Politecnico di Milano
Energy Department
Doctoral Programme In Energy and Nuclear Science and Technology

---

Multidimensional simulations of external gear pumps

Doctoral Dissertation of:
Javier Martínez Rubio

Supervisor:
Prof. Federico Piscaglia

Co-supervisor:
Dr. Andrea Montorfano

Tutor:
Prof. Angelo Onorati

The Chair of the Doctoral Program:
Prof. Carlo Enrico Bottani

2017 – XXX Cycle STEN

To the memory of
my grandfather Pepe

Acknowledgements

This thesis represents not only my own work but it mainly belongs to a large number of people who one way or another collaborated to make this possible.

First and foremost I wish to express my sincere gratitude to my supervisor, Prof. Federico Piscaglia. Non only was he responsible of me deciding to start this Ph.D. degree some years ago, but the intense and stimulating discussions with him, his support and understanding have also been the key to achieving this milestone. This would not be possible either without the close collaboration with my co-supervisor, Dr. Andrea Montorfano. Their technical skills and expertise have always been a motivation for me to work harder and move forward.

I would like to extend this gratitude to the rest of professors and researchers in the Internal Combustion Engine Group at Politecnico di Milano. Special thanks to my tutor, Prof. Angelo Onorati, for his aid and support in the Ph.D. planning. Furthermore, I would like sincerely acknowledge Prof. Tommaso Lucchini. In all the projects I collaborated with him, he always stimulated my critical thinking and encouraged me with optimism and open mentality to challenging problems.

During this Ph.D. I was fortunate to be selected for a *Progetto Rocca* doctoral fellowship at Massachusetts Institute of Technology. This was all thanks to Prof. Emilio Baglietto. I owe him not just the opportunity of such a boost in my career, but also an immense amount of knowledge and self confidence I have gained by working with him. Being one of the greatest experts in the field of CFD I have ever met, his excellent advice, his attention and guidance both in the technical and human side have always motivated me and I will always be in debt with him for this. I hope this has only been the start of a much longer collaboration.

Prof. Baglietto was also responsible for my internship at Argonne National Laboratory, where I am very honored to have had Dr. Elia Merzari serving as my host. I owe him my gratitude for such an opportunity, for the great outcome of our collaboration and for all the connections I could establish at ANL thanks to him.

Leaving aside the academical support I must also acknowledge the important contribution of my friends and family.

Above all, I would like to express my appreciation to my parents. They have always been there, supporting, helping and advising in every decision. Giving their points of view and respecting my own. The same holds for every other member of my family, always worrying about me and taking care of everything I ever needed.

During these years I have made lots of friends form all around the world. First of all, special thanks to three very important colleagues for me: Dr. Yan Wu, Dr. Lorenzo Nocivelli and soon-to-be Doctor Isabella Verri. You have been present in all the good and bad moments and I will always remember all I have learned from you during these three years. Thanks also to two other future doctors: Davide Paredi and Filippo Giussani, real friends to lean on and share some laughs with at any time.

It is important to keep you focused while working, but it is just as important to relax and enjoy yourself when you go back home. I would like to acknowledge Francesco D'Aquino, my roommate in Milan for over a year and a true friend for over two.

Moving to Boston, I am truly fortunate to have found three of the greatest people I have met and whom I hope to maintain as lifelong friends: Anita Mollo, Benedetta Petracca and Paolo Minelli. Thank you all for making me feel home so many kilometers away. The experience at MIT would not have been the same without you.

In Chicago I was lucky to meet Jenae Armstrong. Thank you for your care, for the adventures and for all we have shared during my period there, and will keep sharing from now on.

Last but not least, some friends will always be there, year after year, nearby or in different continents. José Luis Hernández, Sergio Moya and Amaia Bretos, I am lucky to have you and this thesis also belongs to you.

Abstract

As a result of their low manufacturing costs, good performance and their capability to work in large pressure head ranges, external gear pumps have become widely employed in a variety of applications: from hydraulic fluid power systems to the petrochemical industry. The study presented here covers three important aspects related to the two- and three-dimensional simulations of gear pumps when a finite-volume method with a collocated grid is used (which is probably the most common approach in commercial codes). First of all, large pressure heads and small gaps in the gearing region result in extreme pressure gradients that significantly limit the stability of the pressure-velocity coupling algorithms employed in the solution of the fluid equations. Besides, results dependence on user-defined matrix solution parameters still exists in codes like OpenFOAM®, which was the main CFD tool selected for this thesis. Dealing with this problem, implementing alternative pressure-velocity coupling algorithms and selecting the best compromise in accuracy and convergence speed is the first scope of this thesis. Secondly, when the 3D simulation of helical pumps is considered, most available mesh motion algorithms require cycles of mesh deformation and replacement, which results in a tremendous computational cost. A simple, general and fully automatic approach to handle the required mesh modification process in two- and three-dimensional gear pump simulations (for both spur and helical gears) is presented here with the main aim of minimizing the use of computational resources while maintaining generality and ease of use. Finally, when a three-dimensional turbulent flow is considered, the case complexity typically forces the use of the U-RANS approach, since a properly resolved LES simulation would required excessive refinement leading to an enormous computational cost. As a compromised solution, a hybrid model (belonging to the so-called second generation U-RANS models) has been developed to overcome some of the deficiencies of classical U-RANS approaches while achieving a significant cost reduction over LES.

Keywords: helical gear pump, p-U coupling, dynamic mesh, hybrid turbulence model

Contents

List of Figures

List of Tables

Abbreviations and Nomenclature

Abbreviations

1D,2D,3D	One-, two- , three-dimensional
OpenFOAM®	Open Field Operation And Manipulation
ACMI	Arbitrary Coupled Mesh Interface
ALE	Arbitrary Lagrangian-Eulerian
AMI	Arbitrary Mesh Interface
ANL	Argonne National Laboratory
CDS	Central Differencing Scheme
CFD	Computational Fluid Dynamics
DDES	Delayed Detached Eddy Simulation
DES	Detached-Eddy Simulation
DIC	Diagonal Incomplete-Cholesky
DILU	Diagonal Incomplete LU
DNS	Direct Numerical Simulation
EGP	External Gear Pump
ELES	Embedded LES
ERCOFTAC	European Research Community On Flow, Turbulence And Combustion
IDES	Improved DES
JAEA	Japan Atomic Energy Agency
LDV	Laser Doppler Velocimetry
LES	Large Eddy Simulation
LHS	Left Hand Side
LU	Lower-Upper
LUST	Linear Upwind Stabilized Transport
LWR	Light Water Reactors
MIM	Momentum Interpolation Method
MIT	Massachusetts Institute of Technology
MMIM	Majumdar MIM
NACA	National Advisory Committee for Aeronautics
NLEVM	Nonlinear Eddy Viscosity Model
NS	Navier-Stokes
OMIM	Original MIM
p-U	Pressure-velocity
PANS	Partially Averaged Navier-Stokes
PBiCG	Preconditioned Bi-Conjugate Gradient

PCG	Preconditioned Conjugate Gradient
PDE	Partial Differential Equation
PIMPLE	Transient SIMPLE
PISO	Pressure Implicit with Splitting of Operators
PITM	Partially Integrated Transport Model
PRNS	Partially Resolved Numerical Simulation
QUICK	Quadratic Upstream Interpolation for Convective Kinematics
RANS	Reynolds Averaged Navier-Stokes
RHS	Right Hand Side
SAS	Scale-Adaptive Simulation
SBES	Stress Blended Eddy Simulation
SDES	Shielded DES
SEM	Spectral Element Method
SFR	Sodium Fast Reactors
SIMPLE	Semi-Implicit Method for Pressure Linked Equations
SIMPLEC	SIMPLE-Consistent
SIMPLER	SIMPLE-Revised
SST	Shear Stress Transport
STRUCT	Controlled STRUCT hybrid formulation with user defined parameters
STRUCTAf	STRUCTL with arithmetic averaging of frequency scales
STRUCTAf_V	STRUCTAf with integration length based on cell volume
STRUCTAt	STRUCTL with arithmetic averaging of time scales
STRUCTAt_V	STRUCTAt with integration length based on cell volume
STRUCTF	STRUCTL with geometric averaging based on face reconstruction
STRUCTL	STRUCT-Local formulation
STRUCTL_V	STRUCTL with integration length based on cell volume
STRUCTT	STRUCT-Transport formulation
STRUCTT-Tf_r	STRUCTT with no diffusion term and time scale computed as $T = 1/(\beta f_r)$
STRUCTT-T$k\epsilon$	STRUCTT with no diffusion term and time scale computed as $T = k/(\beta\epsilon)$
TKE	Turbulent kinetic energy
U-RANS	Unsteady RANS
VLES	Very Large Eddy Simulation

Greek Symbols

α	Involute gear pressure angle or STRUCTT-like models constant
α_u	Velocity equations under-relaxation factor
α_ϕ	Under-relaxation factor for generic flow variable ϕ
$\bar{\phi}$	Reynolds averaged, resolved or linear interpolation of a generic variable ϕ
β	Angle of attack for NACA 0012 airfoil profile case or constant

	of STRUCTT model
β_a	User-defined parameter for the maximum allowable change in the mesh non-orthogonal correction step
β_P	Auxiliary variable in Pascau MIM: $\beta_P = A_P^t / A_P^{nt}$
Δt	Time step size
Δx	Dimension of current cell in x-direction
Δx^+	Dimensionless wall cell size in streamwise direction
$\delta x_w, \delta x_e$	Cell-center to cell-center distance from current cell to west or east cells
Δy	Dimension of current cell in y-direction
$\delta y_n, \delta y_s$	Cell-center to cell-center distance from current cell to north or south cells
Δz^+	Dimensionless wall cell size in spanwise direction
Δ	Generic size of computational grid
Δ_f	Filter width in LES modeling
Δ_Γ	Parameter fixing the speed of the transition in the complete interface method
δ_{ij}	Kronecker delta
ϵ	Turbulence Dissipation Rate
ϵ_r	Contact ratio
γ	Blending parameter in the separated interface method
Γ_ϕ	Diffusion coefficient for the generic flow variable ϕ
Γ_{lim}	Parameter fixing the starting angle of the transition in the complete interface method
$\hat{\phi}$	Time average of a generic variable ϕ
λ	Blending parameter in the complete interface method
$\langle\phi\rangle$	Filtered generic variable ϕ in LES modeling
μ	Dynamic molecular viscosity
μ_t	Dynamic eddy viscosity
μ_{SGS}	Subgrid-Scale dynamic viscosity
ν	Kinematic laminar viscosity
ν_t	Kinetic eddy viscosity
$\nu_{contact}$	Artificial viscosity for the treatment of contact points
ν_{SGS}	Subgrid-Scale kinematic viscosity
ν_{tot}	Sum of laminar and artificial viscosity for the treatment of contact points
ω	Specific Turbulence Dissipation Rate
Ω_{ij}	Rotation tensor
$\overline{\Omega^*}$	Dimensionless rotation invariant: $\overline{\Omega^*} = k/\epsilon\sqrt{2\overline{\Omega_{ij}}\,\overline{\Omega_{ij}}}$
ϕ	Generic flow variable or STRUCT reduction function
ϕ'	Correction in the p-U coupling algorithm or deviation from Reynolds average of generic field ϕ
ϕ''	Second correction of variable ϕ in PISO algorithm or fluctuating part in LES filtering
ϕ^*	Random reference constant for provided parameters in application examples section

ϕ^m	Generic field ϕ at iteration m
ϕ^{**}	Single time corrected generic variable ϕ in PISO algorithm
ϕ_P^0	Generic variable ϕ at cell P at the previous iteration
ϕ_P^{k-1}	Generic variable ϕ at cell P at the previous time step
ϕ_{ref}	Reference value of a generic variable ϕ for the computation of the normalization factor
ψ	Generic flow variable
ρ	Density
σ_ϕ	Turbulent Prandtl number for the generic variable ϕ
τ_{ij}	Component i, j of the shear stress tensor
τ_{ij}^{LES}	LES stress tensor
τ_{ij}^{RANS}	RANS stress tensor

Roman Symbols

$\overline{S^*}$	Dimensionless shear invariant: $\overline{S^*} = k/\epsilon\sqrt{2\overline{S_{ij}}\,\overline{S_{ij}}}$
$\overline{II}$	Second invariant of the resolved velocity gradient tensor
$\mathbf{S_e}$	Face normal vector with magnitude equal to surface area
A'_P	Modified diagonal coefficient: $A'_P = A_P/\alpha_\phi$
A_b	Mass residual
A_d	Addendum
A_E, A_W, A_N, A_S	Off-diagonal coefficients
A_P	Diagonal coefficient
A_P^t	Time dependent contribution of A_P
A_P^{nt}	Non-time dependent contribution of A_P
a_{ij}	Anisotropic stress
b'_P	Modified source term: $b'_p = b_p + (1-\alpha_\phi)A_P\phi_P^0/\alpha_\phi$
B_P	Modified source term including time dependent contributions $B_P = b'_p + A_P^t u_P^{k-1}$
b_P	Source term
C_D	Drag coefficient
C_L	Lift coefficient
C_P	Pressure coefficient
D	Diameter in rod array case or nozzle width in triple array case
d	Clearance distance from tooth head to pump casing, or input variable of $f(d)$ function
D_d	Dedendum
D_i	Inlet diameter in application examples section
D_P	Pressure gradient contribution pre-multiplier in the definition of u_P: $D_P = \Delta x \Delta y/A_P$
D_p	Pitch diameter
$d_{\min}$	Parameter defining the minimum distance between gears
$d_{contact,i}$	Minimum distance between gears at contact point i
D_{t_m}	Diffusion term in t_m transport equation in STRUCTT-like

	models
$f(d)$	Smooth step function applied to the mesh generation process and to the contact point treatment
f_1, f_2, f_μ	Damping functions for low Reynolds models
f_ϵ	Residual to total ϵ ratio in PANS model
f_B	Blending functions for *blended* hybrid models
f_e^+	Weighting factor: $f_e^+ = \Delta x_P / 2\delta_{x_e}$
f_I	Interface function for $interface$ hybrid models
f_k	Residual to total k ratio in PANS model
f_m	Modeled fields frequency scale
f_r	Resolved fields frequency scale
f_{x_P}	x-direction pressure gradient contribution in u_P equation
G	Convolution kernel of LES filtering
H	Diffuser height in asymmetric diffuser case
h	Hill height in periodic hill case, or channel half-width for channel flow case
H_P	Contribution to u_P different from the pressure gradient
h_P	Contribution to u_P different from the pressure gradient in the case of under-relaxed velocity: $h_P = H_P + (1 - \alpha_u)u_P^0$
k	Turbulent kinetic energy or the user-defined constant in $f(d)$ function
k_ν	Maximum viscosity multiplier in the determination of $\nu_{contact}$
L, l	Generic distance, length scale, or integral length scale
L_i, L_o	Inlet and outlet pipe length in application examples section
L_{vK}	Von Karman length scale
M	Mach number or torque per unit length
M_z	Torque
n	Rotational speed
N, S, E, W	Cell located at North, South, East or West of current cell
n, s, e, w	North, south, east or west face of current cell
$n_{\theta_i}, l_{\theta_i}, r_{\theta_i}$	Number of cells, length and grading ratio in the i fragment of the gear profile edge in the mesh generation process
NB, nb	Neighbor
P	Pitch or generic point
p	Pressure
P_h	Helix pitch for helical gears
P_k	Production of turbulent kinetic energy
p_m	Pressure at m iteration or modified pressure $p_m = p + \frac{2}{3}\rho k$
$P_{contact,i}$	Determined location of i contact point between gears
q	Volumetric flow per unit length
q_z	Volumetric flow
R	Integration length in STRUCTL-like models
r	Fillet radius, or model activation parameter in STRUCT hybrid approach
r_c	Radial direction in cylindrical coordinates
r_i	Radial index in the node location method for the generation

	of the hypermatrix of cell vertices
R_p	Pitch radius
R_t	Turbulent Reynolds number $R_t = k^2/\nu\epsilon$
r_{ge}	Radius of epicycloid and hypocycloid generating circumference in cycloid tooth profile geometry
r_{split}	Profile splitting parameter
Re	Reynolds number
Re_y	Dimensionless wall distance $Re_y = y\sqrt{k}/\nu$
Re_τ	Reynolds number based on friction velocity and channel half-width: $Re_\tau = u_\tau h/\nu$
S_{ij}	Strain tensor
S_{t_m}	Time depending source term in t_m transport equation for the derived STRUCTT-like models
$spacing$	Additional separation between gears center points
St	Strouhal number
T	Temperature or time scale in t_m equation in STRUCTT-like models or gear rotation period
t	Time
T^*	Normalized mean temperature in triple jet case
t_c	Tangential direction in cylindrical coordinates
T_c, T_h, T_m	Cold, hot and mixed-mean temperature for triple jet case
t_D	Reference time for Taylor-Green vortices case
t_i	Tangential index in the node location method for the generation of the hypermatrix of cell vertices
t_m	Modeled fields time scale
t_{m0}	Locally computed modeled turbulent frequency scale: $f_{m0} = \epsilon/k$
t_{m0}	Locally computed modeled turbulent time scale: $t_{m0} = k/\epsilon$
TFI^*	Normalized temperature fluctuation in triple jet case
U, V, W	Generic velocity in x, y or z-direction
u^*	Velocity computed at momentum predictor step
U_b	Bulk velocity
u_i	Component i of velocity vector, occasionally substituted by (u, v, w)
u_τ	Friction velocity
U_{lid}	Reference lid velocity for the lid-driven cavity case
U_{ref}, U_∞	Reference velocity
V, V_P	Volume, volume of cell P
V_c, V_h, V_m	Cold, hot and mean discharge velocity for triple jet case
V_{exit}	Reference velocity for triple jet case
W	Power
x_i	Component i of position vector, occasionally substituted by (x, y, z)
y^+	Dimensionless wall distance
y_1, y_2	Distance from cell center to the closest point in *gea*1 or *gear2*
Z	Number of teeth

$\mathbf{S}$	Source term in velocity equations for the explicit porosity model

Chapter 1

Objectives and scope

1.1 Introduction

External gear pumps (EGP) play a major role in the framework of positive displacement turbomachinery [64]. Manufacturing simplicity and derived low cost, robustness, high volumetric efficiency and large pressure head range (from low pressure lubricating pumps to medium-high pressure open and close loop hydraulic circuits), make external gear pumps a popular and reliable choice for a wide variety of applications. While their rigid design gives them the ability to pump high viscosity fluids in the petrochemical industry, the tight internal clearances employed in their construction make them also well suited to handle applications that require precise flow control, including among others: metering applications of polymers, fuels or chemical additives.

Notwithstanding their relatively simple manufacturing, operation of gear pumps involve many complex phenomena both from the mechanical and from the fluid motion point of view. In fact, from the early models developed several decades ago [12, 83, 142], experimental evaluation and numerical modeling of external gear pumps has captivated the attention of many researchers. Available investigations focus on different aspects of the operation of this type of pumps, considering for instance the study of the inter-teeth pressure and forces distribution [32], fluid leakage [102], noise production [55], cavitation [46] or the effects of teeth geometry on the flow ripple [84].

1.2 Working principle

External gear pumps belong to the so called positive displacement pumps. As any other fluid power system, their mission is to supply hydraulic energy to the fluid from the mechanical energy introduced to the system typically by means of a shaft. In particular, positive displacement pumps are characterized by the presence of a suction side with an expanding cavity which brings fluid towards the pump, and a discharge side with a collapsing cavity that impulses the fluid out of the pump. This kind of systems are considered constant flow machines. Neglecting the volumetric efficiency, typically very high, a positive displacement machine would produce the same flow at a given speed independently of the discharge pressure.

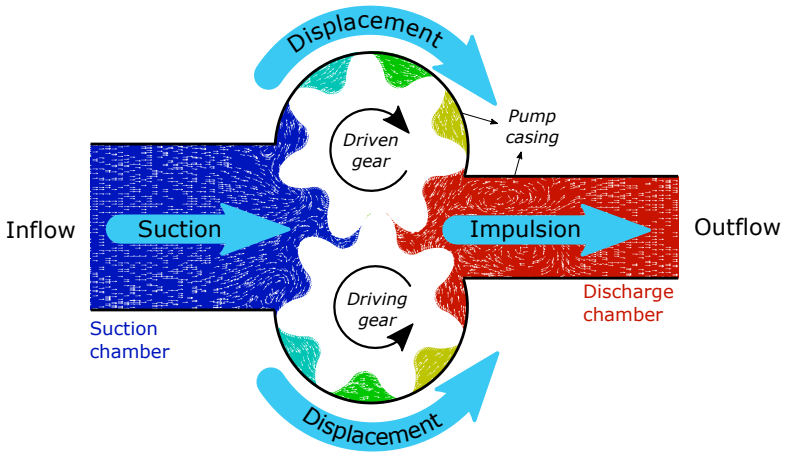


Figure 1.1: Definition of regions and energy transfer phases in an external gear pump

When a gear pump is used, the expansion and reduction of the volume is obtained by means of two generally identical gearwheels that mesh and rotate towards the suction chamber, as shown in Fig. 1.1. On the suction side, the fluid is trapped between the gears and the casing of the pump. On the discharge or pressure side, the fluid is squeezed out as the two gears rotate against each other. The power is externally introduced to the pump through a drive shaft, connected to the so-called 'driving gear', while the other one is dragged by meshing and referred as 'driven gear'. The simplicity of the system, involving a minimum number of parts increases the reliability of this kind of pumps compared to others with a more complex design. As a result, gear pumps can operate at high speeds and up to moderate pressure heads. According to the working principle described above, the energy transformation (mechanical to hydraulic) occurs in three phases [27]:

- **Suction:** The motion of both gears generate an opening volume that is continuously filled by the fluid. Inlet pressure, typically atmospheric, pushes the liquid towards the vacuum created at the opening volume.
- **Displacement:** Volumes of fluid are trapped in the region between the gears and the casing and are transported towards the discharge side.
- **Impulsion:** The volume containing the liquid is here reduced, impulsing the fluid towards the outlet pump.

Among the most important parameters defining the operation of a gear pump we can consider: pressure distribution on the teeth, presence of cavitation, pressure and flow ripple and volumetric efficiency. These parameters determine both the performance of the pump and the possible mechanical failure of the gears and pump casing, typically related to a fatigue process.

In order to reduce the stress suffered by the gears and casing, indentation of relief grooves is a common choice. These decompression slots (as shown in Fig. 1.2) allow to improve the fatigue life of the different elements of the pump. However they increase the liquid leakage, reducing the suction capability of the pump by decreasing its volumetric efficiency.

Figure 1.2: Decompression slots in the compensation plates of a gear pump [27]

In fact, the presence of relief grooves is of great importance when computing the volumetric efficiency of the pump. This parameter depends mainly on the following factors [27].

- **Leakage:** As shown in Fig. 1.3 fluid can find the way from the pressure side to the suction side through several paths allowing us to define:
  - Relief groove leakage: fluid from the meshing inter-teeth volume that leaks trough the volume trapped between contact points, due to the effect of the relief grooves.
  - Axial leakage: Fluid passage between the gears and the side plates.
  - Radial leakage: Movement of fluid through the teeth and the casing towards adjacent inter-teeth volumes.
- **Compressibility:** For a real pump the volumetric efficiency is also a function of the fluid working conditions.
- **Cavitation:** when cavitation occurs volumetric efficiency can be drastically reduced [27].

As it will be detailed in the next section (section 1.3), different simulation approaches will be more or less suitable for the correct prediction of the volumetric efficiency. While cavitation will not be considered in the present study, the proposed improvements will allow for a realistic estimation of the liquid leakage and compressibility effects.

1.3 State of the art

Regarding numerical simulations, predictions of external gear pumps behavior have been obtained from a variety of approaches. First of all, several mathematical and semi-empirical methods can be found in the literature. As an example, Manring *et al.* [84] studied the theoretical mass flow though external gear pumps neglecting leakages or cavitation effects. Others focused on the prediction of the point where cavitation starts: Myllykylä [98] developed a semi-empirical method

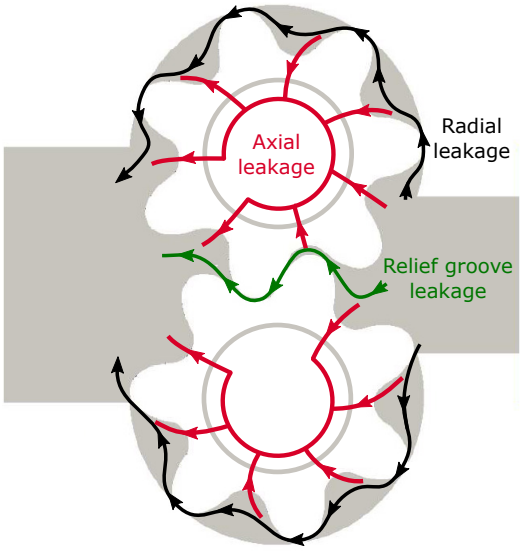


Figure 1.3: Liquid leakage paths in a gear pump

to determine suction capabilities of pumps before cavitation occurs and Khalaf [68] studied the performance of external gear pumps under marginal suction conditions. Some other authors studied these pumps from the point of view of the pressure pulsations. For instance, Eaton *et al.* [32] developed an analytical model based on an equivalent hydraulic circuit to predict the pressure ripple. Edge and Johnston [31] presented a method to estimate the flow ripple from pressure ripple measurements. While mathematical models and theoretical derivations can provide accurate predictions in some cases, their applications is typically restricted to very particular designs and difficult to generalize without the appropriate calibration.

The next degree of generalization consists in the use of very sophisticated 1D models. Generally speaking, 1D codes model gear pumps by considering the fluid region to be divided in control volumes, each of which represents the volume of the fluid trapped between two teeth and the casing, or between the teeth of both gears in the gearing region. Properties within the chambers are uniform and the evolution of these properties depends on the interaction with the adjoining chambers by variable orifices. Despite the extreme assumptions, successful description of the phenomena can be found for simple straight-cut gears [11, 80, 96]. The main drawbacks of these approaches are: the difficult description of the geometrical volumes and throat areas a function of the shaft angular position for a general user-given profile with given clearance values, and the lack of a detailed description of the pressure distribution in the teeth. Besides consideration of three-dimensional effects and the extension to helical gears becomes critical.

With the advent of low-cost large-scale computing, the use of Computational Fluid Dynamics (CFD) has become a promising solution. A complete description of the fluid region in the pump is now available without the need of extreme assumptions or complex definitions of volume zones and throat areas. Besides CFD is the only simulation tool that allows us to treat the problem in its full com-

plexity, considering for instance three-dimensional effects, fluid compressibility, turbulence, cavitation and fluid structure interaction. In CFD, the fluid region is divided in discrete cells (computational mesh) where the governing equations are solved [35]. Among the different methods available nowadays, the finite-volume method is the preferred choice for many commercial codes for the solution of the fluid motion conservation equations (Navier-Stokes equations). Finite-volume methods are simple, robust and a natural choice when the partial differential equations to be solved are conservation laws. Finite-volumes are based on the enforcement of the integral form of the conservation laws in each of the small control volumes defined by the computational mesh. Finite-volume methods can further be divided as function of how the variables are stored in the computational grid. In the so called staggered grid arrangement, scalar variables (such as density, pressure, enthalpy...) are stored in the cell centers of the control volumes, whereas momentum variables (velocity) is stored at the cell faces. This method benefits from a higher stability and robustness but substantially increases the memory storage requirements and calculation time when compared to its alternative, the co-located grid arrangement [35]. In the co-located grid arrangement all variables are stored in the cell centers, which significantly simplifies the implementation of solvers. This, together with the reduced memory storage requirements has made co-located grid arrangement gain popularity in the recent years in both general-purpose and commercial flow solvers. However some problems arise from the use of co-located grids. In particular the most significant one is the so called checkerboard pressure. Due to the nature of the Navier-Stokes equations, pressure appears in the momentum equations inside a gradient term. The application of central-difference spatial discretization to this term in a co-located grid, produces a decoupling of pressure and velocity cell values, leading to saw-tooth oscillations. As it will be explain in Chapter 2, this problem was addressed by using a technique proposed long ago, the momentum interpolation methods [112]. However the typically required equations under-relaxation combined with the use of momentum interpolation methods can generate additional problems that had not been addressed in the main CFD code used in this thesis, OpenFOAM®, by the time this work was performed.

When finite-volume methods with co-located grids are used for the simulation of external gear pumps, the huge pressure gradients between adjoining lateral chambers or in the gears meshing zone accentuate the pressure-velocity decoupling problem and therefore the commented problems. The first part of the thesis will focus on the study of Momentum Interpolation Methods (MIM) correcting these problems, as well as their influence on the accuracy and convergence speed of several pressure-velocity coupling algorithms, with the purpose of selecting the most appropriate choice to be used in the simulation of gear pumps.

The second critical aspect considered in this thesis lies on the meshing process. Given the complexity of the moving boundaries in the gears meshing region and the narrow gear-to-gear and gear-to-casing gaps, mesh generation and motion is critical. Arbitrary Lagrangian-Eulerian (ALE) formulations [28] are a common choice and have been successfully applied in literature [14, 61, 133, 140]. In the usual Eulerian method, equations including convective terms,are solved in a steady

mesh. With a pure Lagrangian method the mesh moves with the fluid particles and the convective term vanishes. ALE approach tries to adopt an intermediate solution. However, when ALE methods are used, even if mesh deformation is combined with local re-meshing, mesh quality frequently needs to be improved and most authors rely on a mesh replacement strategy (see for example [13, 14, 69]). Regularly, an un-structured mesh is created for the initial position of the gears. As the gears rotate, mesh points are displaced following the gears motion and the quality of the resulting mesh is controlled. When the quality lies below a given threshold, a new mesh is generated for that particular position of the gears. Variables are then interpolated from one mesh into the other and simulation proceeds with the new mesh, repeating the process. The approach is frequently used due to its simplicity. However, it is clear that the mesh generation process can consume a significant amount of computational power, since it needs to be performed every small angle of gear rotation. Besides, interpolation of variables between meshes generate numerical errors which are difficult to control, and a good mesh quality in regions near contact points or circumferential clearance gaps is hard to achieve.

The method can be however efficient when applied to spur gears (straight-cut), since the 3D mesh generation can be simplified to the extrusion of a 2D mesh, which is faster to create. This is not the case when helical gears are considered. Some alternative methods are based on: dynamic non-structured mesh deformation with cells refinement/agglomeration [133] which requires a very small time step not to reduce significantly the mesh quality; deformation of a block-structured mesh based on a Laplacian equation solved in an un-structured mesh [140], which requires therefore the management of two meshes in parallel with the related computational cost; and mesh superposition [9], which suffers again from numerical diffusion given by mesh-to-mesh interpolation. The difficulty when a helical gear is considered increases significantly. In fact, some authors [59] even relay on a series of spur gears rotated according to a helix angle, eliminating the smoothness of the helical surface, leading to a simpler but not very realistic method. Other commercial CFD software include the capabilities of generating a single structured deforming mesh that adapts as the gears rotate. Typically the user has to define an interface between the gears that is used for mesh generation purposes. Having this idea in mind, Chapter 3 in this thesis will described a new mesh motion strategy which tries to reduce to a minimum the number of parameters defined by the user, while considering a general method that can be applied to spur or helical gears with any user-given profile. In the same chapter, different options for the treatment of the contact points between gears will also be described.

Last but not least, even if velocities in the chambers of the pump are no high enough for the computed Reynolds number to be in the turbulent range, it has been experimentally shown [15] that the stirring produced by the gears and the associated variations of volume and pressure, result in an injection of energy that cannot be dissipated by molecular viscosity and therefore makes turbulence arise. In the simulation of turbulent flows, Direct Numerical Simulations (DNS) and Reynolds Averaged Navier-Stokes simulations (RANS), or their unsteady counterpart (U-RANS) represent the two extremes for turbulence modeling [35]. DNS

is the most accurate and simplest approach from a conceptual point of view, as it consists in solving all scales contained in the flow without any averaging or approximation, thus incurring in the highest computational cost (wall time and hardware resources such as processors and memory). On the other hand, RANS simulations solve for the mean flow with an appropriate model for the turbulent energy transfer among the various scales. Large Eddy Simulations (LES) lie between the two extremes. LES is based on the numerical solution of large scales of fluid structures combined with a proper modeling of the effect of small-scale motions. The stringent requirements regarding computational grid size and time step make DNS and very often LES too, impractical in most industrial applications, due to limitations in processing power and storage capabilities. Reasonable computational cost and acceptable accuracy make RANS modeling the most widely used approach. For most applications either the geometry is too large or the Reynolds number too high to leverage LES and U-RANS is the preferred method. As an example, computational requirements in turbo-machinery simulations are increased by a factor of of $10^5 - 10^7$ when switching from U-RANS to LES [88]. However, RANS does not solve for the unsteady flow structures, the intrinsic characteristic of turbulent flows, and indeed it may fail to reproduce relevant flow physics for many engineering problems. Hybrid models try to bridge the gap between the two approaches (U-RANS and LES) by controlling the amount of turbulent kinetic energy that is modeled. Even if the statistical assumptions used in the derivation of RANS and LES equations differ, they share significant similarities [44] that can be leveraged in the derivation of hybrid models. The potential benefits of such approach can be understood when considering the numerous hybrid models available in literature. Some of the most well-known ones include: Detached-Eddy Simulations (DES) [125], Embedded LES (ELES) [21], Scale-Adaptive Simulations (SAS) [91] and Partially Averaged Navier-Stokes (PANS) [49]. For the present thesis a new hybrid model has been developed focusing on its application to external gear pumps, leveraging the STRUCT model developed by Giancarlo Lenci [75].

1.4 Objectives

The main aim of this project is to provide advanced algorithms, all from the point of view of the pressure-velocity coupling, mesh motion strategy and turbulence modeling, to facilitate the study of external gear pumps with a co-located grid finite-volume CFD approach such as OpenFOAM®. In particular the main contributions and objectives of the present thesis can be summarized as:

- Study and **correction of the MIM** used in OpenFOAM® for the pressure-velocity coupling algorithms in incompressible solvers.
- Study of the **influence of MIM on the accuracy and convergence speed** of the solvers, aiming to determine the most appropriate solver for the simulation of gear pumps.
- Development and implementation of a **new mesh generation and mesh motion strategy** for the simulation of 2D and 3D (spur or helical) gear

pumps. Such approach should be general (the same approach should work for any user-given gear profile), it should minimize the number of user-defined parameters and incur in the lowest achievable computational cost.

- Development and implementation of different models for the **treatment of the contact point** between gears when the previously mentioned mesh motion strategy is used.
- Development of a **hybrid turbulence model** to reduce the computational cost that would be derived from the use of LES approach, aiming to correct some of the inherent deficiencies of U-RANS for applications such as 3D gear pumps.

1.5 Organization

The logical organization of the thesis covers the objectives mentioned in section 1.4 in an structured manner.

- This chapter serves as an introduction. It defines the working principles of external gear pumps and presents the main difficulties found when considering their study through CFD simulations.
- Chapter 2 covers the study of Momentum Interpolation Methods and its application to the pressure-velocity coupling algorithms used in OpenFOAM® solvers. In particular the dependence of steady state solutions on velocity under-relaxation factors and time step size is explained and the available corrections are tested and applied to simple cases. As a next step the influence of MIM in both accuracy and convergence speed of different p-U coupling algorithms is evaluated and extracted conclusions are used to select the most appropriate solver for the simulation of gear pumps.
- Chapter 3 minutely describes the developed mesh motion algorithms. Special emphasis is given to the starting mesh generation tool, the topological changes required in the mesh, and to its implementation in OpenFOAM®. Examples of its application to a variety of gear profiles are given in order to show the capabilities of the approach. Furthermore, being linked to the mesh motion strategy, the implemented algorithms for the treatment of the contact point between gears is here described.
- Chapter 4 contains the work developed during a one-year visiting period at Massachusetts Institute of Technology (MIT), where a new hybrid turbulence model (STRUCT) was developed. The main idea behind the model is described in detail and several variants of the model are introduced together with their testing on simplified cases that resemble the physical phenomena found in the flow through a gear pump. The base RANS model in which the hybrid approach is based has also been implemented in a very different code (the Spectral Element Method code Nek5000) at Argonne National Laboratory (ANL), as a first step in the implementation of the hybrid approach. Notes on the partial results obtained during the two-month visiting period

at ANL will also be given, to show the capabilities of the underlying RANS model behind STRUCT.

- Chapter 5 shows examples of the application of the algorithms explained above to one commercial gear pump taken as a reference. This comes from the collaboration with a private company (whose name will not be given for confidentiality reasons), interested in the developed algorithms for the simulations of some of their gear pump models.

- Chapter 6 serves as a summary and proposes possible improvements of the presented methods as well as future work to further approach the simulation of gear pumps under more realistic conditions.

Chapter 2

P-U coupling algorithms

2.1 Introduction

Checkerboard pressure is one of the most significant problems arising from the use of co-located grids for fluid dynamics simulations using the finite-volume method. When a central-difference spatial discretization scheme is applied to the pressure gradient in velocity equations, the decoupling of pressure and velocity cell values leads to saw-tooth pressure oscillations. Aiming to face this problem, Rhie and Chow [112] proposed a technique for momentum based interpolation of mass fluxes on cell faces, imitating the staggered grid discretization. This interpolation is based on formulating a discretized momentum equation for the face, so that the computation of the driving pressure force involves the pressure value at the nodes adjacent to the face in question, and therefore at the node itself. This technique removes the pressure checkerboarding problem for the most part, which is the reason of its wide acceptance and intensive use in unstructured grid solvers. This Momentum Interpolation Method (MIM) underwent extensive development for complex geometries [19], unsteady flows [17, 67, 118, 143] or flows with large body forces [18].

Velocity under-relaxation is usually required to achieve convergence. When this is the case, the Original Momentum Interpolation Method (OMIM) by Rhie-Chow presents some additional problems. Majumdar [82] and Miller *et al.* [92] independently reported that solutions obtained with the original Rhie-Chow interpolation method were dependent on the velocity under-relaxation factor. Furthermore, the use of very small under-relaxation factors could make the checkerboard pressure reappear. The corrected version of momentum interpolation proposed in [82], termed as the Majumdar Momentum Interpolation Method (MMIM) completely eliminates this dependency.

Additionally, Choi [17] was the first to report that original Rhie-Chow interpolation is also time step-size dependent and proposed a correction similar to that of Majumdar. However, some years later, Yu [143] showed that the correction proposed by Choi did not cancel the dependence, and suggested an alternative modification to eliminate the problem. Some other authors extended the derivations for the use of first and second order time discretization schemes. In particular, based on the work from Cubero and Fueyo [22], Pascau [103] proposed an

alternative MIM that would correctly eliminate the time step dependency in a more generalized approach.

In the current chapter the OMIM will be described and the dependence of steady state solution on under-relaxation factors and time step size will be explained. Some of the proposed corrections available in literature will be implemented and tested in the OpenFOAM® technology, which will be used to evaluate the importance and performance of the corrections. Once some conclusions have been extracted on this respect, a systematic study of several MIM on the solution accuracy and convergence speed will be applied to four well-known segregated pressure-velocity coupling algorithms, namely, SIMPLE (Semi-Implicit Method for Pressure Linked Equations) [105], SIMPLER (SIMPLE-Revised) [104], SIMPLEC (SIMPLE-Consistent) [29] and PISO (Pressure Implicit with Splitting of Operators) [63].

2.2 Rhie-Chow OMIM

In the following section the OMIM will be described in detail. The dependency of steady state OMIM solutions to the velocity under-relaxation factor (α_u) and time step size will be discussed. First, the steady equations will be analyzed to show the dependence on α_u while the time step dependence will be discussed later.

The governing equations for a steady laminar incompressible flow, in the absence of other body or external forces are:

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

$$\frac{\partial}{\partial x_j}(\rho u_i u_j) = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} \tag{2.2}$$

where x_i is the cartesian framework, u_i is the velocity component in the i coordinate, p is the pressure and ρ denotes the density. Assuming a Newtonian fluid, shear stress τ_{ij} can be determined by:

$$\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\mu\frac{\partial u_k}{\partial x_k}\delta_{ij} \tag{2.3}$$

where μ is the dynamic dynamic viscosity and δ_{ij} is the Kronecker delta. For the sake of simplicity, the OMIM will be described in a two-dimensional orthogonal grid as the one shown in Fig. 2.1.

Under this considerations, the conservation equation for a general flow variable ϕ in the steady laminar case described above, in the absence of any source term, can be written as:

$$\frac{\partial}{\partial x}(\rho u\phi) + \frac{\partial}{\partial y}(\rho v\phi) = \frac{\partial}{\partial x}\left(\Gamma_\phi\frac{\partial \phi}{\partial x}\right) + \frac{\partial}{\partial y}\left(\Gamma_\phi\frac{\partial \phi}{\partial y}\right) \tag{2.4}$$

where u and v are the x and y components of the velocity field and Γ_ϕ is the diffusion coefficient. Integrating Eq. (2.4) in the computational cell, and applying

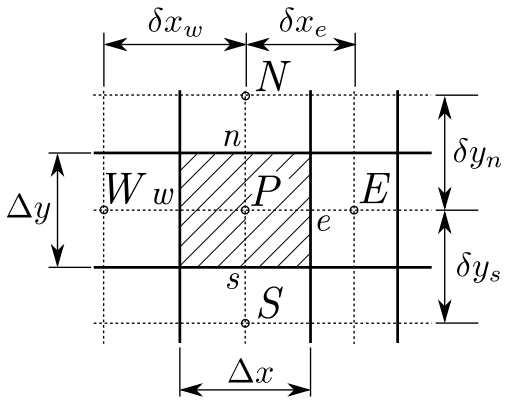


Figure 2.1: Two-dimensional orthogonal grid. N, S, E and W correspond to neighbor cells of cell P; n, s, e and w denote cell P faces; Δx and Δy are cell P dimensions in the x and y spatial coordinates; δy_n, δy_s, δx_e and δx_w correspond to cell-center to cell-center distances from cell P to neighbor cells

Green-Gauss theorem the semi-discretized form of the equation can be written as follows:

$$\begin{aligned}\Delta y\left[(\rho u\phi)_e-(\rho u\phi)_w\right]+\Delta x\left[(\rho v\phi)_n-(\rho v\phi)_s\right]= \\ \Delta x\left[\frac{\Gamma_n}{\delta y_n}(\phi_N-\phi_P)-\frac{\Gamma_s}{\delta y_s}(\phi_P-\phi_S)\right] \\ +\Delta y\left[\frac{\Gamma_e}{\delta x_e}(\phi_E-\phi_P)-\frac{\Gamma_w}{\delta x_w}(\phi_P-\phi_W)\right]\end{aligned} \tag{2.5}$$

where central differencing scheme has been used for diffusive terms. In the discretization of divergence terms, many schemes are available in literature (a group of classical schemes includes: first-order upwind, second-order upwind, central differencing scheme, QUICK [78]). Second and higher-order schemes have been widely used in applications involving orthogonal and uniform meshes, but the stability of higher-order schemes for applications involving turbulent flows in complex geometries is not guaranteed and convergence may be difficult to achieve. For these applications, first or second-order methods are best suited.

Application of the first-order upwind discretization scheme to convective terms of Eq. 2.5 yields the following final discretized form of Eq. (2.4).

$$A_P\phi_P=A_E\phi_E+A_W\phi_W+A_N\phi_N+A_S\phi_S+b_P \tag{2.6}$$

where

$$\begin{aligned}
A_E =& \frac{\Gamma_e \Delta y}{\delta x_e} + \max(-\rho u_e \Delta y, 0) \\
A_W =& \frac{\Gamma_w \Delta y}{\delta x_w} + \max(\rho u_w \Delta y, 0) \\
A_N =& \frac{\Gamma_n \Delta x}{\delta y_n} + \max(-\rho v_n \Delta x, 0) \\
A_S =& \frac{\Gamma_s \Delta x}{\delta y_s} + \max(\rho v_s \Delta x, 0) \\
b_P =& -\max(\rho u_e \Delta y, 0)(\phi_e - \phi_P) + \max(-\rho u_e \Delta y, 0)(\phi_e - \phi_E) \\
& -\max(-\rho u_w \Delta y, 0)(\phi_w - \phi_P) + \max(\rho u_w \Delta y, 0)(\phi_w - \phi_W) \\
& -\max(\rho v_n \Delta x, 0)(\phi_n - \phi_P) + \max(-\rho u_n \Delta x, 0)(\phi_n - \phi_N) \\
& -\max(-\rho v_s \Delta x, 0)(\phi_s - \phi_P) + \max(\rho u_s \Delta x, 0)(\phi_s - \phi_S)
\end{aligned} \tag{2.7}$$

where deferred-correction procedure [35] is used for the term b_P.

A_P coefficients can be determined by

$$A_P = A_E + A_W + A_N + A_S + A_b \tag{2.8}$$

where A_b is the mass residual. The A_b term is usually dropped since divergence-free conditions are required for the velocity field:

$$A_b = \rho u_e \Delta y - \rho u_w \Delta y + \rho v_n \Delta x - \rho v_s \Delta x \tag{2.9}$$

Eq. (2.8) is true as long as conservative discretization schemes are used [35]. It should also be noticed that a diagonally dominant matrix ($A_P \geq \sum_{NB} |A_{NB}|$ where NB refers to the neighbor cells of cell P) is a sufficient condition for convergence of iterative methods and the inequality must be satisfied at least at one node [35].

In order to reach convergence under-relaxation factors are typically applied to dependent variables to limit the change in consecutive iterations. From Eq. (2.6) we can write the final value of ϕ_P as:

$$\phi_P = \frac{\alpha_\phi}{A_P}(A_E \phi_E + A_W \phi_W + A_N \phi_N + A_S \phi_S + b_P) + (1 - \alpha_\phi)\phi_P^0 \tag{2.10}$$

where superscript 0 is used for the quantities calculated at the previous iteration and α_ϕ is the under-relaxation factor for variable ϕ. Based on the above discussion, Eq. (2.6) can be written as:

$$A'_P \phi_P = A_E \phi_E + A_W \phi_W + A_N \phi_N + A_S \phi_S + b'_P \tag{2.11}$$

where $A'_P = A_P/\alpha_\phi$ and $b'_P = b_p + \frac{(1-\alpha_\phi)}{\alpha_\phi} A_P \phi_P^0$.

Let us now consider the equation for x-component of the velocity field, u. Before under-relaxation u_P would follow an equation similar to Eq. (2.6). If the

pressure term is separated from the source we can write:

$$u_P = \frac{\sum_{NB} A_{NB} u_{NB} + b_P}{A_P} - \frac{\Delta y (p_e - p_w)}{A_P} \tag{2.12}$$

which can also be expressed in general form:

$$u_P = H_P - D_P (\nabla p)_P \tag{2.13}$$

where $H_P = \frac{\sum_{NB} A_{NB} u_{NB} + b_P}{A_P}$, $D_P = \frac{\Delta x \Delta y}{A_P}$ and $(\nabla p)_P$ is the x component of the pressure gradient in cell P. Therefore, for the two contiguous cells E and P we could write the following equations:

$$u_P = \frac{(\sum_{NB} A_{NB} u_{NB} + b_P)_P}{(A_P)_P} - \frac{\Delta y (p_e - p_w)_P}{(A_P)_P} \tag{2.14}$$

$$u_E = \frac{(\sum_{NB} A_{NB} u_{NB} + b_P)_E}{(A_P)_E} - \frac{\Delta y (p_e - p_w)_E}{(A_P)_E} \tag{2.15}$$

Mimicking formulation followed for u_E and u_P, Rhie-Chow proposed a pseudo-equation for the face velocity u_e:

$$u_e = \frac{(\sum_{NB} A_{NB} u_{NB} + b_P)_e}{(A_P)_e} - \frac{\Delta y (p_E - p_P)}{(A_P)_e} \tag{2.16}$$

which can also be expressed as:

$$u_e = H_e - D_e (\nabla p)_e \tag{2.17}$$

where H_e is the first term in the RHS of Eq. (2.17), $D_e = \frac{\delta x_e \Delta y}{(A_P)_e}$ and $(\nabla p)_e$ is the x component of pressure gradient in face e. In the OMIM of Rhie and Chow, unknown terms of RHS of Eq. (2.16) are obtained by linear interpolation as follows:

$$\begin{aligned}\left(\frac{\sum_{NB} A_{NB} u_{NB} + b_P}{A_P}\right)_e = & f_e^+ \left(\frac{\sum_{NB} A_{NB} u_{NB} + b_P}{A_P}\right)_E \\ & + (1 - f_e^+) \left(\frac{\sum_{NB} A_{NB} u_{NB} + b_P}{A_P}\right)_P\end{aligned} \tag{2.18}$$

$$\frac{1}{(A_P)_e} = f_e^+ \frac{1}{(A_P)_E} + (1 - f_e^+) \frac{1}{(A_P)_P} \tag{2.19}$$

where f_e^+ is a weighting factor that for the mesh shown in Fig. 2.1 can be determined as $f_e^+ = \frac{\Delta x_P}{2 \delta x_e}$. Therefore, Eq. (2.17) becomes:

$$u_e = \overline{H_e} - \overline{\left(\frac{1}{A_e}\right)} \Delta y (p_E - p_P) \tag{2.20}$$

where the over-bar denotes linear interpolation.

A similar procedure can be applied for other face velocities of cell P. For the sake of simplicity and brevity, this procedure is shown only for face e.

Substituting Eq. (2.18) into Eq. (2.16), considering Eqs. (2.14) and (2.15), and re-ordering terms we obtain:

$$\begin{aligned} u_e = &\left[f_e^+ u_E + (1-f_e^+)u_P\right] \\ &- \frac{\Delta y(p_E - p_P)}{(A_P)_e} \\ &+ \left(f_e^+ \frac{\Delta y(p_e - p_w)_E}{(A_P)_E} + (1-f_e^+)\frac{\Delta y(p_e - p_w)_P}{(A_P)_P}\right) \end{aligned} \tag{2.21}$$

The first term in the RHS of Eq. (2.21) corresponds to the linear interpolation of cell values, while the last two terms can be regarded as a correction term that smooths the pressure field, and removes the undesired checkerboard behavior. Assuming $D_e \approx \overline{D_e}$ and $\overline{(D\nabla p)_e} \approx \overline{D_e}\,\overline{\nabla p_e}$, and considering Eqs. (2.13) and (2.17), Eq. (2.21) can be re-written as:

$$u_e = \overline{u_e} + \overline{D_e}\left(\overline{\nabla p_e} - (\nabla p)_e\right) \tag{2.22}$$

which is the classical Rhie-Chow interpolation formula.

However, in general, under-relaxation is required for the momentum equations. When considering under-relaxation, Eq. (2.14) becomes:

$$\begin{aligned} u_P &= \frac{\alpha_u \left(\sum_{NB} A_{NB} u_{NB} + b_P\right)_P}{(A_P)_P} + (1-\alpha_u)u_P^0 - \frac{\alpha_u \Delta y(p_e - p_w)_P}{(A_P)_P} \\ &= \alpha_u \left[H_P - \frac{\Delta y(p_e - p_w)_P}{(A_P)_P}\right] + (1-\alpha_u)u_P^0 \\ &= h_P - \frac{\alpha_u \Delta y(p_e - p_w)_P}{(A_P)_P} \end{aligned} \tag{2.23}$$

where $h_P = \alpha_u H_P + (1-\alpha_u)u_P^0$. Similarly, for cell E we find:

$$u_E = h_E - \frac{\alpha_u \Delta y(p_e - p_w)_E}{(A_P)_E} \tag{2.24}$$

and $h_E = \alpha_u H_E + (1-\alpha_u)u_E^0$.

If Rhie-Chow momentum interpolation is now introduced, following Eq. (2.20) we can write:

$$\begin{aligned} u_e &= \overline{h_e} - \alpha_u \overline{\left(\frac{1}{A_e}\right)} \Delta y(p_E - p_P) \\ &= \left(f_e^+ h_E + (1-f_e^+)h_P\right) - \alpha_u \overline{\left(\frac{1}{A_e}\right)} \Delta y(p_E - p_P) \end{aligned} \tag{2.25}$$

Using definitions of h_E and h_P in terms of H_P and H_E, u_e can be written as:

$$u_e = \alpha_u \underbrace{\left[\overline{H_e} - \overline{\left(\frac{1}{A_e}\right)} \Delta y(p_E - p_P)\right]}_{\text{Momentum interpolation}} + (1-\alpha_u) \underbrace{\left[f_e^+ u_E^0 + (1-f_e^+)u_P^0\right]}_{\text{Linear Interpolation}} \tag{2.26}$$

The above discussion makes it clear that the direct application of Rhie-Chow technique to the under-relaxed momentum equations would yield a solution that will not converge to the desired momentum interpolation (first term), but will contain a portion of linear interpolation. Therefore, when a small enough under-relaxation factor is used, the second term in Eq. (2.26) will be the main contribution, and pressure oscillations may re-appear, since no momentum interpolation is used.

In order to understand the time step dependency let us consider the transient Navier-Stokes equations under the same assumptions of Eqs. (2.1) and (2.2):

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}(\rho u_i) = 0 \tag{2.27}$$

$$\frac{\partial (\rho u_i)}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_i u_j) = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} \tag{2.28}$$

Following a similar procedure as the one used in the steady case, and using implicit first order Euler scheme for the time derivative term, the discretized form of x-velocity equation now yields:

$$A_P^t(u_P - u_P^{k-1}) + A_P^{nt} u_P = A_E u_E + A_W u_W + A_N u_N + A_S u_S + b_P \tag{2.29}$$

where $A_P^t = \frac{\rho V_P}{\Delta t}$ represents the part of the diagonal coefficient related to the time derivative term, and A_P^{nt} contains the rest of diagonal terms. b_P contains now all explicit terms except for the one coming from the discretization of the time derivative term.

Before under-relaxation, separating the pressure term from b_P, u_P follows an equation similar to (2.12)

$$u_P = \frac{\sum_{NB} A_{NB} u_{NB}}{A_P} + \frac{b_P + A_P^t u_P^{k-1}}{A_P} - \frac{\Delta y (p_e - p_w)}{A_P} \tag{2.30}$$

where $A_P = A_P^t + A_P^{nt}$.

Considering now under-relaxation, the equation for x-velocity at P becomes:

$$\begin{aligned} u_P =& \frac{\alpha_u \left(\sum_{NB} A_{NB} u_{NB} + b_P + A_P^t u_P^{k-1} \right)_P}{(A_P)_P} - \frac{\alpha_u \Delta y (p_e - p_w)_P}{(A_P)_P} + (1 - \alpha_u) u_P^0 \\ =& \frac{\alpha_u \left(\sum_{NB} A_{NB} u_{NB} + B_P \right)_P}{(A_P)_P} - \frac{\alpha_u \Delta y (p_e - p_w)_P}{(A_P)_P} \end{aligned} \tag{2.31}$$

where B_P now reads: $B_P = b_p + A_P^t u_P^{k-1} + \frac{1-\alpha_u}{\alpha_u} A_P u_P^0$

Considering the equivalent equation for cell E, a direct application of Rhie-Chow momentum interpolation technique (following Eq. (2.16)) yields the following equation for the face velocity:

$$u_e = \frac{\alpha_u \left(\sum_{NB} A_{NB} u_{NB} + B_P \right)_e}{(A_P)_e} - \frac{\alpha_u \Delta y (p_E - p_P)}{(A_P)_e} \tag{2.32}$$

Bringing back the definition of B_P and after some reorganization:

$$
\begin{aligned}
&u_e = \\
&\alpha_u \left[f_e^+ \left(\frac{\sum A_{NB} u_{NB} + b_P}{A_P} \right)_E + (1 - f_e^+) \left(\frac{\sum A_{NB} u_{NB} + b_P}{A_P} \right)_P - \frac{\Delta y (p_E - p_P)}{(A_P)_e} \right] \\
&+ (1 - \alpha_u) \left[f_e^+ u_E^0 + (1 - f_e^+) u_P^0 \right] \\
&+ \alpha_u \left[f_e^+ \frac{(A_P^t)_E}{(A_P)_E} u_E^{k-1} + (1 - f_e^+) \frac{(A_P^t)_P}{(A_P)_P} u_P^{k-1} \right]
\end{aligned} \tag{2.33}
$$

From Eq. (2.33) we can extract several conclusions. First, we can see that even at convergence (where $u_P = u_P^{k-1} = u_P^0$ and $u_E = u_E^{k-1} = u_E^0$) the previous expression yields a value of u_e that depends both on α_u and on Δt (let us remember that Δt is present in A_P^t and in A_P). Second, it should be noticed that when a very small time step is used, the fractions involved in the last term of Eq. (2.33) tend to unity:

$$
\lim_{\Delta t \to 0} \frac{A_P^t}{A_P} = \lim_{\Delta t \to 0} \frac{\rho V_P / \Delta t}{A_P^{nt} + \rho V_P / \Delta t} = 1 \tag{2.34}
$$

Therefore face velocity tends to the linear interpolation $u_e = f_e^+ u_E^0 + (1 - f_e^+) u_P^0$, which means that the pressure oscillations may re-appear for very small time steps, even if velocity equations are not under-relaxed.

In the following sections two methods correcting these deficiencies of the OMIM will be described. While some other of the alternatives introduced in section 2.1 have also been implemented, they will not be shown for the sake of brevity.

2.3 Majumdar MIM

In order to correct the under-relaxation factor dependency of the steady state solution, Majumdar [82] forced the convergence of u_e to the full momentum interpolation value by applying explicit relaxation on the face velocity:

$$
u_e = \alpha_u \left[\overline{H_e} - \overline{\left(\frac{1}{A_e} \right)} \Delta y (p_E - p_P) \right] + (1 - \alpha_u) u_e^0 \tag{2.35}
$$

Obviously, in an iterative procedure, u_e will converge to the momentum interpolation value, as described in Eq. 2.20. Expressing $\overline{H_e}$ in terms of H_E and H_P, and these in terms of h_E and h_P, similarly to what has been done in Eqs. 2.23 and 2.24, it follows:

$$
\begin{aligned}
u_e = &\left[f_e^+ h_E + (1 - f_e^+) h_P \right] + \alpha_u \overline{\left(\frac{1}{A_e} \right)} \Delta y (p_E - p_P) \\
&+ (1 - \alpha_u) \left[u_e^0 - f_e^+ u_E^0 - (1 - f_e^+) u_P^0 \right]
\end{aligned} \tag{2.36}
$$

where the second line of Eq. (2.36) is commonly known as Majumdar correction.

2.4 Pascau MIM

Pascau [103] proposed an alternative MIM to be used in the solution of transient problems when first or second order temporal discretization schemes are used. As done in Eq. (2.29), the x-velocity equation can be reformulated for a general 1^{st} or 2^{nd} order backward discretization scheme:

$$A_P^t(c_0 u_P - c_1 u_P^{k-1} - c_2 u_P^{k-2}) + A_P^{nt} u_P = A_E u_E + A_W u_W + A_N u_N + A_S u_S + f_{x_P} \quad (2.37)$$

where the term related to the pressure gradient in x-direction, $\Delta y(p_e - p_w)$, has been denoted as f_{x_P} for simplicity, and c_0, c_1, c_2 represent the time discretization coefficients. From Eq. (2.37):

$$\begin{aligned} u_P = {} & \alpha_u \left[\frac{\sum_{NB} A_{NB} u_{NB} + b_P}{A_P^{nt} + c_0 A_P^t} \right] + \alpha_u \left[\frac{f_{x_P}}{A_P^{nt} + c_0 A_P^t} \right] \\ & + \alpha_u \left[\frac{c_1 A_P^t u_P^{k-1}}{A_P^{nt} + c_0 A_P^t} + \frac{c_2 A_P^t u_P^{k-2}}{A_P^{nt} + c_0 A_P^t} \right] \\ & + (1 - \alpha_u) u_P^0 \end{aligned} \quad (2.38)$$

which can also be expressed as:

$$\begin{aligned} u_P = {} & \frac{\alpha_u}{1 + c_0 \beta_P^t} \left[\frac{\sum_{NB} A_{NB} u_{NB} + b_P}{A_P^{nt}} \right]_P + \frac{\alpha_u}{1 + c_0 \beta_P^t} \frac{f_{x_P}}{A_P^{nt}} \\ & + \frac{\alpha_u c_1 \beta_P^t}{1 + c_0 \beta_P^t} u_P^{k-1} + \frac{\alpha_u c_2 \beta_P^t}{1 + c_0 \beta_P^t} u_P^{k-2} \\ & + (1 - \alpha_u) u_P^0 \end{aligned} \quad (2.39)$$

where $\beta_P = A_P^t / A_P^{nt}$. The face velocity is then reconstructed as:

$$\begin{aligned} u_e = {} & \frac{\alpha_u}{1 + c_0 \beta_e^t} \left[\frac{\sum_{NB} A_{NB} u_{NB} + b_P}{A_P^{nt}} \right]_e + \frac{\alpha_u}{1 + c_0 \beta_e^t} \frac{f_{x_e}}{A_e^{nt}} \\ & + \frac{\alpha_u c_1 \beta_e^t}{1 + c_0 \beta_e^t} u_e^{k-1} + \frac{\alpha_u c_2 \beta_e^t}{1 + c_0 \beta_e^t} u_e^{k-2} \\ & + (1 - \alpha_u) u_e^0 \end{aligned} \quad (2.40)$$

where the pressure gradient term is $f_{x_e} = \Delta y(p_E - p_P)$, and β_e^t is the linear interpolation of β_P^t. It can be shown that this MIM is both under-relaxation factor and time step size independent [22,103]. The proof will be omitted here for the sake of brevity, and can be found in [103].

2.5 P-U coupling algorithms

Due to the simplicity of their implementation and the lower peak memory requirements, segregated pressure-velocity coupling algorithms are commonly preferred nowadays over coupled algorithms. In this section four p-U coupling algorithms for incompressible flows are described together with a summary of the iterative procedure followed in each of them.

2.5.1 SIMPLE

The SIMPLE algorithm (Semi-Implicit Method for Pressure Linked Equation) [105] is probably one of the most used nowadays and it sets the basis for the derivation of the rest of models that will be explained later. The basic idea of the SIMPLE algorithm is to update velocity and pressure field in each iteration so that continuity is always satisfied, and velocity equations approach progressively their solution. To do this, a *projection method* is used. For simplicity, SIMPLE equations will be shown for the case of OMIM.

At the beginning m-iteration, known pressure p^{m-1} is used to compute a velocity field u^* by solving the velocity equation in the so called momentum predictor step. For instance, the semi-discretized form of x-velocity is:

$$A_P u_P^* = \sum_{nb} A_{nb} u_{nb}^* + b_P - \Delta y(p_e^{m-1} - p_w^{m-1}) \tag{2.41}$$

where the pressure gradient contribution to the source term has been separated from the rest of source terms, represented in b_P. Typically velocity-equation would be under-relaxed, as explained in the OMIM section:

$$u_P^* = \frac{\alpha_u \left(\sum_{nb} A_{nb} u_{nb}^* + b_P\right)_P}{(A_P)_P} + (1-\alpha_u) u_P^{m-1} - \frac{\alpha_u \Delta y (p_e^{m-1} - p_w^{m-1})_P}{(A_P)_P} \tag{2.42}$$

$$u_P^* = \frac{\alpha_u \left(\sum_{nb} A_{nb} u_{nb}^* + B_P\right)_P}{(A_P)_P} - \frac{\alpha_u \Delta y (p_e^{m-1} - p_w^{m-1})_P}{(A_P)_P} \tag{2.43}$$

Using the OMIM, face velocities will follow an equation similar to 2.25:

$$u_e^* = \frac{\alpha_u \left(\sum_{nb} A_{nb} u_{nb}^* + B_P\right)_e}{(A_P)_e} - \frac{\alpha_u \Delta y (p_E^{m-1} - p_P^{m-1})}{(A_P)_e} \tag{2.44}$$

Since u^* does not satisfy continuity, velocity has to be updated using a correction u' so that $u^m = u^* + u'$. Pressure field is updated by adding a small correction that should vanish as the solution approaches convergence $p^m = p^{m-1} + p'$. Pressure does not have its natural equation in incompressible flows, therefore, an equation for p' is derived so as to guarantee that final velocity field u^m is divergence free. At convergence, an equation similar to 2.45 should yield:

$$u_e^m = \frac{\alpha_u \left(\sum_{nb} A_{nb} u_{nb}^m + B_P\right)_e}{(A_P)_e} - \frac{\alpha_u \Delta y (p_E^m - p_P^m)}{(A_P)_e} \tag{2.45}$$

Subtracting Eq. 2.44 from 2.45:

$$u_e' = \frac{\alpha_u \left(\sum_{nb} A_{nb} u_{nb}'\right)_e}{(A_P)_e} - \frac{\alpha_u \Delta y (p_E' - p_P')}{(A_P)_e} \tag{2.46}$$

In the SIMPLE algorithm the first term in the RHS of Eq. 2.46 is neglected so that velocity correction becomes:

$$u_e' = -\frac{\alpha_u \Delta y (p_E' - p_P')}{(A_P)_e} \tag{2.47}$$

This neglection is thought to be the main reason for the difficulty of SIMPLE algorithm to achieve convergence in some cases. Using Eq. 2.47 face velocity becomes:

$$u_e^m = u_e^* - \frac{\alpha_u \Delta y(p'_E - p'_P)}{(A_P)_e} \tag{2.48}$$

Since u_e^* is now known, following the same procedure for the other velocity component v and for the rest of faces, one may use continuity equation to derive an equation for p'. Eq. 2.1 can be integrated in the cell volume by using Green-Gauss theorem leading to:

$$\sum_e \rho \mathbf{u_e} \mathbf{S_e} = 0 \tag{2.49}$$

$$\Delta y(\rho u_e^m) - \Delta y(\rho u_w^m) + \Delta x(\rho v_n^m) - \Delta x(\rho v_s^m) = 0 \tag{2.50}$$

When Eq. 2.48 and similar equations for u_w^m, v_n^m and v_s^m are introduced in Eq. 2.50, an equation for p' is obtained. Calculation of p' allows us to correct pressure and velocity fields. For pressure correction an under-relaxation factor is usually introduced, so that new pressure becomes $p^m = p^{m-1} + \alpha_P p'$. Cell center velocities are computed by an equation derived from 2.48:

$$u_P^m = u_P^* - \frac{\alpha_u \Delta y(p'_e - p'_w)}{(A_P)_P} \tag{2.51}$$

where p'_e and p'_w are obtained by linear interpolation of cell center values.

A slight variation of the SIMPLE algorithm (hereafter referred to as *simpleP-corr*) can be considered where pressure equation is formulated directly to solve for p and not for p' (referred to as *simple*). In fact, substituting u_e^* from Eq. 2.44 into Eq. 2.48:

$$u_e^m = \frac{\alpha_u \left(\sum_{nb} A_{nb} u_{nb}^* + B_P\right)_e}{(A_P)_e} - \frac{\alpha_u \Delta y(p_E^m - p_P^m)}{(A_P)_e} \tag{2.52}$$

where the first term in the RHS is the part of the velocity field without the pressure term. Introducing Eq. 2.52 into continuity equation yields an equation for p^m. Now pressure will not be directly taken to be the calculated value of p^m but it will be relaxed following $p^m = \alpha_p p^m + (1 - \alpha_p) p^{m-1}$. New cell center velocities would be computed as:

$$u_P^m = \frac{\alpha_u \left(\sum_{nb} A_{nb} u_{nb}^* + B_P\right)}{A_P} - \frac{\alpha_u \Delta y(p_e^m - p_w^m)}{A_P} \tag{2.53}$$

For the simulations shown in this thesis, pressure under-relaxation factor α_p has been set to $1 - \alpha_u$ as recommended by [35].

2.5.2 SIMPLE-C

The SIMPLE-C (SIMPLE-Consistent) algorithm [29] tries to reduce the lack of convergence of the standard SIMPLE algorithm by avoiding the brutal assumption made from Eq. 2.46 to Eq. 2.47. It has shown to accelerate convergence in problems where the pressure-velocity coupling is the main source of deterrent to obtaining a solution. SIMPLE-C procedure is shown next.

From Eq. 2.46 we can derive the following:

$$(A_P)_e u'_e = \alpha_u \left(\sum_{nb} A_{nb} u'_{nb} \right)_e - \alpha_u \Delta y (p'_E - p'_P) \tag{2.54}$$

By subtracting $\sum_{nb} A_{nb} u'_e$ from both sides of the equation we reach:

$$\left((A_P)_e - \sum_{nb} A_{nb} \right) u'_e = \alpha_u \left(\sum_{nb} A_{nb} \left(u'_{nb} - u'_e \right) \right)_e - \alpha_u \Delta y (p'_E - p'_P) \tag{2.55}$$

First term in the RHS of Eq. 2.55 is now neglected, leading to a new relation between velocity and pressure corrections:

$$u'_e = - \frac{\alpha_u \Delta y (p'_E - p'_P)}{(A_P - \sum_{nb} A_{nb})_e} \tag{2.56}$$

Face velocity now becomes:

$$u^m_e = u^*_e - \frac{\alpha_u \Delta y (p'_E - p'_P)}{(A_P - \sum_{nb} A_{nb})_e} \tag{2.57}$$

By using this definition of face velocity in continuity equation we can again obtain a pressure equation for p'. New pressure would be obtained from under-relaxation $p^m = p^{m-1} + \alpha_p p'$ and cell center velocities are computed from an equation derived from 2.57:

$$u^m_P = u^*_P - \frac{\alpha_u \Delta y (p'_e - p'_w)}{(A_P - \sum_{nb} A_{nb})} \tag{2.58}$$

Again, a slightly different version of the explained SIMPLE-C (*simpleCPcorr*) can be obtained by deriving an equation for p^m and not for p' (*simpleC*). Substituting u^*_e from Eq. 2.44 into Eq. 2.57:

$$u^m_e = \frac{\alpha_u \left(\sum_{nb} A_{nb} u^*_{nb} + B_P \right)_e}{(A_P)_e} - \frac{\alpha_u \Delta y (p^{m-1}_E - p^{m-1}_P)}{(A_P)_e} - \frac{\alpha_u \Delta y (p'_E - p'_P)}{(A_P - \sum_{nb} A_{nb})_e} \tag{2.59}$$

which can be transformed into:

$$\begin{aligned} u^m_e = & \frac{\alpha_u \left(\sum_{nb} A_{nb} u^*_{nb} + B_P \right)_e}{(A_P)_e} \\ & - \alpha_u \left(\frac{1}{(A_P)_e} - \frac{1}{(A_P - \sum_{nb} A_{nb})_e} \right) \Delta y (p^{m-1}_E - p^{m-1}_P) \\ & - \frac{\alpha_u \Delta y (p^m_E - p^m_P)}{(A_P - \sum_{nb} A_{nb})_e} \end{aligned} \tag{2.60}$$

First two terms depend only on already calculated variables and can be computed. Inserting u^m_e into continuity equation yields an equation for p^m. Then, as it happened in SIMPLE, pressure is under-relaxed $p^m = \alpha_p p^m + (1 - \alpha_p) p^{m-1}$. New cell velocities are computed as:

$$
\begin{aligned}
u_P^m = & \frac{\alpha_u \left(\sum_{nb} A_{nb} u_{nb}^* + B_P\right)}{A_P} \\
& - \alpha_u \left(\frac{1}{A_P} - \frac{1}{A_P - \sum_{nb} A_{nb}}\right) \Delta y (p_e^{m-1} - p_w^{m-1}) \\
& - \frac{\alpha_u \Delta y (p_e^m - p_w^m)}{A_P - \sum_{nb} A_{nb}}
\end{aligned}
\tag{2.61}
$$

2.5.3 PISO

PISO (Pressure Implicit with Splitting of Operators) [63] is another derivation of the standard SIMPLE algorithm. Once velocity and pressure have been corrected with the standard simple procedure $u^{**} = u^* + u'$, $p^{**} = p^{m-1} + p'$, new corrections ($p^m = p^{**} + p''$, $u^m = u^{**} + u''$) may be considered to avoid neglecting the first term in RHS of Eq. 2.46 which now becomes:

$$
u_e'' = \frac{\alpha_u \left(\sum_{nb} A_{nb} u_{nb}'\right)_e}{(A_P)_e} - \frac{\alpha_u \Delta y (p_E'' - p_P'')}{(A_P)_e} \tag{2.62}
$$

Since u' has already been calculated, first term of the RHS can now be computed. Following ($u^m = u^{**} + u''$), face velocity is now:

$$
u_e^m = u_e^{**} + \frac{\alpha_u \left(\sum_{nb} A_{nb} u_{nb}'\right)_e}{(A_P)_e} - \frac{\alpha_u \Delta y (p_E'' - p_P'')}{(A_P)_e} \tag{2.63}
$$

Using again continuity equation (note that u_e^{**} already satisfies continuity), Eq. 2.63 yields an equation for p''. Cell pressure field is then updated to $p^m = p^{**} + p''$ and cell velocities are calculated as:

$$
u_P^m = u_P^{**} + \frac{\alpha_u \left(\sum_{nb} A_{nb} u_{nb}'\right)}{A_P} - \frac{\alpha_u \Delta y (p_e'' - p_w'')}{A_P} \tag{2.64}
$$

As explained before for the rest of algorithms standard PISO (known hereafter as *pisoPcorr*) can be modified so that pressure equation solves directly for p^m (*piso*). After the first corrector step, u^{**} and p^{**} are linked by an equation similar to Eq. 2.52:

$$
u_e^{**} = \frac{\alpha_u \left(\sum_{nb} A_{nb} u_{nb}^* + B_P\right)_e}{(A_P)_e} - \frac{\alpha_u \Delta y (p_E^{**} - p_P^{**})}{(A_P)_e} \tag{2.65}
$$

Substitution of u_e^{**} in 2.63 yields:

$$
u_e^m = \frac{\alpha_u \left(\sum_{nb} A_{nb} (u_{nb}^* + u_{nb}') + B_P\right)_e}{(A_P)_e} - \frac{\alpha_u \Delta y (p_E^{**} - p_P^{**})}{(A_P)_e} - \frac{\alpha_u \Delta y (p_e'' - p_w'')}{(A_P)_P} \tag{2.66}
$$

which can also be expressed as:

$$
u_e^m = \frac{\alpha_u \left(\sum_{nb} A_{nb} u_{nb}^{**} + B_P\right)_e}{(A_P)_e} - \frac{\alpha_u \Delta y (p_E^m - p_P^m)}{(A_P)_e} \tag{2.67}
$$

where again the first term in the RHS is the part of the velocity field u^{**} not containing the pressure term. Continuity yields now an equation for p^m, which should not be under-relaxed, and cell velocity can be then updated to:

$$u_P^m = \frac{\alpha_u \left(\sum_{nb} A_{nb} u_{nb}^{**} + B_P\right)}{A_P} - \frac{\alpha_u \Delta y (p_e^m - p_w^m)}{A_P} \tag{2.68}$$

2.5.4 SIMPLE-R

The SIMPLE-R (SIMPLE-Revised) algorithm [104] is another slight variation of SIMPLE-like methods. For some particular flow problems, neglecting the $\sum_{nb} A_{nb} u'_{nb}$ terms in pressure correction equation produces too large pressure corrections (which then require under-relaxation). It could also occur that initial guess of pressure field is much worse than initial guess of velocity field. Using standard SIMPLE algorithm, the good initial velocity field will be destroyed in the momentum predictor step by using a wrong pressure field. To deal with these situations, it is convenient to set a separate equation for pressure computation, which will be solved at the beginning of the iteration, and to construct then a pressure correction equation that will only be used to correct the velocity field. Following these steps, a face velocity could be computed by using u^{m-1}:

$$u_e = \frac{\alpha_u \left(\sum_{nb} A_{nb} u_{nb}^{m-1} + B_P\right)_e}{(A_P)_e} - \frac{\alpha_u \Delta y (p_E^m - p_P^m)}{(A_P)_e} \tag{2.69}$$

If this face velocity is used in the continuity equation we obtain an equation for p^m. It is clear that no approximation has been made here and no term has been neglected. Calculated p^m is now used to construct and solve velocity equations:

$$(A_P/\alpha_u)\, u_P^* = \sum_{nb} A_{nb} u_{nb}^* + B_P - \Delta y (p_e^m - p_w^m) \tag{2.70}$$

Once u^* is known, Eqs. 2.48 and 2.50 can be used to derive a pressure correction equation for p', following the exact same steps of SIMPLE algorithm. However, in this case, pressure is not corrected (p' is not added to current estimation of p), but it is only used to correct velocity field, following Eq. 2.51

For each of the solvers presented here, the introduced names correspond to the OMIM (such as *simpleC*), while they will be referred to as "solver name" + *Majumdar* when MMIM is used (such as *simpleCMajumdar*).

2.6 Numerical experiments

In order to test the effect of the MIM used in the simulations, three simple test cases have been selected. First of all, the consistency of Pascau correction is tested in a transient case where the analytical solution can be found, the well-known Taylor-Green vortex. Later on, accuracy of the several MIM is tested on a lid-driven laminar cavity. Extracted conclusions are used to perform a more systematic study in the same lid driven cavity case and finally to a more complex case, the study of the flow around a NACA 0012 airfoil profile is chosen, to check

the performance of some of the MIM when the convergence of the solvers might be limited by that of the turbulence quantities equations. As pointed out by B. Yu et al. [143], face velocities are used three times in the overall procedure of the solution. They are used to determine the coefficients in the discretization of the equation, they are used to derive the pressure equation and they are considered in the mass residual coefficient if it is not dropped. According to their suggestion, in all simulations shown here, the same procedure to calculate u_e will be used for all the three times it is required.

For each of the cases analyzed here, convergence of the solvers is checked by calculating equation residuals for continuity and all velocity components equations. Convergence is assumed to be achieved when all normalized residuals go below a given threshold. In OpenFOAM® normalized residual for a given variable ϕ following Eq. 2.6 are calculated as follows:

$$res_\phi = \frac{\sum_{i=1}^{N} | \left[A_P \phi_P - \sum_{nb} A_{nb} \phi_{nb} - b_P \right]_i |}{normFactor_\phi} \tag{2.71}$$

where the first summation is performed in all the cells and normalization factor is calculated as:

$$normFactor_\phi = \sum_{i=1}^{N} \left| \left[A_P \phi_P - \sum_{nb} A_{nb} \phi_{nb} \right]_i - \left[A_P - \sum_{nb} A_{nb} \right]_i \phi_{ref} \right| + \tag{2.72}$$

$$+ \sum_{i=1}^{N} \left| \left[b_P \right]_i - \left[A_P - \sum_{nb} A_{nb} \right]_i \phi_{ref} \right| \tag{2.73}$$

where ϕ_{ref} is a reference value for the field, calculated as mean value over the total number of cells ($\phi_{ref} = \sum_i \left[\phi_P \right]_i / N$). At convergence normalization factor becomes $normFactor_\phi = 2 \sum_{i=1}^{N} |\left[b_P \right]_i|$. Continuity error is computed as a volume weighted average of the mass residual A_b calculated as explained in Eq. 2.9.

Simple linear matrix solvers have been used for all cases. Pressure and pressure correction equations are solved with a Preconditioned Conjugate Gradient solver (PCG), using diagonal incomplete-Cholesky preconditioner (DIC), while velocity and turbulence quantities are solved using Preconditioned Bi-conjugate Gradient solver (PBiCG) and Diagonal Incomplete LU preconditioner (DILU). Each outer iteration, matrices are solved until an absolute residual of 10^{-12} or a relative residual (compared to initial residual in present iteration) of 10^{-3} is achieved.

2.6.1 Taylor-Green vortex

In this section, the accuracy of the Pascau MIM is tested by solving the decay of two-dimensional vortices, the so-called Taylor-Green problem. For this case, Navier-Stokes equations can be analytically solved. Using a non-dimensional formulation (considering unit viscosity $\nu = 1$ and unit maximum initial velocity $u_{\max}|_0 = v_{\max}|_0 = 1$) the solution is given by:

$$
\begin{aligned}
u(x,y,t) &= -e^{-2t}\cos(x)\sin(y) \\
v(x,y,t) &- e^{-2t}\sin(x)\cos(y) \\
p(x,y,t) &= \frac{-e^{-4t}\cos(2x)\sin(2y)}{4}
\end{aligned}
\tag{2.74}
$$

The domain extension considered for this case is a square (of side $L = 2\pi$) domain $[0 - 2\pi] \times [0 - 2\pi]$ and periodic conditions are applied at boundaries. Fields of the analytical solution (used for initialization) are shown in Fig. 2.2.

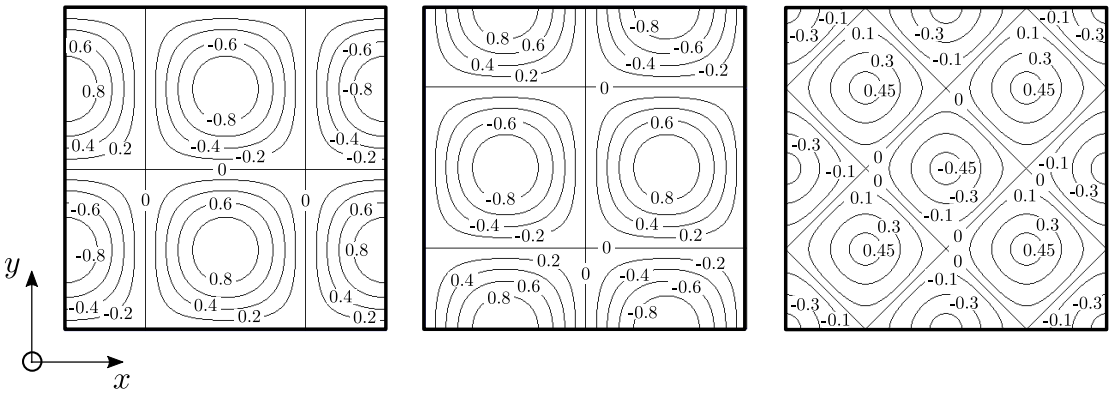


Figure 2.2: Taylor-Green vortices analytical solution of u (left), v (center) and p (right) at $t = 0$

A simple orthogonal uniform mesh of 80×80 cells has been considered for this case. MIM were first compared by analyzing the profiles of $u(x), v(x), p(x)$ at $y = \pi/2$ at different times. As an example Fig. 2.3 shows the comparison between the two MIM when $\Delta t = 0.005 t_D$, being $t_D = L/u_{\max}|_0$, when a second order backward time discretization scheme is used.

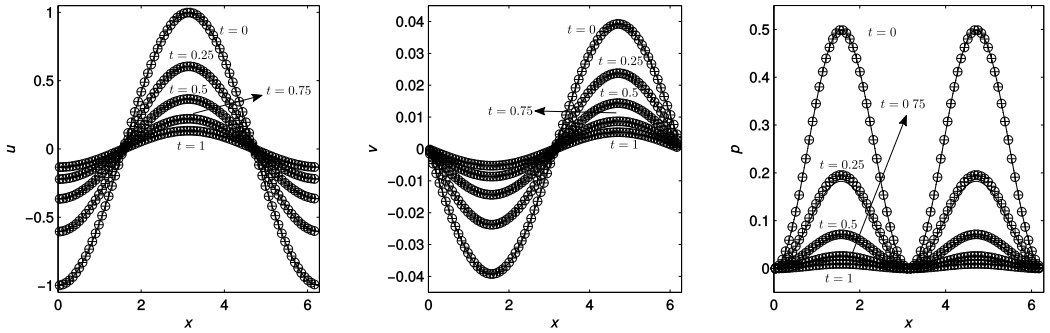


Figure 2.3: Velocity and pressure profiles at $y = \pi/2$ for several times. Results are shown for OMIM ($\circ\circ\circ$); Pascau MIM ($+++$) in comparison with analytical solution (—)

In this case a transient SIMPLE algorithm is used: continuity and velocity equations are iteratively solved as described in section 2.5.1 and time advances when equations are converged achieving an absolute residual lower than 10^{-12}. This is a too small value for a realistic case but it is achievable in such a simple academic problem. Under-relaxation factors for this case are $\alpha_u = 0.8$ and $\alpha_p =$

0.2. For this example and the rest of cases in this section, second order Central Differencing Scheme (CDS) has been used for the discretization of advection terms.

Given the very small differences between the solutions, the actual comparison is evaluated by analyzing the differences between MIM in the prediction of peak x-velocity at $t = 0.5$. Two p-U coupling algorithms are considered: PIMPLE (with $\alpha_u = 0.8$ and $\alpha_p = 0.2$) and PISO (no under-relaxation). Besides, results are shown for two time discretization schemes: first and second order Euler backward discretization. The results as a function of the time step size are shown in Fig. 2.4

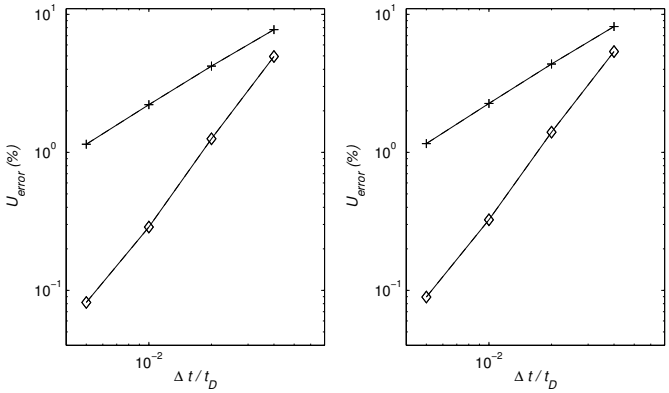


Figure 2.4: Error in the determination of peak x-velocity at $t = 0.5$ for first order $(+ + +)$ and second order $(\diamondsuit\diamondsuit\diamondsuit)$ time discretization schemes, using OMIM (—) or Pascau MIM $(- - -)$, and PIMPLE (left) or PISO (right) p-U coupling algorithms

The comparison makes evident that the differences are minimum for any Δt. When using PISO algorithm no under-relaxation is used so that the differences are only due to the time step dependence. When PIMPLE algorithm is used both α_u and Δt dependencies could cause results to be different, but it has been show that for this case the differences are not meaningful at all. While no significant differences were obtained when Pascau correction was used in this case, the implementation of solvers is much more complex (it requires the splitting of velocity equation and the storage of many additional fields) and the extension to the moving mesh case of the gear pump becomes critical. Therefore, and considering that the time step dependency is much lower that the under-relaxation factor dependency (as already shown by many authors [22, 103, 143]), only Majumdar correction for under-relaxation factor dependency will be analyzed in the rest of cases.

2.6.2 Lid-driven cavity

The second study case is a two-dimensional laminar lid-driven cavity (as shown in the schematic in Fig. 2.5) for $Re = 1000$ and $Re = 5000$. This case was chosen because it represents one of the simplest cases available. The simple geometry allows us to use a perfectly orthogonal and uniform mesh and the boundary conditions can be clearly specified. Therefore stability, accuracy and robustness depends mainly on the MIM used for its solution. When the higher Reynolds number

is used, even if the flow remains laminar, instabilities are more likely to appear in the numerical solvers and this provides also important information about the MIM used.

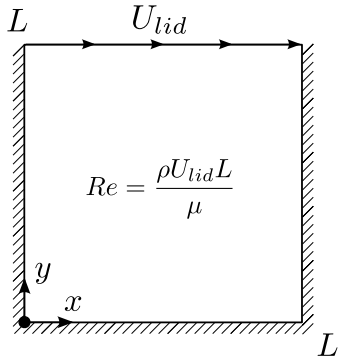


Figure 2.5: Cavity case geometry

Vertical velocity profiles at the center-line are considered for comparison with respect to the benchmark solutions from Ghia *et al.* [47]. Different mesh types have been studied, as shown in Fig. 2.6. Information regarding number of cells and maximum non-orthogonality for those four meshes is shown in Table 2.1.

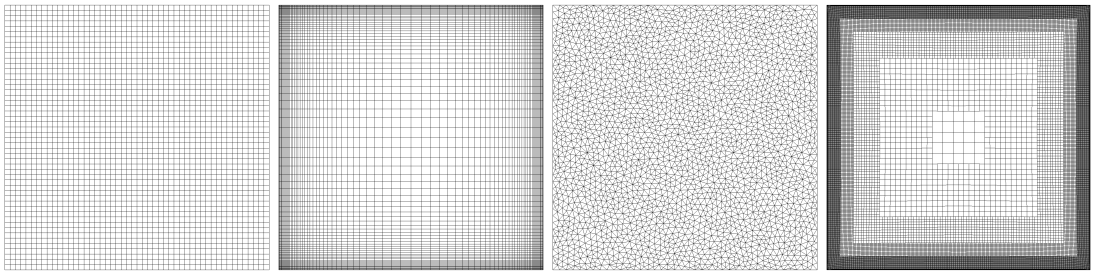


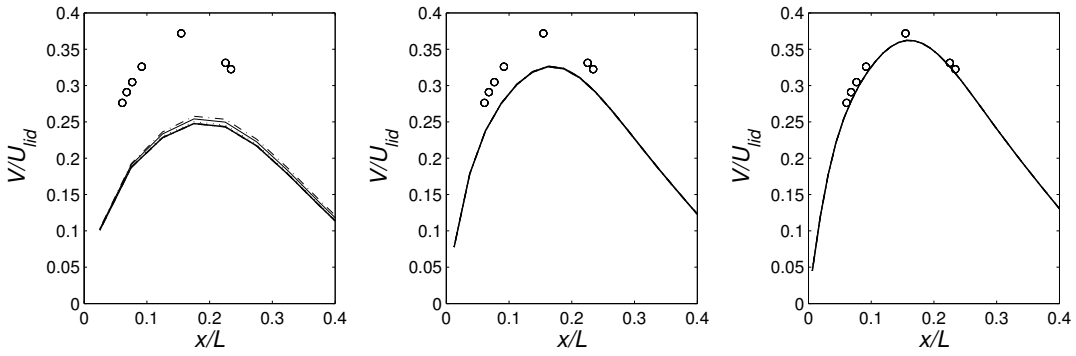
Figure 2.6: Mesh types tested for cavity case; from left to right: uniform orthogonal, non-uniform orthogonal, triangles and unstructured hex-dominant

Table 2.1: Mesh characteristics for cavity case

Mesh	Number of cells	Maximum non-orthogonality
Uniform orthogonal	2500	0
Non-uniform orthogonal	6400	0
Triangles	4862	37.6
Hex-dominant	48825	47.6

First, a comparison of OMIM and Majumdar MIM using the SIMPLE algorithm is performed in a uniform mesh with increasing resolution. Results are shown in Fig. 2.7.

Fig. 2.7a shows that if a 20×20 mesh is used, there is a 4% difference in peak velocity between $\alpha_u = 0.2$ and 0.8. Dependency of the solution quickly reduces with mesh refinement, while Majumdar correction completely eliminates



(a) 20×20 cells (b) 40×40 cells (c) 80×80 cells

Figure 2.7: Vertical velocity profiles along horizontal center-line for OMIM (thin lines) and Majumdar correction (thick lines) for several under-relaxation factors: $\alpha_u = 0.2$ ($-\cdot-$); $\alpha_u = 0.5$ (—); $\alpha_u = 0.8$ ($\cdots$). Results are compared to benchmark solution ($\circ\ \circ\ \circ$). Mesh refinement increases from left to right: 20×20, 40×40, 80×80

the dependency. As shown in Fig. 2.7, for a resolution higher than 40×40 the difference in the solution is insignificant.

If the error in the determination of vertical velocity at point (0.1548, 0.5) (peak velocity location in the center-line for reference data) is represented with respect to the number of cells for the uniform mesh case (Fig. 2.8), it becomes clear that α_u dependency of OMIM is almost negligible for a fine enough mesh, while Majumdar solution is always independent of α_u.

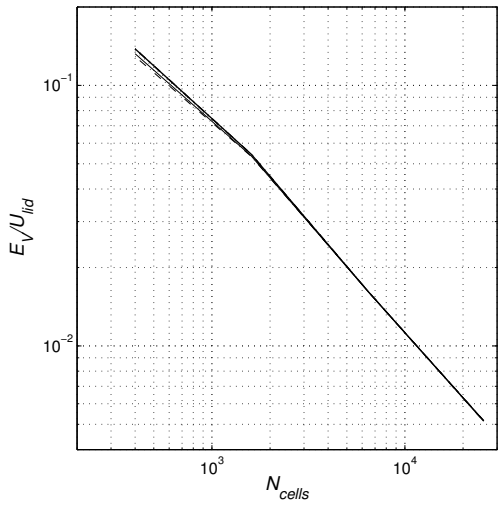


Figure 2.8: Error in the determination of vertical velocity at (0.1548, 0.5) for OMIM (thin lines) and Majumdar correction (thick lines) for several under-relaxation factors: $\alpha_u = 0.2$ ($- \cdot -$); $\alpha_u = 0.5$ (—); $\alpha_u = 0.8$ ($\cdots$)

Given these results the study is now extended to the different pressure-velocity coupling algorithms presented in section 2.5 and to the different meshes considered in Fig. 2.6. It was first verified that for a given mesh, the solution obtained by all the pressure-velocity coupling algorithms was the same for a fixed value of α_u with OMIM or for any value of α_u if the Majumdar correction was applied.

For the case of $Re = 1000$ convergence speed of each combination algorithm - momentum interpolation method, has been checked for two possible values of α_u, 0.2 and 0.8, and for the four meshes described in Table 2.1. In particular, the required number of iterations to achieve a residual lower than 10^{-6} for both velocity components and continuity is represented in Fig. 2.9.

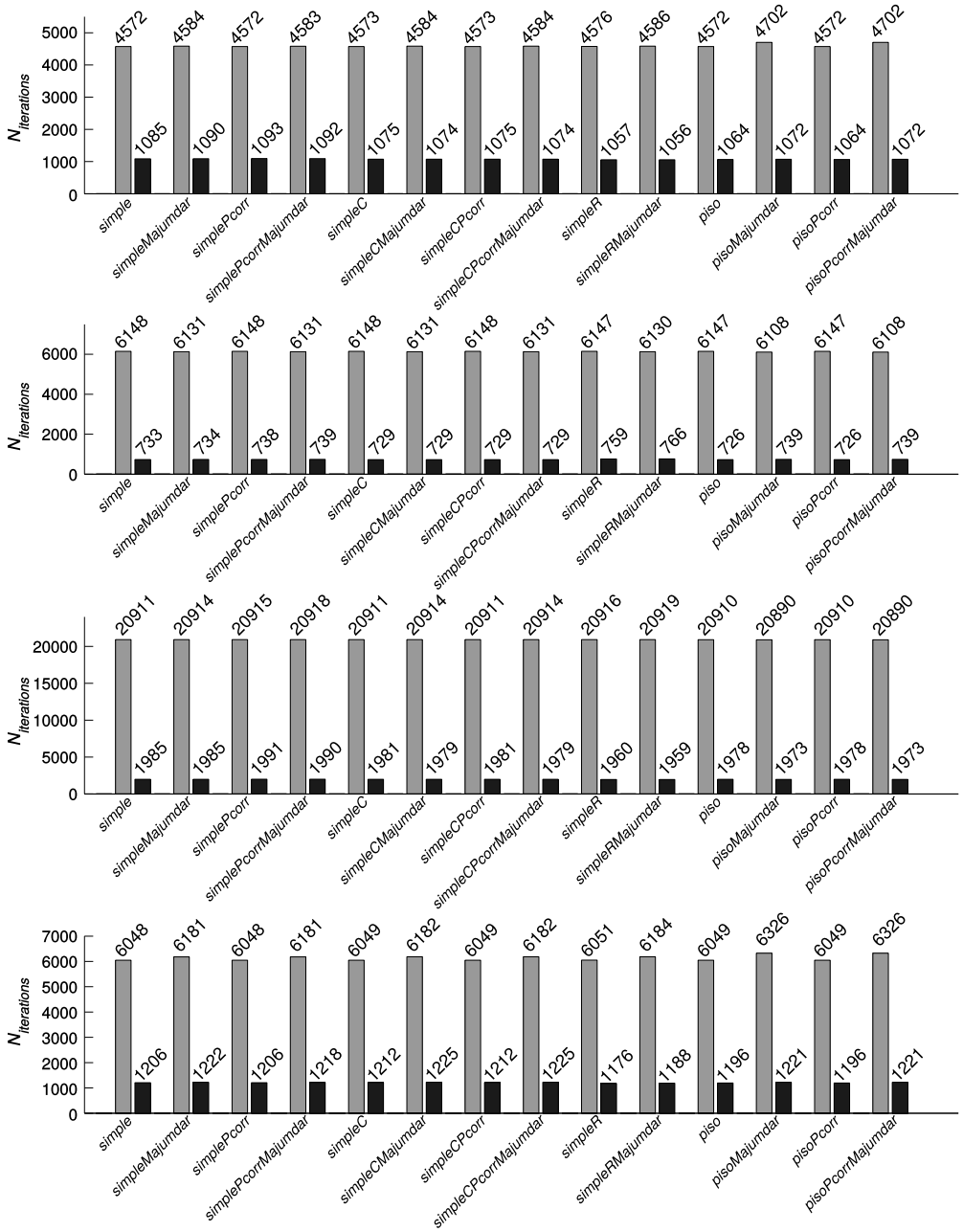


Figure 2.9: Required number of iterations to reduce all residuals below 10^{-6} when α_u is fixed to $\alpha_u = 0.2$ (grey) and $\alpha_u = 0.8$ (black). Results are reported for uniform (first), non-uniform orthogonal (second), hex-dominant (third) and triangles (fourth) meshes

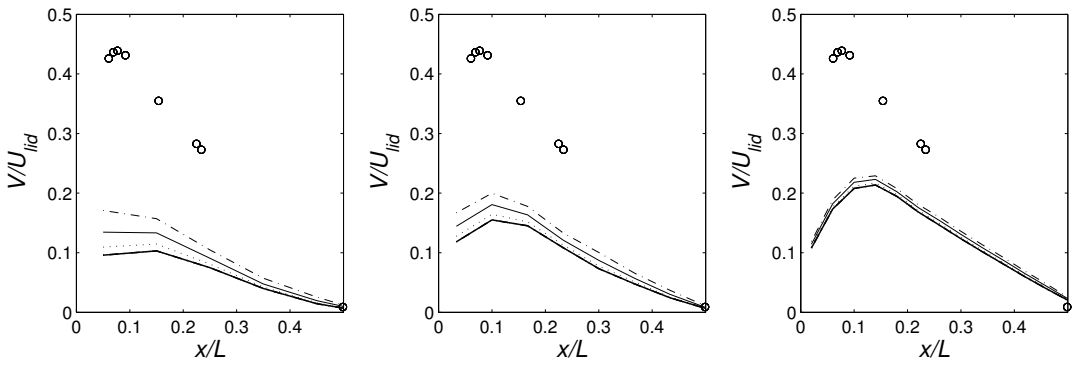
Analysis of Fig. 2.9 shows that no significant difference in terms of speed of convergence is observed for this case and for the algorithms considered, and

that Majumdar correction does not influence the convergence speed. For a given mesh, number of iterations for all solvers is approximately the same, as far as the same velocity under-relaxation is considered. Certainly, all cases with higher under-relaxation factor ($\alpha_u = 0.8$) converge faster than those with a lower under-relaxation factor ($\alpha_u = 0.2$). As a reminder, all PISO-derived and SIMPLER-derived algorithms solve twice for a pressure correction equation, and no pressure under-relaxation is performed, while for the rest of the algorithms pressure under-relaxation is set to $\alpha_p = 1 - \alpha_u$ and only one pressure correction equation is solved.

When considering the pressure correction equation, solving for pressure correction p' instead of a direct solution for p^m slightly increases the number of iterations until convergence. For a given solver, the influence of applying Majumdar correction does not show a clear trend. For instance, when considering the non-uniform mesh, Majumdar correction accelerates convergence for $\alpha_u = 0.2$ for almost all solvers but it has a negative influence when $\alpha_u = 0.8$. Considering the hex-dominant mesh, the influence is the opposite, slightly positive for $\alpha_u = 0.8$ and slightly negative for $\alpha_u = 0.2$.

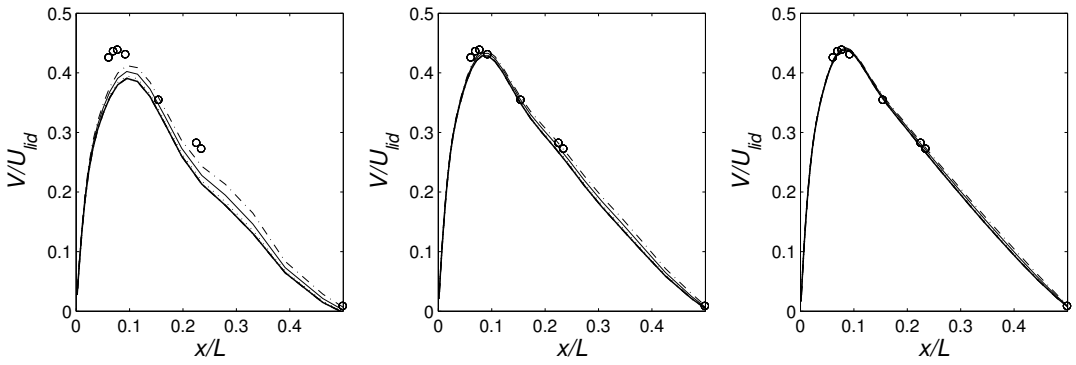
When comparing solvers, SIMPLEC-like solvers tend to converge slightly faster but again no clear trend is observed.

Taking into account the small differences found for $Re = 1000$, the study is now extended to the case of $Re = 5000$. Firstly, mesh convergence of standard OMIM and Majumdar correction is tested, using in this case either a uniform mesh or a non-uniform orthogonal mesh. For this higher Reynolds case differences are more evident. For the uniform orthogonal mesh, only coarse meshes allow solvers to converge. Fig. 2.10 shows how solution changes with α_u for fairly coarse uniform meshes, while Fig. 2.11 shows the profiles with more refined non-uniform meshes. Again, Majumdar correction completely removes the dependency as it can be seen in Fig. 2.12 where errors in the determination of vertical velocity at (0.06072, 0.5), corresponding to peak velocity in the solution reference, are shown.



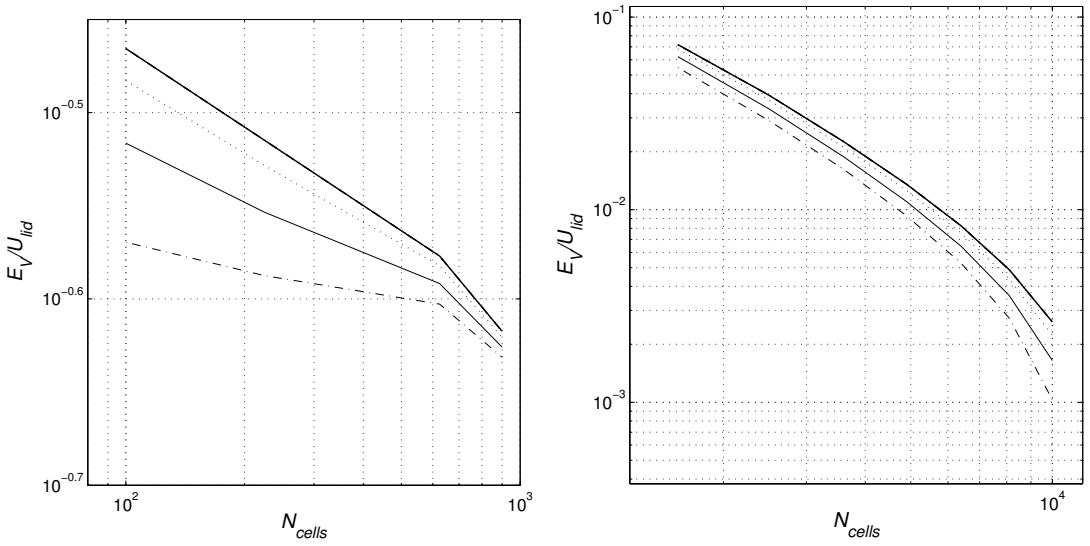
(a) 10×10 cells (b) 15×15 cells (c) 25×25 cells

Figure 2.10: Vertical velocity profiles along horizontal center-line obtained with uniform orthogonal meshes for OMIM (thin lines) and Majumdar correction (thick lines) for several under-relaxation factors: $\alpha_u = 0.2$ ($- \cdot -$); $\alpha_u = 0.5$ (—); $\alpha_u = 0.8$ ($\cdots$). Results are compared to benchmark solution ($\circ\circ\circ$). Mesh refinement increases from left to right: 10×10, 15×15, 25×25



(a) 40×40 cells (b) 60×60 cells (c) 80×80 cells

Figure 2.11: Vertical velocity profiles along horizontal center-line obtained with non-uniform orthogonal meshes for OMIM (thin lines) and Majumdar correction (thick lines) for several under-relaxation factors: $\alpha_u = 0.2$ $(- \cdot -)$; $\alpha_u = 0.5$ $(—)$; $\alpha_u = 0.8$ $(\cdots)$. Results are compared to benchmark solution $(\circ \circ \circ)$. Mesh refinement increases from left to right: 40×40, 60×60, 80×80



(a) Uniform mesh (b) Non-uniform mesh

Figure 2.12: Error in the determination of vertical velocity at (0.06070, 0.5) for OMIM (thin lines) and Majumdar correction (thick lines) for several under-relaxation factors: $\alpha_u = 0.2$ $(- \cdot -)$; $\alpha_u = 0.5$ $(—)$; $\alpha_u = 0.8$ $(\cdots)$. Results are shown for uniform mesh (left) and non-uniform orthogonal mesh (right)

Following the procedure of the previous case, convergence speed is evaluated for the four meshes described in Table 2.1. As the Reynolds number increases, convergence becomes more difficult and, depending on the value of under-relaxation factor, it might not be achievable. It is also observed that, if convergence is achieved, the solution is exactly the same as long as the same α_u is used for any of the algorithms, or Majumdar correction is applied. Number of iterations required

to achieve convergence is represented in Fig. 2.13, where (-1) refers to cases that do not converge.

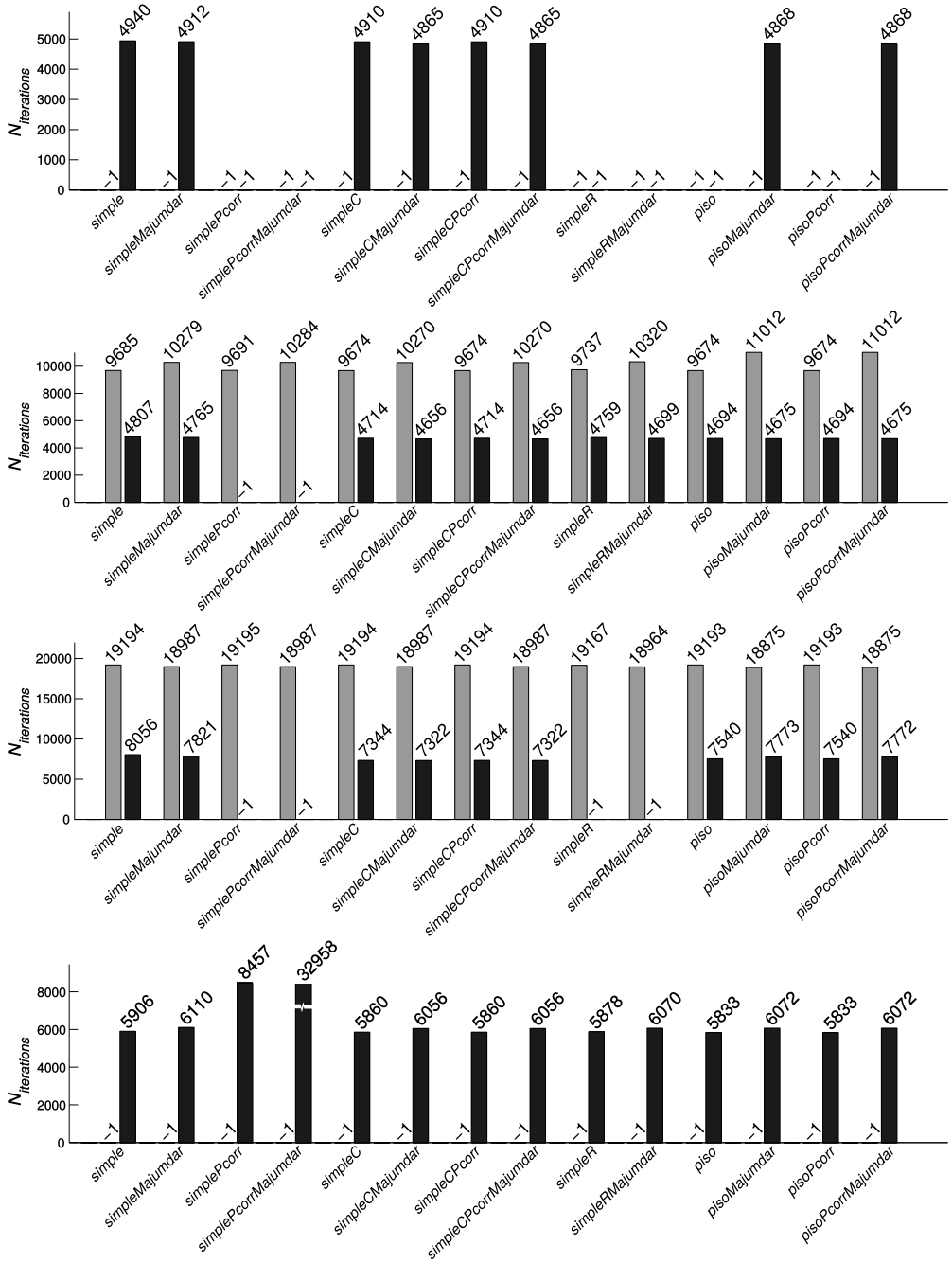


Figure 2.13: Required number of iterations to reduce all residuals below 10^{-6} when α_u is fixed to $\alpha_u = 0.2$ (grey) and $\alpha_u = 0.8$ (black). Results are reported for uniform (first), non-uniform orthogonal (second), hex-dominant (third) and triangles (fourth) meshes

With the exception of few cases (*simplePcorr-simplePcorrMajumdar* in non-uniform orthogonal and hex-dominant meshes, and *simpleR-simpleRMajumdar* in hex-dominant mesh), a large velocity under-relaxation factor $\alpha_u = 0.8$ helps the convergence of the cases (which might not be achievable for $\alpha_u = 0.2$ as in the

triangles mesh), or increases convergence speed (as in the non-uniform orthogonal or the hex-dominant mesh). When considering Majumdar correction applied to SIMPLE or SIMPLE-C, the general trend is that it slightly improves convergence speed, even if the effect can be opposite for some combinations of mesh-α_u. In the selection of algorithm, SIMPLE-C is the most promising, with or without Majumdar correction (considering than only one pressure correction equation is solved for iteration).

Summing up, when fully converged results are achieved, Majumdar correction eliminates α_u-dependency, without changing convergence speed significantly. Dependency of OMIM on α_u is quickly reduced as mesh refinement increases. When several algorithms are considered, SIMPLE-C seems the most promising alternative to the standard SIMPLE algorithm. Under these considerations, effects of momentum interpolation and p-U coupling algorithm will now be tested on a more complex geometry where coupling between velocity field and turbulence variables becomes important.

2.6.3 NACA 0012 airfoil profile

The turbulent flow ($Re \approx 6 \cdot 10^6$ based on the chord length) around a NACA 0012 airfoil profile is studied under incompressible conditions ($M = 0.15$). As proposed in ERCOFTAC references, a structured C-grid of 14336 cells (225×65 points with 129 points on airfoil surface) is employed, giving an approximate average $y^+ \approx 1$ over the airfoil profile. Fig. 2.14 shows a representation of the mesh, which satisfies the characteristics shown in Table 2.2.

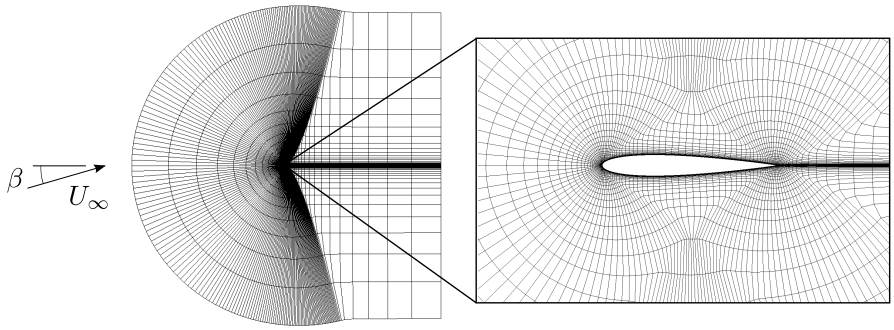


Figure 2.14: Representation of the mesh used in NACA 0012 case

Table 2.2: Mesh characteristics for cavity case

Number of cells	14336
Max. non-orthogonality	77.2
Avg. non-orthogonality	10.2
Max. skewness	0.62

A free-stream turbulence intensity of 0.052% together with a free-stream turbulent viscosity $\nu_t = 0.009\nu$ are used to set the turbulence boundary conditions. First order upwind spatial discretization is used for divergence terms of turbulence vari-

ables and LUST (Linear Upwind Stabilized Transport) [141] is used for the convective term in velocity equations. While velocity and pressure under-relaxation are set to different values during this study, turbulence variables under-relaxation factor has been fixed to 0.8.

Typical flow features are used for the evaluation of numerical solutions. In particular, experimental results for pressure distribution [52], lift and drag coefficients [72] on the upper wall for different angles of attack (β) are used as reference. The mesh used in this study is chosen after a mesh convergence analysis. Standard SIMPLE (*simple*) algorithm with OMIM and $\alpha_u = 0.7$ is used with four increasingly refined meshes; coarse (3584 cells), medium (14336 cells), fine (57344 cells) and very fine (229376 cells) leading to the drag coefficient values and pressure coefficient profiles shown in Figs. 2.16 and 2.15 for $\beta = 0$.

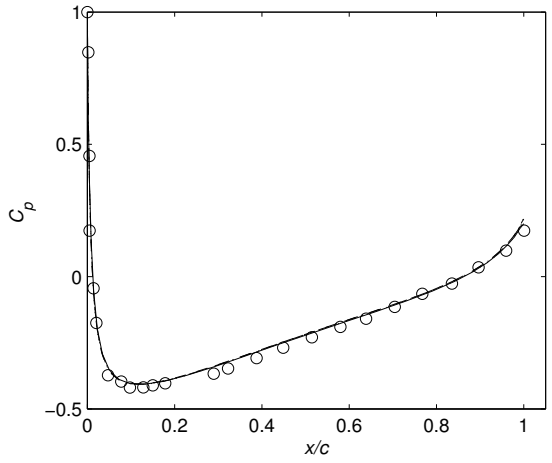


Figure 2.15: Mesh convergence of pressure coefficient on upper wall: coarse(—); medium (− − −); fine(− · −); very fine (· · ·). Results are compared to experimental results (∘ ∘ ∘)

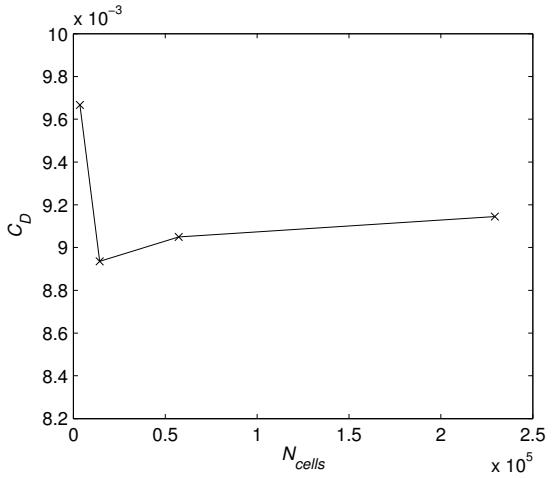
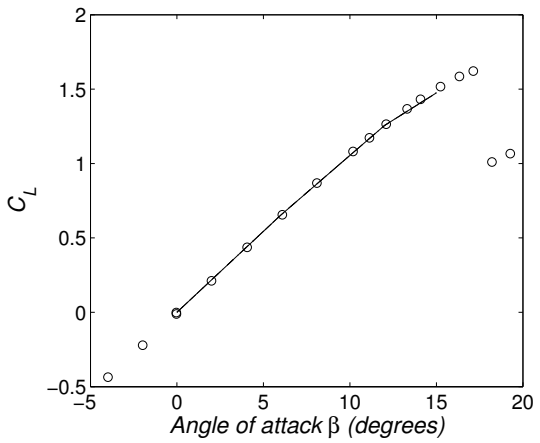


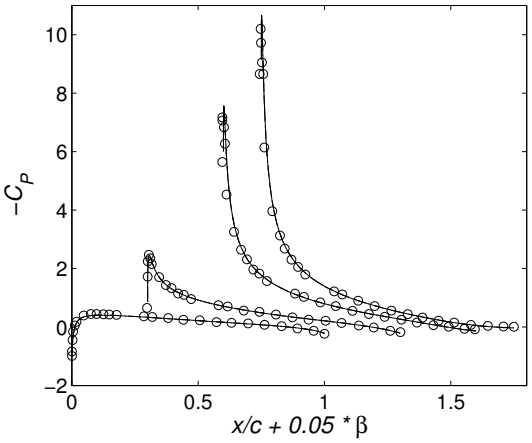
Figure 2.16: Mesh convergence of drag coefficient

According to this, medium mesh is selected for the study, since it is the coarsest mesh for which mesh convergence is acceptable.

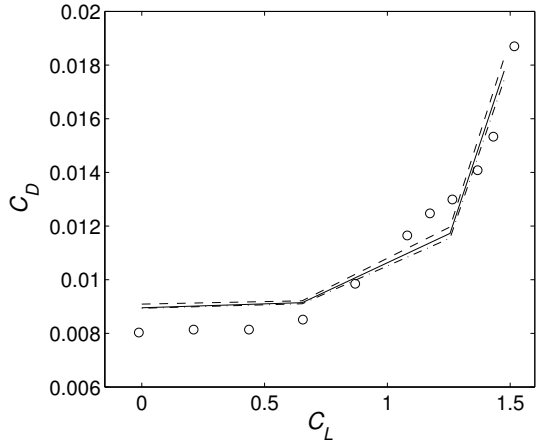
As a starting point for momentum interpolation comparison, standard SIMPLE algorithm with OMIM or MMIM is used to compute C_P, C_D and C_L for a range of angles of attack β: 0, 6, 12 and 15 degrees and for different values of α_u: 0.2, 0.5 and 0.8. Fig. 2.17 shows the results obtained with OMIM while Fig. 2.18 shows the same results when Majumdar correction is considered in the solver. For each case, simulation was run until a residual lower than 10^{-10} was achieved for all variables.



(a) Lift coefficient (C_L)



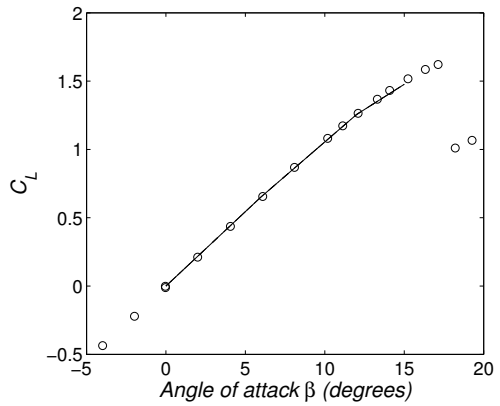
(b) Pressure coefficient (C_P)



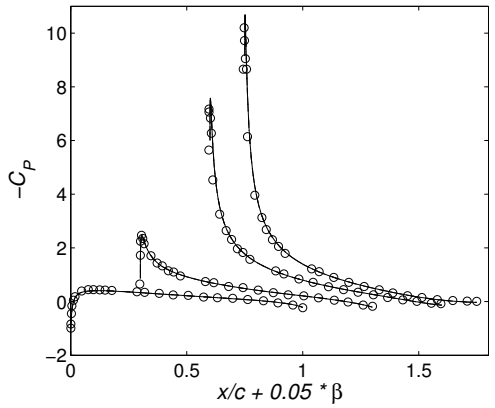
(c) Drag coefficient (C_D)

Figure 2.17: C_L, C_P and C_D coefficients SIMPLE algorithm with OMIM for several angles of attack β: 0, 6, 12 and 15 degrees. Results are shown for $\alpha_u = 0.2$ ($- - -$); $\alpha_u = 0.5$ (—); $\alpha_u = 0.8$ ($- \cdot -$)

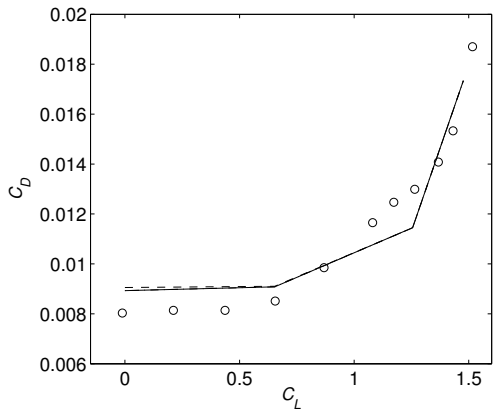
While dependency of C_L (Fig. 2.17a) and C_P (Fig. 2.17b) on α_u is negligible, showing that calculated pressure field barely changes with α_u, results for C_D (Fig. 2.17c) do change more significantly. It is observed that small variations in velocity field produce larger variations in velocity gradients and therefore drag force computation. This barely affects C_L, for which pressure force is the main contribution, but it does affect C_D.



(a) Lift coefficient (C_L)



(b) Pressure coefficient (C_P)



(c) Drag coefficient (C_D)

Figure 2.18: C_L, C_P and C_D coefficients SIMPLE algorithm with Majumdar correction for several angles of attack β: 0, 6, 12 and 15 degrees. Results are shown for $\alpha_u = 0.2$ ($- - -$); $\alpha_u = 0.5$ (—); $\alpha_u = 0.8$ ($- \cdot -$)

When considering the same cases with SIMPLE algorithm and MMIM, dependence of results on α_u is almost completely eliminated (see Fig. 2.18) for the level of convergence achieved in the simulations. In order to better evaluate how much of the dependency is eliminated, a single case with $\beta = 12$ degrees (where OMIM shows the highest dependency) is considered. Calculated C_L and C_D for SIMPLE algorithm with and without Majumdar correction are shown in Table 2.3 and Table 2.4.

While for a standard convergence Majumdar correction only partially eliminates the α_u-dependence, when convergence is good enough, Majumdar correction reduces the relative change in predicted value by two orders of magnitude. The lack of convergence of a case due to turbulence could therefore limit the effect of canceling under-relaxation factor dependency of the Majumdar correction.

We proceed now to evaluate how Majumdar correction affects the convergence

Table 2.3: Dependency of C_D, C_L on α_u for a convergence of $res < 10^{-7}$

Coefficient		OMIM	Majumdar
C_L	$\alpha_u = 0.2$	1.2551915	1.2557273
	$\alpha_u = 0.8$	1.2552688	1.2554411
	Rel. Change	0.006 %	0.022%
C_D	$\alpha_u = 0.2$	0.011967419	0.011588629
	$\alpha_u = 0.8$	0.011548746	0.011450845
	Rel. Change	3.498 %	1.189%

Table 2.4: Dependency of C_D, C_L on α_u for a convergence of $res < 10^{-12}$

Coefficient		OMIM	Majumdar
C_L	$\alpha_u = 0.2$	1.2543421	1.2554539
	$\alpha_u = 0.8$	1.2552675	1.2554399
	Rel. Change	0.133 %	0.001%
C_D	$\alpha_u = 0.2$	0.011948155	0.011446795
	$\alpha_u = 0.8$	0.011549183	0.011451306
	Rel. Change	3.339 %	0.039%

speed of each of the pressure-velocity coupling algorithms described in Section 2.5. For the case of $\beta = 12$ degrees, for each algorithm with or without Majumdar correction, the number of iterations required to achieve residuals lower than 10^{-6} and 10^{-7} is computed for two values of α_u: 0.2 and 0.8 and represented in Fig. 2.19.

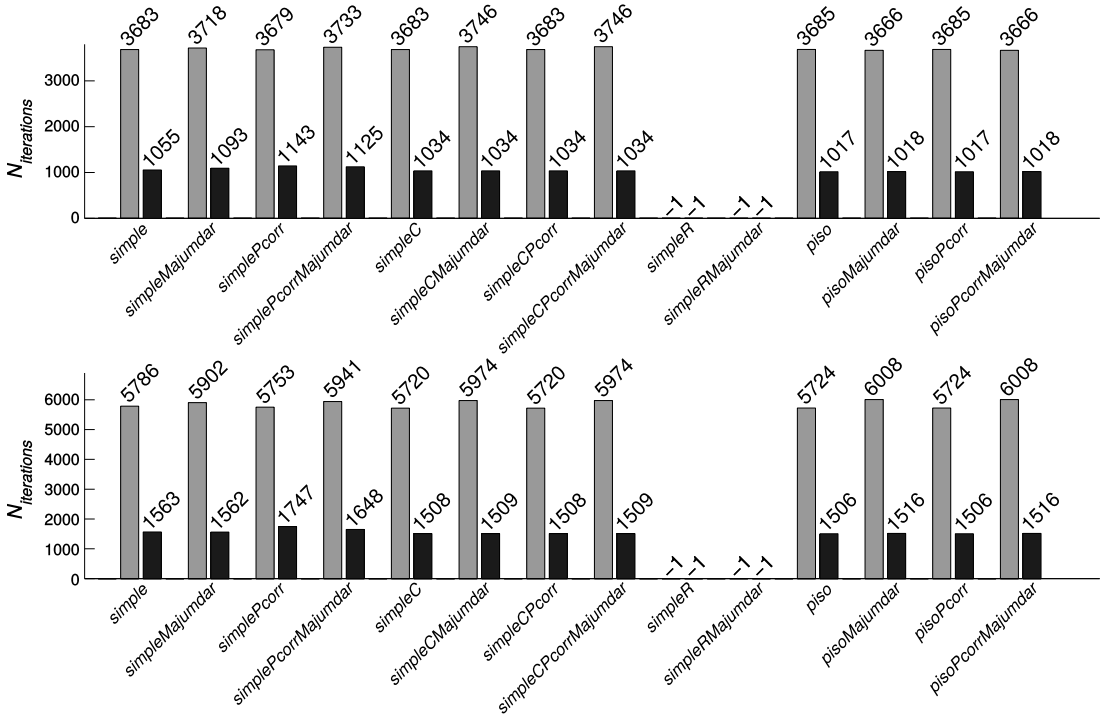


Figure 2.19: Required number of iterations to reduce all residuals below 10^{-6} (top) or 10^{-7} (bottom) when α_u is fixed to $\alpha_u = 0.2$ (grey) and $\alpha_u = 0.8$ (black)

First it should be noted that SIMPLE-R algorithm did not converge to the level of residuals considered here. For both *simpleR*, *simpleRMajumdar*, minimum achieved residual for velocity equations was about 10^{-4}. When comparing the rest of cases, Majumdar correction tends to increase the required number of iterations for a low value of α_u, and has almost no effect when considering the larger value of α_u. SIMPLE-C is again the best performing algorithm here for both small and large values of velocity under-relaxation factor.

2.7 Conclusions

In this chapter the Original Momentum Interpolation by Rhie-Chow, used to cancel the checkerboarding pressure problem in co-located finite-volume codes, has been explained in detail. Inherent problems to this approach have been identified and some of the available solutions in literature to eliminate velocity under-relaxation factor and time step size dependency have been implemented and tested in OpenFOAM®. Several pressure-velocity coupling algorithms have also been considered (with the required implementation of some of them, which were not available in the code by the time this work was developed).

The difficulty in the implementation of the time step dependency problem, its complex extension to the dynamic mesh case present in the gear pump problem, and the small influence observed in the study of the Taylor-Green vortex problem, drove us not into considering its application for the current study. Majumdar correction for under-relaxation factor dependency has been implemented and has shown to be able to correctly eliminate the α_u dependency of the OMIM in simple cases, with no significant influence on the convergence speed and a simple implementation. The analysis of a lid-driven cavity case showed that influence of α_u is almost negligible for a Reynolds number of $Re = 1000$ as far as the mesh refinement is enough to approach the benchmark solution, while higher dependency is found for a larger Reynolds number $Re = 5000$. In both cases Majumdar MIM correctly eliminates the dependency but generates velocity profiles further from the benchmark profiles. When a more complex case as a NACA 0012 airfoil profile is considered, convergence of turbulence quantities makes full independence of the solution with respect to α_u not achievable. However, Majumdar correction significantly reduces the difference in computed values of drag and lift coefficients. In principle, Majumdar correction could be applied to any pressure-velocity coupling algorithm. Simulations show that in general the correction does not greatly modify the number of iterations required to achieve a given level of convergence. When comparing pressure-velocity coupling algorithms, SIMPLE-C was the best performing option for the numerical simulations carried out here.

Considering this, SIMPLE-C together with Majumdar MIM was selected to be applied to gear pumps simulations, as a better performance and correct independence from under-relaxation factors is expected.

Chapter 3

Mesh motion strategy

3.1 Introduction

Probably the highest complexity in the CFD simulation of gear pumps is found in the mesh motion strategy. CFD offers a promising alternative to the very simplified approaches (theoretical, semi-empirical and 1D simulations) presented in the first paragraphs of section 1.3, which are typically difficult to generalize for a new given geometry and usually cannot provide an insight of the physical phenomena taking place inside the pump. However, the complex motion of the gears and the narrow clearances make mesh handling critical in the CFD approach. While the mesh manipulation already presents difficulties when treated in a 2D approximation, the case is inherently three-dimensional. The presence of axial clearances, relief grooves and cylindrical inlet and exit pipes makes this point clear, and it is even more obvious when a helical pump is considered. Computation of the leakage is also extremely difficult given the very tight clearances involved. As a matter of fact, most approaches available in literature are used to perform simulations where clearances are much higher than the their realistic value, in order to avoid the use of too refined meshes and the corresponding computational cost [14, 27, 69]. The most common approach is the use of Arbitrary Lagrangian-Eulerian formulations, as introduced in section 1.3. The problem is that they are generally applied by considering the deformation of an unstructured mesh that continues moving until its quality reduces excessively, moment in which it needs to be replaced (see for instance [14]). The generation of a new mesh is both time and memory consuming. If this needs to be performed often, the computational cost is excessive. Even if the mesh generation process can be reduced for spur gears (where a 2D mesh can be created and extruded to complete the 3D one) the process is still expensive in the global computational cost of the simulation. Besides when dealing with helical gears, that is not an option (some authors [59] considered the use of a number of rotated spur gears to define the helical gear, but the lack of accuracy of such approach is evident). Other methods that combine deformation with mesh refinement-agglomeration suffer again from the computational cost derived from a mesh modification process in which the number and/or addressing of cells, vertices, and internal faces unavoidably change.

For any of the previously described methods, and some others mentioned in

section 1.3, an additional problem arises when considering the interpolation of variables from mesh to mesh. The errors introduced in this process are inevitable and difficult to control.

Depending on the application, a proper estimation of the leakage and the flow characteristics might not be necessary, reducing the stringency in the correct description of the clearance gaps and relaxing the limitations on the time step. This, together with the simplicity of most of these approaches explain their wide application in literature.

In this chapter a new fully-automatic mesh motion strategy for the simulations of gear pumps is presented. The main idea is to reduce the computational cost and the mesh-to-mesh interpolation errors derived from the use of any of the alternatives specified before, while maintaining a general applicability. The first sections will introduce the operations involved in the dynamic deformation of the mesh. After this, two possible methodologies for the treatment of boundary mesh interfaces will be described, detailing the required topological changes. Then the implementation of the method in OpenFOAM® will be outlined. The last sections will cover different strategies that have been implemented for an efficient parallelization of the mesh with a proper performance of the proposed algorithm, and several options for the treatment of the (one or more) contact point(s) between gears. Throughout the chapter a set of examples of the applicability of the method to several types of gear and tooth profiles will be provided.

3.2 Overview of the method

One of the main goals when considering a dynamic mesh handling method is to reduce the mesh motion modification process while maintaining a good quality. Mesh modification is not significantly time consuming as long as the number of cells, points and faces remain constant and no reordering is required, or at least it is kept to a minimum. For instance in the re-meshing method the regeneration of the entire mesh, or a part of it, involves the calculation of the new vertices, faces and cells. Progressive refinement or agglomeration of cells also alters the number and addressing of cells, faces and points, leading to a tremendous increase in the computational cost.

With this idea in mind, the best option we can consider is to reduce the mesh modification to the motion of cell vertices. If only the point coordinates (actually only some of the points) are modified, cells, points and internal faces labeling remain constant and the mesh motion proceeds fast.

The proposed mesh motion algorithm considers the splitting of the fluid domain in three mesh regions. Fig. 3.1 shows an example of the separation of regions in the two-dimensional case. This case will be explain in detail and it will be further extended to the three-dimensional case in section 3.7

The region *"fixedCells"* contains inlet and outlet ports, in which mesh points are kept fixed. Cells around the gears are separated in *"gear1"* and *"gear2"* regions. For all cells contained in *"gear1"* or *"gear2"*, vertices motion proceeds in two steps, as shown in Fig. 3.2. From an "undisplaced" situation (Fig. 3.2a, which is generated as described in section 3.4) a rigid rotation along the axis of each gear

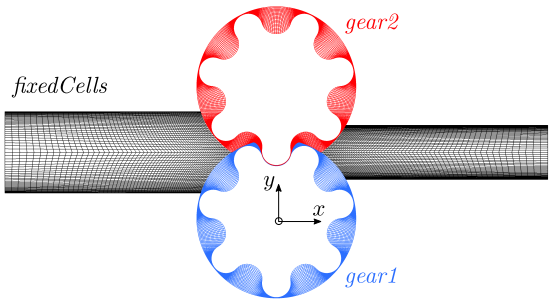
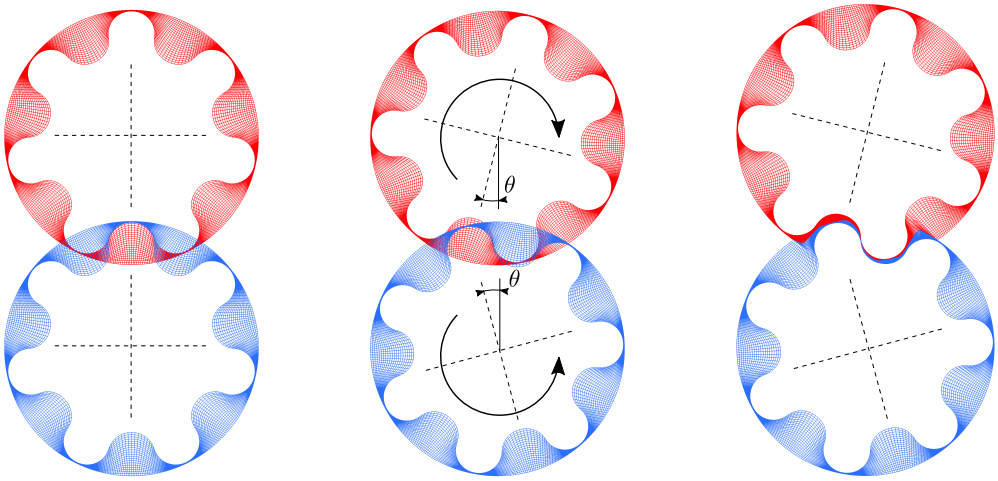


Figure 3.1: Definition of mesh regions

is first considered (arriving to Fig. 3.2b) and a projection (described in section 3.6) to a common interface is applied, avoiding overlapping cells, arriving to the situation shown in Fig. 3.2c. The projection consists in scaling the points of each gear down to a common line that lies always between the two gear boundaries. This line will be called hereafter interface, and its calculation is explained in section 3.5.



(a) Initial undisplaced mesh (b) Rotated mesh (c) Projected mesh

Figure 3.2: Mesh motion steps

Point coordinates of the "undisplaced mesh" are saved in memory during the calculation. For a given time, the rigid rotation step will rotate these points an angle equal to the integral of the rotational speed from the initial time up to the current time. Thus, we avoid the accumulation of errors (which can be dangerous given the low clearances), and we always have access to a "non-projected mesh" so that we do not need to un-project the points. It should be stated that only a list with the coordinates of the undisplaced points (belonging to *"gear1"* or *"gear2"* zones) is saved in memory, and not the entire mesh, or the entire list of points. Minimizing the memory requirements is important when considering a large case like this.

3.3 Profile geometry

The first step in the mesh generation process is the definition of the geometry, and in particular the definition of the gear tooth profile. In general two options are considered for this case (see Fig. 3.3). Either a classical involute or cycloid profile can be selected, by providing the required parameters, or the user can define the profile as a list of points. When that is the case, the user must also provide the pitch diameter ($D_p = 2R_p$) together with the list of points.

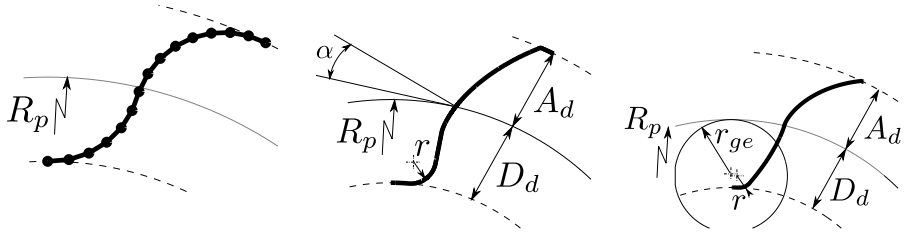


Figure 3.3: Tooth profile: user-given list of points (left), involute profile (center), cycloid profile (right)

If an involute profile is chosen, the definition is complete if the following parameters are defined: number of teeth (Z), pitch diameter ($D_p = 2R_p$), pressure angle (α), addendum (A_d), dedendum (D_d), and fillet radius (r). Similarly the cycloidal profile is determined by providing the number of teeth (Z), pitch diameter ($D_p = 2R_p$), radius of the rolling circumference that generates both epicycloid and hypocycloid parts of the profile (r_{ge}) (note that for two equal gears, equality of rolling circumference diameter is a necessary condition for the profiles to be conjugate) and fillet radius (r).

Besides, for any of the three cases mentioned, a fit in the manufacturing process is assumed, and the lash can be specified by a *backlash* angle, and a clearance distance d, as shown in Fig. 3.4.

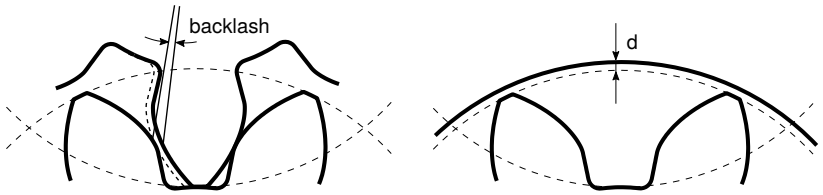


Figure 3.4: Definition of *backlash* angle and d clearance, applicable to any of the described profiles

The last parameter to be defined in any of the profiles chosen is intrinsic to CFD simulations. The requirement of positive volume cells forbids us from simulating the real contact between profiles. We are therefore forced to maintain a distance between the gears that are supposed to be in contact and apply some external model to avoid the fluid to move through the contact point. The separation is defined by assuming the distance between centers to be enlarged from the theoretical value (D_p) by a user specified value: *spacing* as shown in Fig. 3.5 (Note that the *backlash* angle is assumed to be zero in the figure for the sake of

clarity).

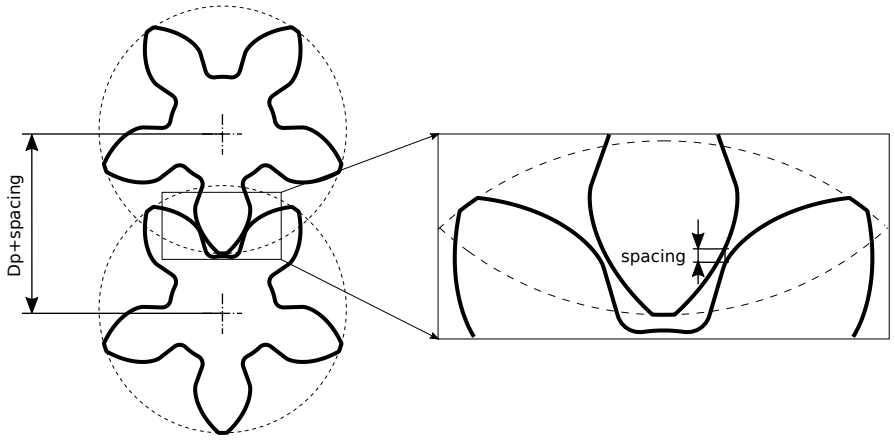


Figure 3.5: Definition of *spacing* distance, applicable to any of the described profiles

3.4 Undisplaced mesh generation

Once the profile is defined, the next step is the generation of an undisplaced mesh, such as the one shown in Fig. 3.2a. Given the symmetry, the process starts with the generation of the mesh corresponding to the user-provided half-tooth profile. An application has been created in order to generate a block-structured mesh, using user-defined number of cells and grading ratios, as shown in Fig. 3.6.

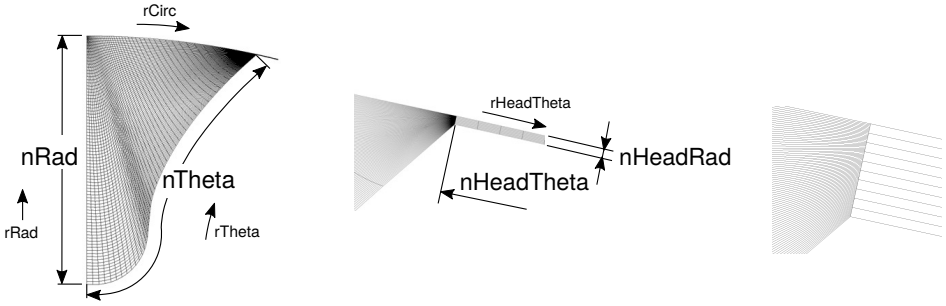


Figure 3.6: Mesh generation for involute or cycloid profiles (left), zoom of the top land region (center) and closer zoom to the non-conformal interface (right)

Given the low clearances, a non-conformal interface is allowed in the top of the tooth (land region), in order to reduce the number of cells between the top of the tooth and the casing (see right side of Fig. 3.6), which could negatively influence the convergence of the solver. For involute or cycloid profiles, the position of the non-conformal interface is clear, given the sudden change in the steepness of the profile. For user-defined rounded profiles a similar strategy is offered, but the position of the interface is determined by an additional parameter (r_{split}), as shown in Fig. 3.7.

Furthermore, the number or cells and grading ratios (nTheta and rTheta for instance) are not necessarily one single value. The user is allowed to divide the

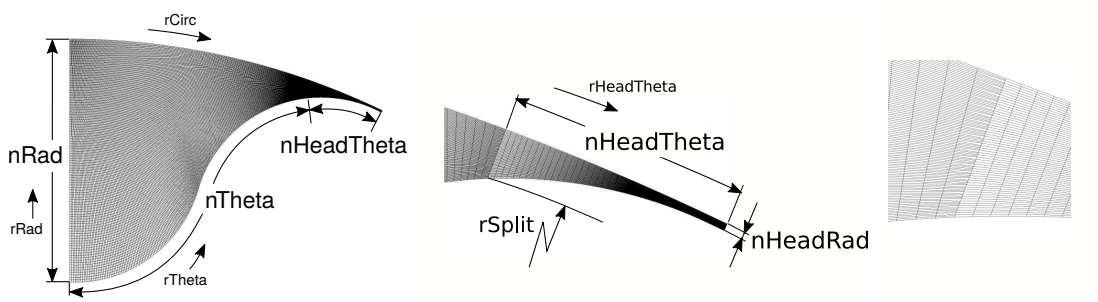


Figure 3.7: Mesh generation for user-defined profiles (left) and zoom of the top land region (center) and closer zoom to the non-conformal interface (right)

total length covered by nTheta cells (length denoted as lTheta) into intervals (lTheta= $[l_{\theta_1}, l_{\theta_2}...l_{\theta_k}]$), where l_{θ_k} represents a ratio of the total l_θ (so that $\sum_k l_{\theta_k} = 1$). With this division the scalar values (nTheta and rTheta) become now vectors: nTheta= $[n_{\theta_1}, n_{\theta_2}...n_{\theta_k}]$ and rTheta= $[r_{\theta_1}, r_{\theta_2}...r_{\theta_k}]$, so that the k fragment of the entire edge (l_θ) represents l_{θ_k} of the total length, and contains n_{θ_k} cells, whose spacing grows according to the r_{θ_k} expansion ratio. This is applied to the three edges (lTheta,lCirc and lRad), as shown in Fig. 3.8. This allows the user to add some refinement near the walls to create for instance a boundary layer both near the gears and near the pump casing, or to distribute the cells in a more general way, in order to improve the refinement in a given region if desired. On the other hand, the head cells are always assumed uniformly distributed in the radial direction, and nHeadTheta and rHeadTheta continue to be scalar values.

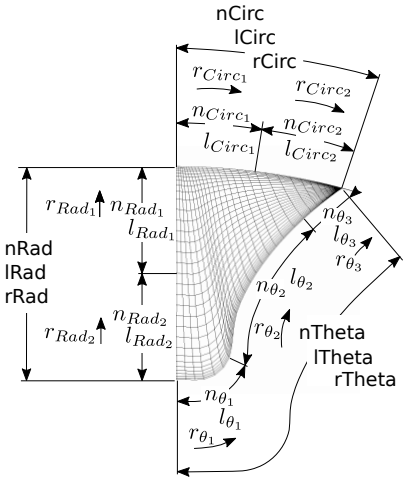


Figure 3.8: Intervals definition of grid spacing and expansion ratios

Additionally, the user is given the option of further improving the quality of the mesh by reducing the non-orthogonality of the cells near wall boundaries (gears or casing). Let us consider point P (see Fig. 3.9) in a mesh generated with the algorithm described so far. Let us define r_1 and r_2 distances from P to O_1 and O_2 ($r_1 = |\overline{\mathbf{O_1P}}|$ and $r_2 = |\overline{\mathbf{O_2P}}|$), and let us also consider tg_1 and tg_2, tangent lines to the wall boundaries at points O_1 and O_2 respectively. If we wanted the edge $\overline{\mathbf{O_2P}}$ to be orthogonal to tg_2 (to reduce the non orthogonality of the cells near

O_2) we would need to move P to R_2. Similarly to reduce the non orthogonality near O_1, P should be moved to R_1.

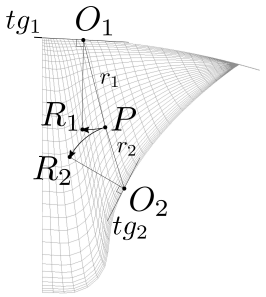


Figure 3.9: Schematic explaining useful information for final improvement of undisplaced mesh

It is clear that moving P to R_2 is important when P is close to O_2, and moving it to R_1 is important when P is close to O_1. In other words, the vicinity to the wall should be emphasized when determine the weighting function that approximate P to R_1 or to R_2. In order to mathematically express this, the function $f(d)$ defined in Eq. (3.1) and Fig. 3.10 is considered.

$$f(d) = 1 - \frac{1}{1 + e^{-k\left(\frac{d}{d_{min}k_d} - 1\right)}} \tag{3.1}$$

where k and k_d are user-defined constants (with default values of 1 and 0.25) and d_{min} is the distance of the mesh edge that joins O_1 and O_2.

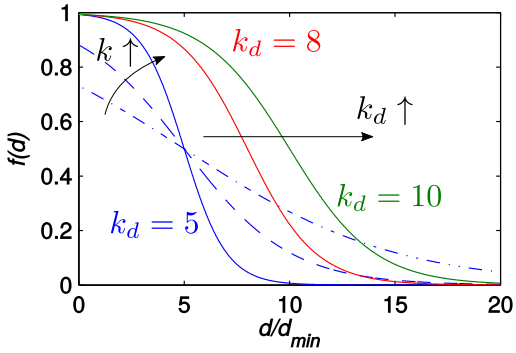


Figure 3.10: Blending function

Using this function r_1 and r_2 are used to determine $r_1' = r_1 f(r_2)/f(0)$ and $r_2' = r_2 f(r_1)/f(0)$. When the P is close to O_1, $r_1 < r_2$. In that case $f(r_1)/f(0)$ will be very small while $f(r_2)/f(0)$ will approach unity so that $r_1' << r_2'$. Using this information the final location of P (denoted as P') will be given by Eq. 3.2.

$$\mathbf{OP'} = \mathbf{OP} + \beta_a \frac{f(\min(r_1', r_2'))}{f(0)} \left(\frac{r_2'}{r_1' + r_2'} \mathbf{OR_1} + \frac{r_1'}{r_1' + r_2'} \mathbf{OR_2} \right) \tag{3.2}$$

where O is the coordinates origin and β_a is a user-defined parameter that determines the maximum allowable change. $f(\min(r_1', r_2')$ is used to ensure that points

located far from both walls are not modified, and only non-orthogonality near the walls is corrected.

It can be easily deduced from Eq. 3.2 that the location of P' will be closer to R_1 if P was initially closer to O_1 and closer to R_2 otherwise.

The described method is applied to all the interior points of the mesh. The evolution from the initially described method with scalar definitions of nTheta, rTheta, etc. to the intervals definition, to the non-orthogonality corrected mesh is shown in Fig. 3.11.

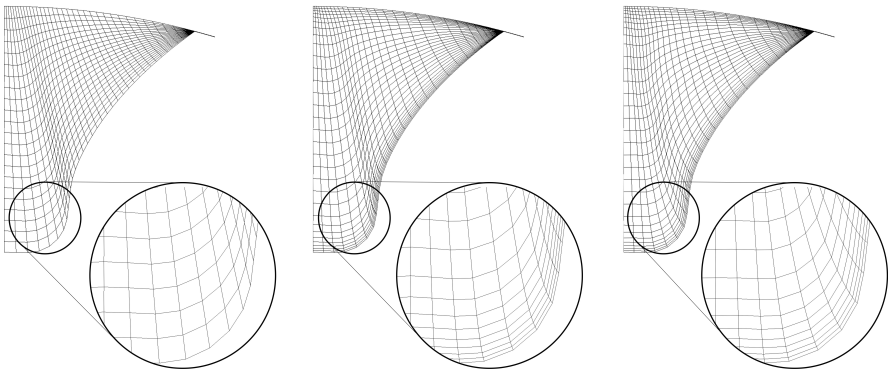


Figure 3.11: Mesh improvements: initial mesh (left); interval definition of cells distribution (center); improvement of non-orthogonality (right)

Once the initial half-tooth mesh is generated, this can be replicated until completing the entire gear. The second gear is obtained as a rotation and displacement of the first. This method produces as a result the "undisplaced meshes" shown in Fig. 3.12

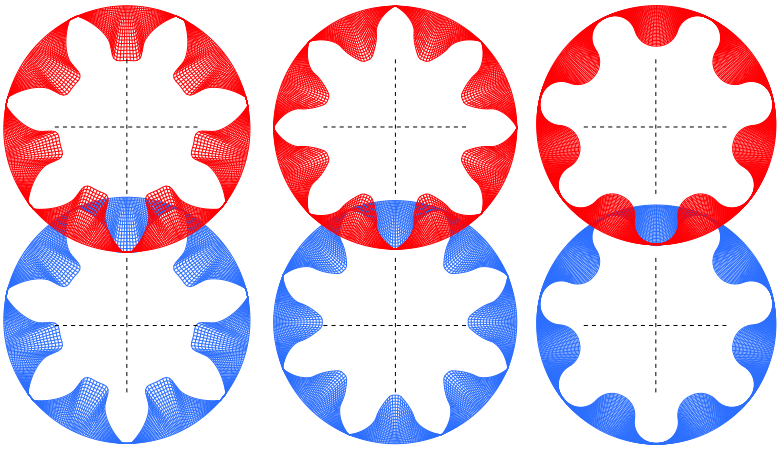


Figure 3.12: Examples of complete "undisplaced" meshes: involute (left), cycloid (center) and user-defined (right) profiles

3.5 Interface calculation

As described in section 3.2 once the "undisplaced mesh" has been generated (Fig. 3.2a), it is rotated to the position determined by the current time (Fig. 3.2b). Before proceeding with the last step (projection of the mesh to Fig. 3.2c), the line on which to perform the projection needs to be created. We need to define a line that always lies within the two gear walls for any given profile. Several alternatives have been considered with this scope and will be individually explained next.

3.5.1 Complete interface

The first group of alternatives will include those methods that allow us to calculate the entire interface in one single step. As an example, one of the methods that could be considered for a rounded profile consists in building the interface by the medium points of the segments joining "corresponding points". The corresponding point of the top of the tooth of one of the gears would be the closest valley of the other gear. The medium point of this segment would be a point of the interface (see Fig. 3.13). In order to determine these points, the list of points used to define upper and lower gear boundaries could be directly the mesh vertices, or another user-given list of points with increased point density.

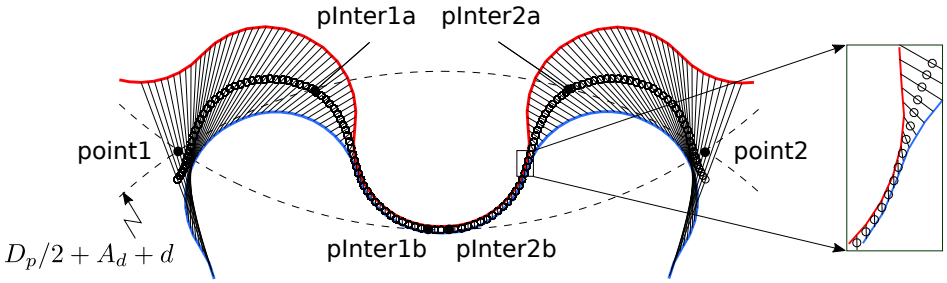


Figure 3.13: Calculation of an interface based on medium points

To ensure the smoothness of the mesh after projection, the interface should start from point1 and finish in point2 (see Fig. 3.13). Therefore some additional correction is needed at the beginning and end of the interface line. The first idea was to determine the intersections between the calculated interface and the external circumferences with radius $(D_p/2 + A_d + d)$. The interface is then limited to the central part between the intersections (between point1a and point2a in Fig. 3.14), and completed with arcs in both ends.

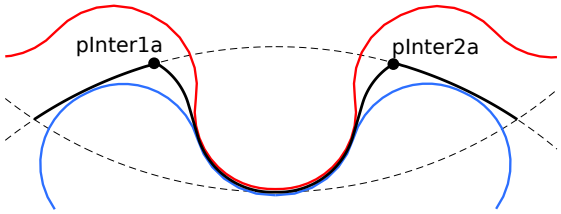


Figure 3.14: Correction of the interface left and right limits

However, following the gear rotation, the arc used to complete the interface

should change from the circumference centered in one of the gears to the one centered in the other gear. In order to smooth this transition a blending function is proposed. As shown in Fig. 3.15, if the interface calculated with middle points is completed with an arc centered in the lower gear, curveA would be generated. If it is completed with the arc centered in the upper we would arrive to curveB.

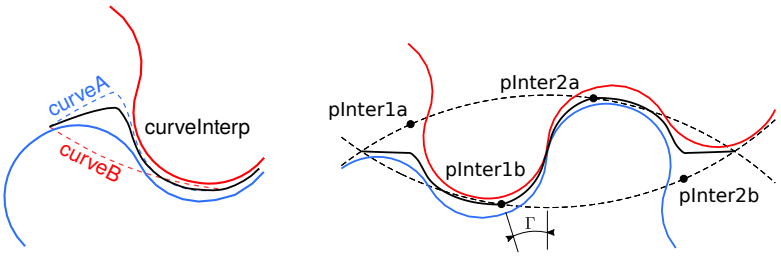


Figure 3.15: Final blending of the interfaces

As the gears rotate we would require a smooth transition from curveA to curveB. This is achieved by considering a blending between the two as shown in Eq. (3.3).

$$\text{curveInterp} = \lambda\ \text{curveA} + (1-\lambda)\ \text{curveB} \tag{3.3}$$

where λ is a blending parameter that depends on the position of the gears. This relation is expressed as $\lambda(\Gamma)$ where Γ (see Fig. 3.15) is the angle between the vertical line and the line that joins the center of the lower gear with its first valley at the right of point1.

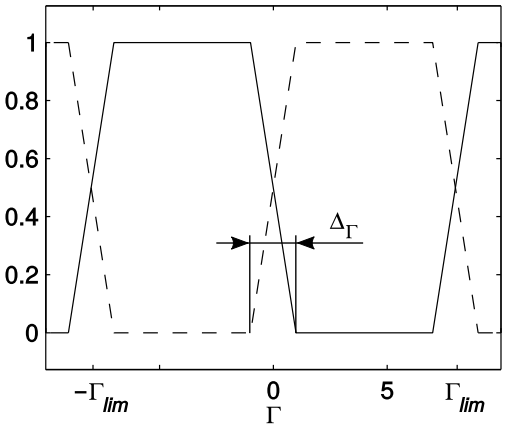


Figure 3.16: Definition of blending factor $\lambda(\Gamma)$: curveA weight (λ) (—); curveB weight $(1-\lambda)$ $(- - -)$

The same strategy applies to the curve completing the interface in the vicinity of point2. Parameters Γ_{lim} and Δ_Γ determine when and how fast the blending proceeds, and can be both specified by the user.

The described strategy works properly when smooth rounded profiles are considered, but it could fail when very low clearances are used. Therefore some other alternatives were studied, as presented in the next section.

3.5.2 Separated interface

The second method we propose consists in separating the profiles of the gears in the gearing region in pairs of half-tooth profiles, as the ones marked in Fig. 3.17. A "medium line" is calculated between each pair of half-tooth poly-lines, and is then completed with arcs with the corresponding center. This methodology (as shown in Fig. 3.18) is applied to the pairs in the inner part of the gearing region. For the interface near point1 and point2 some additional blending is performed as it will be explained later. The separation in pairs guarantees the applicability of the method to gears with any number of teeth.

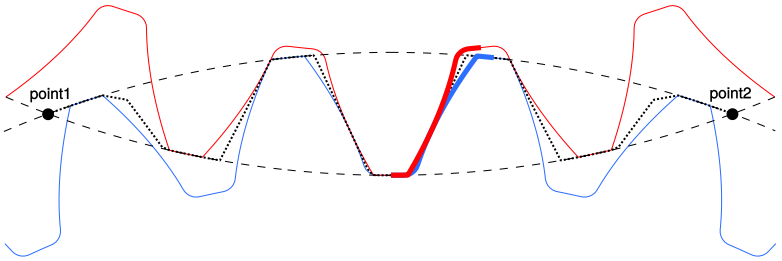


Figure 3.17: Definition of the *separated interface* method. A pair of half-tooth profiles has been marked in the center of the figure

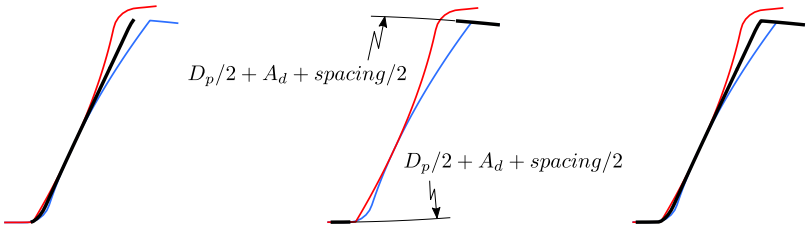


Figure 3.18: Definition of the single interface of a profile pair: medium line (left); arcs (medium); complete single interface (right)

While the generation of the medium line can seem straight forward, a generally applicable algorithm for any profile in any position of the gears is not so obvious. Two alternatives were found to perform properly for the tested profiles: the hereafter called *arcs* method and the *frechét* method.

Arcs method

The *arcs* method starts by considering the half-tooth profiles with the same distribution of points of the gear wall boundary mesh vertices (without increased point density). A series of arcs is then defined; starting from the top of one tooth, an arc co-centered with the gear that tooth belongs to, is drawn until it intersects the other profile. The intersection determines the end point of the arc, and the medium point is the first point of the interface. The process continues with arcs that keep using the same center and start in the points of each of the profiles. The distance between the extremes of the arcs keeps reducing until a minimum distance is found (near the contact point). When this happens arcs centered in the

center of the other gear are considered and the process continues until all points of both profiles have been used. The procedure is shown in Fig. 3.19. The use of mesh vertices guarantees that the medium line will always lays between the two profiles as long as gears wall boundaries do not overlap.

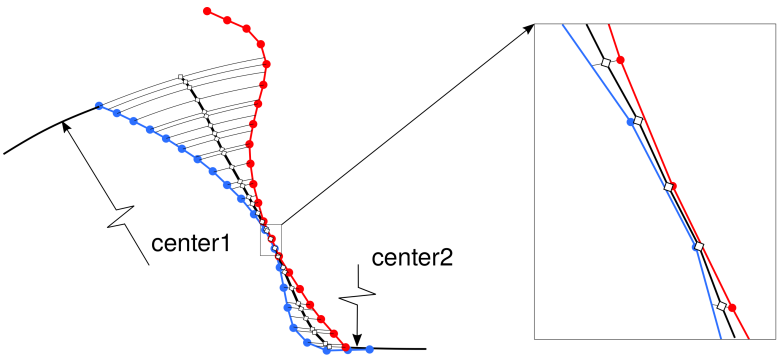


Figure 3.19: Definition of the *arcs* methodology

Fréchet method

The idea behind the *fréchet* method came from the so-called Fréchet distance, a mathematical algorithm that allows the calculation of the similarity between two polygonal curves. This algorithm has been further extended in literature for many applications [8, 97] including for instance the determination of a "medium line" that minimizes the maximum weak Fréchet distance to two given profiles [56]. However this method was found to be too time consuming for its application to our problem. The proposed method (see Fig. 3.20) starts by considering the profiles with a high enough point resolution (not only the gear wall boundary vertices are considered). Examining two starting points (f_1, f_2, one from each profile), the length of $\overline{f_1 f_2}$ is determined. Keeping f_1 fixed, a new segment $\overline{f_1 f_2'}$ is inspected. If the segment is shorter ($|\overline{f_1 f_2'}| < |\overline{f_1 f_2}|$), next point in the profile (f_2'') is studied, leading to the segment $\overline{f_1 f_2''}$ whose length is compared to the last. This process is repeated until the new segment becomes larger than the previous (in the example $|\overline{f_1 f_2''''}| > |\overline{f_1 f_2'''}|$). Medium points of first and last segments (m_0 and m_{00}) are then used to determine the middle point between them i_0 which will be part of the interface. After this process finishes for f_1 (which was kept fixed), the point of the last segment from the other profile (f_2''') becomes fixed and the process is repeated. However, the determination of middle points m_x and i_x is only considered when the fixed point belongs to profile used to determine the first point m_0. This avoids the generation of non-smooth interface curves.

It is clear that when this method is applied to diverging lines many more points will be generated (the change of fixed point will occur more often). Therefore for the case of tooth profiles (Fig. 3.20), the first step to consider is the determination of a pair of points (one from each profile) for which the distance between them is minimum (they will be near the contact point). Then the explained methodology is applied moving towards each direction (diverging lines in both cases). Additional difficulties are found when considering the peculiar case of points near the

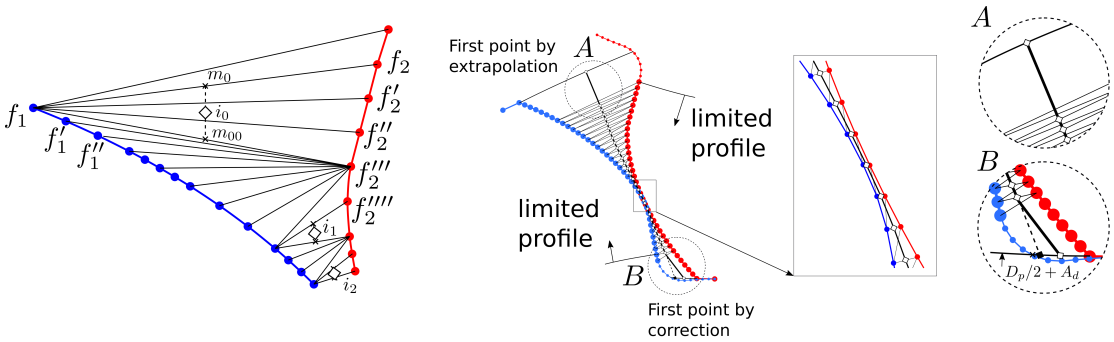


Figure 3.20: Definition of the *fréchet* method. Basic definition (left); application to an involute profile (center); zoom (right)

top of one tooth. In fact the direct application of the method from the contact point until the last point could generate problems. To avoid them, before the method is applied, the profiles are limited as shown in Fig. 3.20, so that the part of the profile including the fillet radius is not considered in the calculations. For the part of the interface that has not been calculated due to this limited profile two options are considered (the code will choose the most appropriate one): either the last two calculated points of the interface are used to extrapolate the location of the last one (as in case A in Fig. 3.20), or this particular point is calculated by using the previously described *arcs* method (as in case B in Fig. 3.20). Notice that if the extrapolation method (as applied in case A) was used in case B, the determined location of the last point would be too close to the gear profile. In this case the second method (the one actually applied in case B) is preferred.

While this method is more complex and slightly more time consuming, *fréchet* is applicable to all: rounded, involute, or cycloid profiles, and possible problems in the contact region when clearances are reduced can usually be solved by increasing the point density in the profiles used for the determination of the interface.

Blending

With any of the two methods described above (*arcs* or *fréchet*), the medium lines are generated and completed with arcs as it was explained in Fig. 3.17. However, as it was shown with the "complete interface" calculation in section 3.5.1, additional blending is required near point1 and point2. In this case the blending is performed in a different way.

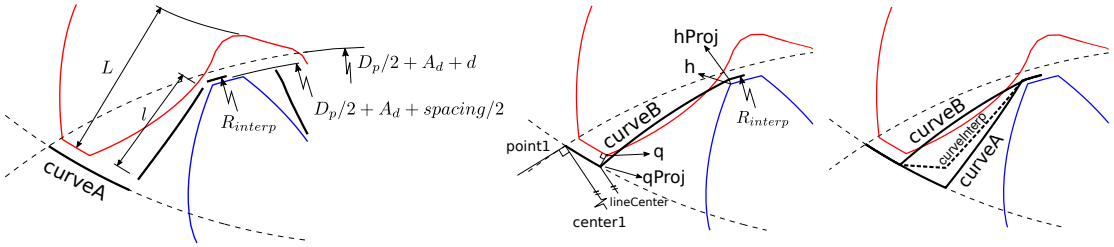


Figure 3.21: Defnition of blending algorithm for "separated interface" method: curveA (left), curveB (center), curveInterp (right)

Following Fig. 3.21, *arcs* or *fréchet* methods would provide the central part of the first curve, curveA (the length of this line is l in Fig. 3.21). For inner "medium lines" these medium lines were completed with arcs of radius $(D_p/2 + A_d + spacing/2)$ (see Fig. 3.18). However as we approach point1, these arcs should approach the external circumference arcs, of radius $(D_p/2 + A_d + d)$. To make this change smooth, the radius of the arcs (R_{interp}) is progressively increased by a blending function as shown in Eq. 3.4;

$$R_{interp} = (D_p/2 + A_d) + \gamma(spacing/2) + (1 - \gamma)d \tag{3.4}$$

where γ is a coefficient reduces as the medium line approaches point1, following Eq. 3.5.

$$\gamma = \min(C_\gamma l/L, 1) \tag{3.5}$$

where l is the length of the medium line, L is the whole dept $(L = A_d + D_d)$ and C_γ is a user defined value which sets the speed and starting point of the blending.

Besides, the medium line should smoothly approach an arc to improve mesh quality. As shown in Fig. 3.21, this is performed by creating other curve (curveB) and defining the final interface as a blending (see Eq. 3.6).

$$\text{curveInterp} = \gamma\ \text{curveA} + (1 - \gamma)\ \text{curveB} \tag{3.6}$$

where γ is the same blending parameter previously defined in Eq. 3.5 and curveB will be defined next. As the tooth approaches point1, the length of the medium line (l) decreases, γ decreases and curveInterp approaches curveB. As shown in Fig. 3.21, curveB starts with an arc of radius $(D_p/2 + A_d + d)$, until point qProj. This point is determined by the projection of the point where the steepness of the profile suddenly changes (for involute or cycloid, see Fig. 3.6), or the user defined location of the non-conformal interface (see Fig. 3.7), (point q) in the external circumference of radius $(D_p/2 + A_d + d)$. Similarly point hProj is the projection of point h in the circumference of radius R_{interp}. The second part of curveB consists of an arc which goes through hProj and qProj and whose center is located in the line lineCenter, parallel to the line that joins point1 with the center of the lower gear. Last part of curveB is just an arc of radius R_{interp}. With this method we ensure curveB will eventually turn into just an arc of radius $(D_p/2 + A_d + d)$ and therefore the final interface curveInterp will smoothly approach it too.

A similar strategy is performed when the tooth approaching point1 belongs to the other gear, or when it is happening in the proximity of point2.

3.6 Mesh projection

Following the procedure introduced in section 3.2, last step to finally arrive to Fig. 3.2c is the mesh projection. The undisplaced mesh (section 3.4) has been rotated and its cells need to be projected to the calculated interface (section 3.5). An example of the projection process that will be explained in this chapter is shown in Fig. 3.22.

Only the "radial edges" for which the last point lies within point1 and point2 are considered for projection. For example $\overline{S_1S_2}$ is projected while $\overline{Q_1Q_2}$ is not

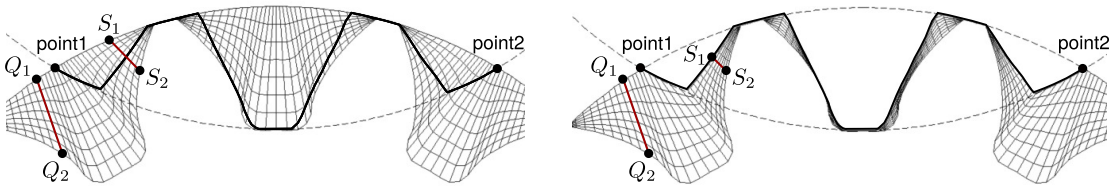


Figure 3.22: Mesh projection step for lower gear of a 2D mesh of an involute profile: non-projected mesh (left), projected mesh (right)

modified (as shown in Fig. 3.22). Two possibilities for the projection algorithm have been studied: linear or spline projection.

Linear projection

The basic projection method is based on scaling the list of connected cell vertices following the line that joins the first and the last in the radial direction. As shown in Fig. 3.23, the list of points $\{l_1, ...l_n\}$ must be projected. First, intersection l_{inter} between the straight line $\overline{l_1 l_n}$ and the calculated interface is computed.

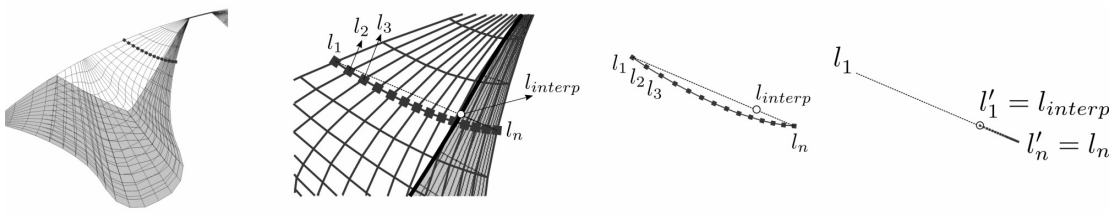


Figure 3.23: Linear projection

The entire list of points is then scaled so that the final position of point l_i, l'_i is obtained by Eq. 3.7.

$$\mathbf{l'_i} = \mathbf{l_n} + (\mathbf{l_i} - \mathbf{l_n}) \frac{|\overline{l_{interp} l_n}|}{|\overline{l_1 l_n}|} \tag{3.7}$$

Even if fast, this simple method might fail when fine curved meshes are used, since curved lines are becoming straight as shown in Fig. 3.24.

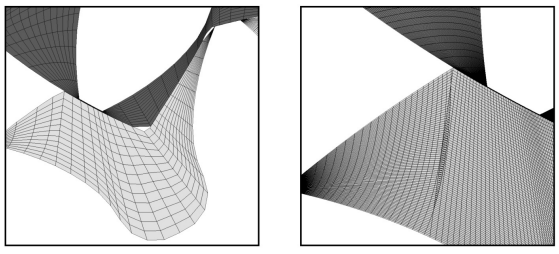


Figure 3.24: Example of application of linear projection: coarse mesh (left), fine curved mesh (right)

Therefore other projection method could be based on scaling points in a similar manner, but using as a reference a spline.

Spline projection

The spline projection method works in the same manner as the linear one (see Fig. 3.25).

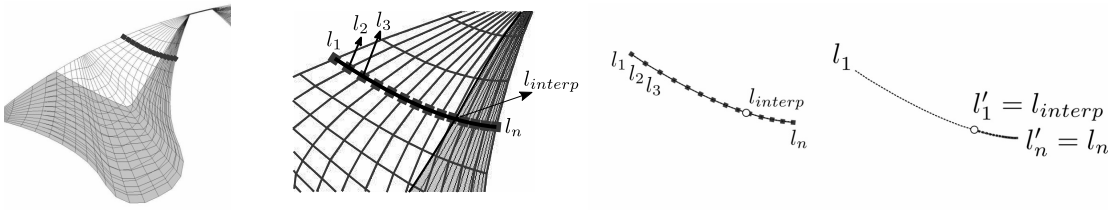


Figure 3.25: Spline projection

However, in this case, the distances are calculated as the integrated length of the spline segments. Thanks to this, the method would also work for fine curved meshes, as shown in Fig. 3.26.

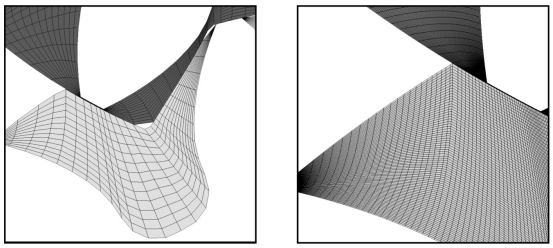


Figure 3.26: Example of application of linear projection: coarse mesh (left), fine curved mesh (right)

With both linear or spline projection methods, one additional parameter is considered. If the initial mesh has been generated with some user-specified grading ratios for wall refinement, the application of such refinement in the zones where the distance between gears is minimum would generate cells with extremely high aspect ratio. To avoid this, the user is given the possibility to consider a uniform distribution of points when the cells are located far from point1 or point2, and the initial graded distribution of the undisplaced mesh when approaching these points. The final location of each point will be a blending of both strategies, using as blending factor $\chi = \min(|\overline{l_{interp}l_n}|/(A_d + D_d), 1)$. Final position $\mathbf{l"_i}$ is then determined by Eq. (3.8).

$$\mathbf{l"_i} = \mathbf{l_n} + \frac{|\overline{l_{interp}l_n}|}{|\overline{l_1 l_n}|}\left[\chi(\mathbf{l_i} - \mathbf{l_n}) + (1-\chi)(\mathbf{l_1} - \mathbf{l_n})\frac{n-i}{n-1}\right] \tag{3.8}$$

Further corrections

Any of the projection methods shown above can be further corrected in the vicinity of point1 and point2. The projection of only some of the edges generates holes in the mesh that cannot be easily handled by the interface boundary conditions; therefore it is convenient to eliminate them. An easy way to solve the problem is to make sure there is always one cell vertex in each of those points. For instance,

looking at point1, after rotation of the non-projected mesh, the closest cell vertex to point1 is selected and moved to the coordinates of point1. When the linear projection is being used the rest of vertices in the radial direction can be moved following Eq. 3.7 (see Fig. 3.27).

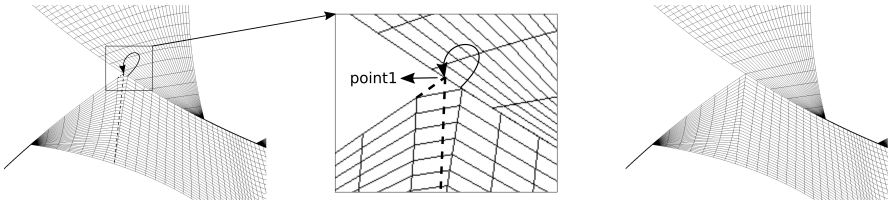


Figure 3.27: Projection of cell vertex to point1 when linear projection is used

When the spline projection is selected (see Fig. 3.28), and following the notation introduced in Fig. 3.25) l_1 is moved to $l_1' =$ point1 by a vector $\mathbf{d_1}$ ($\mathbf{d_1} = \mathbf{point1} - \mathbf{l_1}$).

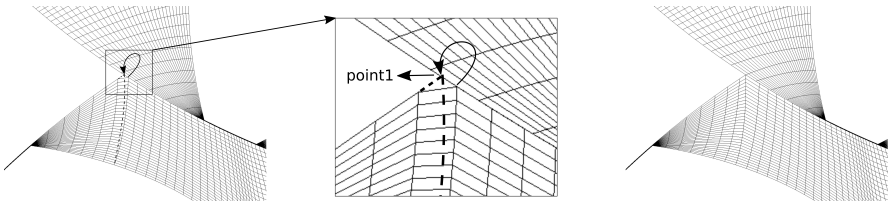


Figure 3.28: Projection of cell vertex to point1 when spline projection is used

For the rest of points $\{l_2, l_3 ... l_n\}$ the displacement $\mathbf{d_i}$ is scaled with their distance to l_n according to Eq. 3.9 for the linear projection and the equivalent, following the curved edge, for the spline projection.

$$\mathbf{d_i} = \mathbf{d_1} \frac{|\overline{l_i l_n}|}{|\overline{l_1 l_n}|} \tag{3.9}$$

3.7 Extension to 3D meshes

So far the method has been applied to two-dimensional meshes. The extension to three-dimensional straight-cut and helical meshes is straight forward. As a matter of fact, the final outcome of the entire strategy for the two-dimensional case is a map of displacement vectors for some of the vertices of the mesh. The method is slightly different for straight-cut and helical meshes, so it will be explained separately.

Straight-cut gear meshes can be generated by the extrusion of a 2D mesh. For these cases, the calculated field of displacements can be applied to transform the points of each *z-level* of the mesh, as shown in Fig. 3.29.

This would be the fastest 3D case scenario since no re-calculation is needed. Interface and projection are calculated only once and the displacement field is mapped to the rest of *z-levels*. In fact, in OpenFOAM®, this is done even in the two-dimensional case, since cells are always visualized as thee-dimensional and

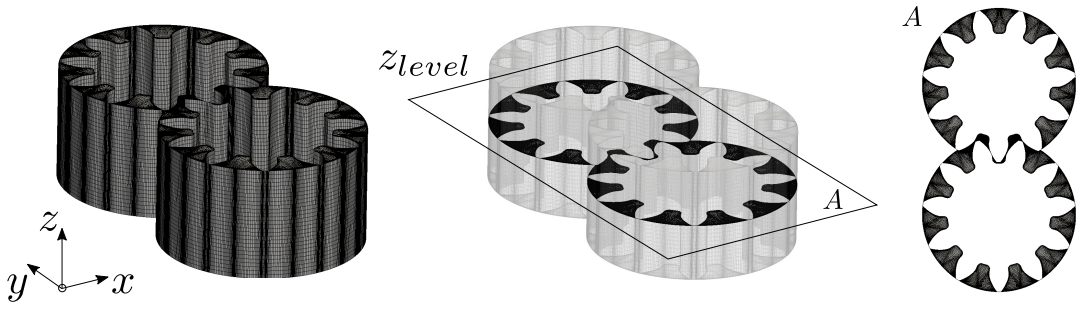


Figure 3.29: Application of the proposed method to a 3D straight-cut gear mesh

the two-dimensionality is applied when considering *empty* boundary conditions, which in our case would be applied to the top and bottom faces in the z direction.

For helical gears, the non-projected mesh can be generated as a straight-cut mesh (extruded from the 2D case), plus an additional rigid rotation of points with respect to the gear axis depending on the pitch of the helix P_h. The angle of rotation will be proportional to the difference in z coordinate between a given point and the lower base of the gear, following Eq. 3.10, as shown in Fig. 3.30.

$$\theta_i = 2\pi / P_h (z_i - z_0) \tag{3.10}$$

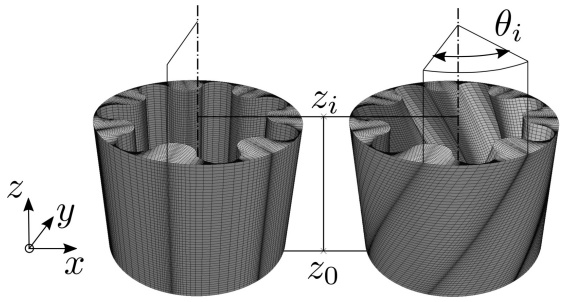


Figure 3.30: Generation of non-projected mesh for helical gears

The interface calculation and projection algorithm can then be applied to each *z-level* (see Fig. 3.31) with slight modifications.

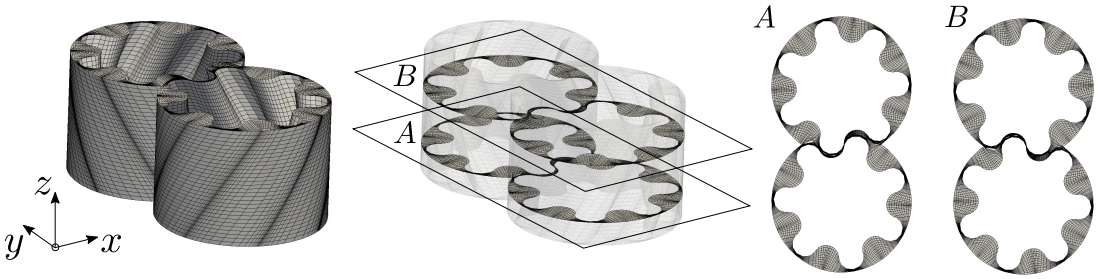


Figure 3.31: Application of the proposed method to a 3D helical gear mesh

First of all, the map of displacements is different for each *z-level* and therefore the calculation of interface and projection must be performed once for each *z-level*.

For a given level z_i the algorithm is exactly the same as in the 2D case, but it must consider that the rotation angle for the given z is given by $\theta_i = \theta + 2\pi/P_h(z_i - z_0)$.

Secondly, the critical problem appears when considering the correction of mesh holes near point1 and point2 (as explained in section. 3.6 for the 2D case). If we focus on the cells near point1 (which represents a line in the 3D case, line1), if 2D procedure is followed and only the closest vertex is moved to point1 at each *z-level*, holes will still appear and very concave cells will be generated, as shown in Fig. 3.32.

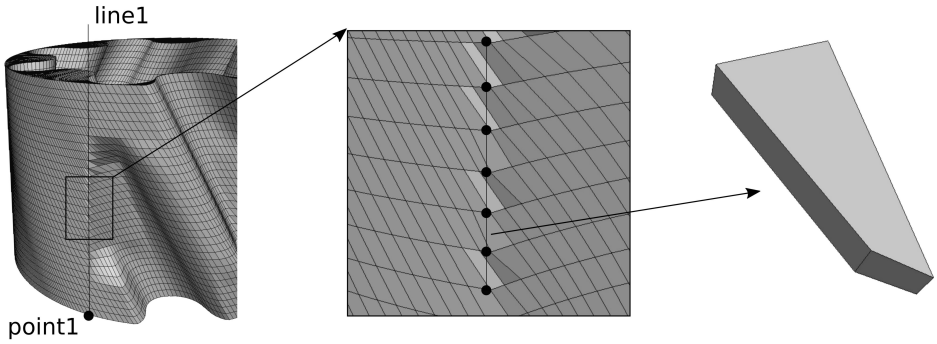


Figure 3.32: Problem of vertex projection to point1 in the case of helical gear meshes

In this case, the problem is not only affecting the interface boundary condition, but concave cells will also limit the convergence of solvers. In order to make the gaps disappear, we need to make one of the edges of those cells become vertical, and align it with line1. It must be only one edge per *z-level*, since otherwise we would be generating zero-area faces, which are of course not accepted by the code. As shown in Fig. 3.33, the proposed method consists in identifying the edges that are crossing line1, choosing the one in the center (if more than one is crossing) and projecting upper and lower vertices adequately to the point1 of the respective *z-level*, for that edge to become aligned with line1.

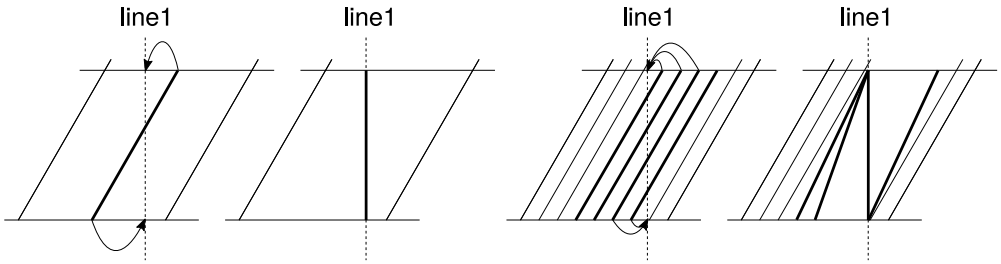


Figure 3.33: Mesh overview before and after the correction: coarse meshes (left); fine meshes (right)

As a consequence the previously concave cells become wedge-like cells as shown in Fig. 3.34. The treatment of interface boundary conditions is simplified, and the concavity of the cells is corrected, even if a little higher non-orthogonality might appear for some cells.

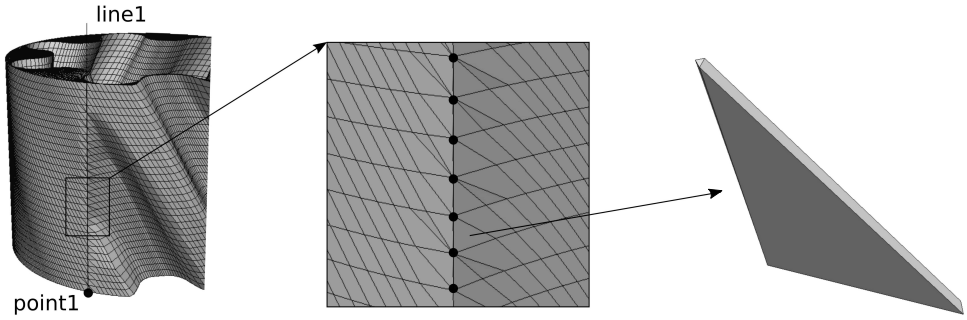


Figure 3.34: Application of the projection strategy in line1 and wedge-like generated cells

3.8 Fixed cells meshing

While the process explained so far applies to the generation and motion of all cells contained in the *gear1* and *gear2* regions (see Fig. 3.1), no information has been given regarding the meshing process of the *fixedCells* region. In fact, since neither the cells nor the points or faces need to be modified during the simulation, any generation strategy can be followed to mesh that region. As an example, Fig. 3.35 shows a block-structured 2D mesh generated with a commercial meshing tool, while 3.36 shows two 3D meshes, block-structured and unstructured hex-dominant respectively, generated with OpenFOAM® *blockMesh* and *snappyHexMesh* utilities.

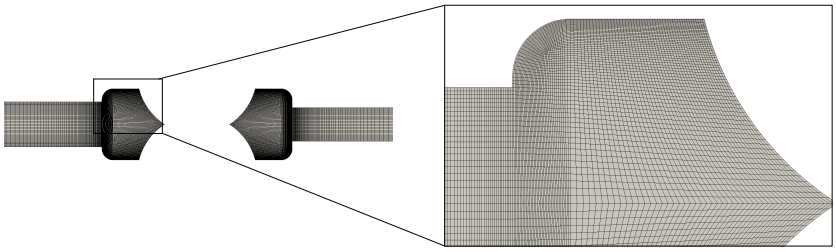


Figure 3.35: Generation of *fixedCells* mesh region with a commercial mesh generator

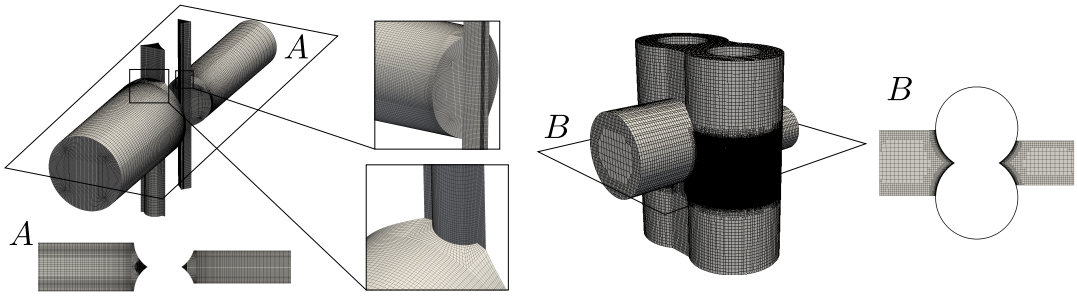


Figure 3.36: Generation of *fixedCells* mesh region with open-source applications: *blockMesh* (left) and *snappyHexMesh* (right)

Depending on the strategy followed for the topological changes in the mesh (detailed in the next section), the *fixedCells* region might need to include layers of cells between the circumferential boundaries of the rotating *gear1-gear2* and

the casing of the pump. Furthermore, if the axial leakage needs to be simulated, additional layers of fixed cells would be required at the top and bottom of the rotating gears, as it will be detailed in section 3.10.

3.9 Topological changes

The proposed strategy is based on the separation of the the entire mesh in three independently moving regions. The contact boundaries between these meshes must be treated as interfaces. In its current version, OpenFOAM-dev offers two alternatives to treat this kind of conditions: Arbitrary Mesh Interface (AMI) and Arbitrary Coupled Mesh Interface (ACMI). While the former can be used to communicate two simple non-conformal overlapping mesh boundaries, the latter allows for the specification of a second boundary condition in the non-overlapping region.

For the proposed method, communication between boundaries can be solved using exclusively AMI conditions or both AMI and ACMI conditions. However, even for the ACMI condition, OpenFOAM does not allow a given boundary to have two interface conditions (one interface in the overlapping part with one boundary and other interface condition in the non-overlapping part with another boundary). While the code could have been extended to allow this kind of behavior, a much easier solution was found, modifying the topology of the mesh in order to allow faces to move within boundary patches, We should note here that the only change of topology that we are considering is the motion of faces from one boundary to another, which is computationally fast since neither internal faces, nor cells or vertices are being modified.

3.9.1 AMI conditions

The first method, based on the use of AMI conditions, assumes that the region of *fixedCells* (see Fig. 3.1) includes also some layers of cells in the space between *gear1* and the pump casing, as well as between *gear2* and the casing (see Fig. 3.37).

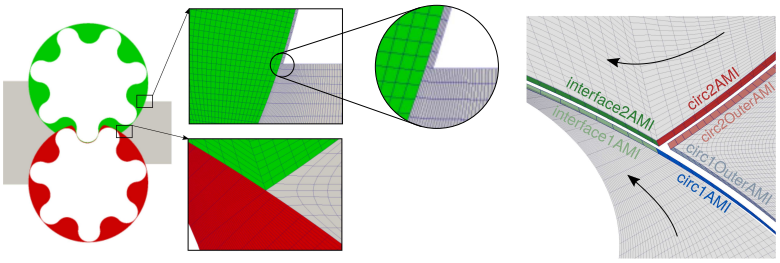


Figure 3.37: Boundary definition for the use of only AMI conditions

Adding layers of cells of the *fixedCells* region all along the casing enables the definition of an AMI condition between the moving *gear1* and *gear2* regions and the steady *fixedCells* region. The boundaries circ1AMI and circ1OuterAMI can be defined with AMI conditions. At the beginning of each time step, after moving the

mesh, interpolation weights are updated. While equations are solved, variables are interpolated using these weights from one boundary to the other each iteration. Exactly the same procedure is followed for circ2AMI and circ2OuterAMI, and for interface1AMI and interface2AMI since the same surface is covered by every pair of boundary patches (the boundaries fully overlap in couples). However, as the gears rotate (following the arrows in Fig. 3.37), faces located for instance in circ1AMI should eventually be moved to interface1AMI, since they are not in contact with faces from circ1OuterAMI any more, but they start overlapping faces of interface2AMI. When the projection of cell vertices to point1 and point2 is performed, the change of faces from one boundary patch to another is sudden. When this projection is not performed, there will be a period where the face of circ1AMI will be located at a position where it sees both faces from circ1OuterAMI and from interface2AMI. In that case, for the sake of simplicity, when the projection weight are below 20% Newman conditions (*zeroGradient*) are applied. The face is changed from one patch to the other when the x coordinate of the face center changes its position with respect to to the x coordinate of point1.

3.9.2 ACMI and AMI conditions

The second method, based on ACMI conditions, does not require the mentioned layers of cells near the casing of the pump, since ACMI allows two boundary conditions (an interface or, for example a *nonSlipWall* condition) for a given face in a patch. A weighted sum of the conditions is applied as a function of the overlapping between the patch and the other interface patch. This is performed by OpenFOAM® by duplicating the faces in the ACMI boundary (see Fig. 3.38). The previous circ1AMI is now two boundaries, circ1ACMI_blockage and circ1ACMI_couple, and the same happens with the other interface boundaries between *gear1*, *gear2* and *fixedCells* regions. If there is overlapping between circ1ACMI_couple and circ1OuterACMI_couple, boundary is treated like a normal AMI interface. However when a face in circ1ACMI_couple, which is duplicated in the patch circ1ACMI_blockage does not overlap circ1OuterACMI_couple, the condition to be applied is the corresponding to a *wall* patch, for the given variable.

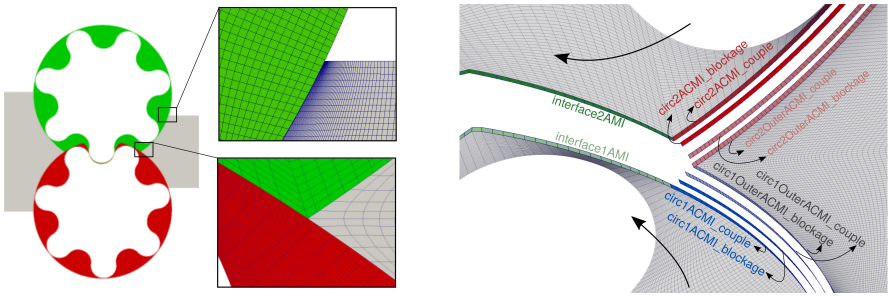


Figure 3.38: Boundary definition for the use of ACMI and AMI conditions

Regarding the topological changes, a slight modification is now required. As the gears rotate, faces which are for instance in circ1ACMI_couple, and duplicated in circ1OuterACMI_blockage, should be moved to interface1AMI (let us remark that the condition here is still AMI since no double boundary condition is required between

interface1AMI and interface2AMI). The two changing faces (in circ1ACMI_couple and circ1ACMI_blockage) are then transformed to one only face in interface1AMI.

The described strategy is applied in a similar way to the faces of circ2ACMI_couple and circ1ACMI_blockage. In the vicinity of point2 a similar topological change needs to be applied, since a single face from interface1ACMI would need to become two faces, one belonging to circ1ACMI_couple and other belonging to circ1ACMI_blockage.

3.10 Velocity mapping for axial leakage estimation

In the three-dimensional simulation, in a first degree of approximation, the top and bottom boundaries of *gear1* and *gear2* cell regions would behave as fixed walls. In fact those faces would be in contact with the compensation plates (shown in Fig. 1.2) and therefore, as the mesh rotates the boundary condition to be applied to those faces is simply a non-slip condition. However, if the axial leakage needs to be estimated, the minimum space between the rotating gears and the fixed compensation plates boundaries needs to be simulated. Fig. 3.39 represents an schematic of such a case (distances have been magnified for visualization purposes).

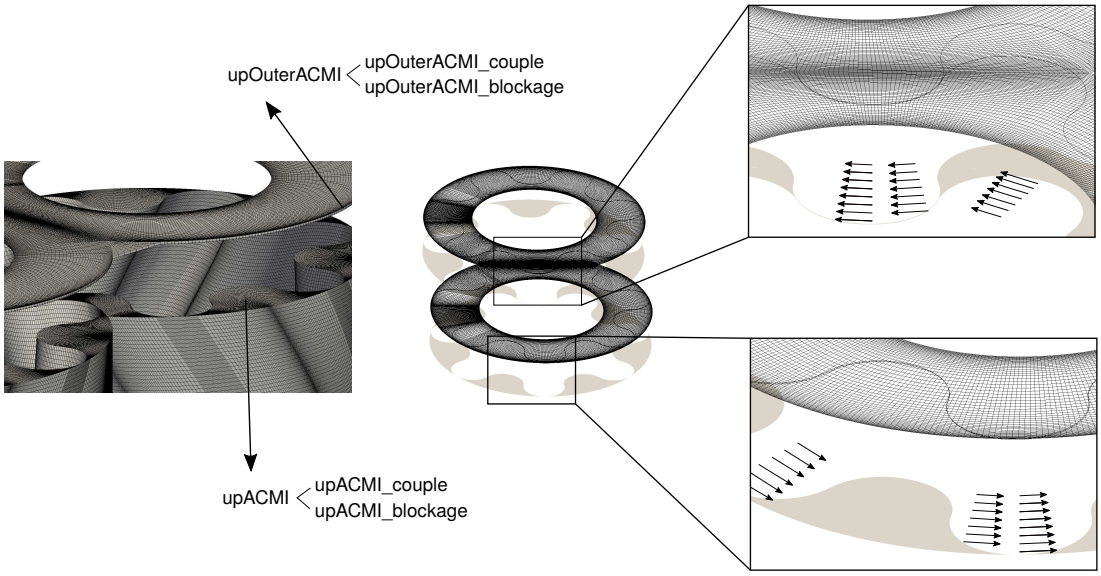


Figure 3.39: Simulation of the space between gears cells and compensation plates

The layers of cells separated from the upper boundary of the *gears* cells are static. They can be created with the rest of the *fixedCells* region. The problem arises when considering the boundary conditions of the bottom faces of this layers of cells (upOuterACMI) , which is in contact with the top boundary of the *gears* cells (upACMI). As the gears rotate, faces from upOuterACMI will either be in contact with faces from upACMI (they should then behave as an interface allowing the fluid to move through them) or they will not be in contact with any face, and should then behave as a boundary. In the latter case, the boundary condition to be applied would come from the rotating wall at the top of the gear. This can be treated in OpenFOAM® using the already mentioned ACMI conditions. However, in this case, the boundary condition to apply to the upOuterACMI_blockage faces is not

just a zero velocity. In fact if we pay attention to the gears meshing region (see Fig. 3.40), a face behaving as a moving wall in upOuterACMI_blockage would need a boundary condition that changes in time as the gears move.

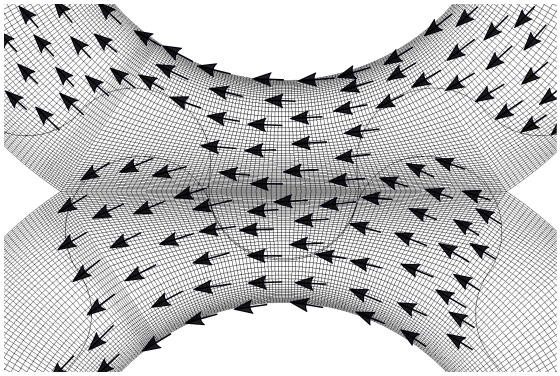


Figure 3.40: Velocity mapping

This has been implemented by determining, for each face center and run-time, whether its position lies on top of *gear1* or on top of *gear2* wall boundaries, and assigning as boundary condition, the value corresponding to a solid rotation with axis in the center of *gear1* or that of *gear2*.

3.11 Implementation in OpenFOAM®

The mesh motion strategy has been implemented in the *-dev* version of OpenFOAM, following the rules of structured coding and avoiding code repetition. The collaboration diagram for the generated classes is shown in Fig. 3.41.

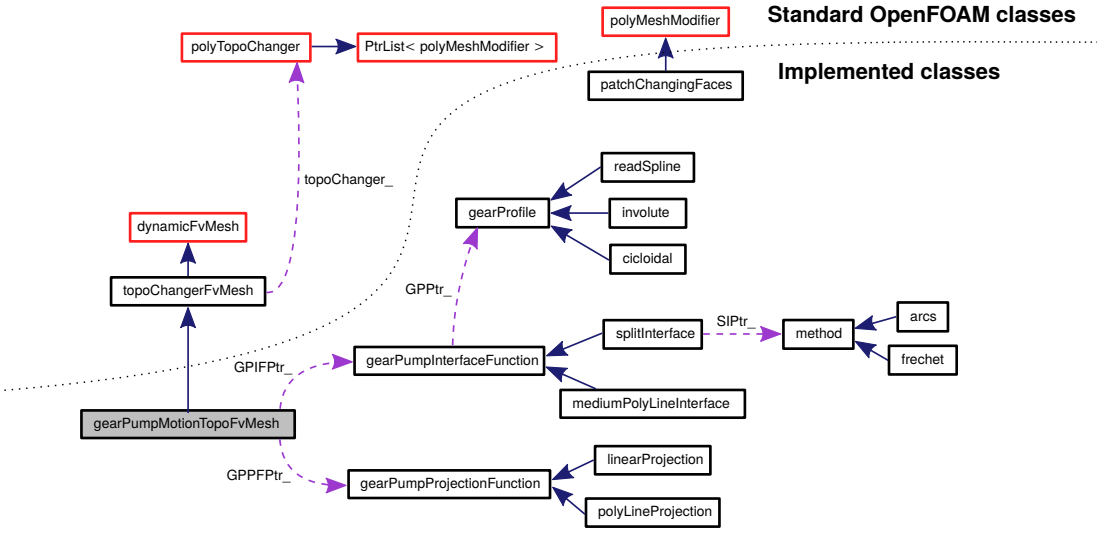


Figure 3.41: Collaboration diagram of the implemented classes and connection with base OpenFOAM classes

The main class gearPumpMotionTopoFvMesh derives from OpenFOAM® class topoChangerFvMesh. One of the benefits of doing this, is that any available solver

in OpenFOAM® can directly use the implemented classes without the need of being re-compiled. This piece of code gains therefore versatility.

In gearPumpMotionTopoFvMesh a run-time selection (the user specifies in the appropriate dictionary the class she/he would like to use) of a gearPumpInterfaceFunction class, that determines the 2D interface for a given rotation angle, and of a gearPumpProjectionFunction class that calculates the map of displacements to project the undisplaced mesh to the current interface, is performed. The profile is given by the class gearProfile that again can be selected run-time among any of the previously mentioned: involute (involute), cycloidal (cycloidal) or a point list given by the user (readSpline).

Regarding topological changes, OpenFOAM® base class (topoChangerFvMesh) contains a pointer to a list of polyMeshModifiers. The changes explained in section 3.9 are implemented in the patchChangingFaces class, directly derived from polyMeshModifier.

The operations performed by the main class can be summarized as follows. First the mesh is loaded and names of cellZones, boundaries, profile selection, and others are checked. Once the mesh has been checked, the labels of mesh points must be organized in order to simplify the mesh projection algorithm, as it will be explained later. Then the location of the undisplaced mesh points is determined. This is all done in the constructor of the class. During the simulation, mesh is updated by rotating the undisplaced points (saved in memory), determining for each *z-label* the 2D interface, and projecting the points as it has been explained previously. Once the point coordinates have changed the AMI and/or ACMI boundary conditions are checked and, if required, the topological changes are applied.

The movement of points can only be performed if their ordering is known. Since the mesh could have been renumbered, an algorithm has been implemented in order to create a hypermatrix of labels in which a point label can be found (see Fig. 3.42) by providing the cellZone it belongs to (*gear1* or *gear2*), the index of its *z-coordinate* (z_i), the tangential coordinate (t_i) and the radial coordinate (r_i). This search is only done once in the constructor of the class and information is stored and used thereafter. Even if the memory requirements slightly increase because of these variables, the point search algorithm can be slow, so that it is convenient to execute it just once.

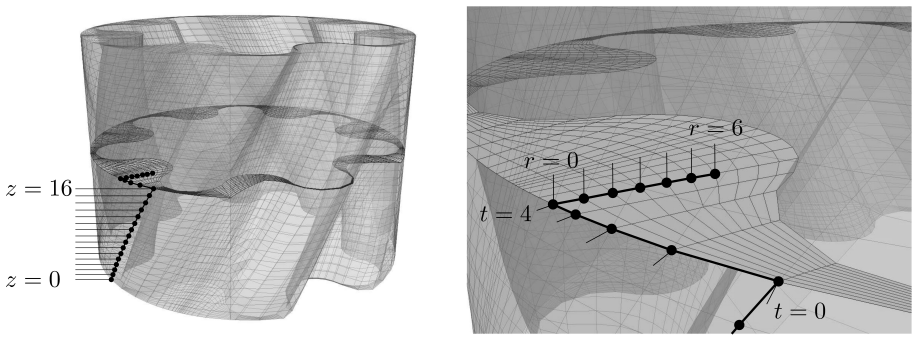


Figure 3.42: Point labels organization

Point search starts form a random point in the lowest *z-level* at the external radial boundary ($r = 0$). Considering this as a reference ($z = r = t = 0$), any other point can be located once the indexes are known following the solid line in Fig. 3.42. As shown in Fig. 3.43, this is performed by locating points connected to an initial one in the opposite face of a hexahedral cell. When the cell is not hexahedral due to the presence of non-conformal interfaces (see Figs. 3.7 and 3.6) face normal vectors ($\mathbf{S_{f,i}}$) are used to determine the direction in which to proceed with the point search.

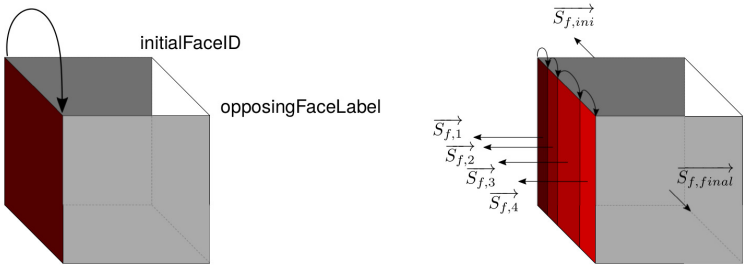


Figure 3.43: Point research algorithm

As mentioned before, the undisplaced mesh is used in the motion algorithm. Since the loaded mesh contains point coordinates which have been projected to an interface, the projection needs to be undone in order to determine the coordinates of the undisplaced mesh points. This is performed as follows (see Fig. 3.44). For each *z-level* the first point before point1 (pointLeft at tangential index $t_{pointLeft}$) and the first point after point2 (pointRight at tangential index $t_{pointRight}$) are identified. The coordinates of all points between $t_{pointLeft}$ and $t_{pointRight}$ (with any radial index $r = 0...r_{Max}$) are replaced by the corresponding undisplaced location, which is copied (rotated back) from the points before point that had not been projected.

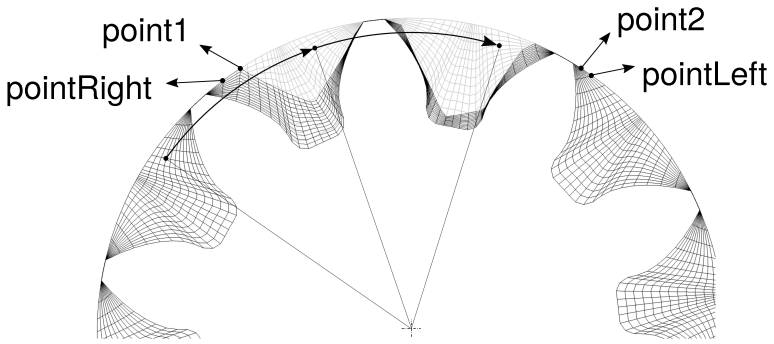


Figure 3.44: Creation of unprojected mesh in the class constructor

Having access to all mesh information, gearPumpMotionTopoFvMesh is also used to determine gear contact points locations (section 3.13), to map velocities for axial leakage calculation (section 3.10) and to provide additional functionalities required by some of the other classes exposed in Fig. 3.41.

3.12 Parallelization

The way it has been described, the algorithm would works as long as it has access to all the cells between a minimum and a maximum *z-level* and belonging to either one of the gear cellZones or both. Therefore the maximum level or parallelization obtained for a 2D mesh would be the one shown in Fig. 3.45.

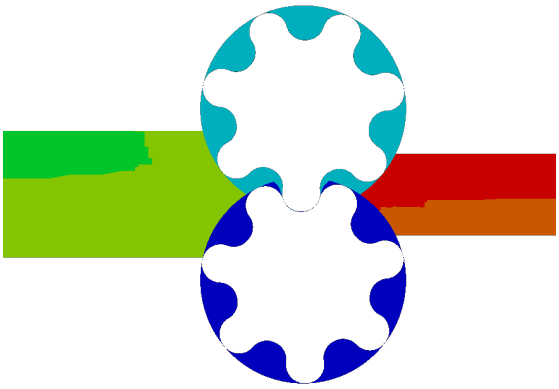


Figure 3.45: Parallelization of a 2D mesh

Each of the gears is kept in a different processor while any algorithm can be used to distribute the rest of cells of the *fixedCells* region.

When a 3D mesh is considered the parallelization can also be applied in the z direction leading to the structure of processors shown in Fig. 3.46. Again, the rest of the domain (*fixedCells*) can be decomposed following any other method.

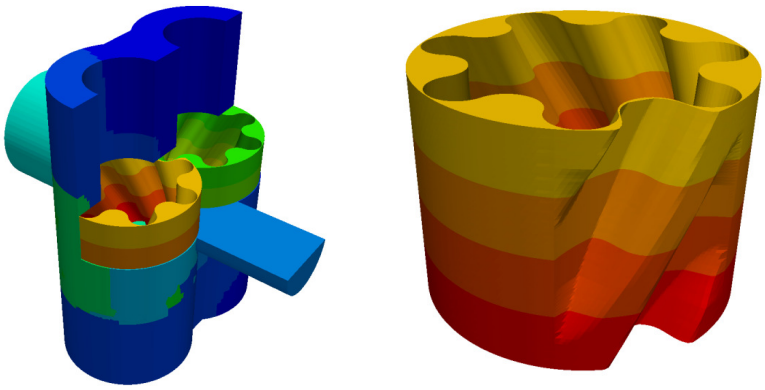


Figure 3.46: Parallelization of a 3D mesh

The only required modification appears when the projection of cell edges to line1 and line2 is considered in the case of 3D helical gears. In this case additional communication between processors is required. As shown in Fig. 3.47, projection or red edges leads to the situation shown in blue lines, where some points have been projected by one of the processors but not by the immediately upper one. Point synchronization between processors allows us to correct problematic points leading to the green edges and reaching the final situation in Fig. 3.47.

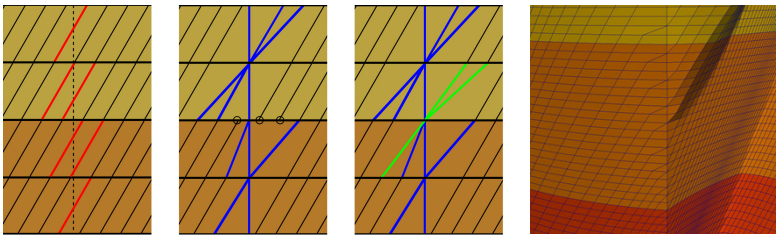


Figure 3.47: Point synchronization in 3D helical meshes with edge correction

The simple mesh decomposition method described so far does not produce a properly balanced parallelization in terms of shared faces and points between processors, even if the number of cells is homogeneously distributed between processors. The main issue behind any other parallelization method is that, following the algorithm used to determine the hypermatrix with point labels, we need the local processor to have information about all the cells contained in a given gear cellZone within two *z-levels* to create these matrices. If that is not the case, the local processor cannot access such information from the neighbor ones and even if it could, the process would be too time consuming. However a properly balanced decomposition is required, particularly when such a large case is under study.

The proposed solution is the following. Since we cannot access the required coordinates of some points from the local processor (because those points might belong to cells that are not contained in it), and we would not like to load the entire mesh from every local processor (that would be too time consuming), what we can do is previously define two files containing a hypermatrix with the global index structure (globalMatrixOfPoints) and coordinates (globalUndisplacedPoints) of the undisplaced points in *gear1* or *gear2* regions (not all the points of the mesh) and read those files from every processor. Once these files are read, each processor will determine how much of this information it needs (depending on the cells contained in it), reduce the size of these hypermatrices, and save the required information to be used in its motion algorithm.

Fig. 3.48 explains the global hypermatrix size reduction. Let us consider a case in which the parallelization proceeded so that one processor contains the cells in blue in the left of the figure while other contains the cells in yellow. globalMatrixOfPoints and globalUndisplacedPoints contain information of all the points that would need to be modified in the entire mesh (points belonging to *gear1* or *gear2* regions), and are represented for the entire gray meshes at the left of the figure. When the processor in blue reads these files it will see that the cells it contains belong all to *gear2*, but it contains cells in all the *zLevels*. In that case it will save the information of the hypermatrix of the entire *gear2*. On the other hand, the yellow processor needs only some of the *zLevels* of *gear2* and some others of *gear1*, and that is the information it will save, as shown in the right of Fig. 3.48. The mesh motion strategy proceeds so that the projection is only computed when needed, and the interface is only calculated for the *zLevels* it is required and only once for each *zLevel* (since the same interface is used to compute projection of cells in *gear1* and *gear2*).

An application has been created to determine or update the required files.

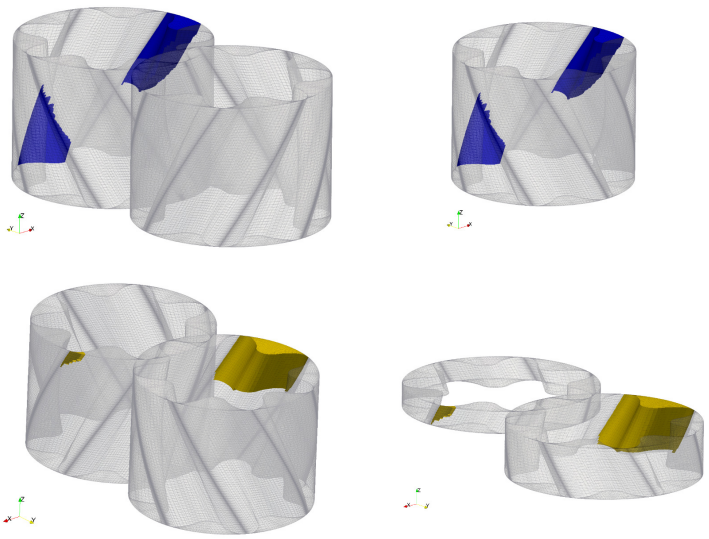


Figure 3.48: New parallelization method and global hypermatrix size reduction

With this new method any kind of parallelization method could be applied to our case.

3.13 Contact point treatment

One important problem in the use of the CFD approach for the simulation of gear pumps is the treatment of the contact point, whose hydrodynamic mechanism has itself concerned several authors [57, 58]. In real pumps, one of the gears drives the other, and the contact between them occurs in one or more points (one or more contact lines in the three-dimensional case). In fact, the contact ratio (ϵ_r) is typically higher than one, which means that before a pair of teeth separate, the contact between the next pair has started. This is particularly important when a 2D approach is considered, since decompression slots cannot be simulated and the increasing pressure in the trapped volume when $\epsilon_r > 1$ typically leads to numerical instabilities. The most realistic way of simulating the contact point would be to connect gear wall boundaries, reducing therefore to zero the spacing between the gears. This method however would tremendously complicate the dynamic mesh handling, since the topological changes to continuously attach and detach the boundaries would substantially increase the computational cost. Besides, the grid tolerances would make it difficult to ensure tangential contact between the moving boundaries. Other approach that can be found in literature [14] is based generating solid walls in inner faces of the domain. Typically this can be applied when the re-meshing method is used as mesh motion strategy, since the contact wall faces remain constant as the mesh deforms (the contact point is however separating from the theoretical position in that case), and the location of these faces can be corrected once a new mesh is generated substituting the current one. If such a method was to be applied in the mesh motion strategy described in this thesis, the topological change to reorder the faces (internal faces need to

become wall boundary faces) would again induce a much higher computational cost, significantly reducing the efficiency of the proposed method. Three possible methods are considered to overcome these difficulties.

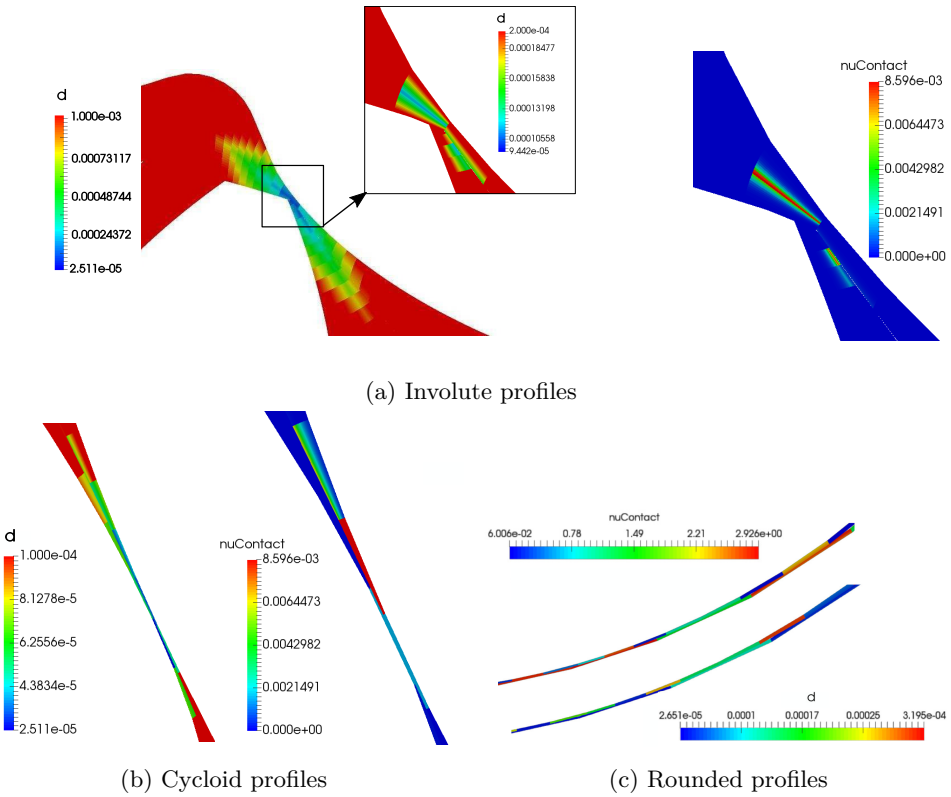
The first one is based on modifying the fluid viscosity in the vicinity of the contact point, and it is a common approach in literature [13, 26, 40]. If the liquid dynamic viscosity is artificially increased in a small region around the contact point, this would act as a solid spot, preventing the liquid to flow through, and therefore modeling the real behavior. In [27], the artificially increased viscosity linearly reduces as the distance of the cells to the contact point increases, until the viscosity returns to the property of the liquid at a given distance from the contact point. This approach does not allow the user many options to control how fast she/he wants to increase the viscosity when approaching the contact point. A better behavior can be obtained when the same idea is applied but a smoother decaying function is used. For instance, in [13], the contact point is simulated in OpenFOAM® by considering a varying viscosity, which behaves as described in Eq. (3.11).

$$\nu_{tot} = \nu + \nu_{contact}(d) = \max(\nu, \nu_{\max} f(d)) \tag{3.11}$$

where $\nu_{contact}$ is the additional viscosity, ν is the fluid kinematic viscosity, $\nu_{\max} = k_\nu \nu$ is the maximum value of the viscosity on the contact point ($k_\nu = 1000$ in our cases, is a user-defined constant) and $f(d)$ is the same blending function described in the mesh generation section in Eq. (3.1). In this case d represents the distance from the cell center to the furthest gear and $d_{\min}$ is the minimum distance between the gears, which in our case has been approximated by the *spacing* parameter. Considering Eq. 3.1 and Fig. 3.10 it is clear that k controls the slope of the curve, and therefore how fast we would like the artificial viscosity to increase, while k_d defines the extension of the contact point region, and therefore how far from the actual contact point we would like the artificial viscosity to be high. Too high values of k would lead to numerical instabilities due to the sudden and strong variation of viscosity between a cell and its neighbors, while k_d should be kept as small as possible, as long as the fluid is stopped from flowing through the contact point, to limit the region with artificially modified properties. For instance, $k = 5$ and $k_d = 3$ are suggested in [13]. The problem we found when the proposed method was applied, is related to the way d is calculated. Castilla *et al.* [13] determined d by using a modified version of wallDist OpenFOAM® class. In fact they used wallDist to determine the distance from every cell center to the closest face in the *gear1* wall patch (y_1) and the to *gear2* wall patch (y_2), defining $d = \max(y_1, y_2)$. Besides, in order not to consider an increased viscosity near the backlash angle, where gears are close to each other but liquid should not be impeded to flow, the x component of the vector pointing from every cell center to the driving gear was considered, determining when the proximity of gears occurs because of an actual contact point, and not because of the backlash clearance.

The described method has been implemented in OpenFOAM®. However, it was found that wallDist functions, based on calculating Poisson [34] or Eikonal [138] equations, are approximate methods, whose accuracy diminishes with the degree of mesh distortion; as soon as non-orthogonality or skewness of the mesh increases

the validity of the method fails. When the method is applied with the proposed mesh motion strategy, the presence of AMI interfaces also difficults the resolution of the PDE that is solved to determine y_1 and y_2, leading to oscillations.



(a) Involute profiles

(b) Cycloid profiles

(c) Rounded profiles

Figure 3.49: Instabilities in the calculation of d and $\nu_{contact}$ oscillations

In Fig. 3.49 the method has been applied to involute, cycloid and rounded profiles. The oscillations in the calculated distance d produces the same behavior in $\nu_{contact}$, which leads to numerical instabilities. The class wallDist adds also the possibility to correct the predicted distances for the boundary cells, where the approximate distance is replaced by an explicit geometrical calculation. However, the problem we are facing affects to inner cells, and therefore there is no parameter we can modify to correct the distance calculation using that method. A possible alternative is to consider a different method to determine the proximity of any cell to the contact point.

The proposed alternative (the second contact point treatment we consider in this thesis) consists in geometrically determining the location of the contact point (or points), the minimum distance between gears for each of them ($d_{contact}$), and manually calculating distances from every cell to it. Therefore in this case, the variable d is not the distance to the furthest gear wall boundary any more, but the distance from every cell center to the contact point under consideration. The parameter $d_{\min}$ in Eq. (3.1) is still *spacing* but the new artificial viscosity is now calculated according to Eq. 3.12.

$$\nu_{contact}(d) = k_\nu \nu f(d_{contact}) f(d) \tag{3.12}$$

where $f(d_{contact})$ will approach zero when $d_{contact}$ is large, and will approach unity when $d_{contact}$ is smaller than $k_d spacing$. In that case the viscosity blending for each cell will be given by $k_\nu f(d)$.

In the three-dimensional case, the location of the contact point is determined for each $zLevel$ and the distance for the cells with vertices in $zLevel$ and $zLevel+1$ is calculated with respect to the expected position of the contact point, the middle point between the contact determined at $zLevel$ and the one determined at $zLevel+1$.

Since the profile could have been defined by the user by a list of points, there is not analytical solution for the determination of the contact point, so a numerical strategy must be employed. The approach we will follow is based on the *fréchet* method that was described in section 3.5.2. As shown in Fig. 3.50, the minimum distance between profile nodes is found, and the middle point of the projection segment is considered as the contact point.

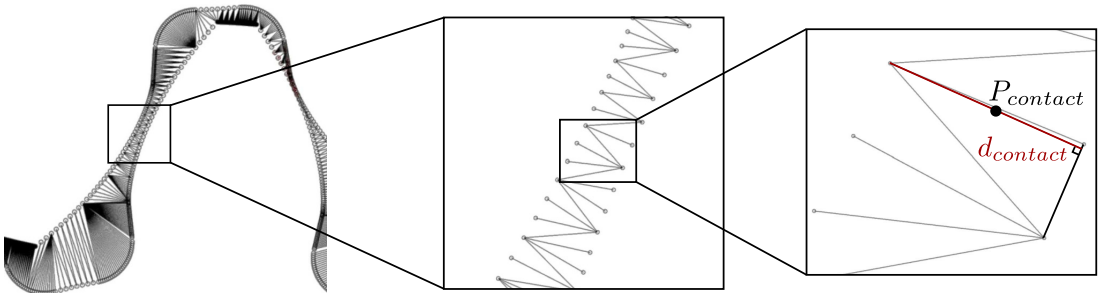


Figure 3.50: Contact point ($P_{contact}$) and minimum distance ($d_{\min}$) determination

The user can also decide to avoid the backlash zone (for the reasons explained before), and to consider more than one contact point. For instance, when the specified number of contact points to locate is three (see Fig. 3.51), the code will look for the three locations where distances are minimum, corresponding to different teeth pairs (avoiding in this case backlash zones), and will return the location of those points together with the minimum distance found in each case.

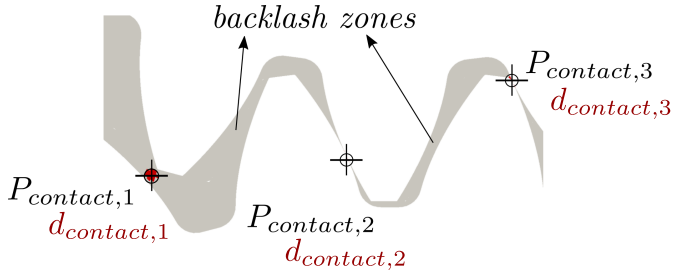


Figure 3.51: Location of more than one contact point, avoiding backlash zone

It is clear that, even if contact does not exist yet (as in $P_{contact,1}$) the point should be identified so that the artificially increased viscosity is not added suddenly when the contact starts, but instead it smoothly increases in time, as the distance reduces, to avoid numerical instabilities. Of course, this can occur as fast

as the the user decides by controlling k_d (artificially viscosity will start increasing when $d_{contact}$ starts approaching $k_d spacing$).

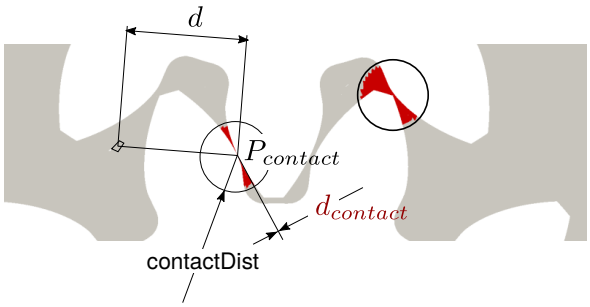
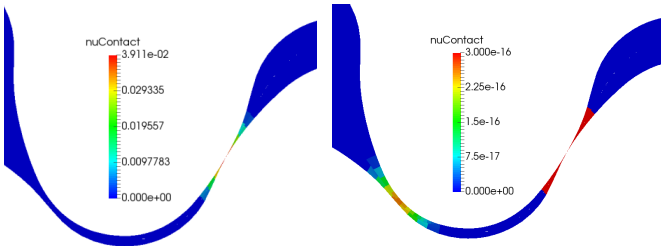


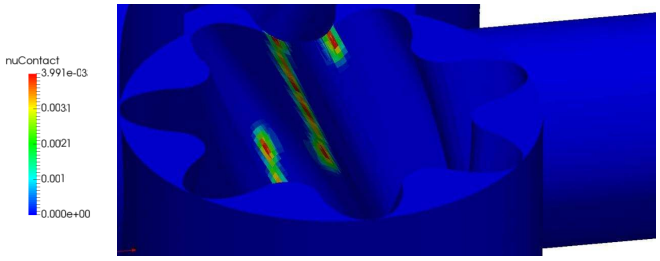
Figure 3.52: Definition of contactDist and other variables in the proposed contact point treatment formulation

Furthermore, in order to limit the cells in which the operation (Eq. (3.12)) is evaluated, the user can define a new parameter that restricts the application of the increased viscosity to the cells withing a given distance (contactDist) from the contact point (see Fig. 3.52). This is optional and added only to reduce the computational cost, since cells far from the contact point will not see any increased viscosity according to Eq. (3.12).

Fig. 3.53 shows the result of the application of the proposed method to a 2D and a 3D case. The reader can notice how the the problem with $\nu_{contact}$ oscillations disappears.



(a) Two-dimensional case: scaled to the global $\nu_{contact}$ range (left); scaled to a minimum $\nu_{contact}$ range (right)



(b) Three-dimensional case

Figure 3.53: Application of the proposed methodology to 2D and 3D cases

The third and last method we could consider, is based on assuming the existence of an artificial porous media in the contact region. While the previous algorithm is used to locate the cells within a given distance of the contact points, OpenFOAM® functions can be used to include explicit terms in the velocity equations that increase as a function of the maximum velocity reached in each cell. For instance the power law porosity model can be applied to the cells within a selected cellZone (the cells marked in red in Fig. 3.52). In that case the source term added to the velocity equation is represented by Eq. (3.13).

$$\mathbf{S} = -\rho C_0 |\mathbf{u}|^{C_1 - 1} \mathbf{u} \tag{3.13}$$

where C_0 and C_1 are user-defined constants that should be adjusted for the case under study.

3.13.1 Implementation

The additional term in velocity equations for this last case is completely handled by OpenFOAM® and all the required modifications are in this case related to selection of cells under the desired cellZone that should be updated every time step.

When considering the other two options, the implementation in OpenFOAM® tries again to maximize the applicability of the implemented methods to any given case. As a result, the additional terms that contain the extra viscous term, are added to the velocity equations by using fvOptions, always present in the declaration of the velocity equations in any OpenFOAM® solver. The implemented version works both in incompressible and compressible solvers. While the method proposed by Castilla *et al.* [13] does not required any additional information from the mesh main class (gearPumpTopoFvMesh), the new proposed method communicates with it to obtain the list of contact points and minimum distances. When the third method is used, the additional terms in the velocity equations are automatically handled by OpenFOAM® and all the required modifications are related to the selection of cells under the required cellZone that should be updated every time step. The collaboration diagram for the implemented classes is presented in Fig. 3.54.

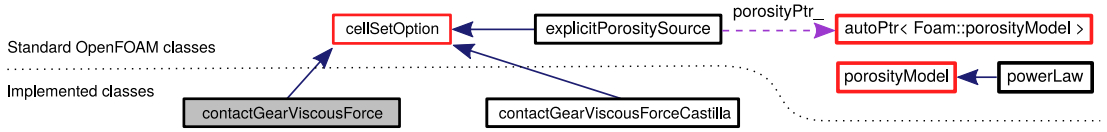


Figure 3.54: Collaboration diagram for contact point treatment implementations

3.14 Conclusions

The present chapter has introduced one of the main contributions of the thesis, a new mesh motion strategy for the simulation of 2D and 3D spur or helical gear pumps. Several options are given regarding the different algorithms involved in

the strategy. In particular, based on the experience in the use of the code, the separated interface methods is recommended: the *arcs* method has shown to be superior when cycloidal or involute tooth profiles are considered, while the *fréchet* was the preferred choice when smooth curved profiles were studied. Regarding the mesh projection process, unless the computational cost needs to be kept to a minimum, the spline projection should be applied, since it solves the problems derived from the use of linear projection methods. When considering the topological changes, if no extra complex modeling is added to the cases (compressibility, cavitation...) the use of ACMI-AMI conditions improves the convergence of the code and simplifies the meshing process. However the use of only AMI conditions might be required if the pump under study has circumferential grooves that require therefore a non-circular boundary in the casing. Regarding parallelization methods, for a small 2D case, the simple parallelization based on *gear1*, *gear2* and *fixedCells* cellZones is convenient. When a more complex and large 3D case is under study, the benefits of a better balanced parallelization method overcomes the increased memory requirements of saving additional files as explained on section 3.12. For the treatment of the contact point, among the implemented methods, the one proposed by Castilla *et al.* has shown some limitations for our case, and the one based on a explicit porosity source has shown to be difficult to adjust, therefore the second newly proposed method based on artificial viscosity is recommended and has been used in the cases studied in this thesis.

Chapter 4

Hybrid turbulence modeling

4.1 Introduction

The present chapter introduces a new developed hybrid turbulence model, based on the STRUCT approach proposed by [75]. As introduced in section 1.3, hybrid models try to bridge the gap between Unsteady RANS and LES. Despite the large amount of hybrid formulations available in literature, hybrid models have not achieved a widespread adoption in engineering applications yet [75], possibly because of the undesirable behavior observed in several applications (see for instance lack of grid convergence [41,127], or significant deviation from experimental results [24, 53, 123]). The scope of the proposed formulation is to address some of the shortcomings observed in available hybrid models, namely: robustness, grid convergence and ease of use. Unlike many of the available models, STRUCT does not base its activation on the size of the computational grid (Δ), but on locally computed flow quantities. The activation of models based on Δ recognizes zones of valid applicability of LES-like mode. On the other hand, STRUCT looks for those zones in which U-RANS assumptions loose their validity and therefore the resolution of turbulent structures is necessary.

In this chapter U-RANS and LES approaches for turbulence modeling will first be defined, together with their strengths and weaknesses. Then the rationale behind the proposed STRUCT model will be identified and the different strategies followed in the development of the model will be introduced. At the end of the chapter several of the tests performed for the validation of the approach will be presented, starting with the validation of the implementation of the baseline RANS method behind STRUCT and finishing with the application of the several versions of STRUCT.

4.2 Statistical description of turbulent flows

Let us consider the Navier-Stokes equations (NS) as presented in chapter 2: continuity equation (2.27) and velocity equations (2.28). Direct Numerical Simulations will directly solve the NS equations for all scales of turbulence, with no need for any additional model. However, as identified in the seminal work by Reynolds [110], even if all the information from the DNS simulation of an engineering applica-

tion was available, we would need to perform statistical operations to obtain useful information. From an engineering point of view, averaged variables contain the most relevant information and, therefore, different approaches have been proposed in order to directly solve for statistical quantities and reduce the computational burden, instead of performing the statistics after the simulation. This averaging process certainly involves a loss of information and a derived modeling error which will depend on the chosen closure. Nevertheless, the computational resources needed for a stringent DNS are so extreme that the application to simulations other than limited small geometries with moderate Reynolds numbers is impractical in most cases.

4.2.1 U-RANS models

Reynolds Averaged Navier Stokes equations are based on the definition of a Reynolds averaging operator [108] for any quantity ϕ, hereafter denoted as $\overline{\cdots}$, with certain properties [129]. For instance the operator must be linear $\overline{\phi+\psi} = \overline{\phi}+\overline{\psi}$, constant-preserving $\overline{a\phi} = a\overline{\phi}$, idempotent $\overline{\overline{\phi}} = \overline{\phi}$ and commutative with space and time translations: $\overline{\partial\phi/\partial x} = \partial\overline{\phi}/\partial x$ and $\overline{\partial\phi/\partial t} = \partial\overline{\phi}/\partial t$. The fluctuations with respect to this mean are denoted with a prime: $\phi = \overline{\phi} + \phi'$.

When applied to NS equations, non-linearities in the velocity equations generate an extra term, the Reynolds stress tensor $-\rho\overline{u_i'u_j'}$, that must be closed with the RANS model τ_{ij}^{RANS}. Assuming an incompressible fluid with constant density for simplicity, averaged velocity equations are shown in Eq. (4.1).

$$\frac{\partial(\rho\bar{u}_i)}{\partial t} + \frac{\partial}{\partial x_j}(\rho\bar{u}_i\bar{u}_j) = -\frac{\partial\bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial\bar{u}_i}{\partial x_j} + \frac{\partial\bar{u}_j}{\partial x_i}\right)\right] + \frac{\partial}{\partial x_j}\tau_{ij}^{RANS} \tag{4.1}$$

The operator can be defined in several manners: as the temporal mean for statistically steady flows, a finite-time temporal average for flows with slow variation of statistical properties (slower than the characteristic turbulent time-scales), or as a phase average for flows with some basic frequency. In general, the U-RANS notation is applied whenever the computation is time-dependent.

The problem arises when this approach is directly applied without any recalibration of coefficients to a transient simulation, aiming at resolving some of the unsteady features of the flow. When the unsteadiness is large in time and space (much larger than the turbulence fluctuations), such as slowly varying boundary conditions, the interaction between the significantly slower large-scale fluctuations and turbulence can be neglected, and hence the application of such approach is valid. However, when this difference in scales (hereafter called scale separation) does not exists, U-RANS approach is typically unsuitable to handle the situation. This is for instance the case of internal instabilities of the flow, such as bluff body flows in which large vortical structures disintegrate into smaller structures downstream. Even if a proper phase average can be defined for these cases, a significant amount of the interaction is unresolved by the U-RANS approach which can typically lead to errors even in the determination of the most insensitive quantity for these cases, the vertex shedding frequency, given by the Strouhal number (St) [113].

A variety of models exists, providing different formulation for the closure problem. Typically this leads to a classification [119] depending on whether the Reynolds stress tensor is directly related to the mean flow field by an algebraic relation (algebraic equation models or zero-equation models) or with the aid of transported parameters that define local turbulence quantities (transport equation models). Furthermore, transport models can be subdivided, attending the number of transport equations required for the problem closure. Two-equation models are currently the most popular choice both in academics and engineering applications [111].

The Reynolds stress tensor, τ_{ij}^{RANS} is a second-order symmetric tensor that can be decomposed in an isotropic and a traceless part:

$$\rho\overline{u_i'u_j'} = -\tau_{ij}^{RANS} = \frac{2}{3}\rho k\delta_{ij} + a_{ij} \tag{4.2}$$

where k is the turbulent kinetic energy ($k = \frac{1}{2}\overline{u_i'u_i'}$) and a_{ij} is the anisotropic stress. The isotropic part is typically considered inside a modified pressure term $\bar{p}_m = \bar{p} + \frac{2}{3}\rho k$.

A simple but common closure for a_{ij} consists in assuming that the effect of turbulence can be represented by an increased viscosity that enhances the transport of mass momentum and energy, the so-called Boussinesq eddy-viscosity assumption [7]. This is mathematically expressed as the assumption of a_{ij} being aligned with the resolved rate of strain tensor $\bar{S}_{ij} = \frac{1}{2}\left(\frac{\partial\bar{u}_i}{\partial x_j} + \frac{\partial\bar{u}_j}{\partial x_i}\right)$, through the eddy viscosity μ_t.

$$a_{ij} = -2\mu_t\bar{S}_{ij} = -\mu_t\left(\frac{\partial\bar{u}_i}{\partial x_j} + \frac{\partial\bar{u}_j}{\partial x_i}\right) \tag{4.3}$$

Despite its extended use, this carries fundamental limitations. In fact the assumption is not supported by experimental evidence [108] and does not hold even for the simplest cases. As a result, even an optimal evaluation of the eddy viscosity cannot translate into perfect results. However, its simplicity and numerical stability justifies its industrial success. As it will be explained in section 4.3.2, a higher order of approximation can be obtained by using the so-called non-linear eddy viscosity models.

In general, U-RANS models do not have strong requirements on mesh quality (much lower that those of LES). The required resolution is smaller both in the inner domain and near the wall, where there are not mesh constraints in spanwise and streamwise directions, and wall-functions can be applied to reduce the constraints in the wall-normal direction. U-RANS can also benefit from the flow symmetries, significantly reducing the computational cost. U-RANS converge asymptotically to the modeled solution (not to the DNS solution) and, once a reasonable convergence is achieved, they are not very sensitive to changes in the grid size [136]. Moreover, U-RANS models are numerically very stable due to the increased diffusivity. However, U-RANS models provide poor averaged results in complex geometries when compared to LES. They perform poorly when the scale separation assumption is not satisfied, and are unable to provide a description of unsteady flow quantities.

4.2.2 LES models

Large Eddy Simulation is based on the low-pass spatial filtering of the turbulent motions with a mathematically well-established formalism [77]. The LES filtering operator applied to a variable ϕ yields a smoothed counterpart $\langle\phi\rangle$ in which the scales smaller than the filter width (Δ_f) have been removed, and a fluctuating part ϕ'' so that $\phi = \langle\phi\rangle + \phi''$. This filtering is represented by a convolution product $\langle\phi\rangle = G \star \phi$, where G is the convolution kernel. Note that the filter G is not *a priori* a Reynolds operator: in general $\langle\langle\phi\rangle\rangle \neq \langle\phi\rangle$.

Similarly to the case of RANS modeling, the application of the filter to NS equations produces an additional term (see Eq. (4.4)) that needs to be modeled τ_{ij}^{LES}.

$$\frac{\partial(\rho\langle u_i\rangle)}{\partial t} + \frac{\partial}{\partial x_j}(\rho\langle u_i\rangle\langle u_j\rangle) = -\frac{\partial\langle p\rangle}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial\langle u_i\rangle}{\partial x_j} + \frac{\partial\langle u_j\rangle}{\partial x_i}\right)\right] + \frac{\partial}{\partial x_j}\tau_{ij}^{LES} \tag{4.4}$$

In most LES, the filtering operation is rather a concept behind the development of the method than a explicitly applied procedure [38]. The filter size Δ_f is usually set equal to the grid size Δ which therefore determines the cut-off scale in the filter. For historical reasons, $\tau_{ij}^{LES} = -\rho(\langle u_i u_j\rangle - \langle u_i\rangle\langle u_j\rangle)$ is typically referred to as subgrid-scale tensor. The strategy is to resolve most of the turbulent kinetic energy of the flow, mainly contained in the large scale fluid structures or eddies, which are highly affected by the geometry of the domain, while providing a suitable model for the subgrid-scale motions, which can be considered similar and statistically isotropic.

Most commonly used closures use again an eddy viscosity assumption. The SGS tensor is modeled by Eq. (4.5)

$$\tau_{ij}^{LES} - \frac{1}{3}\tau_{kk}^{LES}\delta_{ij} = 2\mu_{SGS}\langle S_{ij}\rangle \tag{4.5}$$

where $\langle S_{ij}\rangle$ is the strain rate tensor of the resolved scales $\langle S_{ij}\rangle = \frac{1}{2}\left(\frac{\partial\langle u_i\rangle}{\partial x_j} + \frac{\partial\langle u_j\rangle}{\partial x_i}\right)$. Typically the SGS viscosity explicitly is computed shown in Eq. (4.6).

$$\mu_{SGS} = \rho(C_m\Delta)^2 OP_m(\langle\mathbf{u}\rangle) \tag{4.6}$$

where C_m is the constant associated to the model m and OP_m is a differential operator acting on the resolved velocity field.

The use of SGS models has some inherent limitations. In order to capture most of the turbulent kinetic energy and the assumptions behind LES modeling to be valid (such us the isotropy of SGS scales), the cut-off wave number must lie within the inertial sub-range. Moreover, a wall-resolved LES simulation of industrial applications, typically characterized by large complex geometries and high Reynolds numbers, will not only limit the mesh resolution in the inner domain, but stringent near-wall grid spacing is required all in the wall-normal ($y^+ \approx 1$), streamwise ($\Delta x^+ \approx 100$) and spanwise ($\Delta z^+ \approx 30$) directions [116].

In general LES requires extremely fine good quality grids (sensitivity to cells aspect ratio and non-orthogonality is higher for LES than for U-RANS). Besides, even if LES should revert to DNS for a sufficiently refined grids, grid convergence

can be non-monotonic for complex flows [126]. Their quality can also strongly depend on the inlet resolved turbulence, for which synthetic eddy methods need to be used if no other turbulent data is available, and on the choice of spatial discretization schemes [86]. When the mesh is coarser than required, LES models will not revert to U-RANS, but will introduce significant errors and produce nonphysical behavior. On the other hand, the potential accuracy of LES is much higher than that of U-RANS models. Furthermore, LES can provide an accurate description of unsteady flow quantities, interesting for many engineering applications (thermal fluctuations, vibrations etc.), which cannot be properly determined by U-RANS.

4.2.3 Hybrid models

When considering Eqs. (4.1) and (4.4) the similarity between U-RANS and LES is evident. Furthermore, the structural similarity is obvious when considering models that use the eddy viscosity assumption (see Eqs. (4.5) (4.3)). The first hybrid concept, the Very Large Eddy Simulations (VLES) was proposed by Speziale [130] long ago, as a first attempt to extend the eddy-resolving capabilities to grids coarser than those needed in LES. A variety of models have been proposed ever since. However a universally accepted definition of what a hybrid model represents does not exist yet. While many approaches combine U-RANS and LES closures, some others with extended use (such as Scale-Adaptive Simulation (SAS) [91] or Partially Averaged NS (PANS) [49]) do not imply any LES closures, while still aim at achieving an intermediate behavior between U-RANS and LES. In general however, hybrid models are usually designed to work on coarser grids than those required for LES, and achieve a higher accuracy that what is possible through U-RANS. Some studies show that the computational cost of hybrid methods is reduced by a factor of $0.07Re^{0.46}$ when compared to LES [50]. The extensive list of approaches available in literature makes it difficult to summarize the possibilities of hybrid modeling. One example of categorization of hybrid models groups them in three types, depending on the hybridization approach [51]: *segregated* models, *interfaced* models and *blended* models.

In *segregated* models, also referred as *zonal decomposition* models [116], the global computational domain is divided into subdomains before the start of the computation. Some of them will be treated with the RANS method while others will be computed using LES, creating therefore an *embedded* LES (ELES). Grid resolution is relaxed in the RANS approach, which might also benefit from suppressed spatial directions thanks to the statistical homogeneity of the flow. The discontinuity between RANS and LES regions is artificially handled by coupling conditions, one of the biggest issues in this kind of approach [39]. Examples of such approach include the works presented by Quéméré *et al.* [109], Georgiadis *et al.* [43], Davidson [23], Bagget [3] or Tucker *et al.* [137]. However the most famous approach in this category is the Detached Eddy Simulation, proposed by Spalart *et al.* [100, 125, 126], and its further corrections: Delayed Detached Eddy Simulation (DDES) [124], Improved-DDES [54, 122], Shielded-DES [90], Stress Blended Eddy Simulation [89]. In the same category we group models that split the use of RANS and LES, not in the space domain but in the frequency domain, the so

called Nonlinear Disturbance Equation Method proposed by Morris *et al.* [20,94]. The low-frequency or steady part is modeled by RANS, while the high-frequency fluctuating part is computed by LES.

Interface and *blended* models are sometimes grouped in the same category, *universal* models [116]. These models aim at reducing the resolved turbulent kinetic energy dissipation induced by the U-RANS model, leading to a weaker damping of high frequencies, which yields more irregular LES-like flow fields. While *blended* models typically use an interface function (f_I) for the transition between U-RANS and LES: $\tau_{ij} = f_I \tau_{ij}^{RANS} + (1 - f_I)\tau_{ij}^{LES}$ (most clear example is the formulation proposed by Germano [45]), *blended* models rescale the U-RANS stress tensor by a function (f_B) without the explicit consideration of the LES tensor: $\tau_{ij} = f_B \tau_{ij}^{RANS}$ (even if f_B can be selected so that LES equations are retrieved). An example of this last case is given by Speziale [131, 132].

Other possible categorization groups together those models in which no grid size dependence exists in their formulation. They were identified by Fröhlich *et al.* [39] as *second generation* U-RANS models, as they can be interpreted as extensions of the U-RANS closures. This is of major important in this thesis, since the developed model belongs to this category. Two main examples should be cited inside this group, PANS and SAS.

Partially Averaged Navier Stokes models (PANS) were first introduced in 2003 [49] and then described extensively by Girimaji [48]. The main idea is to decompose flow variables in a resolved and a residual part using an arbitrary filter. This is achieved by considering two user-defined parameters (f_k,f_ϵ), that describe the ratio between residual (unresolved) and total turbulent kinetic energy ($f_k = k_u/k$) and turbulent dissipation rate ($f_\epsilon = \epsilon_u/\epsilon$). Transport equations can then be derived for the unresolved k_u and ϵ_u. Note that these equations resemble the standard $k-\epsilon$ equations (4.8) with modified turbulent Prandtl numbers (σ_{k_u} and σ_{ϵ_u}) and a modified constant in the dissipation term of ϵ_u equation, that become a function of f_k and f_ϵ. Using the eddy-viscosity assumption, the turbulent viscosity is in this case computed as $\mu_t = C_\mu \rho k_u^2/\epsilon_u$. Other variants of a similar idea can be found in the Partially Integrated Transport Model (PITM) by Schiestel and Dejoan [117] or the Partially Resolved Numerical Simulation by Shih and Liu [121].

Scale-adaptive Simulations (SAS) is other example worth mentioning, due to its successful application to a variety of industrial flows. Developed by Menter *et al.* [91], the model was born when revisiting the $k - kL$ model by Rotta [114]. It was found that the transport equation for the turbulent length-scale introduces the second derivative of the velocity field and the Von Karman length scale (L_{vK}) appears as a natural length-scale. L_{vK} is formulated based on the ratio of parameters corresponding to the first and second derivatives of the spatially resolved velocity. For the sake of brevity let us only considered the application of the SAS to the $k-\omega$ SST model [33]. In the SST-SAS model an extra production term is added to the ω equation. This extra term, typically named Q_{SAS} increases the turbulent dissipation frequency proportionally to $(L/L_{vK})^2$, where L is the integral length scale ($L = \sqrt{k}/\beta^{*0.25}\omega$), avoiding the regions with large gradients of k or ω (such as boundary layers where the U-RANS behavior should be recovered).

The activation of the model, and the subsequent decrease of turbulent viscosity (derived from the increased dissipation frequency production term), occurs when the modeled integral length scale exceeds the Von Karman length scale, a length scale of the resolved velocity. In particular the model will activate in regions with strong instabilities such as massively separated flows, while it will revert to the standard $k-\omega$ SST in regions of stationary flow. One of the inherent limitations of this model is that SAS will not produce a correct physical description unless significant unsteadiness occurs in the resolved fields. In fact, the model was shown to perform worse than the base $k-\omega$ SST in cases like an asymmetric diffuser [24], where the mild separation caused by a slight adverse pressure gradient represents a huge challenge for hybrid models.

4.3 STRUCT

In the present section the STRUCT formulation will be described. First of all, the rationale and objectives of the model will be introduced, together with the desirable behavior. Then the baseline RANS model will be explained in detail, paying attention to the capabilities of NLEVM over classical linear models. In the next section the different hybridization approaches proposed by Lenci [75] and the alternative variants developed for this thesis will be defined. Finally a validation of the implementation of the RANS model in simple academic test cases will be presented, followed by more challenging test cases where different hybrid closures will be studied, resembling the physical phenomena that they should be able to model when applied to a gear pump.

4.3.1 Rationale

STRUCT aims at improving the robustness and grid convergence of available hybrid models. The activation of STRUCT will not be based on the computational grid size belonging therefore to the previously introduced *second generation* U-RANS models. In fact, unlike many of the hybrid approaches in literature, STRUCT will not try to identify the regions in which mesh refinement is enough to allow the activation of the LES-like behavior. On the contrary, the activation will be based on identifying the zones where poor performance of the baseline U-RANS model is expected, zones where resolution of flow structures is necessary. For instance, when considering the previously introduced concept of scale separation, U-RANS is expected to work properly when a sufficient separation exists, but wrong behavior will be predicted otherwise (see section 4.2.1). STRUCT will work by comparing resolved and modeled flow scales, and the overlap of these two scales will trigger the activation of the model.

The objectives followed in the development of STRUCT were pointed out clearly by [75] in the graphic representation shown in Fig. 4.1.

In Fig. 4.1 potential accuracy identifies the capabilities of the models to produce accurate time-averaged fields, while flow description is related to the ability of the models to resolve complex spectral content of unsteady flows, one of the main limitations of U-RANS approaches. Ease of use includes for instance, re-

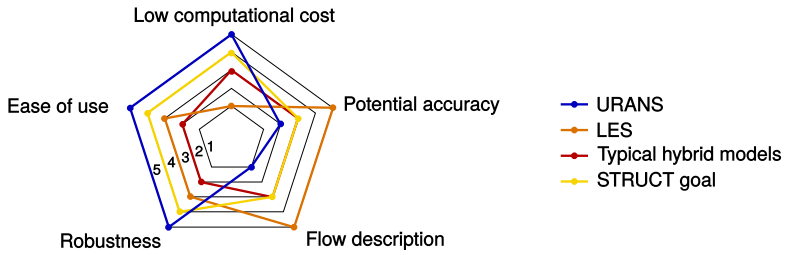


Figure 4.1: STRUCT goals compared to other classical approaches

ducing the necessity of case-dependent parameters, complex boundary conditions (such as synthetic turbulent conditions for LES) or mesh quality constraints. Robustness, on the other hand, refers to the capability to tolerate small perturbations in boundary conditions or numerical methods without this causing significant differences in the results. The low computational cost is the main goal of hybrid models as explained before, improving U-RANS results without incurring the the mesh refinement limitations of LES. As it can be inferred from Fig. 4.1, the goal of STRUCT is not to improve the flow description or accuracy with respect to other available hybrid models, but to focus on reducing the computational cost, ease of use and robustness.

4.3.2 NLEV RANS model

The accuracy of a hybrid model can substantially depend on the choice of the underlying RANS model. As mentioned in section 4.2.1 one of the limitations of many U-RANS models (and also LES closures) is the isotropic eddy-viscosity assumption. The assumption of a_{ij} being aligned with $\bar{S}_{ij}$ (see Eq. (4.3)) causes very limited accuracy in complex flows including among others those involving swirl [66], impingement [1], strong curvature [65] or turbulence-induced secondary motions [93]. Besides classical linear models are obviously unable to predict anisotropy even in simple cases [6]. It has been even proposed that linear U-RANS should be replaced completely with NLEVMs in industrial applications [60]. The choice of a nonlinear RANS closure is also common in hybrid modeling from the very first hybrid approaches [130] and high-order models in LES have also proven to increase accuracy [25].

As pointed out by Pope [107], the Boussinesq assumption has two downsides, the limitations of an isotropic approach and the eddy-viscosity assumption itself. Therefore in 1975 he introduced the so-called nonlinear eddy viscosity models (NLEVM) to correct the former issue. These models are based on extending the eddy-viscosity assumption by assuming that the anisotropy stress tensor responds to the more general form described in Eq. (4.7).

$$a_{ij} = \rho\overline{u_i'u_j'} - \frac{1}{3}\rho\overline{u_k'u_k'}\delta_{ij} = -2\mu_t\bar{S}_{ij} + f(\bar{S}_{ij}, \bar{\Omega}_{ij}, k, \epsilon...) \tag{4.7}$$

where f describes the nonlinear stress-strain relation, and is expressed through a coordinate-invariant polynomial function of k, ϵ and the averaged velocity gradient tensor (contained in the strain $\bar{S}_{ij}$ and rotation $\bar{\Omega}_{ij} = \frac{1}{2}\left(\frac{\partial\bar{u}_i}{\partial x_j} - \frac{\partial\bar{u}_j}{\partial x_i}\right)$ tensors).

This explicit modification increases only slightly the computational cost of the linear counterpart (no additional Partial Differential Equation (PDE) needs to be solved, but provides increased accuracy in complex flows). Multiple variants exist in literature with different definitions of f. Original models by Shih, Zhu and Lumley [120], Lien *et al.* [79] and Speziale [128] are some of the most commonly used.

In the case of STRUCT the cubic NLEVM proposed by Baglietto and Ninokata [4, 5] has been selected. On the base of a physical interpretation of [120] they reformulated the model coefficients to extend its applicability. The model uses the well know two-equation $k-\epsilon$ model in a low Reynolds version. Transport equations for k and ϵ are shown in Eq. (4.8).

$$\begin{aligned}\frac{\partial(\rho k)}{\partial t}+\frac{\partial}{\partial x_j}\left(\rho\bar{u}_j k\right)&=\frac{\partial}{\partial x_j}\left[\left(\mu+\frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right]+P_k-\rho\epsilon\\ \frac{\partial(\rho\epsilon)}{\partial t}+\frac{\partial}{\partial x_j}\left(\rho\bar{u}_j \epsilon\right)&=\frac{\partial}{\partial x_j}\left[\left(\mu+\frac{\mu_t}{\sigma_\epsilon}\right)\frac{\partial \epsilon}{\partial x_j}\right]+C_{\epsilon_1}f_1\frac{\epsilon}{k}P_k-C_{\epsilon_2}f_2\rho\frac{\epsilon^2}{k}\end{aligned} \tag{4.8}$$

where P_k is the production of turbulent kinetic energy (see Eq. (4.9)) and the turbulent viscosity (μ_t) is computed according to Eq. (4.10).

$$P_k=-\rho\overline{u_i'u_j'}\frac{\partial\bar{u}_i}{\partial x_j} \tag{4.9}$$

$$\mu_t=\rho f_\mu C_\mu\frac{k^2}{\epsilon} \tag{4.10}$$

The low Reynolds formulation is achieved by the functions introduced in ϵ equation (f_1, f_2) and the turbulent viscosity damping function (f_μ). f_1 and f_2 were selected from the example of [79] based himself in the length-scale formulation of Norris and Reynolds [101] and read Eqs. (4.11) and (4.12), while f_μ was defined following [4], as shown in Eq. (4.13)

$$f_1=1+\frac{1}{P_k}1.33\left(1-0.3e^{-R_t^2}\right)\left(P_k+2\mu\frac{k}{y^2}\right)e^{-0.00375Re_y^2} \tag{4.11}$$

where $R_t=\frac{k^2}{\nu\epsilon}$ and $Re_y=\frac{y\sqrt{k}}{\nu}$

$$f_2=1-0.3e^{-R_t^2} \tag{4.12}$$

$$f_\mu=1-e^{-0.029Re_y^{0.5}-0.00011Re_y^2} \tag{4.13}$$

The model is also assembled respecting the realizability constraints: non-negativity of the turbulent normal stresses ($\overline{u_i'u_i'}\geq 0$, where index summation rule does not apply) and Schwarz inequality between any fluctuation ($\overline{u_i'u_i'}\,\overline{u_j'u_j'}\geq\overline{u_i'u_j'}^2$). This is expressed as a non constant C_μ coefficient as shown in Eq. (4.14).

$$C_\mu=\frac{C_{a0}}{C_{a1}+C_{a2}\bar{S}^*+C_{a3}\bar{\Omega}^*} \tag{4.14}$$

where $\bar{S}^*$ and $\bar{\Omega}^*$ are dimensionless parameters, function of the averaged strain and rotation tensors, as shown in Eq. 4.15

$$\bar{S}^* = \frac{k}{\epsilon}\sqrt{2\bar{S}_{ij}\bar{S}_{ij}} \ , \ \bar{\Omega}^* = \frac{k}{\epsilon}\sqrt{2\bar{\Omega}_{ij}\bar{\Omega}_{ij}} \tag{4.15}$$

The nonlinear strain-stress relation is then defined by Eq. (4.16):

$$\begin{aligned} a_{ij} = & -2\mu_t\bar{S}_{ij} \\ & + 4f_\mu k\left(\frac{k}{\epsilon}\right)^2 \frac{1}{C_{nl6} + C_{nl7}\overline{S}^{*3}}\Bigg[\quad C_{nl1}\left(\bar{S}_{ik}\bar{S}_{kj} - \frac{1}{3}\bar{S}_{kl}\,\bar{S}_{kl}\delta_{ij}\right) \\ & \qquad + C_{nl2}\left(\bar{\Omega}_{ik}\bar{S}_{kj} + \bar{\Omega}_{jk}\bar{S}_{ki}\right) \\ & \qquad + C_{nl3}\left(\bar{\Omega}_{ik}\,\bar{\Omega}_{jk} - \frac{1}{3}\bar{\Omega}_{kl}\bar{\Omega}_{kl}\delta_{ij}\right)\Bigg] \\ & + 8f_\mu k\left(C_\mu\frac{k}{\epsilon}\right)^3\Bigg[\quad C_{nl4}\left(\bar{S}_{kI}\bar{\Omega}_{lj} + \bar{S}_{kj}\bar{\Omega}_{li}\right)\bar{S}_{kl} \\ & \qquad + C_{nl5}\left(\bar{S}_{kl}\bar{S}_{kl} - \bar{\Omega}_{kl}\bar{\Omega}_{kl}\right)\bar{S}_{ij}\Bigg] \end{aligned} \tag{4.16}$$

Well assessed values of coefficients appearing in the $k-\epsilon$ equations (Eq. (4.8)) are taken from Launder and Spalding [74] (as shown in Table 4.1), while the coefficients appearing in the nonlinear formulation (Eqs. (4.16) and (4.14)) are shown in Table 4.2

Table 4.1: Standard $k-\epsilon$ coefficients

σ_k	σ_ϵ	C_{ϵ_1}	C_{ϵ_2}
1.0	1.3	1.44	1.92

Table 4.2: NLEVM coefficients

C_{a0}	C_{a1}	C_{a2}	C_{a3}	C_{nl1}	C_{nl2}	C_{nl3}	C_{nl4}	C_{nl5}	C_{nl6}	C_{nl7}
0.6667	3.9	1.0	0.0	0.8	11.0	4.5	-5.0	-4.5	1000	1.0

Hereafter the baseline model will be referred to as BagliettoNLEVM.

4.3.3 STRUCT models

As introduced in section 4.3.1 STRUCT hybrid formulation bases its activation in the comparison of resolved and modeled scales. In particular frequency scales have been chosen. A resolved scale (f_r) is computed from the resolved velocity field, while several options are considered for the definition of the modeled frequency (f_m). The model is then activated by following Eq. 4.17.

$$\mu_t = \begin{cases} \mu_t^{RANS} & f_r < f_m \\ \phi \mu_t^{RANS} & f_r \geq f_m \end{cases} \tag{4.17}$$

where μ_t^{RANS} is computed according to Eq. (4.10) and ϕ is a reduction parameter that represents how much of the total turbulent kinetic energy is resolved in the selected structures.

The frequency scale for the resolved structures f_r is then defined as a function of the second invariant of the resolved velocity gradient tensor ($\overline{II}$) following Eq. (4.18).

$$f_r = \sqrt{|\overline{II}|} \tag{4.18}$$

where $\overline{II}$ is determined as:

$$\overline{II} = -\frac{1}{2} \frac{\partial \bar{u}_i}{\partial x_j} \frac{\partial \bar{u}_j}{\partial x_i} \tag{4.19}$$

The second invariant has been used by some authors to identify coherent structures [30,115], and is able to describe regions of poor U-RANS performance when reaching the rapid distortion limit [75]. When used to activate the model, it also allows us to describe regions of poor U-RANS performance because of the non existence of scale separation. Besides, it will avoid the model activation in regions of simple shear flow (such as near the walls), since f_r will vanish impeding the model activation.

Several versions of STRUCT have been proposed by [75], depending on the different definitions of f_m and ϕ and they will be presented in the following sections.

Controlled STRUCT

The simplest version uses user-defined constant values of f_m and ϕ. While an estimation of f_m can be obtained from a representative value of ϵ/k in a previously computed U-RANS simulation, a robust value of $\phi = 0.6$ is sufficient to enhance accuracy over U-RANS [76]. In this case the model lacks completeness but it serves to provide preliminary results.

STRUCTL

Increasing the degree of completeness of the model, Lenci [75] proposed a local geometric averaging procedure for the determination of f_m. Based on a Taylor series expansion, f_m is determined as $f_m = 1/t_m$ where t_m is computed following Eq. (4.20).

$$t_m = e^{\ln t_{m0} + \min\left(\max\left(\frac{R^2}{10}\nabla^2(\ln t_{m0}), -\ln 2\right), \ln 2\right)} \tag{4.20}$$

where $t_{m0} = k/\epsilon$ is related to the locally computed modeled turbulent time scale and R represents the averaging length, proportional to the modeled turbulent length scale, as shown in Eq. (4.21).

$$R = C_R \frac{k^{3/2}}{\epsilon} \tag{4.21}$$

where C_R is a user-defined constant, and $C_R = 2$ is suggested in [75].

In equation Eq. (4.20) we can appreciate that large variations of t_m with respect to t_{m0} are avoided by limiting the predicted value of t_m ($0.5t_{m0} \leq t_m \leq 2t_{m0}$). In this version however, ϕ parameter must still be provided by the user, and a sufficiently small value (close to zero) is recommended $\phi = 10^{-10}$.

STRUCTT

Last step of completeness allows the closure of both f_m and ϕ. STRUCTT uses the generalized space-time Lagrangian average (based on the work from [87]) to provide a transport equation (Eq. (4.22)) for t_m (let us recall $f_m = 1/t_m$).

$$\frac{\partial(t_m)}{\partial t} + \frac{\partial}{\partial x_j}\left(\bar{u}_j t_m\right) = \frac{\partial}{\partial x_j}\left[\left(\frac{L^2}{T}\right)\frac{\partial t_m}{\partial x_j}\right] + S_{t_m} \tag{4.22}$$

where T and L are respectively length and time modeled scales, locally defined as shown in Eq. (4.23).

$$L = \sqrt{0.09}\frac{k^{3/2}}{\epsilon} \;\;,\;\; T = \frac{1}{\beta}\frac{k}{\epsilon} \tag{4.23}$$

The source term in Eq. (4.22) is computed as shown in Eq. (4.24).

$$S_{t_m} = \min\left(\max\left(\frac{1}{T}(t_{m0} - t_m), -\frac{2t_m}{\Delta t}\right), \frac{2t_m}{\Delta t}\right) \tag{4.24}$$

where Δt is the simulation time step size, and is used to limit the change of t_m in its transport equation, increasing the stability.

Using the initial conditions $t_m|_{t=0} = t_{m0}|_{t=0}$ and assigning $t_m = t_{m0}$ at boundaries, Eq. (4.22) allows us to determine the modeled frequency f_m. The control parameter ϕ is eliminated by replacing Eq. (4.17) with Eq. (4.25):

$$\mu_t = \mu_t^{RANS} = \min\left(\frac{f_m}{\alpha f_r}, 1\right) \tag{4.25}$$

Recommended values for α and β are given in Table 4.3

Table 4.3: STRUCTT coefficients

α	β
1.35	0.01

Other variants

The previously described versions were proposed by Lenci [75]. As it will be shown in the results section, some downsides were found when testing these variants, and some alternatives have been proposed.

First of all, the controlled version (STRUCT) is meaningful for initial tests, but as explained before it is a non-closed model. The determination of an appropriate ϕ parameter for a given case is not straight forward. Besides, a constant value of

f_m might not be appropriate for a case where very different turbulent scales can be found in different parts of the domain under study. Therefore we can center the discussion in the other two methods proposed by Lenci.

When considering STRUCTL, it was found that the the extension of the geometric averaging distance (R) was sometimes excessive (see Fig. 4.27). In fact the consideration of only local information to predict the behavior of t_m somewhere far from the local cell, leads to overestimation and underestimation of the geometric average t_m that gets limited by the explicit bounds in Eq. (4.20), leading to an nonphysical checkerboarding activation of the model.

Some alternatives have been considered to deal with this problem. One option is to consider a more local average. In the model we will call **STRUCTL_V** hereafter, integration length is given by the size of the local cell $R = \sqrt[3]{V}$ where V is the volume of the local cell, and t_m is still computed from Eq. (4.20).

Alternatively, one may think of using an arithmetic average instead of the geometric averaging method described in [75]. By considering the polynomial Taylor expansion of t_m, the modeled frequency could be determined by:

$$t_m = t_{m0} + \frac{R^2}{10}\nabla^2 t_{m0} \tag{4.26}$$

where again R could be determined by the integral length scale $R = C_R k^{3/2}/\epsilon$, in the model we will call **STRUCTAt** or by the local cell size $R = \sqrt[3]{V}$ in the model we will refer to as **STRUCTAt_V**.

The procedure could also be applied directly to the modeled frequency and not to the time scale, following Eq. (4.27).

$$f_m = f_{m0} + \frac{R^2}{10}\nabla^2 f_{m0} \tag{4.27}$$

which leads to **STRUCTAf** and **STRUCTAf_V** models, following the same nomenclature explained above. Note that f_{m0} is defined as $1/t_{m0}$. Following the idea of reducing the averaging integration length, the local average could also be determined from a face reconstruction procedure, where only neighbor cells information is considered as defined in the model **STRUCTF** (see Eq. (4.28)).

$$t_m = \frac{1}{A}\sum_i A_i (t_{m0})_i \tag{4.28}$$

where A is the total area of the cell and $(t_{m0})_i$ is the value of local time scale t_{m0} interpolated at face i, of area A_i.

The problem of using face averaging or a small integration length is that, first of all, the result will be dependent on the mesh size which is something we would like to reduce. Second, for a small enough grid, the process has very limited effect, and f_m will be close to f_{m0} so that we are actually not considering any averaging of the modeled scales. Besides, ϕ parameter must still be fixed by the user: the model therefore remains open.

Some attempts to close these versions have been implemented, following the viscosity damping expression shown in Eq. (4.25), but considering the different options presented above for the determination of f_m. Results were however unsatisfactory and will not be presented in the thesis.

Given the reasons exposed above, next ideas were applied to versions derived from the closed STRUCTT model presented in section 4.3.3. When considering Eq. (4.22) two terms appear in the right hand side (RHS) of the equation. Following the definitions of L and T in Eq. (4.23), the diffusion term (which we will define as D_{t_m}) can be expressed as shown in Eq. 4.29.

$$D_{t_m} = \frac{\partial}{\partial x_j}\left[\left(\frac{L^2}{T}\right)\frac{\partial t_m}{\partial x_j}\right] = \frac{\partial}{\partial x_j}\left[\beta\left(0.09\frac{k^2}{\epsilon}\right)\frac{\partial t_m}{\partial x_j}\right] \tag{4.29}$$

while the temporal dependent source term S_{t_m}, when no limitations are applied, can be expressed as Eq. (4.30):

$$S_{t_m} = \frac{1}{T}\left(\frac{k}{\epsilon} - t_m\right) \tag{4.30}$$

In the simulation performed in this thesis, when D_{t_m} was compared to the other source term S_{t_m}, it was found that the effect of the diffusion part can be neglected, as it will be shown in section 4.5.2. All effects were caused by the time dependent source term, which is actually the real Lagrangian average proposed by [87]. Therefore, the first simplification we can perform is to consider only S_{t_m}, neglecting D_{t_m}. Besides, the bounding limits on S_{t_m} were never applied (they were proposed to guarantee stability if required, but this was never the case), so they could also be avoided. In Eq. (4.30), T defines the length in time of the averaging procedure. Instead of using the modeled time scale defined in Eq. (4.23), it was proposed to use the resolved time scale as the time integration length $T = 1/f_r$. Integration is therefore extended in zones with high fluctuations, high variations of the flow, where higher averaging is required. This modeled will be defined as **STRUCTT-Tf_r**.

The rest of the work employed in this topic was focused on the extension of the model (initially intended to be applied to internal flows) to external aerodynamics applications. In fact, inlet turbulence in these cases is usually characterized by very low frequency fluctuations, which leads to very low f_m and incorrect activation of STRUCT, that predicts a small value of f_r but still higher than f_m in zones where no structures are present and no model activation should occur. To correct these deficiencies, different alternatives were proposed for the determination of f_r from the resolved velocity field, in order to avoid the incorrect activation of the model. Since the internal flow inside a gear pump is the application considered in this thesis, no information will be given on the other variants of the modeled developed during the PhD.

4.4 Implementation

The different versions have been again implemented in OpenFOAM®, following the rules of structured coding and avoiding code repetition. In this case, the main class containing k and ϵ equations, as well as the definition of the nonlinear part of the Reynolds stress tensor, has been implemented as a double template class, to allow the user to define further developments using either RANS or LES

approaches, and to conserve the same formulation for incompressible or compressible cases. In our case no LES functions are needed and all models derive form a instantiation of the double template class, that sets the turbulent model to be a nonlinearEddyViscosity RAS model. The different versions implement their own definition of f_m and viscosity blending function, solving for the required variables if necessary and defining their constants. The collaboration diagram is shown in Fig. 4.2. A new user can always add additional versions without the need to modify or re-compile any of the existing base classes.

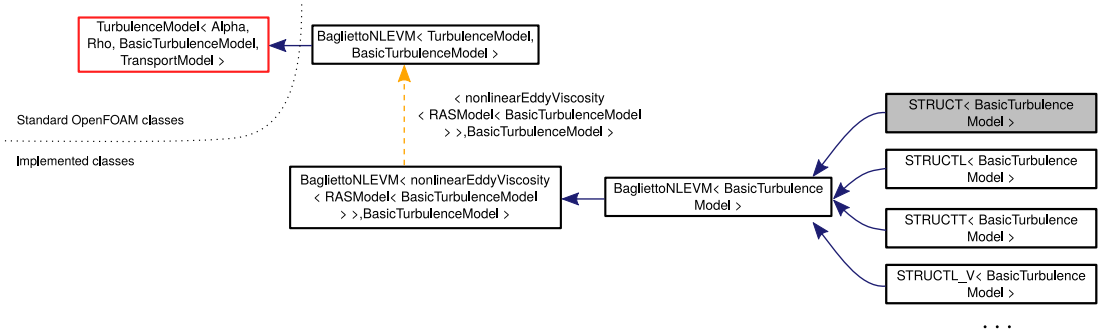


Figure 4.2: Collaboration diagram of the turbulence classes implemented in OpenFOAM®

Furthermore, the first steps for the implementation of STRUCT models in the Spectral Element Method (SEM) code Nek5000 have also been taken. Nek5000 uses Galerkin projection methods and high-order Lagrange polynomial basis functions with Gauss-Lobatto-Legendre points for efficient quadrature. It is highly scalable, spectrally accurate and has been greatly optimized for parallel computations. Its efficiency and accuracy make it suitable to perform DNS and LES calculations, which has been the main purpose of its use in the recent years. In fact the code did not include any $k-\epsilon$ RANS model, so that the implementation had to be done from the very beginning. In particular several low Reynolds linear and nonlinear $k-\epsilon$ models have been implemented in order to demonstrate the performance of the baseline NLEVM model used in STRUCT in a code with minimum numerical diffusion and high accuracy. Namely the low Reynolds linear models by Launder and Sharma [73], Chien [16], Nagano and Tagawa [99] and nonlinear models by Lien *et al.* [79] and Baglietto and Ninokata [4] (the one used in STRUCT) have been implemented and tested. Not being related to the rest of the work presented in this thesis, which was performed under the OpenFOAM® framework, the implementation in Nek5000 will not be commented here.

4.5 Numerical experiments

The results section will be divided in two parts. First, the correct implementation of the nonlinear RANS model will be tested in simple academic tests, both in OpenFOAM® and in Nek5000, focusing on cases where the nonlinearity of the models play an important role. In the second part several cases will be considered for the tests of the different hybrid approaches explained before.

4.5.1 NLEV RANS model

Three cases have been chosen in this section. First the classical turbulent channel flow has been simulated, paying attention to the comparison with other linear and nonlinear formulations, and observing the predictions of the anisotropy of the Reynolds stresses in this simple case. Then two cases with turbulence-induced secondary motions have been simulated: a fully developed square channel flow, which was tested in Nek5000 and a rod bundle with a triangular array configuration, tested in OpenFOAM®.

Fully developed turbulent channel flow

The first case is the simplest wall-bounded turbulent flow, the flow in a plane channel. Simulations have been carried out both in OpenFOAM® and Nek5000.

The fully developed turbulent channel flow at $Re_\tau = u_\tau h/\nu = 392.24$, where u_τ and h refer respectively to the friction velocity and the channel half-width has been studied in Nek5000. Moser DNS results [95] have been taken as a reference. The domain has been simplified by considering only half of a 2D channel. Periodic conditions have been applied in the streamwise direction while symmetry conditions are applied at the channel centerline. An explicit force is added in the streamwise direction to maintain the same average velocity field as that of the DNS results. First, two meshes (shown in Fig. 4.3) with the characteristics shown in Table 4.4 have been considered to check the mesh convergence of BagliettoNLEVM. One single spectral element is considered in the streamwise direction while 8 or 12 elements with progressively increasing size are considered in the wall normal direction.

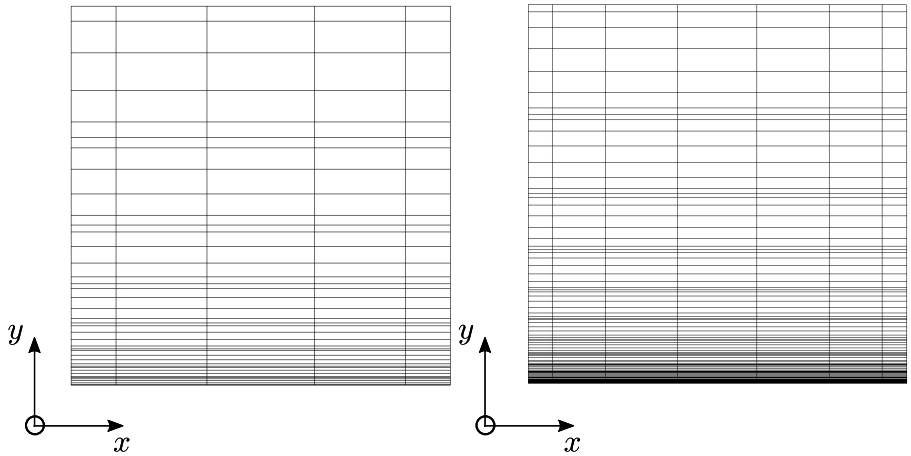


Figure 4.3: Meshes considered for channel flow case in Nek5000: coarse (left) fine(right)

Grid convergence is checked by studying the small variations observed in velocity and turbulent kinetic energy profiles, shown in wall units in Fig. 4.4.

Given the simplicity and fast convergence of the case, the finer mesh has been used to compare BagliettoNLEVM with other linear and nonlinear $k-\epsilon$ models implemented in Nek5000, as well as the already available $k-\omega$ model. Fig. 4.5 and Fig. 4.6 show respectively velocity and TKE profiles.

Table 4.4: Channel flow Nek5000 mesh characteristics

Mesh	Number of elements	Polynomial order	y^+
Coarse	8	6	1.0
Fine	12	8	0.1

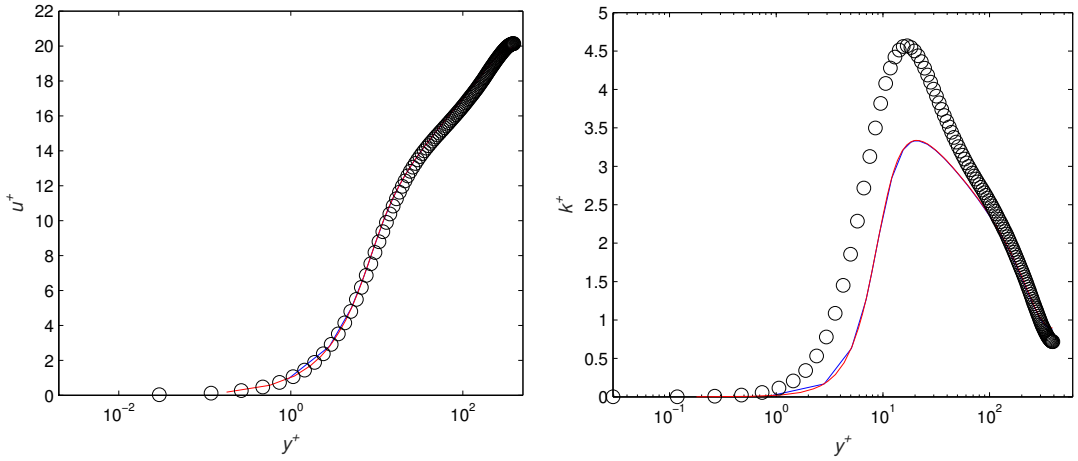


Figure 4.4: Mesh convergence of velocity (left) and TKE (right) profiles; coarse mesh (blue); fine mesh (red); reference DNS (○ ○ ○)

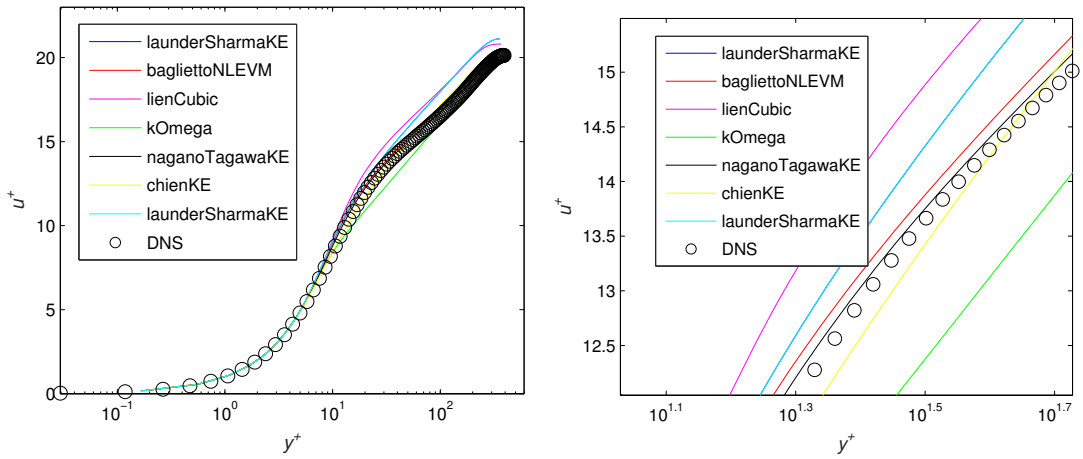


Figure 4.5: Velocity profiles (left) and zoomed profiles (right) for several models in comparison with reference DNS

The $k-\epsilon$ models considered here differ mainly in two aspects: the linear/nonlinear definition in the eddy viscosity assumption (Lien is the only nonlinear model together with BagliettoNLEVM), and the damping functions used for the low Reynolds treatment. When considering k and u profiles, the model by Nagano and Tagawa seems to give the best compromise, predicting accurately both quantities. BagliettoNLEVM is one of the best in the determination of the velocity profile but is one of the worst when considering TKE.

If the different components of the Reynolds stress tensor are also considered,

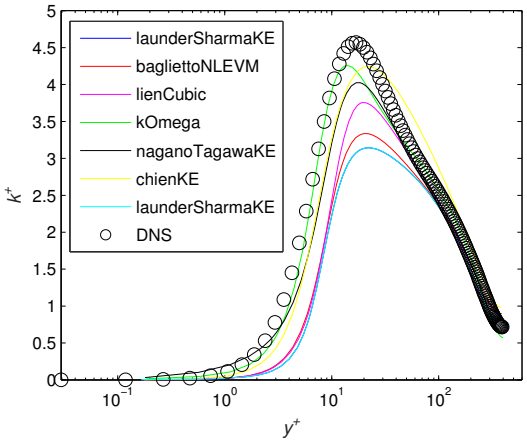


Figure 4.6: TKE profiles for several models in comparison with reference DNS

only nonlinear models are capable of capturing the anisotropy. When considering linear models, the three normal stresses are coincident in one single line for each of them.

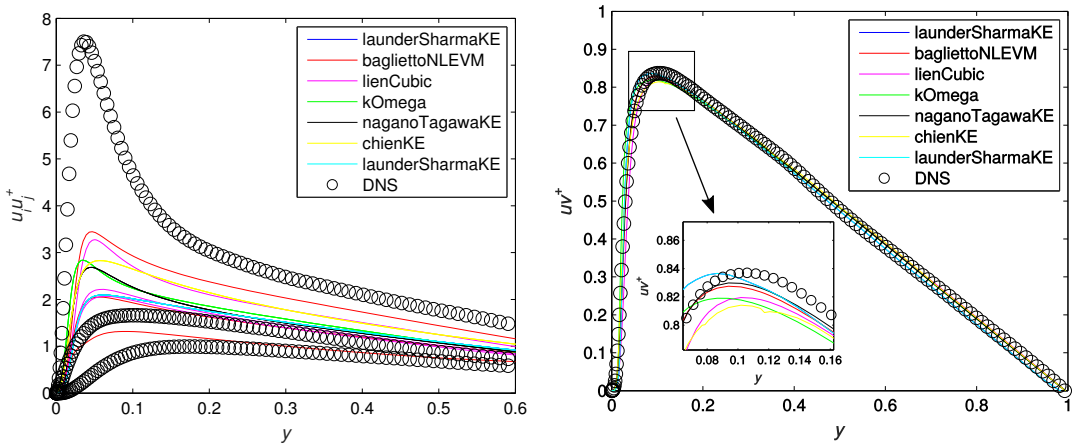


Figure 4.7: Reynolds stress tensor components: normal stresses $\overline{u'u'}$, $\overline{v'v'}$, $\overline{w'w'}$ (left); shear stress $\overline{u'v'}$(right)

However it can be checked that only the general trend can be captured, and none of the nonlinear models gave an accurate prediction near the walls. When BagliettoNLEVM is compared to Lien, Fig. 4.7 shows that BagliettoNLEVM gives more accurate prediction of the anisotropy.

OpenFOAM® has also been used to perform similar simulations, in this case at a higher Reynolds case $Re_\tau = 642.54$. DNS data from Iwamoto *et al.* is used as a reference. One-dimensional case has been studied with the same boundary conditions as those considered in Nek5000. Mesh schematic and characteristics are shown in Fig. 4.8 and Table 4.5.1.

Figure 4.8: OpenFOAM® mesh for channel flow

Number of cells	y^+
100	0.3

Table 4.5: Channel flow OpenFOAM® mesh characteristics

Fig. 4.9 shows velocity and TKE profiles for BagliettoNLEVM in comparison with other nonlinear model Lien, and two classical linear models available in OpenFOAM® that accept low Reynolds treatments: kOmegaSST and LaunderSharmaKE.

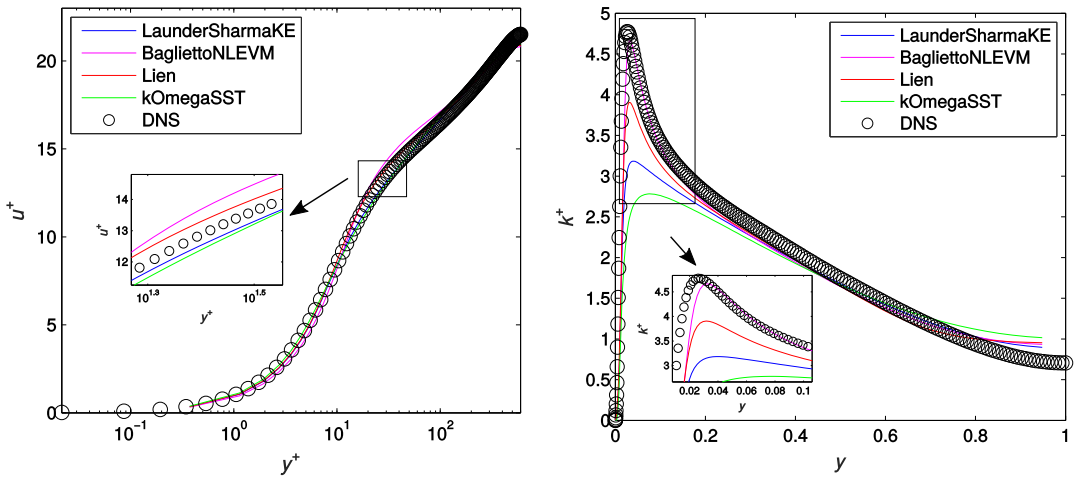


Figure 4.9: Velocity (left) and turbulent kinetic energy (right) profiles for several models in comparison with reference DNS

Results show in this case that all models give a good prediction of the velocity field, even if perhaps BagliettoNLEVM separates a little more from the DNS solution in the buffer layer, but it is by far the most accurate model in predicting peak value and global profile of turbulent kinetic energy. Fig. 4.10 shows the different components of the Reynolds stress tensor, testing the capacity of the models to predict anisotropy of the flow (left) and shear stress (right).

It is clear here that BagliettoNLEVM gives much better prediction of the anisotropy when compared to Lien, which however predicts shear stress more accurately. Results are in agreement with similar tests of BagliettoNLEVM [5].

In general BagliettoNLEVM, seems to be a good compromise between accuracy in the determination of velocity and TKE, and ability to capture the anisotropy

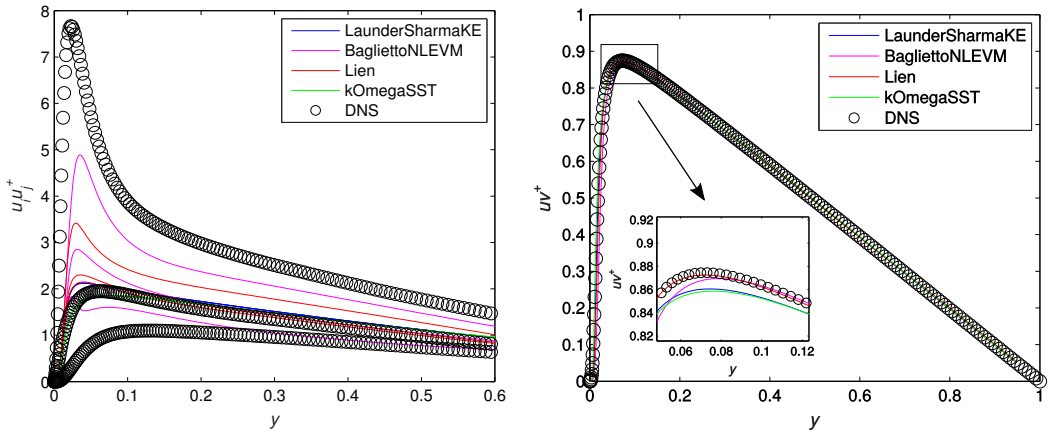


Figure 4.10: Reynolds stress tensor components: normal stresses $\overline{u'u'}$, $\overline{v'v'}$, $\overline{w'w'}$ (left); shear stress $\overline{u'v'}$(right)

of the flow. In the next cases we will consider situations in which this anisotropy influences significantly the flow, leading to secondary motions.

Square channel flow

The fully developed turbulent straight square duct flow is a canonical problem regarding turbulence-induced secondary flows. Anisotropy in the cross-sectional plane generate net flows in the directions perpendicular to the bulk flow, the so-called secondary flows, or flows of the second kind. These counter-rotating streamwise vortices in the corner of the duct, with an associated velocity much smaller than that of the streamwise flow, are caused by a fine balance involving the gradients of the Reynolds stresses and the pressure gradient, being therefore a challenging task for turbulence modeling. In this study, BagliettoNLEVM has been used in Nek5000 to simulate a simplified two-dimensional square channel, to test the capabilities to predict the secondary motions in a square channel at $Re_\tau = 2hu_\tau/\nu = 600$ where u_τ is computed by averaging wall shear stress over all wall cells. DNS results by Huser and Biringen are used as a reference [62].

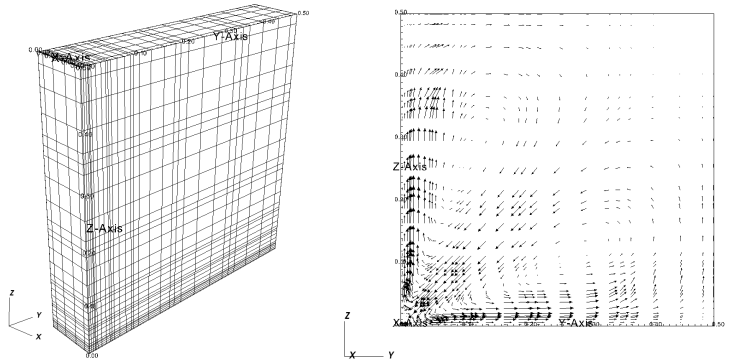


Figure 4.11: Nek5000 mesh (left) and schematic of turbulence induced secondary flows (right)

A 6×6 element mesh with 5^{th} order polynomials has been used in Nek5000

as shown in Fig. 4.11. Capabilities of BagliettoNLEVM to compute secondary motions is tested by comparing streamwise and secondary velocity components at several y locations with the reference DNS data, as shown in Figs. 4.12 and 4.13. Profiles have been dimensionalized with the bulk velocity U_{ref}.

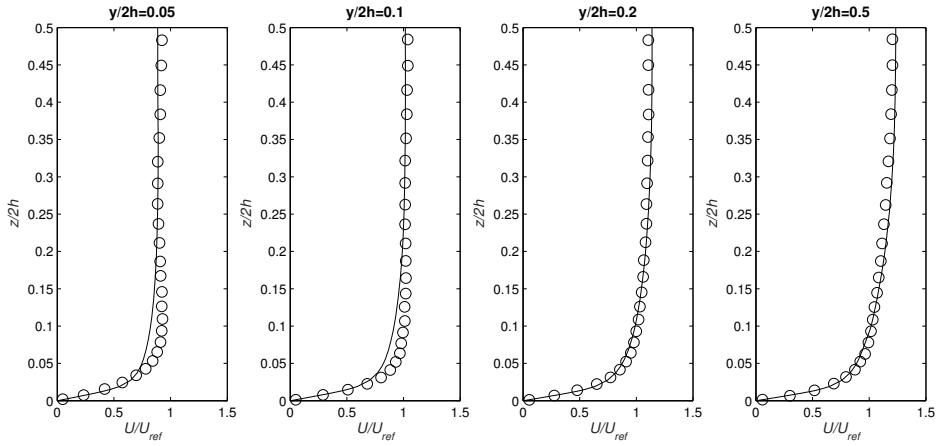


Figure 4.12: Streamwise velocity profiles at several y locations: BagliettoNLEVM(—) in comparison with reference DNS (∘ ∘ ∘)

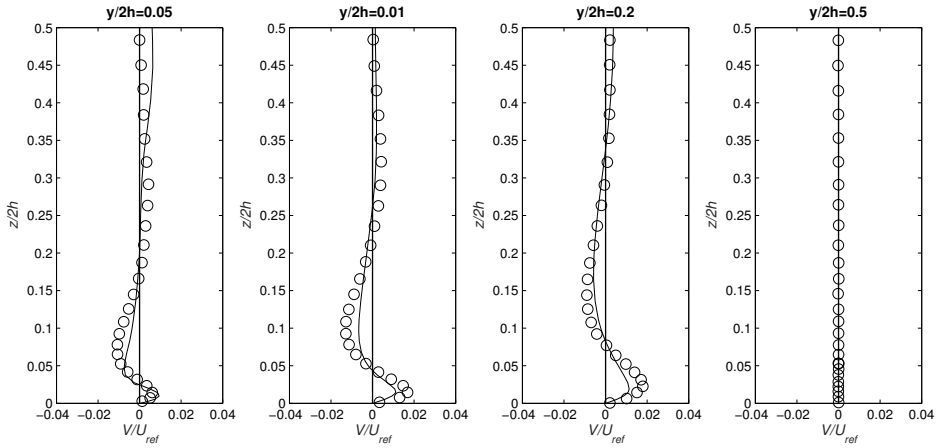


Figure 4.13: Secondary velocity profiles at several y locations: BagliettoNLEVM(—) in comparison with reference DNS (∘ ∘ ∘)

Furthermore, Reynolds normal stresses have also been computed and compared to DNS reference data in Fig. 4.14.

Good agreement is found in streamwise and secondary velocity, with slight under-prediction of the peak values of this last quantity. When comparing Reynolds stress components, the trend is qualitatively captured adequately, in particular for streamwise fluctuations. Relative importance between stresses seems also properly estimated by the model, even if significant errors are found when comparing absolute values.

Similarly to the case presented here, next case will evaluate another flow in which secondary flows appear, using for this case OpenFOAM® code.

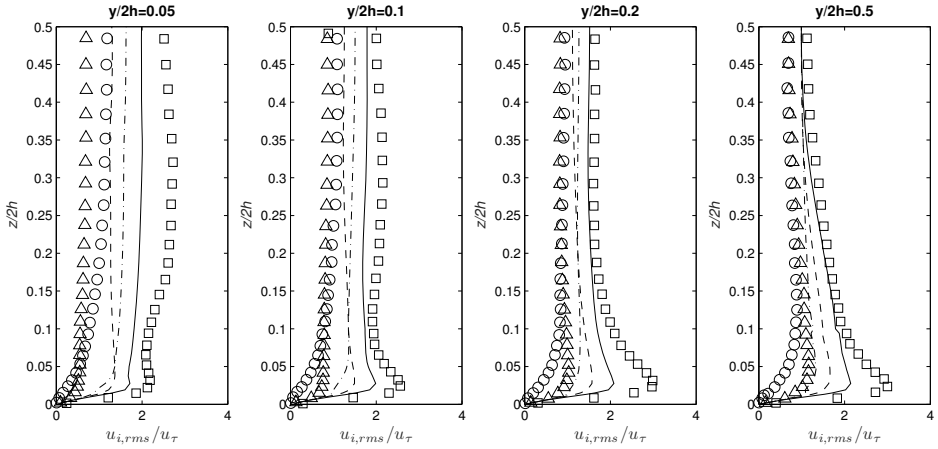
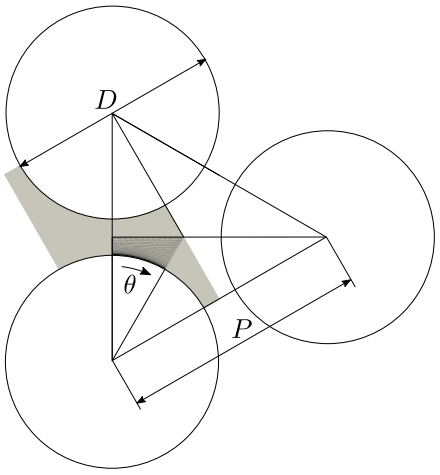


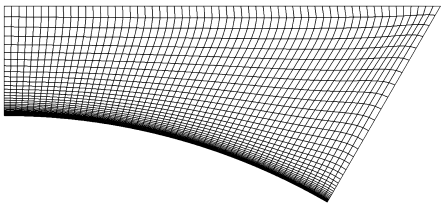
Figure 4.14: Reynolds normal stresses profiles at several y locations: BagliettoNLEVM: $\sqrt{\overline{u'u'}}/u_\tau$(—) $\sqrt{\overline{v'v'}}/u_\tau$(− · −) $\sqrt{\overline{w'w'}}/u_\tau$(− − −); in comparison with reference DNS: $\sqrt{\overline{u'u'}}/u_\tau$(□□□) $\sqrt{\overline{v'v'}}/u_\tau$(○ ○ ○) $\sqrt{\overline{w'w'}}/u_\tau$(△△△);

Triangular array rod bundle

The flow inside a tightly packed triangularly arranged rod bundle has been studied in OpenFOAM®. This is another simple configuration where anisotropy plays an important role and simple models fail to reproduce the secondary flows and shear stress distributions. This is a common configuration representative of the flow inside tightly packed rod bundles of common use in nuclear power reactor cores or in industrial heat exchangers. In particular the case under study is triangular array with pitch to diameter ratio P/D of 1.17 at $Re = 181200$. The 2D domain can be further reduced thanks to the symmetries (see Fig. 4.15a), arriving to the mesh shown in Fig. 4.15b, where symmetry conditions are applied to all boundaries excepting the cylinder wall, and whose characteristics are presented in Table 4.6.



(a) Use of domain symmetries



(b) Rod bundle mesh

Figure 4.15: Domain reduction for the generation of final mesh in the rod bundle case

Table 4.6: Triangular rod array mesh characteristics

Number of cells	$y^+_{\max}$
2400	1.0

The secondary flows inside the sub-channel generate a more uniform distribution of axial velocity [5]. The accuracy of the model is first tested by comparing the distribution of axial velocity in radial direction (normal to the wall) at $\theta = 30deg$ (see Fig. 4.15a) with the experimental results from Mantlic *et al.* [85]. Analysis is also extended to the comparison of wall shear stress. Results are shown in Fig. 4.16.

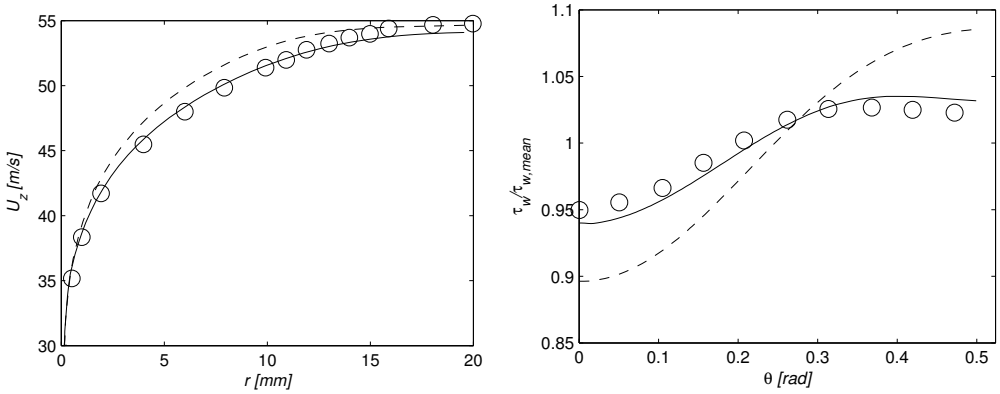


Figure 4.16: Axial velocity profiles in radial direction at $\theta = 30\ deg$ (left) and wall shear stress (right). Results shown for BagliettoNLEVM (—) and Lien ($- - -$) in comparison with experimental data ($\circ\circ\circ$)

The adequacy of the model for this case is clear when considering the improvement in the predictions with respect to Lien nonlinear model.

BagliettoNLEVM has in general proved to be capable of providing accurate predictions in the academic cases studied here. It has also shown to be able to capture the influence of the anisotropy on macroscopic variables such as velocity and wall shear stress distribution in practical cases.

Once the proper behavior of the NLEVM has been tested, the study proceeds with the simulations involving the different versions of STRUCT defined in section 4.3.3.

4.5.2 Hybrid models

In this sections the different models proposed in section 4.3.3 will be tested in a variety of cases that have been selected either because they represent simplifications of some of the phenomena that take place inside a gear pump, or because they are typical test cases for hybrid models, since typical RANS tend to fail due to the limitations of such approach. For all cases incompressible conditions apply, and reliable DNS or experimental data is used as a reference to compare models predictions. When considering a simulation performed with a hybrid model, it is worth mentioning the procedure followed to extract information about the veloc-

ity mean fluctuations. When that is the case, the triple decomposition applies for every instantaneous variable ϕ, as shown in Eq. (4.31).

$$\phi = \overbrace{\hat{\phi} + \underbrace{\tilde{\phi} + \phi''}_{\phi'}}^{\overline{\phi}} \tag{4.31}$$

where $\overline{\phi}$ is the resolved variable and $\hat{\phi}$ is the time average. It is a widely accepted approach in the validation of models to compare prediction of time averaged fields and averaged fluctuations (second order moments). Considering for instance the velocity field, a simulation will provide the time average of the resolved fields $\hat{\overline{u}}$ which will be used as an approximation of the actual average following Eq. (4.32).

$$\hat{\overline{u}} = \widehat{\hat{u} + \tilde{u}} = \hat{u} - \widehat{u''} \approx \hat{u} \tag{4.32}$$

Similarly, velocity fluctuations are defined as the variance of u with respect to time ($\widehat{u'u'}$) for velocity fluctuations the approximation in Eq. (4.33) is considered:

$$\widehat{u'u'} = \widehat{(u - \hat{u})(u - \hat{u})} = \hat{\tilde{u}}\hat{\tilde{u}} + \widehat{u''u''} + 2\,\widehat{\tilde{u}u''} \approx \hat{\tilde{u}}\hat{\tilde{u}} + \widehat{u''u''} \tag{4.33}$$

where the first term of the approximation is obtained from the variance of the resolved velocity with respect to the time average ($\hat{\tilde{u}}\hat{\tilde{u}} \approx \widehat{(\overline{u} - \hat{\overline{u}})(\overline{u} - \hat{\overline{u}})}$) and the second term is obtained by averaging the modeled residual stresses ($\hat{\tau}_{ij} = -\rho\widehat{u_i''u_j''}$). These are very common assumptions.

Periodic hill

The flow over periodic hills is a common test case, typically intended for studying LES models performance in the presence of separation and reattachment. It consists of polynomial-shaped obstacles mounted on a flat plane with a recirculation region in their wake. Highly resolved LES simulations [37] have been used as a reference for this case. Mean streamlines and geometry details are shown in Fig. 4.17. Reynolds number based on bulk velocity U_b and hill height h is $Re = 10595$.

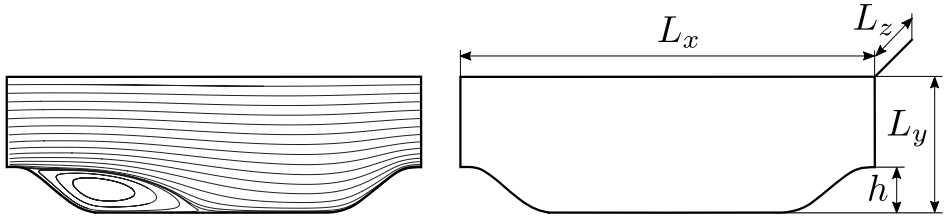


Figure 4.17: Streamlines of mean velocity field (left) and geometric parameters (right) of the 2D periodic hill configuration. $L_x = 9h$, $L_y = 3.035h$, $L_z = 2h$

Two meshes have been considered for this case as shown in Fig. 4.18 with the characteristics shown in Table 4.7.

Upper and lower walls should be modeled using RANS due to the selection of the resolved frequency parameter, that vanishes near the wall. They have enough resolution to apply non slip velocity condition. For the rest of boundaries

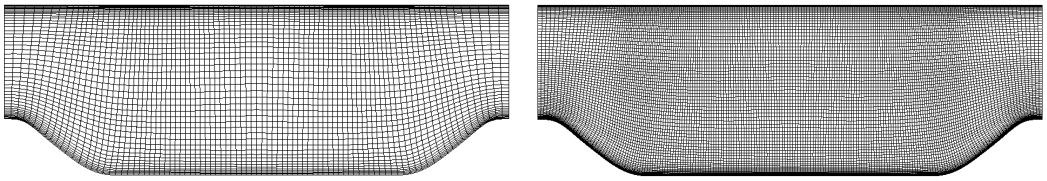


Figure 4.18: Coarse (left) and fine(right) meshes for periodic hill case

Table 4.7: Periodic hill case mesh characteristics

Mesh	Number of cells	Max. non-orthogonality [deg]	$y^+_{\max}$
Coarse	≈ 82000	17	1.0
Fine	≈ 60000	14	0.5

periodic conditions apply, and an explicit pressure gradient has been added to the velocity equations to maintain U_b mean velocity through the inlet-outlet patch. Simulations are carried out using LUST [141] scheme for convective terms and second order backward temporal discretization. Fields have been averaged for at least 10 convective times in each simulation. Many simulations have been performed and only some results will be shown here for the sake of brevity.

First, a 2D simulation was carried out to determine the reference f_{m0} required by the controlled STRUCT. Then standard BagliettoNLEVM, STRUCT, STRUCTL and STRUCTT were tested in both meshes. As an example Figs. 4.19 and 4.20 show a comparison of the predictions of x and y-velocity profiles for all four models in the coarse mesh.

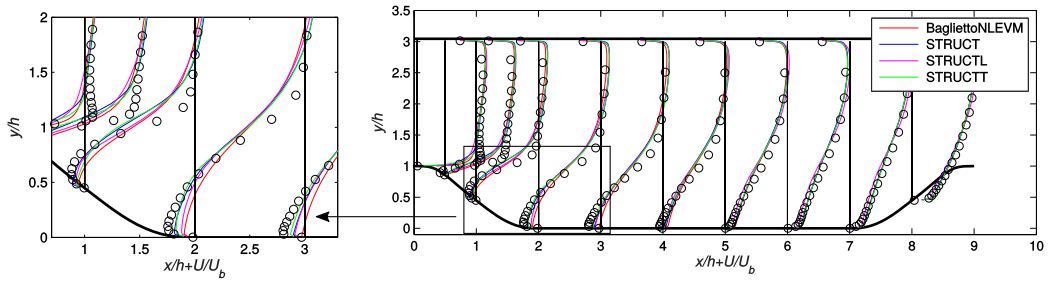


Figure 4.19: Mean x-velocity profiles for standard models

It is clear that the RANS model provides the worst prediction of the separation and reattachment of the flow and incurs in the highest error for both velocity components. STRUCT and STRUCTT give in general similar predictions while STRUCTL approaches that of BagliettoNELVM, worse than the rest. Fig. 4.21 shows the activation regions for the three hybrid models, represented by r. r is the viscosity damping function in the case of STRUCTT, while it only identifies activation regions in a binary fashion for STRUCT and STRUCTL.

It is worth noticing that only the controlled STRUCT generates activation of the model in the right side of the domain where the slight increase of flow

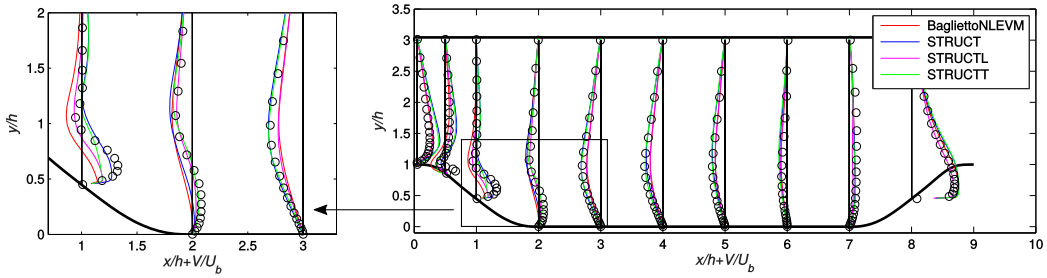


Figure 4.20: Mean y-velocity profiles for standard models

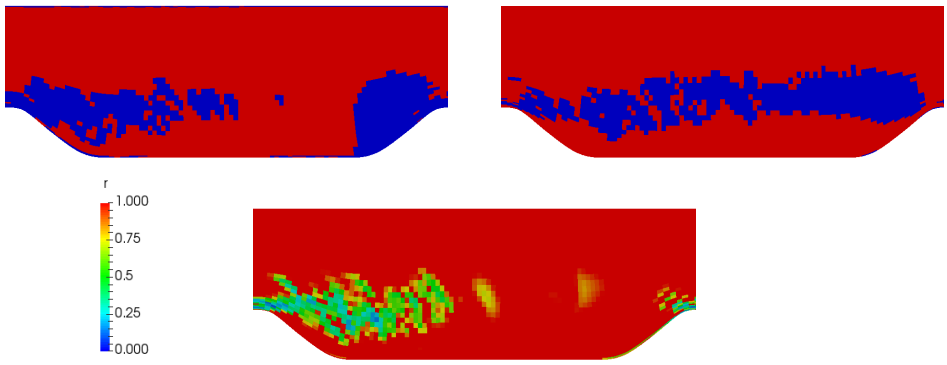


Figure 4.21: Activation regions for standard hybrid models in coarse mesh: STRUCT (top left); STRUCTL (top right); STRUCTT(bottom)

strain overcomes the fixed modeled frequency, while it does not when the modeled frequency is obtained by an averaging procedure (as in STRUCTL or STRUCTT).

Both streamwise and wall normal velocity profiles were sufficiently insensible to mesh resolution in all cases and in good agreement with reference results. However, even if mesh convergence was achieved for the baseline BagliettoNLEVM when considering the two 3D meshes presented before (differences in turbulent kinetic energy profiles were also minimum), it was found that the hybrid approaches were significantly mesh dependent when considering k profiles. As an example, Figs. 4.23, 4.24 and 4.25 show the mesh dependency of STRUCTT model, the one for which the largest differences were found between the results obtained with the coarse and fine meshes. STRUCTT activation regions for the two meshes is shown in Fig. 4.22.

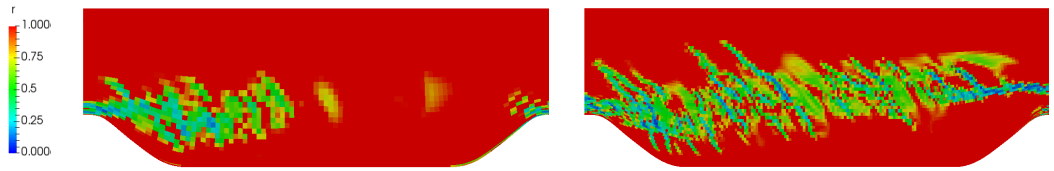


Figure 4.22: Activation of STRUCTT in coarse (left) and fine (right) meshes

Fig. 4.25 shows that in general, mesh refinement improves the prediction of TKE, over-predicted in the coarse mesh in the middle of the channel. As shown

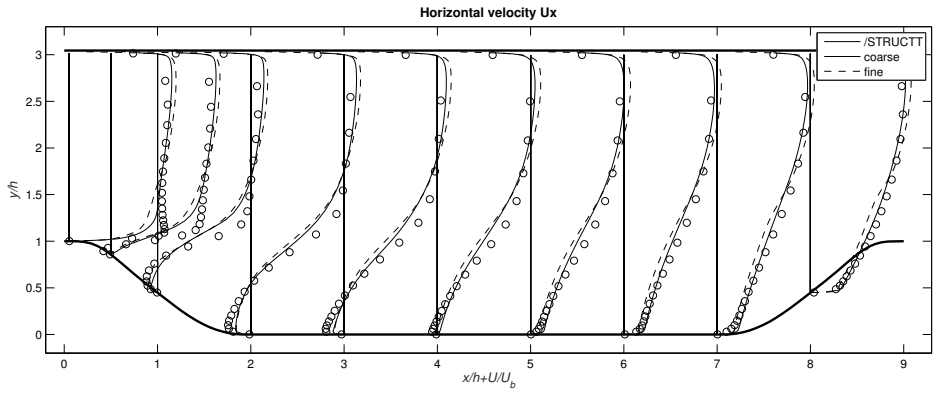


Figure 4.23: Mesh dependence of x-velocity profiles for STRUCTT: coarse mesh (—); fine mesh (– – –)

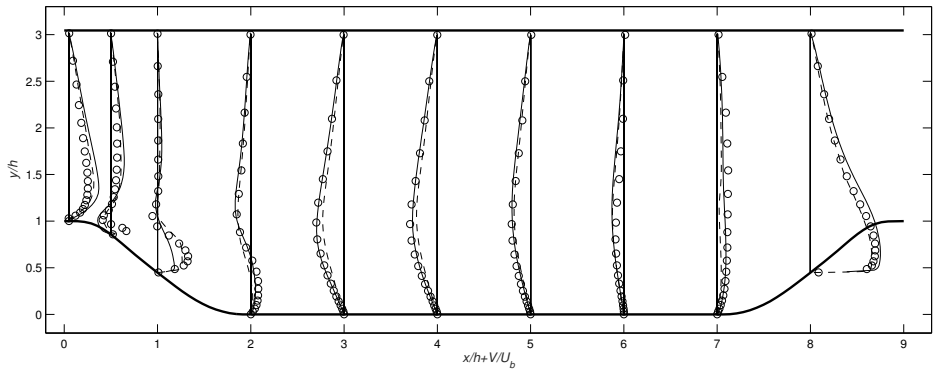


Figure 4.24: Mesh dependence of y-velocity profiles for STRUCTT: coarse mesh (—); fine mesh (– – –)

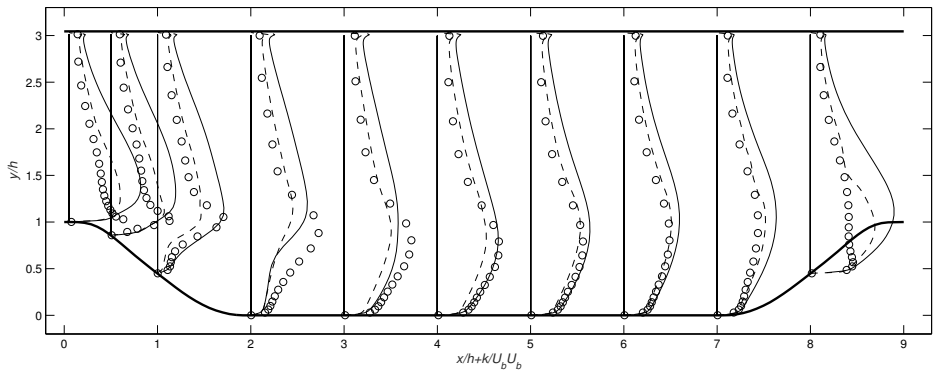


Figure 4.25: Mesh dependence of TKE profiles for STRUCTT: coarse mesh (—); fine mesh (– – –)

by Fig. 4.22, more structures are resolved in the fine mesh and this leads to an improvement in the results.

While both results and activation regions for STRUCTL seem realistic, it was noticed that activation of the model was not only due to physical issues but there were also numerical problems involved. In fact, it was noticed that the long

integration distance R used for the geometric averaging procedure (see Eq. (4.21)) was generating a checkerboard activation, due to the limitations of the average defined in Eq. (4.20). This occurred in both coarse and fine meshes. For instance, if a line in the streamwise direction is considered (see Fig. 4.26) profiles of f_m and f_r can help us identified where activation occurs and why (see Fig. 4.27).

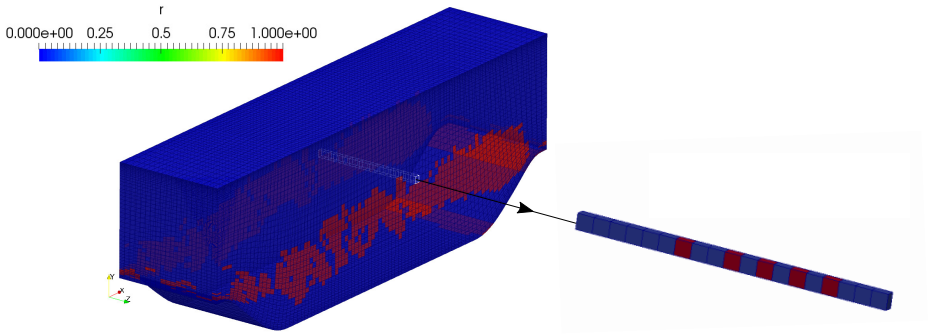


Figure 4.26: Activation along a streamwise line for STRUCTL in coarse mesh

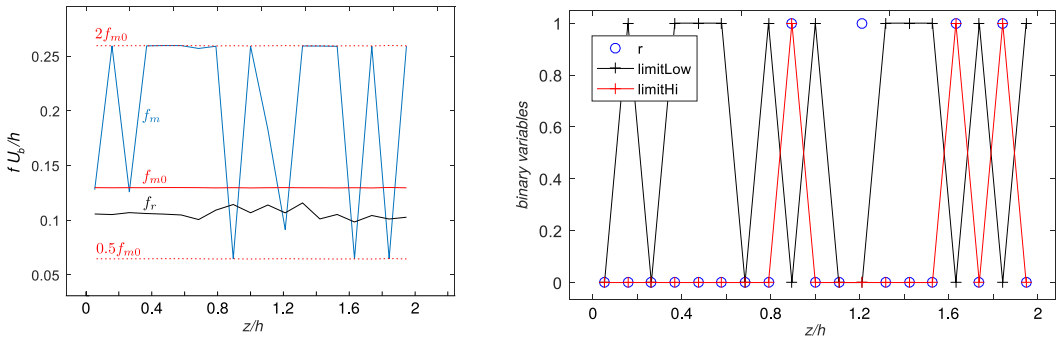


Figure 4.27: Dimensionless frequency profiles (left) and activation and limitations (right) along streamwise line

It can be derived from Fig.4.27 that if no averaging of f_{m0} was considered for the determination of f_m (then $f_m = f_{m0}$) activation would never occur in that line since $f_r < f_{m0}$. Variations of f_{m0} in the vicinity of that streamwise line are small but however the integration distance R computed with Eq. (4.21) is very large, leading to a predicted value of f_m far from the values of f_{m0} around the local cells, and leading to the limitations of f_m to the top value of $2f_{m0}$ (limitHi in Fig. 4.27) or the bottom value of $0.5f_{m0}$ ((limitLow in Fig. 4.27). This huge oscillations cause an artificial activation of the model ($r = 1$), only due to the averaging procedure used, as defined in Eq. (4.20).

In fact, the integral length scale (l) is typically estimated by $l = C_\mu^{3/4} k^{3/2}/\epsilon$. The integration distance is therefore $R = C_R/C_\mu^{3/4} l \approx 12l$. We are using local information to extrapolate the behavior $12l$ away form the local cell, which is the cause of the unphysical activation. C_R was probably assigned on the basis of optimizing results in some other cases, however this case shows that the chosen value is too large. In order to avoid this behavior we can either reduce the activation region to a more logical value, or consider alternative averaging procedures, as

described in the different variants proposed in section 4.3.3.

The main conclusions extracted from all the simulations and parametric studies of the different closures of STRUCTL-like models are given next. First for all STRUCTL, STRUCTAt, STRUCTAt_V, STRUCTAf or STRUCTAf_V, integration distance should be kept sufficiently small to avoid spurious activation. As an example Fig. 4.28 shows the activation and active limitations of f_m for STRUCTL when the integration distance is computed as C'_R times the integral length scale ($R = C'_R C_\mu^{3/4} k^{3/2}/\epsilon$) for different values of C'_R.

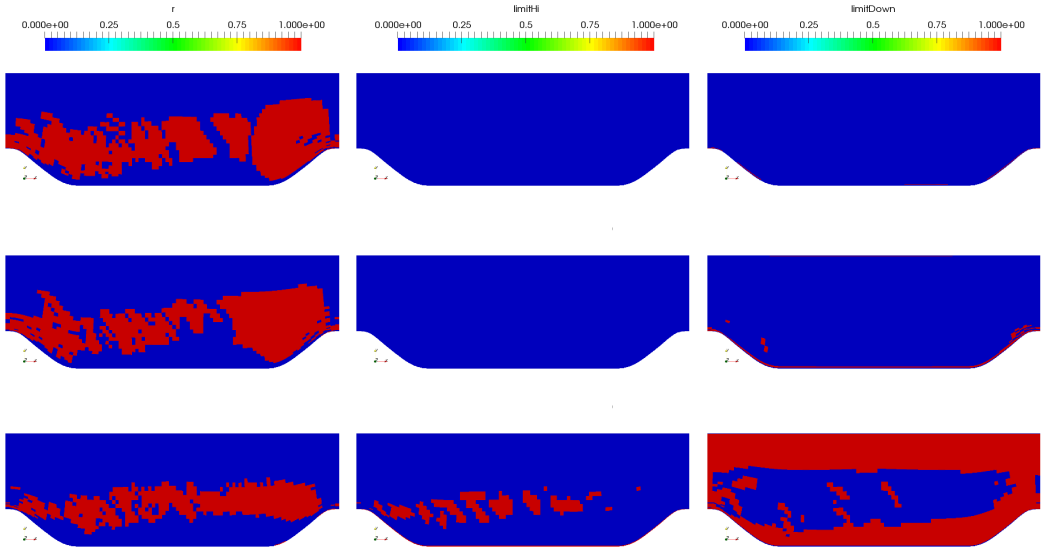


Figure 4.28: Activation and limitation regions for STRUCTL in coarse grid: $C'_R = 1$ (top); $C'_R = 2$ (medium); $C'_R = 10$ (bottom)

When _V-like models are used (R computed from cell size) no significant changes are observed in activation regions. In fact R does not vary greatly in the inner part of the domain, and using an integration distance based on R of based on the grid size, which is quite uniform, does not produce a meaningful difference as long as integration distance is similar. It was noticed that in _V-like models, upper limitations apply near the wall boundaries, which however does not affect the activation of the model (see Fig. 4.29).

Results obtained with any of the models mentioned here were very similar as long as the integration distance was kept to a small value. When comparing standard STRUCTL, STRUCTAt or STRUCTAf in their standard form (using $C_R = 2$), with the _V-like alternative: STRUCTF, STRUCTL_V, STRUCTAt_V or STRUCTAf_V with a small value of C_R, for instance $C_R = 2$, it can be noticed that the nonphysical limitations worsen the performance of the first group of models. For instance, Fig. 4.30 shows this comparison for velocities and TKE, in the vicinity of zone where the highest differences were observed for each variable in the coarse mesh. Even if differences are not very large, the figure shows how models can be grouped in two categories that give similar results for the same value of C_R: those for which the integration distance is calculated as a function of the cell size, therefore small, and those for which the distance is calculated as

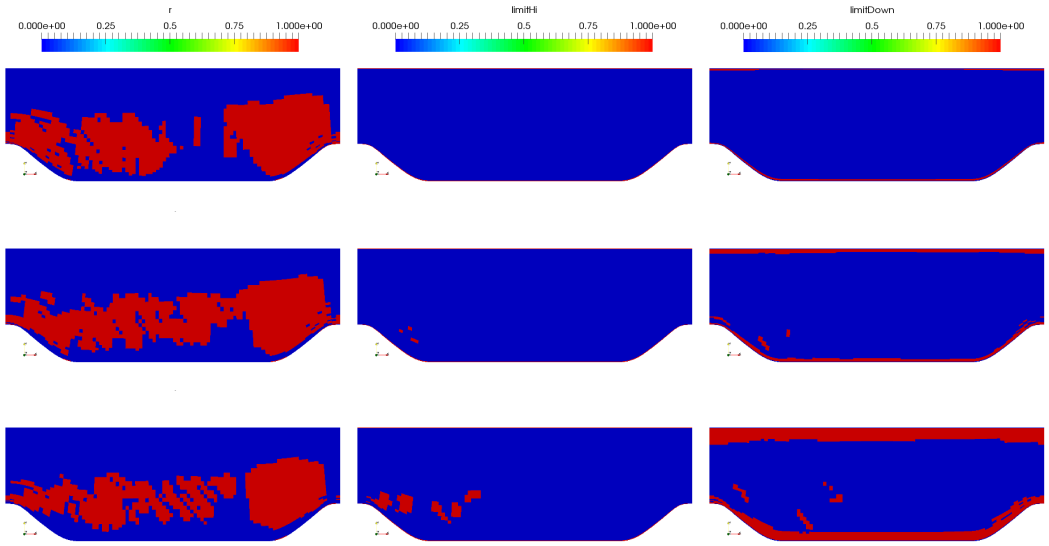


Figure 4.29: Activation and limitation regions for STRUCTL_V in coarse grid: $C_R = 2$ (top); $C_R = 5$ (medium); $C_R = 10$ (bottom)

a function of the turbulent length scale, that suffer from limitations and generate a higher error.

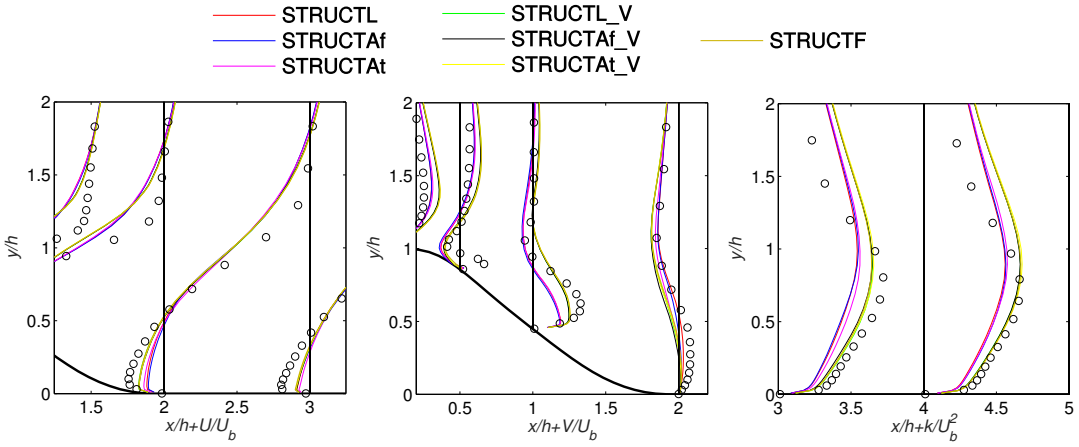


Figure 4.30: x-velocity (left), y-velocity (center) and TKE profiles (right) for standard and _V-like models in coarse mesh

Similarly, the separate effect of the integration distance can be studied. For example, Fig. 4.31 shows the effect on the results of variations of C'_R for STRUCTL, when the integration distance is computed according to $R = C'_R C_\mu^{3/4} k^{3/2}/\epsilon$.

Again, Fig. 4.31 shows that results improve progressively with the decrease of C'_R, as the cells with limited f_m disappear (see Fig. 4.28), even if profiles are accurately captured in any case.

It could be argued that local turbulent time scale could possibly be used as an approximation of f_m without any averaging ($f_{m0} = f_m$). However Lenci [75] already tested this option and concluded that some kind of averaging is necessary.

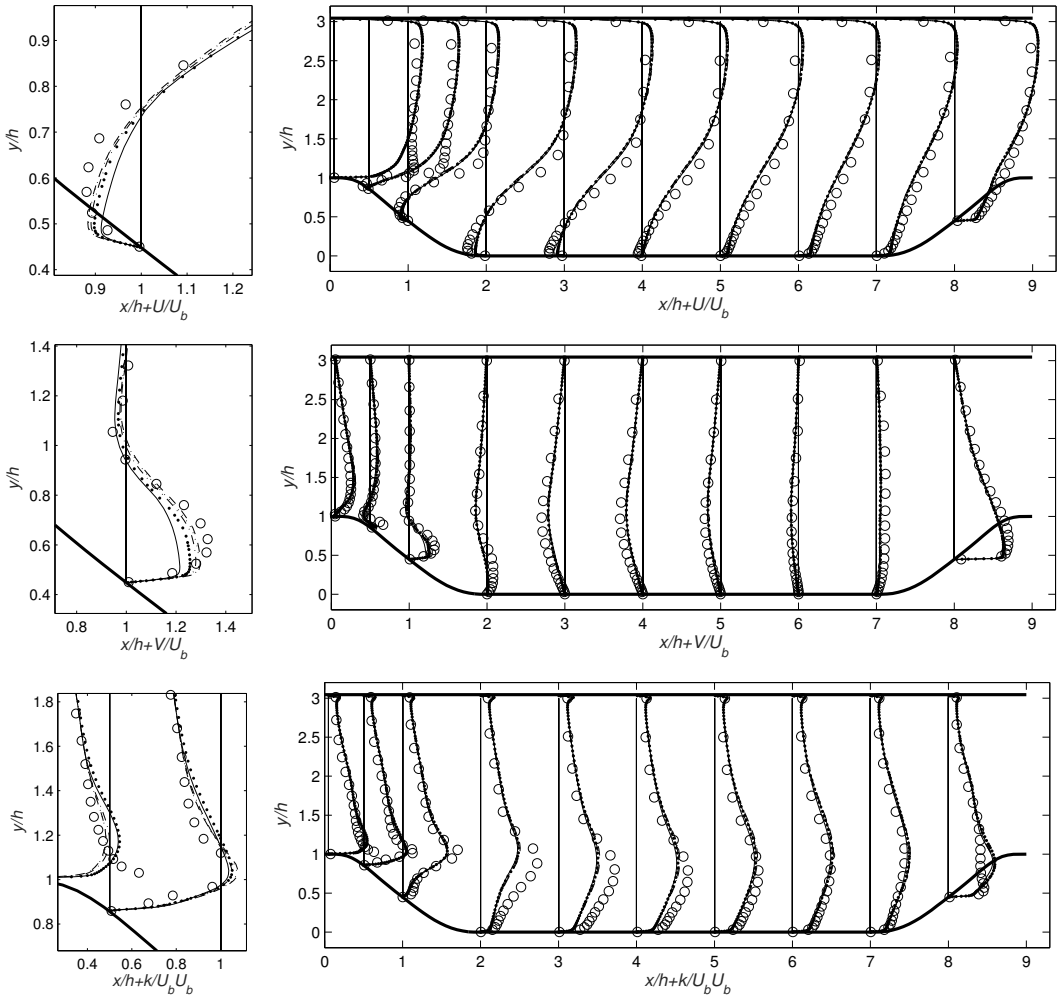


Figure 4.31: x-velocity (top), y-velocity (center), TKE (bottom) profiles (right) and zoomed profiles (left) for standard STRUCTL: $C'_R = 0.25$ ($- - -$); $C'_R = 0.5$ ($\cdots$); $C'_R = 1$ ($- \cdot -$); $C'_R = 2$ ($\bullet\bullet\bullet$); $C'_R = 10$ (—)

If this is true, the use of the alternative models proposed in this section could behave as a too small averaging when mesh is refined. Besides, STRUCTL-like models are still unclosed models, since the user must provide the ϕ parameter, whose optimal value depends on the case.

Given the very small variations found in this case, the study is extended to the analysis of the wall bounded flow around a square cylinder, where the vortex shedding approximates also the physics of a similar vortex shedding in a gear pump.

Flow around a square cylinder

In this section, the flow past a square cylinder is analyzed. This represents a common example of flow with shedding and massive separation, typically used to show the performance of hybrid models over U-RANS. U-RANS generally predicts a single mode fluctuation, far from the complex spectrum evidenced experimen-

tally [88]. This case is also useful because the distinction between regions with clear scale separation, and those in which model activation should be necessary, is clear.

The case consists in a fully developed flow through a channel of width $14L$ that encounters a square cylinder of side L in its center, as shown in Fig. 4.32. Reynolds number based on the bulk velocity and square side is $Re = 21400$. Laser Doppler Velocimetry (LDV) measurements performed by Lyn *et al.* [81] will be taken as a reference for the comparison of velocity and velocity fluctuation profiles.

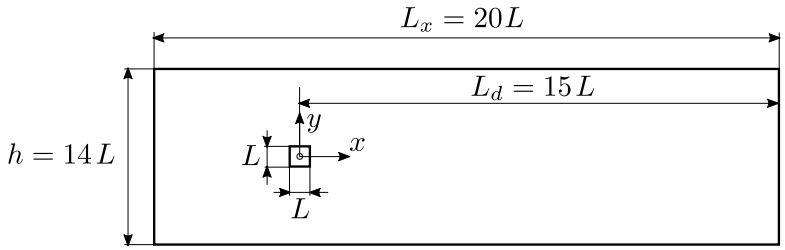


Figure 4.32: Square cylinder case geometry

As suggested [139] $L_z = 5L$ has been chosen for the length of the domain in the spanwise direction where periodic boundary conditions are applied.

The experimental domain has been enlarged in the x-direction, in order to apply a uniform velocity boundary condition at inlet, that develops and matches experimental results at $x/L = -3$, and to reduce the effect of the outlet boundary condition, as suggested by [75]. The mesh used for the simulations extends therefore from $x/L = -35$ to $x/L = 31$ and has been proved to produce mesh converged results for the baseline U-RANS. The grid is illustrated in Fig. 4.33 together with the characteristics shown in Table 4.8.

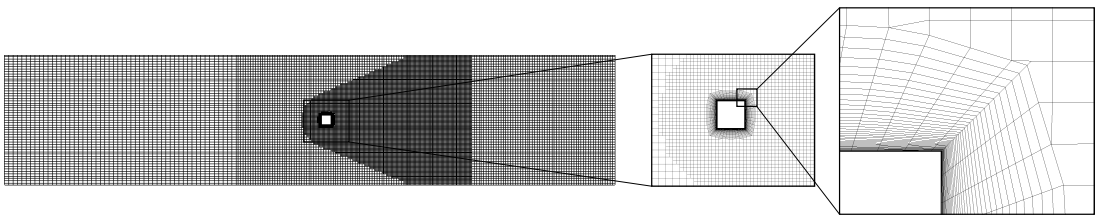


Figure 4.33: Square cylinder case mesh

Table 4.8: Square cylinder case mesh characteristics

Number of cells	Max. non-orthogonality [deg]
≈ 680000	43

Mesh resolution near the square cylinder walls has been increased to satisfy $y^+ < 1$ while classical wall functions are used in upper and lower wall boundaries, ensuring $y^+ > 30$ for their correct applicability. Preliminary U-RANS results have been use to provide the reference turbulent frequency scaled f_{m0}, required for the

controlled STRUCT simulations. U-RANS and hybrid models simulations have been averaged for about 4 convective times (considering the entire streamwise length of $66L$), which was shown to be enough to provide converged statistics.

Given the problems related to the use of STRUCTL explained before with the large integration distance, and trying to avoid the mesh dependence if a small integration distance is considered, or any of the suggested models (STRUCTAf, STRUCTAt...) is used, all of which are still open models (ϕ is still a user-defined parameter for all of them), the study of this case has been centered in completely closed models, the ones derived from the STRUCTT version.

As a starting point, results obtained from the baseline BagliettoNLEVM are compared to those obtained with STRUCT (where results improved as the parameter ϕ was reduced and therefore $\phi = 10^{-10}$ was used) and with the standard version of STRUCTT. Fig. 4.34 shows the comparison of velocity profiles with experimental data while in Fig. 4.35 the non-zero components of the Reynolds stress tensor are compared to the reference LDV measurements.

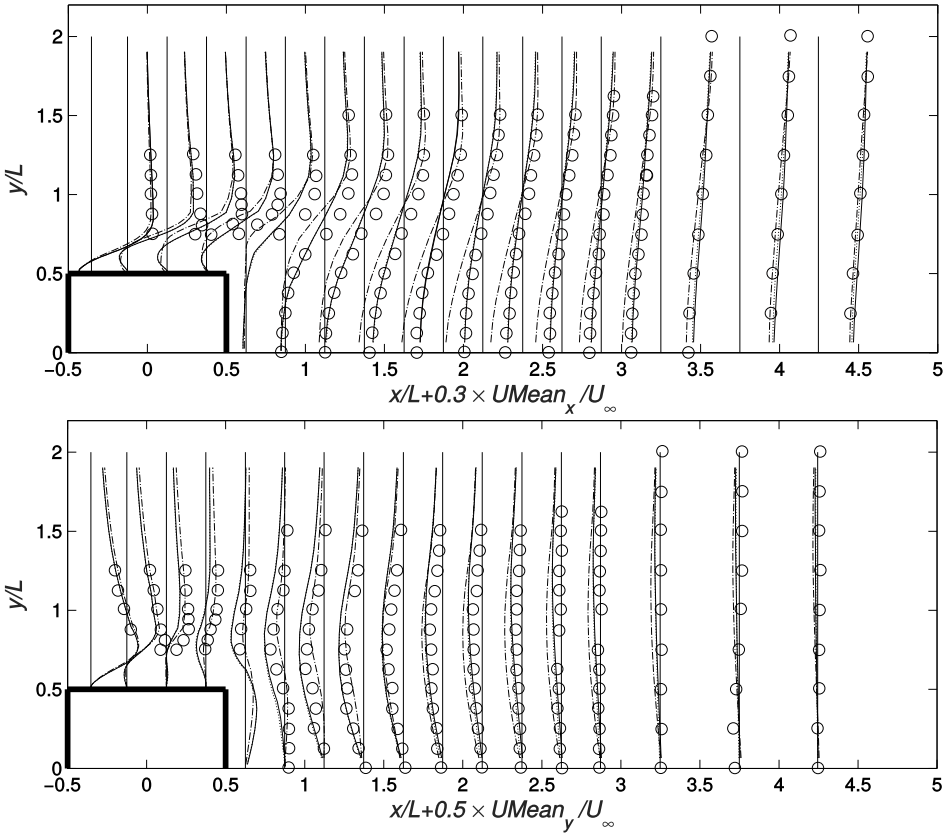


Figure 4.34: x-velocity (top) and y-velocity (bottom) profiles comparison for: BagliettoNLEVM ($- \cdot -$); STRUCT ($\cdots$); STRUCTT(—)

It is clear that the performance of both hybrid approaches is much better than that of the baseline nonlinear RANS model, which predicts an extended region with unrealistic negative x-velocity. When comparing both hybrid approaches, differences are minimum in velocity profiles and not clear when comparing velocity fluctuations: while controlled STRUCT provides slightly better accuracy in the

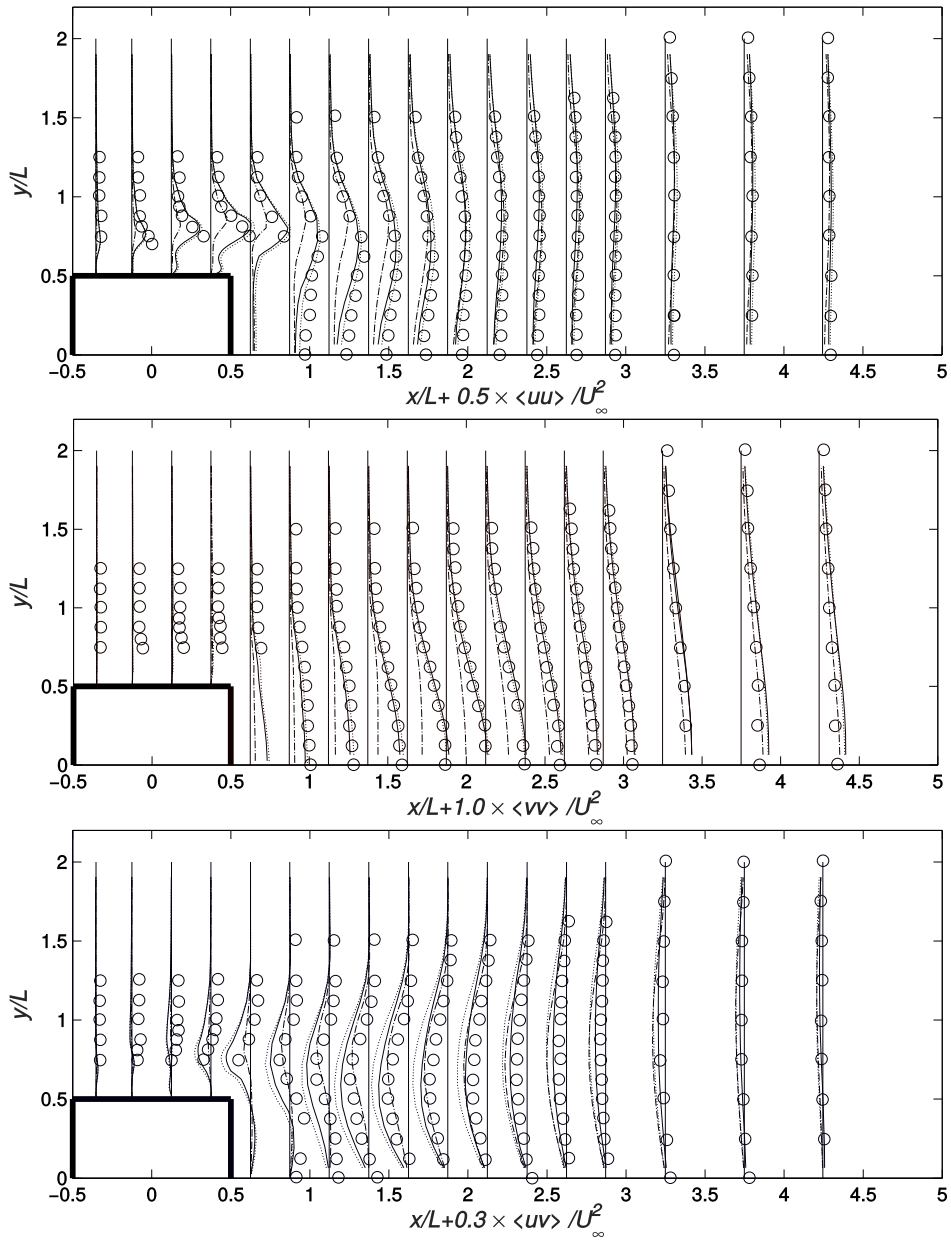


Figure 4.35: Velocity fluctuation profiles $\overline{u'u'}$ (top); $\overline{v'v'}$ (center); $\overline{u'v'}$ (bottom). Comparison of: BagliettoNLEVM ($- \cdot -$); STRUCT ($\cdots$); STRUCTT(—)

determination of streamwise fluctuations, STRUCTT is marginally better when the uv time co-variance is considered.

Activation of STRUCT and STRUCTT in a z section of the domain is shown in Fig. 4.36, while Fig. 4.37 shows the comparison of instantaneous vorticity fields obtained with the baseline U-RANS and those obtained with STRUCTT.

Activation contours show how the model activates in the zones where high velocity deformation is expected. It also shows how the model would never activate if the obstacle was not present leading to the RANS solution (which is accurate enough) in the case of a pure channel flow. Furthermore the figures show how the

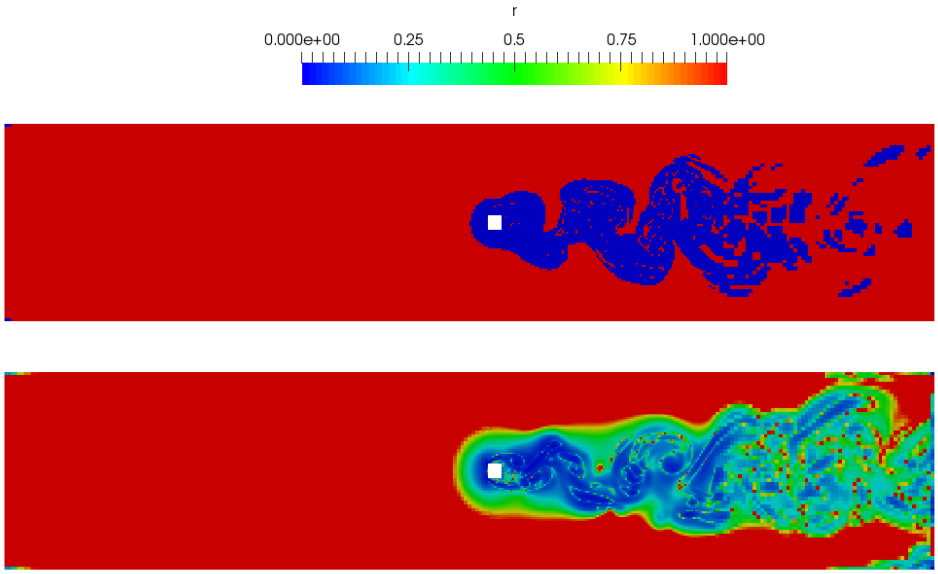


Figure 4.36: Hybrid mode activation parameter for STRUCT (top) and STRUCTT (bottom)

large shedding structures decay into smaller ones, being finally dissipated, and canceling the activation of the hybrid model near the outlet boundary.

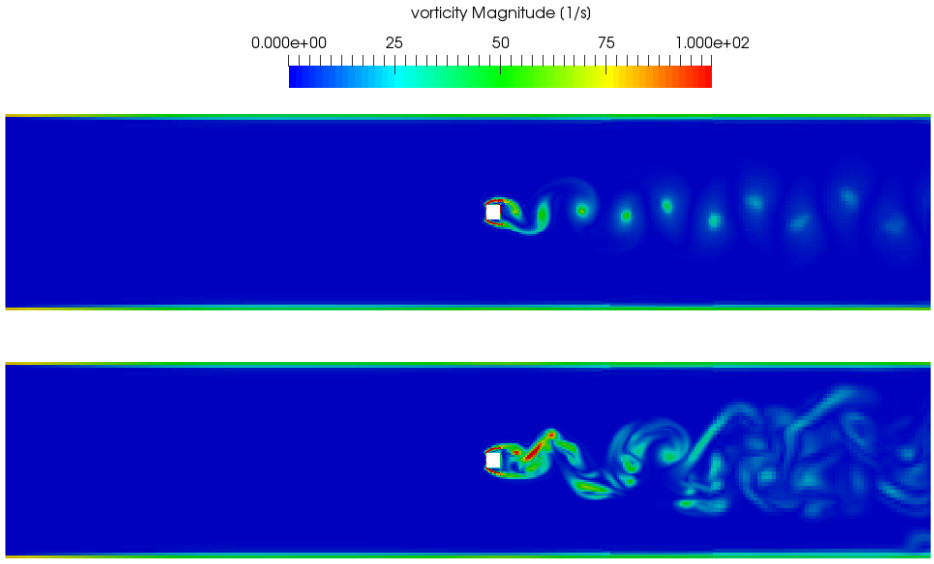


Figure 4.37: Vorticity magnitude for BagliettoNLEVM (top) and STRUCTT (bottom)

Wen vorticity fields are compared, top contour plot of Fig. 4.37 shows how U-RANS predicts a single mode, almost periodic pattern, not being able to capture the breakdown of large vortices into smaller ones downstream of the obstacle as expected, which does appear in the vorticity pattern showed by STRUCTT model.

Even if results were satisfactory for STRUCTT a more detailed study was performed. Fig. 4.38 represents contour plots of the relevant frequency scales: f_r, f_{m0} and f_m, while Fig. 4.39 shows the behavior of the same variables along the

domain centerline past the obstacle ($y = z = 0$ and $x/L > 1$).

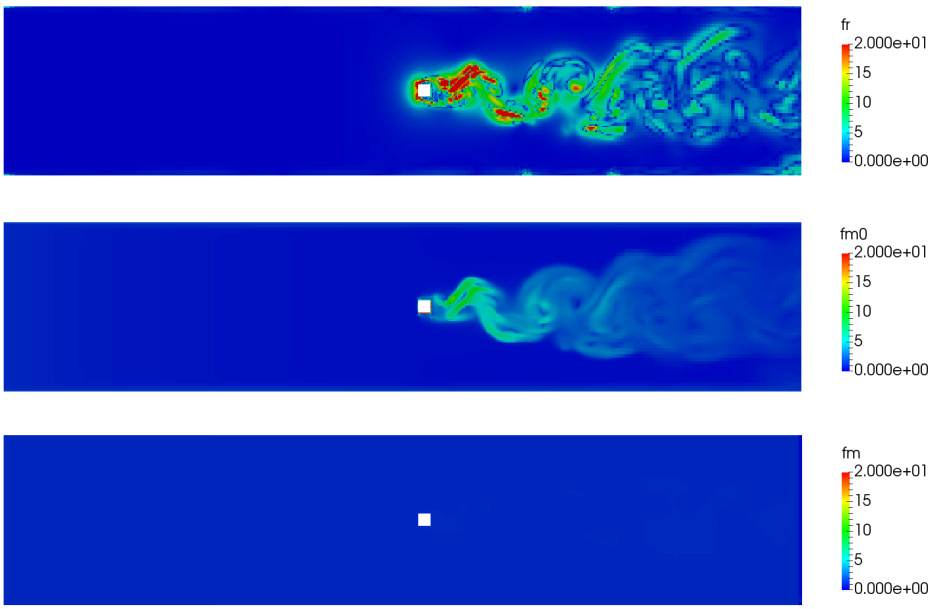


Figure 4.38: Relevant frequency scales for STRUCTT instantaneous solution

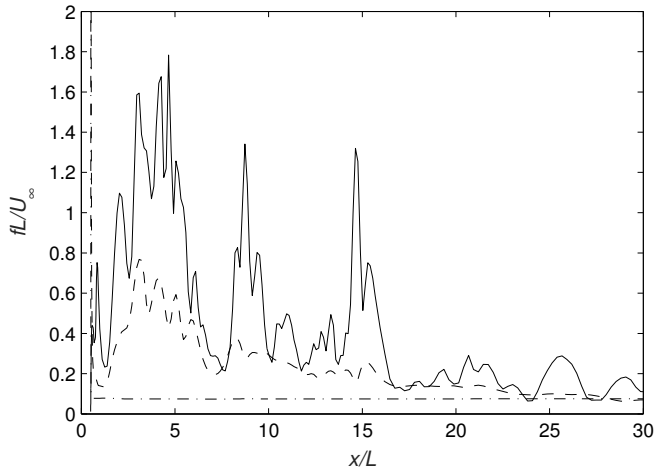


Figure 4.39: Relevant frequency scales for STRUCTT along centerline: f_r (—); f_{m0} ($-\cdot-$); f_m ($- - -$).

Figs. 4.38 and 4.39 show that f_m is not so much of an average of f_{m0}. The small value chosen for β parameter avoids significant variations of t_m and consequently f_m remains close to its initial value, close to the initial condition of f_{m0} at the starting point of the simulation. If we also pay attention to the different terms involved in the additional equation solved for STRUCTT model (Eq. (4.22)), in particular the source and diffusion terms D_{tm} and S_{t_m}, and we plot their values in the centerline as done for the frequency scales, we obtain Fig. 4.40. The relative weight of S_{t_m} is much higher than that of D_{t_m}. This behavior is maintained in the rest of cases studied. Therefore the model could be simplified by neglecting

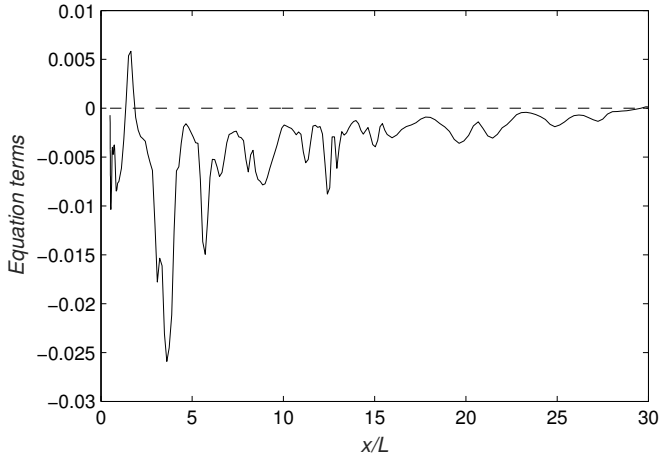


Figure 4.40: RHS terms in the equation for t_m in STRUCTT model: S_{t_m} (—); D_{t_m} (− − −)

D_{t_m}. Furthermore, we would like f_m to adapt faster to an average of f_{m0} in the vicinity of the regions with resolved scales. This could be achieved by considering the integration parameter in the Lagrangian average procedure to be $T = 1/f_r$. As mentioned in section 4.3.3 the model following these ideas has been called STRUCTT-Tf_r.

When the same α and β parameters are used, results are insensible to the selection of the time scale T since, as explained before, f_m remains approximately equal to the initial value of f_{m0}. In order to determine the best general behavior of STRUCTT (without the diffusion term in t_m equation) for the two possible definitions of T: $T = k/(\beta\epsilon)$ (from now on STRUCTT-T$k\epsilon$) and $T = 1/(\beta f_r)$ (STRUCTT-Tf_r), a systematic study has been carried out, varying the parameters α and β respectively in the ranges $(0.5 - 2.0)$ and $(0.001 - 1)$. The mean errors in velocity fields and velocity fluctuations were computed considering all profiles for which experimental data was available. Simulations will not be shown here for the sake of brevity. This study has also been extended to the two other cases that will be presented next: the diffuser case and the triple jet case. When considering the diffuser case, it was found that a sufficiently high value of α was required to guarantee the activation of the model and meaningful results. With this in mind $\alpha = 2$ was chosen for both models as the best compromise. Regarding the effect of β, when $T = k/(\beta\epsilon)$ behavior improved progressively for low values of β, even if this means no actual averaging is performed in f_{m0}. The optimum was therefore $\beta = 0.001$. When $T = 1/(\beta f_r)$, optimum results were found with $\beta = 0.1$. The simulations with STRUCTT-Tf_r were also found less dependent on the choice of α and β in the tested ranges.

Streamwise velocity and velocity fluctuation profiles for the optimum configuration of each model (STRUCTT-T$k\epsilon$ and STRUCTT-Tf_r) are shown in Fig. 4.41.

For this case results are very similar to the ones shown before with the different constants. STRUCTT-T$k\epsilon$ is slightly better predicting velocity profiles but slightly worse in the determination of the x-velocity time variance profiles.

The two fully closed models will next be applied to two additional cases, with

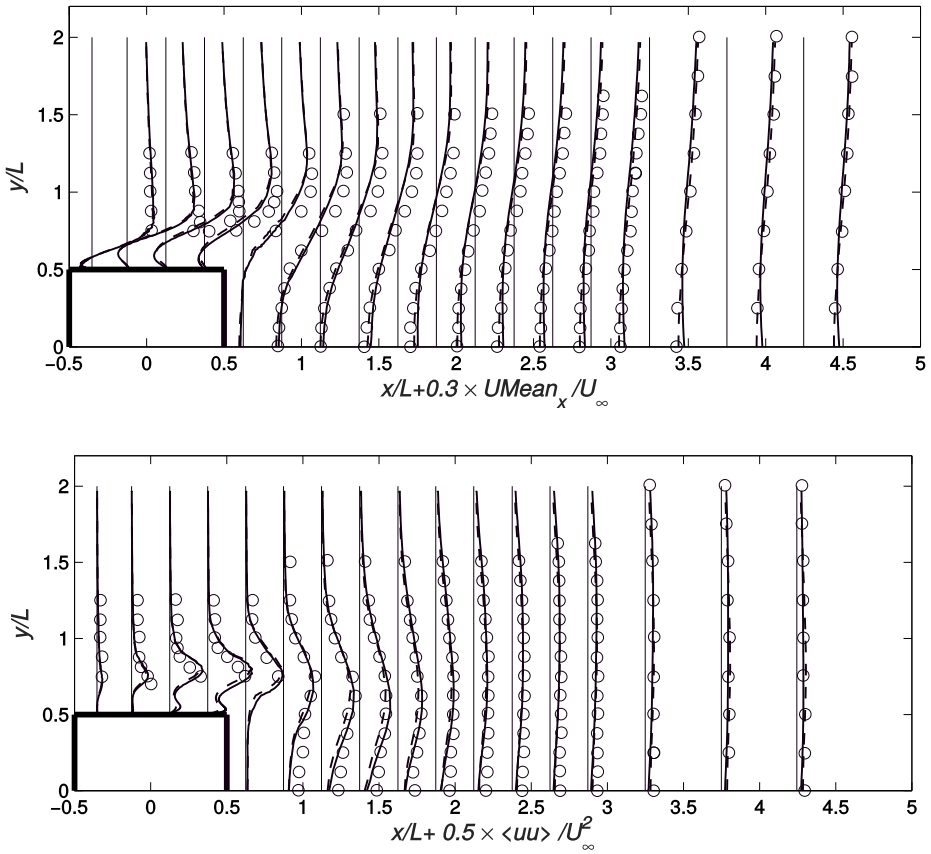


Figure 4.41: x-velocity (top) and velocity fluctuations (bottom) for STRUCT ($- - -$) and STRUCTT-Tf_r (—)

the selection of optimal constants explained before.

Asymmetric diffuser

The plane asymmetric diffuser is a classical case for the study of mild flow separation against a slight adverse pressure gradient. The mild separation makes this test challenging for hybrid models. For instance, Davidson [24] performed simulations with the SAS using $k-\omega$ SST as baseline U-RANS and highlighted the poor performance of the SAS model, which showed lower accuracy than the simulations performed with the standard $k-\omega$ SST in the same mesh. The configuration that will be studied here is shown in Fig. 4.42. A fully developed channel flow encounters a diffuser section with a slope of 10 degrees. The flow separates and a recirculation region appears near the end of the slope. The Reynolds number for the case considered here is $Re = 20000$, based on the height of the inlet channel H and the bulk velocity U_{ref}.

Two meshes have been studied for this case. A 2D mesh which will be used to obtain steady state solution of linear and nonlinear RANS models and 3D one, extrusion of the former in the spanwise direction a total of $L_z = 3H$, which will be used to test U-RANS and hybrid models. Fig. 4.43 shows a representation of them, together with the characteristics specified in Table 4.9.

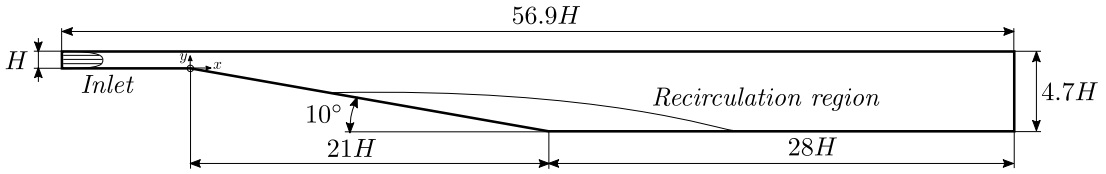


Figure 4.42: Asymmetric diffuser geometry

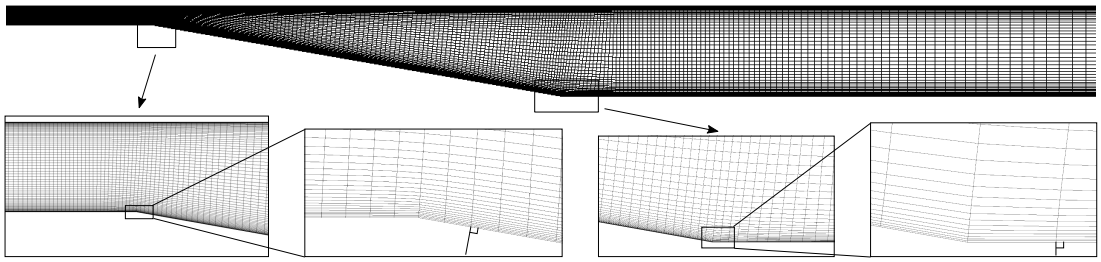


Figure 4.43: Asymmetric diffuser mesh

Table 4.9: Asymmetric diffuser mesh characteristics

Mesh	Number of cells	Max. non-orthogonality [deg]	$y^+_{\max}$
2D	≈ 20400	7.8	0.1
3D	≈ 980000		

Wall mesh refinement has been increased in an attempt for the models to predict the separation point. Periodic conditions apply in the spanwise direction. A fully developed one-dimensional turbulent channel flow simulation (such as the one shown in section 4.5.1), performed with BagliettoNLEVM, has been considered to determine velocity and turbulence quantities profiles at the inlet boundary. Hot wire anemometry measurements from Buice [10] will be used as a reference. Anderson and Eaton [2] estimated the maximum random errors to be 3% for averaged velocity profiles, 5% for velocity time variance and 10% for velocity covariance.

First, simulations are performed with classical RANS models and BagliettoNLEVM in the two-dimensional mesh. Fig. 4.44 shows averaged velocity and velocity fluctuations profiles in comparison with experimental data.

Linear and nonlinear $k-\epsilon$ models fail to predict the separation of the flow. In fact for these models the flow never separates, and streamwise velocity is positive in the entire domain. On the other hand $k-\omega$ SST predicts quite accurately velocity and streamwise fluctuation profiles, but it over-predicts wall normal velocity fluctuations and velocity co-variance. The nonlinear eddy viscosity assumption does not seem to produce a significant improvement in this case.

When considering the performance of hybrid models it was first noticed that the proposed STRUCTT-like models would not activate unless a sufficiently high value of α was used. It seems that when α is not high enough, there are not sufficiently large instabilities in the flow to trigger the hybrid model activation. According to the study mentioned in the last case, STRUCTT-T$k\epsilon$ and STRUCTT-Tf_r have

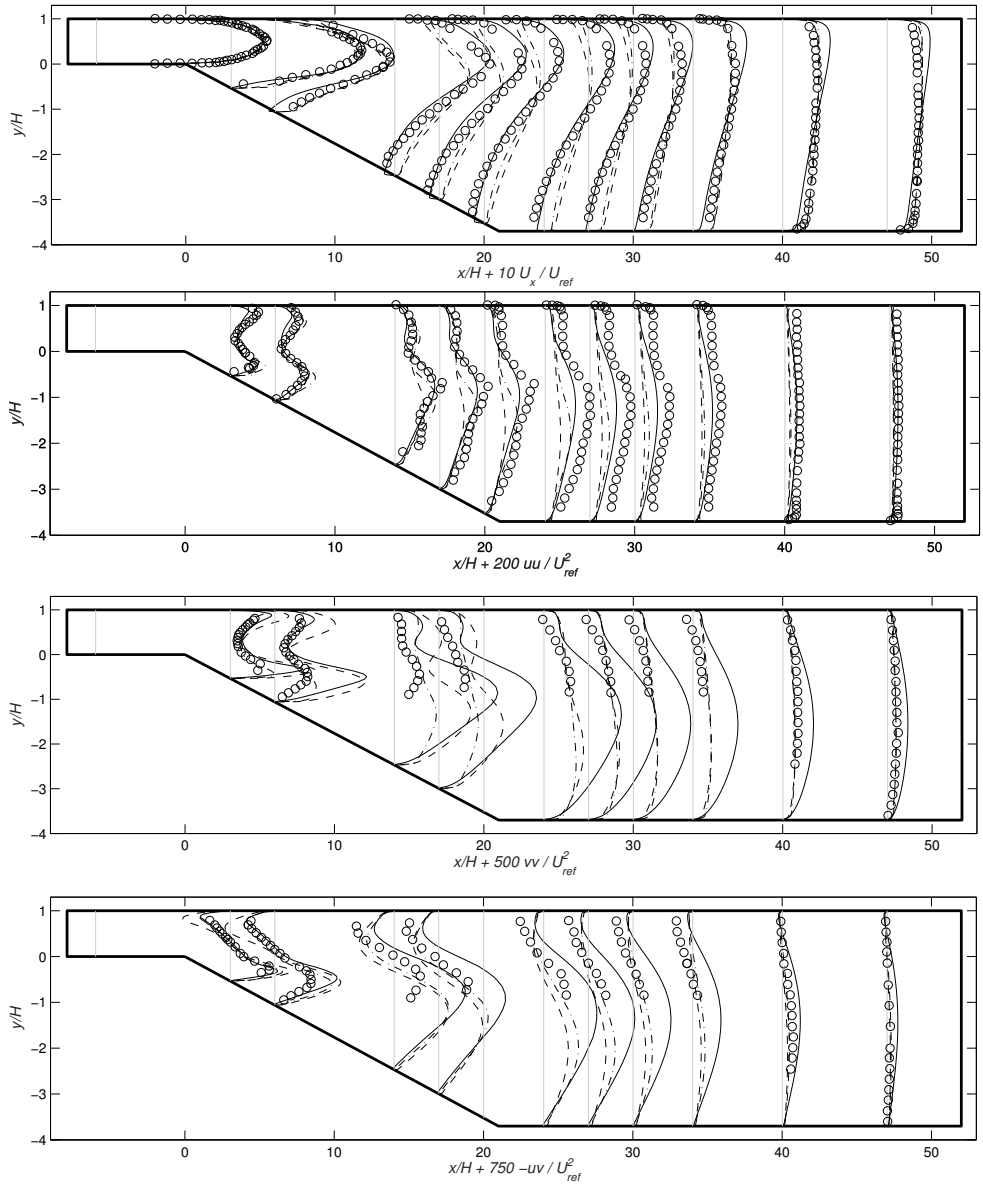


Figure 4.44: Streamwise velocity (first) and velocity fluctuation profiles: uu variance (second), vv variance (third) and uv covariance (fourth) for the two-dimensional simulations: $k-\omega$ SST (—); Launder Sharma $k-\epsilon$ ($-\,-\,-$); BagliettoNLEVM ($-\cdot-$); experimental data ($\circ\,\circ\,\circ$)

been used, using the best globally performing parameters for each of them, to solve the asymmetric diffuser case in the three-dimensional mesh. Activation of both models is shown in Fig. 4.45.

Activation of STRUCTT-Tf_r seems to be enlarged with respect to that of STRUCTT-T$k\epsilon$. In fact, for the latter, f_m stays close to the initial f_{m0}, taken from the inlet channel flow, which is higher than the value downstream. STRUCTT-Tf_r adapts faster to the value downstream, providing a lower value of f_m and therefore enlarging the activation of the model. As commented before, a parametric study was perform to determine the parameters defining the best global behav-

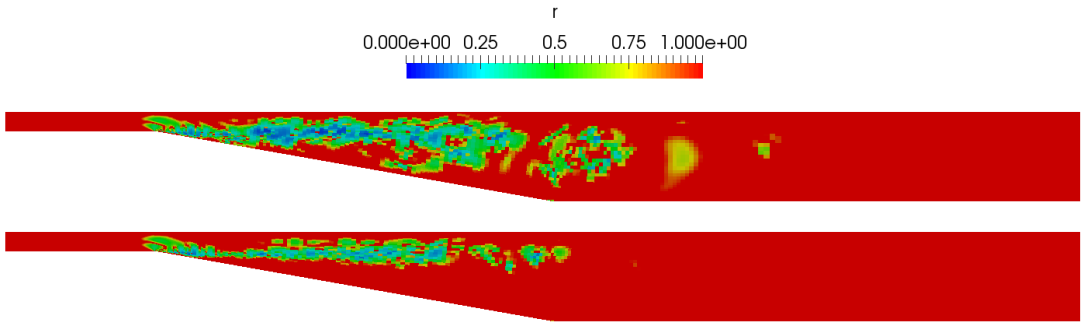


Figure 4.45: Hybrid models activation regions for diffuser case: STRUCTT-Tf_r (top); STRUCTT-T$k\epsilon$ (bottom)

ior of STRUCTT-T$k\epsilon$ and STRUCTT-Tf_r. Fig. 4.46 shows how the velocity fields predicted by STRUCTT-T$k\epsilon$ vary significantly more than those predicted by STRUCTT-Tf_r for different β values.

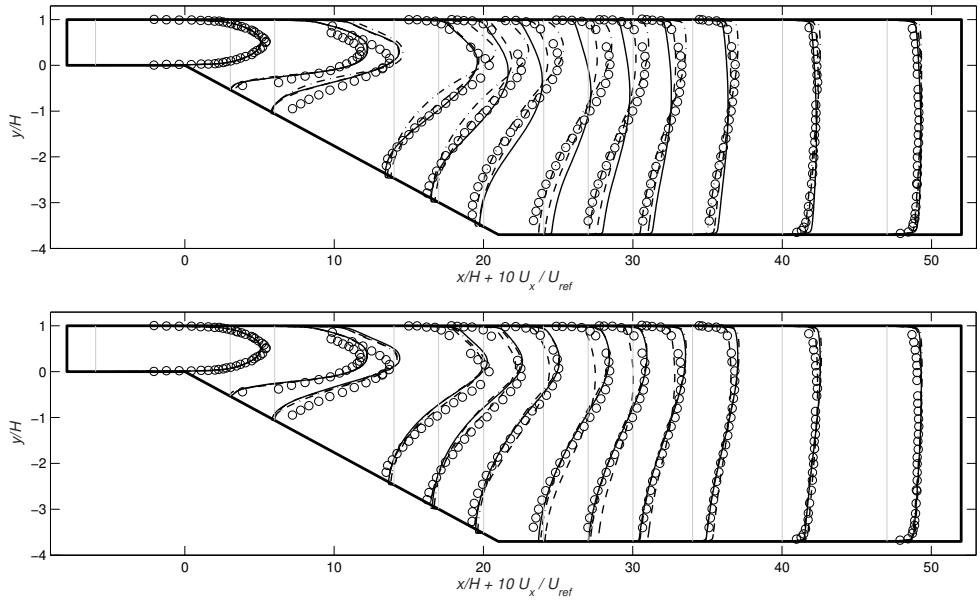


Figure 4.46: Streamwise velocity profiles for STRUCTT-T$k\epsilon$ (top) and STRUCTT-Tf_r (bottom): $\beta = 0.001$ (—); $\beta = 0.01$ (− − −); $\beta = 0.1$ (− · −); experimental data (∘ ∘ ∘)

When the globally best combination of α and β are used for each models, the results of velocity and velocity fluctuation profiles are shown in Fig. 4.47

Both models predict an early separation of the flow. STRUCTT-Tf_r gives a slightly better prediction of velocity profiles, but both cases improve significantly the behavior of the baseline U-RANS, and also the results from $k-\omega$ SST. When considering velocity fluctuations there is a clear overestimation of the velocity variance and covariance peaks in the vicinity of the separation point. This overestimation is however significantly smaller than the one obtained with other classical hybrid approaches (see for instance [24]). Of the two models, STRUCTT-Tf_r ameliorates the over-predicted peaks obtained with STRUCTT-T$k\epsilon$ but does

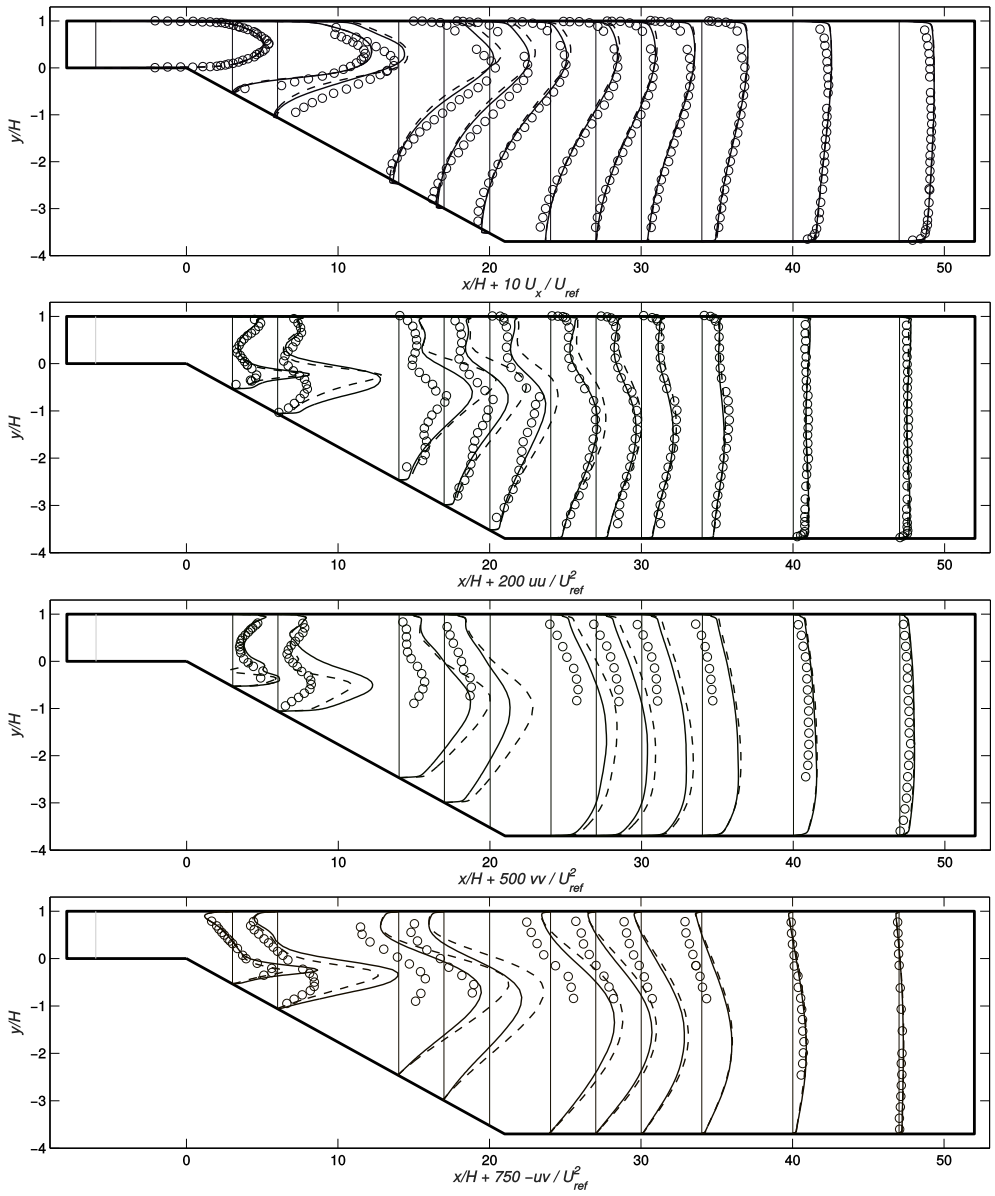


Figure 4.47: Streamwise velocity (first) and velocity fluctuation profiles: uu variance (second), vv variance (third) and uv covariance (fourth) for the three-dimensional simulations: STRUCTT-T$k\epsilon$ ($- - -$); STRUCTT-Tf_r (—); experimental data ($\circ\circ\circ$)

not fully correct the problem.

So far, models have been tested in cases characterized by similar physical phenomena to the flow through a gear pump (separation of the flow in a curved hump in the periodic hill, or vortex shedding around a square cylinder), and cases typically complex for the performance of hybrid models (separation against mild pressure gradient in the asymmetric diffuser). Last case will evaluate the capacity of the model to predict oscillatory mixing of flow streams, in the so-called triple jet case.

Triple jet

Thermal striping is a complex phenomena of particular concern in the nuclear engineering field. It arises when turbulent streams of different temperatures mix in the vicinity of sensitive structural materials, generating temperature fluctuations that can cause thermal fatigue. This has been intensively studied in light water reactors (LWR) when considering coolant mixing in T-Junctions [106] and in Sodium Fast Reactors (SFR) where the oscillatory mixing of hot sodium coolant streams from the core sub-assemblies and cold sodium flows from control rod channels and blanked fuel assemblies, can damage the upper-plenum materials [42]. Accurate simulation of this phenomena is essential for design and operation of nuclear reactors.

The configuration chosen here is a parallel triple jet sodium flow, presented as a benchmark for striping phenomena by the Japan Atomic Energy Agency (JAEA) under the name PLAJEST. This involves a large series or related experiments [70, 71, 134, 135] which will be used as a reference for comparisons of velocity, temperature and temperature fluctuations. The experimental setup is shown in Fig. 4.48.

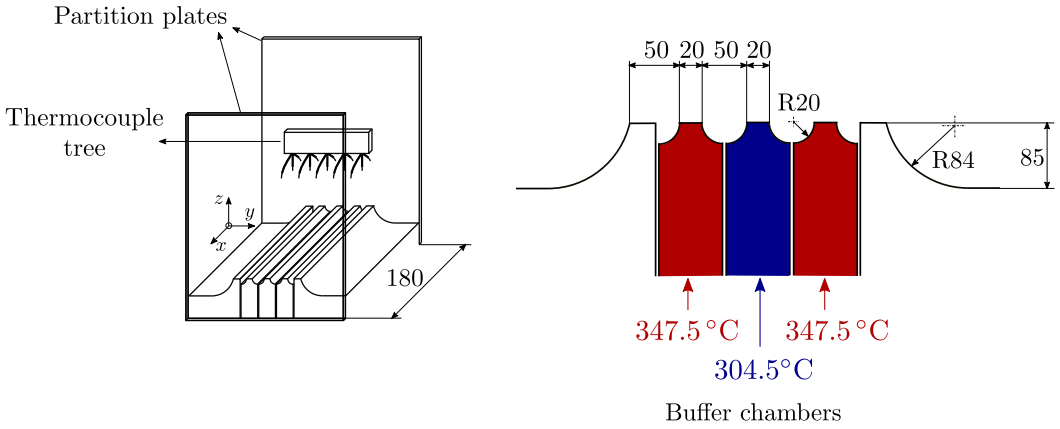


Figure 4.48: Schematic of the experimental setup for PLAJEST benchmark. Distances are expressed in mm

Only one case will be studied, referred to as A1 iso-velocity case, for which the experimental conditions are shown in Table 4.10.

Table 4.10: Experimental conditions for the triple jet case

Case	Outer-slits/hot jets		Center -slit/cold jet		Mixture		
	V_h (m/s)	T_h (°C)	V_c (m/s)	T_h (°C)	V_m (m/s)	ΔT (°C)	T_m (°C)
A1	0.51	347.5	0.51	304.5	0.51	43	333.2

where V_h and V_c are the mean velocity of hot and cold jets and V_m is the mean discharged velocity. Discharged temperature difference is simply defined as $\Delta T = T_h - T_c$ and the mixed-mean temperature is estimated as $T_m = (V_c T_c + 2V_h T_h)/(3V_m)$.

The computational domain and mesh schematic is shown in Fig. 4.49, while mesh characteristics are shown in Table 4.11. A very coarse mesh has been con-

sidered in this case in order to show the applicability of the models in such case and the improvements with respect to the baseline U-RANS, in a mesh created for this type of approach. The origin is placed at the center of the cold jet on the nozzle outlet and on the partition plate wall surface. Liquid sodium density, heat capacity and transport properties have been expressed as a polynomial function of temperature, as described in [36]. Walls are treated with wall functions, ensuring the proper behavior of these and they are considered well isolated. Lateral and top boundaries are treated as outlet boundaries as recommended by [144]. Fully developed turbulent flow has been considered at the different inlet boundaries.

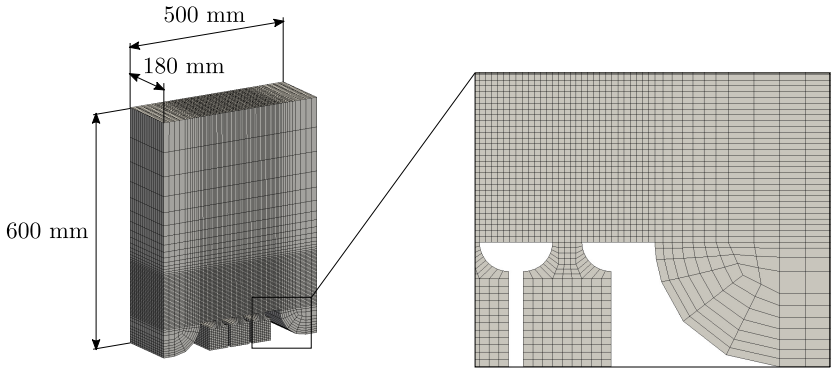


Figure 4.49: Schematic of the computational mesh

Table 4.11: Periodic hill case mesh characteristics

Number of cells	Max. non-orthogonality [deg]	y^+_{min}	$y^+_{\max}$
≈ 8300	31	45	230

RANS models, as expected, are not able to predict properly temperature fluctuations [144], which forces the use of U-RANS and LES techniques. Here we will analyze the performance of the selected STRUCTT-like methods in comparison with U-RANS solutions.

Fig. 4.50 shows the predicted temperature contours at the middle section ($x/D = 4.5$ where $D = 20$ mm is the nozzle width) for BagliettoNLEVM, STRUCTT-T$k\epsilon$ and STRUCTT-Tf_r.

While predictions of the two hybrid models considered here are similar, differences with the baseline RANS model are obvious. BagliettoNLEVM predicts a much larger diffusion of temperature, probably due to high estimation of turbulent viscosity that enhances the mixing of coolant flows. Regarding the comparison of hybrid approaches, Fig. 4.51 shows the activation regions for both models, which are again very similar.

In order to compare the models, experimental measurements of mean velocity, temperature and temperature fluctuations in the center plane ($x/D = 4.5$) at $z = 100$ will be used. Simulation results have been averaged for over 10 seconds. All velocity components are normalized with the discharge velocity $V_{exit} = 0.51$ m/s. Temperature is normalized according to Eq. (4.34).

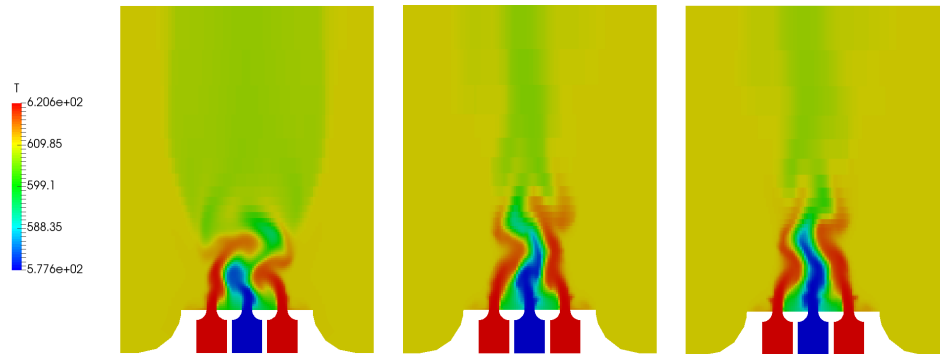


Figure 4.50: Velocity magnitude (top) and temperature (bottom) contours: BagliettoNLEVM (left), STRUCTT-T$k\epsilon$ (center); STRUCTT-Tf_r (right)

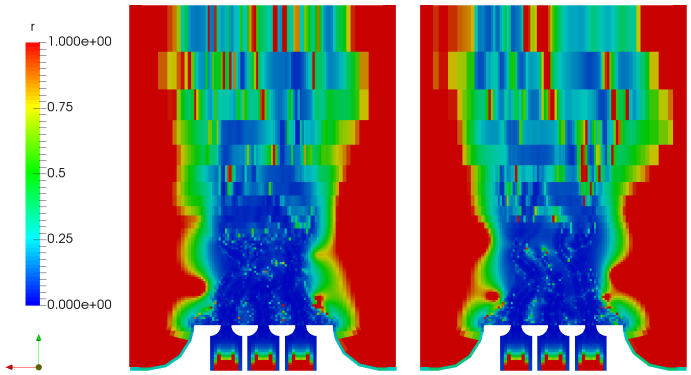


Figure 4.51: Hybrid model activation regions: STRUCTT-T$k\epsilon$ (left); STRUCTT-Tf_r (right)

$$T^* = \frac{T - T_c}{\Delta T} \tag{4.34}$$

Temperature fluctuation are compared by determining the normalized temperature fluctuation intensity (TFI^*), defined with a normalization of the root mean square of T, as shown in Eq. (4.35)

$$TFI^* = \frac{T_{rms}}{\Delta T} \tag{4.35}$$

Fig 4.52 shows the comparison of averaged velocity and mean temperature and temperature fluctuations along the described centerline.

Results for hybrid models are again superior to those of nonlinear U-RANS. It appears that BagliettoNLEVM overpredicts turbulent viscosity producing excessive mixing, that translates into under-prediction of velocity and temperature profile peaks. If we focus on the velocity fields, results of both hybrid models are very similar. Non of them succeeds into capturing the trend of horizontal and vertical velocities near $y = 0$ probably due to insufficient mesh resolution. Magnitude of vertical velocity fluctuations is slightly over-predicted but still close to the experimental value. Both hybrid models capture accurately the mean temperature

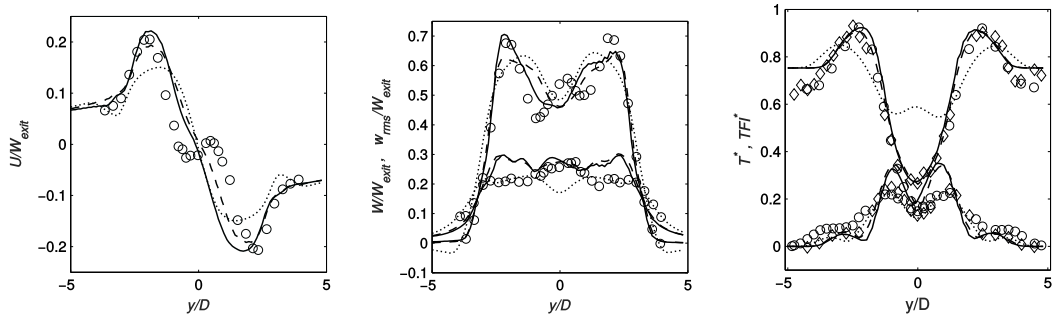


Figure 4.52: Comparison of x-velocity (left), z-velocity and z-velocity fluctuations (center) and temperature and temperature fluctuations (right) along centerline ($x/D = 4.5$) at $z = 100$ mm: BagliettoNLEVM ($\cdots$); STRUCTT-T$k\epsilon$ (—); STRUCTT-Tf_r ($- - -$); experimental data first set ($\circ\circ\circ$); experimental data second set ($\diamondsuit\diamondsuit\diamondsuit$)

distribution and they predict the right trend in temperature fluctuations even if the peaks of these are slightly over-predicted by both models. BagliettoNLEVM fails to predict even mean temperature profiles.

4.6 Conclusions

In the present chapter new hybrid models have been presented. On the basis of the STRUCT model developed by Lenci [75] several alternatives have been proposed and tested. The chapter introduces the theory behind hybrid models and gives some notes about some of the most typically used ones nowadays. Attention is then paid to the description of BagliettoNLEVM, the baseline nonlinear RANS model of STRUCT, on the rationale behind STRUCT approach and on the different versions of STRUCT proposed by Lenci. The implementation of the nonlinear model is first tested in several academic cases, focusing on those which are challenging for linear models and for which nonlinear models should improve the results. BagliettoNLEVM has proved to provide accurate predictions of velocity and turbulent kinetic energy in the considered academic cases, and to improve the description of turbulence anisotropy when compared to other classical alternatives. Given these results, the different versions of STRUCT are tested in a variety of cases. Failure of some of the approaches for some cases leads to the proposition of alternative models. In particular STRUCTT-Tf_r, a derivation of the STRUCTT version of STRUCT, has been proposed and compared to the latter, giving equal or better results and being based on more physical assumptions. Both have been proved to always improve accuracy with respect to the baseline U-RANS solution, in cases with increasing complexity. Some of the cases chosen here resemble the phenomena found in a gear pump and therefore the model is proposed for its use in the simulation of three-dimensional gear pumps.

Chapter 5

Application examples

5.1 Introduction

This chapter will provide some examples of the application of the methods proposed in previous chapters to the simulation of gear pumps. Since the project was developed in collaboration with a gear pump manufacturing company, one of their helical gear pumps has been selected for this study. In particular an oil pumping external gear pump of reduced dimensions, wiht $z = 7$ teeth, with a large range of pressure head and velocity working conditions will be analyzed. Due to confidentiality reasons, neither the profile nor the working conditions, geometry details or fluid properties will be given, and results will be referenced to unspecified random parameters which will be denoted by a star '$*$'. For brevity only some examples of the performed simulations will be given.

Regarding p-U coupling a transient SIMPLE-C method with Majumdar correction will be applied to all simulations. With respect to the mesh motion strategy, the rounded profile of the studied pump is a clear example for which *fréchet* method is preferred. Interface boundary conditions are treated with the AMI-ACMI boundaries as described in section 3.9.2. The generalized parallelization method described in section 3.12 has been used together with *scotch* decomposition. The new method proposed in this thesis for the treatment of contact points (section 3.13) has been used for all cases unless stated otherwise. Laminar, classical linear models or BagliettoNLEVM (section 4.3.2) have been used for turbulence modeling as specified in each case. Absolute pressure and total pressure have been fixed respectively at inlet and outlet boundaries depending on the working conditions in each case, and mass flow and torque have been computed and compared when possible to experimental data. The lack of extensive experimental measurements makes it difficult to provide a clear validation of the proposed strategy, however working condition maps of the gear pump under study will be provided together with the simulation predictions, and observed tendencies will be given for other parameters, for which no experimental results were available.

5.2 2D cases

This section provides a broad analysis of different working conditions of the studied gear pump. The speed of the calculations allow us to perform a large number of simulations to study different parameters.

5.2.1 Mesh dependency

As a starting point, laminar simulations of the 2D simplification of the helical pump have been performed. Geometric variables are shown in Fig. 5.1 while Fig. 5.2 shows two of the meshes used in this section, whose characteristics are specified in Table 5.1.

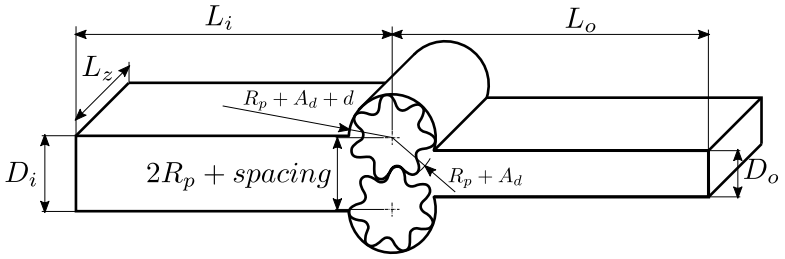


Figure 5.1: Two-dimensional gear pump geometry

In the 2D simulations the pump is assumed to be infinitely long in the z direction and therefore the behavior would be the corresponding to spur gears. When comparing to real measured data, results will be scaled in the z-direction assuming a L_z length that is selected to maintain the same inlet area as the real case (its details cannot be provided), following Eq. (5.1).

$$\frac{\pi D_i^2}{4} = D_i L_z \tag{5.1}$$

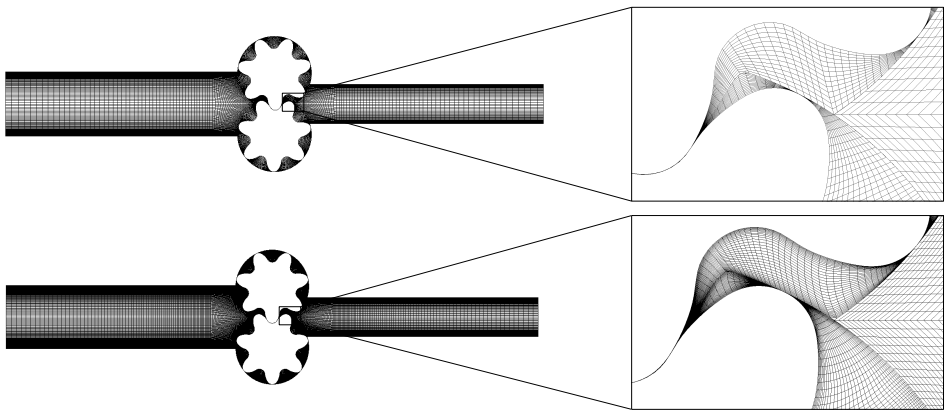


Figure 5.2: Two-dimensional gear pump meshes

Inlet and outlet boundaries have been placed far from the region of interest ($L_i = L_o = 9R_p$) to guarantee that the vicinity of the boundary is not affecting the results. Mesh characteristics obviously change as the mesh moves and deforms.

Table 5.1: Characteristics of 2D gear pump meshes

Mesh	Number of cells	Max. non-orthogonality [deg]	Avg. non-orthogonality [deg]	Max. Skewness
Coarse	≈ 21800	62	16	2.21
Fine	≈ 58700	65	12	2.6

The given parameters are determined by the worst conditions at which each mesh was found in its movement. Initial mesh was generated trying always to pass OpenFOAM® *checkMesh* application, which guarantees an adequate performance of the solvers.

Given the symmetry of the 2D mesh, all results are periodic, with a periodicity that is given by $T/(2Z)$ where T is the rotation period of each gear, as obtained by the rotational speed n. Fig. 5.3 shows for instance the integrated volumetric flow through the pump, per unit length in z direction q, when the pump runs at $n/n^* = 1500$ under a pressure head of $\Delta p/\Delta p^* = 150$, computed in each of the meshes for $d/d^* = 10$ and $spacing/spacing^* = 20$, after a steady fluctuations range has been reached. No contact point treatment is considered in this case.

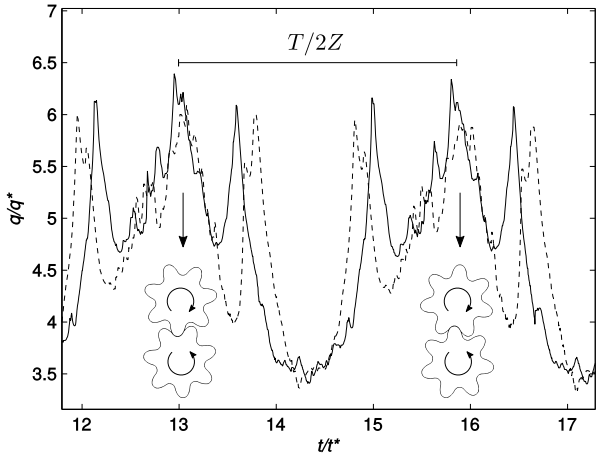


Figure 5.3: Volumetric flux through the 2D pump: coarse mesh (– – –); fine mesh (—)

While the maximum and minimum mass flow is similar between the two meshes, the oscillations magnitude and position change with the mesh resolution. However the averaged volumetric flow per unit length for these two meshes are respectively $\widehat{q_{coarse}}/q^* = 4.731$ and $\widehat{q_{fine}}/q^* = 4.704$.

Integration of pressure and viscous forces at gears wall boundaries allow us to determine the total torque M per unit length for each mesh at each time step, as shown in Fig. 5.4.

Large oscillations are observed in the time evolution in torque for both meshes. In fact, the main contribution comes from pressure forces. Large pressure oscillations probably due to the convergence of the solver lead to similar oscillations in the computed torque. When integrating the torque and computing the time av-

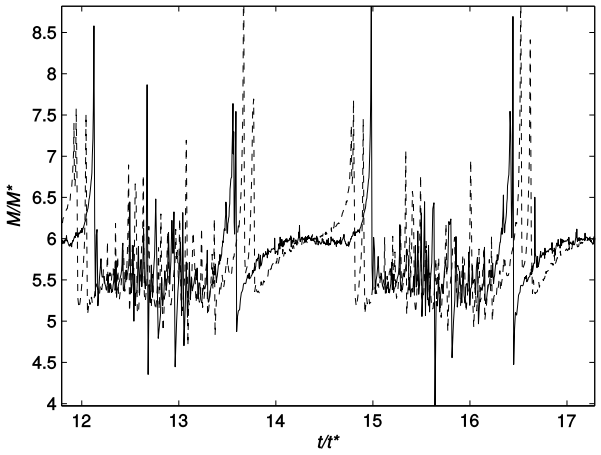


Figure 5.4: Total torque at gear wall boundaries: coarse mesh (– – –); fine mesh (—)

erage values, $\widehat{M_{coarse}}/M^* = 5.7517$ and $\widehat{M_{fine}}/M^* = 5.763$ are obtained for coarse and fine mesh respectively.

The difference in mass flow and torque (and therefore power) is in the order of 0.2%.

If the contact point is properly treated in the fine mesh, the results differ greatly. Fig. 5.5 shows the comparison of volumetric flow per unit length with and without the point treatment, while Fig. 5.6 shows the velocity vectors in the vicinity of the contact point.

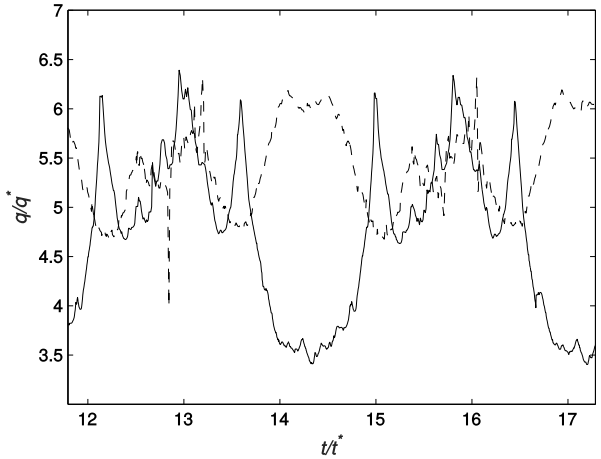


Figure 5.5: Volumetric flux through the 2D pump: with contact point treatment (– – –); without contact point treatment (—)

Given the low computational cost the fine mesh has been chosen for further analysis, and contact point treatment will always be used hereafter.

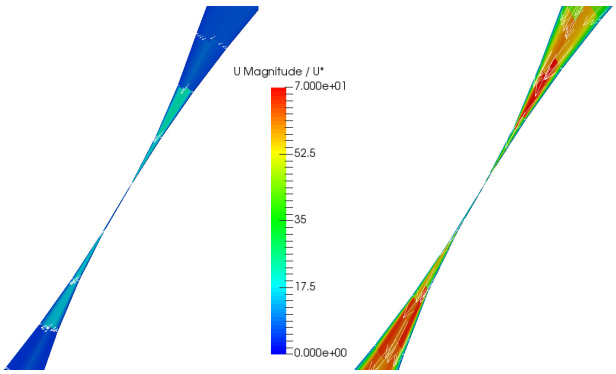


Figure 5.6: Flow through contact point: with contact point treatment (left); without contact point treatment (right)

5.2.2 Internal clearances

As explained in section 1.2, one of the parameters that can significantly influences the volumetric efficiency of a pump are the internal clearances that lead to the radial leakage. This strongly depends on the manufacturing process. Typically clearances can range from 1 to 20 μm. When the fine mesh is considered at $n/n^* = 1500$, $\Delta p/\Delta p^* = 239$ and $spacing/spacing^* = 20$, and radial clearance d is varied, volumetric flow and total torque (considering already L_z length units in the z-direction) vary according to Fig. 5.7.

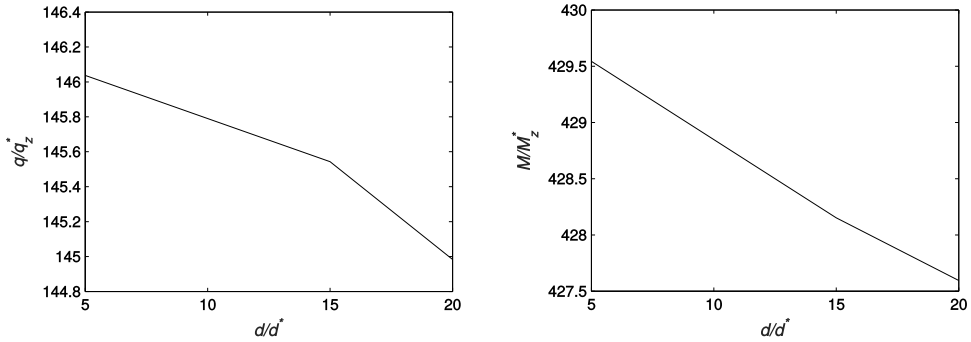


Figure 5.7: Volumetric flow and total torque as a function of circumferential clearance d

The simulations provided the expected tendency. For the same pressure head, a higher clearance reduces the pump delivered flow and the required power.

When simulations at d/d^*= 5, 15 and 20 are considered at the same exact angle and velocity fields are extracted at one of the limits between chambers (see Fig. 5.8).

When the velocity in tangential t_c-direction, U_t is represented as a function of the distance to the gear wall r_c in the normal direction, Fig. 5.9 can be obtained.

Two conclusions can be extracted from Fig. 5.9. First, the parabolic shape of the profiles informs of a laminar flow, as expected in such a reduced section. Secondly, the integration of those curves determines the loss of output flux of the pump, one of the contributions to reduce the volumetric efficiency, and it can be

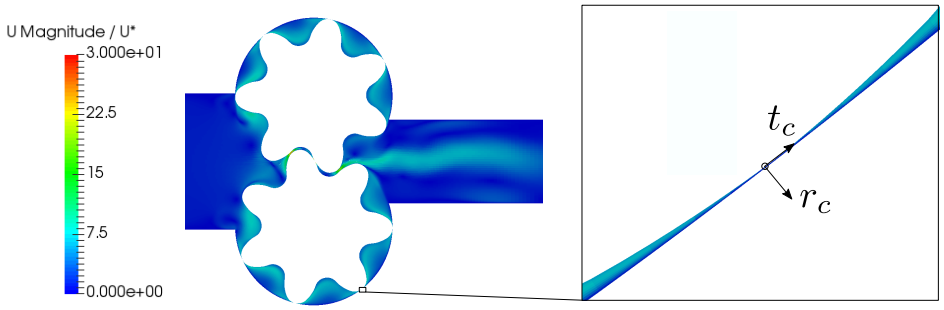


Figure 5.8: Schematic of the region around the studied clearance section

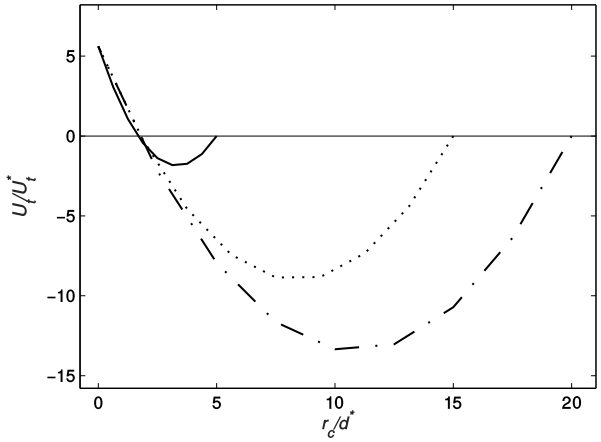


Figure 5.9: Tangential velocity at clearance: $d/d^* = 5$ (—); $d/d^* = 15$ ($\cdots$); $d/d^* = 20$ ($- \cdot -$)

deduced from Fig. 5.9 that it drastically increases with d.

When selecting the artificial separation defined as *spacing*, no difference was found in the results when it was varied in the same range as d if the contact point was treated.

5.2.3 Turbulence modeling

Two turbulence models have been tested in this case: $k-\omega$ SST and BagliettoNLEVM. For this particular case and working conditions the results did not vary much with the model chosen when considering global variables as the averaged flux, for which differences were in the range of 1%. This does not imply however that larger differences might exist when studying other working conditions, pump designs or structure or generated vortices.

Fig. 5.10 shows a comparison of the contours of turbulent kinetic energy for the two turbulence models selected here, while Fig. 5.11 compares the average flux through the pump as determined by the simulations performed with no model (laminar), $k-\omega$ SST or BagliettoNLEVM.

Two main conclusions can be inferred from these results. First, vortex shedding coming out of the pump as a result of the gear motion do generate production of energy that cannot be dissipated by the laminar viscosity, leading to the prediction

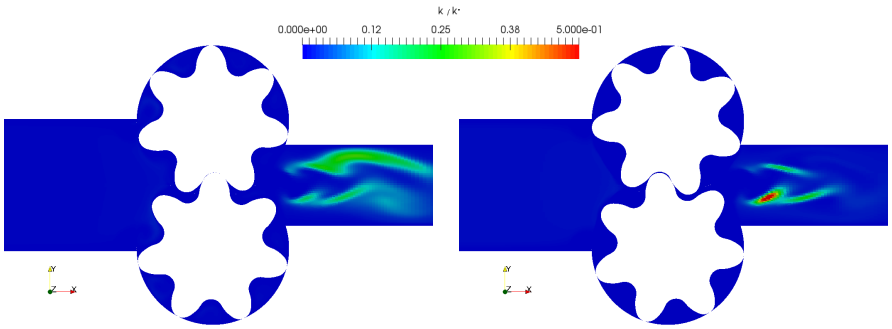


Figure 5.10: Turbulent kinetic energy contours for $k-\omega$ SST (left) and BagliettoNLEVM (right)

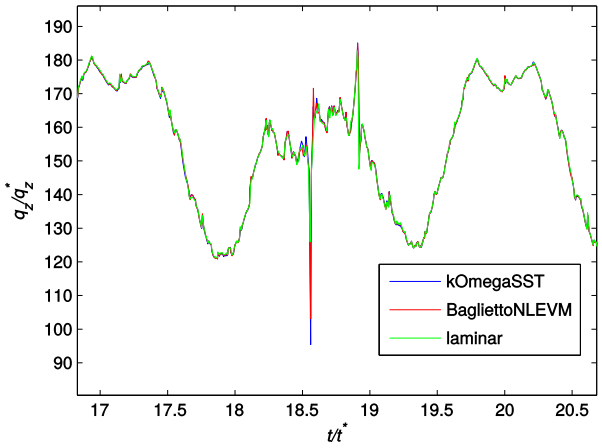


Figure 5.11: Averaged flux through the pump in laminar and turbulent cases

of a non zero turbulent kinetic energy. Second, the similar profiles in the mass flux ripple informs that turbulent does not play an important role in this particular case.

5.2.4 Power-torque performance map

In this section simulations with the fine mesh have been carried out considering fixed $spacing/spacing^* = 20$, $d/d^* = 10$ and no turbulence model, in a several working conditions, varying the pressure head and the rotational speed of the pump independently. Fig. 5.12 shows for instance the evolution of velocity and pressure fields when rotational speed is kept fixed and output pressure is rised.

At higher output pressure the extension of the wake in the center of the outlet channel seems to be reduced. In fact, a reduction of the total mass flow is also expected as the pressure rises, since this causes a reduction on the volumetric efficiency.

Similarly, Fig 5.13 shows the change in velocity contours while Fig. 5.14 compares the volumetric flow evolution when different rotational speeds are considered for the same pressure head.

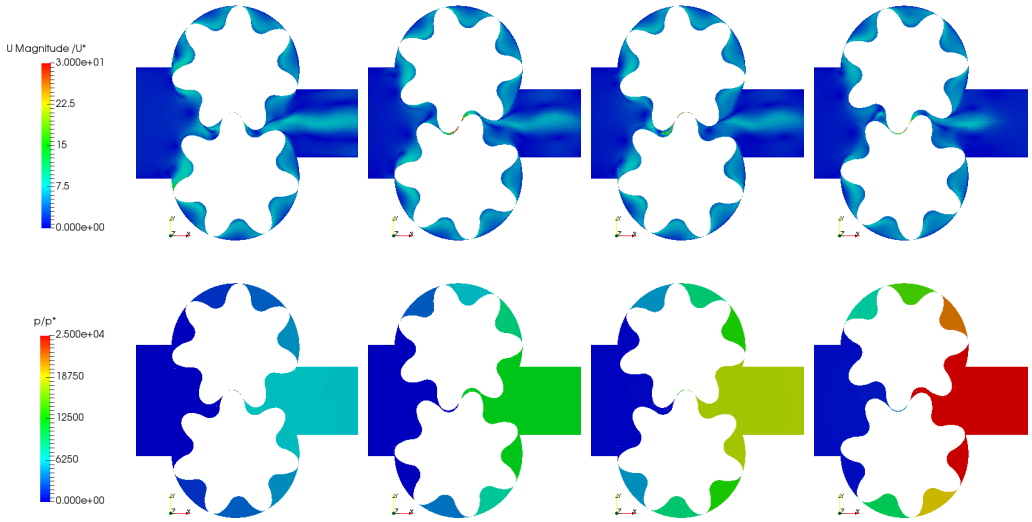


Figure 5.12: Velocity (top) and pressure (bottom) contours for pump simulations at $n/n^* = 1500$. Cases from left to right correspond to: $\Delta p/\Delta p^* = 50$, 100, 150 and 250

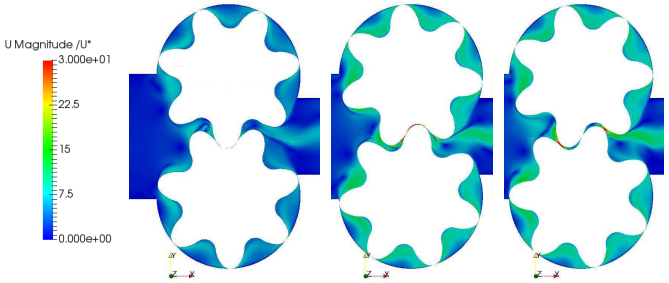


Figure 5.13: Velocity contours for pump simulations at $\Delta p/\Delta p^* = 50$ varying the rotational speed: $n/n^* = 1000$ (left), 1500 (center), 2000 (right)

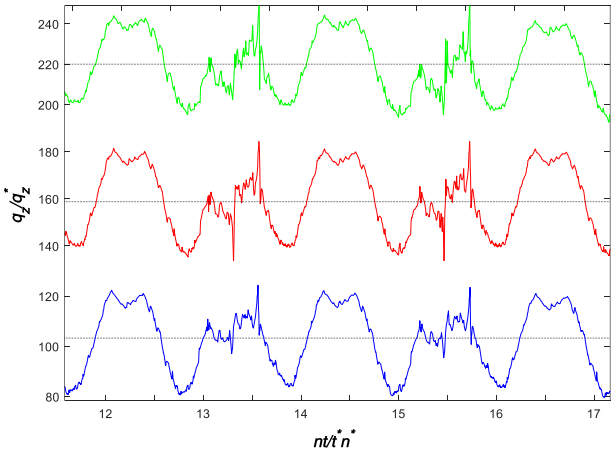


Figure 5.14: Volumetric flux for pump simulations at $\Delta p/\Delta p^* = 50$ varying the rotational speed: $n/n^* = 1000$ (blue), 1500 (red), 2000 (green) togehter with the mean value $(\cdots)$

The rotation velocity seems to have a reduced importance on the scaled volumetric flow estimation. The flow ripple is similar for the three cases, for the particular pump under study.

The computation of pressure and viscous forces determines again the total torque required for both gears, the sum of which can be compared to the experimental value. When this is performed at the several working conditions tested here, Fig. 5.15 is obtained, where torque and power are plot against their experimental value in the real three-dimensional pump.

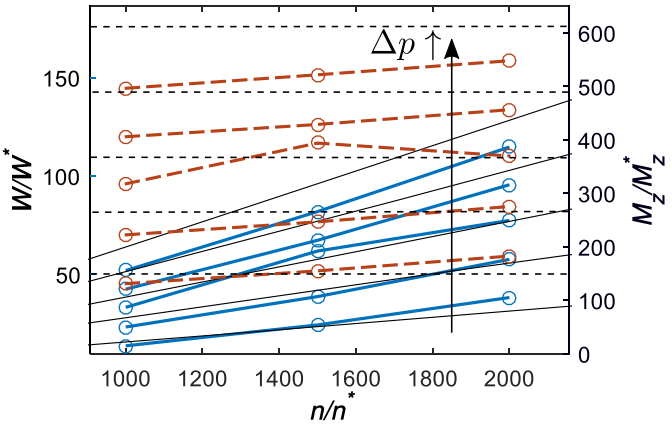


Figure 5.15: Power-torque performance map. Power (—) and torque (– – –) are represented as a function of the rotational speed n for increasing pressure head: $\Delta p/\Delta p^* = 50$, 100, 150, 200, 250. Comparison between simulation results (colored) and experimental results (black)

The two-dimensional simulations provide reasonable agreement with experimental data, even if some major differences can be found. First while torque variation with rotational speed is minimum in the real pump, the simulation provides in general an increase in the required torque for increasing velocities. It should also be commented that the error in the estimation of torque and power increases for increasing pressure heads. We should always consider that these are two-dimensional simulations, an artificial extrapolation of the real pump behavior. The real pump is helical, it has pressure relief groves, cylindrical inlet and outlet pipes and some additional three-dimensional effects (such as axial leakage for instance) cannot be taken into account. In any case two-dimensional simulations can provide an approximation of the expected power-torque maps and the behavior of the pumps when design parameters or operation conditions are varied.

5.3 3D case

In this section some information regarding the three-dimensional simulation of the gear pump will be provided. While temporal limits did not allow me to perform the all the required tests, the starting point is given and future work will be described in order to continue the work started here.

A block structured mesh has been generated for the *fixedCells* region using OpenFOAM® *blockMesh* utility, while the helical gears region has been created with the method presented in section 3 using the parameter $d/d^* = 10$ and

$spacing/spacing^* = 20$. Fig. 5.16 shows an schematic of the entire mesh, whose characteristics are specified in Table 5.2.

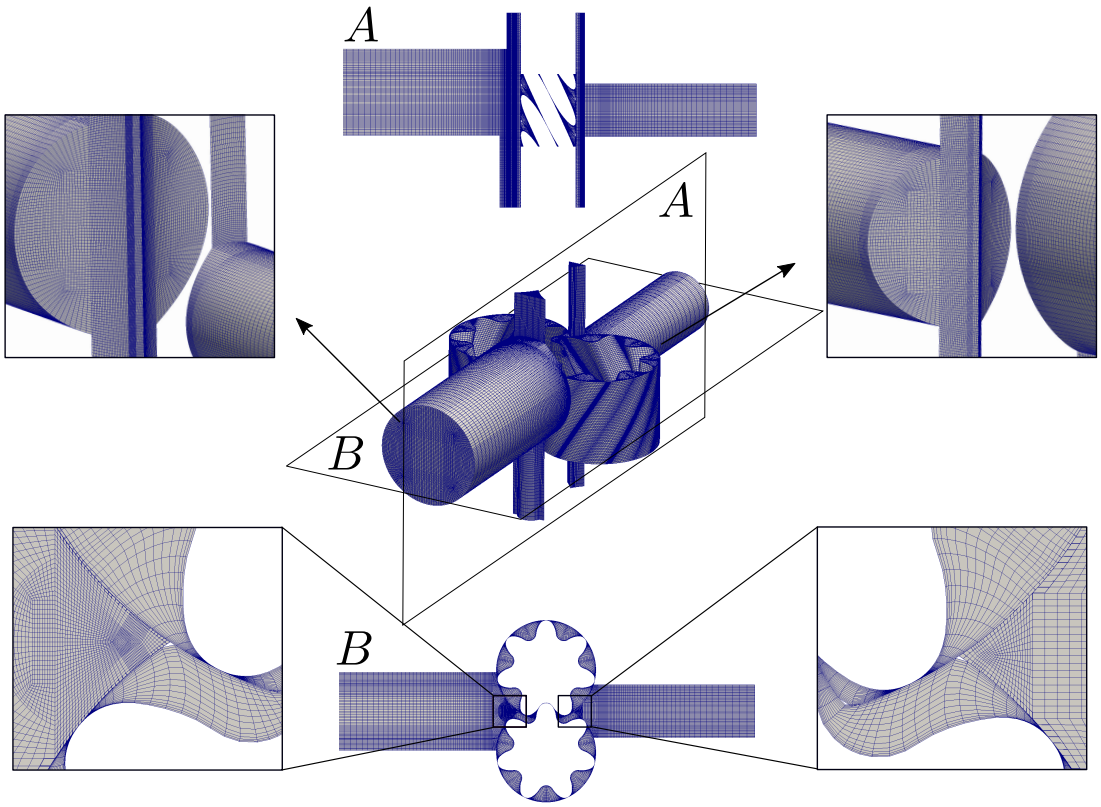


Figure 5.16: Three-dimensional gear pump mesh

Table 5.2: Characteristics of 3D gear pump meshes

Number of cells	Max. non-orthogonality [deg]	Avg. non-orthogonality [deg]	Max. Skewness
$\approx 1.42M$	67	19	1.88

The complexity of the *fixedCells* mesh region increases when the 3D case is considered. Since this fixed part also needs to be modified when studying the influence of some of the parameters (such as d or *spacing*), a script using OpenFOAM® *blockMesh* application has been creating in order to automatically generated the mesh by providing input parameters such as d, *spacing*, pipe lengths and diameters.

The conditions $n/n^* = 1500$, $\Delta p/\Delta p^* = 250$ have been used to start the simulation. The current state is shown in Fig. 5.17.

Pressure distribution is obviously similar for the different planes considered since the chambers are connected in the helical structure. No significant vortical structures appear at this point yet. When considering the inlet pipe, flow moves

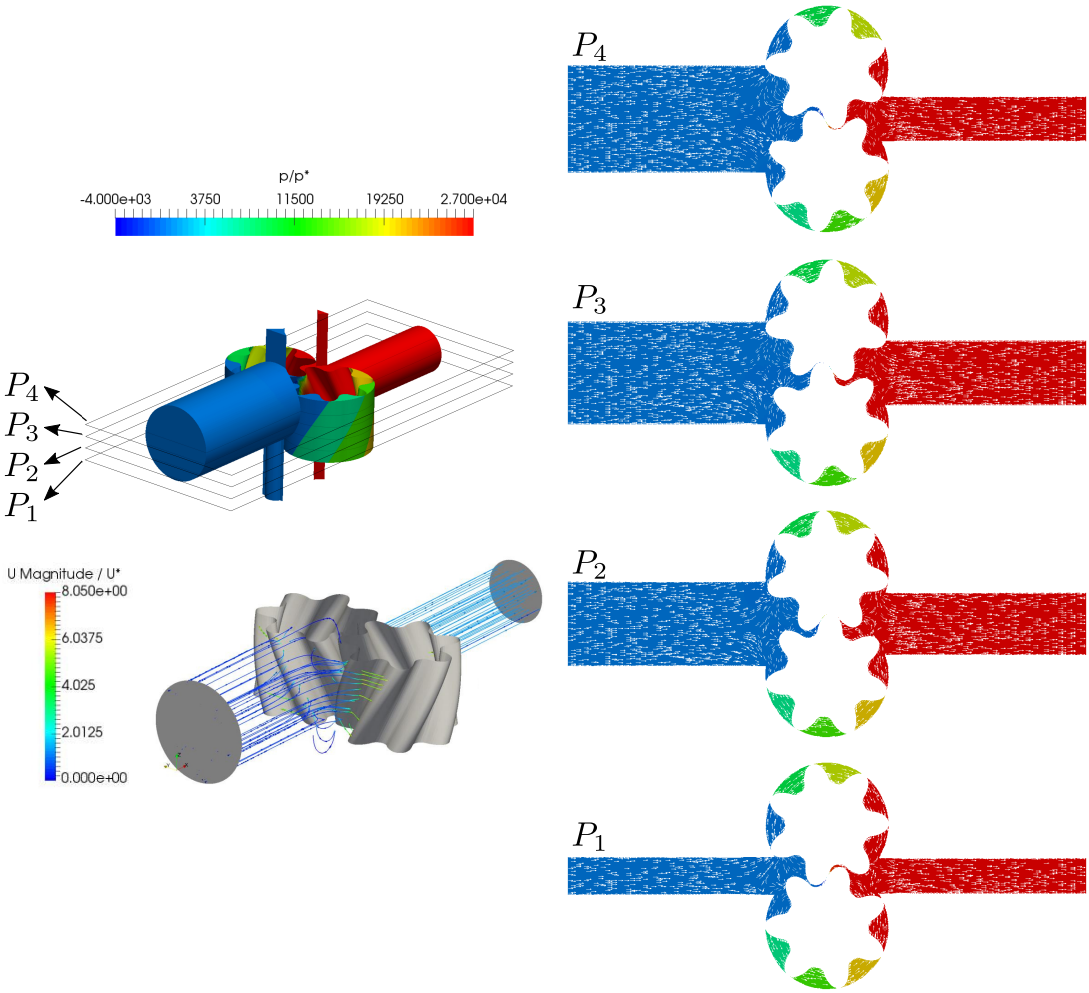


Figure 5.17: Pressure contours with velocity glyph and velocity colored streamlines

towards the open sections in the top and bottom of the pipe before entering the gears region. Further details and analysis could be performed as it will be commented in the future work section (section 6.4).

Chapter 6

Conclusions

This final chapter intends to serve as a summary of the developed strategies, the performed simulations and the results obtained in the thesis. The main contributions are divided in three different sections: the improvements in the pressure-velocity coupling algorithms for co-located grids in the finite-volume framework, the developed mesh motion strategy, topological changes and contact point treatment methods, and the leveraged hybrid turbulence model. These will be followed by a section suggesting possible future lines of research to continue this work.

6.1 Pressure-velocity coupling

Finite-volume methods are probably the most common choice in the field of CFD, in particular when considering commercial software. Furthermore, within these methods, co-located grids are typically employed in order to reduce the memory storage requirements of the alternatives. The most important problem associated with this kind of approach are the nonphysical pressure oscillations, the so called checkerboarding problem, due to the applied discretization methods and the nature of the Navier-Stokes equations. This problem was addressed long ago by applying the Original Momentum Interpolation by Rhie-Chow. This approach is of common use nowadays, but it has some inherent limitations that have not been undertaken in codes like OpenFOAM®, which still suffered from these limitations at the time this thesis work was performed.

In this thesis the OMIM has been analyzed and two of the inherent problems have been identified, namely: the dependence of the steady state solutions on velocity equations under-relaxation factors and the time step size dependency. Available corrections for both problems have been found in literature and have been successfully implemented in OpenFOAM®. The study of the Taylor-Green vortex problem demonstrated the small influence of the latter issue (the time step size dependency), and the increased computational cost and difficult generalization for moving boundary problems, reason for which it has not been considered for the rest of the study. Regarding the under-relaxation factor dependency, the Majumdar correction has been implemented and tested in the simple two-dimensional lid-driven cavity case at two Reynolds numbers and in the NACA 0012 airfoil profile, proving to correctly eliminate the limitation of OMIM.

Other aspect to consider when dealing with incompressible Navier-Stokes equation is the pressure-velocity coupling algorithm. During this thesis the list of classical algorithms accessible in OpenFOAM®: SIMPLE and PISO, has been enlarged with some alternatives available in literature: SIMPLE-C, SIMPLE-R. A systematic study on the accuracy and convergence speed of each algorithm in combination with OMIM with or without Majumdar correction has been performed. SIMPLE-C proved to converge equally or faster than the rest of algorithms considered here while Majumdar correction did not show to affect the number of iterations required to achieve convergence. Therefore, SIMPLE-C with Majumdar correction was selected to be applied to the rest of simulations performed in this thesis.

6.2 Mesh motion strategy

Mesh motion is probably the most complex problem to face in the simulation of gear pumps. The intricacy of the gears motion and the tight clearances make gear pumps a difficult challenge for CFD simulations. In fact, in order to avoid this difficulty many researchers consider sophisticated 1D models, whose lack of physical accuracy and difficult generalization impedes their wide application in the field. The most common alternative is the use of ALE formulations with mesh deformation and re-meshing. The induced mesh-to-mesh interpolation errors and the computational cost of such approach motivated the development of a new mesh motion strategy as defined in chapter 3.

A fully automatic mesh motion strategy has been developed with the aim of reducing the computational cost of the mesh manipulation steps in the application to external gear pumps, while maintaining general applicability and ease of use. The proposed strategy is based on the rotation and deformation of single mesh blocks corresponding to each gear. A synthetic algorithm determines a line/surface between the two gears in the meshing region, which is used as a reference for a mesh projection process. Difficulty arises when considering the interaction between the different mesh blocks. Boundaries between mesh blocks should allow the free flow of the fluid from one block to another. The motion of the different mesh blocks generates the need of the boundary conditions applied to a given face to be changed as the simulation proceeds. This is performed by applying topological changes to the mesh, avoiding any required interaction of the user. Further modifications of the algorithm have been proposed in order to ensure its operation for any given parallelization methods, so as to provide a good balance of cells and faces processor distribution to guarantee a proper scalability.

The utilities presented here, allow not only for the mesh motion, but also for the generation of the initial mesh, for which quality can be controlled by adjusting cells distribution, near wall refinement and non-orthogonality reduction near wall boundaries. The method has been tested in spur and helical two- and three-dimensional gears, with a variety of profiles: involute, cycloidal and user-defined rounded profiles.

Finally in chapter 3 a new method has been proposed to treat the contact point between gears. In order to reduce the computational cost that would be

derived from topological changes in internal faces, alternative methods have been studied to impede the flow from going through gear-to-gear gap in the vicinity of the contact point. Difficulties were found in the application of a strategy proposed in literature when very small realistic clearances were used, and this inspired the development of an algorithm, based on smoothly increased viscosity, to handle the problem. This is can be applied to both two- and three-dimensional meshes with one or more contact points/lines and with any given gear profiles.

6.3 Hybrid turbulence modeling

In the recent years, experimental evidence has proved that the stirring produce by the gears motion generate energy that cannot be dissipated by molecular viscosity and make therefore turbulence arise. While DNS and LES could be prohibitively expensive from the computational point of view, inherent limitations of U-RANS methods when applied to cases with no clear scale separation motivated the development of a hybrid model. In Chapter 4, the hybridization technique named STRUCT has been implemented and tested in OpenFOAM® together with its baseline nonlinear RANS model, BagliettoNLEVM, which has also been implemented and tested in Nek5000 SEM code. Several versions of the model were available in literature and the first task was to analyzed and test them in several study-cases.

The nonlinear eddy viscosity model BagliettoNLEVM, was found to provide accurate description of mean velocity and turbulent kinetic energy fields in a set of academic cases, and to provide better prediction of the anisotropy of the Reynolds stress tensor, when compared to other classical RANS models. In particular the classical fully developed turbulent channel flow was first analyzed, followed by some more complex cases, where the non-linearities in the strain-stress relation of turbulent fluctuations generate the so called secondary motions, namely: the turbulent square channel flow and the flow in a triangular array rod bundle.

Additional cases were then considered for the testing of hybrid approaches. While controlled STRUCT is an open model (case-related information is required for the model to work properly), STRUCTL version was found to fail in some simple cases due to the use of local information to extrapolate behavior far from the local cell. Some alternatives have successfully been tested but introduced additional mesh dependency and still relayed on case-related parameters. Therefore the close version, FTT, was considered and tested. It was found that the some of the terms in the additional equation proposed for STRUCTT played no actual role in the cases and therefore could be removed. Additionally, alternative time scales could be used in the model derivation to provide more a physical behavior. Following the extracted information, a new closed version, named STRUCTT-Tf_r was proposed in this thesis and has proved to provide equal or better behavior than the predecessors in the cases considered.

The choice of test-cases has been made attending to their complexity and/or searching similar physical phenomena than that of the flow through a gear pump. In particular the flow a period hill considers flow separation in a smoothly curved hill (similar somehow to the rounded profiles in a gear pump). The flow past

a square cylinder is used to study the vortex shedding structures coming out of the interaction, resembling again those generated by the motion of gears in the pump. Finally two more complex cases have been considered, the separation against a slight adverse pressure gradient in a plane asymmetric diffuser, typically challenging for hybrid models, and the thermal striping phenomena in a triple jet, where the interaction of turbulent structures determine the mixing of different temperatures streams.

6.4 Final conclusions and future work

In conclusion, algorithms in the fields of pressure-velocity coupling, mesh motion strategies and turbulence modeling have been developed, aiming at providing the tools to facilitate the study of external gear pumps in the co-located grid finite-volume code OpenFOAM®. Without any previous work in this field in OpenFOAM®, this thesis sets the basis to encourage a further test and analysis, with the final goal of providing the most realistic simulation of an external gear pump.

The current work could be extended following the next research lines:

- Use the provided tools to complete the simulation of the three-dimensional pump considering decompression slots and circumferential grooves and test the influence of those in the predicted volumetric efficiency.
- Evaluate numerically the relative importance of radial and axial leakage with and without pressure relief grooves.
- Test the performance of the proposed hybrid model for this application, in comparison with traditional U-RANS approaches.
- Analyze the influence of pressure boundary conditions and test test different approaches to simulate the effect of the discharge circuit and how this influences the predicted performance parameters of the pump.
- Evaluate the importance of the liquid compressibility, in particular for the two-dimensional studies where no pressure groof can be simulated.
- Compare the performance of the proposed mesh motion strategy against traditional re-meshing approaches.
- Use the proposed tools to extend the study to solve for the energy transfer with additional Fluid Solid Interaction (FSI) methods.
- Test the capabilities of the method to predict risk of cavitation, and in a further analysis, perform the multiphase simulation to compare predictions.
- The gears in the pump are subjected to radial forces, due to which the gear axes follow orbits instead of being fixed in space. The mesh motion strategy could also be slightly modified to take into account the this kind of motion and the effect of these considerations on the predictions could be analyzed.

- Once the method has gained sufficient maturity the study could also be extended to determine the noise produced by vibrations caused by pressure oscillations.

Bibliography

[1] A. Abdon and B. Sundén. Numerical investigation of combined impingement and convection heat transfer. *Ann. N. Y. Acad Sci.*, 934:417–423, 2001. doi: 10.1111/j.1749-6632.2001.tb05878.x.

[2] S.D. Anderson and J.K. Eaton. Experimental study of a pressure-driven, three-dimensional, turbulent boundary layer. *AIAA Journal*, 25(8):1086–1092, 1987. doi: 10.2514/3.9747.

[3] J.S. Bagget. On the feasibility of merging LES with RANS for the near-wall region of attached turbulent flows. In *Annual Research Briefs - Center for Turbulence Research*, pages 267–277, 1998.

[4] E. Baglietto and H. Ninokata. Anisotropic eddy viscosity modeling for application to industrial engineering internal flows. *Int. J. Transp. Phenom.*, 8(2):237–248, 2006.

[5] E. Baglietto and H. Ninokata. Improved turbulence modeling for performance evaluation of novel fuel designs. *Nucl. Technol.*, 158(2):237–248, 2007. doi: 10.13182/NT07-A3839.

[6] H. BenSaid, G. Mompean, and H. Naji. On the evaluation of linear and non-linear models using DNS data of turbulent channel flows. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 34(4):469–476, 2012. doi: 10.1590/S1678-58782012000400007.

[7] J.V. Boussinesq. Essai sur la théorie des eaux courantes. Mémoires présentés par divers savants à l'Académie des sciences. *NASA Technical Memorandum*, XXXIII(1):1–680, 1877.

[8] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *Proc. 31st VLDB Conference*, pages 853–864, 2005.

[9] D. Bruce, M. Wilson, and S. Generalis. Flow Field Analysis of Both the Trilobal Element and Mixing Disc Zones within a Closely Intermeshing, Co-Rotating Twin-Screw Extruder. *International Polymer Processing*, 12:323–330, 1997. doi: 10.3139/217.970323.

[10] C.U. Buice and J.K Eaton. Experimental investigation of flow through an asymmetric plane diffuser. In *Annual Research Briefs - Center for Turbulence Research*, pages 117–120, 1995.

[11] P. Casoli, A. Vacca, and G. Franzoni. A numerical model for the simulation of external gear pumps. In *Proc. of the 6th JFPS Int. Symposium on Fluid Power*. Tsukuba, Japan, Nov. 2005. doi: 10.5739/isfp.2005.705.

[12] G. Castellani. Pompe ad ingranaggi a Denti Dritti. In *Progetto delle Dentature, Giornata Mondiale della Fluidodinamica, FLUID'67*. Milan, Italy, 1967.

[13] R. Castilla, P.J. Gamez-Montero, D. del Campo, G. Raush, and M. Garcia-Vilchez. Three-Dimensional Numerical Simulation of an External Gear Pump With Decompression Slot and Meshing Contact Point. *Journal of Fluids Engineering*, 137, 2015. doi: 10.1115/1.4029223.

[14] R. Castilla, P.J. Gamez-Montero, N. Ertük, A. Vernet, M. Coussirat, and E. Codina. Numerical simulation of the turbulent flow in the suction chamber of a gear pump using deforming mesh and mesh replacement. *International Journal of Mechanical Sciences*, 52:1334–1342, 2010. doi: 10.1016/j.ijmecsci.2010.06.009.

[15] R. Castilla, J. Wojciechowsky, P.J. Gamez-Montero, A. Vernet, and E. Codina. Analysis of the turbulence in the suction chamber of an external gear pump using time resolved particle image velocimetry. *Flow Measurement and Instrumentation*, 19(6):337–384, 2008. doi: 10.1016/j.flowmeasinst.2008.06.005.

[16] K.Y. Chien. Predictions of channel and boundary-layer flows with a low-reynolds number turbulence model. *AIAA Journal*, 20(1):33–38, 1982. doi: 10.2514/3.51043.

[17] S.K. Choi. Note on the Use of Momentum interpolation Method for Unsteady Flows. *Numerical Heat Transfer, Part A: Applications*, 36(5):545–550, 1991. doi: 10.1080/104077899274679.

[18] S.K. Choi. Use of the momentum interpolation method for flows with large body force. *Numerical Heat Transfer, Part B: Fundamentals*, 43(3):267–287, 2003. doi: 10.1080/713836204.

[19] S.K. Choi, H.Y. Nam, and M. Cho. Use of momentum interpolation method for numerical solution of incompressible flows in complex geometries: choosing cell face velocities. *Numerical Heat Transfer, Part B: Fundamentals*, 23(1):21–41, 1993. doi: 10.1080/10407799308914888.

[20] T. Chyczewsky, P. Morris, and L. Long. Large-eddy simulation of wall bounded shear flow using the nonlinear disturbance equations. *AIAA Paper 2000-2007*, 2000. doi: 10.2514/6.2000-2007.

[21] D. Cokljat, D. Caridi, G. Link, R. Lechner, and F.R. Menter. Embedded LES Methodology for General-Purpose CFD Solvers. In *Proceedings of the 6th International Symposium on Turbulence and Shear Flow Phenomena*, 2009. Seul, Korea.

[22] A. Cubero and N. Fueyo. A compact momentum interpolation procedure for unsteady flows and relaxation. *Numerical Heat Transfer, Part B: Fundamentals*, 52:507–529, 2007. doi: 10.1080/10407790701563334.

[23] L. Davidson. Hybrid LES/RANS: a combination of a one-equation SGS model and a $k - \omega$ model for predicting recirculating flows. In *Proceedings of ECCOMAS CFD conference*, 2001. doi: 10.1002/fld.512.

[24] L. Davidson. Evaluation of the SST-SAS model: Channel flow, asymmetric diffuser and axi-symmetric hill. In *Proceedings of the European Conference on Computational Fluid Dynamics (ECCOMAS CFD 2006)*, pages 1–20, 2006. Egmond aan Zee, The Neatherlands.

[25] J.W. Deardorff. Stratocumulus-capped mixed layers derived from a three-dimensional model. *Boundary-Layer Meteorol*, 18:495–527, 1980. doi: 10.1007/BF00119502.

[26] D. del Campo, R. Castilla, G. Raush, P.J. Gamez-Montero, and E. Codina. Numerical Analysis of External Gear Pumps Including Cavitation. *J. Fluids Eng.*, 134(8), 2012. doi: 10.1115/1.4007106.

[27] D. del Campo Sud. *Analysis of the Suction Chamber of External Gear Pumps and their Influence on Cavitation and Volumetric Efficiency.* PhD thesis, Escola Tècnica Superior d'Enginyeries Industrial i Aeronàutica de Terrassa, 2012.

[28] J. Donea, A. Huerta, J.P. Ponthot, and A. Rodríguez-Ferrán. *Arbitrary Lagrangian-Eulerian methods.* Encyclopedia of computational mechanics, 2004.

[29] J.P. Van Doormaal and G.D. Raithby. Enhancements of the SIMPLE method for predicting incompressible fluid flows. *Numerical Heat Transfer*, 7:147–163, 1984. doi: 10.1080/01495728408961817.

[30] Y. Dubief and F. Delcayre. On coherent-vortex identification in turbulence. *Journal of Turbulence*, 1:1–22, 2000. doi: 10.1088/1468-5248/1/1/011.

[31] M. Eaton, P.S. Keogh, and K.A. Edge. The 'secondary source' method for the measurement of pump pressure ripple characteristics. Part 1: Description of the method. In *Proceedings of The Institution of Mechanical Engineers Part A: Journal of Power and Energy*, volume 204, 1990. doi: 10.1243/pime_proc_1990_204_006_02.

[32] M. Eaton, P.S. Keogh, and K.A. Edge. The modelling, prediction and experimental evaluation of gear pump meshing presures with particular reference to aero-engine fuel pumps. *Journal of Systems and Control Engineering*, 220:33–40, 2006. doi: 10.1243/09596518JSCE183.

[33] Y. Egorov and F.R. Menter. Development and application of SST-SAS turbulence model in the DESIDER project. In *Advances in Hybrid RANS-LES Modelling. Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, pages 261–270, 2008. doi: 10.1007/978-3-540-77815-8_27.

[34] E. Fares and W. Schroder. Differential Equation for Approximate Wall Distance. *Int. J. Numer. Meth. Fluids*, 39:743–762, 2002. doi: 10.1002/fld.348.

[35] J.H. Ferziger and M. Perič. *Computational Methods for Fluid Dynamics.* Springer, 3^{rd} edition, 2002.

[36] J.K. Fink and L. Leibowitz. Thermodynamic and transport properties of sodium liquid and vapor. *ANL/RE-95-2*, 1995.

[37] J. Frohlich, C.P. Mellen, W. Rodi, L. Temmerman, and M.A. Leschziner. Highly resolved large-eddy simulation of separated flow in a channel with streamwise periodic constrictions. *J. Fluid Mech.*, 526:19–66, 2005. doi: 10.1017/S0022112004002812.

[38] J. Frölich and W. Rodi. *Closure strategies for turbulent and transitional flows.* Cambridge University Press, 2002.

[39] J. Frölich and D. Von Terzi. Hybrid LES/RANS methods for the simulation of turbulent flows. *Prog. Aerospace Sci.*, 44(5):349–377, 2008. doi: 10.1016/j.paerosci.2008.05.001.

[40] P.J. Gamez-Montero, R. Castilla, D. del Campo, N. Ertürk, G. Raush, and E. Codina. Influence of Interteeth Clearances on the Flow Ripple in a Gerotor Pump for Engine Lubrication. In *Proc. Inst. Mech. Eng., Part D*, volume 226, pages 930–942, 2012. doi: 10.1177/0954407011431545.

[41] S.E. Gant. Reliability issues of LES-related approaches in an industrial context. *Flow, Turbul. Combust.*, 84:325–335, 2010. doi: 10.1007/s10494-009-9237-8.

[42] O. Gelineau and M. Sperandio. Thermal fluctuation problems encountered in lmfrs. In *Specialistic Meeting on "Correlation Between Material Properties and Thermohydraulics Conditions in LMFBRs"*, 1994. Aix-en-Provence, France.

[43] N.J. Georgiadis, J. Iwan, D. Alexander, and E. Reshotko. Development of a hybrid RANS/LES method for compressible mixing layer simulations. *AIAA Paper 2001-0289*, 2001. doi: 10.2514/6.2001-289.

[44] M. Germano. From RANS to DNS: Towards a bridging Model. In *Direct and Large-Eddy Simulation III*, pages 225–236. Springer, 1999.

[45] M. Germano. On the hybrid RANS-LES of compressible flows. *Notes Numer. Fluid Mech. Multidiscip. Des.*, 130:253–263, 2015. doi: 10.1007/978-3-642-31818-4.

[46] J. Ghazanfarian and D. Ghanbari. Computational fluid dynamics investigation of turbulent flow inside a rotary double external gear pump. *J. Fluids Eng.*, 132(2):021101, 2014. doi: 10.1115/1.4028186.

[47] U. Ghia, K.N. Ghia, and C.T. Shin. High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of Computational Physics*, 48:387–411, 1982. doi: 10.1016/0021-9991(82)90058-4.

[48] S.S. Girimaji. Partially-Averaged Navier-Stokes Model for Turbulence: A Reynolds-Averaged Navier-Stokes to Direct Numerical Simulation Bridging Method. *J. Appli. Mech.*, 73(3):413–421, 2006. doi: 10.1115/1.2151207.

[49] S.S. Girimaji, R. Srinivasan, and E. Jeong. PANS Turbulence model for seamless transition between RANS and LES:fixed-point analysis and preliminary results. In *Proceedings of the 4th ASME-JSME Joint Fluids Engineering Conference*, pages 1–9, 2003. Honolulu, HI, USA. doi: 10.1115/FEDSM2003-45336.

[50] H. Golapan, S. Heinz, and M.K. Stöllinger. A unified RANS-LES model: computational development, accuracy and cost. *Journal of Computational Physics*, 249:249–274, 2013. doi: 10.1016/j.jcp.2013.03.066.

[51] H. Gopalan and R. Jaiman. Numerical study of the flow interference between tandem cylinders employing non-linear hybrid URANS-LES methods. *Journal of Wind Engineering and Industrial Aerodynamics*, 142:111–129, 2015. doi: 10.1016/j.jweia.2015.03.017.

[52] N. Gregory and C.L. O'Reilly. Low-Speed Aerodynamic Characteristics of NACA 0012 Aerofoil Sections, including the Effects of Upper-Surface Roughness Simulation Hoar Frost. *NASA R&M 3726 (1970).*

[53] M.S. Gritskevich, A.V. Garbaruk, T. Frank, and F.R. Menter. Investigation of the thermal mixing in a T-Junction flow with different SRS approaches. *Nucl. Eng. Des.*, pages 83–90, 2014. doi: 10.1016/j.nucengdes.2014.03.010.

[54] M.S. Gritskevich, A.V. Garbaruk, J. Schutze, and F.R. Menter. Development of DDES and IDDES Formulations for the $k-\omega$ Shear Stress Transport odel. *Flow, Turbulence and Combustion*, 88(3):431–449, 2012. doi: 10.1007/s10494-011-9378-4.

[55] M. Gutes, P.J. Gamez Monter, R. Castilla, and E. Codina. Journal Bearing Performance in Gear Pumps. In *Proc. 1st Int. FPNI-PhD Symposium.* Germany, Sept. 2000.

[56] S. Har-Peled and B. Raichel. The fréchet distance revisited and extended. *ACM Transactions on Algorithms (TALG)*, 10(3), 2014. doi: 10.1145/2532646.

[57] M. Hartinger. *CFD Modeling of Elastohydrodynamic Lubrication*. PhD thesis, Imperial College, 2007.

[58] M. Hartinger, M.L. Dumont, S. Ioannides, D. Gosman, and H. Spikes. CFD Modeling of a Therman and Shear-Thinning Elastohydrodynamic Line Contact. *ASME J. Tribol.*, 130(4), 2008. doi: 10.1115/1.2958077.

[59] A.S. Heisler, J.J. Moskwa, and F.J. Fronczak. Simulated helical gear pump analysis using a new CFD approach. In *Proceedings of the ASME 2009 Fluids Engineering Division Summer Meeting*, 2009. Vail, Colorado, USA. doi: 10.1115/FEDSM2009-78472.

[60] A. Hellsten and S. Wallin. Explicit algebraic reynolds stress and non-linear eddy-viscosity models. *Int. J. Com. Fluid Dyn.*, 23:349–361, 2009. doi: 10.1080/10618560902776828.

[61] G. Houzeaux and R. Codina. A finite element method for the solution of rotary pumps. *Computers & Fluids*, 36(4):667–679, 2007. doi: 10.1016/j.compfluid.2006.02.005.

[62] A. Huser and S. Biringen. Direct numerical simulation of turbulent flow in a square duct. *J. Fluid Mech.*, 27:65–95, 1993. doi: 10.1017/S002211209300299X.

[63] R.I. Issa, A.D. Gosman, and A.P. Watkins. The computation of compressible and incompressible recirculating flows by a non-iterative implicit scheme. *Journal of Computational Physics*, 62:66–82, 1986. doi: 10.1016/0021-9991(86)90100-2.

[64] J. Ivantysyn and M. Ivantysynova. *Hydrostatic Pumps and Motors*. Tech Books Int., New Delhi, India, 2003.

[65] X. Jing-lei, M. Hui-yang, and H. Yu-ning. Nonlinear turbulence models for predicting strong curvature effects. *Applied Mathematics and Mechanics*, 29(1):31–42, 2008. doi: .

[66] W.P. Jones and D. Lentini. A realisable non-linear eddy viscosity/diffusivity model for confined swirling flows. *International Journal of Heat and Fluid Flow*, 29:1612–1627, 2008. doi: 10.1016/j.ijheatfluidflow.2008.08.005.

[67] Y. Kawaguchi, W.Q. Tao, and H.Ozoe. Checkerboard Pressure Predictions Due to the UnderrelaxationFactor and Time Step Size for a Nonstaggered Grid with Momentum Interpolation Method. *Numerical Heat Transfer B*, 41(1):85–94, 2002. doi: 10.1080/104077902753385027.

[68] A.H. Khalaf. *The design and performance of gear pumps with particular reference to marginal suction condition*. PhD thesis, Cranfield Institute of Technology, 1989.

[69] H. Kim, H. Marie, and S. Patil. Two-dimensional CFD analysis of a hydraulic gear pump. *American Society for Engineering Education*, 2007.

[70] N. Kimura, H. Miyakoshi, and H. Kamide. Experimental investigation on transfer characteristics of temperature fluctuation from liquid sodium to wall in parallel triple-jet. *Int. J. Heat Mass Transfer*, 50:2024–2036, 2007. doi: 10.1016/j.ijheatmasstransfer.2006.09.030.

[71] N. Kimura, A. Tokuhiro, and H. Miyakoshi. An experimental investigation on thermal striping. Part II: Heat transfer and temperature measurement results. In *NURETH-8*, volume 3, pages 1724–1734, 1997.

[72] C.L. Ladson, A.S. Hill, and W.G. Johnson. Pressure Distributions from high Reynolds Number Transonic Tests of a NACA 0012 Airfoil in the Langley 0.3 Meter Transonic Cryogenic Tunnel. *NASA TM 100826 (1987)*.

[73] B.E. Launder and B.I. Sharma. Application of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc. *Letters in Heat and Mass Transfer*, 1(2):131–138, 1974. doi: 10.1016/0094-4548(74)90150-7.

[74] B.E. Launder and D.B. Spalding. The numerical computation of turbulent flows. *Comp. Meth. in Appl. Mech. and Eng.*, 3:269–289, 1974. doi: 10.1016/0045-7825(74)90029-2.

[75] G. Lenci. *A methodology based on local resolution of turbulent structures for effective modeling of unsteady flows*. PhD thesis, Massachusetts Institute of Technology, 2016.

[76] G. Lenci and E. Baglietto. A structure-based approach for topological resolution of coherent turbulence: overview and demonstration. In *NURETH-16*, 2015. Chicago IL.

[77] A. Leonard. Energy cascade in large eddy simulations of turbulent fluid flows. *Advances in Geophysics*, 18-A:237–248, 1975. doi: 10.1016/S0065-2687(08)60464-1.

[78] B.P. Leonard. A Stable and Accurate Convective Modelling Procedure Based on Quadratic Upstream Interpolation. *Comput. Meth. Appl. Mech. Eng.*, 19:59–98, 1979. doi: 10.1016/0045-7825(79)90034-3.

[79] F. Lien, W. Chen, and M. Leschziner. Low-reynolds-number eddy-viscosity modelling based on non-linear stress-strain/vorticity relations. In *Proceedings of the 3rd Symposium on Engineering Turbulence Modelling and and Experiments*, pages 91–100, 1996. Crete, Greece.

[80] C. Liping, Z. Yan, Z. Fanili, Z. Jianjun, and T. Xianzhao. Modeling and Simulation of Gear Pumps based on Modelica/Mworks. In *Proceedings of the 8th Modelica Conference*, 2011. Dresden, Germany.

[81] D. Lyn, S. Einav, W. Rodi, and J.H. Park. A laser-doppler velocimetry study of ensemble-averaged characteristics of the turbulent near wake of a square cylinder. *J. Fluid Mech.*, 304:285–319, 1995. doi: 10.1016/j.ijheatfluidflow.2011.02.001.

[82] S. Majumdar. Role of underrelaxation in momentum interpolation for calculation of flow with non-staggered grids. *Numerical Heat Transfer*, 13:125–132, 1988. doi: 10.1080/10407788808913607.

[83] S. Mancò and N. Nervergna. Simulation of an External Gear Pump and Experimental Verification. In *JHPS International Symposiuim on Fluid Power*. Tokio, Japan, 1989.

[84] N.D. Manring and S.B. Kasaragadda. The theoretical flow ripple of an external gear pump. *Journal of Dynamic Systems Measurement and Control-transactions of The ASME*, 125, 2003. doi: 10.1115/1.1592193.

[85] F. Mantlik, J. Heina, and J. Chervenka. Results of local measurements of hydraulic characteristic in triagunlar pin bundle. In *UJV-3778-R*, 1976. Rzez, Czech Republic.

[86] J. Martínez, F. Piscaglia, A. Montorfano, A. Onorati, and S. M. Aithal. Influence of spatial discretization schemes on accuracy of explicit LES: Canonical problems to engine-like geometries. *Computers & Fluids*, 117:62–78, 2015. doi: 10.1016/j.compfluid.2015.05.007.

[87] C. Meneveau, T.S. Lund, and W.H. Cabot. A Lagrangian dynamic subgrid-scale model of turbulence. *Journal of Fluid Mechanics*, 319:233–242, 1996. doi: 10.1017/S0022112096007379.

[88] F.R. Menter. *Best Practice: Scale-Resolving Simulations in ANSYS CFD*. ANSYS Inc 1-70, 2012.

[89] F.R. Menter. Stress-Blended Eddy Simulation (SBES) - A new Paradigm in hybrid RANS-LES Modeling. *Sixth HRLM Symposium*, 2016. Strasbourg University, France.

[90] F.R. Menter and M. Kuntz. *Adaptation of eddy-viscosity turbulence models to unsteady separated flow behind vehicles*, volume 19. Springer, 2004.

[91] F.R. Menter, M. Kuntz, and R. Bender. A scale-adaptive simulation model for turbulence flow predictions. In *41st Aerospace Sciences Meeting and Exhibit.*, pages 1–11, 2003. Reno, NV, USA. doi: 10.2514/6.2003-767.

[92] T.F. Miller and F.W. Schmidt. Use of a Pressure-Weighted Interpolation Method for the Solution of Incompressible Navier-Stokes Equations on a Non-Staggered Grid System. *Numerical Heat Transfer*, 14:213–233, 1988. doi: 10.1080/10407788808913641.

[93] F. Mompean, S. Gavrilakis, L. Machiels, and M.O. Deville. On predicting the turbulence-induced secondary flows using nonlinear $k - \epsilon$ models. *Physics of Fluids*, 8(7):1856–1868, 1996. doi: 10.1063/1.868968.

[94] P.J. Morris, L.N. Long, A. Bangalore, and Q. Wang. A parallel three-dimensional computation aeroacoustics method using nonlinear disturbance equations. *Journal of Computational Physics*, 133:56–74, 1997. doi: 10.1006/jcph.1997.5646.

[95] R.D. Moser, J. Kim, and N.N. Mansour. Direct numerical simulation of turbulent channel flow up to $Re_\tau = 590$. *Physics of Fluids*, 11(4):943–945, 1999. doi: 10.1063/1.869966.

[96] E. Mucchi, G. D'Elia, and G. Dalpiaz. Simulation of the running in process in external gear pumps and experimental verification. *Meccanica*, 47:621–637, 2012. doi: 10.1007/s11012-011-9470-9.

[97] M.E. Munich and P. Perona. Continuous dynamic time warping for translation- invariant curve alignment with applications to signature verification. In *7th Int. Conf Comp. Vision*, pages 108–115, 1999. doi: 10.1109/ICCV.1999.791205.

[98] J. Myllykylä. *Semi-emperical model for the suction capability of an external gear pump*. PhD thesis, Tampere University of Technology, 1999.

[99] Y. Nagano and M. Tagawa. An improved k-epsilon model for boundary layer flows. *Journal of Fluids Engineering*, 112:33–39, 1990.

[100] N.V. Nikitin, F. Nicoud, B. Washito, K. D. Squires, and P.R. Spalart. An approach to wall modeling in large-eddy simulations. *Physics of Fluids*, 12(7):1629–1632, 2000. doi: 10.1063/1.870414.

[101] L.H. Norris and W.C. Reynolds. Turbulent channel flow with a moving wavy boundary. *Report No. FM-10*, 1975. Department of Mechanical Engineering, Stanford University, USA.

[102] F. Paltrinieri, M. Borghi, and M. Milani. Studying the flow field inside lateral clearances of external gear pumps. In *3rd FPNI-PhD Symposium on Fluid Power*. Spain, June 2004.

[103] A. Pascau. Cell face velocity alternatives in a structured collocated grid for the unsteady Navier-Stokes equations. *Int. J. Numer. Meth. Fluids*, 65:812–883, 2011. doi: 10.1002/fld.2215.

[104] S.V. Patankar. A calculation procedure for two dimensional elliptic situations. *Numerical Heat Transfer*, 14:409–425, 1984. doi: 10.1080/01495728108961801.

[105] S.V. Patankar and D.B. Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, 15:1787–1806, 1971. doi: 10.1016/0017-9310(72)90054-3.

[106] C. Peniguel, M. Sakiz, S. Benhamadouche, J. Stephan, and C. Vinderinho. Presentation of a Numerical 3D Approach to Tackle Thermal Striping in a PWR Nuclear T-Junction. In *Proceedings of the 2003 ASME Pressure Vessels and Piping Conference*, pages 125–132, 2003. Cleveland, OH, USA.

[107] S.B. Pope. A more general effective-viscosity hypothesis. *Journal of Fluid Mechanics*, 72(2):331–340, 1975. doi: 10.1017/S0022112075003382.

[108] S.B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

[109] P. Quéméré and P. Sagaut. Zonal multidomain RANS/LES simulations of turbulent flows. *International Journal for Numerical Methods in Fluids*, 40:903–925, 2002. doi: 10.1002/fld.381.

[110] O. Reynolds. On the dynamical theory of incompressible viscous fluid and the determination of the criterion. *Philos. Trans. R. Soc. London*, 1895. http://www.jstor.org/stable/90643.

[111] W.C. Reynolds. Computation of turbulent flows. *Annu. Rev. Fluid Mech.*, 8:183–208, 1976. doi: 10.1146/annurev.fl.08.010176.001151.

[112] C. Rhie and W.L. Chow. A numerical study of the turbulent flow past an isolated airfoil with trailing edge separation. *AIAA J.*, 21:1525–1532, 1983. doi: 10.2514/6.1982-998.

[113] W. Rodi, J.H. Ferziger, M. Breuer, and M. Pourquiée. Status of large eddy simulation: results of a workshop. *J. Fluids Eng.*, 119:248–262, 1997. doi: 10.1115/1.2819128.

[114] J.C. Rotta. Uber eine Methode zur Berechnung turbulenter Scherströmungen. *Aerodyn. Versuchanstalt Göttingen, Rep. 69A14*, 1968.

[115] G. Rousseaux, S. Seifer, V. Steinberg, and A. Wiebel. On the lamb vector and the hydrodynamic charge. *Experiments in Fluids*, 42(2):291–299, 2007. doi: 10.1007/s00348-006-0238-2.

[116] P. Sagaut. *Large eddy simulation for incompressible flows: An introduction*. Scientific Computation, Springer, 3^{rd} edition, 2006.

[117] R. Schiestel and A. Dejoan. Towards a new partially integrated transport model for coarse grid and unsteady turbulent flow simulations. *Theor. Comput. Fluid Dyn.*, 18:443–468, 2005. doi: 10.1007/s00162-004-0155-z.

[118] W.Z. Shen, J.A. Michelsen, and J.N. Sorensen. Improved Rhie-Chow interpolation for unsteady flow computations. *AIAA Journal*, 39(12):2406–2409, 2001. doi: 10.2514/2.1252.

[119] T. Shih and N. Liu. A Realizable Reynolds Stress Algebraic Equation Model. *NASA Technical Memorandum*, (205993), 1993.

[120] T. Shih, J. Zhu, and J.L. Lumley. A realizable reynolds stress algebraic equation model. *NASA Technical Memorandum 105993*, 1993.

[121] T.H. Shih and N.S. Liu. Partially resolved numerical simulation. From RANS towards LES for engine turbulent flows. In *42nd AIAA Aerospace Sciences Meeting & Exhibit.*, pages 1–14, 2004. Reno, NV, USA.

[122] M.L. Shur, P.R. Spalart, M.K. Strelets, and A.K. Travin. A Hybrid RANS-LES Approach With Delayed-DES and Wall-Modelled LES Capabilities. *International Journal of Heat and Fluid Flow*, 29(6):1638–1649, 2008. doi: 10.1016/j.ijheatfluidflow.2008.07.001.

[123] B.L. Smith, J.H. Mahaffy, K. Angele, and J. Westing. Report of the OECD/NEA-Vattenfall T-Junction Benchmark exercise No. NEA/CSNI/R(2011)5. 2011.

[124] P. Spalart, S. Deck, M. Shur, K. Squires, M.K. Strelets, and A. Travin. A New Version of Detached-Eddy Simulation, Resistant to Ambiguous Grid Densities. *Theoretical and Computational Fluid Dynamics 0935-4964*, pages 181–195, 2006. doi: 10.1007/s00162-006-0015-0.

[125] P. Spalart, W.H. Jou, M.K. Strelets, and S. Allmaras. Comments on the feasibility of LES for wings, and on a hybrid RANS/LES approach. *Proceedings of the First AFOSR International Conference on DNS/LES, Advances in DNS/LES*, pages 137–147, 1997. Ruston, LA, USA.

[126] P.R. Spalart. Detached-Eddy Simulation. *Annu. Rev. Fluid Mech.*, 41:181–202, 2009. doi: 10.1149/annurev.fluid.010908.165130.

[127] P.R. Spalart. Philosophies and fallacies in turbulence modeling. *Prog. Aerosp. Sci.*, 72:1–15, 2015. doi: 10.1016/j.paerosci.2014.12.004.

[128] C.G. Speziale. On nonlinear $k-l$ and $k-\epsilon$ models of turbulence. *J. Fluid Mech.*, 178:459–475, 1987. doi: 10.1017/S0022112087001319.

[129] C.G. Speziale. A review of the Reynolds stress models for turbulent shear flows. *Report 95-15 ICASE*, 1995.

[130] C.G. Speziale. Computing non-equilibrium turbulent flows with time-dependent RANS and VLES. *Proceedings of the 15th International Conference on Numerical Methods in Fluid Dynamics*, 1996. Monterey, CA, USA.

[131] C.G. Speziale. A combined large-eddy simulation and time-dependent RANS capability for high-speed compressible flows. *Journal of Scientific Computing*, 13(3):253–274, 1998. doi: 10.1023/A:1023266932231.

[132] C.G. Speziale. Turbulence modeling for time-dependent RANS and VLES: a review. *AIAA Journal*, 36(2):173–184, 1998. doi: 10.2514/2.7499.

[133] W. Strasser. CFD Investigation of Gear Pump Mixing Using Deforming/Agglomerating Mesh. *Journal of Fluids Engineering*, 129:476–484, 2006. doi: 10.1115/1.2436577.

[134] A. Tokuhiro and N. Kimura. An experimental investigation on thermal striping: mixing phenomena of a vertical non-buoyant jet with two adjacent buoyant jets as measured by ultrasound doppler velocimetry. *Nuclear Engineering and Design*, 188(1):49–73, 1999. doi: 10.1016/S0029-5493(99)00006-0.

[135] A. Tokuhiro and N. Kimura. An experimental investigation on thermal striping. mixing phenomena on a vertical non-buoyant jet with two adjacent buoyant jets as measured by ultrasound doppler velocimetry. *Nucl. Eng. Des.*, 188:49–73, 1999.

[136] A. Travin, M. Shur, P. Spalart, and M. Strelets. On URANS solutions with LES-like behaviour. In *Proceedings of ECCOMAS*, pages 1–20, 2004.

[137] P. Tucker and L. Davidson. Zonal $k-l$ based large eddy simulations. *Computers & Fluids*, 33:267–287, 2004. doi: 10.1016/S0045-7930(03)00039-2.

[138] P.G. Tucker, C.L. Rumsey, R.E. Bartels, and R.T. Biedron. Transport equation based wall distance computations aimed at flows with time-dependent geometry. *NASA 2003-212680*, 2003.

[139] P.R. Voke. *Flow Past a Square Cylinder: Test Case LES2*, volume 5. Springer, 1997. Dordrecht.

[140] J. Vande Voorde, J. Vierendeels, and E. Dick. Development of a Laplacian-based mesh generator for ALE calculations in rotary volumetric pumps and compressors. *Comput. Methods Appl. Mech. Engrg*, 193(39-41):4401–4415, 2004. doi: 10.1016/j.cma.2003.12.063.

[141] H. Weller. Controlling the computational modes of the arbitrarily structured C grid. *Monthly Weather Review*, 140(10):3220–3234, 2012. doi: 10.1175/MWR-D-11-00221.1.

[142] W.E. Wilson. Performance Criteria for Positive Displacement Pumps and fluid Motors. In *Proc. of ASME semi-annual meeting*. June, 1948.

[143] B. Yu, W.Q. Tao, J.J. Wei, Y. Kawaguchi, T. Tagawa, and H. Ozoe. Discussion on momentum interpolation method for collocated grids of incompressible flow. *Numerical Heat Transfer, Part B: Fundamentals*, 42(2):85–94, 2002. doi: 10.1080/10407790190053879.

[144] Y.Q. Yu, E. Merzari, J.W. Thomas, A. Obabko, and S.M. Aithal. Steady and unsteady calculations on thermal stripping phenomena in triple-parallel jet. *Nuclear Engineering and Design*, 312:429–437, 2017. doi: 10.1016/j.nucengdes.2016.06.015.