Università degli Studi di Firenze
Facoltà di Ingegneria

Corso di Laurea in
Ingegneria Meccanica - Mechanical Engineering

Implementation and Validation of RANS Turbulence Models for Heat Transfer Analysis in an Object Oriented CFD Code

Relatori:

Prof. Bruno Facchini

Prof. Ennio Carnevale

Ing. Luca Mangani

Candidato:

Cosimo Bianchini

Anno Accademico 2005/2006

To the first Ing. Bianchini

Nothing is more dangerous than an idea
when it is the only one you have.
Emile Chartier

Acknowledgements

I would firstly like to thank Ing. Luca Mangani for the crystalline example of abnegation and the never ending enthusiasm in researching.
Moreover, an heartfelt thankyou to Prof. Bruno Facchini because trusting my capabilities, offered me the opportunity to give myself up to this work.
I am grateful to Dott. Ing. Antonio Andreini for the support and the always precise suggestions given in developing this thesis.
I would also like to mention Marco, because his advices made me thinking the first time to OpenFOAM, Emiliano, Lorenzo, Luca, Matteo and Riccardo for the help and the mirth given. I don' t want to forget my mates Filippo, Francesco, Lorenzo, Roberto without whom efforts in this last period would have been much more difficult to bear.
More than a thankyou is reserved to my family, always proud and respectful of my choices, for everything they have done during these long but joyful years of studying.

Abstract

A new object oriented code named OpenFOAM, released by OpenCFD Ltd and aimed at solving continuum mechanics problems, has been provided with suitable tools for predicting flows involving heat transfer in conditions typical of turbomachinery cooling systems. Expansions have been added in two directions: an All-Mach pressure based solver for co-located arrangement and several RANS eddy viscosity turbulence models for Low Reynolds grids have been implemented.

OpenFOAM supports operator-based implicit and explicit second and fourth-order Finite Volume (FV) discretization in three dimensional space and on curved surfaces.

The solver is a development of the SIMPLE algorithm to include compressibility effects and to avoid checkerboarding profiles without the need of staggered arrangement.

Implemented turbulent models space from a set of different Low Reynolds $k-\varepsilon$ models, to models of $k-\omega$ class ending to Two Layer models. Realizability constraints have been used too.

Tests for validation here presented concern geometries for impingement as well as film and internal duct cooling. Both standard reference cases, Cooper et al. [1993] and Sinha et al. [1991], and in-house experiments, Facchini and Surace [2006] and Arcangeli et al. [2006], are shown.

Contents

List of Figures

List of Tables

Nomenclature

Symbols

A	generic coefficient matrix, generic diagonal coefficient in algebraic equation	[ ],[A]
a	generic out of diagonal coefficient in algebraic equation	[A]
b	mass imbalance	$[kg/s]$
c	speed of sound	$[m/s]$
C_p	constant pressure specific heat	$[J/(kg \cdot K)]$
C_μ	coefficient for eddy viscosity	[ ]
$C_{\varepsilon 1}, C_{\varepsilon 2}$	coefficients in turbulent dissipation equation	[ ]
$C_{\omega 1}, C_{\omega 2}$	coefficients in specific turbulent dissipation equation	[ ]
d	$\frac{\Delta x}{a}$	$[m/A]$
E	specific energy	$[J/kg]$
F	external force, surface mass flux	$[N], [kg/s]$
f	external force per unit volume	$[N/m^3]$
f_μ	damping function for eddy viscosity	[ ]
f_1, f_2	damping functions for ε equation	[ ]
F_1	blending function for $k-\omega$ turbulence models	[ ]
G	explicit part of turbulent kinetic energy production	$[kg/(m\ s^3)]$

h	specific static enthalpy	$[J/kg]$
h_0	specific total enthalpy	$[J/kg]$
H	remainder	[ ]
HTC	heat transfer coefficient	$[W/(m^2K)]$
i, j, k	generic counters	[ ]
i	specific internal energy	$[J/kg]$
$\vec{J}$	generic flux vector	$[\Phi/(m^2 s)]$
j	generic face flux	$[\Phi/(m^2 s)]$
k	thermal conductivity, turbulent kinetic energy	$[W/(m\,K)]$, $[m^2/s^2]$
k^+	non-dimensional turbulent kinetic energy $= \frac{k}{u_\tau^2}$	[ ]
l_t	turbulent length scale	$[m]$
l_μ	turbulent length scale for eddy viscosity	$[m]$
l_ε	turbulent length scale for dissipation	$[m]$
m	mass	$[kg]$
$\vec{n}$	normal versor	[ ]
n	number of cell	[ ]
p	pressure	$[Pa]$
P_k	turbulent kinetic energy production term	$[kg/(m\ s^3)]$
P	pitch	$[m]$
Pr	Prandtl number	[ ]
Pr_t	turbulent Prandtl number	[ ]
Q	source term	$[\Phi/s]$
q	source term per unit volume	$[\Phi/(m^3 s)]$
$\dot{Q}$	rate of heat	$[W]$
$\vec{q}$	heat flux vector	$[W/m^2]$
R	gas constant	$[J/(kg\,K)]$
Re_y	Reynolds y number $= \frac{\sqrt{k}y}{\nu}$	[ ]
Re_t	Reynolds turbulent number $= \frac{k^2}{\varepsilon\nu}$	[ ]
S	work done by body forces per unit volume, surface	$[W/m^3], [m^2]$

S_{ij}	rate of strain tensor	$[s^{-1}]$
t	time	$[s]$
T	temperature	$[K]$
T^+	non dimensional temperature	[ ]
T_s	turbulent time scale	$[s]$
Tu	turbulent intensity $= \frac{u'}{\overline{U}} \approx \frac{\sqrt{\frac{2}{3}k}}{\overline{U}}$	$[\%]$
u, v, w	velocity components in x,y,z direction	$[m/s]$
$\vec{U}$	velocity vector	$[m/s]$
U_i	generic velocity component	$[m/s]$
u_τ	friction velocity	$[m/s]$
u_0	equivalent friction velocity	$[m/s]$
u^0	pseudo velocity	$[m/s]$
x, y, z	cartesian coordinates	$[m]$
y	wall distance	$[m]$
y^+	non-dimensional wall distance $\frac{yu_\tau}{\nu}$	[ ]
W	rate of work	$[W]$
$\mathscr{T}$	Reynolds stress tensor	$[Pa]$

Greek symbols

α	thermal diffusivity, relaxation factor	$[m^2/s]$, [ ]
β^*	constant for turbulent kinetic energy production term $k-\omega$ class	[ ]
γ	specific heat ratio, blending factor, $\frac{\overline{v'}}{\overline{w'}}$	[ ], [ ], [ ]
Γ	diffusivity coefficient	$[1/sm]$
δ_{ij}	Kronecker delta	[ ]
$\Delta x, \Delta y$	dimensions of bidimensional control volume	$[m]$
ε	turbulent kinetic energy dissipation	$[m^2/s^3]$
ε^+	non-dimensional turbulent kinetic energy dissipation $= \frac{\nu\varepsilon}{u_\tau^4}$	[ ]

η	adiabatic effectiveness	[ ]
$< \eta >$	averaged adiabatic effectiveness	[ ]
κ	Von Karman's constant = 0.41	[ ]
λ	distance weighting factor, bulk viscosity	[ ], $[kg\ /(m\ s)]$
μ	viscosity	$[kg\ /(m\ s)]$
μ_t	eddy viscosity	$[kg\ /(m\ s)]$
ν	kinematic viscosity	$[m^2/s]$
ρ	density	$[kg/m^3]$
$\sigma_k, \sigma_\varepsilon, \sigma_\omega$	coefficients for eddy viscosity in k, ε and ω equation	[ ]
τ	shear stress	$[Pa]$
Φ	generic scalar, viscous dissipation	$[\Phi], [W/m^3]$
Ψ	$\frac{1}{RT}$ - compressibility factor	$[kg/J]$
χ	net flux through surface	$[\Phi/s]$
ω	specific turbulent kinetic energy dissipation	$[s^{-1}]$
Ω	volume	$[m^3]$

Subscripts

b	body
c	coolant
E, W, N, S	east, west, north, south cell neighbor
e, w, n, s	east, west, north, south face
EE	right (east) neighbor of east neighbor
g	hot gas
m	counter for inner iteration
n	normal direction
nb	neighbor
P	cell of reference node
ref	reference

s	surface
w	wall

Superscripts

n	outer iteration counter
$'$	correction, Reynolds fluctuation
$''$	Favre fluctuation
$*$	guessed
$\bar{}$	Reynolds averaged
$\tilde{}$	Favre averaged

Introduction

Nature and Significance of present research

One of the most demanding task in gas turbine design is the proper evaluation of all phenomena involved with heat transfer. Of particular interest are the compound cooling systems for the hot components of the engine. Precise investigations for the zones in which heat transfer to solid walls is higher is fundamental as long as actual design criteria force cooling system to more and more limiting conditions. Thinking about blade cooling for example, the need for improvements in overall performances tend to increase temperature in turbine vanes, at the same time the amount of available cooling air is wanted to be as low as possible to keep work losses at the minimum. For combustors the situation is not so different with the new lean premixed flames design criteria reducing the quantity of air for liner cooling. Accurate heat transfer measurements are however very expensive to support due to the complexity of geometries, the high costs of measuring device and the long set up necessary to collect reliable data. This is why CFD analysis is becoming more and more popular in each phase of the design process.

Nevertheless, CFD simulations for evaluation of thermal loads and effectiveness of the cooling devices in gas turbine engines are one of the most complex to face. Complexity arises especially because of the need for advanced physical modeling and flexibility in geometrical mesh handling. Typical example for the necessity of particular models is the treatment of turbulence.

Modeling turbulence results very important in many fluxes but moreover it is basilar in well predicting impinging jets. Complexity grows because the good functioning of these models is quite case dependent. Viceversa other kinds of cooling devices are very demanding in terms of mesh handling: internal ducts are usually improved with non aerodynamic turbulator such as ribs or pins, where a very fine grid is requested to resolve the complex structure of the local flow.

The capability to handle hybrid unstructured meshes and the availability of a quite large set of turbulence models become a must for a CFD code aimed at solving heat transfer related problems. Unfortunately these requirements result to be quite strict and not many softwares worldwide can satisfy them. That is why, in the last decade, both industrial and academic researchers have been widely using only few well known commercial solvers.

Commercial solvers aspire to resolve many kinds of flow and so they are often provided with a big amount of different packages for various models. This generality often results in poor performances in terms of calculation times due to a waste of system resources for a large part of packages that are not used in standard simulations.

Apart from this, the main drawback, according to expert users, stands in how commercial CFD codes behave like a "black box solution maker". Especially in case of disagreements with experiments, it is not of secondary importance to perfectly know under which hypothesis numerical predictions could be generated. Information on the process used to obtain convergence can be found only on often poor of details user manuals: source codes are obviously unaccessible.

Advanced users in heat transfer applications sometimes need the use of "ad hoc" models or modifications suitable for specific cases. User subroutine features provided by commercial packages become quickly inadequate as the complexity of modifications grows. Furthermore R&D department of big

companies usually need to tune built-in models in order to feed calculations tools with their design practice frequently based on detailed and expensive experimental tests.

Objectives and present contributions

The objective of this thesis is to show the capabilities of a new open-source software environment where it is possible to implement new models, renew the existing ones and experiment with model combinations. The OpenFOAM package (Field Operation And Manipulation) is an object-oriented numerical simulation toolkit written in C++ language. Besides its advanced basic native CFD features its essential characteristic, in opposition to commercial solvers, is the opportunity to build new models and solvers with high simplicity and in less time than with standard Fortran based codes. Object-oriented programming of C++ drastically reduces the probability of bugs introduction with a consequent reduction in debugging time. This thesis describes the attempt to build a CFD package suitable for typical steady state heat transfer analysis which could be able to assist gas turbine design process. As will be described later, to reach such goal it was necessary to introduce specific features in the package in order to overcome the limitations of built-in approaches: first of all a compressible steady state solver capable of handling transonic flows, then a set of turbulence two-equations closures with particular reference to a detailed near wall treatment.

As confirmation of the work done a set of validation testcases were performed. In particular, in this thesis, we will focus our attention on the validation of the code with some complex configurations typical of heat transfer problems such as ribbed channel, film cooling and impingement cooling.

Outline of thesis

Chapter 1 is based on a review of classical arguments concerned with Computational Fluid Dynamics. All the different parts numerical solution methods are composed by, are briefly presented with no aim of completeness but to give an hint of the approach used in this thesis.

Chapter 2 presents main features of the base code OpenFOAM with an overview on the C++ philosophy of programming, a description of how the linear systems are built and an example of the structure of one of the most common classes. Moreover an example on how to write a postprocessing utility is given.

Chapter 3 describes the solver used. First general themes regarding solution algorithms such as pressure velocity coupling and segregation are introduced. Thus the ancestor of the solver used is described and finally expansion to co-located approach and compressibility effects are presented.

Chapter 4 introduces the problem of modeling turbulence as first thing. Eddy viscosity RANS models are introduced first deriving standard $k - \varepsilon$ model, then presenting a number of different models for Low Reynolds grids. Finally the specific OpenFOAM class for turbulence models is described and commented.

Chapter 5 reports the results for the simulations done to validate the models proposed. After an initial test over a flat plate, typical geometries for turbomachinery cooling devices are presented. First three cases for impingement cooling and secondly two film cooling set-ups. In both cases single and multiple hole configurations are analyzed. Last comes a ribbed duct for internal cooling of a trailing edge of blade.

Chapter 6 gives the conclusions of this research work with particular emphasis on the weaknesses and suggests the path to follow for future developments.

Chapter 1

Basic CFD

This chapter introduces and discusses the basilar aspects of Computational Fluid Dynamics starting from a general overview, with the aim of organizing such branch into the more complex environment of fluid flow predictions, and going through the structural parts in which numerical methods can be subdivided. In particular all basic aspects of numerical prediction techniques are touched, starting from the governing equations of the mathematical model, giving a brief introduction to discretization methods and finite approximation and mentioning methods for solving the resulting system. In the following particular interest will be posed on the Finite Volume Method.

1.1 Overview

Prediction of heat transfer and fluid flow processes can be obtained by two main methods: experimental investigation and theoretical calculation.

Even if reliable information about physical processes could only be obtained by actual measurement, experimental investigations involving full scale equipment often result to be too expensive. The alternative is so to use scaled models and conditions, at the end extrapolate the results to full scale.

This scaling, however, is not completely free from errors: general rules for correct scaling are often unavailable, phenomena may not be scalable (i.e. combustion, turbulence, etc ...), measuring instruments errors may weight more.

Such problems can be avoided with the help of a mathematical model able to suitably represent the physical process. Theoretical predictions consist in working out the consequences of such model. For fluid dynamics problems, the mathematical model basically consists in a set of partial differential equation. If classical mathematics techniques were to be used for solving such equations, there would be little hope of predicting many cases of practical interest with closed form solution. Hopefully, with the development of numerical methods and the availability of large digital computers, closure can be found almost for any practical problem.

Furthermore, computer analysis offers several advantages with respect to experimental investigation: low cost, speed, complete and detailed information, capability in simulating both realistic and ideal conditions. At the same time, numerical calculation are not free from disadvantages: it can happen that a suitable mathematical model for describing the physical conditions cannot be found, it is possible for prediction with a very limited objective not to be cheaper than experiments, problems involving complex geometries, strong non linearity may be harder, longer and again more expensive to solve.

It appears clearly now how experiments and computations must coexist and interact to have wide and reliable predictions [Patankar, 1980].

Let's now analyze in detail what a numerical solution method is composed by. As above written, the starting point of a numerical method is the **Mathematical Model**, i.e. the set of partial differential equations and boundary conditions. Fluid dynamic science teaches that exact conservation laws describe the behavior of all flows: no matter the type of flow, it will respect the general governing equations. General purpose methods however

are often impractical, if not impossible, to solve so it is more convenient to include simplifications in the mathematical model and develop a solution method designed for that particular set of equations. Then a suitable **Discretization Method**, approximating the set of differential equations by a system of algebraic equations for the variables at a number of discrete points in space and time, is necessary. The most important discretization methods are: Finite Difference Method (FDM), Finite Volume Method (FVM) and Finite Element Method (FEM). The discrete locations where the variables want to be calculated, are defined by the **Numerical Grid**. The numerical grid is a discrete representation of the flow domain (both in space and time) through the use of a finite number of subdomains such as elements, control volumes etc...

Then a **Finite Approximation** technique has to be selected taking in consideration the choice for the discretization method and the numerical grid. This choice influence a lot the accuracy of the solution as well as the development, coding, debugging and the speed of the solution method. More accurate approximations involve, in fact, more nodes and give usually a fuller coefficient matrix. A compromise between accuracy and efficiency is always necessary. Once this large system of non-linear algebraic equations has been built by discretization techniques, it must be solved using a **Solution Method**. Such methods use successive linearization of the equations and the resulting linear systems are almost always solved by iterative techniques. Usually there are two levels of iterations: inner iterations, within which the linear equation are solved, and the outer iterations, that deal with the non linearity and coupling of the equations. As last point, it is important to determine suitable **Convergence Criteria**. It is fundamental to well set stopping conditions for both the inner and the outer cycles in order to obtain accurate solution in an efficient way.

Once defined, numerical methods must be checked to posses certain prop-

erties in order to establish whether a method is appropriate or not. The most important properties are:

- **Consistency**: discretization should become exact as the grid spacing tends to zero. In other words truncation error, i.e. the difference between exact and discretized equation, must go to zero as $\triangle t \rightarrow 0$ and $\triangle x \rightarrow 0$.

- **Stability**: errors appearing in the course of numerical solution process do not magnify. For iterative methods, stable methods are the ones that do not diverge.

- **Convergence**: the solution of the discretized equation tends towards the exact solution as the grid spacing tends to zero. It is a very difficult property to demonstrate, it is usually accepted to test grid-independence for a solution.

- **Conservation**: solution must respect conservation of physical quantities both on local and global scale. It is a very important property because limits solution error. Even if on fine grids non-conservative schemes can also lead to correct solutions, conservative ones are usually preferred.

- **Boundedness**: solution should lie within proper bounds. Boundedness is difficult to guarantee and often unbounded schemes have stability and convergence problems too.

- **Realizability**: guarantee a model to give physically realistic solutions for the phenomena it is representing. Usually in connection with phenomena too complex to be directly simulated.

- **Accuracy**: is the property of well approximating the exact solution, in other words limiting modeling, discretization and iteration errors.

1.2 Physical principles and Mathematical model

In this section conservation equations for mass, momentum and energy for non reacting mono-phase compressible flows, will be described.

Firstly equations are derived in the most general form as possible. The same approach has been used to obtain all the basic equation for fluid motion: apply the appropriate fundamental physical principle to a suitable model of the flow and then extract the mathematical equations which embody such physical principles. The fluid flow has been modeled with infinitesimal control volume fixed in space with the fluid moving through it. As a consequence equations are proposed in the differential conservation form. To switch from one form to another it must be remembered the concept of the substantial derivative:

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + (\vec{U} \cdot \nabla) \, , \tag{1.1}$$

meaning that the rate of increase of a scalar of fluid particle, on the left hand side of Eq. 1.1, is equal to the rate of increase of fluid element plus the net rate of flow of the scalar out of fluid element, on the right hand side of the same equation [Anderson, jr., 1995].

Secondly an appropriate model for the fluid is proposed: simplifications for the cases of interest such as ideal gas and newtonian fluid are introduced.

Third passage in manipulating the set of governing equations, i.e. steady-state simplification and turbulence modeling, is explicitly done in chapter 4 but has been widely used in all the cases of need in the previous chapters too.

1.2.1 Continuity equation

The fundamental physical concept standing behind the continuity equation is that mass is conserved. In other words the rate of increase of mass in fluid element must equal the net rate of flow of mass into fluid element or the rate

of change of mass in particle is null [Malalasekera and Versteeg, 1995]:

$$\frac{D\,m}{Dt} = 0\ . \tag{1.2}$$

With reference to Fig. 1.1 the rate of increase of mass in the fluid element is:

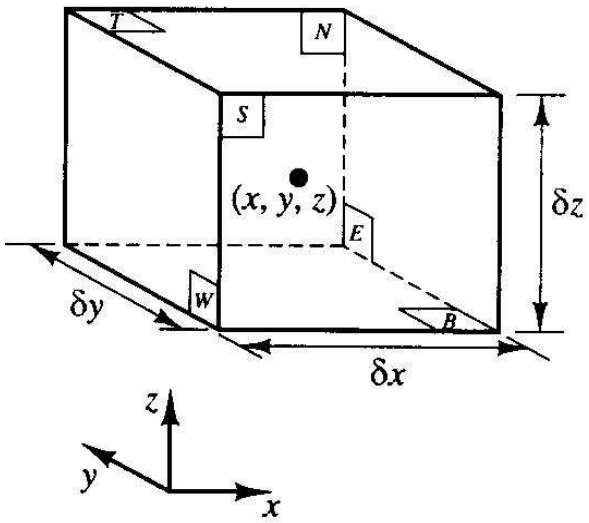


Figure 1.1: Fluid element for conservation laws.

$$\frac{\partial}{\partial t}(\rho\ \delta x\,\delta y\,\delta z) = \frac{\partial \rho}{\partial t}\ \delta x\,\delta y\,\delta z\ , \tag{1.3}$$

thus the net rate of flow of mass into fluid element is:

$$\begin{aligned}
&\left(\rho u - \frac{\partial(\rho u)}{\partial x}\ \frac{1}{2}\delta x\right)\delta y\,\delta z - \left(\rho u + \frac{\partial(\rho u)}{\partial x}\ \frac{1}{2}\delta x\right)\delta y\,\delta z \\
&+ \left(\rho v - \frac{\partial(\rho v)}{\partial y}\ \frac{1}{2}\delta y\right)\delta x\,\delta z - \left(\rho v + \frac{\partial(\rho v)}{\partial y}\ \frac{1}{2}\delta y\right)\delta x\,\delta z \\
&+ \left(\rho w - \frac{\partial(\rho w)}{\partial z}\ \frac{1}{2}\delta z\right)\delta x\,\delta y - \left(\rho w + \frac{\partial(\rho w)}{\partial z}\ \frac{1}{2}\delta z\right)\delta x\,\delta y \\
&\qquad = -\left(\frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} + \frac{\partial(\rho w)}{\partial z}\right)\delta x\,\delta y\,\delta z\ .
\end{aligned} \tag{1.4}$$

Eq. 1.3 and Eq. 1.4 together give the well known differential continuity equation in conservation form:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{U}) = 0\ . \tag{1.5}$$

1.2.2 Momentum equation

From Newton's second law directly follows the momentum equation: the rate of increase of momentum of fluid particle equals the sum of forces on fluid particle:

$$m\frac{D\vec{U}}{Dt} = F_s + F_b \; . \tag{1.6}$$

It is easier to distinguish external forces between surface forces F_s and body forces F_b. The first ones being defined upon control volume boundaries, the second ones on the volume itself. It is common to separate surface forces into pressure, the hydrostatic part of the stress tensor, and viscous stress, the deviatoric part.

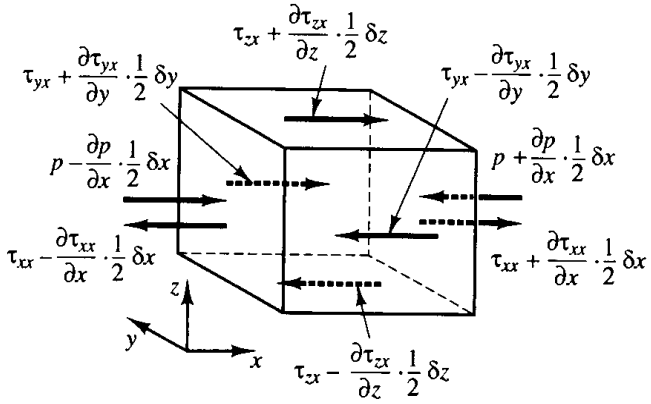


Figure 1.2: Stress component in the x direction.

To derive x-momentum equation it is better start with a balance of surface forces in the x-direction on the (E,W) faces (see Fig. 1.2):

$$\begin{aligned}
&\left[\left(p - \frac{\partial p}{\partial x}\,\frac{1}{2}\delta x\right) - \left(\tau_{xx} - \frac{\partial \tau_{xx}}{\partial x}\,\frac{1}{2}\delta x\right)\right] \delta y\, \delta z \\
&+ \left[-\left(p + \frac{\partial p}{\partial x}\,\frac{1}{2}\delta x\right) + \left(\tau_{xx} + \frac{\partial \tau_{xx}}{\partial x}\,\frac{1}{2}\delta x\right)\right] \delta y\, \delta z \\
&\qquad\qquad = \left(-\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x}\right) \delta x\, \delta y\, \delta z \; ,
\end{aligned} \tag{1.7}$$

then on (N,S) faces:

$$-\left(\tau_{yx}-\frac{\partial\tau_{yx}}{\partial y}\,\frac{1}{2}\delta y\right)\delta x\,\delta z+\left(\tau_{yx}+\frac{\partial\tau_{yx}}{\partial y}\,\frac{1}{2}\delta y\right)\delta x\,\delta z=\frac{\partial\tau_{yx}}{\partial y}\,\delta x\,\delta y\,\delta z, \quad (1.8)$$

and on (T,B):

$$-\left(\tau_{zx}-\frac{\partial\tau_{zx}}{\partial z}\,\frac{1}{2}\delta z\right)\delta x\,\delta y+\left(\tau_{zx}+\frac{\partial\tau_{zx}}{\partial z}\,\frac{1}{2}\delta z\right)\delta x\,\delta y=\frac{\partial\tau_{zx}}{\partial z}\,\delta x\,\delta y\,\delta z. \quad (1.9)$$

Putting Eq. 1.7, Eq. 1.8 and Eq. 1.9 together we obtain the x-component of the momentum equation:

$$\frac{\partial(\rho u)}{\partial t}+\nabla\cdot\left(\rho u\vec{U}\right)=\frac{\partial(-p+\tau_{xx})}{\partial x}+\frac{\partial\tau_{yx}}{\partial y}+\frac{\partial\tau_{zx}}{\partial z}+f_{bx}\,, \quad (1.10)$$

similarly can be done for y-direction:

$$\frac{\partial(\rho v)}{\partial t}+\nabla\cdot\left(\rho v\vec{U}\right)=\frac{\partial\tau_{xy}}{\partial x}+\frac{\partial(-p+\tau_{yy})}{\partial y}+\frac{\partial\tau_{zy}}{\partial z}+f_{by}\,, \quad (1.11)$$

and z-direction:

$$\frac{\partial(\rho w)}{\partial t}+\nabla\cdot\left(\rho w\vec{U}\right)=\frac{\partial\tau_{xz}}{\partial x}+\frac{\partial\tau_{yz}}{\partial y}+\frac{\partial(-p+\tau_{zz})}{\partial z}+f_{bz}\,. \quad (1.12)$$

1.2.3 Energy equation

The energy equation is derived from the first principle of thermodynamics stating that the rate of change of energy of a fluid particle is equal to the rate of heat addition to the fluid particle plus the rate of work done on the particle

$$m\frac{DE}{Dt}=\dot{Q}+W\,. \quad (1.13)$$

Let's determine the rate of heat addition and the work done on the particle in terms of more suitable physical quantities such as temperature gradients, shear and normal stresses.

Referring to Fig. 1.3, the net rate of heat transfer due to heat flow in the x-direction is:

$$\left[\left(q_x+\frac{\partial q_x}{\partial x}\,\frac{1}{2}\delta x\right)-\left(q_x-\frac{\partial q_x}{\partial x}\,\frac{1}{2}\delta x\right)\right]\delta y\,\delta z=-\frac{\partial q_x}{\partial x}\,\delta x\,\delta y\,\delta z\,, \quad (1.14)$$

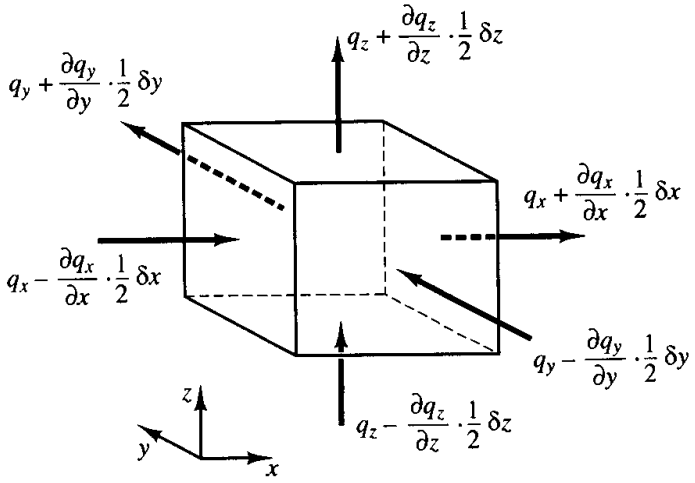


Figure 1.3: Heat flux vector.

and including contributions for the y and z directions, the rate of heat transfer added to the particle per unit volume is:

$$-\frac{\partial q_x}{\partial x} - \frac{\partial q_y}{\partial y} - \frac{\partial q_z}{\partial z} = -\nabla \cdot \vec{q}\,. \tag{1.15}$$

Furthermore, from Fourier's law:

$$\vec{q} = -k\nabla T, \tag{1.16}$$

giving finally:

$$\dot{Q} = -\nabla \cdot (k\nabla T)\ \delta x\, \delta y\, \delta z\ . \tag{1.17}$$

The net rate of work done on the particle by surface forces acting on x-direction is:

$$\begin{aligned}
&\left[\left(pu - \frac{\partial pu}{\partial x}\,\frac{1}{2}\delta x\right) - \left(pu + \frac{\partial pu}{\partial x}\,\frac{1}{2}\delta x\right)\right]\delta y\,\delta z \\
&+\left[-\left(\tau_{xx}u - \frac{\partial \tau_{xx}}{\partial x}\,\frac{1}{2}\delta x\right) + \left(\tau_{xx}u + \frac{\partial \tau_{xx}}{\partial x}\,\frac{1}{2}\delta x\right)\right]\delta y\,\delta z \\
&+\left[-\left(\tau_{yx}u - \frac{\partial \tau_{yx}u}{\partial y}\,\frac{1}{2}\delta y\right) + \left(\tau_{yx}u + \frac{\partial \tau_{yx}u}{\partial y}\,\frac{1}{2}\delta y\right)\right]\delta x\,\delta z \\
&+\left[-\left(\tau_{zx}u - \frac{\partial \tau_{zx}u}{\partial z}\,\frac{1}{2}\delta z\right) + \left(\tau_{zx}u + \frac{\partial \tau_{zx}u}{\partial z}\,\frac{1}{2}\delta z\right)\right]\delta x\,\delta y \\
&\qquad = \left[\frac{\partial(u(-p+\tau_{xx}))}{\partial x} + \frac{\partial(u\tau_{yx})}{\partial y} + \frac{\partial(u\tau_{zx})}{\partial z}\right]\delta x\,\delta y\,\delta z\ . \quad (1.18)
\end{aligned}$$

Following the same approach for the y and z direction as in Eq. 1.18 and summing, the total rate of work done on the fluid particle by surface stress is:

$$W = \left(-\nabla\cdot(pU) + \frac{\partial \tau_{ij}U_i}{\partial x_j}\right)\delta x\,\delta y\,\delta z\ . \quad (1.19)$$

Including Eq. 1.19 and Eq. 1.17 into Eq. 1.13, the final form of the energy equation in obtained, eventually adding the rate of work done by body forces as a source term S_E:

$$\frac{\partial \rho E}{\partial t} + \nabla\cdot\left(\rho E\vec{U}\right) = -\nabla\cdot\left(p\vec{U}\right) + \frac{\partial \tau_{ij}U_i}{\partial x_j} + \nabla\cdot(k\nabla T) + S_E\ . \quad (1.20)$$

For compressible flows it is however more correct, and surely less complicated to treat, to express energy equation in terms of another variable: total enthalpy in which pressure energy is directly included is one of the choices:

$$h_0 = E + \frac{p}{\rho} = i + \frac{1}{2}\left(u^2+v^2+w^2\right) + \frac{p}{\rho}\ . \quad (1.21)$$

Substituting Eq. 1.21 into Eq. 1.20, one may obtain energy equation in terms of total enthalpy:

$$\frac{\partial \rho h_0}{\partial t} + \nabla\cdot\left(\rho h_0\vec{U}\right) = \frac{\partial p}{\partial t} + \frac{\partial \tau_{ij}U_i}{\partial x_j} + \nabla\cdot(k\nabla T) + S_h\ , \quad (1.22)$$

sometimes the term $\frac{\partial \tau_{ij}U_i}{\partial x_j}$ is referred to as the viscous dissipation term Φ.

1.2.4 A model for the fluid

As above mentioned, a Mathematical Model result in being more accurate and easier to handle if suitable simplifications are introduced in the general equations to be used. The aim of this section is so to reduce the area of interest to problems concerned with heat transfer in turbomachinery, giving a specific model to the fluid.

In order to simplify the above derived equations into a simpler set of equation still capable of well modeling such flows, at least three assumptions can be made.

First hypothesis is the assumption of the fluid to be a **perfect gas**, meaning that it follows the well-known equation of state:

$$p = \rho RT. \tag{1.23}$$

Second assumption concerns the modeling of the viscous stress. Fluids of interest are considered **Newtonian**, meaning that viscous stresses, in analogy with hookean elasticity, are linearly proportional to the rates of deformation. A first coefficient μ named dynamic viscosity, relates stress to linear deformation, a second coefficient λ called second viscosity or bulk viscosity takes into account volumetric deformation effect:

$$\tau_{ij} = \mu \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right) + \delta_{ij} \lambda \left(\nabla \cdot \vec{U} \right). \tag{1.24}$$

It may noticed that, following such definition, mechanical and thermodynamic pressure may not coincide. in fact following this definition τ_{ij} is not implicitly traceless. Since little is known about λ it is common to avoid this inconsistency simply assuming that $\lambda = -\frac{2}{3}\mu$. It is also true that the term $\left(\nabla \cdot \vec{U} \right)$ is usually very small and some authors proposed to directly neglect the effects due to volumetric deformation [White, 1991].

Third comes the modeling of **thermophysical properties**. Even if the validation runs have been performed with constant thermophysical proper-

ties, fluid properties are treated, not to loose generality for the proposed model, as function of both temperature and pressure.

All these assumptions and simplifications lead to the final set of governing equations:

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho U_j)}{\partial x_j} = 0 \,, \tag{1.25}$$

$$\begin{aligned} \frac{\partial (\rho U_i)}{\partial t} + \frac{\partial (\rho U_i U_j)}{\partial x_j} &= \\ -\frac{\partial p}{\partial x_i} &+ \frac{\partial}{\partial x_j}\left(\mu\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) - \frac{2}{3}\mu\delta_{ij}\frac{\partial U_j}{\partial x_j}\right) + F_{bi} \,, \end{aligned} \tag{1.26}$$

$$\frac{\partial \rho h_0}{\partial t} + \frac{\partial (\rho h_0 U_j)}{\partial x_j} = \frac{\partial p}{\partial t} + \frac{\partial}{\partial x_j}\left(k\frac{\partial T}{\partial x_j}\right) + \Phi + S_h \,, \tag{1.27}$$

$$p = \rho R T \,, \tag{1.28}$$

where Φ is the viscous dissipation:

$$\Phi = \tau_{ij}\frac{\partial U_i}{\partial x_j} = \mu\left[\frac{1}{2}\left(\frac{\partial U_j}{\partial x_i} + \frac{\partial U_i}{\partial x_j}\right)^2 - \frac{2}{3}\left(\nabla \cdot \vec{U}\right)^2\right]. \tag{1.29}$$

1.3 Discretization approaches

After the physical mathematical model has been derived, the goal is to manipulate it in a form suitable for computer calculations. First step on this path regards the discretization of the equations. The main task of a discretization approach is to convert a partial differential equation, valid on the entire domain, into a set of discrete algebraic equation, one for every node considered. The value at the node is, of course, put in relation with "neighbor" nodes, the simultaneous satisfaction of all the equation in the set then give the numerical solution. The most popular discretization techniques are presented and discussed:

- **Finite Difference Method** (FDM) approximates conservation equations in differential form substituting partial derivatives via truncated Taylor series expansions or polynomial fitting. Even if, in principle,

it can be applied to all kind of grids, actual applications are limited to structured grids where grid lines are used as local coordinate lines. In such a way in fact, it is easy to obtain higher-order schemes. The biggest drawback of FDM is that it does not enforce conservation, consequently it is very hard to get reliable simulations of complex geometries and use is restricted to the simple ones.

- **Finite Volume Method** (FVM) works with the integral form of the conservation equations. The domain is divided into finite volumes whose centroid represent the relative node. Grid just define boundaries in between different volumes and need not to be related to any metrics. Interpolation is used to express variable values at the surface in terms of nodal values. This method is conservative by construction as long as surface integrals for volumes sharing the same face are equal for both of them. The disadvantage of FVM in comparison with FDM is that building higher than second order schemes for 3D simulation is much more difficult due to the three levels of approximation introduced: interpolation, differentiation and integration. Because of its very physical approach, ease to be understood and implemented, FVM is the most widely used approach.

- **Finite Element Method** (FEM) may appear similar to FVM, the distinguish feature is the weight function: conservation equations are multiplied by a weight function before being integrated. Solution is supposed to adhere within each volume to a shape function constructed from values at the corners of the elements. Such an hypothesis is substituted into the conservation equations whose derivative with respect to nodal value is put to zero, selecting so the residual minimizing allowed function. The main advantage in using FEM is the ability in dealing with arbitrary geometries, while the main drawback, common

to all integral methods, is that the resulting matrix may result not well structured meaning that efficient solving method are difficult to implement.

Discussing which of the previously described methods should be used, exact satisfaction of scalar conservation over control volumes has been focused to be the most important properties to search for. It is fundamental in fact that all the simplifications and discretizations to be introduced in the governing equations, always respect exactly basilar physical principles they are built on. The physical laws to be obeyed are all conservation laws so conservation should be perfectly observed. Exact conservation can be imposed externally or intrinsically respected by the discretization method. The second choice seemed the most logic and that is why implicitly conservative methods were selected. Finite Volume Method has so been chosen, as a consequence from now on it will be the method of reference and will be discussed more in detail, see Sec. 1.3.1.

1.3.1 Finite Volume Method

As above mentioned, FVM is based on conservation equations in the integral form. To obtain such equations from the ones derived in Sec. 1.2, integration over a finite volume must be performed. To better fix ideas and not to get lost in heavy calculation, detailed presentation of this method will be done considering the generic conservation equation for a transported scalar Φ:

$$\int_S \vec{J_\Phi} \cdot \vec{n} dS = \underbrace{\int_S \rho\Phi(\vec{U} \cdot \vec{n}) dS}_{Convection} - \underbrace{\int_S \Gamma_\Phi(\nabla\Phi \cdot \vec{n}) dS}_{Diffusion} = \underbrace{\int_\Omega q_\Phi d\Omega}_{Source} . \quad (1.30)$$

Such an equation applies over each control volume and the entire domain as well, underlying once again the main feature of FVM: global conservation. To obtain an algebraic equation the three integral must be approximated by quadrature formulae. In the following discussion about approximation tech-

niques the bi-dimensional quadrilateral CV in Fig. 1.4 has been considered.

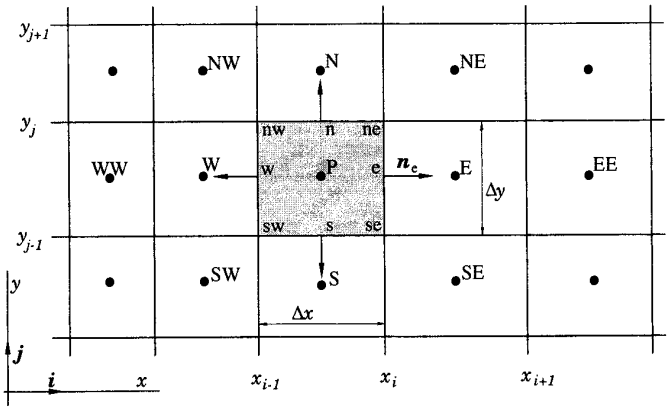


Figure 1.4: Typical bi-dimensional quadrilateral control volume.

1.3.1.1 Approximation of Volume Integrals: Source Terms

Source terms require integration over the entire CV. Approximations of such integrals basically reduce in calculating an averaged value $\overline{q}$ for the source terms to be multiplied by the volume

$$Q_P = \int_\Omega q d\Omega = \overline{q}\,\Omega\ . \tag{1.31}$$

The first and easiest approximation just replaces the mean value with the value at the CV center node

$$Q_P = q_P\,\Omega\ . \tag{1.32}$$

Such approximation is obviously exact if q is constant or varies linearly, in any other case is second order accurate. Higher order of accuracy schemes can be obtained via shape function or interpolation.

1.3.1.2 Approximation of Surface Integrals

The net flux through the CV boundaries is obtained summing over the six faces, the surface integral of the total flux normal vector j composed by convective and diffusive contributions:

$$\int_S \vec{J} \cdot \vec{n} dS = \int_S J_n dS = \sum_k \int_{S_k} j dS = \sum_k \chi_k \ . \tag{1.33}$$

To exactly calculate surface integrals, the value of j is needed everywhere on the face S_k. This is of course impossible as long as only nodal values of Φ are computed, it is so necessary to introduce some approximations. Usually two level of approximation are introduced:

- the integral is approximated in terms of the variable values at one or more locations on the cell face (*face approximation*)
- the cell-face values are approximated in terms of nodal values (*nodal approximation*).

For each type several approximation modes have been proposed. For the face approximation:

- midpoint rule - the value over the face is equal to the value at cell-face center $\chi_e = j_e S_e$. This approximation is of second-order accuracy.
- trapezoid rule - the value on the face is the mean between face extremes (cell vertex) shared with 'neighbor' cells $\chi_e = S_e \frac{1}{2}(j_{ne} + j_{se})$. This approximation is second order too.
- Simpson's rule - the value on the face is a combination of the value at the center of the face and the values at the vertices $\chi_e = S_e \frac{1}{6}(j_{ne} + 4j_e + j_{se})$. This approximation is forth order.

Of course to maintain such levels of accuracy, nodal approximation of, at least, the same order must be chosen. Extension in three dimensions is

quite direct for both the midpoint and trapezoid rule, but gets more complicated for higher-order approximation. Nodal approximation is discussed later Sec. 1.4 being a task requiring deeper treatment.

It has already been cleared that $\vec{J}$ is composed by two different contributions: convection and diffusion. These two terms indeed behave in a quite different manner as will be shown in the following.

1.3.1.3 Diffusive flux

Discretization of the diffusive fluxes gives, using midpoint rule and assuming linear variation:

$$\int_S \Gamma_\Phi(\nabla\Phi \cdot \vec{n})dS = \sum_k S_k(\rho\Gamma_\Phi\nabla\Phi)_k = \sum_k S_k(\rho\Gamma_\Phi)_k(\nabla\Phi)_k \ . \quad (1.34)$$

Let's see how it is not that difficult to compute gradients on the boundary face e:

$$(\nabla\Phi)_e = \frac{\Phi_E - \Phi_P}{x_E - x_P} \ . \quad (1.35)$$

It is however possible to compute the gradient in other way that is:

$$(\nabla\Phi)_P = \frac{1}{\Omega}\sum_k S_k\Phi_k \ , \quad (1.36)$$

and find the corresponding value on the face e interpolating $(\nabla\Phi)_P$ and $(\nabla\Phi)_E$ with techniques to be proposed later in Sec. 1.4.

1.3.1.4 Convective flux

The discretization of the convective fluxes gives using the midpoint rule:

$$\int_S \rho\Phi(\vec{U} \cdot \vec{n})dS = \sum_k S_k(\rho\Phi U_n)_k = \sum_k S_k(\rho U_n)_k\Phi_k = \sum_k F_k\Phi_k \ . \quad (1.37)$$

Transport of Φ across the boundary faces has been computed in quite a direct way, the problem has now shifted in finding an expression for Φ_k involving only nodal values. This pretty much is the most crucial passage of the entire discretization process and so the following section is entirely dedicated to interpolation and differentiation schemes.

1.4 Finite Approximation: Interpolation and Differentiation Schemes

It has already been shown how it is not possible, especially for high order approximation, to compute the algebraic equations only in terms of cell center values but values on the boundary are also needed. The aim of this section is to present some of the most common schemes for interpolation in order to get values at locations other than computational nodes. As in previous sections the flow field is supposed to be known, in addition density and transport coefficients are also determined everywhere.

1.4.1 Upwind Interpolation UDS

The idea is to approximate Φ_e with the value at the node upstream. So, depending on the sign of $\vec{U} \cdot \vec{n}$ on the face of reference, it will either be the right or the left one:

$$\Phi_e = \begin{cases} \Phi_P \text{ if } (\vec{U} \cdot \vec{n})_e > 0 \\ \Phi_E \text{ if } (\vec{U} \cdot \vec{n})_e < 0 \, . \end{cases} \tag{1.38}$$

This approximation is equivalent to using a backward forward difference approximation for first derivative, hence the name Upwind Differencing Scheme (UDS). Such interpolation scheme result in being stable, boundedness criterion always satisfied, but numerically diffusive. Consider in fact Taylor series expansion:

$$\Phi_e = \Phi_P + (x_e - x_p) \left(\frac{\partial \Phi}{\partial x} \right)_P + \frac{(x_e - x_p)^2}{2} \left(\frac{\partial^2 \Phi}{\partial x^2} \right)_P + H \, . \tag{1.39}$$

UDS neglect all terms but the first so it is a first order scheme. The first neglected term behave like a diffusive flux:

$$j_e^d = \Gamma_e \left(\frac{\partial \Phi}{\partial x} \right)_e , \tag{1.40}$$

meaning that scalars are diffused normally and parallel to the flow. The coefficient of numerical diffusion is proportional to grid dimension and to mass flux, moreover is magnified for multidimensional oblique to the grid flows. To avoid such inaccuracy, fundamental in case of shocks or rapid changes, mesh must be thickened a lot because the scheme is only first order accurate.

1.4.2 Linear Interpolation CDS

Again in analogy with finite difference differential approximation, linear interpolation can be implemented for FVM: the value at CV-face center is a linear interpolation, with distance as weighting factor, of the values at the centers;

$$\Phi_e = \Phi_E \lambda_e + \Phi_P (1 - \lambda_e) \, , \tag{1.41}$$

with $\lambda_e = (x_e - x_P)/(x_E - x_P)$. Taylor series expansion for Φ_E at point x_P shows that Eq. 1.41 is second order accurate vanishing all first order terms in the truncation error. In fact:

$$\Phi_e = \Phi_E \lambda_e + \Phi_P (1 - \lambda_e) - \frac{(x_e - x_p)(x_E - x_p)}{2} \left(\frac{\partial^2 \Phi}{\partial x^2} \right)_P + H \, . \tag{1.42}$$

Having eliminated the numerical diffusion, the leading term of the truncation error is quadratic and may produce oscillatory solutions. In spite of such oscillatory behavior CDS, being the simplest second order, is one the most widely used differencing scheme.

1.4.3 Self Filtered Centered Interpolation SFCD

SFCD is a scheme that try to get the best of the two precedent schemes, namely the boundedness of the UDS and the accuracy of CDS. It is in fact a central difference scheme with a native filter to remove unphysical estrema whenever they would arise. This filter is basically a local shifting towards an

upwind scheme in such zones. This is done blending CDS and UDS together by a blending factor γ, ranging $0-1$, depending on computed face $\nabla\Phi$:

$$\Phi_e^{SFCD} = \gamma_e \Phi_e^{CDS} + (1-\gamma_e)\Phi_e^{UDS}. \tag{1.43}$$

The blending factor will be as close as possible to unity in case of steep gradients in order to get sharp discontinuities, while in other more uniform zones stability is reached with γ close to zero. The main drawback of this scheme is the introduction of additional non linearity due to the dependency of γ on Φ values, resulting for some steady flow in oscillatory solutions [CD , 2004].

1.4.4 Algebraic Equation

Using the discretization and approximation techniques together, general transported scalar conservation equation Eq. 1.30 can be finally written as an algebraic equation for each nodal point P:

$$A_P \Phi_P + \sum_{nb} A_{nb} \Phi_{nb} = Q_P \ . \tag{1.44}$$

If both the A_P and the A_{nb} are known, this is the case if the flow field is given, the equation is linear and putting together the equation for each node it is possible to built up a matrix representing the differential equation over the entire domain:

$$\underline{\underline{A}} \cdot \underline{\Phi} = \underline{Q} \ , \tag{1.45}$$

in which matrix A is in general quite sparse, full of zeroes because influence for Φ_P is usually limited to a small number of other points, actually for FVM is strictly limited to the number of neighbor nodes. Φ is the solution vector and Q is the source term both of dimension equal to the number of cells.

1.5 Grid

As already mentioned, the solution domain should also be discretized into a finite number of control volumes. The discrete representation of the spatial domain is called grid or mesh. Meshes can firstly be subdivided into:

- Structured mesh: fluid domain of interest is covered with three families of lines (ξ, η, ζ) never intersecting other lines of the same family and normally intersecting all lines of the other families in sequence. Nodal point $P_{i,j,k}$ is defined by the intersection of the three lines (ξ_i, η_j, ζ_k). In this grid arrangement it is always possible to identify neighbor nodes moving on the three lines intersecting at the nodal point.

- Unstructured mesh: fluid domain of interest is covered with a number of finite volumes P_i of arbitrary shape sharing the boundaries. Contrarily to the structured grid where flexibility in gridding complex geometries is limited by the stiff organization of mesh elements, unstructured mesh offers the maximum freedom in defining cells both in terms of shape and location.

A comparison between a typical example of a structured and an unstructured mesh can be done looking at Fig. 1.5 and Fig. 1.6.

On the contrary of FDM, FVM can also deal with unstructured grid. This is fundamental for heat transfer involved simulations where typically non-aerodynamic geometries are encountered. The main difference between FDM and FVM however is that in FVM grids CV are defined via cell boundaries more than cell centers, even if this could be done. Once the boundary surfaces are defined the coordinate of cell center are calculated assuring a higher degree of accuracy in substituting the mean with the center value. Viceversa if centers are defined and boundary calculated, Central Difference Scheme for derivatives are more accurate at CV faces because the face is midway between the two nodes.

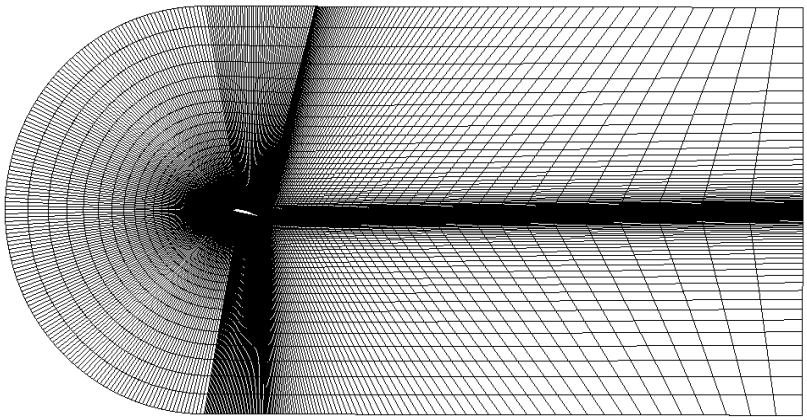


Figure 1.5: Typical structured C grid example.

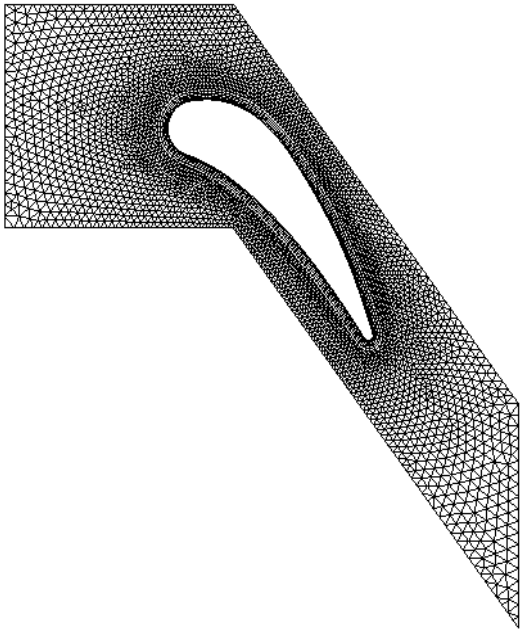


Figure 1.6: Typical hybrid tetrahedral esaedral unstructured mesh example.

A new class of methods, the so called meshless methods, has been proposed in the last years to avoid handling meshes that sometimes can result in being the most expensive step in the entire solution process. The idea is

not to create a mesh but just to store data for the cell centers, actually it is a no-sense to talk about cells when there are no more, so they will be referred to as nodes. An area of influence is hence defined for each node generating a local mesh. Local control volumes can overlap without loosing conservative properties: conservation is respected no matter the shape, dimension and orientation of the control volumes.

1.6 Solution method

Previous sections were dealing with building up the algebraic equations to be solved for a generic scalar Φ transported by a known flow field. It is not however of secondary importance to underline that the flow field cannot be known a priori and scalar quantities for which conservation equations were derived ($\rho, \rho U_i, \rho h_0, p$) are responsible for the development of the flow field itself. So the equations are not independent one from the other but on the contrary different scalar conservation laws result to be tightly coupled. In other words the Navier-Stokes Equations cannot be solved separately but must be treated like a real system meaning that in the conservation equation for scalar Φ_1, all other Φs cannot be considered known but should be treated as variables too. This results in a strong non-linearity of the system, think about convective or viscous dissipation terms for example.

The mechanism used to eliminate such non-linearity is a guess and correct approach leading to an iterative cycle at the end of which the solution will satisfy all the equations at the same time. Starting from an initial guess of all Φs, the system is solved taking the values for Φ_i as constant in the equation for Φ_j. This solution is used to update coefficients of the matrix representing the system and the cycle repeats until solution is reached. As the reader may easily believe it is not so straight forward that such a method really leads to convergence. Many different algorithms can be used to improve convergence properties for this cycle usually called **outer cycle**, most of which separate

the system in equations to be solved sequentially and not contemporary. These are called segregated solvers, in contrast with simultaneous solvers, and will be analyzed in detail later in Sec.3.1.

Once the Navier-Stokes Equations are linearized, again they appear in the final form:

$$\underline{\underline{A}} \cdot \underline{\Phi} = \underline{Q} \tag{1.46}$$

where A is, this time, the sparse coefficient matrix for all the five conservation laws and the eqaution of state, Φ is the solution vector for all variables and Q is the source vector. Linear systems are not difficult to solve, think about Gauss elimination techniques or Cramer rule, but, since typically the number of cells for a CFD simulation varies from some thousands to several millions and CPU time is a very requested and still limited resource, solving in an efficient and fast way it is a must.

Methods for solving linear systems are commonly classified into two main groups: direct methods and iterative methods. Direct methods handle the matrix with linear algebra rules to reduce it in a form from which extraction of final solution can be done quite easily. Every well posed linear system can be solved directly but the quite sparse matrix A (every node depends on the value of not so many other nodes) gives decomposition matrices that are not sparse increasing a lot the computational cost. Iterative methods instead starts from a guessed solution and use the system to successively refine it. If each iteration is cheap and the number of iteration is small enough, this is usually the case in CFD, iterative methods result in being much faster than direct ones. This iterative cycle is called **inner cycle**. The algorithms used in this thesis are the Incomplete Cholesky preconditioned Conjugate Gradient (ICCG), Incomplete Cholesky preconditioned Biconjugate Gradient (BICCG) and the Algebraic Multi Grid (AMG). For details on such methods see Ferziger and Peric [2002], Lacor [2006], Murthy and Mathur [2002].

To avoid instability due to the non-linearity of the system as a consequence of the coupling between the equations, underelaxation techniques are widely used [Patankar, 1980]. The concept standing behind underrelaxation is to "relax" changes in variable values multiplying the increments from the previous value by an underrelaxation factor (more commonly referred to as relaxation factor) α_Φ with values in the interval $[0, 1]$:

$$\Phi^n = \Phi^{n-1} + \alpha_\Phi(\tilde{\Phi}^n - \Phi^{n-1}) \,, \tag{1.47}$$

where $\tilde{\Phi}^n$ on the right hand side in the exact result of the linear system at outer iteration n while Φ^n on the left hand will the best available value for Φ at iteration n. Eq. 1.47 is sometimes referred to as **explicit relaxation**, meaning that matrix A is not modified. It is however possible to express the same relation in an implicit way involving a modification of the matrix itself. Considering the linearized equation for Φ_P:

$$A_P\tilde{\Phi}_P^n + \sum_{nb} A_{nb}\Phi_{nb}^n = Q_P \,, \tag{1.48}$$

and using the Eq. 1.47 it is possible to write a new expression for Φ_P at iteration n:

$$\begin{aligned} \Phi_P^n = \Phi_P^{n-1} + \alpha_\Phi \left(\frac{Q_P - \sum_{nb} A_{nb}\Phi_{nb}^n}{A_P} - \Phi_P^{n-1} \right) \Rightarrow \\ \frac{A_P}{\alpha_\Phi}\Phi_P^n + \sum_{nb} A_{nb}\Phi_{nb}^n = Q_P + \frac{1-\alpha_\Phi}{\alpha_\Phi} A_P \Phi_P^{n-1} \,. \end{aligned} \tag{1.49}$$

This technique is known as **implicit relaxation**. Stability is increased because the diagonal dominance of the matrix A is higher: $\frac{A_P}{\alpha_\Phi}\Phi_P^n > A_P\Phi_P^n$. Unfortunately optimum underrelaxation factors are problem dependant, moving towards lower values to increase stability and to higher values to speed up convergence. A general strategy could be to start with low relaxation factors for early iteration and move to higher values reaching convergence. It is now time to clarify what really convergence is and how it can be established.

Using iterative methods, the most important parameter to check how exactly a prediction satisfy the equation is the residual defined as:

$$\text{Res} = Q - A \cdot \Phi \ .$$

Res is a vector, with cell number size, and it is not a very convenient factor to establish whether or not an equation is converged. The global parameter usually used is:

$$\text{GRes} = \sum_n \text{Res} = \sum_n (Q - A \cdot \Phi) \ , \tag{1.50}$$

where n correspond to the number of cells in fluid domain. Equations are checked separately so 5 different residuals plus the ones for turbulence will be followed. Usually, and this is the case, residuals are normalized with respect to some mean value, trying to account relative errors more than absolute. In this work the normalization factor is defined as follows:

$$\begin{aligned} \Phi_{ref} &= \overline{\Phi} \ , \\ S &= A \cdot \Phi \ , \\ S_{ref} &= A \cdot \Phi_{ref} \ , \\ \text{Norm} &= \sum \left(|S - S_{ref}| + |Q - S_{ref}| \right) \ . \end{aligned}$$

As already hinted, two nested iterative cycles are present: the inner one is directly related to the solution of linear system at iteration n, the outer one regards the algorithms of segregated solvers going from the initial guess at step 0 to actual iteration at time n. Convergence criteria must be set up and controlled for both inner and outer iteration cycles. Outer iterations control parameter is Initial Residuals, while inner iterations are managed by Final Residual:

$$\begin{aligned} \text{InitialResidual} &= \frac{\sum_n (Q_{i0} - A_{i0} \cdot \Phi_{i0})}{\text{Norm}_{i0}} \ , \\ \text{FinalResidual} &= \frac{\sum_n (Q_{im} - A_{im} \cdot \Phi_{im})}{\text{Norm}_{im}} \ . \end{aligned}$$

Chapter 2

OpenFOAM

The aim of this chapter is to introduce the reader into the main features of OpenFOAM as support code for CFD simulations. In particular focus will be posed more on the way it is born and developed than on its practical using. OpenFOAM in fact is not thought to be a ready-to-use software, even if it can be used as a standard simulation package, but to offer a backing to CFD programmers in building their own codes. Being at the core just a C++ library, first a brief introduction to such programming language is necessary. Then OpenFOAM itself is described in detail giving both a "philosophical" introduction and practical examples on how to work with its classes and its applications.

2.1 Introduction to C++

Giving such a small introduction to C++ is intended here as describing the supported methodologies of programming and some of the tools to make them convenient. Such methodologies are technically called programming paradigms. Supporting a programming paradigm does not mean to merely enable such programming style but to provide facilities to make such paradigm convenient to use in terms of ease, safety, efficiency. Support must not be

limited to language facilities to directly use the paradigm, but must be extended at compile-time and run-time checks.

The most common programming paradigms are briefly introduced:

- **Procedural Programming**: the focus is on the processing,i.e. the algorithm needed to perform the desired computation. Languages provides facilities for passing and returning the "most suitable" arguments to the "most suitable" functions in the "most suitable" way. Discussions regard the concept of "most suitable".

- **Modular Programming**: also known as the data-hiding principle, is concerned with subdiving programs into smaller modules data are hidden within. This need reflects a constant increase in program size.

- **Data Abstraction**: this paradigm is dealing with user-defined data types. Data types suitable for the problem are newly defined providing a full set of operations for each type. Connected and fundamental for doing it, are the concepts of operator overloading and polymorphism, also called late binding.

- **Object-Oriented Programming**: the concept introduced with such a paradigm is the concept of inheritance in connection with class hierarchy. Base classes are defined as the common structure and then specialized into derived classes. This can actually be done even with previous paradigm but now the possibility of sharing (inherit) members from base to derived class is introduced.

- **Generic Programming**: the aim is to parameterize algorithm in such a way they can work for a variety of suitable types and data structures. Generic Programming introduces the concept of containers: classes that can hold a collection of elements of different types.

C++ was designed to support data abstraction, object-oriented and

generic programming in addition to traditional C programming techniques such as procedural and modular programming. It was not meant to force one particular programming style upon all users [Stourstroup, 1997].

It would be better to clarify some of the above introduced concepts, in particular: polymorphism, inheritance and containers.

Connected with data abstraction paradigm is the concept of operator overloading meaning that the same function can actually have different meaning for different classes. Polymorphism is the capability of deciding at run-time which of the several overloaded member functions to call.

Inheritance is the ability of one class to inherit capabilities or properties from another class. To fix ideas an example is given. Consider the class of the *Human-beings*, it has some specific properties (two arms walk, intelligence, etc) plus some others (giving birth to live young, warm blood, etc) derived from the class *Mammal* which is belonging to. Equally Mammals have inherited some of their properties to other base classes such as *Vertebrate* or *Animals*. In such a way a sort of chain is built to correctly link a property to the right class.

Thinking about containers the best example is a list. Lists are basically a collection of elements stored in a row. It is possible to define a list of integer as well as a list of people, meaning that the class definition is not bound by the type of objects it is listing. It is common to refer to containers as templates. Templates are a compile-time mechanism so that their use result in no run-time overhead compared to hand written code [Davis, 2005].

2.2 What is OpenFOAM

The OpenFOAM (Field Operation And Manipulation) code is an object-oriented numerical simulation toolkit for continuum mechanics, written in C++ language, released by OpenCFD Ltd (`http://www.opencfd.co.uk`) [Ope, c]. It is so capable to support all the above discussed features typical of

C++ programming: it enables the construction of new types of data specific for the problem to be solved (i.e. a virtual class for turbulence Model with virtual functions such as ε, k, μ_t, etc ...), the bundling of data and operations into hierarchical classes preventing accidental corruptions (i.e. a base class for storing mesh data and a derived class for acceding them), a natural syntax for user defined classes (i.e. operator overloading) and it easily permits the code re-use for equivalent operations on different types (i.e. templating) [Jasak, 1996, Jasak et al., 2004, Juretic, 2004].

Let's see a little more in detail what are the specificity of OpenFOAM in helping CFD programmers. First and most important thing is that the toolkit implements operator-based implicit and explicit second and fourth-order Finite Volume (FV) discretization in three dimensional space and on curved surface.

Differential operators can be treated like finite volume calculus (fvc) or finite volume method (fvm) operators.

The first approach performs explicit derivatives returning a field, the second one is an implicit derivation converting the expression into matrix coefficients. The idea standing behind is to think about partial differential equations in terms of a sum of single differential operators that can be discretized separately with different discretization schemes. At the moment the following, self explaining, implicit differential operator are defined:

- `fvm::ddt` $= \frac{\partial}{\partial t}$,
- `fvm::d2dt2` $= \frac{\partial^2}{\partial t^2}$,
- `fvm::div` $= \sum_i \frac{\partial}{\partial x_i}$,
- `fvm::laplacian` $= \sum_i \frac{\partial^2}{\partial x_i^2}$.

In addition implicit treatment of source terms is done by: `fvm:Sp` and `fvm:SuSp`.

Explicit equivalent for the previous operators are defined and furthermore other common operators such as curl, gradient, etc ... are implemented.

Building different types of partial differential equations is now only a matter of combining in a different way the same set of basic differential operators. Just to give an example of the capability of such a top-level code, let's consider a standard equation like momentum conservation:

$$\frac{\partial \rho \vec{U}}{\partial t} + \nabla \cdot \left(\rho \vec{U} \vec{U} \right) - \nabla \cdot \left(\mu \nabla \vec{U} \right) = -\nabla p \, . \tag{2.1}$$

It can be implemented in an astonishingly almost natural language which is ready to compile source C++ code:

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

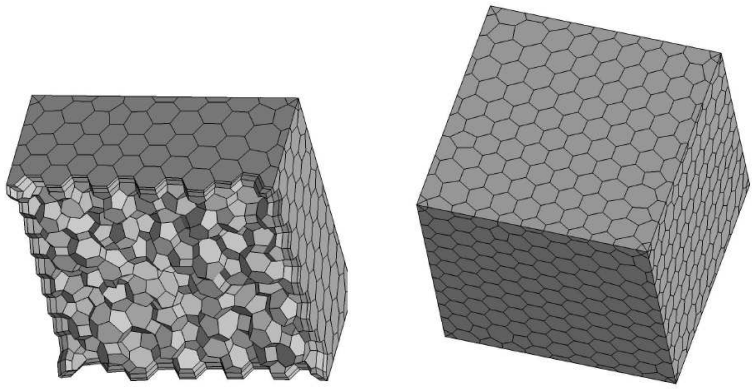
letting programmers concentrate their efforts more on the physics than on programming.

Such example clearly shows that OpenFOAM programmers do not think in terms of cells or faces but in terms of objects (U, rho, phi, etc ...) defined as a field of values, no matter what dimension, rank or size, over mesh elements such as points, edges, faces etc. Just to fix ideas the velocity field for example is defined at every cell centroid and boundary-face centers, with its given dimensions and the calculated values for each direction, and represented by just a single object `U` of the class `GeometricField` see Sec. 2.4.

Another important feature allowed by object programming is the dimensional check, physical quantities objects are in fact constructed with a

reference to their dimensions and so only valid dimensional operations can be performed. Avoiding errors and permitting an easier understanding, come directly as a consequence of an easier debug.

Next thing to be mentioned is the flexibility in handling different types of meshes. OpenFOAM native grid engine can, in fact, handle every mesh of arbitrary polyhedra bounded by arbitrary polygons, see Fig. 2.1.



(a) (b)

Figure 2.1: Polyhedral mesh example.

In addition, even if never directly tested by the author, automatic mesh motion and runtime topological changes such as attach/detach boundaries, cell layer addition/removal are allowed.

It has already pointed out, how OpenFOAM was thought not really to be a ready to use code but to be as much flexible as possible in defining new models and solvers in the simplest way. Its strength in fact stands in being open not only in terms of source code but, what's more, in its inner structure and hierarchical design, giving the user the opportunity to fully extend its capability. Moreover, the possibility of using top-level libraries containing a set of models for the same purpose which refer to the same interface, guarantees programmers for smooth and efficient integration with the "built-in" functionalities. New models can be added to the appropriate

model table at link-time and become available in the same manner as the supplied models.

Most of the selections necessary to set up calculations are done at run-time, meaning that options can change while the code is running. For further information about how to use and how to program OpenFOAM see Ope [a,b].

2.3 Matrix structure in OF

It is interesting to describe how OpenFOAM is building and storing the matrix A of coefficients for one of the discretized equations. This array is decomposed in three parts: the diagonal coefficients, the upper triangular matrix u and the lower triangular matrix l. Off diagonal coefficients represent the influence of neighboring cells. Each face is shared by two different cells one of which is the owner and the other is properly called neighbor. The owner cell for a general face is the one with lower cell index. The lower matrix is so returning coefficients for all the owner cells while the upper one for the neighbors. Just to clarify consider line i of array A, it corresponds to equation for cell of index i. The number of non-zero coefficients on this line equals the number of faces plus the diagonal coefficient that multiply the solution at cell i itself. On the left side of diagonal coefficient (rows j from 1 to i-1), coefficients for faces between cell i and lower index are stored. This means that l array is actually storing coefficients for owner cells of faces bounding cell i not owned by cell i itself. At the contrary, coefficients for cells who share boundaries owned by cell i, once again the cells with index higher than i, are stored in the u array. OpenFOAM programmers usually works with FVM, so a special class of matrix `fvMatrix` in which the influence is only limited to adjacent cells is defined and normally used. It is also possible however to work with the base class `lduMatrix` where this limitation is not present; this is very important if FEM want to be used.

2.4 Working with main classes in OF

Just to give an example of how object oriented programming in OpenFOAM is working, the class `GeometricField` is shown and described in its structure and interaction with other classes. First of all, it would be better introduce the class with its Unified Modeling Language UML graphical representation in Fig. 2.2.

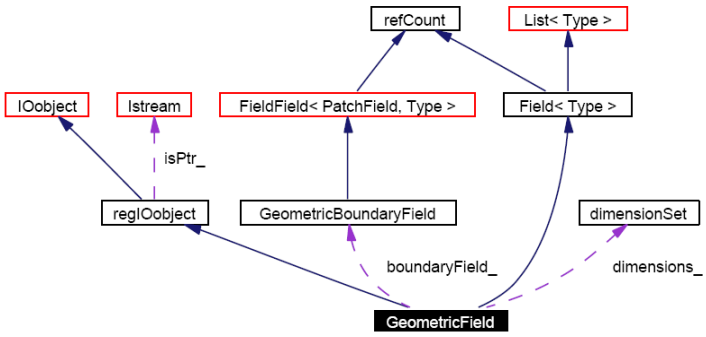


Figure 2.2: UML diagram for GeometricField class.

To interpret these graphs one should read it in this way: a box represents a class, a solid arrow stands for public inheritance and a dashed arrow indicates usage, with the edge of the arrow labeled with the variable responsible for the relationship. Meaning that `GeometricField` has a `boundaryField` and is a `Field<Type>`. As above hinted at, `GeometricField` could represent any tensor field; it consists of an `internalField` and a `boundaryField`. The first one, belonging to `Field<Type>` class, stores a list of values of appropriate tensor rank for each computational point, the second, namely a `GeometricBoundaryField`, stores boundary values subdiving it into one `patchField` for each boundary surface. Reference to dimensions is stored in the `dimensionSet` that basically is the vector of the exponents for the fundamental units. It could be of 5 $[kg, m, s, K, mol]$ or 7 $[kg, m, s, K, mol, A, cd]$

elements.

`GeometricField` is a template and gives a good example of generic programming: this virtual class in fact contains Fields of scalar, vector and tensor type defined on surfaces and volumes. One can in fact define something like a `surfaceScalarField` as well as a `volTensorField`.

Now that a brief introduction to one of the most common classes in OpenFOAM has been done, next step will be to see how programmers actually works with it.

2.5 Examples

We already said that OpenFOAM is primarily a C++ library used to create executables, usually called applications. Such applications can be divided into two main groups: solvers, designed to solve specific continuum mechanics problems, and utilities, to perform tasks involving data manipulation. In addition, programmers can use OpenFOAM native classes to implement new user-defined classes.

A simple post processing utility for the calculation of wall heat transfer coefficient, is presented as example, see appendix A for the complete code of the main file is wallHTC.C.

Before the declaration of the main (line 8) several libraries should be included (lines 1-5) in order to get access at their members. Then the case is started (line 13), times at which the solution exists are read (lines 15-23) and the mesh is created (line 25).

A cycle over the times present is started (line 28) and fields are initialized calling a header file called createFields.H (line 36), see appendix B for the source code. In such file, thermo physical model is loaded calling an autopointer to the basicThermo (line 1). basicThermo give access to all state variables and so `volScalarField` are defined for enthalpy, temperature and density as well as for fluid properties such as thermal diffusivity and con-

stant pressure specific heat (lines 7-40). Velocity field is read as an I/O object (lines 46-57) and a mass flux over cell boundaries are calculated (line 59). Again an autopointer is used to call for the turbulence model (lines 61-70) and finally I/O objects for reading the results are defined for wall heat flux and heat transfer coefficient (lines 73-99). Wall adiabatic temperature, calculated as a results of an adiabatic simulation, is read from results directory (lines 14-25). Once all the used fields are created the main is calculating heatflux over all cell surfaces as the normal to the face gradient of the enthalpy multiplied by interpolated thermal diffusivity (lines 38-40). Wall boundary are then selected with an if cycle (line 44-46) and the value of heatflux on those patches is passed to wallHeatFlux (line 48). Heat transfer coefficient is calculated following the definition (line 53), manipulating data only on wall boundaries. From HTC also Nusselt number is calculated (line 58) and some string are plotted to show some of the output for such computation (lines 62-83). Before closing the cycle the newly defined output scalar fields are written in the solution directory (lines 88-92).

Chapter 3

All Mach Pressure Based Solver

The problem of developing a suitable solver for heat transfer predictions in turbomachinery is not an easy task. This chapter is aiming at giving a general overview on the specific needs for such a solver, then various classes of solvers are presented and criticized to explain the whys of the choice of a pressure-based solver. Hence a detailed derivation of standard algorithms for this class is given. Finally the developed solver is presented with particular attention to the aspects that were found of much trouble for the programmers.

3.1 Segregated solvers

During the derivation of the set of conservation equations, it has been repeated many times that equations are coupled meaning that coefficients of one depend on solution of another. It has been underlined that such coupling results in non-linear systems and the resolution of this non-linearity is left to the iterative process. What has not been specified yet is how the system is solved. Considering for example the final expression derived for

the discretized Navier-Stokes Equations:

$$A_{[m\times n,m\times n]}\Phi_{[m\times n]} = Q_{[m\times n]} \ , \tag{3.1}$$

where $\Phi_{[m\times n]}$ is a vector of dimensions equal to the number of cells n times the number of unknowns m.

For computational costs reasons, this system is usually not solved entirely but sequentially, in other words the system matrix is decomposed into one smaller matrix for each equation and such matrices are solved in sequence for the scalar of reference:

$$A_{\rho[n,n]}\Phi_{\rho[n]} = Q_{\rho[n]} \ , \tag{3.2}$$

$$A_{u[n,n]}\Phi_{u[n]} = Q_{u[n]} \ , \tag{3.3}$$

$$A_{v[n,n]}\Phi_{v[n]} = Q_{v[n]} \ , \tag{3.4}$$

$$A_{w[n,n]}\Phi_{w[n]} = Q_{w[n]} \ , \tag{3.5}$$

$$A_{p[n,n]}\Phi_{p[n]} = Q_{p[n]} \ , \tag{3.6}$$

$$A_{h[n,n]}\Phi_{h[n]} = Q_{h[n]} \ . \tag{3.7}$$

In this way it is clear that the coupling between the different equations must be treated with another strategy. An algorithm to iteratively link such discretized equations is needed in order to reach convergence. It is easy to guess that a number of different algorithm has been proposed to organize the steps in which the process should be parted.

3.2 The pressure velocity coupling

The two main classes in which standard segregated solvers can be classified are: density-based and pressure-based solvers. Each of these classes is naturally applied to a certain class of flows: high Mach compressible flows for the density-based and low Mach almost incompressible for the pressure-based.

The different nature of conservation equations in case of subsonic or supersonic flows makes the use of these classes outside the range of Mach

number of reference a difficult task. Unsteady compressible Navier-Stokes Equations are in fact parabolic-hyperbolic in nature, but their incompressible counterparts are of elliptic parabolic type.

However, heat transfer in turbomachinery applications, and other well known cases of no interest here, involve flows usually covering a wide range of Mach regimes. In particular, it usually happens that different Mach conditions simultaneously arise in the same domain. Such situations makes the accurate solution of viscous flows governing equations a complex task since this kind of flows does not fall completely into any of the categories before cited. In order to solve such flows specific solvers should be created trying to enlarge range of applicability to the maximum, at the best to All-Mach flows.

The most famous method proposed to change behavior of the incompressible Navier-Stokes Equations and fit them over the compressible shape is the so-called artificial compressibility technique: a fake time derivative is added to the incompressible conservation equations rendering them hyperbolic and solvable with standard time-marching techniques developed for the compressible form of the equations. Such methods are applied to steady state solution and require a different set of variables in the compressible and incompressible flow regime. That is why a different path to solution has been searched for.

3.3 Problems associated with density based solvers

Conservation equations derived in chapter 1 as well as the other later derived in chapter 4 can be cast into a general form as:

$$\frac{\partial \rho \Phi}{\partial t} + (\nabla \cdot \vec{J}) = Q \ , \tag{3.8}$$

where the total flux $\vec{J}$ is composed by a convective and a diffusive term:

$$\vec{J} = \underbrace{\rho\vec{U}\Phi}_{convective} - \underbrace{\Gamma\nabla\Phi}_{diffusive} \quad . \tag{3.9}$$

In addition an equation of state to describe the relationship between the various thermodynamic variables is needed. This set of non-linear coupled equation should be solved for the unknowns ρ, $\vec{U}$, p, h.

It is so customary to associate each of the equations to one basic variable: density for continuity, velocity components for momentum, enthalpy for energy. In addition the equation of state can be considered as the equation for pressure. At very low Mach number however, the pressure and the density become very weakly related and in the idealized limit of incompressible flows, the density is completely decoupled from the pressure. For such a reason neither pressure nor density can be directly associated with the incompressible continuity equation and it assumes the role of a compatibility condition on the velocity field.

For a sequential solution of the equations a mechanism to couple continuity and momentum should be searched, otherwise it is also possible to use the fully compressible equations even in incompressible flows. In reality in fact, all fluids are compressible and the pressure is always related to the density even though the relationship may become very weak. However, because equations are solved numerically, constraint are imposed by numerical considerations: truncation and round-off errors.

The perfect gas law for a nearly incompressible flow in which the speed of sound can be considered constant, states that:

$$p = \rho RT = \frac{c^2}{\gamma}\rho \Rightarrow \Delta p = \frac{c^2}{\gamma}\Delta\rho \; . \tag{3.10}$$

Introducing reference values it is possible to rewrite Eq. 3.10 in a non-dimensional form:

$$\frac{\Delta p}{\rho_0 u_0^2} = \frac{1}{\gamma M a_0^2}\frac{\Delta\rho}{\rho_0} \; . \tag{3.11}$$

This expression clearly points out, that even though changes in density may become small or infinitesimal, as long as Ma number tends to zero too their ratio is finite and changes in pressure may result large. If sensibility in solving both terms is established to be:

$$\frac{\Delta p}{\rho_0 u_0^2} > 10^{-n} , \tag{3.12}$$

$$\frac{\Delta \rho}{\rho_0} > 10^{-m} , \tag{3.13}$$

then it is imposed the Mach number to be grater than $10^{-\frac{m+n}{2}}$. In addition for $Ma = 0$ there is a singularity point for the Navier-Stokes Equations, rendering them improperly scaled in the limit of incompressibility.

The above discussion highlights the problems associated with the use of density as a primary variable for computing low Mach flows or mixed compressible and incompressible flows [Karki, 1986].

Development of a computational scheme valid for the whole range of Mach number, needs a switch to pressure as a primary independent variable in preference to density.

3.4 SIMPLE Algorithm

This section is written to illustrate in detail one of the segregated solver algorithm introduced in Sec. 3.1. The algorithm presented here it the original version of the Semi-Implicit Method for Pressure Linked Equations (**SIMPLE**) published for the first time in Patankar and Spalding [1972] and nowadays very well known worldwide. This old version was derived for incompressible or at maximum weakly compressible flows and expansion to fully compressible flows will further be investigated. The main idea is to convert continuity equation into an equation for pressure or better a pressure corrector and to use a guess and correct procedure to reach solution: pressure corrector equally zero everywhere. Since continuity contains discrete face velocities a

connection to relate such velocities to a pressure field is needed: the direct link is the momentum equation.

Consider the discretized x and y momentum equation with notation referred to Fig. 3.1:

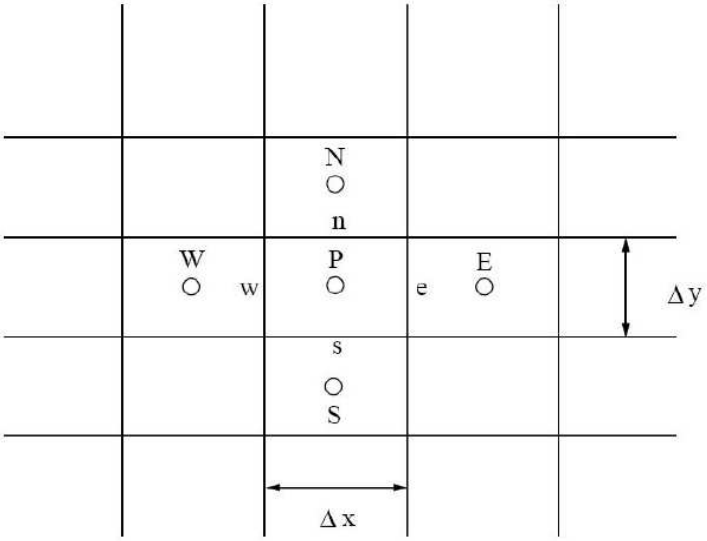


Figure 3.1: Bidimensional control volume.

$$a_e u_e = \sum a_{nb} u_{nb} + (p_P - p_E)\Delta y + Q_e \ , \tag{3.14}$$

$$a_n v_n = \sum a_{nb} v_{nb} + (p_S - p_N)\Delta x + Q_n \ . \tag{3.15}$$

To initialize the SIMPLE calculation a pressure field p^* is guessed, hence Eq. 3.14 and Eq. 3.15 are solved with such pressure distribution to obtain a first guess value for velocity u^* and v^*:

$$a_e u_e^* = \sum a_{nb} u_{nb}^* + (p_P^* - p_E^*)\Delta y + Q_e \ , \tag{3.16}$$

$$a_n v_n^* = \sum a_{nb} v_{nb}^* + (p_S^* - p_N^*)\Delta x + Q_n \ . \tag{3.17}$$

A pressure corrector is then defined as the difference between the correct and

the guessed pressure, and similarly is done for velocity correctors too:

$$p' = p - p^* , \tag{3.18}$$

$$u' = u - u^* , \tag{3.19}$$

$$v' = v - v^* . \tag{3.20}$$

Subtraction of Eq. 3.16 from Eq. 3.14 and of Eq. 3.17 from Eq. 3.15 gives two equations for pressure and velocity correctors:

$$a_e u'_e = \sum a_{nb} u'_{nb} + (p'_P - p'_E)\Delta y , \tag{3.21}$$

$$a_n v'_n = \sum a_{nb} v'_{nb} + (p'_S - p'_N)\Delta x . \tag{3.22}$$

Given that correctors tend towards zero as long as solution is converging, dropping terms such as $\sum a_{nb} u'_{nb}$ and $\sum a_{nb} v'_{nb}$ is not giving any error when convergence is reached. This is however the biggest approximation of the SIMPLE algorithm and its consequences will be discussed later in this section. Rewriting Eq. 3.21 and Eq. 3.22, one obtains:

$$u'_e = (p'_P - p'_E) d_e , \tag{3.23}$$

$$v'_n = (p'_S - p'_P) d_n , \tag{3.24}$$

where $d_e = \frac{\Delta y}{a_e}$ and $d_n = \frac{\Delta x}{a_n}$. Thus far only momentum equations has been considered, continuity should be used to impose mass conservation:

$$\begin{aligned} &[F_e - F_w] + [F_n - F_s] = 0 \Rightarrow \\ &[(\rho u A)_e - (\rho u A)_w] + [(\rho v A)_n - (\rho v A)_s] = \\ &\qquad [\rho_e u_e - \rho_w u_w]\,\Delta y + [\rho_n v_n - \rho_s v_s]\,\Delta x = 0 . \end{aligned} \tag{3.25}$$

The face mass flux F_i is then split in two parts corresponding to the two components of the exact solution of the velocity u^* and u':

$$[F^*_e - F^*_w] + \left[F'_e - F'_w\right] + [F^*_n - F^*_s] + \left[F'_n - F'_s\right] = 0 . \tag{3.26}$$

Eq. 3.23 and Eq. 3.24 can be inserted in Eq. 3.26 to give:

$$F_e^* + \rho_e d_e \Delta y \left(p'_P - p'_E\right) - F_w^* + \rho_w d_w \Delta y \left(p'_P - p'_W\right) + F_n^* + \rho_n d_n \Delta x \left(p'_P - p'_N\right) - F_s^* + \rho_s d_s \Delta x \left(p'_P - p'_S\right) = 0 \ , \quad (3.27)$$

that rearranged gives the equation for the pressure corrector:

$$A_P p'_P = \sum_{nb} a_{nb} p'_{nb} + b \ , \quad (3.28)$$

where:

$$a_E = \rho_e d_e \Delta y \ , \quad (3.29)$$

$$a_W = \rho_w d_w \Delta y \ , \quad (3.30)$$

$$a_N = \rho_n d_n \Delta x \ , \quad (3.31)$$

$$a_S = \rho_s d_s \Delta x \ , \quad (3.32)$$

$$A_P = \sum_{nb} a_{nb} \ , \quad (3.33)$$

$$b = F_w^* - F_e^* + F_s^* - F_n^* \ . \quad (3.34)$$

At this point all the necessary equations have been deduced: they should be included into an organic iterative process to obtain a solution. The overall algorithm foresees the following steps:

1. Guess a pressure field p^*.
2. Discretize and solve the momentum equations using the guessed value p^* for the pressure terms. Eq. 3.16 and Eq. 3.17 yield the u^* and v^*.
3. Find the pressure correction source term b calculating mass imbalance with Eq. 3.34.
4. Discretize and solve the pressure correction equation: Eq. 3.28.
5. Calculate velocity corrections using Eq. 3.23 and Eq. 3.24.
6. Correct pressure and velocity fields: Eq. 3.18, Eq. 3.19 and Eq. 3.20.

7. Solve discrete equations for each scalar Φ using the continuity satisfying velocity field for the convection term.

8. Check convergence: IF satisfied stop, ELSE use p as a new guess p^* and start again from 2.

It is now time to come back to Eq. 3.21 to clarify what are the consequences of dropping the $\sum_{nb} a_{nb}$. It has already been underlined how at convergence nothing is changed, so changes are limited on the rate of convergence.

Such dropping in fact places the entire burden of correcting velocities upon pressure corrector. This will lead to a velocity field still satisfying continuity but also to an overpredicted pressure field. The pressure correction equation is indeed susceptible to divergence unless some under-relaxation is used during the iterative process. Implicit underrelaxation is usually preferred because it also increase the diagonal dominance of the matrix, but this cannot be case. Being a corrector the history of convergence of p' is without any particular meaning, in other words values of p' at previous iteration do not resolve the same, or almost the same, equation, as a consequence p'^{n-1} can be very different from p'^{n} and implicit relaxation will violate continuity. What is done to avoid this obstacle is to relax the new values for pressure and not the corrector:

$$p^{new} = p^* + \alpha_p \, p' \; . \tag{3.35}$$

To avoid overpredictions of pressure, velocities are also relaxed. A correct choice of under-relaxation factors α is essential for cost-effective simulations. Too large a value may lead to oscillatory or even divergent solution while small values comport extremely slow convergence. Unfortunately, the optimum values of under-relaxation factors are flow dependent and must be sought on a case-by-case basis [Malalasekera and Versteeg, 1995, Murthy and Mathur, 2002].

Other variants of the same method have been proposed in the course of years:

- **SIMPLER** (SIMPLE-Revised): use a discretized equation for pressure to obtain p^* from an initial guess of velocity v^{**} and pressure p^{**} fields [Patankar, 1980].
- **SIMPLEC** (SIMPLE-Consistent): use the same algorithm of SIMPLE but with a better approximation for Eq. 3.21 and Eq. 3.22 [Doormal and Raithby, 1984].
- **PISO** (Pressure Implicit with Splitting Operators): add a second corrector step to the standard SIMPLE algorithm meaning that $p^{**} = p^* + p'$ is only a starting guess for the second corrector step [Issa, 1986].

At this point the method is completely described, it is not however secondary to better precise how to treat difficulties arising with the use of such a solver. In particular, more attention must be paid on the problem of checkerboarding solutions and extension to fully compressible flows.

3.5 Pressure Checkerboarding

As you may already have noticed this method requires values both on cell (capital letters) and face (minuscole) centers so it can be directly applied only on staggered grids, fluid domains on which velocities and pressures are computed at different locations. Staggered approach is however quite heavy in terms of computer memory to be allocated and complicated to organize. If the number of elements increase to actual standard size for CFD simulation of industrial interest, this approach should be left for a co-located approach.

This is indeed not the case for most of standard CFD codes, so a solution must be found in order to be able to use such method. In particular, using OpenFOAM cell centered approach, all face values such as ρ_e, u_e, F_e

are unknown and cell values should be interpolated with some interpolation scheme. With a variety of different interpolation schemes available in OpenFOAM (centered, upwinded, TVD, NVD schemes, etc ...), the choice for velocity and density has been left to the user. Some words more should be spent on face flux (advective velocities) interpolation scheme. The easiest solution could be to use a linear interpolation to obtain values on faces and still apply the above derived method:

$$F_e = \frac{F_P + F_E}{2} \ . \tag{3.36}$$

Unfortunately this approach is not bringing any good, in fact it allows for non physical checkerboarding solutions.

Checkerboarding profiles are due to the fact that linear interpolation brings continuity, i.e. Eq. 3.25, to be independent from the value of velocity on the cell of reference, in other words conservation of mass at cell P depends only on neighboring values, as a consequence odd and even nodal values result to be decoupled:

$$\sum F_i = F_e - F_w + F_n - F_s = \frac{1}{2}\left(F_E - F_W + F_N - F_S\right) \ . \tag{3.37}$$

If such a pattern is satisfying momentum equation too, the checkerboarding would persist in the final solution; unfortunately this is the case. To demonstrate it, take the x-direction momentum equation Eq. 3.14 and make a balance over cell P:

$$u_P = d_P(p_w - p_e) + ... \tag{3.38}$$

$$= d_P\left(\frac{p_W + p_P}{2} - \frac{p_E + p_P}{2}\right) + ... = d_P\left(\frac{p_W - p_E}{2}\right) + ... \tag{3.39}$$

The dependency of u_P on p_P, using linear interpolation, is canceled and velocity is only "feeling" neighboring pressure values. This means that velocity and pressure at the same location do not influence each other, as a consequence checkerboarding patterns of velocity and pressure will satisfy momentum equation.

Dramatically both momentum and mass conservation equation are solved by the checkerboarding pattern and so once it is formed it will never smooth. Even if difficultly perfect checkerboarding profiles for velocity or pressure arise, due to irregularities in the mesh, boundary conditions or physical properties, such tendency often shows itself in unphysically wiggly fields.

To avoid this unacceptable behavior two methods are proposed in literature:

- use a staggered grid: fluid domain discretization on which velocities and pressures are computed at different locations,
- use a co-located grid but a different interpolation method for face mass fluxes.

First approach takes advantage of storing values for velocity and pressure precisely at the points required for the implementation and no interpolation is needed. Primary drawback for the staggered grid arrangement is the increment of metric complexity due to the use of different grid for different variables. This method results as a consequence to be very expensive computationally speaking for curvilinear coordinates and unapplicable to unstructured mesh, so in practise it has been abandoned in the course of years.

Second approach is preventing from checkerboarding expressing face velocities in terms of adjacent cell pressure rather than alternate values. Even if more complex to understand, this approach is less dependent on geometries and gridding strategies and moreover it is the only chance to continue using a co-located code. Advanced interpolation techniques are so computed in order to avoid such checkerboarding on co-located grids.

3.6 Co-located grid: Rhie - Chow interpolation

The interpolation technique named after Rhie and Chow [1983], and also known as momentum interpolation, has been used in the solver. Substantially it consists in defining a pseudo velocity field (on cells) excluding the pressure gradient term to the original formulation of velocity, hence such pseudo velocity is interpolated linearly and finally the pressure gradient is evaluated directly on the face and summed. A monodimensional domain has been considered in the following derivation. It begins with the definition of the pseudovelocity:

$$u_P^0 = u_P + \left(\frac{dl}{A}\nabla p\right)_P = u_P + \frac{(p_E - p_W)}{2A_P} \,, \tag{3.40}$$

then interpolate it linearly on face centers

$$u_e^0 = \frac{u_P^0 + u_E^0}{2} \,, \tag{3.41}$$

to finally reconsider pressure contributions with gradients computed precisely on faces:

$$u_e = u_e^0 - \left(\frac{dl}{A}\nabla p\right)_e = u_e^0 - \frac{(p_E - p_P)}{2A_e} \,. \tag{3.42}$$

Substituting Eq. 3.40 into Eq. 3.41 and into Eq. 3.42, a final expression for face velocity is obtained:

$$u_e = \frac{(u_P + u_E)}{2} + \frac{1}{2}\left(\frac{(p_E - p_W)}{2A_P} + \frac{(p_P - p_{EE})}{2A_E}\right) - \frac{(p_E - p_P)}{2A_e} \,. \tag{3.43}$$

Similar expressions may be derived for the other faces.

Putting together such expressions and summing mass fluxes F_i, it is clear that explicit dependance on cell values do not cancel out as with linear interpolation. Continuity and momentum equation now do feel pressure at point P meaning that wavy gradients are smoothed and so checkerboarding is avoided.

It is interesting to see how the behavior of continuity equation changes with the introduction of this interpolation also known as added dissipation

scheme. Rewrite Eq. 3.43 considering $A_e = A_E = A_P$, this happens in case of uniform grid and uniform flow:

$$u_e = \frac{(u_P + u_E)}{2} + \frac{1}{4A_P}\left((p_E - p_W + 2p_P) + (p_P - p_{EE} - 2p_E)\right) \ . \quad (3.44)$$

Writing down the Taylor series expansion for pressure and reorganizing:

$$\left(\frac{\partial^2 p}{\partial x^2}\right)_P = \frac{(p_W - p_E - 2p_P)}{\Delta x^2} + O(\Delta x^2) \ , \quad (3.45)$$

so

$$\begin{aligned} u_e &= \frac{(u_P + u_E)}{2} - \frac{1}{4A_P}\left(\left(\frac{\partial^2 p}{\partial x^2}\right)_P - \left(\frac{\partial^2 p}{\partial x^2}\right)_E\right)\Delta x^2 = \\ &= \frac{(u_P + u_E)}{2} - \frac{1}{4A_P}\left(\frac{\partial^3 p}{\partial x^3}\right)_e \Delta x^3. \end{aligned} \quad (3.46)$$

Substituting this expression into continuity equation for constant density:

$$\begin{aligned} u_e - u_w &= \frac{(u_E - u_W)}{2} - \frac{1}{4A_P}\left(\frac{\partial^3 p}{\partial x^3}\right)_e \Delta x^3 + \frac{1}{4A_P}\left(\frac{\partial^3 p}{\partial x^3}\right)_w \Delta x^3 = \\ &= \frac{(u_E - u_W)}{2} - \frac{1}{4A_P}\left(\frac{\partial^4 p}{\partial x^4}\right)_P \Delta x^4 = 0 \ , \end{aligned} \quad (3.47)$$

it comes out that a fourth order derivative is added to continuity. Derivative terms of even order are known in literature as dissipation terms, as a consequence a more diffusive behavior is expected from this equation [Anderson, jr., 1995]. More diffusion means more influence of the center value on its direct neighbors sweeping away all unphysical wiggles.

With the implementation of such interpolation scheme the problem of checkerboarding patterns is solved and the same overall algorithm can be applied also in this case of co-located variables. For a better reference see also Davidson [2005], Murthy and Mathur [2002], Mangani [2006].

Before going through the derivation of a different pressure correction equation valid for compressible fluid flows, it is better to specify how to use values of pressure and velocity known at step 6 of the SIMPLE algorithm, for solving conservation equations for the other quantities. Solution of passive

scalars are better computed via the use of face flux, on which continuity is respected, more than cell velocity, on which only momentum is exactly satisfied. If this advice is followed in fact, conservation equations are perfectly conservative.

3.7 Including compressibility effects

The above derived method is valid for incompressible or weakly compressible flows, it is possible however to extend it to fully compressible flows introducing a compressibility factor into the equation for the pressure corrector. To expand the proposed method to compressible flows dependence of density on pressure should be explicitly pointed out and included in the equation for the pressure corrector. This feature is essential in extending applicability of this method to flows at high Mach: outside the weak compressibility zone. Several authors in the course of years have worked on this problem, a selection of articles for further reference is: Karki [1986], Peric et al. [1993], Karki and Patankar [1989], McGuirk and Page [1990], Lien and Leschziner [1993], Moukalled and Darwish [1999], Politis and Giannakoglou [1996], Rhie [1989], Shyy and Braaten [1988].

Let's begin defining density, on the shape of Eq. 3.18, as the sum of a guessed and a corrector density:

$$\rho = \rho^* + \rho' \ . \tag{3.48}$$

Using a compressibility factor Ψ defined as:

$$\Psi = \frac{\partial \rho}{\partial p} = \frac{1}{RT} \ , \tag{3.49}$$

it is possible to write:

$$\rho' = \Psi p' \ . \tag{3.50}$$

As a consequence:

$$F_e = (\rho u)_e A_e = (\rho^* + \rho')_e (u^* + u')_e A_e = \\ (\rho^* u^*)_e A_e + \rho^*_e u'_e A_e + \rho'_e u^*_e A_e + \rho'_e u'_e A_e \ , \quad (3.51)$$

that, dropping last term because infinitesimal of second order and substituting ρ' with Eq. 3.50 (upwinding is used in the following), gives:

$$F_e = F^*_e + \rho^*_e u'_e A_e + \Psi p'_P u^*_e A_e \ . \quad (3.52)$$

Repeating the same reasoning for all faces a different pressure correction equation is obtained:

$$\begin{aligned} & F^*_e + \rho^*_e d_e \Delta y \left(p'_P - p'_E\right) + \Psi p'_P u^*_e \Delta y \\ & - F^*_w + \rho^*_w d_w \Delta y \left(p'_P - p'_W\right) - \Psi p'_W u^*_w \Delta y \\ & + F^*_n + \rho^*_n d_n \Delta x \left(p'_P - p'_N\right) + \Psi p'_P u^*_n \Delta x \\ & \qquad - F^*_s + \rho^*_s d_s \Delta x \left(p'_P - p'_S\right) - \Psi p'_S u^*_s \Delta x = 0 \ . \end{aligned} \quad (3.53)$$

Reducing to the same notation

$$A_P p'_P = \sum_{nb} a_{nb} p'_{nb} + b \ , \quad (3.54)$$

where:

$$\begin{aligned} a_E &= \rho^*_e d_e \Delta y, & (3.55)\\ a_W &= \rho^*_w d_w \Delta y + \Psi u^*_w \Delta y, & (3.56)\\ a_N &= \rho^*_n d_n \Delta x, & (3.57)\\ a_S &= \rho^*_s d_s \Delta x + \Psi u^*_s \Delta x, & (3.58)\\ A_P &= \sum_{nb} a_{nb} - \Psi u^*_w \Delta y - \Psi u^*_s \Delta x + \Psi u^*_e \Delta y + \Psi u^*_n \Delta x, & (3.59)\\ b &= F^*_w - F^*_e + F^*_s - F^*_n. & (3.60) \end{aligned}$$

Eq. 3.54 include two different types of terms for the pressure correction p'_P: first type involving velocity, a convective term $\Psi p'_P u^*_e \Delta y$, and second

involving pressure difference, a diffusion like term $\rho_e^* d_e \Delta y \, (p'_P - p'_E)$. The relative importance of the two terms depends on the Mach number of the flow. At low Mach numbers, the diffusional part dominates and the equation exhibits an elliptic behavior. In the supersonic regime, the convective term is much larger than the diffusional term and the mass flux is governed solely by the upstream pressure. This reflects the correct hyperbolic behavior of pressure for supersonic flows. The nature of the equation is such that the transition from the subsonic to supersonic regime is gentle and the transonic flow calculations are smoothly treated.

3.8 Critical aspects

During the development of the code for the proposed solution algorithm, critical aspects have been found to be the treatment of the boundary conditions for the pressure corrector and the underrelaxation of the pressure corrector equation.

The starting problem was to understand which was the more suitable boundary condition for the pressure corrector. Being a "fictitious" variable at convergence equally null, every boundary condition satisfying this constraint could be considered physical for the problem. However it is true that on boundaries on which the pressure is specified there is no need of correction ($p^* = p$) and so p' is fixed to zero. At the same time boundaries on which the mass flow is specified need no mass correction that means no difference of p' across that boundary: null Neumann condition.

Information about boundary conditions is propagated into the domain along characteristic lines. Characteristics for compressible fluid flow are the Mach waves. Theory of characteristics reveals how, at the inlet boundary, one of the conditions should be extrapolated from interior values. Extrapolated boundaries are traduced in mathematics with Neumann type boundary conditions. The null gradient constraint is standardly applied to pressure, a

condition that respects the physics, therefore a mathematical enquiry should establish whether or not such condition on pressure correction equation results in a well posed problem. In case it results to be untrue and other types of boundary conditions are needed to respect well posedness, a solution to maintain physical constraint is needed. This mathematical and physical investigation took quite a long time because none of the previously cited authors clearly mention the problem of boundary condition for pressure corrector.

In the weakly compressible form derived in Sec. 1.2.4, pressure corrector equation is a Poisson type equation that is an elliptic problem. A first derivative constraint, Neumann boundary condition, is suitable for this kind of problem: mathematically the problem results to be well-posed. In the fully compressible version however a convective term is added and the behavior of the equation turns hyperbolic. For hyperbolic equations, unless one uses the unsteady form, resulting in an initial boundary value problem, a Dirichlet boundary condition is needed at the inlet too, to satisfy well-posedness constraint. Physically, it has already been underlined, specifying pressure corrector means to impose a pressure. To avoid inlet pressure not to change, it is uploaded at every iteration no more via the pressure corrector but via extrapolation from the interior using both value and gradient on first cell.

This passage is somehow the bottleneck of the entire process. To improve the speed of convergence for inlet pressure better would be to increase the diffusive behavior of the equation near the inlet. This can be done using, instead of the null fixed value boundary condition, a mixed (Robin type) boundary condition:

$$c\phi_w + (1-c)\left(\frac{\partial\phi}{\partial n}\right)_w = c\phi_{ref} + (1-c)\left(\frac{\partial\phi}{\partial n}\right)_{ref}. \tag{3.61}$$

This boundary condition imposes a value, depending on previously defined reference quantities ϕ_{ref} and $\left(\frac{\partial\phi}{\partial n}\right)_{ref}$, both zero in the case, to a linear combination of wall value and normal gradient weighted on scalar c. Again

it is better underline how the combination of this boundary condition on the pressure corrector and the extrapolation of inlet values for pressure from the internal domain, is mathematically and physically valid for the entire range of Mach number.

It has already been shown (see Eq. 3.35) how underrelaxation for the pressure corrector is necessary to obtain convergence. Furthermore an additional weighting factor α_{mix}, here improperly referred to as a relaxation factor, has been included in the pressure correction equation, scaling convective terms, to balance the relative influence of the diffusion term over convective term:

$$A_P = \sum_{nb} a_{nb} + \alpha_{mix}(-\Psi u_w^* \Delta y - \Psi u_s^* \Delta x + \Psi u_e^* \Delta y + \Psi u_n^* \Delta x) \ . \quad (3.62)$$

Just to show how the nature of the two relaxation factors is completely different, typical values for them two are indicated: $\alpha_p = 0.01$ and $\alpha_{mix} = 0.9$.

Chapter 4

Turbulence Modeling

This chapter is focused on the modeling of turbulence with the aim of finding closure to the set of conservation equations proposed in chapter 1, namely from Eq. 1.25 to Eq. 1.28. After a brief introduction to turbulent phenomena, classical approaches to turbulence modeling are presented with special interest in the RANS approach. Hence typical heat transfer simulation failures connected with standard turbulence models are discussed. Main part concerns the presentation of the many turbulence models capable of partially avoiding such defects in the prediction of turbulent effects, implemented in OpenFOAM for this thesis. Finally OpenFOAM `turbulenceModel` class is presented and discussed as a help for future OpenFOAM programmers with the will of further developments.

4.1 The physics of turbulence

It is known that, for low velocities, wall-bounded flows are smooth and adjacent layers of fluid slide past each other in an orderly fashion. When velocity is high instead the flow become intrinsically unsteady and chaotic even with constant boundary conditions: the flow is said to be turbulent.

Turbulence phenomena may be described with the following characteris-

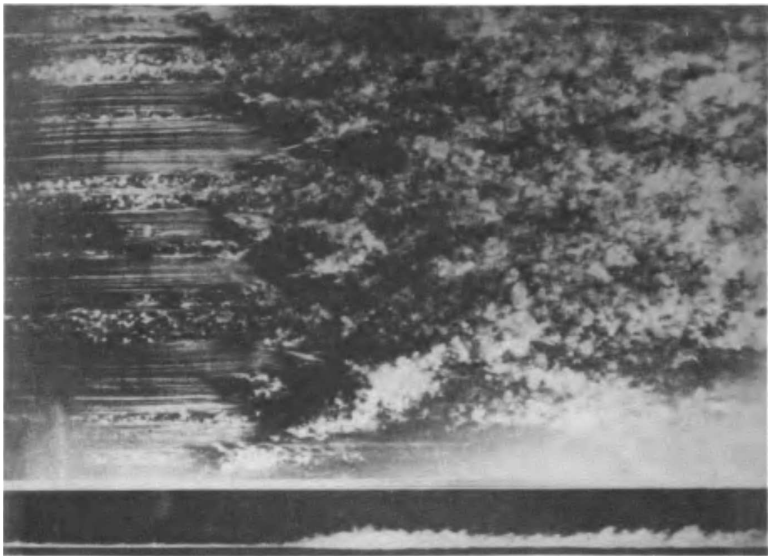


Figure 4.1: Smoke visualization of air flow over a flat plate.

tics [White, 1991, Malalasekera and Versteeg, 1995]:

- **Fluctuations** in time superimposed upon a mean value for each quantities: $\Phi(t) = \overline{\Phi} + \Phi'(t)$. Connected with this is the extension to three dimensions also for bidimensional flows: velocity fluctuates always in 3D.

- **Eddies** or fluid packets intermingle and fill the shear layer. Eddies with a wide range of length scales, coexist at the same time and at the same place.

- **Random** variations in fluid properties. Such variations are not white-noise shaped, each quantity has a specific continuous energy spectrum.

- **Self-sustaining** motion. New eddies replace those lost by viscous dissipation. Kinetic energy is handed down from bigger to smaller eddies in what is termed the energy cascade. Large scales (energy-containing subrange) contain most of the energy while smallest scales (dissipation subrange) dissipate into heat such energy. Scales laying in

between belong to the so called inertial subrange: a transport region for the cascade process.

- **Mixing** is much stronger than in laminar flows. Turbulent eddies actively increase diffusion by moving in the three dimensions. Heat transfer and friction are strongly enhanced.

Transition from laminar and turbulent regime always occurs when a certain, called critical, Reynolds number is reached. Starting from such a description, one cannot hope turbulence to be isotropic: only the smallest eddies in fact are non-directional, with bigger ones strongly dependent on mean flow direction.

4.2 Models for Turbulence

Such impressive complications do not involve the flow not to respect Navier-Stokes Equations in the general form (see Eq. 1.25), but certainly do reduce abruptly the chance of being able to solve such equations. It should be marked out, in fact, that a wide range of temporal and length scales for eddy dimensions means a wide range of scales to be computed and solved. Navier-Stokes Equations still exactly model turbulent flows and the typical unsteady turbulence phenomena connected with, but at computational costs that often results prohibitive. The need for simplifications in the treatment of turbulence has gained during the course of the years to several class of models: each involving different range of turbulence scale of interest. All these approaches are presented here briefly in decreasing computational complexity order:

- **Direct Numerical Simulation** (DNS): it is not a model, Navier-Stokes Equations are computed in their most general form meaning that the complete spectrum of involved frequencies and length scales

are solved. This approach is only feasible at current time for Low Reynolds number flows, due to limitation on computer resources, and used mainly as validation test for other approaches or as an help in understanding turbulence physics.

- **Large Eddy Simulation** (LES): remembering that energy is contained mainly in the largest scales, this model reduces the range of interest only to the biggest vortices. It employs, in fact, a time dependent three dimensional computation of the large-eddy structure and a model for the small scales. It basically consists in filtering in space the Navier-Stokes Equations with a high-pass filter, resolving for scales that actually are the energy-containing scales and modeling dissipation subrange behavior. The cut-off scale usually lies into the inertial subrange [Davidson, 2006]. LES is becoming more and more popular in the CFD community, particular complexities in treating boundary conditions and the need for wide computer resources still limit the use to simple geometries and specific areas of interest in which turbulence modeling is more than fundamental (turbulent combustion, wake effects, chemical reactions).

- **Reynolds Averaged Navier Stokes** (RANS): Navier-Stokes Equations are averaged in time on a period big enough to contain also lowest frequency oscillation. The unsteady behavior of the turbulent flows is completely neglected, turbulence become a steady phenomenon simply considering the effects of fluctuations onto the mean flow. At the state of art RANS simulations are the standard for flows involving heat transfer of industrial interest. Difficultly in fact unsteady phenomena result in being determinant for such simulations. The ease of implementation and the speed in solving, supported by a good accuracy in modeling mean flows, are reasonably believed to make this approach

still practised in middle-term. Since this thesis is treating turbulence with a RANS approach, a deeper introduction to such an approach is needed. This will be the subject of next section.

- **Hybrid**: this class is aiming in taking advantage of low computational cost of the RANS approach and the good confidence of LES simulation. The idea is to use the two approaches in different areas where flow conditions are diverse. In particular LES is used only in the zone were RANS simulations are likely to fail, namely separating zone.

4.3 RANS approach

It is already known that this approach simply consists of averaging in time (Reynolds Averaging) the exact Navier-Stokes Equations, here reported for a better reference:

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho U_j)}{\partial x_j} = 0 \,, \tag{4.1}$$

$$\begin{aligned} \frac{\partial (\rho U_i)}{\partial t} + \frac{\partial (\rho U_i U_j)}{\partial x_j} &= \\ -\frac{\partial p}{\partial x_i} &+ \frac{\partial}{\partial x_j}\left(\mu\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) - \frac{2}{3}\mu\delta_{ij}\frac{\partial U_j}{\partial x_j}\right) + F_{bi} \,, \end{aligned} \tag{4.2}$$

$$\frac{\partial \rho h_0}{\partial t} + \frac{\partial (\rho h_0 U_j)}{\partial x_j} = \frac{\partial p}{\partial t} + \frac{\partial}{\partial x_j}\left(k\frac{\partial T}{\partial x_j}\right) + \Phi + S_h \,, \tag{4.3}$$

$$p = \rho R T \,. \tag{4.4}$$

Mathematical definition of Reynolds Averaging is given below:

$$\overline{f}(x) = f - f' = \lim_{t\to\infty} \frac{1}{t}\int_0^t f(x,\tau)d\tau \,. \tag{4.5}$$

Before applying this operator to the complete set of equations, it would be better see what happen if it is applied to instantaneous fluid properties:

$$\overline{\Phi} = \overline{\overline{\Phi} + \Phi'} = \overline{\overline{\Phi}} + \overline{\Phi'} \,. \tag{4.6}$$

Here comes the interesting part: the average of the mean value is equal to itself, the average of the fluctuating part is zero:

$$\overline{\overline{\Phi}} = \overline{\Phi} \,, \tag{4.7}$$

$$\overline{\Phi'} = 0 \,, \tag{4.8}$$

all this comes directly out from the definition of the period t to be big enough to include all effects of unsteady turbulence. If products appears inside the averaging operator however, fluctuation do not completely cancel out:

$$\overline{\Psi\Phi} = \overline{(\overline{\Phi} + \Phi')(\overline{\Psi} + \Psi')} = \overline{\Phi}\overline{\Psi} + \overline{\Phi'}\overline{\Psi} + \overline{\Phi}\overline{\Psi'} + \overline{\Phi'\Psi'} = \overline{\Phi}\overline{\Psi} + \overline{\Phi'\Psi'} \,. \tag{4.9}$$

It must be underlined that also Favre (density-weighted) average:

$$\tilde{f}(x) = f - f'' = \lim_{t\to\infty} \frac{1}{\overline{\rho}\, t} \int_0^t \rho f(x,\tau)\, d\tau = \frac{\overline{\rho f(x)}}{\overline{\rho}} \,, \tag{4.10}$$

is very important for compressible flows because it exactly respects the following simplification [Knight, 1997]:

$$\overline{\rho U_i U_j} = \overline{\rho}\tilde{U}_i\tilde{U}_j + \overline{\rho U_i'' U_j''} \,. \tag{4.11}$$

Remembering such a clarification, in the following derivation Reynolds and Favre averaging will be distinguished.

After this introduction to averaging, it is better start using this operator on the Navier-Stokes Equations. First start with continuity:

$$\overline{\frac{\partial \rho}{\partial t}} + \overline{\frac{\partial(\rho U_j)}{\partial x_j}} = \frac{\partial \overline{\rho}}{\partial t} + \frac{\partial(\overline{\rho}\tilde{U}_j)}{\partial x_j} = 0 \,, \tag{4.12}$$

followed by momentum equation:

$$\begin{aligned}
&\overline{\frac{\partial \rho U_i}{\partial t}} + \overline{\frac{\partial(\rho U_i U_j)}{\partial x_j}} = -\overline{\frac{\partial p}{\partial x_i}} + \overline{\frac{\partial}{\partial x_j}\left(\mu\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) - \frac{2}{3}\mu\delta_{ij}\frac{\partial U_j}{\partial x_j}\right)} + \overline{F_{bi}} \Rightarrow \\
&\frac{\partial \overline{\rho U_i}}{\partial t} + \frac{\partial(\overline{\rho U_i U_j})}{\partial x_j} = -\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu\left(\overline{\frac{\partial U_i}{\partial x_j}} + \overline{\frac{\partial U_j}{\partial x_i}}\right) - \frac{2}{3}\mu\delta_{ij}\overline{\frac{\partial U_j}{\partial x_j}}\right) + \overline{F_{bi}} \Rightarrow \\
&\frac{\partial \overline{\rho}\tilde{U}_i}{\partial t} + \frac{\partial(\overline{\rho}\tilde{U}_i\tilde{U}_j)}{\partial x_j} + \frac{\partial(\overline{\rho U_i'' U_j''})}{\partial x_j} = \\
&\qquad -\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu\left(\frac{\partial \overline{U_i}}{\partial x_j} + \frac{\partial \overline{U_j}}{\partial x_i}\right) - \frac{2}{3}\mu\delta_{ij}\frac{\partial \overline{U_j}}{\partial x_j}\right) + \overline{F_{bi}} \,,
\end{aligned} \tag{4.13}$$

and finally energy:

$$\overline{\frac{\partial \rho h_0}{\partial t}} + \overline{\frac{\partial (\rho h_0 U_j)}{\partial x_j}} = \overline{\frac{\partial}{\partial x_j}(k\frac{\partial T}{\partial x_j})} + \overline{\Phi} + \overline{S_h} \Rightarrow$$
$$\frac{\partial \overline{\rho}\tilde{h_0}}{\partial t} + \frac{\partial (\overline{\rho}\tilde{h_0}\tilde{U_j})}{\partial x_j} + \frac{\partial (\overline{\rho h_0'' U_j''})}{\partial x_j} = \frac{\partial}{\partial x_j}(k\frac{\partial \overline{T}}{\partial x_j}) + \overline{\Phi} + \overline{S_h} \ , \quad (4.14)$$

where $\overline{\Phi}$ is:

$$\overline{\Phi} = \frac{\mu}{2}\overline{\left(\frac{\partial \overline{U_i}}{\partial x_j} + \frac{\partial U_i'}{\partial x_j} + \frac{\partial \overline{U_j}}{\partial x_i} + \frac{\partial U_j'}{\partial x_i}\right)^2} \ . \quad (4.15)$$

The RANS equations look pretty much like the exact Navier-Stokes Equations, Eq. 4.1, except for an extra term in momentum and energy equation, depending on fluctuating quantities. These terms appear inside the divergence operator and they are often moved to the right hand side as companion terms of laminar viscous stress and conduction terms. That is why they are in literature famous as Reynolds stresses and turbulent heat fluxes.

RANS approach will now used from now on in this thesis and so to clarify notation all the bars, apart from where specifically needed, are dropped with the implicit notation that $a = \bar{a}, \tilde{a}$ and $a' = a', a''$.

After the Reynolds Averaging, temporal dependency is of no interest anymore, so simplification of steady-state can finally be introduced, just drop the time derivative term, and the RANS equations presented:

$$\frac{\partial (\rho U_j)}{\partial x_j} = 0 \ , \quad (4.16)$$

$$\frac{\partial (\rho U_i U_j)}{\partial x_j} = \quad (4.17)$$
$$-\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) - \frac{2}{3}\mu\delta_{ij}\frac{\partial U_j}{\partial x_j} - \overline{\rho U_i' U_j'}\right) + F_{bi} \ ,$$

$$\frac{\partial (\rho h_0 U_j)}{\partial x_j} = \frac{\partial}{\partial x_j}(k\frac{\partial T}{\partial x_j} - \overline{\rho h_0' U_j'}) + \overline{\Phi} + S_h \ , \quad (4.18)$$

$$p = \rho R T \ . \quad (4.19)$$

It comes out that modeling turbulence is another way of saying for finding a way to treat the two terms: $\overline{\rho U_i' U_j'}$ and $\overline{\rho h_0' U_j'}$.

These terms introduce 9 more unknown into the Navier-Stokes Equations system, that before was determined but now is undetermined with 9 degree of freedom. Constraints should be introduced if the system want to be solved: introducing these constraints is modeling turbulence.

Modeling these terms is also known as the closure problem, explicitly pointing out that the number of unknowns [15: density, velocity (3), pressure, Reynolds stresses (6), enthalpy, turbulent heat fluxes (3)] is larger than the number of equations (6).

Different levels of approximation as been proposed to model in a more complex or simpler way those terms, depending on the needed accuracy and the available computer resources. Main class of models are quickly introduced again in order of decreasing computational complexity. To better understand energy equation has been neglected (drop off 1 equation and 4 unknowns) and turbulent heat flux is implicitly treated like Reynolds stress.

- Differential Stress Models, more commonly called **Reynolds Stress Transport Model** (RSTM) or Second Moment Closure models, solve a separate scalar transport equation for each stress component and closure is reached. The biggest drawback of this category of models can be guessed easily: it is computationally very expensive compared with the following methods. Moreover, stability is far from being easy to obtain and, even if they include more turbulence physics than eddy viscosity models with advection and production terms (energy-in terms) derived exactly, they still require modeling for important terms quote: redistribution and dissipation terms [Apsley, 2006].

- **Eddy Viscosity Models**: the RSTMs are the only models not taking advantage of Boussinnesq assumption, other models in fact, to prevent solving additional differential equations, relate Reynolds stress tensor

to strain rate or, in other words, to velocity gradients:

$$-\rho\overline{U_i'U_j'} = \mathscr{T} = \mu_t\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} - \frac{2}{3}\delta_{ij}\frac{\partial U_k}{\partial x_k}\right) - \frac{2}{3}\rho k\delta_{ij} \qquad (4.20)$$

where last two terms are added to be consistent with continuity and turbulent kinetic energy definition, that is to have the proper trace for turbulent stress tensor [Davidson, 2003, Knight, 1997]. The problem now shifts to a way of guessing the eddy viscosity μ_t, that is why the following classes of turbulent models are called Eddy Viscosity Models. The main advantage of such methods is the facility of implementation in viscous solvers: just replace molecular viscosity μ with an effective viscosity $\mu_{eff} = \mu + \mu_t$ and the same for thermal diffusivity α with an effective thermal diffusivity $\alpha_{eff} = \alpha + \alpha_t$.

This thesis is proceeding further with eddy viscosity models so the last and really implemented form of the Navier-Stokes Equationsis presented. Moreover a change in variable for the energy equation has been done solving for static enthalpy instead of total:

$$\frac{\partial(\rho U_j)}{\partial x_j} = 0\,, \qquad (4.21)$$

$$\frac{\partial(\rho U_i U_j)}{\partial x_j} = \qquad (4.22)$$

$$-\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\mu_{eff}\left(\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) - \frac{2}{3}\delta_{ij}\frac{\partial U_j}{\partial x_j}\right) - \frac{2}{3}\rho k\delta_{ij}\right] + F_{bi}\,,$$

$$\frac{\partial(\rho h U_j)}{\partial x_j} = \frac{\partial}{\partial x_j}(\alpha_{eff}\frac{\partial h}{\partial x_j}) + U_i\frac{\partial p}{\partial x_i} + \overline{\Phi} + S_h\,, \qquad (4.23)$$

$$p = \rho RT\,. \qquad (4.24)$$

Even if theoretical foundation is limited to simple shear flows, eddy viscosity models are the most commonly used because they are very cheap in CPU time, well known in their pros and cons and global predictions usually do not fail so much. At the contrary, the main drawback is the little turbulence physics they are based on: anisotropy and history effects are completely neglected. Due to the mathematical structure of the model in fact, in the

best case one can hope to accurately represent only one Reynolds Stress. Applicability of such models should be restricted in cases where stress are almost mono directional. This is the case almost never but often it is possible to individuate a major stress, the idea is to well predict the main one and neglect if the others are not that well modeled.

This category is subdivided again into:

- **Algebraic Models** also called zero equation models because do not solve any extra partial differential equation and μ_t is obtained from mean flow properties and a prescribed, geometry dependent characteristic length.
- **One Equation Models**, in which a length scale is also prescribed but an extra partial differential equation is solved for k.
- **Two Equations Models** where transport equations are specified for both k and its rate of dissipation (absolute ε or specific ω), and μ_t is then computed locally from the value of these scalars.

The fundamental limitation of zero equation models is the assumption that Reynolds Stress can be modeled on mean flow: turbulence does not in fact respond instantly to changes in mean flow but rather adjust over a time scale typical of the turbulent structure. One equation models, even if feels the influence of fluctuating properties via turbulent kinetic energy equation, do still depend on geometry via the definition of a characteristic length. First, in order of complexity, class of model to be independent on the definition of an appropriate length scale is the two equations class. This property is fundamental in predicting many type of flows, for example recirculating flows: mixing length in fact cannot be used for flows in which diffusion and convection are not negligible.

The following will be consistent with the choice of two equations models as a good compromise between area of applicability and computational

feasibility.

4.4 Standard $k-\varepsilon$ turbulence model

The landmark model for eddy viscosity two equations models is the $k-\varepsilon$ model by Jones and Launder [1972] which appeared in 1972 and nowadays known as the standard $k-\varepsilon$ model.

Before building and analyzing the structure of the transport equation for turbulent kinetic energy k and turbulent kinetic energy dissipation ε, a general overview common to all $k-\varepsilon$ models is needed, in order to understand how k and ε are related to the wanted scalar μ_t.

First turbulent length scale l_t is computed locally as:

$$l_t = \frac{k^{\frac{3}{2}}}{\varepsilon} \,, \tag{4.25}$$

and second, following the definition of length scale, one obtains:

$$\mu_t = C_\mu \rho k^{\frac{1}{2}} l_t = C_\mu \rho \frac{k^2}{\varepsilon} \,. \tag{4.26}$$

4.4.1 Equation for turbulent kinetic energy

An exact equation for k is not difficult to derive from Navier-Stokes Equations with some manipulations [Wilcox, 1998]:

$$\begin{aligned} \frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho k U_j)}{\partial x_j} = \frac{\partial}{\partial x_j}\left(\mu \frac{\partial k}{\partial x_j}\right) \\ - \frac{\partial}{\partial x_j}\left(\frac{\rho}{2}\overline{U_j' U_i' U_i'} + \overline{U_j' p'}\right) - \rho \overline{U_i' U_j'} \frac{\partial U_i}{\partial x_j} - \mu \overline{\frac{\partial U_i'}{\partial x_k}\frac{\partial U_i'}{\partial x_k}} \,. \end{aligned} \tag{4.27}$$

Terms on the first line need no further modeling so efforts should be directed in modeling the last three blocks. First of these three comes the turbulent diffusion of kinetic energy: transport of velocity fluctuations by the fluctuations themselves, almost always modeled by use of gradient-diffusion assumption:

$$-\left(\frac{\rho}{2}\overline{U_j' U_i' U_i'} + \overline{U_j' p'}\right) \approx \left(\frac{\mu_t}{\sigma_k}\frac{\partial k}{\partial x_j}\right) \,, \tag{4.28}$$

where σ_k is the so called Turbulent Prandtl number whose value is usually close to unity.

Second term represents the rate of production (positive source term) due to mean flow and is commonly approximated, with the help of Boussinesq, by:

$$P_k = -\rho\overline{U_i'U_j'}\frac{\partial U_i}{\partial x_j} = \mu_t\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} - \frac{2}{3}\delta_{ij}\frac{\partial U_k}{\partial x_k}\right)\frac{\partial U_i}{\partial x_j} - \frac{2}{3}\rho k\frac{\partial U_j}{\partial x_j} \ . \quad (4.29)$$

In order to make equations the most implicit as possible, in computational fluid dynamic it is common to split the implicit and the explicit part of the production term:

$$P_k = \underbrace{G}_{explicit} \underbrace{- \frac{2}{3}\rho k\frac{\partial U_j}{\partial x_j}}_{implicit} \ . \quad (4.30)$$

Last term is the sink term or dissipation for which a second equation is needed:

$$-\mu\overline{\frac{\partial U_i'}{\partial x_k}\frac{\partial U_i'}{\partial x_k}} = \rho\varepsilon \ . \quad (4.31)$$

Hence the final form of the k equation is:

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho k U_j)}{\partial x_j} - \frac{\partial}{\partial x_j}\left[(\mu + \frac{\mu_t}{\sigma_k})\frac{\partial k}{\partial x_j}\right] = P_k - \rho\varepsilon \ . \quad (4.32)$$

4.4.2 Equation for turbulent kinetic energy dissipation

Although it is possible to derive an exact equation for dissipation directly from Navier-Stokes Equations the modeling applied to it is so severe that it is best to regard the entire equation as a model [Ferziger and Peric, 2002]. The most common form for such an equation is:

$$\frac{\partial(\rho\varepsilon)}{\partial t} + \frac{\partial(\rho\varepsilon U_j)}{\partial x_j} - \frac{\partial}{\partial x_j}\left[(\mu + \frac{\mu_t}{\sigma_\varepsilon})\frac{\partial\varepsilon}{\partial x_j}\right] = C_{\varepsilon 1}P_k\frac{\varepsilon}{k} - \rho C_{\varepsilon 2}\frac{\varepsilon^2}{k} \ . \quad (4.33)$$

In Tab. 4.1 standard values for the introduced constants are given.

Such equations should be solved together with the Navier-Stokes Equations and, as one could guess and see, coupling is pretty strong. But as long

C_μ	$C_{\varepsilon 1}$	$C_{\varepsilon 2}$	σ_k	σ_ε
0.09	1.44	1.92	1.0	1.3

Table 4.1: Standard values for standard $k-\varepsilon$ coefficients.

as Eq. 4.32 and Eq. 4.33 are much stiffer than the others, they are usually solved segregated in order to increase convergence properties. In addition it is necessary to use under-relaxation techniques in iterative methods.

4.4.3 Drawbacks

After a so straight forward presentation of this model, it would be better to enquire the main and well known characteristic drawbacks.

The profiles of turbulent scalars are typically much more peaked near the wall than mean velocity profile. Such peaks are difficult to capture if a grid, tuned on mean velocity needs, is used also on turbulent equations, better will be to use a finer grid for k and ε. This is not usually possible and near wall treatment become a very important task.

At wall complete no-slip condition applies, meaning that both mean and fluctuating part of velocity vanish. As a consequence k is exactly null on wall boundary and the same stands for μ_{eff}. In near wall layer turbulent stresses, usually much higher, decreases to values of comparable magnitude with the viscous one. At high Reynolds number, this results in very large flow gradients near the wall, moreover solid boundaries selectively damp fluctuations normal to the wall.

This is usually avoided via wall-functions meaning that the flow is not resolved in the near-wall region and theoretical profiles between boundary surface and first near-wall node are assumed and superimposed. The main disadvantage of this approach is that theoretical profiles are only known and justified in near-equilibrium boundary layers, in addition grid should be chosen with care, avoiding first node y^+ to be out of the range 15-150 with

an optimal range of 30-50 [Apsley, 2006].

Another approach could be to model near wall damping directly inside the turbulence model. It is so possible to refine the grid near the wall working with $y^+ \approx 1$ and include the effects of molecular viscosity in the coefficients of the turbulent model. These models, known as Low Reynolds, even if much more computationally expensive are fundamental in case near wall behavior is needed. Many models belonging to this class are described in the following sections.

The standard $k-\varepsilon$ however is a High Reynolds model meaning it must be supported by wall-functions, if at least approximately the near wall behavior wants to be catched. So let's come back at the concept of wall function to see what is the standard formulation for near wall turbulence quantities.

4.4.4 Wall functions

For both velocity and turbulent kinetic energy no-slip condition stands at the wall. To force the correct near-wall behavior however it is important to have a good guess for the "mean" value of turbulent viscosity for $\vec{U}$ and of production and dissipation term for k.

For velocity, once the right profile for k and ε is imposed, nothing has to be done and wall functions only permit to correctly calculate the right wall shear stress:

$$\tau_w = \rho \nu_{e,w} \frac{U_P}{y_P} \,, \tag{4.34}$$

where the subscript P stands for first node from the wall, $\nu_{e,w}$ is the effective kinematic viscosity at the wall

$$\nu_{e,w} = \nu \cdot \begin{cases} 1 & \text{for } y_P^+ < y_\nu^+ \\ \frac{y_P^+}{y_\nu^+ + \frac{1}{\kappa}\ln(1+\kappa(y_P^+ - y_\nu^+))} & \text{for } y_P^+ > y_\nu^+ \end{cases} \tag{4.35}$$

and y_ν^+ is a non-dimensional matching length whose typical value is 7.17 for smooth walls.

In the equation for k, P_k and ε should be replaced at the first node with an averaged value:

$$P_k^{av} = \frac{(\tau_w/\rho)^2}{\kappa u_0 y_P}\left\{\ln[1+\kappa(y_P^+ - y_\nu^+)] - \frac{\kappa(y_P^+ - y_\nu^+)}{1+\kappa(y_P^+ - y_\nu^+)}\right\} , \quad (4.36)$$

$$\varepsilon_{av} = \frac{u_0^3}{\kappa y_P}\left[\ln\left(\frac{y_P}{y_\varepsilon}\right)+1\right] , \quad (4.37)$$

where u_0 is the "equivalent" friction velocity:

$$u_0 = C_\mu^{0.25} k_P^{0.5} , \quad (4.38)$$

and

$$y_\varepsilon = \frac{2k_P\nu}{C_\mu^{0.5} u_0} . \quad (4.39)$$

As a consequence dissipation equation is not solved onto the first layer of cells being the near wall node value superimposed [Apsley, 2006].

Furthermore near wall temperature profiles should also be corrected when High Reynolds turbulence models are used. The log-law can be adapted to provide an equation that relates the value of temperature at the near wall node T_P, to that at the wall T_w and to the value of the wall heat flux rate q_w.

$$T_P^+ = \rho\, C_p\, C_\mu^{0.25}\, k_P^{0.5}\, \frac{T_w - T_P}{q_w} , \quad (4.40)$$

where

$$T_P^+ = \sigma_t\left[\frac{1}{\kappa}\ln\left(\frac{y_P^+ \exp(\kappa C_\mu^{0.25}\, y_\nu^+)}{y_\nu^+}\right) + y_\nu^+ C_\mu^{0.25}\left(\frac{Pr}{\sigma_t} - 1\right)\right] . \quad (4.41)$$

This equation is read in two different ways depending on physical boundary condition for temperature. If fixed wall heat flux simulations want to be performed, the numerical condition for T will be fixed value given by:

$$T_w = T_P + T_P^+ \frac{q_w}{\rho\, C_p\, C_\mu^{0.25}\, k_P^{0.5}} . \quad (4.42)$$

If vice versa temperature is imposed on the boundary, the temperature gradient must be held fixed numerically:

$$q_w = k \left(\frac{\partial T}{\partial n} \right)_w = \rho C_p C_\mu^{0.25} k_P^{0.5} \frac{T_w - T_P}{T_P^+} \ . \tag{4.43}$$

Here notation can be confusing so it would be better underline that k on the left hand side is the thermal conductivity while k_P on the right hand side is the turbulent kinetic energy at first node [Iacovides, 2004, Apsley, 2006]. The values of T_w and q_w depend on T_P so they change during the iteration process. When convergence is reached the value of T_P, T_w, q_w will however exactly satisfy the wall function.

4.4.5 Stability

After this quite long digression, some words more should be spent on the problems connected with the standard $k - \varepsilon$ model. If $k - \varepsilon$ is applied on regions of low turbulence numerical instability may arises. The problem is connected with the modeling of the destruction term in the ε equation: ε^2/k. Even if both quantities goes to zero, the rate of decreasing is not always correct [Davidson, 2003]. Even if they are known to fail in well predicting adverse pressure gradient flows, turbulence model of the $k - \varepsilon$ class are still the most used two equations models world wide [Wilcox, 1998].

4.5 Failures in predictions for heat transfer

Apart from the above mentioned well-known defects of the standard $k - \varepsilon$, other problems, specifically connected with heat transfer simulations, arise making it no more suitable for such predictions. These new failures of standard turbulence models are presented and a way to solve them is hinted. Later implemented models are described in detail.

The correct modeling of turbulent quantities is fundamental in conducting heat transfer simulations, because of the simultaneous importance of

well predicting both the near wall behavior and the complex structures of the main flow. Correct predictions of thermal quantities and gradients inside boundary layers are necessary to establish whether or not the cooling system is efficient. At the same time such properties strongly depend on the development of the free stream flow.

Usage of wall function approach has to be avoided because of the unpredictability of boundary thermal gradient and the failure in predicting transitional flows and low Reynolds as well as adverse pressure gradient flows.

Modeling flows close to solid walls requires integration of the two equations over a fine grid in order to capture turbulent peaks, see Sec. 4.4, as well as corrections for Low Reynolds number effects. Several so called Low Reynolds $k-\varepsilon$ models have been proposed in the course of years, see Patel et al. [1985] and Wilcox [1998] for a review. The idea standing behind such models is to damp turbulent viscosity near the wall through a damping function f_μ going towards zero as the distance from the wall is reducing. Constants multiplying source terms in the turbulent dissipation equation are in some cases also damped. The basic structure of the models is the same for all of them differing in the tuning of the damping functions and some extra sources in dissipation equation.

It is known from literature that in high strain rate regions eddy viscosity models overpredict turbulent kinetic energy: this problem is sometimes referred to as "stagnation point anomaly" [Durbin, 1996]. The cause of such an overprediction stands in the modeling of the production term P_k supposed to be proportional to the rate of strain S_{ij}. When S_{ij} is too large, and that happens not only near stagnation points but also in swirling flows and even in turbine passages, such proportionality fails to be physical or better the errors with the simplifications become unacceptable. To avoid such overprediction of turbulent kinetic energy, turbulent time scale, normally computed as the

ratio between k and ε, is bounded. Such a bound was derived from a sort of "realizability" constraint for Reynolds stress tensor [Medic and Durbin, 2002].

Having modeled near wall behavior, now the focus shifts towards well matching good near wall predictions with suitable modeling of flow structures far from the wall. Being the standard $k-\varepsilon$ model still one of the best in predicting High Reynolds zones, Two Layer $k-\varepsilon$ models have been implemented. Such methods, in fact, consist in patching together a one equation model in the near wall layer and a two equations High Reynolds model, namely standard $k-\varepsilon$ model or the renormalized group variant, in the outer layer [Rodi, 1991].

It has already been mentioned how $k-\varepsilon$ models are likely to fail in predicting adverse pressure gradient flows. Another class of model, writing the second equation in terms of specific dissipation ω, is instead better performing for separating flows. Several models have been proposed all sharing the same defect in depending from freestream values. Interesting is the $k-\omega$ in the SST (Shear Stress Transport) variant: it includes the modification of the standard $k-\omega$ to avoid sensitivity to (quite arbitrary) freestream values of ω [Menter, 1994, 1993]. The basic idea is similar to Two Layer models: two different approaches are merged together to model the two different flow regions. The sublayer and logaritmic model is the standard $k-\omega$, chosen because of its robustness, the absence of damping function and Dirichlet type boundary conditions. From the wake region and outside the boundary layer the standard $k-\varepsilon$, written in terms of ω, has been preferred due to its good compromise in predicting different kind of flows.

Summarizing it is kind of easy to obtain turbulence models that well perform in wall bounded $k-\omega$ or in frees shear flows $k-\varepsilon$ but since most complex flows include both types of regions, the task is to well match both behaviors.

4.6 Low Reynolds $k - \varepsilon$

As already said, models of this class all share the same basic structure, differing only in tuning of the damping functions namely f_μ, f_1, f_2. First and biggest change is the introduction of the damping function f_μ introduced to mimic the direct effect of molecular viscosity on the shear stress:

$$\mu_t = C_\mu f_\mu \frac{k^2}{\varepsilon} \ . \tag{4.44}$$

Little is known about such function but to maintain agreement with standard $k - \varepsilon$ in the High Reynolds zone it must tend to unity in the fully turbulent logarithmic layer. In addition an "experimental" curve can be obtained calculating an expression for f_μ for a bidimensional boundary layer and substituting measured data for k^+, ε^+, $\overline{uv}^+$:

$$f_\mu = \frac{-\overline{uv}^+ \varepsilon^+}{C_\mu k^{+2} (dU^+/dy^+)} \ . \tag{4.45}$$

It exhibits an almost constant value for $y^+ < 15$, a linear increase up to $y^+ = 60$ and a gradual approach toward unity.

Moreover equation for turbulent kinetic energy dissipation is also damped in its source terms and an extra term is added:

$$\frac{\partial(\rho\varepsilon)}{\partial t} + \frac{\partial(\rho\varepsilon U_j)}{\partial x_j} - \frac{\partial}{\partial x_j}\left[(\mu + \frac{\mu_t}{\sigma_\varepsilon})\frac{\partial \varepsilon}{\partial x_j}\right] = C_{\varepsilon 1} f_1 P_k \frac{\varepsilon}{k} - \rho C_{\varepsilon 2} f_2 \frac{\varepsilon^2}{k} + E \ , \tag{4.46}$$

The function f_2 is introduced primarily to incorporate Low Reynolds number effects in the sink term. The effects of such a function is generally limited to the viscous sublayer and nuances in the shapes do not exert large influence on the overall results. E and f_1 usually influence magnitudes at peak or at the wall. Equation for k remains substantially invariant with only the addition of an extra dissipation terms for some models.

Damping functions are in general expressed in terms of two turbulent non dimensional numbers:

$$Re_y = \sqrt{k}y/\nu \,, \tag{4.47}$$

$$Re_t = k^2/\varepsilon\nu \,, \tag{4.48}$$

where here, and from now on, y is wall distance if not specified. After this description of the commonalities between such models, each model is analyzed in its own specific formulation.

4.6.1 Abe Kondoh Nagano

First model to be presented is the model proposed by Abe et al. [1994]. This model will be referred to from now on with the name AKN. Equation of reference as well as values for the coefficients are reported:

$$\frac{\partial(\rho\varepsilon)}{\partial t} + \frac{\partial(\rho\varepsilon U_j)}{\partial x_j} - \frac{\partial}{\partial x_j}\left[(\mu + \frac{\mu_t}{\sigma_\varepsilon})\frac{\partial\varepsilon}{\partial x_j}\right] = C_{\varepsilon 1}P_k\frac{\varepsilon}{k} - \rho C_{\varepsilon 2}f_2\frac{\varepsilon^2}{k} \,, \tag{4.49}$$

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho k U_j)}{\partial x_j} - \frac{\partial}{\partial x_j}\left[(\mu + \frac{\mu_t}{\sigma_k})\frac{\partial k}{\partial x_j}\right] = P_k - \rho\varepsilon \,. \tag{4.50}$$

This model uses another non-dimensional number to express the damping functions, it is again a sort of Reynolds number calculated via turbulence dissipation:

$$y^* = \frac{y\varepsilon^{0.25}}{\nu^{0.75}} \,. \tag{4.51}$$

The production term in turbulence dissipation equation is not damped so only two damping functions are present in this model:

$$f_\mu = \left[1 - \exp\left(\frac{y^*}{14.0}\right)\right]^2 \left\{1 + \frac{5.0}{Re_t^{0.75}}\exp\left[-\left(\frac{Re_t}{200}\right)^2\right]\right\} \,, \tag{4.52}$$

$$f_2 = \left[1 - \exp\left(\frac{y^*}{3.1}\right)\right]^2 \left\{1 + 0.3\exp\left[-\left(\frac{Re_t}{6.5}\right)^2\right]\right\} \,. \tag{4.53}$$

C_μ	$C_{\varepsilon 1}$	$C_{\varepsilon 2}$	σ_k	σ_ε
0.09	1.5	1.9	1.4	1.4

Table 4.2: Standard values for Abe et al. Low Reynolds $k-\varepsilon$ coefficients.

In Tab. 4.2 standard values for the introduced constant are given. Wall boundary condition for ε are specified in such a way:

$$\varepsilon_w = 2\frac{\nu k}{y^2} \ , \tag{4.54}$$

where this condition is intended imposed on first near wall node. For k, it is better remind the no-slip condition on solid surfaces.

4.6.2 Chien

The following is the model proposed by Chien [1982], known to the author from [Yoder and Georgiadis, 1999] and referred to as CH later on.

$$\frac{\partial(\rho\varepsilon)}{\partial t} + \frac{\partial(\rho\varepsilon U_j)}{\partial x_j} - \frac{\partial}{\partial x_j}\left[(\mu + \frac{\mu_t}{\sigma_\varepsilon})\frac{\partial \varepsilon}{\partial x_j}\right] = C_{\varepsilon 1}P_k\frac{\varepsilon}{k} - \rho C_{\varepsilon 2}f_2\frac{\varepsilon^2}{k} - 2\frac{\mu}{y^2}\exp\left(-0.5y^+\right)\varepsilon \ , \tag{4.55}$$

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho k U_j)}{\partial x_j} - \frac{\partial}{\partial x_j}\left[(\mu + \frac{\mu_t}{\sigma_k})\frac{\partial k}{\partial x_j}\right] = P_k - \rho\varepsilon - 2\frac{\mu k}{y^2} \ . \tag{4.56}$$

The viscosity damping function:

$$f_\mu = \left[1 - \exp\left(-0.0015y^+\right)\right] \ , \tag{4.57}$$

is expressed this time in terms of:

$$y^+ = \frac{yu_\tau}{\nu} \ . \tag{4.58}$$

Looking carefully at such an equation, it appears how difficult can be to implement y^+ for cells far from the wall. It is not that easy in fact to

define what really u_τ is for an internal cell. This computation was done with reference to the friction velocity value of the closest wall cell.

The other damping function is instead modeled with the standard turbulent Reynolds number:

$$f_2 = \left\{ 1 - 0.22 \exp\left[-\left(\frac{Re_t}{6} \right)^2 \right] \right\} . \tag{4.59}$$

In Tab. 4.3 standard values for the introduced constant are given.

C_μ	$C_{\varepsilon 1}$	$C_{\varepsilon 2}$	σ_k	σ_ε
0.09	1.35	1.8	1.0	1.3

Table 4.3: Standard values for Chien Low Reynolds $k - \varepsilon$ coefficients.

Wall boundary value for ε will be held fixed to 0, as one may notice in fact an extra term has been introduced into turbulence kinetic energy equation taking into account the correct near wall behavior of the sink term in that equation. It is unnecessary to repeat that wall boundary condition for k is a null Dirichlet condition.

4.6.3 Chen Lien Leschziner

This is the model proposed in [Chen et al., 1996]. For brevity, this model will be referred to from now on with the name CLL. Equation of reference as well as values for the coefficients are reported:

$$\frac{\partial(\rho\varepsilon)}{\partial t} + \frac{\partial(\rho\varepsilon U_j)}{\partial x_j} - \frac{\partial}{\partial x_j}\left[(\mu + \frac{\mu_t}{\sigma_\varepsilon}) \frac{\partial \varepsilon}{\partial x_j} \right] = C_{\varepsilon 1} G \frac{\varepsilon}{k} + (C_{\varepsilon 3} - \frac{2}{3} C_{\varepsilon 1}) \rho\varepsilon \frac{\partial (U_j)}{\partial x_j} + C_{\varepsilon 1} P'_k \frac{\varepsilon}{k} - \rho C_{\varepsilon 2} f_2 \frac{\varepsilon^2}{k} , \tag{4.60}$$

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho k U_j)}{\partial x_j} - \frac{\partial}{\partial x_j}\left[(\mu + \frac{\mu_t}{\sigma_k}) \frac{\partial k}{\partial x_j} \right] = P_k - \rho\varepsilon . \tag{4.61}$$

In this model damping functions are again only two:

$$f_\mu = [1 - \exp(-0.0198 Re_y)] \left[1 + \frac{5.29}{Re_y}\right] , \tag{4.62}$$

$$f_2 = \left[1 - 0.3 \exp\left(-Re_t^2\right)\right] , \tag{4.63}$$

but source term in ε equation does not remain unchanged. An extra production term is doping the near wall behavior:

$$P_k' = 1.33 f_2 \left[P_k + 2\mu \frac{k}{y^2} \exp\left(-0.00375 Re_y^2\right)\right] . \tag{4.64}$$

In Tab. 4.4 standard values for the introduced constant are given.

C_μ	$C_{\varepsilon 1}$	$C_{\varepsilon 2}$	$C_{\varepsilon 3}$	σ_k	σ_ε
0.09	1.44	1.92	-0.33	1.0	1.219

Table 4.4: Standard values for Chen et al. Low Reynolds $k - \varepsilon$ coefficients.

Wall boundary condition for ε are specified in such a way:

$$\varepsilon_w = \frac{P_k}{\rho} + 2\frac{\nu k}{y^2} . \tag{4.65}$$

Again k is zero at solid boundaries.

4.6.4 Hwang Lin

This is the model proposed by Hwang and Lin [1998]. This model will be called also with the acronym HW. Equation of reference as well as values for the coefficients are reported.

The peculiarity of such a model, even if shared with many other models is that it is solving for $\tilde{\varepsilon}$ instead of ε. The two variables are linked together by the following equations:

$$\tilde{\varepsilon} = \varepsilon - \hat{\varepsilon} , \tag{4.66}$$

$$\hat{\varepsilon} = 2\nu |\nabla \left(k^{0.5}\right)|^2 . \tag{4.67}$$

An extra term depending on $\hat{\varepsilon}$ will also be added in the kinetic energy equation:

$$\frac{\partial(\rho\tilde{\varepsilon})}{\partial t}+\frac{\partial(\rho\tilde{\varepsilon}U_j)}{\partial x_j}-\frac{\partial}{\partial x_j}\left[(\mu+\frac{\mu_t}{\sigma_\varepsilon})\frac{\partial\tilde{\varepsilon}}{\partial x_j}\right]= C_{\varepsilon1}G\frac{\tilde{\varepsilon}}{k}-\frac{2}{3}C_{\varepsilon1}\rho\tilde{\varepsilon}\frac{\partial(U_j)}{\partial x_j}-\frac{\partial}{\partial x_j}\left[0.5\mu\frac{\tilde{\varepsilon}}{k}\frac{\partial k}{\partial x_j}\right]-\rho C_{\varepsilon2}f_2\frac{\tilde{\varepsilon}^2}{k}\ , \quad (4.68)$$

$$\frac{\partial(\rho k)}{\partial t}+\frac{\partial(\rho kU_j)}{\partial x_j}-\frac{\partial}{\partial x_j}\left[(\mu+\frac{\mu_t}{\sigma_k})\frac{\partial k}{\partial x_j}\right]=P_k-\frac{\partial}{\partial x_j}\left[0.5\mu\frac{k}{\varepsilon}\frac{\partial\hat{\varepsilon}}{\partial x_j}\right]-\rho\varepsilon\ . \quad (4.69)$$

Apart from the definition of this new parameter y_λ:

$$y_\lambda=y\sqrt{\frac{\tilde{\varepsilon}}{\nu k}}\ , \quad (4.70)$$

used in the definition of the viscosity damping function

$$f_\mu=\left[1-\exp\left(-0.01y_\lambda-0.008y_\lambda^3\right)\right]\ , \quad (4.71)$$

the main difference stands in the tuning of σ constants with the change in y_λ:

$$\sigma_k = \left[1.44-1.1\exp\left(-\frac{y_\lambda}{10}\right)\right]\ , \quad (4.72)$$

$$\sigma_\varepsilon = \left[1.3-1.0\exp\left(-\frac{y_\lambda}{10}\right)\right]\ . \quad (4.73)$$

The effect of this tuning is to damp more diffusivity factor, without constraining production terms. In Tab. 4.5 standard values for the introduced constant are given.

C_μ	$C_{\varepsilon1}$	$C_{\varepsilon2}$
0.09	1.44	1.92

Table 4.5: Standard values for Hwang and Lin Low Reynolds $k-\varepsilon$ coefficients.

Wall boundary value for $\tilde{\varepsilon}$ and k as well are fixed to 0.

4.6.5 Lam Bremhorst

This model was proposed by Lam and Bremhorst [1981]. It will be referred to from now on with the name LB. Equation of reference as well as values for the coefficients are reported:

$$\frac{\partial(\rho\varepsilon)}{\partial t} + \frac{\partial(\rho\varepsilon U_j)}{\partial x_j} - \frac{\partial}{\partial x_j}\left[(\mu + \frac{\mu_t}{\sigma_\varepsilon})\frac{\partial\varepsilon}{\partial x_j}\right] = C_{\varepsilon 1} f_1 G\frac{\varepsilon}{k} - \frac{2}{3}C_{\varepsilon 1} f_1 \rho\varepsilon\frac{\partial(U_j)}{\partial x_j} - \rho C_{\varepsilon 2} f_2 \frac{\varepsilon^2}{k} \ , \quad (4.74)$$

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho k U_j)}{\partial x_j} - \frac{\partial}{\partial x_j}\left[(\mu + \frac{\mu_t}{\sigma_k})\frac{\partial k}{\partial x_j}\right] = P_k - \rho\varepsilon \ . \quad (4.75)$$

The three damping functions are:

$$f_\mu = \left[1 - \exp\left(-0.0165 Re_y\right)\right]^2 \left[1 + \frac{20.5}{Re_t}\right] \ , \quad (4.76)$$

$$f_1 = \left[1 - \left(\frac{0.05}{f_\mu}\right)^{3.0}\right] \ , \quad (4.77)$$

$$f_2 = \left[1 - \exp\left(-Re_t^2\right)\right] \ . \quad (4.78)$$

In Tab. 4.6 standard values for the introduced constant are given. Wall

C_μ	$C_{\varepsilon 1}$	$C_{\varepsilon 2}$	σ_k	σ_ε
0.09	1.44	1.92	1.0	1.3

Table 4.6: Standard values for Lam and Bremhorst Low Reynolds $k-\varepsilon$ coefficients.

boundary condition for ε is of type zero gradient normal to walls. Obviously k is respecting no slip condition.

4.6.6 Lien Leschziner

This is the model proposed in [Lien and Leschziner, 1993] to conform with one equation model by Wolfshtein (1969), a companion model with closure formulae according with Norris and Reynolds length scale is presented here

after. This model will be referred to from now on with the name LW. Equations of reference as well as values for the coefficients are reported:

$$\frac{\partial(\rho\varepsilon)}{\partial t}+\frac{\partial(\rho\varepsilon U_j)}{\partial x_j}-\frac{\partial}{\partial x_j}\left[(\mu+\frac{\mu_t}{\sigma_\varepsilon})\frac{\partial\varepsilon}{\partial x_j}\right]= C_{\varepsilon 1}\left(P_k+P'_k\right)\frac{\varepsilon}{k}-\rho C_{\varepsilon 2}f_2\frac{\varepsilon^2}{k}\ , \quad (4.79)$$

$$\frac{\partial(\rho k)}{\partial t}+\frac{\partial(\rho k U_j)}{\partial x_j}-\frac{\partial}{\partial x_j}\left[(\mu+\frac{\mu_t}{\sigma_k})\frac{\partial k}{\partial x_j}\right]=P_k-\rho\varepsilon\ . \quad (4.80)$$

Like in CLL model, two damping functions and a modified production term lead all the modification to the original equations:

$$f_\mu = \left[1-\exp\left(-0.016Re_y\right)\right]\left[1-\exp\left(-0.263Re_y\right)\right]^{-1}\ , \quad (4.81)$$

$$f_2 = \left[1-0.3\exp\left(-Re_t^2\right)\right]\ , \quad (4.82)$$

$$P'_k = \frac{\rho C_{\varepsilon 2}f_2k^{1.5}\exp\left(-0.00222Re_y^2\right)}{3.53y\left[1-\exp\left(-0.263Re_y\right)\right]}\ . \quad (4.83)$$

In Tab. 4.7 standard values for the introduced constant are given. Wall

C_μ	$C_{\varepsilon 1}$	$C_{\varepsilon 2}$	σ_k	σ_ε
0.09	1.44	1.92	1.0	1.3

Table 4.7: Standard values for Lien and Leschziner Low Reynolds $k-\varepsilon$ coefficients.

boundary condition for ε are specified in such a way:

$$\varepsilon_w = C_{\varepsilon 1}\frac{P'_k}{\rho C_{\varepsilon 2}f_2} \quad (4.84)$$

For the companion model, called LNR [Lien, 1992], everything but the following remain invariant:

$$f_\mu = \left[1-\exp\left(-0.0198Re_y\right)\right]\left(1+\frac{5.29}{Re_y}\right)^{-1}\ , \quad (4.85)$$

$$D_\varepsilon = \frac{y\sqrt{k}}{y\sqrt{k}+2\nu 0.45C_\mu^{-0.75}}\ , \quad (4.86)$$

$$P'_k = \frac{\rho C_{\varepsilon 2}f_2k^{1.5}\exp\left(-0.00057Re_y^2\right)}{0.42C_{\varepsilon 1}yC_\mu^{-0.75}D_\varepsilon}\ . \quad (4.87)$$

4.6.7 Realizability

Excessive levels of turbulence kinetic energy are predicted by standard two-equation models in region of large rate of strain. This was originally recognized in stagnation point flows, but it will be seen here that it is a more widespread anomaly. The problem is connected with a big overproduction of k, in case a moderate level of k is subjected to a large of rate of strain. Such over production can be due to, remember that k equation is exact, an underestimation of the sink term or/and an overestimation of turbulent viscosity. These ideas could be merged together into a bound for the local turbulent time scale T_s [Medic and Durbin, 2002]. In detail, first step is to reformulate the expression for μ_t:

$$\mu_t = C_\mu f_\mu \rho \, u^2 \, T_s \ , \tag{4.88}$$

where u is the velocity scale and T_s is the local turbulence time scale. For the previously proposed models $u = k$ and $T_s = k/\varepsilon$ and so ε equation is rewritten in terms of T_s:

$$\frac{\partial(\rho\varepsilon)}{\partial t} + \frac{\partial(\rho\varepsilon U_j)}{\partial x_j} - \frac{\partial}{\partial x_j}\left[(\mu + \frac{\mu_t}{\sigma_\varepsilon})\frac{\partial \varepsilon}{\partial x_j}\right] = C_{\varepsilon 1}\frac{P_k}{T_s} - \rho C_{\varepsilon 2}\frac{\varepsilon}{T_s} \ . \tag{4.89}$$

To constrain Reynolds Stress Tensor to be positive defined, a limiter is then applied to T_s:

$$T_s = \min\left[\frac{k}{\varepsilon}, \frac{\alpha}{\sqrt{6}C_\mu \|S^*\|}\right] \ , \tag{4.90}$$

in which $S^* = \frac{1}{2}G$ and $\alpha = 0.6$ usually fits. As a consequence at large rate of strain P_k is growing like $|S|$ rather than $|S|^2$.

The option of selecting the realizability constraint or not has been implemented in all previous models with a switch to be controlled by the user. This correction will be referred to in future chapters, when brevity is a must with the postscript "Real".

4.7 Two Layer

Improvements in well predicting near wall behavior of previous models were not followed by a similar enhancement in the far from the wall region. Usually at the contrary Low Reynolds models spoils good behavior of standard $k-\varepsilon$ in the outer layer. Better would be to join the advantages of the two models. This approach was proposed by [Rodi, 1991]: in which a one equation model $(k-l)$ in the near wall region match standard $k-\varepsilon$ High Reynolds models at a certain patching point. Several variant of this model has been implemented: following Wolfshtein or Norris and Reynolds closure formulae in the wall layer, with standard $k-\varepsilon$ or the RNG version. For simplicity only the Wolfshtein variant is going to be described.

It would be better start with the definition of such a patching point: the location at which the damping function f_μ reaches the value of 0.95. It is better show how f_μ is defined:

$$f_\mu = \left[1 - \exp\left(-\frac{Re_y}{A_\mu}\right)\right] . \tag{4.91}$$

In order to avoid that non-turbulent zones far from the wall were calculated with the one equation model, a user controlled parameter defines the wall layer thickness meaning that above such value f_μ was not checked any more and the standard $k-\varepsilon$ directly applied.

The $k-l$ models are based on the following equations, expressing both dissipation and eddy viscosity via a turbulent length scale:

$$\varepsilon = \frac{k^{\frac{3}{2}}}{l_\varepsilon} , \tag{4.92}$$

$$\mu_t = C_\mu \rho \sqrt{k} l_\mu . \tag{4.93}$$

For this model two characteristic lengths are defined:

$$l_\mu = C_l \, y f_\mu , \tag{4.94}$$

$$l_\varepsilon = C_l \, y \left[1 - \exp\left(-\frac{Re_y}{A_\varepsilon}\right)\right] . \tag{4.95}$$

In Tab. 4.8 standard values for the introduced constants are given.

C_μ	C_l	A_ε	A_μ
0.09	2.5	5	62.5

Table 4.8: Standard values for Two Layer $k-\varepsilon$ coefficients.

4.8 The $k-\omega$ class

The first complete turbulence model was proposed by Kolmogorov in 1942, in addition to the same equation for k, he developed a second equation for the parameter ω representing the rate of dissipation of energy in unit volume and time. The reciprocal of ω serves as a local turbulent time scale while the length scale is prescribed by $k^{0.5}/\omega$. Due to unavailability of computer resources this model went with no application till the 70's when Saffman formulated a $k-\omega$ model able to predict effects of adverse pressure gradients and to integrate through the viscous sub-layer [Wilcox, 1998]. After that, many variants has been proposed making $k-\omega$ models the second most used class of two-equation turbulence models.

As hinted in Sec. 4.5, both $k-\varepsilon$ and $k-\omega$ class of models have been shown their own strengths and weaknesses in the course of the years. $k-\varepsilon$ in particular is recognized as the best choice for free shear flows (mixing layer, jets and wakes) and reached good accuracy also in wall bounded flows. $k-\omega$ certainly performs better in wall bounded flows, especially the ones with adverse pressure gradients, although its freestream values dependency is notorious, in addition it is credited with a higher numerical stability [Bredberg, 2001].

4.8.1 Original $k-\omega$

The model introduced now is proposed in [Wilcox, 1998] as the evolution of the well-known $k-\omega$ model presented by Wilcox in 1988. New dissipation coefficients, maintaining the high precision for boundary layers and remov-

ing overprediction of free shear layer spreading rates, have been introduced. Thus such model is applicable to both wall-bounded and free shear flows and the minimum requirement is satisfied. Definition of eddy viscosity directly follows from the $k-\varepsilon$ class one but due to main authors different notation sometimes there is ambiguity regarding the "old" standard $k-\varepsilon$ C_μ constant. Therefore it is not superfluous to write:

$$\mu_t = \rho\frac{k}{\omega} \ . \tag{4.96}$$

Then the equation for "turbulence frequency" is presented together with standard k equation:

$$\frac{\partial(\rho\omega)}{\partial t} + \frac{\partial(\rho\omega U_j)}{\partial x_j} - \frac{\partial}{\partial x_j}\left[(\mu + \frac{\mu_t}{\sigma_\omega})\frac{\partial\omega}{\partial x_j}\right] = C_{\omega 1}P_k\frac{\omega}{k} - \rho C_{\omega 2}\omega^2 \ , \tag{4.97}$$

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho k U_j)}{\partial x_j} - \frac{\partial}{\partial x_j}\left[(\mu + \frac{\mu_t}{\sigma_k})\frac{\partial k}{\partial x_j}\right] = P_k - \beta^*\rho\omega k \ . \tag{4.98}$$

β^*	$C_{\omega 1}$	$C_{\omega 2}$	σ_k	σ_ω
0.09	0.5555	0.075	2	2

Table 4.9: Standard values for $k-\omega$ coefficients.

Wall boundary condition for ω is specified in such a way:

$$\omega_w = 10.0\frac{6.0\mu}{\rho C_{\omega 2} y^2} \ . \tag{4.99}$$

Again this expression is intended as an imposition on the first near wall node, noticing that the value for ω is going towards infinity as distance from the wall is reducing to zero.

4.8.2 Baseline model

The idea of the base line model is to retain the robust and accurate formulation of the Wilcox $k-\omega$ model in the near wall region, and to take advantage of the free stream independence of the $k-\varepsilon$ model in the outer part of the boundary layer. To achieve this a $k-\omega$ formulation of standard $k-\varepsilon$ is derived and merged together the previous model via a blending function F_1 being one in the near wall region to activate standard $k-\omega$ and zero outside activating $k-\varepsilon$. The blending will take place in the wake region of the boundary layer. The main difference between the models for the two regions is the appearance of an extra cross-diffusion term (scalar products of gradients) in the ω equation [Menter, 1994].

$$\frac{\partial(\rho\omega)}{\partial t}+\frac{\partial(\rho\omega U_j)}{\partial x_j}-\frac{\partial}{\partial x_j}\left[(\mu+\mu_t\alpha_\Omega)\frac{\partial\omega}{\partial x_j}\right]= \frac{C_1P_k}{\nu_t}-\rho C_2\omega^2+\frac{2\rho\alpha_\varepsilon(1-F_1)}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial\omega}{\partial x_j}\ , \quad (4.100)$$

$$\frac{\partial(\rho k)}{\partial t}+\frac{\partial(\rho k U_j)}{\partial x_j}-\frac{\partial}{\partial x_j}\left[(\mu+\mu_t\alpha_K)\frac{\partial k}{\partial x_j}\right]=P_k-\beta^*\rho\omega k\ . \quad (4.101)$$

First the blending function F_1 should be implemented, so we need some auxiliary functions such as:

$$CD_{k\omega} = \max\left[2\alpha_\varepsilon\omega^{-1}\frac{\partial k}{\partial x_j}\frac{\partial\omega}{\partial x_j},10^{-20}\right]\ , \quad (4.102)$$

$$arg_1 = \min\left\{\max\left[\frac{\sqrt{k}}{0.5\omega y},\frac{500\nu}{\omega y^2}\right],\frac{4\alpha_\varepsilon k}{CD_{k\omega}y^2}\right\}\ , \quad (4.103)$$

to finally write:

$$F_1=\tanh\left(arg_1^4\right)\ . \quad (4.104)$$

The four constants are hence blended:

$$\alpha_\Omega = F_1\alpha_\omega + (1-F_1)\alpha_\varepsilon \ , \tag{4.105}$$

$$\alpha_K = F_1\alpha_{k\omega} + (1-F_1)\alpha_{k\varepsilon} \ , \tag{4.106}$$

$$C_1 = F_1 C_{\omega 1} + (1-F_1)C_{\varepsilon 1} \ , \tag{4.107}$$

$$C_2 = F_1 C_{\omega 2} + (1-F_1)C_{\varepsilon 2} \ . \tag{4.108}$$

Values for the introduced constants can be found in Tab. 4.10.

β^*	$C_{\varepsilon 1}$	$C_{\varepsilon 2}$	$C_{\omega 1}$	$C_{\omega 2}$	$\alpha_{k\varepsilon}$	$\alpha_{k\omega}$	α_ε	α_ω
0.09	0.4404	0.0828	0.5976	0.075	1	0.5	0.856	0.5

Table 4.10: Standard values for $k-\omega$ BSL coefficients.

At the first near wall node ω is specified to be:

$$\omega_w = 10.0\frac{6.0\mu}{\rho C_{\omega 2} y^2} \ . \tag{4.109}$$

4.8.3 Shear Stress Transport model

One of the major differences between eddy-viscosity and full Reynolds-stress models is that the latter accounts for the important effect of the transport of the principal turbulent shear stress $\tau_{12} = -\overline{\rho u' v'}$. The base for such an equation is Bradshaw's assumption that the shear stress in boundary layer is proportional to the turbulent kinetic energy:

$$\tau_{12} = \rho a_1 k \ . \tag{4.110}$$

For conventional two-equation models the principal shear stress can be computed as:

$$\tau_{12} = \mu_t \Omega = \rho\sqrt{\frac{\text{Production}_k}{\text{Dissipation}_k}}a_1 k \ , \tag{4.111}$$

where Ω is the vorticity and a_1 is a constant.

In adverse pressure gradient flows the ratio of production to dissipation could be much larger than one, meaning that Bradshaw's hypothesis is substantially violated. To satisfy Eq. 4.110 within the framework of eddy-viscosity models μ_t should be bounded in such a way:

$$\mu_t = \frac{a_1 k}{\max[a_1\omega, \Omega F_2]} \ . \tag{4.112}$$

Of course original formulation of the eddy-viscosity must be maintained for free shear layers so the same blending function approach as for the baseline model is adopted.

First come the equation of conservation for turbulent properties:

$$\frac{\partial(\rho\omega)}{\partial t} + \frac{\partial(\rho\omega U_j)}{\partial x_j} - \frac{\partial}{\partial x_j}\left[(\mu + \mu_t\alpha_\Omega)\frac{\partial\omega}{\partial x_j}\right] = \frac{C_1 P_k}{\nu_t} - \rho C_2\omega^2 + \frac{2\rho\alpha_\varepsilon(1-F_1)}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial\omega}{\partial x_j} \ , \tag{4.113}$$

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho k U_j)}{\partial x_j} - \frac{\partial}{\partial x_j}\left[(\mu + \mu_t\alpha_K)\frac{\partial k}{\partial x_j}\right] = P_k - \beta^*\rho\omega k \ . \tag{4.114}$$

Second the bounding function F_2 is constructed:

$$arg_2 = \max\left[2\frac{\sqrt{k}}{0.09\omega y}, \frac{500\nu}{\omega y^2}\right] \ , \tag{4.115}$$

$$F_2 = \tanh\left(arg_2^2\right) \ . \tag{4.116}$$

Then the blending function is built:

$$CD_{k\omega} = \max\left[2\rho\alpha_\varepsilon\omega^{-1}\frac{\partial k}{\partial x_j}\frac{\partial\omega}{\partial x_j}, 10^{-20}\right] \ , \tag{4.117}$$

$$arg_1 = \min\left\{\max\left[\frac{\sqrt{k}}{0.09\omega y}, \frac{500\nu}{\omega y^2}\right], \frac{4\alpha_\varepsilon k}{CD_{k\omega}y^2}\right\} \ , \tag{4.118}$$

$$F_1 = \tanh\left(arg_1^4\right) \ . \tag{4.119}$$

Finally constants are blended like in BSL model:

$$\alpha_\Omega = F_1\alpha_\omega + (1-F_1)\alpha_\varepsilon \,, \tag{4.120}$$

$$\alpha_K = F_1\alpha_{k\omega} + (1-F_1)\alpha_{k\varepsilon} \,, \tag{4.121}$$

$$C_1 = F_1 C_{\omega 1} + (1-F_1)C_{\varepsilon 1} \,, \tag{4.122}$$

$$C_2 = F_1 C_{\omega 2} + (1-F_1)C_{\varepsilon 2} \,. \tag{4.123}$$

β^*	$C_{\varepsilon 1}$	$C_{\varepsilon 2}$	$C_{\omega 1}$	$C_{\omega 2}$	$\alpha_{k\varepsilon}$	$\alpha_{k\omega}$	α_ε	α_ω	a_1
0.09	0.44035	0.0828	0.59761	0.075	1	0.5	0.856	0.5	0.31

Table 4.11: Standard values for $k-\omega$ SST coefficients.

Wall boundary condition for ω is specified in such a way:

$$\omega_w = 10.0\frac{6.0\mu}{\rho C_{\omega 2} y^2} \,. \tag{4.124}$$

Another variant of the $k-\omega$ SST, later in this thesis also called simply SST, proposed to extend applicability beyond aerodynamic applications has been implemented following the instructions given in CFX guide ANS [2004].

Modifications regards eddy viscosity:

$$\mu_t = \frac{a_1 k}{\max[a_1\omega, \sqrt{2}SF_2]} \,, \tag{4.125}$$

the limiter $CD_{k\omega}$:

$$CD_{k\omega} = \max\left[2\rho\alpha_\varepsilon\omega^{-1}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j}, 10^{-10}\right] \,, \tag{4.126}$$

a bound imposed on production term:

$$P_k = \min\left[G - \frac{2}{3}\rho k\frac{\partial U_j}{\partial x_j}, 10\varepsilon\right] \,, \tag{4.127}$$

and the value of $C_{\omega 1} = 0.5532$.

4.9 Implementing a Turbulence Model in OpenFOAM

This section is aiming at helping future programmers of OpenFOAM in implementing their own turbulence models. It basically is a description of how to define a derived class from the virtual class `turbulenceModel`.

Let's take standard $k - \varepsilon$ as a clear example. It is defined by two files: the header file, kEpsilon.H, and the main file, kEpsilon.C.

First have a look at the header file: at the beginning (line 37-38) the compressiblekEpsilon model is defined, other header files are included (turbulenceModel at line 40) and namespaces are opened (lines 44-48).

Crucial passage is at lines 55-57 where kEpsilon class is inheriting turbulenceModel class with all its members.

Private data are then defined (lines 61-71). Hence TypeName, constructor and destructor are declared (lines 77-93).

Member functions, inherited from virtual class turbulenceModel, are declared to externally access variables such as: `mut()` for turbulent viscosity (lines 99-102), `DkEff()` for diffusion coefficient in the turbulent kinetic energy equation (lines 105-111), `DepsilonEff()` for diffusion coefficient in the turbulent kinetic energy dissipation equation (lines 114-120), `alphaEff()` for effective thermal diffusivity (lines 123-129), `k()`, `epsilon()`, `omega()`.

Two important methods are defined at line 146 and line 148, namely the `volTensorField R()` and the `fvVectorMatrix divRhoR(volVectorField& U)`. These two members are responsible for communication between turbulence model and the turbulent solver. The first one returns the Reynolds stress tensor and the second one the matrix to include viscous and turbulence effects in the momentum equation. The method `correct()` (line 150) is called at every iteration by the solver with the duty of implementing and solving turbulent equations, update and write values for turbulent quantities, especially the eddy viscosity. Last method is the `read()` important to read coefficients needed by the model.

The main file kEpsilon.C follows the same structure of the header file: includes, namespaces and constructors (lines 27-103).

The turbulent stress tensor is calculated at lines 111-129 inside the `R()`:

$$R() = \frac{2}{3}\,k\,I - 2\mu_t symm(\nabla\vec{U}) \;, \tag{4.128}$$

where symm is the operator below defined:

$$symm(A) = \frac{1}{2}(A + A^T) \;, \tag{4.129}$$

and I is the identity matrix. The `divRhoR()` (lines 132-138) is passing the turbulence contribution to momentum equation in an at least partially implicit (the laplacian part) way, rendering the UEqn matrix more stable. Usual form of such a function is:

$$divRhoR(U) = -\nabla\cdot\left(\mu_{eff}\nabla\vec{U}\right) - \nabla\cdot\left(\mu_{eff} dev2\left(\nabla\vec{U}\right)\right) \;, \tag{4.130}$$

where dev2 is the operator below defined:

$$dev2\,(A) = A - \frac{2}{3} tr\,(A)\,I \;. \tag{4.131}$$

After the read declaration, `correct()` is initialized (line 162).

Many important operations are done inside such method: production term is calculated, in tensorial notation:

$$P_k = G - \frac{2}{3}\rho k\left(\nabla\cdot\vec{U}\right) = 2\mu_t(\nabla\vec{U})^T : dev\left(symm(\nabla\vec{U})\right) - \frac{2}{3}\rho k\left(\nabla\cdot\vec{U}\right) \;, \tag{4.132}$$

equations for ε and k are implemented and solved, wall function are imposed at line 156, and finally eddy viscosity is calculated line 196.

Chapter 5

Results

In this chapter the runs for validation of the proposed Low Reynolds models will be presented. At the beginning a general test was conducted over a flat plate with the aim of selecting the best performing Low Reynolds $k-\varepsilon$ models. After this first selection attention was addressed to cases of interests for turbomachinery cooling systems and to models of known better performance (realizable $k-\varepsilon$, Two Layer, $k-\omega$ SST). Simulations have been conducted firstly against standard test, available from literature as reference cases, and secondly against experiments performed at the Energy Engineering Department of the University of Florence.

Real gas turbine cooling systems nowadays combine both internal and external cooling methods with the opposite aim of increasing the internal heat transfer coefficient the first and isolating the metal the second. The two ways of protecting solid walls have been analyzed with tests over impingement and film cooling geometries. First, results for impinging jets are shown: an axial-symmetric impingement test documented by European Research Community On Flow, Turbulence And Combustion (ERCOFTAC) and a three dimensional impingement geometry tested experimentally inhouse in a single and a five holes configuration. Second comes external cooling with the well-known test of a film of coolant over a flat plate by

Sinha for single hole tests and a multiple holes full-coverage test done on an experiment conducted at this department. In addition a last case simulating an internal cooling system with ribs and pedestals is presented. Comparisons are made with an experimental investigation done again at this department for the European project: Aerothermal Investigations on Turbine Endwalls and Blades (AITEB2).

5.1 Generalities

Before starting the analysis of the obtained results, not to weigh down the treatment of the subject, some features common to all simulations will be discussed in this section.

Due to the big number of implemented turbulence models, a shortcut has been used to name most of them: the acronyms before defined and summarized in Tab. 5.1 are widely used in substitution of authors' full name.

Table 5.1: Acronyms for the various turbulence models.

$k-\varepsilon$ Low Reynolds by Abe et al.	AKN
$k-\varepsilon$ Low Reynolds by Chien	CH
$k-\varepsilon$ Low Reynolds by Chen et al.	CLL
$k-\varepsilon$ Low Reynolds by Hwang and Lin	HW
$k-\varepsilon$ Low Reynolds by Lam and Bremhorst	LB
$k-\varepsilon$ Low Reynolds by Lien and Leschziner	LW
$k-\varepsilon$ Low Reynolds by Lien	LNR
Realizability constraint postscript	Real
Two Layer	TL
$k-\omega$ SST	SST

All cases but the flat plate and the internal blade cooling are run with the second order advection scheme SFCD described in Sec. 1.4. Flat plate was run with an upwind scheme while the AITEB2 case with a deferred approach

mixing implicit and explicit derivatives:

$$\Phi_e^{Def,n} = \Phi_e^{UDS,n} + \gamma \left(\Phi_e^{SFCD,n-1} - \Phi_e^{UDS,n-1} \right), \tag{5.1}$$

to obtain blended second order accuracy at convergence, $\Phi_e^{UDS,n} = \Phi_e^{UDS,n-1}$, but convergence properties typical of upwind schemes.

Even if case dependent, it is preferred to give a general indication for the values of suggested underrelaxation factors of the various quantities in Tab. 5.2.

Table 5.2: Typical values for the underrelaxation factors used.

Velocity	$0.5 - 0.7$
Enthalpy	$0.5 - 0.7$
Turbulent kinetic energy	$0.4 - 0.6$
Turbulent dissipation	$0.4 - 0.6$
Turbulent specific dissipation	$0.6 - 0.8$
Turbulent viscosity	$0.8 - 1.0$
Pressure corrector	$0.005 - 0.01$
Density	0.1
α_{mix}	$0.9 - 1.0$

Standardly convergence criteria for the inner cycle are imposed as a relative tolerance, ratio of final residual on initial residual, of $[10^{-3} - 10^{-5}]$ for all quantities but pressure corrector that can be solved less accurately. For outer iteration a good convergence criterion was considered initial residual for all quantities equal to 10^{-6}. Not always however has been possible to obtain such levels of convergence for all quantities.

5.2 Flat plate case

A simple test of a flow over a flat plate has been considered the best choice to start the validation and selection process of the Low Reynolds $k - \varepsilon$ turbulence models. Because of the large amount of accessible data, the ease

and velocity of the test, corrections and tuning could be carried out quickly and accurately.

The flow field being modeled is tested against experimental data reported by Wieghardt and Tillman [1951] and later included in the 1968 AFOSR-IFP Stanford Conference [Coles and Hirst, 1969].

Of particular interest is the possibility of testing correct near wall behavior of difficult to measure quantities such as turbulent kinetic energy and turbulent dissipation. Due to experimental limitation in fact, physical profiles of k and ε can be found almost only for flat plate simulations. Results are so reported in terms of non dimensional k and ε plotted versus non dimensional wall distance at an axial location where flow is fully developed, see Fig. 5.2 and Fig. 5.3.

The geometry of reference is a flat plate 6 meter long with test section in the fully developed zone at a distance $X = 4.687\,m$ from the inlet, posed directly at the beginning of the solid wall. Test conditions result in a weakly compressible, inlet $Ma = 0.2$, fully turbulent flow, inlet turbulent intensity $Tu = 5.76\,\%$ and length scale $l_t = 34.1\,mm$. These inlet conditions were in practice transformed in conditions for velocity, density, turbulent kinetic energy and turbulent dissipation:

$$U = Ma\sqrt{\gamma RT}\,, \tag{5.2}$$

$$\rho = \frac{p}{RT}\,, \tag{5.3}$$

$$k = \frac{3}{2}\|\vec{U}\|^2 Tu^2\,, \tag{5.4}$$

$$\varepsilon = \frac{k^{\frac{3}{2}}}{l_t}\,. \tag{5.5}$$

The plate has been considered adiabatic and the inlet temperature was $T_{in} = 294.4\,K$. Details about applied boundary conditions are listed in Tab. 5.3.

Near wall behavior of both turbulent kinetic energy and turbulent kinetic energy dissipation has been properly checked, through the boundary layer

Table 5.3: Flat plate - Computational boundary conditions.

Inlet temperature	294.4	K
Inlet velocity	68.9	ms^{-1}
Outlet pressure	101400	Pa
Inlet turbulent kinetic energy - k	23.6	m^2s^{-2}
Inlet turbulent dissipation - ε	3365	m^2s^{-3}

up to a very little minimum y^+. Due to the relative simplicity of the case, runs could be performed on a very fine grid: first node $y^+ \approx 0.1$. The total number of cells for the mesh was 17641.

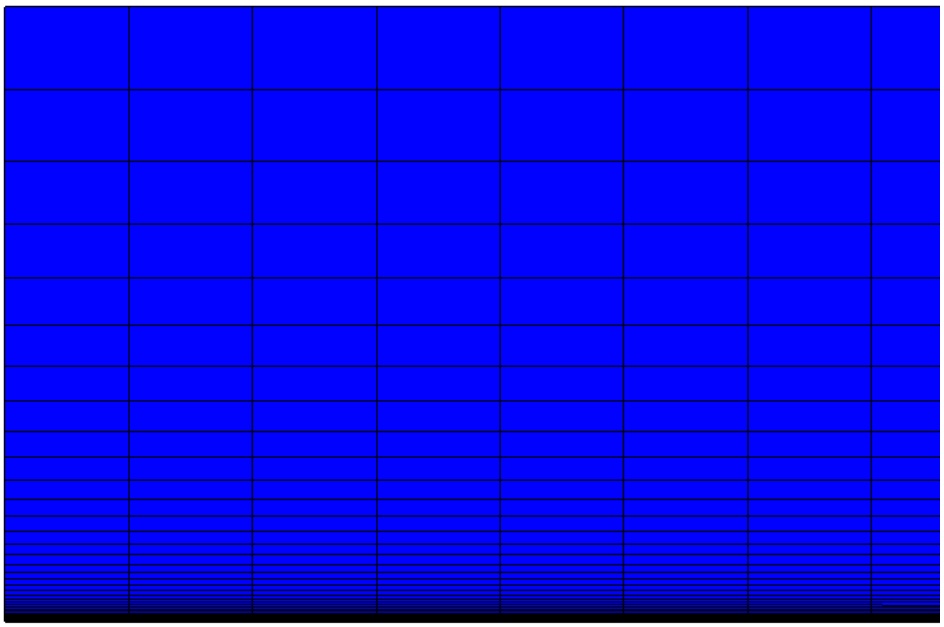


Figure 5.1: Flat plate - Near wall mesh detail.

To drop dimensions from the turbulent kinetic energy and dissipation, the following non-dimensionalizations has been used:

$$k^+ = \frac{k}{u_\tau^2}\,, \tag{5.6}$$

$$\varepsilon^+ = \frac{\nu\varepsilon}{u_\tau^4}\,, \tag{5.7}$$

where:

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}} \tag{5.8}$$

and τ_w is the proper wall shear stress. It is better underline the world proper to point out the differences between Low Reynolds and wall functions treatment. Shear stress is in fact directly computed with velocity wall normal gradient:

$$\tau_w = \|\vec{\tau}_w\| = \|\mathscr{T}\vec{n} - [(\mathscr{T}\vec{n}) \cdot \vec{n}]\,\vec{n}\| \ , \tag{5.9}$$

where $\mathscr{T}$ is the Reynolds stress tensor defined in Eq. 4.20, calculated at the wall. The abscissa is instead reporting the non-dimensional wall distance, that is a sort of wall Reynolds number:

$$y^+ = \frac{u_\tau y}{\nu} \ . \tag{5.10}$$

Apart from CH and HW, all models result to be in good agreement with turbulent kinetic energy experimental data for $y^+ \geq 40$. The tendency, excluded the above mentioned and the LB model, is to totally miss the peak registered for $y^+ \simeq 10$. Lam and Bremhorst estimation viceversa is higher than experimental registration. Level of approximation results in being quite uniform for the different models with CLL and AKN slightly better in overlapping free stream values for k^+.

Also for turbulent dissipation, models well predict outer layer behavior: all models apart from LNR basically coincide for $y^+ \geq 40$. The major disagreements are registered inside the viscous layer. There is no agreement in fact in predicting the peak both in terms of positioning and values. Due to the different boundary conditions imposed to the dissipation, near wall behavior is quite different for each model: LB fails in predicting the peak, AKN and HW results in having pretty high wall values for ε. The models that best suit experimental data reported by Patel et al. [1985] are LNR and CLL.

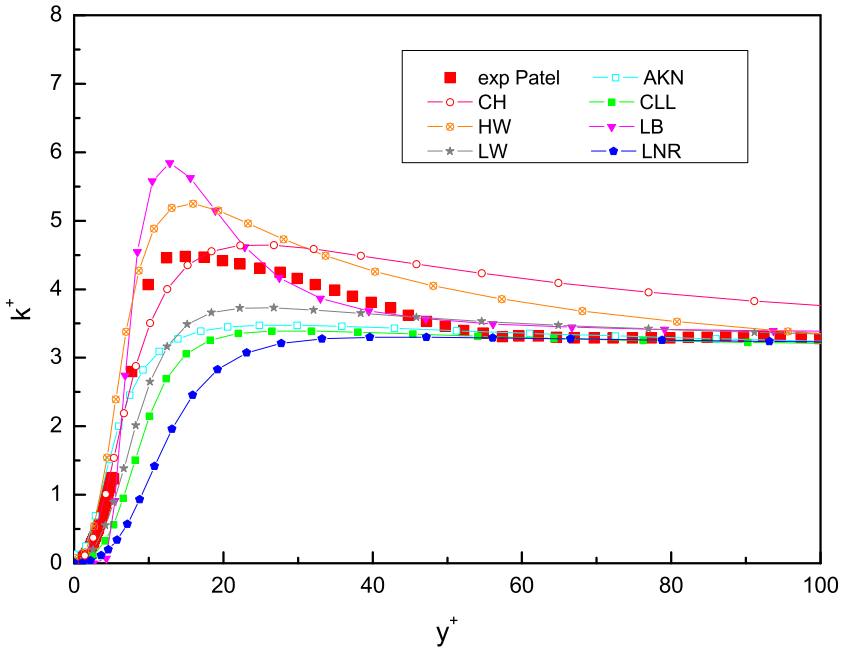


Figure 5.2: Flat plate - k^+ profile.

Of all the tested Low Reynolds models, the CLL model has been chosen as the most reliable one and used as example for Low Reynolds models in most of the following cases.

It is however interesting to show the profiles of the eddy viscosity for some of the models. Profiles look pretty much the same: an high eddy viscosity results from the inlet conditions, then it decreases when the boundary layer is developing and finally a zone of maximum shear stress is found at the end of the boundary layer. Values for μ_t are all fundamentally in agrement at the section of interest where the fluid results fully developed for all the models, with CLL and LNR slightly lower. The two models not presented in these maps, namely the LB and the HW, have shown much higher values for the eddy viscosity: LB predictions were not that far from the other models but

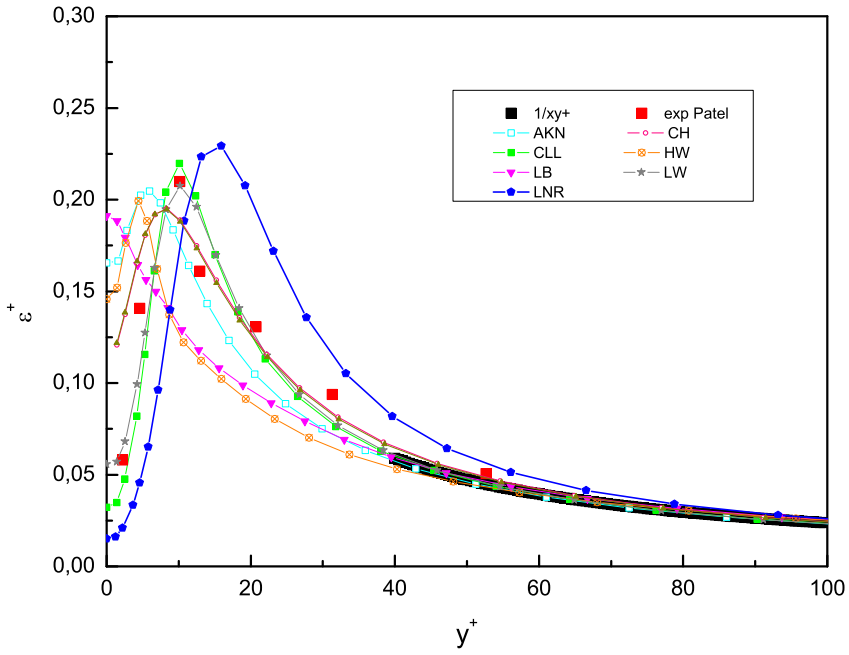


Figure 5.3: Flat plate - ε^+ profile.

HW presented averaged values of μ_t almost 50 times higher.

(a) AKN (b) CH (c) CLL (d) LNR (e) LW

Figure 5.4: Flat plate - Distribution of eddy viscosity along the plate $[kg\ m/s]$.

5.3 Impingement Cooling

Among various possible techniques to enhance internal heat transfer rate, impingement cooling certainly presents very high cooling potentiality, thus it's commonly found both in typical blade and combustor cooling systems, operating in a wide range of design conditions. The idea of such a cooling device is to increase heat transfer using jets of coolant directly blowing over the hot wall. From a numerical point of view, impinging jet flows present several interesting aspects allowing deep evaluation of turbulence models.

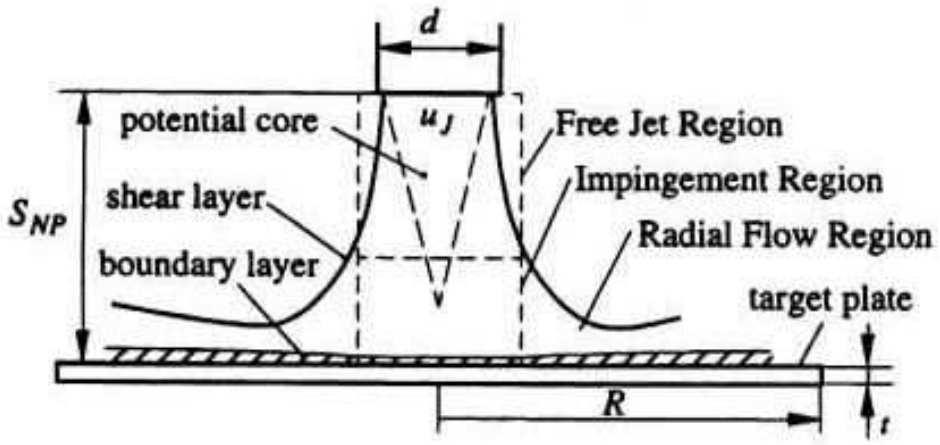


Figure 5.5: Impinging jet.

Normally impinging circular jets can be approximated as axial-symmetric problems. The problem of a 2-D jet of air blowing on a flat plate has been considered as a first test case: the ERCOFTAC C25.

Usually however in turbomachinery impingement the jet is not completely free to develop, in fact the curvature of the walls and the jet to jet interaction laterally bound it. This would not affect simulations so much, what is really interfering with heat transfer predictions is the cross flow generated by the draining of the impinged air. See Fig. 5.6 for a typical example of blade impingement. For combustors cooling the situation in any case is

not so different.

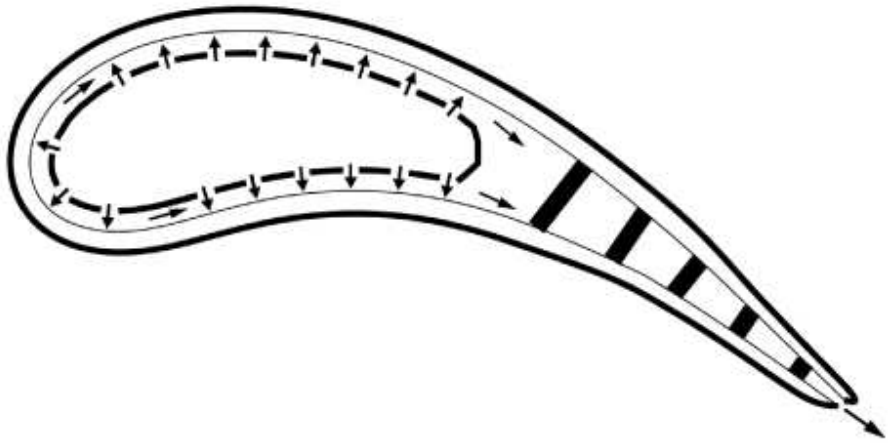


Figure 5.6: Schematic blade impingement cooling.

That is why an experimental set up was organized at the Energy Engineering Department of the University of Florence during the European project LOPOCOTEP (LOw POllutant COmbustor TEchnical Programme). This simulation is a three-dimensional array of jets impinging a flat plate with a wall to bound the outflow from one side. Two different cases were extrapolated from this set up: a first one with just the first row of holes opened and a second one with all the five rows blowing coolant.

After this small introduction it is better start analyzing the cases one by one.

5.3.1 ERCOFTAC case

As above reported, this case has been modeled to reproduce experimental data collected in an extensive set of measurements of a turbulent jet impinging orthogonally onto a large plane surface. In the experimental campaign

test has been done against different changeable parameters such as Reynolds number $Re = 2.3 \cdot 10^4$ and $7 \cdot 10^4$, diameter of the pipe D, the eight of the jet above the plate $Z = 2 - 10\ D$, see Fig. 5.7 to clarify notation. Further reference for the experimental campaign can be found in [Cooper et al., 1993] as well as in the ERCOFTAC online database.

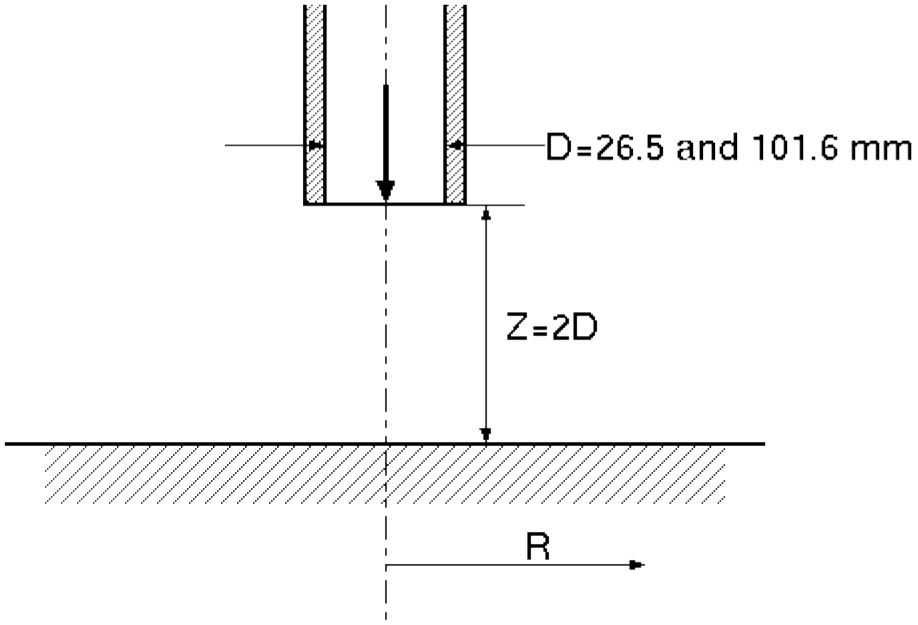
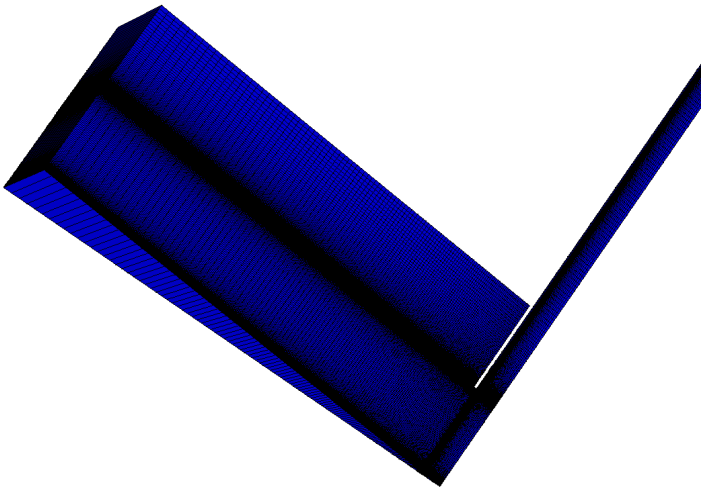


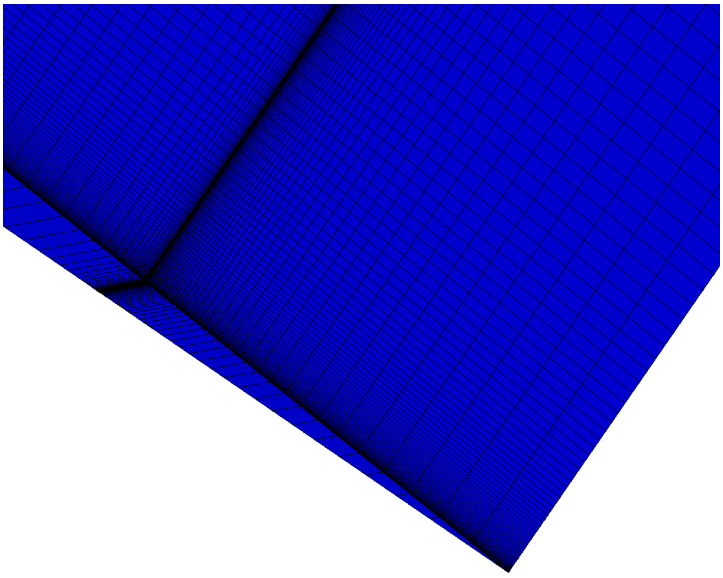
Figure 5.7: ERCOFTAC - Schematic view of experimental set up.

Geometry has been simplified in an axial-symmetric jet. Correct velocity and turbulence conditions at the end of the pipe were obtained from the development of a 50D upstream extruded inlet hole, see Fig. 5.8 for details on the grid. The total number of cells for this case was 68907.

The plate, heated at constant heat flux $\dot{q} = 200\ W\, m^{-2}$, has been computed like a fixed gradient temperature wall (Neumann type boundary condition), while all other walls has been considered adiabatic, zero heat flux. Under these conditions the flow results almost incompressible. Only one test condition was reproduced for validation purposes, with flow conditions



(a) Total



(b) Particular

Figure 5.8: ERCOFTAC - Entire geometry and particular of the grid around the stagnation point.

reported in Tab. 5.4.

Table 5.4: ERCOFTAC - Flow conditions.

Re number	23000	
D	101.6	mm
Z/D	2	
Temperature wall normal gradient	7057.16	Km^{-1}
Inlet temperature	293	K
Inlet velocity	4.53	ms^{-1}
Outlet pressure	100000	Pa
Inlet turbulent kinetic energy - k	0.07695	m^2s^{-2}
Inlet turbulent dissipation - ε	0.3531	m^2s^{-3}

Results for this test are reported in terms of Nusselt number as a function of radial distance from the center. Nusselt number is calculated from the heat transfer coefficient. To better check the predictions, heat flux is calculated again using enthalpy wall gradient. In short, equations to calculate Nu from the known flow field are:

$$\dot{q} = \alpha \frac{\partial h}{\partial n} , \tag{5.11}$$

$$HTC = \frac{\dot{q}}{T - T_{ad}} , \tag{5.12}$$

$$Nu = \frac{HTC \cdot L}{k} , \tag{5.13}$$

where HTC is the heat transfer coefficient, T_{ad} is the adiabatic wall temperature, for this low speed simulation approximately equal to inlet static temperature, L is the characteristic length: the diameter of the pipe, and $k = C_p\,\alpha$ is the thermal conductivity.

The turbulence models tested for this case are, after the selection over the flat plate: CLL, CLL realizable, AKN, Two Layer, $k - \omega$ SST.

The predictions of all two-equation models used in this validation case are in good agreement with the experimental data far from the stagnation point. As known in literature [Durbin and M. Benhia, 1998], in the area around the

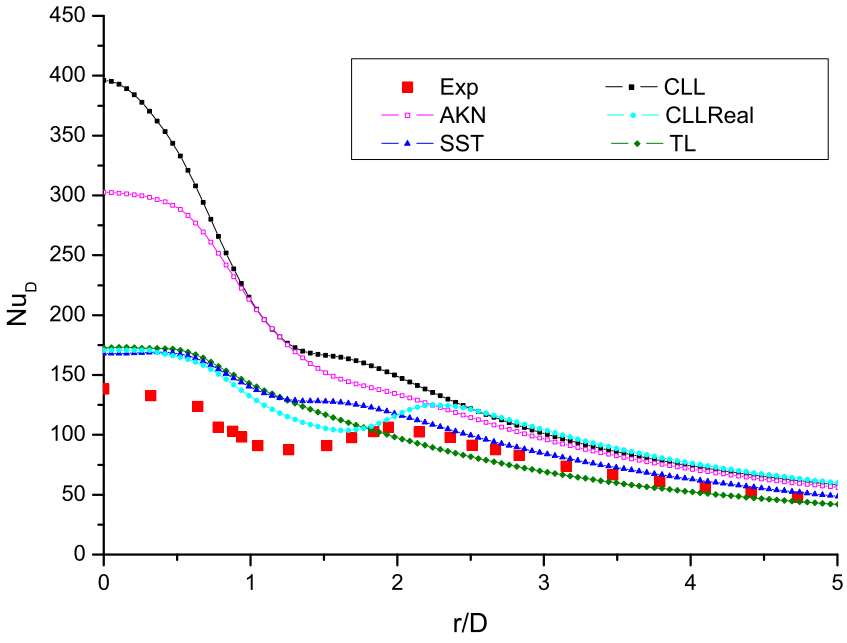


Figure 5.9: ERCOFTAC - Nusselt number distribution along radius.

stagnation point, $k-\varepsilon$ Low Reynolds models without realizability constraint fail and dramatically overpredict the peak in the heat transfer coefficient (error of almost 200%).

At the same time Two Layer, SST and realizable models show about the same peak value. The local maximum at $r/D \approx 2$ is not seen by the Two Layer and is slightly predicted by the SST. On the contrary, the realizable CLL is well predicting such peak only shifting a bit towards higher values of the radius.

Interesting could be to have a look at the flow field and at the predictions for turbulent kinetic energy. The velocity and pressure distributions really match one with the other with no possibility of differentiating the maps, they all look like the one reported in Fig. 5.10 that is the AKN.

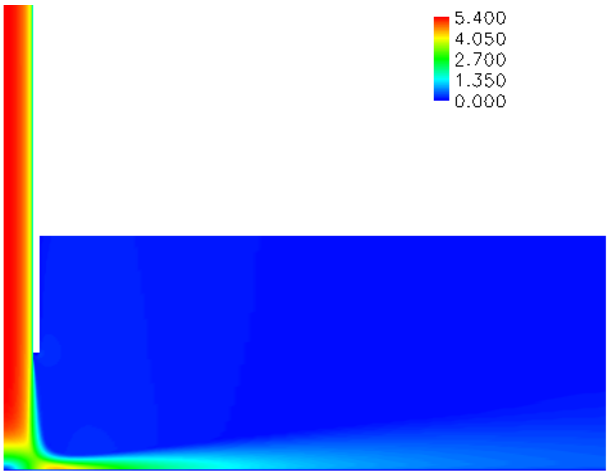


Figure 5.10: ERCOFTAC - Velocity magnitude on the symmetry plane $[m/s]$.

For turbulent kinetic energy one cannot say the same, see Figs. from 5.11 to 5.15: first of all it may be noticed how two of the models, the realizable CLL and the $k-\omega$ SST, have an almost laminar core in the jet that is going up to the wall. What happen in fact is that both these two models limit production term in k equation when the rate of strain is high in this case near the stagnation point. It is shown finally how the $k-\omega$ SST implicitly satisfy the "realizability" constraint proposed in Sec. 4.6.7 thanks to the limit on the eddy viscosity, see Eq. 4.112. The same effect can be seen, even though with less clearness, in the next cases too, see Fig. 5.20. In any case the trends for all models are in very good agrement, with again the $k-\omega$ SST a little more diffusive.

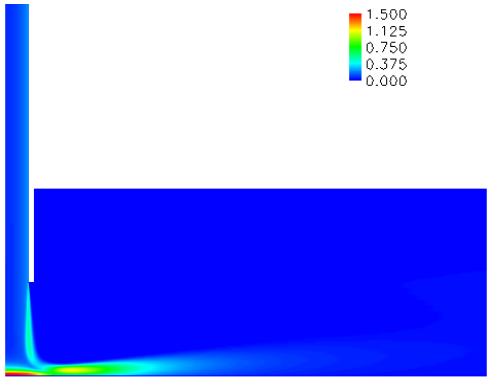


Figure 5.11: ERCOFTAC - AKN turbulent kinetic energy distribution on symmetry plane $[m^2/s^2]$.

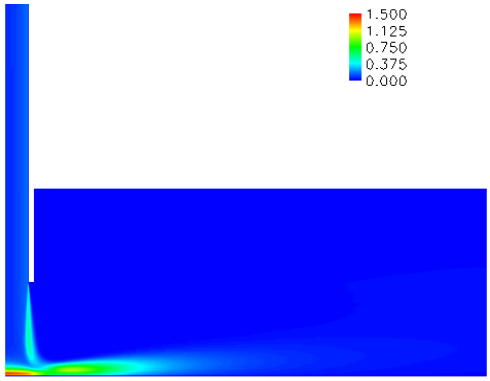


Figure 5.12: ERCOFTAC - CLL turbulent kinetic energy distribution on symmetry plane $[m^2/s^2]$.

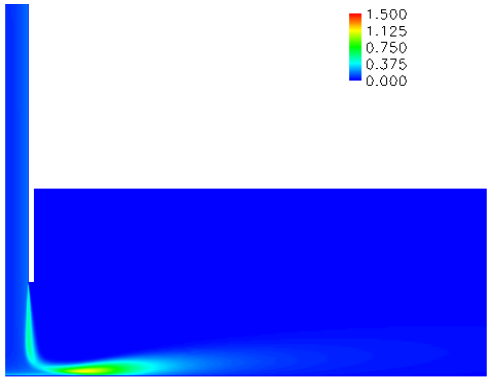


Figure 5.13: ERCOFTAC - CLLReal turbulent kinetic energy distribution on symmetry plane $[m^2/s^2]$.

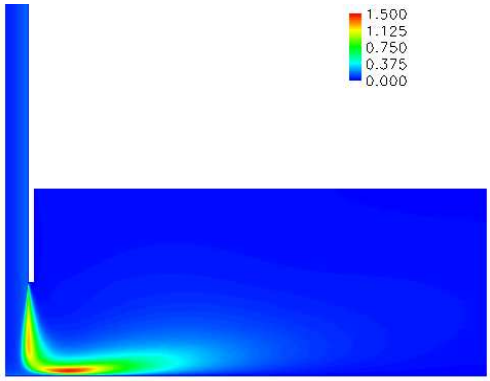


Figure 5.14: ERCOFTAC - SST turbulent kinetic energy distribution on symmetry plane $[m^2/s^2]$.

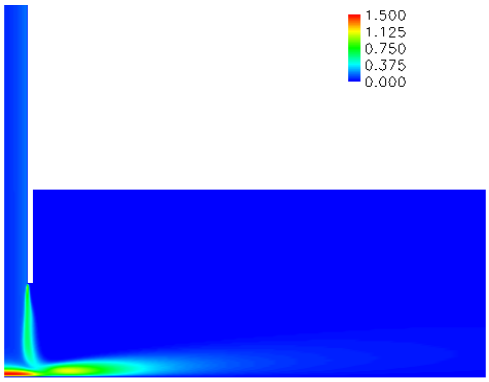


Figure 5.15: ERCOFTAC - TL Norris and Reynolds turbulent kinetic energy distribution on symmetry plane $[m^2/s^2]$.

5.3.2 Single hole case

For further validation of the three best performing models from the previous simulation: Two Layer, $k-\omega$ SST, realizable CLL model, a test against in-house collected experimental data seemed the best solution. This case is therefore modeled to reproduce the following situation: coolant is injected from a plenum through a perforated plate and impacts over a flat plate at uniform heat flux. The holes on the plenum compose an array of $10-11$ spanwise rows per 9 streamwise holes. This test is simulating the behavior of the first row only, while in the following $3-2$ rows (the array of holes is staggered) for a total of 5 jets are impinging. For further details refer to [Facchini and Surace, 2006, Surace, 2004].

As can be seen from the grid, see Fig. 5.16, jet spreading is bounded by a wall from one side; the other two vertical boundaries are symmetry planes: a cutting hole plane and a hole-to-hole symmetry. Main geometrical parameters are the diameter of the pipe $D = 4\,mm$ and the height of the

jet over the plate $Z = 3.75\,D$. The grid on which simulations were run was globally composed by 386732 cells.

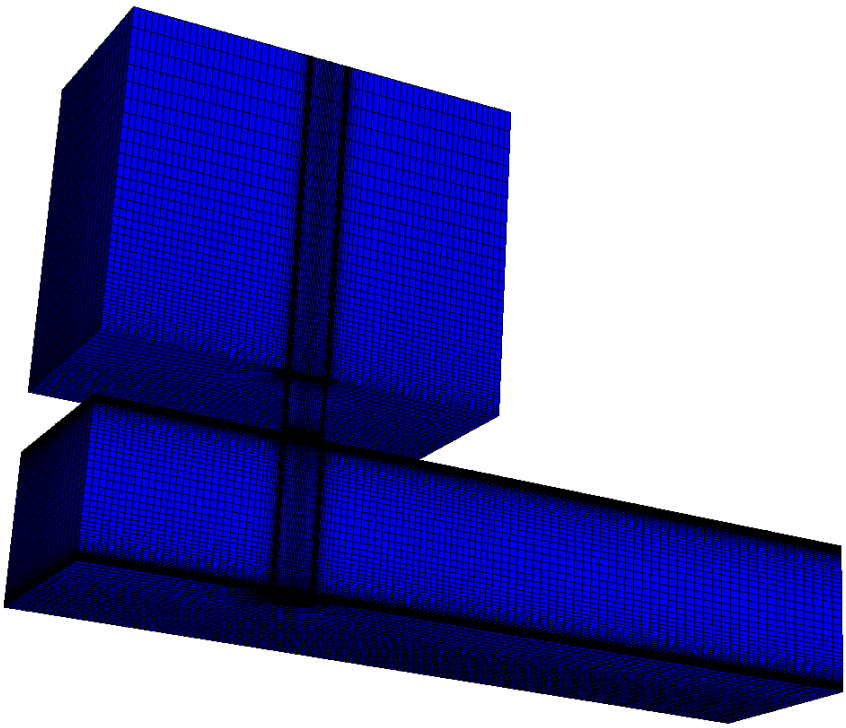


Figure 5.16: Impingement single hole - Grid.

Main flow parameters can be seen in Tab. 5.5, these conditions have been applied at the computational domain as the boundary conditions presented in Tab. 5.6.

Simulations have been validated in terms of heat transfer coefficient calculated with respect to inlet static temperature circa coincident to inlet total temperature. Adiabatic simulations have been done too, in order to check whether this approximation could be done or not. Heat transfer coefficient is plotted along the cutting hole symmetry line versus non-dimensional distance from the stagnation point.

Table 5.5: Impingement single hole - Flow conditions.

Inlet Total Temperature	308.2	K
Inlet Total Pressure	86341	Pa
Outlet Pressure	85101	Pa
Inlet Turbulence intensity - Tu	unknown $\rightarrow \leq 0.5\%$	
Turbulence length scale - l_t	unknown $\rightarrow 0.10\,D$	
Inlet mass flux - $\dot{m}$	0.452	$kg\,m^{-1}$
Wall Heat flux	3000	$W\,m^{-2}$

Table 5.6: Impingement single hole - Computational boundary conditions.

Inlet Temperature	308.2	K
Outlet Pressure	85101	Pa
Inlet turbulent kinetic energy	$3.145 \cdot 10^{-6}$	$m^2 s^{-2}$
Inlet turbulent dissipation	$2.292 \cdot 10^{-6}$	$m^2\,s^{-3}$
Inlet Velocity	0.28956	$m\,s^{-1}$
Temperature wall normal gradient	111649	$K\,m^{-1}$

First thing to notice from Fig. 5.17 is that, contrarily to ERCOFTAC test, realizable CLL model fails in well predicting heat transfer coefficient around the stagnation point. Moreover, due to the potential core that is not extinguished at the wall, two unphysical spurious peaks are predicted at $X/D \approx 1$.

Two Layer and SST result in being almost equivalent both for the peak level and the far from the stagnation point values, with the Two Layer predictions slightly lower everywhere on the impinged surface.

From a designer point of view however, what is more important is a parameter that well represent some mean value of the heat transfer coefficient. Averaging has not been done for this case so, to give an idea of the bi-dimensional distribution of HTC over the plate, map of wall heat transfer coefficient are contrasted in Fig. 5.18. Scale of reference is the same used for the experimental results so it is obviously scaled on its maximum 375 and

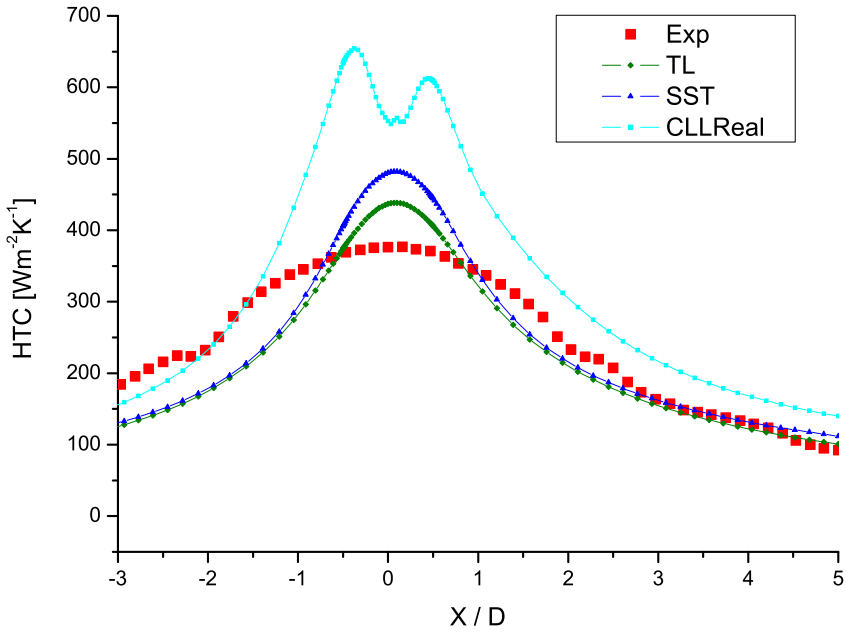
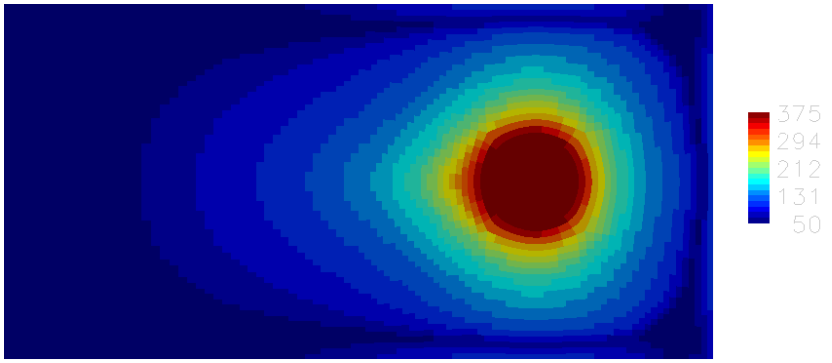


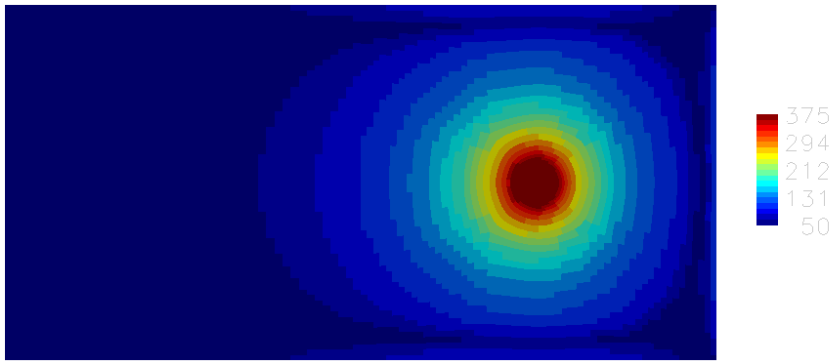
Figure 5.17: Impingement single hole - Heat transfer coefficient on impinged wall along symmetry line.

minimum value 50 $[W\,m^{-2}\,K^{-1}]$.

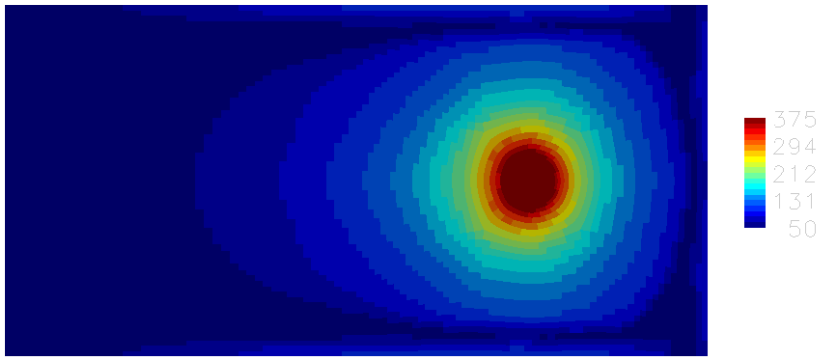
It can be noticed how the shapes of these isolines differ: Two Layer presents almost circular contour lines while realizable CLL tends to have higher values downstream on the symmetry line at the same distance from the stagnation point. The SST model lies in between this two bounds. This tendency is however smoothed downstream. What is interesting is the bubble of higher HTC right on the hole-to-hole symmetry plane. This is due by the interaction of the jet with the wall confining the flow in addition at the jet to jet interaction. The effect of non constrained single jet in fact decreases with the distance but in this case practically the mass flowed towards the right side, see Fig. 5.18, cannot be drained on that side because of the solid wall. Streamlines are thus curved by the wall and, not powerful enough to



(a) Realizable Chen Lien Leschziner



(b) Two Layer



(c) k-ω SST

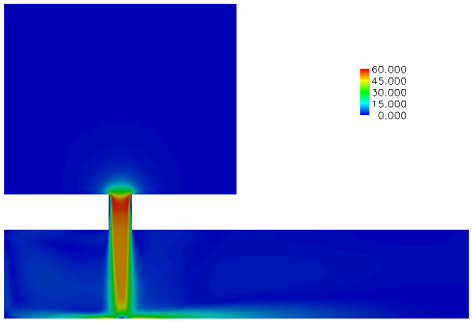
Figure 5.18: Impingement single hole - Heat transfer coefficient distribution on impinged wall $[W/(m^2K)]$.

penetrate the jets, goes in between them. Crossflow rounds the reference jet and increases the velocity near the symmetry plane. This results in an higher than expected HTC.

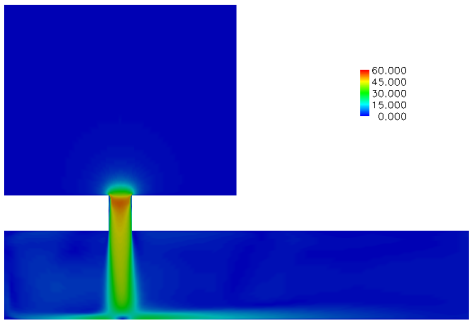
In order to check which of the three distributions is more physical, probably comparisons are better done evaluating the flow field on the symmetry plane: velocity and turbulent kinetic energy are reported in Fig. 5.19 and Fig. 5.20.

Both velocity magnitude and k do not perfectly agree in terms of qualitative distributions, but if for velocity contour levels pretty much coincide, for turbulent kinetic energy this is not true, and realizable CLL presents a maximum value more than two times lower with respect to the other two models. It has already been underlined in Sec. 5.3.1, that also $k-\omega$ SST uses a sort of "realizability" constraint, in this case though, the low k zone is much less developed and values stand in the middle between the Two Layer and the realizable CLL. It is interesting to see how the really low values of viscosity inside the potential core of the $k-\varepsilon$ Low Reynolds model, increase velocity and the higher heat transfer coefficient is easily explained.

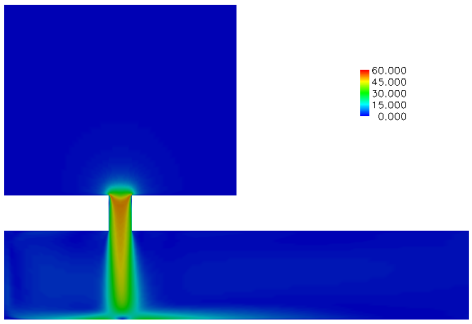
Very important in validation process for new CFD packages is the testing against equivalent models released in commercial codes. A very wide investigation of the same geometry has been done with StarCD by Di Carmine [2004]. Comparisons between the results are in perfect agreement one with the other in terms of a qualitative analysis. Quantitatively speaking the differences between the two results are however more than acceptable with maximum errors for the heat transfer coefficient, on peak values, of the order of 5%.



(a) Realizable CLL

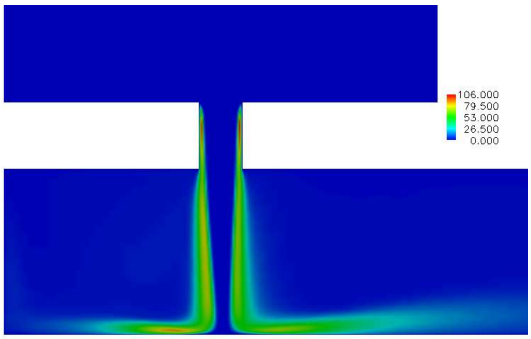


(b) Two Layer

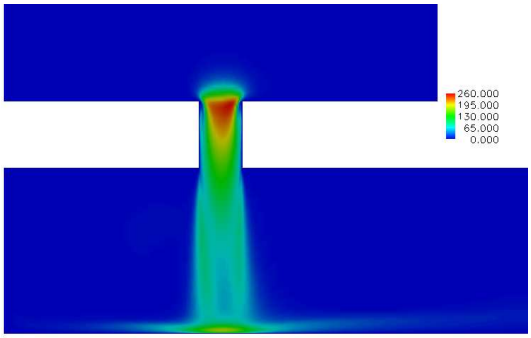


(c) k-ω SST

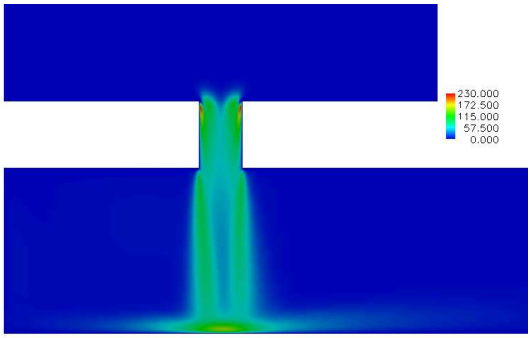
Figure 5.19: Impingement single hole - Velocity magnitude on cutting hole symmetry plane $[m/s]$.



(a) Realizable CLL



(b) Two Layer



(c) k-ω SST

Figure 5.20: Impingement single hole - Turbulent kinetic energy on cutting hole symmetry plane $[m^2/s^2]$.

5.3.3 Multiple holes case

This case refer to the same set of experiments of the previous test. As said the number of jets is increased to 5: the grid is doubled and basically two consecutive rows of holes are cut by the symmetry lines. Being a staggered pattern of holes, the holes results three on the first symmetry line and two on the second. For this multi-hole simulation the plenum is bigger and the imposed inlet mass flow is approximately five times more than single hole case. Computational boundary conditions follow exactly the previous single hole test. In any case a much heavier grid, 1704740 cells, was necessary, see Fig. 5.21.

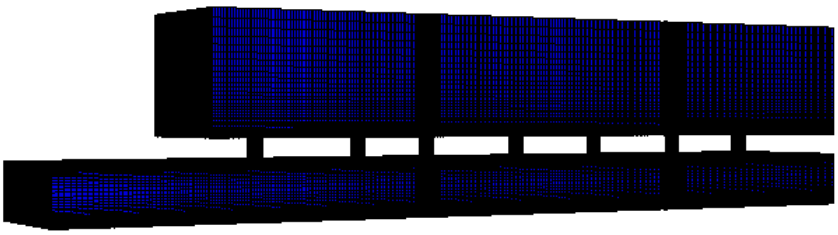


Figure 5.21: Impingement five holes - Grid.

For this geometry, only the Two Layer and k-ω SST models have been tested against experimental results, again in terms of heat transfer coefficient on the impinged wall.

First the heat transfer coefficient on the symmetry line is plotted as a function of the non-dimensional distance from the first hole stagnation point. In this case however, there are two cutting holes symmetry lines so both experimental and numerical data are sampled onto the two different symmetry lines (path1 and path2) and then merged together in the zone where a relative minimum is localized, see Fig. 5.22.

Even if obtained results are in good agreement with experimental data

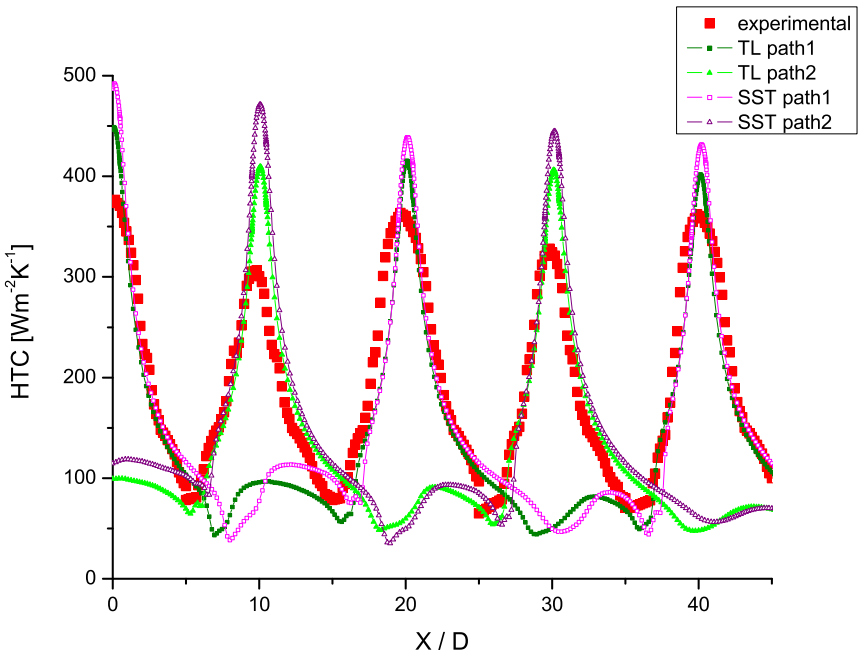
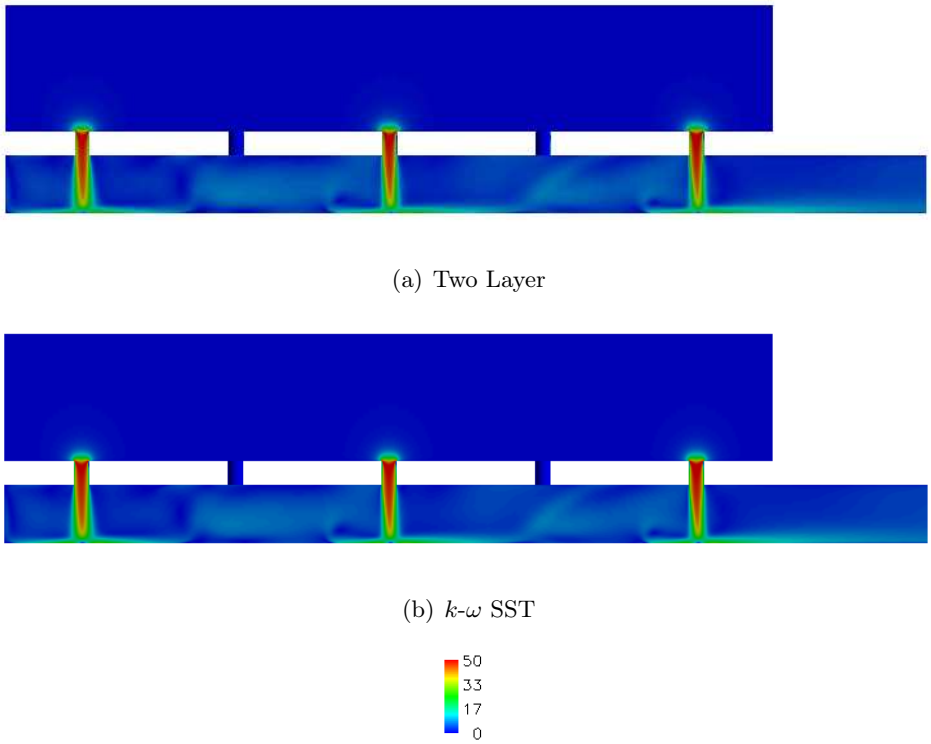


Figure 5.22: Impingement five holes - Heat transfer coefficient along center lines.

far from the stagnation point, it should be noticed that predictions for the peak values are quite different from measured data. Higher discrepancies on the even peaks are probably due to errors in the experimental measurements [Facchini and Surace, 2006]. Comparing the two models, Two Layer predicts peak values a 10% better of the k-ω SST giving basically identical results outside the stagnation points area. In any case, it should be considered that temperature gradients are quite small: a better agreement is expected for higher values of wall heat flux.

Again symmetry line behavior of HTC is not a really good parameter to establish whether or not the cooling device is effective. Even though it quite well shows maximum and minimum values, no information on the mean value, felt by the wall, are obtained. A wall HTC map is hence presented in



(a) Two Layer

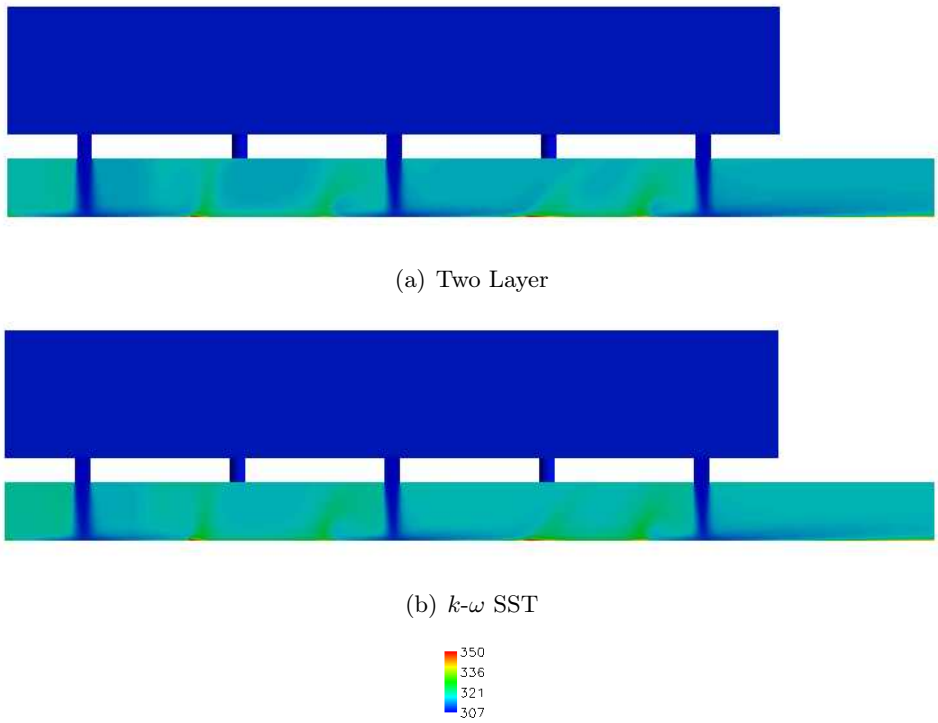
(b) k-ω SST

Figure 5.23: Impingement five holes - Velocity magnitude on symmetry plane $[m/s]$.

Fig. 5.25 to both give an idea of an averaged value and permit an evaluation of flow structures. For this last purpose is of particular help the picture of the flow field reported in Fig. 5.23 and Fig. 5.24.

As one may notice either in the velocity or in the temperature plot, the two models are perfectly in agreement in the prediction of the structure of motion. The effects due to cross flow over the plate magnify from left to right obviously following the increase in mass flow. Moreover the two columns of holes are not completely decoupled, the effect of such coupling is seen in the smearing on the symmetry plane at the height of the corresponding jet as well as in the increase of heat transfer coefficient, see Fig. 5.22, on the opposite path.

As already pointed out for single hole configuration, the decrease of HTC

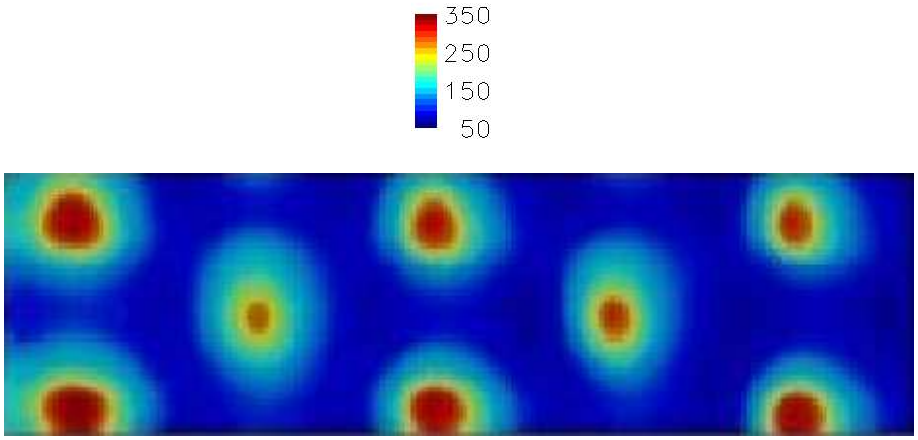


(a) Two Layer

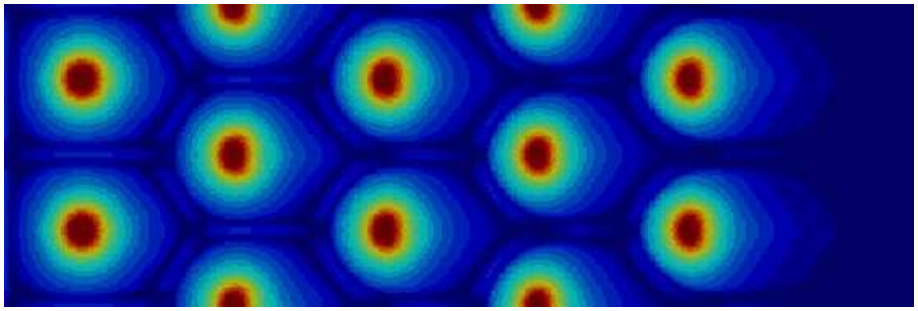
(b) k-ω SST

Figure 5.24: Impingement five holes - Temperature distribution on symmetry plane [K].

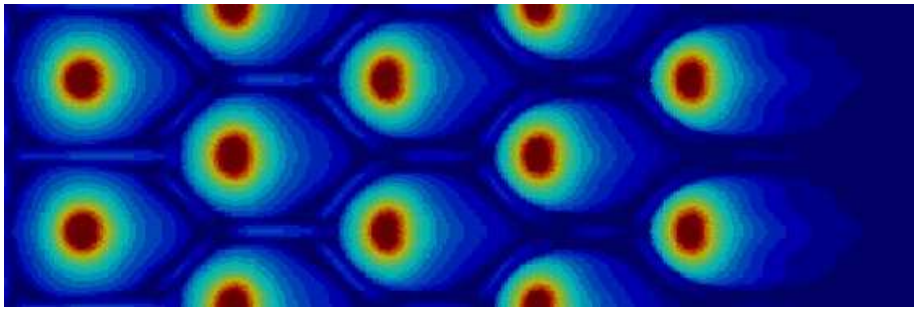
downstream the jet is faster for the Two Layer than the other model, resulting in more roundish contour plots. The bubbles of higher heat transfer coefficient due to passage of more streamlines deviated by the jet now multiply because of the interaction of the same phenomenon on different adjacent holes. This can be noticed also in Fig. 5.22, looking at the local maximum in correspondence with the hole on the opposite path.



(a) Experimental



(b) Two Layer



(c) k-ω SST

Figure 5.25: Impingement five holes - Heat transfer coefficient distribution on impinged wall $[W/(m^2K)]$.

5.4 Film Cooling

Among the different techniques used to cool the hot parts of internal engines, one of the most used is the so called film cooling. It basically consists of an injection of cooling air from the wall to cool, into the hot gas flow with the aim of producing a thin film of cold fluid that isolates the walls from the main flow.

Figure 5.26: Typical film cooling holes on a gas turbine blade.

Because of the complex interaction between air and hot gases during mixing, many different injection hole shapes and distributions have been studied and a great amount of research work is still on going.

First step done has been again the validation of some of the models proposed against a case well-known in literature. For this family of situations, classified as film cooling simulations, the main reference is the experimental investigation done by Sinha et al.. It simply is a study of a single jet mixing in cross flow over a flat plate. Single row configuration was mainly consid-

ered in order to have a reference geometry to compare results for different turbulence models.

As can be clearly seen in Fig. 5.26 in fact, single jet simulation over a flat plate are not of particular interest for turbomachinery design. Multi jet configurations are the standard and should therefore be investigated.

In particular, most recent developments in drilling capabilities allow the manufacturing of wide arrays of micro-holes (diameters below $1\,mm$), currently referred to as effusion cooling. Even if this technique does not produce a film wall protection as in standard film cooling, its most efficient cooling feature is the heat removed by the passage of coolant inside the holes (heat sink effect): the great number of holes and their high length/diameter ratio (with angles below 30°) allows to heavily increase the overall cooling effectiveness [Gustafsson, 2001]. Effusion cooling represents the base in the thermal design of modern aero-engine combustors and its use in the cooling of turbine endwalls is also investigated. Even if film wall protection may not represent the main cooling effect in effusion technique, the prediction of mixing between coolant and cross flow and the corresponding assessment of adiabatic effectiveness, still represent one of the most difficult task in CFD analysis [Andreini et al., 2005].

Second case for the film cooling family was so chosen to represent experimental investigations done on an effusion cooling system done at the Energy Engineering Department of the University of Florence [Arcangeli et al., 2006]. These experiments represent a real full coverage turbine end-wall cooling device.

5.4.1 Sinha case

As above introduced, experimental data and geometries for this case are based on tests made by Sinha et al. [1991]. The geometry is a flat plate with a single row of circular holes obliquely connecting the main fluid domain with

a plenum at higher pressure. A jet of coolant is as a consequence entering from the hole into the hot gas domain, see Fig. 5.27.

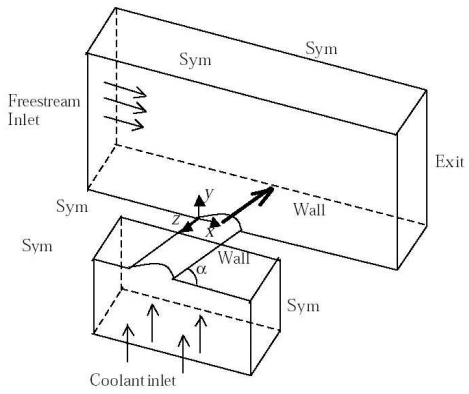


Figure 5.27: Sinha - Calculation domain.

It is important to report some fundamental geometrical dimensions such as hole diameter $D = 12.7\,mm$ and coolant inlet area $A_{ci} = 1.210\,mm$.

Computational domain, as usual, reproduce the minimum periodic geometry meaning that two symmetry planes are imposed, one passing through hole axis and the other half spanwise pitch. Of particular importance for this case is the computational grid in the zone around the hole, where the jet entrains in the mean flow. Magnifications of the used mesh are reported in Fig. 5.28, globally the mesh is composed by 177468 cells.

Experimental flow conditions are listed in Tab. 5.7, and are traduced in computational boundary conditions listed in Tab. 5.8.

The performances of five turbulence models were analyzed: the two Low Reynolds $k - \varepsilon$ models CLL and AKN, with the CLL also in the realizable variant, the Two Layer and the $k - \omega$ Shear Stress Transport.

Results were checked in terms of adiabatic effectiveness, a parameter that describe how much wall temperature is influenced by hot gas temperature

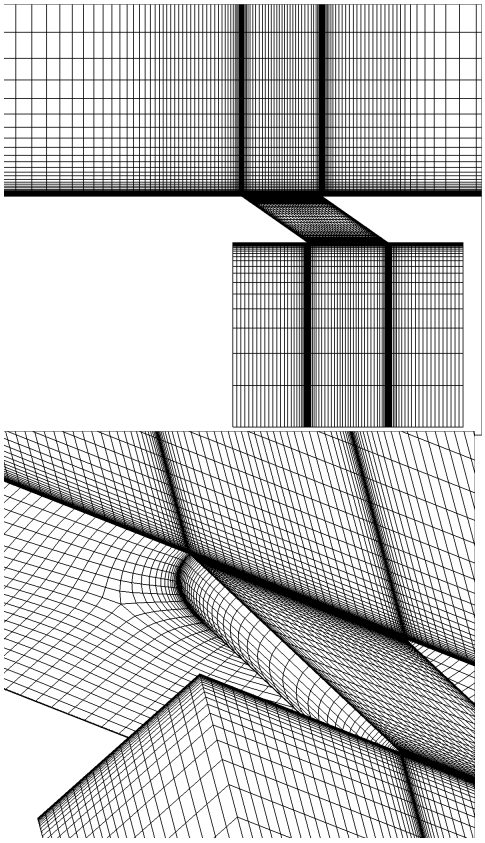


Figure 5.28: Sinha - Near wall mesh details.

Table 5.7: Sinha - Flow conditions.

Cross flow temperature	300	K
Coolant temperature	153	K
Pressure	101325	Pa
Density ratio - DR	2.0	
Blowing-rate - M	0.5	
Momentum ratio - I	0.125	
Turbulence level - Tu	≤ 0.2	%
Cross flow velocity	20	$m\,s^{-1}$
Re_c	15700	
Wall heat flux - $\dot{q}$	0	$W\,m^2$

Table 5.8: Sinha - Computational boundary conditions.

Freestream inlet temperature	300	K
Coolant inlet temperature	153	K
Outlet pressure	101325	Pa
Freestream inlet velocity	20	ms^{-1}
Coolant inlet velocity	0.2618	ms^{-1}
Freestream inlet turbulent kinetic energy	0.002385	m^2s^{-2}
Coolant inlet turbulent kinetic energy	$4.19 \cdot 10^{-7}$	m^2s^{-2}
Freestream inlet turbulent dissipation	7.021	m^2s^{-3}
Coolant inlet turbulent dissipation	0.09305	m^2s^{-3}
Temperature wall normal gradient	0	$K\,m^{-1}$

or by the coolant:

$$\eta = \frac{T_w - T_g}{T_c - T_g} \ , \tag{5.14}$$

where T_w is wall local temperature, T_g stands for hot gas temperature and T_c for coolant temperature.

To validate the behavior of the different turbulence models, local effectiveness is plotted along spanwise lines at different distances downstream the hole, see Figs. 5.29, 5.30, 5.31.

The general trend of all the numerical simulations is to fairly be less diffusive than experimental survey in lateral direction, resulting more efficient than measured data near the symmetry plane and at the same time faster loosing effectiveness moving spanwise. This tendency is not really evident at 1D, where also experiments predict a coherent jet, but increases streamwise, the coherency of the jets can be seen in Fig. 5.32. Similar results are found also with commercial solvers [Andreini et al., 2005].

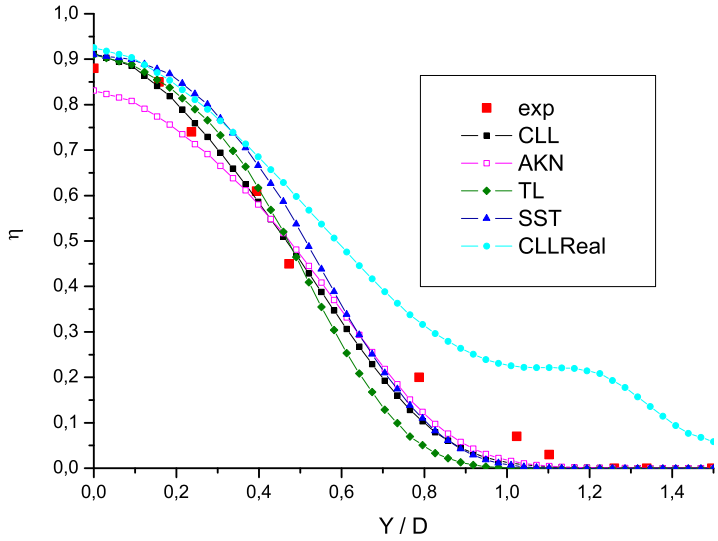


Figure 5.29: Sinha - Effectiveness spanwise distribution at X = 1D.

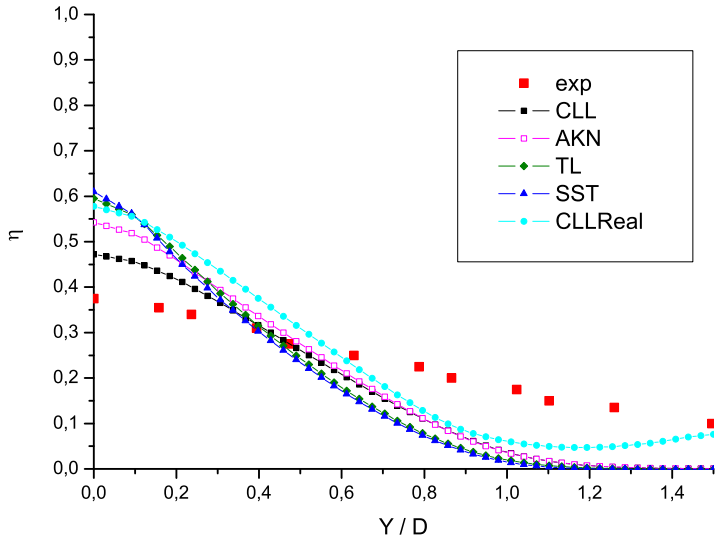


Figure 5.30: Sinha - Effectiveness spanwise distribution at X = 10D.

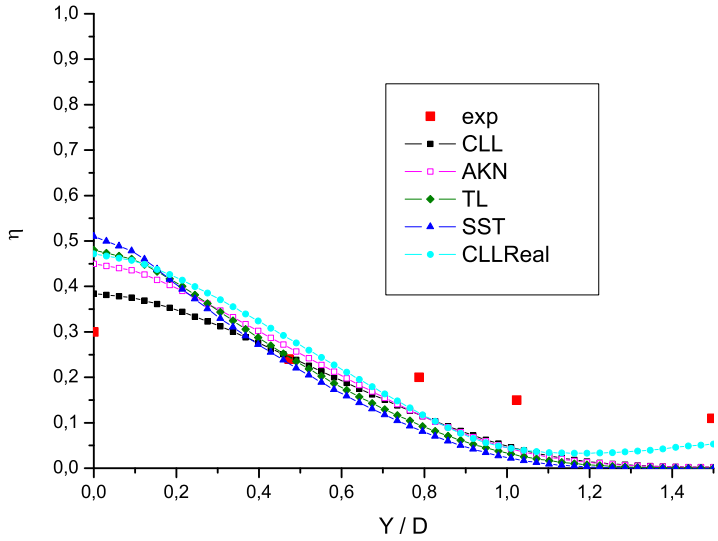


Figure 5.31: Sinha - Effectiveness spanwise distribution at X = 15D.

This behavior, known in literature [Azzi and Lakehal, 2002, Azzi and Jubran, 2003, Lakehal, 2002], is due to the assumption of turbulence isotropy that results in an under prediction of the $\overline{\rho u'v'}$ term in Reynolds stress tensor. There is no way of avoiding this failure except to dope Boussinnesq assumption magnifying the ratio between spanwise and normal fluctuations:

$$\gamma = \frac{\overline{v'^2}}{\overline{w'^2}} \,. \tag{5.15}$$

The cross stresses for spanwise direction should therefore be calculated using a different eddy viscosity. Standard expression for Reynolds stress tensor suddenly changes to:

$$-\overline{\rho u_i' u_j'} = \frac{\mu_{t,ij}}{\mu_t} \mathscr{T}, \tag{5.16}$$

where

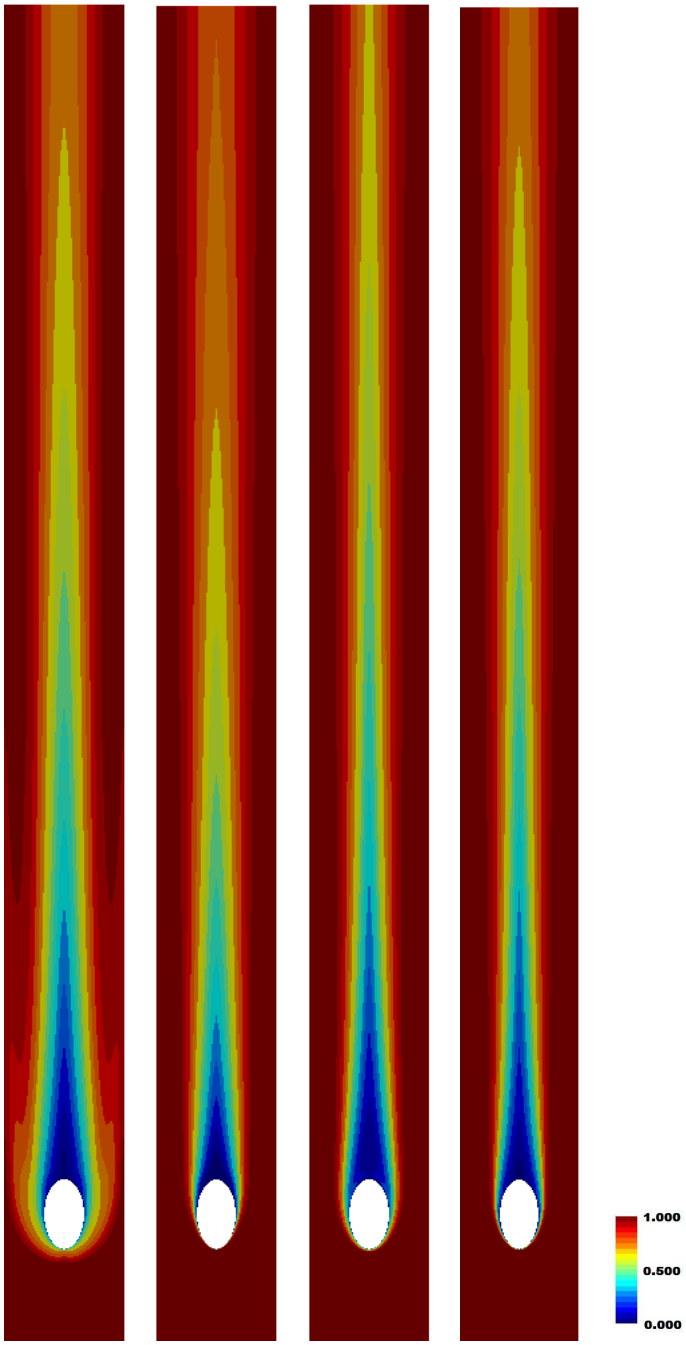
$$\mu_{t,ij} = \begin{pmatrix} \mu_t & \gamma\mu_t & \mu_t \\ \gamma\mu_t & \mu_t & \mu_t \\ \mu_t & \mu_t & \mu_t \end{pmatrix} \,. \tag{5.17}$$

In the above derivation x is the streamwise direction, y the spanwise direction and z axis points normally to the wall.

Apart from the realizable CLL that is giving quite out of range prediction one diameter downstream, all models are substantially equivalent, giving a fairly good agreement between numerical and experimental results. To better understand why the realizable CLL is so overpredicting the effectiveness for $y/D > 0.8$, you can refer to Fig. 5.32.

The baffle underlined at the end of the spanwise direction may be a consequence of a poor developing of the boundary layer predicted by the model. Low values of turbulent viscosity do not dissipate the horse-shoe vortex upstream of the hole. A low pressure zone drives coolant gas coming from the plenum around the jet. As a consequence there is more spreading of the film in spanwise direction and the evidence is a local maximum for the effectiveness at $Y/D \approx 1.2$. This behavior is not limited to this model and even if less accentuated can be seen also in simulations with commercial solvers.

Centerline local η is an important parameter to show the decay of effectiveness in flow direction and being so able to decide where to drill again in case a limit on temperature is imposed at the wall. Its behavior is shown in Fig. 5.33.



(a) CLLReal (b) CLL (c) SST (d) TL

Figure 5.32: Sinha - Effectiveness distribution over the wall [ ].

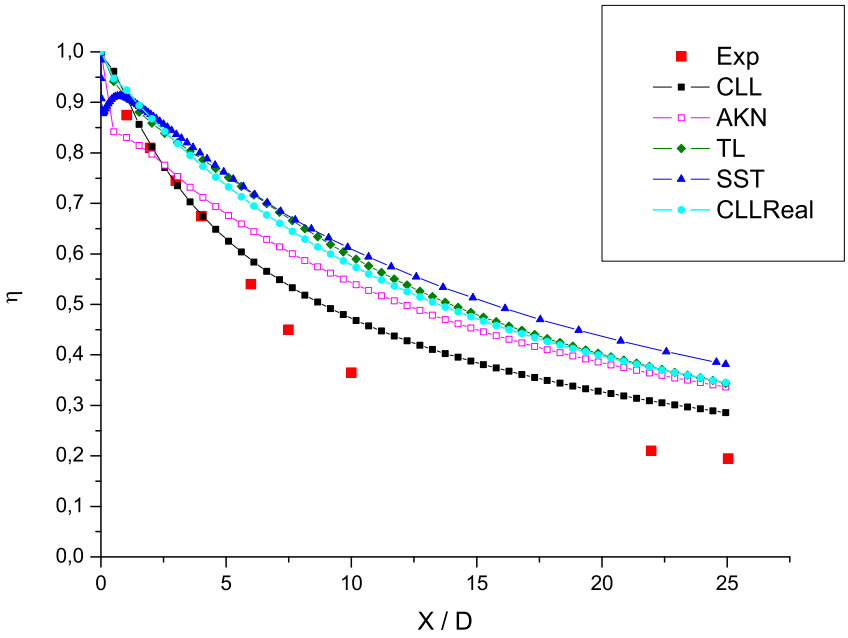
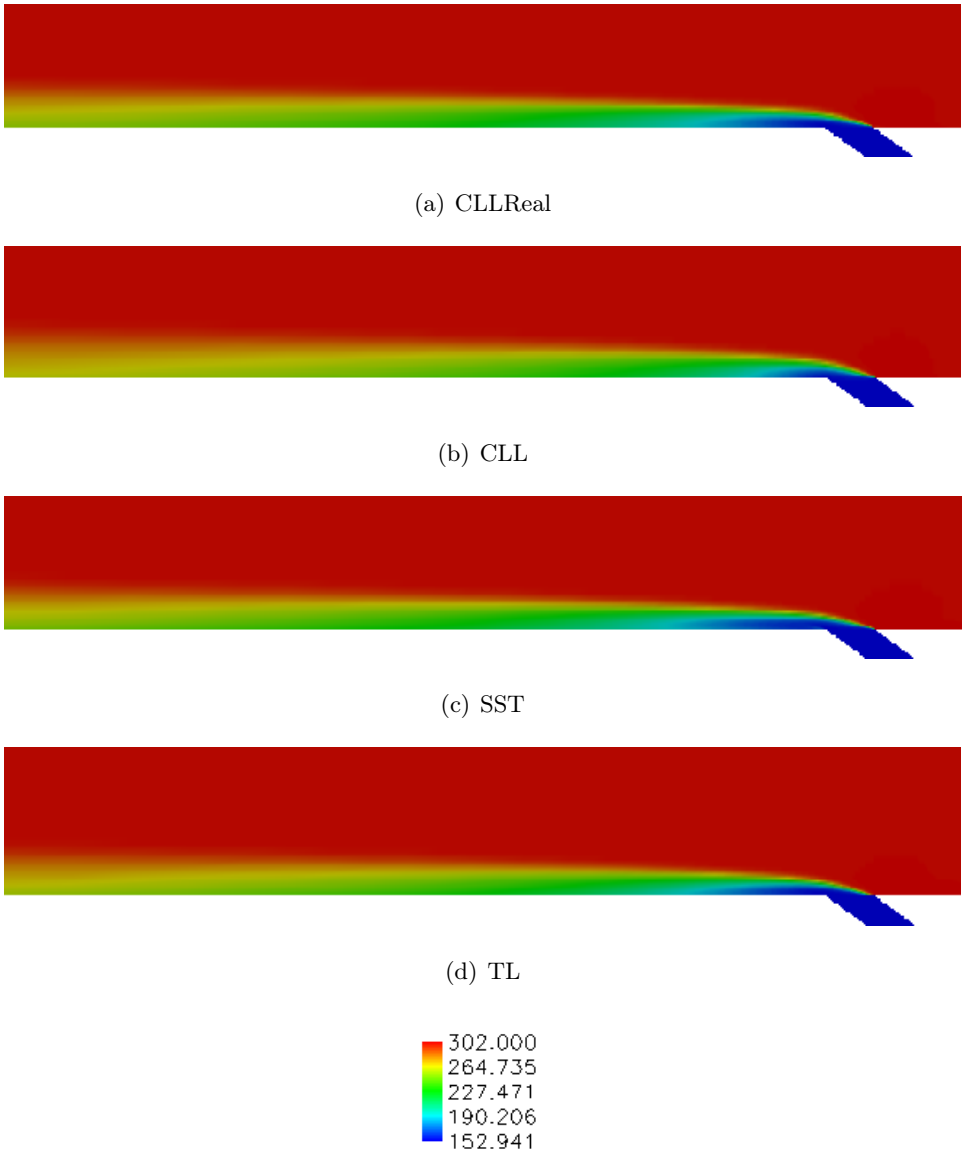


Figure 5.33: Sinha - Local center line film cooling effectiveness.

For the same reason already explained, numerical simulations results much more effective on the center line with the CLL model slightly more in agreement especially for $1 < X/D < 5$, but still in overprediction both at the hole and far downstream. What is interesting to notice is the local minimum presented by $k-\omega$ SST. As one may also see in Fig. 5.34, looking at the thicker and longer cold zone just next to the hole, it is the result of a deeper penetration of the jet. The more the jet is entraining, the more the hot gas goes round it and as a consequence wall temperature rises on the downstream zone really close to the hole.



(a) CLLReal

(b) CLL

(c) SST

(d) TL

Figure 5.34: Sinha - Temperature distribution on symmetry plane $[K]$.

It is important to repeat how these local parameters, even if very important in understanding the flow and validating the models, are of no particular interest for the practical design of cooling systems. It is in fact much more efficient in representing the real effectiveness of the system, some sort of mean parameter such as the laterally averaged effectiveness $< \eta >$ defined as:

$$< \eta > = \frac{1}{P/2} \int_0^{P/2} \eta \, dl \tag{5.18}$$

where P is the pitch between two consecutive rows, here plotted versus streamwise distance from the hole in Fig. 5.35.

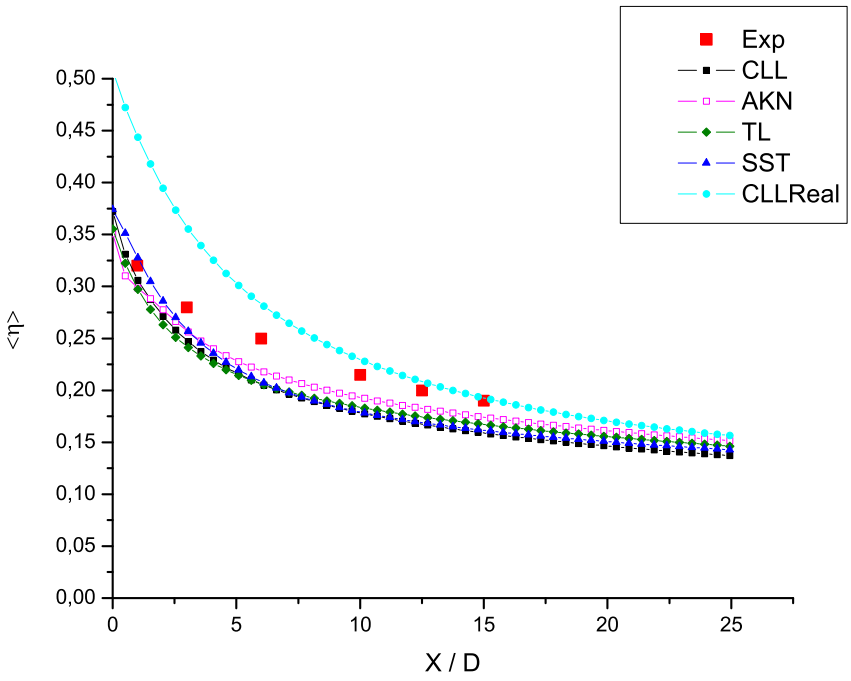


Figure 5.35: Sinha - Laterally averaged effectiveness.

5.4.2 Effusion case

Even if this case was set up to reproduce an effusion cooling system, whose main cooling effect is related with the heat-sink behavior of the coolant flowing into the channels, it must be underlined that numerical simulations were done in adiabatic conditions. This kind of simulations help designers in predicting flow development and temperature distributions also in cases where the wall heat flux is not null. The geometry for this case is a six staggered holes flat plate interposed in between a plenum and a channel at lower pressure. To enhance numerical stability, the plenum has been meshed as six different smaller plena each one with the same inlet mass flow imposed to respect total experimental cooling air mass flow. The computational grid shown in Fig. 5.37 is two million elements. A summary of experimental

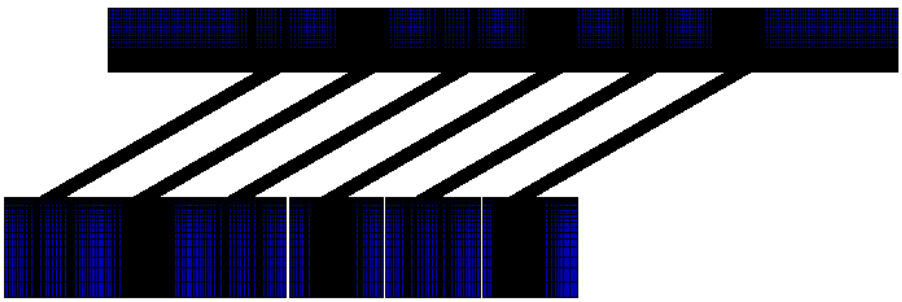


Figure 5.36: Effusion - Grid.

flow conditions can be found in Tab. 5.9. This set of physical conditions was translated into computational boundary conditions in the manner shown in Tab. 5.10.

Results are reported in terms of spanwise averaged adiabatic effectiveness, see Eq. 5.18, in Fig. 5.38. Together with experimental data, predictions using a correlative approach have been reported too. In particular L'Ecuyer and Soechting correlation with Sellers superposition criterion was used for

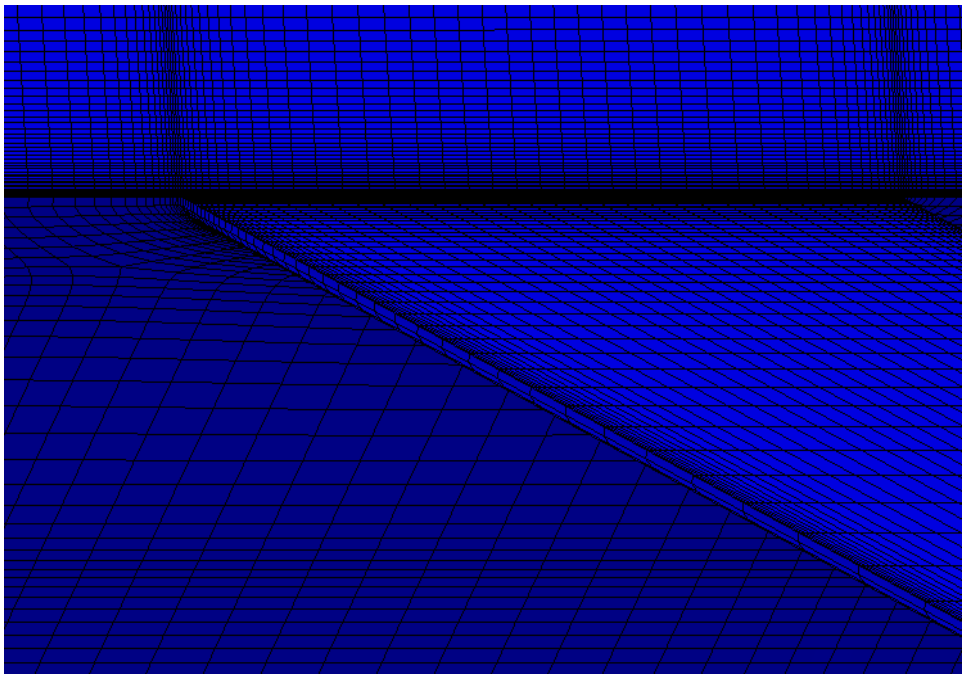


Figure 5.37: Effusion - Grid particular.

Table 5.9: Effusion - Flow conditions.

Cross flow temperature	323	K
Coolant temperature	298	K
Wall heat flux - $\dot{q}$	0	$W\,m^{-2}$
Pressure	$6.0 \cdot 10^4$	Pa
Density ratio - DR	1.103	
Blowing-rate - M	0.2	

the results found in [Arcangeli et al., 2006]. First thing to explain is the absence of strong peaks in experimental data. This is explained by the combination of two separate effects: the non perfectly adiabatic surface and the lack of resolution for the instruments. First statement comes as a consequence of a registered behavior of the plate upstream the first hole: values of effectiveness on that zone are not null meaning that conduction inside the solid is not exactly zero. Moreover the instrumental apparatus could not

Table 5.10: Effusion - Computational boundary conditions.

Freestream inlet temperature	323	K
Coolant inlet temperature	298	K
Wall temperature normal gradient	0	$K\,m^{-1}$
Outlet pressure	$6.0 \cdot 10^4$	Pa
Freestream inlet velocity	42.0	$m\,s^{-1}$
Coolant inlet velocity	0.1505	$m\,s^{-1}$
Freestream inlet turbulent kinetic energy	6.615	$m^2\,s^{-2}$
Coolant inlet turbulent kinetic energy	$3.398 \cdot 10^{-6}$	$m^2\,s^{-2}$
Freestream inlet turbulent dissipation	873.6	$m^2\,s^{-3}$
Coolant inlet turbulent dissipation	$2.573 \cdot 10^{-6}$	$m^2\,s^{-3}$

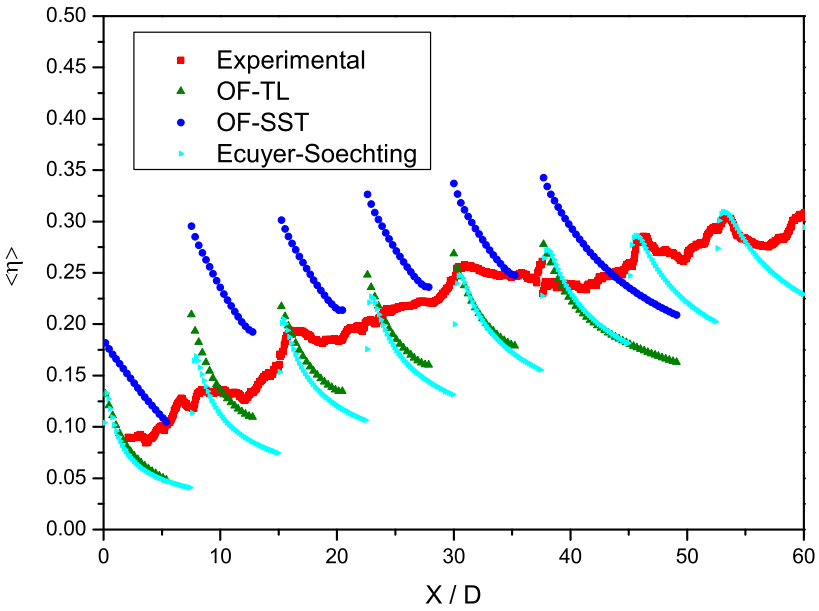
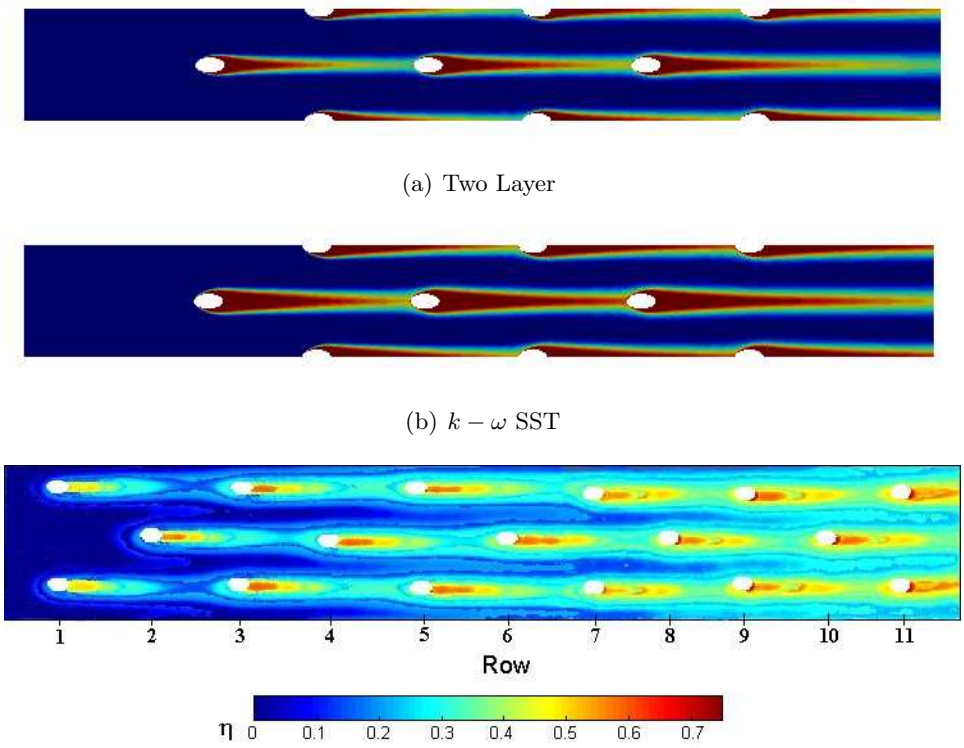


Figure 5.38: Effusion - Spanwise averaged effectiveness.

collect data in the zone very close to the hole. If areas are just a minimum percentage of overall surface one may think this error to be negligible but the area around the hole is the zone at highest value of effectiveness and so

its contribution to the weighted average is sensible.

Second thing to be noticed is that Two Layer model strongly improves matching of both experimental and correlative data in comparison to $k-\omega$ SST. Two Layer is still slightly in over prediction for the peak values especially for even peaks, for the odd ones in fact peak values result in being on the same level of the previous hole, meaning that rows interaction is very weak. This lack is due to the assumption of isotropic turbulence behavior that result, as seen in Sec. 5.4.1, in very coherent jets limiting lateral diffusion. Such effect is even stronger for the $k-\omega$ SST.



(a) Two Layer

(b) $k-\omega$ SST

(c) experimental

Figure 5.39: Effusion - Comparison between numerical and experimental effectiveness.

By looking at the two-dimensional effectiveness map in Fig. 5.39 the reason why SST turbulence model overpredicts previously plotted data comes

out. The jet predicted by SST model presents in fact a different shape: the area interested by the coolant effects is both larger and longer than in Two Layer. A thinner structure of the film, see also Fig. 5.40 and Fig. 5.41, maintains effectiveness values almost unitary even at distance from the hole equal to half pitch in streamwise direction.

Regarding correlative analysis: both models qualitatively well predict the decay of the spanwise effectiveness downstream the holes.

Moreover SST results in a larger film which is able to keep its cooling potential less affected by the interaction with the main flow in the shear layer.

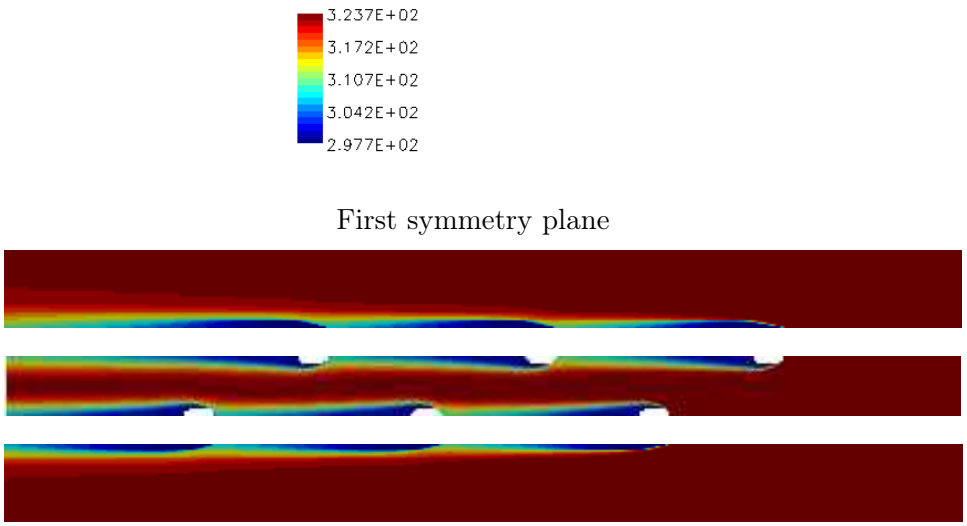


Figure 5.40: Effusion - Temperature distribution on symmetry planes SST $[K]$.

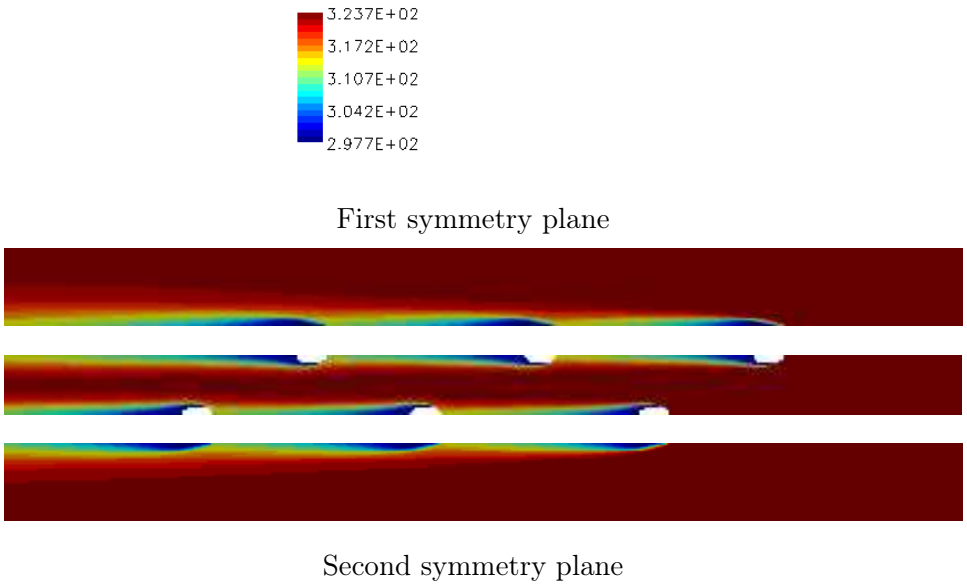


Figure 5.41: Effusion - Temperature distribution on symmetry planes TL $[K]$.

5.5 Blade internal cooling - ducts

Even though only hinted in previous sections, it should be clear how cooling of hot parts of turbomachinery is reached using simultaneously many different techniques, see also Fig. 5.42. The most efficient and effective cooling techniques, film and impingement cooling, have already been presented but there is another important cooling method to be tested.

Even if not very well suited for high temperature typical of leading edge of first stage turbine blades, internal cooling with ducts is still used in less extreme situations because of its low manufacturing cost and the ease of design.

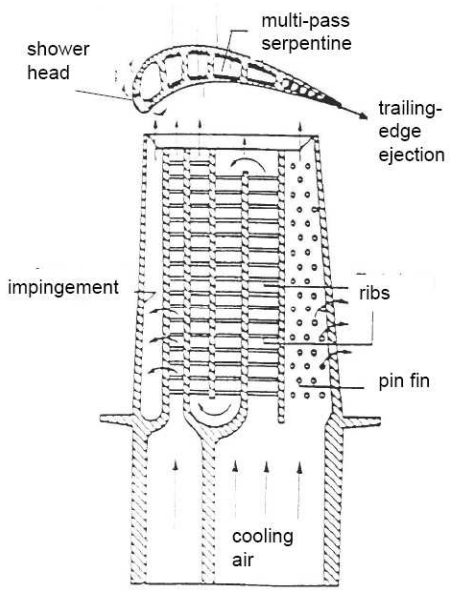


Figure 5.42: Typical compound cooling system of a blade.

Moreover in zones where geometrical constraints are particularly strict and interaction with external flow risky in term of aerodynamic performances, other techniques results very complicated to be inserted. That is why attention was posed on the axial cooling duct at the trailing edge of a turbine blade. The channel is ribbed and interaction with pedestals is also

simulated. Ribs are usually inserted to increase the overall efficiency and effectiveness of the cooling device, to reduce the air mass flow and more uniformly distribute the heat transfer coefficient on the wall. At the same time some drawbacks, such as higher manufacturing costs and higher pressure loss, should be considered.

5.5.1 AITEB2 case

This case refers to a typical cooling system of the trailing edge of a blade, chosen as a case for the European project AITEB2. Referring to Fig. 5.43, the fluid is flowing from right to left and the outlet is at the trailing edge of the blade. The light blue surfaces are the symmetry planes in the radial

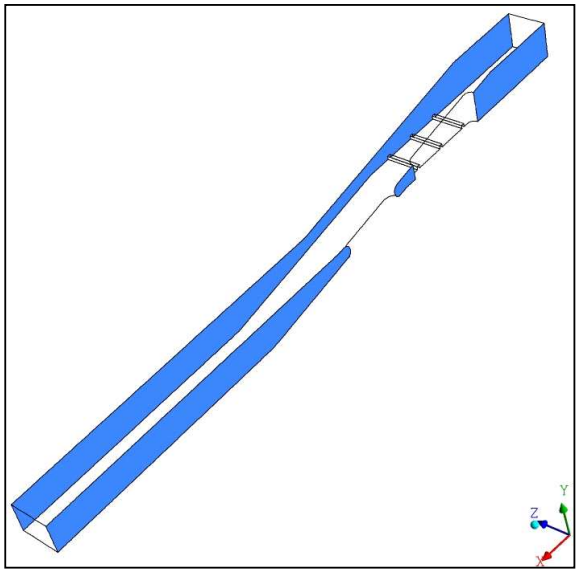


Figure 5.43: AITEB2 - Geometry of reference.

direction, meaning that their normal vector is parallel to the height of the blade itself. In correspondence with the first pedestal three ribs are inserted onto the suction side, to increase turbulent fluctuations and so improve heat transfer where it is most needed. As one may guess this kind of simulations is quite difficult to predict because the effects of the ribs is hard to model.

That is why the grid has been thickened in the area above the turbulator ribs, see Fig. 5.44. A particular of the grid around pedestal can be seen in Fig. 5.45. Total number of cells is approximately two millions. The characteristic length to make distances non-dimensional has been chosen to be the width of the pedestal: $d = 4\ mm$.

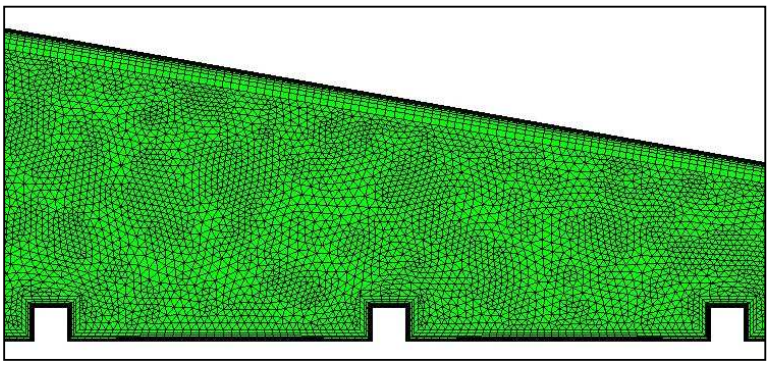


Figure 5.44: AITEB2 - Particular of the grid around the ribs.

Flow conditions are reported in Tab. 5.11.

Table 5.11: AITEB2 - Flow conditions.

Inlet temperature	326.24	K
Inlet mass flow rate	$2.31528 \cdot 10^{-3}$	$kg\,s^{-1}$
Intensity of turbulence	0.05	
Turbulence length scale	1	mm
Wall fixed temperature	313	K
Outlet pressure	$4.3 \cdot 10^{4}$	Pa

A new boundary conditions for inlet surfaces that defines inlet mass and not the velocity has been successfully used in this simulation. This boundary condition is quite important in case testing against experimental data, taken from imposed mass experiments, is done. Inlet velocities are usually unknown and their value is calculated via density and the inlet area. Before solving the flow it is impossible to know static pressure and so density

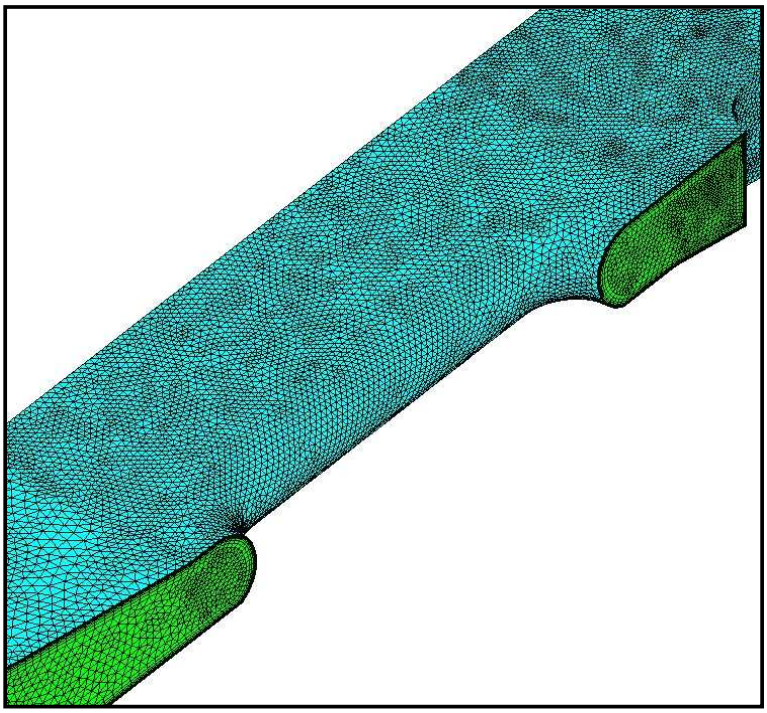


Figure 5.45: AITEB2 - Particular of the grid around the pedestal.

at the inlet. As a consequence mass could not be strictly respected if fixed velocity is imposed at the inlet. Moreover the deferred scheme described in Sec. 5.1 was applied. For Two Layer two different runs have been conducted to convergence: a first one with $\gamma = 0.5$ blends upwind and central difference, a second $\gamma = 1$ pure second order.

Results are reported in terms of local heat transfer coefficient on the centerline of the ribbed wall, see Fig. 5.46. Experimental data were not collected on the top surface of the ribs and so also corresponding numerical values have not been considered.

Looking at this graph, it appears clearly how again numerical results are much less diffusive than experimental data, with higher peaks and lower valleys in the zones where steeper gradients are present. In addition a more oscillatory behavior is manifested near the maximum with two or more local

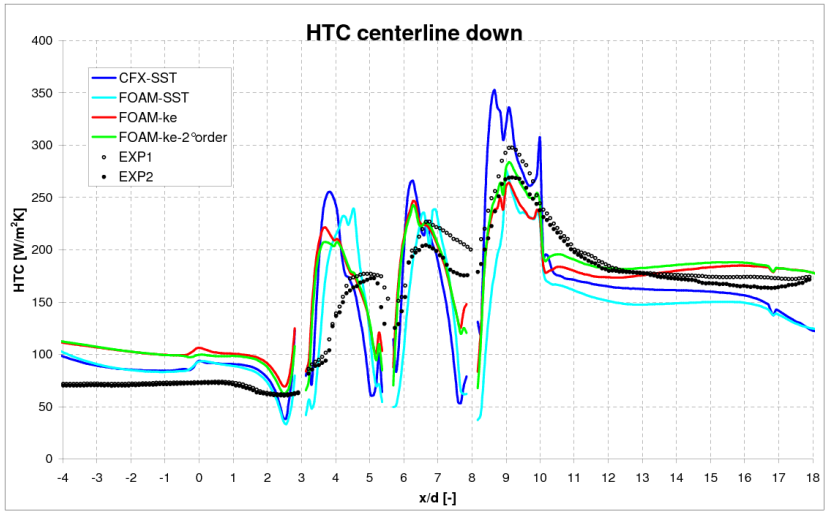


Figure 5.46: AITEB2 - Heat transfer coefficient on centerline of ribbed wall.

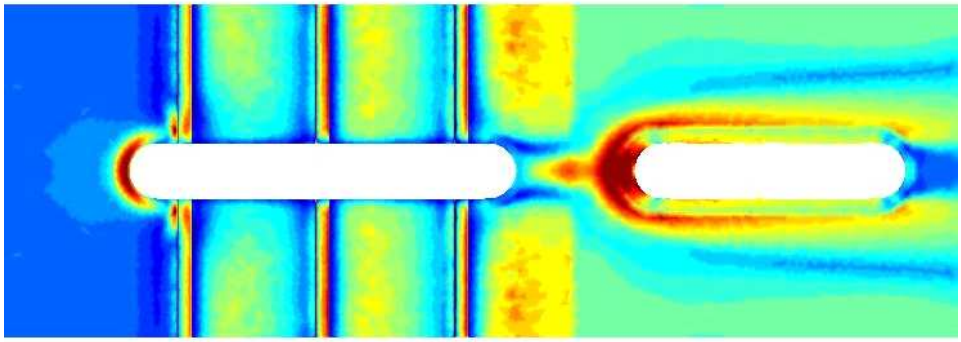
maxima after each rib, this tendency can be seen also near minima but with less intensity. What is really unpredicted by CFD results is the superposition effect registered by experiments. Especially for the SST model the three peaks remain of the same order of magnitude with a total increase from first to third of less than 20%, experiments register instead an increment of almost 100% on the value of heat transfer coefficient. It is interesting to point out how the Two Layer anticipate the maximum for the first two ribs resulting in being more efficient really close to the upstream rib. The higher levels of the heat transfer coefficient upstream the first rib have been checked not to depend on wrong inlet turbulent value, actually experimental inlet turbulence level was unknown but even a run with lower inlet values did not change the upstream behavior. What is even more satisfying is the improvement in matching experimental data in comparison with a simulation done with CFX [ANS, 2004]. Opposite to OpenFOAM $k-\omega$ SST model CFX

do have an increase from first to third rib but peak values result much higher especially after the first rib.

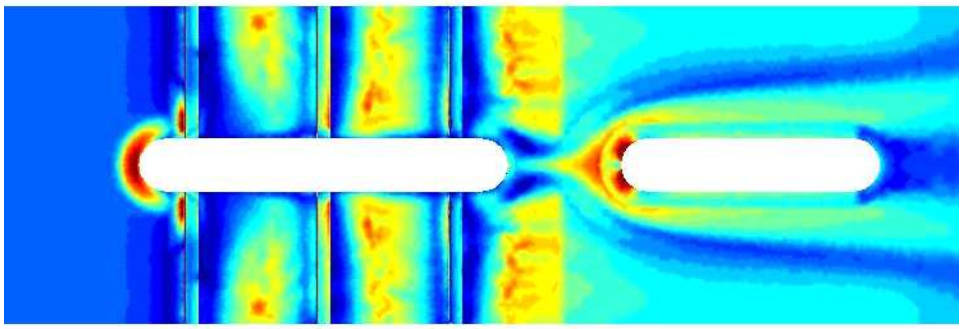
Again something that better show the overall cooling effect of the system was needed, that is why maps of heat transfer coefficient on the ribbed wall, starting from the point corresponding to the zero of previous graph, are plotted in Fig. 5.47.

These plots basically confirm what already noted about centerline values: superposition effect is almost null for the SST, numerical predictions tend to underestimate HTC close to the rib both upstream and downstream, less homogeneous values result also in more peaked trends.

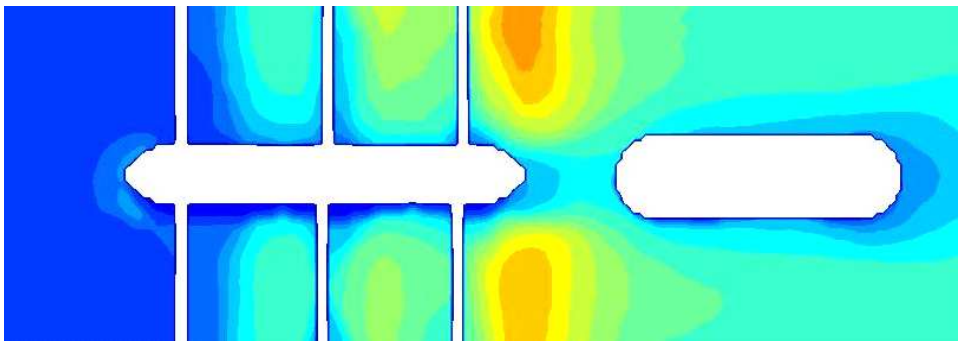
What can be added is the effect of nose of the pedestals where an horse shoe vortex is formed. As one may notice experimental results register a slight increase, especially for front pedestal, for numerical predictions as expected the "realizable" SST tends to limit the effect of the stagnation point, Two Layer at the contrary results in higher peaks right next to pedestal noses.



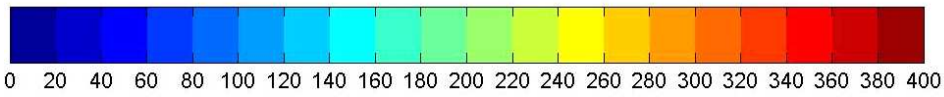
(a) Two Layer 2^{nd} order



(b) $k-\omega$ SST



(c) experimental



HTC color scale

Figure 5.47: AITEB2 - Heat transfer coefficient map $[W/(m^2K)]$.

Chapter 6

Conclusions and future developments

In this chapter presented work will be analyzed and discussed, with the aim of establishing whether these tools are adequate for CFD heat transfer research scopes or not.

First the implemented models will be criticized for their ability in reproducing expected results. Hence the history of development and the process adopted to reach the goal will be treated. Thus general strengths and foibles of OpenFOAM will be underlined, in order to understand which are the real potentialities and limits of this code.

Weaknesses encountered in the use of these tools will be exposed and finally a path for future expansions is drawn.

6.1 Conclusions

Numerical results have been compared mainly with experimental data obtaining in some cases good agreement but failing in some others, see chapter 5 for further details. Disagreements could be due to errors in implementing and solving the codes but also to a poor adherence of the model to the physics.

An a priori knowledge of the models behavior in the different situations is thus requested to decouple these effects. That is why models have always been tested against cases, representing the state of art for turbomachinery cooling, known in literature. Moreover the real aim of this investigation was to render OpenFOAM as reliable as commercial codes in the predictions for heat transfer simulations. So of primary importance was the adherence to experimental data in the cases where commercial solvers are in good agreement too. For the in-house cases in fact, a quite wide database of previous simulations, done with commercial solvers (ANS [2004], CD [2004]), was available. Results of commercial and OpenFOAM solvers are in substantial agreement with major disagreements concentrated in zones where trusting commercial solvers is questionable, meaning that more evident failures can be attributed to the lack of modeling more than incapacity in solving.

In research and industrial environment, it is fundamental to correctly predict the efforts necessary to obtain a goal, in order to establish which are the most convenient developments to invest on. From this point of view, OpenFOAM surely offers great advantages to programmers in comparison with other codes. With the acquired knowledge of language structure, programming is becoming faster and faster and major objectives can be hit.

In implementing such models the adopted strategy has been to mainly separate the activity of programming, namely developing new models, and running, that is solving cases of interest. First impingement geometry was run when all Low Reynolds $k-\varepsilon$ models were ready and tested on the simple case of the flat plate. Only very few changes at the codes have been induced by the results obtained on complex geometries, after the preliminary test was conducted.

After this long testing OpenFOAM is resulted very flexible for the implementation of new turbulence models, solver algorithms, boundary condition types and physical models. Furthermore it shows a quite good stability and

precision both for single processor and multi processor parallel calculations. Nowadays however OpenFOAM standard release is not ready to be treated as a ready to run code, its selection of models is too small and only rough estimates can be done. For complex cases such the ones presented in this thesis, specific model implementation is necessary. Theoretically there are no limitations on changes to be done in the code, all files are opened and built in a way to facilitate the necessary additions and modifications. The problem stands in the complexity of the code itself: even if rough, it has a wide variety of different tools, and in the programming language often quite difficult to interpret. Some of the needed expansions cannot be done at an exterior level, modification of external modules, but should be deeply inserted in the core of the code. Just to make an example of something really tried without success at present time is the implementation of arbitrary cyclic boundaries, namely non conformal cyclic boundaries. This and the other drawbacks are however related to the ignorance of the core coding, with time it will be possible to clarify and open all the still black boxes that compose the code.

6.2 Weaknesses

At current time there still are some unwanted and unresolved behavior of the presented models. First, and more notable, problem is connected with residual calculation.

Referring to Eq. 1.51 it can be seen how if some high values cells are immersed in a field of much lower values, the mean value and, as a consequence, the normalization factor result doped. This gives very low values for initial residuals even after few iterations, when the flow cannot be evolved from the initial guess, loosing the possibility of really checking the convergence for that specific equation. The way to avoid this unwanted behavior is taken from commercial solvers: redefine the normalization factor as the inlet flux

of the quantity of reference. With this definition, no matter the distribution, the production and dissipation terms, the normalization factor will always be constant and limited to acceptable values. Common in literature is to make reference to root mean square residuals so the will is to change not only normalization factor but also the way of summing.

Partially connected with this is the bounding strategy. It is quite common that, if starting guessed fields are not close enough to final solution, initial iterations, but not only unfortunately, gives negative solutions on some cells also for positively defined quantities (k, ε, ω, etc ...). This is not acceptable and maintaining such negative values will bring to unphysical solutions, so positive values are superimposed after the solution of the linear system on the cells with negative values. At convergence this problem should not manifest itself, the problem is in fact to properly correct such cells during the iteration process to enable convergence. The three strategies found in literature were to substitute negative values with a case dependent reference value or an almost null quantity, or to bound directly eddy viscosity possibly changing turbulent dissipation to maintain turbulent fields consistent. All options were tried without particular success, even if these inconsistency were limited to a very small number of cells, approximately 1/10000.

Residuals however do not only have the above described problem. It takes too much effort to have an always decreasing history of convergence: often when they are reduced of three-four orders from the initial value, the decrease ceases and it is very hard to reach higher levels of convergence. This is probably due to the used convective schemes: advances are expected using other blended schemes like Gamma or hopefully better deferred approach.

Another inconsistency, usually small but somehow influencing a bit the solution, comes out in cases where there are High Reynolds walls, namely walls on which first node is in the order of tens. This situation arises when there are walls on which no particular interest is posed on in terms of near

wall behavior and thickening the grid will be too expensive. On such meshes Low Reynolds models do not reliably predict first node values but at the same time the use of wall functions is at least questionable on a Low Reynolds model.

6.3 Future Developments

It is important, starting from the weakness above underlined, to define a path to follow for future developments, establishing priorities as function of the needs and ease of development. Some of the ideas to be proposed are already in phase of testing, some are in course of stabilization, some others are at current time not started yet.

Wall function for temperature should be implemented together with a revise of given turbulent wall functions that are based on a too simplified version. Temperature wall function is in phase of testing while the work on turbulent wall function is to be started.

The big failure of isotropic model for film cooling cases, is tried to be solved with the implementation of the anisotropic correction, see Sec. 5.4.1, upon the Two Layer model. At this point only the structure of the model is ready to run, with selection of the area of application and the proper doping of the Reynolds stress to be done.

Another great target not to miss, is the development of a conjugate solver, enabling to solve also for conduction inside solid walls. This task is of high interest for the study of film cooling but is actually fundamental for effusion cases. Such expansion is already running on single CPU, further testing is however needed to validate the results, but it must be made adaptable to parallel calculations.

As already hinted, other advection schemes should be tried in order to solve the low convergence problem. Experience suggests to implement a deferred correction in order to enhance stability. One of the cases presented

here, see Sec. 5.5.1, was already using this development. Further testing is needed for a confirmation of the good behavior in helping convergence.

The last expansion to be mentioned will give even more freedom in the meshing process. The idea is to remove the constraint on cyclic boundaries to be conformal or better to loosen the one to one relationship between cells on the two coupled surfaces.

Appendix A

wallHTC.C

```
001    #include "fvCFD.H"
002    #include "basicThermo.H"
003    #include "compressible/turbulenceModel/turbulenceModel.H"
004    #include "fixedGradientFvPatchFields.H"
005    #include "wallFvPatch.H"
006
007    // * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
008
009    int main(int argc, char *argv[]) {
010
011
012    #   include "addTimeOptions.H"
013    #   include "setRootCase.H"
014
015    #   include "createTime.H"
016
017        // Get times list
018        instantList Times = runTime.times();
019
020        // set startTime and endTime on -time and -latestTime options
021    #   include "checkTimeOptions.H"
022
023        runTime.setTime(Times[startTime], startTime);
024
025    #   include "createMesh.H"
026
027
028        for (label i=startTime; i<endTime; i++)
029        {
030            runTime.setTime(Times[i], i);
031
032            Info<< "Time = " << runTime.timeName() << endl;
033
034            mesh.readUpdate();
035
036            #   include "createFields.H"
037
038            surfaceScalarField heatFlux =
```

```
039
040                                 fvc::interpolate(alpha) * fvc::snGrad(h);
041
042
043
044             forAll(mesh.boundary(), patchi)
045             {
046                 if (typeid(mesh.boundary()[patchi]) == typeid(wallFvPatch))
047                 {
048                 wallHeatFlux.boundaryField()[patchi] =
049                 heatFlux.boundaryField()[patchi];
050
051
052
053                 HTC.boundaryField()[patchi] =
054                 wallHeatFlux.boundaryField()[patchi]
055                 / (T.boundaryField()[patchi]
056                 - Tadiabatic.boundaryField()[patchi]+SMALL);
057
058                 Nu.boundaryField()[patchi] =
059                 HTC.boundaryField()[patchi]
060                 / kond.boundaryField()[patchi] * L.value();
061
062                 Info<<  "\nPatch " << patchi
063                     << " named " << mesh.boundary()[patchi].name()
064
065                     << "\n Wall heat fluxes [W] " <<
066                     sum(mesh.magSf().boundaryField()[patchi]
067                   * heatFlux.boundaryField()[patchi])
068
069                     << "\n Specific Wall heat flux [W/m2]:  min: " <<
070                     min(heatFlux.boundaryField()[patchi])
071                     << " max: " <<
072                     max(heatFlux.boundaryField()[patchi])
073                     << " average: " <<
074                     average(heatFlux.boundaryField()[patchi])
075
076                     << "\n HTC [W/(m2*K)]: min: "<<
077                     min(HTC.boundaryField()[patchi])
078                     << " max: " <<
079                     max(HTC.boundaryField()[patchi])
080                     << " average: " <<
081                     average(HTC.boundaryField()[patchi])
082
083                     << nl << endl;
084
085                 }
086             }
087
088             wallHeatFlux.write();
089
090             HTC.write();
091
092             Nu.write();
```

```
093
094         }
095
096         return(0);
097     }
```

Appendix B

createFields.H

```
001  autoPtr<basicThermo> thermo
002  (
003      basicThermo::New(mesh)
004  );
005
006  Info << "Reading field h\n" << endl;
007  const volScalarField& h = thermo->h();
008
009  Info << "Reading field T\n" << endl;
010  const volScalarField& T = thermo->T();
011
012  Info << "Reading field Tadiabatic\n" << endl;
013
014  volScalarField Tadiabatic
015  (
016      IOobject
017      (
018          "Tadiabatic",
019          runTime.timeName(),
020          mesh,
021          IOobject::MUST_READ,
022          IOobject::AUTO_WRITE
023      ),
024      mesh
025  );
026
027  volScalarField rho
028  (
029      IOobject
030      (
031          "rho",
032          runTime.timeName(),
033          mesh
034      ),
035      thermo->rho()
036  );
037
038  const volScalarField alpha = thermo->alpha();
039
```

```
040  const volScalarField Cp = thermo->Cp();
041
042  const volScalarField kond = Cp * alpha;
043
044  dimensionedScalar  L("L", dimensionSet(0,1,0,0,0), 0.1016);
045
046  volVectorField U
047  (
048      IOobject
049      (
050          "U",
051          runTime.timeName(),
052          mesh,
053          IOobject::MUST_READ,
054          IOobject::AUTO_WRITE
055      ),
056      mesh
057  );
058
059  #include "compressibleCreatePhi.H"
060
061  autoPtr<compressible::turbulenceModel> turbulence
062  (
063      compressible::turbulenceModel::New
064      (
065          rho,
066          U,
067          phi,
068          thermo()
069      )
070  );
071
072
073  volScalarField wallHeatFlux
074  (
075      IOobject
076      (
077          "wallHeatFlux",
078          runTime.timeName(),
079          mesh,
080          IOobject::NO_READ,
081          IOobject::AUTO_WRITE
082      ),
083      mesh,
084      dimensionedScalar("wallHeatFlux", dimensionSet(1,0,-3,0,0), 0.0)
085  );
086
087  volScalarField HTC
088  (
089      IOobject
090      (
091          "HTC",
092          runTime.timeName(),
093          mesh,
```

```
094         IOobject::NO_READ,
095         IOobject::AUTO_WRITE
096     ),
097     mesh,
098     dimensionedScalar("HTC", dimensionSet(1,0,-3,-1,0), 0.0)
099 );
100
101 volScalarField Nu
102 (
103     IOobject
104     (
105         "Nu",
106         runTime.timeName(),
107         mesh,
108         IOobject::NO_READ,
109         IOobject::AUTO_WRITE
110     ),
111     mesh,
112     dimensionedScalar("Nu", dimensionSet(0,0,0,0,0), 0.0)
113 );
```

Appendix C

kEpsilon.H

```
001
002   =========                 |
003   \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
004    \\    /   O peration     |
005     \\  /    A nd           | Copyright (C) 1991-2005 OpenCFD Ltd.
006      \\/     M anipulation  |
007
008   License
009       This file is part of OpenFOAM.
010
011       OpenFOAM is free software; you can redistribute it and/or modify it
012       under the terms of the GNU General Public License as published by the
013       Free Software Foundation; either version 2 of the License, or (at your
014       option) any later version.
015
016       OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
017       ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
018       FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
019       for more details.
020
021       You should have received a copy of the GNU General Public License
022       along with OpenFOAM; if not, write to the Free Software Foundation,
023       Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
024
025   Class
026       kEpsilon
027
028   Description
029       Standard k-epsilon turbulence model.
030
031   SourceFiles
032       kEpsilon.C
033       kEpsilonCorrect.C
034
035
036
037   #ifndef compressiblekEpsilon_H #define compressiblekEpsilon_H
038
039   #include "turbulenceModel.H"
```

```
040
041
042
043    namespace Foam { namespace compressible { namespace turbulenceModels
044    {
045
046    /*---------------------------------------------*\
047                Class kEpsilon Declaration
048    \*---------------------------------------------*/
049
050    class kEpsilon :
051        public turbulenceModel
052    {
053        // Private data
054
055            dimensionedScalar Cmu;
056            dimensionedScalar C1;
057            dimensionedScalar C2;
058            dimensionedScalar C3;
059            dimensionedScalar alphak;
060            dimensionedScalar alphaEps;
061            dimensionedScalar alphah;
062
063            volScalarField k_;
064            volScalarField epsilon_;
065            volScalarField mut_;
066
067
068    public:
069
070        //- Runtime type information
071        TypeName("kEpsilon");
072
073        // Constructors
074
075            //- from components
076            kEpsilon
077            (
078                const volScalarField& rho,
079                const volVectorField& U,
080                const surfaceScalarField& phi,
081                basicThermo& thermophysicalModel
082            );
083
084
085        // Destructor
086
087            ~kEpsilon()
088            {}
089
090
091        // Member Functions
092
093            tmp<volScalarField> mut() const
```

```
094             {
095                 return mut_;
096             }
097
098             //- Return the effective diffusivity for k
099             tmp<volScalarField> DkEff() const
100             {
101                 return tmp<volScalarField>
102                 (
103                     new volScalarField("DkEff", alphak*mut_ + mu())
104                 );
105             }
106
107             //- Return the effective diffusivity for epsilon
108             tmp<volScalarField> DepsilonEff() const
109             {
110                 return tmp<volScalarField>
111                 (
112                     new volScalarField("DepsilonEff", alphaEps*mut_ + mu())
113                 );
114             }
115
116             //- Return the effective turbulent thermal diffusivity
117             tmp<volScalarField> alphaEff() const
118             {
119                 return tmp<volScalarField>
120                 (
121                     new volScalarField("alphaEff", alphah*mut_ + alpha())
122                 );
123             }
124
125             tmp<volScalarField> k() const
126             {
127                 return k_;
128             }
129
130             tmp<volScalarField> epsilon() const
131             {
132                 return epsilon_;
133             }
134
135             tmp<volScalarField> omega() const
136             {
137                 return epsilon_/(k_+k0_);
138             }
139
140             tmp<volTensorField> R() const;
141
142             tmp<fvVectorMatrix> divRhoR(volVectorField& U) const;
143
144             void correct();
145
146             //- Read turbulenceProperties dictionary
147             bool read();
```

```
148     };
149
150
151
152
153     } // End namespace turbulenceModels } // End namespace compressible
154     } // End namespace Foam
155
156
157
158     #endif
159
160
```

Appendix D

kEpsilon.C

```
001    /*---------------------------------------------------------------------------*\
002    =========                 |
003    \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
004     \\    /   O peration     |
005      \\  /    A nd           | Copyright (C) 1991-2005 OpenCFD Ltd.
006       \\/     M anipulation  |
007    -------------------------------------------------------------------------
008    License
009        This file is part of OpenFOAM.
010
011        OpenFOAM is free software; you can redistribute it and/or modify it
012        under the terms of the GNU General Public License as published by the
013        Free Software Foundation; either version 2 of the License, or (at your
014        option) any later version.
015
016        OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
017        ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
018        FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
019        for more details.
020
021        You should have received a copy of the GNU General Public License
022        along with OpenFOAM; if not, write to the Free Software Foundation,
023        Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
024
025    \*---------------------------------------------------------------------------*/
026
027    #include "kEpsilon.H" #include "addToRunTimeSelectionTable.H"
028    #include "wallFvPatch.H"
029
030    // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
031
032    namespace Foam { namespace compressible { namespace turbulenceModels
033    {
034
035    // * * * * * * * * * * * * * * * Static Data Members * * * * * * * * * * * //
036
037    defineTypeNameAndDebug(kEpsilon, 0);
038    addToRunTimeSelectionTable(turbulenceModel, kEpsilon, dictionary);
039
```

```
040    // * * * * * * * * * * * * * * * * * Constructors  * * * * * * * * //
041
042    // from components
043    kEpsilon::kEpsilon (
044        const volScalarField& rho,
045        const volVectorField& U,
046        const surfaceScalarField& phi,
047        basicThermo& thermophysicalModel
048    ) :
049        turbulenceModel(typeName, rho, U, phi, thermophysicalModel),
050
051        Cmu(turbulenceModelCoeffs_.lookup("Cmu")),
052        C1(turbulenceModelCoeffs_.lookup("C1")),
053        C2(turbulenceModelCoeffs_.lookup("C2")),
054        C3(turbulenceModelCoeffs_.lookup("C3")),
055        alphak(turbulenceModelCoeffs_.lookup("alphak")),
056        alphaEps(turbulenceModelCoeffs_.lookup("alphaEps")),
057        alphah(turbulenceModelCoeffs_.lookup("alphah")),
058
059        k_
060        (
061            IOobject
062            (
063                "k",
064                runTime_.timeName(),
065                mesh_,
066                IOobject::MUST_READ,
067                IOobject::AUTO_WRITE
068            ),
069            mesh_
070        ),
071
072        epsilon_
073        (
074            IOobject
075            (
076                "epsilon",
077                runTime_.timeName(),
078                mesh_,
079                IOobject::MUST_READ,
080                IOobject::AUTO_WRITE
081            ),
082            mesh_
083        ),
084
085        mut_
086        (
087            IOobject
088            (
089                "mut",
090                runTime_.timeName(),
091                mesh_,
092                IOobject::NO_READ,
093                IOobject::AUTO_WRITE
```

```
094             ),
095             Cmu*rho_*sqr(k_)/(epsilon_ + epsilonSmall_)
096         )
097     { #   include "wallViscosityI.H" }
098
099
100     // * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * //
101
102     tmp<volTensorField> kEpsilon::R() const {
103         return tmp<volTensorField>
104         (
105             new volTensorField
106             (
107                 IOobject
108                 (
109                     "R",
110                     runTime_.timeName(),
111                     mesh_,
112                     IOobject::NO_READ,
113                     IOobject::NO_WRITE
114                 ),
115                 ((2.0/3.0)*I)*k_ - (mut_/rho_)*2*dev(symm(fvc::grad(U_))),
116                 k_.boundaryField().types()
117             )
118         );
119     }
120
121
122     tmp<fvVectorMatrix> kEpsilon::divRhoR(volVectorField& U) const {
123         return
124         (
125         - fvm::laplacian(muEff(), U) - fvc::div(muEff()*dev2(fvc::grad(U)().T()))
126         );
127     }
128
129
130     bool kEpsilon::read() {
131         if (turbulenceModel::read())
132         {
133             turbulenceModelCoeffs_.lookup("Cmu") >> Cmu;
134             turbulenceModelCoeffs_.lookup("C1") >> C1;
135             turbulenceModelCoeffs_.lookup("C2") >> C2;
136             turbulenceModelCoeffs_.lookup("C3") >> C3;
137             turbulenceModelCoeffs_.lookup("alphak") >> alphak;
138             turbulenceModelCoeffs_.lookup("alphaEps") >> alphaEps;
139             turbulenceModelCoeffs_.lookup("alphah") >> alphah;
140
141             return true;
142         }
143         else
144         {
145             return false;
146         }
147     }
```

```
148
149
150     void kEpsilon::correct() {
151         if (!turbulence_)
152         {
153             // Re-calculate viscosity
154             mut_ = rho_*Cmu*sqr(k_)/(epsilon_ + epsilonSmall_);
155             mut_.internalField() = min( mut_.internalField(),1.0);
156             mut_.correctBoundaryConditions();
157             mut_.boundaryField() = min(mut_.boundaryField(), 1.0);
158             return;
159         }
160
161         turbulenceModel::correct();
162
163         volScalarField divU = fvc::div(phi_/fvc::interpolate(rho_));
164
165         if (mesh_.moving())
166         {
167             divU += fvc::div(mesh_.phi());
168         }
169
170         tmp<volTensorField> tgradU = fvc::grad(U_);
171         volScalarField G = 2*mut_*(tgradU() && dev(symm(tgradU())));
172         tgradU.clear();
173
174     #   include "wallFunctionsI.H"
175
176         // Dissipation equation
177         tmp<fvScalarMatrix> epsEqn
178         (
179             fvm::ddt(rho_, epsilon_)
180         + fvm::div(phi_, epsilon_)
181         - fvm::laplacian(DepsilonEff(), epsilon_)
182         ==
183             C1*G*epsilon_/k_
184         + fvm::SuSp((C3 - (2.0/3.0)*C1)*rho_*divU, epsilon_)
185         - fvm::Sp(C2*rho_*epsilon_/k_, epsilon_)
186         );
187
188         epsEqn().relax();
189
190     #   include "wallDissipationI.H"
191
192         solve(epsEqn);
193         bound(epsilon_, epsilon0_);
194
195
196         // Turbulent kinetic energy equation
197
198         tmp<fvScalarMatrix> kEqn
199         (
200             fvm::ddt(rho_, k_)
201         + fvm::div(phi_, k_)
```

```
202         - fvm::laplacian(DkEff(), k_)
203         ==
204             G - fvm::SuSp(2.0/3.0*rho_*divU, k_)
205         - fvm::Sp(rho_*epsilon_/k_, k_)
206         );
207 
208         kEqn().relax();
209         solve(kEqn);
210         bound(k_, k0_);
211 
212 
213         // Re-calculate viscosity
214         mut_ = rho_*Cmu*sqr(k_)/epsilon_;
215         mut_.internalField() = min( mut_.internalField(),1.0);
216         mut_.correctBoundaryConditions();
217         mut_.boundaryField() = min(mut_.boundaryField(), 1.0);
218     #   include "wallViscosityI.H"
219 
220     }
221 
222 
223     // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *//
224 
225     } // End namespace turbulenceModels } // End namespace compressible
226     } // End namespace Foam
227 
228     // ************************************************************************//
```

Bibliography

K. Abe, T. Kondoh, and Y. Nagano. A new turbulence models for predicting fluid flow and heat transfer in separating and reattaching flows-i. flow field calculations. *International Journal of Heat Mass Transfer*, 37:139–151, 1994.

J. D. Anderson, jr. *Computational Fluid Dynamics, the basic with applications*. McGraw-Hill, US, 1995.

A. Andreini, C. Carcasci, S. Gori, and M. Surace. Film cooling system numerical design: adiabatic and conjugate analysis. *ASME paper*, (HT2005-72042), 2005.

CFX Manual - version 10. ANSYS, 2004.

D. Apsley. Computational hydrolics. Technical report, University of Manchester, 2006.

L. Arcangeli, M. Surace, L. Tarchi, D. Coutandin, and S. Zecchi. Correlative analysis of effusion cooling systems. *ASME Turbo Expo*, (GT2006-90405), 2006.

A. Azzi and B. A. Jubran. Numerical modeling of film cooling from short length steam-wise injection holes. *Heat and mass transfer*, (39):345–353, 2003.

A. Azzi and D. Lakehal. Perspectives in modeling film cooling of tur-

bine blades by transcending conventional two-equation turbulence models. *Journal of turbomachinery*, (124):472–484, 2002.

J. Bredberg. On two equation eddy-viscosity models. Technical Report Internal report 01/8, Chalmers University of Technology, 2001.

StarCD Methodology - version 3.24. CD adapco Group, 2004.

W. L. Chen, F. S. Lien, and M. A. Leschziner. Low-reynolds-number eddy-viscosity modelling based on non-linear stress-strain/vorticity relations. *Engineering turbulence modelling and experiments*, 3:91–100, 1996.

K. Y. Chien. Predictions of channel and boundary-layer flows with a low reynolds number turbulence model. *AIAA Journal*, 20(1):33–38, 1982.

D. E. Coles and E. A. Hirst. Computation of turbulent boundary layers. In *AFOSRIFPStanford Conference*, volume 2. Stanford University, 1969.

D. Cooper, D. C. Jackson, B. E. Launder, and G. X. Liao. Impinging jet studies for turbulence model assessment. part i: Flow-field experiments. *Int. J. Heat Mass Transfer*, (36):2675–2684, 1993.

L. Davidson. An introduction to turbulence models. Technical Report Publication 97/2, Chalmers University of Technology, 2003.

L. Davidson. Numerical methods for turbulent flows. Technical Report MTF071, Chalmers University of Technology, 2005.

L. Davidson. Turbulence modeling. Technical Report MTF270, Chalmers University of Technology, 2006.

S. R. Davis. *C++ for Dummies*. Wiley Publishing, Inc, 2005.

E. Di Carmine. Studio numerico del raffreddamento per impingement nelle turbomacchine: Simulazione di geometrie reali. Master's thesis, Università degli Studi di Firenze, Facoltà di Ingegneria, 2004.

J. P. V. Doormal and G. D. Raithby. Enhancements of the simple method for predicting incompressible fluid flows. *Numerical Heat Transfer*, 7:147–163, 1984.

P. A. Durbin. On the $k-\varepsilon$ stagnation point anomaly. *International journal of heat and fluid flow*, 17(1):89–90, 1996.

P. A. Durbin and S. P. M. Benhia. Prediction of heat transfer in an axisymmetric turbulent jet impinging on a flat plate. *International journal of heat mass tranfer*, 41(12):1845–1855, 1998.

B. Facchini and M. Surace. Impingement cooling for modern combustors: experimental analysis of heat transfer and effectiveness. *Experiments in Fluids*, (40):601–611, 2006.

J. H. Ferziger and M. Peric. *Computational Methods for Fluid Dynamics*. Springer, Germany, 2002.

K. M. B. Gustafsson. *Experimental studies of effusion cooling*. PhD thesis, Chalmers University of Technology, Department of Thermo and Fluid Dynamics, 2001.

C. B. Hwang and C. A. Lin. Improved low-reynolds-number $k-\varepsilon$ model based on direct numerical simulation data. *AIAA Journal*, 36(1):38–43, 1998.

H. Iacovides. Current practice and recent developments in wall functions. Technical report, Department of mechanical Aerospace and Manufactoring Engineering UMIST, 2004.

R. I. Issa. Solution of the implicitly discretized fluid flow equations by operator splitting. *Journal of Computational Physics*, 62:40–65, 1986.

H. Jasak. *Error Analysis and Estimation for the Finite Volume Method*

with Applications to Fluid Flows. PhD thesis, Imperial College of Science, Technology and Medicine, 1996.

H. Jasak, H. Weller, and N. Nordin. In cylinder cfd simulation using a c++ object-oriented toolkit. *SAE Technical Papers*, (2004-01-0110), 2004.

W. P. Jones and B. E. Launder. The prediction of laminarization with a two-equation model of turbulence. *Int. J. Heat and Mass Transfer*, (15): 301–314, 1972.

F. Juretic. *Error Analysis in Finite Volume*. PhD thesis, Imperial College of Science, Technology and Medicine, 2004.

K. C. Karki. *A Calculation Procedure for Viscous Flows at all Speed in complex Geometries*. PhD thesis, University of Minnesota, Faculty of the Graduated School, 1986.

K. C. Karki and S. V. Patankar. Pressure based calculation procedure for viscous flows at all speeds in arbitrary configurations. *AIAA Journal*, 27 (9):1167–1174, 1989.

D. D. Knight. Numerical simulation of compressible turbulent flows using the reynolds-averaged navier-stokes equations. *AGARD*, R(819):5.1–5.22, 1997. AGARD FDP Special Course on "Turbulence in Compressible Flows" at VKI.

C. Lacor. Solution of time dependent reynolds averaged navier-stokes equations with the finite volume method. Technical report, Vrije Universitiet Brussel, 2006.

D. Lakehal. Near-wall modeling of turbulent convective heat transport in film cooling of turbine blades with the aid of direct numerical simulation data. *Journal of turbomachinery*, (124):485–498, 2002.

C. K. G. Lam and K. Bremhorst. A modified form of the $k - \varepsilon$ model for predicting wall turbulence. *Journal of fluids engineering*, 103:456–460, 1981.

F. S. Lien. *Computational modeling of 3-D flow in complex ducts and passages*. PhD thesis, University of Manchester, Institute of science and technology, 1992.

F. S. Lien and M. A. Leschziner. A pressure-velocity solution strategy for compressible flow and its application to shock/boundary-layer interaction using second-moment turbulence closure. *Journal of fluids engineering*, 115:717–725, 1993.

W. Malalasekera and H. K. Versteeg. *Computational Fluid Dynamics*. Longman Scientific, England, 1995.

L. Mangani. Openfoam: Basic theory and preliminary result. Technical report, University of Florence, Energy Engineering Department - Ansaldo Energia, 2006.

J. J. McGuirk and G. Page. Shock capturing using a pressure-correction method. *AIAA Journal*, 28(10):1751–1757, 1990.

G. Medic and P. A. Durbin. Towards improved prediction of heat transfer on turbine blades. *Journal of turbomachinery*, 124:187–192, 2002.

F. R. Menter. Zonal two equation $k - \omega$ turbulence models for aerodynamic flows. *AIAA Paper*, (93-2906), 1993.

F. R. Menter. Two equation eddy viscosity turbulence model for engineering applications. *AIAA Journal*, 32:1598–1604, 1994.

F. Moukalled and M. Darwish. A high-resolution pressure-based algorithm for fluid flow at all speeds. Technical Report SPC-99-4003, European Office of Aerospace Research and Development, 1999.

J. Y. Murthy and S. R. Mathur. Numerical methods in heat, mass and momentum tranfer. Technical Report ME 608, School of Mechanical Engineering Purdue University, 2002.

OpenFOAM Programmers Guide. OpenCFD Limited, a.

OpenFOAM User Guide. OpenCFD Limited, b.

http://www.opencfd.co.uk. OpenCFD Limited, c.

S. V. Patankar. *Numerical Heat Transfer and Fluid Flow*. Taylor & Francis, US, 1980.

S. V. Patankar and D. B. Spalding. A calculation procedure for heat, mass and momentum tranfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Tranfer*, (15):1787–1805, 1972.

V. C. Patel, W. Rodi, and G. Sheuerer. Turbulence models for near wall and low reynolds number flows: a review. *AIAA Journal*, 23(9):1308–1319, 1985.

M. Peric, Z. Lilek, and L. Demirdzic. A collocated finite volume method for predicting flows at all speed. *Journal of Numerical Methods in Fluids*, 16: 1029–1050, 1993.

E. S. Politis and K. C. Giannakoglou. A pressure-based algorithm for high speed turbomachinery flows. *International journal for numerical methods in fluids*, 25:63–80, 1996.

C. M. Rhie. Pressure based navier-stokes solver using the multigrid method. *AIAA Journal*, 27(8):1017–1018, 1989.

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

W. Rodi. Experience with two-layer models combining the $k-\varepsilon$ model with a one-equation model near the wall. Technical Report 91-0216, AIAA, 1991.

W. Shyy and M. E. Braaten. Applications of a generalized pressure correction algorith for flows in complicated geometries. *Advances and Applications in Computational Fluid Dynamics - ASME Winter Annual Meeting*, pages 109–119, 1988.

A. K. Sinha, D. G. Bogard, and M. E. Crawford. Film-cooling effectiveness downstream of a single row of holes with variable density ratio. *ASME Journal of Turbomachinery*, 113:442–449, 1991.

B. Stourstroup. *The C++ programming language*. Addison Wesley, 1997.

M. Surace. *Investigation of impingement systems for gas turbine combustor cooling*. PhD thesis, University of Florence, Energy Engineering Department, 2004.

F. M. White. *Viscous Fluid Flow*. McGraw-Hill, US, 1991.

K. Wieghardt and W. Tillman. On the turbulent friction layer for rising pressure. Technical Report TM-1314, NACA, 1951.

D. C. Wilcox. *Turbulence Modeling for CFD, 2nd Edition*. DCW Industries, US, 1998.

D. A. Yoder and N. J. Georgiadis. Implementation and validation of the chien $k-\varepsilon$ turbulence model in the wind navier-stokes code. Technical Report 209080, NASA, Glenn Research Center, Cleveland Ohio, 1999.