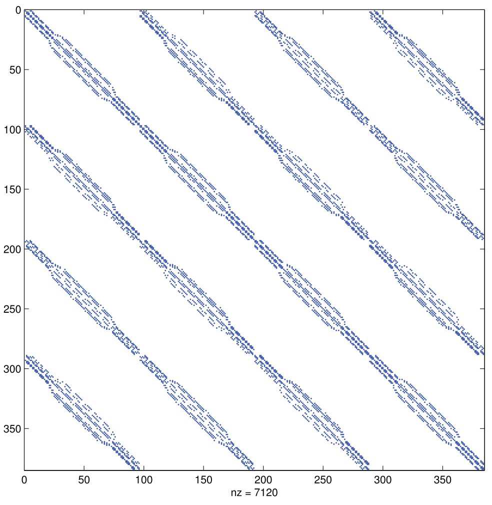


Figure 3.14: **Matrix pattern visualization of $\mathcal{D}(I + \mathcal{F}_1)\mathcal{G}$ with $N_{lhs} = 0$**

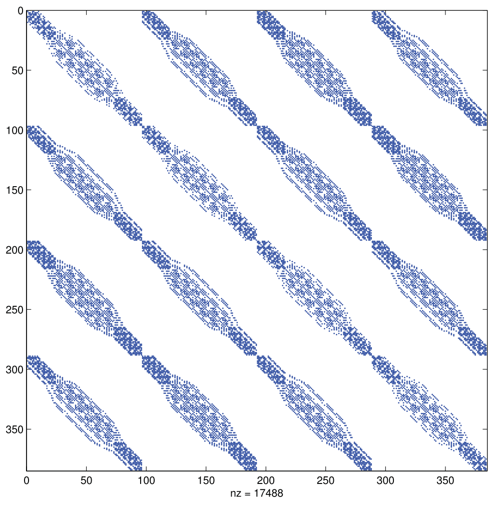


Figure 3.15: **Matrix pattern visualization of $\mathcal{D}(I + \mathcal{F}_1)\mathcal{G}$ with $N_{lhs} = 1$**

3.4 Flamelet Numerical Implementation

All variables at the time t^n are given, and a time stepping method designed to upgrade the solution to the time t^{n+1} is here described.

Let's start by using the mixture fraction transport equation in a non conservative form

$$\frac{\partial Z}{\partial t} + u_i \frac{\partial Z}{\partial x_i} = \frac{1}{\rho} \frac{\partial}{\partial x_i} \left[\frac{\mu}{\sigma} \frac{\partial Z}{\partial x_i} \right], \tag{3.18}$$

and then, let's introduce an explicit time discretization that will lead us to the following form

$$Z^{n+1} = Z^n - \Delta t \left[u_i \frac{\partial Z}{\partial x_i} \right]^n + \frac{\Delta t}{\rho^n} \frac{\partial}{\partial x_i} \left[\frac{\mu}{\sigma} \frac{\partial Z}{\partial x_i} \right]^n. \tag{3.19}$$

In order to use the same sparse approximate solver used for the incompressible model (that is the *numerical kernel* of our protocode PRIN-3D), the equation (3.19) will be forced into an implicit structure inserting fictitious viscosity and θ coefficients (ν_0 and θ_0) yielding

$$\mathcal{A}\, Z^{n+1} = Z^n - \Delta t \nu_0 \theta_0 \frac{\partial^2}{\partial x_i \partial x_i} Z^n - \Delta t \left[u_i \frac{\partial Z}{\partial x_i} \right]^n + \frac{\Delta t}{\rho^n} \frac{\partial}{\partial x_i} \left[\frac{\mu}{\sigma} \frac{\partial Z}{\partial x_i} \right]^n \tag{3.20}$$

with

$$\mathcal{A} = \left[I - \Delta t \nu_0 \theta_0 \frac{\partial^2}{\partial x_i \partial x_i} \right].$$

Let's mutiply both sides of the (3.20) by the N^{th} order sparse approximate inverse of $\mathcal{A}$ (see section 3.3), which is

$$\tilde{\mathcal{A}}_N^{-1} = +(\nu\theta\Delta t\ \nabla^2) + (\nu\theta\Delta t\ \nabla^2)^2 + \cdots + (\nu\theta\Delta t\ \nabla^2)^N$$

and introduce an iterative index k obtaining

$$Z^{n+1,k+1} = \left[\Delta t \nu_0 \theta_0 \frac{\partial^2}{\partial x_i \partial x_i}\right]^{N+1} Z^{n+1,k} + \\ + \tilde{\mathcal{A}}_N^{-1} \left\{ Z^n - \Delta t \nu_0 \theta_0 \frac{\partial^2}{\partial x_i \partial x_i} Z^n - \Delta t \left[u_i \frac{\partial Z}{\partial x_i}\right]^n + \frac{\Delta t}{\rho} \frac{\partial}{\partial x_i} \left[\frac{\mu}{\sigma} \frac{\partial Z}{\partial x_i}\right]^n \right\}. \tag{3.21}$$

Typically just a few iterations over k are necessary.

The value of Z^{n+1} on every node of the computational mesh is now calculated, as well as the value of the scalar dissipation rate $\chi^{n+1} = 2\ D\ \frac{\partial Z^{n+1}}{\partial x_i} \frac{\partial Z^{n+1}}{\partial x_i}$ where $D = \frac{\mu}{\rho \sigma}$. Looking up in the Flamelet's table created in preprocessing[4], the corresponding values of ρ^{n+1} (and of course $, T^{n+1}, \mu^{n+1}, etc..$) can be evaluated, giving in this way a satisfying estimation of the *source term* $-\frac{\partial \rho}{\partial t}$ in the continuity equation, and leading to the following time stepping method

$$(\rho u_j)^{n+1} - \Delta t\, \nu_0\, \theta_0 \frac{\partial^2}{\partial x_i \partial x_i} (\rho u_j)^{n+1} + \Delta t \frac{\partial P}{\partial x_j}^{n+1} = (\rho u_j)^n - \\ - \Delta t \frac{\partial}{\partial x_i} (\rho u_j u_i)^n + \Delta t \frac{\partial}{\partial x_i} \left[\mu \left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j}\right) - \frac{2}{3} \mu \frac{\partial u_k}{\partial u_k} \delta_{ij}\right]^n - \\ - \Delta t\, \nu_0\, \theta_0 \frac{\partial^2}{\partial x_i \partial x_i} (\rho u_j)^n \tag{3.22}$$

$$\frac{\partial}{\partial x_i} (\rho u_i)^{n+1} = - \frac{\rho^{n+1} - \rho^n}{\Delta t}. \tag{3.23}$$

With the same Incompressible Navier-Stokes Equation solver, the vector field $(\rho u_j)^{n+1}$ and the pressure field P^{n+1} can be determined. By dividing the former by ρ^{n+1}, the velocity vector field u_j^{n+1} can also be determined. Now the cycle can be started all over again. Given the complex, non-linear and inter-dependent nature of such equations, the best way to achieve numerical

[4]This preprocessing procedure is carried out by means of a C++ numerical code called FlameMaster [15]. For more informations: http://www.stanford.edu/group/pitsch/FlameMaster.htm

stability in many cases is to underrelax the output of the Flamelet Library (i.e. $\rho^{n+1}, T^{n+1}, \mu^{n+1}, etc..$). There is no optimum value for the underrelaxing factor, it may vary according to the Reynolds number and geometry and it is essentially determined from a trade off between avoiding instabilities and reaching the steady state as soon as possibile.

Some considerations ought to be made for the diffusive term (containing the real dynamic viscosity μ) in equation (3.22). The gradient of diffusive contribution of the isotropic part of the velocity strain tensor $-\frac{2}{3}\mu\frac{\partial u_k}{\partial u_k}\delta_{ij}$ will be included in the pressure variable and so it will be neglected. Regarding the other diffusive terms, considering that

$$\frac{\partial}{\partial x_i}\left[\mu\left(\frac{\partial u_j}{\partial x_i}+\frac{\partial u_i}{\partial x_j}\right)\right]=\frac{\partial}{\partial x_i}\left(\mu\frac{\partial u_j}{\partial x_i}\right)+\frac{\partial}{\partial x_i}\left(\mu\frac{\partial u_i}{\partial x_j}\right) \tag{3.24}$$

a different choice has to be made for the spatial differential operators in equation (3.24). The first term is the one that in the $\mu = cost$ (incompressible) case gives birth to the classic Laplacian operator so the same numerical operators used in the momentum equation are chosen. The second term has to vanish in the incompressible case so divergence and pressure gradient operators are used.

$$\frac{\partial}{\partial x_i}\left[\mu\left(\frac{\partial u_j}{\partial x_i}+\frac{\partial u_i}{\partial x_j}\right)\right]=\frac{\partial}{\partial x_i}\left(\mu\frac{\partial u_j}{\partial x_i}\right)+\mu\frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_i}\right)+\frac{\partial \mu}{\partial x_i}\frac{\partial u_i}{\partial x_j}=$$
$$\frac{\partial}{\partial x_i}\left(\mu\frac{\partial u_j}{\partial x_i}\right)_{Momentum}+\mu\frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_i}\right)_{Divergence}+\frac{\partial \mu}{\partial x_i}\frac{\partial u_i}{\partial x_j}_{Divergence}$$

3.5 $k - \epsilon$ Numerical Implementation

The first step in order to start setting up this kind of simulation is to estimate the correct mesh size. This is needed because the Low Reynolds number $k-\epsilon$ model is meant to integrate the mean velocity, k and ϵ fields, all the way down to the wall. Therefore, the viscous sublayer is needed to be filled in with at least 4 or 5 computational cells. As an example of how this procedure can be done, the mesh size estimation in the Channel flow case (see [17]) is illustrated.

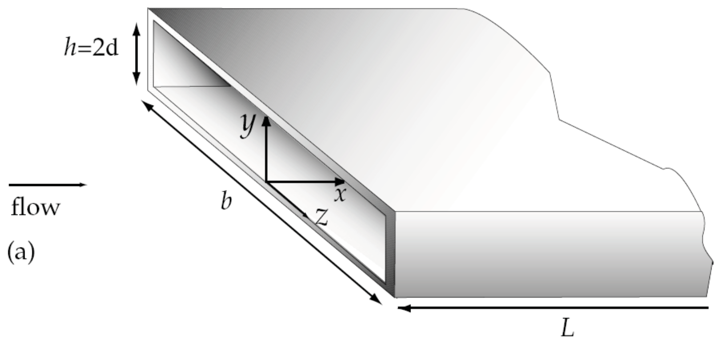


Figure 3.16: **Sketches of a channel flow (source [17])**

The procedure to empirically estimate the thickness of the viscous sublayer can be started by calculating the friction coefficient c_f. If the Reynolds number

$$Re = \frac{2\delta\bar{U}}{\nu}$$

based on the bulk velocity

$$\bar{U} = \frac{1}{\delta}\int_0^\delta \langle U\rangle dy$$

(see Figure 3.16) is known, consulting the diagram in Figure 3.17, a good guess for c_f can be made.

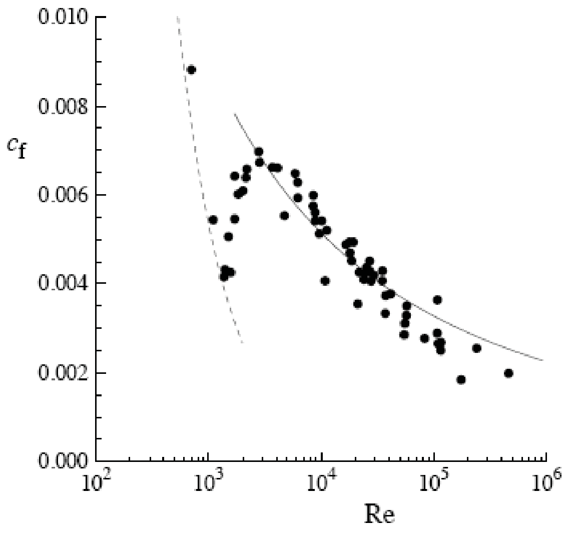


Figure 3.17: **The skin friction coefficient against the Reynolds number for channel flow (source [17])**

By the friction coefficient's definition

$$c_f = \frac{\tau_w}{\frac{1}{2}\rho U_0^2}, \tag{3.25}$$

where $U_0 = \langle U \rangle_{y=\delta}$ is the maximum mean velocity of the inflow profile, the τ_w can be evaluated, and in this way also the friction velocity

$$u_\tau = \left(\frac{\tau_w}{\rho}\right)^{1/2} = \left(-\frac{\delta}{\rho}\frac{dp_w}{dx}\right)^{1/2} \tag{3.26}$$

can be calculated, allowing us to finally obtain the viscous length

$$\delta_\nu = \frac{\nu}{u_\tau} \tag{3.27}$$

that defines the thickness of the viscous sublayer (for Channel flows $\approx 5\ \delta_\nu$).

All the details Regarding the numerical solving procedure are not reported here, but rather a brief description of the updating process from t^n to t^{n+1} is given. This is because of the identical mathematical structure that mean flow equations (2.11)-(2.12) and $k-\epsilon$ equations (2.13)-(2.14) (or (2.16)-(2.14)) share with incompressible Navier-Stokes equations (2.5)-(2.6) so that the same *numerical kernel* (see sections 3.1-3.3) can be applied.

To have a non zero velocity field to begin with, a few laminar iterations are performed before the actual $k-\epsilon$ numerical solver takes place. Knowing the turbulent viscosity

$$\nu_T = C_\mu \frac{k^2}{\epsilon} \tag{3.28}$$

at time step n, the mean velocity field ($\langle U_i \rangle$, $i = 1,2,3$) can be directly updated to time step $n+1$, because the term

$$\frac{\partial}{\partial x_i}\left[\nu_T \left(\frac{\partial \langle U_j \rangle}{\partial x_i} + \frac{\partial \langle U_i \rangle}{\partial x_j}\right)\right] \tag{3.29}$$

is positioned on the right-hand side of the numerical discretization of equation (2.12). With this mean velocity field, the k and ϵ numerical equations, that also have a suitable numerical kernel structure, can now be solved. The boundary and initial conditions for k and ϵ fields are respectively chosen as follows [10]:

- inflow:

  $$k_{in} = 1.5\ (T_{in}\ \langle U_{in} \rangle)^2 \qquad \nu_{T_{in}} = (0.1 \rightarrow 100)\ \nu \qquad \epsilon_{in} = C_\mu \frac{k^2}{\nu_{T_{in}}}$$

  with $\quad T_{in} = 10^{-6} \rightarrow 10^{-1}$

- solid walls:

  $$k = 0 \qquad \epsilon = 0 \qquad \nu_T = 0$$

- outflow: extrapolation is used.

Because some terms in transport equations (2.13) and (2.14) (or (2.16) and (2.14)) are divided by k and ϵ, a zero initial condition for k and ϵ fields can not be assumed. One of the possible choices is to adopt instead the maximum value that k and ϵ assumes on the boundary.

Chapter 4

Convection

Let's consider the simple advection equation of the momentum vector ρu_j

$$\frac{\partial \rho u_j}{\partial t} + \frac{\partial \rho u_i u_j}{\partial x_i} = 0 \tag{4.1}$$

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = 0 \tag{4.2}$$

by subtracting equation (4.2) from (4.1) it is obtained the equation

$$\frac{\partial u_j}{\partial t} + u_i \frac{\partial u_j}{\partial x_i} = 0, \tag{4.3}$$

in the 1D case it is obtained the inviscid Burgers' equation

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = 0 \tag{4.4}$$

or in conservative form

$$\frac{\partial u}{\partial t} + \frac{\partial (u^2/2)}{\partial x} = 0 \tag{4.5}$$

for which exact solutions exists.

The purpose here is to test advection schemes and their monotone properties. Godunov's theorem makes it impossibile for any high order (larger than 1) scheme which is not dependent from the solution itself, to satify these qualities. The usage of flux limiters is still one of the best options in order to satisfy these requirements but it leads to very tough programming tasks especially in the 3D case [7].

Such schemes can also be extended to a general 1D equation

$$\frac{\partial u}{\partial t} + \frac{\partial f\left(u\right)}{\partial x} = 0 \tag{4.6}$$

and regarding its discretization, a finite volume technique will be adopted: considering the integral form of (4.6) over the i^{th} computational cell

$$\frac{d}{dt}\int_{x_{i-1/2}}^{x_{i+1/2}} u\left(x,t\right)dx = f\left(u\left(x_{i-1/2},t\right)\right) - f\left(u\left(x_{i+1/2},t\right)\right) \tag{4.7}$$

integrating it in time from t^n to t^{n+1} and dividing by $\Delta x = x_{i+1/2} - x_{i-1/2}$ we obtain

$$\frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} u\left(x,t^{n+1}\right)dx - \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} u\left(x,t^{n}\right)dx =$$
$$-\frac{1}{\Delta x}\left[\int_{t^n}^{t^{n+1}} f\left(u\left(x_{i+1/2},\tau\right)\right)d\tau - \int_{t^n}^{t^{n+1}} f\left(u\left(x_{i-1/2},\tau\right)\right)d\tau\right].$$

Considering now the flux function time average

$$\tilde{f}_{i+1/2} = \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} f\left(u\left(x_{i+1/2},\tau\right)\right)d\tau$$

and the mean value of u over the i^{th} computational cell at time t^n

$$\bar{u}_i^n = \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} u\left(x,t^n\right)dx$$

the following expression is obtained

$$\bar{u}_i^{n+1} = \bar{u}_i^n - \frac{\Delta t}{\Delta x}[\tilde{f}_{i+1/2} - \tilde{f}_{i-1/2}]. \tag{4.8}$$

All the convection schemes differ in the way they evaluate the numerical flux F and/or its time averaged counterpart $\tilde{F}$ [12] and this will be examined in subsequent sections. But let's focus for a moment on the cell mean value $\tilde{u}$. In general this variable can be evaluated by means of a reconstruction of the function itself based on nodal values and subsequent integration, leading to a linear combination of nodal values of u with some weights. With equally spaced cells (or in general node centered ones) and adopting a linear reconstruction, the mean value $\bar{u}$ is equal to the nodal (cell center) value of the function $u = \bar{u}$. In this case the formula (4.8) can be obtained with a finite difference discretization technique, with central differencing discretization [5].

4.1 Burgers' Equation Test Case

The following analytical solution of the Burgers' Inviscid equation for the given initial condition represented in Figure 4.1 will be used as a test case

$$u(x) = \begin{cases} 0 & x \le x_i \\ (x - x_i)/t & x_i < x \le x_1 \\ U_{max} & x_1 < x \le x_2 \\ 0 & x_2 \le x. \end{cases} \tag{4.9}$$

In order not to make the expansion wave (departing from x_i) reach the discontinuity (departing from x_f) and not to make the latter reach the end of the domain ($x = L$), the final time of the simulation will be chosen as

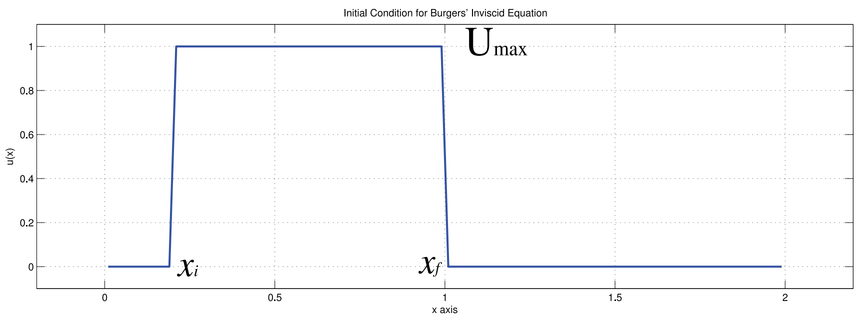


Figure 4.1: **Burgers initial condition with $L = 2$, $U_{max} = 1$, $x_i = L/10$, $x_f = L/2$.**

$$T_{fin} = \frac{2}{U_{max}} min(L - x_f, x_f - x_i)$$

4.2 Non Conservative Schemes

Lagragian approach

Considering an explicit time discretization of the (4.4), that is to say

$$u^{n+1} = u^n - \Delta t \left[u \frac{\partial u}{\partial x} \right]^n$$

which is equivalent, considering the first order Taylor's polynomial expansion, to the following expression

$$u(x)^{n+1} = u(x - u\Delta t)^n + O(\Delta t^2).$$

Dropping the second order term and introducing a spatial discretization finally yields

$$u_i^{n+1} = u^n|_{x_i - \Delta t\, u_i}$$

which is only first order accurate and of course the value $u^n|_{x_i - \Delta t u_i}$ must be obtained by interpolation. Higher interpolative orders guarantee a lower amount of artificial viscosity. This method is of course non conservative (there is no numerical flux to be estimated) and, accordingly, it is not shock capturing.

4.3 Conservative Schemes

All conservative schemes come down to the same time stepping method (see equations (4.6)-(4.8))

$$u_i^{n+1} = u_i^n - \lambda[\tilde{F}_{i+1/2} - \tilde{F}_{i-1/2}]$$

($\lambda = \frac{\Delta t}{\Delta x}$) and all the schemes that will be described will differ in the way the time averaged flux $\tilde{F}$ (over the time interval t^n and $t^n + \Delta t$) is being estimated at the control volumes surfaces.

Upwind Differential Scheme

The Upwind Differential Scheme (UDS) is a first order and very diffusive conservative scheme. The numerical flux is defined as follows

$$\tilde{F}_{i-1/2} = \frac{1}{2} u_{i-1}^{+} u_{i-1} + \frac{1}{2} u_i^{-} u_i \tag{4.10}$$

Lax Friedrichs Scheme

This scheme is a high order central explicit scheme with additional artificial viscosity meant to stabilize an otherwise totally unstable scheme. The numerical flux is

$$\tilde{F}_{i-1/2} = -\frac{\Delta x}{2\Delta t}(u_i^n - u_{i-1}^n) + \frac{\frac{1}{2}u_{i+1}^2 + \frac{1}{2}u_i^2}{2} \tag{4.11}$$

n^{th} **Order Upwind Interpolation Scheme**

The idea is to interpolate the value of $f_{i-1/2} = f_{i-1/2}^- + f_{i-1/2}^+$ at the face $i - 1/2$ with an upwind interpolation that is to say

$$F_{i-1/2}^+ = \begin{cases} \frac{1}{2}\left(f_i^+ + f_{i+1}^+\right) & \textit{linear} \\ -\frac{1}{8}f_{i-2}^+ + \frac{3}{4}f_{i-1}^+ + \frac{9}{8}f_i^+ & \textit{quadratic} \\ \frac{1}{16}f_{i-3}^+ - \frac{5}{16}f_{i-2}^+ + \frac{15}{16}f_{i-1}^+ + \frac{5}{16}f_i^+ & \textit{cubic} \end{cases} \tag{4.12}$$

$$F_{i-1/2}^- = \begin{cases} \frac{1}{2}\left(f_i^- + f_{i+1}^-\right) & \textit{linear} \\ +\frac{9}{8}f_{i-1}^- + \frac{3}{4}f_i^- - \frac{1}{8}f_{i+1}^- & \textit{quadratic} \\ +\frac{5}{16}f_{i-1}^- + \frac{15}{16}f_i^- - \frac{5}{16}f_{i+1}^- + \frac{1}{16}f_{i+2}^- & \textit{cubic} \end{cases} \tag{4.13}$$

The Richtmyer Two-Step Lax-Wendroff Method

The Lax Friedrichs Scheme is overall only first order accurate. The goal is now increasing the time-average accuracy with this second order scheme with

$$\tilde{F}_{i-1/2} = f(u_{i-1/2}^{n+\frac{1}{2}})$$

where

$$u_{i-1/2}^{n+\frac{1}{2}} = \frac{1}{2}\left(u_{i-1}^n + u_i^n\right) - \frac{\lambda}{2}\left(f(u_i^n) - f(u_{i-1}^n)\right) \tag{4.14}$$

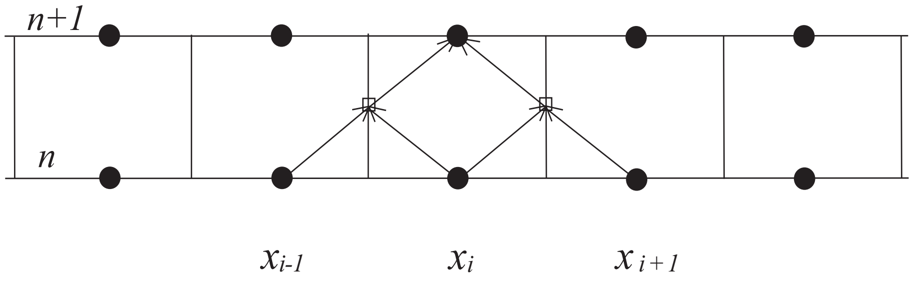


Figure 4.2: **the Richtmyer two-step Lax-Wendroff scheme.**

Multilevel Lagrangian Conservative Schemes

The Richtmyer scheme is extremely oscillating and a scheme which is more accurate in time and space, with a simple control on spurious viscosity effects, will be introduced. The first step is interpolating with a given order the nodal values of the velocity u from the first grid (*grid 1*) to a second grid (*grid 2*) which is derived from *grid 1* by refining the mesh with additional $nsbgr - 1$ cells or $nsbgr$ cell centers in between two cell centers of *mesh 1*

$$u_{i1}^{n} \quad \rightarrow \quad u_{i2}^{n}. \tag{4.15}$$

The time averaging of the flux will be carried out with Gaussian integration of the analytical flux $f(u)$ on the cell's face

$$\begin{aligned} \tilde{f} = \frac{1}{\Delta t}\int_{t^n}^{t^n+\Delta t} f(t)_{i-1/2}\, dt \simeq \dots \\ \dots \simeq \frac{1}{2}\left[f(u^{n+\frac{3-\sqrt{3}}{6}})_{i-1/2} + f(u^{n+\frac{3+\sqrt{3}}{6}})_{i-1/2}\right] = \tilde{F} \end{aligned} \tag{4.16}$$

this time averaging is third order accurate[1]. The values of the velocity at the Gauss points in time are calculated as follows:

[1]Usual Gaussian quadrature rule refers to points $\pm\sqrt{1/3}$ for the evaluation of $\int_{-1}^{1}\varphi(t)\,dt$. With the change of variables $\bar{t} = \frac{t+1}{2}\Delta t$ is posible to see that $\int_{0}^{\Delta t}\varphi(t(\bar{t}))\,d\bar{t} = \Delta t/2\int_{-1}^{1}\varphi(t)\,dt = \Delta t/2\left[\varphi\left(\frac{3-\sqrt{3}}{6}\Delta t\right) + \varphi\left(\frac{3+\sqrt{3}}{6}\Delta t\right)\right]$

$$u^{n+\frac{3-\sqrt{3}}{6}} = u^n \left(x_{i-1/2} - \frac{3-\sqrt{3}}{6} \, \Delta t \; u^*_{i-1/2} \right) \tag{4.17}$$

$$u^{n+\frac{3+\sqrt{3}}{6}} = u^n \left(x_{i-1/2} - \frac{3+\sqrt{3}}{6} \, \Delta t \; u^*_{i-1/2} \right) \tag{4.18}$$

$$\tag{4.19}$$

with

$$u^*_{i-1/2} = \frac{1}{2} \left(u_{i-1} + u_i \right) \tag{4.20}$$

and the values of $u^n \left(x_{i-1/2} - \frac{3-\sqrt{3}}{6} \, \Delta t \; u^*_{i-1/2} \right)$ and $u^n \left(x_{i-1/2} - \frac{3+\sqrt{3}}{6} \, \Delta t \; u^*_{i-1/2} \right)$ must be obtained with a *nearest value* interpolation from the u_{i2} array of interpolated velocities, providing the correct amount of artificial viscosity while preserving a high-order spatial and temporal resolution.

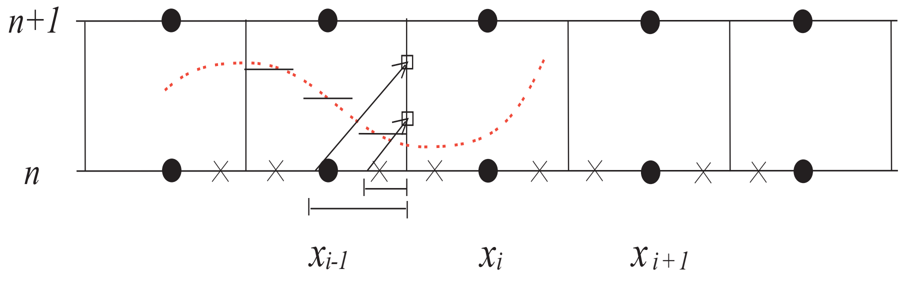


Figure 4.3: **Multilevel Lagrangian Conservative scheme.**

4.4 Flux Limiters

The purpose of Flux Limiters is to blend, by means of a *limiter function* ϕ, low order (monotonic but extremely diffusive) and high order (accurate but oscillating in proximity of discontinuities) approximations of the velocity flux. A splitting into postive and negative velocities is necessary. The numerical flux is estimated as follows

$$F^+_{i-1/2} = f^{LOW+}_{i-1/2} - \phi(r^+_{i-1/2}) \left[f^{LOW+}_{i-1/2} + f^{HIGH+}_{i-1/2} \right]$$
$$F^-_{i-1/2} = f^{LOW-}_{i-1/2} - \phi(r^-_{i-1/2}) \left[f^{LOW-}_{i-1/2} + f^{HIGH-}_{i-1/2} \right]$$

with

$$r^+_{i-1/2} = \frac{(u_{i-1} - u_{i-2})(u_i - u_{i-1})}{(u_i - u_{i-1})^2}$$
$$r^-_{i-1/2} = \frac{(u_i - u_{i+1})(u_{i-1} - u_i)}{(u_{i-1} - u_i)^2}$$

two algorithms for the evaluation of the *limiter function* have been implemented

SuperBee

$$\phi(r) = max[0, min(2r, 1), min(r, \beta_{sb})] \tag{4.21}$$
$$\lim_{r \to +\infty} \phi = \beta_{sb} \tag{4.22}$$

vanLeer

$$\phi(r)_{vl} = \frac{r + |r|}{1 + r} \tag{4.23}$$
$$\lim_{r \to +\infty} \phi = \beta_{vl}. \tag{4.24}$$

4.5 Schemes Comparison

In this section some numerical comparisons among the convection schemes above described are shown. The logarithm of the time average of the taxi-cab norm[2] of the difference between the analytic solution and the numerical solution (error), that can be be written as

$$\log\left(\widetilde{\|u_a - u_{num}\|_1}\right)$$

is reported in tables as confronting index of good approximation for every scheme being tested.

Prediction correction schemes

With these schemes a lagrangian prediction of u values at t^{n+1}, with different interpolating orders, is made, then both fluxes at times t^n and t^{n+1} with a particular reconstruction order, are evaluated, and finally the temporal mean with a trapezoidal time integration is obtained.

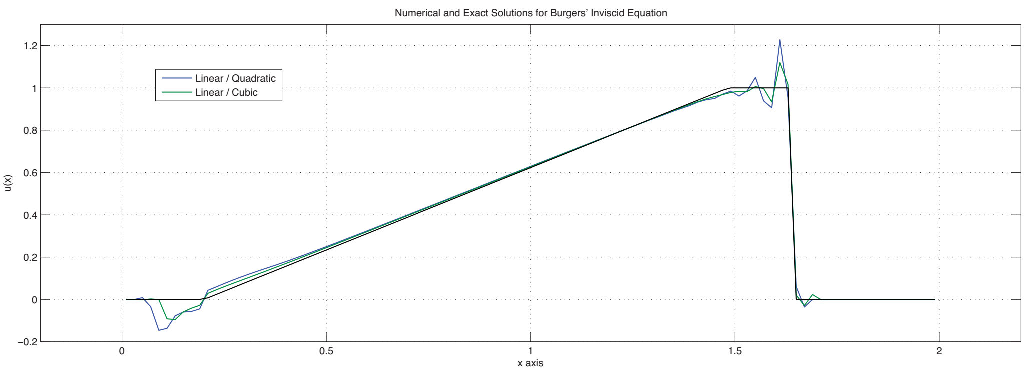


Figure 4.4: **prediction (linear) - correction.**

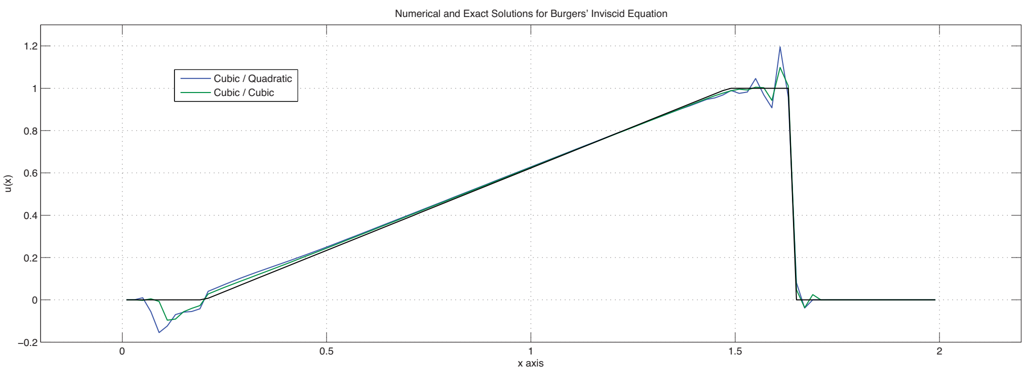


Figure 4.5: **prediction (cubic) - correction.**

Flux Order↓ \ Predictor→	NEAREST	LINEAR	CUBIC
LINEAR	n/a	n/a	n/a
QUADRATIC	n/a	0.0521	0.0495
CUBIC	n/a	0.0373	0.0350

The not available symbol n/a indicates that the simulation is unstable or the results are very poor. This is the case respectively, for the linear flux reconstruction (that means central differencing) and for the nearest reconstruction.

Prediction correction schemes with limiters

Here two different limiters (van Leer and SuperBee) are applied to some prediction-correction schemes with different interpolating order on the lagrangian prediction and with quadratic order for fluxes evaluation.

[2]The taxi-cab norm or 1-norm of vector v is $\|v\|_1 = \sum_i |v_i|$

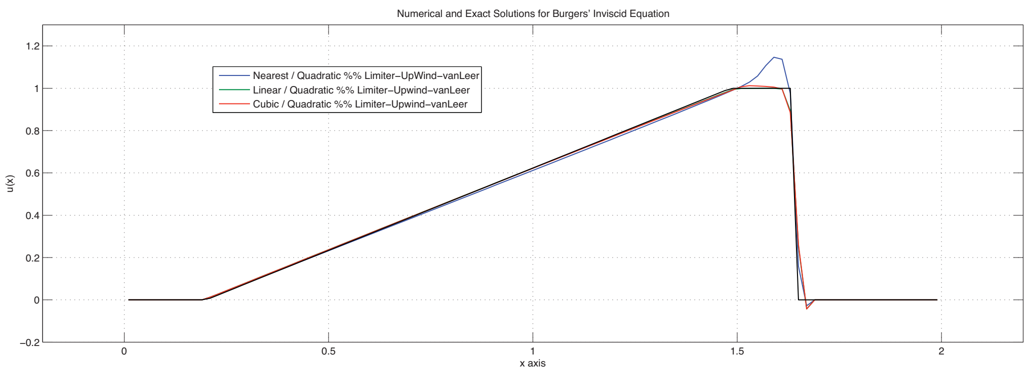


Figure 4.6: **prediction - correction with quadratic order reconstruction and flux limiter van Leer.**

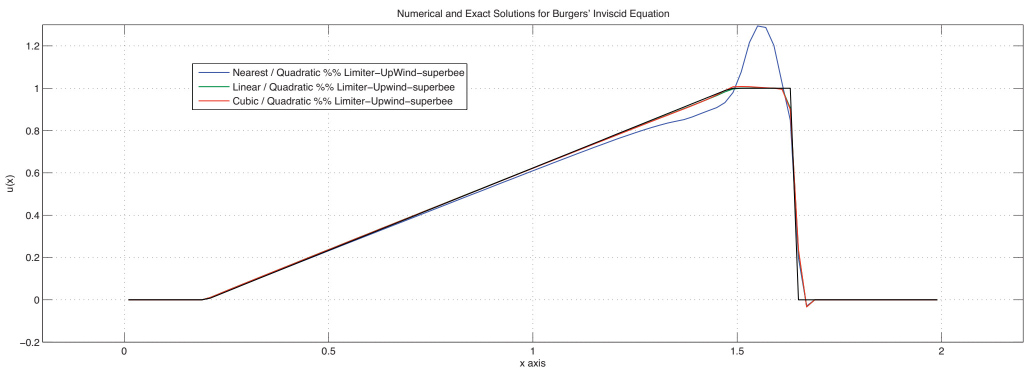


Figure 4.7: **prediction - correction with quadratic order reconstruction and flux limiter SuperBee.**

Predictor↓ \ Limiter→	van Leer	SuperBee
NEAREST	0.0313	0.0378
LINEAR	0.0162	0.0153
CUBIC	0.0175	0.0157

Multilevel Lagrangian Conservative Schemes (MLCS)

In this case MLCS schemes with 2 and 3 sub-grid points and different reconstruction orders, are compared with the Richtmyer two-step Lax-Wendroff method

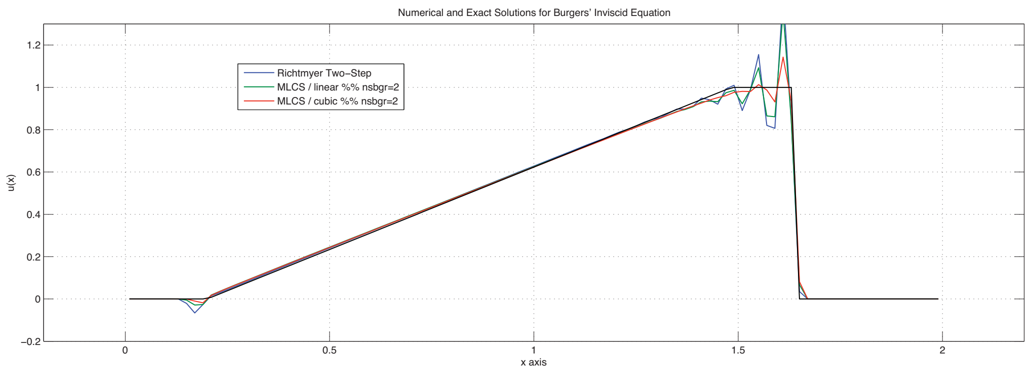


Figure 4.8: **Multilevel Lagrangian Conservative scheme with 2 sub-grid points vs. Lax-Richtmyer.**

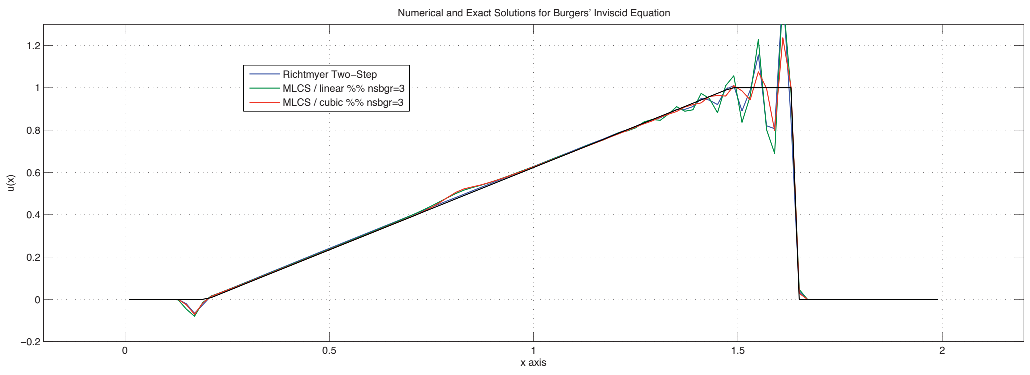


Figure 4.9: **Multilevel Lagrangian Conservative scheme with 3 sub-grid points vs. Lax-Richtmyer.**

Predictor↓ \ n. Sub-grid→	2	3
LINEAR	0.423	0.628
CUBIC	0.0277	0.0366

Lax-Richtmyer → 0.0521

Multilevel Lagrangian Conservative Schemes (MLCS) with limiters

Here the van Leer and SuperBee limiters are applied to the above tested MLCS schemes

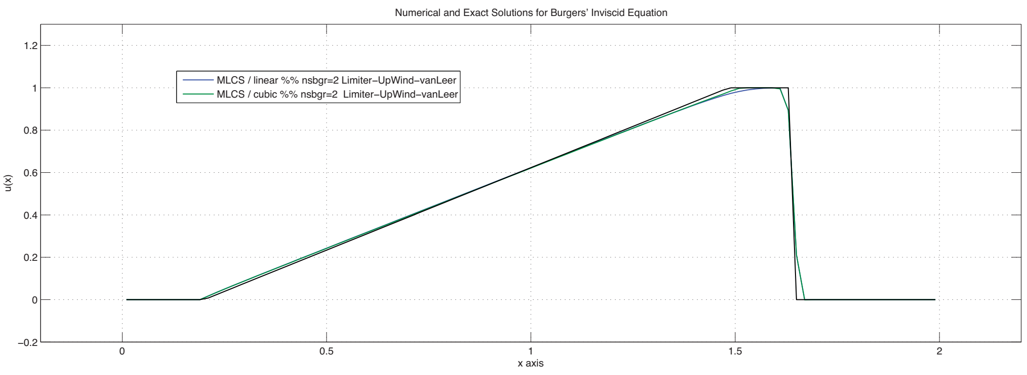


Figure 4.10: **MLCS with 2 sub-grid points and van Leer limiter.**

Predictor↓ \ n. Sub-grid→	2	3
LINEAR	0.0214	0.0163
CUBIC	0.0194	0.0143

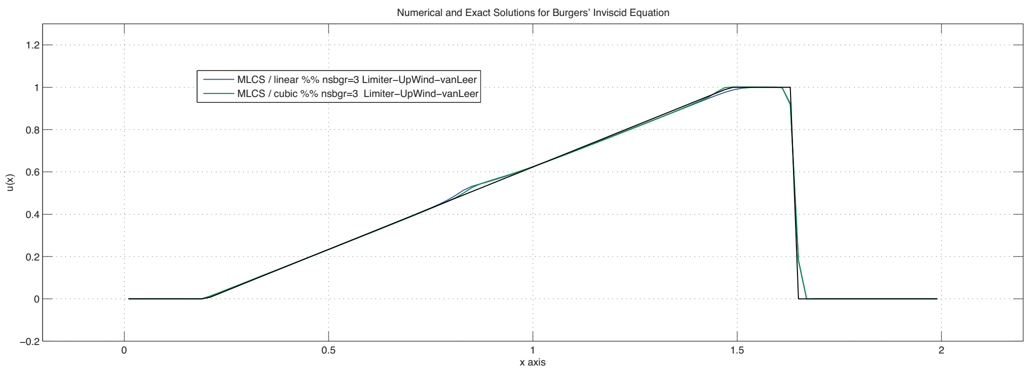


Figure 4.11: **MLCS with 3 sub-grid points and van Leer limiter.**

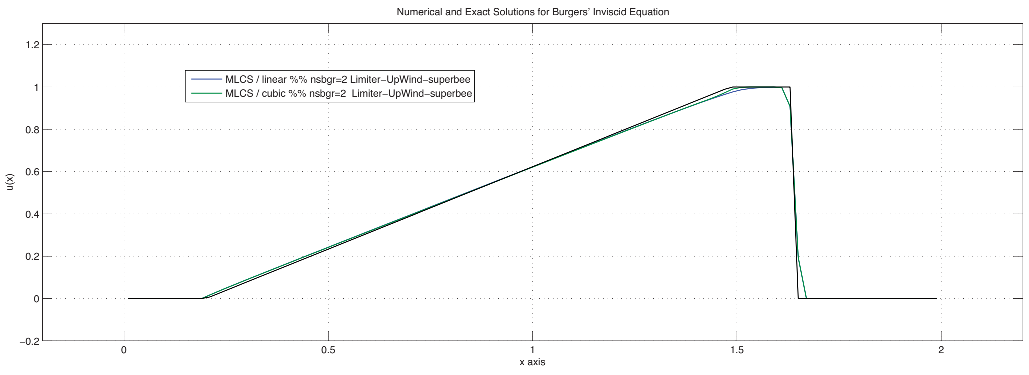


Figure 4.12: **MLCS with 2 sub-grid points and SuperBee limiter.**

Predictor↓ \ n. Sub-grid→	2	3
LINEAR	0.0206	0.0154
CUBIC	0.0189	0.0148

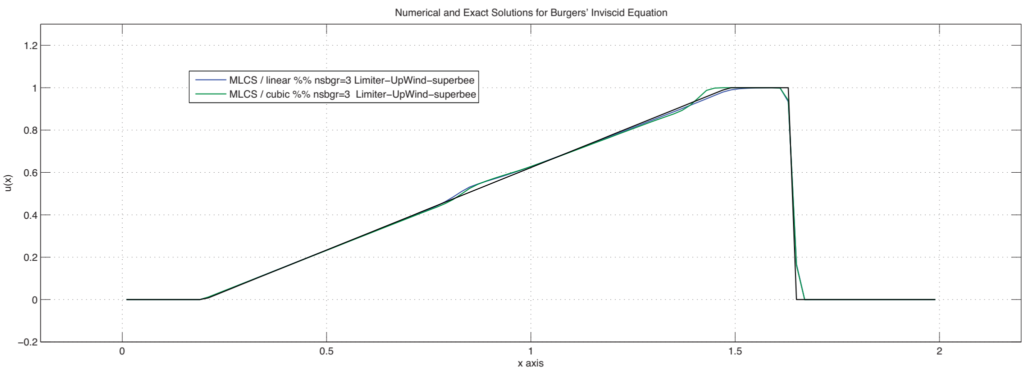


Figure 4.13: **MLCS with 3 sub-grid points and SuperBee limiter.**

Chapter 5

Code overview

One of the primary reasons for writing a computational fluid dynamic code from scratch in an high level algebraical language such as Matlab, Scilab or Python, is to exploit their characteristics of clean and simple but also effective coding. Using these kind of languages is a suitable choice taking especially into account their "natural" handling of fundamental linear algebra objects like matrices, and the extensive amount of libraries and functions available for a lot of simple and complex operations like matrix manipulation or 3D graphics and visualization.

5.1 PRIN-3D General Design

One of the most important characteristics that has to be included in the design of the code is modularity; the idea is to write a code that enables the user to perform several tests ranging from algebraic analysis of the equations' structure to modular implementation of virtually any kind of Fluid Dynamic model. With this in mind, we can take a tour of the basic structure of the code and start with a basic user-input example.

5.1.1 3D Computational Domain

Three-dimensional geometry handling is one of the toughest steps in a CFD code design. The purpose was not to realize an "industrial" CFD code, so the idea of dealing with a totally generic geometry has been dropped. An effort has been made to guarantee the maximum flexability with respect to the user's demands in within a certain class of 3D domains. The actual version of the code is intended to handle a specific class of three-dimensional domains, *right prisms*. The user defines a basic 2D polygon (with points assigned in a counterclockwise manner in the Y-Z plane) and the height of the prism which extends along the X-dimension, as illustrated in Figure (5.1).

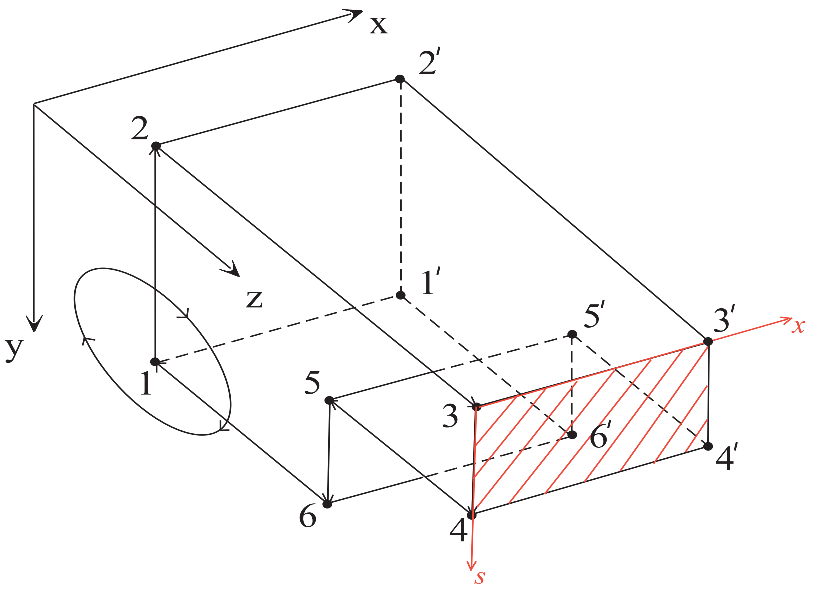


Figure 5.1: **Computational domain example**

There are essentially two **types** of boundary faces in this class of domains:

1. **type A :** faces with normals orthogonal to the X-axis, which are always rectangular

2. **type B :** faces with normals parallel to the X-axis, which are polygons defined with subsets of points taken from the previously defined basic 2D polygon (this will be better explained later on)

Boundary conditions available for the velocity's components *u,v,w* are *Dirichlet* (for all faces) and *Extrapolation* (for Outflow faces only) [1]. It is possible to have multiple inflows/outflows but, due to the actual variable collocation, which will be discussed later on, the code works better if different inflows or outflows are not defined on adjacent faces. Every point on type B faces is defined by the set of (dimensional) coordinates (z,y) and dirichlet boundary conditions for the three velocity components can be defined as functions of (z,y). These set of faces are not rectangular in general, they can be of any polygonal shape and this is user-defined. Exactly the same thing can be done for type A faces which, on the contrary, are ALL rectangular. A set of *dimensionless* coordinates (x,s) can then be defined to locate any point on their surface. For example, the set of (x,s) coordinates has been drawn on a type A face in Figure (5.1) and the coordinates (x=0,s=0) indicate the upper-right vertex (point 3), whereas the coordinates (x=1,s=1) the lower-right vertex (point 4'). An example of a kind of velocity outlet that can be assigned on a face normal to the Z-axis is showed in Figure (5.2).

It is also possible to assign *lid driven* faces or even *swirled.* Extrapolation for outflow faces has proven to be quite effective with the implicit time stepping scheme (see Chapter 3 for further details) but it was originally meant to be used with explicit schemes. This particular boundary condition (see Figure 5.3) consists in using the velocity profile at time t^n in flow section 'P' immediately before (with respect to the flow direction) the outlet section 'O' as the dirichlet boundary condition for the ouflow face at time t^{n+1} (this

[1] *Pressure Inlet* and *Neumann* are supported in the code but non tested yet

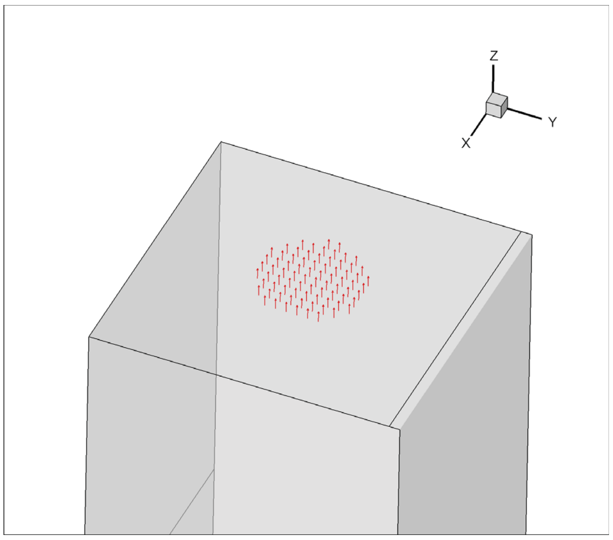


Figure 5.2: **W component hat function given on a outflow face**

is why this is NOT an homogenous Neumann boundary condition!). Same boundary conditions apply for transported scalars and other model-related quantities (Turbulent Energy K, Energy Dissipation ε, Mixture Fraction Z, etc...) and have to be defined by the user.

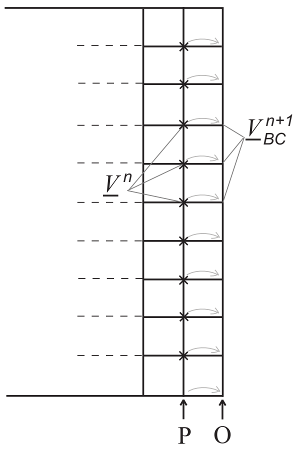


Figure 5.3: **Extrapolation for outflow faces**

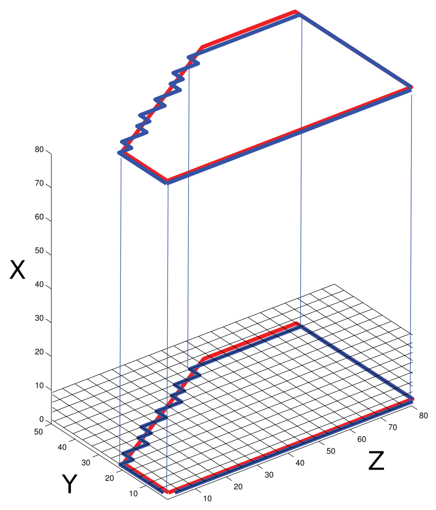


Figure 5.4: **Staircase approximation of user defined domain boundary.**

The computational grid is a 3D block-structured grid obtained by a staircase approximation of the user defined boundary geometry and the mesh is uniformally spaced ($\Delta x = \Delta y = \Delta z$). This allows an isotropical distrubution of the truncation error of the discrete spatial operators, simplicity in the code design and it is very suitable for Large Eddy Simulations. In Figure 5.4 the red line is the user defined polygon which is repeated along the X-direction whereas the blu line is the actual staircase approximation and our real computational domain boundary.

In the case of velocity boundary conditions, for example, the user has to specify for every face of the domain a two-variable function (function of dimensionless coordinates (x,s) for faces of type A, function of dimensional coordinates (z,y) for faces of type B) that defines the boundary condition itself and an integral mean value used to calculate the integral velocity flux contribution of that particular face. Once the association of the boundary velocities on the multi-rectangle is made, these are adjusted in order to account for the divergence-free constraint.

5.1.2 Grid System

There are typically three possibile variable arrangements for structured grids: *Colocated*,*Partially-Staggered* and *Staggered*. PRIN-3D can switch from Control Volumes to Finite Difference method according to what advection scheme is adopted [5]. Nodes can therefore be intended as CV centers or collocation points of the spatial discretization of the PDE we intend to solve.

- **Regular Grid System** This variable arrangement is by far the most developer-friendly but troublesome choice. In the 3D case there is only one mesh to handle and velocity boundary conditions are very easy to assign but on the other hand having pressure nodes right on the domain boundary may force an embarrassing assignment for pressure boundary conditions which is not normally done in the incompressible case. The worst effect of this arrangement is the well-known checkerboard effect on the pressure field

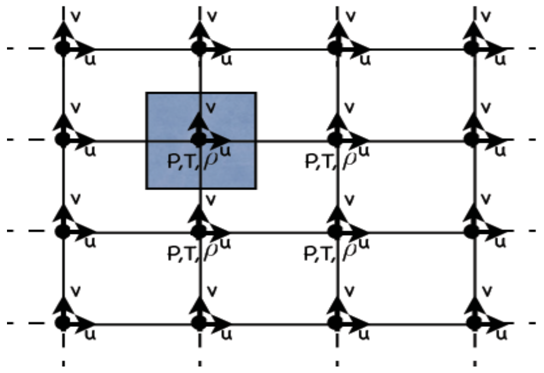


Figure 5.5: **Regular Grid System**

- **Staggered Grid System**

Such arragement (Harlow-Welch arrangement) allows very reduced usage of interpolations for flux computation on cell faces and the pressure field is totally oscillation-free. The worst drawback is the handling of four different computational grids (for the *u,v,w,p* variables) which makes the boundary conditions treatment quite troublesome from a programmer's point of view. The overall system of equations in this case leads to a rank deficient matrix which can be made invertible and good conditioned by simply specifying the pressure value in a single node.

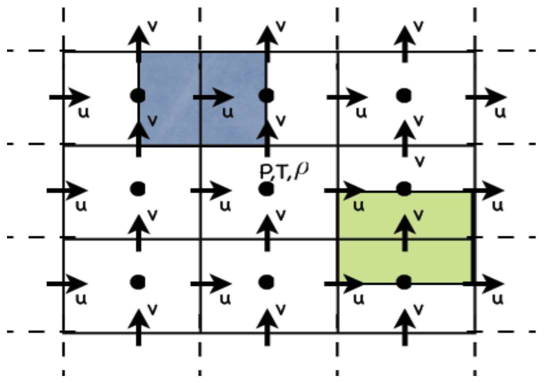


Figure 5.6: **Staggered Grid System**

- **Partially Staggered Grid System**

  With this arragement there are only two grids (pressure grid and velocity grid) and velocity boundary conditions are easily defined keeping at the same time pressure nodes inside the computational domain. This arragement gives birth to highly rank defective system of equations with pressure checkerboard effect.

The best choice in order to achieve the best numerical results would be the Harlow-Welch arragement but this choice has been dropped in order to

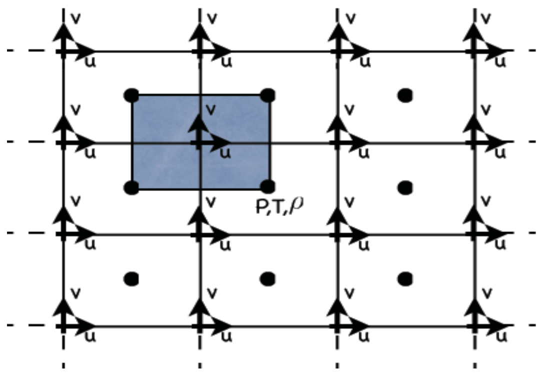


Figure 5.7: **Partially Staggered Grid System**

prevent the programming efforts from being exclusively focused on the mesh generation and handling. The regular grid system option, though extremely developer-friendly, would have lead to poor numerical results. The Partially staggered grid system appears to be a fair trade off between these two choices.

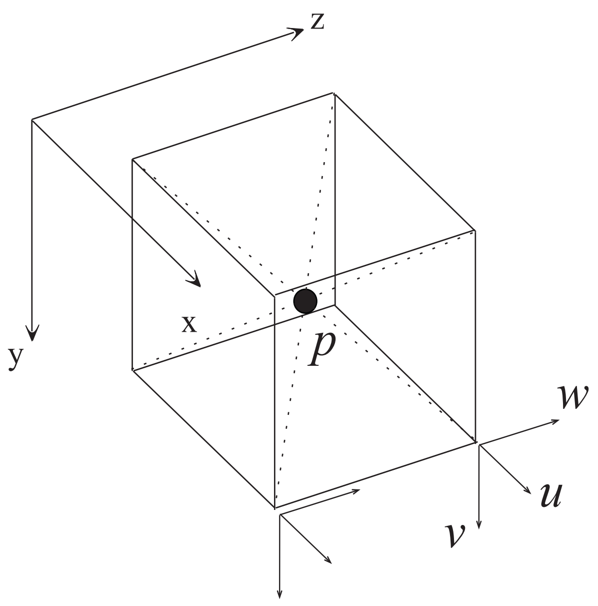


Figure 5.8: **3D Partially Staggered Grid System**

The main drawback are oscillating pressure fields which can be efficiently smoothed by linearly interpolating (8 point average) its values on the velocity nodes and adopting sparse approximate inverses that give a biharmonic structure to the pressure equation (see Chapter 3 for further solver related issues).

5.1.3 Discrete Operators

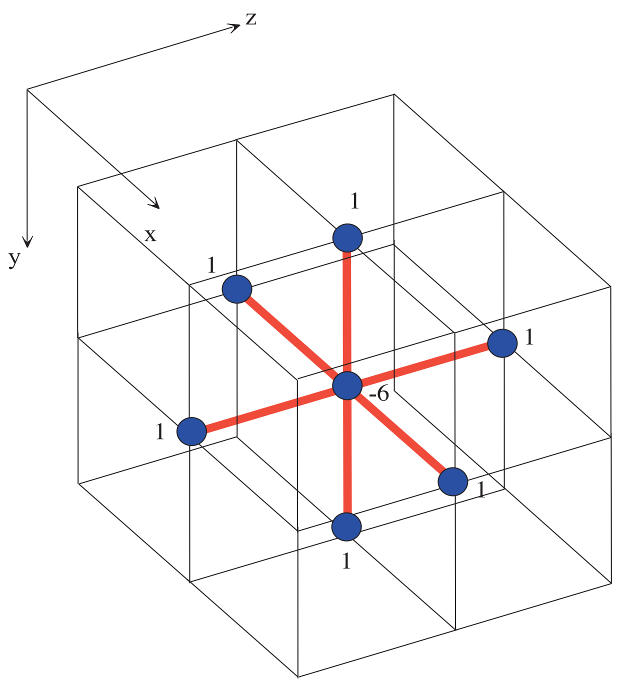


Figure 5.9: **Laplacian 3D Stencil**

The control volume for the momentum equations is a cube centered in the velocity nodes with vertices being the 8 surrounding pressure nodes (see Figure 5.8). The discretization of the diffusive contribution is shown in Figure 5.9 where the central velocity node is labeled with the -6 weight. The dis-

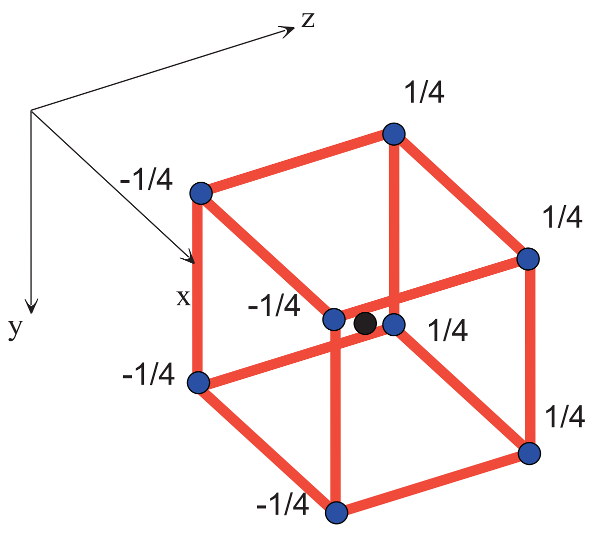


Figure 5.10: **Z-Divergence operator / Pressure Gradient - 3D Stencil**

crete pressure gradient is computed by a central difference of 4-averaged pressures: for example, to compute the pressure gradient along the Z-direction in a velocity node (say the black one in Figure 5.10 the weights are $\pm 1/4$ for pressure nodes (in blue) surrounding the velocity node. Interpreting, instead, in Figure 5.10 blue nodes as velocity nodes and the central black node as a pressure node the divergence stencil (only the derivative with respect to Z) is obtained. Such Divergence operator is the opposite of the transpose of the Pressure Gradient one.

5.2 User Input Example

The best way of getting started with PRIN-3D is to guide the reader through a demostrantive simulation that is available in the first code release (apart from many others). Everything the user needs to do is to fill in a *user input* m-file. We will start with a simple incompressible fully three-dimensional flow that can show the code's capability. Open from the Matlab editor the

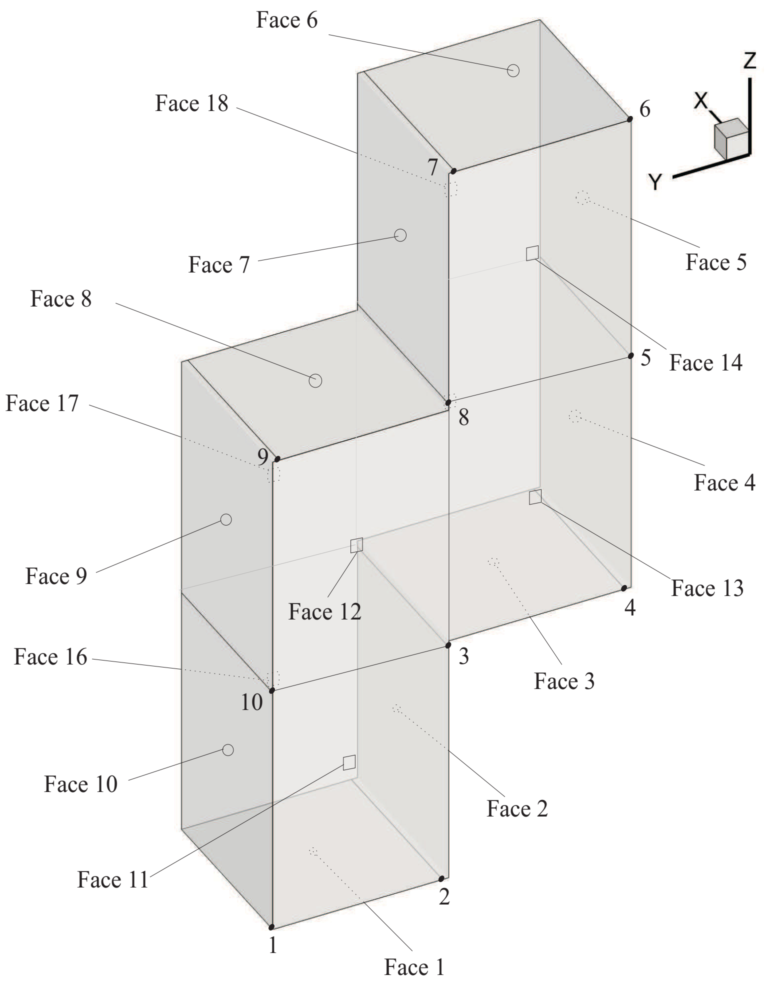


Figure 5.11: **Fully-3D simulation**

`Mainuvwp.m` file and uncomment under the `‘% User Defined Script’` section just the code line `‘uifile=‘Fully3D.m’;’` which will load the user defined geometry and boundary condition data contained in such script once the main script `Mainuvwp.m` is launched. By starting the `Mainuvwp.m` script the program will start and at specific iteration itervals a dumping of processed data will occur on the hard drive in ascii format and the simulation will stop with a user defined criterion. But let’s analyze the user defined script in the first place. All the user defined parameters of this example can be seen in the file `./UserInput/Fully3D.m`. Let’s analyze, chunk by chunk, this file, which is the only file the user has to edit. The file starts with the following code lines:

```
% Fully3D Demo

% Box Dimensions
Lx=.5; Ly=1; Lz=2;
%                                                        (    ∧ )
% YZ-Plane polygon, must be assigned counter-clockwise in the |y−z− > plane
y1=.5; z1=1/3; z2=2/3;
yp=[Ly y1*Ly y1*Ly 0 0 0 y1*Ly y1*Ly Ly Ly Ly ];
zp=[0 0 z1*Lz z1*Lz z2*Lz Lz Lz z2*Lz z2*Lz z1*Lz 0];
xp=[0 Lx]; nfaces=10+4+4;

% Containing Mesh Factor
cmshf=.2;
```

Reference lengths `Lx, Ly, Lz` for each dimension MUST be assigned in order to give roughly the maximum extent, along each direction, of the computational domain. The two arrays `yp, zp` describe the 2D basic polygon that

MUST be assigned in a counterclockwise manner in the Y-Z plane as shown in Figure 5.1. The `xp` array contains just two entries specifying the position along the X-coordinate of the two Y-Z boundary faces. In general such planes will always be located at `0` and `Lx`. The value of the Containing Mesh Factor `cmshf` will be used by the `./GeoMesh&BCProcessing/MeshOperGen3D.m` routine to define the actual boundaries of the containing mesh which is a rectangular prism defined by the following set of coordinates `(xm,ym,zm); (xM,ym,zm); (xm,yM,zm); ...; (xM,yM,zM)` that in a Matlab language style are:

```
% Containing Mesh Definition
xm=-cmshf*Lx+min(xp); xM=cmshf*Lx+max(xp);
ym=-cmshf*Ly+min(yp); yM=cmshf*Ly+max(yp);
zm=-cmshf*Lz+min(zp); zM=cmshf*Lz+max(zp);
```
[2]

Faces 1 to 10 (type A faces) are automatically defined but still we haven't yet decided what are the set of vertices of the basic 2D polygon that make up the several lateral faces, i.e. type B faces, from 11 to 14 which are identical to, respectively, faces 15 to 18. This is done in the following code lines:

```
nfl=4;
maskptinfl=[1 0 0 0 % 1
1 0 0 0 % 2
1 1 1 0 % 3
0 0 1 0 % 4
0 0 1 1 % 5
```

[2]these are not the actual lines in the `MeshOperGen3D.m` routine, they are meant for explanatory purposes only

```
0 0 0 1 % 6
0 0 0 1 % 7
0 1 1 1 % 8
0 1 0 0 % 9
1 1 0 0]; % 10
maskptinfl=logical(maskptinfl);
```

The variable `nfl` defines the number of distinct lateral faces (*in this case* it is 4 but we could have defined any number of faces from 1 to 8 [3]) and variable `maskptiinfl` must be a `nfaces`-by-`nfl` logical matrix whose i-th column has `1` in every position indicating the set of vertices forming the `(i+nfaces-2 nfl)`-th face. In this case there are 10 faces of the first type and these are the first in the boundary face numbering sequence. The first column of `maskptiinfl` refers to the first face (in the numbering sequence) of the second type, i.e. Face 11. Only faces from 11 to 14 need to be defined, since, as previously stated, faces from 15 to 18 are one by one respectively identical. For example, by calling `find(maskptinfl(:,3))` the numbers corresponding to the set of points that define Face 10+3 will be printed to the screen.

We want to give inflow Dirichlet boundary conditions on face 15 and similar outflow conditions for face 5 and 6. Just to make things a little more tricky we will give circular plug flow jets. The code lines to do this are:

```
% Velocity BC and Fluid Info
mu0=2.303724665081446e-04; % Tuning this value will change the Reynolds Number
rho0=1.205; % Physical value for air density [kg/m^3]
```

[3] 1 lateral face would be a face having the user defined basic 2D polygon as its edge; 8 lateral faces would be a set of triangles made up, for example, by the following sets of nodes {1,2,10},{2,10,3},{3,10,9},{3,8,9},{3,8,4},{5,8,4},{5,8,7},{5,6,7}

```
ni0=mu0/rho0;
Vin=-.001;
Vout=.0005;
ain=z1*Lz/4; % inflow Z semi-axis
bin=(1-y1)*Ly/4; % inflow y semi-axis
xin=.25;
sin=.25;
z0=.5*z1*Lz; y0=.5*(1-y1)*Ly+y1*Ly; % inflow orifice center
hVin= (z,y) Vin*hat((((z-z0)/ain).^2+((y-y0)/bin).^2),1); % Y = hat(X,Delta)
hVout= (x,s) Vout*hat((((x-.5)/xin).^2+((s-.5)/sin).^2),1);
ReLref=2*mean([ain,bin]);
```

In order to assign a velocity profile on Face 15 we need to use dimensional coordinates (z,y) and a handle function of such coordinates must be defined. For the inflow, the handle is 'hVin' (as we will se later on, the name is unessential) and for faces in the Z-Y plane (i.e. type B) the (z,y) coordinates must indicate the actual position on the face and must be dimensional. The function `hat(x,L)` defined in the folder `./Other` returns 1 if $-L/2 < x < L/2$ otherwise 0. The inlet plug flow is defined using the ellipse' formula where `z0` and `y0` clearly define the center of Face 15 and `ain` and `bin` the ellipse's (dimensional) semi-axes. The same thing is done for the outflow faces 4 and 6 for which an identical handle can be defined but the coordinates to use are (x,s) (since Faces 4 and 6 have a normal orthogonal to the X-axis thus belonging to the type A boundary faces) which are both dimensionless (both ranging from 0 to 1). The center of such face is simply located at (.5,.5).
Moreover, this is where the user has to assign the reference length for the Reynolds number, which is computed later on after the velocity boundary conditions have been corrected. The Reynolds number being displayed in the

code's text output is based on such reference length and on the maximum boundary velocity module.

We now need to assign such handles to each of the faces and to assign a desired flow rate. It is possible to assign the value of each velocity component on every previously defined face but the user will not have to give such values in terms of u, v, w but rather in terms of V_n, V_t, V_e where V_n is the velocity component normal to the face and pointing outside of the domain and V_t, V_e are the two remaining tangential components. Figures 5.12 and 5.13 show the rule adopted for the positive values of such velocity components.

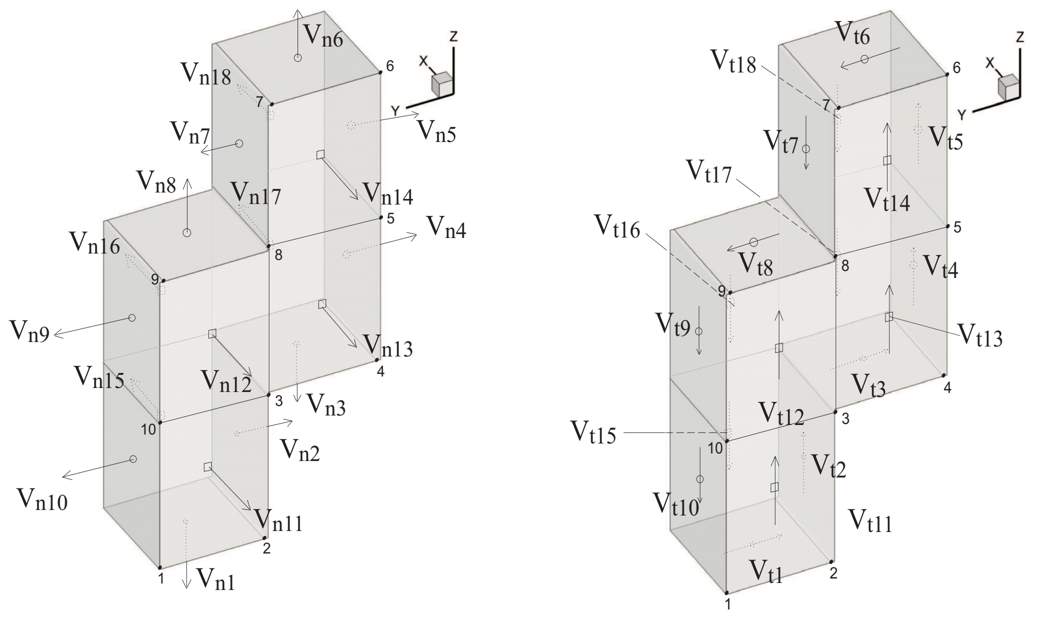


Figure 5.12: **Positive V_n and V_t direction**

All the handles associated with every component are predefined as zero-functions in the following preprocessing code lines (which shall not be included in the *user input* m-file)

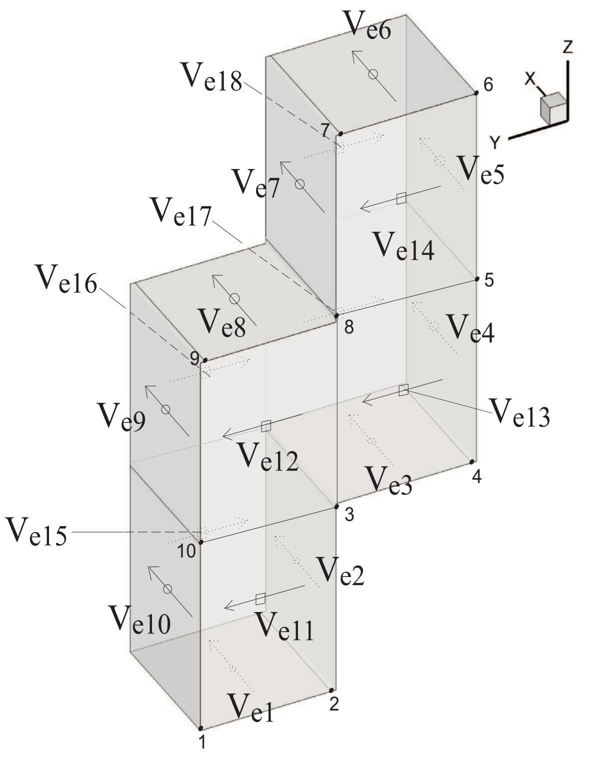


Figure 5.13: **Positive V_e direction**

```
chVn=cell(1,nfaces);
for ic=1:nfaces, chVnic=null; end; chVe=chVn; chVt=chVn;
Vn=zeros(1,nfaces); Ve=zeros(1,nfaces); Vt=zeros(1,nfaces);
```

Variables `chVn`,`chVt`,`chVe` are preallocated `1`-by-`nfaces` cell arrays of handles that return flat zero velocity profiles on every face. The user will assign a predefined function handle (in this example this has already been done with `hVin` and `hVout`) to be inserted in the correct entry of `chVn`,`chVt`,`chVe`. Variables `Vn`,`Vt` and `Ve` are simply `1`-by-`nfaces` arrays of doubles which are intended as the surface-integral-mean values (i.e. flow rates in the case of normal velocity components) assigned by the user for each face. Now, if the flag variable `Qcheckflag` is set to 0, these values are corrected (in order for the total flow rate to sum up to zero for the divergence free constraint) by the routine `./GeoMesh&BCProcessing/vdirich.m` and then the routine `./GeoMesh&BCProcessing/Qcheck.m` (the trickiest one of the whole code) corrects the actual velocity boundary vectors on the multi-rectangle so that their surface integral (that MUST be computed with the same criterion used to approximate the divergence operators) for each face is exactly the one computed by the `vdirich` routine.

In order to assign handles `hVin`, `hVout` and define the values of the surface-integral-average of the velocity components keep in mind the number of the faces on which such boundaries conditions want to be imposed and, for this example, such assignement will be as follows:

```
Vn([15 4 6])=[Vin,Vout,Vout];
chVn([15 4 6])=hVin,hVout,hVout;
chV=chVn;chVe;chVt;
Qcheckflag=0
```

Setting `Qcheckflag` to 1 will force the inflow boundary conditions to be exactly how the user has defined them and correct only the outflow faces in order to satisfy the divergence-free constraint.
Another step, essential for simulating reactive flows, but still needed in order for a simple incompressible flow simulation to run is the assignment of the boundary conditions for dynamic viscosity μ and density ρ done as follows:

```
% BC for Fluid density and viscosity
chMu=challoc([1,nfaces],mu0);
chRho=challoc([1,nfaces],rho0);
Rhobc=ones(1,nfaces)*rho0;
```

```
Mubc=ones(1,nfaces)*mu0;
```

where the function `challoc([n,m],v)` stored in the `./Other` folder creates an n-by-m cell array of handle functions of two coordinates (x,s) or (y,z) returning all always the same constant value `v`. In the incompressible case, of course, density and viscosity on the boundary is all set to `mu0` and `rho0`. We then need to specify the type of boundary condition for each face:

```
%%% specify the type of boundary condition for each face
% 'D' -> dirichlet
% 'E' -> extrapolation
bctype=repmat('D',[1,nfaces]);
```

Faces labeled with `'D'` are Dirichlet faces and boundary conditions for these faces are stationary throughout the whole simulation. Boundary condition on faces labeled with `'E'` (that must be outflow faces) will be updated as previously described, at every step. Finally we have to specify the type of face, whether it has to be treated as a wall, an inflow/outflow face or a lid driven face.

```
%%% specify the label the for boundary face
% 'W' -> wall
% 'I' -> inflow
% 'O' -> outflow
% 'L' -> lid
bclabel=repmat('W',[1,nfaces]);
```

```
bclabel(15)='I'; bclabel([4,6])='O';
```

We now want to decide the mesh size. By specifing the value of `nng` in the following lines

```
nng=160;
maskcnc=false(2,10);
```

we choose a mesh size `h` which is computed in the following manner

```
h=max([xM-xm,yM-ym,zM-zm])/(nng-1)
```

For approximately cubic geometries `Lx`∼`Ly`∼`Lz` the number of unknowns will be very sensitive to the value of `nng`. The variable `maskcnc` is a 2-by-`(nfaces-2 nfl)` logical matrix that, for this particular simulation, can be set completely to false. As it will be illustrated in other tutorials, that will be available on-line, it will be necessary to correct some vectors that are assigned by nearest interpolation to the multi-rectangle. For instance by setting maskcnc(2,5)=1 the code will erase all the velocity boundary vectors in the last set of points aligned along the X-direction of Face 5 (this applies only to faces of the type A) i.e. all the points at s=1 in face 5.[4] If we were dealing with geometry in Figure 5.1 this would erase all the boundary velocity vectors on the 6-6' segment.

Now the geometry and boundary definitions are all set. The very last step

[4]This works perfectly for faces aligned with the mesh whereas for oblique inflow/outflow faces velocity flux correcting routines still need to be upgraded

is to define the solver options, the model-related parameters and many other parameters. The basic options are:

```
%%% BASIC OPTIONS
invmethodImplicitoSpai=struct('N',1,'N_lhs',0,'press_ord',2,'parallelize',0);
itrslvr=struct('iteri_max',1,'iterk_max',1,'iterl_max',1);
options=struct('Solver',0,'invmethod',invmethodImplicitoSpai,'itrslvr',itrslvr,
'typgeoin','Extrusion','Periodic','no','TimeSteppingMethod','Implicit','Model',
'Laminar-Incompressible','rTfin',10,'istatsupdate',200,'dEkdt_tol',0.01,'beta',0.02,
'rdt',.2,'inormerrp',1e06,'isumdiv',1e06,'igraf',1e06,'isave',200,'toldiv',1e-004,
'th',.5,'Qvischeck',1,'maskcnc',maskcnc,'typcflux','Lagrangian','interpflux','linear',
'limitertype','superbee');
```

Structures `invmethodImplicitoSpai` and `itrslvr` define specific options of the iterative solver (see Chapter 3 for further details), in general the setting showed here are pretty much the optimum for almost every case.

- `invmethodImplicitoSpai`
  - 'N' : Order of the approximate sparse inverse
  - 'N_lhs' : Laplacian's power to be left on the right hand side of (3.17)
  - 'press_ord' : Pressure variable ordering (i.e. number of pressure colors)
  - 'parallelize' :
    - 0 : Sequential block Gauss-Siedel solving
    - 1 : Parallel block Jacobi solving. This is possible only if the Star-P software has been installed with Matlab. Star-P is a client-server parallel-computing platform that's been designed

to work with high level languages (hll) such as MATLAB®, or Python and has built-in tools to expand hll computing capability through addition of libraries and hardware-based accelerators. The programming effort in setting up such parallel computation is very low since there is absolutely no need for an MPI based cluster expertise management. A beneficial description of Star-P for many users is that Star-P is a global array syntax language. By providing a global array syntax in Star-P, the user variable *App* refers to the entirety of a distributed object on the back end server. The abstraction of an array that contains many elements is a powerful construct. With one variable name such as *App*, you are able to package up a large collection of numbers. This construct enables higher level mathematical operations expressed with a minimal amount of notation. On a parallel computer, this construct allows you to consider data on many processors as one entity. By contrast, message passing or "node-oriented" languages force you as a programmer to consider only local data and create any global entity completely outside the scope of the language. Data is passed around through explicit calls to routines such as send and receive or SHMEM get and put. The lack of support for the global entity places more of a cognitive burden on you, the programmer. Star-P allows users to implement their programs in parallel without having to master the intricacies of MPI in Fortran, C, or C++ [1].[5]

- `itrslvr`
  - 'iteri_max' : maximum number of i-iterations

[5] for more details see http://www.interactivesupercomputing.com/

- 'iterk_max' : maximum number of k-iterations
- 'iterl_max' : maximum number of l-iterations

These structures are inserted in the `options` structure that contains the other following fields:

- `'Solver'`
  - `0` : Standard Approximate Sparse Inverse solver (see Chapter 3);
  - `1` : Full pressure segregation, extremely slow and memory demanding, only to be used in the 2D case! (see Chapter 3 for further details);
  - `?` ... whatever new solver you would like to implement;
- `'typegeoin'`
  - `'Extrusion'` : this field specifies basically the role of the `MeshOperGen3D.m` routine which interprets the user-input data as previously described *extruding* the 2D Y-Z polygon assigned by the user;
- `'Periodic'`
  - `'no'` : all user defined boundary data will be respected, all operators are created with Dirichlets boundary conditions;
  - `'x-x'` : all of type B faces become periodic, this is needed when the user wants to run essentially 2D simulations. This is done by assigning 2D boundary conditions, i.e. all boundary velocity vectors orthogonal to the X-direction.
- `'TimeSteppingMethod'`
  - `'Implicit'` : Implicit discretization is essentially allowed by the linearity of the diffusive flux and allows larger time steps though approximate sparse inverse give an upper bound (see Chapter 3)

  - `‘Explicit’` : (NOT AVAILABLE) Everything is ready to set up, for example, a Runge-Kutta time integration of fluxes

- `‘Model’`

  - `‘Laminar-Incompressible’` : Basic Incompressible flow model
  - `‘RANS - K-EPS -Incompressible’` : Low-Reynolds K-Epsilon incompressible model with further model-related options
  - `‘Laminar-Slightly-Compressible’` : Laminar Combustion simulation with precomputed Flamelet libraries

- `‘rTfin’` : real number indicating the fraction of the extimated final simulation time `Tfin` (variable present in the `Solver.m` routine) at which to stop the simulation. `Tfin` is taken as: `Tfin=rTfin max([Lref^2 /ni0,Lref/Vref]);`

- `‘beta’` : a good value for this parameter is `0.02` this is an upper bound for the group `dt*th*ni0/h^2` (and so for the time step `dt`) due to a very down to earth error analysis of the sparse inverse approximation (see Chapter 3)

- `‘istatusupdate’` : must be an integer and indicates the number of time-steps intervals over which the derivative of the field’s kinetic energy is computed. This can be quite an expensive calculation especially with large number of nodes, a wise choice might be setting this value to between 100-1000 depending on the machine’s speed.

- `‘dEkdt_tol’` : the fraction of dEkdt0 (the initial total kinetic energy integral derivative) to stop the simulation at.

- `‘rdt’` : real number indicating the fraction of the minimum allowed `dt` chosen among 3 *candidates* that is to say `dtcand=[.5*h^2/ni0,h/Vref,` `beta*h^/(th*ni0)];` the first and the last value are suggested respectively by the monoticity criterion for explicit parabolic problems and (as previously stated) inverse sparse error control, the second one is simply the CFL condition; The final `dt` will be `dt=rdt*min(dtcand);`

- `'inormerrp'` : the number of time step intervals over which the norm of the pressure-segregated equation is computed.

- `'isumdiv'` : the number of time step intervals over which the sum of all the divergence equations is computed.

- `'igraf'` : the number of time step intervals over which the processed solution is being displayed with Matlab's output graphics.

- `'isave'` : the number of time step intervals over which the processed solution is being stored to the hard drive. If a simulation is started on the 31st of October 2008 at 17.53 (and 31 seconds!) the saving paths will be

  - on Windows: `C:\PostProcessing\31Oct2008T175331`
  - on MacOSX: `/Users/username/Documents/PostProcessing/31Oct2008T175331`
  - on Linux: `/home/username/Documents/PostProcessing/31Oct2008T175331`

- `'toldiv'` : tolerance on the overall sum (in sign) of the divergence equations.

- `'th'` : value of θ for the implicit time stepping scheme (see Chapter 3).

- `'Qvischeck'` : setting this value to `1` will enable a check on the velocity boundary conditions making it possible for the user to edit the `maskcnc` logical matrix and delete unwanted

- `'maskcnc'` : this is not an editable field, we are just inserting the logical matrix `maskcnc` in the `options` structure

- `'interpflux'` : sets the interpolation order of the Matlab interp3 routine being used in case of any kind of Convection scheme needing Lagrangian interpolation, this field can therefore be `'nearest'`,`'linear'` or `'cubic'` (`'spline'` wont work since the domain is immersed in NaNs)

- `'typcflux'` : possible options are [6]

  - `'NoConvFlux'` : No convective flux being computed, i.e. Stokesian flow

  - `'Quadratic'` : QUICK scheme with 1D splitting, (zero order accurate in time)

  - `'Lagrangian'` : (NON CONSERVATIVE) Lagrangian upwind interpolation

  - `'Predictor3DCorrector1D'` : Lagrangian upwind interpolation and evaluation of predicted and actual fluxes with QUICK scheme with 1D splitting (first order accuarate in time)

  - `'PredictorCorrectorTVD'` : Lagrangian upwind interpolation and evaluation of predicted and actual fluxes with QUICK scheme with 1D splitting (first order accuarate in time) with limiters functions specified by the field `'limitertype'`

    * `'superbee'`

    * `'vanleer'`

- `'MLCS'` : Multilevel Lagrangian Conservative Scheme

[6]all of these schemes are explained in detail in Chapter 4

Chapter 6

Numerical Results

In this chapter some of the most relevant numerical simulations that have been so far carried out with PRIN-3D, will be presented, ranging from Incompressible Flow simulations to Turbulent Low-Reynolds and Reacting flows, all with different geometrical layouts. Most of these simulations were meant as test cases for code validation and some other for debugging the code's full three-dimensional potentiality. All the post-processing analysis has been done with `Tecplot`$^{\text{TM}}$ by loading ascii data files containing processed variables that are dumped to a specific hard drive folder (see Chapter 5) during the simulation. All the data is arranged with tecplot finite-element data format. For every simulation there is a table showing basic numerical and geometrical data such as the three reference lengths for each direction Lx, Ly and Lz, the number of velocity nodes $nvel$, the number of pressure nodes or cells ncp (in some cases the number of pressure cells along the x, y and z direction are shown and are, respectively, $ncpx$, $ncpy$ and $ncpz$), the Reynolds and the cell Reynolds number respectively Re and Re_{cell}, the number of boundary faces $nfaces$ and other model-related parameters.

6.1 Fully 3D

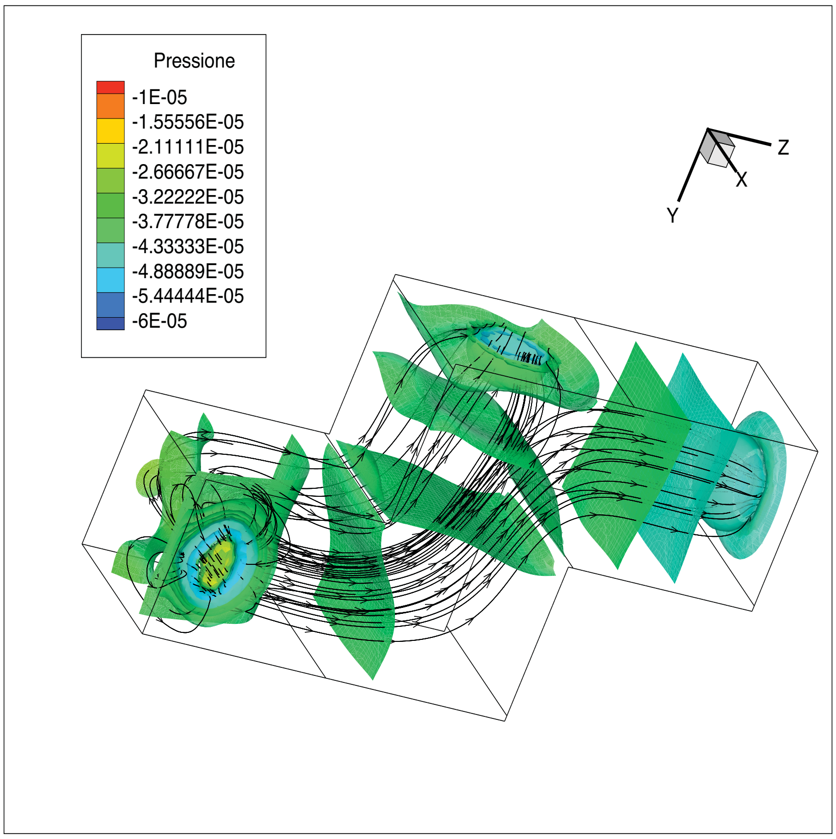


Figure 6.1: **Post-processing of a fully 3D Incompressible Navier Stokes simulation.**

This simulation is intended to test the code for any bugs concerning the velocity boundary conditions assignement and management. There are two Dirichlet outflows with circular plug flow velocity profiles (one with negative V component the other with W positive component) and one inflow where a negative U component has been assigned as showed in Figures 6.3 and 6.2. The code has automatically checked and reassigned such velocity boundary conditions in order to satisfy the divergence-free constraint and this was an

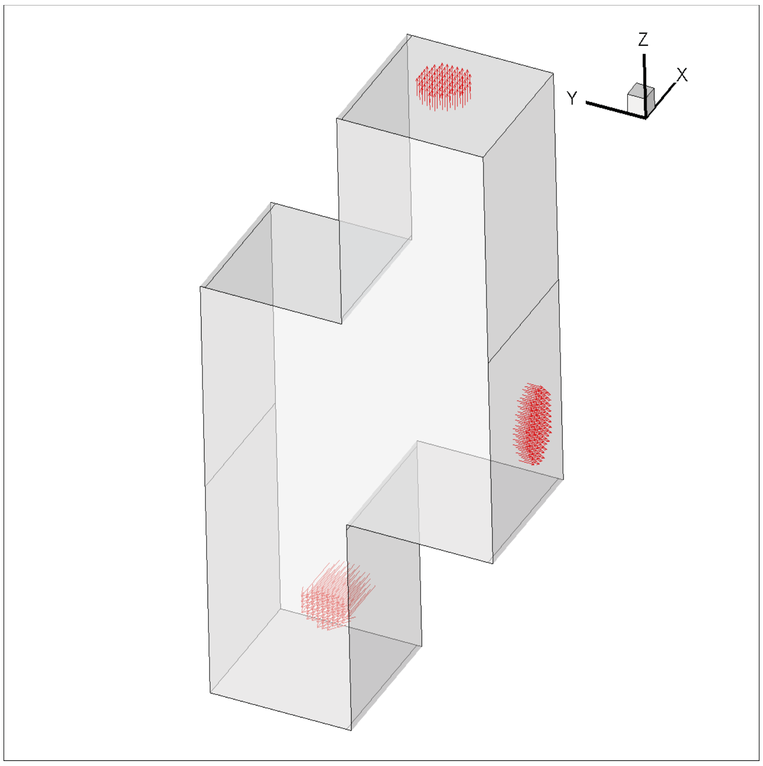


Figure 6.2: **Rappresentation of dirichlet velocity boundary conditions.**

important test for the right-hand side equation generators because the inflow and outflow conditions are totally 3D. Since this was a simple test with no physical relevance, a non conservative Lagrangian convective scheme (with 3D linear interpolation with low computational cost) has been adopted. This type of convective scheme has proven to be less numerically dissipative than a TVD high order flux reconstruction with dimensional splitting as it will be shown later on with some other Laminar Incompressible examples. Predictor-Corrector convective schemes with Lagrangian prediction are also available in the code and could be a better option for accuracy improvement. This simulation has been carried out with the second type of pressure variable ordering (see Chapter 3) because it has been tested that with the forth pressure variable ordering and block Gauss-Siedel solving, the pressure field evolves

$Lx = .5$	$nvel = 110862$
$Ly = 1$	$Re = 31.54$
$Lz = 2$	$Re_{cell} = 1.904$
$ncp = 119700$	$nfaces = 18$

Table 6.1: **General simulation data for Fully3D**

with plenty of oscillations causing the flow to oscillate as well several times while reaching for the steady state, which can cause numerical instabilities. It is possible to notice from Figure 6.1 that the pressure isosurfaces' values indicate that it decreases downstream (as it should be).

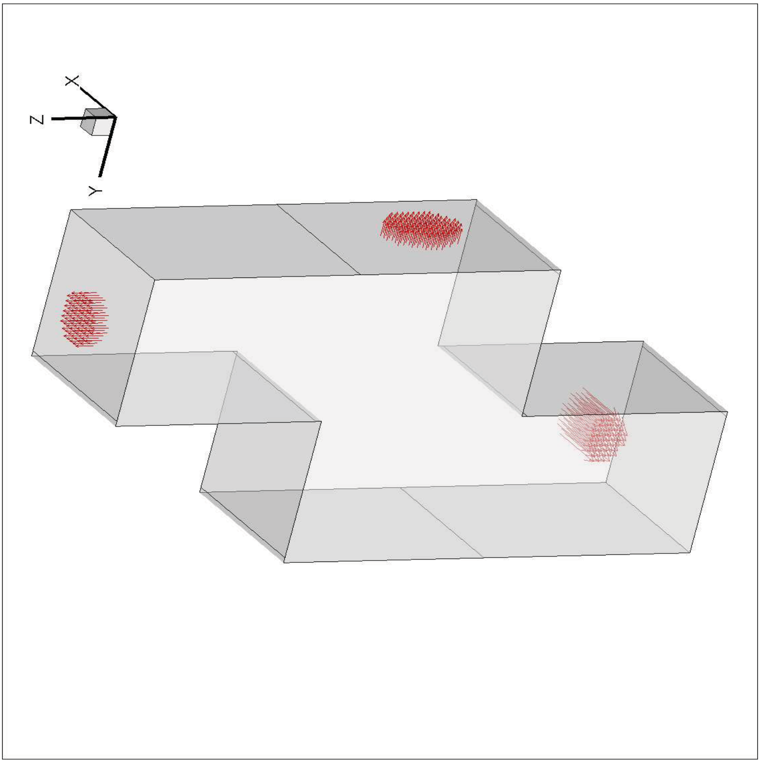
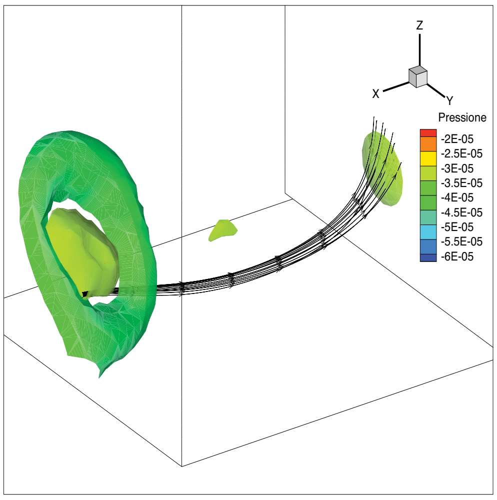


Figure 6.3: **Rappresentation of dirichlet velocity boundary conditions.**

It is interesting to zoom into the inflow area Figure 6.4 and notice that there is a stagnation point on the wall right in front of the inlet, and that right on the edge of the inflow orifice there are recirculating streamlines. (Figure 6.5)



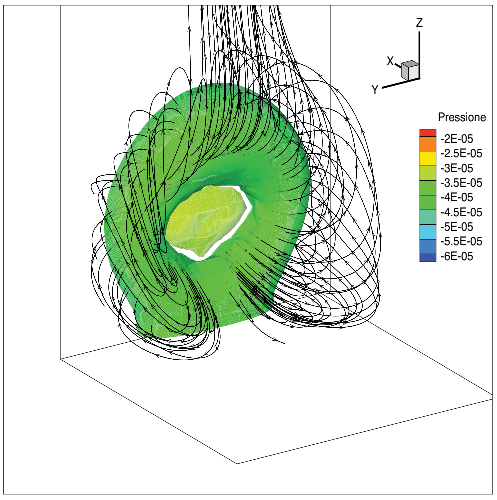


Figure 6.4: **Stagnation point zoom in.**

Figure 6.5: **Inflow recirculating streamlines.**

6.2 LID 2D

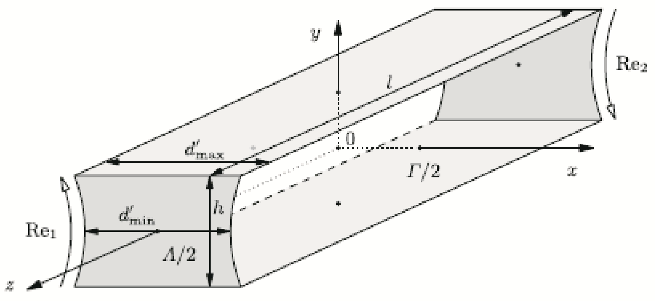


Figure 6.6: **Illustration of the experimental set up in (source [2]). The side-wall curvature is exaggerated**

Despite the fact that the code is meant to run 3D simulations, by setting the `‘Periodic’` field in the options struct to `‘x-x’` and by giving bi-dimensional velocity boundary conditions (all boundary velocity vectors lying in the Z-Y plane) a 2D simulation can be set up. A smart choice would be to tune the geometry parameters in order to reduce as much as possible the spanwise nodes so that the total processing time can be more efficient and the preprocessing time (LU factorization with UMFPACK) and memory usage (which has been found to be the real bottleneck in many simulations, especially for bulk 3D domains) will be reduced. On the same geometry, two different convection schemes available in the code have been tested, which are Lagrangian (non conservative) and a predictor-corrector conservative scheme with TVD flux reconstruction by means of spatial splitting. Moreover, for both of these schemes, two different configurations have been used:

1 Figure 6.8 : double lid driven cavity with moving faces at Y=0 and Y=0.04 with $Re_p = 100$

2 Figure 6.12 : single lid driven face at Y=0 and Y=0.04 with $Re = 700$

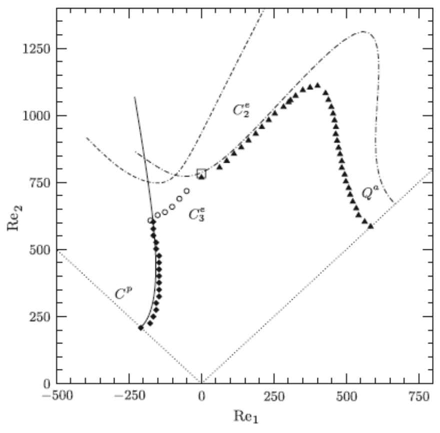


Figure 6.7: **Stability domain in the Re_1 vs Re_2 plane. Experimental critical Reynolds number Re_2 as function of Re_1 (*filled symbols*) in comparison with numerical neutral curves for different modes : C^p cooperative instability - *filled diamond, continous line*, C^e_2 and Q^a respectively centrifugal instability and quadripolar instability - *filled triangles, dashed-dotted line*, C^e_3 open circul *double dash, dotted line* (source [2])**

The data for these simulation has been taken from [2]. In Figure (6.7) it is clearly shown that in both cases no instabilities should rise, assuring that the basic 2D flow will not breakdown into a fully 3D phenomena. From Figures 6.8 and 6.12 it can be seen that the spanwise dimension is very short (in this case only 3 spanwise cells have been used)

$Lx = .02\ Ly$	$Ly = .04$	$Lz = .04$
$numiter_i = 2$	$numiter_k = 1$	$numiter_l = 1$
`‘Periodic’=‘x-x’`	pres. var. arrangement 2	$ncp = 111747$
$ncpx = 3$	$ncpx(np) = 4$	$ncpz = 193$
$ncp(np) = 148996$	$ncpy = 193$	$nvel = 110592$

Table 6.2: **General Simulation Data for LID 2D**

6.2.1 Double LID Driven Cavity $Re_p = 100$

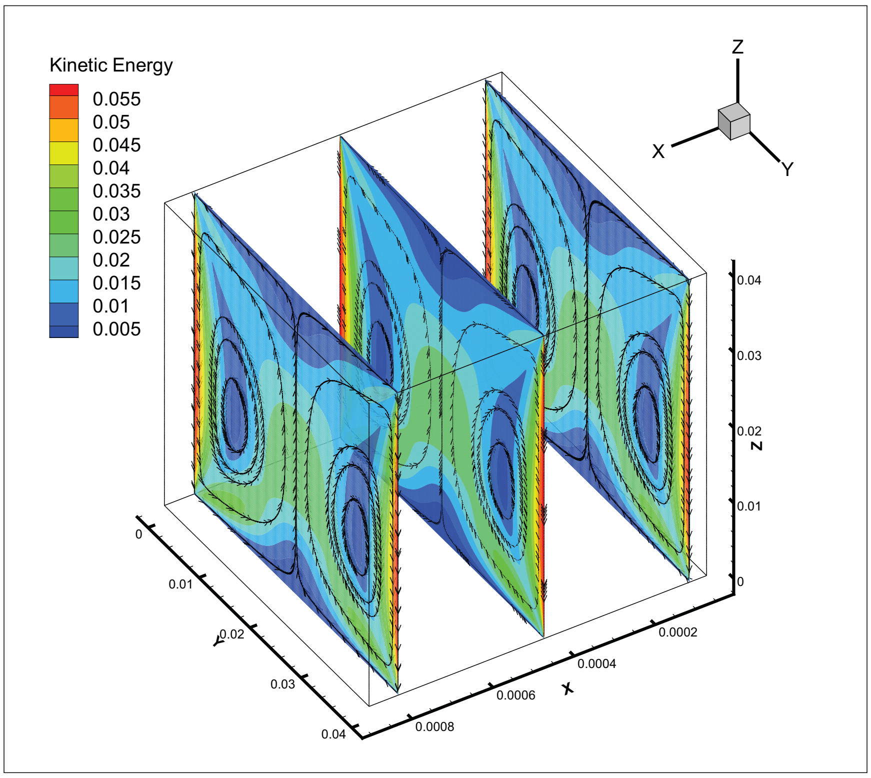


Figure 6.8: **X-plane slices of a double lid driven cavity with $Re_p = 100$.**

In this post-processing three slices along the periodic direction have been created and pulled apart in order to show that the data is truly bi-dimensional. As shown in Table 6.3 the TVD SuperBee scheme inserts a greater amount of artificial viscosity into the numerical results, whereas the Lagrangian convection scheme, though not conservative, proves to be less dissipative. This is due essentially to the fact that the latter is a genuinely 3D upwind interpolation whereas the TVD limiter function blends a high order flux (QUICK) with an upwind low order (UDS) which is very diffusive. This numerical effect is magnified in multidimensional problems if the flow is oblique to the grid; the truncation error then produces diffusion in the direction normal to the flow as well as in the streamwise direction, a particularly serious type of error. Peaks or rapid variation in the variables will be smeared out and, since the rate of error reduction is only first order, very fine grids are required to obtain accurate solutions. [5]

In Figures 6.9, 6.10 and 6.11 a good agreement between PRIN-3D numerical results 6.11 and the experimental and numerical results presented in [2], respectively 6.10 and 6.9, is shown.

TVD SuperBee	Lagrangian
1.7123e-008	1.842e-008

Table 6.3: **Double LID Driven Cavity - Comparison of numerical integration of Kinetic Energy for different convection schemes**

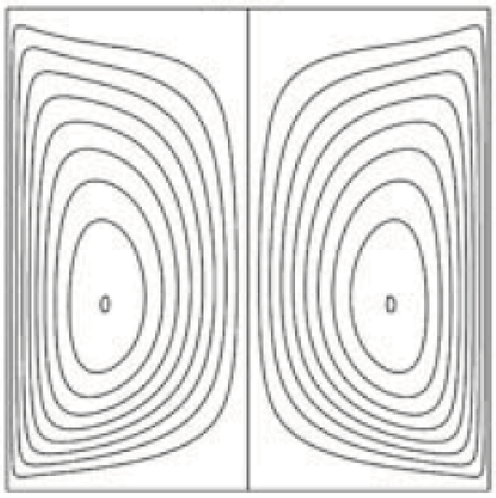


Figure 6.9: **Numerical results in (source [2])**

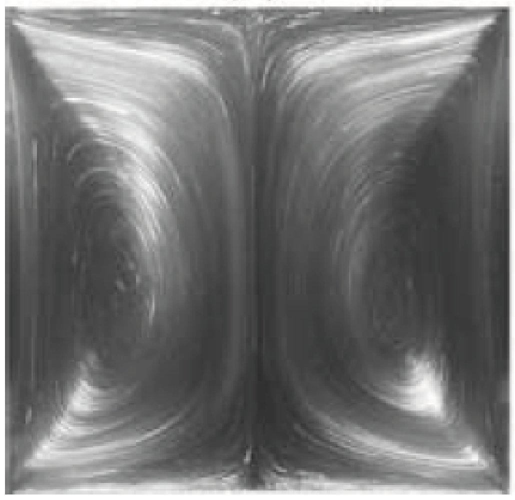


Figure 6.10: **Experimental results in (source [2])**

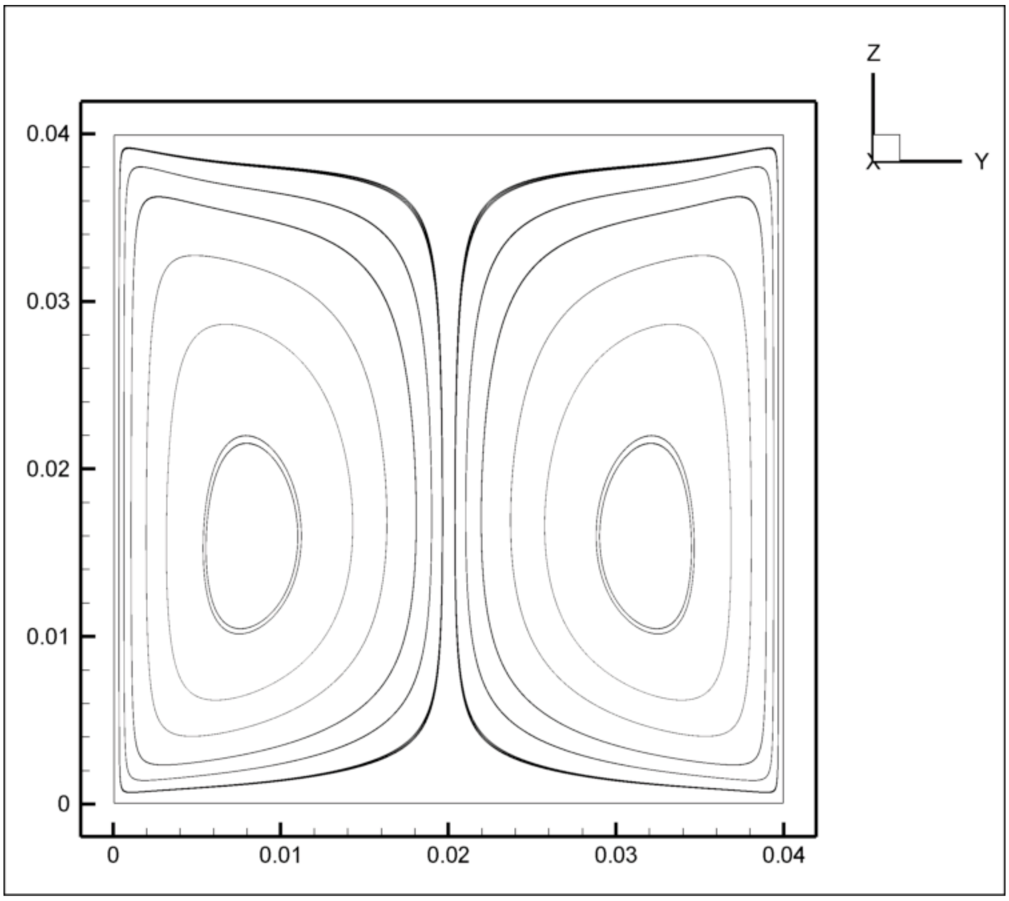


Figure 6.11: **PRIN-3D numerical results**

6.2.2 Lid Driven 2D Cavity $Re = 700$

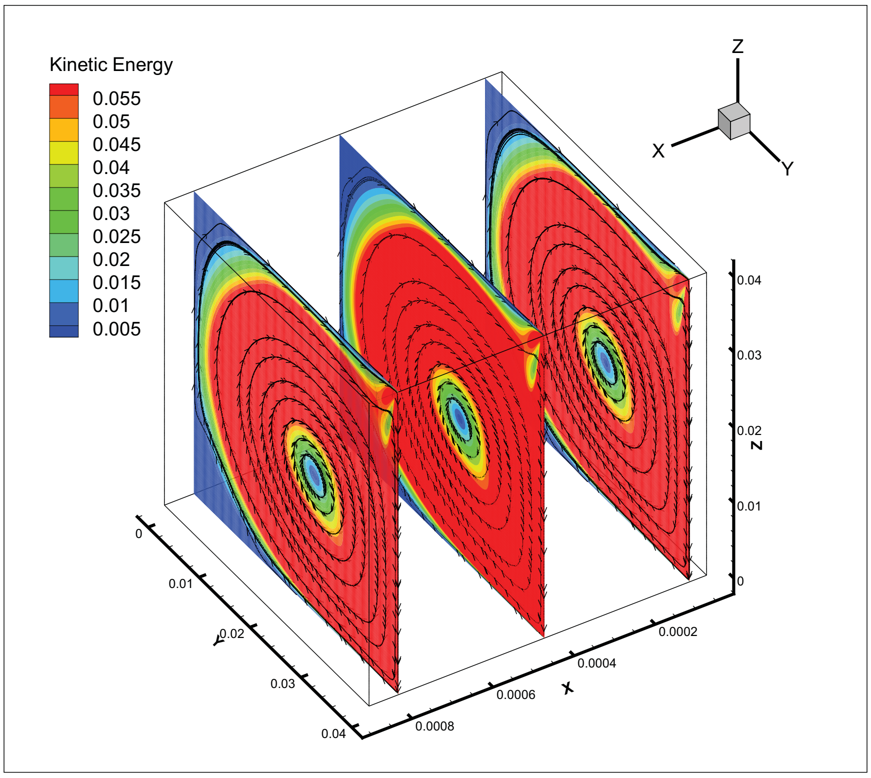


Figure 6.12: **3 X-plane slices of a single lid driven cavity with $Re = 700$.**

The second configuration that has been tested is the classic LID Driven Cavity at a Reynolds number that guarantees that no instabilities should rise.

TVD SuperBee	Lagrangian
4.025e-007	5.081e-007

Table 6.4: **LID Driven Cavity - Comparison of numerical integration of Kinetic Energy for different convection schemes**

It can be seen that in the first case (double LID driven cavity at $Re = 100$)

the amount of energy being dissipated by the TVD scheme is approximately 7% of the total kinetic energy of the Lagrangian case. On the other hand, with the single LID driven cavity at Reynolds 700, the dissipated energy is 20%. This is probably due to the fact that with higher velocity gradients (i.e. with a higher Reynolds number) the limiter function overdamps the velocity field switching too often to the low order UDS flux reconstruction.

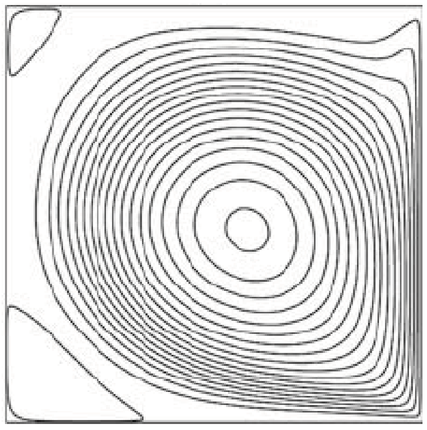


Figure 6.13: **Numerical results in (source [2])**

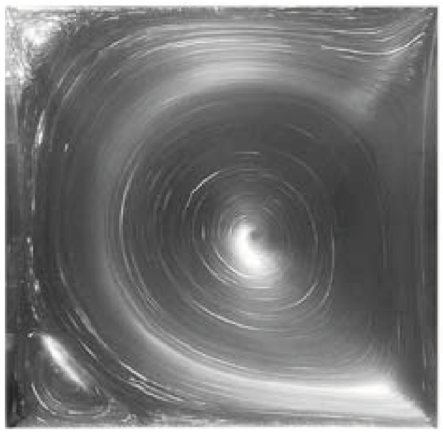


Figure 6.14: **Experimental results in (source [2])**

This simulation is actually 3D but, just like in double LID case, boundary velocity vectors are given exactly in the Y-Z plane with periodic boundary conditions along the X-direction. This guarantees that the velocity field is truly bi-dimensional as it can be gathered by Figure 6.12

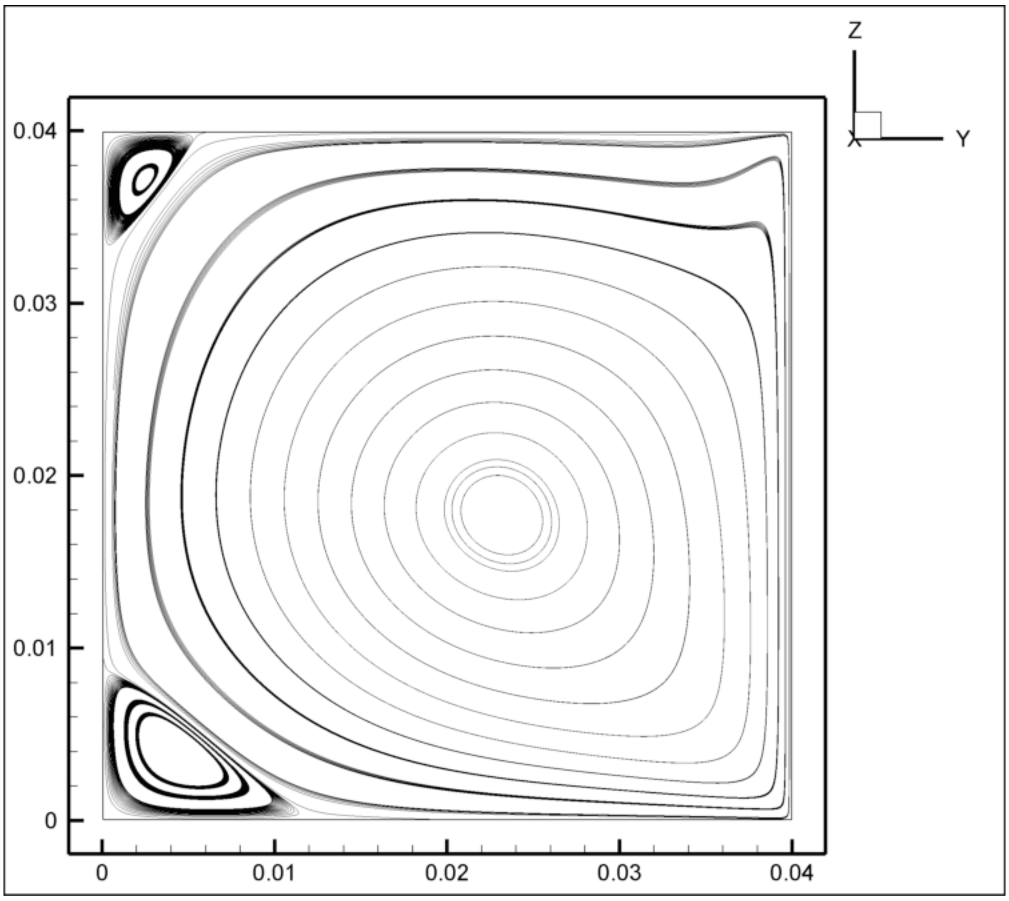


Figure 6.15: **PRIN-3D numerical results**

6.2.3 Double Lid Driven 3D Cavity

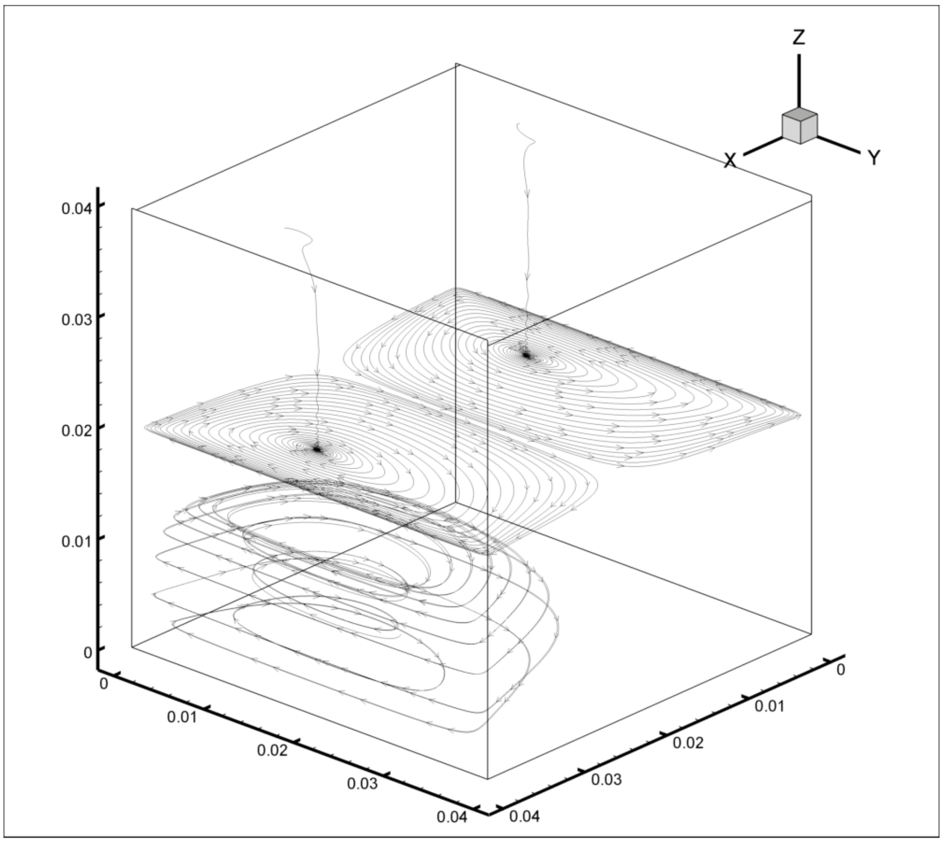


Figure 6.16: **Double lid driven 3D cavity.**

A straightforward extension to the 3D case of the previously described double LID driven 2D simulation is the one presented here. The two parallel moving lids are faces $X = 0$ and $X = 0.04$ whereas all of the other faces are solid walls. By taking a central slice (i.e. at $Z = 0.02$), shown in Figure 6.16, it is possible to notice that the velocity field is pretty much bi-dimensional, even though not identical to the analog 2D case. As a matter of fact, it can be seen that the vortex core is actually drawing mass from the neighbor zones into the central plane giving birth to 4 double nested helix structures one descending and the other one ascending (symmetrically located with respect

$Lx = .04$	$Ly = .04$	$Lz = .04$
$numiter_i = 1$	$numiter_k = 1$	$numiter_l = 1$
`'Periodic'='no'`	pres. var. arrangement 2	$ncp = 456533$
$ncpx = 77$	$ncpy = 77$	$ncpz = 77$
$nvel = 438976$	$Re_c = 100$	$Re_c = 1.284$

Table 6.5: **General Simulation Data for LID Driven Cavity 3D**

to the $Z = 0.02$ plane and the $X = 0.02$ plane) of which only one is shown in Figure 6.17 for the sake of clarity.

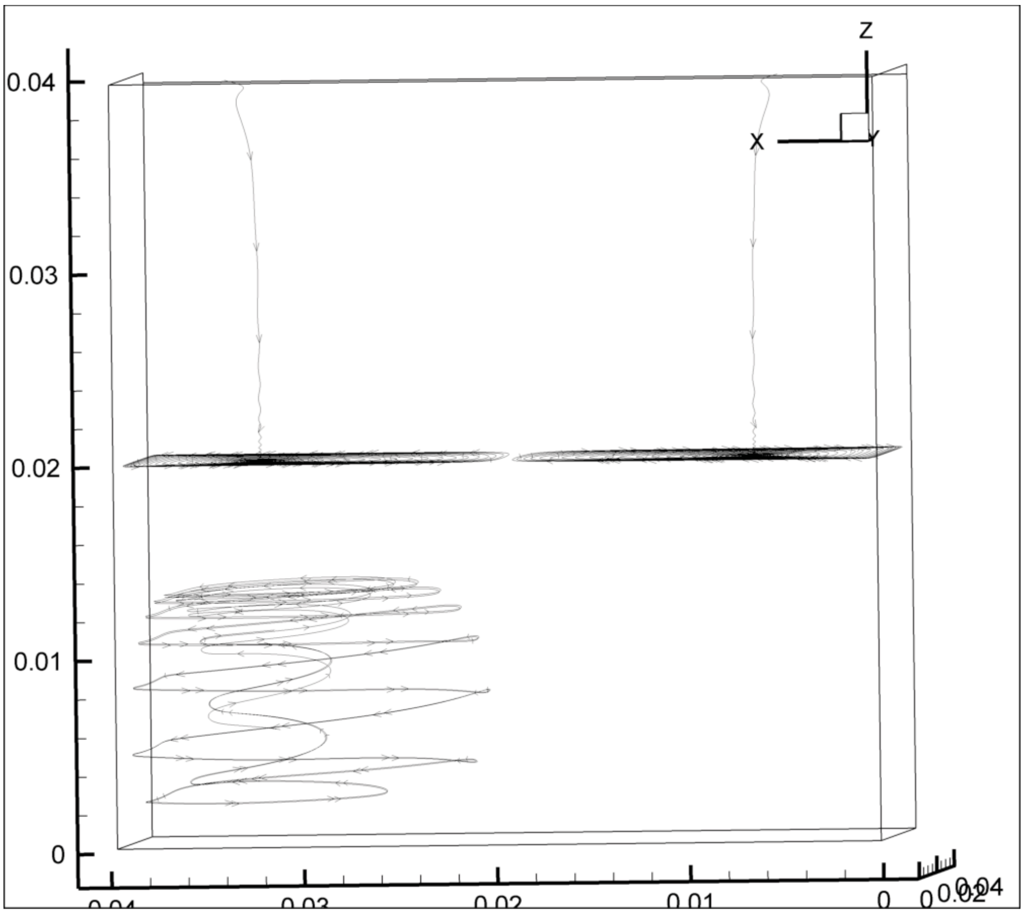


Figure 6.17: **Double lid driven cavity 3D close-up.**

6.3 Combustion Simulation

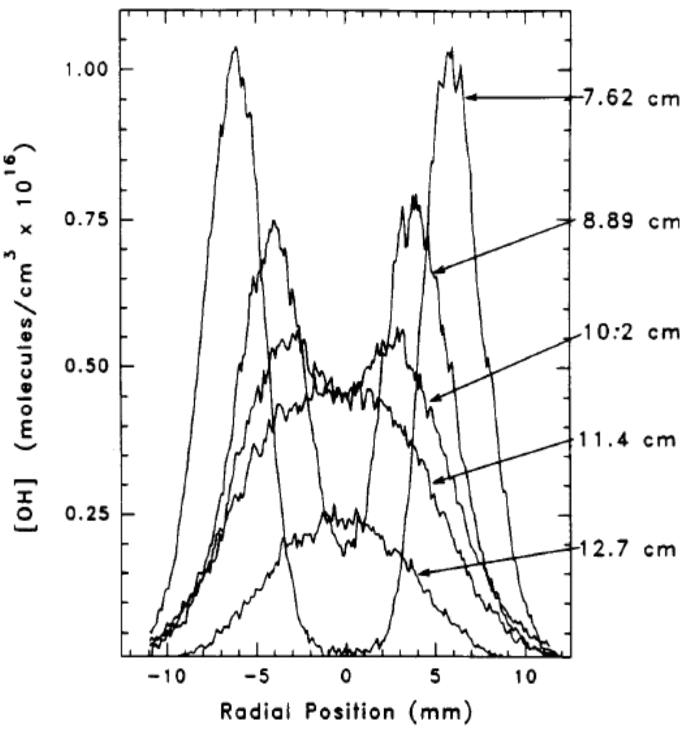


Figure 6.18: **Santoro flame OH concentration profile** $rdt = 0.6$ $rlxFL = 0.8$**. (source [18]**)

The main objective now is to validate the flamelet combustion numerical implementation starting with experimental data retrieved from [18]. The aim of this article is the investigation of chemical and thermofluiddynamic properties of an axial-symmetric diffusion flame with several different types of oxidizer and fuel. The focus will be on methane-air laminar diffusion flames for which flamelet libraries will have been preprocessed. The experimental setup consists in a coannular burner made up of a 1.1-cm-diameter fuel tube and a concentric 10.2-cm-diameter air annulus which is used to establish the laminar diffusion flames. The air and fuel flow rates are respectively set to 1300 cm^3/s and 9.8 cm^3/s. Considering the geometrical layout of the apparatus, this yields a fuel and air mean velocities of 0.103 m/s and 0.161 m/s. The very first attempts in simulating such flame have been carried out with a cartesian 2D geometry. The flames evolve in the Y-Z plane while spanwise periodic boundary conditions (along the X-direction) are imposed. In Figure 6.19 it is clearly shown that the simulation is truly bi-dimensional and this can be argued by noticing that the Mixture Fraction's isosurfaces are identically repeated along the X-direction.

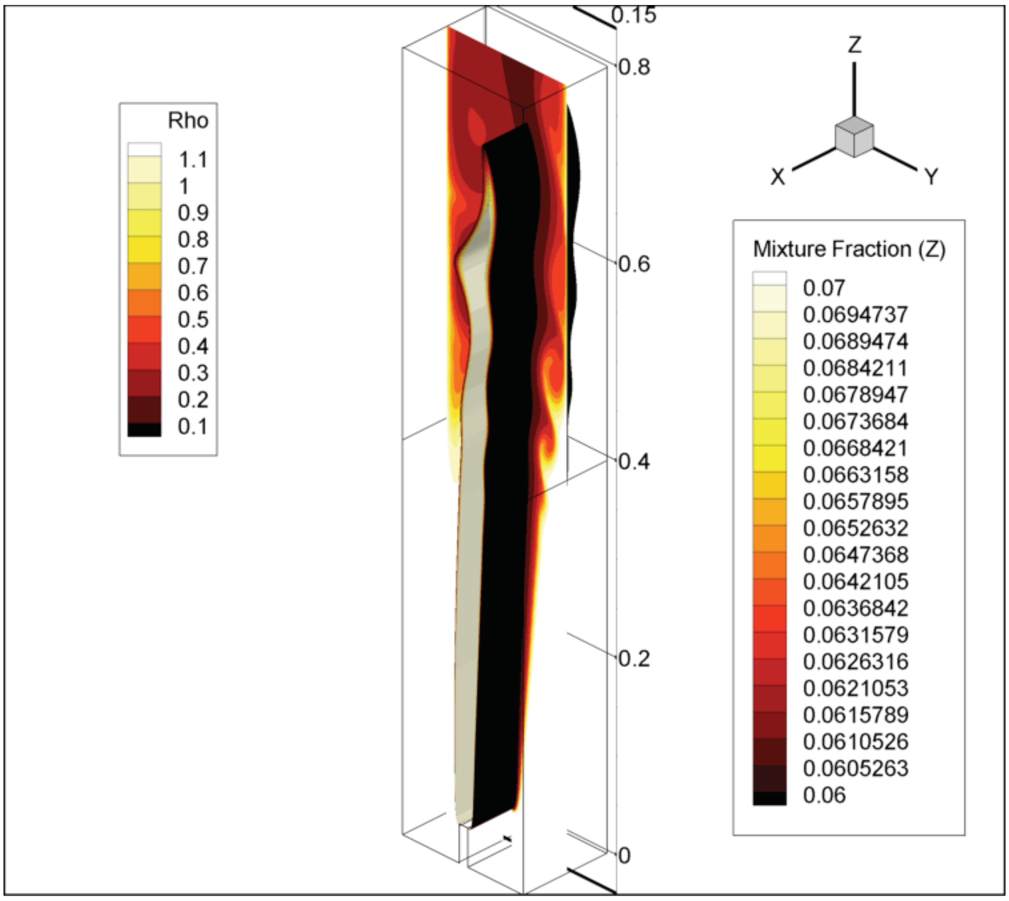


Figure 6.19: **Contour of density (mid X-plane) and isosurfaces of Mixture Fraction**

6.3.1 2D Methane-air flame

The simulation data in Table 6.6 is common to all of the numerical results presented in this section. The oxidizer's mean velocity is kept at 0.161m/s for all cases whereas three different fuel velocities are being tested: Flame 1) 0.103 m/s, Flame 2) 0.5 m/s, Flame 3) 0.015 m/s. This is done for the sake of testing the code's basic respondance to inlet b.c. variations. Plug flow velocity Dirichlet boundary conditions are given for the lateral air co-flow whereas Poiseuille Dirichlet boundary conditions are assigned for the central methane injector. On the outflow face at Z=0.8 extrapolation b.c. are assigned for all the transported species. On lateral walls homogenous Dirichlet b.c. are issued for the Mixture Fraction and this has proven to work correctly even though, for the flamelet model's approximations, all solid wall boundaries should be adiabatic walls and, therefore, homogenous Neumann b.c. should be imposed. Our choice can be explained by taking into account that the time stepping procedure used to update the Mixture Fraction transport is explicit and by the fact that the flame never reaches the lateral walls apart from zones near the outflow section which are not relevant due to the typical presence of an error region associated to ouflow b.c. . The time evolution of the whole simulation (which only makes sense if the solver is set to provide a good time resolution, i.e. *large* values for $numiter_i$, $numiter_k$ and $numiter_l$) is very sensitive to the mixture fraction underrelaxing parameter but, on the other hand, the steady state configuration is not.

$Lx = .01$	$Ly = .15$	$Lz = .8$	`'Periodic'='x-x'`	pres.arrang. 2
$Re_{OX} = 112.44$	$Re_{OXcell} = 1.284$	$numiter_i = 1$	$numiter_k = 1$	$numiter_l = 1$
$ncp = 200715$	$ncpx = 5$	$ncpy = 87$	$ncpz = 463$	$nvel = 438976$

Table 6.6: **Simulation general data**

SANTORO 2D Flame 1

In this case the fuel rate is set at full speed (i.e. the true value in [18]) and the flame surface's position (i.e. the isosurface of the maximum OH concentration or maximum Temperature value contour) is not stationary. Its position should be fixed in space and as it can be seen in Figure 6.18 at 7.62 cm the flame should be approximately 1 cm wide.

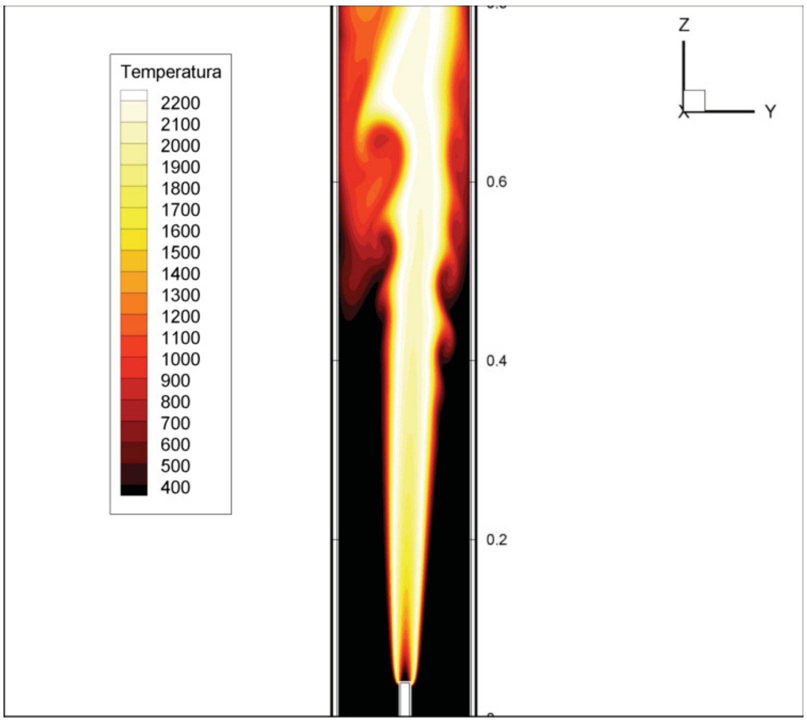


Figure 6.20: **Santoro Flame 1 temperature profile**

`‘rdt’=0.6`	`‘rlxFL’=0.8`	$V_{fuel} = 0.103\ m/s$	$Re_{FUEL} = 65.03$	$Re_{FUELcell} = 9.976$

Table 6.7: **Flame 1 simulation data**

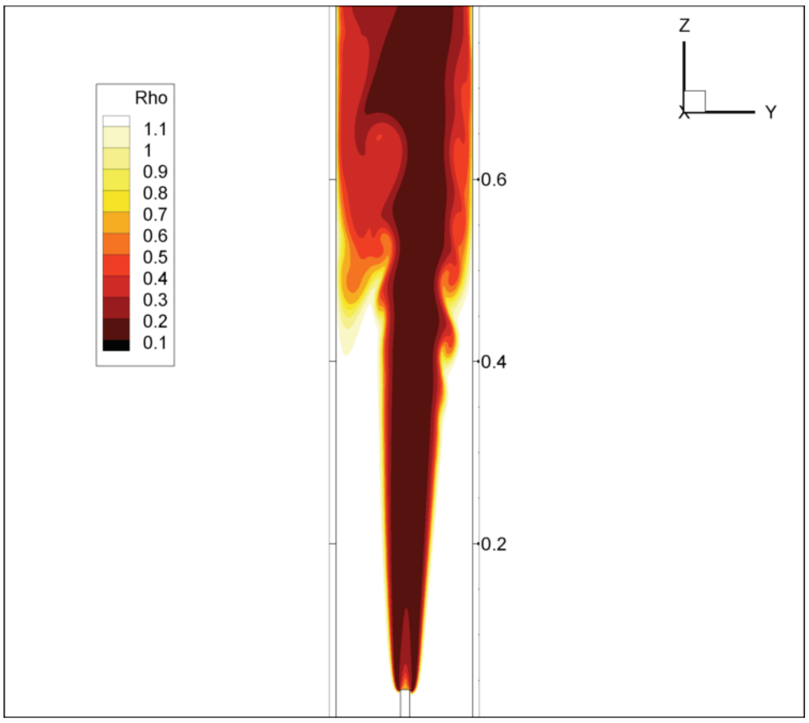


Figure 6.21: **Santoro Flame 1 density profile**

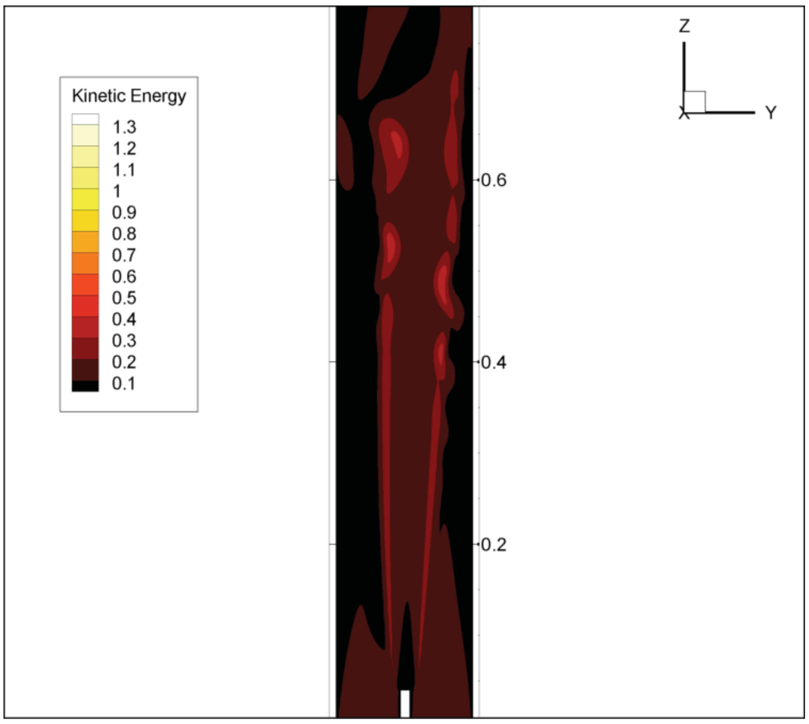


Figure 6.22: **Santoro Flame 1 Kinetic Energy profile**

`‘rdt’=0.6`	`‘rlxFL’=0.4`	$V_{fuel} = 0.5\ m/s$	$Re_{FUEL} = 31.52$	$Re_{FUELcell} = 4.84$

Table 6.8: **Flame 2 simulation data**

SANTORO 2D Flame 2

Since in Flame 1 the simulation yielded a non-physical unsteady behavior, the simulation presented here wanted to test if by lowering the fuel Reynolds number a true steady state but with a shorter flame would have been obtained. This has been the case, in fact as shown in Figure 6.23 the flame surface assumes a typical shape and it reaches approximately 4.5 cm (less than 7.62 as expected). Nonetheless, the zone ranging from 5 cm to 8 cm still shows unsteady behavior probably due to the extrapolating outflow boundary conditions.

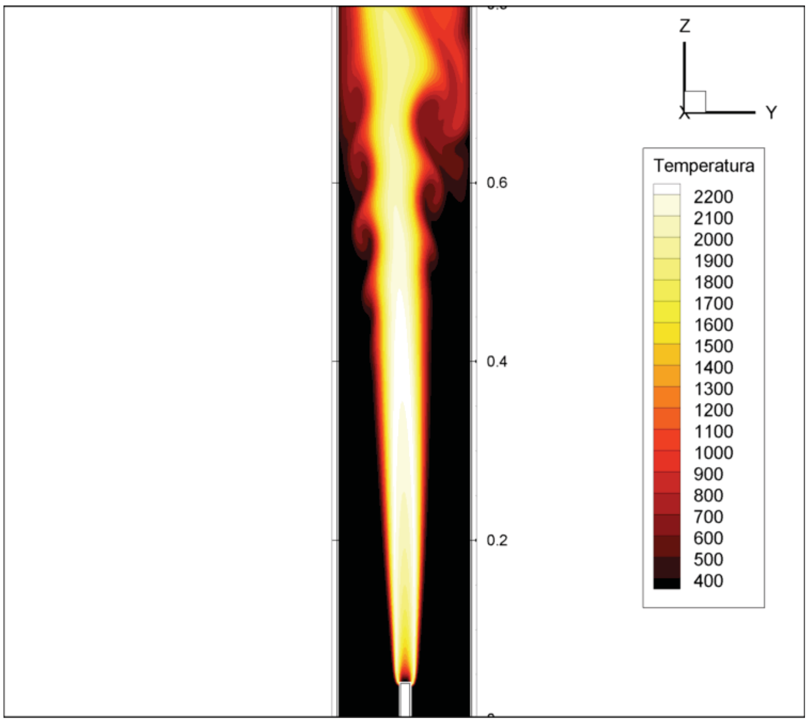


Figure 6.23: **Santoro Flame 2 temperature profile**

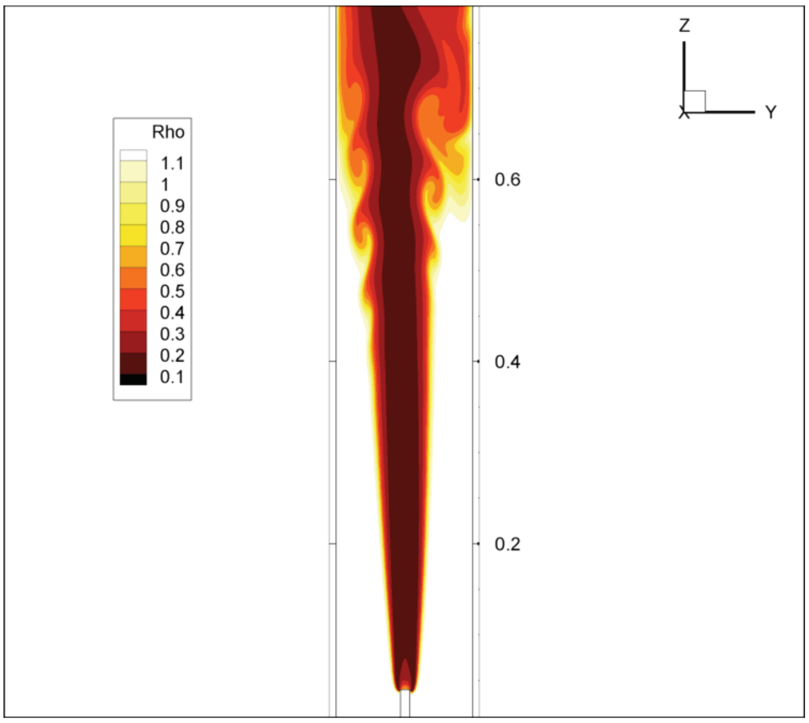


Figure 6.24: **Santoro Flame 2 density profile**

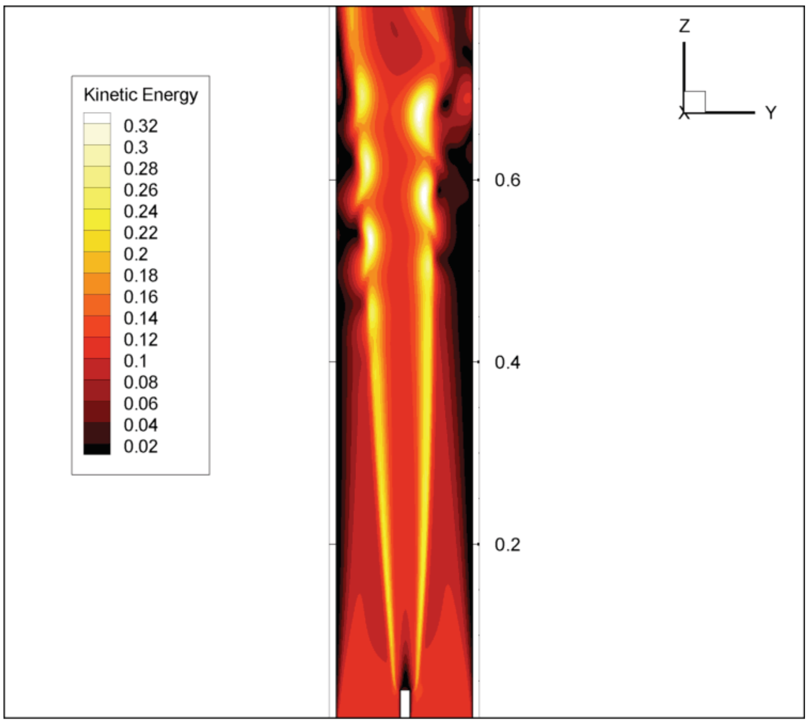


Figure 6.25: **Santoro Flame 2 Kinetic Energy profile**

SANTORO 2D Flame 3

Same as Flame 2 but with fuel flow rate reduced to 10 % of the original value (Flame 1). Extrapolating b.c. are less effective on the flame's configuration which is steady and rather short (approx. 2 cm).

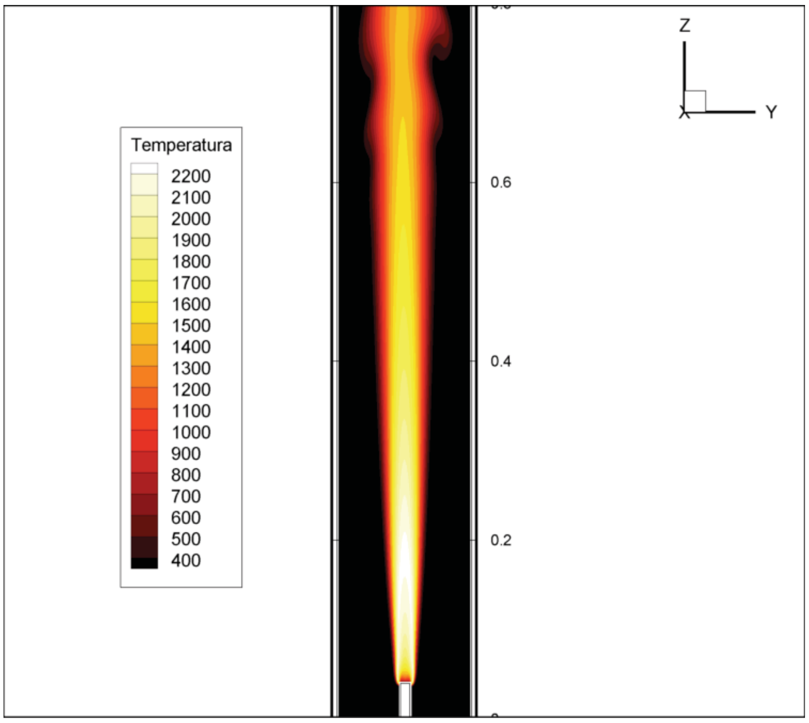


Figure 6.26: **Santoro Flame 3 temperature profile**

`'rdt'=0.6`	`'rlxFL'=0.4`	$V_{fuel} = 0.5\ m/s$	$Re_{FUEL} = 31.52$	$Re_{FUELcell} = 4.84$

Table 6.9: **Flame 3 simulation data**

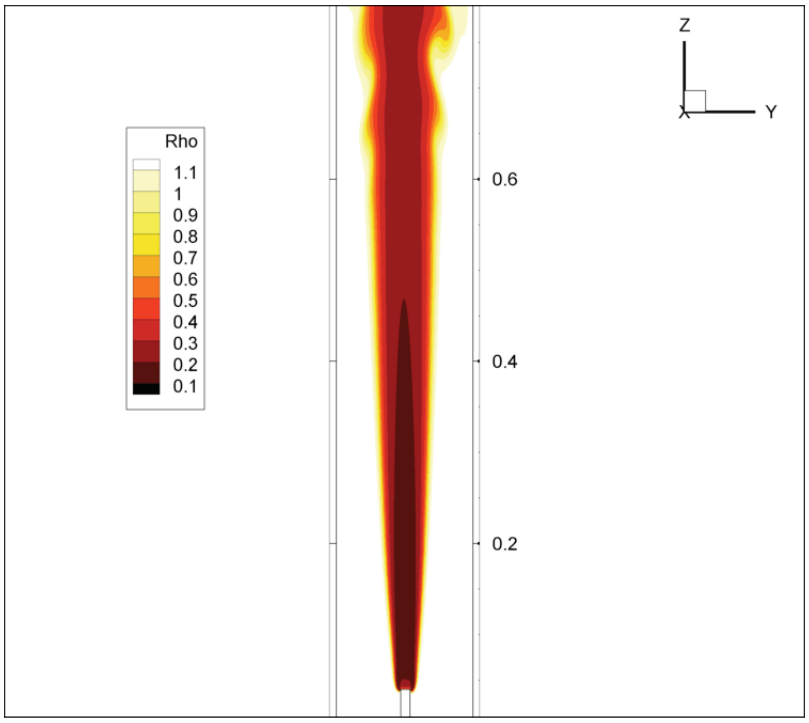


Figure 6.27: **Santoro Flame 3 density profile**

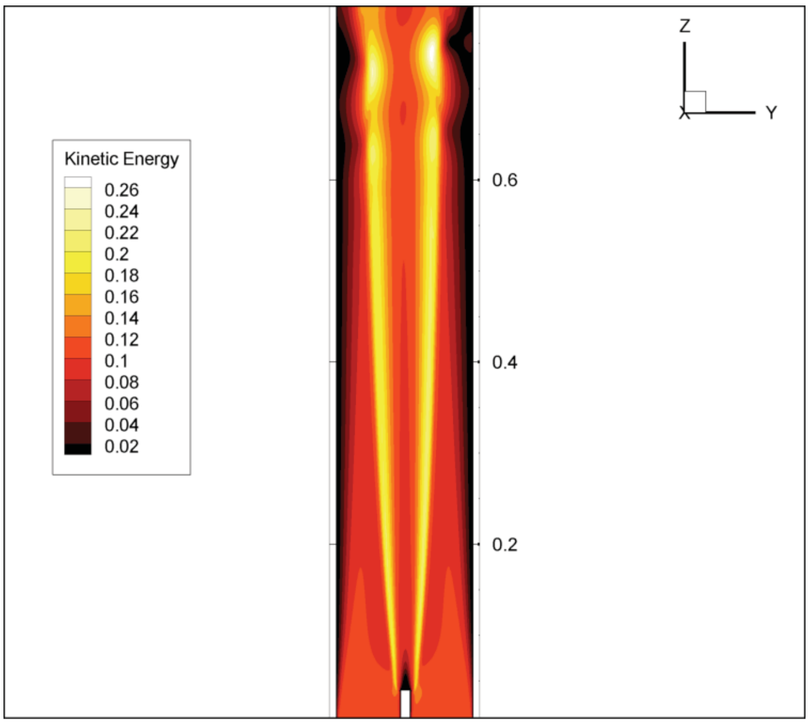


Figure 6.28: **Santoro Flame 3 Kinetic Energy profile**

6.3.2 SANTORO 3D Flame

A truly 3D combustion simulation reproducing the 3D boundary conditions of the experiment in [18], is here going to be set up . The computational domain is now a 3D box with a single inflow and outflow face with the `‘Periodic’` option field set to `‘no’`. Axialsymmetric boundary conditions (as far as the staircase approximation’s limited reconstruction capability is concerned) are reproduced by using combinations of circular hat functions. On the same inflow face, starting from its center, radial functions for the mixture fraction, normal velocity component, density and dynamic viscosity are assigned by the user, in order to reproduce the geometry of the coannular burner and the concentric air annulus inflows.

In Table 6.10 all PRIN-3D numerical parameters, common to 3D flame simulations, are shown.

`‘rdt’=0.1`	`‘rlxFL’=0.1`	$nvel = 392040$	$ncp = 200715$	$ncpx = 67$
$ncpy = 67$	$ncpz = 91$	$Lx = .11$	$Ly = .11$	$Lz = .15$

Table 6.10: **3D Flame simulation data**

Plug inflow conditions

$Re_{FUEL} = 63.59$	$Re_{FUELcell} = 9.56$	$Re_{OX} = 109.94$	$Re_{OXcell} = 16.53$

Table 6.11: **3D Flame simulation data - Plug Flow**

As a first simulation attempt, for both fuel and oxidizer’s velocity inflow conditions, plug profiles have been adopted. Moreover, because of the high number of pressure cells and of the bulk geometry that leads to a quite filled

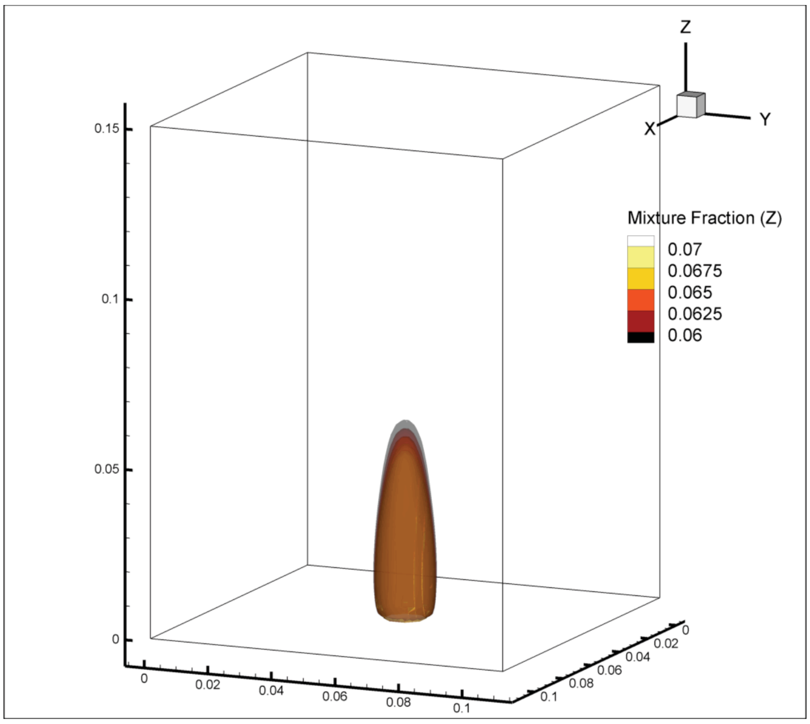


Figure 6.29: **Santoro 3D flame Z profile.**

pseudo-elliptic matrix pattern, it has been tried to launch the simulation with a 4 pressure variables arrangement (see section 3.3).

Because of these choices, as can it be seen in Figure 6.29, the flame, in its maximum extension, is shorter than expected and in addition it collapsed due to the known stability issues associated with the four pressure variable arrangement.

As it can be observed from Figures 6.30 and 6.31, the effect of combustion on the flow is to generate mass flux source cells. This due to the fact that in the reacting cells temperature rises causing density to drop and particles to be locally pushed away from the reacting front. In particular, from Figure 6.31 it is clear how such positive-divergence front , where non stoichiometric reactions are taking place, stays ahead of the actual flame surface.

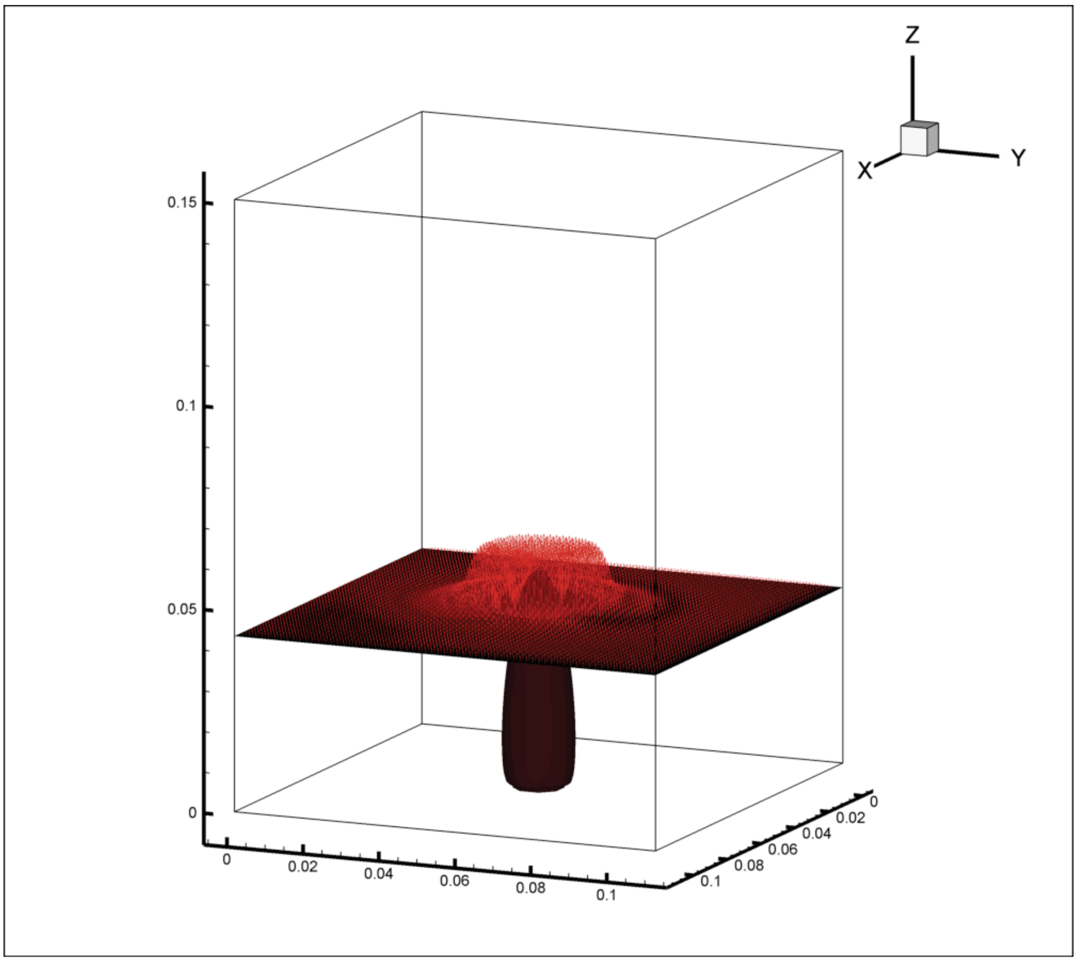


Figure 6.30: **Santoro 3D flame. Mass flux source cells.**

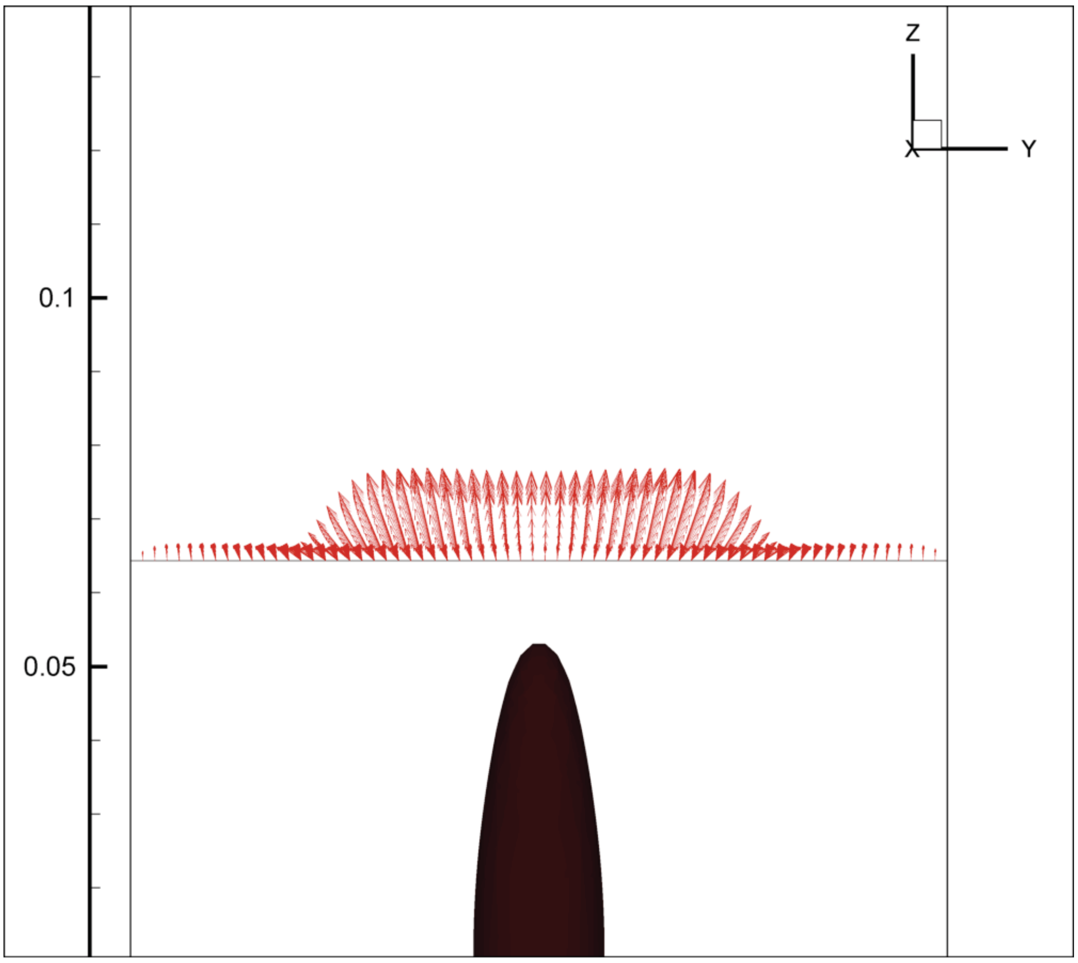


Figure 6.31: **Santoro 3D flame. Chemical reaction front.**

Poiseuille inflow conditions

After the first simulation and observing that in [18] is specified that fuel and oxidizer supply ducts are long enough to obtain fully developed flows, it has been decided to impose for both fuel and oxidizer inlet velocities, Poiseuille conditions Figure 6.32.

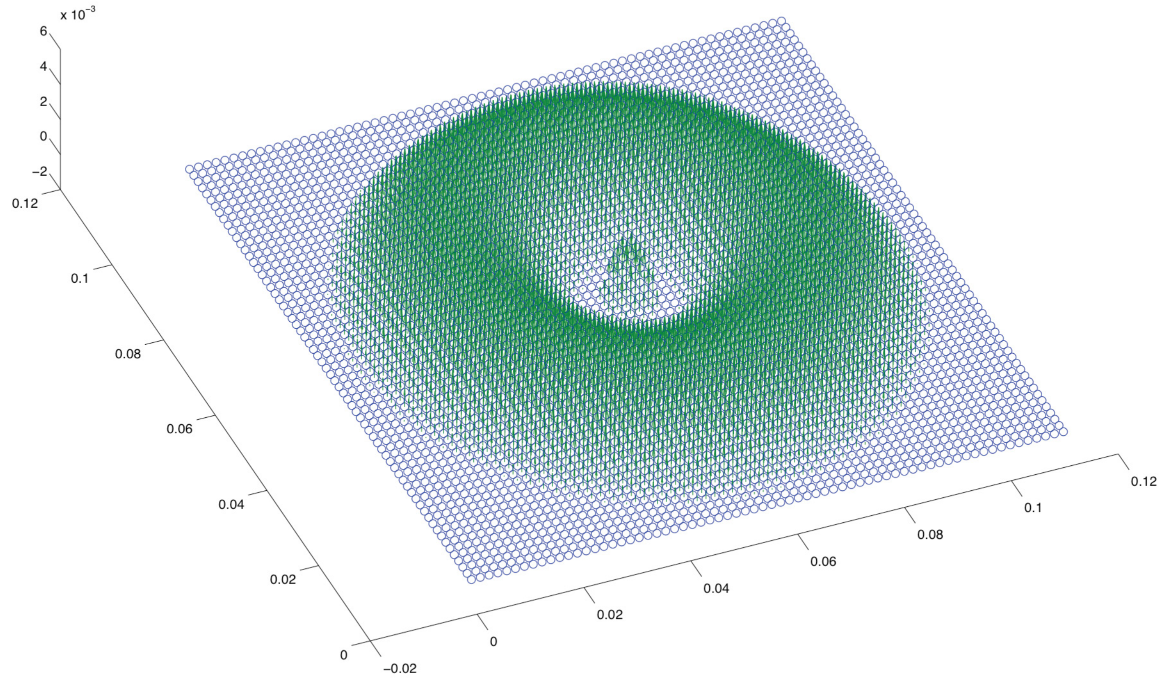


Figure 6.32: **Santoro 3D flame. Poiseuille inflow conditions.**

This means that radial hat functions representing this inlet conditions are:

$$w_{FUEL} = 1 - \frac{r^2}{R_1^2}$$

for the fuel, and

$$w_{AIR} = R_2^2 - r^2 + \left(R_2^2 - R_1^2\right) \frac{\ln\left(R_2/R_1\right)}{\ln\left(R_1/R_2\right)}$$

for the oxidizer, where R_1 and R_2 are respectively the inner and outer radius of the air annulus.

$Re_{FUEL} = 127.18$	$Re_{FUELcell} = 19.12$	$Re_{OX} = 164.91$	$Re_{OXcell} = 24.80$

Table 6.12: **3D Flame simulation data**

Considering that maintaining the same flow rate, such profiles have higher maximum velocities than the plug ones, a longer flame is obtained.

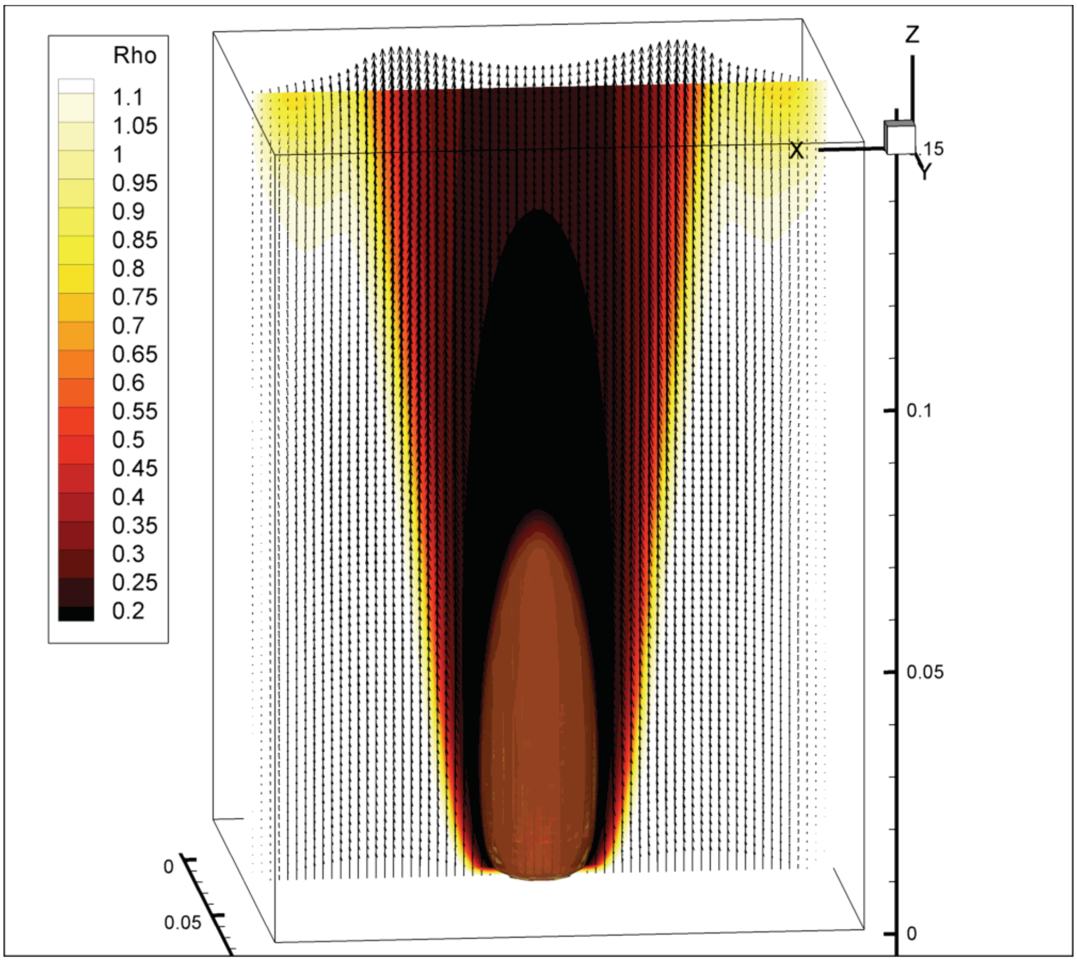


Figure 6.33: **Santoro 3D flame. Density profile and velocity vectors slice.**

Cutting the flame with a $Z = 7.62$ plane (Figure 6.35), a temperature profile in this plane (Figure 6.36) is obtained.

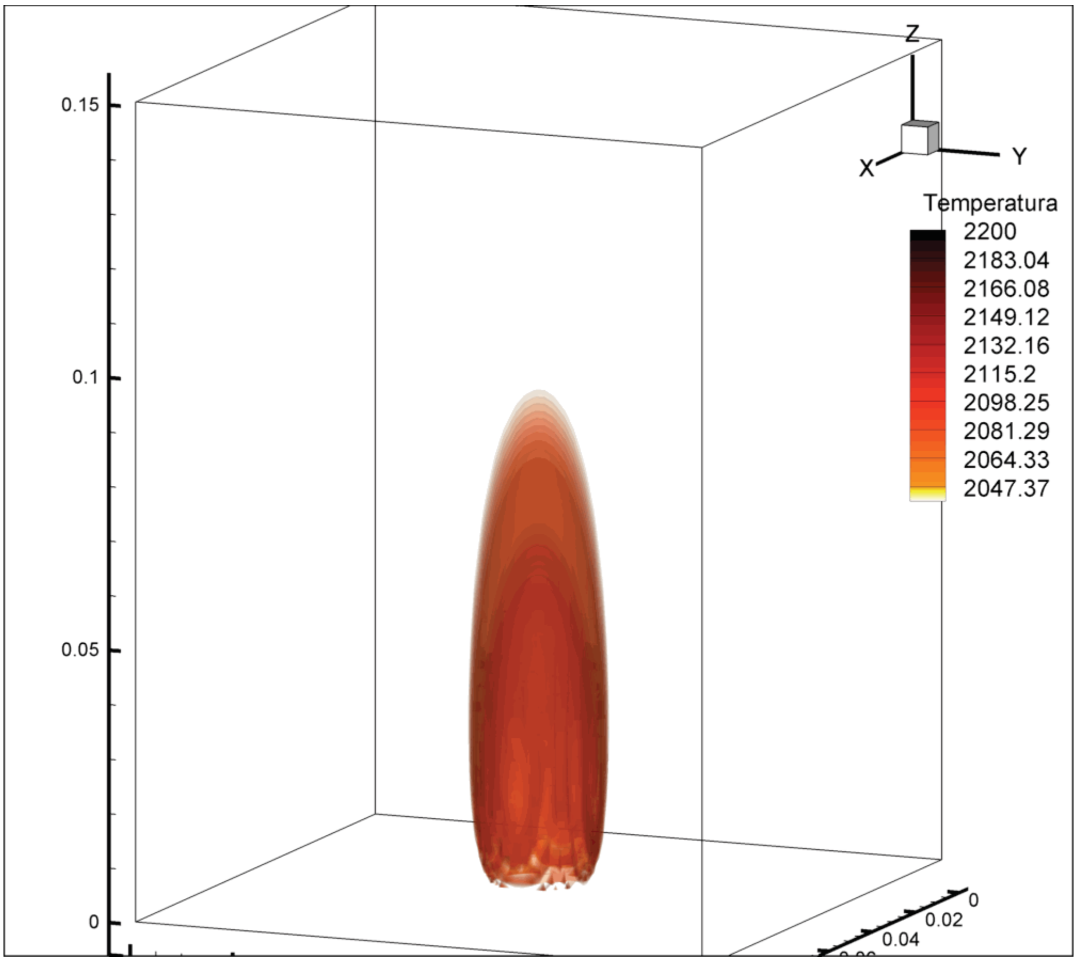


Figure 6.34: **Santoro 3D flame. Temperature isosurfaces.**

The maximum temperature profile (flame surface intersection with plane $Z = 0.75$) has approximately a 1 centimeter diameter, and this is what the experimental results show in Figure 6.18.

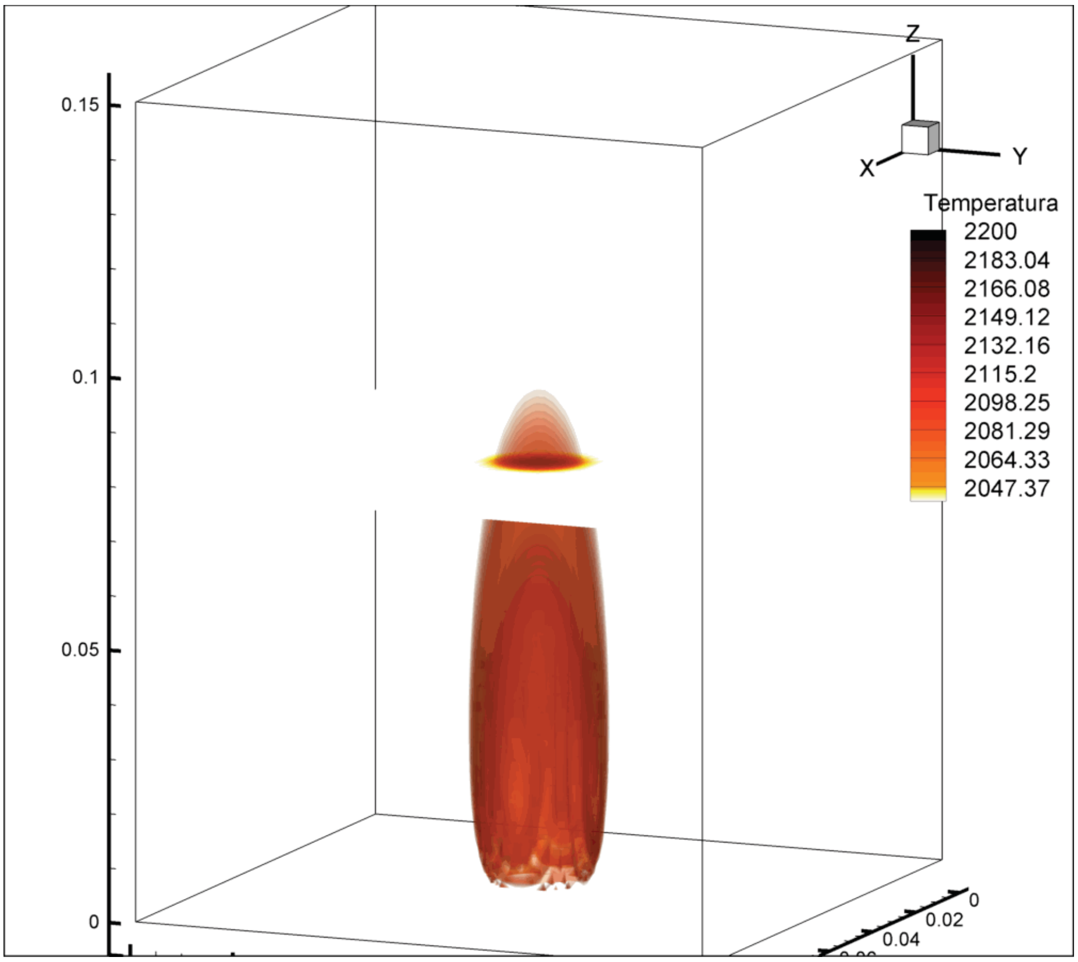


Figure 6.35: **Santoro 3D flame. Temperature isosurfaces with slicing plain at $Z = 0.75$ meters.**

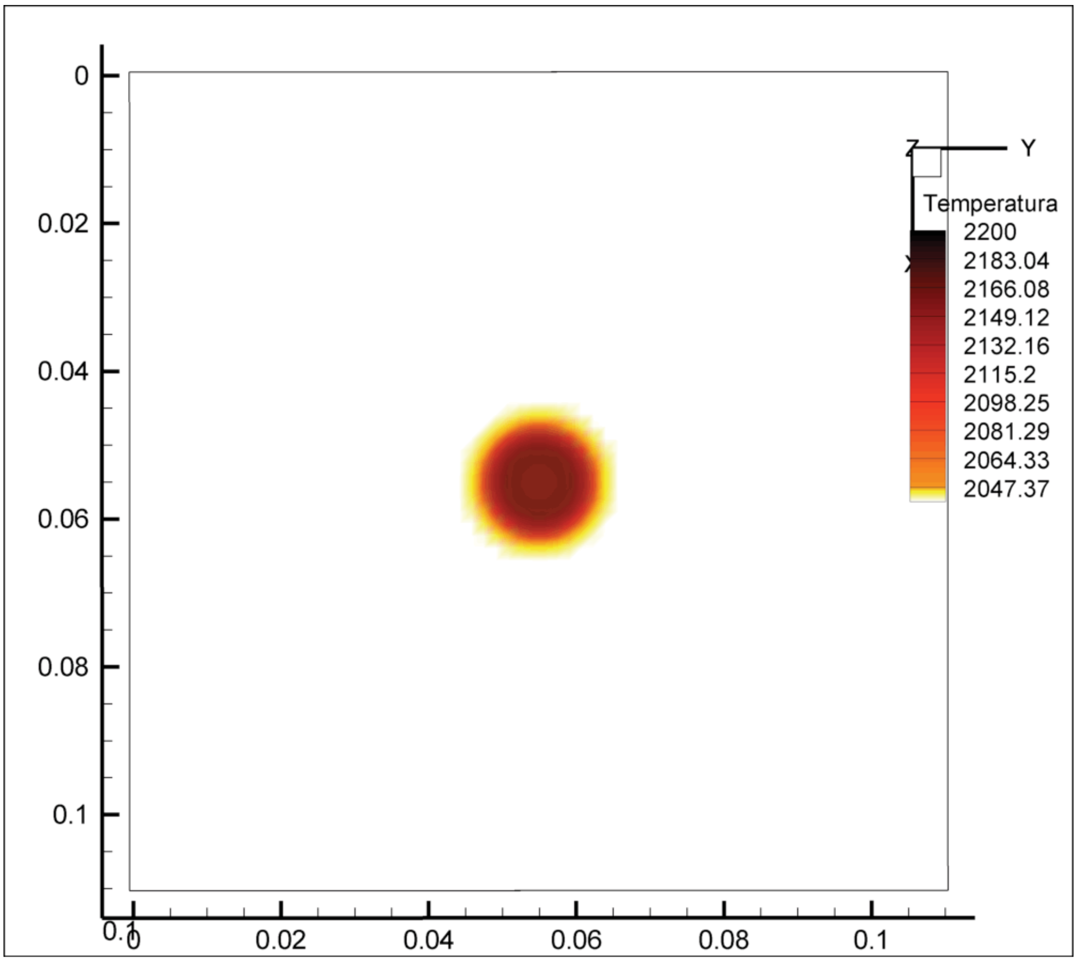


Figure 6.36: **Santoro 3D flame. Temperature profile on plain at** $z = 0.75$ **meters.**

6.4 Turbulent Flow Simulation

This simulation is intended to test the numerical implementation of the Low Reynolds k-ϵ model in a classic test case, i.e. the channel flow (see Figure 3.16). Mesh size has been estimated as described in section 3.5, and all the simulation data is shown in Table 6.13. In order to obtain a fully developed flow, a modified version of the outflow extrapolation technique is being implemented: while still adopting section's P velocity profile (see Figure 5.3) at time t^n for defining boundary conditions at time t^{n+1} in the outflow section O, the same profile is, for every global iteration, used as the inflow velocity profile. Same procedure has been adopted for the k and ϵ scalar fields. This has been a first attempt to implement streamwise (`'z-z'`) periodic boundary conditions.

$Re = 3500$	$c_f = 0.0065$	$u_\tau = 1.737$	$\delta_\nu = 0.00011$
$k_{in} = 0.10\frac{U_{in}^2}{2}$	$\mu_T = 25\ \mu_{in}$	$ncp = 762045$	$mesh\ size = 0.000112$
$Lx = 0.0002$	$Ly = 0.04$	$Ly = 0.2$	`'Periodic'='x-x'`

Table 6.13: **Turbulent Flow Simulation data**

In Figures 6.37 and 6.38 the flood contours of the Rate of Energy Dissipation (ϵ) and the Turbulent Kinetic Energy (k) are shown. It is interesting to notice how such fields, though slightly varying along the streamwise direction, show to interdependently do so in order to achieve a correct distribution of the turbulent viscosity field as shown by the good agreement between Figures 6.40 and 6.39.

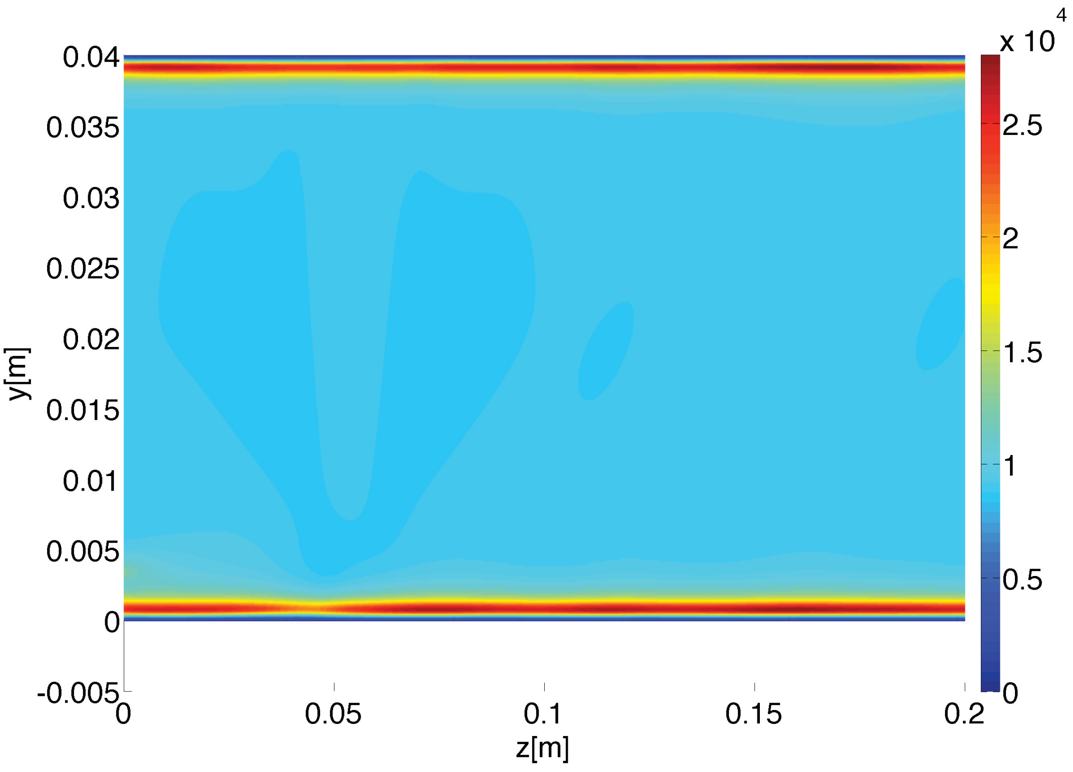


Figure 6.37: **Flooded contour of the Rate of Energy Dissipation (ϵ)**

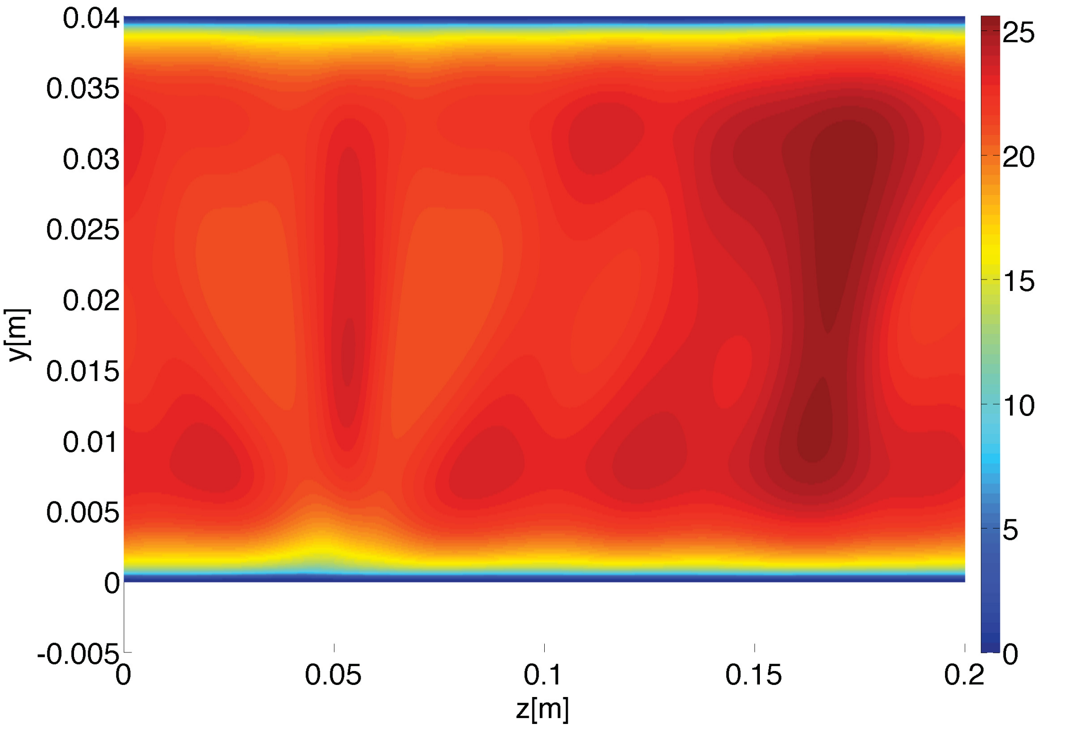


Figure 6.38: **Flooded contour of the Turbulent Kinetic Energy (k)**

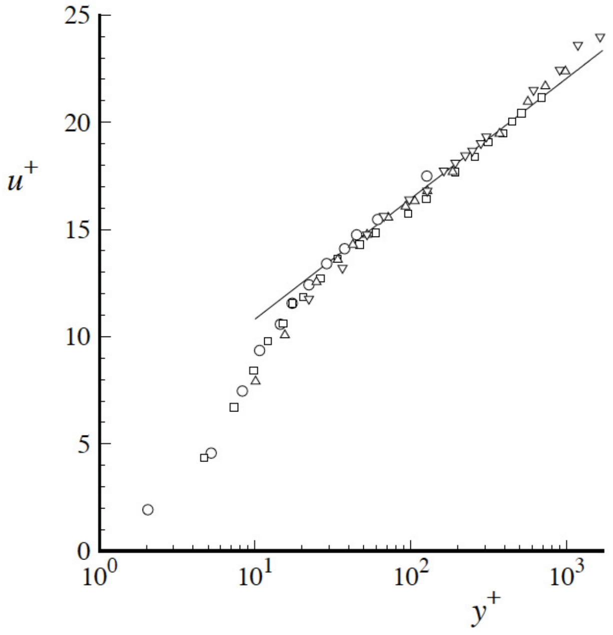


Figure 6.39: **Mean velocity profiles in fully developed turbulent channel flow measured by [22] (source [17]).**

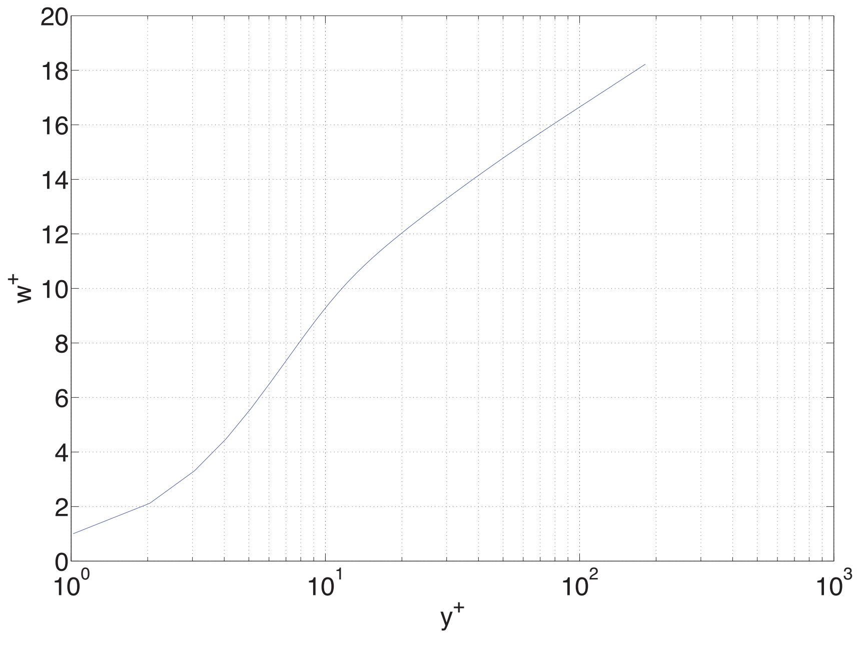


Figure 6.40: **PRIN-3D Numerical Results**

Bibliography

[1] *Star-P Programming Guide for Use with MATLAB.*

[2] T. Siegmann-Hegerfeld S. Albensoeder. Two- and three-dimensional flows in nearly rectangular cavities driven by collinear motion of two facing walls. *Exp Fluids*, 45:781–796, 2008.

[3] S. Badia and R. Codina. Algebraic pressure segregation methods for the incompressible naveri-stokes equations. *Arch Comput Methods Eng*, 15:343–369, 2008.

[4] G.K. Batchelor. *An Introduction to Fluid Dynamics.* Cambridge University Press, New York, 1973.

[5] J.H. Ferziger and M. Peric. *Computational Methods for Fluid Dynamics.* Springer-Verlag, Berlin, 2002.

[6] A. George and J. W. Liu. *Computer Solution of Large Sparse Positive Definite Systems.* Prentice-Hall,New Jersey, 1981.

[7] V. Grazioso. Su alcune correlazioni tra i metodi di ricostruzione e le proprietà funzionali delle soluzione numeriche ottenibili con schemi conservativi. Master's thesis, Università degli Studi di Napoli 'Federico II', 2005.

[8] D. Silvester H. Elman and A. Wathen. *Finite Elements and Fast Iterative Solvers.* Oxford University Press, Oxford, 2005.

[9] V. E. Howle J. Shadid H. Elman and R. Tuminaro. A parallel block multi-level preconditioner for the 3d incompressible navier-stokes equations. *J. Comput Phys*, 187:504–523, 2003.

[10] K.A. Hoffmann and S.T. Chiang. *Computational Fluid Dynamics*. Engineering Education System, Wichita, KS - USA, 2000.

[11] A. Erisman I. Duff and J. Reid. *Direct Methods for Sparse Matrices*. Oxford University Press, Oxford, 1986.

[12] R.J. Leveque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, New York, 2002.

[13] C. Meola and G. de Felice. *Fondamenti lineari per la Fluidodinamica Numerica*. Edizioni l'Ateneo, Napoli, 1996.

[14] R. Peyret, editor. *Handbook of Computational Fluid Mechanics*. Academic Press, London, 2000.

[15] H. Pitsch. Creating a flamelet library for the steady flamelet model or the flamelet/progress variable approach. User manual of: FlameMaster, A C++ Computer Program for 0D Combustion and 1D Laminar Flame Calculations, September 2006.

[16] H. Pitsch and N. Peters. A consistent flamelet formulation for non-premixed combustion considering differential diffusion effects. *Combustion and Flames*, 114:26–40, 1998.

[17] S.B. Pope. *Turbulent Flows*. Cambridge University Press, New York, 2000.

[18] R. J. Santoro R. Puri and K. C. Smyth. The oxidation of soot and carbon monoxide in hydrocarbon diffusion flames. *Combustion and Flames*, 97:125–144, 1994.

[19] C. M. Rhie and W. L. Chow. Numerical study of the turbulent flow past an airfoil with trailing edge separation. *AIAA Journal*, 21:1525–1532, 1983.

[20] D.E. Rosner. *Transport Processes in Chemically Reacting Flow Systems*. Dover Publicatons, Mineola, N.Y. - USA, 2000.

[21] A. Quarteroni R. Sacco and F. Saleri. *Matematica Numerica*. Springer-Verlag, Milano, 2008.

[22] T. Wei and W. W. Willmarth. Reynolds-number effects on the structure of a turbulent channel flow. *J. Fluid Mech.*, 204:57–95, 1989.