Università della Calabria

Facoltà di Ingegneria

Dipartimento di Difesa del Suolo "V. Marone"

Dottorato di Ricerca in Ingegneria Idraulica per l'Ambiente ed il Territorio

XXIII Ciclo

Dissertazione per il conseguimento del titolo di Dottore di Ricerca in Ingegneria Idraulica per l'Ambiente ed il Territorio

COHERENT STRUCTURES OF TURBULENCE IN WALL-BOUNDED TURBULENT FLOWS

di

Stefania Angela Ciliberti

(Settore Scientifico Disciplinare: ICAR/01)

Coordinatore:
Prof. Francesco Macchione

Supervisore:
Prof. Giancarlo Alfonsi, Ph. D.

Arcavacata di Rende (Cosenza), Italia – Novembre 2011

Abstract

Direct Numerical Simulation (DNS) of a fully developed turbulent channel flow represents a powerful tool in turbulence research: it has been carried out to investigate the main characteristics of wall-bounded turbulence. It consists of solving numerically the Navier-Stokes equations with physically-consistent accuracy in space and time. The major difficulty in performing turbulence calculations at values of the Reynolds number of practical interest lies in the remarkable amount of computational resources required. Recent advances in high performance computing, especially related to hybrid architectures based on CPU/GPU, have completely changed this scenario, opening the field of High Performance Direct Numerical Simulation of turbulence (HPDNS), to which new and encouraging perspectives have been associated with the development of an advanced numerical methodology for studying in detail turbulence phenomena.

The research activities related to the Ph. D. Program concerns the high performance direct numerical simulation of a wall-bounded turbulent flow in a plane channel with respect to the Reynolds number dependence in order to investigate

coherent structures of turbulence in the wall region. The objectives of the research have been achieved by means the construction and the validation of DNS turbulent flow databases, that give a complete description of the turbulent flow. The Navier-Stokes equations that governs the flow of a three-dimensional, fully developed, incompressible and viscous fluid in a plane channel have been integrated and a computational code based on a mixed spectral-finite difference scheme has been implemented. In particular, a novel parallel implementation of the Navier-Stokes solver on GPU architectures have been proposed in order to perform simulations at high Reynolds numbers. In order to deal with large amount of data produced by the numerical simulation, statistical tools have been developed in order to verify the accuracy of the computational domain and describe the energetic budgets that govern the energy transfer mechanisms close to the wall. Flow visualization has been provided in order to identify and evaluate the temporal and morphological evolution of flow structures in the wall region by using numerical methods for their extraction from a three-dimensional time-dependent turbulent flow database.

The first part of the Ph. D. thesis concerns a literature review of the main contributes about wall-bounded turbulence, the DNS, coherent structures and high performance Navier-Stokes solver. The historical background has been described in detail, from the earlier works until the most recent ones, underlying their importance and limitations. The problem statement concerns also the study of the physical-mathematical characteristics of the system of the Navier-Stokes equation for a three-dimensional, fully developed, incompressible and viscous flow. The second part is devoted to the study of the numerical method for the integration of the Navier-Stokes equations. A mixed spectral-finite difference technique for the numerical integration of the governing equations is devised: Fourier decomposition in both streamwise and spanwise directions and finite difference method along the wall-normal direction are used, while a third-order Runge-Kutta algorithm coupled with the fractional-step method are used for time advancement and for satisfying the incompressibility constraint. A parallel computational codes has been developed for multicore architectures; furthermore, in order to simulate the turbulence phenomenon at high Reynolds numbers, a novel parallel computational model has been developed and implemented for hybrid CPU/GPU computing systems. The third part of the Ph. D.

thesis concerns the analysis of numerical results, in order to evaluate the relationship between turbulence statistics, energy budgets and flow structures, allowing to increase the knowledge about wall-bounded turbulence for developing new predictive models and for the control of turbulence.

Sintesi

La simulazione numerica diretta (DNS, Direct Numerical Simulation) della turbolenza di un fluido all'interno di un canale piano rappresenta un utile strumento nel campo della fluidodinamica al fine di valutare nel dettaglio le caratteristiche principali di un corrente turbolenta di parete. La maggiore difficoltà che storicamente si è incontrata nell'ambito della simulazione della turbolenza considerando valori del numero di Reynolds di pratico interesse ingegneristico è rappresentata dall'onerosità delle risorse computazionali richieste: di conseguenza, per un lungo periodo di tempo, l'analisi si è concentrata su semplici casi di fluidi turbolenti. I progressi nel settore del calcolo ad alte prestazioni ha favorito la nascita di una nuova filiera di ricerca, la simulazione numerica diretta ad alte prestazioni (HPDNS, High Performance Direct Numerical Simulation) della turbolenza, consentendo lo sviluppo di una modellistica numerica molto avanzata.

Le attività sviluppate nel corso del Dottorato di Ricerca riguardano la simulazione numerica diretta della turbolenza di parete mediante il ricorso a tecniche di supercalcolo, al variare del numero di Reynolds, al fine di poter studiare le

strutture coerenti della turbolenza che si originano in prossimità della parete. Tale obiettivo è stato perseguito mediante la simulazione e la validazione di database numerici di un fluido tridimensionale, turbolento, incomprimibile e viscoso all'interno di un canale piano. Le equazioni di Navier-Stokes, che governano il moto di un fluido viscoso incomprimibile all'interno di un canale piano, sono state integrate mediante il ricorso ad un codice di calcolo basato su uno schema numerico di tipo misto spettrale-differenze finite. Si è proceduto allo sviluppo di un originale modello computazionale per l'implementazione del solutore di Navier-Stokes sulle nuove piattaforme di supercalcolo basate sulle schede grafiche, allo scopo di simulare la turbolenza considerando elevati valori del numero di Reynolds. Lo studio delle peculiarità del campo di moto simulato mediante il ricorso alla simulazione diretta è stato condotto mediante lo sviluppo di un set di strumenti statistici per la verifica del livello di accuratezza del dominio di calcolo e per la descrizione dei bilanci di energia che governano i processi di produzione, trasporto e dissipazione dell'energia cinetica turbolenta in prossimità della parete solida. Grazie alla visualizzazione scientifica è stato possibile, inoltre, identificare e valutare l'evoluzione temporale e morfologica delle strutture coerenti in prossimità della parete, attraverso il ricorso a metodi per l'estrazione delle strutture vorticose da un database tridimensionale le cui variabili sono funzioni note nel tempo e nello spazio.

La prima parte della tesi di Dottorato è incentrata sulla descrizione delle nozioni fondamentali della turbolenza e fornisce una panoramica sui principali riferimenti di letteratura che riguardano la DNS, il problema della turbolenza in un canale piano e le procedure numeriche per lo sviluppo di un solutore di Navier-Stokes ad alte prestazione e per l'estrazione delle strutture vorticose. La seconda parte riguarda lo studio di un metodo numerico per l'integrazione delle equazioni di Navier-Stokes. Tale metodo si basa su uno schema di tipo misto spettrale-differenze finite per l'integrazione delle equazioni, mentre per quanto riguarda l'avanzamento temporale si ricorre all'algoritmo di Runge-Kutta al terzo ordine di accuratezza, associato al metodo *fractional-step* per il soddisfacimento del vincolo di incomprimibilità del fluido. Sulla base dell'algoritmo numerico utilizzato per l'integrazione delle equazioni, è stato possibile sviluppare un nuovo modello computazionale: una prima versione parallela del solutore è stata sviluppata per calcoli di tipo *CPU-intensive*,

mentre per la generazione di database della DNS ad elevati numeri di Reynolds si è proceduto allo sviluppo di un nuovo algoritmo per il sistema di calcolo ibrido basato sul co-processing tra CPU e GPU. La terza ed ultima parte della tesi ha riguardato l'analisi dei risultati numerici, allo scopo di valutare le relazioni intercorrenti tra le statistiche della turbolenza, i bilanci energetici in prossimità della parete e le strutture vorticose, grazie cui è stato possibile fornire un utile contributo all'analisi dei fenomeni turbolenti per elevati valori del numero di Reynolds.

Acknowledgements

I would like to express my deep gratitude to my supervisor, Prof. Giancarlo Alfonsi for his guidance throughout this work, his contributions of time and ideas and opportunity to study at the *Dipartimento di Difesa del Suolo "V. Marone"* at the University of Calabria.

I would like to thank also Dr. Leonardo Primavera for helpful discussions about the research activities and Dr. Eng. Marco Mancini for his precious advices, support and encouragement during the Ph. D. Program.

I would also direct my thanks to my collegues, in particular Agostino Lauria, for the friendship they offered me during these years and to my special friends Antonella, Lorenzo and Manuela for their smiles, their sincere advices and support.

A lovely thank you to Marco, for his patience and love.

Last but not least, my eternal gratitude and love are guaranteed to my mother Anna, my father Santino, my brother Francesco and my grandmother Letizia for their understanding, help, great encouragement and constant support.

Contents

List of Tables

List of Figures

List of Symbol

Roman Symbols (upper-case)

$A =$ wetted area (in R_H)

$A_{ij} = \partial u_i / \partial x_i$ = velocity-gradient tensor

A_{l_1} = area of leg 1 (of hairpin vortex)

A_{l_2} = area of leg 2 (of hairpin vortex)

B_r = number of bytes read per kernel

B_w = number of bytes written per kernel

$C = 5.5 =$ additive constant (in law of the wall)

$C_f = 2\tau_w / \rho \bar{u}_b^2 =$ bulk-velocity skin-friction coefficient

$C_n =$ constants (in structure functions)

$C_K \cong 0.5 \pm 0.05$ = Kolmogorov constant

D = discriminant of the characteristic equation of A_{ij}

D_ε = viscous diffusion rate

D_{ij} = viscous diffusion rate term

D_K = viscous diffusion rate term (related to the kinetic energy)

$E_p =$ efficiency of parallel calculation (processors)

$E_{pd} =$ efficiency of parallel calculation (processors and disk)

$E_f =$ effectiveness of parallel calculation (processors)

$E_{fd} =$ effectiveness of parallel calculation (processors and disk)

$E(k) =$ energy spectrum

$F_{u'}, F_{v'}, F_{w'} =$ flatness factors of velocity fluctuations

$K =$ average kinetic energy of the turbulent field

L = reference length (generic)

$L_x, L_y, L_z =$ dimensions of computational domain along x, y, z

$L_x^+, L_y^+, L_z^+ =$ dimensions of computational domain along x, y, z (wall units)

$N =$ grid points in each direction of computing domain

$N_x, N_y, N_z =$ number of grid points of computing domain along x, y, z

N_x^{mv}, N_y^{mv} = grid points along the x- and y directions within minimum volume

$P =$ wetted perimeter (in R_H)

$P, Q, R =$ scalar invariants of the velocity-gradient tensor A_{ij}

$P_\varepsilon^1 =$ mixed production rate of dissipation

$P_\varepsilon^2 =$ production rate of the dissipation by mean-velocity gradient

$P_\varepsilon^3 =$ gradient production rate

$P_\varepsilon^4 =$ turbulent production rate

$P_K =$ turbulent kinetic-energy production term

P_{ij} = Reynolds-stress production rate tensor

P_{ij}^b = Reynolds-stress anisotropic production term

$Q_1, Q_2, Q_3, Q_4 =$ first-, second-, third- and fourth-quadrant (in quandrant analysis)

$R_H = A/P =$ hydraulic radius

R_{ij} = velocity-correlation tensor

$Re = UL/\nu$ = Reynolds number based on U and L

$Re_b = \bar{u}_b h/\nu =$ bulk-velocity Reynolds number (based on $\bar{u}_b$ and h)

$Re_c = \bar{u}_c h/\nu =$ centreline-velocity Reynolds number (based on $\bar{u}_c$ and h)

$Re_d = \bar{u}_\infty d/\nu =$ Reynolds number based on $\bar{u}_\infty$ and d (circular cylinder)

$Re_h =$ Reynolds number (in flow over a backward-facing step)

$Re_K = u_\eta \eta/\nu$ = Reynolds number based on u_η and η

$Re_l = ul/\nu$ = Reynolds number based on u and l

$Re_\delta =$ Reynolds number based on Blasius boundary-layer displacement thickness

$Re_\lambda = u\lambda/\nu$ = Reynolds number based on u and λ

$Re_\tau = u_\tau h/\nu =$ friction-velocity Reynolds number (based on u_τ and h)

$Re_{\tau H} = u_\tau R_H/\nu =$ Reynolds number based on u_τ and R_H

$Re_\theta = \bar{u}_\infty \theta/\nu =$ momentum-thickness Reynolds number (based on $\bar{u}_\infty$ and θ)

$S_{ij} =$ rate-of-strain tensor

$\bar{S}_{ij} =$ mean rate-of-strain tensor

$S'_{ij} =$ fluctuating rate-of-strain tensor

$S_p =$ speedup of parallel calculation (processors)

$S_{pd} =$ speedup of parallel calculation (processors and disk)

$S_{u'}, S_{v'}, S_{w'} =$ skewness factors of velocity fluctuations

$T =$ time as a dimension (also nondimensional runtime of computational code)

$T_\varepsilon =$ turbulent transport rate

$T_{tot} =$ total mean stress of the turbulent field

$T_{ij} =$ turbulent transport rate term

$T_{ij}^b =$ anisotropy transport rate

$T_K =$ turbulent transport rate term (related to the kinetic energy)

$U =$ reference velocity (generic)

$W_{ij} =$ rate-of-rotation tensor

$Y =$ turbulent dissipation rate

Roman Symbols (lower-case)

$b_{ij} =$ Reynolds-stress anisotropy tensor

$d =$ circular-cylinder diameter

$d_{ij} =$ dissipation-rate anisotropy tensor

$e_{ijk} =$ alternating-unit tensor

f_{seq} = fraction of elapsed time inherently sequential (of computational code)

f_{seq}^{CPU} = sequential fraction of computational components in elapsed time

f_{par}^{CPU} = parallel fraction of computational components in elapsed time

$f_{seq}^{I/O}$ = sequential fraction of I/O components in elapsed time

$f_{par}^{I/O}$ = parallel fraction of I/O components in elapsed time

$h =$ plane-channel half height

k = wavenumber

$k'(n_p, n_d)$ = scaling function for overhead (in computational code)

$k''(n_p, n_d)$ = scaling function for overhead (in computational code)

$l =$ integral length scale (of turbulence)

$l_1 =$ leg one (of hairpin vortex)

$l_2 =$ leg two (of hairpin vortex)

$n_p =$ number of processors

$p =$ pressure

$\overline{p} =$ mean pressure

$p' =$ fluctuating pressure

$r =$ separation distance in structure function (also radial coordinate)

$r, \theta, z =$ cylindrical coordinates

$s^+ = su_\tau / \nu$ = riblet spacing (wall units)

$t = l/u$ = integral time scale of turbulence (also time coordinate)

$t_i =$ unit tangent

$t/u_\tau h$ = non-dimensional time unit

$t^+ = tu_\tau^2 / \nu =$ time unit (wall units)

t_{DB} = data-base calculated time

u = integral velocity scale (of turbulence)

u_i = velocity vector (u,v,w)

$\mathbf{u}$ = velocity vector (symbolic notation)

$\overline{u}_i$ = mean-velocity vector

$\overline{u}_b$ = bulk mean x-velocity

$\overline{u}_c$ = mean centerline x-velocity (channel)

$\overline{u}_\infty$ = free-stream velocity

$u_\tau = \sqrt{\tau_w/\rho}$ = friction velocity

$u_\eta = \nu^{1/4}\varepsilon^{1/4}$ = Kolmogorov velocity microscale

$u^+ = \overline{u}/u_\tau$ = nondimensional mean x-velocity (wall units)

u_i' = fluctuating-velocity vector (u',v',w')

$-\overline{u'v'}$ = Reynolds shear stress (divided by density)

u'_{rms},v'_{rms},w'_{rms} = *rms* velocity fluctuations

x_i = spatial coordinates (x,y,z)

x_i^+ = spatial coordinates (wall units, x^+,y^+,z^+)

Greek Symbols (upper-case)

Δu = velocity difference (in structure function)

Δt^+ = temporal resolution (wall units)

Δx^+ = spatial resolution along x (wall units)

Δy_c^+ = spatial resolution along y at center (wall units)

Δy_w^+ = spatial resolution along y at wall (wall units)

Δz^+ = spatial resolution along z (wall units)

Π_ε = pressure transport rate

Π_{ij} = velocity pressure-gradient term

$\Pi_{s,ij}$ = pressure-strain term

$\Pi_{d,ij}$ = pressure-diffusion term

Π_{ij}^{b} = anisotropy velocity pressure gradient term

Π_{K} = velocity pressure-gradient terms (related to the kinetic energy)

Greek Symbols (lower-case)

β = base of the floating-point representation

δ_{ij} = Kronecker's delta

δ_{n} = departure from K41

ε = average dissipation-rate of turbulent kinetic energy

ε_{ij} = dissipation rate term

ε_{ij}^{b} = anisotropy dissipation rate

$\zeta_{n} = (n/3) - \delta_{n}$ = scaling exponent in K62

$\kappa = 0.40$ = von Kármán constant

$\eta = \nu^{3/4}/\varepsilon^{1/4}$ = Kolmogorov spatial microscale

η^{+} = Kolmogorov spatial microscale (wall units)

θ = momentum thickess (also azimuthal coordinate)

λ = Taylor scale of turbulence (also eigenvalue)

λ_{cr} = real part of the complex eigenvalue pair of A_{ij}

λ_{ci} = imaginary part of the complex eigenvalue pair of A_{ij}

μ = fluid dynamic viscosity

ν = fluid kinematic viscosity

ν/u_{τ} = viscous-length unit

ν/u_{τ}^{2} = viscous-time unit

ρ = fluid density

$\tau_{w} = \mu(d\overline{u}/dy)_{w}$ = mean shear stress at wall

$\tau_{ij} = \overline{u_{i}'u_{j}'}$ = Reynolds-stress tensor (divided by density)

$\tau_{\eta} = \nu^{1/2}/\varepsilon^{1/2}$ = Kolmogorov temporal microscale

$\tau_\eta^+ =$ Kolmogorov temporal microscale (wall units)

φ_i = empirical eigenfunction (KL decomposition)

ψ_i = vector function (KL decomposition)

$\omega_k = e_{ijk}\,\partial u_i / \partial x_j$ = vorticity vector

$\boldsymbol{\omega} = \nabla \times \boldsymbol{u}$ = vorticity vector

Acronyms

ALE = Arbitrarian Lagrangean Eulerian (formulation)

API = Application Programming Interface

CBE = Cell Broadband Engine

CFD = Computational Fluid Dynamics

CPU = Central Processing Unit

CUDA = Compute Unified Device Architecture

DDMP = Domain Decomposition Message Passing (programming model)

DDR = Double Data Rate

DNS = Direct Numerical Simulation (of turbulence)

DRAM = Dynamic Random Access Memory

EB = Exa Byte (10^{18})

ES = Earth Simulator (computer)

FDL = Fluid Dynamics Laboratory

FFT = Fast Fourier Transform

FMA = Fused Multiply Add

FPGA = Field Programmable Gate Array

FSI = Fluid Structure Interaction

Flop/s = Floating Point Operation per second

GB = Giga Byte (10^9)

GB/s = Giga Byte per second (10^9)

GFlop/s = Giga Floating Point Operation per second (10^9)

GPGPU = General Purpose Graphic Processing Unit

HPC = High Performance Computing

HPDNS = High Performance Direct Numerical Simulation (of turbulence)

HPF = High Performance Fortran

ILP = Instruction Level Parallelism

I/O = Input/Output (operations)

KL = Karhunen Loèwe (decomposition)

LSI = Large Scale Integration

MB = Mega Byte (10^6)

MB/s = Mega Byte per second (10^6)

MFlop/s = Mega Floating Point Operation per second (10^6)

MIMD = Multiple Instruction Multiple Data (computer)

MPI = Message Passing Interface

MPP = Massively Parallel Processors

NPB = NAS Parallel Benchmarks

PB = Peta Byte (10^{15})

PB/s = Peta Byte per second (10^{15})

PFlop/s = Peta Floating Point Operation per second (10^{15})

RAM = Random Access Memory

rhs = right-hand-side

rms = root mean square

RSH = Refined Similarity Hypothesis

SIMD = Single Instruction Multiple Data (computer)

SISD = Single Instruction Single Data (computer)

SMP = Symmetric Multi Processor

SMT = Symmetric Multi Threaded

SPMD = Single Program Multiple Data (computer)

TB = Tera Byte (10^{12})

TB/s = Tera Byte per second (10^{12})

TFlop/s = Tera Floating Point Operation per second (10^{12})

ULP = Unit of Least Precision

VIV = Vortex Induced Vibration

VL = Vector Length

VLSI = Very Large Scale Integration

ZPGFPBL = Zero Pressure Gradient Flat Plate Boundary Layer

Chapter 1
Introduction

Over the years, the issue of turbulence has attracted the attention of academic research because of the challenges that its understanding represents for scientists and the impact on engineering applications. From a physical point of view, a turbulence flow is unpredictable, since it is impossible to give an exact deterministic prediction of its evolution due to the non-linearity of the Navier-Stokes equations that represent the governing equations of these phenomena. In the last decades, researchers have made intense efforts for trying to overcome this difficulty, by using both the experimental approach and the theoretical approach of the scientific method; the key element is represented by *modern fluid technology* that provides the tools for understanding and analyzing the results of theory and experiments.

The study of turbulence near walls has been consistently studied and represents an important scientific field to understand since it has a strong impact in engineering, environmental and industrial applications; for example, it is responsible of drag on surface and dispersion of scalars and pollutants in a variety of phenomena.

One method for studying turbulence is to use laboratory techniques such as hot wire anemometry (HWA), laser Doppler anemometry (LDA), ultrasonic Doppler velocimetry (UDV) and particle imaging velocimetry (PIV). A second class of methods involves numerical simulations related to the integration of the three-dimensional unsteady Navier-Stokes equations on a computational domain with adequate accuracy in space and time.

Considering the numerical approach, several methods can be used such as the Reynolds Averaged Navier-Stokes equations (RANS) and the Large Eddy Simulation (LES). The instantaneous range of scales in turbulent flows increases rapidly with the Reynolds number. As a result, the engineering computation of turbulent flows therefore relies on simpler descriptions; instead of solving for the instantaneous flow-field, the statistical evolution of the flow is sought. Approaches based on the RANS equations are most prevalent and involve computing one-point moments such as mean velocity and turbulent kinetic energy. Instead, LES directly computes the large energy containing scale, while modeling the influence of small scales. In order to avoid modeling at any scale and for a complete description of turbulent flow, where the flow variables, such as velocity and pressure, are known as a function of space and time can only be obtained by numerically solving the Navier-Stokes equations. These numerical solutions are named Direct Numerical Simulations (DNS); it allows to deal with any range of scale of turbulence flows and, furthermore, allows to understand the role of viscosity, that is responsible of the balance between energy production, transport and dissipation close to the walls.

One disadvantage of DNS is the necessity of a paramount number of computational resources for the simulation of the Navier-Stokes equations at Reynolds numbers of practical interest. Furthermore, the DNS produces a large amount of data related to the fluid flow field (velocity and pressure) that need to be analyzed in order to understand the role of heat, mass and momentum transport during the turbulent phenomena.

Recent advances in high performance computing, especially related to hybrid architectures based on CPU/GPU computing, are revolutionizing and affecting scientific research simulation by providing several orders of magnitude of increased computing performance, allowing researchers to work with more accurate,

computationally expensive, approximations and numerical methods. Clearly, one has to rethink the computational models in order to create scalable, high performance applications based on this new technology.

The present thesis is related to the design and implementation of novel computational models and algorithms on hybrid CPU/GPU architectures, consisting of multicore CPUs and discrete GPU cards in order to solve the three-dimensional, fully turbulent, incompressible and viscous flows by using the DNS tool. Moreover, in order to deal with large amount of data produced by the numerical simulations, several statistical tools, event-detection techniques and methods for the eduction of coherent structures eduction have been considered.

The thesis is organized as follows:

- *Chapter 2* provides a perspective on turbulence fundamentals, about the scales of turbulence and mechanisms that govern the energy-transfers in a fluid flow;

- *Chapter 3* shows the state-of-the-art about direct numerical simulation, the problem of the plane channel, the turbulent flow structures and high performance computing techniques for solving numerically the Navier-Stokes equations over complex computational domains;

- *Chapter 4* describes the state-of-the-art of high performance computing techniques and metrics for performance measure. In particular, it describes recent advances on hybrid architectures CPU/GPU;

- *Chapter 5* is devoted to the design and development of novel computational models and algorithms for the numerical integration of the Navier-Stokes equations related to a three-dimensional, fully developed, incompressible and viscous flow in a plane channel. The Navier-Stokes solver is based on a mixed spectral-finite difference method associated to a third-order Runge-Kutta procedure for time advancement and the fractional-step method for satisfying the incompressibility constraint;

- *Chapter 6* shows the results of the DNS of a fully developed channel flow at Reynolds number $\mathrm{Re}_\tau = 200, 400, 600$; in particular, turbulence statistics are analyzed to verify the accuracy of the simulations and to evaluate the relationships between the Reynolds stresses, kinetic energy, dissipation and anisotropy budgets;

- *Chapter 7* presents the numerical results in terms of flow structures, morphological evolution in time and strength of a vortex population, in conjunction with ejection and sweep events, by considering the DNS results obtained at $\mathrm{Re}_\tau = 200, 400, 600$;

- *Chapter 8* provides conclusions, final remarks and future works.

Chapter 2

Turbulence Fundamentals

2.1 Introduction

In fully developed turbulence – the state in which the averaged quantities change slowly with respect to the downstream distance or time – there is a rough equilibrium between the rate at which turbulence energy is produced (i.e. transferred from the mean flow to the turbulence) and the rate at which it is dissipated (i.e. transferred to the internal energy of the fluid by the action of viscosity). Such flows are called *equilibrium turbulent flows*.

Many turbulent flows of engineering interest are dominated by shear, such as wall-shear layers and free-shear layers. This means that the largest changes in the velocity occur in direction that is approximately normal to the main-flow one. A *homogeneous turbulent flow* is a flow whose state is independent on the location from a statistical viewpoint. This means that the measurement of any averaged quantity gives identical results at any point of the flow field. The simplest homogeneous flow is the *isotropic* one, in which no strain of any kind exists. Thus,

no energy is added to turbulence (there is no production) and turbulence decays. In laboratory, a homogeneous and isotropic flow can be obtained by passing a fluid flow through a passive grid, that produces uniform turbulence. Numerically (and not physically) it is possible to force the large scales of an isotropic flow, so that an isotropic turbulent state is maintained in a statistical steady state.

In turbulent flows there is a wide range of length *scales*. The most noticeable are the *integral* or outer scale and the *Kolmogorov scale*. The *Taylor scale* is a third length scale, lying between these two, that is used to characterize homogeneous flows. Thus, turbulence can be characterized by a number of length scales, at least one for the energy-containing range and one for the dissipative range. Whether a turbulence is simple or not depends on how many scales are necessary to describe the energy-containing range of scales. For example, if a turbulent flow involves more than one production mechanism (such as shear and buoyancy) there will be more than one important length scale. Other cases are possible. As an example, turbulence may be produced by shear in a boundary layer, which is then subjected to a strain rate. For a while, the turbulence will have two length scales, one corresponding to the initial boundary-layer turbulence and the other associated to the strain rate to which the flow is subjected.

Turbulence may also have different length scales in different directions. Consider a turbulent flow having a single length scale l in the energy-containing range. One can take this scale to be the integral length one (it will be proportional to a length that characterizes the geometry of the flow), that can be in general defined in terms of the fluctuating-velocity autocorrelation coefficient in some direction (note that correlations tend to emphasize large-scale effects and hide small-scales effects). Consider also that turbulence is characterized by a single velocity scale u, that can be in general taken as being the *rms* fluctuating velocity. A Reynolds number can be defined as:

$$\mathrm{Re}_l = \frac{ul}{\nu} \tag{2.1}$$

where ν is the fluid kinematic viscosity.

Many turbulent flows have a large mean velocity and small fluctuations. The Taylor's hypothesis is the assumption that a large convection velocity U_c sweeps a frozen turbulent state past the position of interest.

The present chapter is organized as follows: in Section 2.2, the main issues on scales are discussed; then, Section 2.3 describes the scale of turbulence, based on Kolmogorov theory; finally, in Section 2.4 the energy-transfer mechanisms in turbulent flows are analyzed in detail, with reference to the energy-cascade model of Kolmogorov.

2.2 Issues on scales

According to Reynolds decomposition, the dependent variables of the Navier-Stokes equations can be decomposed into *mean* and *fluctuating parts*, and averaged, so that the Reynolds Averaged Navier-Stokes equations (RANS) are obtained (in *Appendix A*, an outline is given of a number of turbulence-transport equations, as derived from the RANS equations). The equation of the mean flow $\overline{u}_i\overline{u}_i/2$ in the stationary-flow case is:

$$\rho\overline{u}_j\frac{\partial}{\partial x_j}\left(\frac{1}{2}\overline{u}_i\overline{u}_i\right)=\frac{\partial}{\partial x_j}\left(T_{ij}^{tot}\overline{u}_i\right)-T_{ij}^{tot}\frac{\partial\overline{u}_i}{\partial x_j} \tag{2.2}$$

or also:

$$\rho\overline{u}_j\frac{\partial}{\partial x_j}\left(\frac{1}{2}\overline{u}_i\overline{u}_i\right)=\frac{\partial}{\partial x_j}\left(T_{ij}^{tot}\overline{u}_i\right)-T_{ij}^{tot}\overline{S}_{ij} \tag{2.3}$$

where

$$T_{ij}^{tot}=\overline{T}_{ij}-\rho\overline{u_i'u_j'}=-\overline{p}\delta_{ij}+2\mu\overline{S}_{ij}-\rho\overline{u_i'u_j'} \tag{2.4}$$

is the total mean stress of the turbulent field. The contribution of the turbulent motion to the mean stress is $-\rho\overline{u_i'u_j'}$, the *Reynolds-stress tensor*. This tensor is symmetric, the diagonal components are normal stresses (negative pressures) and the off-diagonal components are shear stresses.

The dissipation term in Eq. (2.2) can be written as:

$$T_{ij}^{tot}\overline{S}_{ij} = 2\mu\overline{S}_{ij}\overline{S}_{ij} - \rho\overline{u_i'u_j'}\overline{S}_{ij} \tag{2.5}$$

where the first term on the rhs of Eq. (2.5) is the viscous part of the dissipation of kinetic energy of the mean flow, while the second term is the turbulent part. Since the turbulent stress provides the turbulent part of the dissipation, the kinetic energy of turbulence gains from this dissipation. For this reason, the term $-\rho\overline{u_i'u_j'}\overline{S}_{ij}$ is called the *turbulent kinetic energy production term*. By inserting Eq. (2.4) in Eq. (2.2), one obtains:

$$\overline{u}_j\frac{\partial}{\partial x_j}\left(\frac{1}{2}\overline{u}_i\overline{u}_i\right) = \frac{\partial}{\partial x_j}\left(-\frac{\overline{p}}{\rho}\overline{u}_j + 2\nu\overline{u}_i\overline{S}_{ij} - \overline{u_i'u_j'}\overline{u}_i + \overline{u_i'u_j'}\overline{S}_{ij} - 2\nu\overline{S}_{ij}\overline{S}_{ij}\right) \tag{2.6}$$

where:

- the first three terms on the rhs of Eq. (2.6) represent the flux of work associated, respectively, with mean pressure, mean viscous stress and turbulent stress;
- the last two terms represent the dissipation related to turbulent stress and viscous stress, respectively.

A length scale l, a velocity scale u and a time scale $t = l/u$ can be introduced. These scales lead to the following relations:

$$\frac{\partial\overline{u}_i}{\partial x_i} \approx \frac{u}{l} \tag{2.7}$$

$$-\overline{u_i'u_j'} \approx u^2 \tag{2.8}$$

$$\overline{S}_{ij} \approx \frac{u}{l} \tag{2.9}$$

$$\overline{u_i' u_j'} \overline{S}_{ij} \approx u l \overline{S}_{ij} \overline{S}_{ij} \tag{2.10}$$

$$-\overline{u_i' u_j'} \overline{S}_{ij} \approx u l \overline{u} \overline{S}_{ij} \tag{2.11}$$

in which both mean and fluctuating components are scaled relatively to the velocity scale u. By comparing Eqs. (2.10) and (2.11) with the first and second viscous terms of Eq. (2.6), respectively, one obtains:

$$\frac{-\overline{u_i' u_j'} \overline{u}_i}{2\nu \overline{S}_{ij} \overline{u}_i} \approx \mathrm{Re}_l \tag{2.12}$$

$$\frac{\overline{u_i' u_j'} \overline{S}_{ij}}{2\nu \overline{S}_{ij} \overline{S}_{ij}} \approx \mathrm{Re}_l \tag{2.13}$$

where $\mathrm{Re}_l = ul/\nu$ is the Reynolds number as in Eq. (2.1), based on the integral scales. This result shows that the terms associated with the turbulent stress are Re_l times larger than the terms associated with the viscous terms. Being Re_l usually very large, the viscous terms in equation Eq. (2.6) can usually be neglected, meaning that the structure of a turbulent flow tends to be independent of viscosity.

The equation that governs the mean kinetic energy of the turbulent field $\overline{u_i' u_i'}/2$ is obtained by multiplying the momentum equation of the instantaneous field by u_i, taking the average of all terms, and subtracting Eq. (2.6). In the stationary-flow case, one obtains:

$$\overline{u}_j \frac{\partial}{\partial x_j}\left(\frac{1}{2}\overline{u_i' u_i'}\right) = \frac{\partial}{\partial x_j}\left(-\overline{\frac{p'}{\rho} u_j'} + 2\nu \overline{u_i' S_{ij}'} - \frac{1}{2}\overline{u_i' u_i' u_j'} - \overline{u_i' u_j'} \overline{S}_{ij} - 2\nu \overline{S_{ij}' S_{ij}'}\right) \tag{2.14}$$

where

$$S'_{ij} = \frac{1}{2}\left(\frac{\partial u'_i}{\partial x_j} + \frac{\partial u'_j}{\partial x_i} \right) \tag{2.15}$$

is the strain rate tensor of the fluctuating field. In Eq.(2.14):

- the first three rhs terms denote, respectively, the net flux of work associated with the fluctuating pressure, the fluctuating viscous stress and the turbulent stress;
- the last two terms denote the turbulent and viscous dissipations.

It is to be noted that the turbulent production term $-\overline{u'_i u'_j}\overline{S}_{ij}$ in Eqs. (2.6) and (2.14) has opposite sign. This term allows to exchange kinetic energy between mean flow and turbulence. In general, the exchange energy involves a loss to mean flow and a gain to turbulence. The last term in Eq. (2.14) is the rate at which the fluctuating viscous stresses perform deformation work against the fluctuating strain rate. It denotes a loss of energy and, being quadratic in S'_{ij}, is the *viscous dissipation of turbulent kinetic energy*. Unlike the dissipation related to mean viscous stresses in Eq. (2.14), this term is essential to the dynamics of turbulence and cannot be neglected. Equation (2.15) can be written as:

$$\overline{u}_j \frac{\partial K}{\partial x_j} = P_K - \varepsilon - \frac{\partial D_j}{\partial x_j} \tag{2.16}$$

stating that the mean of turbulent kinetic energy of fluctuations is balanced by the production, the dissipation and the diffusion of this energy. In the case of steady, homogeneous, pure-shear flow, all averaged quantities except $\overline{u}_i$ are independent of position and $\overline{S}_{ij}$ is a constant, and Eq. (2.16) reduces to:

$$P_K = \varepsilon \tag{2.17}$$

i.e. the balance between the production rate of turbulent kinetic energy by Reynolds stresses and the rate of viscous dissipation of turbulent fluctuations. By using Eqs. (2.8) and (2.9), one has:

$$-\overline{u_i' u_j'} \approx u^2 \tag{2.18}$$

$$\overline{S}_{ij} \approx \frac{u}{l} \tag{2.19}$$

By using Eq. (2.11), one has:

$$\overline{u_i' u_j'} \overline{S}_{ij} \approx u l \overline{S}_{ij} \overline{S}_{ij} \tag{2.20}$$

and from Eq. (2.17) one obtains:

$$u l \overline{S}_{ij} \overline{S}_{ij} \approx \nu \overline{S_{ij}' S_{ij}'} \tag{2.21}$$

or also:

$$\frac{\overline{S_{ij}' S_{ij}'}}{\overline{S}_{ij} \overline{S}_{ij}} \approx \frac{u l}{\nu} = \mathrm{Re}_l \tag{2.22}$$

Thus, being usually Re_l very large:

$$\overline{S_{ij}' S_{ij}'} >> \overline{S}_{ij} \overline{S}_{ij} \tag{2.23}$$

showing that the fluctuating-strain rate S_{ij}' is much larger than the mean-strain rate $\overline{S}_{ij}$. Since strain rates have dimensions of T^{-1}, the eddies, contributing most to the dissipation of kinetic energy, have very small convection time scales as compared with the time scale of the mean flow. Accordingly, the direct interaction between the

fluctuating strain rate and the mean strain rate is negligible for large Reynolds numbers. The above considerations suggest that the scales for fluctuations should be different from the scales for the mean flow.

2.3 Scales of turbulence

The integral scales l, u and $t = l/u$ previously introduced are the scales for the large turbulence eddies and are applied to the production term of Eq. (2.17). Note that the time scale t represents the life time of the large eddies (large eddy turn-over time) and that to the large eddies is associated a kinetic energy per unit mass proportional to u^2. Some smaller scales can be introduced, namely a length scale η, a velocity scale u_η and a time scale $\tau_\eta = \eta/u_\eta$, characterizing the small dissipative turbulence scales. They are the *Kolmogorov microscales* ([1], [2]) and are applied to the dissipation term of Eq. (2.17).
Equation (2.17) can be written in terms of integral scales and microscales as:

$$P_K = u^2 \frac{u}{l} = \nu \frac{u_\eta^2}{\eta^2} = \varepsilon \tag{2.24}$$

showing that, being ε ($[\varepsilon] = L^2/T^3$) the same as the average rate of energy input, the viscous dissipation of energy can be estimated from the large-scale inviscid dynamics. Thus, the dissipation may be interpreted as a passive process that proceeds at a rate dictated by the inviscid inertial behavior of the large eddies.
For the isotropic limit of the homogeneous flow, Eq. (2.24) becomes:

$$\varepsilon = \frac{u^3}{l} \tag{2.25}$$

that allows an immediate estimation of the power of a turbulent flow (ε is the rate of energy dissipation per unit mass) from the estimate of characteristic sizes and velocities.

Letting $l \to \eta$ and $u \to u_\eta$, one has:

$$u_\eta = \left(\eta \varepsilon^{1/3}\right) \tag{2.26}$$

and from the last proportionality of Eq. (2.25):

$$u_\eta = \eta \left(\frac{\varepsilon}{\nu}\right)^{1/2} \tag{2.27}$$

By eliminating u_η from Eqs. (2.26) and (2.27), one obtains the Kolmogorov length scale:

$$\eta = \left(\frac{\nu^3}{\varepsilon}\right)^{1/4} \tag{2.28}$$

The insertion of Eq. (2.28) in Eq. (2.26) or Eq. (2.27) gives the Kolmogorov velocity scale:

$$u_\eta = \left(\nu \varepsilon\right)^{1/4} \tag{2.29}$$

and the ratio between Eqs. (2.28) and (2.29) gives the Kolmogorov time scale:

$$\tau_\eta = \left(\frac{\nu}{\varepsilon}\right)^{1/2} \tag{2.30}$$

Expressions (2.28)-(2.30) show that the small-scale motion of turbulence only depends from the momentum diffusivity (kinematic viscosity) and that the large-scale energy supply for dissipation ε. Thus, the dissipation occurs only at the level of small-scale motion and at this level turbulence is homogeneous and isotropic. The Reynolds number in these scales:

$$\mathrm{Re}_K = \frac{u_\eta \eta}{\nu} = 1 \tag{2.31}$$

shows that the small-scale motion is slow and viscous. The two sets of scales, the scales of the large eddies l, u, t and those of the small eddies η, u_η, τ_η lead to the so-called *two-scale turbulence model*.

From Eq. (2.25) and Eqs. (2.28)-(2.30), one also has:

$$\frac{l}{\eta} = \mathrm{Re}_l^{3/4} \tag{2.32}$$

$$\frac{u}{u_\eta} = \mathrm{Re}_l^{1/4} \tag{2.33}$$

$$\frac{t}{\tau_\eta} = \mathrm{Re}_l^{1/2} \tag{2.34}$$

Another length scale can be introduced in this context, in particular referred to the case in which u_η would be amplified to u of large scales, i.e. the (unknown so far) length scale x associated to

$$\tau_\eta^{-1} = \frac{u_\eta}{\eta} = \frac{u}{x} \tag{2.35}$$

when u_η tends to u. One can expect that η amplifies to l, but this is not exactly true. By combining Eqs. (2.25) and (2.35), one obtains:

$$u^2 \frac{u}{l} = \nu \frac{u_\eta^2}{\eta^2} = \nu \frac{u^2}{x^2} \tag{2.36}$$

and from the first and last term of Eq. (2.36):

$$\frac{x}{l} = \frac{1}{\mathrm{Re}_l^{1/2}} << 1 \tag{2.37}$$

This result shows that the unknown scale must be smaller than the integral scale, and, on the basis of Eq. (2.35), also greater than the Kolmogorov scale. The new scale is the Taylor microscale λ, such as:

$$\tau_\eta^{-1} = \frac{u_\eta}{\eta} = \frac{u}{\lambda} \tag{2.38}$$

Being defined by the velocity of large scales and the time of small scales, the Taylor scale is neither large nor small, but rather an intermediate scale. The Taylor scale is frequently estimated as:

$$S'_{ij} = \frac{u}{\lambda} \tag{2.39}$$

Equation (2.34) may be rearranged in terms of Eq. (2.38) as follows:

$$u^2 \frac{u}{l} = \nu \frac{u^2}{\lambda^2} \tag{2.40}$$

that gives:

$$\frac{\lambda}{l} = \frac{1}{\mathrm{Re}_\lambda} = \frac{1}{\mathrm{Re}_l^{1/2}} \tag{2.41}$$

where

$$\mathrm{Re}_\lambda = \frac{u\lambda}{\nu} \tag{2.42}$$

is the Reynolds number based on the Taylor scale.

Equation (2.24) may also be rearranged in terms of Eq. (2.31) as follows:

$$\frac{\eta}{l} = \frac{1}{\mathrm{Re}_\eta^3} = \frac{1}{\mathrm{Re}_l^{3/4}} \tag{2.43}$$

The ratio between Eqs. (2.41) and (2.43) gives:

$$\frac{\eta}{\lambda} = \frac{1}{\mathrm{Re}_\eta} = \frac{1}{\mathrm{Re}_\lambda^{1/2}} = \frac{1}{\mathrm{Re}_l^{1/4}} \tag{2.44}$$

or also:

$$\mathrm{Re}_\eta = \mathrm{Re}_\lambda^{1/2} = \mathrm{Re}_l^{1/4} \tag{2.45}$$

where

$$\mathrm{Re}_\eta = \frac{u\eta}{\nu} \tag{2.46}$$

is the Reynolds number based on the Kolmogorov scale.
Equations (2.41) and (2.43) also give the following additional result:

$$\left(\frac{\eta}{\lambda}\right)^2 = \frac{\lambda}{l} \tag{2.47}$$

2.4 Energy transfer in turbulent flows

There is a widely-accepted picture of the energy-transfer mechanism in turbulent flows, the *energy-cascade model of Kolmogorov*. According to this model, the mean flow, induced by a pressure gradient or other kind of forces, works on the large-scale motions of turbulence (the *largest eddies*), increasing the turbulent energy production. Through a variety of processes, including vortex stretching, energy is transferred to ever small scales (the *smallest eddies*), until it reaches scales that are small enough because of viscosity to dissipate the kinetic energy into the internal

energy of the fluid. At high Reynolds numbers, between the large scales at which turbulent energy is produced and the small scales at which is destroyed, lies a range of scales at which neither process is very important. This is called the *inertial subrange* [1]. The rate of energy input (per unit of mass) at the largest scales is equal to the energy throughput from the large to the small scales. This rate, in turn, is equal to the energy dissipated by the smallest scales.

As the cascade proceeds, the successive generation of smaller eddies lose information on the large-scale structure of the flow. Thus, the anisotropy of the large scales fades and the small-scales become statistically isotropic. With respect to Kolmogorov (the K41 theory), the new concept that is introduced is related to the notion of *local isotropy*, the isotropy of the small scales. Further, Kolmogorov postulated that the statistics of these isotropic scales would have universal behavior, independent from the way the flow is produced. The scales at which this approximately occurs are known as the *universal equilibrium subrange*. This is further divided into a dissipation subrange (the very smallest scales) and the inertial subrange, those scales larger than the dissipation scales (where viscosity become dominant), but smaller than the large anisotropic scales that define the flow. The width of the inertial subrange increases as the Reynolds number increases, so that the anisotropic large scales encroach less and less on the small scales.

In the inertial subrange, K41 provides predictions for the way the statistics behave of velocity differences across a separation distance r. By defining this difference:

$$\Delta u(r) = u(x_i + r) - u(x_i) \tag{2.48}$$

where x_i is a reference point, K41 predicts that its statistical average (ensemble or time average) will be only a function of ε and r itself.

In the inertial subrange, energy is cascaded to smaller scales, and the structure function $\left\langle \left[\Delta u(r)\right]^n \right\rangle$ (being n a positive integer) obeys the relation [1]:

$$\left\langle \left[\Delta u(r)\right]^n \right\rangle = C_n (\varepsilon r)^{n/3} \tag{2.49}$$

An exact relation exists for the 3rd – order structure function [2] is:

$$\left\langle \left[\Delta u(r) \right]^3 \right\rangle = -\frac{4}{5}\varepsilon r \tag{2.50}$$

Known as the Kolmogorov's "4/5" law, that shows that the energy flux from large to small scales is unidirectional on average.

For $n = 2$, the variance $\left\langle \left[\Delta u(r) \right]^2 \right\rangle$ will increase as $r^{2/3}$, or:

$$\left\langle \left[\Delta u(r) \right]^2 \right\rangle = \varepsilon^{2/3} r^{2/3} \tag{2.51}$$

giving the Kolmogorov's "2/3" law. In this case, the Fourier transform of Eq. (2.51) yields the -5/3 spectrum, that has been verified in many flows. More in particular, the distribution of energy over the scales of turbulence is usually described in terms of wavenumber, although this is a concept that is strictly applicable only to homogeneous flows. In this case, it is possible to represent the velocity field as a Fourier series. Consider the one-dimensional Fourier series:

$$u(x) = \sum \hat{u}(k) e^{ikx} \tag{2.52}$$

the energy spectrum is:

$$E(k) = \frac{1}{2} u(k) u^*(k) \tag{2.53}$$

where the asterisk denotes the complex conjugate. The energy spectrum gives the distribution of turbulent energy in terms of wavenumber or inverse length scale $\overline{u_i' u_i'} = \int E_i(k) dk$.

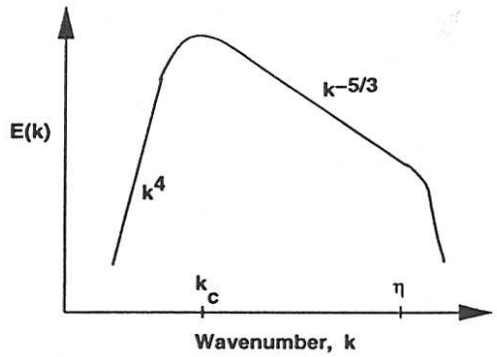

Figure 2.1 – Sketch of typical energy spectrum of turbulence.

In the inertial subrange, energy is cascaded to smaller scales. As there is no production in this range, the rate at which the energy is transferred to the smaller scales must be equal to the rate at which it is dissipated at the smallest scales.
Kolmogorov argued on dimensional grounds that the three-dimensional energy spectrum in this region must have the form:

$$E(k) = C_K \varepsilon^{2/3} k^{-5/3} \tag{2.54}$$

where $E(k)$ is the three-dimensional energy spectrum, ε is the rate of energy dissipation and C_K is the Kolmogorov constant ($1.4 \div 2.2$).

A sketch of typical energy spectrum of turbulence in logarithmic coordinates is shown in Fig. 2.1. Figure 2.2 reports experimental evidence of Eq. (2.54) in different flows [3]. With reference to Fig. 2.1, at low wavenumber the spectrum is proportional to k^4. This region is followed by a peak at wavenumber k_c. The length scale corresponding to the peak is an important characteristic length scale of turbulence and is approximately the integral scale l. The peak in the spectrum is followed by the inertial subrange, whose length depends on the type of flow and on the Reynolds number.

Finally, there is a sharp decrease of the energy spectrum near the Kolmogorov length scale η. Note that, being the integral scale the length scale corresponding to the peak

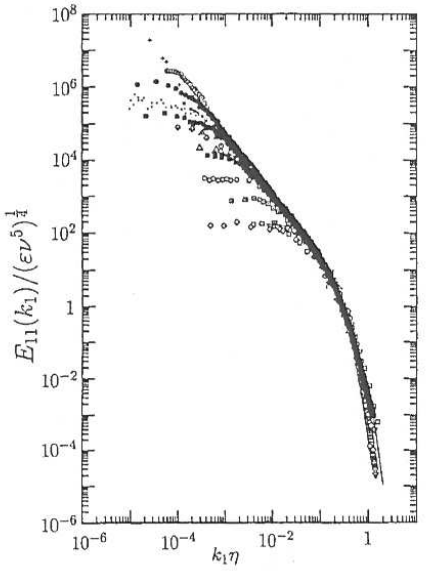

Figure 2.2 – Experimental evidence of Eq. (2.70) in different flows [3].

of the energy spectrum, it is the length scale characteristic of the energy-containing scales. Thus, it can also be defined as:

$$l = \frac{\int_0^\infty E(k)\,dk}{\int_0^\infty \frac{E(k)}{k}\,dk} \tag{2.55}$$

The following can be noted. In 1941, Kolmogorov suggested that, through the cascade process, the energy would lose detailed information about the mechanism of energy production. If the number of steps of the cascade was sufficiently great, it could be presumed that all information would be lost.

The small scales would know only how much energy they received (ε, in equilibrium). They might be expected to be isotropic, having lost all information about the anisotropy of the energy-containing scales. This state of isotropy would actually exist only at infinite Reynolds number (infinitely many steps in the cascade). At any finite Reynolds number, the small scales would be expected to be less anisotropic than the energy-containing scales, but still somewhat anisotropic. It is

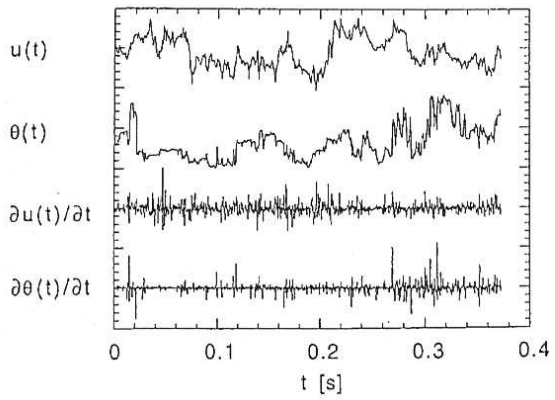

Figure 2.3 – Time series of turbulent quantities [6].

also worth mentioning the fact that there is a permanent anisotropy even in the smallest scales of the velocity spectrum of a shear flow [4]. The anisotropy exists because of the energy from the mean flow is fed into one component and must be redistributed to the other two.

However, while the amount of anisotropy remains fixed as the Reynolds number increases, it steadily decreases when consider as a proportion of the total mean square velocity gradient. In this sense, it can be said that

the velocity spectrum becomes increasingly isotropic in the small scales as the Reynolds number increases.

The situation in real turbulent flows is more complex than the aforementioned scenario. In three-dimensional turbulence, a significant amount of energy is transferred in both directions of the spectrum (from large to small scales and from small to large ones) and the most important processes of transferring energy to small scales are vortex stretching and the shearing of small vortices by larger ones. The energy transfer toward the small scales has been found by [5] to be about twice the reverse flow in homogeneous isotropic turbulence. There is some discussion about this reverse flow of energy, called *backscatter*. This term refers to the transfer of energy in the spectrum in the direction from small to large scales.

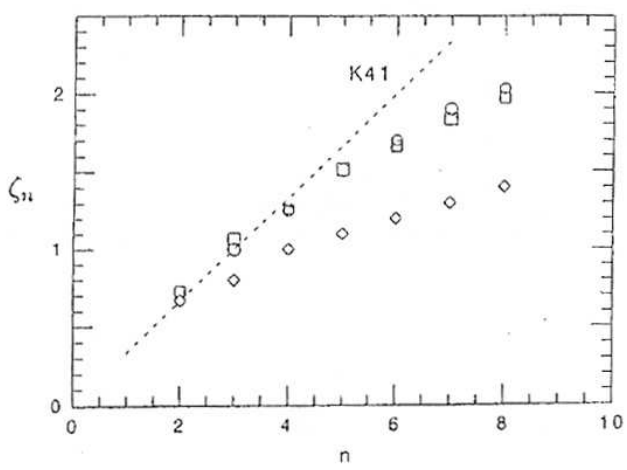

Figure 2.4 – Inertial subrange scaling exponents ζ_n for velocity and scalar [8].

There is no question that, taking averages over long times or large regions of space, the energy transfer in the spectrum of three-dimensional turbulence is from large scales to small. In two-dimensional turbulence, the energy cascade goes in the opposite direction, since there is no vortex stretching, vortices coalesce to form larger vortices and this is the mechanism of energy transfer. It is possible in three-dimensional turbulence if one considers short time averages or averages over small regions of physical space, to have vortex coalescence and hence, locally and temporarily, energy transfer in the reverse direction. Some initial instabilities are of two-dimensional nature and for some time the energy transfer will be toward the reverse direction, until the flow will become fully three-dimensional. On the other hand, many flows of technological importance that are young (not fully developed) have highly anisotropic remnants of initial instabilities and turbulent structures that are highly anisotropic may have for limited times and over limited regions energy transfer in the reverse direction. Other problems exist. The dissipation rate has been found to vary both spatially and temporally within the flow. Because the ultimate fate of the turbulence energy is at the small scales, the dissipation rate is related to the sharp gradients of the velocity that occur there. Thus, the dissipation is a function of various combinations of the velocity derivatives.

If one observes a time series of turbulent quantities (Fig. 2.3), it can be noted that the velocity fluctuations themselves are close to Gaussian, while those of their

temporal derivatives are strongly non-Gaussian. This result implies that the dissipation in turbulent flows is spotty, or intermittent (see also [7]). There are instants in which it is large, followed by quieter moments (note that in the framework of K41 the governing parameter is the average dissipation rate).

The intermittent character of turbulence also implies evolutionary processes of complex structures in the flow. If high-order structure functions are constructed from a time series of turbulent velocity fluctuations, a linear trend in the scaling exponent (as predicted by the K41) is not found. The trend is nonlinear with increasing departure from K41 with increasing n (Fig. 2.4). The departure from linearity at higher orders is caused by the intermittent nature of turbulence. It appears that the two quantities ε and r are not sufficient to determine the form of the structure function. Thus, a new parameter must enter. Usually the integral length scale is introduced, giving:

$$\left\langle \left[\Delta u(r) \right]^n \right\rangle = (\varepsilon r)^{n/3} \left(\frac{l}{r} \right)^{-\delta_n} = r^{\zeta_n} \tag{2.56}$$

$$\zeta_n = \frac{n}{3} - \delta_n \tag{2.57}$$

where ζ_n is the scaling exponent, and δ_n is the departure from K41 (for $\delta_n = 0$, Eq. (2.57) reduces to K41, Eq. (2.49)), where [9] assumed that the dissipation rate is log-normally distributed (the Kolmogorov-Obukhov K62 theory, or Refined Similarity Hypothesis, RSH). Note that the cascade concept and the postulate of local isotropy are tightly bound. If local isotropy does not hold, the implication is that the small scales are not universal and that there is direct interaction between the large and the small scales.

Chapter 3

Literature Review

3.1 Introduction

The Direct Numerical Simulation of turbulence (DNS) consists of solving numerically the Navier-Stokes equations with physically-consistent accuracy in space and time. If the mesh is fine enough, the time step is short enough and the numerical scheme is designed to minimize the dispersion and dissipation errors, one obtains an accurate three-dimensional time-dependent solution of the governing equations, in which the only errors are those introduced by the residual approximations incorporated in the numerical scheme and in the number-representation technology of the computing machine. For this reason, there is a fundamental difference between a solution of the Navier-Stokes equations as an exercise of numerical mathematics – whatever complex it may result – and a solution of the same equations with the aim to obtain a precise correlation of the results with the turbulence physics; in the latter case, the accuracy of the calculations has to be deeply monitored. It results that the major difficulty in performing turbulence

calculations at values of the Reynolds number approaching those of practical engineering interest lies in the remarkable amount of computational resources required (the degree of freedom involved with the solution of the Navier-Stokes equations roughly increase $\approx Re^{9/4}$) and the consequence of this fact has been – for a rather long period of time – that only simple flows have been analyzed with the DNS.

The advent of high performance computing systems has completely changed this scenario, opening the field of the High Performance Direct Numerical Simulation of turbulence (HPDNS). Generally speaking, the procedure for the execution of high performance Navier-Stokes calculations involves a number of steps, such as:

- development and testing of a new numerical method or application of an existing numerical technique for the solution of the Navier-Stokes equations on a given spatial and temporal computational domain;
- implementation of the Navier-Stokes solver onto a given high performance computing architecture;
- execution of the numerical simulations with appropriate resolutions in space and time;
- use of the resulting numerical databases for the calculation of appropriate turbulent-flow quantities.

There exist works in which DNS results are presented actually giving the appropriate emphasis to their physical realism, but scarce relevance to the computational technologies used for their attainment. There are works in which the implementation of Navier-Stokes solvers onto high-performance computers is extensively described, but afterwards no results of physical relevance are presented. There are works in which DNS databases are presented and used for the eduction of turbulent-flow structures with different techniques, without specifying in detail how the latter databases have been obtained. Finally, there are a few works in which the development of a procedure including the four above steps can be on the contrary clearly recognized.

In this chapter, an attempt is made to present the aforementioned body of literature in the most-possibly systematic manner. Additional issues that are discussed are related to the twofold relationship that establishes between

computational scientists and high performance computers, in the following sense. On one hand the computational fluid dynamics develops a Navier-Stokes solver and afterwards he needs to implement the latter solver onto a given computing architecture. On the other hand the computational scientists has also to be able to understand how the use of a newly-introduced computing architecture may bring advantages with regard to a more efficient use of a given computational technique, or even to the adoption of a completely different numerical method.

This chapter is organized as follows: Section 3.2 describes the main contributes about DNS and numerical methodologies adopted for numerical simulations; Section 3.3 shows the description of turbulence in a plane channel and the major works about it; then, Section 3.4 shows the main literature contributes about flow structures and theory about their identifications; finally, Section 3.5 describes the conjunction between DNS and high performance computing, in order to study wall-bounded flows at high-Reynolds numbers.

3.2 Early times in DNS

While the CDC 6600 and 7600 (rolled out in 1963 and in 1969, respectively) have been the first (scalar) computing systems that have been later denoted as "supercomputers", probably (see also the 1998 review of Moin and Mahesh [10]) the first works to which the DNS acronym has been then associated, are those of S.A. Orszag and co-workers, and in particular that of Orszag and Patterson [11] where isotropic turbulence was calculated onto a 32^3 grid-point computational volume at $\mathrm{Re}_\lambda = 35$, using a CDC-7600 computer. Rogallo [12] studied the effects of mean shear, irrotational strain and rotation in homogeneous turbulence by transforming the governing equations using an extension of the Orszag-Patterson algorithm (see also the 1984 review of Rogallo and Moin [13]). The numerical isotropic turbulence was also investigated in terms of higher-order-derivative correlations by Kerr [14] on a Cray computer, on up to a 128^3 grid-point volume and up to ($\mathrm{Re}_\lambda = 82.9$). Later, the

Table 3.1 – Synthetic prospectus of early times DNS of shear-flow turbulence

Authors	*Re*	*Type of Flow*	*Computing System(s)*
Moser and Moin [16]	$\mathrm{Re}_{\tau H} = 168$	Curved channel	Cray X-MP
Kim et al. [17]	$\mathrm{Re}_{\tau H} = 180$	Plane channel	Cray X-MP
Spalart [18]	$\mathrm{Re}_{\theta} = 225 \div 1410$	Flat plate	-
Gavrilakis [19]	$\mathrm{Re}_{\tau H} = 75$	Square duct	Cray 2
Huser and Biringen [20]	$\mathrm{Re}_{\tau H} = 150$	Square duct	-
Eggels et al. [21]	$\mathrm{Re}_{\tau H} = 90$	Circular pipe	Cray X-MP

structure of intense vorticity in isotropic turbulence has been studied by Jiménez et al. [15]. Right after the appearance of [11], [12] and [14], a number of pioneering works on wall turbulence started to appear, that, in practice, qualified as milestone works with regard to both the type of flow considered and the numerical methods used. Moser and Moin [16] studied the flow of an incompressible fluid in a curved channel. They used a numerical method based on Fourier expansions in the $\theta -$ and $z -$ directions, and Chebychev-polynomials expansion in the $r -$ direction (normal to the walls). This is probably the first case in which the fully spectral Fourier-Chebychev method is used that has characterized so many subsequent works of P. Moin, J. Kim, and co-workers. The Reynolds number was $\mathrm{Re}_{\tau} = 168$, 128 Fourier modes were used along the periodic directions, while 64 Chebychev modes were used along the $z -$ direction. Kim et al. [17] studied the flow in the plane channel at $\mathrm{Re}_{\tau} = 180$ using a 4×10^6 grid-point volume, on a Cray X-MP machine. With 40s for each time step, about 250 CPU hours were necessary to run 10 non-dimensional times. Spalart [18] computed the turbulent flat-plate boundary layer with zero pressure gradient (ZPGFPBL) up to $\mathrm{Re}_{\theta} = 1410$ and up to about 10^7 grid points. The flow in a square duct has been investigated by Gavrilakis [19] and Huser and Biringen [20]. Respectively, fully finite difference and mixed spectral-high-order finite difference computational algorithms were used in these cases. The flow in a circular pipe has been explored by Eggels et al. [21] where the fully finite difference technique originally introduced by Schumann [22] was used. In Table 3.1, a synthetic prospectus of the aforementioned works is reported (in order to compare internal-

flow cases with different cross-sectional geometries, the Reynolds number $\mathrm{Re}_{\tau H} = u_\tau R_H / \nu$ is used).

3.3 Plane channel

In this Section, DNS works in the field of shear-flow turbulence are considered, with reference to the flow in a plane channel. Overall, *x* is the streamwise direction, *y* is the wall-normal direction, *z* is the spanwise direction, and the flow is considered homogeneous in the streamwise and spanwise directions. Overall, the aim of the simulations is that of calculating a given number of time steps of the statistically-steady turbulent flow, to build up a DNS database. Accurate DNS calculations of the turbulent flow in a plane channel have been carried out by Kim et al. [17], Lyons et al. [23], Kasagi et al. [24], Antonia et al. [25], Rutledge and Sleicher [26], Moser et al. [27], Abe et al. [28], Iwamoto et al. [29], Del Alamo and Jiménez [30], Del Alamo et al. [31], Tanahashi et al. [32], Iwamoto et al. [33], Hoyas and Jiménez [34], Hu et al. [35], Alfonsi and Primavera [36], at different values of the Reynolds number. In these works the system of the governing equations is mainly solved in the framework of the fractional-step method, in conjunction with Runge-Kutta algorithms for time marching. More in particular, in Refs. [17, 23-27, 29-31, 33, 35] the unsteady three-dimensional Navier-Stokes equations in rotational form are integrated in space by using either the fully spectral Fourier-Chebychev numerical technique originally introduced by Kim and Moin [37] or minor variants of the latter, or also fully spectral techniques introduced by other authors.

In Ref. [28] the Navier-Stokes equations in non-conservative form are integrated by means of a fully finite- difference algorithm, with a grid-stretching law along the direction orthogonal to the solid walls. In Refs. [32] and [34] mixed spectral-high-order finite difference numerical schemes are used.

In Ref. [36] the Navier-Stokes equations in conservative form are integrated by means of the mixed Fourier-finite difference method originally introduced by Alfonsi et al. [38], where a grid-stretching law of hyperbolic-tangent type is inserted along the direction orthogonal to the walls.

As concerns boundary conditions, periodic conditions are imposed along the streamwise and spanwise directions, and no-slip conditions are enforced at the solid walls (for an extensive review about boundary conditions in incompressible Navier-Stokes problems, one can refer to the 2006 review of Rempfer [39]). In Tab. 3.2, some characteristic quantities of simulations of Refs. [17, 23-36] are reported, where the values of the mean-flow quantities can be verified using the experimental correlations suggested by Dean [40].

The numerical simulation of wall-bounded turbulent shear flows requires a remarkably large number of grid points in all spatial directions, where the grid spacing determines the scales that are represented. In the wall-normal direction it is possible to distribute the grid points with variable spacing, so that the viscous sublayer and the buffer layer result satisfactorily resolved. As the Reynolds number increases, more points are required. In the spanwise direction, a reasonable criterion to be followed is that of resolving the streaks in the vicinity of the wall. A number of works (see also the 1991 review of Robinson [41], and the 2001 review work of Panton [42]) indicate that the mean streak spacing is about 100 wall units, i.e. $\lambda_m^+ = \lambda_m u_\tau / \nu \cong 100$, where the mean width of the high-speed wall-layer structures is $\cong 20\nu/u_\tau \div 40\nu/u_\tau$. Thus, in order to resolve $20\nu/u_\tau$ and assuming 4 grid points for each eddy, the grid spacing along the spanwise direction should be $\Delta z^+ \cong 5$. As concerns the streamwise direction, estimates based on the streamwise extent of the wall-layer structures indicate that the required grid resolution is about half than that adopted for the spanwise direction. As concerns the smallest-scale resolution requirements, the criterion of resolving the Kolmogorov space and time scales (or appropriate multiple of the latter) is extensively followed. In Tab. 3.3, a number of characteristic computational parameters of simulations of Refs. [17, 23-36] are reported, while in Table (3.4) the data related to the resolution of the calculations are shown, in conjunction with the computing system used (if it is given). From the data reported in Tables (3.2), (3.3) and (3.4) it can be verified that simulations of Refs. [17, 23-36] have been globally executed in a framework of high temporal resolution (the resolutions in time obey to the rather stringent Courant-number requirements associated to the use of a given numerical technique, usually more demanding with

Table 3.2 – Characteristic quantities of simulations of Refs. [17, 23-36]

References	Re_τ	$\overline{u}_c/\overline{u}_b$	$10^3 C_f$	$10^{-7}\varepsilon$	η^+	τ_η^+
[35]	90	1.17	10.32	1.0	1.59	3.54
[29]	110	1.16	9.73	1.9	1.66	3.76
[35]	130	1.16	9.27	3.2	1.72	3.96
[23][24][29]	150	1.16	8.89	5.2	1.77	3.15
[17][25][26][27][28][30][32][35][36]	180	1.16	8.44	9.1	1.84	3.40
[29]	300	1.15	7.29	45.3	3.06	4.23
[35]	360	1.14	6.92	80.4	3.14	4.57
[27][28]	395	1.14	6.74	107.6	3.18	4.76
[25][29][32]	400	1.14	6.71	111.9	3.19	4.78
[30][31]	550	1.14	6.13	303.8	3.34	5.49
[27]	590	1.14	6.01	378.6	3.38	5.66
[28]	640	1.14	5.87	488.4	3.42	5.86
[29]	650	1.14	5.85	513.7	3.43	5.90
[35]	720	1.13	5.68	706.2	3.48	6.17
[32]	800	1.13	5.52	983.0	3.54	6.46
[31]	950	1.13	5.25	1680.8	3.64	6.96
[35]	1440	1.12	4.68	6163.9	3.89	8.35
[31]	1900	1.12	4.33	14633.6	3.07	9.44
[34]	2003	1.12	4.26	17251.7	3.11	9.66
[33]	2320	1.12	4.09	27270.4	3.21	10.31

respect to the Kolmogorov-time-scale requirement) and high spatial resolution near the walls, in an attempt to accurately resolve the wall turbulent-flow structures. In a number of works, attempts have been made to devise reliable but less stringent criteria - with respect to those of the Kolmogorov scales - for the accuracy of DNS calculations. Grötzbach [43] devised a number of rules for DNS accuracy, such as:

- to select a domain of size sufficiently large to record all the relevant large-scale structures;
- to select a normal-to-the-wall grid-width distribution in such a way as to resolve the steep gradients of the velocity field near the wall (i.e. to have at least 3 grid points in the viscous sublayer);
- to select a computational grid such as the mean-grid width result smaller than the smallest relevant turbulent elements ($\Delta \leq \pi\eta$);
- to select $\Delta t \leq \tau_\eta$.

Other viewpoints exist. According to Ref. [10], the smallest resolved length scale has to be $O(\eta)$ not exactly equal to η, in the sense that reliable first- and second-order statistics are obtained whenever the resolution is fine enough to accurately

Table 3.3 – Computational parameters of simulations of Refs. [17, 23-36]
[(*) simulation D, (**) series 1, (***) series 2]

Ref.	Re_τ	L_x	L_y	L_z	L_x^+	L_y^+	L_z^+	N_x	N_y	N_z
[35]	90	48h	2h	24h	4320	180	2160	256	61	256
[29]	110	5πh	2h	2πh	1728	220	691	48	65	48
[35]	130	24h	2h	12h	3120	260	1560	196	81	196
[23]*	150	13.7h	2h	6.3h	1900	300	950	85	65	85
[24]	150	5πh	2h	2πh	2356	300	942	128	96	128
[29]	150	3.5πh	2h	πh	1178	300	471	64	97	64
[17]	180	4πh	2h	2πh	2262	360	1131	192	129	160
[25]	180	4.5πh	2h	2h	810	360	360	128	129	128
[26]	180	4πh	2h	4πh/3	2262	360	754	144	65	144
[27]	180	4πh	2h	4πh/3	2262	360	754	128	129	128
[28]	180	13.8h	2h	6.4h	2304	360	1152	256	128	256
[30]	180	12πh	2h	4πh	6786	360	2262	-	-	-
[32]	180	4πh	2h	2πh	2262	360	1131	192	193	160
[35]	180	24h	2h	12h	4320	360	2160	256	121	256
[36]	180	2πh	2h	πh	1131	360	565	96	129	64
[29]	300	3.5πh	2h	πh	2356	600	942	128	193	128
[35]	360	12h	2h	6h	4320	720	2160	256	161	256
[27]	395	2πh	2h	πh	2482	790	1241	256	193	192
[28]	395	6.4h	2h	3.2h	2528	790	1264	256	192	256
[25]	400	4.5h	2h	2h	1792	800	768	256	193	192
[29]	400	3.5πh	2h	πh	3142	800	1257	192	257	192
[32]	400	2πh	2h	πh	2513	800	1257	256	385	192
[30]	550	8πh	2h	4πh	13823	1100	6912	-	-	-
[31]**	550	8πh	2h	4πh	13823	1100	6912	1536	257	1536
[31]***	550	πh	2h	πh/2	1728	1100	864	192	257	192
[27]	590	2πh	2h	πh	3707	1180	1854	384	257	384
[28]	640	6.4h	2h	2h	4096	1280	1280	512	256	256
[29]	650	3.5πh	2h	πh	5105	1300	2042	288	257	384
[35]	720	12h	2h	6h	8640	1440	4320	512	321	512
[32]	800	2πh	2h	πh	5027	1600	2513	512	769	384
[31]**	950	8πh	2h	3πh	23876	1900	8954	3072	385	2304
[31]***	950	πh	2h	πh/2	2985	1900	1492	384	385	384
[35]	1440	12h	2h	6h	17280	2880	8640	1024	481	1024
[31]***	1900	πh	2h	πh/2	5969	3800	2985	768	769	768
[34]	2003	8πh	2h	3πh	50341	4006	18878	6144	633	4608
[33]	2320	6πh	2h	2πh	43731	4640	14577	2304	1025	2048

capture most of the dissipation. Thus, the smallest length scale that must be resolved depends on the energy spectrum, being the latter typically larger than η. Moser and Moin [16] have shown that most of the dissipation in a curved channel takes place at scales larger than 15η. In a DNS work the number of grid points, their distribution in space and the time step of the calculations are decided before running the calculation, on the basis of a preliminary evaluation of the Kolmogorov microscales. A widely-

Table 3.4 – Resolution parameters of simulations of Refs. [17, 23-36]
[(*) simulation D, (**) series 1, (***) series 2]

Ref.	Re_τ	Δx^+	Δy_c^+	Δy_w^+	Δz^+	t_{DB}	*Computing System(s)*
[35]	90	16.8	4.7	0.03	8.4	-	-
[29]	110	36.0	5.4	-	14.4	-	-
[35]	130	16.8	-	-	8.4	-	-
[23]*	150	23.4	7.4	0.18	11.2	-	Cray 2
[24]	150	18.4	4.9	0.08	7.4	$2100\nu/u_\tau^2$	Hitachi S-820/80
[29]	150	18.4	4.9	-	7.4	-	-
[17]	180	11.8	4.4	0.05	7.1	$10tu_\tau/h$	Cray X-MP
[25]	180	11.0	4.4	0.05	4.0	-	-
[26]	180	15.7	8.7	0.21	5.23	-	Cray X-MP
[27]	180	17.7	4.4	0.05	5.9	-	-
[28]	180	9.0	5.9	0.20	4.5	$4320\nu/u_\tau^2$	Fujitsu VPP-500
[30]	180	8.9	6.1	-	4.5	$22t\overline{u_b}/L_x$	-
[32]	180	11.8	4.6	0.43	7.1	-	-
[35]	180	16.8	-	-	8.4	-	-
[36]	180	11.8	4.4	0.87	8.8	$10tu_\tau/h$	HP V-2500
[29]	300	18.4	4.9	-	7.4	-	-
[35]	360	16.8	-	-	8.4	-	-
[27]	395	9.7	6.5	0.03	6.5	-	-
[28]	395	9.9	9.6	0.20	4.9	$15800\nu/u_\tau^2$	Fujitsu VPP-500
[25]	400	7.0	5.5	0.05	4.0	-	-
[29]	400	16.4	4.9	-	6.5	-	-
[32]	400	9.8	5.2	0.48	6.5	-	-
[30]	550	8.9	6.7	-	4.5	$10tu_\tau/h$	-
[31]**	550	8.9	6.7	-	4.5	$10tu_\tau/h$	-
[31]***	550	8.9	6.7	-	4.5	$77tu_\tau/h$	-
[27]	590	9.7	7.2	0.04	4.8	-	-
[28]	640	8.0	8.0	0.15	5.0	$24800\nu/u_\tau^2$	Fujitsu VPP-500
[29]	650	17.7	8.0	-	5.3	-	-
[35]	720	16.8	-	-	8.4	-	-
[32]	800	9.8	5.2	0.48	6.5	-	-
[31]**	950	7.6	7.6	-	3.8	$9.2t\overline{u}_b/L_x$	-
[31]***	950	7.8	7.8	-	3.9	$27t\overline{u}_b/L_x$	-
[35]	1440	16.8	9.4	0.12	8.4	-	-
[31]***	1900	7.8	7.8	-	3.9	$22t\overline{u}_b/L_x$	-
[34]	2003	8.2	8.9	-	4.1	$10.3tu_\tau/h$	Marenostrum
[33]	2320	19.0	-	-	7.1	-	Earth Simulator

used procedure for this evaluation is based on the estimation of the rate of dissipation per unit mass ε, as obtained from some mean-flow quantities.

This method has been introduced by Bakewell and Lumley [44] in an experimental work dealing with the pipe-flow case. In the case of the plane channel, one has:

$$\varepsilon \cong \frac{2L_x L_z \tau_w \overline{u}_b}{\rho L_x L_y L_z} = \frac{2L_x L_z \tau_w \overline{u}_b}{2\rho h L_x L_z} = u_\tau^2 \overline{u}_b \tag{3.1}$$

Other relevant issues are related to the verification of the adequacy of the computing-domain dimensions and grid resolution.

The size of the computational domain is adequate if it is large enough to capture all the relevant large-scale turbulent structures. Thus, the velocity fluctuations at streamwise and spanwise separation distances of half the domain dimensions have to be uncorrelated and this circumstance can be verified by monitoring the two-point correlation coefficients of the fluctuating velocities (further observations about the adequacy of the computing domain in DNS can be found in Fishpool et al. [45]).

As concerns grid resolution, one-dimensional energy spectra of the fluctuating velocities are usually monitored. The grid resolution is adequate if the energy density associated with the high wavenumbers results several orders of magnitude lower than the energy density corresponding to low wavenumbers.

An attempt in reducing the computational resources required for DNS calculations in turbulent channel flow has been performed by Jiménez and Moin [46], in the framework of the minimal channel-flow domain.

Channel-flow calculations at $\mathrm{Re}_c = 2000$, 3000 and 5000 were performed on domains significantly smaller than a large turbulent channel, and statistics of mean velocity and turbulence intensity were compared with those obtained in the full channel. Good agreement in the near-wall region was observed for domain sizes greater that 100 wall units in the spanwise direction, and 250-350 wall units in the streamwise direction. Thus, the smallest domain that sustained turbulence was termed *minimal channel*, representing a single unit of near-wall turbulence that allows the isolation of key turbulence structures and the study of their dynamics (additional observations about resolution requirements in DNS can be found in the 2001 review of Friedrich et al. [47]).

3.4 Turbulent-flow structures

A critical aspect in DNS is the use of a computed numerical database for the improvement of our knowledge of turbulence physics. Also in this phase, high performance computing techniques may be highly helpful (see, among others, Alfonsi and Primavera [48]).

3.4.1 Early times in flow structures

A useful notion for the scientific understanding of turbulence physics is that of turbulent-flow structure. Historically, one of the first results in studying the structure of the boundary layer is due to Kline et al. [49] who showed that, very near to the wall, the flow organizes in alternating arrays of unsteady high- and low-speed regions aligned in the streamwise direction, called *streaks*. Thereafter, a considerable amount of work has been accomplished. Techniques for the detection of turbulent events have been introduced, and *Conditional Sampling and Averaging* (see also the 1981 review of Antonia [50]), *Quadrant Analysis* (Willmarth and Lu [51]), *Variable Interval Time Averaging* (VITA, Blackwelder and Kaplan [52]), and *Variable Interval Space Averaging* (VISA, Johansson et al. [53]) are examples of such techniques.

In the *Quadrant Analysis*, the local flow behaviour is divided into quadrants, depending on the sign of the streamwise u' and normal v' velocity fluctuations. Four quadrants are identified and, among them, the second-quadrant Q_2 event ($u'<0, v'>0$, low-speed fluid moving away from the wall) is identified as an *ejection*, while the fourth-quadrant Q_4 event ($u'>0, v'<0$ high-speed fluid moving toward the wall) is identified as a *sweep*. One of the first contributions to the issue of the presence of vortices in the boundary layer is due to Theodorsen [54], who introduced the *hairpin vortex model*. Robinson [41] confirmed the existence of arch vortices and quasi-streamwise vortices, on the basis of DNS results. The composition of a quasi-streamwise vortex with an arch vortex may result in a hairpin vortex, but this conclusion may strongly depends on the particular technique used for vortex detection. The process of evolution of a hairpin vortex involves the development of *vortex legs*. The leg of a vortex, considered in isolation, may appear as a quasi-streamwise vortex. The *vortex head* instead, rises through the flow field, and the

vorticity in the vortex head diminishes (Head and Bandyopadhyay [55], Smith et al. [56]). Conceptual models of boundary-layer turbulence based on vortex dynamics have been provided, among others, by Willmarth and Tu [57] (based on vorticity lines), Offen and Kline [58] (in terms of lifted and stretched horseshoe vortical structures), Praturi and Brodkey [59] (based on the relation between inner and outer region), and Thomas and Bull [60]. Acarlar and Smith [61] described the dynamics of hairpin vortices in the boundary layer in connection with low-speed streaks, shear layers and other phenomena. Robinson [41] devised a model in which quasi-streamwise vortices dominate the buffer region, arch vortices are mainly present in the wake region, while in the overlap layer both structures exist, often as elements of the same vortical structure. An extensive review about boundary-layer turbulence has been performed in 2006 by Alfonsi [62].

3.4.2 Vortical structures

With the advent of DNS, turbulent-flow databases of numerical nature became available, incorporating the possibility of implementing – on huge amounts of data – mathematically-based definitions of vortical structures, for their eduction. Mathematically-founded methods that can be successfully used for the identification of vortical structures of different kind in a turbulent flow have been introduced by Perry and Chong [63] (based on the complex eigenvalues of the velocity-gradient tensor), Hunt et al. [64] (based on the second invariant of the velocity-gradient tensor), Jeong and Hussain [65] (based on the analysis of the Hessian of the pressure), and Zhou et al. [66] (based on the imaginary part of the complex eigenvalue pair of the velocity-gradient tensor). A comparison of the effectiveness of the four aforementioned vortex-eduction criteria can be found, among others, in Alfonsi and Primavera [67n]. Perry and Chong [63] proposed the method of identifying vortices by means of isosurfaces of positive small values of the discriminant D of the characteristic equation of the velocity-gradient tensor, where it has complex eigenvalues (the *D criterion*). Hunt et al. [64n] devised another criterion, in defining an eddy zone as a region characterized by positive values of the second invariant Q of the velocity-gradient tensor (the *Q criterion*). In Wu and Moin [68], [69] a clear evidence can be found of the presence of hairpin vortices in the flat-

plate boundary layer, as extracted from the numerically-simulated velocity field by using the Q criterion. It is shown that the instantaneous flow fields in both transitional and turbulent regions result vividly populated by hairpin vortices. Jeong and Hussain [65] proposed a definition of a vortex by considering the problem of the pressure minimum (the λ_2 *criterion*). According to this method, a vortex is defined as a connected region of the flow with the requirement that the intermediate eigenvalue of B_{ij} be negative $\lambda_2 < 0$. The λ_2 *criterion* represented the basis for a remarkable amount of work, as performed by F. Hussain and co-workers (see Schoppa and Hussain [70] and references therein). Zhou et al. [66] adopted the criterion of identifying vortices by visualizing isosurfaces of appropriate values of the imaginary part of the complex eigenvalue pair of the velocity-gradient tensor (the λ_{ci} or *swirling strength criterion*). The method is frame independent and due to the fact that the eigenvalue is complex only in regions of local circular or spiralling streamlines, it automatically eliminates regions having vorticity but no local spiralling motion. Chakraborty et al. [71] proposed an enhanced criterion, the so-called enhanced swirling strength criterion.

3.5 High performance Navier-Stokes solvers

In this section, an overview of a number of works in which computational codes for the numerical integration of the Navier-Stokes equations are developed and implemented on high performance computers of different kinds is given (Tab. (3.5)). Among the first, Jespersen and Levit [72] tested the performance of a Navier-Stokes solver on a 32768-processor Connection Machine CM-2, against that obtained on a Cray X-MP and Cray 2 computers. Fischer et al. [73] presented a high-efficiency medium-grained parallel spectral-element method for the numerical solution of the unsteady incompressible Navier-Stokes equations, mainly evaluating the optimality of the algorithm-architecture coupling. Two MIMD Intel message-passing vector-hypercube computers (a iPSC/1-VX/d4 and a iPSC/2-VX/d4) and a Cray X-MP machine were used. The first machine was a Intel 286-based system with store-and-forward message passing. The second was a Intel 386-based system with pipeline communication routing. Different flow cases were tested, namely a free-surface

Table 3.5 – Synthetic prospectus of high performance Navier-Stokes solvers.

Author(s)	*Computing System(s)*
Jespersen and Levit [72]	Connection Machine CM-2, Cray X-MP, Cray 2
Fischer et al. [73]	Intel iPSC/1, Intel iPSC/2, Cray X-MP
Pelz [74]	NCUBE/1
Jackson et al. [75]	Intel iPSC/860
Chen and Shan [76]	Connection Machine CM-2
Johan et al. [77]	Connection Machine CM-2, CM-200
Naik et al. [78]	IBM Victor
Fatoohi [79]	Connection Machine CM-2, Intel iPSC/860, Cray Y.MP
Basu [80]	Intel iPSC/860
Briscolini [81]	IBM SP-1, IBM SP-2 thin, IBM SP-2 wide
Floros and Reeve [82]	Inmos T-800, Intel iPSC/860, Meiko CS-2
Prestin and Shtilman [83]	Meiko MK-096
Crawford et al. [84]	IBM SP-2, SGI Power Challenge XL, Cray C-90
Garg et al. [85]	Intel Paragon, Intel iPSC/860, Cray Y-MP
Wasfy et al. [86]	SGI Onyx
Garbey and Vassilevski [87]	Cray T-3E, DEC Alpha cluster
Gropp et al. [88]	ASCI White, ASCI Blue, ASCI Red
Kumar et al. [89]	Fujitsu VPP-700
Hoeflinger et al. [90]	SGI Origin 2000
Dong and Karniadakis [91]	SGI Origin 2000, IBM SP-3, et al.
Itakura et al. [92]	Earth Simulator
Xu [95]	Blue Gene/L
Behara and Mittal [96]	Intel Xeon cluster
Grinberg et al. [97]	IBM Blue Gene, IBM Power 4+, Cray XT-3

"levelling" problem, the external startup flow past a circular cylinder, and the startup flow past a large "roughness" element in a channel. Overall, the efficiency of the parallel calculations reached the level of about 75%.

Pelz [74] tested parallel Fourier pseudospectral algorithms for the solution of the unsteady incompressible Navier-Stokes equations, were the major operation requiring parallelization was the multidimensional FFT. Tests were performed on a 1024-node NCUBE/1 hypercube computer, reporting efficiencies of about 83% in a three-dimensional problem with mesh size of 1283 grid points. Jackson et al. [75] presented a detailed implementation of a parallel pseudospectral code for the Navier-Stokes equations, directed toward the execution of direct numerical simulations of homogeneous turbulence. They used a 32-node Intel iPSC/860 hypercube machine,

reporting better performance results with respect to those obtained on a Cray Y-MP. Chen and Shan [76n] presented spectral calculations on a Connection Machine CM-2 performed with a parallel algorithm for the three-dimensional Navier-Stokes equations directed toward the execution of direct numerical simulations of homogeneous isotropic turbulence. They implemented a 5123 (up to) mesh resolution with periodic boundaries, and report a computational speed 30% faster with respect to correspondent simulations executed on a quad-processor Cray-2 vector machine. Johan et al. [77] presented a finite-element method for computational fluid dynamics implemented on Connection Machine systems CM-2 and CM-200. An implicit iterative solution strategy was implemented, and parallel data structures were built on both nodal and elemental sets to achieve maximum parallelization. The cases tested include the flow around a blunt body and the flow around a small jet plane at both negligible angle of attack in a crosswind. Performance comparisons were also provided with respect to the use of vector-computing machines. Naik et al. [78] considered issues related to the parallelization of implicit finite-difference techniques for the solution of Euler and Navier-Stokes equations, requiring the solution of large linear systems in the form of block tri-diagonal and/or scalar penta-diagonal matrices. Various partitioning and scheduling strategies were described, directed to the alleviation of the effects of global-data dependencies. Analyses of computations, communications and memory requirements were presented. The performance of the methods was verified on the IBM message-passing architecture Victor. Fatoohi [79] presented the results of the parallelization procedure of a three-dimensional Navier-Stokes solver on three different machines, a Connection Machine CM-2, a Intel iPSC/860 and a Cray Y-MP. The solver was based on the Lower-Upper Symmetric-Gauss-Seidel implicit scheme for the formulation of the incompressible Navier-Stokes equations. The three computers were fairly different. The CM-2 was massively-parallel SIMD machine (clock rate of 7 MHz) that included 32K 1-bit serial processors, 1024 64-bit Weitek floating point, 4 GB of memory, and two front-end machines, a Sun 4/490 and a Vax 6230. The iPSC/860 is was moderately-parallel MIMD machine that included 128 nodes interconnected by a seven-dimensional hypercube network, and a front-end machine. Each node had a 64-bit i860 microprocessor, 8 Mbyte of memory and a direct

connect module, responsible for communication between nodes. The Cray Y-MP was a shared-memory MIMD machine with 8 processors, 128 Mwords of main memory, and a 6-ns clock cycle (the peak performance was 3.67 GFlops). The author reported that reasonable performances could be achieved on all the computers tested, resulting though the Cray Y-MP the best performer, as related to the algorithm at hand. A spectral technique for the Navier-Stokes equations was parallelized by Basu [80], on a Intel i/860-based three-processor computer. Briscolini [81] reported on three parallel message-passing implementations of a three-dimensional pseudospectral Navier-Stokes solver for homogeneous turbulence, on three different IBM SP-1 and SP-2 computers. The IBM SP-1 included 16 nodes, where each node was based on a IBM Power Risc, model 370. The two SP-2 included 16 nodes each, where each node was based on a IBM Power 2 Risc, model 390-thin and 390-wide, respectively. Overall, the work showed that the parallelized pseudospectral codes allowed an optimal exploitation of the computational capabilities of the computing machines that were tested. Floros and Reeve [82] presented an implementation of a spectral-element Navier-Stokes solver on three generations of parallel architectures, namely a Inmos T-800 transputer, an Intel iPSC/860 and a Meiko CS-3. While the performances of the older Transputer and iPSC/860 machines was fairly predictable, the behaviour of the code on the SPARC-10 based Meiko CS-2 demonstrated the influence of the internal-memory hierarchy on the computational performance. A mixed spectral-element, pseudospectral and finite-difference scheme for the Navier-Stokes equations has been implemented on a Meiko parallel computer by Prestin and Shtilman [83]. The performance level achieved for the spectral-element code on the 28-node Meiko computer approached 200 MFlops in single precision and 150 Mflops in double-precision arithmetics. Crawford et al. [84] presented benchmark results from the parallel implementation of a hybrid three-dimensional Navier-Stokes solver on different parallel platforms, namely a IBM SP-2, a SGI Power Challenge XL, and a Cray C-90. The solver is based on a mixed spectral element-Fourier expansion technique for complex geometries, in which Fourier space is used in the two homogeneous directions, and the spectral-element discretization in the third direction. A method for an efficient implementation of a combined spectral-finite difference algorithm for the calculation of incompressible stratified turbulent flows

on distributed-memory computers was developed by Garg et al. [85]: they consider the case of the stratified turbulent flow in a channel, as described by the Navier-Stokes equations and the scalar-transport equation. A mixed technique was used, Fourier decomposition and finite differences in space and a semi-implicit Crank-Nicolson, third-order Runge-Kutta scheme in time. A speedup analysis of a model problem was presented for three partitioning schemes. They tested the code on a Intel Paragon machine (128 processors), a Intel iPSC/860 computer (32 nodes), against a Cray Y-MP. They reported efficiencies in a $60\% \div 91\%$ range. Wasfy et al. [86] reported on the development of a parallel semi-implicit iterative finite- element procedure, for the unsteady incompressible fluid-flow equations. Tests were performed on a 6-processor SGI Onyx system, in particular related to three flow cases, the flow over a backward-facing step, the oscillating lid-driven square cavity, and the vortex shedding over a circular cylinder. It was found that, depending on the nature of the problem, an optimum time step and number of iterations exists that minimize the computing time. Garbey and Vassilevski [87] described two different algorithms to solve on parallel computers the unsteady three-dimensional Navier-Stokes equations. The test case chosen was the flow around a circular cylinder. Both algorithms revealed a high parallel efficiency on a parallel computer with uniform architecture as the Cray T-3E (24 processors). On found that the solver for the momentum equation was still scalable, while the pressure solver was not. Gropp et al. [88] presented parallel-performance results of implicit fluid-dynamic simulations based on finite discretization on static grids. It is found that, despite of the fact that large-scale unstructured implicit CFD computations have matured to a point of practical use on distributed/shared-memory architectures, a careful tuning is needed to obtain the best product in terms of efficiency-per-processor and global parallel efficiency. Moreover, the number of cache misses and the achievable memory bandwidth are two important parameters that should be considered in determining an optimal data-storage pattern. Kumar et al. [89] performed an interesting study in which MIMD parallel performances of a three-dimensional unsteady Navier-Stokes solver were presented, as measured onto a distributed-memory vector parallel Fujitsu VPP-700 computer. The numerical technique was based on a time-accurate cell-centered finite-volume method, within a Euler implicit time-marching setting. Three

problems of practical interest in the field of biomedical fluid mechanics were considered as test cases, namely the unsteady flow in an asymmetric constricted vessel, the unsteady flow in an asymmetric dilated vessel, and the unsteady flow in a doubly-constricted vessel. The Fujitsu VPP-700 was a distributed-memory-architecture machine, based on powerful proprietary vector processors. The performance of the parallel code was measured in terms of speedup- and efficiency factors, and was found rather encouraging. About 90% efficiency was obtained with large data sizes, using 16 vector CPUs. Further gains in the speedup could be achieved by dynamic-load distribution and fine-vectorization tuning.

Some authors also investigated on parallel-programming interfaces. Hoeflinger et al. [90] performed a study – a SGI Origin 2000 computer was used – about scalability of Navier-Stokes solvers as implemented with OpenMP, and also about MPI-to-OpenMP code migration. OpenMP is a relatively recent programming interface, targeted at MIMD parallelism, that can rather easily deliver good parallel performance in the presence of a small number of processors (< 16). Success with more processors become more difficult to be obtained. OpenMP makes use of processors by employing them as threads of control in a shared-address space. Directives express the parallelism in the program, that is then implemented in the code, as generated by the compiler. The threads share data by default, but they can also have private data. OpenMP is a industry standard on SMP systems, and is usually available on distributed-memory systems constituted by clusters of SMP nodes. MPI is a more mature message-passing library, and there have been many reports of highly-scalable MPI computational codes for large numbers (up to thousands) of processors. MPI is usually used on SPMD mode, and is based on independent processes that do not share any memory. Parallelism and data transfer in MPI are expressed through subroutine calls. MPI is widely regarded as a scalable parallel-programming paradigm because the programming model causes the user to rewrite a serial application all at once into a domain-decomposed program, that - by its nature - has high locality and whose processors actually interact very little. The authors explored a number of causes of poor scalability involved in the use of OpenMP. A list of key issues was provided that need to be addressed to make OpenMP a more easily-scalable paradigm. Dong and Karniadakis [91] presented a

hybrid two-level parallel paradigm with MPI/OpenMP in the context of high-order methods as implemented in the spectral/hp element framework, to take advantage of the hierarchical structures arising from CFD problems. The test case was the flow around a circular cylinder. The authors took a coarse-grain approach to OpenMP shared-memory parallelization and employed a work-load splitting scheme to reduce the OpenMP synchronizations at a minimum level. The hybrid algorithm showed good scalability with respect to both the problem size and the number of processors with fixed problem size. With the same number of processors, the hybrid model with two OpenMP threads per MPI process was observed to perform better than pure MPI and pure OpenMP on a SGI Origin 2000 computer (250 MHz MIPS R 10000) and a Intel IA-64 cluster (Titan, 800 MHz Itanium), while the pure MPI model performs the best on a IBM SP-3 (Blue Orizon, 375 MHz Power-3) and a Compaq Alpha cluster (Le Mieux, 1 GHz Alpha EV-68). A key new result was that the use of threads facilitated effectively p-refinement, a crucial issue to adaptive discretization using high-order methods. The scalability of hybrid programming in a CFD code on the Earth Simulator computer has been explored by Itakura et al. [92]. The Earth Simulator (ES, see Habata et al. [93], and Yanagawa and Suehiro [94]) is a highly-parallel vector supercomputing system, developed under a Japanese Government's initiative. In May 2002 the ES was acknowledged to be the most powerful computer in the world, with 35.86 TFlops on the Linpack HPC benchmark (87.5% of the system peak performance) and 26.58 TFlops for an atmospheric general-circulation code. Such performances could be attributed to three main architectural features, such as:

- vector processor (based on NEC SX-6 vector technology);
- shared memory;
- high-bandwidth non-blocking interconnection crossbar network.

The ES consists of 640 processor nodes (PN) and an interconnection network (IN), housed in 320 PN cabinets and 65 IN cabinets. The ES is installed in a specially-designed building 65m long, 50m wide and 17m high. In order to build-up the system, a number of hardware technologies have been developed, such as high-density and high-frequency LSI, high-frequency signal transmission, high-density packaging, and a high-efficiency cooling and power supply. One of the characteristic

features of the system is the “one chip vector processor”, with a peak performance of 8 GFlops. This highly-integrated LSI was fabricated using 0.15 m CMOS technology, with eight-layer copper interconnection. The operating system for the ES is based on SUPER-UX, the UNIX operating system for the NEC SX series of scientific supercomputers. There are two types of parallel-programming models available on the ES. One is a flat programming model in which a parallel program is implemented by MPI interfaces only, both within a SMP node and among nodes. The other one is a hybrid programming model, in which a parallel program is written using thread programming within a SMP node and MPI programming among nodes simultaneously. It is generally accepted that is difficult to obtain the same high level of performance using the hybrid programming model, as can be achieved with the flat programming model. Itakura et al. [92] evaluated the scalability of a computational code for the solution of the Navier-Stokes equations on the ES. They found that the hybrid programming model achieved a sustained performance of 346.9 GFlop/s, as compared with the flat programming model that achieved 296.4 GFlops, with 16 PN of the ES for a DNS problem size of 256^3. For small-scale problems however, the hybrid programming model was found not so effective, because of microtasking overhead. More recently, Xu [95] implemented a number of parallel models directed to the exploitation of the capability of massively-parallel computers, up to the tera-scale level, by a fully-spectral Navier-Stokes solver for turbulent-channel-flow DNS. Benchmark tests were executed onto a Blue Gene/L computer. A stabilized finite-element formulation for three-dimensional unsteady incompressible flows was implemented on a distributed-memory parallel computer by Behara and Mittal [96]. The scalability of the computations on a 64-processor Linux cluster was evaluated for problems with various sizes. The cluster was actually constituted by 32 nodes, each node equipped with two Intel Xeon processors, with clock rate of 3.06 GHz. Each processing unit included a main-memory unit of 2 GB RAM and 512 KB L2 cache. Grinberg et al. [97] developed and tested an effective and scalable low-energy-basis preconditioner for elliptic solvers. Computational tests were performed onto different high-performance computers. Some other work has been performed to investigate the possibility of automatically parallelize computational codes for the Navier-Stokes equations. Among others, Agrawal et al. [98] analyzed two Navier-

Stokes solvers implemented in High-Performance Fortran (HPF) and showed that it was necessary to manually insert calls to low-level parallel libraries, to achieve performances comparable to an equivalent message-passing implementation of the same codes. Overall, they have demonstrated that the automatic parallelization of computational codes for scientific applications remains largely unpractical.

As concerns processors, for a given period of time, the performance of microprocessors was rapidly increasing, and the evolution of their architectures was mainly driven by two factors:

- the increasing number of transistors per processor;
- the increasing difference between processor speed and memory-access time.

The main technological advance has been a reduction of the basic VLSI feature size. By looking at the computing power of a chip (the total transistor switching per second), the present transistor capacity has increased more than one order of magnitude with respect to the clock rates of the past two decades. The basic single-chip building block has provided - for a given period of time - increasingly larger capacities, so that more components could be packed on the chip (i.e. memory).

The difference between capacity and speed is more pronounced in the memory technology. From the mid-1980s and the mid-1990s, the capacity of DRAM chips has increased of about three orders of magnitude, actually quadrupling every two years, while the memory-cycle time has increased only by a factor of two. Processor design attempts to exploit the large number of available transistors to reduce the performance degradation due to memory latency, that oscillates from few nanoseconds for a first-level on-chip cache, to milliseconds for mass memory, i.e. I/O devices. Caches are used to reduce the memory-access latency, where thread-level parallelism allow the processor to overlap the memory-access time with other useful work. Multi-threading architectures provide special hardware to support parallelism at the thread-level, in which individual threads can simultaneously exploit instruction-level (ILP) parallelism (several instructions executed at the same time, see, among others, Dulong [99]). As concerns memory, logically-shared but physically-distributed memories are not easy to implement. Memory hierarchy in conventional architectures allows the data that are bound to an address, to be migrated toward the processor that requires them. Load instructions in distributed

shared memory architectures allow data to migrate toward the local memory of the processor that access it. Migrating and/or replicating data across a distributed shared-memory multiprocessor system presents a set of challenges due to issues related to coherence and remote-memory latency. A memory-consistency model for a shared-address space specifies constraints on the order in which memory operations have to be performed (i.e. become visible to the processors). The basic consistency model is the sequential consistency (see Lamport [100]) that, although intuitive, is not easy to be inexpensively implemented. Symmetric multiprocessors usually implement the snooping, a general technique for cache coherency that uses the serialization of the memory access by the shared bus. Another way to implement a coherence protocol is the use of a directory, where a central home unit keeps track of the memory locations shared by multiple caches, and of those that are held exclusively. All accesses are handled by the home that, on the basis of its directory, recognizes the caches that need to be updated or invalidated at any given memory access. This solution is sensitive to the speed of the interconnection network due to the fact that all the operations are handled centrally (see Hagersten et al. [101]).

Nowadays, in large-scale supercomputing systems, two types of parallel architectures can be mainly distinguished:

- distributed-memory parallel systems with cache-based superscalar microprocessors (the option followed for example with the ASCI project in the United States);
- distributed-memory parallel systems with vector processors (the option followed for example with the Earth Simulator project in Japan).

An extensive comparison between parallel cache-based superscalar- and parallel vector computing systems can be found in Oliker et al. [102], as related to several different key scientific-computing areas. The first two vector computers appeared in the early 1970s, and were the Texas Instrument TI-ASC and the CDC STAR-100. The development of vector computers actually began with the advent of the Cray 1 in 1976 (see Russel [103]), so starting in practice the history of high-performance

Table 3.6 – Vector computers manifactured 1972-1996 (data from Ref. [104])

Machine	*Year*	*Cycle Time (ns)*	*Pipes*	*Peak (Flop/s)/cycle*
TI-ASC	1972	60.0	4	4
CDC STAR-100	1973	40.0	1	2
Cray 1	1976	12.5	1	2
Fujitsu VP-200	1982	7.0	2	4
Cray X-MP	1983	9.5	1	2
Hitachi S-810/20	1983	19.0	2	12
NEC SX-2	1984	4.0	4	16
Cray 2	1985	4.1	1	2
Hitachi S-820/80	1987	4.0	4	12
Cray Y-MP	1988	4.3	1	2
Fujitsu VP-2600	1989	3.2	4	16
NEC SX-3	1990	2.9	4	16
Cray C-90	1992	4.0	2	4
NEC SX-4	1996	8.0	8	16

computing. The Cray 1 had 160 MFlop/s peak performance, and at that times, had a tremendous impact onto the scientific-computing

community. Subsequently in Japan, Hitachi, Fujitsu and NEC manufactured their own vector computers (see Oyanagi [105]), though developing different architectures from one another. In Tab. 3.6, a number of vector machines as manufactured from 1972 to 1996 are concisely outlined (data from Espasa et al. [104]). The key aspect of a vector architecture is the Single Instruction Multiple Data (SIMD) execution model. In a traditional scalar processor, the basic data type is a n-bit word. The architecture often exposes a register file of words, and the instruction set is composed of instructions that operate on individual words. In a vector architecture, a vector-data type is present, where a vector is a collection of VL n-bit words.

There may also be a vector-register file (the main innovation incorporated in Cray architectures), differently from the old-times vector machines in which vectors were stored in the main memory. Vector processors perform single operations on entire vectors, while classical processors execute entire micro programs on each data element of a stream. The effect of this difference is that in vector architectures the intermediate vectors produced by each instruction are stored in the vector-register file, while in a scalar processor intermediate values are consumed locally. Overall, the code complexity in such a processor increases.

Chapter 4

High Performance Computing

4.1 Introduction

Direct numerical simulations of turbulent flows require a great amount of computational intensive resources; as the Reynolds number increases the separation between the largest scales and the smaller scales of motions becomes greater, increasing the computational cost. Thanks to recent advances in high performance computing, it is possible to develop new powerful tools for fundamental turbulence research (high performance direct numerical simulation of turbulence).

The period between 1980s-1990s is referred as a *golden age* for parallel computing, because of a great increasing interest on parallelism, innovative architectures in the field of supercomputers and parallel programming models. From an economic point of view, however, the impact of these types of architectures is so strong that the spread of supercomputers for research is limited. So, one assists on the beginning of *distributed computing era*, which introduced the concept of massively parallel models applied to clusters of powerful microprocessors. The technological progress

is facilitated by the rapid improvements of microprocessors: they are economic and guarantee good performances as supercomputers.

Figure 4.1 plots the growth in processor performances since the mid-1980s [106]: in particular, it shows a significantly enhancement in performance that is equal to annual rate of over 50%. Technological improvements on microprocessors enhance the overall capability of computers, whose performances are comparable with those of supercomputers. Moreover, those improvements determine the spread of microprocessor-based computer design: the architectural growing in the number of transistors on semiconductor devices, associated to cost advantages of mass-produced microprocessors, produces, in general, an increasing in computer business, especially in the field of high performance computers (PC and workstation).

Since 2003, the limits of energy power, available instruction-level parallelism and long memory latency slowed this positive trend to about 20% (Fig. 4.1). Because of these limits, there was a transition from high performance microprocessors to higher performance multiple processors per chip, referred as cores, such as Intel and AMD have done since 2004: it is referred as the *multicore revolution*, because chip manufactures scale the number of cores per chip rather than clock frequencies to improve distributed computing. Innovation in architectural setting is reflected also in exploiting multithreading through *thread-level parallelism* (or TLP, in which threads are distributed across different parallel computing nodes) and *data-level parallelism* (or DLP, in which data are distributed across different parallel computing nodes): this switch had a deep impact on the software developer community, because of a different approach from an implicit to an explicit parallel programming model.

Nowadays, computational science and numerical simulations are in the midst of a technological revolution caused by recent trends in hardware that have redefined the concept of “parallelism”: high performance computing is oriented towards heterogeneous platforms, defined both by central and graphics processing units (GPUs). GPU is a processor optimized for 2D and 3D graphics, video, visual computing and display [107] and represents an evolution of the video graphics array (VGA) controller, adopted since the late 1980s, that was a memory controller and display generator connected to DRAM [107], definitively abandoned in 2000.

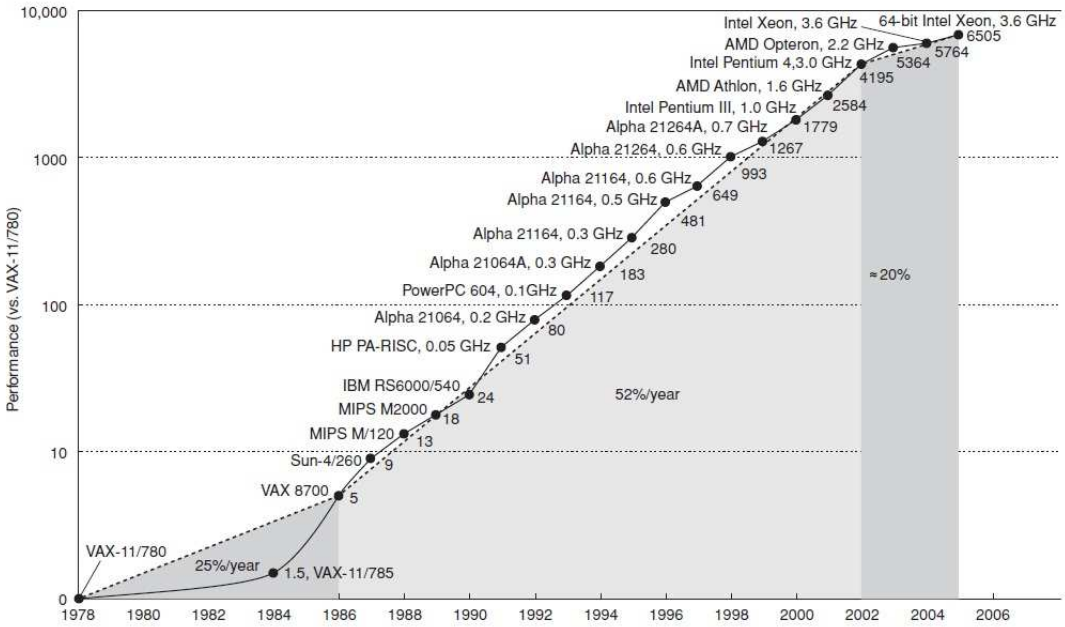

Figure 4.1 – Growth in processor performance since the mid-1980s (adopted from [106]).

Over time, the GPU becomes more and more sophisticated, in terms of programming, especially in the field of advanced graphics due to OpenGL and DirectX models implementations. These application programming interfaces (APIs) allow to accelerate graphics processing functions for more realistic 3D environments, especially for 3D PC Gaming.

Thanks to their massively parallel processors, GPUs became attractive also from a parallel computing point of view, opening the *visual computing era*, defined as the intersection between all graphics advanced processing and parallel computing.

Availability of HPC systems, especially based on CPU/GPU hybrid architectures, can led to significant advances in DNS, since it allows to achieve higher spatial and temporal resolutions and to develop a new generation of numerical solvers with emphasis on high accuracy and stability.

The purpose of the present chapter is to discuss the impact of HPC on DNS, with reference to wall bounded flows. Section 4.2 gives an overview about the most important available and traditional programming models; in Section 4.3, metrics for measuring performances are presented; in Section 4.4, a more detailed discussion about GPUs as parallel computing units is presented; Section 4.5 presents CUDA,

the new parallel environment developed by NVIDIA for GPUs; finally, Section 4.6 gives an overview of multi-GPU architecture.

4.2 Historical view on programming models

Parallel computing architectures started to appear at the end of the 1970s. One can often distinguish between SIMD machines (Single Instruction Multiple Data) and MIMD machines (Multiple Instruction Multiple Data) (see also the 1994 review of Fischer and Patera [108]).

In a SIMD type of architecture a large number of processors (each with its own memory) simultaneously carries out the same sequence of instructions at the same time, while operating on different local data. Regular-mesh explicit algorithms that only require nearest-neighbor communications can be efficiently implemented on such machines. In the past, a noticeable example of fine-grain SIMD machine has been represented by the Connection Machine computers, that, in some cases, have been equipped with up to thousands one-bit processors.

In MIMD architectures each node is a complete computer, with code and data stored in the local memory. MIMD machines provide for a great flexibility of use, but they may involve a remarkable degree of complexity in programming. A further distinction can be made among MIMD machines, distinguishing between machines with shared memory and distributed local memory. In shared-memory computers (Symmetric Multiprocessors, SMP) a limited number of processors, each one able to act independently, shares a unique area of memory that is accessed though a bus. An example of such machines is the Cray X-MP. Usually the scaling capabilities in shared-memory machines is poor, mainly due to the memory-access bottlenecks. This happens because, when the number of processors increases, it is no longer feasible to provide each processor with a parithetic access to the whole memory. Differently, in distributed-memory computers, a large number of processor (each one with its own memory) is connected through a switching and interconnection network, and communicate via explicit message passing (Massively Parallel Processors, MPP). If the switch is fast enough, the memory may be considered as shared, but this is more a programming model than a hardware property.

Since the 1990s, the development of computer processors has evolved toward the production of mass-produced powerful microprocessor chips, that soon became ready to replace the handcrafted processors that have characterized the early times in scientific computing. Since the microprocessor nodes have been rapidly become very fast, one of the issues today in MIMD machines is whether the switching and communication technology will be able to keep pace with that of the microprocessor chips. Modern powerful computing systems often include a given number of nodes connected through a network, each node being formed by a number of shared-memory processors. The network can be a relatively simple infrastructure, like in the case of the most common clusters, or be constituted by powerful proprietary connecting nodes. In fact, a cluster is commonly meant as a machine constituted by a number of CPUs with memory, connected through a network. The point of weakness of the cluster is usually the network, that, in terms of speed, in most cases represents the main bottleneck. A (large) multiprocessor computing machine is on the contrary constituted by a number of CPUs with memory, where the interconnection among the different processors is provided by sophisticated proprietary crossbar units, that are able to provide high-speed communication among the processors.

Three main parallel-programming models exist:

- *data parallel*: this model deals directly with multidimensional data arrays and is particularly useful on SIMD machines, using the Fortran 90 array syntax. The data-parallel model is naturally suited to SIMD hardware. One deals directly with data arrays and acquires neighboring information through data shifts. Several problems in computational fluid dynamics, in which data are carried on a well defined regular mesh, fit this programming model without particular difficulties;
- *message passing*: this model, developed for MIMD machines, leaves to the programmer the task of exchanging the appropriate amounts of data between the local memories and the processor nodes. Probably the most immediate way to adapt a Navier-Stokes code onto a MIMD machine is through the Domain Decomposition Message Passing (DDMP) model. This operation results relatively straightforward for explicit finite-difference schemes for the numerical integration of the compressible-flow equations, in which only local

space-time physics is involved. By decomposing the total computational domain into subdomains – each of them attributed to a processor node in charge of all its arithmetics – only boundary data have to be transmitted between the different subdomains. For schemes that require a global-solving operation such as in the case of the incompressible-flow equations, the problem is less simple, but still it is possible to reduce the algebraic problem to one involving only boundary data;

- *shared memory*: this model treats the total memory of the machine as equally accessible to each node. The shared-memory programming model represents the natural extension of the multitasking model. Synchronization is the key problem and it requires explicit use of barriers and locks. The analysis of the performance of a multiprocessor computing system is important for the determination of the behavior of a computational code on a given machine and, more generally, the clarification of the adequacy of the architectural scheme at hand with respect to a given class of problems (see, among others, Cremonesi et al. [109]).

4.3 Metrics for measuring performances

Measurements can be gathered on existing systems by means, for example, of benchmark applications, the latter aimed to assess the characteristics of an existing or a newly-released computer. Parallel benchmarks extend the traditional sequential benchmarks, providing a wide set of suites that exercise each system component with a targeted workload. The PARKBENCH suite (Dongarra et al. [110], especially oriented to message-passing architectures) and the NPB (NAS Parallel Benchmarks, http://www.nas.nasa.gov/Software/NPB) are examples of commonly-used benchmark suites (see Alfonsi and Muttoni [111] and references therein).

A frequently-used metric for the evaluation of a particular run on a sequential machine is the execution or elapsed time. The elapsed time is a metric of high level because it captures the effects of the software structure, the system components (CPU, memory bandwidth, I/O transfer rate), the workload mix and the execution mode (dedicated vs. multi-programmed system).

The most commonly-used metrics in parallel computers is the *speedup*, that gives the measure of the actual performance of computational code on a multiprocessor system, with respect to an ideal value (generally represented by the execution time of the sequential counterpart). The speedup captures the effects of all the factors that characterize a parallel execution, namely the number of processors, the communications exchanged among the processors, the I/O activity, the structure of the code and - eventually - the simultaneous execution of different activities. Other metrics exist for the evaluation of specific components of a parallel architecture (i.e., Flops/s for the CPU power, MB/s for the channel bandwidth) but, due to the complexity that characterizes a multiprocessor system, the overall performance can hardly be derived from that of its single individual components. Scalability metrics describe the characteristics of a computational code in terms of gain or loss of performance as a function of the number of processors.

Being T_1 the elapsed time of an application when executed on a single processor and T_p the elapsed time of the same application with a given number of processors n_p, the speedup is defined as:

$$S_p = \frac{T_1}{T_p} = \frac{1}{T} \tag{4.1}$$

($T = T_p / T_1$ is the non-dimensional runtime). In most cases, T_1 is the elapsed time of the sequential version considered as a benchmark. In its original form, Amdahl's law states that the speedup is bounded as:

$$S_p \leq \frac{1}{f_{seq} + \left(1 - f_{seq}\right)/n_p} \tag{4.2}$$

(f_{seq} is the fraction of the elapsed time inherently sequential). Very small values of f_{seq} are required to achieve significant speedups, due to the fact that no code can be executed faster than its sequential part. Ideally, when $f_{seq} = 0$ the speedup is linear

($S_p = n_p$, the ideal value of the speedup). Two upper bounds on the speedup can be identified from Eq. (4.2). They are expressed, in terms of the structure of the application by f_{seq}, i.e. the software bound, and in terms of the properties of the parallel architecture by the number of processors n_p, i.e. the hardware bound.

Other metrics can be derived from the speedup. The *efficiency* of an application is defined as:

$$E_p = \frac{S_p}{n_p} = \frac{T_{id}}{T_p} \tag{4.3}$$

($T_{id} = T_1 / n_p$ is the ideal time), i.e. the time spent with one processor divided by the number of processors. The efficiency represents the fraction of time during which the n_p allocated processors are usefully employed, or alternatively the speedup per allocated processor. The maximum value of the efficiency ($E_p = 1$) is obtained when the speedup is linear.

The *effectiveness* of an application is:

$$E_f = \frac{S_p^2}{n_p} \tag{4.4}$$

(also defined as $E_f = S_p / n_p$), that takes into account both the gain and the cost of a computation (Ghosal et al. [112]). The value of n_p corresponding to the maximum effectiveness is the processor working-set of a computational code, i.e. the number of processors that maximize the speedup per unit of cost, so identifying an optimal operating point.

4.4 GPUs as parallel computers

GPGPU (General Purpose Graphics Processing Unit) technology has drastically influenced the HPC market, because GPUs have been considered not only as powerful graphics device but have been used also as high performance manycore processors for accelerating a wide range of scientific applications. In this context, NVIDIA and AMD-ATI, the two most influent worldwide leaders in the graphics card field, proposed proprietary GPGPU programming frameworks: CUDA, developed by NVIDIA, and ATI Stream, developed by AMD-ATI.

The introduction of CUDA, as described in the following sections, opened a new era of improved performance for many applications as a simpler GPU programming: it is a general purpose parallel computing architecture – with a new parallel programming model and instruction set architecture – that leverages the parallel computing engine in NVIDIA GPUs to solve many complex computational problems in a more efficient way than on a CPU [113]. It provides also a framework built around the C programming language, but only runs on NVIDIA GPUs.

ATI Stream technology, instead, is a set of advanced hardware and software technologies that enables AMD graphics processors working in concert with the system's central processors to accelerate applications and to run computational-intensive tasks more efficiently [114]. ATI Stream is, more specifically, a cross platform that only runs on AMD GPUs. Central to the technology is the high-level language Brook+, based on C/C++ languages. Brook+ allows programmers to write CPU codes and synthetically simple GPU kernel functions, that are compiled separately, dividing the code into CPU and GPU components. For ATI Stream SDK to become a more user-friendly programming environment, a more mature development tools need to be added.

Both NVIDIA and AMD support OpenCL (Open Computing Language): it represents the first standard for general purpose parallel programming on heterogeneous systems and supports both data-parallel and task-parallel programming models. The key feature of OpenCL is that it is designed as a parallel platform for programming across CPUs and GPUs. It is an open standard defined by the Khronos Group [115]. This makes it different to either NVIDIA's CUDA or

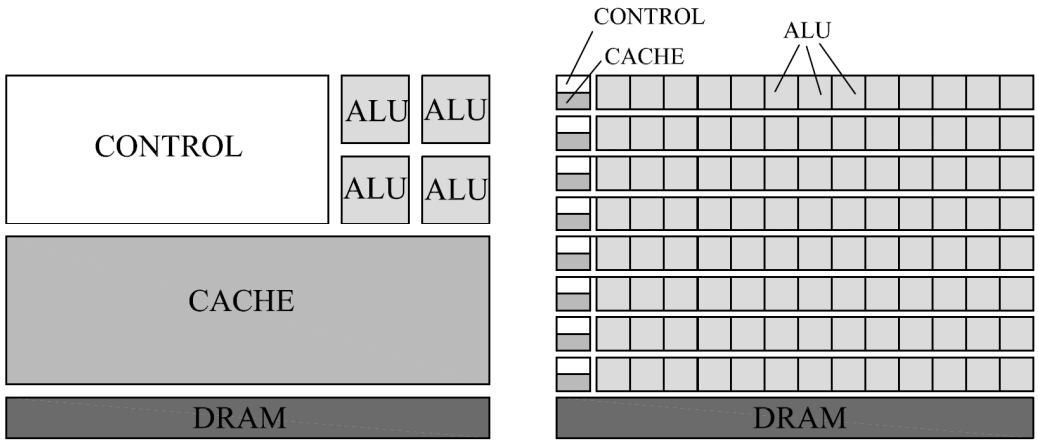

Figure 4.2 – Typical scheme of a multicore CPU (on the left) and of a GPU (on the right).

AMD's ATI Stream, which are proprietary and designed to only work on their respective hardware platforms, becoming attractive to software developers.
Because of its robustness, actually NVIDIA CUDA is considered more mature as programming environment, especially in terms of development tools and stability: for this reason, it is preferred at ATI's Stream SDK and OpenCL, that is still in its infancy.

Combining advantages carried out both from the CPU and GPU, one assists on the development of heterogeneous systems, based on a different processor types: in the field of high performance computing, the main innovative idea consists on executing sequential part of the CPU and numerically intensive part on the GPU [116] in order to increase dramatically the scalability of the cores. More in detail, in the field of microprocessors, it is possible to distinguish between multicore (CPU) and manycore (GPU).

The schematic view of different architectures related to multicore (CPU) and manycore (GPU) are shown in Fig. 4.2. A multicore processor is designed to exploit massive quantities of on-chip resources in an efficient and scalable manner: it consists of two or more powerful cores (ALU, Arithmetic Logic Unit) on a single processor, which perform arithmetic and logical operations, a control unit for instruction execution and a CPU cache for reducing the average time to access memory. An example is given by Intel Core i7 microprocessors, which has four

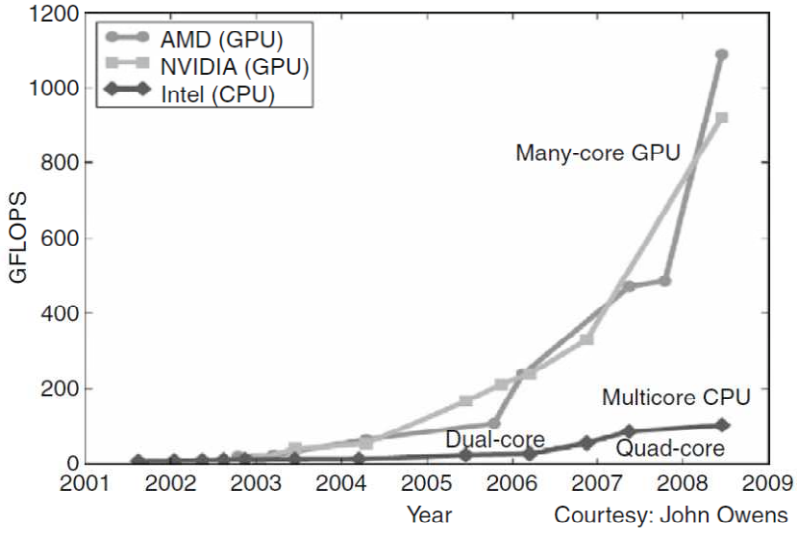

Figure 4.3 – Enlarging performance gap between GPUs and CPUs (adopted from [116]).

processor cores, each of which is an out-of-order: they support hyperthreading technology, designed to maximize the execution speed of sequential programs. This is guaranteed by a sophisticated control logic of sequential instructions, in which a large cache memory allows to reduce the instructions and data access latencies of large complex applications. In contrast, a manycore processor is characterized by a large number of much smaller cores and focuses more on the execution throughput of parallel applications. An example is the NVIDIA GTX280 graphic processing unit (GPU) with 240 cores, each of which is massively multithreaded, in-order, single-instruction issue processor that shares its control and instruction cache with seven other cores [116]. Manycore processors, especially the GPUs, have led the race of floating point performance since 2003. This phenomenon is illustrated in Fig. 4.3. While the performance improvement of general-purpose microprocessors has slowed significantly, the GPUs have continued to improve relentlessly. As of 2009, the ratio between manycore GPUs and multicore CPUs for peak floating point calculation throughput, is about 10 to 1 (1 Tflops *versus* 100 Gflops).

In the next section, GPU architectures and CUDA programming paradigm are described in detail.

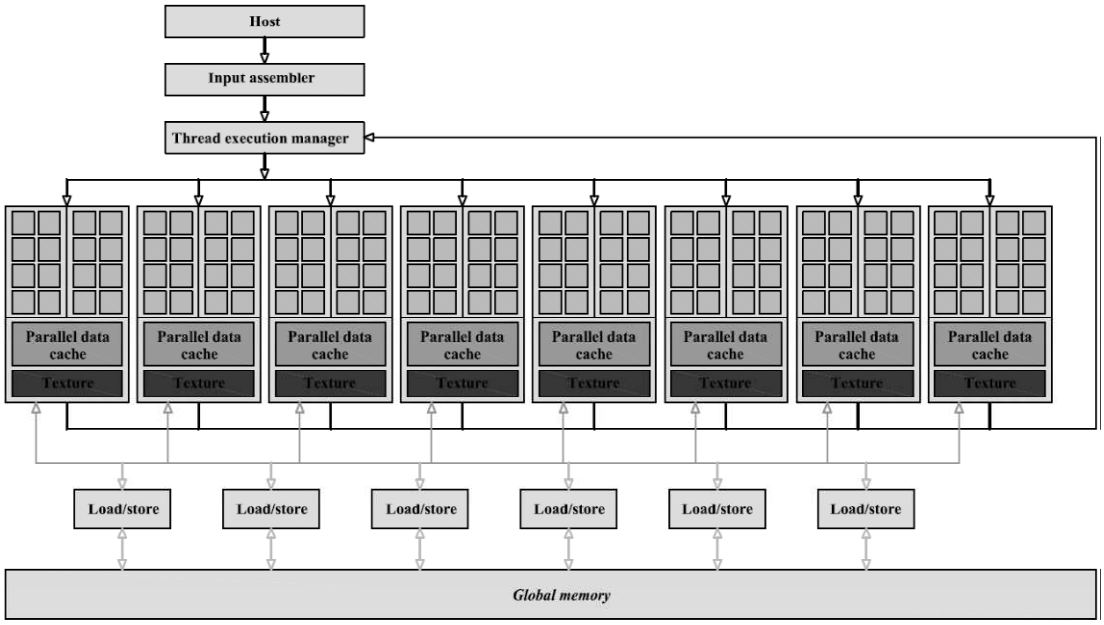

Fig. 4.4 – Architecture of a CUDA-capable GPU (adopted and elaborated from [116]).

4.5 CUDA architecture and programming model

4.5.1 A parallel computing architecture

In November 2006, NVIDIA introduced CUDA, a general purpose parallel computing architecture, with a new parallel programming model and instruction set architecture, that leverages the parallel compute engine in NVIDIA GPUs to solve many complex computational problems in a more efficient way than on a CPU. CUDA comes with a software environment that allows developers to use C as a high-level programming language.

The CUDA Software Development Environment is a suite for advanced programming, that includes numerical libraries (i.e., BLAS, FFT, etc.), C-runtime (as support for executing standard C language and others high-level languages such as Fortran, Java and Python), tools, documentation and samples. In particular, the set of CUDA tools includes the NVIDIA C compiler (nvcc), the CUDA debugger (cuda gdb), the CUDA visual profiler (cuda prof) and more other tools [117].

In order to understand the programming model, it is necessary to focus on hardware architecture of a typical CUDA-capable GPU.

Figure 4.4 shows the architecture of a typical CUDA-capable GPU. It is organized into an array of highly threaded streaming multiprocessors (SMs). In Fig. 4.4, two

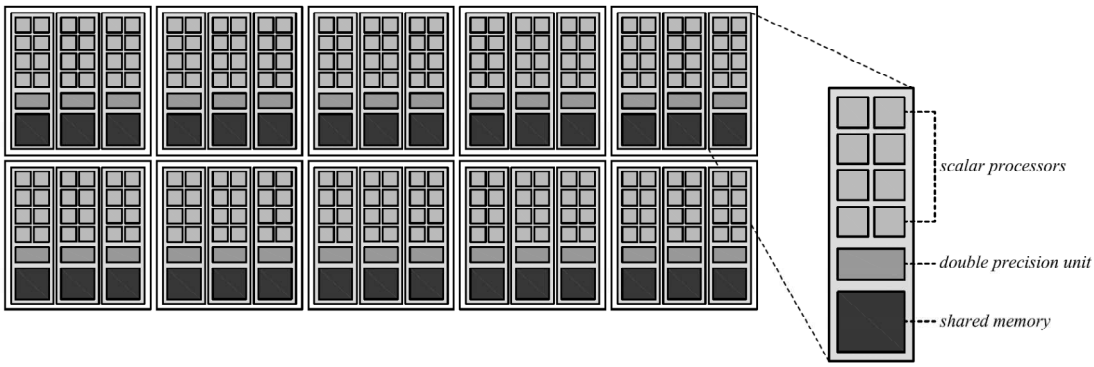

Fig. 4.5 – NVIDIA Tesla 10-Series GPU: a scheme.

SMs form a building block; however, the number of SMs in a building block can vary from different CUDA GPUs generations. Also, each SM has a number of streaming processors (SPs) that share control logic and cache. Each GPU currently comes with up to 4 gigabytes of graphics double data rate (GDDR) DRAM, referred as global memory in Fig. 4.4. These GDDR DRAMs differ from the system DRAMs on the CPU motherboard; for computing applications they function as very-high-bandwidth, off-chip memory, though with somewhat more latency than typical system memory. For massively parallel applications, the higher bandwidth makes up for the longer latency.

The Tesla 10-Series architecture, designed by NVIDIA for scientific computing, is based on a scalable processor array. Figure 4.5 shows a block diagram of the G80, that introduces the CUDA architecture: it has 86.4 GB/s of memory bandwidth, plus an 8 GB/s communication bandwidth with the CPU via the PCI Express. The communication bandwidth is also expected to grow as the CPU bus bandwidth of the system memory grows in the future. The massively parallel G80 chip (Fig. 4.5) has 240 SPs (30 SMs, each with 8 SPs). Each SP has a multiply–add (MAD) unit and an additional multiply unit. With 240 SPs, that's a total of over 1 teraflop. In addition, special function units perform floating-point functions such as square root (SQRT), as well as transcendental functions. Because each SP is massively threaded, it can run thousands of threads per application. A good application typically runs 5000–12000 threads simultaneously on this chip. For those who are used to simultaneous multithreading, note that Intel CPUs support 2 or 4 threads, depending on the machine model, per core. The G80 chip supports up to 1024 threads per SM and up

to about 30000 threads for the chip. It is very important to strive for such levels of parallelism when developing GPU parallel computing applications.

4.5.2 Programming model

The CUDA scalable parallel programming model extends the C and C++ languages to exploit large degrees of parallelism for general applications on highly parallel multiprocessors, particularly GPUs. Since NVIDIA released CUDA in 2007, developers have rapidly developed scalable parallel programs for a wide range of applications, that scale transparently to hundreds of processor cores and thousands of concurrent threads.

CUDA provides three key abstractions: a hierarchy of thread groups, shared memories and barrier synchronization, that provide a clear parallel structure to conventional C code for one thread of the hierarchy. Multiple levels of threads, memory and synchronization provide fine-grained data parallelism and thread parallelism, nested within coarse-grained data parallelism and task parallelism. The programming model scales transparently to large number of processor cores: a compiled CUDA program executes on any number of processors, and only the runtime system needs to know the physical processor count. The programmer writes a sequential program that calls parallel *kernels*: a kernel executes in parallel across a set of parallel threads. The programmer organizes these threads into a hierarchy of threads blocks and grid of thread blocks. A thread block is a set of concurrent threads that can cooperate among themselves through barrier synchronization and through shared access to a memory space private to the block. A grid is a set of thread blocks that may each be executed independently and thus may execute in parallel. When invoking a kernel, the programmer specifies the number of threads per block and the number of blocks comprising the grid. Each thread is given a unique thread ID number **threadIdx** within its thread block, numbered 0,1,2,…,*blockDim*-1, (where *blockDim* is the dimension of the block) and each thread block is given a unique block ID number **blockIdx** within its grid.

All thread creation, scheduling and termination is handled for the programmer by the underlying system. Indeed, a Tesla architecture GPU performs all thread management directly in hardware. The threads of a block execute concurrently and

may synchronize at a synchronization barrier by calling the **_syncthreads()** intrinsic. This guarantees that no thread in the block can proceed until all threads in the block have reached the barrier. After passing the barrier, these threads are also guaranteed to see all writes to memory performed by threads in the block before the barrier. Thus, threads in a block may communicate with each other by writing and reading per-block shared memory at a synchronization barrier. Since threads in a block may share memory and synchronize via barriers, they will reside together on the same physical processor or multiprocessor. The number of thread blocks can, however, greatly exceed the number of processors. The CUDA thread programming model virtualizes the processors and gives the programmer the flexibility to parallelize at whatever granularity is most convenient. Virtualization into threads and thread blocks allows intuitive problem decompositions, as the number of blocks can be dictated by the size of the data being processed rather than by the number of processors in the system. It also allows the same CUDA program to scale to widely varying numbers of processors cores.

To manage this processing element virtualization and provide scalability, CUDA requires that thread blocks be able to execute independently. It must be possible to execute blocks in any order, in parallel or in series. Different blocks have no means of direct communication, although they may coordinate their activities using atomic memory operations on the global memory visible to all threads. This independence requirement allows thread blocks to be scheduled in any order across any number of cores, making the CUDA model scalable across an arbitrary number of cores as well as across a variety of parallel architectures. It also helps to avoid the possibility of deadlock. An application may execute multiple grids either independently or dependently. Independent grids may execute concurrently, given sufficient hardware resources. Dependent grids execute sequentially, with an implicit interkernel barrier between them, thus guaranteeing that all blocks of the first grid complete before any block of the second, dependent grid begins.

Threads may access data from multiple memory spaces during their execution. Each thread has a private local memory. CUDA uses local memory for thread-private variables that do not fit in the thread's registers, as well as for stack frame and register spilling. Each thread block has a shared memory, visible to all threads of the

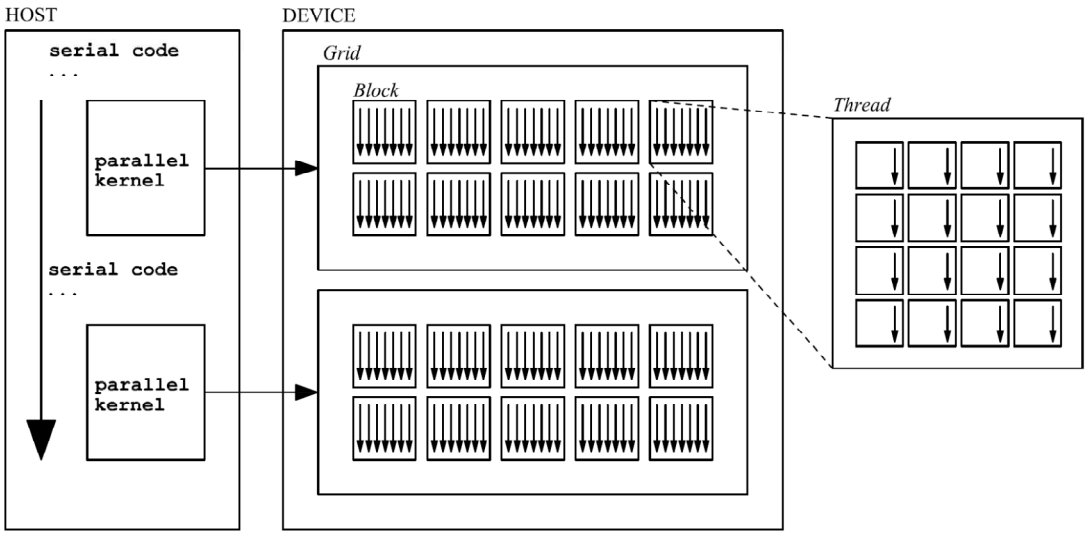

Fig. 4.6 – Heterogeneous programming execution model.

block, which has the same lifetime as the block. Finally, all threads have global memory with the **_shared_** and **_device_** type qualifiers. On a Tesla architecture GPU, these memory spaces correspond to physically separate memories: per-block shared memory is a low-latency on chip RAM, while global memory resides in the fast DRAM on the graphics board.

A program manages the global memory space visible to kernels through calls to CUDA runtime, such as **cudaMalloc()** and **cudaFree()**. Kernels may execute on a physically separate device, as is the case when running kernels on the GPU. Consequently, the application must use **cudaMemcpy()** to copy data between the allocated space and the host system memory.

The CUDA programming model is similar in style to the familiar single-program multiple data (SPMD) model – it expresses parallelism explicitly and each kernel executed on a fixed number of threads. However, CUDA is more flexible than most realizations of SPMD, because each kernel call dynamically creates a new grid with the right number of thread blocks and threads for that application step. The programmer can use a convenient degree of parallelism for each kernel, rather than having to design all phases of the computation to use the same number of threads.

As illustrated by Fig. 4.6, the CUDA programming model assumes that the CUDA threads execute on a physically separate device that operates as a co-processor to the host running the C program. This is the case, for example, when the kernels executes

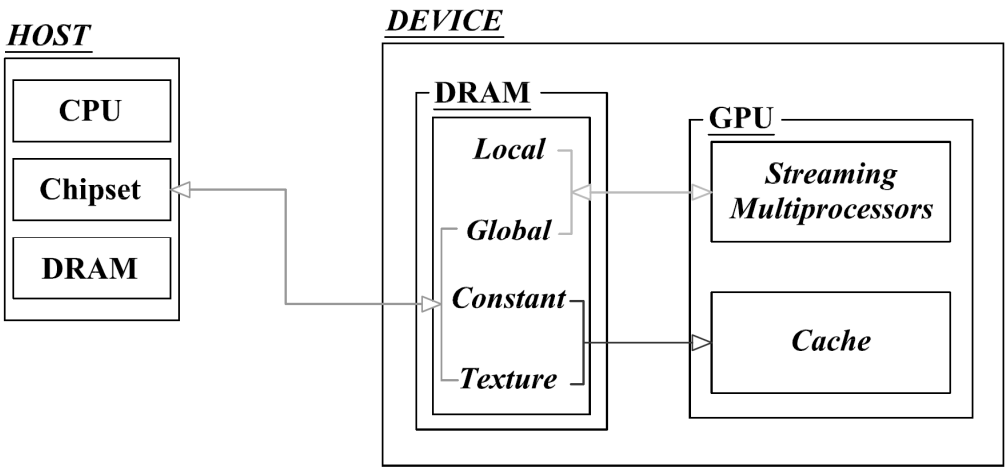

Fig. 4.7 – Memory Architecture of a CUDA-capable GPU.

on a GPU and the rest of the C program executes on a CPU. The CUDA programming model also assumes that both the host and the device maintain their own separate memory spaces in DRAM, referred to as host memory and device memory, respectively (Fig. 4.7). Therefore, a program manages the global, constant and texture memory spaces visible to kernels through calls to the CUDA runtime. This includes device memory allocation and deallocation as well as data transfer between host and device memory (Fig. 4.6). Furthermore, this reflects the reality that devices are typically hardware cards that come with their own dynamic random access memory (DRAM). For example, the NVIDIA T10 processor comes with up to 4 GB (billion bytes, or gigabytes) of DRAM. In order to execute a kernel on a device, the programmer needs to allocate memory on the device and transfer pertinent data from the host memory to the allocated device memory. Similarly, after device execution, the programmer needs to transfer result data from the device memory back to the host memory and free up the device memory that is no longer needed. The CUDA runtime system provides application programming interface (API) functions to perform these activities on behalf of the programmer. From this point on, we will simply say that a piece of data is transferred from host to device as shorthand for saying that the piece of data is transferred from the host memory to the device memory. The same holds for the opposite data transfer direction.

Figure 4.8 shows an overview of the CUDA device memory model for programmers to reason about the allocation, movement, and usage of the various memory types of

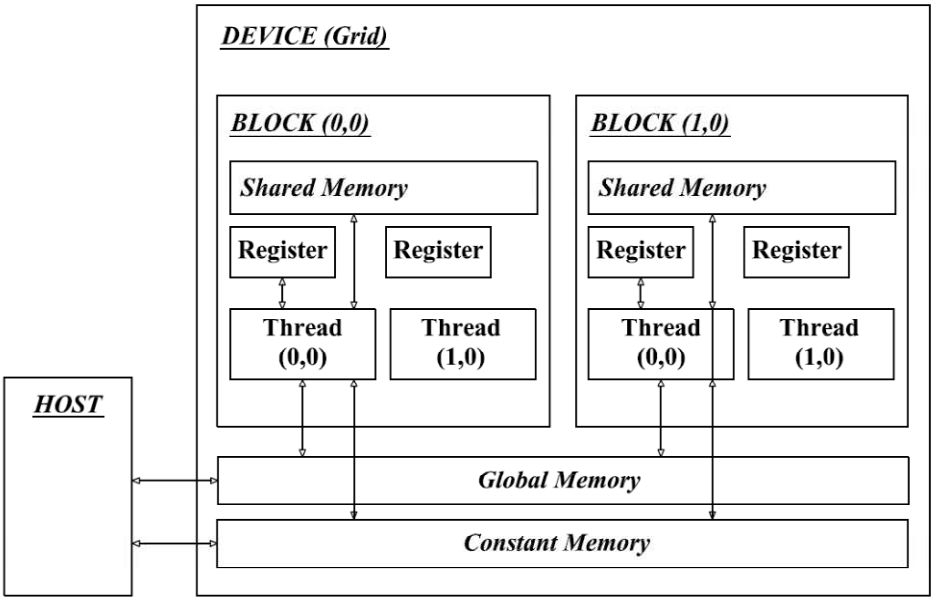

Fig. 4.8 – CUDA device memory model (adopted and elaborated from [116]).

a device. At the bottom of the figure, we see global memory and constant memory. These are the memories that the host code can transfer data to and from the device, as illustrated by the bidirectional arrows between these memories and the host. Constant memory allows read-only access by the device code.

The CUDA memory model is supported by API functions that help CUDA programmers to manage data in these memories. The function **cudaMalloc()** can be called from the host code to allocate a piece of global memory for an object. The reader should be able to notice the striking similarity between **cudaMalloc()** and the standard C runtime library **malloc()**. This is intentional; CUDA is C with minimal extensions. CUDA uses the standard C runtime library **malloc()** function to manage the host memory and adds **cudaMalloc()** as an extension to the C runtime library. By keeping the interface as close to the original C runtime libraries as possible, CUDA minimizes the time that a C programmer needs to relearn the use of these extensions. The first parameter of the **cudaMalloc()** function is the address of a pointer variable that must point to the allocated object after allocation. The address of the pointer variable should be cast to (**void ****) because the function expects a generic pointer value; the memory allocation function is a generic function that is not restricted to any particular type of objects. This address allows the **cudaMalloc()** function to

write the address of the allocated object into the pointer variable. The second parameter of the **cudaMalloc()** function gives the size of the object to be allocated, in terms of bytes. The usage of this second parameter is consistent with the size parameter of the C **malloc()** function.

4.5.3 Precision on GPUs

Regarding numerical accuracy and precision, NVIDIA devices have been designed to follow the IEEE-754-2008 standard for binary floating point arithmetics. As reported in [118], IEEE-754-2008 governs binary floating point arithmetic, specifying number formats, basic operations, conversions and exceptional conditions. Floating-point representation type has a base β (generally assumed to be even) and a precision p; thus, a floating-point number is represented as:

$$\pm\left(d_0 + d_1\beta^{-1} + ... + d_{p-1}\beta^{-(p-1)}\right)\beta^e, \left(0 \le d_i < \beta\right) \tag{4.5}$$

where $\pm d_0 \cdot d_1 \cdot d_2 \cdot ... \cdot d_{p-1}$ is called significand and has p digits [119].

Schematically, the standard can be encoded by three fields: sign, exponent and fraction. The 32- and 64-bit basic binary floating-point formats corresponds to float and double in C language (single and double precision in Fortran 90): the sign can be positive or negative, the exponent encodes the exponent in base 2 and the fraction encodes the significand. A float is a binary format that occupies 32 bits (4 bytes) and its significand has a precision of 24 bits, while a double is a binary format that occupies 64 bits (8 bytes) and its significand has a precision of 53 bits. The IEEE-754 standard supports a lot of operations, such as arithmetic operations, conversion operations, scaling, sign operations and comparison, whose properties are described in Whitehead and Fit-Florea [120]. Particular attention have to be focused on mathematical function accuracy: the accuracy of a floating-point arithmetic operation is determinate by the maximal error introduced by the operation itself. Obviously, the smaller error corresponds to the higher accuracy. When a result cannot be exactly represented, that is when the significand needs too many bits to be represented exactly [116], it is necessary to round it. Arithmetic operations are simple enough

that computing the best floating-point results is easy. For other mathematical operations, such as division and transcendental functions, it is harder: typically, it is necessary to implement iterative approximation algorithms and, if the hardware does not perform a sufficient number of iterations, the result may have an error larger than 0.5 ULP, which is the half the place value of the least significant place, used to measure the precision in numerical calculations. The consequence is that different mathematical libraries cannot achieve the same results for a given input and functions compiled for the GPU using CUDA, compared with those compiled for the CPU, can differ slightly. It influences a lot porting of numerical codes from the CPU to the GPU: with respect to the CPU, the GPU has fused-multiply-add (FMA) operations, that are able to speed up and improve the accuracy of many operations that involve products (i.e., dot product, matrix multiplication, polynomial evaluation and Newton's method). It consists on perform a floating-point multiply-add operation in one step, with a single rounding.

Consequently, the same sequence of operations may give different results according the CPU and the GPU implementation: this is reflected also in parallelizing algorithms. The arithmetic operations on newer generation GPUs are much more accurate.

Finally, the most important key features that guarantee good performances in speed up and accuracy considering GPU's implementation can be synthesized as:

- use of fused-multiply-add operation,
- compare results carefully among implementations, because algorithms can compute the same mathematical quantity but be numerically different,
- know the capability of GPU: device computing capability up to 2.0 version supports both single and double precision IEEE-754-2008 including FMA operations,
- take advantage of the CUDA math library functions, listed on *Appendix C* of the *CUDA C Programming Guide* and supported in device code [120].

4.5.4 Performance metrics for GPUs

A correct evaluation of performances of CUDA programs regards not only on traditional metrics, such as speed up and efficiency, but also measuring the

bandwidth. The evaluation of timing kernel execution can be done by considering CPU or GPU timers. In order to accurately measure the elapsed time for a CUDA call or a sequence of CUDA calls, it is necessary to synchronize the CPU thread with the GPU by calling **cudaThreadSynchronize()** before starting and stopping the CPU timer [121], because all kernels launcher are asynchronous.

The synchronization functions concern also streams and events: **cudaStream-Synchronize()** is devoted to block the CPU thread until all CUDA calls previously launched into the given stream have completed; **cudaEventSynchronize()**, indeed, blocks until a given event in a particular stream has been recorded by the GPU. It is possible to use also CUDA GPU timers, which are API functions that provide to create, destroy and record events, such as **cudaEventRecord()** and **cudaEventElapsedTime()** functions. The first one is devoted to assign the start and stop event into the stream *0*, while the second one returns the time elapsed between the recording of start and stop events, expressed in milliseconds.

The most important element in massively-parallel programming environment based on GPUs is represented by *bandwidth*, that represents the rate at which data can be transferred. That rate is influenced essentially by memory and the order in which it is accessed. Its measure is done by using a theoretical and an effective bandwidth. Theoretical bandwidth depends on hardware specifications. For example, the NVIDIA Tesla C-1060 uses DDR-RAM with a memory clock rate equal to 800 MHz and a 512-bit wide memory interface. So, the peak theoretical memory bandwidth is calculated as follows:

$$\frac{800\times10^{6}\times\frac{512}{8}\times2}{10^{9}}=102.4\,GB/s \tag{4.6}$$

Effective bandwidth, indeed, depends on how data is accessed by the program and it is calculated as follows:

$$Eff.Bandwidth=\frac{\left(\frac{B_r+B_w}{10^{9}}\right)}{time} \tag{4.7}$$

where B_r is the number of bytes read per kernel, B_w is the number of bytes written per kernel and time is expressed in seconds (also effective bandwidth is measured in *GB/s*) [121]. To achieve good performance, it is necessary to understand some basic properties related to the architecture in addition to the programming model: it allows to find the most efficient mapping of the specific application's data-structures to CUDA's domain-based model, in order to minimize communication between host and device and maximize the bandwidth used. Finally, programmers have to avoid the overhead of global synchronization as much as possible because it influences timing performances.

4.6 Multi-GPU architecture

The implementation of CUDA on heterogeneous systems to solve very expensive problems may require *multi-GPU architectures*. Programming model of a multi-GPU architecture is not so different than that described for a single GPU one: the key feature is to exploit efficiently hardware parallelism. In fact, multi-GPU consists of decomposing and distributing a working set across different GPUs in order to improve performances especially in those cases in which the working set exceeds single GPU's memory.

Two configurations of multi-GPU are possible: all GPUs are interconnected in a single network, otherwise GPUs are interconnected across network nodes. Considering the situation about multi-GPU within a node, GPUs can be controlled via a single CPU thread or by using multiple CPU threads belonging to the same process. In this case, processes have its own address space on the CPU side whereas multiple threads can share their address space. The identification of the active (or current) device is the only aspect that defines a difference between multiple cores and multi-GPU: it can be done by using the **cudaSetDevice()** function, that provides the ID of the GPU used. The new multi-GPU programming model can be easily incorporated into the existing multi-threaded CPU code, developing only the compute-intensive portion of the application for the GPUs and adding more calls to transfer data between CPU and GPU. As described in [121], it is necessary to define how the new programming model on multi-GPU is organized. Because of

collaboration between the CPU and the GPU, CUDA provides to establish a context between a CPU thread and the GPU. The context describes the device state, such as streams, events, allocated memory and so on, and it can be associated only to one GPU at any particular instant. Thus, CUDA streams and events are created per device: in the multi-GPU configuration, streams are used whenever an asynchronous call is issued at the current device, while events can be recorded only to a stream of the same device. As previously said, multi-GPU within a node can be controlled by a single CPU thread: it is necessary to set the current device to issue all CUDA calls, except for asynchronous peer-to-peer memcopies. Moreover, asynchronous calls (such as kernels and memcopies) do not block switching the GPU.

Multi-GPU can be controlled also by using multiple CPU threads belonging to the same process or by using multiple CPU processes. In the first case multiple threads are referred to the same process, as in single thread, while in the second case multiple processes have their own address spaces and there is no matter if they are on the same or different nodes. Thus, in order to issue work to p GPUs concurrently, a program have to use p CPU threads, each with its own context: in this case, some type CPU-side parallelism will be needed (OpenMP or MPI).

Chapter 5
The Navier-Stokes Solver

5.1 Introduction

Within the DNS approach and with respect to the channel flow problem, the unsteady three-dimensional Navier-Stokes equations have to be solved numerically in order to understand the mechanics of wall-bounded turbulent flows [17]. Since evaluating simulation databases of turbulence require a great amount of computational resources, the advances in hardware developments have led to increased utilization of high performance computing techniques to perform direct numerical simulations with increasing Reynolds number. An overview of the main approaches for wall-bounded turbulent flows, both from a numerical and computational point of view, is given on *Chapter 3.*

Here, a mixed spectral-finite difference algorithm for the integration of the Navier-Stokes equations applied to the case of the channel flow is considered: after a description of the main components of a Navier-Stokes solver, given in Section 5.2, the numerical scheme adopted and its properties are described in Section

5.4 describes the computing systems on which numerical simulations have been carried out; Section 5.5 describes the sequential implementation and an OpenMP implementation of the Navier-Stokes solver; finally, Section 5.6 shows a novel parallel implementation on a hybrid CPU/GPU architecture (both single- and multi-GPU).

5.2 The components of a Navier-Stokes solver

The advent of high performance computing combined with the development of numerical algorithms for solving physical problems, such as turbulence, has constituted a new "third approach", the advanced computational fluid dynamics. The aim is to synthesize both the advantages of experimental and theoretical approaches in order to analyze the mechanisms of fluid phenomena.

In general, given a problem, the scientific method, based on the numerical approach, consists on identifying the mathematical model, characterized by a set of governing equations, discretizing the computational domain and integrating the equations on the corresponding grid by using a numerical algorithm, in order to obtain an approximate accurate solution [122].

The main components of a direct numerical simulation are reported in the following:

- the mathematical model based on the governing equations;
- the domain discretization that includes a spatial discretization and a numerical discretization;
- the analysis of the stability and the accuracy of the numerical method;
- the evaluation of an appropriate numerical scheme for time marching in order to integrate the governing equations at each point of the computational domain;
- the analysis of the numerical databases of computed velocity and pressure fields in order to provide information about turbulent phenomena near the wall.

More specifically, in the case of the channel flow problem, the first step is to define the governing equations that describe the physics of turbulence: they are represented by the Navier-Stokes equations that, as reported on *Appendix A*, form a system of

three-dimensional time-dependent non-linear partial differential equations. Once a mathematical model is selected, it is necessary to identify the spatial domain in order to analyze turbulent phenomena. After inserting appropriate initial/boundary conditions, the DNS provides information about velocity and pressure in the instantaneous flow field, stored in a numerical database that represent the core of the model. The database is used for quantitative and qualitative studies about wall-bounded turbulence, in terms of statistical correlations, energy budget and vortical structures, referred to the post-processing phase.

5.3 Numerical method

Based on the Navier-Stokes equations for an incompressible viscous fluid in non-dimensional conservative form, as in Eqs. (A.9) and (A.10), a mixed spectral-finite difference scheme has been developed.

The velocity field $u_i = (u,v,w)$ is considered periodic in the streamwise $x-$ and the spanwise $z-$directions; moreover, Eqs. (A.9) and (A.10) are Fourier transformed accordingly:

$$\frac{\partial \hat{u}}{\partial t} + ik_x\left(\widehat{u^2}\right) + \frac{\partial\left(\widehat{uv}\right)}{\partial y} + ik_z\left(\widehat{uw}\right) + ik_x\hat{p} = \frac{1}{\mathrm{Re}}\left(\frac{\partial^2\hat{u}}{\partial y^2} - k^2\hat{u}\right) \tag{5.1a}$$

$$\frac{\partial \hat{v}}{\partial t} + ik_x\left(\widehat{vu}\right) + \frac{\partial\left(\widehat{v^2}\right)}{\partial y} + ik_z\left(\widehat{vw}\right) + \frac{\partial\hat{p}}{\partial y} = \frac{1}{\mathrm{Re}}\left(\frac{\partial^2\hat{v}}{\partial y^2} - k^2\hat{v}\right) \tag{5.1b}$$

$$\frac{\partial \hat{w}}{\partial t} + ik_x\left(\widehat{wu}\right) + \frac{\partial\left(\widehat{wv}\right)}{\partial y} + ik_z\left(\widehat{w^2}\right) + ik_x\hat{p} = \frac{1}{\mathrm{Re}}\left(\frac{\partial^2\hat{w}}{\partial y^2} - k^2\hat{w}\right) \tag{5.1c}$$

$$ik_x\hat{u} + \frac{\partial\hat{v}}{\partial y} + ik_z\hat{w} = 0 \tag{5.2}$$

where ($\mathrm{Re} = u_\tau h/\nu$) is the Reynolds number referred as the friction velocity u_τ and the channel half-width h, the subscript "$\wedge$" indicates variables in Fourier space and $k^2 = k_x^2 + k_z^2$.

The non-linear terms in the momentum Eqs. (5.1a-c) are evaluated pseudospectrally by anti-transforming velocities back in physical space to perform the products (using *Fast Fourier Transforms*). To avoid aliasing errors in transforming the results back to Fourier space, the "$3/2$" rule has been enforced [11].

The scheme uses a mesh staggered along $y-$direction, that enhances the conservation properties of the numerical discretization. Moreover, using a $y-$direction staggered mesh, the pressure is collocated in the center of each cell and all three components of the velocity are collocated at the same point in the center of the side of the cell orthogonal to $y-$axes. Thus, due to steepest gradients near the walls, a stretching law of hyperbolic tangent type has been introduced for the grid points along $y-$direction:

$$y_{str} = Py + (1-P)\left(1 - \frac{\tanh\left[Q(1-y)\right]}{\tanh Q}\right) \tag{5.3}$$

where y indicates the uniform grid and P, Q are two parameters of the points-distribution.

For time advancement, a third-order Runge-Kutta procedure has been implemented; for each Fourier mode, one has in index notation:

$$\begin{aligned}\frac{\hat{u}_i^{(l)} - \hat{u}_i^{(l-1)}}{\Delta t} &= \alpha_l D\left(\hat{u}_i^{(l-2)}\right) + \beta_l D\left(\hat{u}_i^{(l-1)}\right) - \gamma_l C\left(\hat{u}_i^{(l-1)}\right) + \\ &- \delta_l C\left(\hat{u}_i^{(l-2)}\right) - (\alpha_l + \beta_l)\frac{\partial \hat{p}^{(l)}}{\partial x_i}\end{aligned} \tag{5.4}$$

where $l = 1, 2, 3$ denotes the Runge-Kutta sub-steps and:

$$D\left(\hat{u}_i\right)=\frac{1}{\mathrm{Re}}\partial_j\partial_j u_i=\frac{1}{\mathrm{Re}}\left(\frac{\partial^2\hat{u}_i}{\partial y^2}-k^2\hat{u}_i\right) \tag{5.5}$$

$$C\left(\hat{u}_i\right)=\partial_j\left(\widehat{u_i u_j}\right)=ik_x\left(\widehat{u_i u}\right)+\frac{\partial\left(\widehat{u_i v}\right)}{\partial y}+ik_z\left(\widehat{u_i w}\right) \tag{5.6}$$

are, respectively, the diffusive and convective terms; both are treated explicitly and $\alpha_l,\beta_l,\gamma_l,\delta_l$ are constant values:

$$\alpha_1=\frac{4}{15},\ \alpha_2=\frac{1}{15},\ \alpha_3=\frac{1}{6} \tag{5.7a}$$

$$\beta_1=\frac{4}{15},\ \beta_2=\frac{1}{15},\ \beta_3=\frac{1}{6} \tag{5.7b}$$

$$\gamma_1=\frac{8}{15},\ \gamma_2=\frac{5}{12},\ \gamma_3=\frac{3}{4} \tag{5.7c}$$

$$\delta_1=0,\ \delta_2=-\frac{17}{60},\ \delta_3=-\frac{5}{12} \tag{5.7d}$$

$$\sum_{l=1}^{3}\left(\alpha_l+\beta_l\right)=\sum_{l=1}^{3}\left(\gamma_l+\delta_l\right)=1 \tag{5.7e}$$

Time advancement procedure is coupled with the fractional-step method: considering Eq. (5.4), the pressure is interpreted as a projection operator, so velocity and pressure fields are decoupled. At each sub-step l and for each Fourier mode i, an intermediate velocity field is introduced (signed by superscript "*"):

$$\frac{\hat{u}_i^{*(l)}-\hat{u}_i^{(l-1)}}{\Delta t}=\alpha_l D\left(\hat{u}_i^{(l-2)}\right)+\beta_l D\left(\hat{u}_i^{(l-1)}\right)-\gamma_l C\left(\hat{u}_i^{(l-1)}\right)-\delta_l C\left(\hat{u}_i^{(l-2)}\right) \tag{5.8a}$$

Since the intermediate velocity field does not satisfy the incompressibility constraint, the pressure is used to project the intermediate velocity field into a divergence-free one by solving a Poisson problem:

$$\nabla^2 \hat{p}^{(l)} = \frac{1}{\Delta t\left(\alpha_l + \beta_l\right)}\left(\frac{\partial \hat{u}^{*(l)}}{\partial x} + \frac{\partial \hat{v}^{*(l)}}{\partial y} + \frac{\partial \hat{w}^{*(l)}}{\partial z}\right) \tag{5.9}$$

The pressure is obtained by solving Eq. (5.9) and the actual values of the velocity field are computed by using the following:

$$\frac{\hat{u}_i^{(l)} - \hat{u}_i^{*(l)}}{\Delta t} = -\left(\alpha_l + \beta_l\right)\frac{\partial \hat{p}^{(l)}}{\partial x_i} \tag{5.10}$$

No-slip boundary conditions at the walls and periodic conditions in the streamwise and spanwise directions have been applied to the velocity, while a Neumann-type boundary condition has been used for the pressure.

5.4 Computing systems

The numerical simulations have been executed on two specially-assembled hybrid CPU/GPU computing systems, named respectively *FDL-Tesla-1* and *FDL-Tesla-2*.

FDL-Tesla-1 (Fig. 5.1) includes an Intel Core i7 processor at 2.66 GHz, 12 GB of DDR3 RAM and 1 NVIDIA Tesla C-1060 board, based on NVIDIA CUDA technology. The system is also equipped with 1 NVIDIA GeForce GTS 240 with 1GB of GDDR3 memory at 70.4 GB/s bandwidth, while the total number of cores is equal to 112, at 675 MHz. The GeForce board is mainly used for visualization. The storage unit is equipped with 5 hard drives at 7200 rpm and 1 hard drive at 10000 rpm with a total capacity of 2.5 TB.

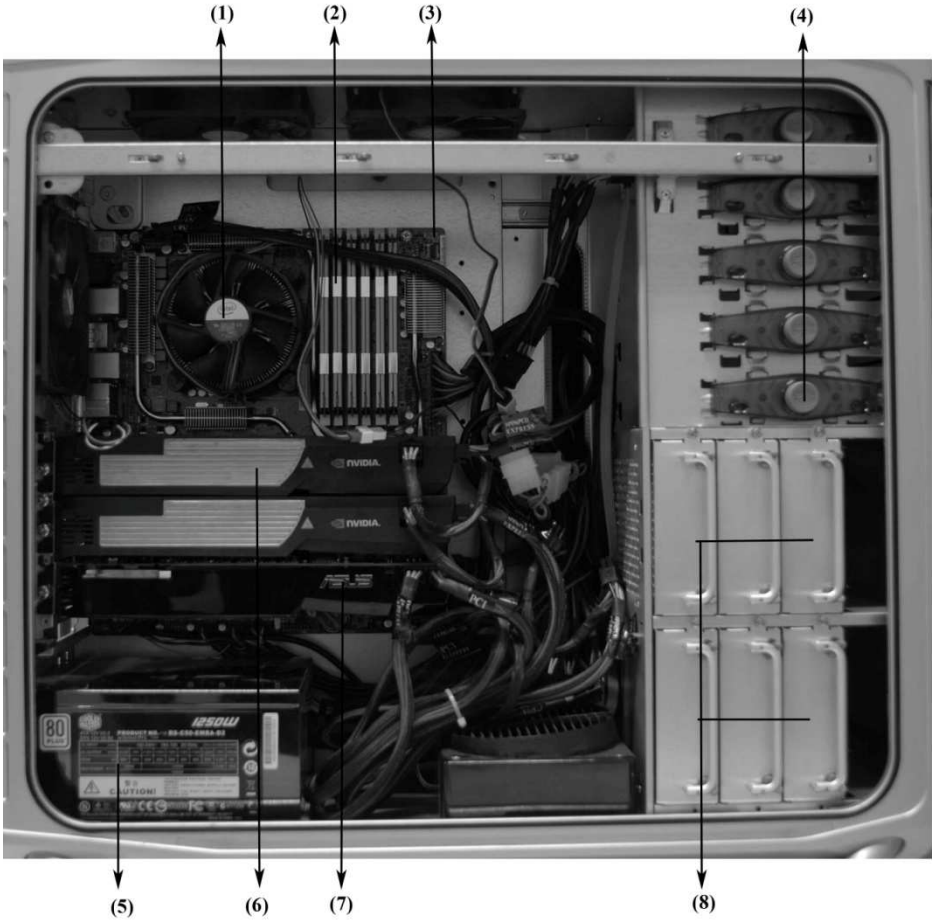

Figure 5.1 – FDL-Tesla-1 computing system: (1) Intel Core i7 (under fan); (2) RAM; (3) motherboard; (5) optical drives; (5) power supply; (6) NVIDIA Tesla C-1060; (7) NVIDIA GeForce GTS 240; (8) hard drives.

FDL-Tesla-2 (Fig. 5.2) includes 2 Intel Xeon 5660 at 2.8 GHz, 48 GB of DDR3 RAM and 3 NVIDIA Tesla C-1060 boards. The system is also equipped with 1 NVIDIA GeForce GTS 450 with 1 GB of GDDR5 memory at 57.7 GB/s bandwidth, while the total number of cores is equal to 192, at 1804 MHz. Also in this case, the GeForce board is mainly used for visualization. The storage unit is equipped with 5 hard drives at 7200 rpm with a total capacity of 5 TB.

Each Tesla board can handle 933 GFLOP/s of single-precision floating point processing, is equipped with 4 GB of GDDR3 memory at 102 GB/s bandwidth and contains 30 multiprocessors; each multiprocessor consists of 8 scalar single-precision floating-point processor cores, 1 double-precision floating-point unit and 16 kB of

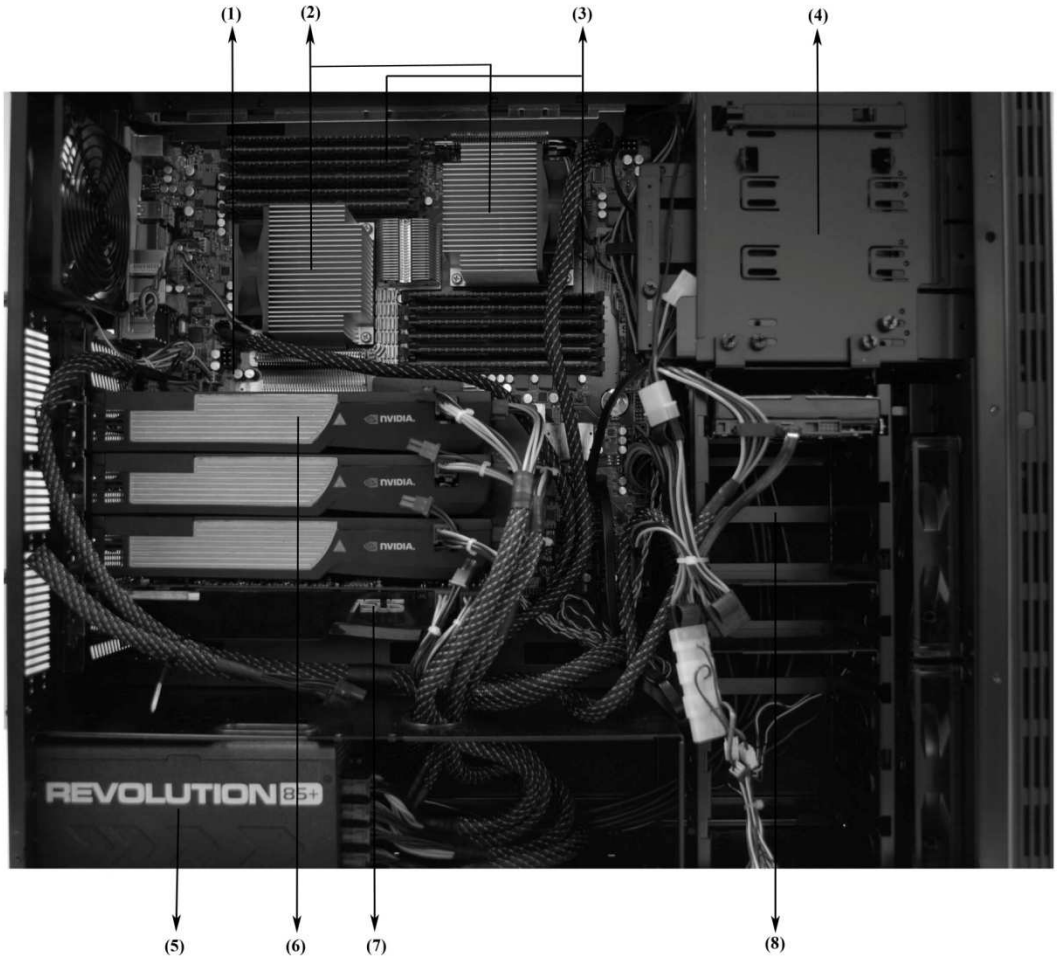

Figure 5.2 – FDL-Tesla-2 computing system: (1) motherboard; (2) Intel Xeon 5660 (under coolers); (3) RAM; (4) optical drives; (5) power supply; (6) NVIDIA Tesla C-1060; (7) NVIDIA GeForce GTS 450; (8) hard drives.

shared memory for threads cooperation. The total number of cores is equal to 240, at 1.3 GHz of processor-clock.

5.5 Parallelization strategies for multicore architectures

5.5.1 The sequential implementation of the Navier-Stokes solver

The Navier-Stokes solver is based on the equations described in the previous section. The implementation of the numerical solver follows these phases:

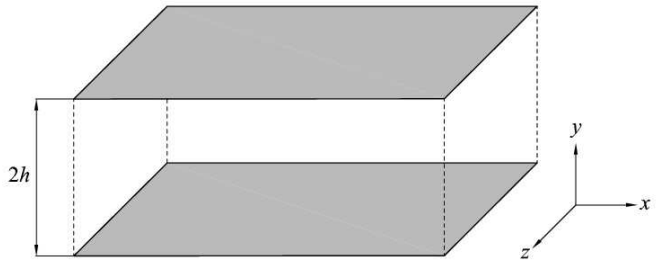

Figure 5.3 – The computational domain: a plane channel.

- pre-processing for the domain discretization and the computation of flow geometry;
- the numerical integration of the Navier-Stokes equations,
- post-processing for numerical databases analysis.

The plane channel is characterized by a simple geometry, consisting of two solid walls (Fig. 5.3), in the coordinate system (x, y, z): the computational domain is discretized by using $N_x \times N_y \times N_z$ grid points along $x-, y-$ and $z-$directions.

The numerical integration of governing equation is based on the following main steps:

- assignment of initial conditions, regarding an equilibrium velocity profile, and of boundary conditions, that consists of no-slip type at the walls and periodic conditions in the streamwise and spanwise direction;
- computation of the intermediate velocity field $\hat{u}_i^{*(l)}$, given the velocity field in the spectral space: for each Runge-Kutta sub-step, 2D *Fast Fourier Transforms (FFTs)* have been applied along $x-$ and $z-$axis, in order to evaluate the non-linear terms of the momentum equation (Eq. (5.8));
- computation of the pressure field $\hat{p}^{(l)}$, given the intermediate velocity field $\hat{u}_i^{*(l)}$: it requires an implementation of a numerical method along $y-$direction for solving a Poisson problem;
- update of the velocity field $\hat{u}_i^{(l)}$ by implementing a third-order Runge-Kutta procedure.

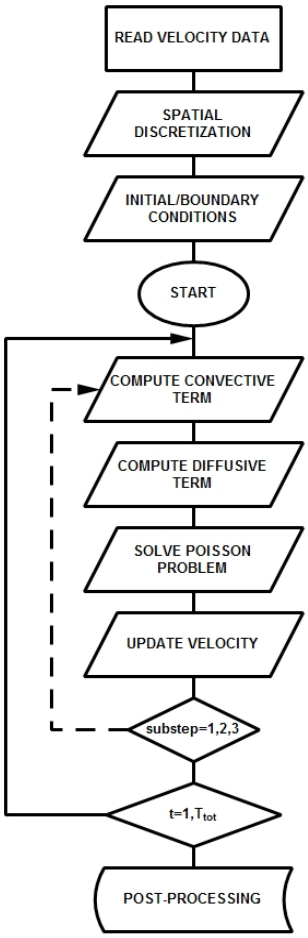

Figure 5.4 – Scheme of the serial implementation of the Navier-Stokes solver.

A synthetic view of the algorithm is given in Fig. 5.4: each block of the flow chart represents a specific routine, implemented by using Fortran 90 language.

The sequential numerical code provides a set of subroutines for allocating/deallocating arrays of data in the host memory space: in particular, those procedures use **allocate()/deallocate()** statements that dynamically provides storage for allocatable/deallocatable arrays, consisting of velocity components, pressure, diffusive and convective terms and other work arrays.

A set of subroutine, **axes_x(), axes_y(), axes_z()**, are devoted to spatial discretization: a regular grid is used along $x-$ and $z-$directions, while a block-regular grid is used along $y-$ direction, according to Eq. (5.3), in order to ensure local refinement near the walls. All data structures are initialized by a subroutine, named **initialization()** (List. 5.1).

```
call allocate_structure()
call initial_conditions()
call axes_x()
call axes_y()
call axes_z()
call initialization()
call initialize_fft(vel,Nx,Ny,Nz,fft_work)
```

Listing 5.1 – Scheme of the preliminary operations of the pre-processing phase

In order to compute the convective (non-linear) term, the following steps are performed:

- anti-transformation of the spectral velocity field to the physical space;
- computation of the velocity products terms;
- transformation of the product terms, in the physical space, back to the spectral one;
- computation of the convective term.

The 2D complex-to-real (C2R) and real-to-complex (R2C) Fourier transforms have been implemented by using the *FFTW* (*Faster Fourier Transform in the West*) library, developed at MIT by Matteo Frigo and Steven J. Johnson [123]. A 2D C2R FFT has been used to transform the velocity field from spectral to physical space by calling the **sfftw_execute_dft_c2r()** subroutine. For each point along $y-$direction and over each $xz-$plane it has been executed in order to compute the physical velocity components **uR, vR, wR**.

As reported in Eq. (5.6), it is necessary to compute products *uu*, *uv*, *uw*, *vv*, *wv*, *ww* for each point of the domain. Then, for each computed term, a 2D R2C FFT has been executed in order to transforming those products back to spectral space by calling **sfftw_execute_dft_r2c()** routine. Finally, the components of the convective term are computed (List. 5.2).

```
call bck_fft(u,v,w,uR,vR,wR)
call perform_products(uR,uR,uu,iy)
…
call perform_products(wR,wR,ww,iy)
call fwd_fft(uu,iy)
…
call fwd_fft(ww,iy)
```

```
call compute_convective_term(Cu,Cv,Cw)
```

Listing 5.2 – Scheme of the computation of the convective term.

According to Eq.(5.5), the components of the diffusive term are computed (List. (5.3)).

```
call compute_diffusive_term(Du,Dv,Dw)
```

Listing 5.3 – Scheme for the computation of the diffusive term.

Given all components of the velocity field and those related to the diffusive and convective terms, the components of the velocity field $\hat{u}_i^* \left(\hat{u}^*, \hat{v}^*, \hat{w}^* \right)$ are computed (List. (5.4)).

```
call compute_velstar(ustar,vstar,wstar)
```

Listing 5.4 – Scheme for the computation of the intermediate velocity field.

Since $\hat{u}_i^*$ does not satisfy the incompressibility constraint, it is necessary to implement a fractional-step method. This is done by implementing a set of subroutines that, given the intermediate velocity field, compute the right side of the Eq. (5.9) by solving a Poisson problem; the *Thomas algorithm* [124] is used to solve the tridiagonal system of linear equation over each $xy-$plane (List. 5.5) corresponding to the Poisson problem.

```
call poisson_rhs()
call compute_diagonals(d,dinf,dsup)
call thomas_algorithm(p)
```

Listing 5.5 – Scheme for the solution of the Poisson problem.

Once the pressure has been computed, a routine for updating velocities is executed, according to Eq. (5.10) (List. 5.6).

```
call update_velocity(ustar,vstar,wstar,p,substep,u,v,w)
```

Listing 5.6 – Scheme for the update of the velocity.

Then, a new iteration can start or data are stored into the DNS database for post-processing phase.

5.5.2 OpenMP implementation for multicore architectures

A parallel version of the numerical procedure suitable for multi-threading computing is developed to exploit multicore architectures of Intel i7 and Intel Dual-Xeon (on *FDL-Tesla-1* and *FDL-Tesla-2*, respectively).

In order to reduce the computing time, the Navier-Stokes solver, described in the previous paragraph, has been implemented in parallel by using OpenMP library [125], according to the steps here reported.

1. Computation of the velocity products terms (List. 5.7):

```
$ omp parallel do
do iy = 2, Ny-1
   call bck_fft(uR,iy)
   call bck_fft(vR,iy)
   call bck_fft(wR,iy)

   call perform_product(uR,vR,wR,uu,uv,uw,vv,vw,ww,iy)

   call fwd_fft(uu,iy)
   call fwd_fft(uv,iy)
   call fwd_fft(uw,iy)
   call fwd_fft(vv,iy)
   call fwd_fft(vw,iy)
   call fwd_fft(ww,iy)
end do
$ omp end parallel do
```

Listing 5.7 – Scheme for the parallel computation of velocity products terms.

2. Application of the de-aliasing filter (List. 5.8):

```
call compute_filter(kx_nz,kz_nz)
$ omp parallel do
do iz = 0, Nz-1
   do iy = 0, Ny-1
      do ix = Nx/2-kx_nz+1, Nx/2
         call filter_x(uu,uv,uw,vv,vw,ww)
      end do
   end do
```

```
end do
$ omp end parallel do
$ omp parallel do
do iz = 0, Nz/2-kz_nz, Nz/2+kz_nz-1
   do iy = 0, Ny-1
      do ix = 0, Nx/2
         call filter_z(uu,uv,uw,vv,vw,ww)
      end do
   end do
end do
$ omp end parallel do
```

Listing 5.8 – Scheme for the parallel computation of the de-aliasing filter.

3. Computation of the convective and the diffusive terms (List. 5.9):

```
$ omp parallel do
do iz = 0, Nz-1
   do iy = 2, Ny-1
      do ix = 0, Nx/2
         call compute_convective_term(Cu,Cv,Cw)
         call compute_diffusive_term(Lu,Lv,Lw)
      end do
   end do
end do
$ omp end parallel do
```

Listing 5.9 – Scheme for the parallel computation of the convective and diffusive terms.

4. Computation of the intermediate velocity field (List. 5.10):

```
$ omp parallel do
do iz = 0, Nz-1
   do iy = 1, Ny
      do ix = 0, Nx
         call compute_intermediate_velocity()
      end do
   end do
end do
```

Listing 5.10 – Scheme for the computation of the intermediate velocity field.

5. Computation of the pressure field. In this case, in order to access in an efficient way the host memory, a different type of data structures memorization is used for solving the Poisson problem: the arrays referred to

diagonals (main diagonal, upper diagonal and lower diagonal) are duplicated in order to solve "two" tridiagonal systems, one for the real part and one for the imaginary part. After solving the tridiagonal systems, the pressure array should be re-organized as the velocity data structure, alternating real and imaginary parts, i.e. each grid point has the corresponding pressure and velocity complex values (List. 5.11).

```
$ omp parallel do
do iz = 0, Nz-1
   do ix = 0, Nx/2
      do iy = 1, nymax-1
         call compute_poisson_rhs(rhs)
      end do

      do iy = 1, nymax-1
         diag(iy) = d(ix,iy,iz)
         diag(iy+nymax-1) = d(ix,iy,iz)
      end do
      do iy = 1, nymax-1
         dinf(iy) = low_d(iy)
         dinf(iy+nymax-1) = low_d(iy)
      end do
      do iy = 1, nymax-1
         dsup(iy) = up_d(iy)
         dsup(iy+nymax-1) = up_d(iy)
      end do
      do iy = 1, nymax-1

      call solve_tridiagonal_sys(dinf,d,dsup,
                                 rhs,pv,2*(nymax-1))

         do iy = 1, nymax-1
            p(2*ix,iy,iz) = pv(iy)
            p(2*ix+1,iy,iz) = pv(iy+nymax-1)
         end do
      end do
   end do
end do
$ omp end parallel do
```

Listing 5.11 – Scheme for the solution of the Poisson problem.

6. Finally, for each Runge-Kutta sub-step, the velocity is updated (List. 5.12).

```
$ omp parallel do
do iz = 0, Nz-1
```

```
   do iy = 2, Ny-1
      do ix = 0, Nx/2
         update_velocity_field(ustar,vstar,wstar,p,sub,u,v,w)
      end do
   end do
end do
$ omp end parallel do
```

Listing 5.12 – Scheme for the update of the velocity.

5.6 Parallelization strategies for manycore architectures

5.6.1 Single-GPU implementation of the Navier-Stokes solver

This section shows a novel parallel implementation for integrating the Navier-Stokes equations on GPU architectures by using the CUDA 3.2 library (http://developer.nvidia.com/cuda-toolkit-32-downloads), while the computational code is written using C language. A synthetic scheme of the numerical procedure is given in Fig. 5.5. Considering a plane channel problem whose data can completely reside on a single-GPU and by recalling the numerical method discussed in Section 5.3, the host is devoted to:

- read the numerical data related to the initial velocity field;
- allocate the required memory space on the device;
- transfer initial data from the host memory to the device memory;
- launch a set of kernels in order to integrate the governing equations that are executed on the device;
- transfer data related to the updated velocity field from the device memory to the host memory;
- store the new data in the DNS database.

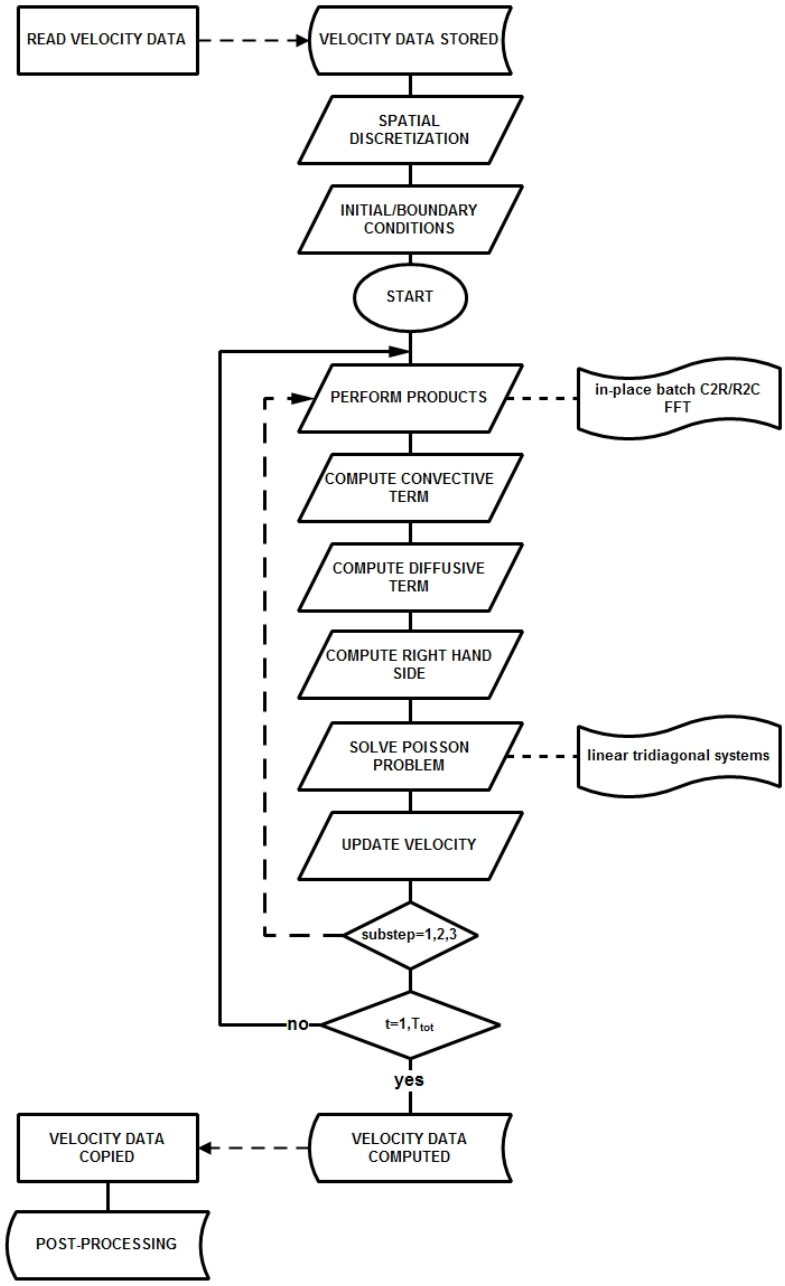

Figure 5.5 – Scheme for the GPU implementation of the Navier-Stokes solver.

CUDA 3.2 provides functions to allocate, deallocate and copy device memory; in particular, data structures are allocated using **cudaMalloc()**, while data transfer of **u,v,w** from host (**h_u,h_v,h_w**) to device (**d_u,d_h,d_w**) memory are managed by **cudaMemcpy()** (List. (5.13)).

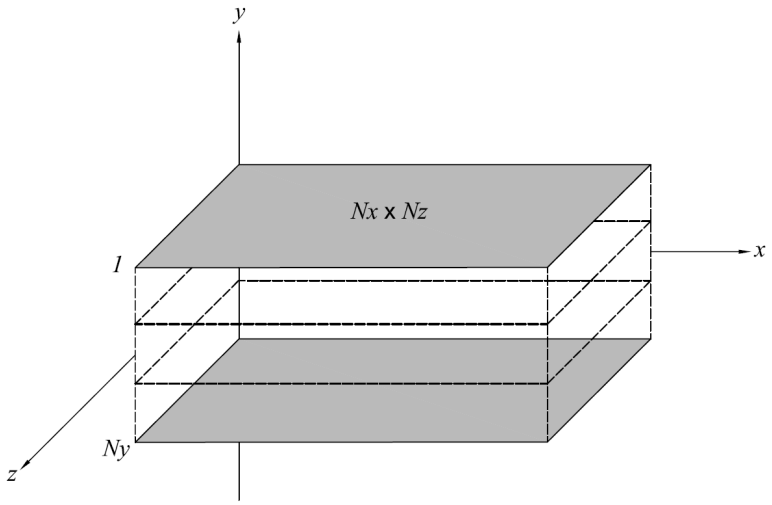

Figure 5.6 – Representation of the *xz*-planes along *y*-direction.

```
cudaMalloc();
cudaMemcpy(d_u, h_u, cudaMemcpyHostToDevice, size);
cudaMemcpy(d_v, h_v, cudaMemcpyHostToDevice, size);
cudaMemcpy(d_w, h_w, cudaMemcpyHostToDevice, size);
```

Listing 5.13 – Scheme of data structures allocation and host-to-device transfer.

For a given velocity field stored on the memory device, batched 2D complex-to-real (C2R) and real-to-complex (R2C) Fourier transforms have been implemented using the CUFFT library provided by NVIDIA [126]. In particular, an in-place batched 2D R2C FFT has been used in order to transform the velocity field from spectral to physical space using a batch of N_y $xz-$planes. It consists on performing a certain number (equal to the total number of grid points along $y-$direction) of Fourier transforms over each $xz-$plane, obtained by cutting the computational domain along $y-$direction, as represented in Fig. 5.6.

After that, each product between the mutual components of the velocity field has been computed by a kernel, named **product_kernel**; then, for each product $\widehat{uu}$, $\widehat{uv}$, $\widehat{uw}$, $\widehat{vv}$, $\widehat{vw}$, $\widehat{ww}$, an in-place batched 2D C2R FFT has been executed in order to transform those quantities back to spectral space. Listing 5.14 shows the pseudo-code that describes those steps.

```
cufftExecC2R(batchBackPlan, (cufftComplex *) d_u, (float *) d_u);
cufftExecC2R(batchBackPlan, (cufftComplex *) d_v, (float *) d_v);
cufftExecC2R(batchBackPlan, (cufftComplex *) d_w, (float *) d_w);

product_kernel<<<blocksPerGrid, threadsPerBlock>>>(d_u, d_v, d_w,
d_uu, d_uv, d_uw, d_vv, d_vw, d_ww);

cufftExecR2C(batchFwdPlan, (float *)d_uu, (cufftComplex *)d_uu);
…
cufftExecR2C(batchFwdPlan, (float *)d_ww, (cufftComplex *)d_ww);
```

Listing 5.14 – Scheme of FFT operations for the solution on non-linear terms.

For each Runge-Kutta sub-step, the diffusive *D* and the convective *C* terms of Eq. (5.8) are computed by using two kernels, **diffusive_kernel** and **convective_kernel**, properly designed in order to minimize memory access by using memory coalescing and increasing the computation/memory ratio. The intermediate velocity field has been calculated using a kernel, named **velstar_kernel**; given all components of the velocity field and all components of diffusive and convective terms, the three components of the intermediate velocity field (**d_ustar, d_vstar, d_wstar**) according to Eq. (5.8a) (List. 5.15) are computed.

```
convective_kernel<<<blocksPerGrid, threadsPerBlock>>>(d_uu, d_uv,
d_uw, d_vv, d_wv, d_ww, d_nu, d_nv, d_nw);

diffusive_kernel<<<blocksPerGrid, threadsPerBlock>>>(d_u,d_v,d_w,
d_lu,d_lv,d_lw);

velstar_kernel<<<blocksPerGrid, threadsPerBlock>>>(d_u ,d_v, d_w,
d_nu, …,d_lu, …,d_ustar, d_vstar,d_wstar);
```

Listing 5.15 – Scheme for the calculation of the intermediate velocity field.

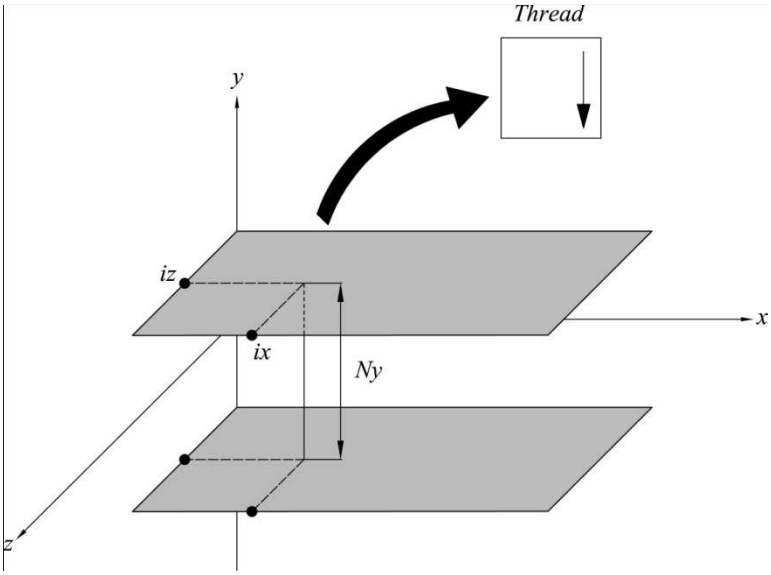

Figure 5.7 – Partitioning of multiple tridiagonal systems among GPU threads.

As said in the previous section, $\hat{u}_i^{*(l)}$ does not satisfy the incompressibility constraint, so it is necessary to implement a fractional-step method. This is done by implementing a set of kernels that, given the intermediate velocity field, compute the right side of Eq. (5.9) and solve a Poisson problem.

In order to solve a Poisson problem, $N_x \times N_z$ tridiagonal systems of linear equations are solved on GPU by redesign the Thomas algorithm. In particular, each thread solves a tridiagonal linear system by using a modified Thomas algorithm, optimized for the GPU (re-use of GPU registers and coalesced memory data access) (Fig. 5.7). Regarding the principal diagonal of the coefficient matrix of the tridiagonal system, for threads belonging to the same warp memory coalescing for load/store instructions is guaranteed. Finally, memory access for the upper and lower diagonals are optimized by storing them in the device constant memory (List. 5.16).

```
tn_kernel<<<blocksPerGrid, threadsPerBlock>>>(rk, h_dt, d_tn,
d_ustar, d_vstar, d_wstar);

solve_poisson_kernel<<<blocksPerGrid,threadsPerBlock>>>(d_dpri,
d_tn, d_p, d_work);
```

Listing 5.16 – Scheme for the solution of the Poisson problem.

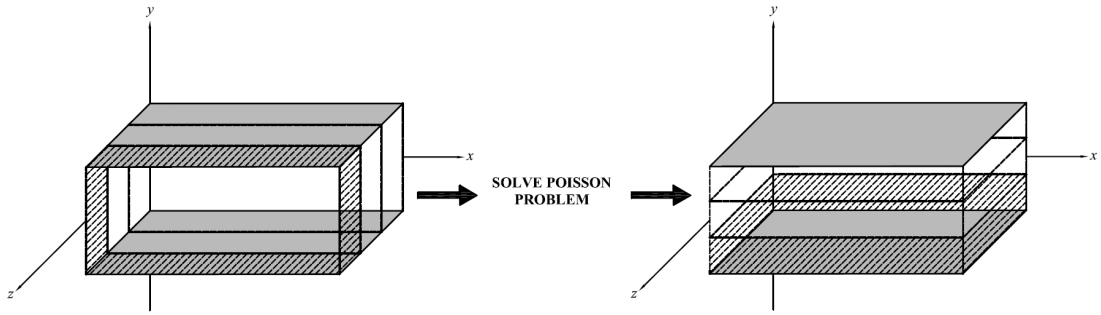

Figure 5.8 – Representation of the slicing and re-slicing computational domain.

After pressure component is computed, a kernel for updating of the velocity field (**update_velocity_kernel**) has been executed, according to Eq. (5.10); then, a device to host data transfer of computed values is performed both for a new iteration or for storing the velocity field into the DNS database (List. 5.17).

```
update_velocity_kernel<<<blocksPerGrid,
threadsPerBlock>>>(d_ustar, d_vstar, d_wstar, d_p, rk, h_dt, d_u,
d_v, d_w);
cudaMemcpy(h_u, d_u, cudaMemcpyDeviceToHost, size);
cudaMemcpy(h_v, d_v, cudaMemcpyDeviceToHost, size);
cudaMemcpy(h_w, d_w, cudaMemcpyDeviceToHost, size);
```

Listing 5.17 – Scheme for the updating of the velocity field and for the device-to-host transfer.

5.6.2 Multi-GPU implementation of the Navier-Stokes solver

In this section, a novel multi-GPU implementation is described that is needed when numerical data cannot completely reside on a single-GPU. In this case, OpenMP library is used for driving devices available on the same computational node, such as in *FDL-Tesla-2*. The code has been implemented such that each thread is responsible for managing its corresponding device, for allocating data on the device and for executing kernels for data transfer from and to the device.

In order to solve the problem on several devices, the most efficient way consists of partitioning data along $y-$axes for computing the intermediate velocity field $\hat{u}^*$ and updating the actual velocity field and partitioning data along $z-$axes for solving

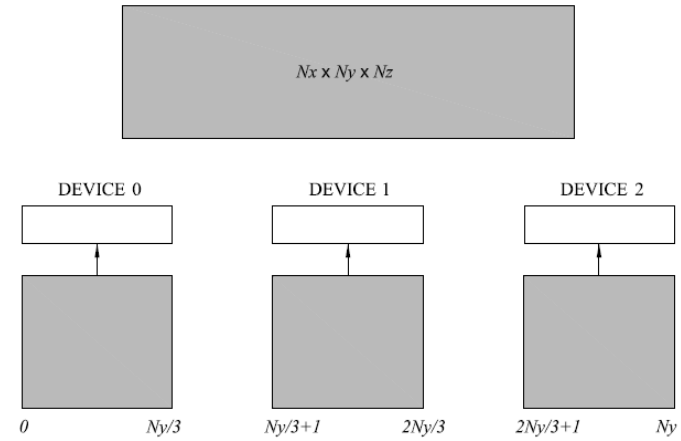

Figure 5.9 – Representation of domain decomposition considering 3 available devices.

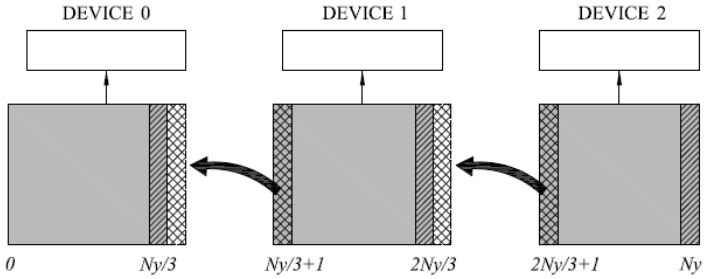

Figure 5.10 – Local data transfers of partitions' boundaries.

the tridiagonal system of linear equations. Figure 5.8 describes, in particular, how the slicing and the re-slicing of the computational domain is performed in order to compute the actual velocity field: at first, a slicing of the computational domain along $z-$axes is required for solving the Poisson problem; then, a re-slicing of the same domain along $y-$axes is performed in order to update the velocity.

With regards to each partition (Fig. 5.9) and in order to compute some intermediate terms, such as the rhs of the Poisson problem (Eq. (5.9)), a local data transfer of boundary layers has to be performed; as represented in Fig. 5.10, each partition p_i needs neighbouring data computed by the device corresponding to partition p_{i+1}, for $(i = 1, N_{DEV} - 1)$, with N_{DEV} number of available devices.

An overall scheme of the numerical algorithm is shown in Fig. 5.11: it is based essentially on the single-GPU implementation, except for the OpenMP library and

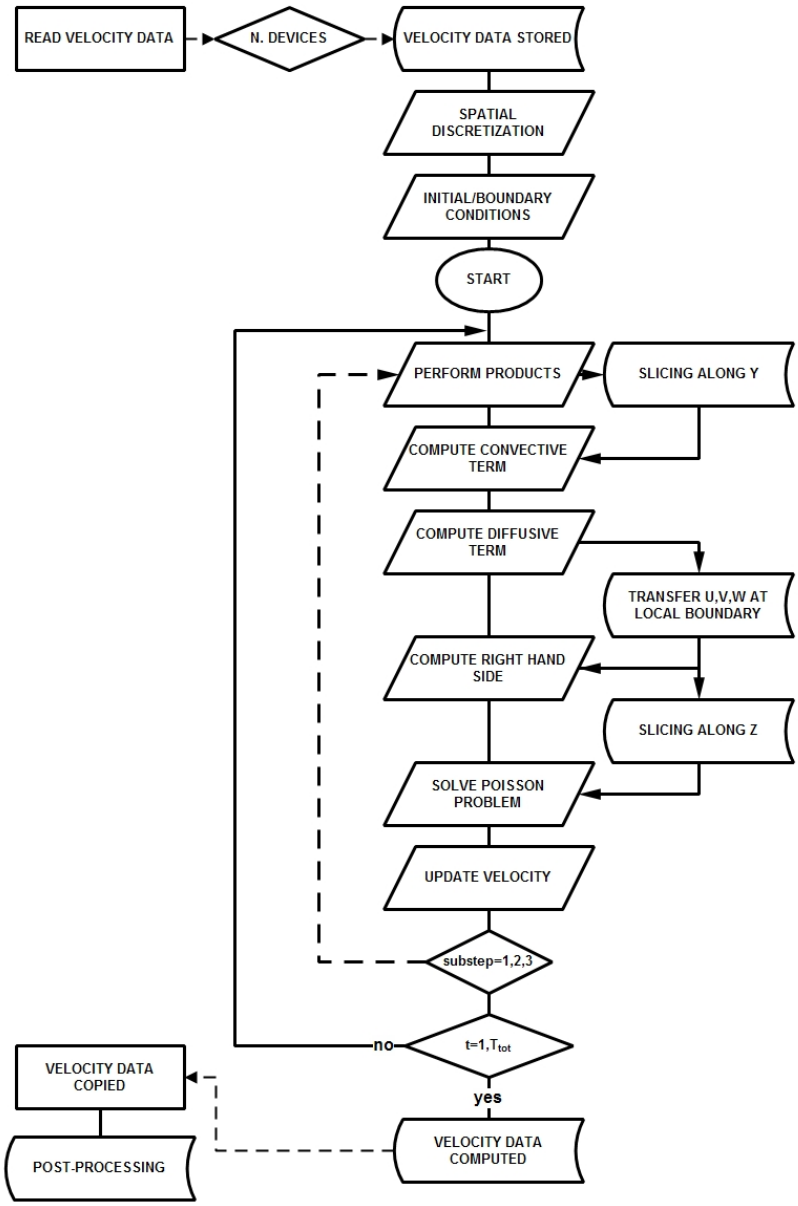

Figure 5.11 – Scheme for the multi-GPU implementation of the Navier-Stokes solver.

the slicing/re-slicing operations among partitions. Listing (5.18) shows the main of the parallel code.

```
void simulation_step
\\ compute C, D and Vstar

velocity_product()
convective_term()
diffusive_term()
velstar_computation()
```

```
\\ boundaries copy from device p+1 to host
nx = d-> slice_xz.nx
ny = d-> slice_xz.ny
nz = d-> slice_xz.nz
cutilSafeCall(cudaMemcpy(d->h->us_buf[d->id], d->us + nx*nz,
nx*nz*sizeof(float2), cudaMemcpyDeviceToHost));
cutilSafeCall(cudaMemcpy(d->h->vs_buf[d->id], d->vs + nx*nz,
nx*nz*sizeof(float2), cudaMemcpyDeviceToHost));
cutilSafeCall(cudaMemcpy(d->h->ws_buf[d->id], d->ws + nx*nz,
nx*nz*sizeof(float2), cudaMemcpyDeviceToHost));

\\ boundaries data transfer from host to device p
#pragma omp barrier
cutilSafeCall(cudaMemcpy(d->us + (ny-1)*nx*nz,
d->h->us_buf[d->id+1], nx*nz*sizeof(float2),
cudaMemcpyHostToDevice));
cutilSafeCall(cudaMemcpy(d->vs + (ny-1)*nx*nz,
d->h->vs_buf[d->id+1], nx*nz*sizeof(float2),
cudaMemcpyHostToDevice));
cutilSafeCall(cudaMemcpy(d->ws + (ny-1)*nx*nz,
d->h->ws_buf[d->id+1], nx*nz*sizeof(float2),
cudaMemcpyHostToDevice));

\\ compute rhs tridiagonal system
\\ slicing along y-direction (xz-plane)
if (d->id == 0) {
cudaMemcpy(d->h->tn_xz + d->slice_xz.memory_offset,
d->tn, d->slice_xz.nx*d->slice_xz.nz*(d->slice_xz.ny-1)
*sizeof(float2), cudaMemcpyDeviceToHost);
} else {
cudaMemcpy(d->h->tn_xz + d->slice_xz.memory_offset +
d->slice_xz.nx*d->slice_xz.nz, d->tn + d->slice_xz.nx*
d->slice_xz.nz, d->slice_xz.nx*d->slice_xz.nz*(d->slice_xz.ny-2)
*sizeof(float2), cudaMemcpyDeviceToHost);
}

#pragma omp barrier
for (int i = 0; i < p->NY ; i++) {
memcpy(d->h->tn_xy + d->slice_xy.memory_offset +
i*d->slice_xy.nx*d->slice_xy.nz,
d->h->tn_xz + i*d->slice_xz.nx*d->slice_xz.nz +
d->slice_xy.domain_offset*d->slice_xy.nx ,
d->slice_xy.nx*d->slice_xy.nz*sizeof(float2));
}

#pragma omp barrier
cudaMemcpy(d->tn, d->h->tn_xy + d->slice_xy.memory_offset,
d->slice_xy.nx*d->slice_xy.nz*d->slice_xy.ny*sizeof(float2),
cudaMemcpyHostToDevice);
```

```
\\ solve tridiagonal system
solve_tridag(p, d);
cudaMemcpy(d->h->p_xy + d->slice_xy.memory_offset, d->p,
d->slice_xy.nx*d->slice_xy.nz*d->slice_xy.ny*sizeof(float2),
cudaMemcpyDeviceToHost);
#pragma omp barrier

\\ re-slicing from xy-plane to xz-plane pattern
for (int i = 0; i < p->NY; i++) {
memcpy(d->h->p_xz + i*d->slice_xz.nx*d->slice_xz.nz +
d->slice_xy.domain_offset*d->slice_xy.nx,
d->h->p_xy + d->slice_xy.memory_offset + i*d->slice_xy.nx*
d->slice_xy.nz,
d->slice_xy.nx*d->slice_xy.nz*sizeof(float2));
d->h->p_xz[i*d->slice_xz.nx*d->slice_xz.nz].x = 0.0;
d->h->p_xz[i*d->slice_xz.nx*d->slice_xz.nz].y = 0.0;
}

#pragma omp barrier
cudaMemcpy(d->p, d->h->p_xz + d->slice_xz.memory_offset,
d->slice_xz.nx*d->slice_xz.nz*d->slice_xz.ny*sizeof(float2),
cudaMemcpyHostToDevice);
\\ update velocity field
update_velocity(p, d);
\\ boundaries data transfer from device p+1 to host
cutilSafeCall(cudaMemcpy(d->h->us_buf[d->id], d->u + nx*nz,
nx*nz*sizeof(float2), cudaMemcpyDeviceToHost));
cutilSafeCall(cudaMemcpy(d->h->vs_buf[d->id], d->v + nx*nz,
nx*nz*sizeof(float2), cudaMemcpyDeviceToHost));
cutilSafeCall(cudaMemcpy(d->h->ws_buf[d->id], d->w + nx*nz,
nx*nz*sizeof(float2), cudaMemcpyDeviceToHost));

\\ boundaries data transfer from host to device p
#pragma omp barrier
if (d->id < d->h->num_devices-1) {
cutilSafeCall(cudaMemcpy(d->u + (ny-1)*nx*nz,
d->h->us_buf[d->id+1], nx*nz*sizeof(float2),
cudaMemcpyHostToDevice));
cutilSafeCall(cudaMemcpy(d->v + (ny-1)*nx*nz,
d->h->vs_buf[d->id+1], nx*nz*sizeof(float2),
cudaMemcpyHostToDevice));
cutilSafeCall(cudaMemcpy(d->w + (ny-1)*nx*nz,
d->h->ws_buf[d->id+1], nx*nz*sizeof(float2),
cudaMemcpyHostToDevice));
}

\\ boundaries data transfer from device p+1 to host
#pragma omp barrier
cutilSafeCall(cudaMemcpy(d->h->us_buf[d->id],
d->u + (ny-2)*nx*nz, nx*nz*sizeof(float2),
cudaMemcpyDeviceToHost));
```

```
cutilSafeCall(cudaMemcpy(d->h->vs_buf[d->id],
d->v + (ny-2)*nx*nz, nx*nz*sizeof(float2),
cudaMemcpyDeviceToHost));
cutilSafeCall(cudaMemcpy(d->h->ws_buf[d->id],
d->w + (ny-2)*nx*nz, nx*nz*sizeof(float2),
cudaMemcpyDeviceToHost));

\\ boundaries data transfer from host to device p
#pragma omp barrier
if (d->id > 0) {
cutilSafeCall(cudaMemcpy(d->u, d->h->us_buf[d->id-1],
nx*nz*sizeof(float2), cudaMemcpyHostToDevice));
cutilSafeCall(cudaMemcpy(d->v, d->h->vs_buf[d->id-1],
nx*nz*sizeof(float2), cudaMemcpyHostToDevice));
cutilSafeCall(cudaMemcpy(d->w, d->h->ws_buf[d->id-1],
nx*nz*sizeof(float2), cudaMemcpyHostToDevice));
}
}
```

Listing 5.18 – Scheme of the main program.

With reference to List. 5.18, the main steps of the computational procedure are here discussed.

1. After domain decomposition and reading velocity data, stored into the host memory, each thread executes a host-to-device memory copy to its corresponding slice (List. 5.19).

```
slice s = d->slice_xz;
update_velocity_kernel<<<blocksPerGrid, d-
>threadsPerBlock>>>(s.nx, s.ny, s.nz, d->us, d->vs, d->ws,
d->p, p->rk[p->step], p->dt, d->u, d->v, d->w);
cutilSafeCall(cudaMemcpy(d->h->us_buf[d->id], d->us + nx*nz,
nx*nz*sizeof(float2), cudaMemcpyDeviceToHost));
cutilSafeCall(cudaMemcpy(d->h->vs_buf[d->id], d->vs + nx*nz,
nx*nz*sizeof(float2), cudaMemcpyDeviceToHost));
cutilSafeCall(cudaMemcpy(d->h->ws_buf[d->id], d->ws + nx*nz,
nx*nz*sizeof(float2), cudaMemcpyDeviceToHost));

s = d->slice_xz

cudaMemcpy(d->u, sizeof(float2),cudaMemcpyHostToDevice);
cudaMemcpy(d->v, sizeof(float2),cudaMemcpyHostToDevice);
cudaMemcpy(d->w, sizeof(float2),cudaMemcpyHostToDevice);
```

Listing 5.19 – Scheme for the host-to-device data transfer.

2. As in the single-GPU implementation, also in the multi-GPU one, each device executes, for each $xz-$plane, a batched 2D complex-to-real (C2R) and real-to-complex (R2C) Fourier transforms (List. 5.20).

```
cufftExecC2R(d->batchBckPlan, (cufftComplex *)d->ut,
(cufftReal *)d->ut);
cufftExecC2R(d->batchBckPlan, (cufftComplex *)d->vt,
(cufftReal *)d->vt);
cufftExecC2R(d->batchBckPlan, (cufftComplex *)d->wt,
(cufftReal *)d->wt);

product_kernel<<<blocksPerGrid, d->threadsPerBlock>>>(N,
scale,
d->ut, d->vt, d->wt, d->uu, d->uv,
d->uw, d->vv, d->wv, d->ww);

cufftExecR2C(d->batchFwdPlan, (cufftReal *)d->uu,
(cufftComplex *)d->uu);
cufftExecR2C(d->batchFwdPlan, (cufftReal *)d->uv,
(cufftComplex *)d->uv);
cufftExecR2C(d->batchFwdPlan, (cufftReal *)d->uw,
(cufftComplex *)d->uw);
cufftExecR2C(d->batchFwdPlan, (cufftReal *)d->vv,
(cufftComplex *)d->vv);
cufftExecR2C(d->batchFwdPlan, (cufftReal *)d->wv,
(cufftComplex *)d->wv);
cufftExecR2C(d->batchFwdPlan, (cufftReal *)d->ww,
(cufftComplex *)d->ww);
```

Listing 5.20 – Scheme for the computation of the product terms.

3. Each device executes a set of kernels, as listed in the following, for computing the convective and the diffusive terms (List. 5.21).

```
convective_kernel<<<blocksPerGrid,d->threadsPerBlock>>>
(s.nx, s.ny, s.nz, s.nx - p->kx_nz+1, s.nx-1, s.nz/2,
p->kz_nz, s.nz/2 + p->kz_nz - 1,
d->uu, d->uv, d->uw, d->vv,
d->wv, d->ww, d->nu, d->nv, d->nw);

diffusive_kernel<<<blocksPerGrid, d->threadsPerBlock>>>
(s.nx, s.ny, s.nz,
d->u, d->v, d->w,
d->lu, d->lv, d->lw);

velstar_kernel<<<blocksPerGrid, d->threadsPerBlock>>>
(s.nx, s.ny, s.nz, rk, dt, tau,
```

```
d->u, …, d->nu, …, d->lu,
d->ustar, d->vstar, d->wstar);
```

Listing 5.21 – Scheme for the computation of the convective and diffusive terms.

4. Before executing the kernel related to the fractional-step method, a clipping of the computational domain along $z-$direction is necessary, as represented in Fig. (5.17). Thus, each device executes the GPU modified *Thomas algorithm* for solving the Poisson problem (List. 5.22).

```
num_ele_z = ceil(Nz/num_devices)
for (i=0; i < h->num_devices; i++) {
h->d_data[i].slice_xy.nx = p->NX;
h->d_data[i].slice_xy.ny = p->NY;
h->d_data[i].slice_xy.nz = num_ele_z;
h->d_data[i].slice_xy.memory_offset =
i*num_ele_z*p->NX*p->NY;
h->d_data[i].slice_xy.domain_offset = i*num_ele_z;
}
tn_kernel<<<blocksPerGrid, d->threadsPerBlock>>>
(s.nx, s.ny, s.nz,
p->rk[p->step], p->dt,
d->tn, d->us, d->vs, d->ws);
solve_poisson_kernel <<<blocksPerGrid, d->threadsPerBlock>>>
(s.nx, s.ny, s.nz,
d->dpri, d->tn, d->p,
d->work);
```

Listing 5.22 – Scheme for the solution of the Poisson problem.

5. Before updating the velocity field, a re-slicing along $y-$direction is performed; thus, each device executes the kernel named **update_velocity_kernel** (List. 5.23).

```
slice s = d->slice_xz;
update_velocity_kernel<<<blocksPerGrid, d->threadsPerBlock>>>
(s.nx, s.ny, s.nz,
d->us, d->vs, d->ws,
d->p, p->rk[p->step], p->dt,
d->u, d->v, d->w);
cutilSafeCall(cudaMemcpy(d->h->us_buf[d->id], d->us + nx*nz,
nx*nz*sizeof(float2), cudaMemcpyDeviceToHost));
cutilSafeCall(cudaMemcpy(d->h->vs_buf[d->id], d->vs + nx*nz,
nx*nz*sizeof(float2), cudaMemcpyDeviceToHost));
cutilSafeCall(cudaMemcpy(d->h->ws_buf[d->id], d->ws + nx*nz,
```

```
nx*nz*sizeof(float2), cudaMemcpyDeviceToHost));
```

Listing 5.23 – Scheme for the updating of the velocity field and for the device-to-host transfer.

6. Finally, each thread transfers the new velocity field from the device memory to the host memory and stores data in the DNS database (List. 5.23).

Chapter 6

Results of the Simulations: Turbulence Statistics

6.1 Introduction

The evaluation of the behavior of fluctuations and pressure of fluid flow variables in wall-bounded turbulence is of fundamental interest in many engineering applications involving the development of efficient turbulence control techniques and closure models. For this reason, it is necessary to investigate the relationships between turbulence statistics, Reynolds-stress, dissipation and anisotropy budgets and, finally, to consider the relevant information that derives from coherent structures, especially with reference to the Reynolds number dependence.

The objective of the present chapter is to present the statistical-energetic results of the numerical simulations of a turbulent channel flow and to evaluate performances of the parallel implementations of the Navier-Stokes solver, described in *Chapter 5*, setting the Reynolds number to be $\mathrm{Re}_\tau = 200, 400, 600$.

The present chapter is organized as follows: Section 6.2 shows the validation of the numerical solver against existing results; Section 6.3 describes the rate budget of the Reynolds shear stress, kinetic energy, dissipation and anisotropy; Section 6.4

analyzes the turbulence statistics with respect to the Reynolds number dependence; finally, in Section 6.5 the evaluation of the performances of the parallel codes is described, both in terms of speedup and efficiency.

6.2 Validation against existing results

Besides the aspect of the strictly numerical reliability of the computational code, it is necessary to verify the ability of the mixed spectral-finite difference method adopted for the numerical integration of the governing equations to simulate the physical properties of turbulence in a plane channel. For this reason, a comparison of the computed results with numerical data obtained by other authors has to be consider.

The comparisons concern all the most important variables, in terms of turbulence intensities and mean flow properties, that allow to verify the reliability both of the spatial/temporal domain in wall turbulence analysis and statistical properties of the numerical databases simulated.

A first screening of the effectiveness of the numerical simulations may be done by determining the accuracy of the computational grid, in space and time, by computing two-point correlation coefficients of velocity fluctuations and energy spectra; furthermore, it is necessary to estimate the Kolmogorov microscale as required by [43].

After this step, it is necessary to evaluate mean flow properties, for a more complete description of the main characteristics of the simulated turbulent flows. Mean flow properties have been evaluated by calculating the bulk mean velocity U_b, the related Reynolds number Re_b, the mean centerline velocity $\overline{u}_c$ and the related Reynolds number Re_c as follows:

$$U_b = \frac{1}{2}\int_{-1}^{1} \overline{u} d\left(\frac{y}{h}\right) \tag{6.1}$$

Table 6.1 – Characteristic parameters of the simulations.

	Computational domain						*Grid points*			*Grid spacing*			
Re_τ	L_x	L_y	L_z	L_x^+	L_y^+	L_z^+	N_x	N_y	N_z	Δx^+	Δ_{yc}^+	Δ_{yw}^+	Δz^+
200	$4\pi h$	$2h$	$2\pi h$	256	181	256	2513	400	1257	9.82	3.87	0.25	6.91
180 [4]	$4\pi h$	$2h$	$4/3\pi h$	128	129	128	2262	360	754	17.7	6.4	0.05	6.9
400	$4\pi h$	$2h$	$2\pi h$	343	321	343	5026	800	2513	16.65	6.36	0.28	7.33
395 [4]	$2\pi h$	$2h$	πh	256	193	182	2482	790	1241	10.0	6.5	-	6.5
600	$4\pi h$	$2h$	$2\pi h$	512	451	512	7540	1200	3770	16.73	6.66	0.30	7.36
590 [4]	$2\pi h$	$2h$	πh	384	257	384	2482	790	1241	9.7	6.5	-	7.2

$$\mathrm{Re}_b = \frac{U_b h}{\nu} \tag{6.2}$$

$$\mathrm{Re}_c = \frac{\bar{u}_c h}{\nu} \tag{6.3}$$

Furthermore, both the values of $\bar{u}_c/U_b$ and C_{fb} are obtained from experimental correlations suggested by [40]:

$$\frac{\bar{u}_c}{U_b} = 1.28\left(2\,\mathrm{Re}_b\right)^{-0.0116} \tag{6.4}$$

$$C_{fb} = 0.073\left(2\,\mathrm{Re}_b\right)^{-0.25} \tag{6.5}$$

while the computed skin friction coefficient are calculated by using the related value of the shear stress at the wall, actually obtained in the computations as follows:

$$C_{fb} = \frac{2\tau_w}{\rho U_b^2} \tag{6.6}$$

$$C_{fc} = \frac{2\tau_w}{\rho \overline{u}_c^2} \tag{6.7}$$

The statistical analysis helps to describe more in detail the wall-bounded phenomena and, in particular, to understand the role of fluctuations of the wall flow variables and their effects in terms of coherent structures. It consists on the evaluation of the root-mean-square of the velocity fluctuations, the skewness and flatness factors and, finally, the Reynolds-shear stress within a plane channel.

Thus, for each database considered, at $\mathrm{Re}_\tau = 200, 400, 600$, a description of the main characteristics parameters of the simulations has been presented in comparison with those of [27].

The generic computational domain is represented in Fig. 5.3; furthermore, by recalling the wall formalism, one as:

$$x_i^+ = x_i \frac{u_\tau}{\nu}, \quad t^+ = t \frac{u_\tau^2}{\nu}, \quad h^+ = \frac{h}{h_\tau}, \quad u^+ = \frac{\overline{u}}{u_\tau}, \quad \mathrm{Re}_\tau = \frac{u_\tau h}{\nu} \tag{6.8}$$

where $\overline{u}$ denotes a mean $x-$velocity, averaged on a generic $xz-$plane and time, $h = \nu / u_\tau$ is the viscous length and h/u_τ is the viscous time unit.

The characteristic parameters of the simulations are reported in Tab. 6.1: it is evident that the grid spacing adopted is in good agreement with that of [27].

In the following sections, a more detailed evaluation of statistical terms and mean-flow variables is presented and analyzed, in order to verify the accuracy of the calculations, both in space and time, and the reliability of the computational domain to capture the most relevant structures.

6.2.1 Plane channel at $\mathbf{Re_\tau = 200}$

Considering the numerical database at $\mathrm{Re}_\tau = 200$, the Kolmogorov spatial microscale, estimated by using the criterion of the average dissipation rate per unit of mass across the width of the channel, is equal to $\eta^+ = 1.89$.

It can be verified that there are 14 grid points in the *y*-direction, within the viscous sublayer $y \leq 5$, that satisfies the following requirements (according to [43]):

- to select a normal-to-the-wall grid width distribution, able to resolve the steep gradients of the velocity field near the wall,
- to select a normal-to-the-wall grid such as the mean grid width ($\Delta\overline{u}$) results smaller than the relevant turbulent elements (i.e. $\Delta\overline{u} \leq \pi\eta$) and,
- to have ($\Delta t \leq \tau_\eta$).

The initial velocity profile evolving with time is interpolated onto the considered computational domain by using the statistically steady state profile obtained by [36]: thus, given an appropriate initial condition, the initial transient flow in the channel is simulated until the turbulent statistically steady state is reached. Five-hundred thousand time steps are calculated with a temporal resolution of $\Delta t = 1 \cdot 10^{-4}\, h/u_\tau$ (that corresponds to $\Delta t^+ = 0.02$), while the Kolmogorov time microscale results $\tau_\eta^+ = 3.54$.

With reference on Tab. 6.1, the adequacy of the computational domain and its grid resolution can be verified by evaluating the two-point correlation coefficients and the energy spectra. In Figs. 6.1-6.4, the two-point correlation coefficients are shown, both in $x-$ and $z-$directions at two *y*-locations: in particular, Figs. 6.1 and 6.2 are referred to $y/h = 0.990$ that is very close to the wall, while Figs. 6.3 and 6.4 are referred to $y/h = 0.0194$ that is very close to the centerline. Because of the fluctuations at $x-$ and $z-$directions are uncorrelated for large separations, the computational domain is considered adequate to capture all the relevant large-scale turbulent structure.

Figure 6.5 shows the one-dimensional energy spectra and demonstrates the adequacy of grid resolutions adopted, being the energy density associated with the high wavenumbers several orders of magnitude lower than the energy density corresponding to low wavenumbers: furthermore, there is no evidence of energy pile-up at high wavenumbers, thanks to numerical filter, based on the "$3/2$ rule", that avoids aliasing errors.

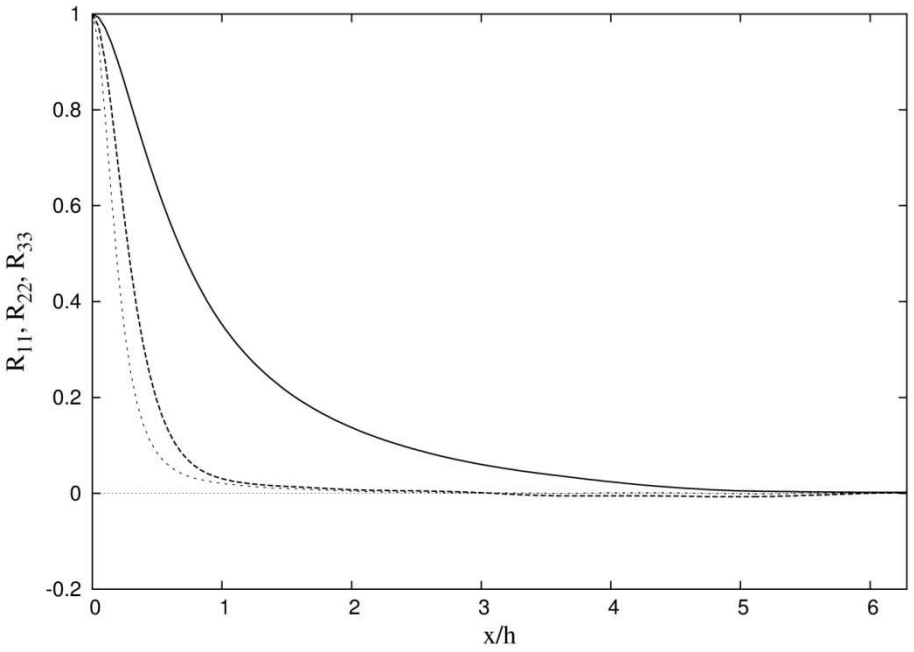

Figure 6.1 – Two-point correlation coefficients of the velocity fluctuations. Streamwise separation at $y/h = -0.990$ (wall): (—) R_{11}; (---) R_{22}; (¨) R_{33}.

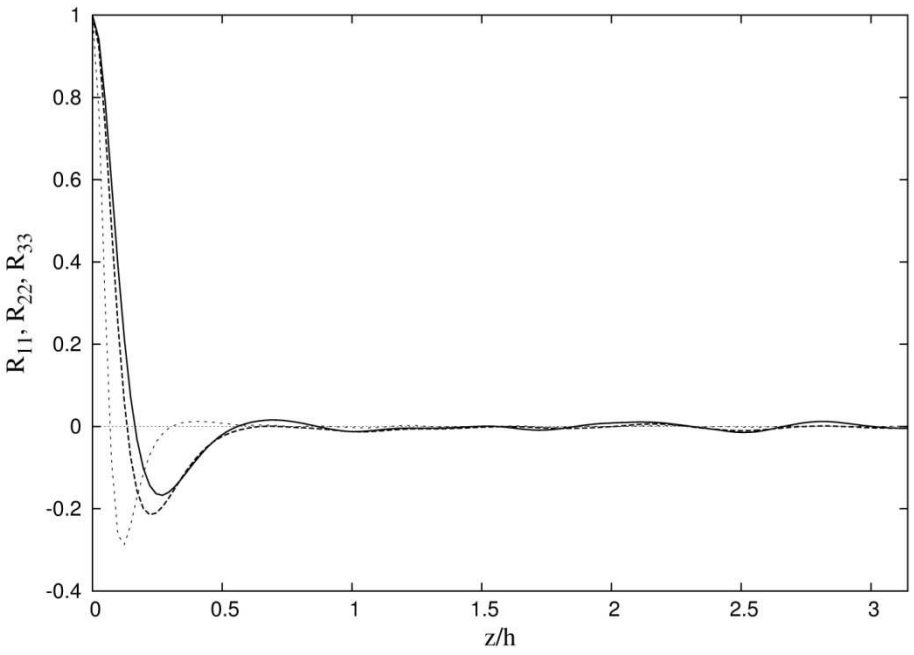

Figure 6.2 – Two-point correlation coefficients of the velocity fluctuations. Spanwise separation at $y/h = -0.990$ (wall): (—) R_{11}; (---) R_{22}; (¨) R_{33}.

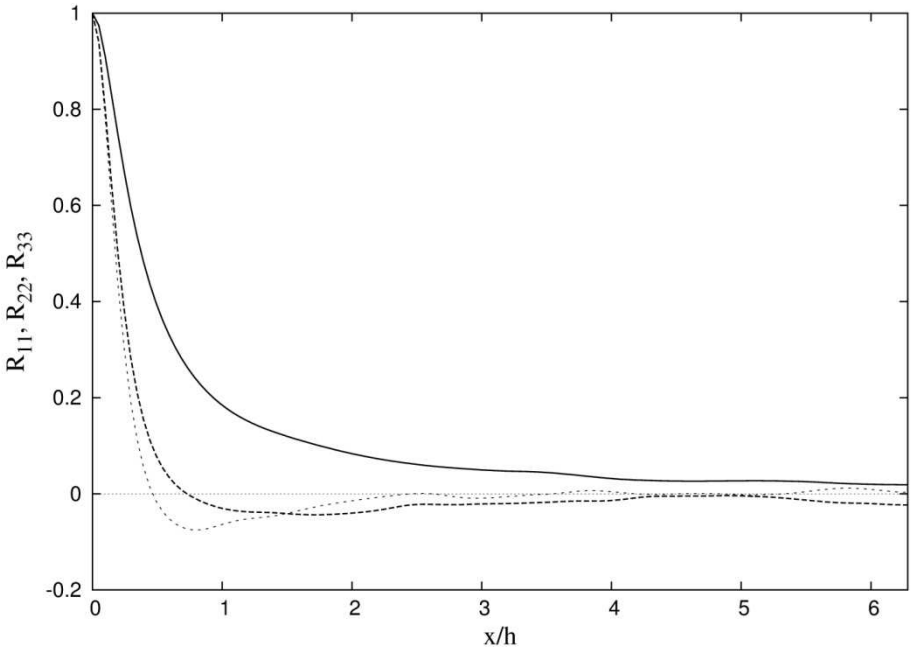

Figure 6.3 – Two-point correlation coefficients of the velocity fluctuations. Streamwise separation at $y/h = -0.0194$ (center): (—) R_{11}; (---) R_{22}; (¨) R_{33}.

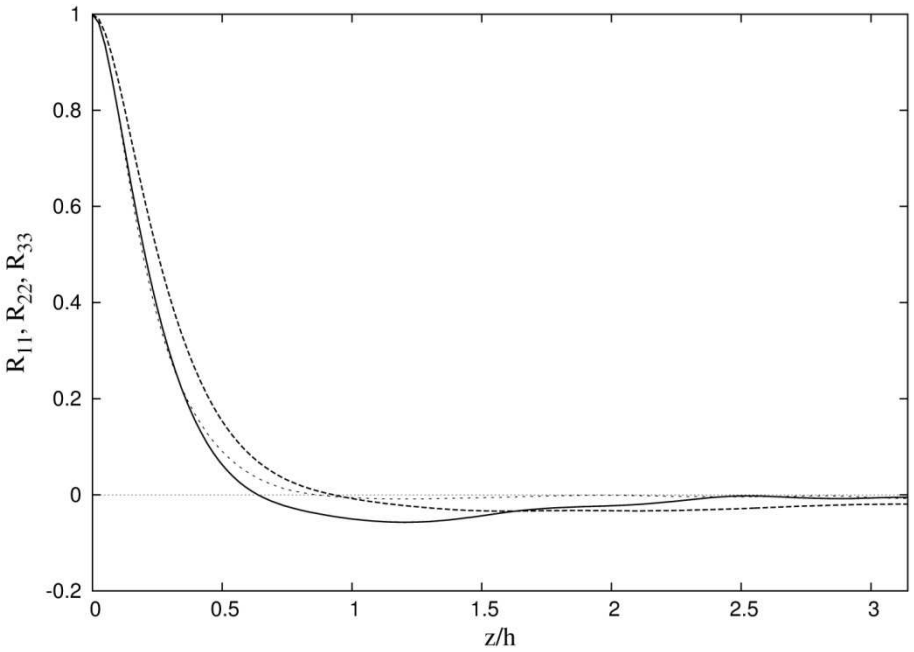

Figure 6.4 – Two-point correlation coefficients of the velocity fluctuations. Spanwise separation at $y/h = -0.0194$ (center): (—) R_{11}; (---) R_{22}; (¨) R_{33}.

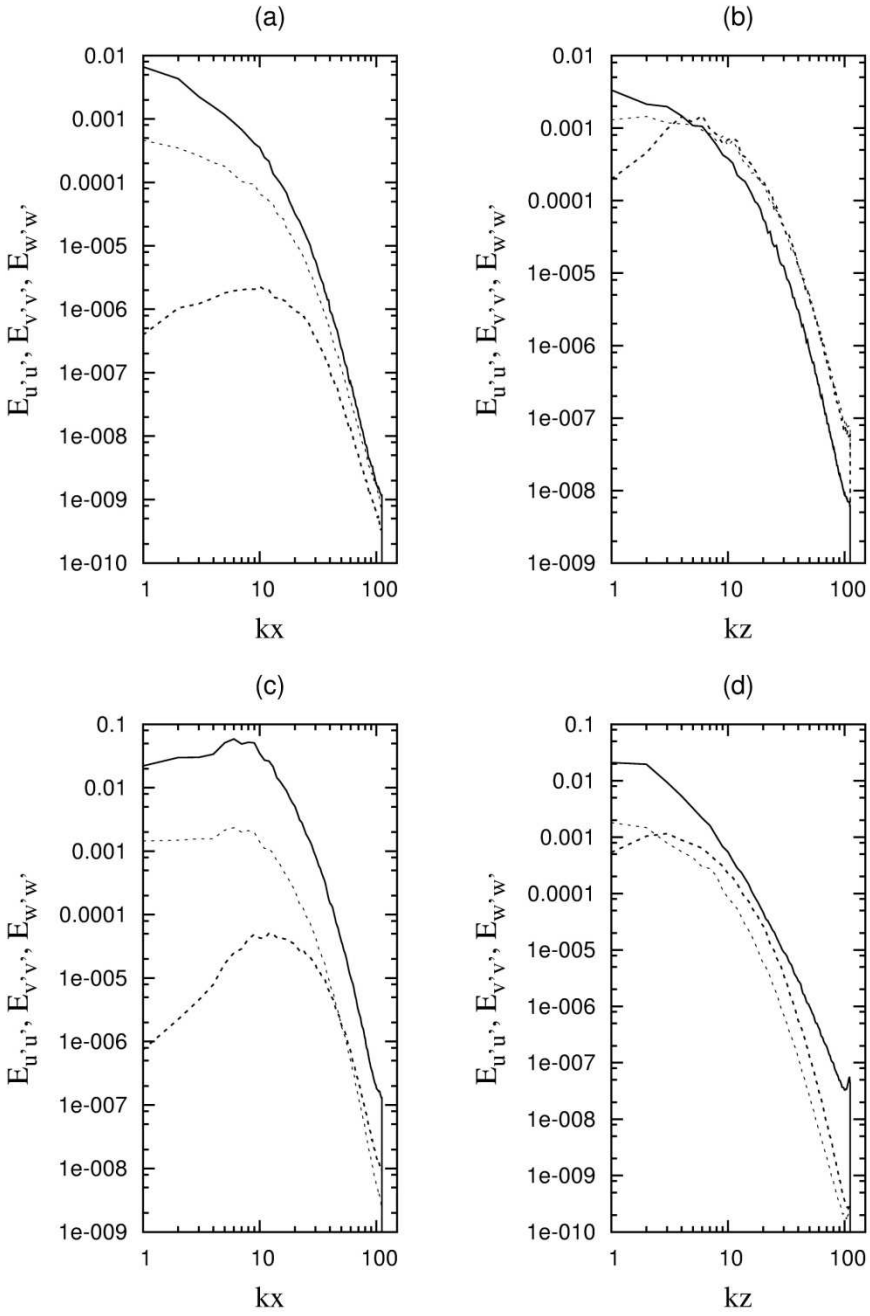

Figure 6.5 – One-dimensional energy spectra. (a) and (b) at $y/h = -0.990$ (wall), (c) and (d) at $y/h = -0.0194$ (center): (—) $E_{u'u'}$; (---); $E_{v'v'}$; (¨) $E_{w'w'}$.

About mean flow properties, the mean-velocity profile normalized by the centerline velocity across the section of the channel in global coordinates $\overline{u}/U_c$ is shown in Fig. 6.6; in Fig. 6.7, the mean velocity profile normalized by the friction velocity in wall coordinates u^+ is compared with the law of the wall and with the results of [27]

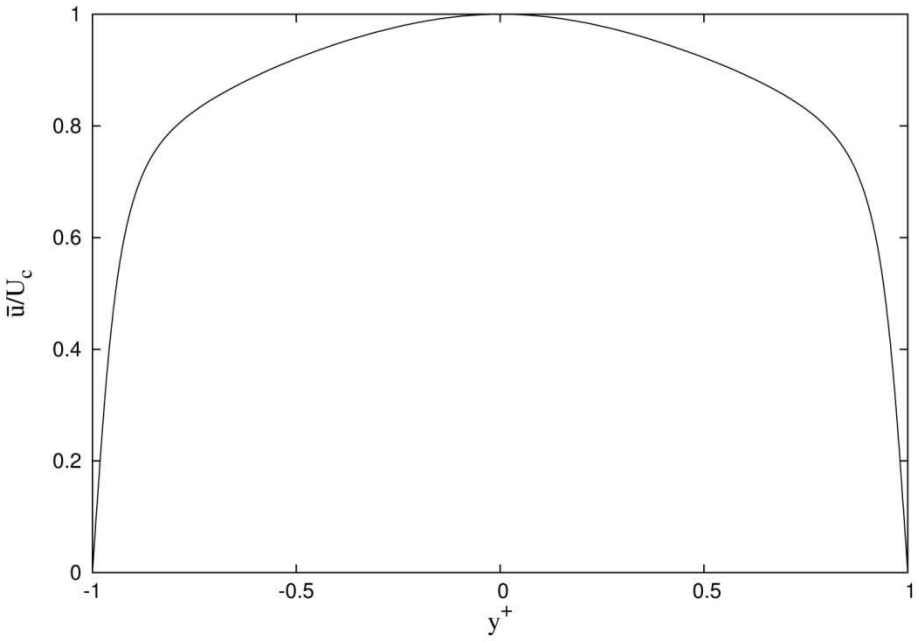

Figure 6.6 – Mean-velocity profile $\bar{u}/U_c$ across the channel in global coordinates y/h.

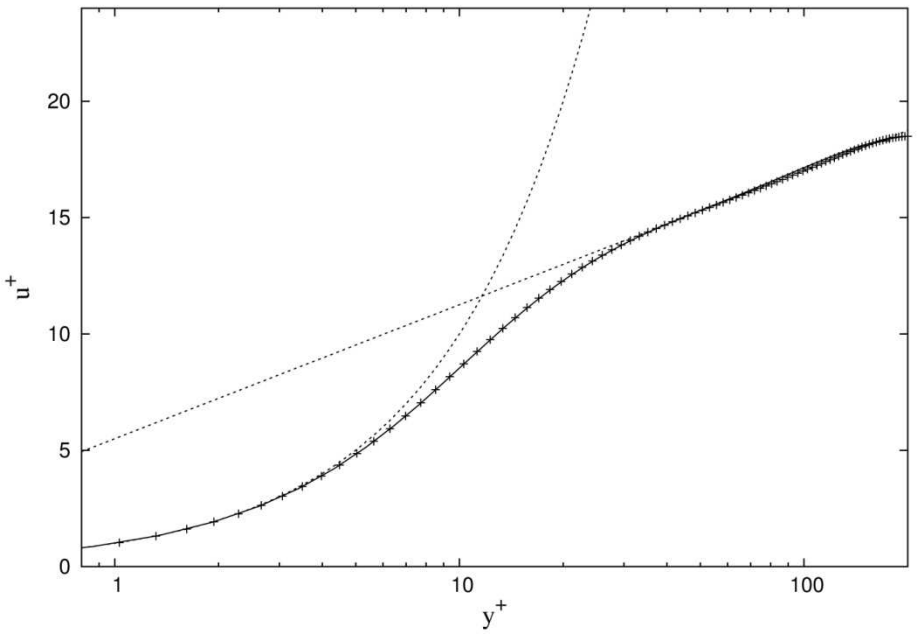

Figure 6.7 – Mean-velocity profile u^+ in wall coordinates y^+: (+) present case study, (—) data from [27]; (---) law at the wall.

at $\mathrm{Re}_\tau = 180$. Also in this case, the comparison between computed results and numerical data is satisfactory.

Table 6.2 – Computed mean flow variables (nominal $\mathrm{Re}_\tau = 200$).

Re_τ	Re_b	Re_c	U_b/u_τ	$\overline{u}_c/u_\tau$	$\overline{u}_c/U_b$	C_{fb}	C_{fc}
200.23	3197.67	3706.26	16.97	18.51	1.16	$7.86 \cdot 10^{-3}$	$6.85 \cdot 10^{-3}$

In order to complete the mean flow analysis, Tab. 6.2 shows the computed mean flow variables by using Eqs. 6.1-6.3 and 6.6-6.7. Figure 6.7 shows that the viscous sublayer is well solved, yielding the linear velocity distribution $u^+ = y^+$ for $y < 5$.

Figures 6.8 and 6.9 report the turbulent intensities $u'_{rms}, v'_{rms}, w'_{rms}$ – the root mean square values of the velocities fluctuations normalized by the friction velocity – in global and wall coordinates, respectively. It is evident the symmetry of profiles about the centerline, that confirms the adequacy of the simulation sample taken for the average. The agreement of computed results with those of [27] is good.

Figures 6.10 and 6.12 show the skewness factors $S_{u'}, S_{v'}, S_{w'}$ of the velocity fluctuations in global and wall coordinates, respectively, in comparison with the results of [27] at $\mathrm{Re}_\tau = 180$. Similar comparisons are presented in Figs. 6.11 and 6.13 for the flatness factors $F_{u'}, F_{v'}, F_{w'}$. Considering both skewness and flatness factors, the profiles in global coordinates show small asymmetries and oscillations and the related values are significantly different from the Gaussian ones (0 and 3, respectively), showing a satisfactory agreement with the computed results of [27]. The same conclusion can be done for the profiles in wall coordinates.

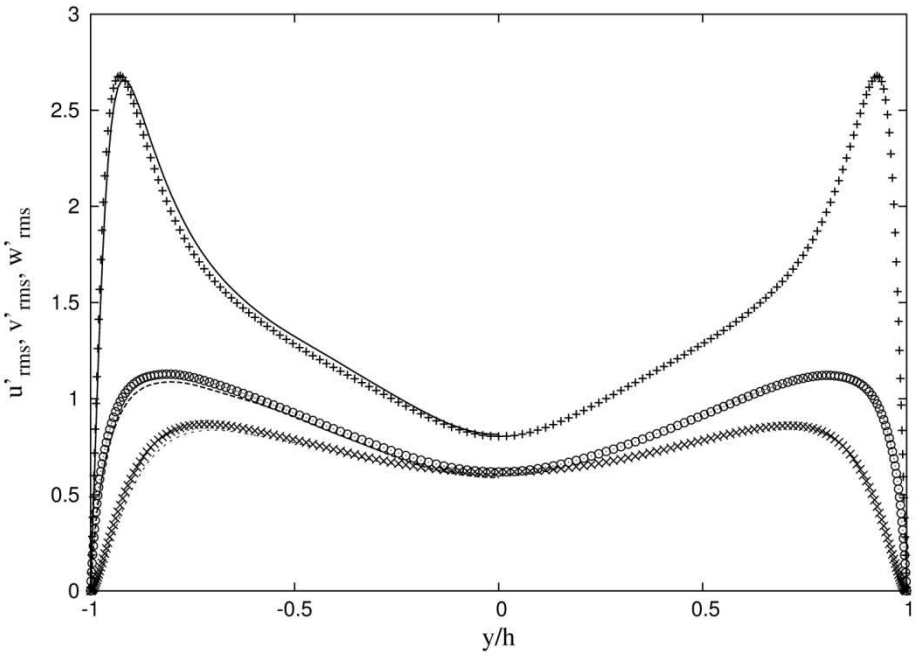

Figure 6.8 – *Rms* of velocity fluctuation in global coordinates. Present case study: (+) u'_{rms} ; (×) v'_{rms} ; (○) w'_{rms} . Data from [27]: (—) u'_{rms} ; (···) v'_{rms} ; (---) w'_{rms} .

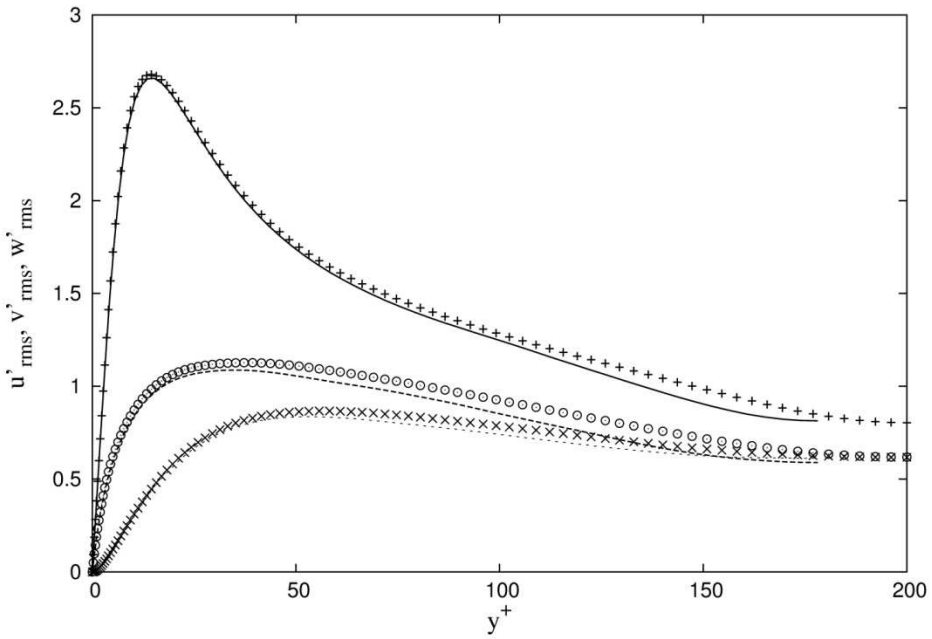

Figure 6.9 – *Rms* of velocity fluctuation in wall coordinates. Present case study: (+) u'_{rms} ; (×) v'_{rms} ; (○) w'_{rms} . Data from [27]: (—) u'_{rms} ; (···) v'_{rms} ; (---) w'_{rms} .

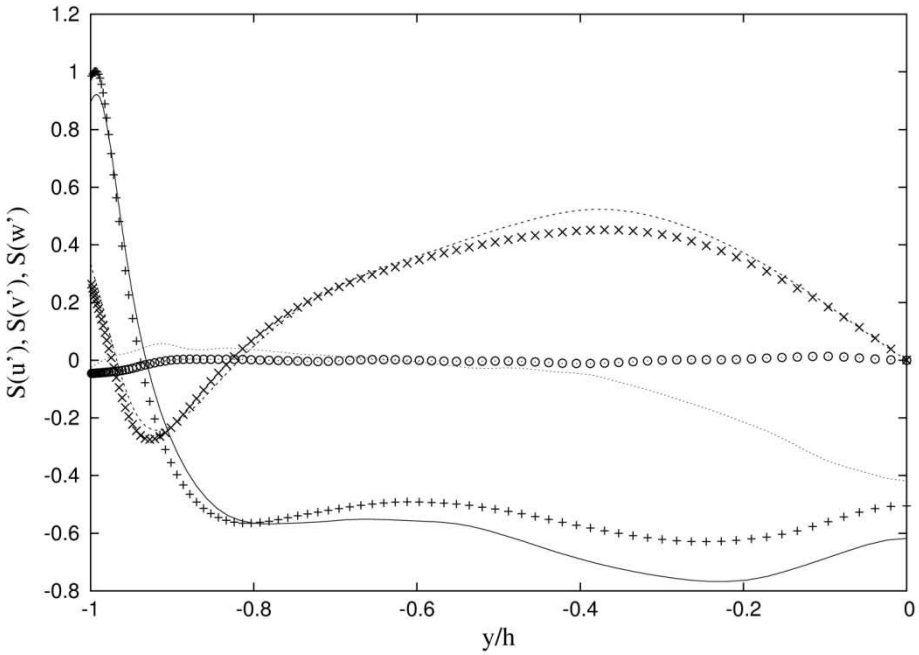

Figure 6.10 – Skewness factors of the velocity fluctuations in global coordinates. Present case study: (+) $S_{u'}$; (×) $S_{v'}$; (○) $S_{w'}$. Data from [27]: (—) $S_{u'}$; (¨) $S_{v'}$; (---) $S_{w'}$.

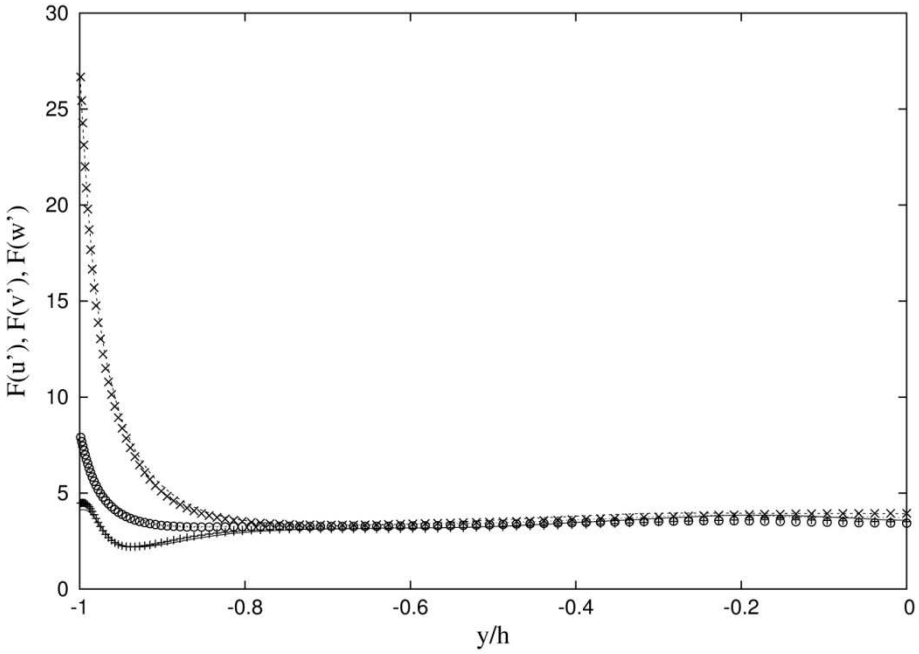

Figure 6.11 – Flatness factors of the velocity fluctuations in global coordinates. Present case study: (+) $F_{u'}$; (×) $F_{v'}$; (○) $F_{w'}$. Data from [27]: (—) $F_{u'}$; (¨) $F_{v'}$; (---) $F_{w'}$.

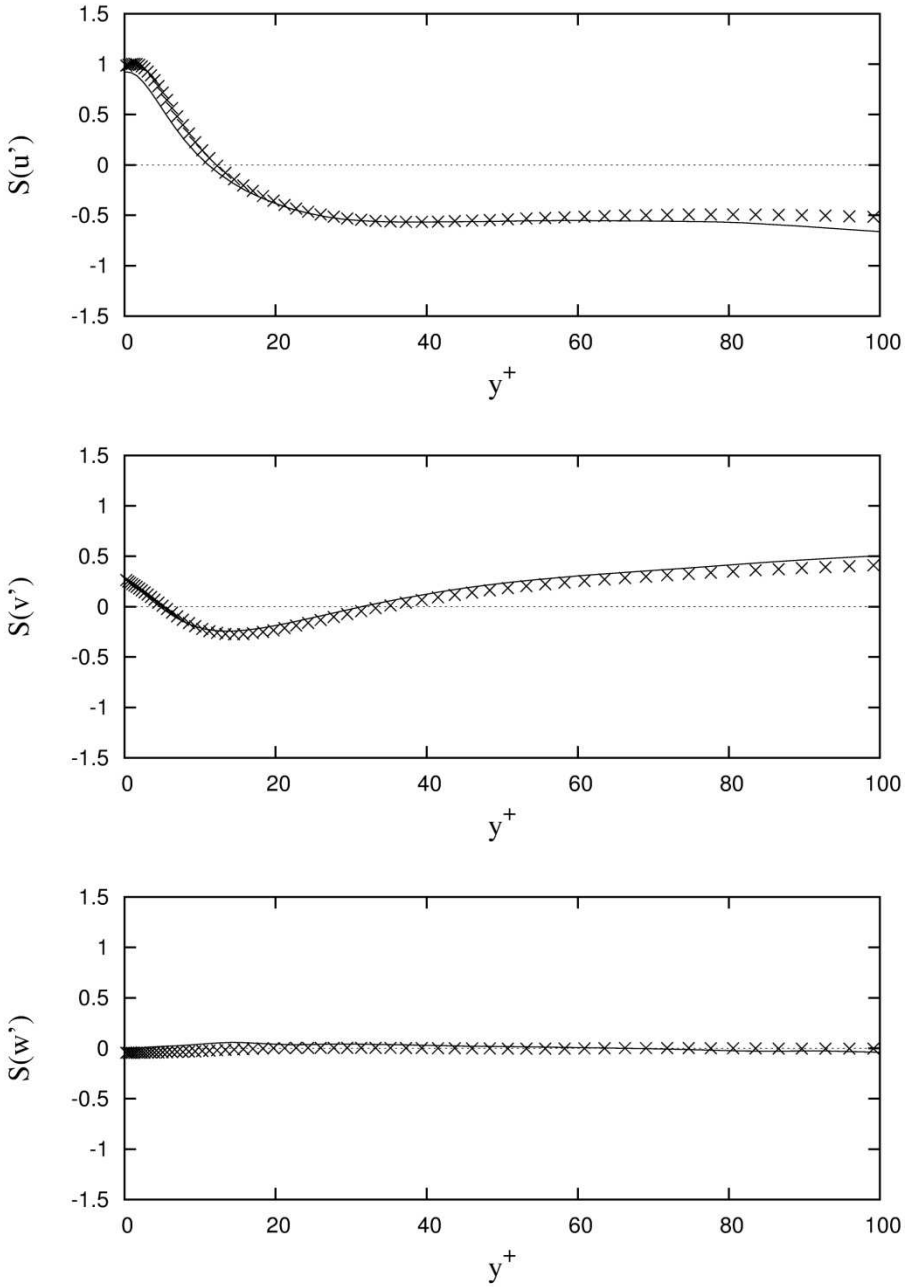

Figure 6.12 – Skewness factors of the velocity fluctuations wall coordinates: (×) present work; (─) data from [27].

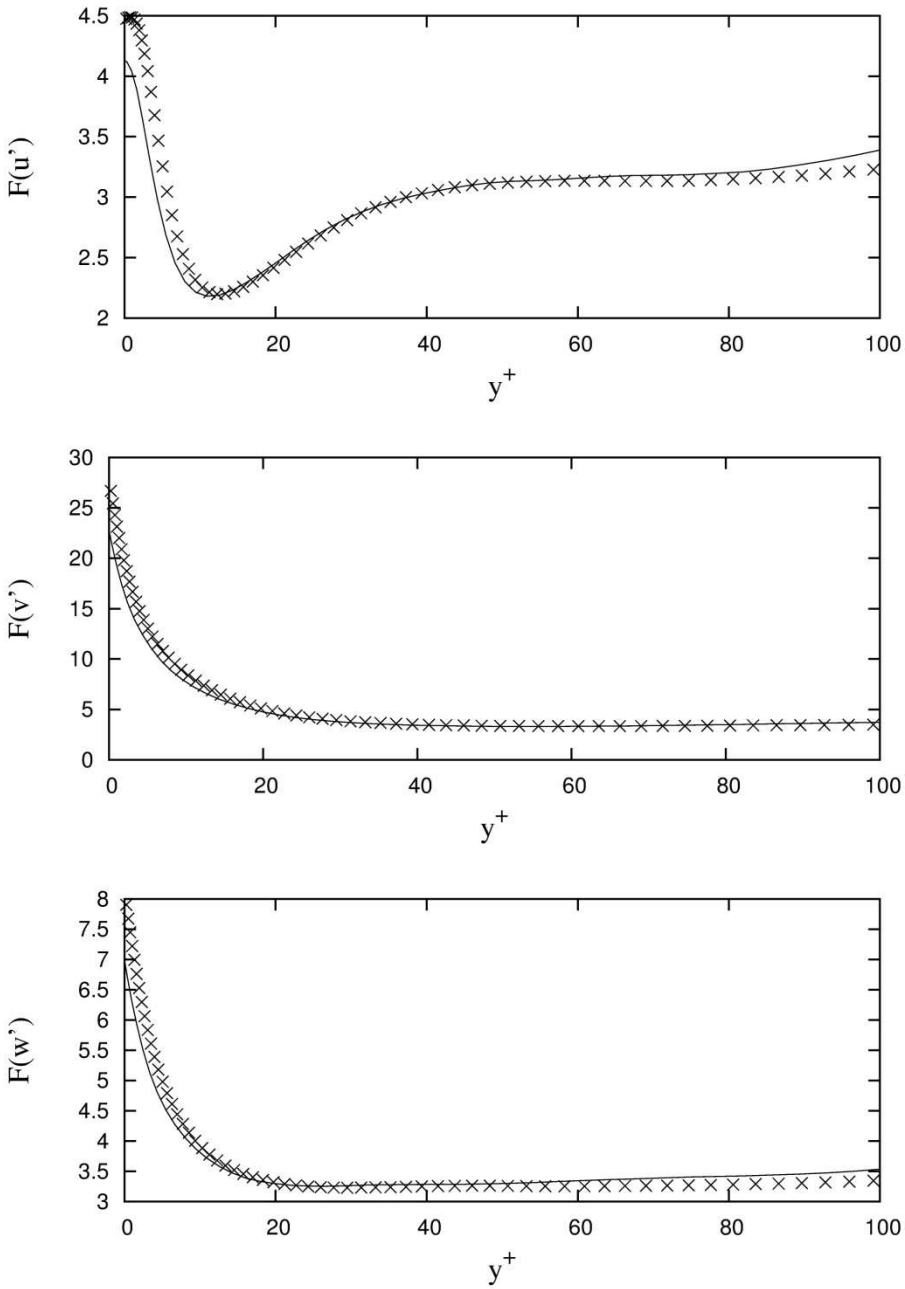

Fig. 6.13 – Flatness factors of the velocity fluctuations wall coordinates: (×) present work; (—) data from [27].

6.2.2 Plane channel at $\mathbf{Re_\tau = 400}$

In this section, the same analysis, done for DNS database at $\mathrm{Re}_\tau = 200$, is performed for DNS database at $\mathrm{Re}_\tau = 400$, in order to verify the reliability of the computational domain, whose parameters are reported on Tab. 6.1.

Table 6.3 – Computed mean flow variables (nominal $\mathrm{Re}_\tau = 400$).

Re_τ	Re_b	Re_c	U_b/u_τ	$\overline{u}_c/u_\tau$	$\overline{u}_c/U_b$	C_{fb}	C_{fc}
399.94	6966.95	7978.80	17.42	19.95	1.14	$6.58\cdot10^{-3}$	$6.02\cdot10^{-3}$

About numerical accuracy, the Kolmogorov spatial microscale, estimated by using the criterion of the average dissipation rate per unit mass across the width of the channel, results $\eta^+ = 2.19$. Along $y-$direction, 16 grid points have been used, within the viscous sublayer, to satisfy the requirements defined by [43].

The initial velocity profile evolving with time is interpolated onto the considered computational domain by using the statistically steady state profile obtained from the database at $\mathrm{Re}_\tau = 200$: thus, given an appropriate initial condition, the initial transient flow in the channel is simulated until the turbulent statistically steady state is reached.

Five-hundred thousand time steps are calculated with a temporal resolution of $\Delta t = 1\cdot10^{-4}\, h/u_\tau$ (that corresponds to $\Delta t^+ = 0.04$), while the Kolmogorov time microscale results $\tau_\eta^+ = 4.79$.

With reference on Tab. 6.1, the adequacy of the computational domain and its grid resolution can be verified by evaluating the two-point correlation coefficients and the energy spectra.

In Figs. 6.14-6.17, the two-point correlation coefficients are shown, both in $x-$ and $z-$directions at two $y-$locations: in particular, Figs. 6.14 and 6.15 are referred to $y/h = 0.988$ that is very close to the wall, while Figs. 6.16 and 6.17 are referred to $y/h = 0.022$ that is very close to the centerline. Also in this case, because of fluctuations along $x-$ and $z-$directions are uncorrelated for large separations, the computational domain is considered adequate to capture all the relevant large-scale turbulent structure [43]. Figure 6.18 shows the one-dimensional energy spectra and demonstrates the adequacy of grid resolutions adopted because of the energy cascade and there is no evidence of energy pile-up at high wavenumbers.

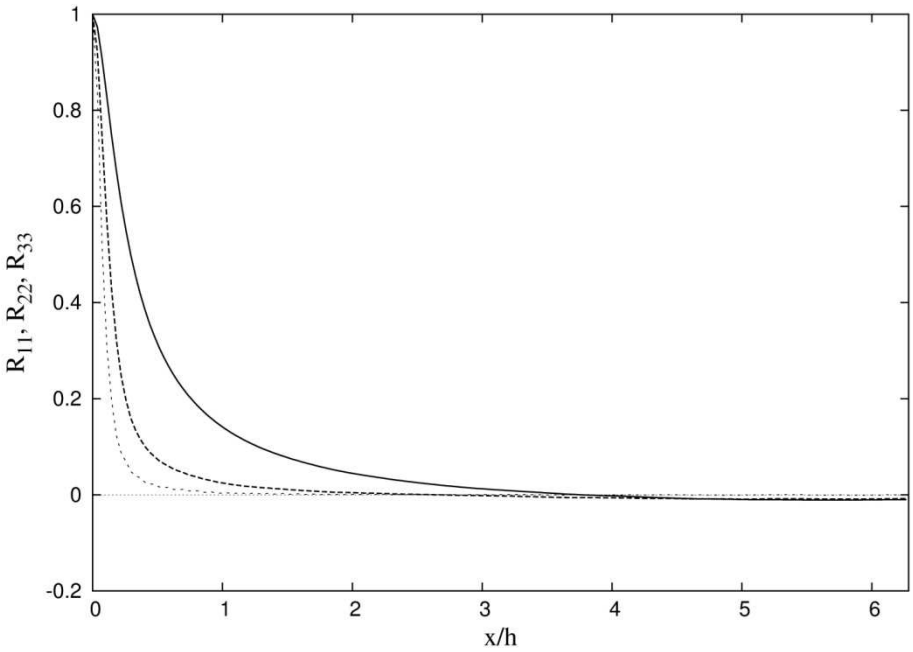

Figure 6.14 – Two-point correlation coefficients of the velocity fluctuations. Streamwise separation at $y/h = -0.988$ (wall): (—) R_{11}; (---) R_{22}; (···) R_{33}.

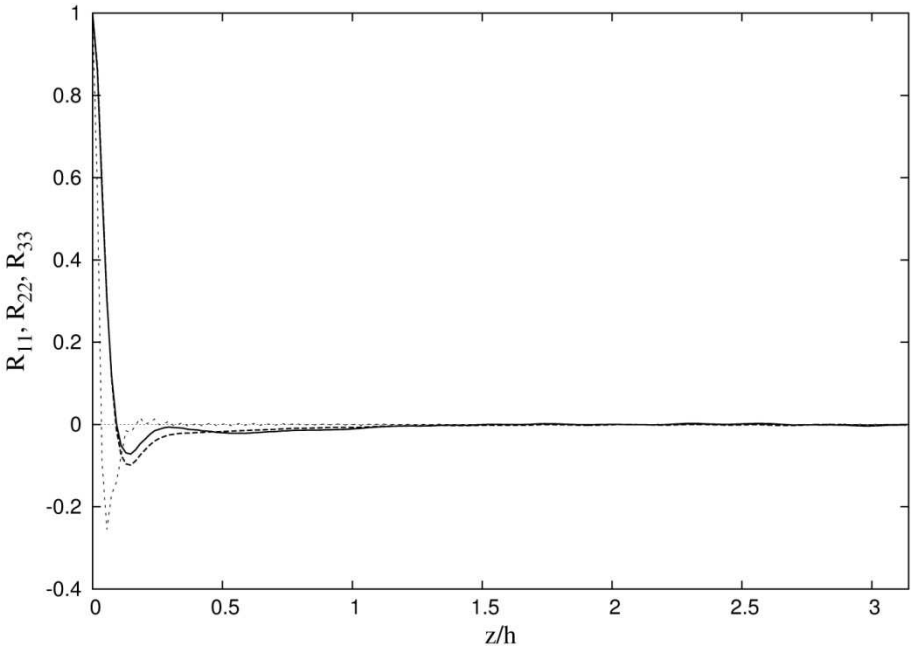

Figure 6.15 – Two-point correlation coefficients of the velocity fluctuations. Spanwise separation at $y/h = -0.988$ (wall): (—) R_{11}; (---) R_{22}; (···) R_{33}.

About mean flow properties, the mean-velocity profile normalized by the centerline velocity across the section of the channel in global coordinates $\overline{u}/U_c$ is shown in

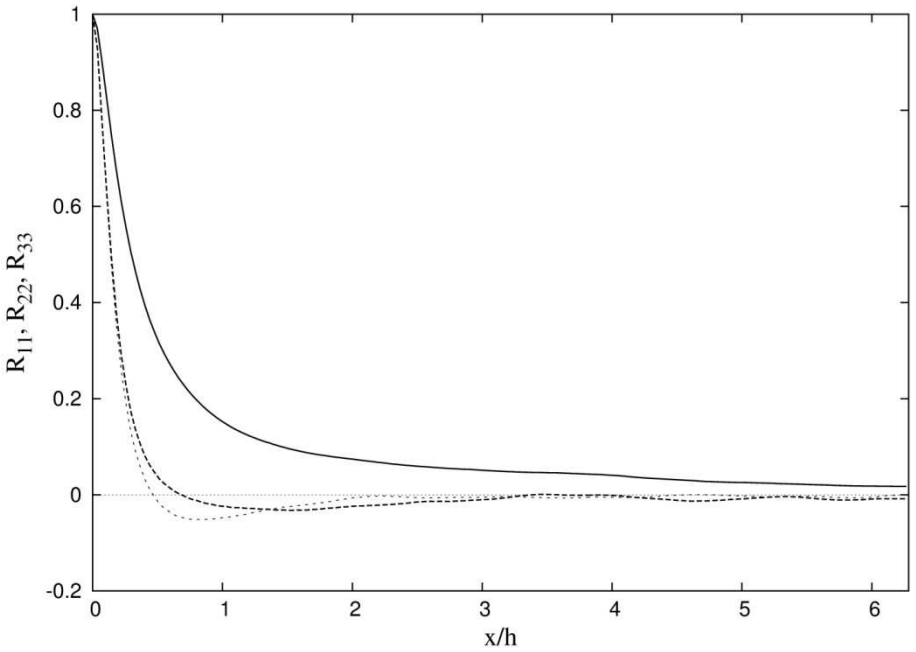

Figure 6.16 – Two-point correlation coefficients of the velocity fluctuations. Streamwise separation at $y/h = -0.022$ (center): (—) R_{11}; (---) R_{22}; (···) R_{33}.

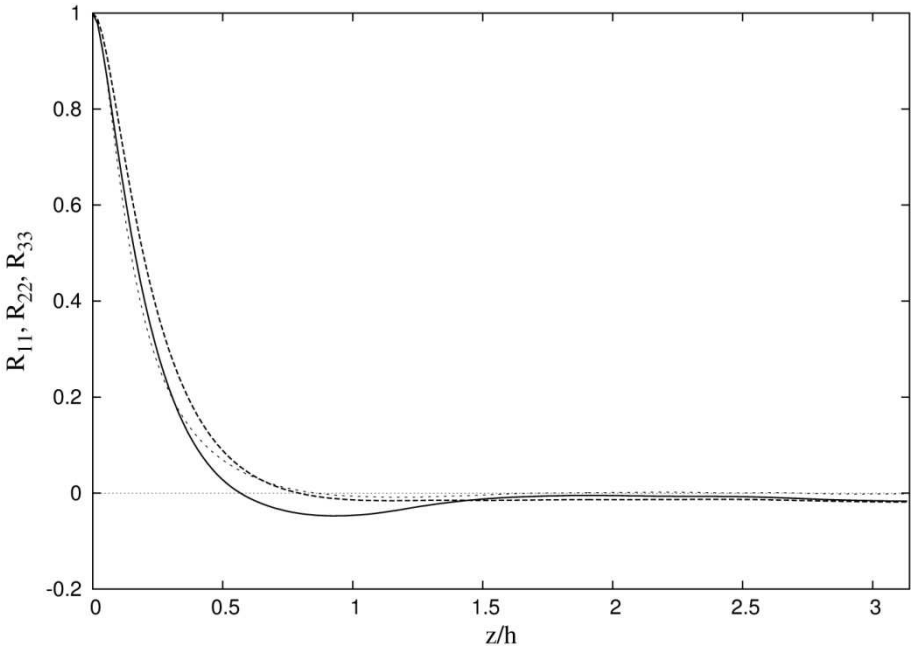

Figure 6.17 – Two-point correlation coefficients of the velocity fluctuations. Spanwise separation at $y/h = -0.022$ (center): (—) R_{11}; (---) R_{22}; (···) R_{33}.

Fig. 6.19; in Fig. 6.20, the mean velocity profile normalized by the friction velocity in wall coordinates u^+ is compared with the law of the wall and with the results of [27] at $\mathrm{Re}_\tau = 395$. Also in this case, the comparison is rather satisfactory.

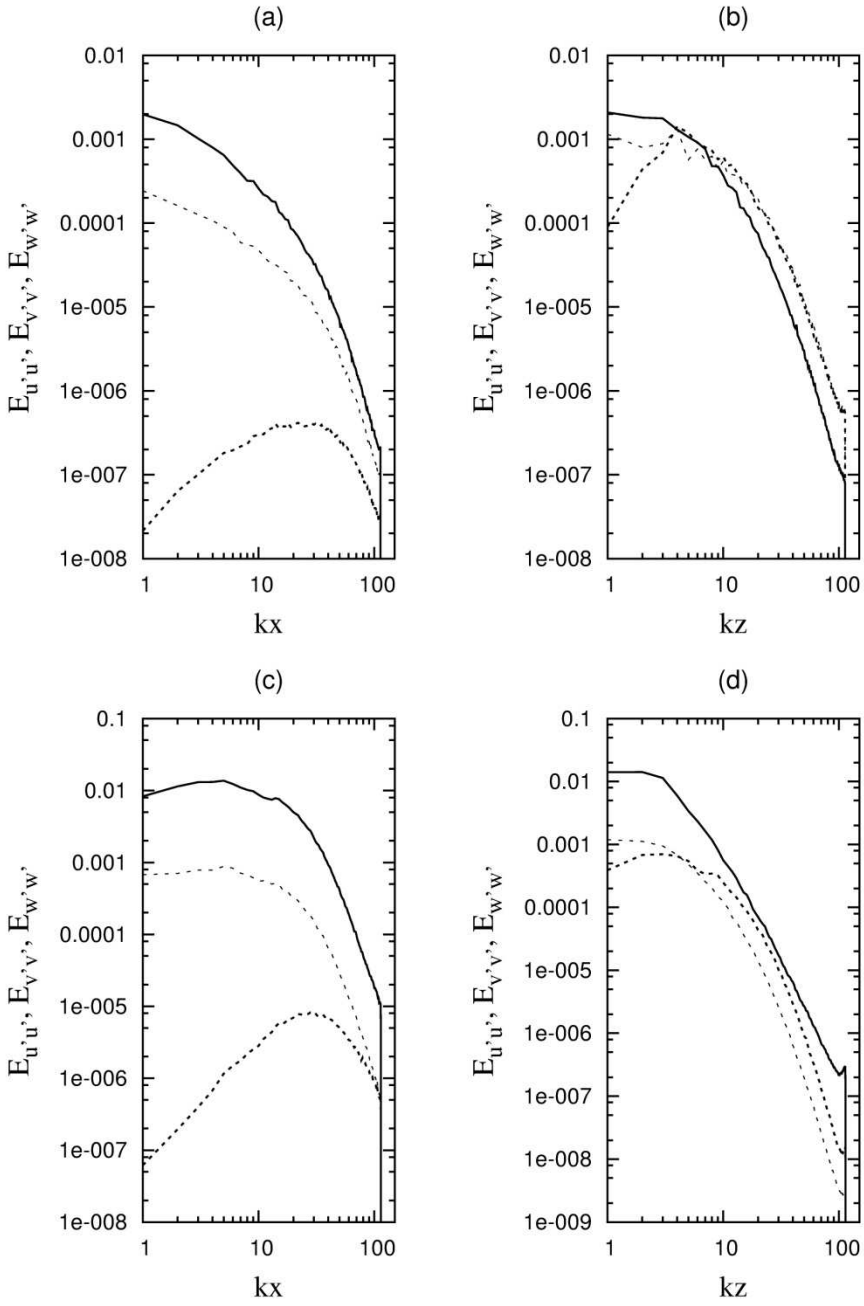

Figure 6.18 – One-dimensional energy spectra. (a) and (b) at $y/h = -0.988$ (wall), (c) and (d) at $y/h = -0.022$ (center): (—) $E_{u'u'}$; (---); $E_{v'v'}$; (···) $E_{w'w'}$.

Fig. 6.20 shows that the viscous sublayer is well solved, yielding the linear velocity distribution $u^+ = y^+$ for $y < 5$. In order to complete the mean flow analysis, Tab. 6.3 shows the computed mean flow variables by using Eqs. 6.1-6.3 and 6.6-6.7.

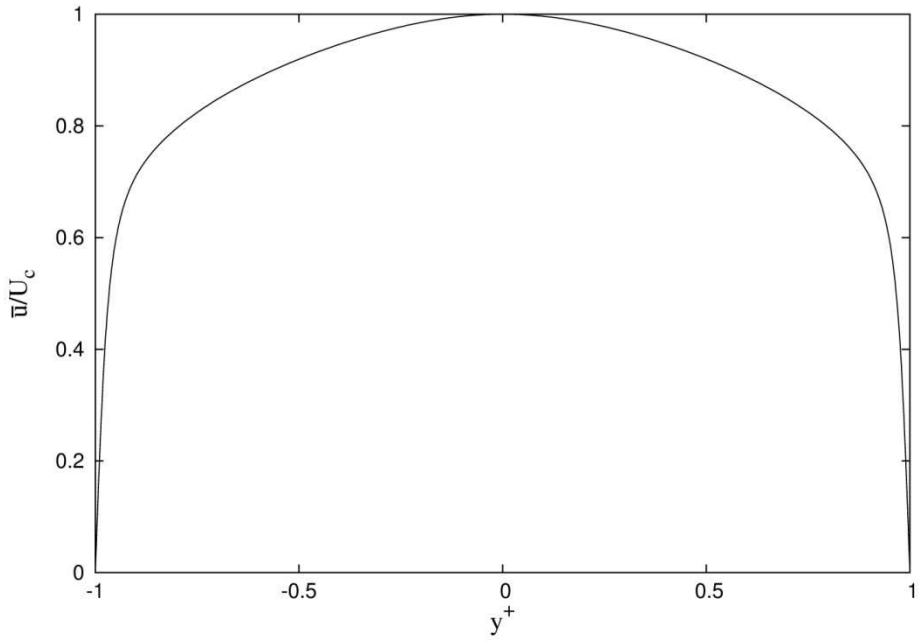

Figure 6.19 – Mean-velocity profile $\overline{u}/U_c$ across the channel in global coordinates y/h.

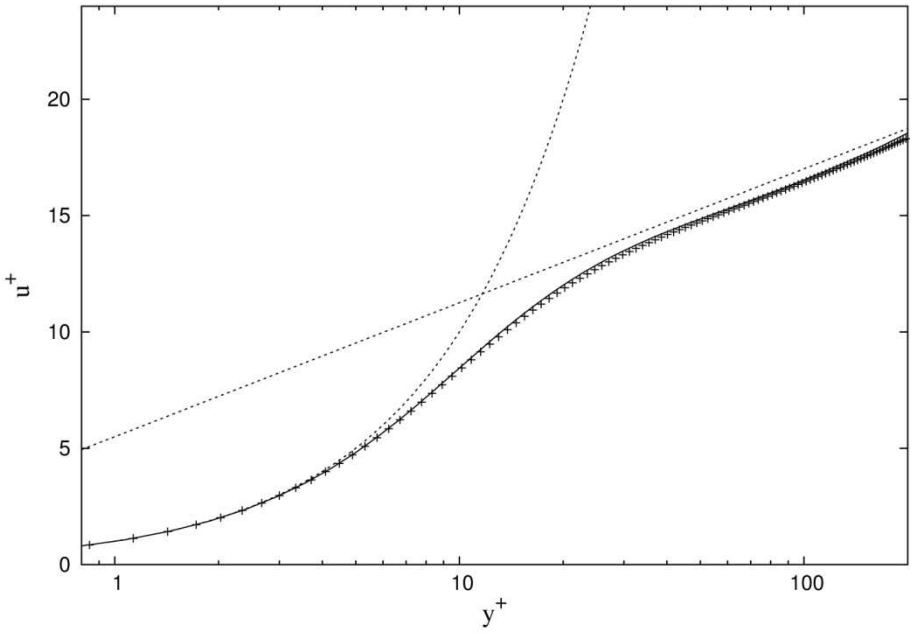

Figure 6.20 – Mean-velocity profile u^+ in wall coordinates y^+: (+) present case study, (—) data from [27]; (---) law at the wall.

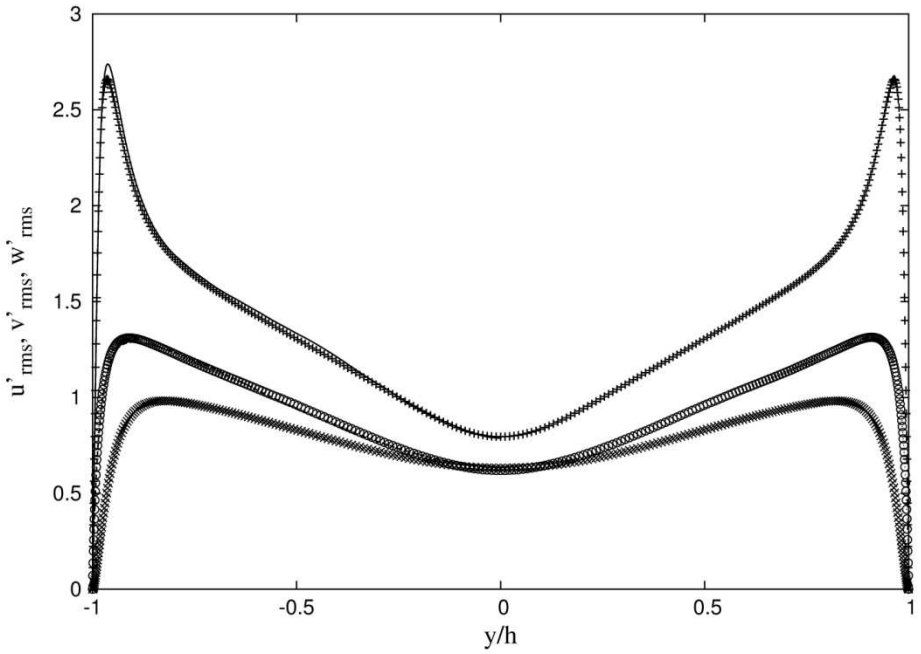

Figure 6.21 – *Rms* of velocity fluctuation in global coordinates. Present case study: (+) u'_{rms} ; (×) v'_{rms} ; (○) w'_{rms} . Data from [27]: (—) u'_{rms} ; (···) v'_{rms} ; (---) w'_{rms} .

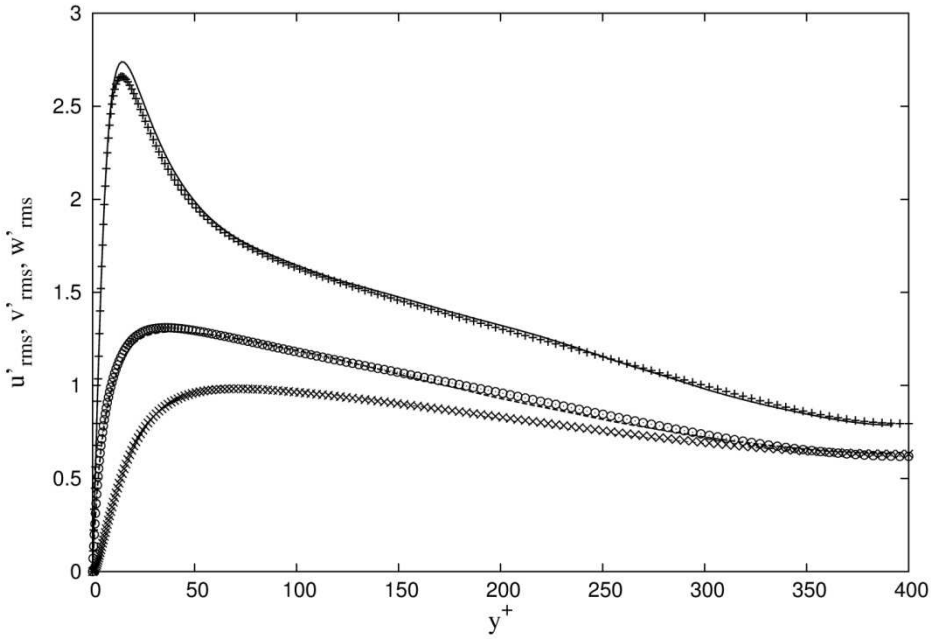

Figure 6.22 – *Rms* of velocity fluctuation in wall coordinates. Present case study: (+) u'_{rms} ; (×) v'_{rms} ; (○) w'_{rms} . Data from [27]: (—) u'_{rms} ; (···) v'_{rms} ; (---) w'_{rms} .

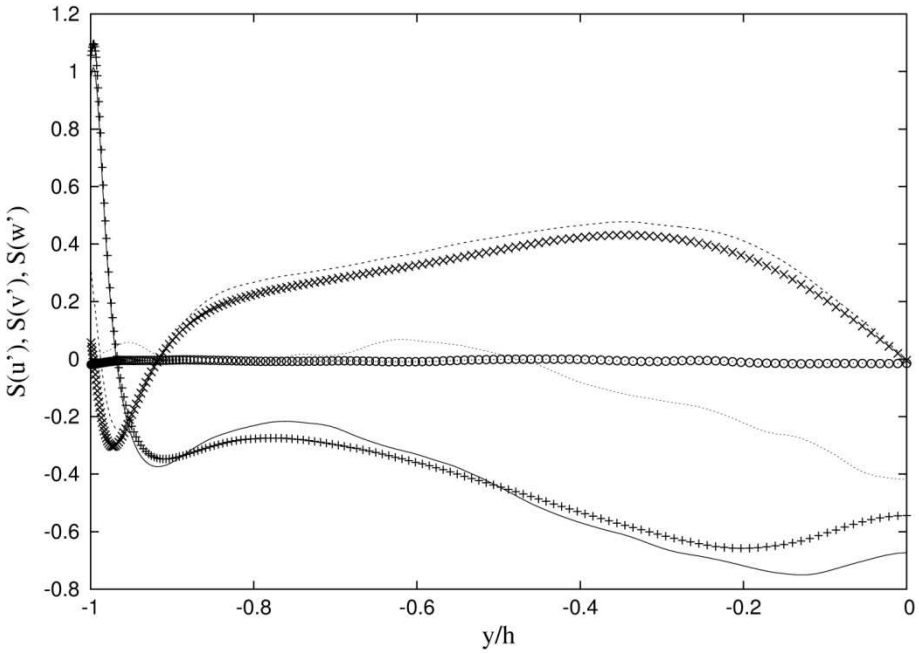

Figure 6.23 – Skewness factors of the velocity fluctuations in global coordinates. Present case study: (+) $S_{u'}$; (×) $S_{v'}$; (○) $S_{w'}$. Data from [27]: (—) $S_{u'}$; (¨) $S_{v'}$; (---) $S_{w'}$.

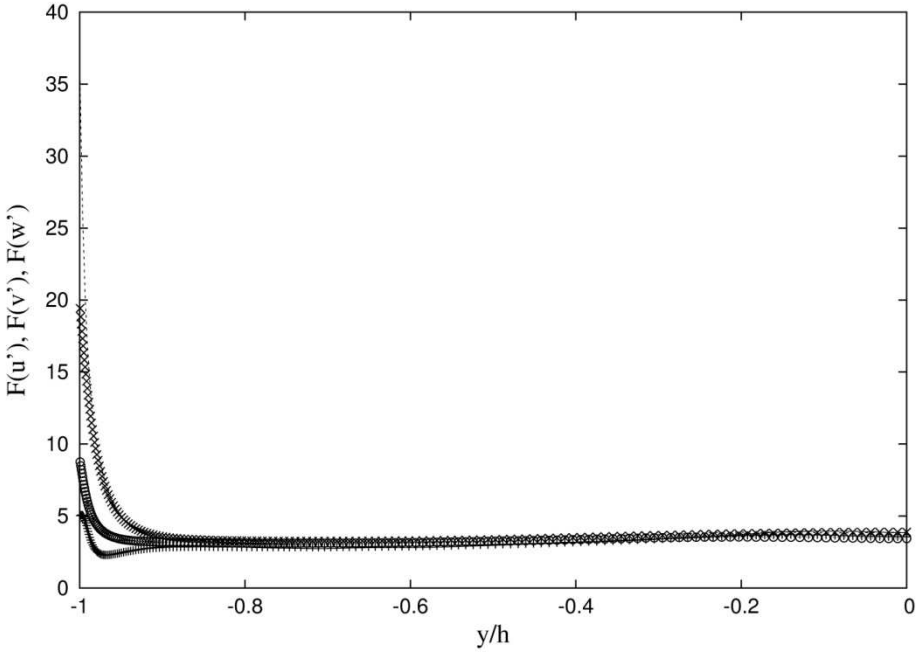

Figure 6.24 – Flatness factors of the velocity fluctuations in global coordinates. Present case study: (+) $F_{u'}$; (×) $F_{v'}$; (○) $F_{w'}$. Data from [27]: (—) $F_{u'}$; (¨) $F_{v'}$; (---) $F_{w'}$.

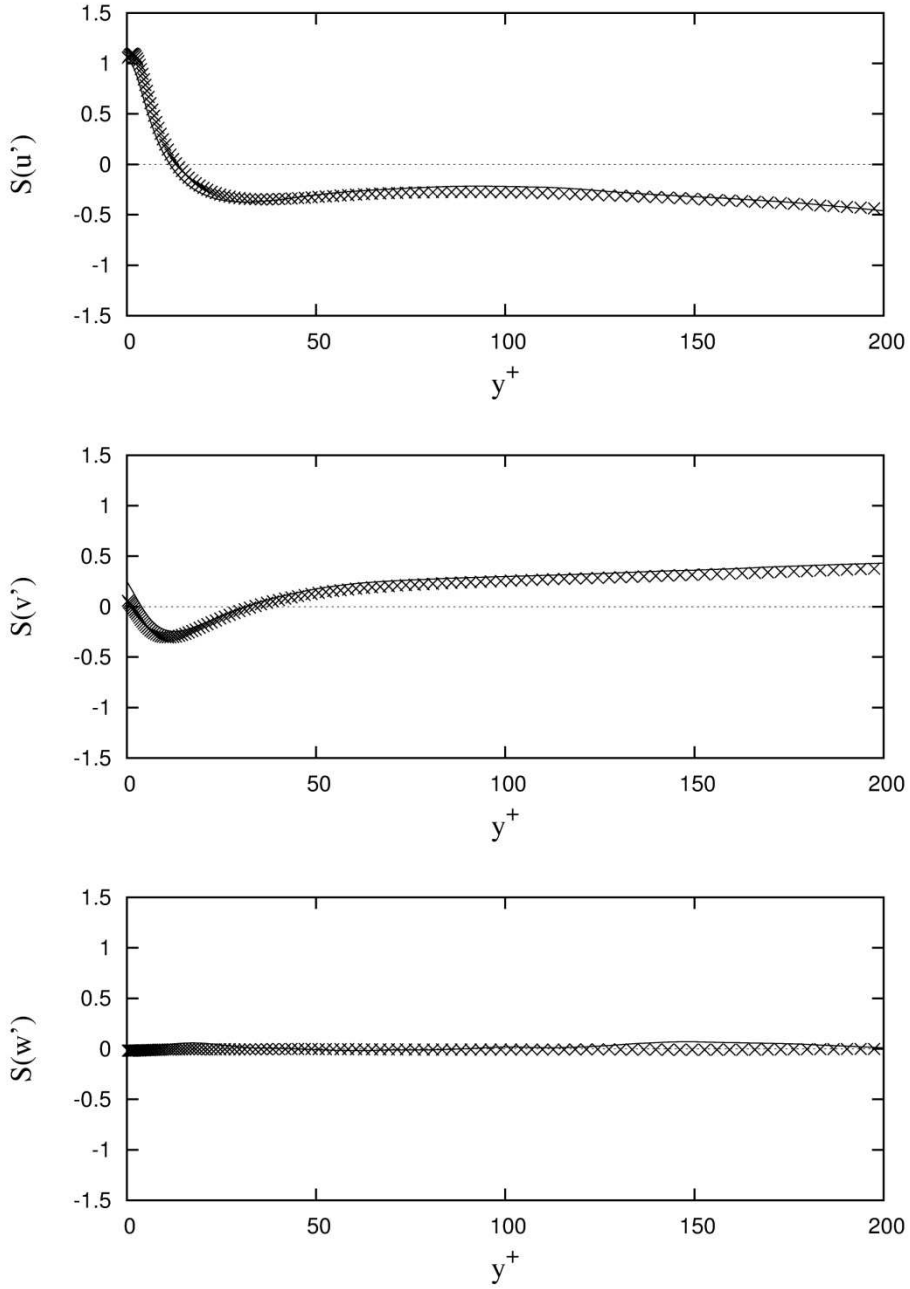

Figure 6.25 – Skewness factors of the velocity fluctuations wall coordinates: (×) present work; (—) data from [27].

Figures 6.21 and 6.22 report the turbulent intensities $u'_{rms}, v'_{rms}, w'_{rms}$ – the root mean square values of the velocities fluctuations normalized by the friction velocity – in global and wall coordinates, respectively. It is evident the symmetry of the profiles about the centerline, that confirms the adequacy of the simulation sample

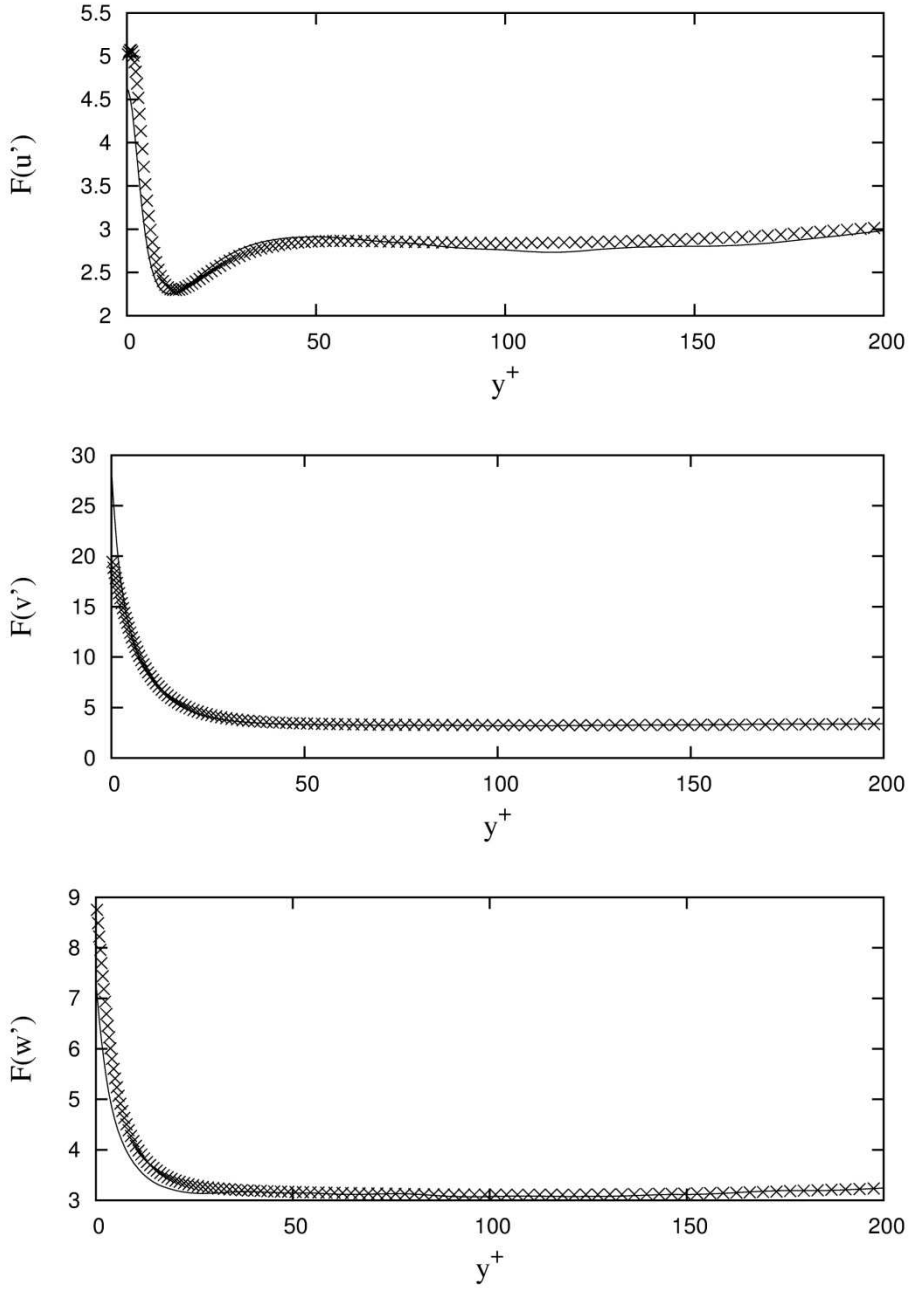

Figure 6.26 – Flatness factors of the velocity fluctuations wall coordinates: (×) present work; (—) data from [27].

taken for the average. The agreement of computed results with those of [27] is good. Figures 6.23 and 6.25 report the skewness factors $S_{u'}, S_{v'}, S_{w'}$ of the velocity fluctuations in global and wall coordinates, respectively, in comparison with the results of [27] at $\mathrm{Re}_\tau = 395$. Similar comparisons are presented in Figs. 6.24 and

6.26 for the flatness factors $F_{u'}, F_{v'}, F_{w'}$. As observed for the last database for both skewness and flatness factors, the profiles in global coordinates show small asymmetries and oscillations and the related values are significantly different from the Gaussian ones (0 and 3, respectively) and the comparison with numerical data of [27] is rather satisfactory. The same conclusion can be done for the same profiles in wall coordinates.

6.2.3 Plane channel at $\mathbf{Re_\tau = 600}$

The Kolmogorov spatial microscale, estimated by using the criterion of the average dissipation rate per unit mass across the width of the channel, results $\eta^+ = 2.42$, while along the y-direction the total number of grid point available within the viscous sublayer is equal to 16. The initial velocity profile evolving with time is interpolated onto the considered computational domain by using one of the statistically steady state profiles obtained from the simulation at $\mathrm{Re}_\tau = 400$: the initial transient flow in the channel is simulated until the turbulent statistically steady state is reached. Two-hundred thousand time steps are calculated with a temporal resolution of $\Delta t = 1 \cdot 10^{-4}\, h/u_\tau$ (that corresponds to $\Delta t^+ = 0.06$), while the Kolmogorov time microscale results $\tau_\eta^+ = 5.87$.

With reference on Table 6.1, the adequacy of the computational domain and its grid resolution can be verified by evaluating the two-point correlation coefficients and the energy spectra. In Figs. 6.27-6.30, the two-point correlation coefficients are shown, both in $x-$ and $z-$directions at two $y-$locations: in particular, Figs. 6.27 and 6.28 are referred to $y/h = -0.990$ that is very close to the wall, while Figs. 6.29 and 6.30 are referred to $y/h = -0.0155$ that is very close to the centerline. Because of the fluctuations at $x-$ and $z-$directions are uncorrelated for large separations, the computational domain is considered adequate to capture all the relevant large-scale turbulent structures. Figure 6.31 is referred to the one-dimensional energy spectra and demonstrates the adequacy of grid resolutions adopted, being the energy density associated with the high wavenumbers several orders of magnitude lower than the energy density corresponding to low wavenumbers; furthermore, also in this case there is no evidence of energy pile-up.

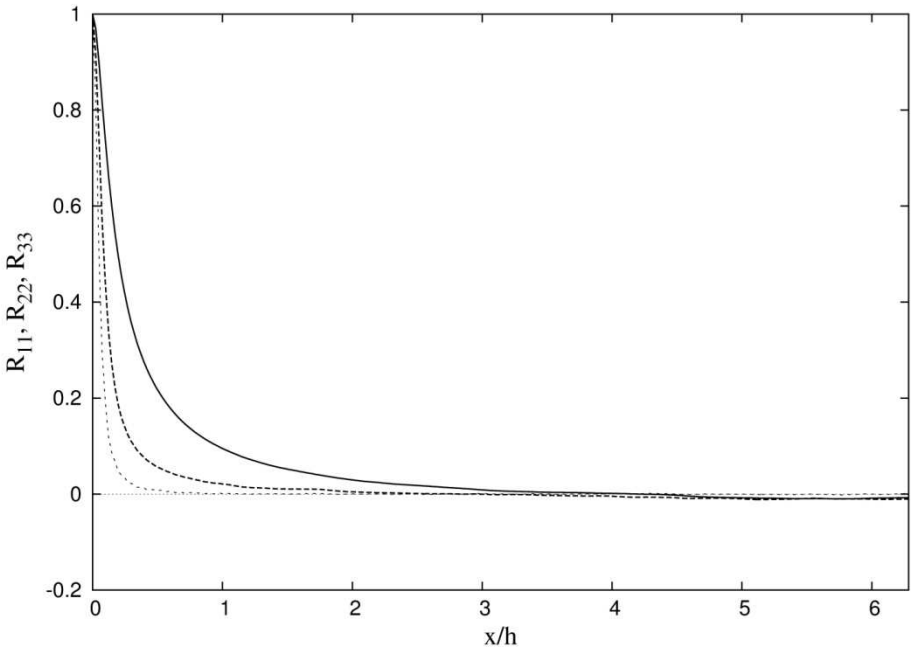

Figure 6.27 – Two-point correlation coefficients of the velocity fluctuations. Streamwise separation at $y/h = -0.990$ (wall): (—) R_{11}; (---) R_{22}; (···) R_{33}.

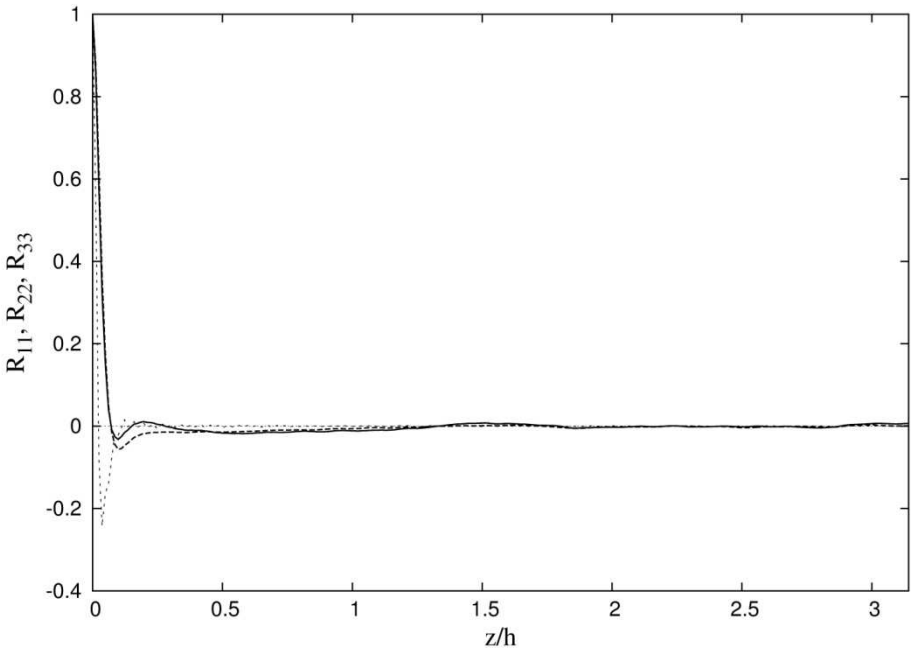

Figure 6.28 – Two-point correlation coefficients of the velocity fluctuations. Spanwise separation at $y/h = -0.990$ (wall): (—) R_{11}; (---) R_{22}; (···) R_{33}.

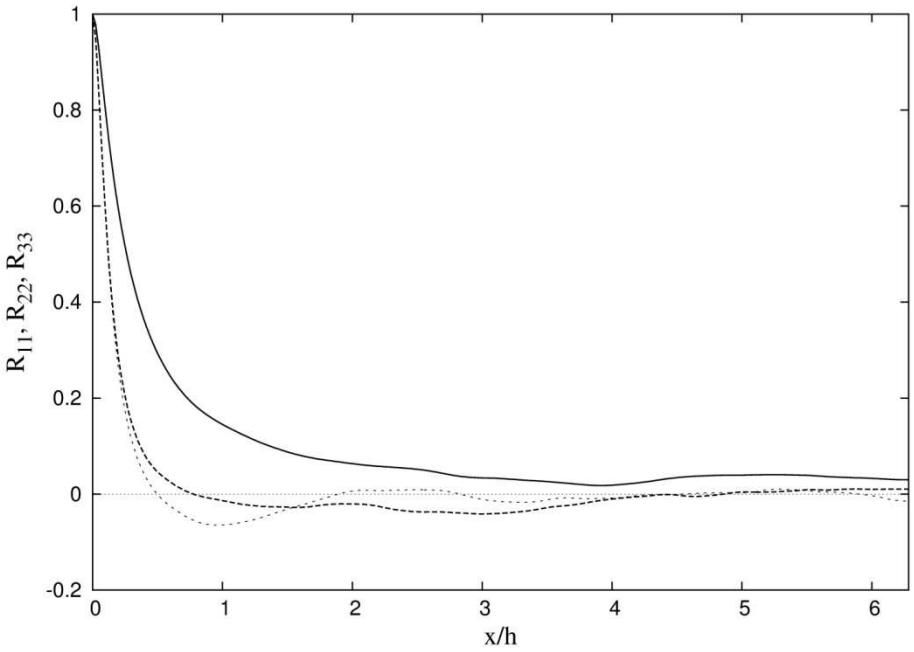

Figure 6.29 – Two-point correlation coefficients of the velocity fluctuations. Streamwise separation at $y/h = -0.0155$ (center): (—) R_{11}; (---) R_{22}; (···) R_{33}.

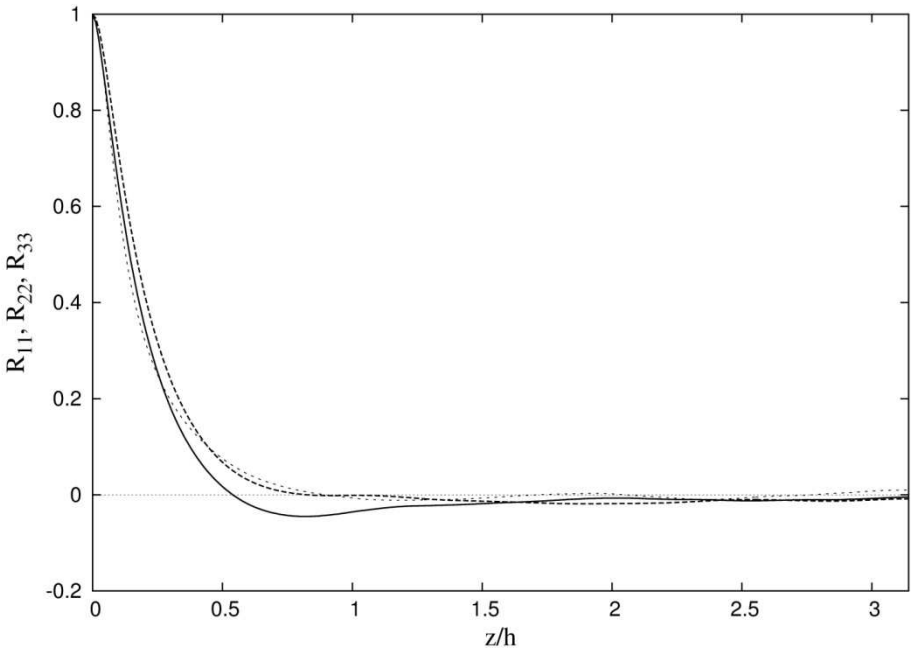

Figure 6.30 – Two-point correlation coefficients of the velocity fluctuations. Spanwise separation at $y/h = -0.0155$ (center): (—) R_{11}; (---) R_{22}; (···) R_{33}.

About mean flow properties, the mean-velocity profile normalized by the centerline velocity across the section of the channel in global coordinates $\overline{u}/U_c$ is shown in Fig. 6.32; in Fig. 6.33, the mean velocity profile normalized by the friction velocity in

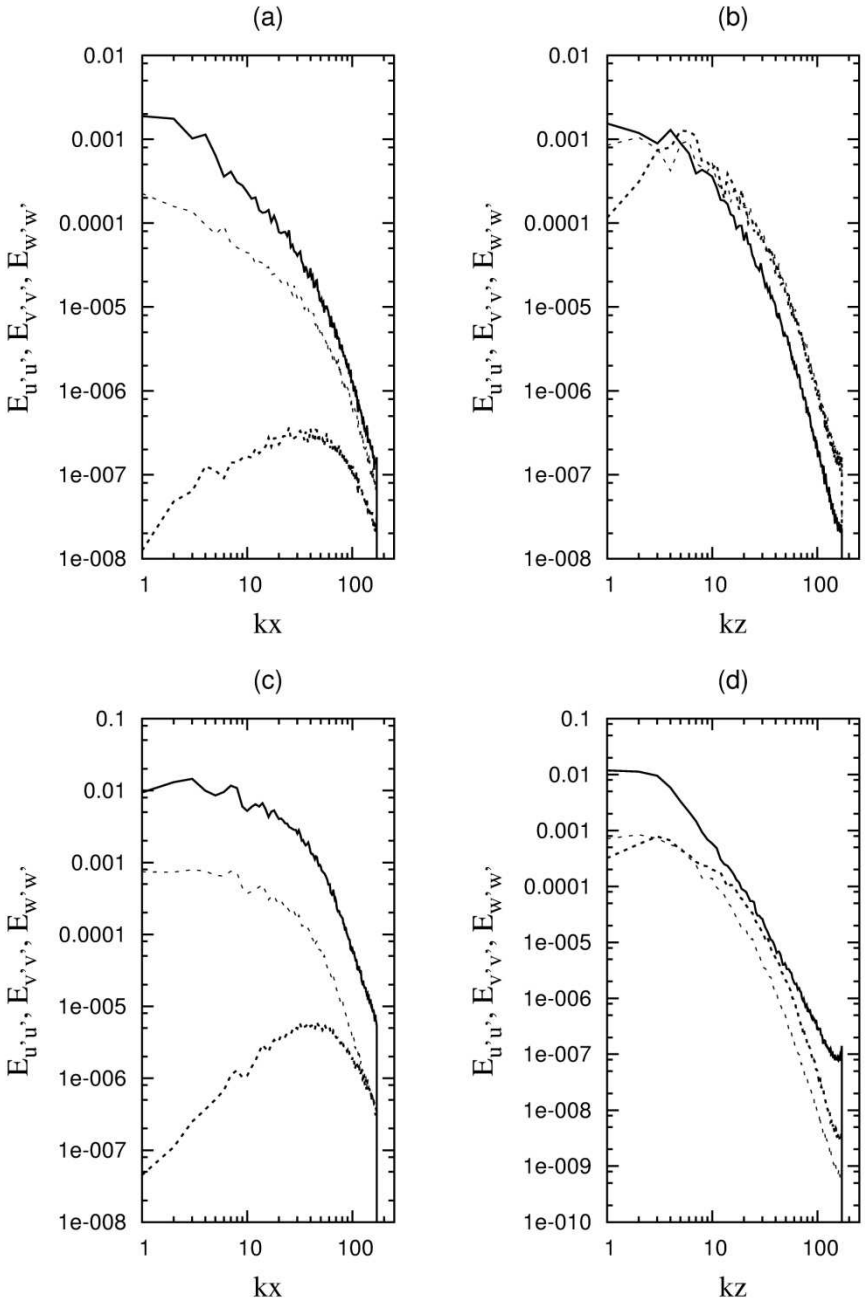

Figure 6.31 – One-dimensional energy spectra. (a) and (b) at $y/h = -0.990$ (wall), (c) and (d) at $y/h = -0.0155$ (center): (—) $E_{u'u'}$; (---); $E_{v'v'}$; (···) $E_{w'w'}$.

wall coordinates u^+ is compared with the law of the wall and with the results of [27] at $\mathrm{Re}_\tau = 590$. Also in this case, the comparison is satisfactory. Fig. 6.33 shows that the viscous sublayer is well solved, yielding the linear velocity distribution $u^+ = y^+$ for $y^+ < 5$.

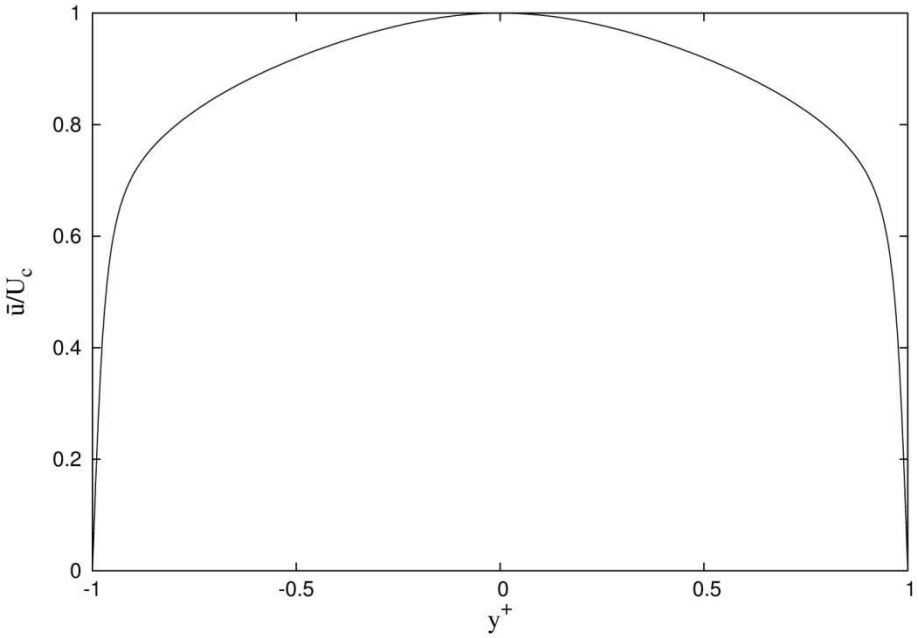

Figure 6.32 – Mean-velocity profile $\overline{u}/U_c$ across the channel in global coordinates y/h.

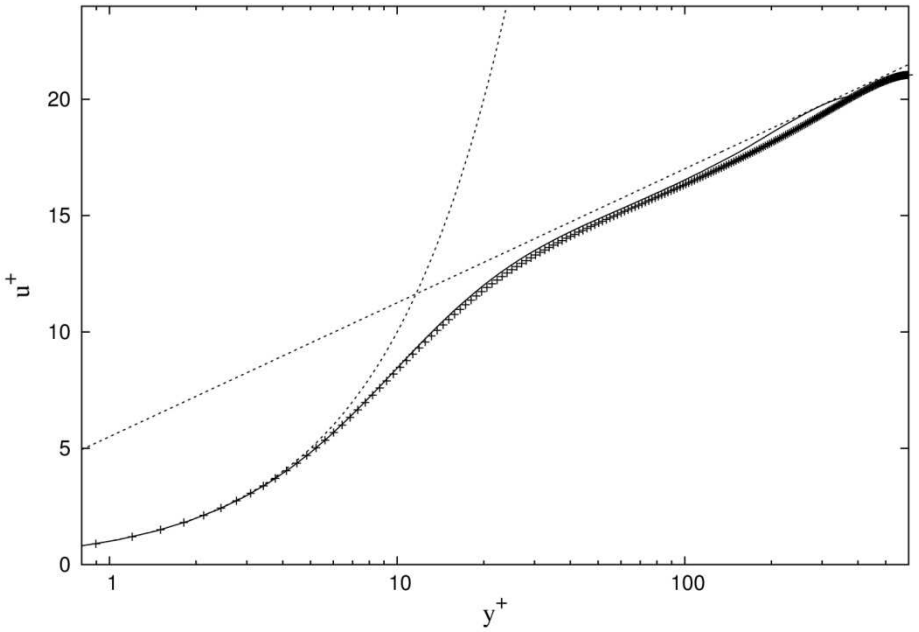

Figure 6.33 – Mean-velocity profile u^+ in wall coordinates y^+: (+) present case study, (—) data from [27]; (---) law at the wall.

In order to complete the mean flow analysis, Tab. 6.4 shows computed mean flow variables by using Eqs. 6.1-6.3 and 6.6-6.7.

Table 6.4 – Computed mean flow variables (nominal $\mathrm{Re}_\tau = 600$).

Re_τ	Re_b	Re_c	U_b/u_τ	$\overline{u}_c/u_\tau$	$\overline{u}_c/U_b$	C_{fb}	C_{fc}
600.55	11106.17	12701.63	18.49	21.15	1.14	$6.86\cdot10^{-3}$	$6.48\cdot10^{-3}$

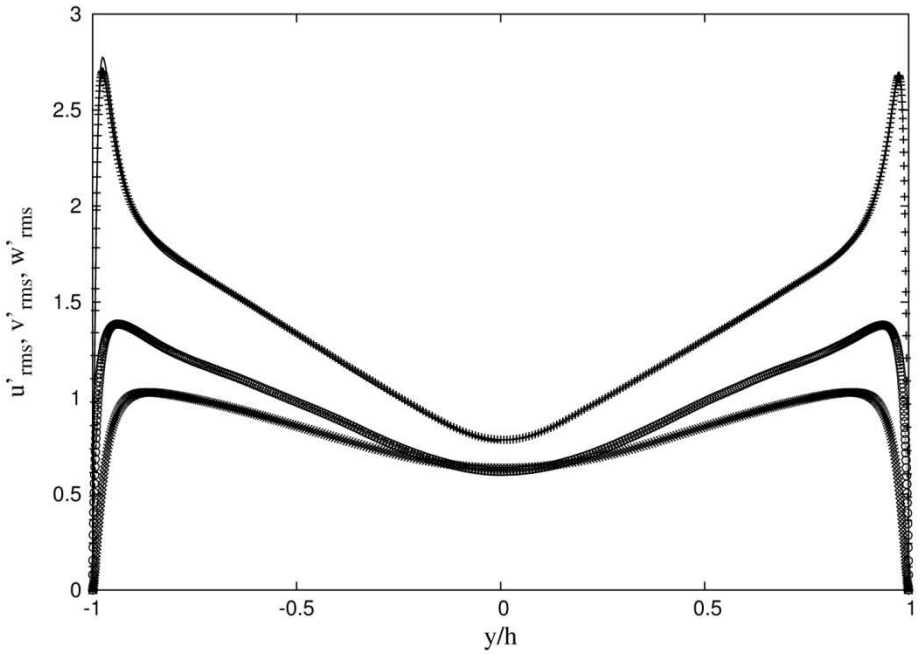

Figure 6.34 – *Rms* of velocity fluctuation in global coordinates. Present case study: (+) u'_{rms} ; (×) v'_{rms} ; (○) w'_{rms} . Data from [27]: (—) u'_{rms} ; (⋯) v'_{rms} ; (---) w'_{rms} .

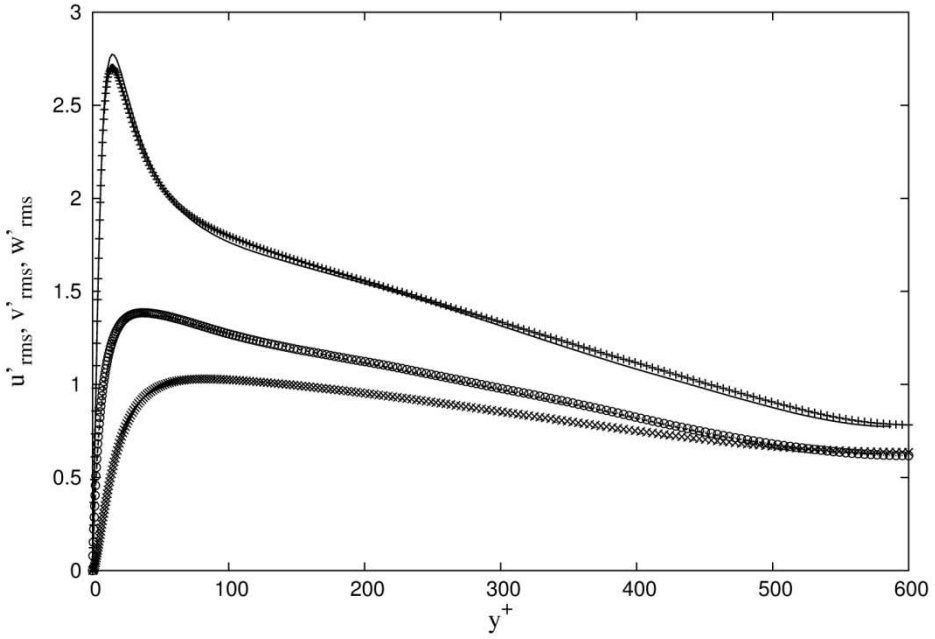

Figure 6.35 – *Rms* of velocity fluctuation in wall coordinates. Present case study: (+) u'_{rms} ; (×) v'_{rms} ; (○) w'_{rms} . Data from [27]: (—) u'_{rms} ; (⋯) v'_{rms} ; (---) w'_{rms} .

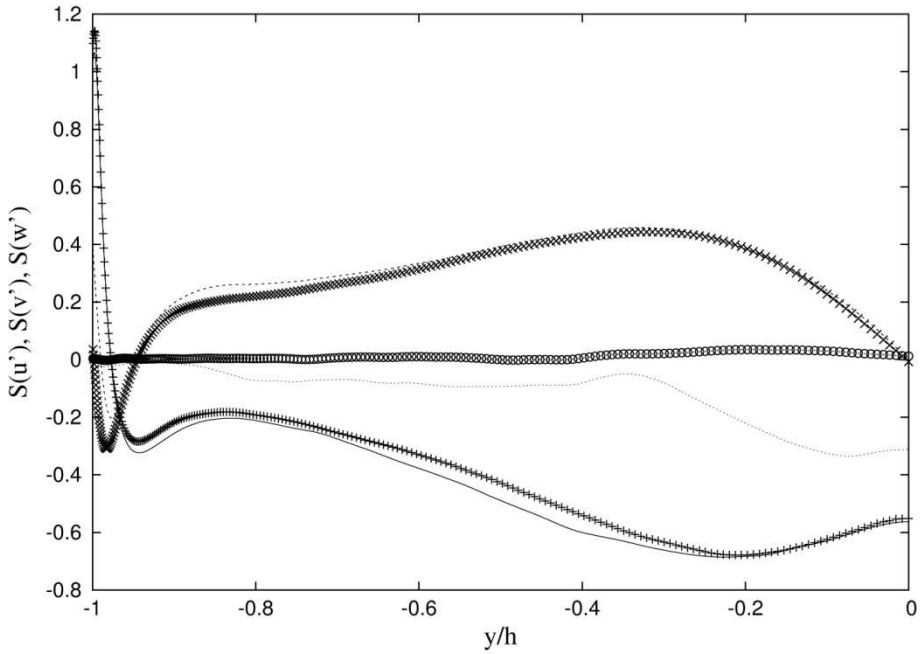

Figure 6.36 – Skewness factors of the velocity fluctuations in global coordinates. Present case study: (+) $S_{u'}$; (×) $S_{v'}$; (○) $S_{w'}$. Data from [27]: (—) $S_{u'}$; (···) $S_{v'}$; (---) $S_{w'}$.

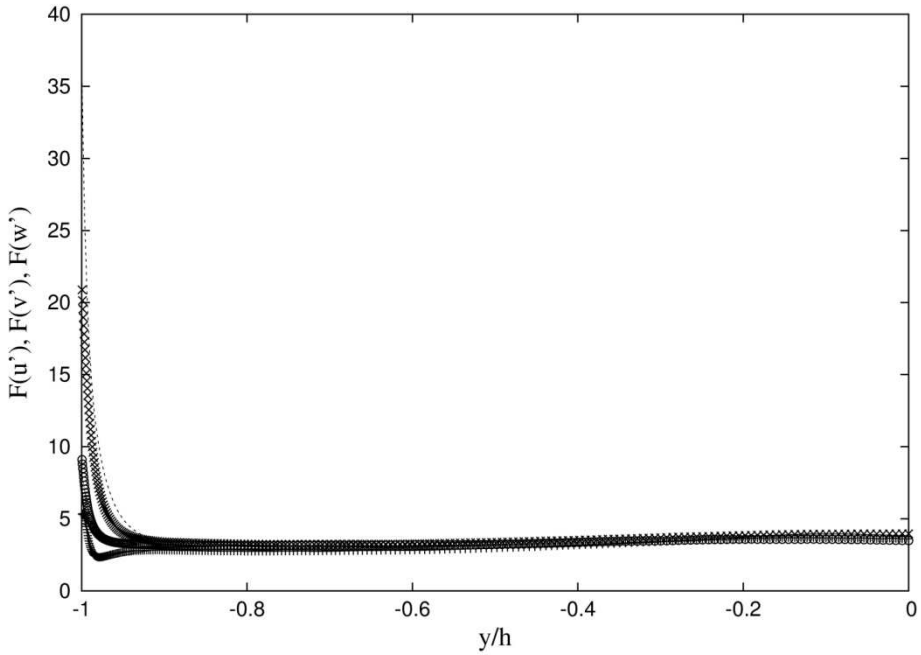

Figure 6.37 – Flatness factors of the velocity fluctuations in global coordinates. Present case study: (+) $F_{u'}$; (×) $F_{v'}$; (○) $F_{w'}$. Data from [27]: (—) $F_{u'}$; (···) $F_{v'}$; (---) $F_{w'}$.

Figures 6.34 and 6.35 report the turbulent intensities $u'_{rms}, v'_{rms}, w'_{rms}$ – the root mean square values of the velocities fluctuations normalized by the friction velocity – in global and wall coordinates, respectively.

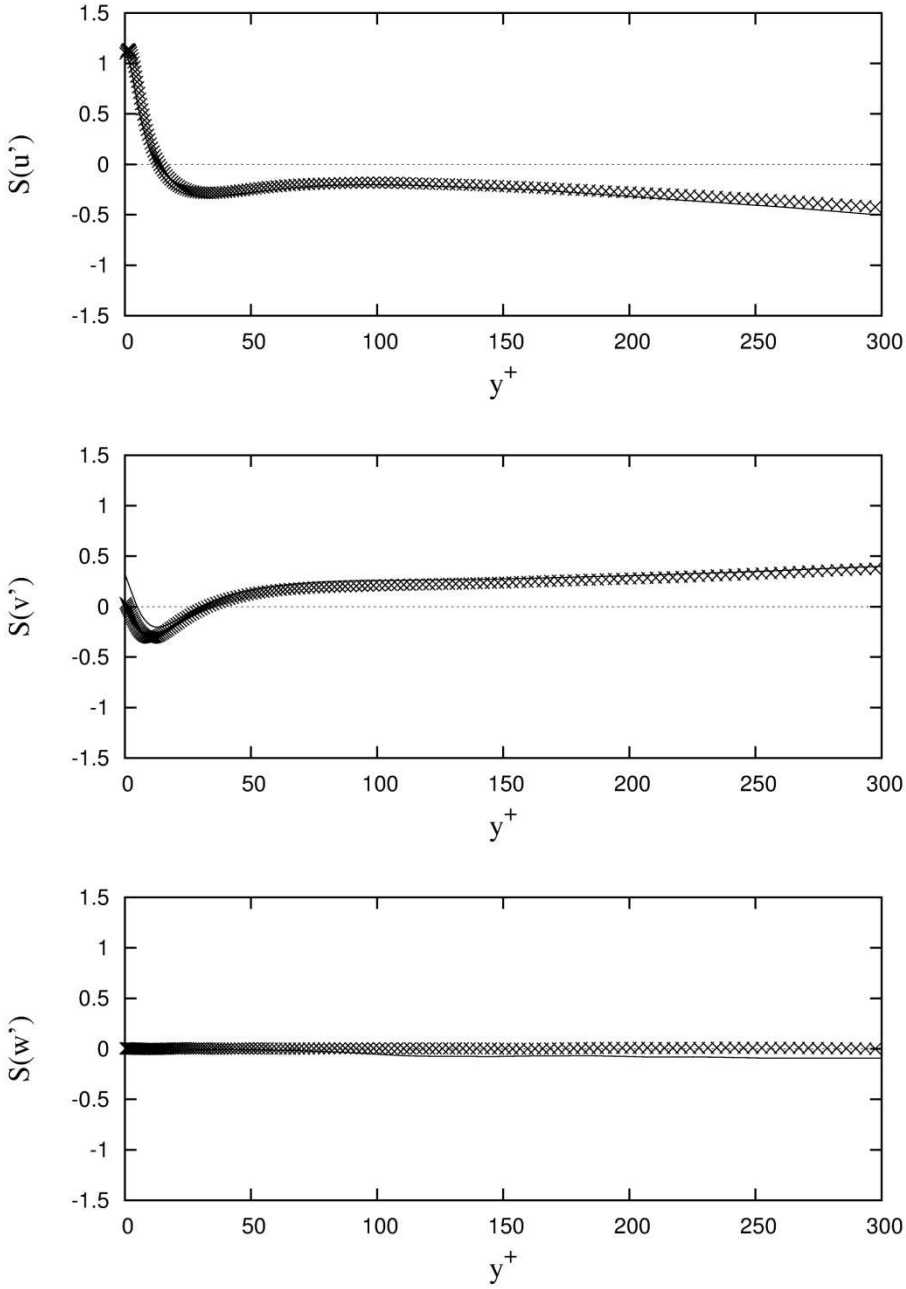

Figure 6.38 – Skewness factors of the velocity fluctuations wall coordinates: (×) present work; (—) data from [27].

It is evident the symmetry of the profiles about the centerline, that confirms the adequacy of the simulation sample taken for the average. The agreement of computed results with those of [27] is good.

Figures 6.36 and 6.38 show the skewness factors $S_{u'}, S_{v'}, S_{w'}$ of the velocity fluctuations in global and wall coordinates, respectively, in comparison with the

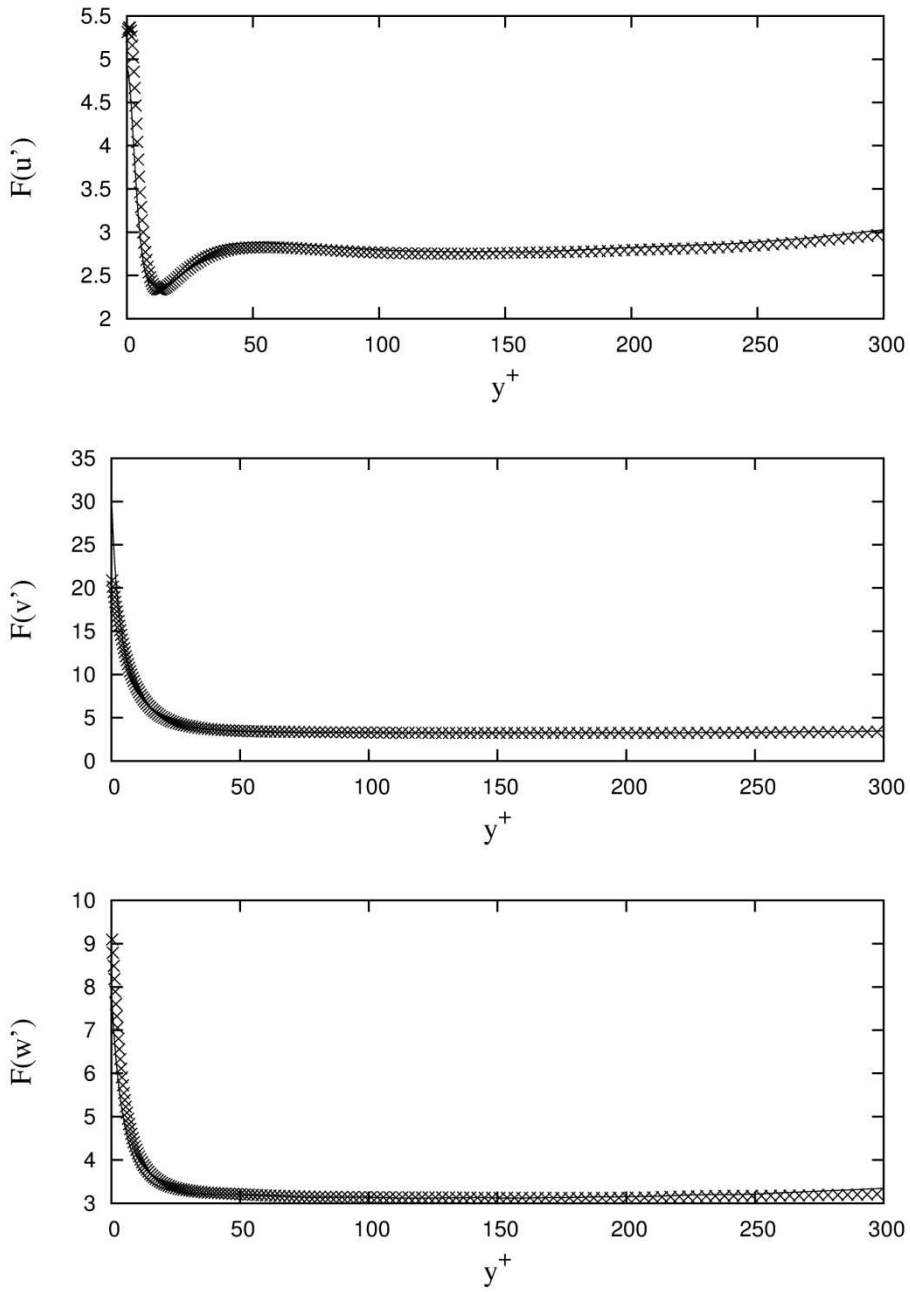

Figure 6.39 – Flatness factors of the velocity fluctuations wall coordinates: (×) present work; (—) data from [27].

results of [27] at $\mathrm{Re}_\tau = 590$. Similar comparisons are shown in Fig. 6.37 and 6.39 for the flatness factors $F_{u'}, F_{v'}, F_{w'}$. For both skewness and flatness factors, the profiles in global coordinates show small asymmetries and oscillations and the related values are significantly different from the Gaussian ones (0 and 3, respectively). The

profiles of both skewness and flatness factors, in global coordinates, show a satisfactory agreement with the computed results of [27]. The same conclusion can be done for the profiles of both skewness and flatness factors in wall coordinates.

6.3 Reynolds shear stress: distribution and budgets

In the field of wall-bounded turbulence, the Reynolds shear stress play an important role for understanding how an active control of the physical mechanisms that occur in the near-wall regions is possible by means models and flow structures analysis. In this section, a detailed description of the numerical results obtained from the DNS of a plane channel at $\mathrm{Re}_\tau = 200, 400, 600$ are shown.

The analysis has been done considering, at first, the values of the Reynolds shear stress $\overline{u'v'}$ in comparison with those of the total shear stress τ_{tot} defined as:

$$\tau_{tot} = -\overline{u'v'} + \frac{1}{\mathrm{Re}_\tau}\frac{\partial \overline{u}}{\partial y} \tag{6.9}$$

Then, starting from the Reynolds stress transport equations, as described in *Appendix A*, the terms referred to the Reynolds stress budget, the dissipation-rate budget and the anisotropy-rate budget are calculated and discussed for each numerical database simulated. The relevant non-zero stresses in this case are $\overline{u_1u_1}, \overline{u_1u_2}, \overline{u_2u_2}, \overline{u_3u_3}$.

6.3.1 Analysis of the terms at $\mathrm{Re}_\tau = 200$

Figures 6.40 and 6.41 report the values of the Reynolds shear stress $\overline{u'v'}$ computed, normalized by the friction velocity, in global and wall coordinates, respectively. Figure 6.40 reports, also, the computed total shear stress τ_{tot}, as in Eq. 6.9. The straight dotted line developing across the channel width in an indicator of the fully developed condition reached in the numerical simulation. In Fig. 6.42, the numerical results are compared with the data of [27] obtained at $\mathrm{Re}_\tau = 180$.

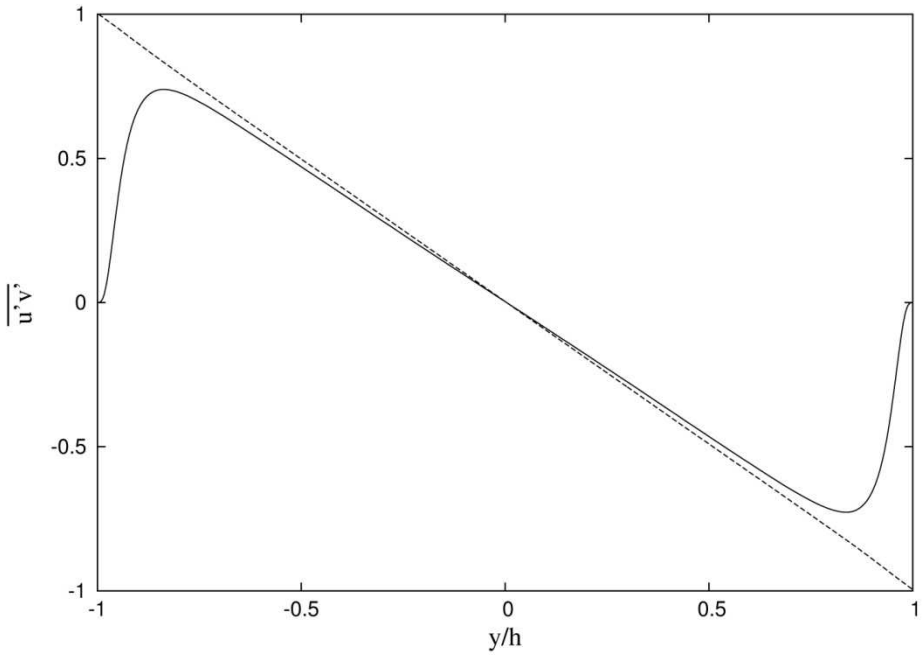

Figure 6.40 – Computed values of Reynolds shear stress and total shear stress normalized by the friction velocity in global coordinates. Present work: (—) $\overline{u'v'}$, (---) τ_{tot}.

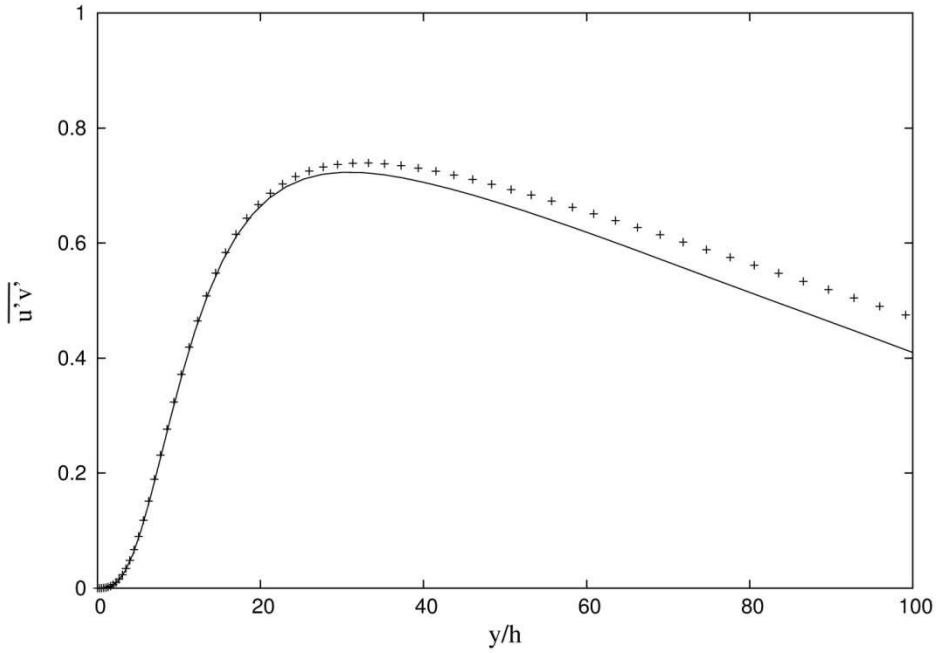

Figure 6.41 – Reynolds shear stress normalized by the friction velocity in wall coordinates. Present work: (+); data from [27] at $\mathrm{Re}_\tau = 180$: (—).

Figures 6.42-6.45 show the terms in the budget of those stresses, while Fig. 6.46 shows the terms of the budget for the turbulent kinetic energy K.

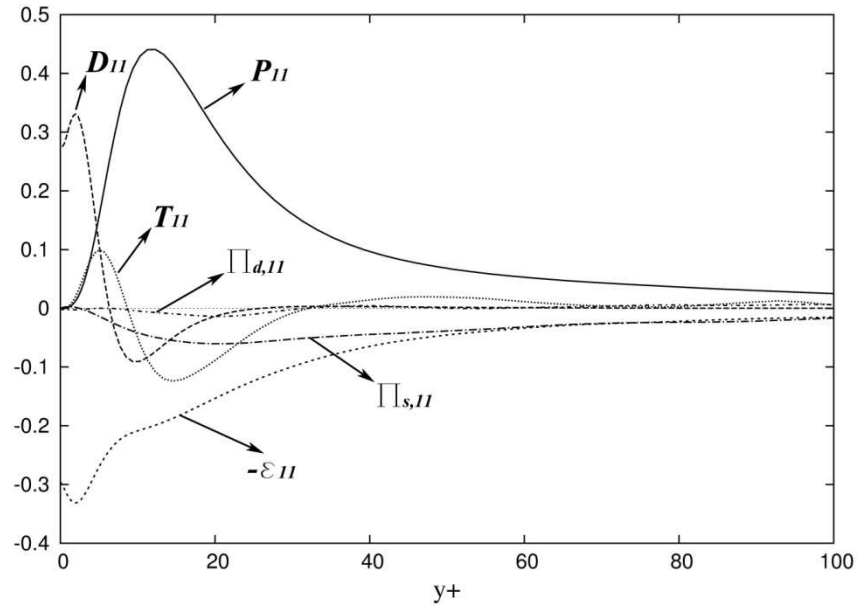

Figure 6.42 – Terms in the budget of $\overline{u_1'u_1'}$ in wall coordinates. P_{11} = Production; T_{11} = Turbulent transport; D_{11} = Viscous diffusion; ε_{11} = Dissipation rate; $\Pi_{s,11}$ = Velocity pressure strain gradient term; $\Pi_{d,11}$ = Velocity pressure diffusion gradient term.

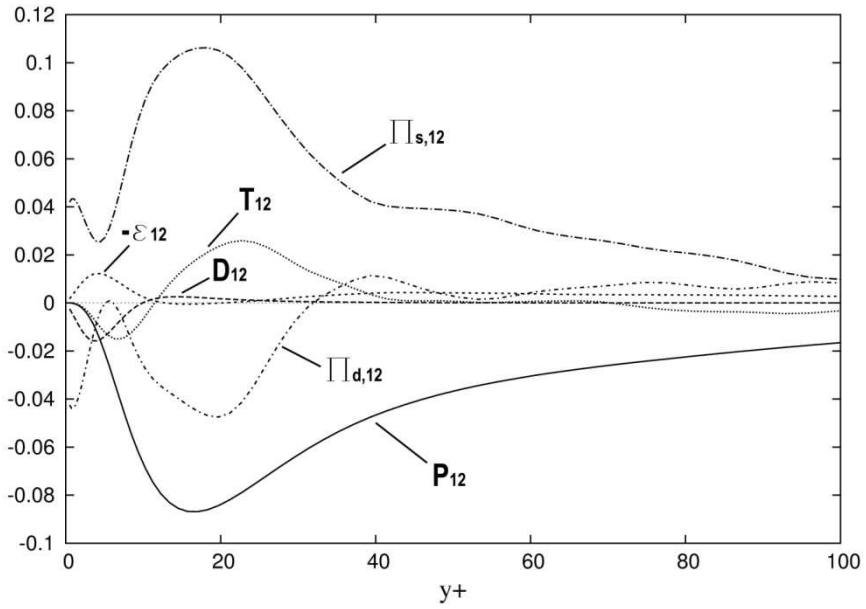

Figure 6.43 – Terms in the budget of $\overline{u_1'u_2'}$ in wall coordinates. P_{12} = Production; T_{12} = Turbulent transport; D_{12} = Viscous diffusion; ε_{12} = Dissipation rate; $\Pi_{s,12}$ = Velocity pressure strain gradient term; $\Pi_{d,12}$ = Velocity pressure diffusion gradient term.

Considering Eq. (A.78), Fig. 6.47 shows the terms that defined the combination of the turbulence kinetic energy and the dissipation rate term.

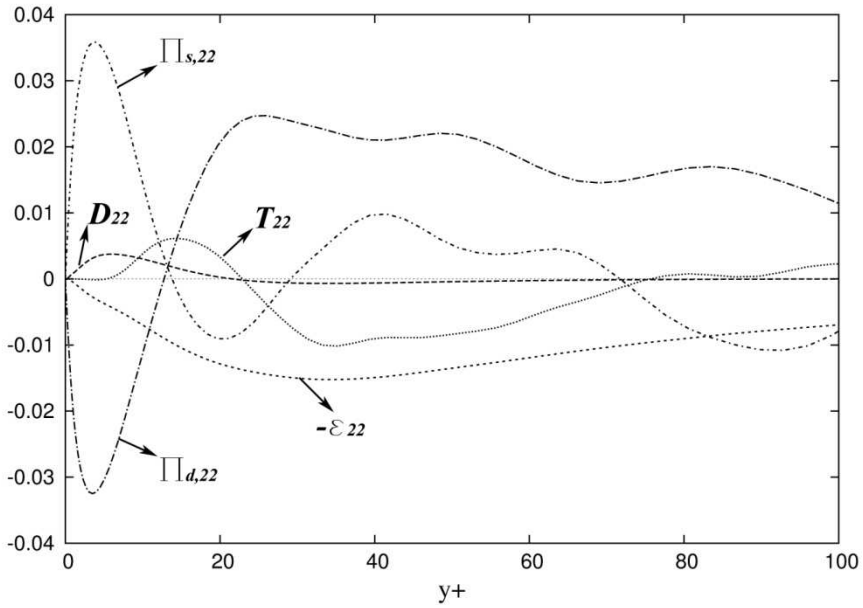

Figure 6.44 – Terms in the budget of $\overline{u_2'u_2'}$ in wall coordinates. T_{22} = Turbulent transport; D_{22} = Viscous diffusion; ε_{22} = Dissipation rate; $\Pi_{s,22}$ = Velocity pressure strain gradient term; $\Pi_{d,22}$ = Velocity pressure diffusion gradient term.

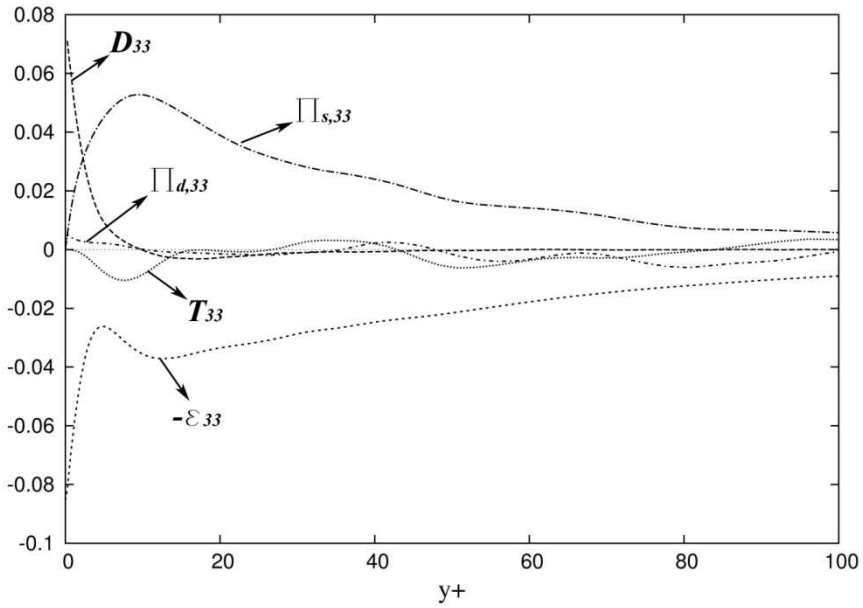

Figure 6.45 – Terms in the budget of $\overline{u_3'u_3'}$ in wall coordinates. T_{33} = Turbulent transport; D_{33} = Viscous diffusion; ε_{33} = Dissipation rate; $\Pi_{s,33}$ = Velocity pressure strain gradient term; $\Pi_{d,33}$ = Velocity pressure diffusion gradient term.

Finally, Figs. 6.48-6.51 show the terms referred to the transport equation for the Reynolds stress anisotropy tensor b_{ij}, as in Eq. (A.68).

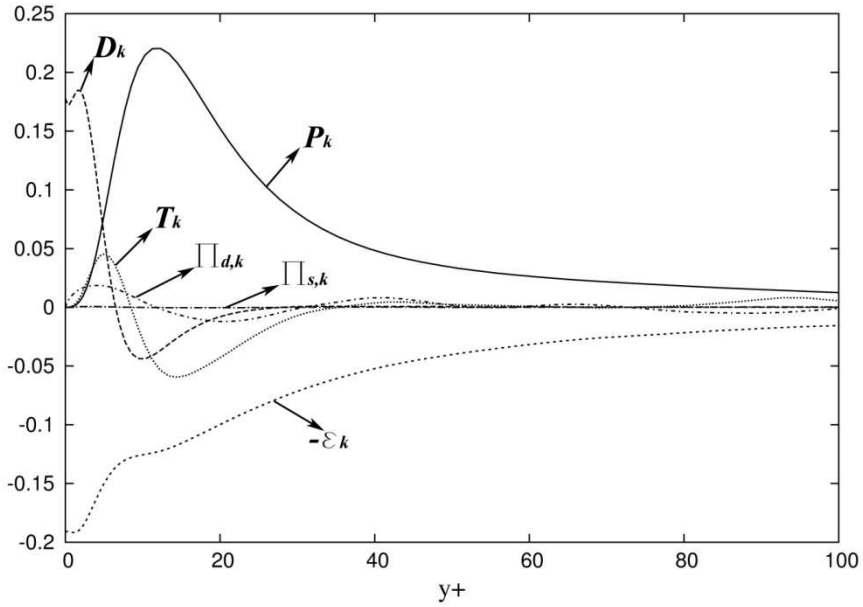

Figure 6.46 – Terms in the budget of the turbulent kinetic energy k in wall coordinates. P_k = Production; T_k = Turbulent transport; D_k = Viscous diffusion; ε_k = Dissipation rate; $\Pi_{s,k}$ = Velocity pressure strain gradient term; $\Pi_{d,k}$ = Velocity pressure diffusion gradient term.

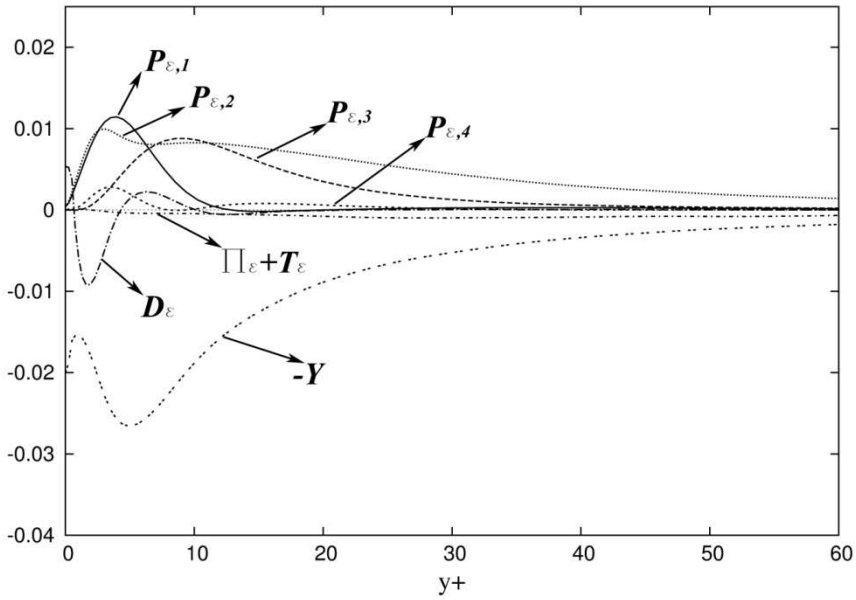

Figure 6.47 – Terms in the budget of the dissipation rate of the turbulence kinetic energy in wall coordinates. $P_{\varepsilon,1}$ = Production by mean velocity gradient; $P_{\varepsilon,2}$ = Mixed production; $P_{\varepsilon,3}$ = Gradient production; $P_{\varepsilon,4}$ = Turbulent production; T_ε = Turbulent transport; D_ε = Viscous diffusion; Y = Dissipation rate; Π_ε = Pressure transport.

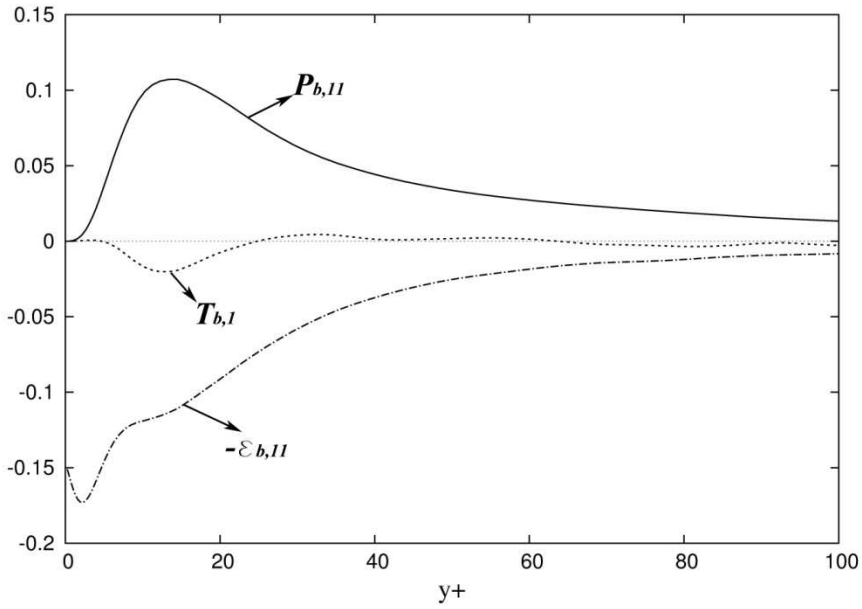

Figure 6.48 – Terms in the budget of the anisotropy rate of the turbulence kinetic energy in wall coordinates. $P_{b,11}$ = Production; $T_{b,11}$ = Turbulent transport; $\varepsilon_{b,11}$ = Dissipation rate.

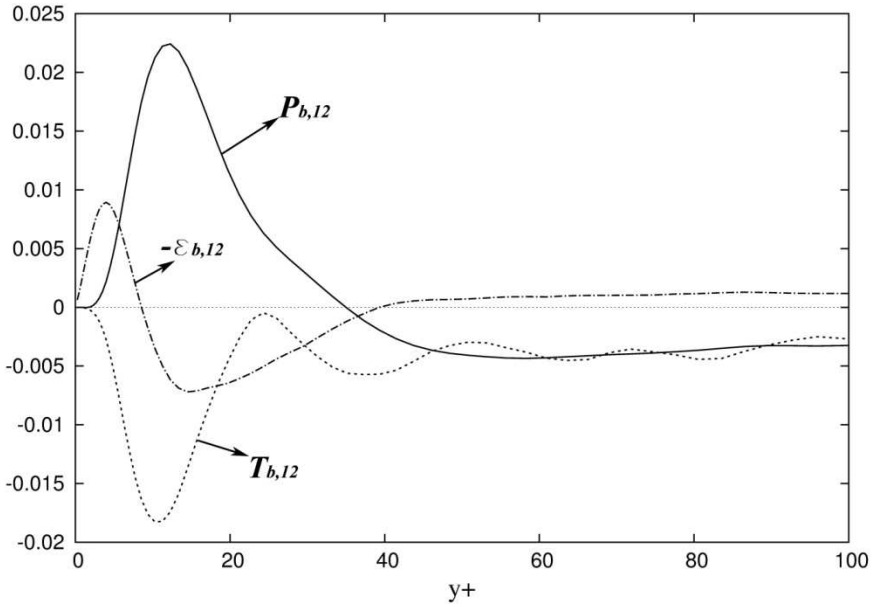

Figure 6.49 – Terms in the budget of the anisotropy rate of the turbulence kinetic energy in wall coordinates. $P_{b,12}$ = Production; $T_{b,12}$ = Turbulent transport; $\varepsilon_{b,12}$ = Dissipation rate.

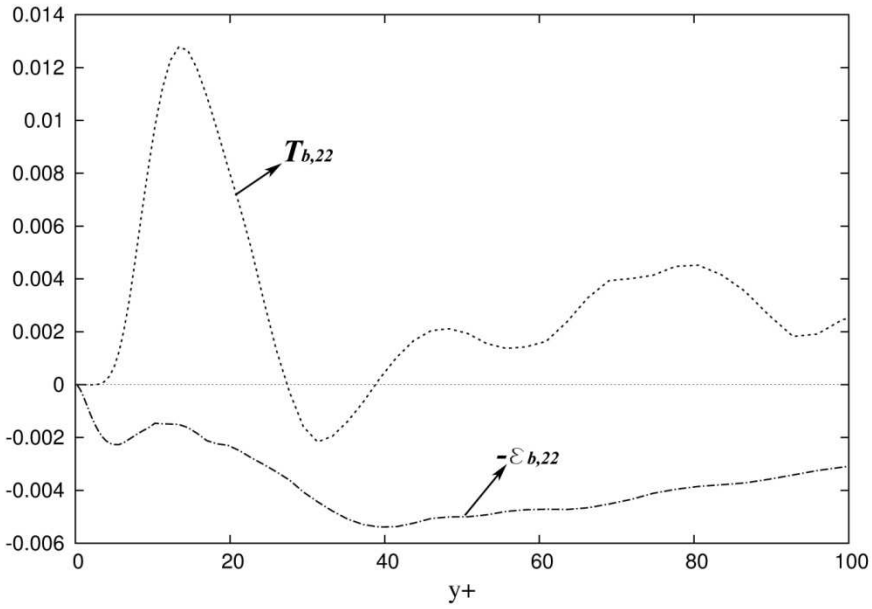

Figure 6.50 – Terms in the budget of the anisotropy rate of the turbulence kinetic energy in wall coordinates. $T_{b,22}$ = Turbulent transport; $\varepsilon_{b,22}$ = Dissipation rate.

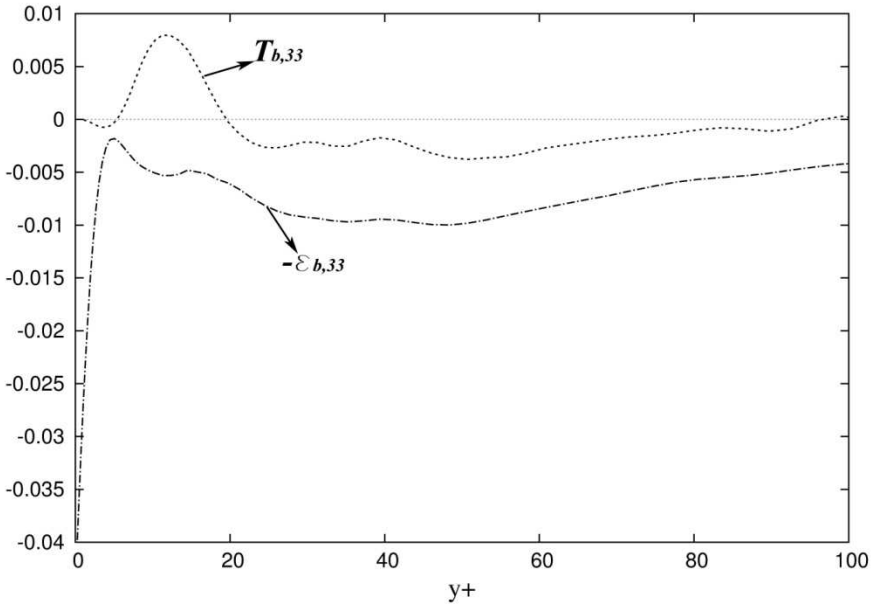

Figure 6.51 – Terms in the budget of the anisotropy rate of the turbulence kinetic energy in wall coordinates. $T_{b,33}$ = Turbulent transport; $\varepsilon_{b,33}$ = Dissipation rate.

6.3.2 Analysis of the terms at $\mathbf{Re_\tau = 400}$

Figures 6.52 and 6.53 report the values of the Reynolds shear stress $\overline{u'v'}$ computed, normalized by the friction velocity, in global and wall coordinates, respectively. Figure 6.52 reports, also, the computed total shear stress τ_{tot}, as in Eq. (6.9). The straight dotted line developing across the channel width in an indicator of the fully developed condition reached in the numerical simulation. In Fig. 6.52, the numerical results are compared with the data of [27] obtained at $\mathrm{Re}_\tau = 395$. Figures 6.54-(6.57) show the terms in the budget of these stresses, while Fig. 6.58 shows the terms of the budget for the turbulent kinetic energy K.

Figure 6.59 shows the terms that defined the combination of the turbulence kinetic energy and the dissipation rate term.

Finally, Figs. 6.60-6.63 show the terms referred to the transport equation for the Reynolds stress anisotropy tensor b_{ij}.

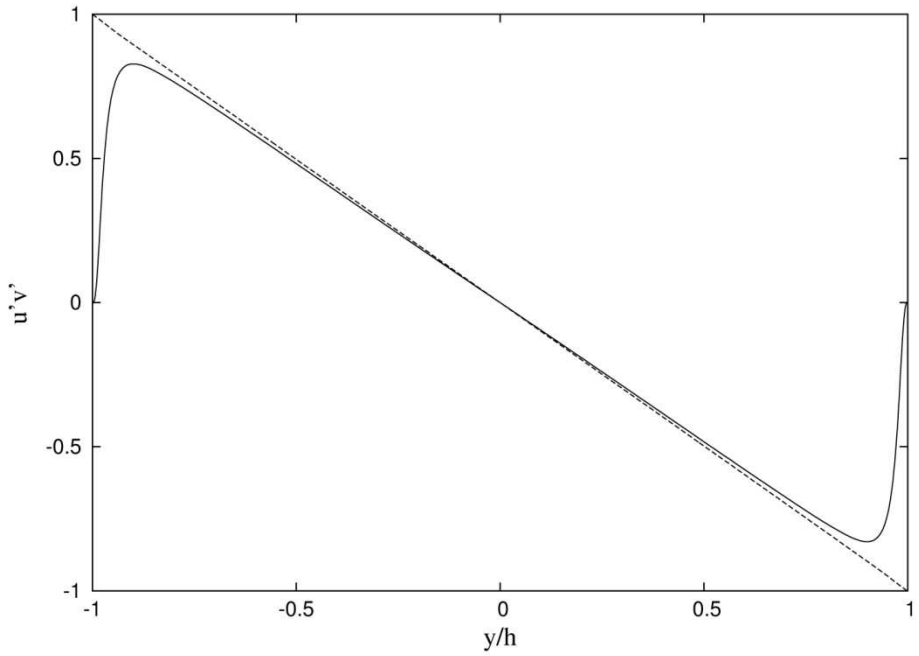

Figure 6.52 – Computed values of Reynolds shear stress and total shear stress normalized by the friction velocity in global coordinates. Present work: (—) $\overline{u'v'}$, (---) τ_{tot}.

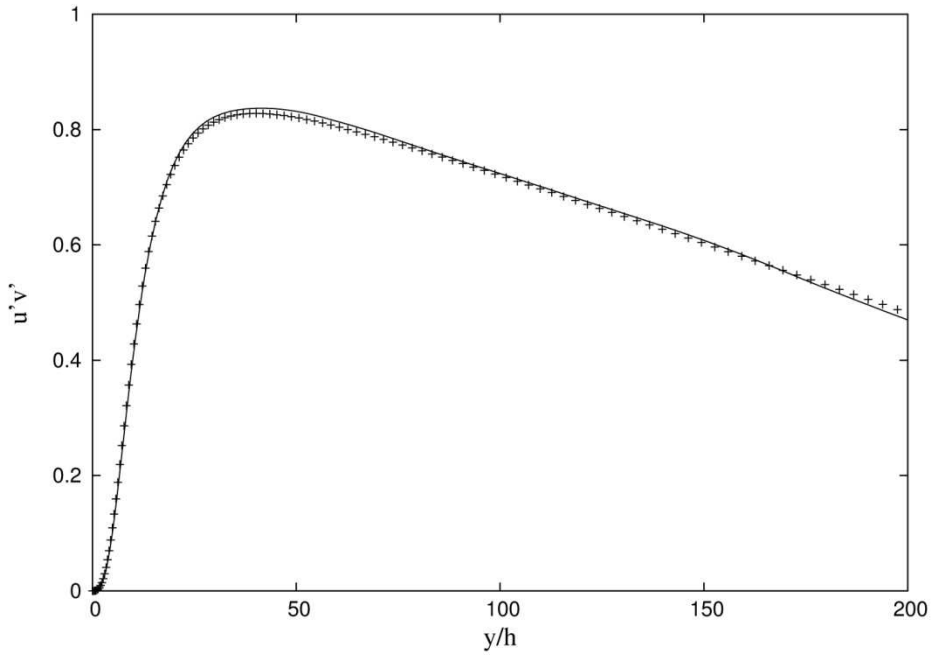

Figure 6.53 – Reynolds shear stress $\overline{u'v'}$ normalized by the friction velocity in wall coordinates. Present work: (+); data from [27] (─).

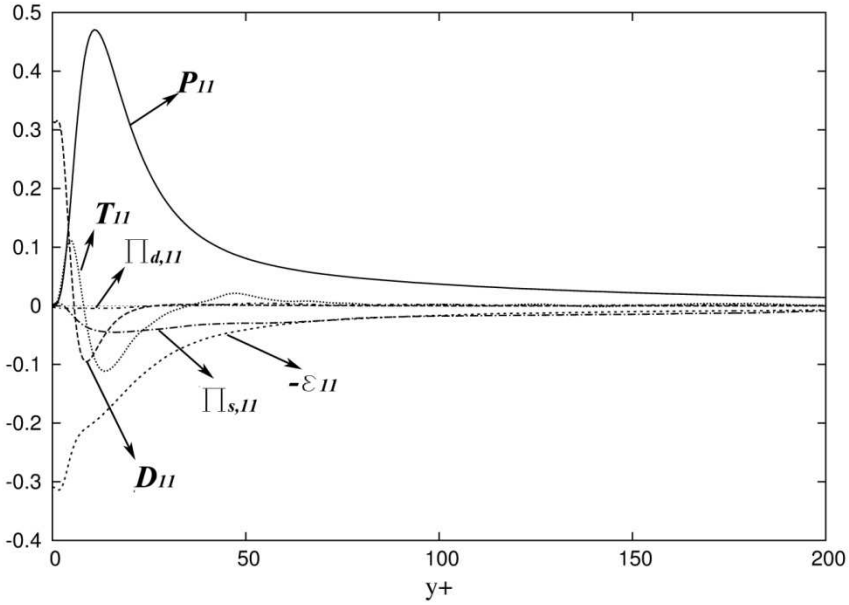

Figure 6.54 – Terms in the budget of $\overline{u_1'u_1'}$ in wall coordinates. P_{11} = Production; T_{11} = Turbulent transport; D_{11} = Viscous diffusion; ε_{11} = Dissipation rate; $\Pi_{s,11}$ = Velocity pressure strain gradient term; $\Pi_{d,11}$ = Velocity pressure diffusion gradient term.

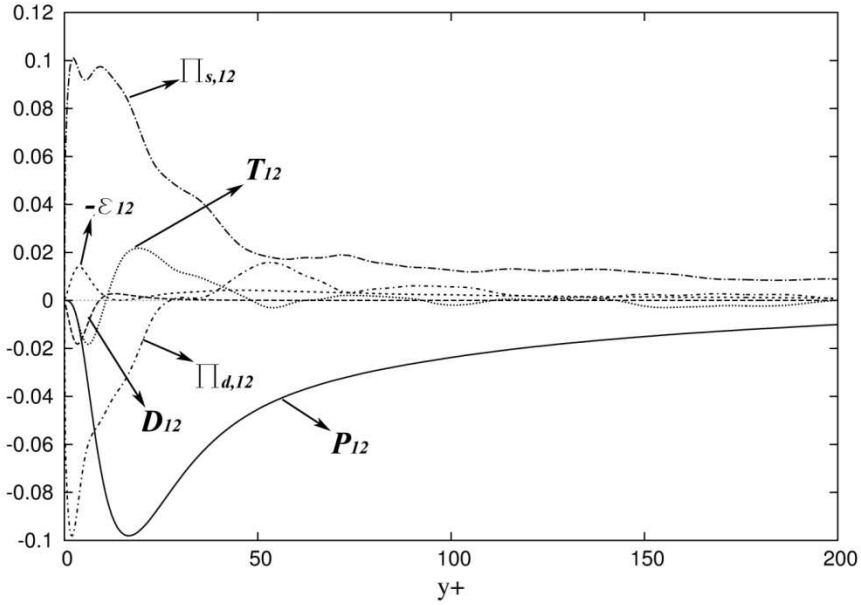

Figure 6.55 – Terms in the budget of $\overline{u_1'u_2'}$ in wall coordinates. P_{12} = Production; T_{12} = Turbulent transport; D_{12} = Viscous diffusion; ε_{12} = Dissipation rate; $\Pi_{s,12}$ = Velocity pressure strain gradient term; $\Pi_{d,12}$ = Velocity pressure diffusion gradient term.

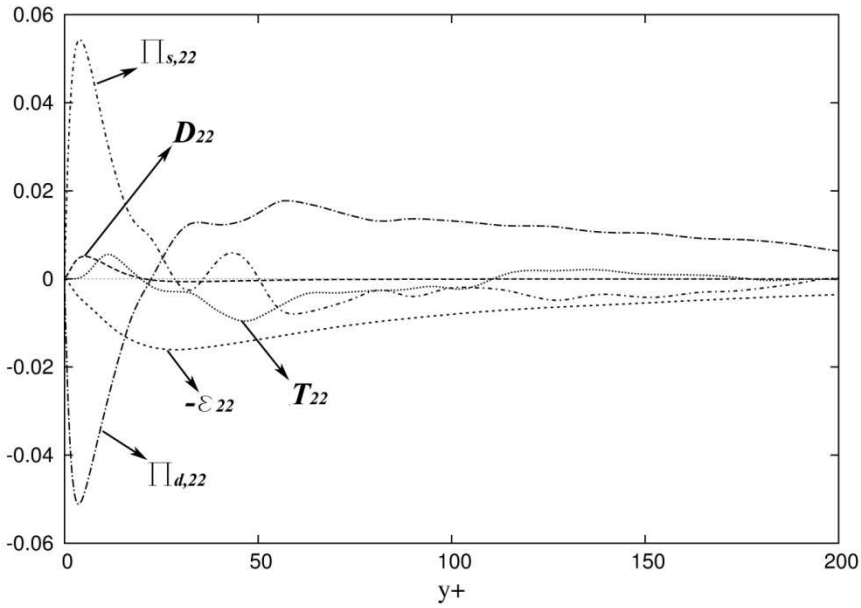

Figure 6.56 – Terms in the budget of $\overline{u_2'u_2'}$ in wall coordinates. T_{22} = Turbulent transport; D_{22} = Viscous diffusion; ε_{22} = Dissipation rate; $\Pi_{s,22}$ = Velocity pressure strain gradient term; $\Pi_{d,22}$ = Velocity pressure diffusion gradient term.

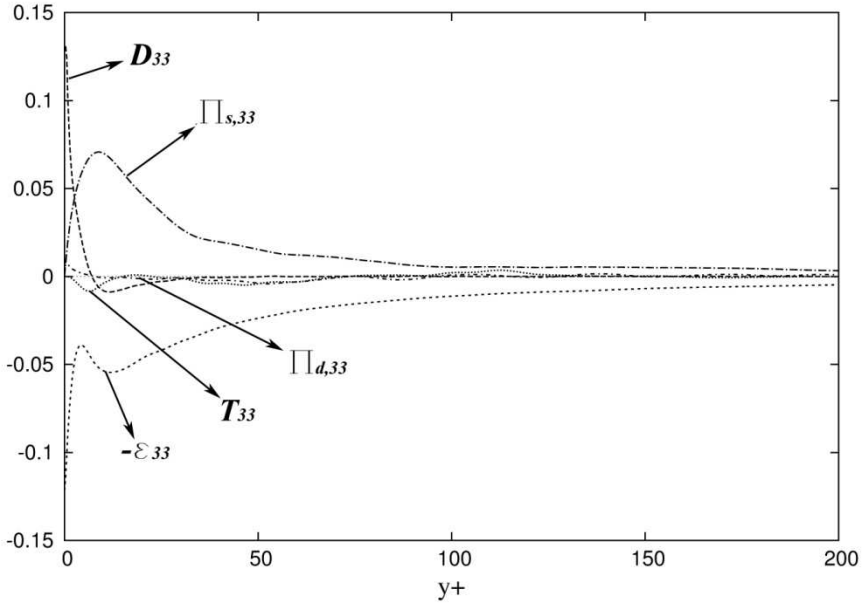

Figure 6.57 – Terms in the budget of $\overline{u_3'u_3'}$ in wall coordinates. T_{33} = Turbulent transport; D_{33} = Viscous diffusion; ε_{33} = Dissipation rate; $\Pi_{s,33}$ = Velocity pressure strain gradient term; $\Pi_{d,33}$ = Velocity pressure diffusion gradient term.

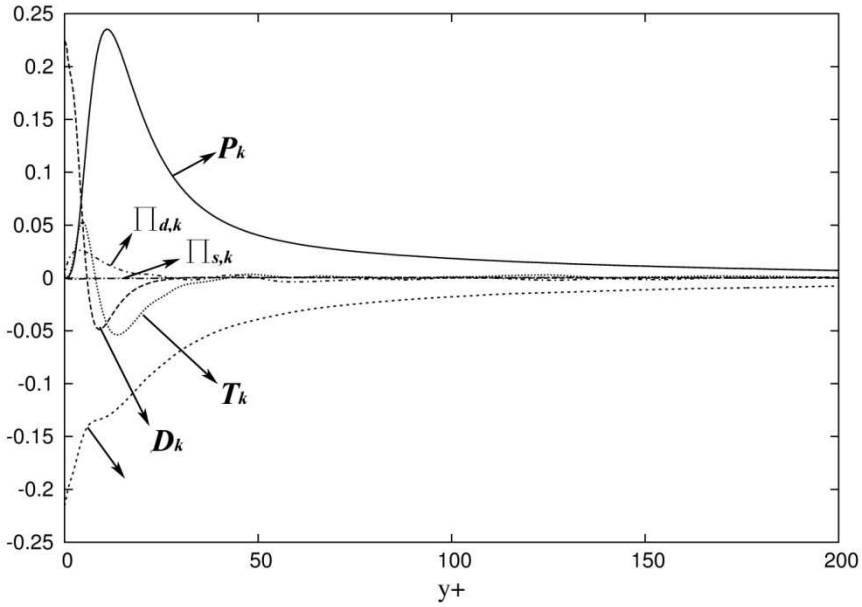

Figure 6.58 – Terms in the budget of the turbulent kinetic energy in wall coordinates. P_K = Production; T_K = Turbulent transport; D_K = Viscous diffusion; ε_K = = Dissipation rate; $\Pi_{s,K}$ = Velocity pressure gradient term; $\Pi_{d,K}$ = Velocity pressure diffusion gradient term.

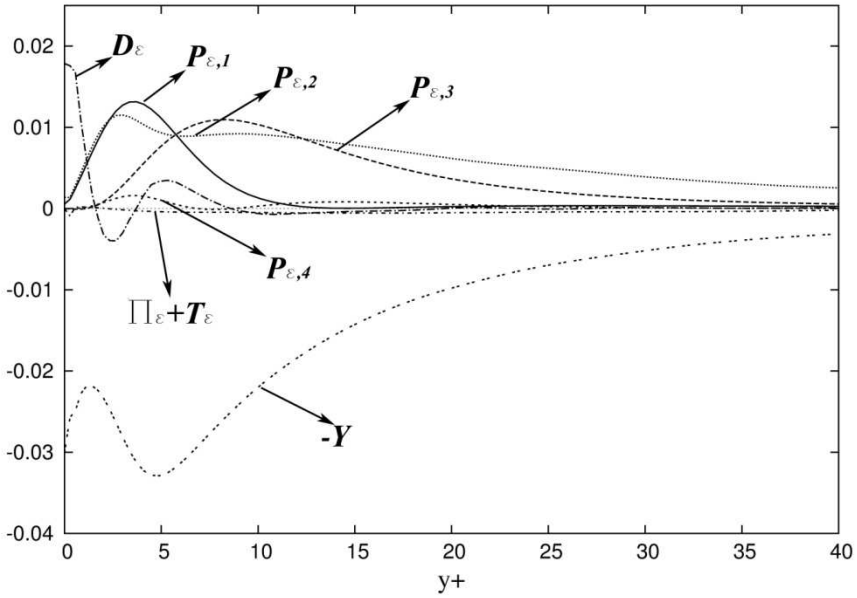

Figure 6.59 – Terms in the budget of the dissipation rate of the turbulence kinetic energy ε in wall coordinates. $P_{\varepsilon,1}$ = Production by mean velocity gradient; $P_{\varepsilon,2}$ = Mixed production; $P_{\varepsilon,3}$ = Gradient production; $P_{\varepsilon,4}$ = Turbulent production; T_ε = Turbulent transport; D_ε = Viscous diffusion; Y = Dissipation rate; Π_ε = Pressure transport.

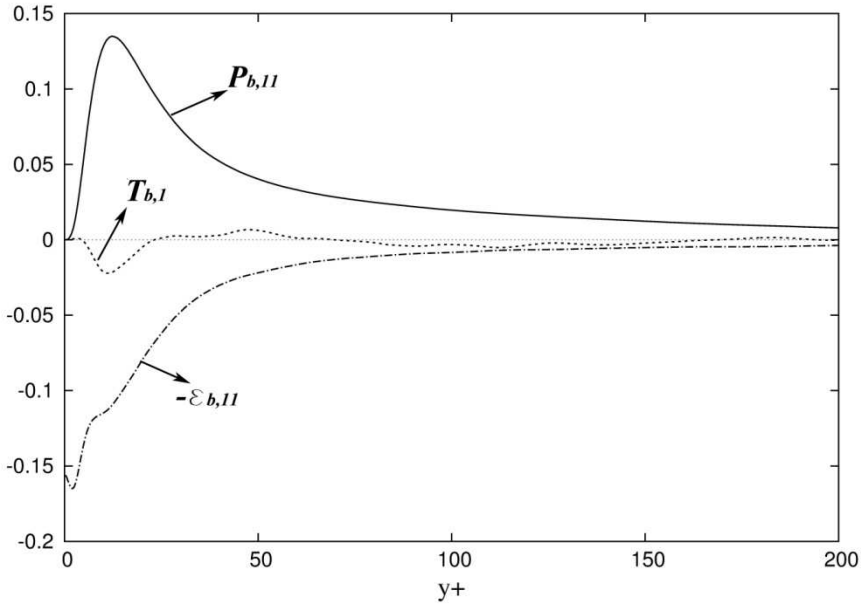

Figure 6.60 – Terms in the budget of the anisotropy rate of the turbulence kinetic energy in wall coordinates. $P_{b,11}$ = Production; $T_{b,11}$ = Turbulent transport; $\varepsilon_{b,11}$ = Dissipation rate.

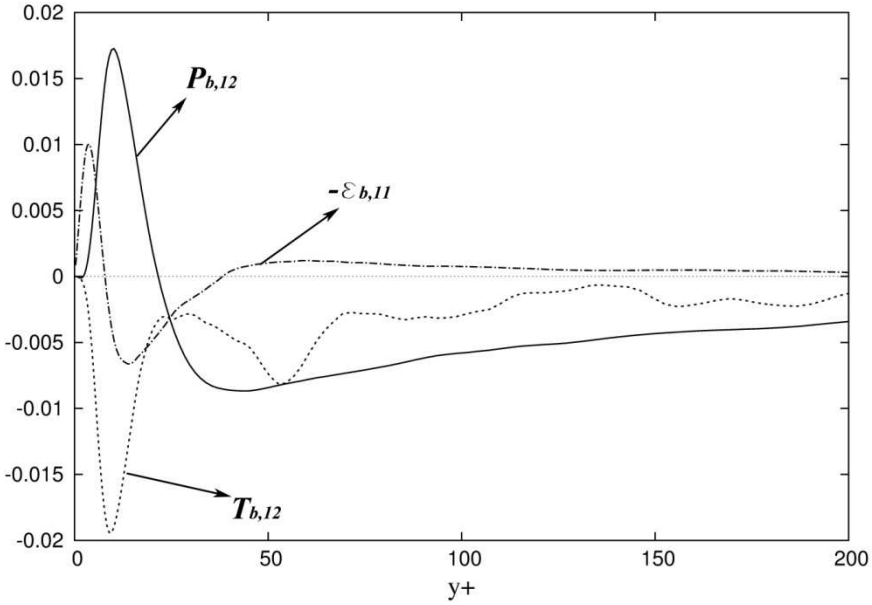

Figure 6.61 – Terms in the budget of the anisotropy rate of the turbulence kinetic energy in wall coordinates. $P_{b,12}$ = Production; $T_{b,12}$ = Turbulent transport; $\varepsilon_{b,12}$ = Dissipation rate.

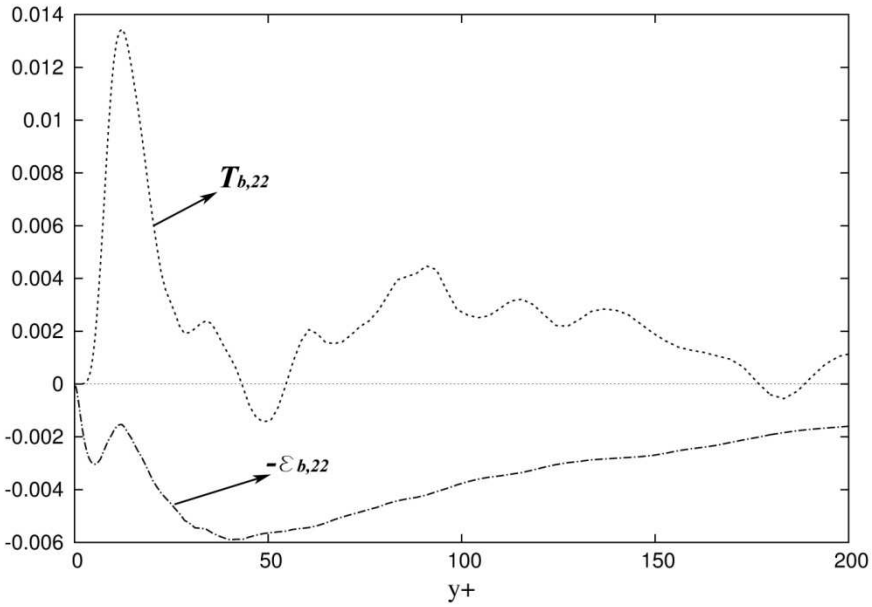

Figure 6.62 – Terms in the budget of the anisotropy rate of the turbulence kinetic energy in wall coordinates. $T_{b,22}$ = Turbulent transport; $\varepsilon_{b,22}$ = Dissipation rate.

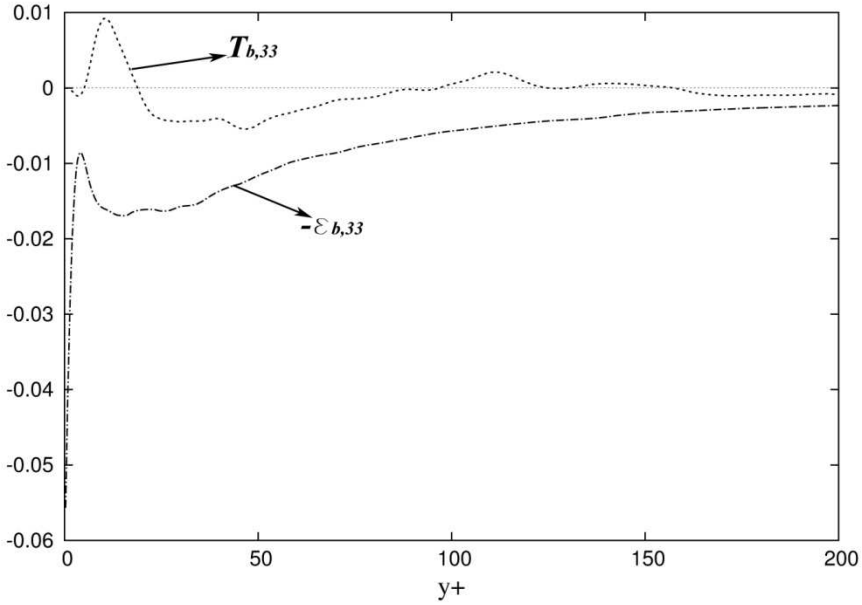

Figure 6.63 – Terms in the budget of the anisotropy rate of the turbulence kinetic energy in wall coordinates. $T_{b,33}$ = Turbulent transport; $\varepsilon_{b,33}$ = Dissipation rate.

6.3.3 Analysis of the terms at $\mathrm{Re}_\tau = 600$

Figures 6.64 and 6.65 report the values of the Reynolds shear stress $-\overline{u'v'}$ computed, normalized by the friction velocity, in global and wall coordinates, respectively. Figure 6.65 reports, also, the computed total shear stress τ_{tot} , as in Eq. (6.9). The straight dotted line developing across the channel width in an indicator of the fully developed condition reached in the numerical simulation. In Fig. 6.67, the numerical results are compared with the data of [27] obtained at $\mathrm{Re}_\tau = 590$.

Figures 6.66-6.69 show the terms in the budget of these stresses, while Fig. 6.70 shows the terms of the budget for the turbulent kinetic energy K.

Figure 6.71 shows the terms that defined the combination of the turbulence kinetic energy and the dissipation rate term.

Finally, Figs. 6.72-6.75 show the terms referred to the transport equation for the Reynolds stress anisotropy tensor b_{ij}.

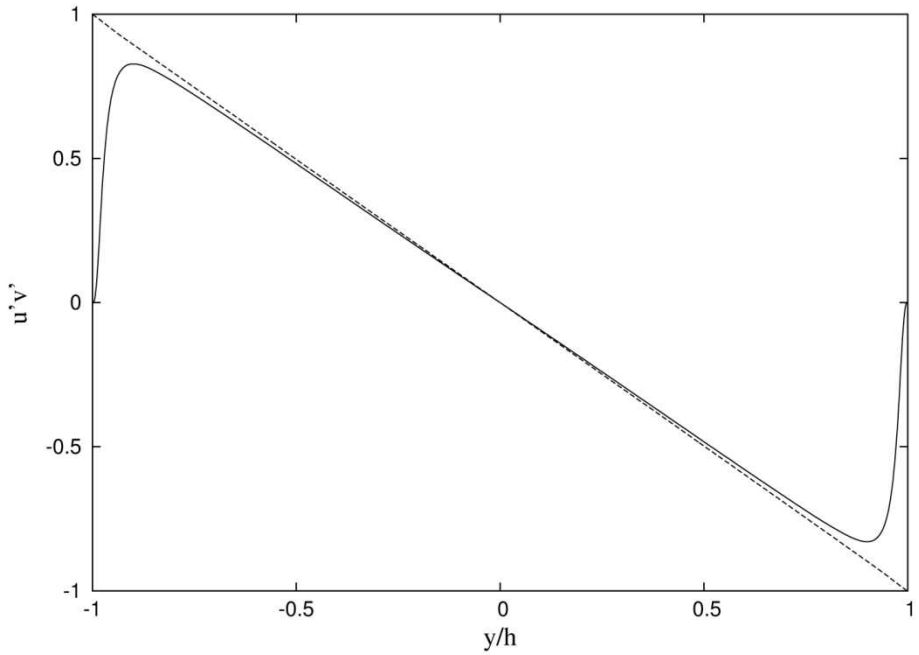

Figure 6.64 – Computed values of Reynolds shear stress and total shear stress normalized by the friction velocity in global coordinates. Present work: (—) $-\overline{u'v'}$, (---) τ_{tot}.

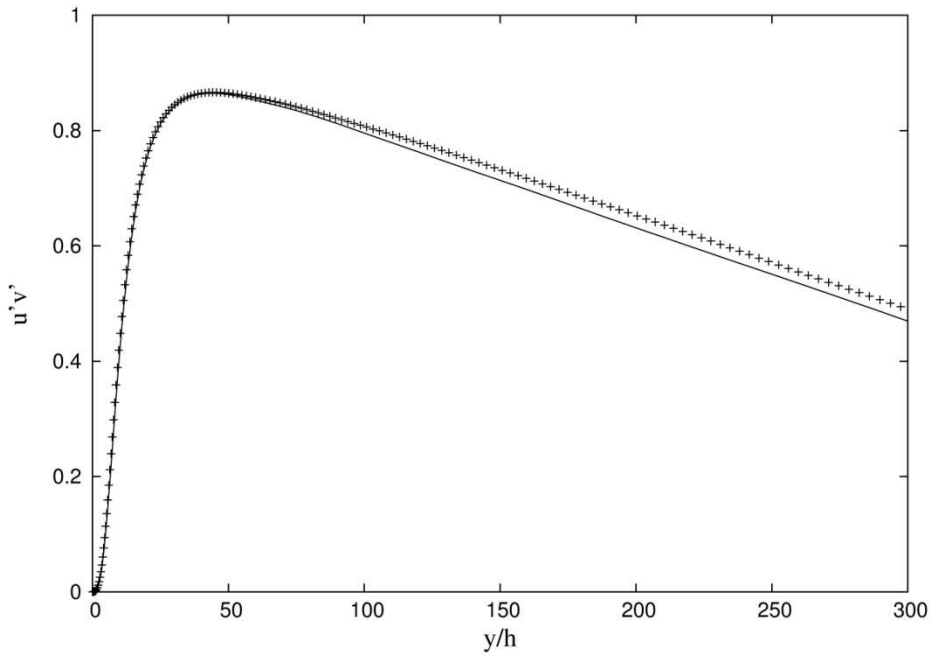

Figure 6.65 – Reynolds shear stress normalized by the friction velocity in wall coordinates. Present work: (+); data from [27] (—).

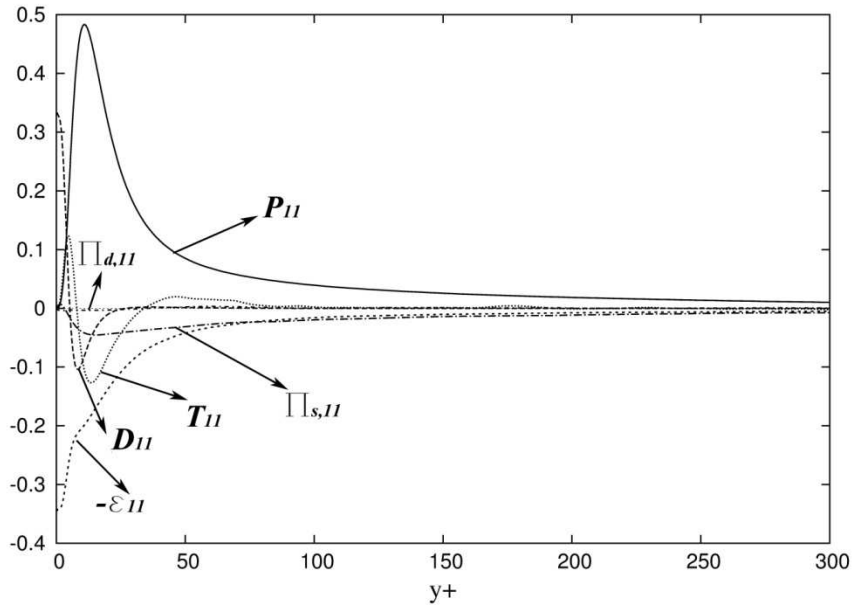

Figure 6.66 – Terms in the budget of $\overline{u_1'u_1'}$ in wall coordinates. P_{11} = Production; T_{11} = Turbulent transport; D_{11} = Viscous diffusion; ε_{11} = Dissipation rate; $\Pi_{s,11}$ = Velocity pressure strain gradient term; $\Pi_{d,11}$ = Velocity pressure diffusion gradient term.

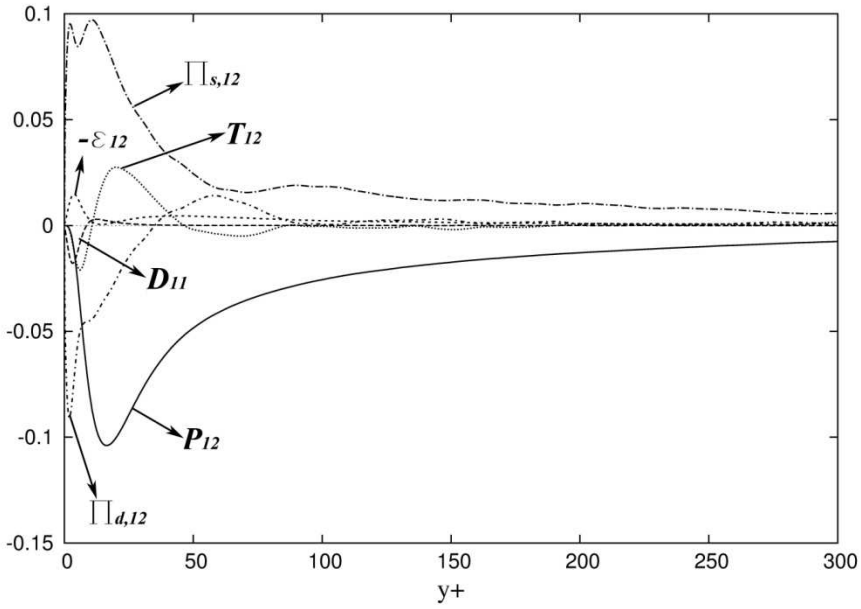

Figure 6.67 – Terms in the budget of $\overline{u_1'u_2'}$ in wall coordinates. P_{12} = Production; T_{12} = Turbulent transport; D_{12} = Viscous diffusion; ε_{12} = Dissipation rate; $\Pi_{s,12}$ = Velocity pressure strain gradient term; $\Pi_{d,12}$ = Velocity pressure diffusion gradient term.

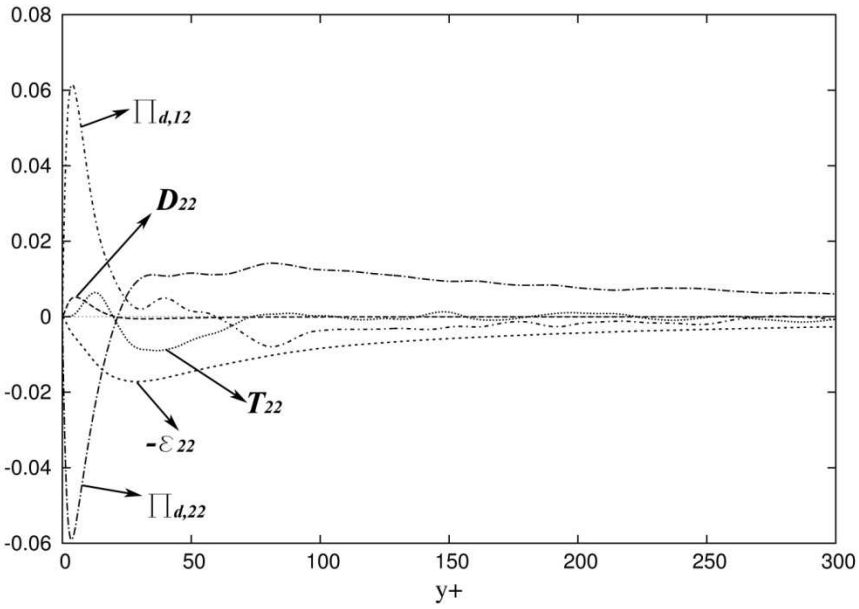

Figure 6.68 – Terms in the budget of $\overline{u_2' u_2'}$ in wall coordinates. T_{22} = Turbulent transport; D_{22} = Viscous diffusion; ε_{22} = Dissipation rate; $\Pi_{s,22}$ = Velocity pressure strain gradient term; $\Pi_{d,22}$ = Velocity pressure diffusion gradient term.

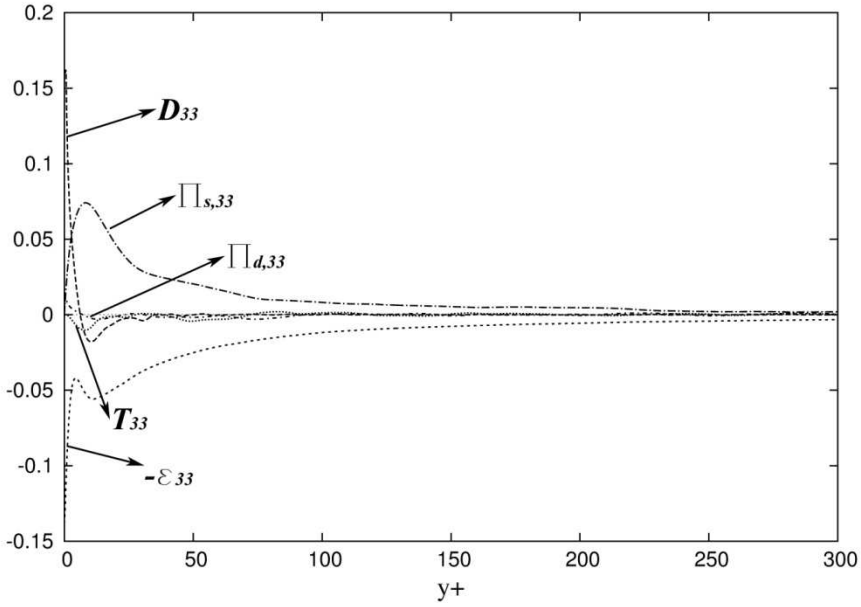

Figure 6.69 – Terms in the budget of $\overline{u_3' u_3'}$ in wall coordinates. T_{33} = Turbulent transport; D_{33} = Viscous diffusion; ε_{33} = Dissipation rate; $\Pi_{s,33}$ = Velocity pressure strain gradient term; $\Pi_{d,33}$ = Velocity pressure diffusion gradient term.

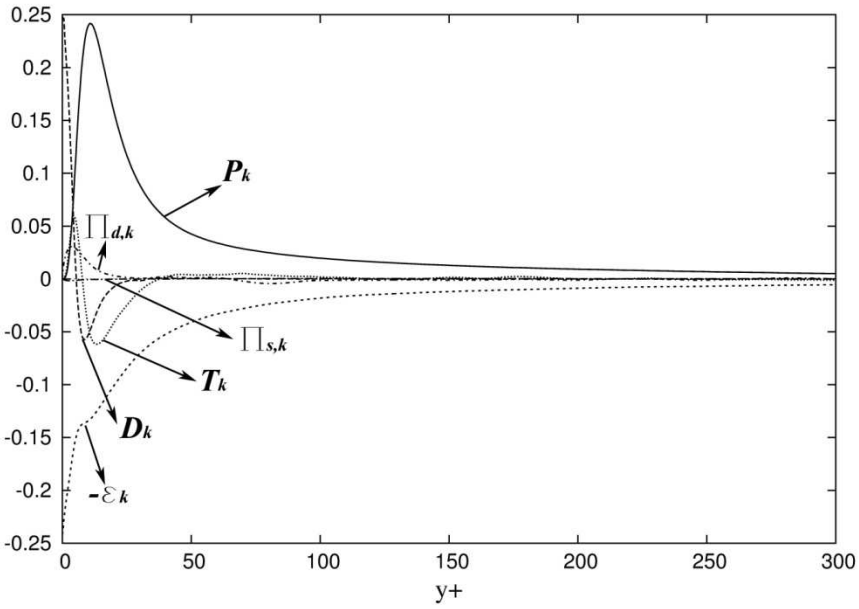

Figure 6.70 – Terms in the budget of the turbulent kinetic energy K in wall coordinates. P_K = Production; T_K = Turbulent transport; D_K = Viscous diffusion; ε_K = Dissipation rate; $\Pi_{s,K}$ = Velocity pressure strain gradient term; $\Pi_{d,K}$ = Velocity pressure diffusion gradient term.

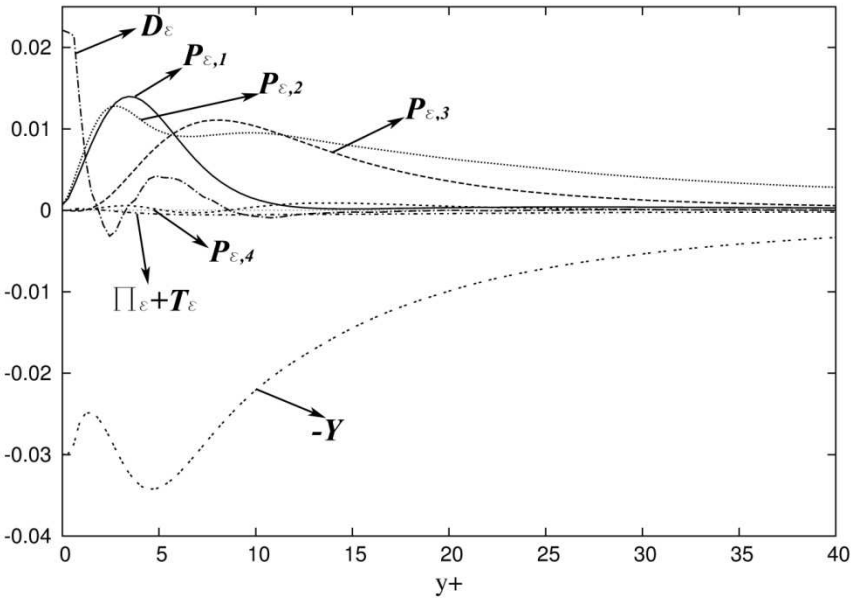

Figure 6.71 – Terms in the budget of the dissipation rate of the turbulence kinetic energy ε in wall coordinates. $P_{\varepsilon,1}$ = Production by mean velocity gradient; $P_{\varepsilon,2}$ = Mixed production; $P_{\varepsilon,3}$ = Gradient production; $P_{\varepsilon,4}$ = Turbulent production; T_ε = Turbulent transport; D_ε = Viscous diffusion; Y = Dissipation rate; Π_ε = Pressure transport.

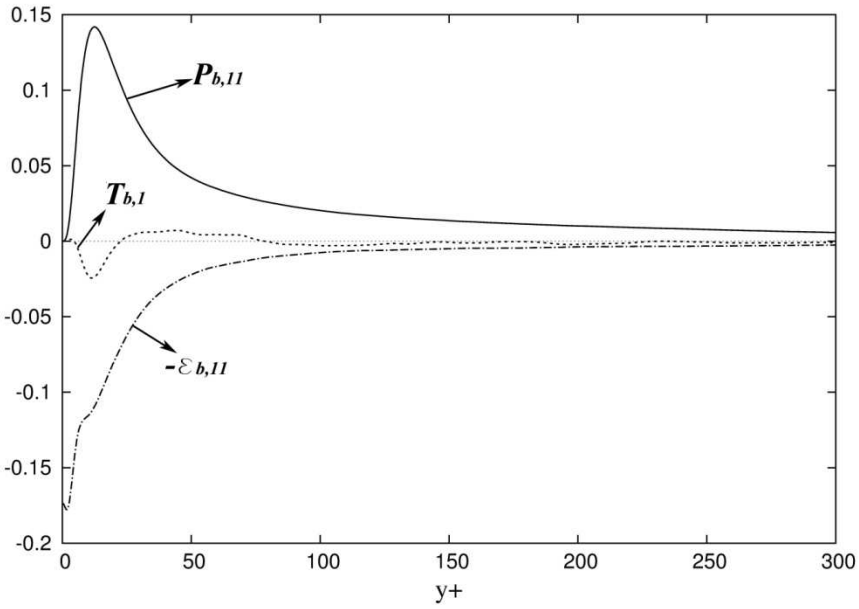

Figure 6.72 – Terms in the budget of the anisotropy rate of the turbulence kinetic energy in wall coordinates. $P_{b,11}$ = Production; $T_{b,11}$ = Turbulent transport; $\varepsilon_{b,11}$ = Dissipation rate.

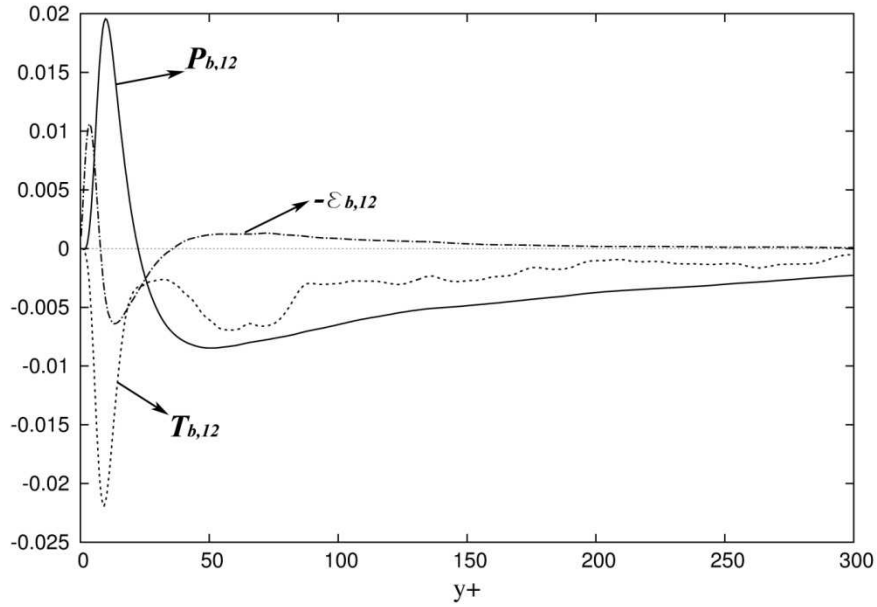

Figure 6.73 – Terms in the budget of b_{12} in wall coordinates. $P_{b,12}$ = Production; $T_{b,12}$ = Turbulent transport; $\varepsilon_{b,12}$ = Dissipation rate.

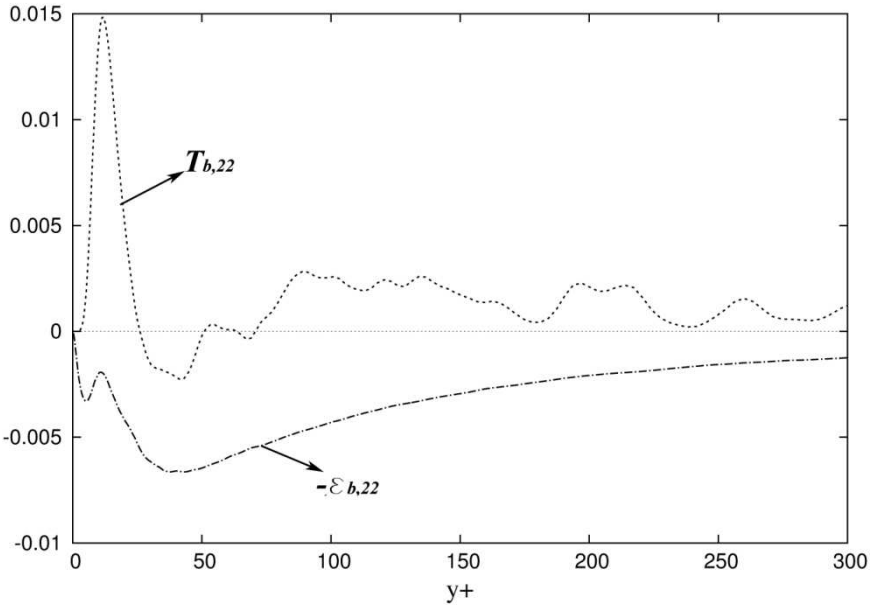

Figure 6.74 – Terms in the budget of the anisotropy rate of the turbulence kinetic energy in wall coordinates. $T_{b,22}$ = Turbulent transport; $\varepsilon_{b,22}$ = Dissipation rate.

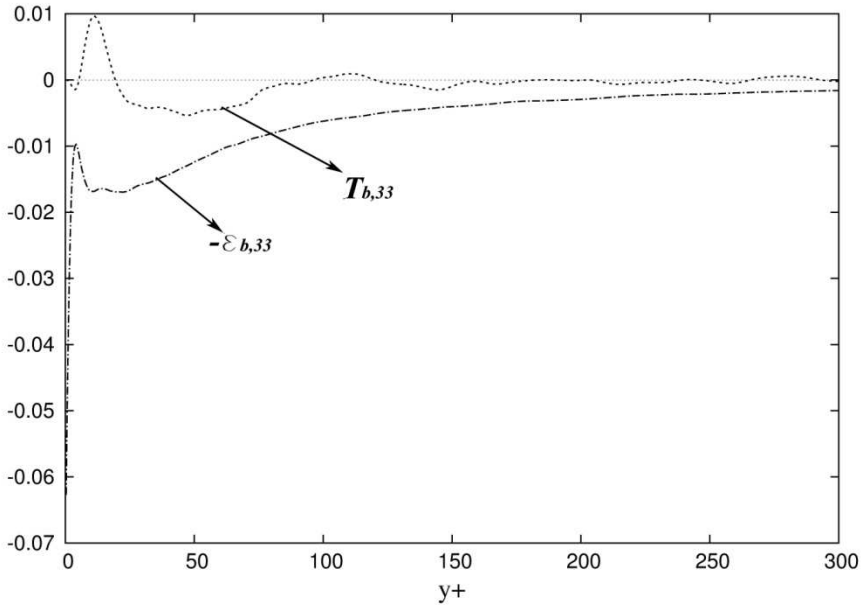

Figure 6.75 – Terms in the budget of the anisotropy rate of the turbulence kinetic energy in wall coordinates. $T_{b,33}$ = Turbulent transport; $\varepsilon_{b,33}$ = Dissipation rate.

6.4 Analysis of Turbulence Statistics with respect to the Reynolds number dependence

The essentially nature of the wall-bounded flows requires a deep analysis of the Reynolds number effects on turbulence quantities in the near-wall region. The availability of accurate DNS databases allows to examine, in particular, the Reynolds number dependence on turbulence statistics, referred to the inner region, in order to understand the behavior and the main characteristics of coherent structures. For this reason, after introducing statistical tools for the numerical scheme validation and showing the most important Reynolds shear stress budget terms, for each DNS database, this section is focused on the analysis of the same statistical variables with respect to the increasing Reynolds number, for a major inspection of the interaction between the inner and the outer layers. In particular, various turbulence statistics, such as the root-mean-square of fluctuations, the Reynolds shear stress, the skewness and flatness factors and the budget terms are evaluated in detail.

6.4.1 Turbulence intensities: discussion

The root mean square of the velocity fluctuations allows to verify the adequacy of the simulation sample taken for the average through the evaluation of the symmetrical trend of its values across the channel. Fig. 6.76 shows the distribution of the *rms* of the velocity fluctuations with the increase of the Reynolds number Re_τ considered. It is evident how the peak of the u'_{rms} progressively moves toward the wall with the increase of Re_τ: for high Reynolds numbers, the compression of vortices is strongest than that occurring for low ones, facilitating the development of ordered streamwise vortical structures.

The Reynolds shear stress distribution along the channel guarantees not only that the statistically steady state of the numerical simulations is reached, but also that the region where viscosity plays an important role is enlarged near walls as the Reynolds number is reduced. Figure 6.77, in particular, shows the shear stress and the total shear stress distributions at varying Reynolds number. Furthermore, the peak value of the Reynolds shear stress increases and moves away from the wall as the Reynolds number increases.

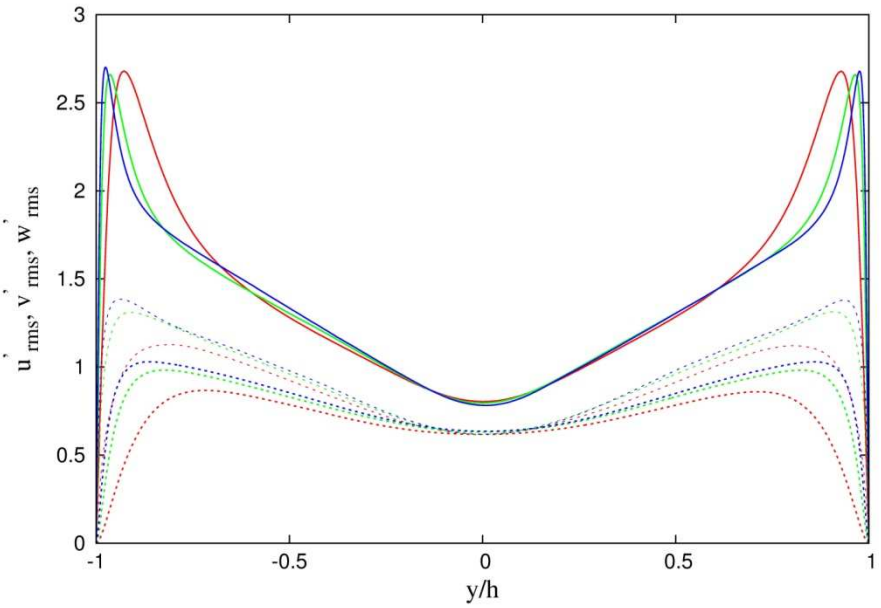

Figure 6.76 – Rms of velocity fluctuations in global coordinates. Present case study: (—) u'_{rms}, (---) v'_{rms}, (···) w'_{rms}; (red) DNS200, (green) DNS400, (blue) DNS600.

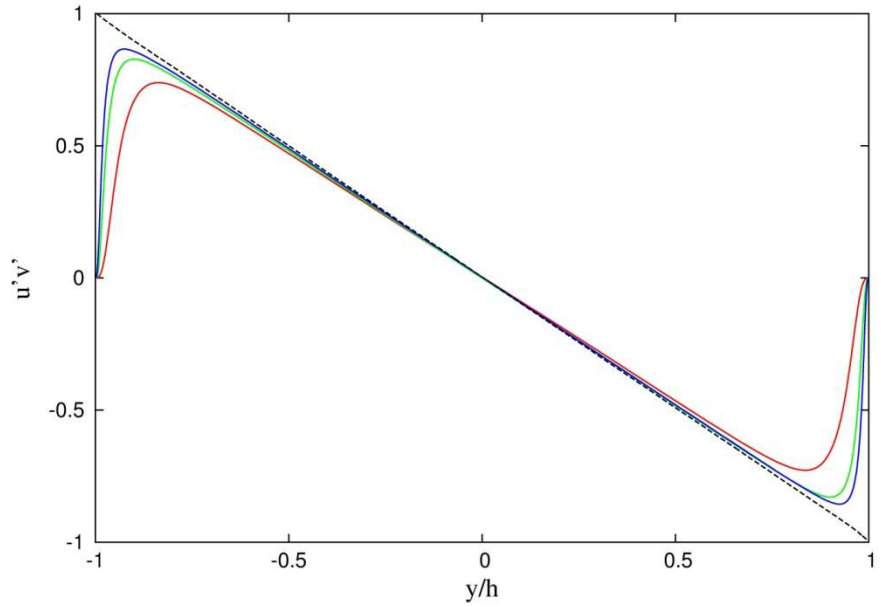

Figure 6.77 – Computed values of Reynolds shear stress and total shear stress normalized by the friction velocity in global coordinates. Present case study: (—) $\overline{u'v'}$, (---) τ_{tot}; (red) DNS200, (green) DNS400, (blue) DNS600.

Figure 6.78 reports the skewness factors in global coordinates, computed for each of the numerical databases available and indicates the asymmetry of the probability

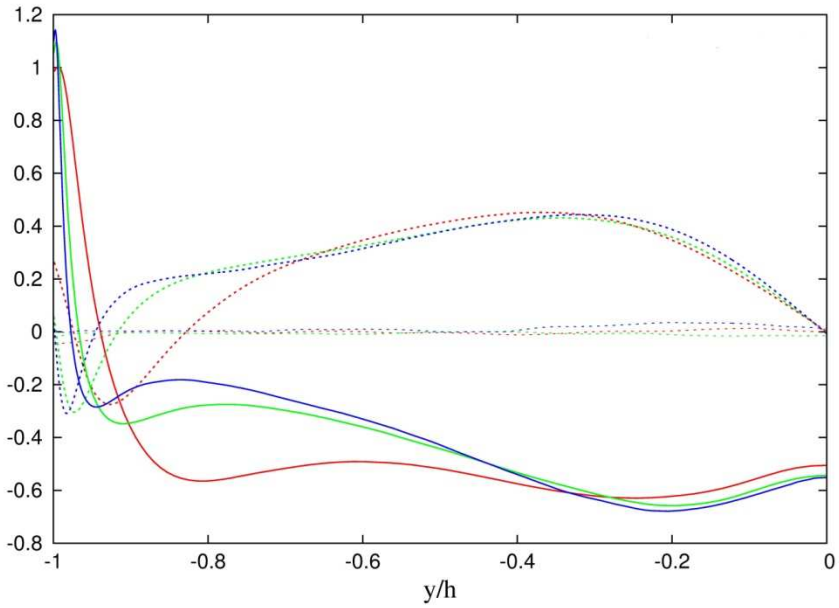

Figure 6.78 – Skewness factors of the velocity fluctuations in global coordinates. Present case study: (—) $S_{u'}$, (---) $S_{v'}$, (···) $S_{w'}$; (red) DNS200, (green) DNS400, (blue) DNS600.

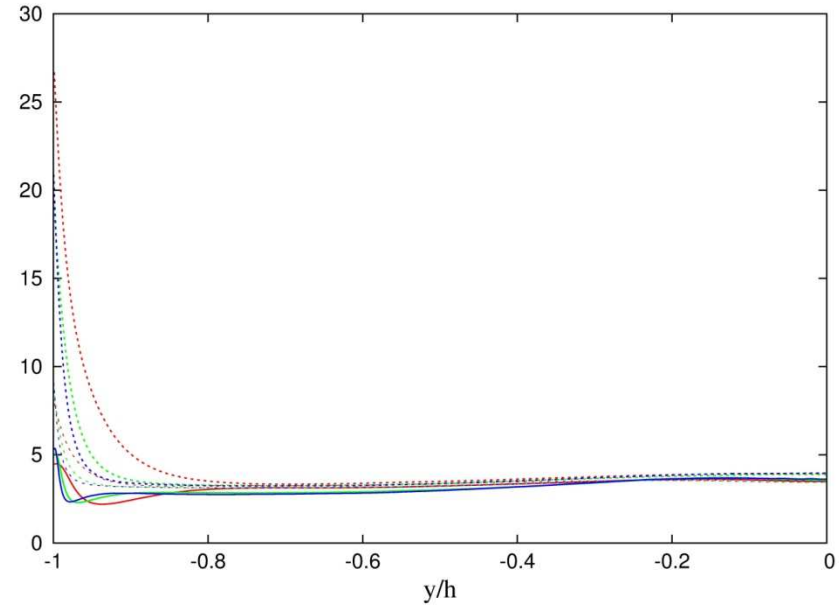

Figure 6.79 – Flatness factors of the velocity fluctuations in global coordinates. Present case study: (—) $F_{u'}$, (---) $F_{v'}$, (···) $F_{w'}$; (red) DNS200, (green) DNS400, (blue) DNS600.

density function of the variables considered. So, it is an indicator of the excursions of velocity fluctuations: it means that vorticity production increases as the Reynolds number increases. Figure 6.79 reports, instead, the flatness factors in global coordinates for the numerical databases available. It indicates the intermittent

Table 6.5 – Effect of the Reynolds number on the main turbulence statistics.

	DB200		***DB400***		***DB600***		***Percentage increase***		
	max	y^+	max	y^+	max	y^+	*% (DB200 vs DB400)*	*% (DB400 vs DB600)*	*% (DB200 vs DB600)*
$\overline{u'v'}$	0.739	33.238	0.828	40.170	0.866	33.238	12	5	17
u'_{rms}	2.680	16.509	2.660	16.599	2.701	16.444	-1	2	1
v'_{rms}	0.867	56.702	0.982	71.436	1.030	83.068	13	5	19
w'_{rms}	1.127	37.252	1.310	36.467	1.385	37.464	16	6	23
$S_{u'}$	1.003	1.315	1.096	1.423	1.141	1.504	9	4	14
$S_{v'}$	0.452	126.944	0.296	133.543	0.208	96.248	-35	-30	-54
$S_{w'}$	0.013	180.773	-0.003	36.467	0.005	86.188	-119	-292	-64
$F_{u'}$	6.488	0.761	6.070	1.131	6.366	0.897	13	6	20
$F_{v'}$	26.679	0.249	19.424	0.280	20.882	0.298	-27	8	-22
$F_{w'}$	7.902	0.249	8.752	0.280	9.094	0.298	11	4	15

character of velocity fluctuations: a higher flatness factor suggests that the relatively large excursion from the mean value are more probable. Table 6.5 shows the effect of the Reynolds number on the main turbulence statistics, such as the Reynolds shear stress, the root-mean-square of the velocity fluctuations, skewness and flatness factors.

6.4.2 Budget in the near-wall region: discussion

This section reports the budgets of the Reynolds stress, the turbulent kinetic energy, dissipation and anisotropy computed considering the DNS databases available in order to evaluate the role of the increase of the Reynolds number and its influence on budget distributions in terms of gain and/or loss energy.

Figure 6.80 shows the Reynolds stress budget for $\overline{u'_1 u'_1}$ component: in particular, the peak value of the production term P_{11} and the turbulent transport T_{11} increase as the Reynolds number increases. Those two terms, associated with the viscous diffusion term D_{11}, represent the gain energy contribution in the near wall region and are balanced by the velocity pressure-gradient term Π_{11} and the dissipation term ε_{11} that represent the loss energy contribution. The flow structures in the near wall

region, with respect to the $\overline{u_1'u_1'}$ component, will be characterized by intense production energy activity and regions of intense dissipation energy activity along the streamwise direction.

With reference to the buffer layer and the role of the $\overline{u_1'u_2'}$ component, instead, the production term P_{12} diminishes because of the Reynolds shear stress diminishes, too, while the dissipation term ε_{12} increases (Fig. 6.81).

The effect of the Reynolds number is strongest considering the $\overline{u_2'u_2'}$ and the $\overline{u_3'u_3'}$ components (Figs. 6.82 and 6.83, respectively); in particular, the pressure diffusion correlation contributes a lot in the budget of the $\overline{u_2'u_2'}$ component and its role is very important especially if it considers $\mathrm{Re}_\tau \geq 400$. Turbulent diffusion T_{22} is not so determinant in the overall budget, while dissipation term ε_{22} is the main responsible of the lost energy. Considering the $\overline{u_3'u_3'}$ component, it assists on the increasing of the viscous diffusion D_{33} and velocity pressure-gradient term Π_{33} as the Reynolds number increases, turbulence transport T_{33} do not play a role in the balance because its values are quasi-zero along the channel, while the dissipation term ε_{33} remains the main responsible of the lost energy in the spanwise direction.

The budget of the turbulence kinetic energy K (Fig. 6.84) has a similar trend as the budget of the Reynolds stress and, in conjunction with the budget of the dissipation rate, represents an important quantity for understanding theoretically turbulence phenomena. In particular, it is possible to note that all terms gradually increase with the increase of the Reynolds number Re_τ; in particular, in the viscous sublayer, the turbulence kinetic energy is gained by the viscous diffusion D_K but lost by dissipation ε_K, while in the logarithmic-law region turbulence kinetic energy is gained exclusively by production term P_K and lost by dissipation ε_K.

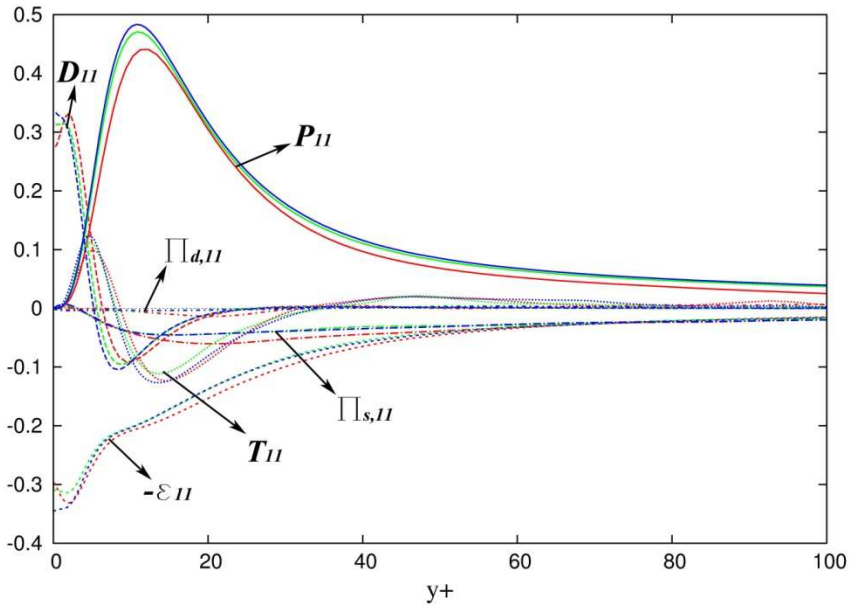

Figure 6.80 – Terms in the budget of $\overline{u_1'u_1'}$ in wall coordinates. P_{11} = Production; T_{11} = Turbulent transport; D_{11} = Viscous diffusion; ε_{11} = Dissipation rate; $\Pi_{s,11}$ = Velocity pressure strain gradient term; $\Pi_{d,11}$ = Velocity pressure diffusion gradient term; (red) DNS200, (green) DNS400, (blue) DNS600.

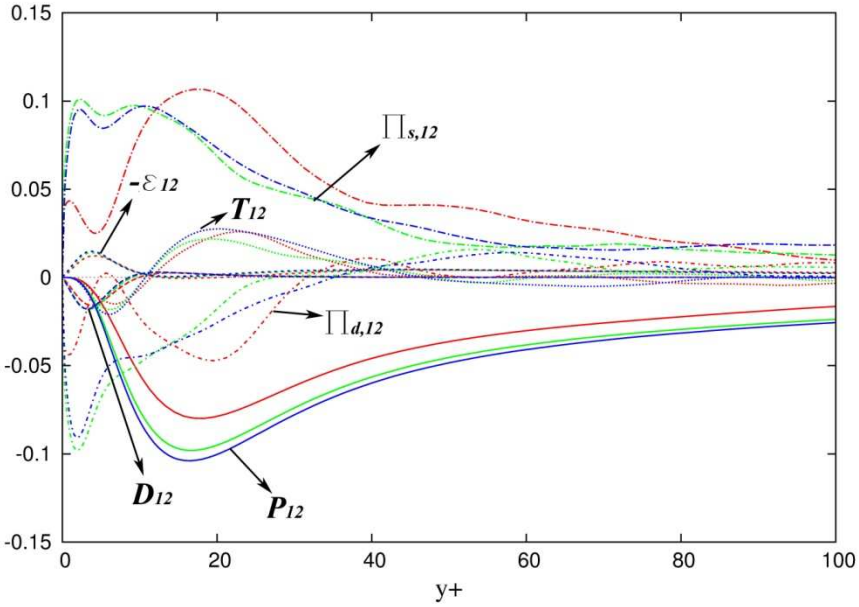

Figure 6.81 – Terms in the budget of $\overline{u_1'u_2'}$ in wall coordinates. P_{12} = Production; T_{12} = Turbulent transport; D_{12} = Viscous diffusion; ε_{12} = Dissipation rate; $\Pi_{s,12}$ = Velocity pressure strain gradient. term; $\Pi_{d,12}$ = Velocity pressure diffusion gradient term; (red) DNS200, (green) DNS400, (blue) DNS600.

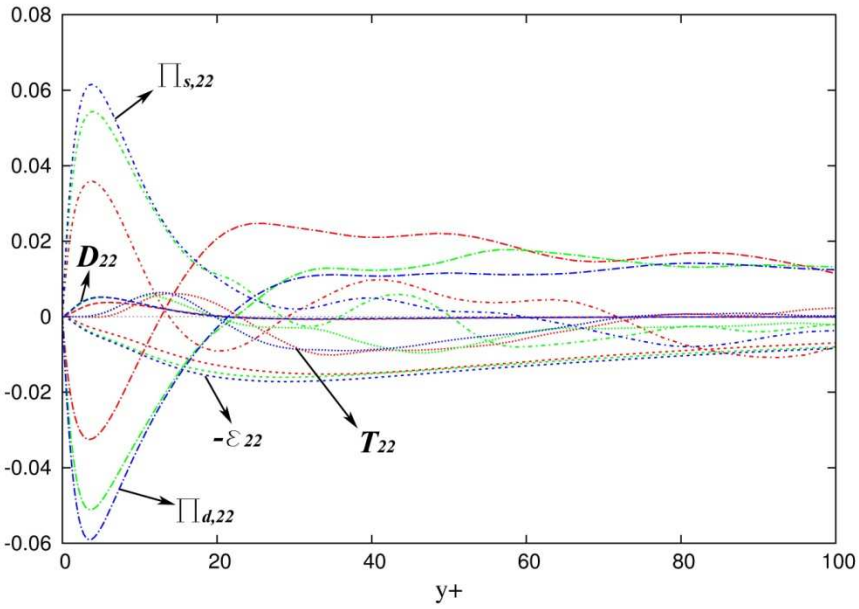

Figure 6.82 – Terms in the budget of $\overline{u_2'u_2'}$ in wall coordinates. T_{22} = Turbulent transport; D_{22} = Viscous diffusion; ε_{22} = Dissipation rate; $\Pi_{s,22}$ = Velocity pressure strain gradient term; $\Pi_{d,22}$ = Velocity pressure diffusion gradient term; (red) DNS200, (green) DNS400, (blue) DNS600.

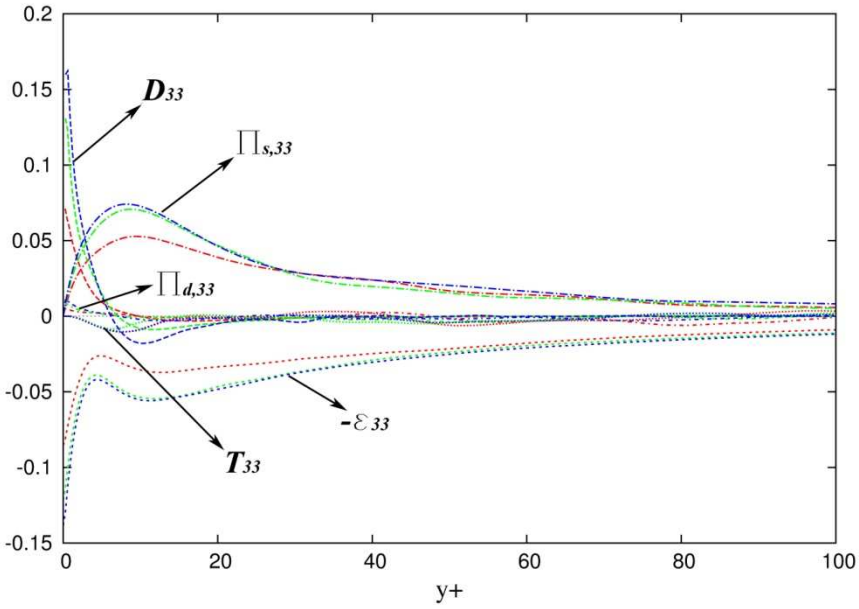

Figure 6.83 – Terms in the budget of $\overline{u_3'u_3'}$ in wall coordinates. T_{33} = Turbulent transport; D_{33} = Viscous diffusion; ε_{33} = Dissipation rate; $\Pi_{s,33}$ = Velocity pressure strain gradient term; $\Pi_{d,33}$ = Velocity pressure diffusion gradient term; (red) DNS200, (green) DNS400, (blue) DNS600.

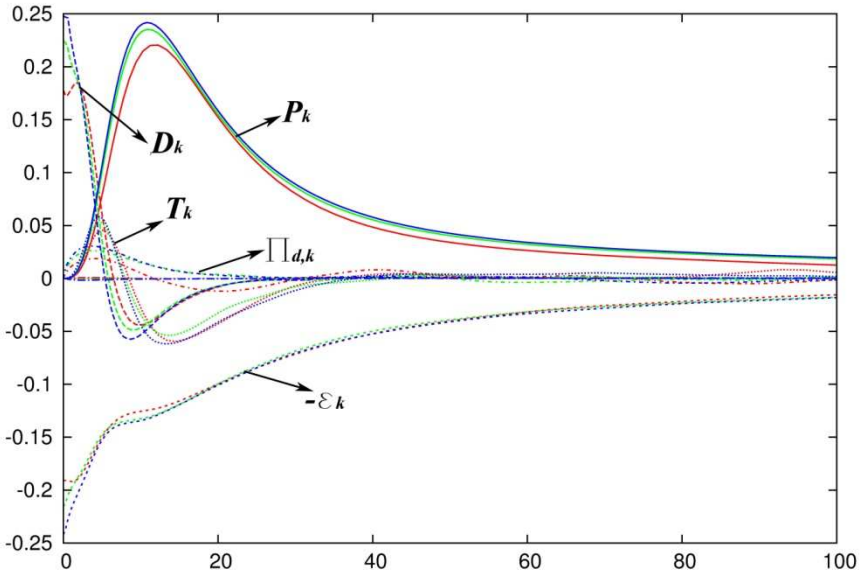

Figure 6.84 – Terms in the budget of the turbulent kinetic energy K in wall coordinates. P_K = Production; T_k = Turbulent transport; D_k = Viscous diffusion; ε_k = Dissipation rate; $\Pi_{s,k}$ = Velocity pressure strain gradient term; $\Pi_{d,k}$ = Velocity pressure diffusion gradient term; (red) DNS200, (green) DNS400, (blue) DNS600.

The dissipation process, instead, is determined by the scalar dissipation rate tensor ε and it is of great interest especially in turbulence modeling: the related values of all the terms that characterized Eq. (A.78) are reported on Fig. 6.85 at varying Reynolds numbers as a function of y^+. As observed in the graph referred to turbulence kinetic energy K also in this case the terms gradually increase with the increase of the Reynolds number. The most important terms that characterized the balance are the production term P_ε and the viscous dissipation term Y_ε, while very near the walls molecular dynamics are prevalent so the viscous diffusion D_ε and the viscous dissipation term Y_ε reaches a local minimum. Figure 6.85 shows, also, how the dissipation rate budget is important as the Reynolds number increase, revealing the strongly non-isotropic nature of the dissipation process at high Reynolds numbers.

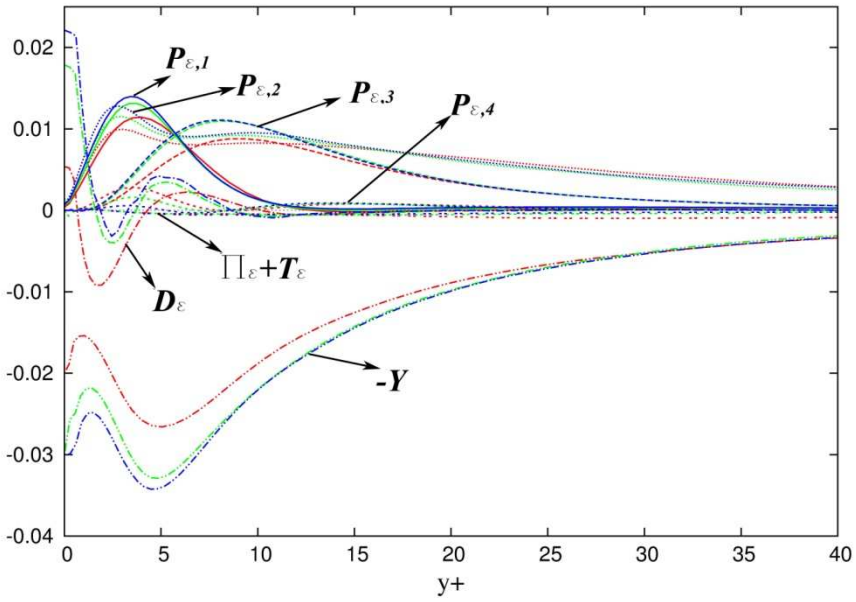

Figure 6.85 – Terms in the budget of the dissipation rate of the turbulence kinetic energy ε in wall coordinates. $P_{\varepsilon,1}$ = Production by mean velocity gradient; $P_{\varepsilon,2}$ = Mixed production; $P_{\varepsilon,3}$ = Gradient production; $P_{\varepsilon,4}$ = Turbulent production; T_ε = Turbulent transport; D_ε = Viscous diffusion; Y = Dissipation rate; Π_ε = Pressure transport; (red) DNS200, (green) DNS400, (blue) DNS600.

The last set of graphics here analyzed are referred to the anisotropy rate budget, as defined in Eq. (A.68), with respect to the Reynolds numbers considered. For the relevant non-zero stresses, the anisotropy rate budget described the highly anisotropic behavior of fluctuations in the near-wall region. About the ($\overline{u_1'u_1'}$) component, within the inner region, it assists on a substantial balance between the production terms $P_{b,11}$ and the dissipation term $\varepsilon_{b,11}$ (Fig. 6.86), while considering the $\overline{u_1'u_2'}$ component and with reference to the viscous sublayer, the most important term that ensures a gain energy is the dissipation term, which an increase in turbulence transport corresponds (Fig. 6.87). At high Reynolds number, the production term is predominant and is balanced by the turbulence transport term. The last two figures, Figs. 6.88 and 6.89, show the terms referred to $\overline{u_2'u_2'}$ and $\overline{u_3'u_3'}$ components. It is evident that the dissipation becomes isotropic in the central region, confirming that fluctuations are characterized by a strong anisotropy in the near wall region.

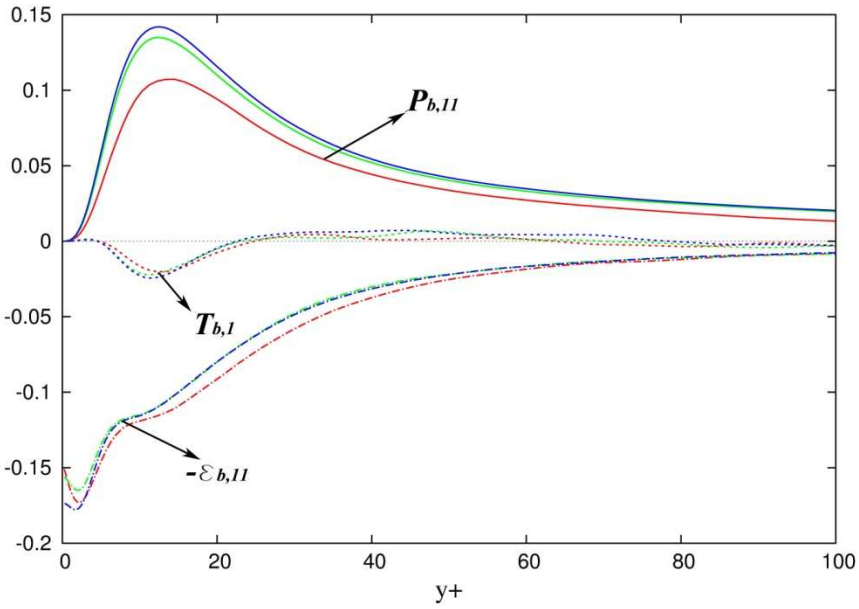

Figure 6.86 – Terms in the budget of b_{11} in wall coordinates. $P_{b,11}$ = Production; $T_{b,11}$ = Turbulent transport; $\varepsilon_{b,11}$ = Dissipation rate; (red) DNS200, (green) DNS400, (blue) DNS600.

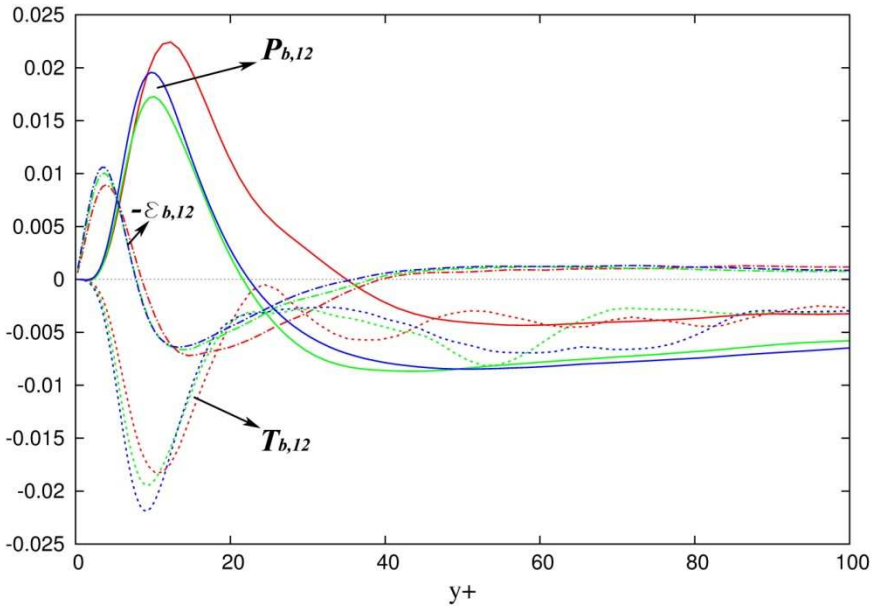

Figure 6.87 – Terms in the budget of b_{12} in wall coordinates. $P_{b,12}$ = Production; $T_{b,12}$ = Turbulent transport; $\varepsilon_{b,12}$ = Dissipation rate; (red) DNS200, (green) DNS400, (blue) DNS600.

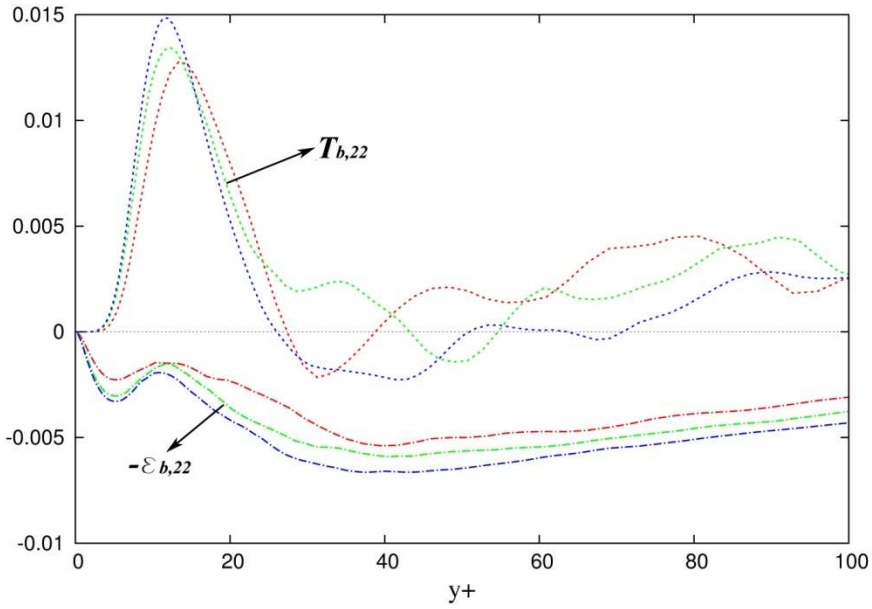

Figure 6.88 – Terms in the budget of b_{22} in wall coordinates. $T_{b,22}$ = Turbulent transport; $\varepsilon_{b,22}$ = Dissipation rate; (red) DNS200, (green) DNS400, (blue) DNS600.

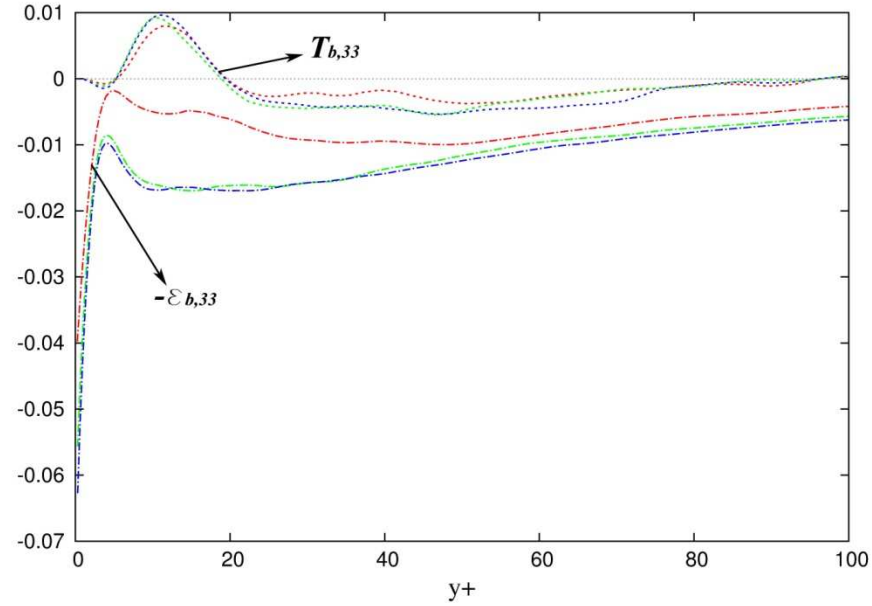

Figure 6.89 – Terms in the budget of b_{33} in wall coordinates. $T_{b,33}$ = Turbulent transport; $\varepsilon_{b,33}$ = Dissipation rate; (red) DNS200, (green) DNS400, (blue) DNS600.

6.5 Parallel performances of the numerical codes

After investigating the numerical accuracy of the simulations and the related main turbulence statistics, this section shows the computational results in terms of performance of the parallel implementations of the Navier-Stokes solver, described in *Chapter 5*.

The parallel performance has been monitored considering the three different spatial discretizations of the computational domain, one for each direct numerical simulation performed, at $\mathrm{Re}_\tau = 200, 400, 600$: the main characteristics are reported on Tab. 6.1.

In order to evaluate the performance of the Navier-Stokes solver, the CPU/GPU implementations, referred to single- and multi-GPU architectures, have been compared with the sequential code (1 CPU) and the OpenMP parallel codes (2, 4, 8, 12 CPUs).

The parallel performance has been evaluated by measuring the overall code execution time at each Runge-Kutta step. The measured computing times do not include the I/O operations. In Tab. 6.6, the run-times with the number of CPUs and of CPUs/GPUs used for each computational domain are reported.

For each of the computational domains considered, it can be noticed that *T* decreases with the number of processors: this result is in agreement with the Amdahl's law [127] about the level of efficiency that can be reached in a parallel computational process by using a multicore architecture. The strongest breakdown of the run-time is performed by using the heterogeneous architectures (CPU/GPU), whose values are two-orders of magnitude lower than the sequential implementation (almost three-orders of magnitude lower than the sequential one considering the greatest

Table 6.6 – Run-time *T* for one Runge-Kutta step with the number of processors

Processors	*Run-time T (s)*		
	DNS at $\mathrm{Re}_\tau = 200$ (DB200)	DNS at $\mathrm{Re}_\tau = 400$ (DB400)	DNS at $\mathrm{Re}_\tau = 600$ (DB600)
1 CPU	7.98	32.19	84.60
2 CPUs	4.23	16.80	43.26
4 CPUs	2.31	9.36	24.45
8 CPUs	1.44	5.37	15.06
12 CPUs	1.20	4.41	12.33
1 CPU + 1 GPU	0.37	1.71	-
3 CPUs + 3 GPUs	-	-	3.32

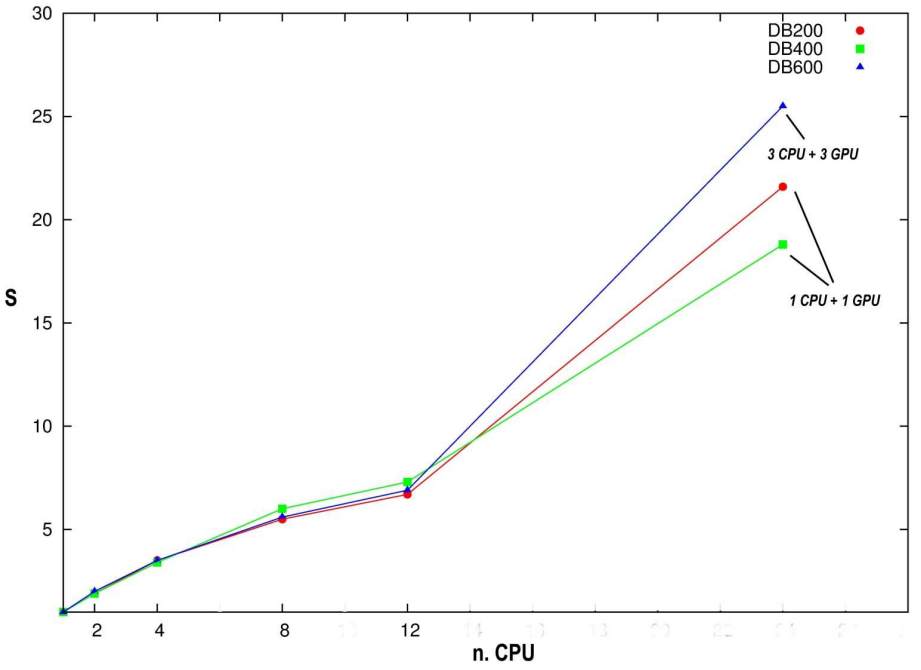

Figure 6.90 – Speedup S with the number of processors for each numerical database: (red) DNS at $\mathrm{Re}_\tau = 200$; (green) DNS at $\mathrm{Re}_\tau = 400$; (blue) DNS at $\mathrm{Re}_\tau = 600$

computational domain, used to perform the simulation at $\mathrm{Re}_\tau = 600$). Figure 6.90 reports the speedup of the calculations with the several architectures for each computational domain: as in Eq. (4.1), the speedup *S* is defined as the run-time per time step with one processor divided by the run-time per time step with a given number of processors. As concerns the parallel implementation on a CPU/GPU computing system, the speedup is defined as the ratio between the total execution time on a CPU and that one on a GPU. The graph of the speedup provides an immediate perception of the performance of the computational code with respect to the linear theoretical value: considering also the performance of the parallel implementation on a CPU/GPU system, it is evident how the CUDA Navier-Stokes solver outperforms significantly the different parallel implementations based on multicore architectures.

Table 6.7 reports the related values of the computed speedups.

Table 6.7 – Parallel performance: speedup *S*.

Processors	*Speedup S (-)*		
	DNS at $Re_\tau = 200$ (DB200)	DNS at $Re_\tau = 400$ (DB400)	DNS at $Re_\tau = 600$ (DB600)
1 CPU	1.0	1.0	1.0
2 CPUs	1.9	1.9	1.9
4 CPUs	3.5	3.4	3.5
8 CPUs	5.5	6.0	5.6
12 CPUs	6.7	7.3	6.9
1 CPU + 1 GPU	21.6	18.8	-
3 CPUs + 3 GPUs	-	-	25.5

Finally, in Tab. 6.8, the efficiency E of the parallel implementations is reported: as defined in Eq. 4.3, the efficiency represents the ratio between the theoretical parallel run-time and the actually measured run-time with a given number of processors. The efficiency of the CPU/GPU parallel implementations is not so significant because of the nature of the computing architecture, based on a massively parallel processor array.

As the efficiency values indicate, the OpenMP parallel implementation of the Navier-Stokes solver ensure good performances as the size of the computational domain increases with the increasing of the number of processors. However, an appreciable degradation of efficiency is observed when running 12 threads considering the computational domain used for simulating turbulence at $Re_\tau = 600$, due to memory access overhead.

Table 6.8 – Parallel performance: efficiency *E*.

Processors	*Efficiency E (-)*		
	DNS at $Re_\tau = 200$ (DB200)	DNS at $Re_\tau = 400$ (DB400)	DNS at $Re_\tau = 600$ (DB600)
1 CPU	1.0	1.0	1.0
2 CPUs	0.94	0.96	0.98
4 CPUs	0.86	0.86	0.87
8 CPUs	0.69	0.75	0.70
12 CPUs	0.55	0.61	0.57

Chapter 7

Results of the Simulations: Flow Structures

7.1 Introduction

A significant advance in wall bounded turbulent flows is achieved by considering studies about coherent structures in a turbulent boundary layer, thanks to the availability of high-quality numerical simulations.

The concept of coherency and evolution of coherent structures offer the possibility to clarify the physical mechanisms through which turbulent energy of mechanical nature is dissipated into heat. Thus, the description of energy transformations brings new perspectives in the modern fluid technology, such as:

- the control of turbulence, that has a relevant engineering impact on the reduction of skin friction in wall bounded flows, the delay of separation in wake flows, the enhancement of mixing in free shear flows and controlled sediment transport in multiphase flows;

- the development of new predictive models for the numerical calculation of high Reynolds number turbulent flows, useful in the evaluation of closures in numerical modeling.

Considering the mean-flow properties of wall-bounded flows in wall units and as concerns the mean-velocity profile, different layers can be distinguished:

- viscous sublayer, $0 < y^+ < 5$, where $u^+ = y^+$;
- the buffer layer, $5 < y^+ < 50$ (recently estimated up to 200 by [128]), the region of maximum average production of turbulent kinetic energy;
- overlap layer, $y^+ > 50$, characterized by the logarithmic law

$$u^+ = \frac{1}{\kappa} \ln y^+ + C \tag{7.1}$$

where κ and C are empirical constants. Some authors ([129], [130], [131], [132]) claim that the pipe-flow data of [133] are satisfactorily interpreted in the overlap layer by a power law in which the relation between u^+ and y^+ is Reynolds number dependent:

$$u^+ = \left(\frac{1}{\sqrt{3}} \ln \mathrm{Re} + \frac{5}{2} \right) \left(y^+ \right)^{3/2(\ln \mathrm{Re})} \tag{7.2}$$

where Re is the Reynolds number based on the mean velocity averaged over the cross section. More recent results confirm on the contrary the validity of the logarithmic law, with the constant parameters estimated as $\kappa = 0.38$ and $C = 4.1$ [128];

- far outer layer, where the law-of-the-wake is valid.

At sufficiently high Reynolds number, two regions can be distinguished: the inner region, the near-wall region in which viscous effects are present and includes the viscous sublayer, the buffer layer and the overlap layer in part; the outer region, essentially inviscid, that includes the rest of the layers. The two region are overlapped and the extent of the overlap increases as the Reynolds number increases.

In the present chapter, the flow structures of a turbulent flow in a plane channel are investigated, in terms of morphological evolution in time and strength of a vortex population, in conjunction with ejection and sweep events, by considering the DNS results obtained at $\mathrm{Re}_\tau = 200, 400, 600$: Section 7.2 contains an overview of some widely-used vortex-identification methods; in Section 7.3, the event-detection techniques are described; finally, in Section 7.4, the results of the numerical simulations performed are presented and discussed.

7.2 Vortical structures eduction methods

7.2.1 The D criterion

Perry and Chong [63] proposed the method of identifying vortices by means of isosurfaces of positive small values of the discriminant of the characteristic equation of the velocity-gradient tensor (deformation-rate tensor), where it has complex eigenvalues.

By considering the system of the Navier-Stokes equations, an arbitrary point can be chosen in the flow field and a Taylor series expansion of each velocity component can be performed in terms of space coordinates with the origin in that point:

$$u_i = A_i + A_{ij} x_j + A_{ijk} x_j x_k + ... \tag{7.3}$$

where the first-order pointwise linear approximation is:

$$u_i = A_i + A_{ij} x_j \tag{7.4}$$

($A_{ij} = \partial u_i / \partial x_i$ is the velocity-gradient tensor). If the origin is located at a critical point, the zero-order terms A_i are equal to zero. Thus, from the characteristic equation of A_{ij} one has:

$$\det\left(\mathbf{A} - \lambda \mathbf{I}\right) = 0 \tag{7.5}$$

$$\lambda^3 + P\lambda^2 + Q\lambda + R = 0 \tag{7.6}$$

where:

$$P = -tr\left(A_{ij}\right) \tag{7.7}$$

$$Q = \frac{1}{2}\left\{\left[tr\left(A_{ij}\right)\right]^2 - tr\left(A_{ij}^2\right)\right\} \tag{7.8}$$

$$R = -\det\left(A_{ij}\right) \tag{7.9}$$

(*tr* is trace, *det* is determinant) are the scalar invariants of the velocity-gradient tensor. In the case of incompressible flow, $P = 0$ (from continuity) and Eq. (7.6) becomes:

$$\lambda^3 + Q\lambda + R = 0 \tag{7.10}$$

The characteristic Eq. (7.10) admits three roots, so that the eigenvalues λ that determine the topology of the local flow pattern are formed on the basis of the values of the remaining non-zero invariants (Q and R).
The discriminant of Eq. (7.10) is:

$$D = \left(\frac{R}{2}\right)^2 + \left(\frac{Q}{3}\right)^3 = \frac{27}{4}R^2 + Q^3 \tag{7.11}$$

and two cases are possible:

- $D > 0$, so that Eq.(7.11) admits two complex and one real solution;
- $D \leq 0$, so that Eq.(7.11) admits three – all real – solutions.

Thus, complex eigenvalues of the velocity gradient tensor occur when $D > 0$.

7.2.2 The Q criterion

Hunt et al. [64] devised another criterion, defining an eddy zone a region characterized by positive values of the second invariant of the velocity-gradient tensor ($Q > 0$). The rate-of-deformation tensor can be split into symmetric and anti-symmetric parts:

$$A_{ij} = S_{ij} + W_{ij} \tag{7.12}$$

S_{ij} being the rate-of-strain tensor (corresponding to the pure irrotational motion) and W_{ij} the rate-of-rotation tensor (corresponding to the pure rotational motion), so that the second invariant of A_{ij} can be written as:

$$Q = \frac{1}{2}\left(W_{ij}W_{ij} - S_{ij}S_{ij}\right) \tag{7.13}$$

where the first term of the right-hand-side of Eq. (7.13) is proportional to the enstrophy density and the second term is proportional to the rate of dissipation of kinetic energy. If Q is large and positive, the rate-of-rotation dominate the strain rate, while if Q is large and negative, the vorticity is low and the rate-of-strain (proportional to the rate-of-dissipation) is large.

7.2.3 The λ_2 criterion

Jeong and Hussain [65] proposed a definition of a vortex by considering the problem of the pressure minimum. The gradient ∇ of the Navier-Stokes equation is taken and the result is decomposed into a symmetric and an anti-symmetric part.
By considering the symmetric part (the anti-symmetric portion is the vorticity-transport equation), one has:

$$\frac{DS_{ij}}{Dt} - \nu\frac{\partial S_{ij}}{\partial x_k \partial x_k} + B_{ij} = -\frac{1}{\rho}\frac{\partial^2 p}{\partial x_i \partial x_j} \tag{7.14}$$

where:

$$B_{ij} = S_{ik}S_{kj} + W_{ik}W_{kj} \tag{7.16}$$

The existence of a local pressure minimum requires two positive eigenvalues of the Hessian tensor of the pressure $\partial^2 p / \partial x_i \partial x_j$.

By neglecting the contribution of the first two terms on the left-hand side of Eq. (7.14), only tensor in Eq. (7.15) is considered to determine the existence of a local pressure minimum due to vortical motion, i.e. the presence of two negative eigenvalues of B_{ij}. Tensor B_{ij} is symmetric by construction, all its eigenvalues are real and can be ordered $\lambda_1 \geq \lambda_2 \geq \lambda_3$. According to this method, a vortex is defined as a connected region of the flow with two negative eigenvalues of B_{ij}.The tensor B_{ij} is symmetric by construction, all its eigenvalues are real and can be ordered $\lambda_1 \geq \lambda_2 \geq \lambda_3$. A vortex is then defined as a connected region of the flow with the requirement that the intermediate eigenvalue of B_{ij}, $\lambda_2 < 0$.

7.2.4 The λ_{ci} criterion

Zhou et al. [66] adopted the criterion of identifying vortices by visualizing isosurfaces of values of the imaginary part of the complex eigenvalue pair of the velocity-gradient tensor. This method is also known as the swirling strength criterion. When $D > 0$, the velocity-gradient tensor has one real eigenvalue λ_1 and a pair of complex-conjugate eigenvalues λ_2, λ_3, that can be written as:

$$\lambda_1 = \lambda_r \tag{7.16}$$

$$\lambda_2 = \lambda_{cr} + i\lambda_{ci} \tag{7.17}$$

$$\lambda_3 = \lambda_{cr} - i\lambda_{ci} \tag{7.18}$$

The swirling strength, given by λ_{ci}, is a measure of the local swirling rate inside the vortex, while the strength of stretching or compression is given by λ_r.

Isosurfaces of the imaginary part of the complex eigenvalue pair of the velocity-gradient tensor can be used to visualize vortices. The method is frame independent and due to the fact that the eigenvalue is complex only in regions of local circular or spiraling streamlines, it automatically eliminates regions having vorticity but no local spiraling motion. With reference to Eq. (7.10) and defining the quantities:

$$J = \left(-\frac{R}{2} + \sqrt{\frac{R^2}{4} + \frac{Q^3}{27}} \right)^{\frac{1}{3}} \tag{7.19}$$

$$K = -\left(+\frac{R}{2} + \sqrt{\frac{R^2}{4} + \frac{Q^3}{27}} \right)^{\frac{1}{3}} \tag{7.20}$$

one has:

$$\lambda_1 = \lambda_r = J + K \tag{7.21}$$

$$\lambda_2 = \lambda_{cr} + i\lambda_{ci} = -\frac{J+K}{2} + \frac{J-K}{2}\sqrt{-3} \tag{7.22}$$

$$\lambda_3 = \lambda_{cr} - i\lambda_{ci} = -\frac{J+K}{2} - \frac{J-K}{2}\sqrt{-3} \tag{7.23}$$

7.2.5 Enhanced λ_{ci} criterion

Chakraborty et al. [71] proposed an enhanced criterion, called the enhanced swirling strength criterion, as follows. In the regions where the eigenvalues of Eq. (7.10) are complex, two parameters are identified:

- the imaginary part of the complex eigenvalue pair (λ_{ci}, the swirling rate);

- the ratio of the real to the imaginary part of the complex eigenvalue pair ($\lambda_{cr}/\lambda_{ci}$, the inverse spiraling compactness) that measures the orbital compactness of the fluid particles involved in the vortical motion.

A vortical region is identified as the points in the flow field that satisfy the dual requirement:

$$\lambda_{ci} \geq (\lambda_{ci})_{th} = \varepsilon^* \tag{7.24}$$

$$\frac{\lambda_{cr}}{\lambda_{ci}} \leq \left(\frac{\lambda_{cr}}{\lambda_{ci}}\right)_{th} = \delta^* \tag{7.25}$$

(*th* is threshold). However, inside an intense vortical structure, the swirling motion dominates and $|\lambda_{cr}/\lambda_{ci}|$ takes negligible values. In [71], the relationships between the previous vortex-eduction methods and the two parameters λ_{ci} and $\lambda_{cr}/\lambda_{ci}$ are determined.

7.2.6 On λ_{ci} threshold value

The issue of determining an appropriate threshold value $(\lambda_{ci})_{th}$ (T^{-1}, where T is time) has been addressed by Alfonsi and Primavera [134].

An adequate threshold value of λ_{ci} is needed, keeping in mind that small threshold values will represent a large number of weak vortical structures – so that the flow-field representation tends to be volume filling – while large threshold values will represent only the most intense vortical cores. Attention is given onto the legs of a generic hairpin vortex and the concept is raised that the value of the circulation around a closed circuit delimiting the external borders of the hairpin legs will discriminate the vortical structure from the rest of the flow field.

The characteristic inverse time scale is introduced:

$$(\lambda_{ci})_{th} = \frac{1}{2}\left(\frac{\oint_{l1} t_i u_i ds}{A_{l1}} + \frac{\oint_{l2} t_i u_i ds}{A_{l2}}\right) \tag{7.26}$$

(*l*1 is the first leg, *l*2 is the second leg, t_i is unit tangent, u_i is velocity, A_{l1} is the cross-sectional area of the first leg, A_{l2} is the cross-sectional area of the second leg), i.e. the average (over the two legs) of the circulation around the closed circuit that delimits each of the hairpin's legs, divided by the area of the legs themselves.

In Eq. (7.26), the area of the hairpin's legs is not easy to evaluate, mainly because it depends on the threshold value of λ_{ci} that is chosen for the representation of the hairpin itself. Moreover, for the determination of the line integral around the closed circuit that delimits each of the hairpin's legs, it would be necessary to know the distribution of the velocity around these circuits. This distribution is also difficult to be determined, mainly due to the highly non uniform character of the velocity field. In fact, slow-moving fluid is lifted away from the wall on the updraft side of the hairpin legs, while fast-moving fluid is moved toward the wall on the downdraft side.

The application of the Stokes theorem to Eq. (7.26) reduces the calculation of the characteristic time scale $(\lambda_{ci})_{th}$ to the evaluation of a prescribed value Ω of the streamwise component of the vorticity ω_x in the legs of the hairpin.

The following calculation is performed. Once a hairpin vortex is qualitatively identified in the flow field, the minimum volume containing the vortical structure is strictly considered, on the basis of a first-sight perception of the vortex itself.

Then, Ω is calculated as:

$$\Omega(z) = \left[\frac{\sum_x \sum_y \omega_x^2(x,y,z)}{N_x^{mv} N_y^{mv}} \right]^{\frac{1}{2}} \tag{7.27}$$

obtaining the *rms* ω_x, averaged along the streamwise $x-$ and vertical $y-$ directions, inside the minimum volume (N_x^{mv} and N_y^{mv} are the grid points along the streamwise and vertical directions in the minimum volume, respectively).

Then Ω is investigated along the spanwise $z-$direction. The distribution of Ω along $z-$direction in the minimum volume exhibits two well-defined peaks, besides some other smaller spikes.

The two peaks reflect the distribution of Ω inside the hairpin legs, i.e. the values that have to be taken into account for the calculation of the characteristic time scale. The final result is obtained:

$$\left(\lambda_{ci}\right)_{th} = \frac{\Omega_{l1} + \Omega_{l2}}{2} \tag{7.28}$$

by further averaging Ω along $z-$direction inside the hairpin legs.
The characteristic inverse time scale (7.27) is then calculated, in the non-dimensional form $\left[\left(\lambda_{ci}\right)_{th} h/\mathrm{Re}_{\tau} u_{\tau}\right]$, by evaluating the field of Ω in the legs of the hairpin. This evaluation univocally depends on the flow field at hand.

7.3 Event-detection techniques

7.3.1 Conditional sampling and averaging

Conditional sampling and averaging is a group of techniques for quantitatively distinguishing particular regions of a flow, including coherent structures (Antonia [50]). A conditional average can be seen as a special type of generalized cross-correlation:

$$R\left(x_i, \Delta x_i, \tau_j\right) = \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} c\left(x_i, t_k\right) f\left(x_i, \Delta x_i, t_k + \tau_j\right) \tag{7.29}$$

where $c\left(x_i, t_k\right)$ is the conditioning function at a point x_i in space and at a time t_k, f is a digital function, N is a number of points to be averaged and (τ_j) is the time delay.

7.3.2 Quadrant analysis

A useful tool for unambiguous definition of the turbulent events occurring in the boundary layer is the quadrant analysis, introduced by Willmarth and Lu [51] (see also [57], [135], [136], [137], [138], [139], [140], [141], [142]). Considering the

quadrant analysis, the local flow behavior is divided into quadrants, depending on the sign of the streamwise and normal velocity fluctuations u' and v', identified as follows:

- Q_1 - first quadrant, where $u' > 0$ and $v' > 0$, denoting an event in which high-speed fluid moves toward the center of the flow field;
- Q_2 - second quadrant, where $u' < 0$ and $v' > 0$, denoting an event in which low-speed fluid moves toward the center of the flow field, away from the wall (*ejection*);
- Q_3 - third quadrant, where $u' < 0$ and $v' < 0$, denoting an event in which low-speed fluid moves toward the wall;
- Q_4 - fourth quadrant, where $u' > 0$ and $v' < 0$, denoting an event in which high-speed fluid moves toward the wall (*sweep*).

The most relevant events are those of the second and the fourth quadrants. Ejections are frequent at a distance from the wall, while sweeps are frequent near the wall. The ejection and sweep events – the events most related with the production of Reynolds stress – are the manifestation of the dynamic processes of evolution of turbulent structures in the boundary layer.

7.3.3 Linear stochastic estimation

The LSE (Linear Stochastic Estimation) technique is a procedure that allows to obtain the best linear approximation to a conditionally averaged flow field $\left\langle u_i'(x_i') \,|\, u_i(x_i) \right\rangle$ where $u_i(x_i)$ is a velocity event specified at point x_i, upon which the flow is conditioned. The best linear estimate of the fluctuating flow field $\widehat{u_i'}(x_i')$ is formulated in terms of an event vector $u_j(x_i)$ as follows:

$$\widehat{u_i'}(x_i') = \sum_{j=1}^{3} L_{ij}(x_i', x_i)\, u_j \tag{7.30}$$

where (L_{ij}) are the linear estimation coefficients to be determined such that the mean square error between the linear estimate $\widehat{u_i'}(x_i')$ and the conditional average $\langle u_i'(x_i') | u_i(x_i) \rangle$ is minimized. The estimated flow field depends on the event vector.
The LSE technique is used by [143] and [144] for the estimation of conditional eddies in the case of homogeneous shear flow, by [145] in the case of the turbulent channel flow and by [66] for the eduction of an initial turbulent structure associated to a Q_2 event to be followed numerically in its subsequent temporal evolution.

7.3.4 VITA Analysis

Another tool is the VITA Analysis (Variable-Interval Time-Averaging), introduced by Blackwelder and Kaplan [146]. In performing the VITA analysis, in a time series of pointwise velocity data one wants to detect the instants in which the highest velocity fluctuations occur. The notation of local average is introduced, an averaging operation over a time interval of the order of the time scale of the phenomenon under study. The method basically consists in the identification of the instants in which the variance of the velocity data in a significant time interval is greater than the variance of the entire series. A localized measure of the turbulent energy is obtained by applying the VITA technique to the square of the velocity and subtracting the localized squared mean value. For this scope, a localized variance is formulated, defined as:

$$\mathrm{var}(x_i,t,T) = \langle u^2(x_i,t,T) \rangle - \langle u(x_i,t,T) \rangle^2 \tag{7.31}$$

Where T is the averaging time. The detection criterion is completed by using a threshold level on the VITA variance signal.
In an analogous way, also the spatial counterpart of VITA can be defined, the VISA analysis (Variable-Interval Space-Averaging, see [53]).

7.4 Results

The present study shows the existence and the frequent occurrence of hairpin packets in wall bounded flows by using high accurate simulations at high Reynolds numbers. The results of the DNS, performed at $\mathrm{Re}_\tau = 200, 400, 600$ in terms of flow structures, are presented. The scientific visualization of vortical structures is done by using Paraview, an open-source, multi-platform data analysis and visualization application (http://www.paraview.org/).

Working with DNS data of turbulent channel flow, [66] adopt the criterion of visualizing isosurfaces (of the square) of the imaginary part of the complex eigenvalue pair of the velocity gradient tensor, that represents the local swirling strength of the vortex. The method is frame independent and, due to the fact that the eigenvalue is complex only in regions of local circular or spiraling streamline, it automatically eliminates regions having vorticity but no local spiraling motion, such as shear layers. The evolution of a single hairpin vortex-like structure in turbulent channel flow is considered. The initial vortical structure is obtained from the two-point spatial correlation of the velocity field by linear stochastic estimation, given a second-quadrant event vector. Initial vortices having vorticity that is weak with respect to the mean value gradually evolve into Ω-shaped vortices that persist for a relatively long time and decay slowly.

Initial vortices that exceed a threshold strength with respect to the mean flow generate new hairpin vortices upstream from the primary vortex. The mechanism of the upstream-process generation is similar to that proposed by [56], with some differences in the details. It is also found that new hairpins generate downstream of the primary hairpin forming, together with the upstream hairpins, a hairpin's packet that propagate coherently. The low-Reynolds number DNS results of [66] are integrated by the PIV measurements of [147] at relatively higher Reynolds numbers, giving rise to a conceptual model founded on the hairpin packet paradigm (the term hairpin is used here to indicate cane, hairpin, horseshoe, omega-shaped vortices, being these structures as variations of a common basic flow structure at different stages of evolution).

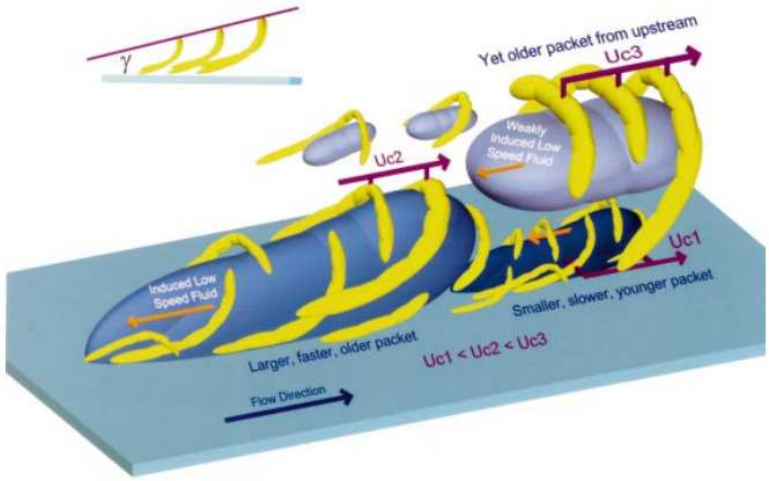

Fig. 7.1 – Scheme of nested packets of hairpin vortices growing up from the wall (adapted from [147]).

In this model (Fig. 7.1) packets of vortices originate at the wall from a disturbance. Firstly, the primary hairpin is formed. It is stretched and intensified by the difference between the streamwise velocity at its legs and head and grows continuously in time evolving into an omega-shaped vortex. If its strength is high enough, it generates a new hairpin, inducing a strong Q_2 event that interacts with the high-speed fluid behind the primary hairpin.

As time progresses, the secondary hairpin grows and begins to create a tertiary hairpin. The resulting packet of hairpins in not symmetric and, with time, the pattern of the vortices in the packet become rather complex. In the buffer layer low-speed streaks form between the hairpin's legs. On the average, larger packets propagate downstream more rapidly with respect to the smaller and the overall dynamics of hairpins does not appear to be governed by just inner or outer variables.

After the application of the swirling-strength criterion, the quadrant analysis of the flow field is considered, in order to visualize also the events that characterized the motion of hairpin-vortical structures and evaluate their coherency, stability and persistency.

More details about coherent structures and theory of vortical structures may be found in [134] and references therein.

7.4.1 Vortical structures in the wall region at $\mathbf{Re_\tau = 200}$

The DNS database related to the plane channel at $\mathrm{Re}_\tau = 200$, whose numerical results are described in *Chapter 6*, is employed in this study.

Thus, after the application of the λ_{ci} vortex detection method to the fluctuating portion of the computed velocity field, a flow field appears, filling the computing domain of turbulent structures adjacent to both upper and lower walls. An overall view of vortical structures at $t^+ = 1$ is shown in Fig. 7.2: hairpins distribution among the two solid walls is shown in Fig. 7.3, while views of hairpin structures over each wall are shown in Figs. 7.4 and 7.5, at the upper and the lower wall, respectively. In these figures, the non-dimensional value of the swirling-strength parameter that characterizes the external surfaces of the vortical structures is $\lambda_{ci} = 9.25 \cdot 10^{-4}$.

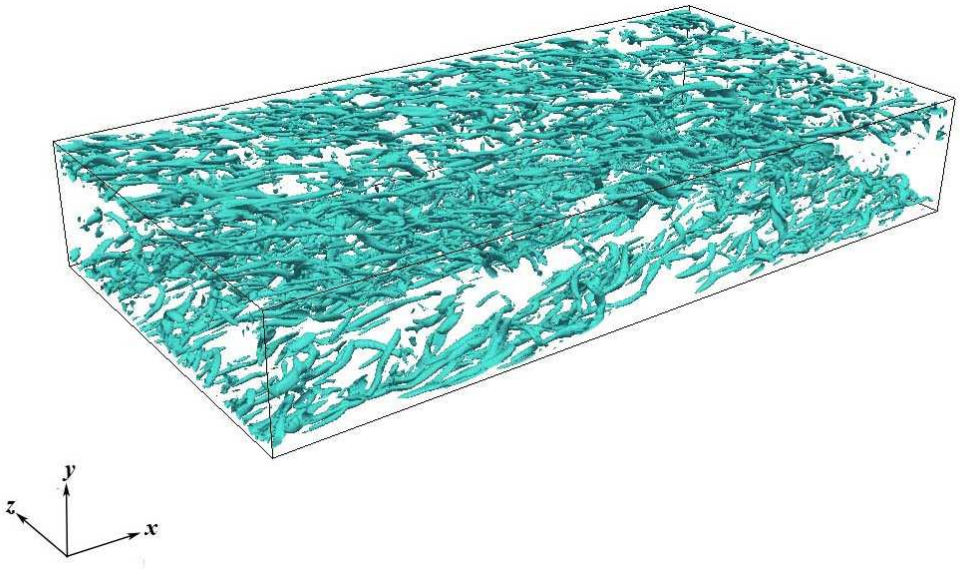

Figure 7.2 – Vortical structures in the computing domain at $\mathrm{Re}_\tau = 200$.

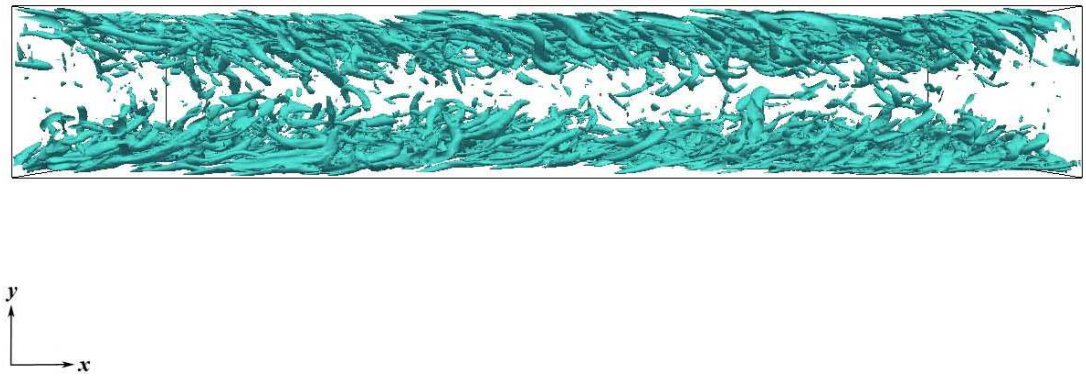

Figure 7.3 – Vortical structures in the computing domain at $\mathrm{Re}_\tau = 200$: lateral view.

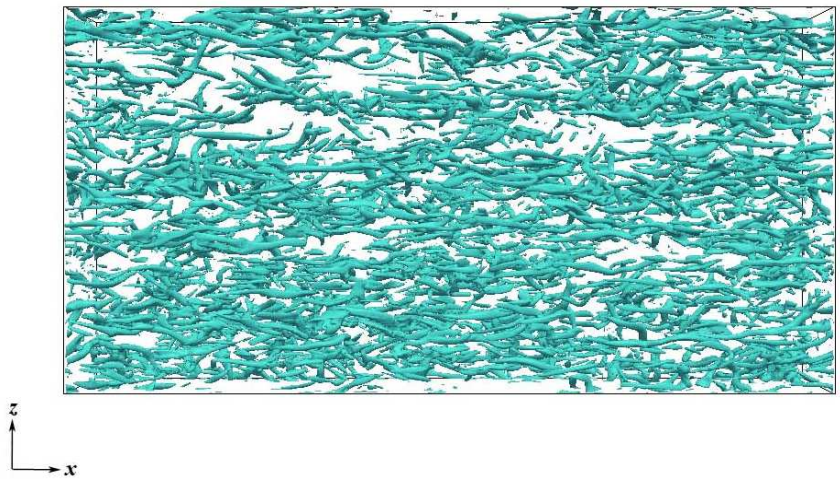

Figure 7.4 – Vortical structures in the computing domain at $\mathrm{Re}_\tau = 200$: inferior wall.

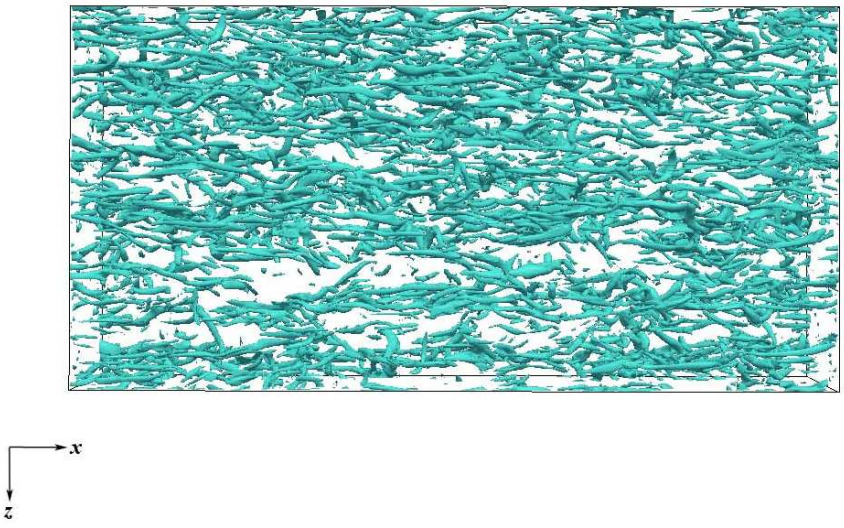

Figure 7.5 – Vortical structures in the computing domain at $\mathrm{Re}_\tau = 200$: superior wall.

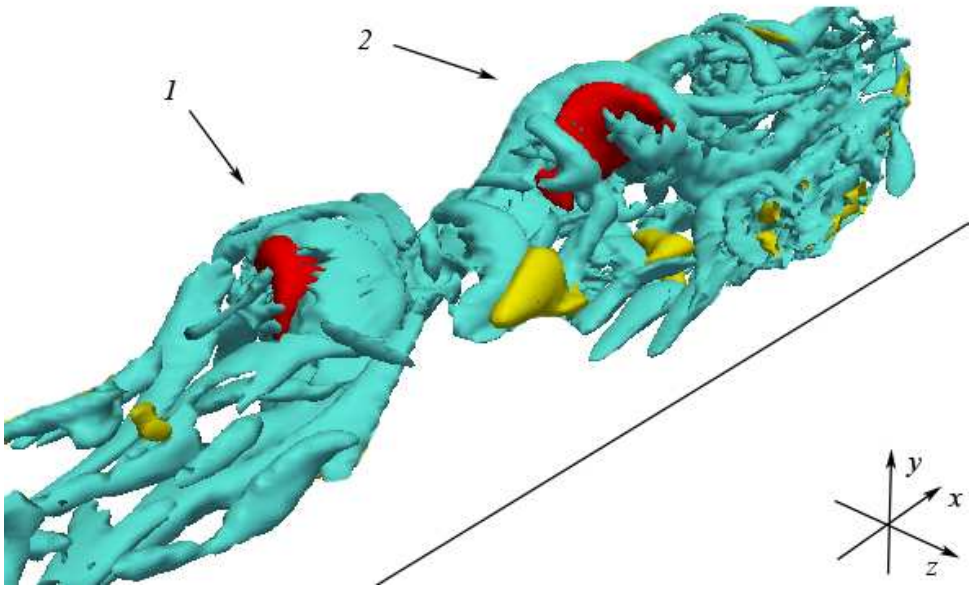

Figure 7.6 – Representation of hairpin vortices and quadrant events at $t^+ = 23$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin in cyan, ejection in red and sweep in yellow.

Considering the DNS database at $\mathrm{Re}_\tau = 200$, it is possible to capture and study the temporal evolution of an isolated turbulent vortex, in order to explain many of the features observed in wall turbulence models by [54] and [55]. Furthermore, the interaction between turbulent events, detected by applying the quadrant analysis technique, and vortical structures is investigated, too. In the following, the analysis of some interesting sequences extracted from the DNS database is described. Cyan color is used to represent vortical structures, red color is used to represent the Q_2 quadrant event (ejections), while yellow color is used to represent the Q_4 quadrant event (sweeps).

The first sequence analyzed regards the evaluation of the relationship between 2nd and 4th quadrant events and dynamics of vortical structures (Figs. 7.6-7.11) considering the time evolution of two hairpin-vortices at the lower wall.

Two main persistent hairpin vortices are visible and denoted , in Fig. 7.6 as *(1)* and *(2)*, at $t^+ = 23$: both of them are characterized by a quite elevation of their heads, while the legs are hidden by the presence of other structures nearby solid walls. In particular, the main primary hairpin is visible and completely developed, while the

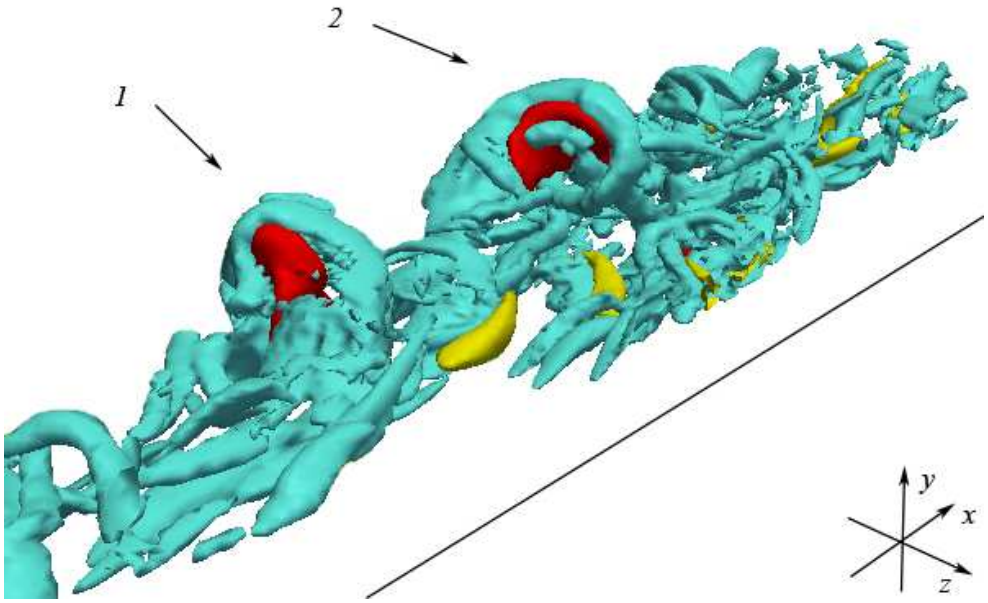

Figure 7.7 – Representation of hairpin vortices and quadrant events at $t^+ = 24$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin in cyan, ejection in red and sweep in yellow.

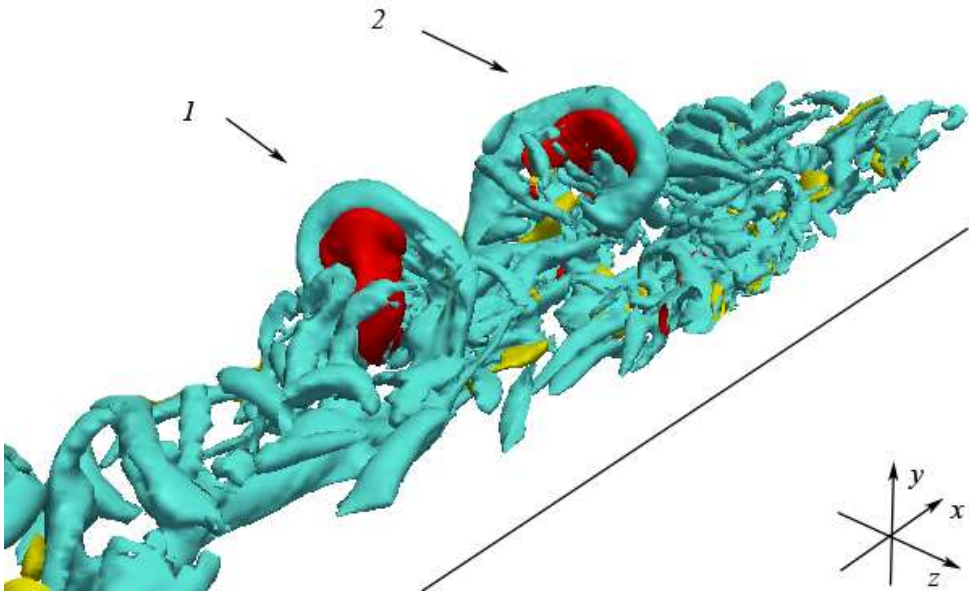

Figure 7.8 – Representation of hairpin vortices and quadrant events at $t^+ = 25$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin in cyan, ejection in red and sweep in yellow.

second one is characterized by the complete development of one leg on the left-side. Below the head of each one, the internal space of the structure is occupied by the ejection isosurfaces, showing that Q_2 event is the main mechanism through which the head of a hairpin is raised upwards (and backward, $u' < 0, v' > 0$). Two sweep

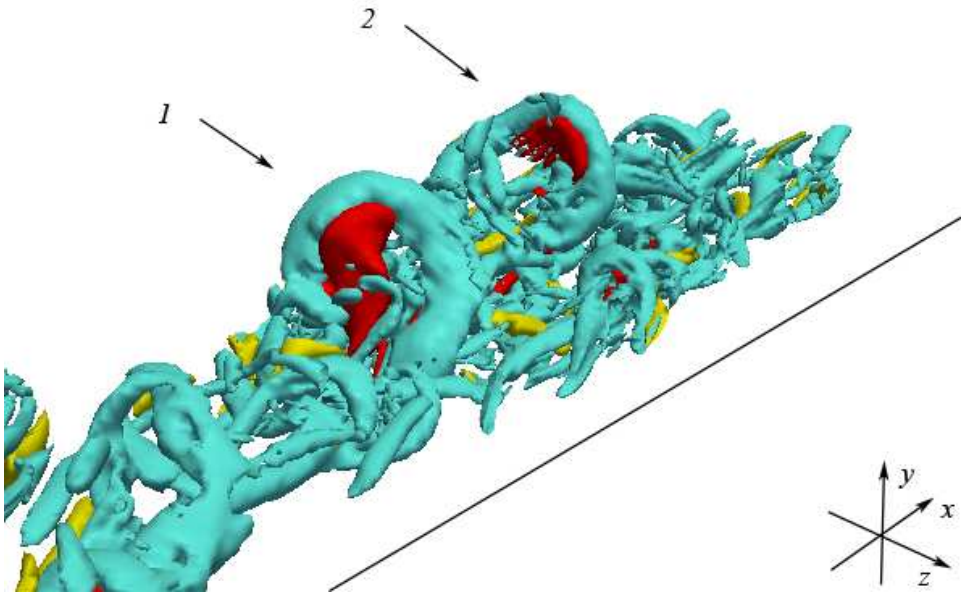

Figure 7.9 – Representation of hairpin vortices and quadrant events at $t^+ = 26$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin in cyan, ejection in red and sweep in yellow.

isosurfaces start to appear at both side of the hairpins, but they do not play a relevant role in this phase.

Figure 7.7, at $t^+ = 24$, shows how both the two hairpins continue to grow due exclusively for the pushing up action of ejections, while sweeps isosurfaces are totally irrelevant.

At $t^+ = 25$ (Fig. 7.8), the primary hairpin is mature and well-defined, assuming the typical $\Omega-$shaped, while the effect of push-up of ejection isosurface on the second hairpin starts to become more evident with the formation of the leg on the right-side.

The primary hairpin continues its development, becoming a persistent and symmetric vortical structure in the flow field at $t^+ = 26$ (Fig. 7.9), while the second one continues its development under a residual influence of ejection, until its disruption (Figs. 7.10 and 7.11). Observing the sequence here described, it is evident the action of the ejections in the first phase of hairpins formation, that allows the growing of their heads upwards and defining the persistent character of the $\Omega-$shaped vortices in the flow field with no presence of sweep events.

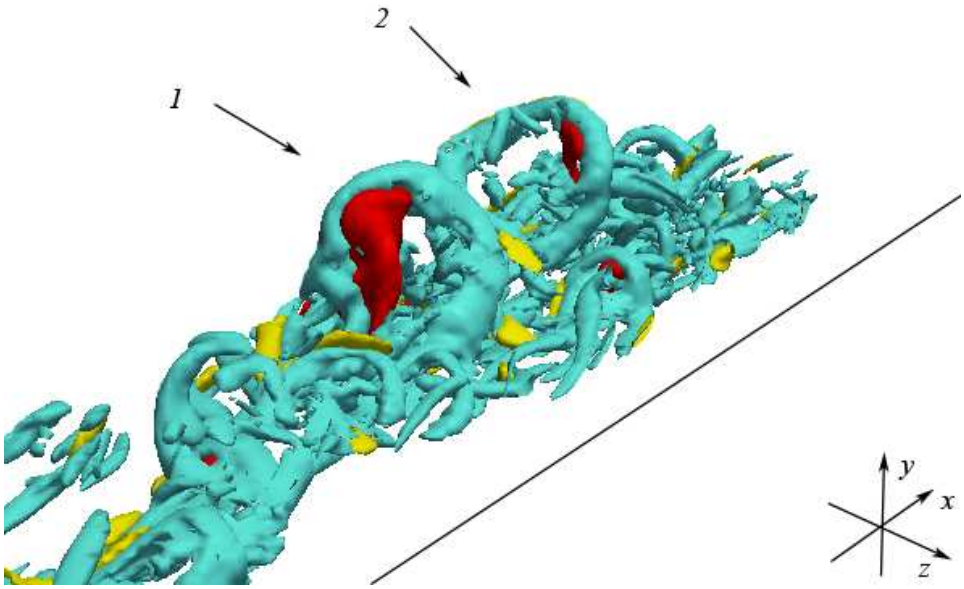

Figure 7.10 – Representation of hairpin vortices and quadrant events at $t^+ = 27$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin in cyan, ejection in red and sweep in yellow.

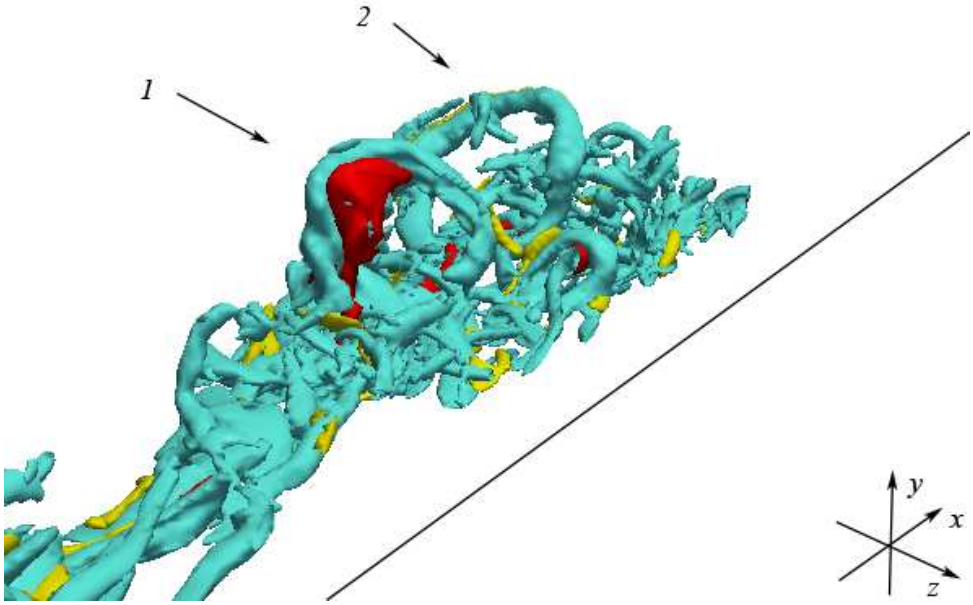

Figure 7.11 – Representation of hairpin vortices and quadrant events at $t^+ = 28$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin in cyan, ejection in red and sweep in yellow.

Figures 7.12-7.17 show another sequence of hairpin-vortical structures, characterized by a couple of hairpins. As shown in Fig. 7.12, the first hairpin is visible, while the second one is going to rise. Also in this case, the ejection isosurface fills the internal space of both the hairpins, while a consistent sweep isosurfaces starts to appear along the right-side of the legs, surrounding also the neck of the primary hairpin.

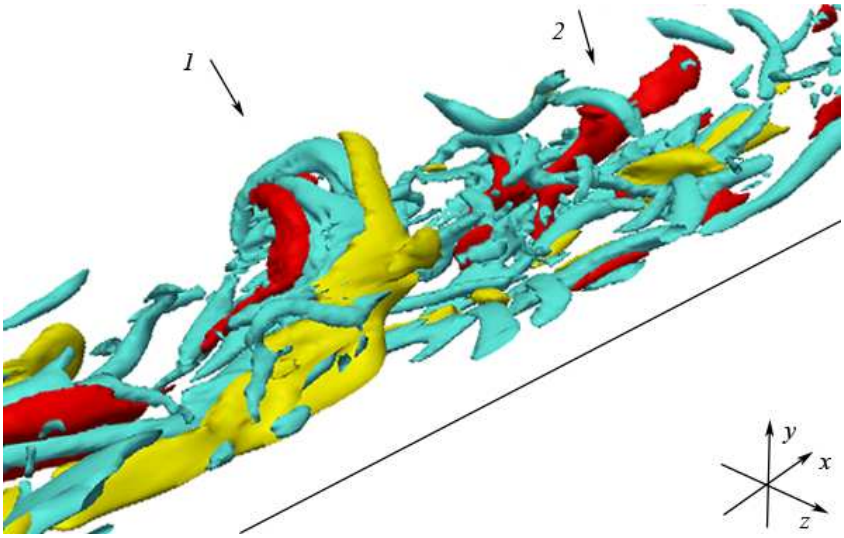

Figure 7.12 – Representation of hairpin vortices and quadrant events at $t^+ = 220$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin in cyan, ejection in red and sweep in yellow.

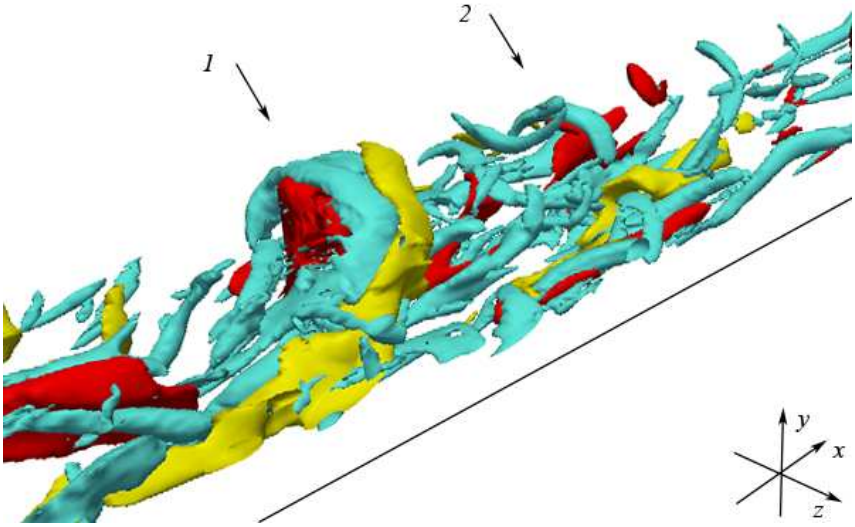

Figure 7.13 – Representation of hairpin vortices and quadrant events at $t^+ = 221$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin in cyan, ejection in red and sweep in yellow.

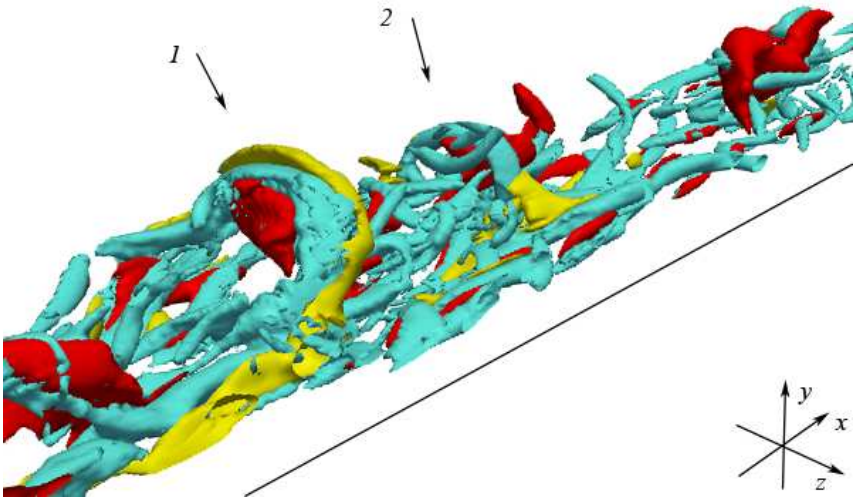

Figure 7.14 – Representation of hairpin vortices and quadrant events at $t^+ = 222$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin in cyan, ejection in red and sweep in yellow.

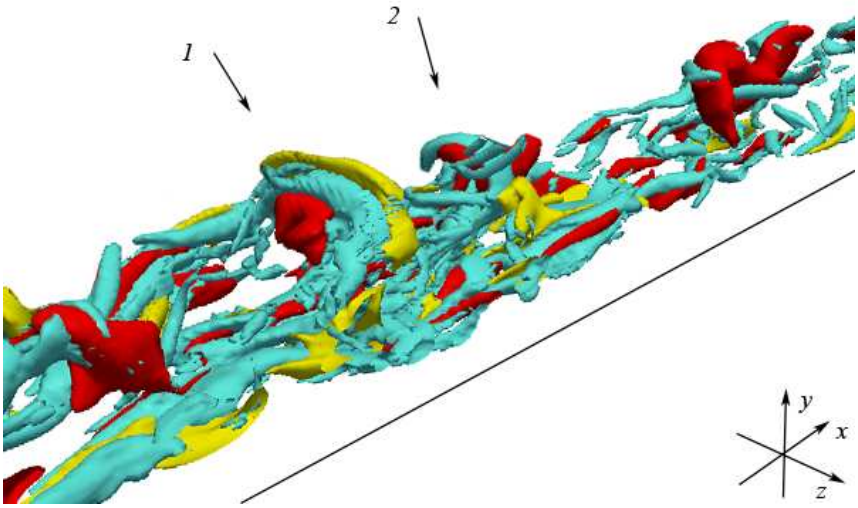

Figure 7.15 – Representation of hairpin vortices and quadrant events at $t^+ = 223$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin in cyan, ejection in red and sweep in yellow.

Figure 7.13 shows, more in detail, the interaction between ejections and sweeps in the primary hairpin. They are responsible of two opposite actions: ejection isosurface tries to push up the hairpin, while sweep isosurface let its legs stay close to the solid wall, providing a downward action also to the neck. This combined mechanism guarantees the morphological evolution and the preservation of the stability of the

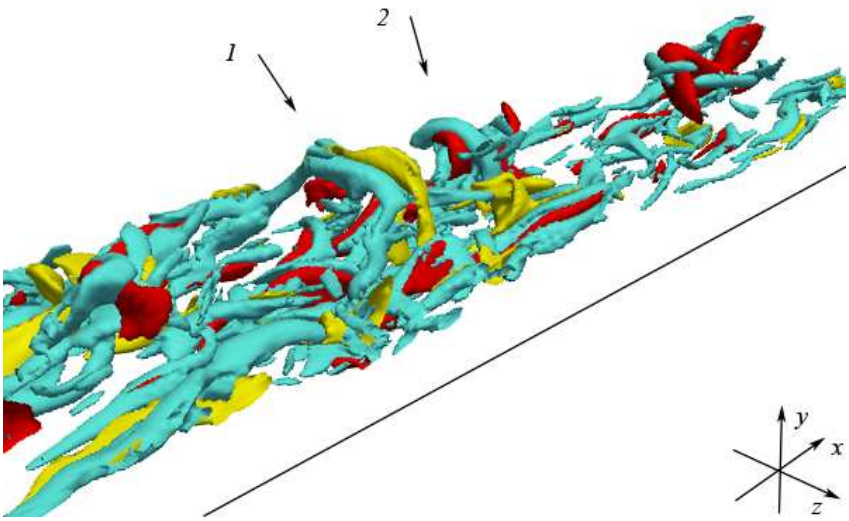

Figure 7.16 – Representation of hairpin vortices and quadrant events at $t^+ = 224$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin in cyan, ejection in red and sweep in yellow.

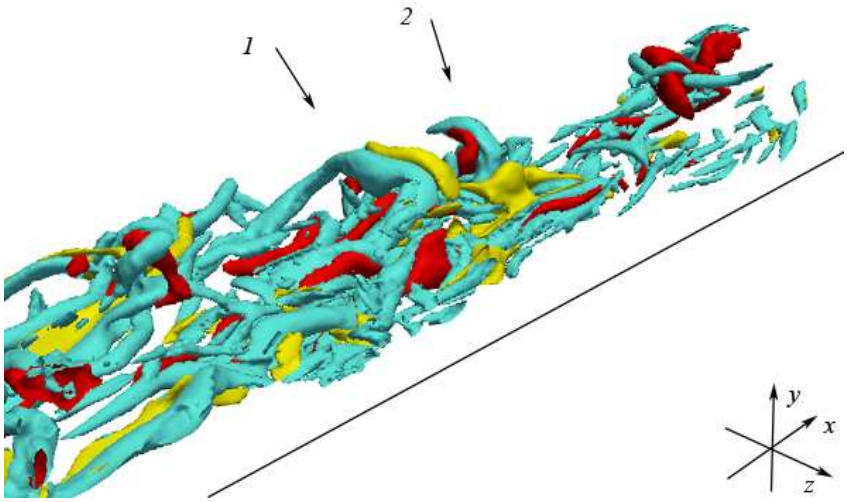

Figure 7.17 – Representation of hairpin vortices and quadrant events at $t^+ = 225$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin in cyan, ejection in red and sweep in yellow.

primary hairpin. The second hairpin, indeed, is going to define its structure under the action of only ejection isosurface. Figures 7.14-7.15 show the predominant event action for defining the time evolution of hairpin *(1)*: the sweep isosurface wraps up the hairpin's neck, forcing to stay close to the solid wall by deforming the head. The hairpin *(2)* is more visible and defined in this phase, characterized only by the action of the ejections. Figures 7.16-7.17, finally, show the disruption of hairpin *(1)* and the evolution and development of hairpin *(2)*, similar to that observed in the previous sequence.

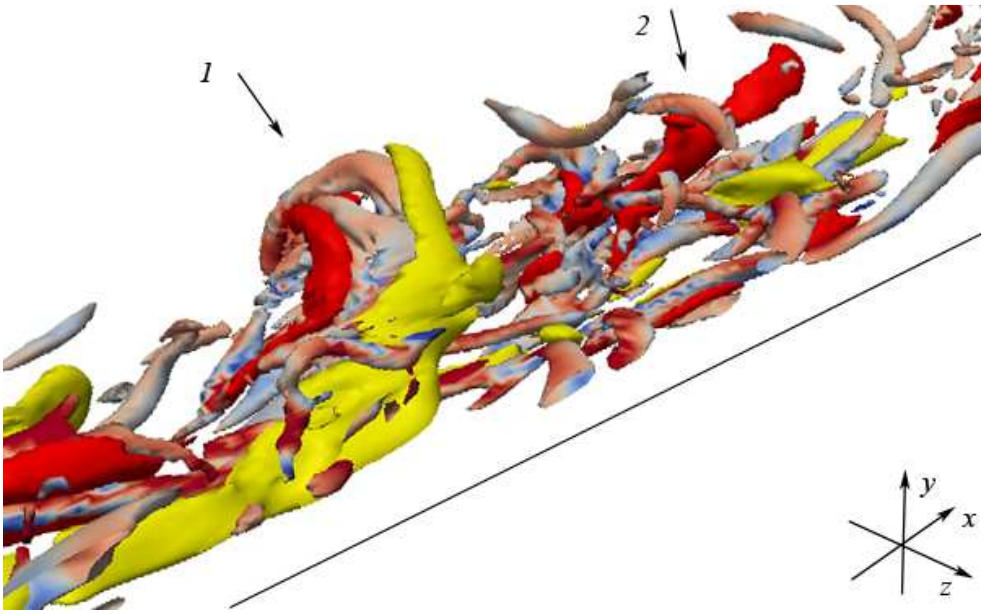

Figure 7.18 – Representation of hairpin vortices and quadrant events at $t^+ = 220$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin colored by using λ_r values from light-blue to light-red, ejection in red and sweep in yellow.

Figures 7.18-7.20 show some plots of the same sequence previously analyzed, whose hairpins are colored by using the λ_r values, that represent a measure of the local strength of stretching or compression of vortical structures. In this case, the intensity of λ_r is defined by using a chromatic scale from light-blue (the lowest values) to light-red (the higher values).

Considering the initial phase (Fig. 7.18), the ejection isosurfaces are responsible of the stretching of the heads, for both hairpins *(1)* and *(2)*, while sweep isosurfaces determine the compression of the corresponding legs.

In Fig. 7.19, the compression involves also the head of hairpin *(1)* because of the predominant effect of sweeps, causing the breaking down of structures (Fig. 7.20).

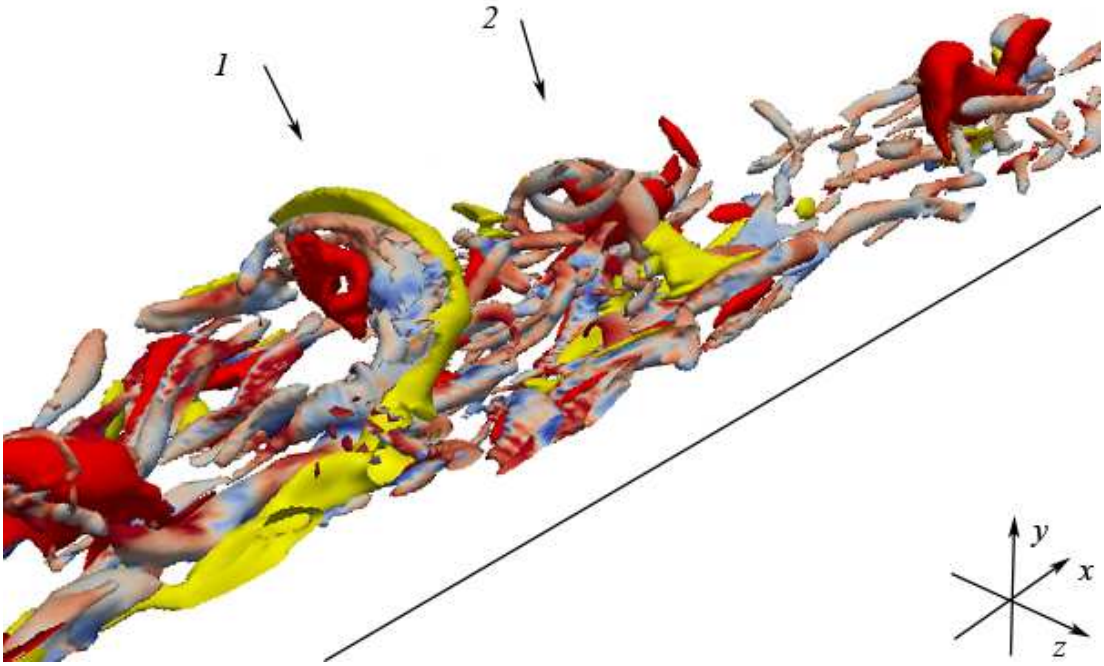

Figure 7.19 – Representation of hairpin vortices and quadrant events at $t^+ = 222$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin colored by using λ_r values from light-blue to light-red, ejection in red and sweep in yellow.

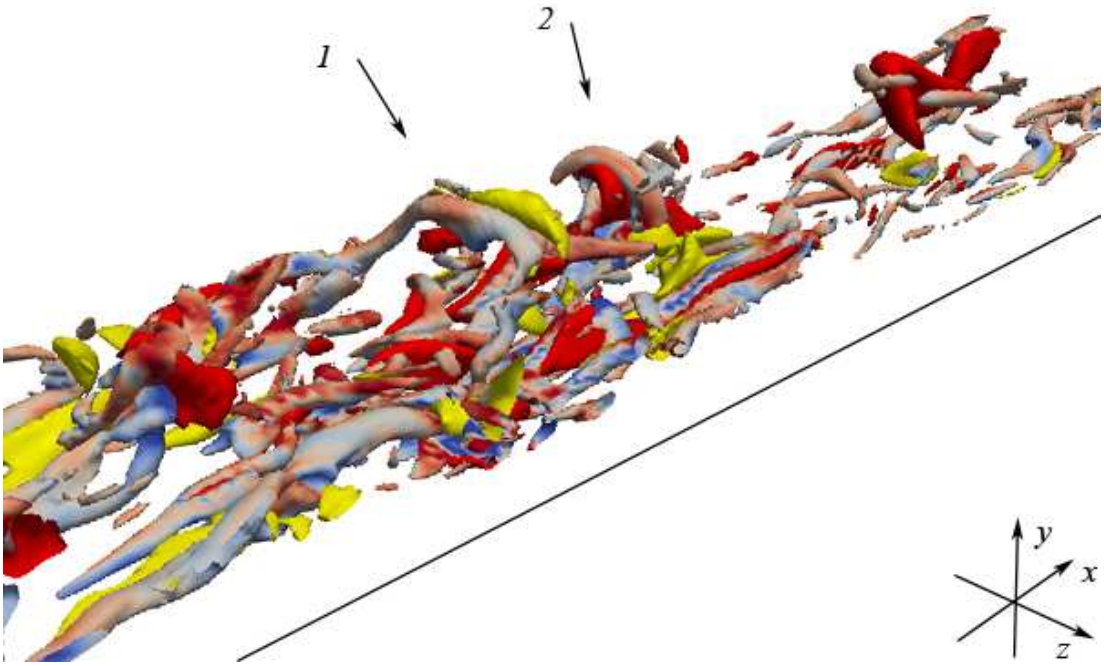

Figure 7.20 – Representation of hairpin vortices and quadrant events at $t^+ = 225$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin colored by using λ_r values from light-blue to light-red, ejection in red and sweep in yellow.

Finally, an interesting sequence (Figs. 7.20-7.30) of a single $\Omega-$shaped hairpin-vortex is described. Also in this case, flow structures are colored by using the λ_r values, while events are represented by using surfaces colored in red and yellow for ejections and sweeps, respectively.

Figure 7.21 shows an isolated structure at $t^+ = 450$: the growing phase is determined only by the ejection isosurfaces. At $t^+ = 451$ (Fig. 7.22), two sweep isosurfaces adjacent to the external side of the neck and legs appear: the sweep event is responsible of the legs compression, while ejection determines the stretching of the head, ensuring the stability of the vortex until the instant $t^+ = 452$ is reached (Fig. 7.23).

Figure 7.24 shows how sweep isosurfaces is going to overlap also the head of the vortex, forcing the vortex itself to stay close to the wall (Figs. 7.24-(7.27).

The predominant action of sweep isosurfaces determines the disruption of the hairpin (Figs. 7.29-7.31).

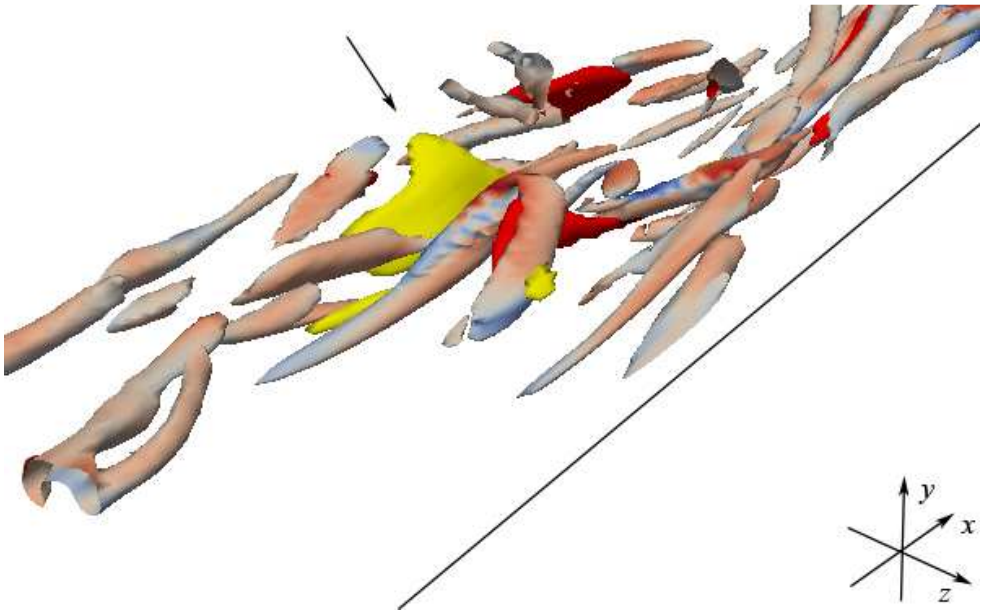

Figure 7.21 – Representation of a single vortex and quadrant events at $t^+ = 450$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin colored by using λ_r values from light-blue to light-red, ejection in red and sweep in yellow.

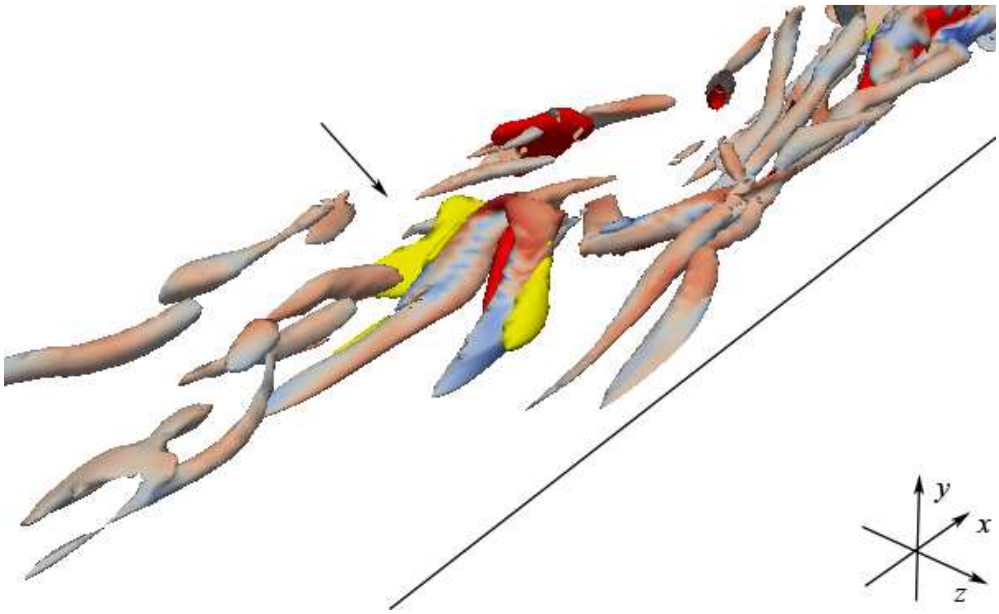

Figure 7.22 – Representation of a single vortex and quadrant events at $t^+ = 451$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin colored by using λ_r values from light-blue to light-red, ejection in red and sweep in yellow.

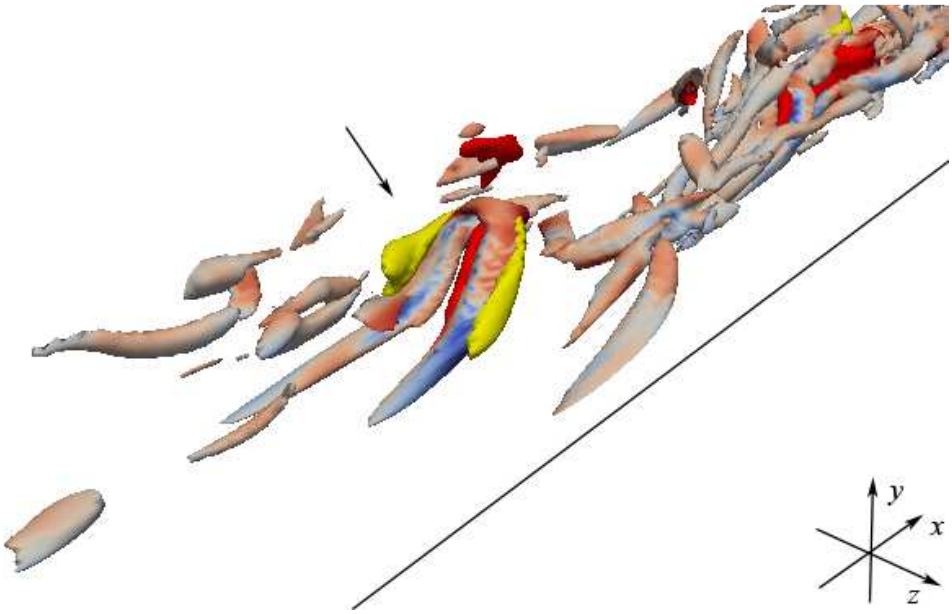

Figure 7.23 – Representation of a single vortex and quadrant events at $t^+ = 452$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin colored by using λ_r values from light-blue to light-red, ejection in red and sweep in yellow.

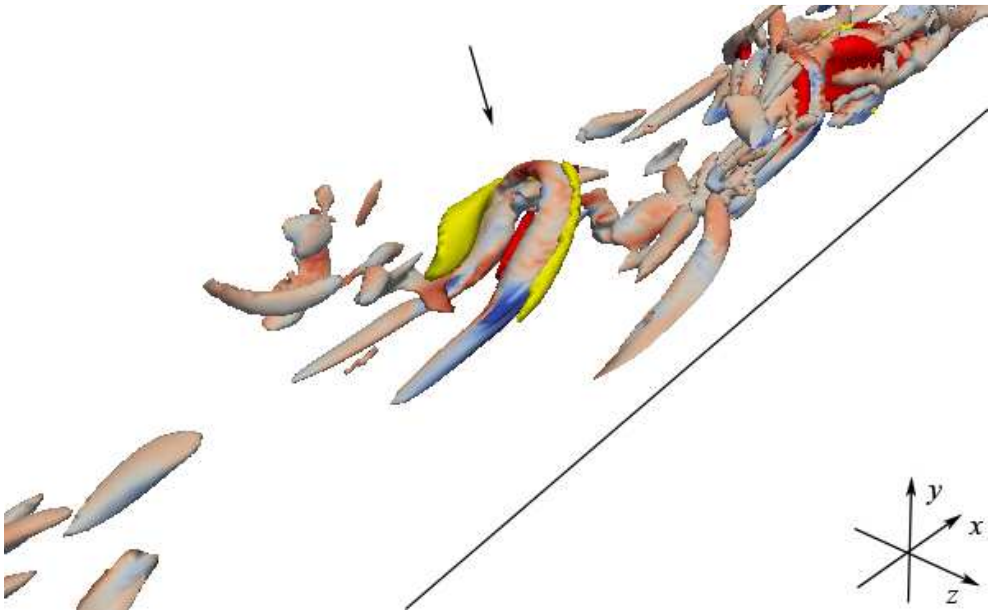

Figure 7.24 – Representation of a single vortex and quadrant events at $t^+ = 453$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin colored by using λ_r values from light-blue to light-red, ejection in red and sweep in yellow.

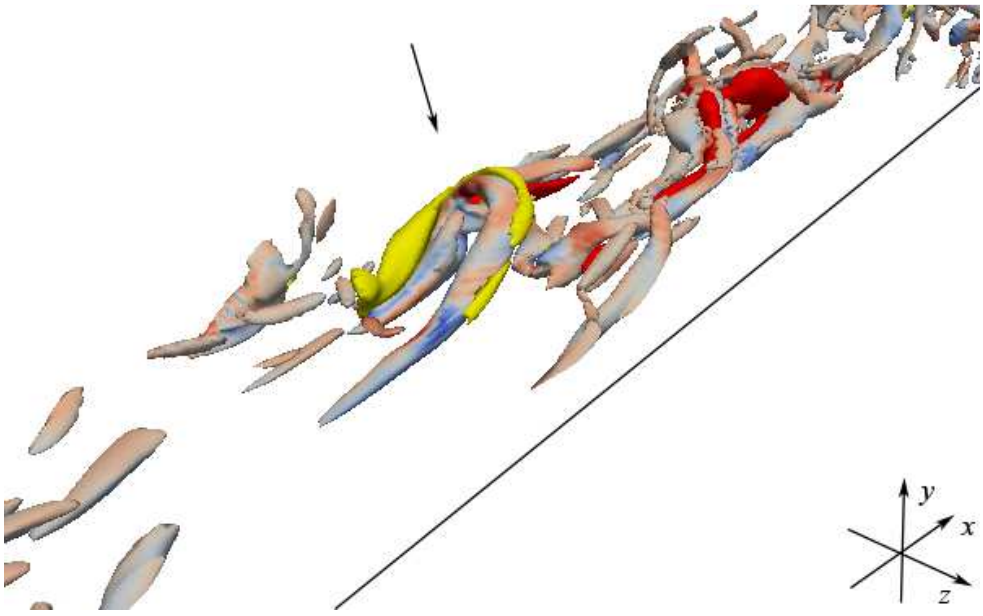

Figure 7.25 – Representation of a single vortex and quadrant events at $t^+ = 454$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin colored by using λ_r values from light-blue to light-red, ejection in red and sweep in yellow.

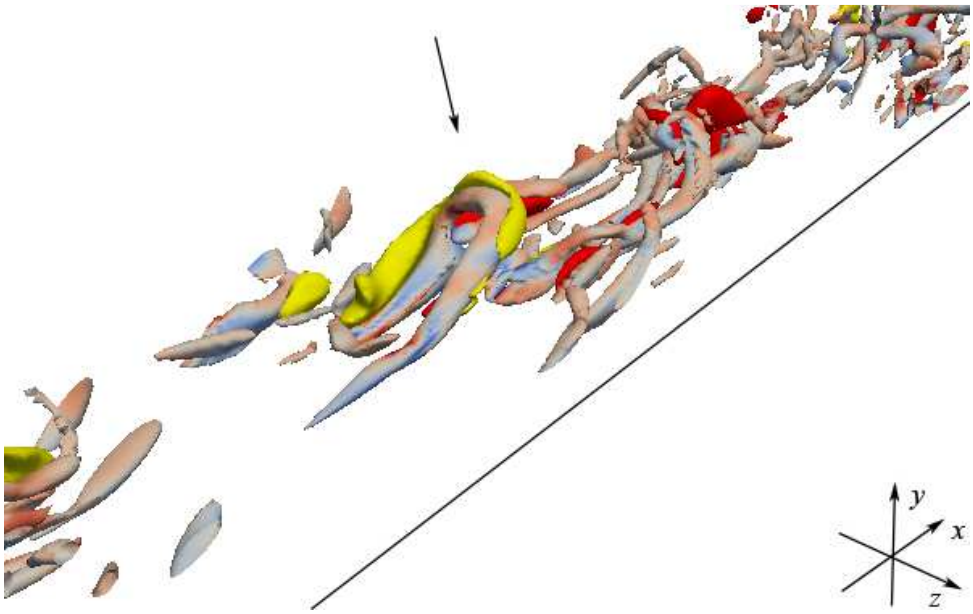

Figure 7.26 – Representation of a single vortex and quadrant events at $t^+ = 455$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin colored by using λ_r values from light-blue to light-red, ejection in red and sweep in yellow.

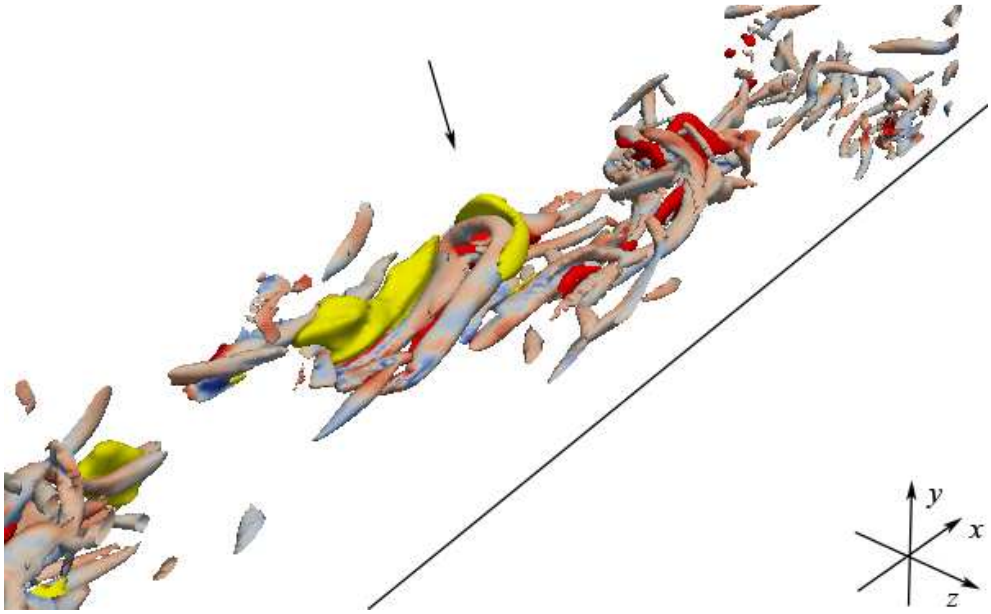

Figure 7.27 – Representation of a single vortex and quadrant events at $t^+ = 456$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin colored by using λ_r values from light-blue to light-red, ejection in red and sweep in yellow.

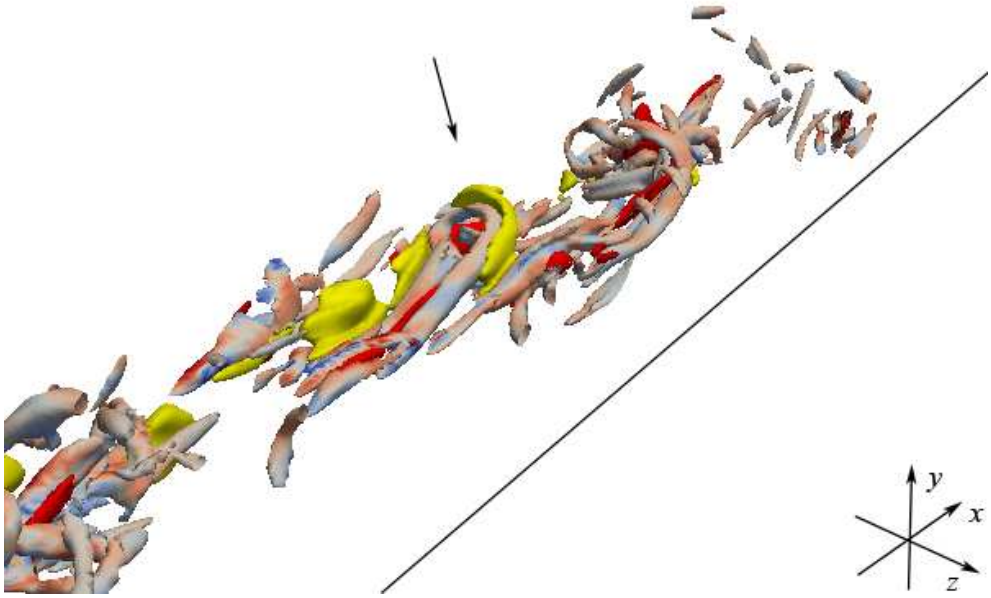

Figure 7.28 – Representation of a single vortex and quadrant events at $t^+ = 457$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin colored by using λ_r values from light-blue to light-red, ejection in red and sweep in yellow.

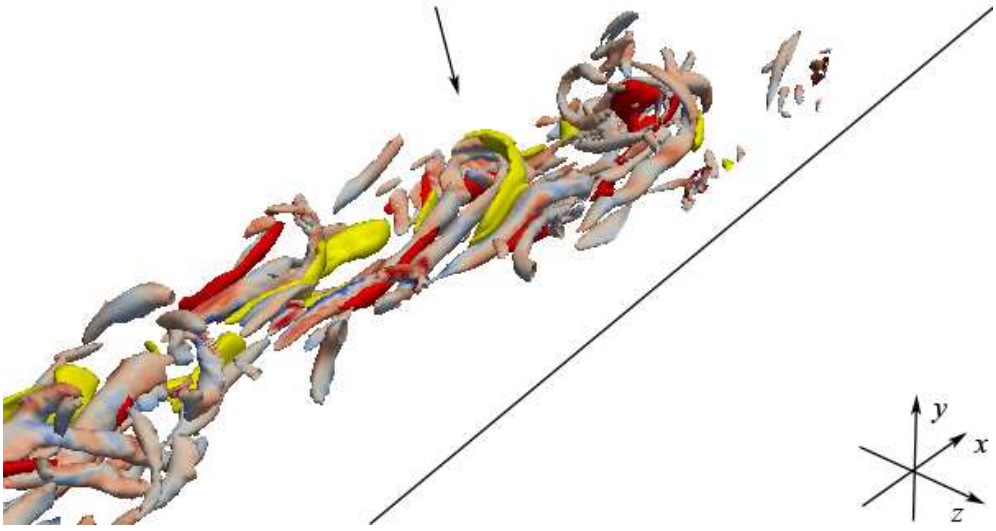

Figure 7.29 – Representation of a single vortex and quadrant events at $t^+ = 458$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin colored by using λ_r values from light-blue to light-red, ejection in red and sweep in yellow.

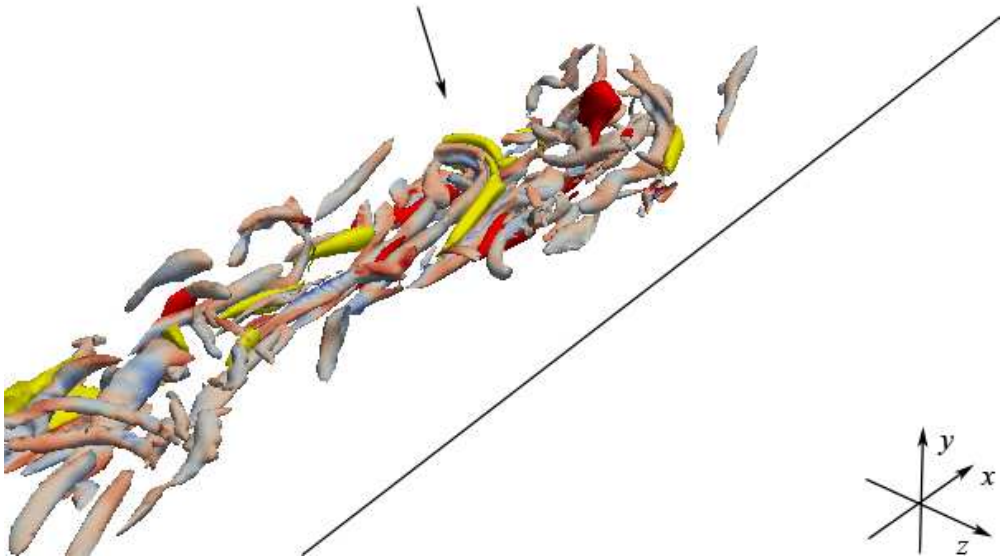

Figure 7.30 – Representation of a single vortex and quadrant events at $t^+ = 459$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin colored by using λ_r values from light-blue to light-red, ejection in red and sweep in yellow.

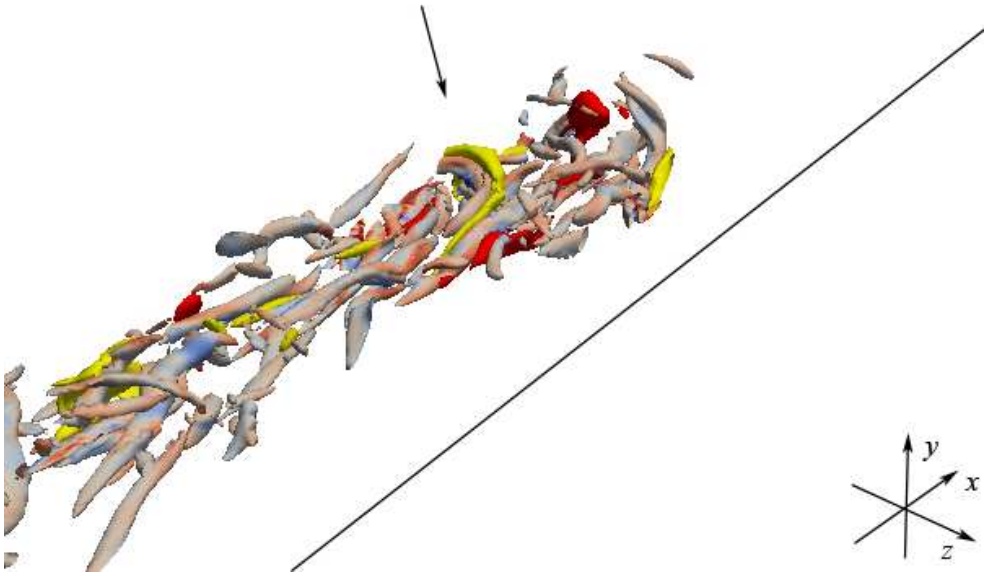

Figure 7.31 – Representation of a single vortex and quadrant events at $t^+ = 460$ and $\lambda_{ci} = 9.25 \cdot 10^{-4}$: hairpin colored by using λ_r values from light-blue to light-red, ejection in red and sweep in yellow.

The major finding of this discussion consists in the clear evidence that the process of morphological evolution of the primary hairpin vortex (the head rises upwards, while the legs stay close to the wall) and the consequent persistency and stability of

vortical structures is due to the combined actions of ejection and sweep events. Where this combined effect is not present the process of development of a persistent hairpin is not successful, as demonstrated by the first sequence analyzed, while the last two sequences described a more stable and complex morphological evolution of a single hairpin in the turbulent flow field.

7.4.2 Vortical structures in the wall region at $\mathbf{Re_\tau = 400}$

Considering the DNS database at $\mathrm{Re}_\tau = 400$ and applying the λ_{ci} criterion to the velocity fluctuations, it is possible to show the flow structures that fill the plane channel. Figure 7.32 shows an overall 3D view of the physical domain: hairpin-vortices are, in this case, many more with respect to the morphological consistency of flow structures educed from the DNS database at $\mathrm{Re}_\tau = 200$, determining a more composite and ordered flow pattern. Hairpins distribution among the two solid walls is shown in Fig. 7.33, while views of hairpin structures over each wall are shown in Figs. 7.34-7.35, at the lower and the upper wall, respectively.

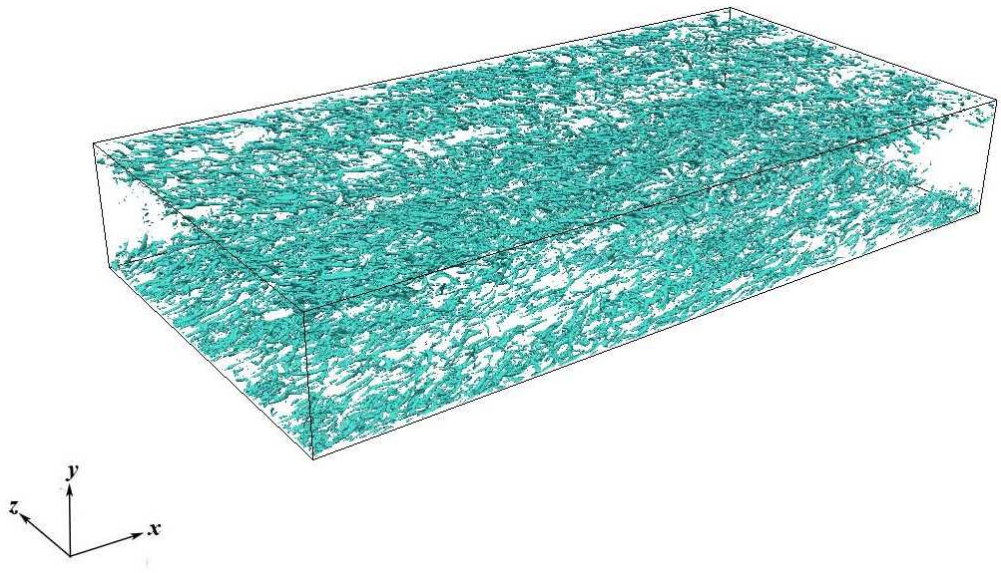

Figure 7.32 – Vortical structures in the computing domain at $\mathrm{Re}_\tau = 400$.

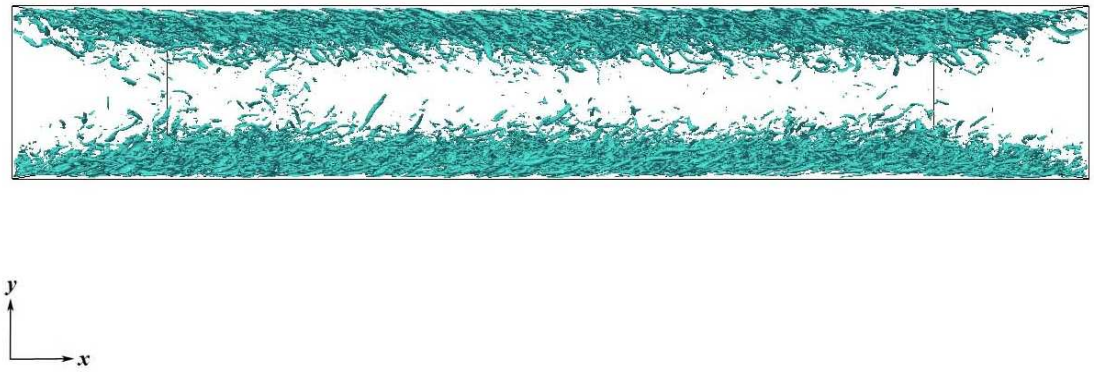

Figure 7.33 – Vortical structures in the computing domain at $\mathrm{Re}_\tau = 400$: lateral view.

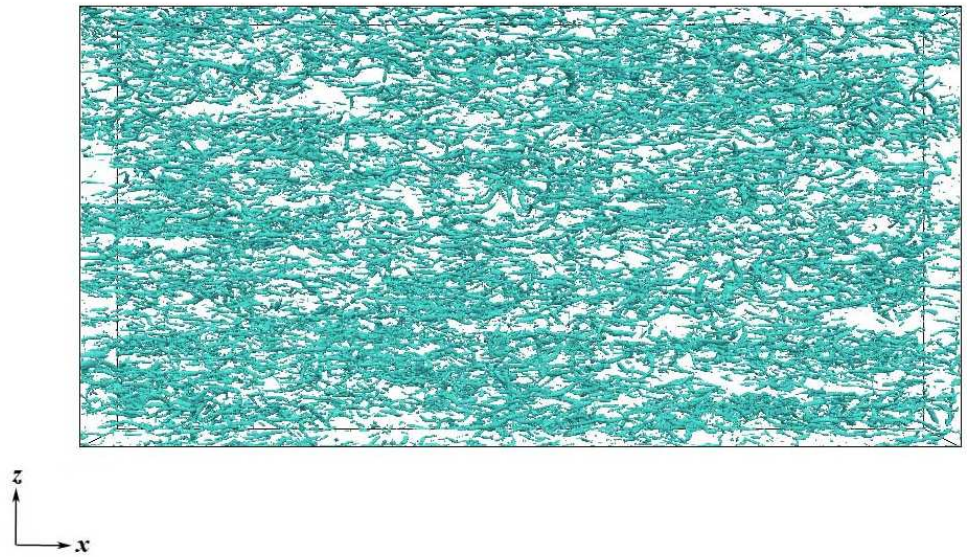

Figure 7.34 – Vortical structures in the computing domain at $\mathrm{Re}_\tau = 400$: inferior wall.

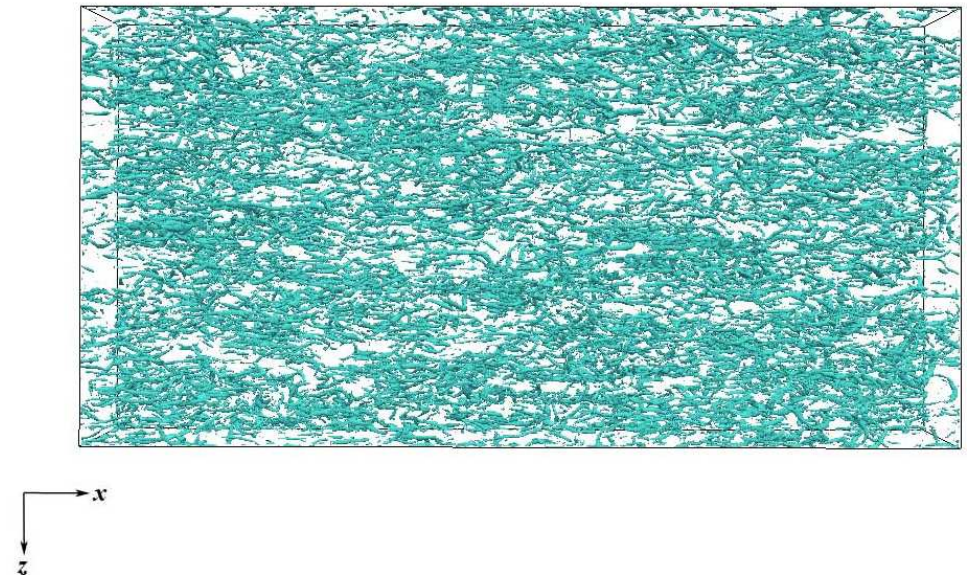

Figure 7.35 – Vortical structures in the computing domain at $\mathrm{Re}_\tau = 400$: superior wall.

One of the first results in studying the structures of the turbulent boundary layer at high Reynolds number, confirmed by numerical results, is related to the visualization of streaks, that describe the flow organization where alternating unsteady arrays of high- and low-speed regions are aligned in the streamwise direction. Figures 7.36-7.38 show the distribution of streaks in the plane channel, considering only the lower wall, in terms of interaction between turbulent events and vortical structures.

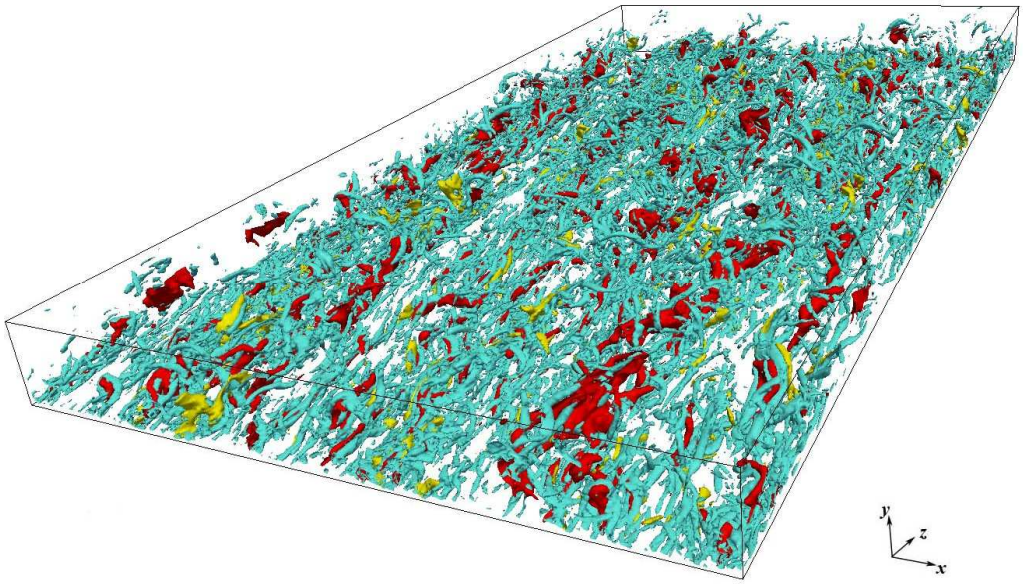

Figure 7.36 – Representation of vortical structures and quadrant events on the lower wall at $t^+ = 3$ and $\lambda_{ci} = 2.625 \cdot 10^{-4}$: hairpin in cyan, ejection in red and sweep in yellow.

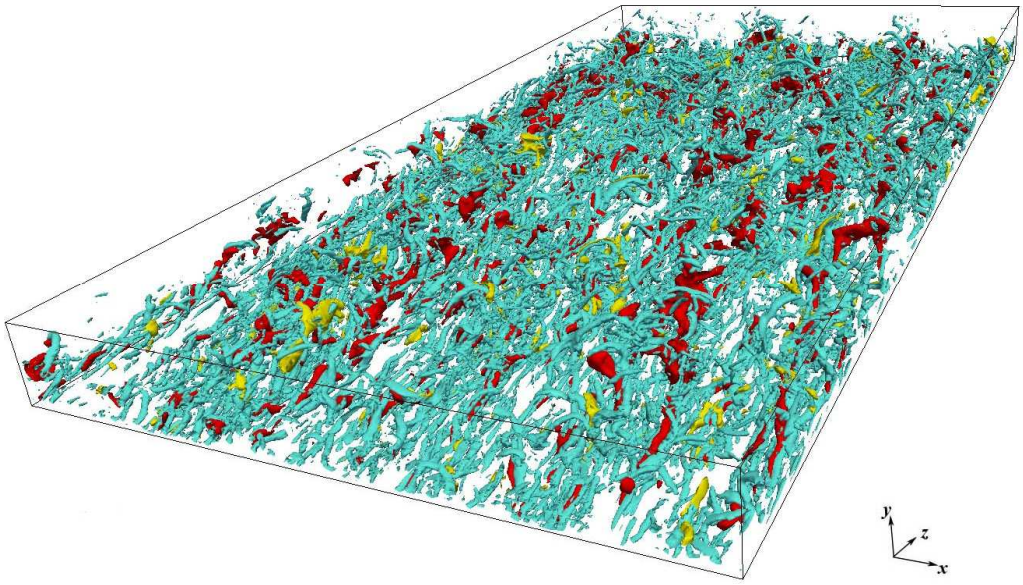

Figure 7.37 – Representation of vortical structures and quadrant events on the lower wall at $t^+ = 4$ and $\lambda_{ci} = 2.625 \cdot 10^{-4}$: hairpin in cyan, ejection in red and sweep in yellow.

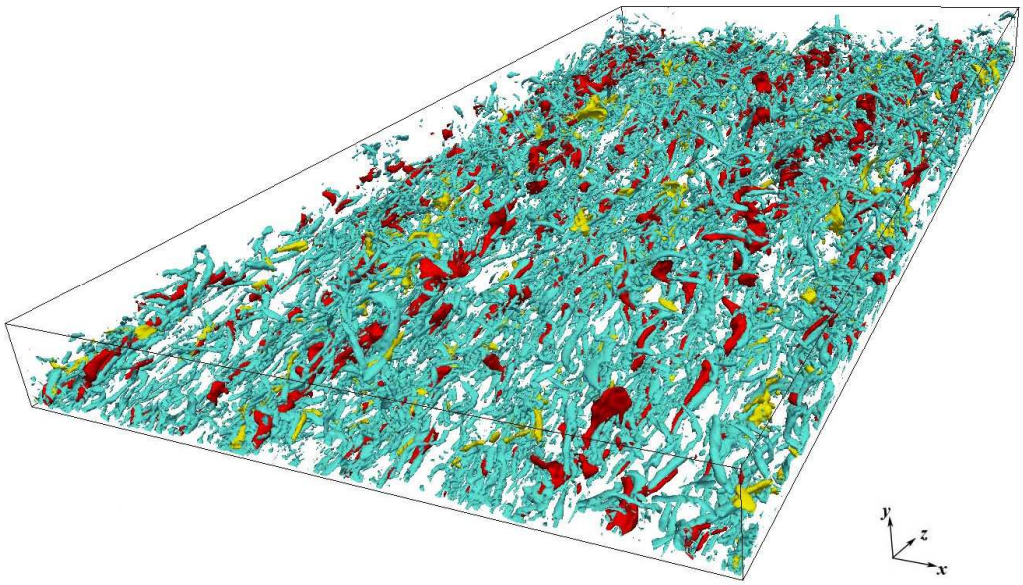

Figure 7.38 – Representation of vortical structures and quadrant events on the lower wall at $t^+ = 5$ and $\lambda_{ci} = 2.625 \cdot 10^{-4}$: hairpin in cyan, ejection in red and sweep in yellow.

In this case, the presence of vortical structures in the turbulent boundary layer, organized as a forest of hairpins, is of direct evidence, thanks to the availability of high accurate DNS database. Moving outward from the wall, many vortices whit different scale, size, strength and orientation appear: this transition suggests the presence of an inner region, with persistent streaky structures, and an outer region, dominated by vortex motions of various size.

Figures 7.39-7.42 show the isosurfaces of λ_{ci} colored by using the local values of the streamwise velocity u, where higher u are represented by red color. A typical flow field organization, where low speed zone (in green) are characterized by smaller scale hairpins in the inner region, overlapped by large scale hairpins in the outer region, characterized by high velocity (in red).

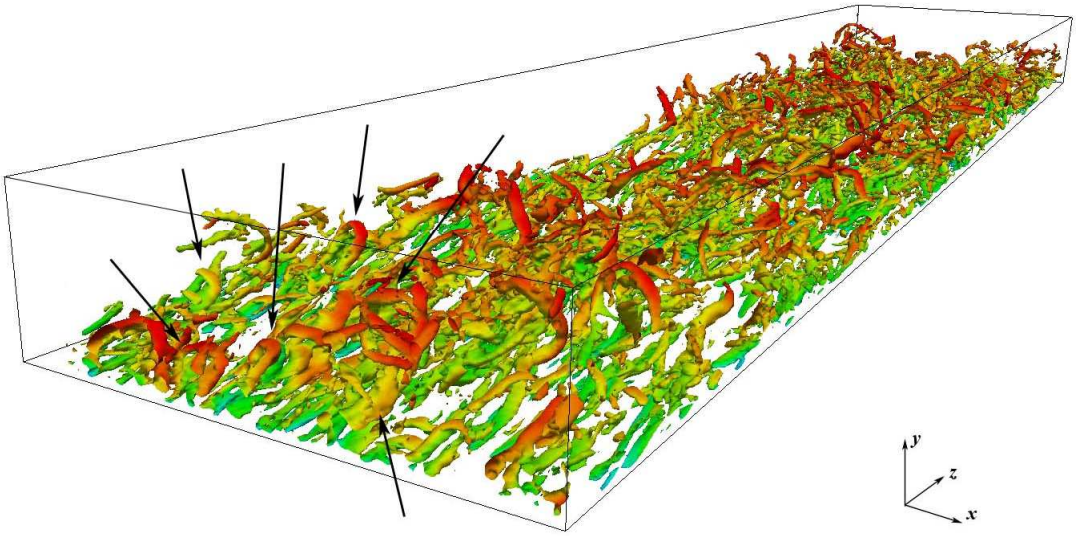

Figure 7.39 – Representation of vortical structures at $t^+ = 180$ and $\lambda_{ci} = 2.625 \cdot 10^{-4}$: hairpin colored by using the local value of *u* velocity.

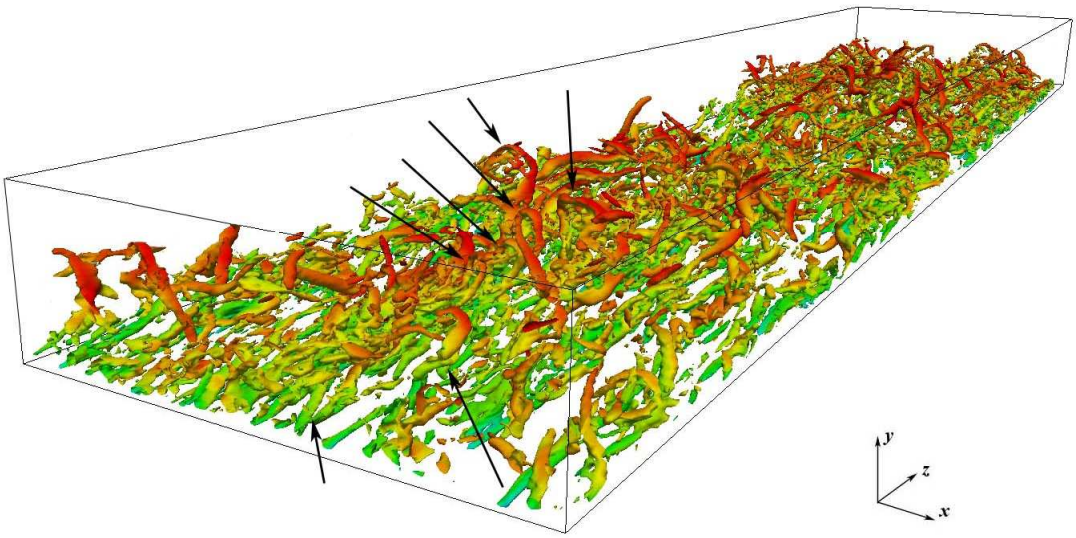

Figure 7.40 – Representation of vortical structures at $t^+ = 181$ and $\lambda_{ci} = 2.625 \cdot 10^{-4}$: hairpin colored by using the local value of *u* velocity.

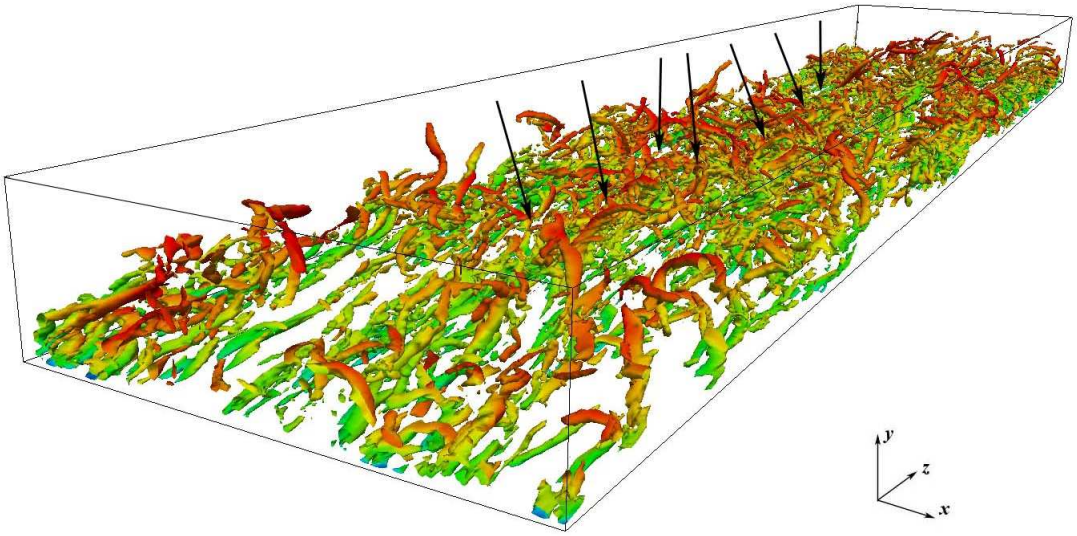

Figure 7.41 – Representation of vortical structures at $t^+ = 182$ and $\lambda_{ci} = 2.625 \cdot 10^{-4}$: hairpin colored by using the local value of *u* velocity.

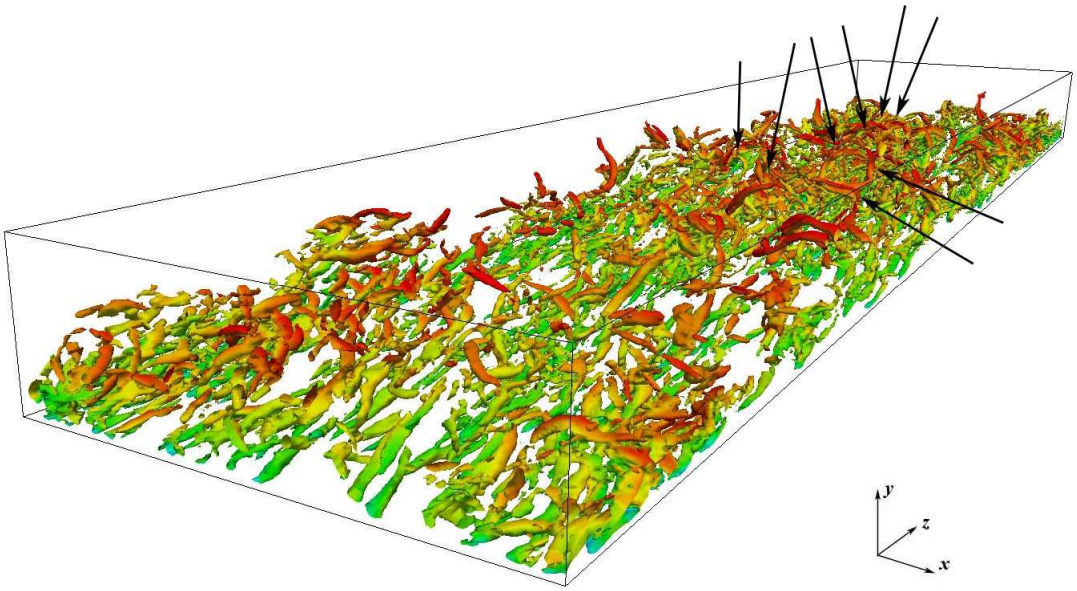

Figure 7.42 – Representation of vortical structures at $t^+ = 183$ and $\lambda_{ci} = 2.625 \cdot 10^{-4}$: hairpin colored by using the local value of *u* velocity.

The arrows indicate vortices that are grouped together in streamwise-aligned packets, that grow upwards and propagate with small velocity, so that their spatial arrangement has a long lifetime.

The same flow structure distribution can be analyzed in terms of interaction between hairpins and events, detected by applying the quadrant analysis (Figs. 7.43-7.46). As represented in Fig. 7.43, ejections occur often in temporal succession: in fact, the most relevant part of the turbulent production process in the boundary layer occurs during outward ejections of low-speed fluid and inrushes of high-speed fluid flow towards the wall. Thus, the near-wall turbulence production process appears as an intermittent cyclic sequence of events, typical of the bursting phenomenon. Arrows indicate the vortical structures and events help to recognize more specifically the hairpins that form the packet. While the packet maintains its symmetry within the flow field, vortices are asymmetric with legs of unequal size, characterized by distortions caused by the stretching and compression actions of other vortices.

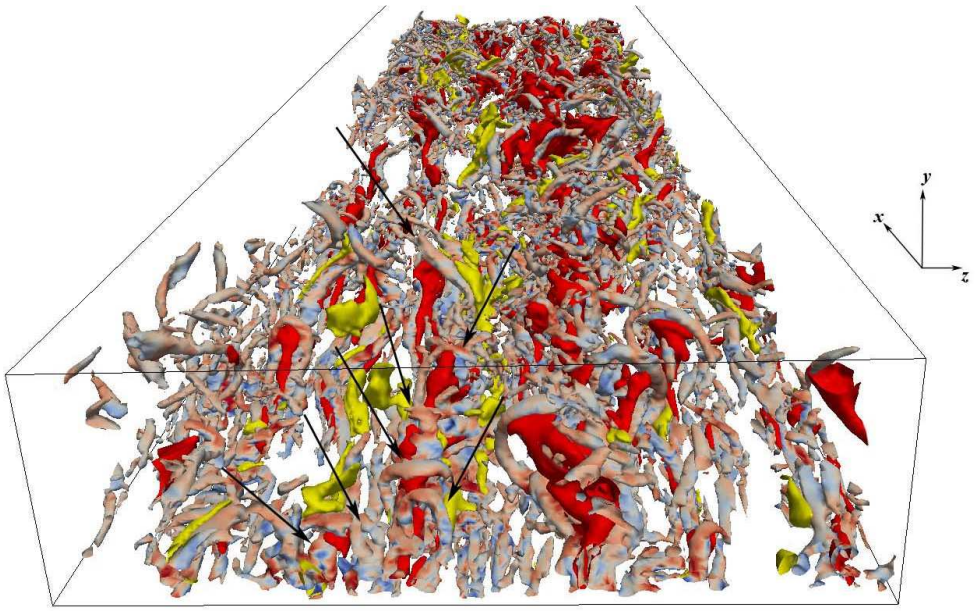

Figure 7.43 – Oblique view of vortical structures and events at $t^+ = 180$ and $\lambda_{ci} = 2.625 \cdot 10^{-4}$: hairpin colored by using λ_r values from light-blue to light-red, ejection in red and sweep in yellow.

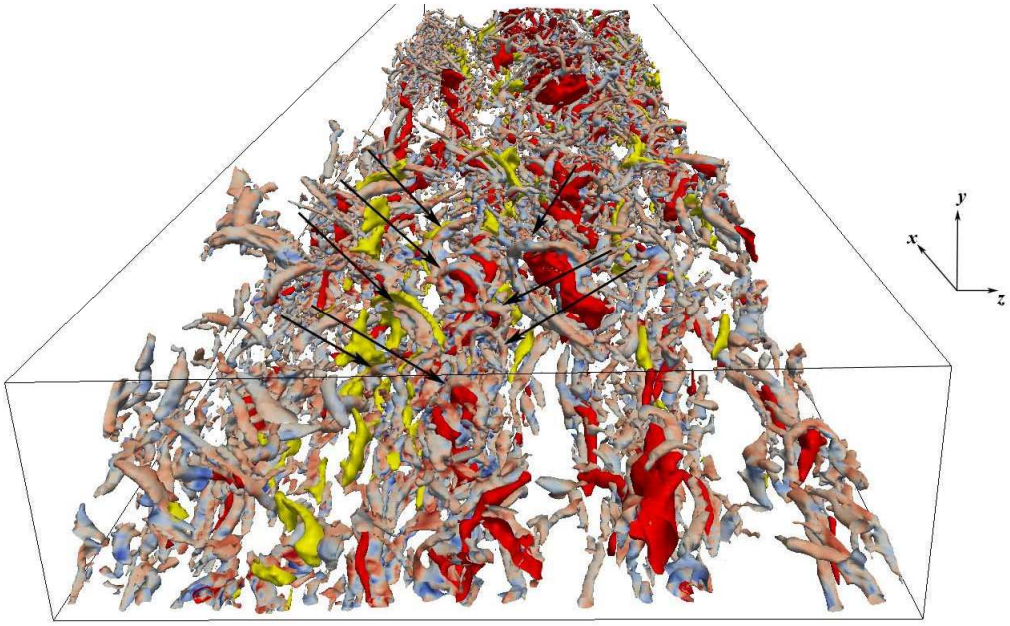

Figure 7.44 – Oblique view of vortical structures and events at $t^+ = 181$ and $\lambda_{ci} = 2.625 \cdot 10^{-4}$: hairpin colored by using λ_r values from light-blue to light-red, ejection in red and sweep in yellow.

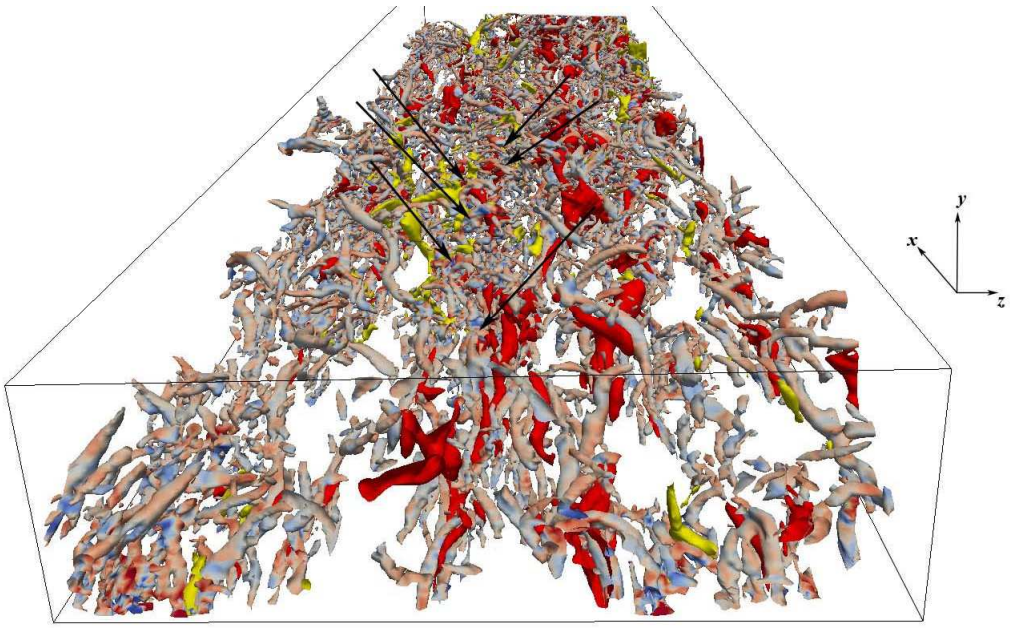

Figure 7.45 – Oblique view of vortical structures and events at $t^+ = 182$ and $\lambda_{ci} = 2.625 \cdot 10^{-4}$: hairpin colored by using λ_r values from light-blue to light-red, ejection in red and sweep in yellow.

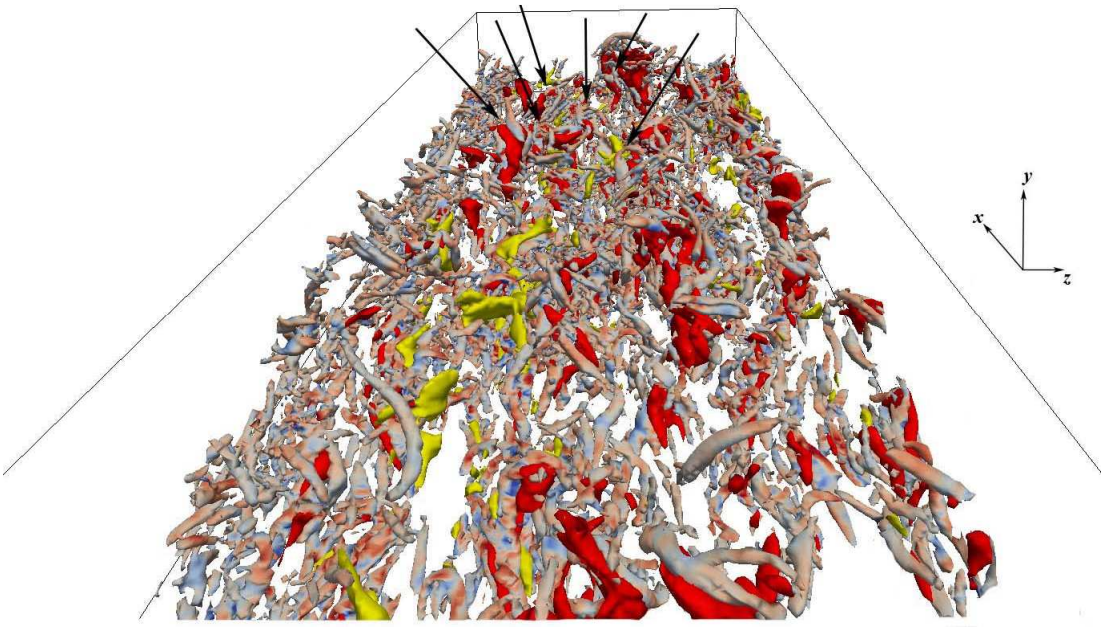

Figure 7.46 – Oblique view of vortical structures and events at $t^+ = 183$ and $\lambda_{ci} = 2.625 \cdot 10^{-4}$: hairpin colored by using λ_r values from light-blue to light-red, ejection in red and sweep in yellow.

Thanks to these results, hairpins packets are observed most clearly, confirming that these configurations (packets and envelope of packets) are most frequently at high Reynolds number flows than in low Reynolds one [134].

7.4.3 Vortical structures in the wall region at $\mathbf{Re_\tau = 600}$

All the previously conclusions done considering the flow structures at $\mathrm{Re}_\tau = 200, 400$ are confirmed and synthesized by those extracted from the computed velocity field at $\mathrm{Re}_\tau = 600$. Also in this case, after the application of the λ_{ci} criterion for vortex-detection to the fluctuations of velocities, it is possible to visualize the flow field, that appears full of turbulent structures adjacent to both the upper and the lower walls, respectively. The representation of vortical structures at $t^+ = 1$ and $\lambda_{ci} = 1.44 \cdot 10^{-4}$ is shown in Fig. 7.47; Fig. 7.48 shows a lateral view of the channel, in order to appreciate the density of vortical structures close to the walls; finally, Figs. 7.49 and 7.50 show the representation of vortical structures at the lower and the upper wall, respectively.

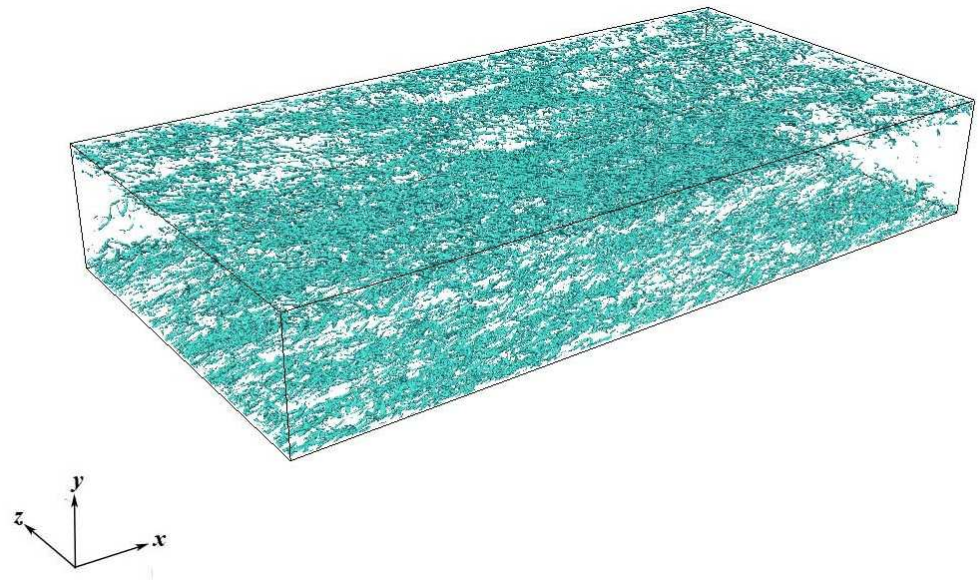

Figure 7.47 – Vortical structures in the computing domain at $\mathrm{Re}_\tau = 600$.

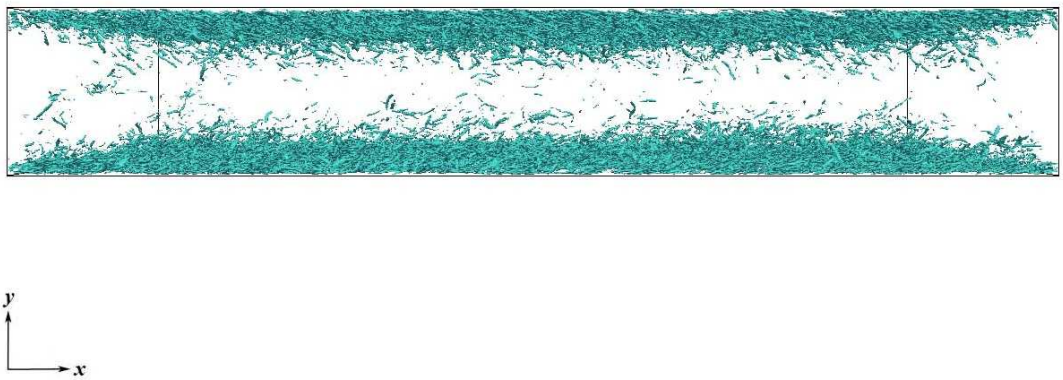

Figure 7.48 – Vortical structures in the computing domain at $\mathrm{Re}_\tau = 600$: lateral view.

Observing figures about flow field at $\mathrm{Re}_\tau = 600$, it is possible to note how complex is the flow structure morphological distribution: the streaks are better defined and the corresponding packets are many more with respect to those founded at lower Reynolds numbers considered in this work.

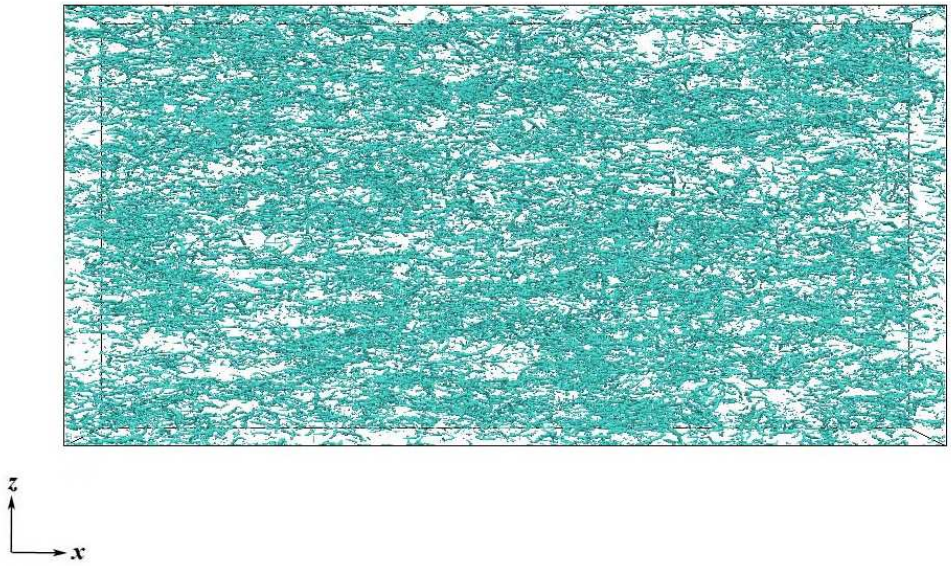

Figure 7.49 – Vortical structures in the computing domain at $\mathrm{Re}_\tau = 600$: inferior wall.

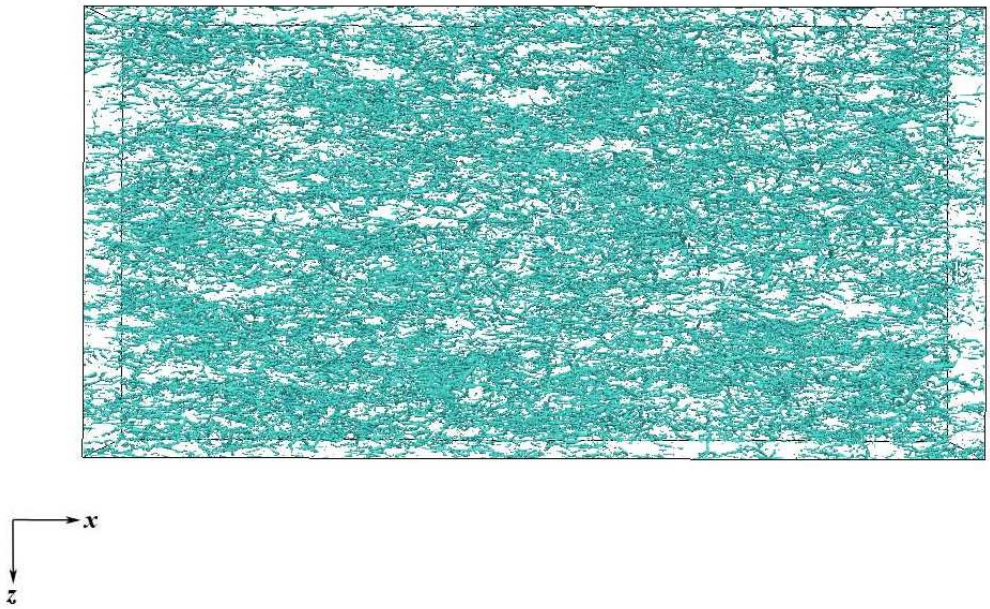

Figure 7.50 – Vortical structures in the computing domain at $\mathrm{Re}_\tau = 600$: superior wall.

Figures 7.51-7.54 show the representation of vortical structures where hairpins are colored by using the local values of the streamwise velocity , in order to evaluate the propagation of the streaks within the plane channel and their interaction near the centerline. There are hairpins with a remarkable elevation of their heads because of stressed by high values of u velocity and hairpins characterized by low values of u velocity that are responsible of the propagation of the packets. Interactions

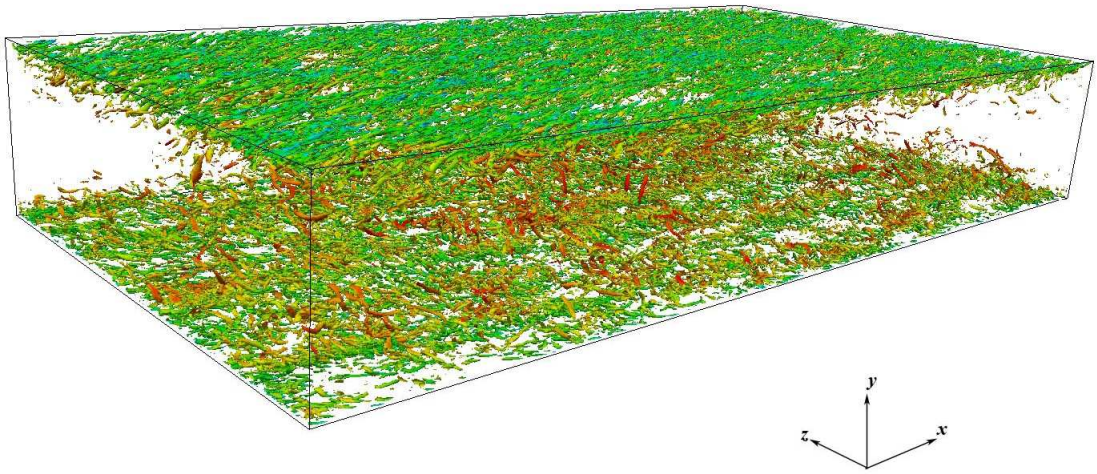

Figure 7.51 – Representation of vortical structures at $t^+ = 45$ and $\lambda_{ci} = 1.44 \cdot 10^{-4}$: hairpin colored by using the streamwise velocity values u.

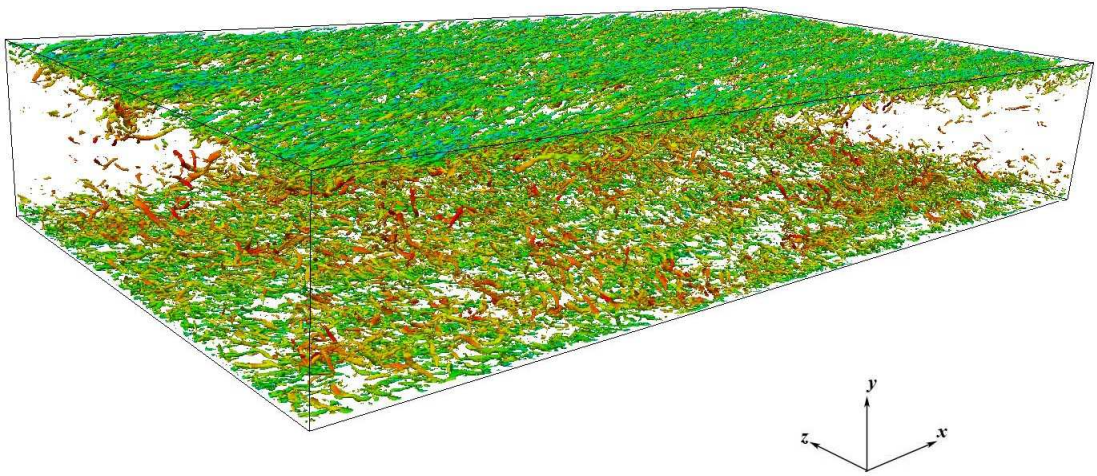

Figure 7.52 – Representation of vortical structures at $t^+ = 46$ and $\lambda_{ci} = 1.44 \cdot 10^{-4}$: hairpin colored by using the streamwise velocity values u.

between vortices increase the complexity of the flow pattern, making the interpretation, understanding and visualization of the channel sometimes very difficult.

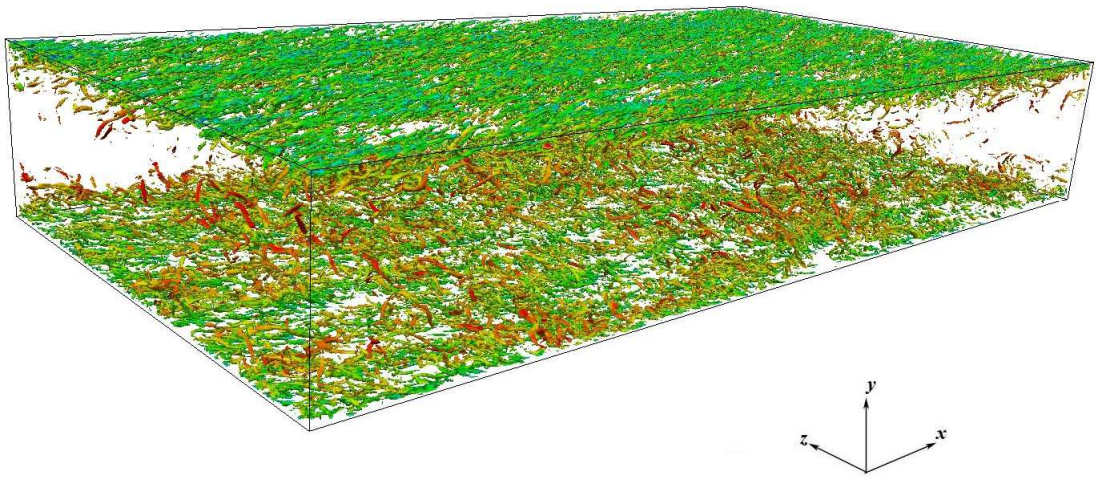

Figure 7.53 – Representation of vortical structures at $t^+ = 47$ and $\lambda_{ci} = 1.44 \cdot 10^{-4}$: hairpin colored by using the streamwise velocity values u.

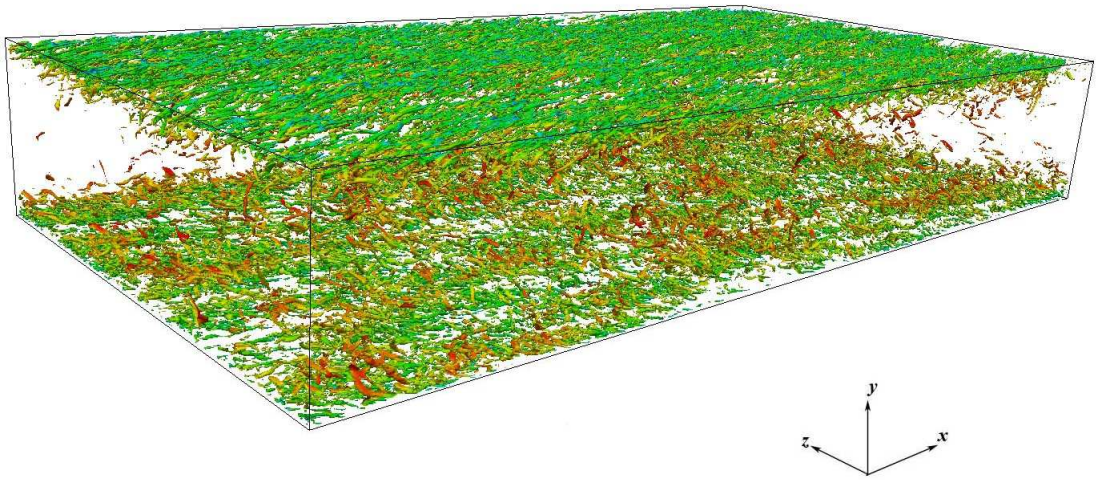

Figure 7.54 – Representation of vortical structures at $t^+ = 48$ and $\lambda_{ci} = 1.44 \cdot 10^{-4}$: hairpin colored by using the streamwise velocity values u.

Figures 7.55-7.58 show the same sequence of samples, from $t^+ = 45$ to $t^+ = 48$, focusing on the interaction between vortical structures and events: the lower wall is considered because here turbulence manifests its main effects.

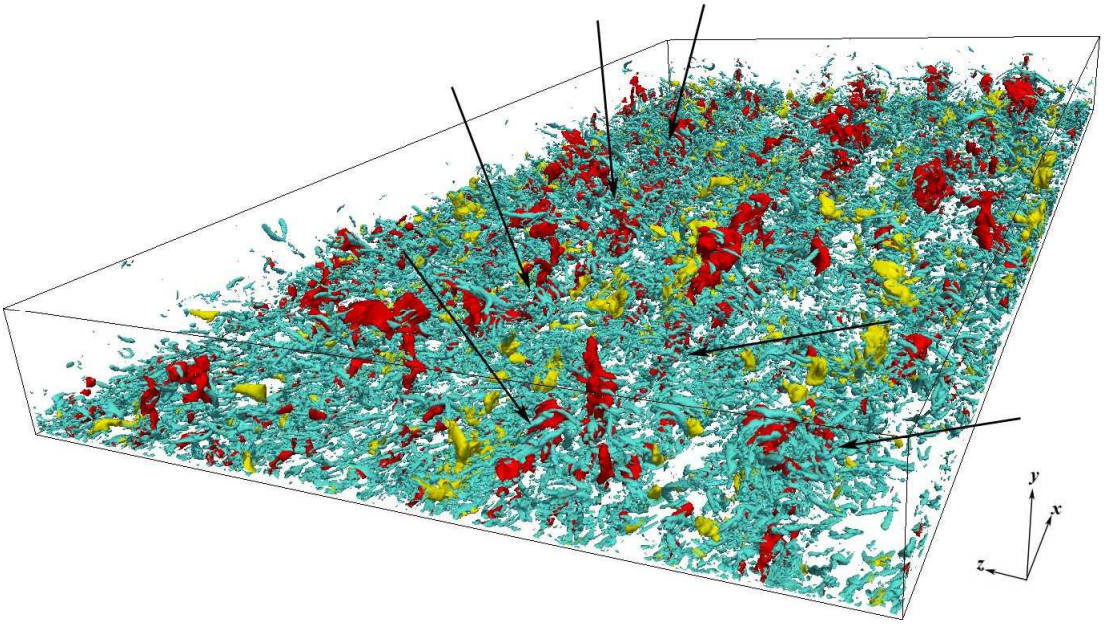

Figure 7.55 – Representation of vortical structures and quadrant events at $t^+ = 45$ and $\lambda_{ci} = 1.44 \cdot 10^{-4}$: hairpin in cyan, ejection in red and sweep in yellow.

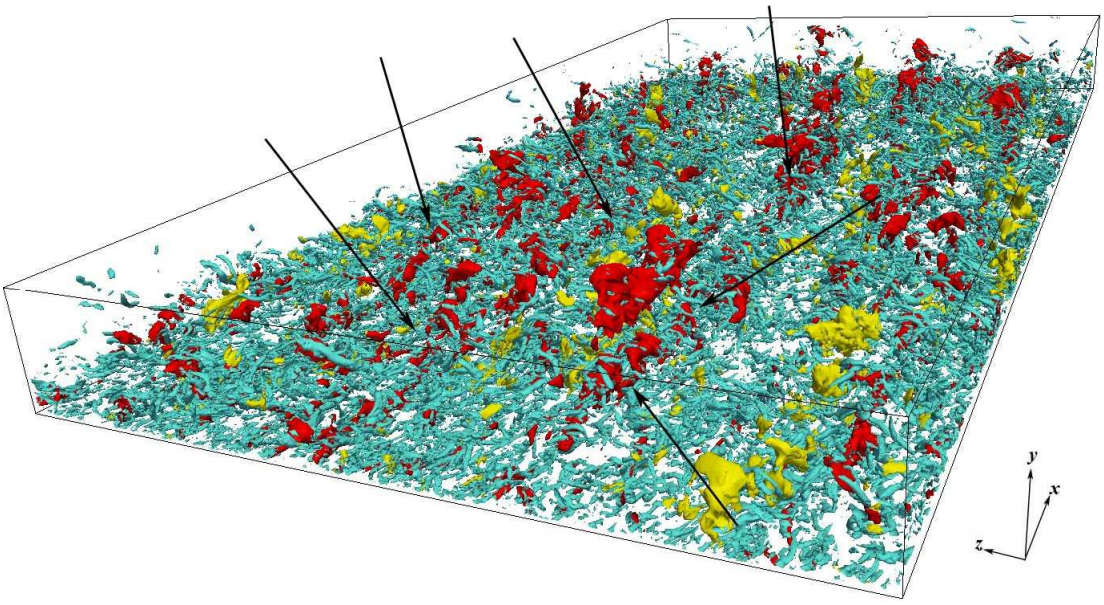

Figure 7.56 – Representation of vortical structures and quadrant events at $t^+ = 46$ and $\lambda_{ci} = 1.44 \cdot 10^{-4}$: hairpin in cyan, ejection in red and sweep in yellow.

It is evident how complex are the events that occur at high-Reynolds numbers: there are many more hairpins that form a lot of packets whose coherent alignment created an induced backflow region inside the flow field: it is much longer than the backflow

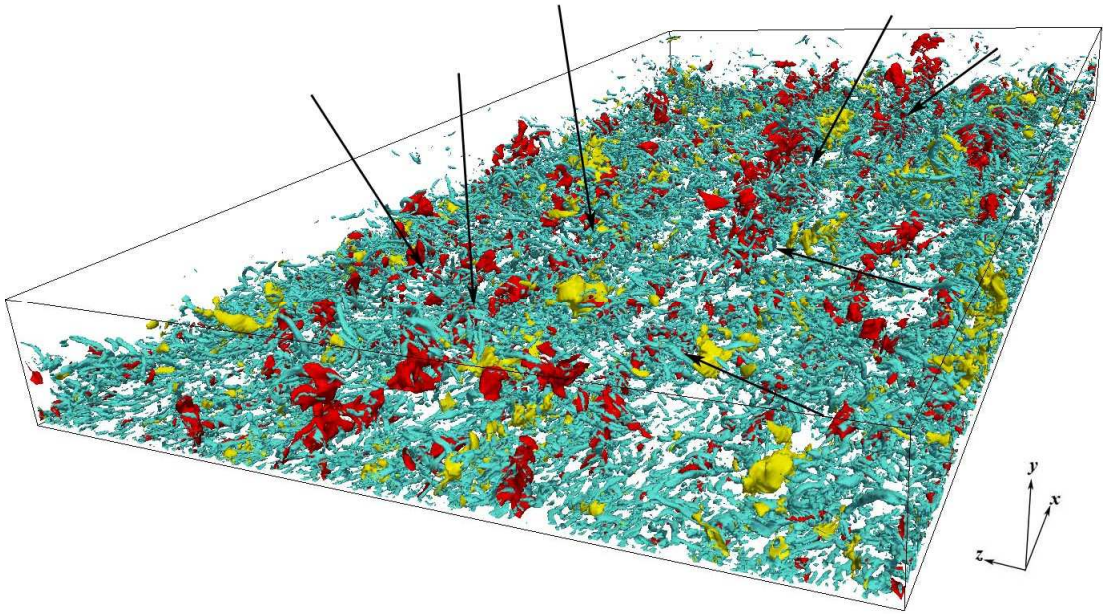

Figure 7.57 – Representation of vortical structures and quadrant events at $t^+ = 47$ and $\lambda_{ci} = 1.44 \cdot 10^{-4}$: hairpin in cyan, ejection in red and sweep in yellow.

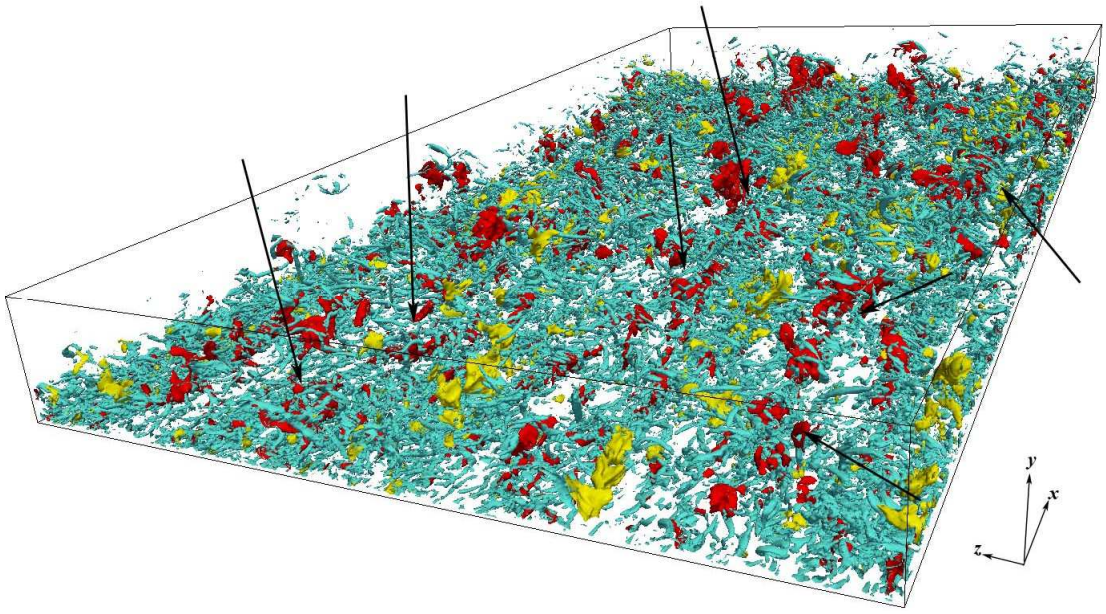

Figure 7.58 – Representation of vortical structures and quadrant events at $t^+ = 48$ and $\lambda_{ci} = 1.44 \cdot 10^{-4}$: hairpin in cyan, ejection in red and sweep in yellow.

induced by a single vortex, as observed by [148] and [149]. Arrows, in particular, indicate some of the packets that are present in the flow field, that confirm the observations done by [150] and [151] about the frequency of ejection events, that

allow the grouping of vortices in turbulent bursts, with respect to sweep events, that pump fluid downward from the outer region of the channel.

Furthermore, the intense action of ejection and sweep events close to the walls suggests how events strongly contribute to the mean Reynolds stress with the increase of the Reynolds number.

Chapter 8
Conclusions

This thesis has proposed a novel implementation for the Direct Numerical Simulation of a three-dimensional, fully turbulent, incompressible and viscous flow on high performance architectures. In particular, the use of CPU/GPU hybrid parallel paradigms has allowed the design and implementation of new computational models and algorithms, suitable for single- and multi-GPU, and more accurate computational domain has been considered, both in space and time, in order to produce a large amount of simulation data related to the problem of turbulent flow in a plane channel at $\mathrm{Re}_\tau = 200, 400, 600$.

The advanced methodology proposed is based on three main tasks: the DNS as a tool of research for obtaining numerical databases of the fluid flow field with adequate accuracy in space and time; the high performance computing techniques, based on the most advanced parallel architectures (such as hybrid CPU/GPU systems) for developing a high performance Navier-Stokes solver, able to simulate the turbulent flow in a plane channel at high Reynolds numbers; the statistical tools

and the analysis of flow structures for performing modeling and controlling turbulence.

The DNS databases have allowed to analyze in detail turbulence statistics for verifying the adequacy of the computational domain and energetic budgets for understanding turbulence dynamics in the near wall regions. The analysis revealed that, with respect to the normal stresses, the wall is characterized by regions with an intense energy production activity and regions characterized by intense energy dissipation activity along the streamwise direction; furthermore, the velocity pressure-gradient plays a very important role considering $\mathrm{Re}_\tau \geq 400$, since it determines a loss of energy due to the increased energy dissipation. About the Reynolds stress budget related to the Reynolds shear stress, the energy production diminishes as the shear stress diminishes, too, causing an increase in dissipation in the inner region. For the first time, the analysis based on DNS results has been extended to the dissipation and the anisotropy rate budgets, giving an important contribute in turbulence modeling. About dissipation rate budget, it was possible to note the strongly non-isotropic nature of the dissipation process at high Reynolds numbers, that caused the loss of energy near the walls. About the anisotropy rate budget, for the relevant non-zero stresses, it described the highly anisotropic behavior of fluctuations in near-wall region, while considering the central region of the channel the behavior of fluctuations is essentially isotropic.

In order to investigate deeply the role of Reynolds stresses in production, transport mean momentum, dissipation of turbulent kinetic energy of inhomogeneous flow, the organized motion of wall turbulence has been analyzed by extracting coherent structures from the fluid flow field, considering DNS numerical data. The coherent structures have been qualitatively analyzed by using a scientific visualization approach, based on plotting the isosurfaces of the turbulent swirling strength, that describes the interaction between hairpins and events, observed in data time series. Flow structures educed from the fluctuating portion of the velocity field at $\mathrm{Re}_\tau = 200$ confirmed the classical theory about vortical structures, allowing to show the morphological evolution of a single hairpin, that provided a means of producing turbulent kinetic energy, and its interaction with events, such as ejections and sweeps, in order to describe its birth, growing and disruption close to the walls.

The flow structures analysis at $\mathrm{Re}_\tau = 400$ allowed to describe the mechanism of hairpin packets generation: they populate a significant fraction of the boundary layer and occur in the streamwise direction with increasing size along the downstream direction and characterized by small dispersion in their propagation. Thanks to high accurate numerical results, the complete view of the flow structures at $\mathrm{Re}_\tau = 600$ is shown for the first time, revealing how ejection and sweep events strongly contribute to the mean Reynolds stress with the increase of the Reynolds number. The analysis of the time series revealed the intense activity of flow structures close to the boundary layer, playing a crucial role in determining skin friction and on dynamics for producing and dissipating turbulent kinetic energy.

Future works will be addressed in order to integrate the Navier-Stokes equations at higher Reynolds numbers to increase significantly the knowledge about wall-bounded turbulence. It may be done by re-thinking and implementing a new parallel implementation on multi-node multi-GPU, starting from the algorithms designed for the present thesis. Furthermore, since coherent structures actually represent a challenging task for the physical description of turbulence phenomena, an interesting application may be related to the use of the proper orthogonal decomposition (POD) technique for flow structures eduction, in order to analyze from an energetic point of view their evolution and eduction from the fluctuating portion of the velocity field.

Appendix A

Equations

A.1 Mass conservation equation

In incompressible-fluid flows, the conservation of mass is expressed by the continuity equation:

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{A.1}$$

In Eq. (A.1) no time dependence appears, so that the continuity equation exhibits the same form in both steady and unsteady flows. Also in switching from a dimensional to a non-dimensional formulation – besides the fact that non-dimensional variables are involved – Eq. (A.1) has the same form. No dimensionless groups are involved, meaning that continuity has a kinematic character, not being influenced by any flow parameter.

A.2 System of Navier-Stokes equations

The fluid-flow momentum equation can be written as:

$$\frac{Du}{Dt} = \frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \frac{\partial \sigma_{ij}}{\partial x_j} \tag{A.2}$$

where σ_{ij} is the viscous-stress tensor (Newtonian fluid):

$$\sigma_{ij} = 2\nu S_{ij} \tag{A.3}$$

and S_{ij} is the strain-rate tensor (in constant-density fluids $S_{ii} = 0$):

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{A.4}$$

so that:

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + 2\nu \frac{\partial S_{ij}}{\partial x_j} \tag{A.5}$$

and being:

$$\frac{\partial^2 u_i}{\partial x_j \partial x_j} = \frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) = 2\frac{\partial S_{ij}}{\partial x_j} \tag{A.6}$$

The system of the Navier-Stokes equations (momentum and continuity equations) can be written as:

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 u_i}{\partial x_j \partial x_j} \tag{A.7}$$

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{A.8}$$

where the convective term of Eq. (A.7) in expressed in non-conservative form and body forces do not appear explicitly.

A widely-used non-dimensional form of Eq. (A.7) and (A.8) is:

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{1}{\mathrm{Re}_\tau} \frac{\partial^2 u_i}{\partial x_j \partial x_j} \tag{A.9}$$

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{A.10}$$

where the dependent variables are now u_i / u_τ and $p / \rho u_\tau^2$ $\mathrm{Re}_\tau = \rho u_\tau h / \mu = u_\tau h / \nu$, and for simplicity their symbols have not been altered in switching from the dimensional to the dimensionless formalism.

Equation (A.7) can be written in symbolic notation as:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u} \tag{A.8}$$

Making use of the vector identities:

$$\nabla \cdot (\mathbf{uu}) = (\mathbf{u} \cdot \nabla)\mathbf{u} + \mathbf{u}(\nabla \cdot \mathbf{u}) = (\mathbf{u} \cdot \nabla)\mathbf{u} \tag{A.9}$$

$$(\mathbf{u} \cdot \nabla)\mathbf{u} = \frac{1}{2} \nabla (\mathbf{u} \cdot \mathbf{u}) - \mathbf{u} \times (\nabla \times \mathbf{u}) = \frac{1}{2} \nabla (\mathbf{u} \cdot \mathbf{u}) - \mathbf{u} \times \boldsymbol{\omega} \tag{A.10}$$

$$(\mathbf{u} \cdot \nabla)\mathbf{u} + \frac{1}{2}\mathbf{u}(\nabla \cdot \mathbf{u}) = (\mathbf{u} \cdot \nabla)\mathbf{u} = \frac{1}{2}\left[\nabla \cdot (\mathbf{uu}) + (\mathbf{u} \cdot \nabla)\mathbf{u}\right] \tag{A.11}$$

where:

$$\boldsymbol{\omega} = \nabla \times \mathbf{u} \tag{A.12}$$

$$\omega_k = \varepsilon_{ijk} \frac{\partial u_i}{\partial x_j} \tag{A.13}$$

is the vorticity, Eq. (A.8) can be written, respectively, in conservative, rotational and skew-symmetric form:

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} \tag{A.14}$$

$$\frac{\partial \mathbf{u}}{\partial t} - \mathbf{u} \times \boldsymbol{\omega} = -\nabla\left(\frac{p}{\rho} + \frac{1}{2}(\mathbf{u} \cdot \mathbf{u})\right) + \nu \nabla^2 \mathbf{u} \tag{A.15}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{1}{2}\left[\nabla \cdot (\mathbf{u}\mathbf{u}) + (\mathbf{u} \cdot \nabla)\mathbf{u}\right] = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} \tag{A.16}$$

The momentum Eq. (A.8) can also be written by considering different forms of the diffusive term.

Making use of the vector identities:

$$\nabla \cdot (\nabla \mathbf{u}) = \nabla^2 \mathbf{u} = \nabla(\nabla \cdot \mathbf{u}) - \nabla \times (\nabla \times \mathbf{u}) = -\nabla \times (\nabla \times \mathbf{u}) = -\nabla \times \boldsymbol{\omega} \tag{A.17}$$

$$\nabla^2 \mathbf{u} + \nabla(\nabla \cdot \mathbf{u}) = \nabla \cdot \left[(\nabla \mathbf{u}) + (\nabla \mathbf{u})^T\right] \tag{A.18}$$

one obtains (^{T} is transpose):

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla \cdot (\nabla \mathbf{u}) \tag{A.19}$$

$$\frac{\partial \mathbf{u}}{\partial t}+(\mathbf{u}\cdot\nabla)\mathbf{u}=-\frac{1}{\rho}\nabla p-\nu\nabla\times(\nabla\times\mathbf{u}) \tag{A.20}$$

$$\frac{\partial \mathbf{u}}{\partial t}+(\mathbf{u}\cdot\nabla)\mathbf{u}=-\frac{1}{\rho}\nabla p-\nu\nabla\times\boldsymbol{\omega} \tag{A.21}$$

$$\partial_t\mathbf{u}+(\mathbf{u}\cdot\nabla)\mathbf{u}=-\frac{1}{\rho}\nabla p+\nu\nabla\cdot\left[(\nabla\mathbf{u})+(\nabla\mathbf{u})^T\right] \tag{A.22}$$

that are forms of Eq. (A.8) in which the viscous term is expressed, respectively, in conservative, divergence-curl, curl and stress-divergence form.

A.3 The Reynolds-Averaged Navier-Stokes equations

According to Reynolds decomposition, the dependent variables in Eqs. (A.7) and (A.8) can be decomposed into mean and fluctuating parts, and averaged, so that the Reynolds-Averaged Navier-Stokes (RANS) equations are obtained:

$$\frac{D\overline{u}}{Dt}=\frac{\partial\overline{u}_i}{\partial t}+\overline{u}_j\frac{\partial\overline{u}_i}{\partial x_j}=-\frac{1}{\rho}\frac{\partial\overline{p}}{\partial x_i}+\frac{\partial\overline{\sigma}_{ij}}{\partial x_j}-\frac{\partial\tau_{ij}}{\partial x_j} \tag{A.23}$$

$$\frac{\partial\overline{u}_i}{\partial x_i}=0 \tag{A.24}$$

where $\overline{\sigma}_{ij}$ is the mean viscous-stress tensor:

$$\overline{\sigma}_{ij}=2\nu\overline{S}_{ij} \tag{A.25}$$

and $\overline{S}_{ij}$ is the mean strain-rate tensor ($\overline{S}_{ii}=0$):

$$\overline{S}_{ij} = \frac{1}{2}\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i} \right) \tag{A.26}$$

Substituting, one has:

$$\frac{\partial \overline{u}_i}{\partial t} + \overline{u}_j \frac{\partial \overline{u}_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + \nu \frac{\partial^2 \overline{u}_i}{\partial x_j \partial x_j} - \frac{\partial \tau_{ij}}{\partial x_j} \tag{A.27}$$

$$\frac{\partial \overline{u}_i}{\partial x_i} = 0 \tag{A.28}$$

where:

$$\rho \tau_{ij} = \rho \overline{u_i' u_j'} \tag{A.29}$$

is the Reynolds-stress tensor.

A.4 Transport equation of the mean-field kinetic energy

The equation governing the kinetic energy of the mean-field $\overline{u}_i \overline{u}_i / 2$ is:

$$\begin{aligned} &\frac{D}{Dt}\left(\frac{1}{2}\overline{u}_i \overline{u}_i \right) = \frac{\partial}{\partial t}\left(\frac{1}{2}\overline{u}_i \overline{u}_i \right) + \overline{u}_j \frac{\partial}{\partial x_j}\left(\frac{1}{2}\overline{u}_i \overline{u}_i \right) = \\ &= \frac{\partial}{\partial x_j}\left(-\frac{\overline{p}}{\rho}\overline{u}_j + 2\nu \overline{u}_i \overline{S}_{ij} - \overline{u_i' u_j'}\overline{u}_i \underbrace{+\overline{u_i' u_j'}\overline{S}_{ij}}_{P_K} - 2\nu \overline{S}_{ij}\overline{S}_{ij} \right) \end{aligned} \tag{A.30}$$

where:

- the first three terms on the rhs of Eq. (A.30) represent, respectively, the net flux of work associated with the mean pressure, mean viscous stress and turbulent stress;

- the last two terms on the rhs of Eq. (A.30) represent, respectively, the dissipation related to turbulent stress (the production of turbulent kinetic energy, P_K) and mean viscous stress.

A.5 Transport equation of the mean turbulent kinetic energy

The equation governing the mean kinetic energy of the turbulent field $\overline{u_i' u_i'}/2$ is [152]:

$$\frac{D}{Dt}\left(\frac{1}{2}\overline{u_i' u_i'}\right) = \frac{\partial}{\partial t}\left(\frac{1}{2}\overline{u_i' u_i'}\right) + \overline{u}_j \frac{\partial}{\partial x_j}\left(\frac{1}{2}\overline{u_i' u_i'}\right) =$$
$$= \frac{\partial}{\partial x_j}\left(-\overline{\frac{p'}{\rho}u_j'} + 2\nu\overline{u_i' S_{ij}'} - \frac{1}{2}\overline{u_i' u_i' u_j'} \underbrace{-\overline{u_i' u_j'}\,\overline{S}_{ij}}_{P_K} \underbrace{-2\nu\overline{S_{ij}' S_{ij}'}}_{\varepsilon}\right) \tag{A.31}$$

where:

- the first three terms on the rhs of Eq. (A.31) denote the net flux of work associated, respectively, with fluctuating pressure, fluctuating viscous stress, and turbulent stress;
- the subsequent turbulence-production term P_K reflects the exchange of kinetic energy between mean flow and turbulence (it exhibits an opposite sign with respect to Eq. (A.30) due to the fact that energy exchange involves in general a loss to mean flow and a gain to turbulence);
- the last term ε on the rhs of Eq. (A.31) is the viscous dissipation of turbulent kinetic energy. Unlike the dissipation related to the mean viscous stress in Eq. (A.30), this term is essential to the dynamics of turbulence.

The turbulent kinetic energy transport Eq. (A.31) is obtained from a contraction of indexes in the Reynolds-stress transport equation.

Equation (A.31), non-dimensionalized by u_τ^4/ν, can be written in the form $K = \tau_{ii}/2 = \overline{u_i' u_i'}/2$ as follows:

$$\frac{DK}{Dt}=\frac{\partial K}{\partial t}+\overline{u}_k\frac{\partial K}{\partial x_k}=P_K+\Pi_K-\varepsilon+T_K+D_K=$$
$$=-\tau_{ik}\frac{\partial \overline{u}_i}{\partial x_k}-\overline{u_i'\frac{\partial p'}{\partial x_i}}-\overline{\frac{\partial u_i'}{\partial x_k}\frac{\partial u_i'}{\partial x_k}}-\frac{1}{2}\frac{\partial}{\partial x_k}\left(\overline{u_i'u_i'u_k'}\right)+\frac{\partial^2 K}{\partial x_k\partial x_k} \tag{A.32}$$

representing the balance between the local rate of change and the convective transport of turbulent kinetic energy on the lhs of Eq. (A.32) and the following terms on the rhs:

- the turbulent kinetic energy production term:

$$P_K=-\tau_{ik}\frac{\partial \overline{u}_i}{\partial x_k} \tag{A.33}$$

- the velocity pressure-gradient term:

$$\Pi_K=-\overline{u_i'\frac{\partial p'}{\partial x_i}} \tag{A.34}$$

- the dissipation rate term (in the form of the isotropic dissipation):

$$\varepsilon=\overline{\frac{\partial u_i'}{\partial x_k}\frac{\partial u_i'}{\partial x_k}} \tag{A.35}$$

- the turbulent transport rate term:

$$T_K=-\frac{1}{2}\frac{\partial}{\partial x_k}\left(\overline{u_i'u_i'u_k'}\right) \tag{A.36}$$

- the viscous diffusion rate term:

$$D_K=\frac{\partial^2 K}{\partial x_k\partial x_k} \tag{A.37}$$

All the terms described above can be calculated by averaging the diagonal terms of the Reynolds-stress tensor Eq. (A.29).

A.6 Reynolds-stress transport equation

The Reynolds-stress transport equation $\tau_{ij} = \overline{u_i' u_j'}$ is obtained from the second moment:

$$\overline{u_i' N u_j' + u_j' N u_i'} = 0 \tag{A.38}$$

The Reynolds-stress transport equation, non-dimensionalized by u_τ^4 / ν, can be cast as [152]:

$$\begin{aligned} &\frac{D\tau_{ij}}{Dt} = \frac{\partial \tau_{ij}}{\partial t} + \overline{u}_k \frac{\partial \tau_{ij}}{\partial x_k} = P_{ij} + \Pi_{ij} - \varepsilon_{ij} + T_{ij} + D_{ij} = \\ &= -\left(\tau_{ik} \frac{\partial \overline{u}_j}{\partial x_k} + \tau_{jk} \frac{\partial \overline{u}_i}{\partial x_k} \right) - \left(\overline{u_i' \frac{\partial p'}{\partial x_j} + u_j' \frac{\partial p'}{\partial x_i}} \right) - 2\overline{\frac{\partial u_i'}{\partial x_k} \frac{\partial u_j'}{\partial x_k}} - \frac{\partial}{\partial x_k} \left(\overline{u_i' u_j' u_k'} \right) + \frac{\partial^2 \tau_{ij}}{\partial x_k \partial x_k} \end{aligned} \tag{A.39}$$

representing the balance between the local rate of change and the convective transport of Reynolds-stress on the lhs of Eq. (A.39) and the following terms on the rhs:

- the Reynolds-stress production rate term:

$$P_{ij} = -\left(\tau_{ik} \frac{\partial \overline{u}_j}{\partial x_k} + \tau_{jk} \frac{\partial \overline{u}_i}{\partial x_k} \right) \tag{A.40}$$

- the velocity pressure gradient terms:

$$\Pi_{ij} = -\left(\overline{u_i' \frac{\partial p'}{\partial x_j} + u_j' \frac{\partial p'}{\partial x_i}} \right) \tag{A.41}$$

It can be split into the pressure-strain $\Pi_{s,ij}$ and the pressure-diffusion $\Pi_{d,ij}$ terms as follows:

$$\Pi_{ij} = \Pi_{s,ij} + \Pi_{d,ij} = \Pi_{ij} = \overline{p'\left(\frac{\partial u_j'}{\partial x_i} + \frac{\partial u_i'}{\partial x_j}\right)} - \left(\frac{\partial}{\partial x_i}\overline{u_j' p'} + \frac{\partial}{\partial x_j}\overline{u_i' p'}\right) \quad \text{(A.42)}$$

It is well known that the pressure-strain term plays an important role on the energy redistribution;

- the dissipation rate term:

$$\varepsilon_{ij} = 2\overline{\frac{\partial u_i'}{\partial x_k}\frac{\partial u_j'}{\partial x_k}} \quad \text{(A.43)}$$

- the turbulent transport rate term:

$$T_{ij} = -\frac{\partial}{\partial x_k}\left(\overline{u_i' u_j' u_k'}\right) \quad \text{(A.44)}$$

- the viscous diffusion rate term:

$$D_{ij} = \frac{\partial^2 \tau_{ij}}{\partial x_k \partial x_k} \quad \text{(A.45)}$$

Repeated indices imply summation over $1,2,3$ and the indices $(1,2,3)$ are used to denote the streamwise x^+, the normal to the wall y^+ and the spanwise z^+ directions, respectively. In the above equation, p is a non-dimensional kinematic pressure. In a fully developed channel, the flow is homogeneous in the streamwise and the spanwise directions. The relevant non-zero stresses in this case are $\overline{u_1' u_1'}, \overline{u_1' u_2'}, \overline{u_2' u_2'}, \overline{u_3' u_3'}$. The main expressions of the Reynolds stress budgets are here reported.

- The Reynolds-stress production rate:

$$P_{11} = -\left(\overline{u_1' u_k'} \frac{\partial U_1}{\partial x_k} + \overline{u_1' u_k'} \frac{\partial U_1}{\partial x_k} \right) = -2\overline{u_1' u_2'} \frac{\partial U_1}{\partial x_2} = -2\left(\overline{u' v'} \frac{\partial U}{\partial y} \right) \tag{A.46}$$

$$P_{12} = -\left(\overline{u_1' u_k'} \frac{\partial U_2}{\partial x_k} + \overline{u_2' u_k'} \frac{\partial U_1}{\partial x_k} \right) = -\left(\overline{v' v'} \frac{\partial U}{\partial y} \right) \tag{A.47}$$

- The velocity pressure gradient term:

$$\Pi_{s,11} = \overline{2p' \frac{\partial u'}{\partial x}},\ \Pi_{d,11} = -2\left(\frac{\partial}{\partial x_i} \overline{u' p'} \right) \tag{A.48}$$

$$\Pi_{s,12} = \overline{p' \frac{\partial u'}{\partial y} + p' \frac{\partial v'}{\partial x}},\ \Pi_{d,12} = -\left[\frac{\partial}{\partial x} \overline{v' p'} + \frac{\partial}{\partial x} \overline{u' p'} \right] \tag{A.49}$$

$$\Pi_{s,22} = \overline{2p' \frac{\partial v'}{\partial y}},\ \Pi_{d,22} = -2\left(\frac{\partial}{\partial x_i} \overline{v' p'} \right) \tag{A.50}$$

$$\Pi_{s,33} = \overline{2p' \frac{\partial w'}{\partial z}},\ \Pi_{d,33} = -2\left(\frac{\partial}{\partial x_i} \overline{w' p'} \right) \tag{A.51}$$

- The dissipation rate term:

$$\varepsilon_{11} = 2\overline{\frac{\partial u_1'}{\partial x_k} \frac{\partial u_1'}{\partial x_k}} = 2\left[\overline{\left(\frac{\partial u'}{\partial x} \right)^2 + \left(\frac{\partial v'}{\partial y} \right)^2 + \left(\frac{\partial w'}{\partial z} \right)^2} \right] \tag{A.52}$$

$$\varepsilon_{12} = 2\overline{\frac{\partial u_1'}{\partial x_k} \frac{\partial u_2'}{\partial x_k}} = 2\left[\overline{\left(\frac{\partial u'}{\partial x} \frac{\partial v'}{\partial x} \right) + \left(\frac{\partial u'}{\partial y} \frac{\partial v'}{\partial y} \right) + \left(\frac{\partial u'}{\partial z} \frac{\partial v'}{\partial z} \right)} \right] \tag{A.53}$$

$$\varepsilon_{22} = 2\overline{\frac{\partial u_2'}{\partial x_k}\frac{\partial u_2'}{\partial x_k}} = 2\left[\overline{\left(\frac{\partial v'}{\partial x}\right)^2 + \left(\frac{\partial v'}{\partial y}\right)^2 + \left(\frac{\partial v'}{\partial z}\right)^2}\right] \tag{A.54}$$

$$\varepsilon_{33} = 2\overline{\frac{\partial u_3'}{\partial x_k}\frac{\partial u_3'}{\partial x_k}} = 2\left[\overline{\left(\frac{\partial w'}{\partial x}\right)^2 + \left(\frac{\partial w'}{\partial y}\right)^2 + \left(\frac{\partial w'}{\partial z}\right)^2}\right] \tag{A.55}$$

- The turbulent transport rate term:

$$T_{11} = -\frac{\partial}{\partial x_k}\left(\overline{u_1'u_1'u_k'}\right) = -\frac{\partial}{\partial y}\left(\overline{u'u'v'}\right) \tag{A.57}$$

$$T_{12} = -\frac{\partial}{\partial x_k}\left(\overline{u_1'u_2'u_k'}\right) = -\frac{\partial}{\partial y}\left(\overline{u'v'v'}\right) \tag{A.58}$$

$$T_{22} = -\frac{\partial}{\partial x_k}\left(\overline{u_2'u_2'u_k'}\right) = -\frac{\partial}{\partial y}\left(\overline{v'v'v'}\right) \tag{A.59}$$

$$T_{33} = -\frac{\partial}{\partial x_k}\left(\overline{u_3'u_3'u_k'}\right) = -\frac{\partial}{\partial y}\left(\overline{w'w'v'}\right) \tag{A.60}$$

- The viscous diffusion rate term:

$$D_{11} = \frac{\partial^2\left(\overline{u_1'u_1'}\right)}{\partial x_k \partial x_k} = \frac{\partial^2\left(\overline{u'u'}\right)}{\partial y^2} \tag{A.61}$$

$$D_{12} = \frac{\partial^2\left(\overline{u_1'u_2'}\right)}{\partial x_k \partial x_k} = \frac{\partial^2\left(\overline{u'v'}\right)}{\partial y^2} \tag{A.62}$$

$$D_{22} = \frac{\partial^2\left(\overline{u_2'u_2'}\right)}{\partial x_k \partial x_k} = \frac{\partial^2\left(\overline{v'v'}\right)}{\partial y^2} \tag{A.63}$$

$$D_{33} = \frac{\partial^2 \left(\overline{u_3' u_3'} \right)}{\partial x_k \partial x_k} = \frac{\partial^2 \left(\overline{w' w'} \right)}{\partial y^2} \tag{A.64}$$

A.7 The Reynolds-stress anisotropy transport equation

The Reynolds-stress tensor can be written as an isotropic part and a deviatoric part, the latter related with the Reynolds-stress anisotropy tensor b_{ij} [153]:

$$\tau_{ij} = \tau_{ij}^I + \tau_{ij}^D \tag{A.65}$$

$$\tau_{ij} = \frac{2}{3} K \delta_{ij} + 2 K b_{ij} \tag{A.66}$$

so that:

$$b_{ij} = \frac{\tau_{ij}}{2K} - \frac{\delta_{ij}}{3} \tag{A.67}$$

The transport equation of b_{ij} can be written as $\varepsilon = \varepsilon_{ii}/2$:

$$\begin{aligned} &\frac{Db_{ij}}{Dt} = \frac{\partial b_{ij}}{\partial t} = \frac{1}{2K} \left(\frac{D\tau_{ij}}{Dt} - \frac{\tau_{ij}}{K} \frac{DK}{Dt} \right) = \\ &= \frac{1}{2K} \left[\left(P_{ij} - \frac{\tau_{ij}}{K} P_K \right) + \Pi_{ij} - \left(\varepsilon_{ij} - \frac{\tau_{ij}}{K} \varepsilon_K \right) + \left(T_{ij} - \frac{\tau_{ij}}{T_K} \right) \right] \end{aligned} \tag{A.68}$$

representing the balance between the local rate of change and the convective transport of turbulent anisotropy of Reynolds-stress on the lhs of Eq. (A.61) and the following terms on the rhs:

- the Reynolds-stress anisotropic production term:

$$P_{ij}^b = \left(P_{ij} - \frac{\tau_{ij}}{K} P_K \right) = -\tau_{ik} \frac{\partial \overline{u}_j}{\partial x_k} + \left(\frac{\tau_{ij} \tau_{ik}}{K} - \tau_{jk} \right) \tag{A.69}$$

- the anisotropy velocity pressure gradient term:

$$\Pi_{ij}^{b} = -\left(\overline{u_i' \frac{\partial p'}{\partial x_j} + u_j' \frac{\partial p'}{\partial x_i}} \right) \tag{A.70}$$

- the anisotropy dissipation rate:

$$\varepsilon_{ij}^{b} = \varepsilon_{ij} - \frac{\tau_{ij}}{K}\varepsilon_K = 2\overline{\frac{\partial u_i'}{\partial x_k}\frac{\partial u_j'}{\partial x_k}} - \frac{\tau_{ij}}{K}\overline{\frac{\partial u_i'}{\partial x_k}\frac{\partial u_i'}{\partial x_k}} \tag{A.71}$$

- the anisotropy transport rate:

$$T_{ij}^{b} = T_{ij} - \frac{\tau_{ij}}{K}T_K = -\frac{\partial}{\partial x_k}\left(\overline{u_i' u_j' u_k'}\right) + \frac{\tau_{ij}}{2K}\frac{\partial}{\partial x_k}\left(\overline{u_i' u_i' u_k'}\right) \tag{A.72}$$

Different forms of Eq. (A.68) can be obtained by decomposing the mean-velocity gradient tensor $\partial \overline{u}_i / \partial x_j$ into a symmetric part $\overline{S}_{ij}$ (the mean strain-rate tensor) and an anti-symmetric part $\overline{W}_{ij}$ (the mean rotation-rate tensor):

$$\frac{\partial \overline{u}_i}{\partial x_j} = \overline{S}_{ij} + \overline{W}_{ij} \tag{A.73}$$

where:

$$\overline{S}_{ij} = \frac{1}{2}\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i} \right) \tag{A.74}$$

$$\overline{W}_{ij} = \frac{1}{2}\left(\frac{\partial \overline{u}_i}{\partial x_j} - \frac{\partial \overline{u}_j}{\partial x_i} \right) \tag{A.75}$$

A.8 Turbulent dissipation-rate transport equation

The turbulent scalar dissipation-rate transport equation $\varepsilon = \varepsilon_{ii}/2$ is obtained from the moment equation as follows:

$$2\nu\overline{\frac{\partial u_i'}{\partial x_j}\frac{\partial}{\partial x_j}\left(Nu_i'\right)} = 0 \tag{A.76}$$

Equation (A.76), non-dimensionalized by u_τ^4/ν, can be cast as [152]:

$$\frac{D\varepsilon}{Dt} = \frac{\partial\varepsilon}{\partial t} + \overline{u}_i\frac{\partial\varepsilon}{\partial x_i} = P_\varepsilon^1 + P_\varepsilon^2 + P_\varepsilon^3 + P_\varepsilon^4 + \Pi_\varepsilon + T_\varepsilon + D_\varepsilon - Y \tag{A.78}$$

Explicating Eq. (A.78), one obtains:

$$\begin{aligned}
&\frac{\partial\varepsilon}{\partial t} + \overline{u}_i\frac{\partial\varepsilon}{\partial x_i} = \\
&= -2\overline{\frac{\partial u_i'}{\partial x_j}\frac{\partial u_k'}{\partial x_j}}\frac{\partial\overline{u}_i}{\partial x_k} - 2\overline{\frac{\partial u_i'}{\partial x_k}\frac{\partial u_i'}{\partial x_m}}\frac{\partial\overline{u}_k}{\partial x_m} - 2\overline{u_k'\frac{\partial u_i'}{\partial x_m}}\frac{\partial^2\overline{u}_i}{\partial x_k\partial x_m} - 2\overline{\frac{\partial u_i'}{\partial x_k}\frac{\partial u_i'}{\partial x_m}\frac{\partial u_k'}{\partial x_m}} + \\
&-\frac{2}{\rho}\frac{\partial}{\partial x_k}\left(\overline{\frac{\partial p'}{\partial x_m}\frac{\partial u_k'}{\partial x_m}}\right) - \frac{\partial}{\partial x_k}\left(\overline{u_k'\frac{\partial u_i'}{\partial x_m}\frac{\partial u_i'}{\partial x_m}}\right) + \\
&-2\overline{\frac{\partial^2 u_i'}{\partial x_k\partial x_m}\frac{\partial^2 u_i'}{\partial x_k\partial x_m}} + \frac{\partial^2\varepsilon}{\partial x_k\partial x_k}
\end{aligned} \tag{A.79}$$

Equation (A.79) represents the balance between the rate of change and the convective transport of turbulent dissipation rate on the lhs of Eq. (A.78) and the following terms on the rhs:

- the mixed production rate of dissipation:

$$P_\varepsilon^1 = -\overline{\frac{\partial u_i'}{\partial x_j}\frac{\partial u_k'}{\partial x_j}}\left(\frac{\partial\overline{u}_i}{\partial x_k} + \frac{\partial\overline{u}_k}{\partial x_i}\right) \tag{A.80}$$

- the production rate of the dissipation by mean-velocity gradient:

$$P_\varepsilon^2 = -\overline{\frac{\partial u_i'}{\partial x_k}\frac{\partial u_i'}{\partial x_m}}\left(\frac{\partial \overline{u}_k}{\partial x_m}+\frac{\partial \overline{u}_m}{\partial x_k}\right) \tag{A.81}$$

- the gradient production rate:

$$P_\varepsilon^3 = -2\overline{u_k'\frac{\partial u_i'}{\partial x_m}}\frac{\partial^2 \overline{u}_i}{\partial x_k \partial x_m} \tag{A.82}$$

- the turbulent production rate:

$$P_\varepsilon^4 = -2\overline{\frac{\partial u_i'}{\partial x_k}\frac{\partial u_i'}{\partial x_m}\frac{\partial u_k'}{\partial x_m}} \tag{A.83}$$

- the pressure transport rate:

$$\Pi_\varepsilon = -\frac{2}{\rho}\frac{\partial}{\partial x_k}\left(\overline{\frac{\partial p'}{\partial x_m}\frac{\partial u_k'}{\partial x_m}}\right) \tag{A.84}$$

- the turbulent transport rate:

$$\begin{aligned} T_\varepsilon = -\Bigg[\nu'\Bigg(&\overline{\frac{\partial u'}{\partial x}\frac{\partial u'}{\partial x}}+\overline{\frac{\partial v'}{\partial x}\frac{\partial v'}{\partial x}}+\overline{\frac{\partial w'}{\partial x}\frac{\partial w'}{\partial x}}+\\ &+\overline{\frac{\partial u'}{\partial y}\frac{\partial u'}{\partial y}}+\overline{\frac{\partial v'}{\partial y}\frac{\partial v'}{\partial y}}+\overline{\frac{\partial w'}{\partial y}\frac{\partial w'}{\partial y}}+\\ &\overline{\frac{\partial u'}{\partial z}\frac{\partial u'}{\partial z}}+\overline{\frac{\partial v'}{\partial z}\frac{\partial v'}{\partial z}}+\overline{\frac{\partial w'}{\partial z}\frac{\partial w'}{\partial z}}\Bigg)\Bigg] \end{aligned} \tag{A.85}$$

- the viscous diffusion rate:

$$D_\varepsilon = \frac{\partial^2 \varepsilon}{\partial x_k \partial x_k} \tag{A.86}$$

- the turbulent dissipation rate:

$$Y = 2\overline{\frac{\partial^2 u_i'}{\partial x_k \partial x_m}\frac{\partial^2 u_i'}{\partial x_k \partial x_m}} \tag{A.87}$$

The main expressions of the dissipation rate tensor are here reported.

- The mixed production rate of the dissipation:

$$P_\varepsilon^1 = -2\frac{\partial \overline{u}}{\partial y}\left(\overline{\frac{\partial u'}{\partial x}\frac{\partial v'}{\partial x} + \frac{\partial u'}{\partial y}\frac{\partial v'}{\partial y} + \frac{\partial u'}{\partial z}\frac{\partial v'}{\partial z}}\right) \tag{A.88}$$

- The production rate of dissipation by mean-velocity gradient:

$$P_\varepsilon^2 = -2\frac{\partial \overline{u}}{\partial y}\left(\overline{\frac{\partial u'}{\partial x}\frac{\partial u'}{\partial y} + \frac{\partial v'}{\partial x}\frac{\partial v'}{\partial y} + \frac{\partial w'}{\partial x}\frac{\partial w'}{\partial y}}\right) \tag{A.89}$$

- The gradient production rate:

$$P_\varepsilon^3 = -2\frac{\partial^2 \overline{u}}{\partial y \partial y}\overline{v'\frac{\partial u'}{\partial y}} \tag{A.90}$$

- The turbulent production rate:

$$P_\varepsilon^4 = -2\left(\overline{\frac{\partial u'}{\partial x}\frac{\partial u'}{\partial x}\frac{\partial u'}{\partial x}} + \overline{\frac{\partial u'}{\partial y}\frac{\partial u'}{\partial x}\frac{\partial u'}{\partial x}} + \overline{\frac{\partial u'}{\partial z}\frac{\partial u'}{\partial x}\frac{\partial w'}{\partial x}} + \right.$$
$$+\overline{\frac{\partial u'}{\partial x}\frac{\partial u'}{\partial y}\frac{\partial u'}{\partial y}} + \overline{\frac{\partial u'}{\partial x}\frac{\partial u'}{\partial z}\frac{\partial u'}{\partial z}} + \overline{\frac{\partial u'}{\partial y}\frac{\partial u'}{\partial y}\frac{\partial v'}{\partial y}} + \overline{\frac{\partial u'}{\partial y}\frac{\partial u'}{\partial z}\frac{\partial v'}{\partial z}} +$$
$$+\overline{\frac{\partial u'}{\partial z}\frac{\partial u'}{\partial y}\frac{\partial w'}{\partial y}} + \overline{\frac{\partial u'}{\partial z}\frac{\partial u'}{\partial z}\frac{\partial w'}{\partial z}} + \overline{\frac{\partial v'}{\partial x}\frac{\partial v'}{\partial x}\frac{\partial u'}{\partial x}} + \overline{\frac{\partial v'}{\partial y}\frac{\partial v'}{\partial x}\frac{\partial v'}{\partial x}} +$$
$$+\overline{\frac{\partial v'}{\partial z}\frac{\partial v'}{\partial x}\frac{\partial w'}{\partial x}} + \overline{\frac{\partial v'}{\partial x}\frac{\partial v'}{\partial y}\frac{\partial u'}{\partial y}} + \overline{\frac{\partial v'}{\partial x}\frac{\partial v'}{\partial z}\frac{\partial u'}{\partial z}} + \overline{\frac{\partial v'}{\partial y}\frac{\partial v'}{\partial y}\frac{\partial v'}{\partial y}} + \quad \text{(A.91)}$$
$$+\overline{\frac{\partial v'}{\partial y}\frac{\partial v'}{\partial z}\frac{\partial v'}{\partial z}} + \overline{\frac{\partial v'}{\partial z}\frac{\partial v'}{\partial y}\frac{\partial w'}{\partial y}} + \overline{\frac{\partial v'}{\partial z}\frac{\partial v'}{\partial z}\frac{\partial w'}{\partial z}} + \overline{\frac{\partial w'}{\partial x}\frac{\partial w'}{\partial x}\frac{\partial u'}{\partial x}} +$$
$$+\overline{\frac{\partial w'}{\partial y}\frac{\partial w'}{\partial x}\frac{\partial v'}{\partial x}} + \overline{\frac{\partial w'}{\partial z}\frac{\partial w'}{\partial x}\frac{\partial w'}{\partial x}} + \overline{\frac{\partial w'}{\partial x}\frac{\partial w'}{\partial y}\frac{\partial u'}{\partial y}} + \overline{\frac{\partial w'}{\partial x}\frac{\partial w'}{\partial z}\frac{\partial u'}{\partial z}} +$$
$$\left. +\overline{\frac{\partial w'}{\partial y}\frac{\partial w'}{\partial y}\frac{\partial v'}{\partial y}} + \overline{\frac{\partial w'}{\partial y}\frac{\partial w'}{\partial z}\frac{\partial v'}{\partial z}} + \overline{\frac{\partial v'}{\partial z}\frac{\partial v'}{\partial y}\frac{\partial v'}{\partial y}} + \overline{\frac{\partial w'}{\partial z}\frac{\partial w'}{\partial w}\frac{\partial w'}{\partial w}} \right)$$

- The pressure transport rate:

$$\Pi_\varepsilon = -\frac{2}{\rho}\frac{\partial}{\partial y}\left(\overline{\frac{\partial p'}{\partial x}\frac{\partial v'}{\partial x} + \frac{\partial p'}{\partial y}\frac{\partial v'}{\partial y} + \frac{\partial p'}{\partial z}\frac{\partial v'}{\partial z}} \right) \quad \text{(A.92)}$$

- The turbulent transport rate:

$$T_\varepsilon = -\frac{\partial}{\partial y}\left\{ \overline{v'\left[\left(\frac{\partial u'}{\partial x}\right)^2 + \left(\frac{\partial v'}{\partial x}\right)^2 + \left(\frac{\partial w'}{\partial x}\right)^2 + \right.} \right.$$
$$+\overline{\left(\frac{\partial u'}{\partial y}\right)^2 + \left(\frac{\partial v'}{\partial y}\right)^2 + \left(\frac{\partial w'}{\partial y}\right)^2} + \quad \text{(A.93)}$$
$$\left. \left. \overline{\left(\frac{\partial u'}{\partial z}\right)^2 + \left(\frac{\partial v'}{\partial z}\right)^2 + \left(\frac{\partial w'}{\partial z}\right)^2} \right] \right\}$$

- the viscous diffusion rate:

$$D_\varepsilon = \nabla^2 \varepsilon = \frac{\partial^2 \varepsilon}{\partial y^2} \quad \text{(A.94)}$$

- The turbulent dissipation rate:

$$
\begin{aligned}
Y = \Bigg[& \overline{\frac{\partial^2 u'}{\partial x \partial x}\frac{\partial^2 u'}{\partial x \partial x}} + \overline{\frac{\partial^2 u'}{\partial y \partial x}\frac{\partial^2 u'}{\partial y \partial x}} + \overline{\frac{\partial^2 u'}{\partial z \partial x}\frac{\partial^2 u'}{\partial z \partial x}} + \overline{\frac{\partial^2 u'}{\partial x \partial y}\frac{\partial^2 u'}{\partial x \partial y}} + \overline{\frac{\partial^2 u'}{\partial y \partial z}\frac{\partial^2 u'}{\partial y \partial z}} + \\
& + \overline{\frac{\partial^2 u'}{\partial y \partial y}\frac{\partial^2 u'}{\partial y \partial y}} + \overline{\frac{\partial^2 u'}{\partial y \partial z}\frac{\partial^2 u'}{\partial y \partial z}} + \overline{\frac{\partial^2 u'}{\partial z \partial y}\frac{\partial^2 u'}{\partial z \partial y}} + \overline{\frac{\partial^2 u'}{\partial z \partial z}\frac{\partial^2 u'}{\partial z \partial z}} + \\
& + \overline{\frac{\partial^2 v'}{\partial x \partial x}\frac{\partial^2 v'}{\partial x \partial x}} + \overline{\frac{\partial^2 v'}{\partial y \partial x}\frac{\partial^2 v'}{\partial y \partial x}} + \overline{\frac{\partial^2 v'}{\partial z \partial x}\frac{\partial^2 v'}{\partial z \partial x}} + \overline{\frac{\partial^2 v'}{\partial x \partial y}\frac{\partial^2 v'}{\partial x \partial y}} + \overline{\frac{\partial^2 v'}{\partial y \partial z}\frac{\partial^2 v'}{\partial y \partial z}} + \\
& + \overline{\frac{\partial^2 v'}{\partial y \partial y}\frac{\partial^2 v'}{\partial y \partial y}} + \overline{\frac{\partial^2 v'}{\partial y \partial z}\frac{\partial^2 v'}{\partial y \partial z}} + \overline{\frac{\partial^2 v'}{\partial z \partial y}\frac{\partial^2 v'}{\partial z \partial y}} + \overline{\frac{\partial^2 v'}{\partial z \partial z}\frac{\partial^2 v'}{\partial z \partial z}} + \\
& + \overline{\frac{\partial^2 w'}{\partial x \partial x}\frac{\partial^2 w'}{\partial x \partial x}} + \overline{\frac{\partial^2 w'}{\partial y \partial x}\frac{\partial^2 w'}{\partial y \partial x}} + \overline{\frac{\partial^2 w'}{\partial z \partial x}\frac{\partial^2 w'}{\partial z \partial x}} + \overline{\frac{\partial^2 w'}{\partial x \partial y}\frac{\partial^2 w'}{\partial x \partial y}} + \overline{\frac{\partial^2 w'}{\partial y \partial z}\frac{\partial^2 w'}{\partial y \partial z}} + \\
& + \overline{\frac{\partial^2 w'}{\partial y \partial y}\frac{\partial^2 w'}{\partial y \partial y}} + \overline{\frac{\partial^2 w'}{\partial y \partial z}\frac{\partial^2 w'}{\partial y \partial z}} + \overline{\frac{\partial^2 w'}{\partial z \partial y}\frac{\partial^2 w'}{\partial z \partial y}} + \overline{\frac{\partial^2 w'}{\partial z \partial z}\frac{\partial^2 w'}{\partial z \partial z}} \Bigg]
\end{aligned}
\tag{A.95}
$$

List of References

[1] Kolmogorov, A. N., 1941a, "The local structure of turbulence in an incompressible fluid for very large Reynolds numbers", *Dokl. Akad. Nauk. SSSR*, **30**, pp. 229-303.

[2] Kolmogorov, A. N., 1941b, "Energy dissipation in locally isotropic turbulence, *Dokl. Akad. Nauk. SSSR*, **32**, pp. 19-21.

[3] Saddoughi, S. G., and Veeravalli, S. V., 1994, "Local isotropy in turbulent boundary layers at high Reynolds numbers", *J. Fluid Mech.*, **268**, pp. 333-372.

[4] Lumley, J. L., 1992, "Some comments on turbulence", *Phys. Fluids A*, **4**, pp. 203-211.

[5] Leslie, D. C., and Quarini, G. L., 1979, "The application of turbulence theory to the formulation of subgrid scale modeling procedures", *J. Fluid Mech.*, **91**, pp. 65-.

[6] Mydlarski, L., and Warhaft, Z., 1996, "On the onset of high-Reynolds number grid generated wind tunnel turbulence", *J. Fluid Mech.*, **320**, pp. 331-368.

[7] Corrsin, S., 1962, "Turbulent dissipation fluctuations", *Phys. Fluids*, **5**, pp. 1301-.

[8] Warhaft Z., 2000, "Passive scalars in turbulent flows", *Annu. Rev. Fluid Mech.*, **32**, 203-240.

[9] Kolmogorov, A. N., 1962, "A refinement of the previous hypotheses concerning the local structure of turbulence in viscous incompressible fluid at high Reynolds number", *J. Fluid Mech.*, **13**, pp. 82-85.

[10] Moin, P., and Mahesh, K., 1998, "Direct numerical simulation: a tool in turbulence research", *Annu. Rev. Fluid Mech.*, **30**, pp. 539-578.

[11] Orszag, S. A., and Patterson, G. S., 1972, "Numerical simulation of three-dimensional homogeneous isotropic turbulence", *Phys. Rev. Lett.*, **28**, pp. 76-79.

[12] Rogallo, R. S., 1981, "Numerical experiments in homogeneous turbulence", *NASA TM-81315*.

[13] Rogallo, R. S., and Moin, P., 1984, "Numerical simulation of turbulent flows", *Annu. Rev. Fluid Mech.*, **16**, pp. 99-137.

[14] Kerr, R. M., 1985, "Higher-order derivative correlations and the alignment of small-scale structures in isotropic numerical turbulence", *J. Fluid Mech.*, **153**, pp. 31-58.

[15] Jiménez, J., Wray, A.A., Saffman, P. G., and Rogallo, R.S., 1993, “The structure of intense vorticity in isotropic turbulence”, *J. Fluid Mech.*, **255**, pp. 65-90.

[16] Moser, R. D., and Moin, P., 1987, “The effects of curvature in wall-bounded turbulent flows”, *J. Fluid Mech.*, **175**, pp. 479-510.

[17] Kim, J., Moin, P., and Moser, R., 1987, “Turbulence statistics in fully developed channel flow at low Reynolds number”, *J. Fluid Mech.*, **177**, pp. 133-166.

[18] Spalart, P. R., 1988, “Direct simulation of a turbulent boundary layer up to”, *J. Fluid Mech.*, **187**, pp. 61-98.

[19] Gavrilakis, S., 1992, “Numerical simulation of low-Reynolds-number turbulent flow through a straight square duct”, *J. Fluid Mech.*, **244**, pp. 101-129.

[20] Huser, A., and Biringen, S., 1993, “Direct numerical simulation of turbulent flow in a square duct", *J. Fluid Mech.*, **257**, pp. 65-95.

[21] Eggels, J. G. M., Unger, F., Weiss, M.H., Westerweel, J., Adrian, R.J., Friedrich, R., and Nieuwstadt, F. T. M., 1994, “Fully developed turbulent pipe flow: a comparison between direct numerical simulation and experiment”, *J. Fluid Mech.*, **268**, pp. 175-209.

[22] Schumann, U., 1975, “Linear stability of finite difference equations for three-dimensional flow problems”, *J. Comput. Phys.*, **18**, pp. 465-470.

[23] Lyons, S. L., Hanratty, T. J., and McLaughlin, J. B., 1991, “Large-scale computer simulation of fully developed turbulent channel flow with heath transfer”, *Int. J. Num. Meth. Fluids*, **13**, pp. 999-1028.

[24] Kasagi, N., Tomita, Y., and Kuroda, A., 1992, "Direct numerical simulation of passive scalar field in a turbulent channel flow", *ASME J. Heat Transf.*, **114**, pp. 598-606.

[25] Antonia, R. A., Teitel, M., Kim, J., and Browne, L. W. B., 1992, "Low-Reynolds-number effects in a fully developed turbulent channel flow", *J. Fluid Mech.*, **236**, pp. 579-605.

[26] Rutledge, J., and Sleicher, C.A., 1993, "Direct simulation of turbulent flow and heat transfer in a channel. Part I: smooth walls", *Int. J. Num. Meth. Fluids*, **16**, pp. 1051-1078.

[27] Moser, R. D., Kim, J., and Mansour, N. N., 1999, "Direct numerical simulation of turbulent channel flow up to", *Phys. Fluids*, **11**, pp. 943-945.

[28] Abe, H., Kawamura, H., and Matsuo, Y., 2001, "Direct numerical simulation of a fully developed turbulent channel flow with respect to the Reynolds number dependence", *ASME J. Fluids Eng.*, **123**, pp. 382-393.

[29] Iwamoto, K., Suzuki, Y., and Kasagi, N., 2002, "Reynolds number effect on wall turbulence: toward effective feedback control", *Int. J. Heat Fluid Flow*, **23**, pp. 678-689.

[30] Del Alamo, J. C., and Jiménez, J., 2003, "Spectra of the very large anisotropic scales in turbulent channels", *Phys. Fluids*, **15**, pp. L41-L44.

[31] Del Alamo, J. C., Jiménez, J., Zandonade, P., and Moser, R. D., 2004, "Scaling of the energy spectra of turbulent channels", *J. Fluid Mech.*, **500**, pp. 135-144.

[32] Tanahashi, M., Kang, S. J., Miyamoto, T., Shiokawa, S., and Miyauchi, T., 2004, "Scaling law of the fine scale eddies in turbulent channel flows up to", *Int. J. Heat Fluid Flow*, **25**, pp. 331-340.

[33] Iwamoto, K., Kasagi, N., and Suzuki, Y., 2005, "Direct numerical simulation of turbulent channel flow at $\mathrm{Re}_\tau = 2320$", *Proc. 6° Int. Symp. on Smart Control of Turbulence*, Tokio, pp. 1-7.

[34] Hoyas, S., and Jiménez, J., 2006, "Scaling of the velocity fluctuations in turbulent channels up to", *Phys. Fluids*, **18**, pp. 011702/1-011702-4.

[35] Hu, Z. W., Morfey, C. L., and Sandham, N. D., 2006, "Wall pressure and shear stress spectra from direct simulations of channel flow", *AIAA J.*, **44**, pp. 1541-1549.

[36] Alfonsi, G., and Primavera, L., 2007, "Direct numerical simulation of turbulent channel flow with mixed spectral-finite difference technique", *J. Flow Visual. Image Proc.*, **14**, pp. 225-243.

[37] Kim, J., and Moin, P., 1985, "Application of a fractional-step method to incompressible Navier-Stokes equations", *J. Comput. Phys.*, **59**, pp. 308-323.

[38] Alfonsi, G., Passoni, G., Pancaldo, L., and Zampaglione, D., 1998, "A spectral-finite difference solution of the Navier-Stokes equations in three dimensions", *Int. J. Numer. Meth. Fluids*, **28**, pp. 129-142.

[39] Rempfer, D., 2006, "On boundary conditions for incompressible Navier-Stokes problems", *Appl. Mech. Rev.*, **59**, pp. 107-125.

[40] Dean, R. B., 1978, "Reynolds-number dependence of skin friction and other bulk flow variables in two-dimensional rectangular duct flow", *ASME J. Fluids Eng.*, **100**, pp. 215-223.

[41] Robinson, S. K., 1991, "Coherent motions in the turbulent boundary layer", *Annu. Rev. Fluid Mech.*, **23**, pp. 601-639.

[42] Panton, R. L., 2001, "Overview of the self-sustaining mechanisms of wall turbulence", *Prog. Aerosp. Sci.*, **37**, pp. 341-383.

[43] Grötzbach, G., 1983, "Spatial resolution requirements for direct numerical simulation of Rayleigh-Benard convection", *J. Comput. Phys.*, **49**, pp. 241-264.

[44] Bakewell, H. P., and Lumley, J. L., 1967, "Viscous sublayer and adjacent wall region in turbulent pipe flow", *Phys. Fluids*, **10**, pp. 1880-1889.

[45] Fishpool, G. M., Lardeau, S., and Leschziner, M. A., 2009, "Persistent non-homogeneous features in periodic channel-flow simulations", *Flow, Turbulence, Combust.*, **83**, pp. 323-342.

[46] Jiménez, J., and Moin, P., 1991, "The minimal flow unit in near-wall turbulence", *J. Fluid Mech.*, **225**, pp. 213-240.

[47] Friedrich, R., Hüttl, T. J., Manhart, M., and Wagner, C., 2001, "Direct numerical simulation of incompressible turbulent flows", *Comput. Fluids*, **30**, pp. 555-579.

[48] Alfonsi, G., and Primavera, L., 2007, "A parallel computational code for the proper orthogonal decomposition of turbulent flows", *J. Flow Visual. Image Proc.*, **14**, pp. 267-286.

[49] Kline, S. J., Reynolds, W. C., Schraub, F. A., and Rundstadler, P. W., 1967, "The structure of turbulent boundary layers", *J. Fluid Mech.*, **30**, pp. 741-773.

[50] Antonia, R. A., 1981, "Conditional sampling in turbulence measurement", *Annu. Rev. Fluid Mech.*, **13**, pp. 131-156.

[51] Willmarth, W. W., and Lu, S. S., 1972, "Structure of the Reynolds stress near the wall", *J. Fluid Mech.*, **55**, pp. 65-92.

[52] Blackwelder, R. F., and Kaplan, R. E., 1976, "On the wall structure of the turbulent boundary layer", *J. Fluid Mech.*, **76**, pp. 89-112.

[53] Johansson, A. V., Alfredsson, P. H., and Kim, J., 1991, "Evolution and dynamics of shear-layer structures in near-wall turbulence", *J. Fluid Mech.*, **224**, pp. 579-599.

[54] Theodorsen, T., 1952, "Mechanism of turbulence", *Proc. 2nd Midwestern Fluid Mechanics Conf.*, Columbus, Ohio, pp. 1-18.

[55] Head, M. R., and Bandyopadhyay, P., 1981, "New aspects of turbulent boundary-layer structure", *J. Fluid Mech.*, **107**, pp. 297-338.

[56] Smith, C. R., Walker, J. D. A., Haidari, A. H., and Soburn, U., 1991, "On the dynamics of near-wall turbulence", *Phil. Trans. R. Soc. London*, **336**, pp. 131-175.

[57] Willmarth, W. W., and Tu, B. J., 1967, "Structure of turbulence in the boundary layer near the wall", *Phys. Fluids Suppl.*, **10**, pp. S134-S137.

[58] Offen, G. R., and Kline, S. J., 1975, "A proposed model of the bursting process in turbulent boundary layers", *J. Fluid Mech.*, **70**, pp. 209-228.

[59] Praturi, A. K., and Brodkey, R. S., 1978, "A stereoscopic visual study of coherent structures in turbulent shear flow", *J. Fluid. Mech.*, **89**, pp. 251-272.

[60] Thomas, A. S. W., and Bull, M. K., 1983, "On the role of wall-pressure fluctuations in deterministic motions in the turbulent boundary layer", *J. Fluid Mech.*, **128**, pp. 283-322.

[61] Acarlar, M. S., and Smith, C. R., 1987, "A study of hairpin vortices in a laminar boundary layer, Part 2. Hairpin vortices generated by fluid injection", *J. Fluid Mech.* **175**, pp. 43-83.

[62] Alfonsi, G., 2006, "Coherent structures of turbulence: methods of eduction and results", *Appl. Mech. Rev.*, **59**, pp. 307-323.

[63] Perry, A. E., and Chong, M. S., 1987, "A description of eddying motions and flow patterns using critical-point concepts", *Annu. Rev. Fluid Mech.*, **19**, pp. 125-155.

[64] Hunt, J. C. R., Wray, A. A., and Moin, P., 1988, "Eddies, streams and convergence zones in turbulent flows", *Proc. 1988 Summer Program*, Center for Turbulence Research, NASA Ames/Stanford University, pp. 193-208.

[65] Jeong, J., and Hussain, F., 1995, "On the definition of a vortex", *J. Fluid Mech.*, **285**, pp. 69-94.

[66] Zhou, J., Adrian, R. J., Balachandar, S., and Kendall, T. M., 1999, "Mechanisms for generating coherent packets of hairpin vortices in channel flow", *J. Fluid Mech.*, **387**, pp. 353-396.

[67] Alfonsi, G., and Primavera, L., 2008, "On identification of vortical structures in turbulent shear flow", *J. Flow Visual. Image Proc*., **15**, pp. 201-216.

[68] Wu, X., and Moin, P., 2009, "Forest of hairpins in a low-Reynolds-number zero-pressure-gradient flat-plate boundary layer", *Phys. Fluids*, **21**, p. 091106-1.

[69] Wu, X., and Moin, P., 2009, "Direct numerical simulation of turbulence in a nominally zero-pressure gradient flat-plate boundary layer", *J. Fluid Mech*., **630**, pp. 5-41.

[70] Schoppa, W., and Hussain, F., 2002, "Coherent structure generation in near-wall turbulence", *J. Fluid Mech*., **453**, pp. 57-108.

[71] Chakraborty, P., Balachandar, S., and Adrian, R.J., 2005, "On the relationships between local vortex identification schemes", *J. Fluid Mech*., **535**, pp. 189-214.

[72] Jespersen, D.C., and Levit, C., 1989, "A computational fluid dynamics algorithm on a massively parallel computer", *Int. J. Supercomp. Appl*., **3**, pp. 9-27.

[73] Fischer, P.F., Ho, L.W., Karniadakis, G.E., Rønquist, E.M., and Patera, A.T., 1988, "Recent advances in parallel spectral element simulation of unsteady incompressible flows", *Comput. Struct*., **30**, pp. 217-231.

[74] Pelz, R.B., 1991, "The parallel Fourier pseudo-spectral method", *J. Comput. Phys*., **92**, pp. 296-312.

[75] Jackson, E., She, Z.S., and Orszag, S.A., 1991, "A case study in parallel computing: I. Homogeneous turbulence on a hypercube", *J. Sci. Comput.*, **6**, pp. 27-45.

[76] Chen, S., and Shan, X., 1992 "High-resolution turbulent simulations using the Connection Machine-2", *Comput. Phys.*, **6**, pp. 643-646.

[77] Johan, Z., Hughes, T.J.R, Mathur, K.K., and Johnsson, S.L., 1992, "A data parallel finite element method for computational fluid dynamics on the Connection Machine system", *Comput. Meth. Appl. Mech. Eng.*, **99**, pp. 113-134.

[78] Naik, N.H., Naik, V.K., and Nicoules, M., 1993, "Parallelization of a class of implicit finite difference schemes in computational fluid dynamics", *Int. J. High Speed Comput*., **5**, pp. 1-50.

[79] Fatoohi, R.A., 1994, "Adapting a Navier-Stokes solver for three parallel machines", *J. Supercomput.*, **8**, pp. 91-115.

[80] Basu, A.J., 1994, "A parallel algorithm for spectral solution of the three-dimensional Navier-Stokes equations", *Parall. Comput.*, **20**, pp. 1191-1204.

[81] Briscolini, M., 1995, "A parallel implementation of a 3-D pseudospectral based code on the IBM 9076 scalable POWER parallel system", *Parall. Comput.*, **21**, pp. 1849-1862.

[82] Floros, N., and Reeve, J.S., 1995, "Evaluation of a spectral-element CFD code on parallel architectures", *Parall. Comput.*, **21**, pp. 1137-1150.

[83] Prestin, M., and Shtilman, L., 1995, "A parallel Navier-Stokes solver: the Meiko implementation", *J. Supercomput.*, **9**, pp. 347-364.

[84] Crawford, C.H., Evangelinos, C., Newman, D., and Karniadakis, G.E., 1996, "Parallel benchmarks of turbulence in complex geometries", *Comput. Fluids*, **25**, pp. 677-698.

[85] Garg, R.P., Ferziger, J.H., and Monismith, S.G., 1997, "Hybrid spectral finite difference simulations of stratified turbulent flows on distributed memory architectures", *Int. J. Numer. Meth. Fluids*, **24**, pp. 1129-1158.

[86] Wasfy, T., West, A.C., and Modi, V., 1998, "Parallel finite element computation of unsteady incompressible flows", *Int. J. Numer. Meth. Fluids*, **26**, pp. 17-37.

[87] Garbey, M., and Vassilevski, Y.V., 2001, "A parallel solver for unsteady incompressible 3D Navier-Stokes equations", *Parall. Comput.*, **27**, pp. 363-389.

[88] Gropp, W.D., Kaushik, D.K., Keyes, D.E., and Smith, B.F., 2001, "High-performance parallel implicit CFD", *Parall. Comput.*, **27**, pp. 337-362.

[89] Kumar, B.V.R., Yamaguchi, T., Liu, H., and Himeno, R., 2001, "A parallel 3D unsteady incompressible flow solver on VPP700", *Parall. Comput.*, **27**, pp. 1687-1713.

[90] Hoeflinger, J., Alavilli, P., Jackson, T., and Kuhn, B., 2001, "Producing scalable performance with OpenMP: experiments with two CFD applications", *Parall. Comput.*, **27**, pp. 391-413.

[91] Dong, S., and Karniadakis, G.E., 2004, "Dual-level parallelism for high-order CFD methods", *Parall. Comput.*, **30**, pp. 1-20.

[92] Itakura, K., Uno, A., Yokokawa, M., Ishihara, T., and Kaneda, Y., 2004, "Scalability of hybrid programming for a CFD code on the Earth Simulator", *Parall. Comput.*, **30**, pp. 1329-1343.

[93] Habata, S., Umezawa, K., Yokokawa, M., and Kitawaki, S., 2004, "Hardware system of the Earth Simulator", *Parall. Comput.*, **30**, pp. 1287-1313.

[94] Yanagawa, T., and Suehiro, K., 2004, "Software system of the Earth Simulator", *Parall. Comput.*, **30**, pp. 1315-1327.

[95] Xu, J., 2007, "Benchmarks on tera-scalable models for DNS of turbulent channel flow", *Parall. Comput.*, **33**, pp. 780-794.

[96] Behara, S., and Mittal, S., 2009, "Parallel finite element computation of incompressible flows", *Parall. Comput.*, **35**, pp. 195-212.

[97] Grinberg, L., Pekurovsky, D., Sherwin, S.J., and Karniadakis, G.E., 2009, "Parallel performance of the coarse space linear vertex solver and low energy basis preconditioner for spectral/hp elements", *Parall. Comput.*, **35**, pp. 284-304.

[98] Agrawal, G., Sussman, A., and Saltz, J., 1995, "An integrated runtime and compile-time approach for parallelizing structured and block structured applications", *IEEE Trans. Parallel Distrib. Syst.*, **67**, pp. 747-754.

[99] Dulong, C., 1998, "The IA-64 architecture at work", *IEEE Comput*., **31**, pp. 24-32.

[100] Lamport, L., 1979, "How to make a multiprocessor computer that correctly executes multiprocess programs", *IEEE Trans. Comput*., **28**, pp. 690-691.

[101] Hagersten, E., Landin, A., and Haridi, S., 1992, "DDM: a cache-only memory architecture", *IEEE Comput.*, **25**, pp. 45-54.

[102] Oliker, L., Canning, A., Carter, J., Shalf, J., and Ethier, S., 2008, "Scientific application performance on leading scalar and vector supercomputing platforms", *Int. J. High Perf. Comput. Appl.*, **22**, pp. 5-20.

[103] Russel, R.M., 1978, "The Cray-1 computer system", *ACM Commun.*, **21**, pp. 63-72.

[104] Espasa, R., Valero, M., and Smith, J.E., 1998, "Vector architectures: Past, present and future", *Proc. ICS 98, Melbourne,* Australia, pp. 425-432.

[105] Oyanagi, Y., 1999, "Development of supercomputers in Japan: hardware and software", *Parall. Comput.*, **25**, pp. 1545-1567.

[106] Hennessy, J. L., and Patterson, D. A., 1990, *Computer Architecture. A Quantitative Approach*, San Francisco, Morgan Kaufmann.

[107] Patterson, D. A., and Hennessy, J. L., 2009, *Computer Organization and Design. The Hardware/Sofware Interface*, San Francisco, Morgan Kaufmann.

[108] Fischer, P.F., and Patera, A.T., 1994, "Parallel simulation of viscous incompressible flows", *Ann. Rev. Fluid Mech.*, **26**, pp. 483-527.

[109] Cremonesi, P., Rosti, E., Serazzi, G., and Smirni, E., 1999, "Performance evaluation of parallel systems", *Parall. Comput.*, **25**, pp. 1677-1698.

[110] Dongarra, J.J., Hey, T., and Strohmaier, E., 1996, "Selected results from the Parkbench benchmark", *Lecture Notes in Computer Science 1124*, Springer, New York.

[111] Alfonsi, G., and Muttoni, L., 2004, "Performance evaluation of a Windows NT based PC cluster for high performance computing", *J. Syst. Archit.*, **50**, pp. 345-359.

[112] Ghosal, D., Serazzi, G., and Tripathi, S.T., 1991, "Processor workingset and its use in scheduling multiprocessor systems", *IEEE Trans. Soft. Eng.*, **17**, pp. 443-453.

[113] NVIDIA®, 2010, *NVIDIA CUDA™ - Programming Guide. Version 3.2*, url: http://developer.download.nvidia.com/compute/cuda/3_2_prod/toolkit/docs/CUDA_C_Programming_Guide.pdf.

[114] AMD®, 2011, *ATI Stream Technology – GPU and CPU Technology for Accelerated Computing*, url: www.amd.com/US/PRODUCTS/TECHNOLOGIES/STREAMTECHNOLOGY/Pages/stream-technology.aspx.

[115] Khronos Group, *OpenCL – The open standard for parallel programming of heterogeneous systems*, www.khronos.org.

[116] Kirk, D. and Hwu, W. W, 2010, *Programming Massively Parallel Processors: a Hands-On Approach*, San Francisco, Morgan Kaufmann.

[117] NVIDIA, 2009, *NVIDIA CUDA™ - Architecture*, url: http://developer.download.nvidia.com/compute/cuda/docs/CUDA_Architecture_Overview.pdf.

[118] IEEE Standards Association, IEEE 754: Standard for Binary Floating-Floating Point Arithmetic, url: http://grouper.ieee.org/groups/754/.

[119] Goldberg, D., 1991, "What Every Computer Scientist Should Know About Floating-Point Arithmetic", *Iss. Comp. Surv.*, pp. 153-230.

[120] Whitehead, N., and Fit-Florea, A., 2011, "Precision & Performance: Floating Point and IEEE 754 Compliance for NVIDIA GPUs", url: http://developer.download.nvidia.com/assets/cuda/files/NVIDIA-CUDA-Floating-Point.pdf.

[121] NVIDIA, 2010, *NVIDIA CUDA™ - CUDA C Best Practice Guide*, url: http://developer.download.nvidia.com/compute/cuda/3_2_prod/toolkit/docs/CUDA_C_Best_Practices_Guide.pdf.

[122] Ferziger, J. H., and Perić, M., 2002, *Computational Methods for Fluid Dynamics*, Springer-Verlag Berlin Heidelberg, New York.

[123] Frigo, M., and Johnson, S. G., 2003, *FFTW User's Manual*, Massachusetts Institute of Technology, url: http://www.fftw.org/fftw2.pdf.

[124] Conte, S.D., and deBoor, C., 1972, *Elementary Numerical Analysis*, McGraw-Hill, New York.

[125] Chapman, B., Jost, G., and van der Pas, R., 2007, *Using OpenMP – Portable Shared Memory Parallel Programming*, The MIT Press, England.

[126] NVIDIA, 2010, *NVIDIA CUDA™ - CUDA CUFFT Library*, url:

http://developer.download.nvidia.com/compute/cuda/3_2_prod/toolkit/docs/CUFFT_Library.pdf.

[127] Karniadakis, G. E., and Orszag, S. A., 1993, "Nodes, modes and flow codes", *Phys. Today*, **46**, pp. 34-42.

[128] Österlund, J. M., Johansson, A. V., Nagib, H. M., and Hites, M. H., 2000, "A note on the overlap region in turbulent boundary layers", *Phys. Fluids*, **12**, pp. 1-4.

[129] Barenblatt, G. I., 1993, "Scaling laws for fully developed turbulent shear layer flows. Part 1. Basic hypothesis and analysis", *J. Fluid Mech.*, **248**, pp. 513-520.

[130] Barenblatt, G. I., and Prostokishin, V. M., 1993, "Scaling laws for fully developed turbulent shear flows. Part 2. Processing of experimental data", *J. Fluid Mech.*, **248**, pp. 521-529.

[131] Zagarola, M. V., and Smits, A. J., 1997, "Scaling of the mean velocity profile for turbulent pipe flow", *Phys. Rev. Lett.*, **78**, pp. 239-242.

[132] Zagarola, M. V., Perry, A. C., and Smits, A. J., 1997, "Log laws or power laws: the scaling in the overlap region", *Phys. Fluids*, **9**, pp. 2094-2100.

[133] Nikuradse, J., 1932, "Gesetzmässigkeiten der turbulenten strömung in glatten rohren", *VDI Forschungheft No. 356* (NACA TT F-10).

[134] Alfonsi, G., and Primavera, L., 2009, "Determination of the threshold value of the quantity chosen for vortex representation in turbulent flow", *J. Flow Visual. Image Proc.*, **16**, pp. 41-49.

[135] Willmarth, W. W., and Sharma, L. K., 1984, "Study of turbulent structure with hot wires smaller than the viscous length", *J. Fluid Mech.*, **142**, pp. 121-149.

[136] Wallace, J. M., Eckelmann, H., and Brodkey, R. S., 1972, "The wall region in turbulent shear flow", *J. Fluid Mech*., **54**, pp. 39-48.

[137] Brodkey, R. S., Wallace, J. M., and Eckelmann, H., 1974, "Some properties of truncated turbulence signals in bounded shear flows", *J. Fluid Mech*., **63**, pp. 209-224

[138] Eckelmann, H., 1974, "The structure of the viscous sublayer and the adjacent wall region in a turbulent channel flow", *J. Fluid Mech*., **65**, pp. 439-459.

[139] Wallace, J. M., Brodkey, R. S., and Eckelmann, H., 1977, "Pattern-recognized structures in bounded turbulent shear flows", *J. Fluid Mech*., **83**, pp. 673-693.

[140] Praturi, A. K., and Brodkey, R. S., 1978, "A stereoscopic visual study of coherent structures in turbulent shear flow", *J. Fluid Mech*., **89**, pp. 251-272.

[141] Kreplin, H. P., and Eckelmann, H., 1979, "Propagation of perturbations in the viscous sublayer and adjacent wall region", *J. Fluid Mech*., **95**, pp. 305-322.

[142] Kreplin, H. P., and Eckelmann, H., 1979, "Behavior of the three fluctuating velocity components in the wall region of a turbulent channel flow", *Phys. Fluids*, **22**, pp. 1233-1239.

[143] Adrian, R. J., 1979, "Conditional eddies in isotropic turbulence", *Phys. Fluids*, **22**, pp. 2065-2070.

[144] Adrian, R. J., and Moin, P., 1988, "Stochastic estimation of organized turbulent structure: homogeneous shear flow", *J. Fluid Mech*., **190**, pp. 531-559.

[145] Adrian, R. J., Moin, P., and Moser, R. D., 1987, "Stochastic estimation of conditional eddies in turbulent channel flow", *Proc. 1987 Summer Progr. Cent. Turb. Res.*, NASA Ames/Standford University, pp. 7-19.

[146] Blackwelder, R. F., and Kaplan, R. E., 1976, "On the wall structure of the turbulent boundary layer", *J. Fluid Mech.*, **76**, pp. 89-112.

[147] Adrian, R., J, Meihart, C. D., and Tomkins, C. D., 2000, "Vortex organization in the outer region of the turbulent boundary layer", *J. Fluid Mech.*, **422**, pp. 1-54.

[148] Grant, H. L., 1958, "The large eddies of turbulent motion", *J. Fluid Mech.*, **4**, pp. 149-190.

[149] Townsend, A. A., 1976, 2nd ed: *The Structure of Turbulent Shear Flow*, Cambridge University Press.

[150] Bogard, D. G., and Tiederman, W. G., 1983, "Investigation of flow visualization techniques for detecting turbulent bursts", *Symposium of Turbulence 1982* (ed. X. B. Reed, G. K. Patterson and J. L. Zakin), p. 289, University of Missouri, Rolla.

[151] Luchik, T. S., and Tiederman, W. G., 1987, "Timescale and structure of ejections and bursts in turbulent channel flows", *J. Fluid Mech.*, **174**, pp. 529-552.

[152] Mansour, N. N., Kim, J., and Moin, P., 1987, "Reynolds-Stress and Dissipation Rate Budgets in a Turbulent Channel Flow", *J. Fluid Mech.*, **194**, pp. 15-44.

[153] Gatski, T. B., 2004, "Constitutive equations for turbulent flows", *Theoret. Comput. Fluid Dyn.*, **18**, pp. 345-369.