Towards a robust parallel solver for large-scale industrial flow simulations

Master Thesis by:

Mátyás Rosta

Advisor:

Dr. Matteo Giacomini

Master's degree in Numerical Methods in Engineering

June, 2022

Abstract

In this work a CFD analysis is done on incompressible viscous flows using Finite Volume schemes implemented in the open-source software OpenFOAM. The objective of this study is twofold: to gain experience with the software and to define a set of best practices when running large-scale cases in OpenFOAM using parallel architectures. The first objective is obtained by testing three academic benchmarks, namely the lid-driven cavity, the flow over a backward facing step, and flow past a circular cylinder. The validation of these results is made by contrasting them to the data available in the literature. The second objective was fulfilled by studying two large-scale industrial problems, laminar flow inside an S-bend and turbulent external flow around a car. For the latter, the DriveAer geometry has been used. The analysis of these high-performance computing studies has been defined in terms of the relative efficiency and speed up for the two problems. The studied cases have been scaled up until 84 CPUs for the S-bend, and until 224 CPUs for the vehicle geometry. Furthermore, the performance of three partitioners, namely the simple, hierarchical, and scotch decomposers have been evaluated.

II

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Computational Fluid Dynamics (CFD) has experienced an exponential growth starting from the 1980s, as numerical models became more sophisticated and computational capacity increased with cluster-architectures [RL21]. This led to the incorporation of CFD techniques in different industrial areas, such as aerospace and the automotive industry. A great benefit of CFD is the ability to study the flow around vehicles or other industrial applications, reducing the cost of a prototype, as it allows to reduce the required number of wind tunnel testing, and to save money by reducing the number of fabricated prototypes.

Incompressible laminar or turbulent flows are ubiquitous in industrial applications.A fluid flow can be treated as incompressible for a Mach number smaller than 0.3, as at this range of velocities the effects of compressibility can be neglected. A Mach=0.3 corresponds to a speed approximately of 360 km/h for air at sea level, so the field of applicability of the incompressible Navier-Stokes equations is plentiful. Just to mention a few examples, industrial flows inside tubes and pipes, pump and turbine applications, utilization in maritime problems such as flow around vessels and submarines[KK11]. In ground transportation it is also applicable, to evaluate the fluid flow over cars, trucks or motorcycles, as well as high-speed train designs [MHB15]. The topic of the application of incompressible Navier-Stokes equations in the automotive industry is explained in more detail below, due to the special interest towards this field.

1.1.1 Passenger and competitive car simulations

The automotive industry found in CFD a very reliable tool that can handle results with acceptable accuracy, affordable cost, and fast turnaround time [ZBFU19]. The study of the flow around a vehicle is very important, as it can define its aerodynamic loads, which directly influence important characteristics of the car such as maneuverability and stability at high speed, traction and fuel consumption [BRG13].

Generally, in automotion, a mesh size on the car surface is approximately 1.5-6 mm [SB13] [ZBFU19], with a first layer thickness around 0.005 mm. This leads to very large-scale models, with up to 190-200 million cells per mesh [ZBFU19].

According to the measures of the International Automobile Federation (FIA), the maximum race speed reached during the 2022 Miami Grand Prix, was 345.6 km/h achieved by Kevin Magnussen [Cha22]. This is just below the approximated limit of incompressible flow applicability. Moreover, access to wind tunnel testing facilities is limited by FIA regulations, hence the computational modelling of the car in a high-fidelity and highly efficient way is crucial. This goal is obtained by constructing a large fluid domain around the car and discretising it with very fine mesh elements around the geometrical surface of the car [RS18][GC20]. The resulting meshes for F1 car simulations also contain several hundreds of millions of elements [BP19].

The above problems require high-performance computing (HPC) techniques to overcome computational limitations such as long computational time, and available memory capacity.

1.2 Numerical methods for Computational Fluid Dynamics

CFD solves a set of governing partial differential equations modelling the conservation of mass, momentum, and energy [DH03].

The approximation of flow equations require robust numerical schemes capable of dealing with convection phenomena, incompressibility, and turbulence effects. These methods can be divided into ones that follows an Eulerian approach to solve computational fluid problems and into those who follow a Lagrangian approach. Among the Lagrangian models the Smooth Particle Hydrodynamics (SPH) method starts to get popularity, however, these techniques are not in a consolidated phase [MCQ16]. The vast majority of the CFD solvers are using Eulerian type numerical scheme, which is based on the idea of control volumes. Among these methods can be found the Finite Difference Method (FDM), Finite Volume Method (FVM), and Finite Element Method (FEM). On these three methods lies the vast majority of the commercial softwares nowadays.

The application of FDM in CFD is not very popular, nonetheless, it is a stable and accurate scheme that provides rapid convergence [Sha16]. This method discretises the domain by defining grid points. The governing partial differential equations are then rewritten into algebraic equations. It is generally applied for free-surface or atmospheric, meteorological and astrophysical problems [Sjo16] [Bha12]. Some softwares that use FDM are ModFlow, HEC-RAS, Flow3D or NASA's Overflow. The advantage of this model with respect to the other ones is its easy implementation. However, these methods present several limitations when geometrically complex domains are considered.

The Finite Volume Method is currently the most widely applied method in Computational Fluid Dynamics. This method gained its popularity due to its locally conservative formulation [Hai17]. The discretisation in this case is done by creating control volumes, and the quantities of interest are evaluated at a certain point of the created volume. It can be in the cell center (Cell-Centered Finite Volume), at the vertices (Vertex-Centered Finite Volume), or at the faces (Face-Centered Finite Volume)[SGH18] and

following, the fluxes of the variables are calculated on the boundary of each cell, in this way imposing their conservation at each cell [Fon18]. This results in very robust numerical schemes. The FVM is implemented in softwares such as Ansys Fluent (CCFV), Ansys CFX (VCFV), Altair Hyperworks, SimScale or open-source codes such as OpenFOAM or SU2.

The Finite Element Method creates a piecewise polynomial approximations of the flow variables. The most common is the application of 2nd order polynomials for velocity and first order polynomials for pressure to fulfill the so-called LBB-condition[DH03]. Among software that use FEM for CFD is COMSOL Multiphysics, LS-Dyna.

High-order methods, such as spectral elements, discontinuous Galerkin, Hybridisable discontinuous Galerkin, etc, have also been successfully employed for CFD simulations, see in [WFA+13], [CKS00], [NPC09]. Nonetheless, their application to large-scale industrial flow problems is still limited.

1.3 Objective of this work

In this work the CCFV method implemented in the open-source software OpenFOAM will be employed to solve industrial flow problems, including internal and external flows around realistic geometries such as an S-bend tube in heating, ventilation and air conditioning (HVAC) systems or the DriveAer car model.

Despite its limited accuracy, FV is the preferred method by the industry, due to its robustness. CCFV relies on a piecewise constant approximations of the flow variables, thus requiring extremely fine meshes to achieve acceptable accuracy.

Hence, High-Performance Computing (HPC) is needed to speed up the computation, reducing time cost of each simulation, and help to overcome hardware and memory limitations.

This work aims to identify a set of rules and best practices for parallel simulations using OpenFOAM. When working with HPC it is important to know the methods one can use for the decomposition of the domain, and the impact they have on the speed of the calculation. Furthermore, it is also interesting to know which is the optimal cell count to be assigned to each processor to obtain the best performance using the available computing facilities.

Specifically, this work aims to investigate the limitations and capabilities of OpenFOAM to perform large-scale flow simulations using parallel computing facilities. To achieve this goal the following steps have been completed:

- literature review on modern aerodynamic internal and external flow problems;
- familiarisation with OpenFOAM for steady and unsteady incompressible laminar flow computations;
- familiarisation with OpenFOAM for steady incompressible turbulent flow computations;
- validation using academic benchmarks;

- accuracy and scalability studies for large-scale laminar internal flow problems using parallel computing facilities;
- accuracy and scalability studies for large-scale turbulent external flow problems using parallel computing facilities;
- validation using industrial benchmarks.

Chapter 2

Approximation of the Finite Volume method in OpenFOAM

In this chapter, the governing equations for incompressible flows are briefly reviewed and a basic introduction to OpenFOAM is provided.

2.1 Governing equations for the fluid flow

The governing equations for an incompressible viscid flow are the conservation laws for mass, momentum and energy. Under the assumption of incompressible flows the energy equation is decoupled from the other two conservation equations. This makes possible to evaluate the velocity and the pressure field without solving for the internal energy. The resulting incompressible steady-state Navier-Stokes equations are:

$$\begin{cases} \boldsymbol{\nabla}\cdot(\boldsymbol{u}\otimes\boldsymbol{u}) - \boldsymbol{\nabla}\cdot(\nu\boldsymbol{\nabla}\boldsymbol{u}) + \boldsymbol{\nabla}p = \mathbf{0} & \text{in } \Omega, \\ \boldsymbol{\nabla}\cdot\boldsymbol{u} = 0 & \text{in } \Omega, \\ \boldsymbol{u} = \boldsymbol{u}_{in} & \text{on } \Gamma_{in}, \\ \boldsymbol{u} = \mathbf{0} & \text{on } \Gamma_{w}, \\ (\nu\boldsymbol{\nabla}\boldsymbol{u} - p\mathbf{I}_d)\boldsymbol{n} = \mathbf{0} & \text{on } \Gamma_{out}, \end{cases} \tag{2.1}$$

The open bounded computational domain $\Omega \in \mathbf{R}^{n_sd}$ is defined in n_{sd} spatial dimensions, with the corresponding Dirichlet (Γ_D) and Neumann (Γ_N) boundaries $\partial\Omega = \Gamma_N \cup \Gamma_D$ and $\Gamma_N \cap \Gamma_D = \emptyset$.

In these equations the first one stands for the conservation of momentum, and the second one for the mass conservation, which in this case is zero divergence constraint on the velocity. They are followed by the Dirichlet and Neumann boundary conditions. The first one is applicable for the imposition of inlet boundary condition or to place physical walls. In the latter case $\mathbf{u}_D$=$\mathbf{0}$. Neumann boundary conditions can be employed, e.g. on outlet boundaries, imposing the pseudo-traction to be zero.

2.2 Treatment of turbulence

It is known that for convection dominated problems a transition from laminar to turbulent flows occurs. Turbulent flows show a complexity due to the chaotic oscillations of the flow motion, as well as multi-scale phenomena in space and time. To handle these problems, several strategies are suitable, e. g. the Reynolds Averaged Navier-Stokes equations, Large Eddy Simulation (LES), and Direct Numerical Simulation (DNS), to mention a few.

Among these techniques, DNS provides a very detailed solution of the turbulent phenomena [Wil06]. Nonetheless, it is extremely time and resource consuming as it solves the Navier-Stokes equations, without the use of any turbulence model, making it computationally unaffordable for industrial applications.

LES solves the largest eddies in the flow, with higher kinetic energy, and models the remaining ones at the subgrid scale [Pop04] [Wil06].

The RANS approach first separates the mean flow and a fluctuation component [Wil06]. For velocity, this reads as **u**=**U**+**u'**, where **U** is the mean component and **u'** is the fluctuating component. It follows that the momentum equation of the Reynolds Average Navier-Stokes model is:

$$\boldsymbol{\nabla}\cdot(\boldsymbol{U}\otimes\boldsymbol{U}) - \boldsymbol{\nabla}\cdot(\nu\boldsymbol{\nabla}\boldsymbol{U} - \overline{\boldsymbol{u'}\otimes\boldsymbol{u'}}) + \boldsymbol{\nabla}P = \mathbf{0}, \tag{2.2}$$

where the third term is known as the Reynolds stress tensor, and it accounts for the turbulent effects. These are commonly modelled using an additional turbulent viscosity, ν_t, contribution leading to:

$$\begin{cases} \boldsymbol{\nabla}\cdot(\boldsymbol{U}\otimes\boldsymbol{U}) - \boldsymbol{\nabla}\cdot((\nu+\nu_t)\boldsymbol{\nabla}\boldsymbol{U}) + \boldsymbol{\nabla}P = \mathbf{0} & \text{in } \Omega, \\ \boldsymbol{\nabla}\cdot\boldsymbol{U} = 0 & \text{in } \Omega, \\ \boldsymbol{U} = \boldsymbol{U}_{\text{in}} & \text{on } \Gamma_{\text{in}}, \\ \boldsymbol{U} = \mathbf{0} & \text{on } \Gamma_{\text{w}}, \\ (\nu\boldsymbol{\nabla}\boldsymbol{U} - p\mathbf{I}_d)\boldsymbol{n} = \mathbf{0} & \text{on } \Gamma_{\text{out}}, \end{cases} \tag{2.3}$$

To determine the turbulent viscosity, several closure models, using one or more equations, have been successfully adopted by the industry. Below, a short overview of some common options is presented.

2.2.1 Spalart-Allmaras model

This model is a one equation turbulence model, that solves a nonlinear equation for the eddy viscosity ($\tilde{\nu}$), from which ν_t is thus computed [SA92]. This model works well in highly viscous regions, as inside boundary layers near physical walls [Wil06][Rum22]. For this reason, it has been used for calculating the flow around airfoils or turbine blades. However, it presents some problems working in the free-stream regions, where it can give nonphysical solutions. It is a very robust turbulence model, and it is commonly employed to get an initial guess before the application of more complex models.

2.2.2 k−ϵ model

This model has been proposed by Launder [LS74][JL72], in order to improve the mixing-length model, and to overcome the need for algebraically prescribing the turbulent length scales for turbulent flows. This is a two equation model, the first equation solves for the turbulent kinetic energy (k), and the second for the turbulent energy dissipation rate (ϵ)[Wil06]. It has been one of the most applied turbulence models, however, it shows some issues when working with large adverse pressure gradients and boundary layer separation.

2.2.3 k−ω SST model

This turbulence model was introduced by Menter [Men93] deriving from the original k-ω model of Wilcox[Wil88]. It is a two equation model, that combines the k-ω and the Shear-Stress Transport models. This model is very popular in the CFD community as it works as a k-ω model down to the viscous sub-layer but the SST formulation gives a behaviour similar to the k-ϵ model in the freestream far from the wall[Rum21]. Nonetheless, it is known to overpredict the production of high turbulence levels around stagnation points and zones with high velocity gradients.

The formulation of this model is done by calculating the turbulence kinetic energy (k), the specific dissipation rate (ω) and from these two data evaluate the turbulent kinematic viscosity (ν_t) in the RANS equations. Mathematically this is calculated as [MKL03]:

- Turbulent kinematic viscosity

$$\nu_t = \frac{a_1 k}{max(a_1\omega, SF_2)} \tag{2.4}$$

  where **S** corresponds to the strain rate.

- Turbulent kinetic energy

$$\frac{\partial k}{\partial t} + U_j\frac{\partial k}{\partial x_j} = P_k - \beta^* k\omega + \frac{\partial}{\partial x_j}\Big[(\nu + \sigma_k\nu_t)\frac{\partial k}{\partial x_j}\Big] \tag{2.5}$$

- Specific dissipation rate

$$\frac{\partial \omega}{\partial t} + U_j\frac{\partial \omega}{\partial x_j} = \alpha S^2 - \beta\omega^2 + \frac{\partial}{\partial x_j}\Big[(\nu + \sigma_{\omega 1}\nu_t)\frac{\partial \omega}{\partial x_j}\Big] + 2(1-F_1)\sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i} \tag{2.6}$$

- Closure coefficients
  The blending functions F_1 and F_2 are defined as [MKL03]:

$$F_1 = tanh\left[\Big[min\Big(max(\frac{\sqrt{k}}{\beta^*\omega y}, \frac{500\nu}{y^2\omega}), \frac{4\sigma_{\omega 2}k}{CD_{k\omega}y^2}\Big)\Big]^4\right]$$

$$F_2 = tanh\Big[\big[max(\frac{2\sqrt{k}}{\beta^*\omega y}, \frac{500\nu}{y^2\omega})\big]^2\Big]$$

Note, that F_1 is zero away from the surface, which gives a k-ϵ behaviour, while it turns to one inside the boundary layer, corresponding to a k-ω model.

The production limiter P_k stands for reducing the turbulence in stagnation regions.

$$P_k = min\Big(\tau_{ij}\frac{\partial U_i}{\partial x_j}, 10\beta^* k\omega\Big)$$

$$CD_{k\omega} = max(2\rho\sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i}, 10^{-10}\Big)$$

The following constants are specified by [MKL03]:

$$\sigma_{k1} = 0.85, \quad \sigma_{k2} = 1$$

$$\sigma_{\omega 1} = 0.5, \sigma_{\omega 2} = 0.856$$

$$\beta_1 = \frac{3}{40}, \beta_2 = 0.0828, \beta^* = 0.09$$

$$\alpha_1 = \frac{5}{9}, \alpha_2 = 0.44$$

$$\Phi = \Phi_1 F_1 + (1 - F_1)\Phi_2$$

The boundary conditions applied to the (2.5)(2.6) are fixed values at inlet and zero gradient at outlet boundaries. Near the walls, all turbulent quantities are set to zero except for ω, for which [Men93] recommends the following formulation:

$$\omega = 10\frac{6\nu}{\beta_1(\Delta y)^2} \qquad \text{at} \qquad y = 0$$

where Δy is the distance between the current position and the following node. Interested readers are referred to [Wil06] for more information.

2.3 Introduction to OpenFOAM

OpenFOAM allows to solve a wide range of fluid mechanics problems, including incompressible, compressible, steady state or transient problems. It is also possible to simulate multi-phase flows, combustion, porous media, heat transfer and buoyancy driven phenomena, etc. For each kind of problems there are one or more solvers available. The ones targeting incompressible single-phase flows are summed up in Table 2.1.

Solver	Transient	Turbulence	Heat transfer	Buoyancy	Combustion	Multi region	Dynamic mesh
icoFoam	x						
simpleFoam		x					
pimpleFoam	x	x					x
pisoFoam	x	x					
chemFoam	x		x		x		
laplacianFoam	x						
potentialFoam							

Table 2.1: Solvers of OpenFOAM working only with incompressible flows, data obtained from [Com21]

In this work, simpleFoam and pisoFoam solvers were used, for steady state and transient simulations, respectively.

Description of the simpleFoam algorithm

simpleFoam implements the so called SIMPLE algorithm [PS72] that stands for semi-implicit method for pressure linked equations. This method is a fractional-step Chorin-Temam projection method, that decouples numerically the pressure and velocity fields, and solves the problem iteratively. For more details about this method one can consult the book written by Jean Donea and Antonio Huerta [DH03].

The method first evaluates the momentum equation without considering the influence of the pressure, giving an intermediate result for velocity, namely called $\boldsymbol{u}$. The pressure field then is evaluated from the incompressibility constraint, which is reformulated into a Poisson problem. Finally, a correction is made on $\boldsymbol{u}$, obtaining a final solution for the velocity field $\mathbf{u}$. The algorithm is detailed in Equation 2.7.

$$\begin{cases} \dfrac{\boldsymbol{u}^k - \boldsymbol{u}^{k-1}}{\Delta t} + \boldsymbol{\nabla}\cdot(\boldsymbol{u}^k \otimes \boldsymbol{u}^{k-1}) - \boldsymbol{\nabla}\cdot(\nu \boldsymbol{\nabla}\boldsymbol{u}^k) = \boldsymbol{s} & \text{in } \Omega, \\ \boldsymbol{u}^k = \boldsymbol{u}_D & \text{on } \Gamma_D, \\ \boldsymbol{n}\cdot(\nu\boldsymbol{\nabla}\boldsymbol{u}^k) = \boldsymbol{t} & \text{on } \Gamma_N, \end{cases} \tag{2.7a}$$

$$\begin{cases} \boldsymbol{\nabla}\cdot(\boldsymbol{\nabla}p) = \dfrac{1}{\Delta t}\boldsymbol{\nabla}\cdot\boldsymbol{u}^k & \text{in } \Omega, \\ \boldsymbol{n}\cdot\boldsymbol{\nabla}p = 0 & \text{on } \Gamma_D, \\ \boldsymbol{n}p = \boldsymbol{0} & \text{on } \Gamma_N, \end{cases} \tag{2.7b}$$

$$\boldsymbol{u} = \boldsymbol{u}^k - \Delta t \boldsymbol{\nabla} p. \tag{2.7c}$$

2.3.1 Organization of an OpenFOAM model

To run a case in OpenFoam, the case setup files must be organised in a specific way. There are three main folders which contain the fundamental information about the model, which are *0*, *constant*, and *system*. The first folder contains the initial information for the model and it must have at least one file per variable that should be initialized. For example, for a laminar incompressible case, the definition of the pressure (p) and the velocity (U) is sufficient, while for a turbulent case using k-ω SST model the turbulent kinetic viscosity (ν_t), the kinetic energy (k) and the specific dissipation rate (ω) must also be initialized in that folder.

In *constant* there must be a folder called *polyMesh* that contains information about the mesh and boundary conditions, and at least two files, one defining the turbulence model that is applied, and the other transport properties, such as the kinematic viscosity of the flow.

The third folder contains files that are required to define the solver and solution control of the problem. In this folder there must be the three following files: *fvSolution*, where one must define a method to solve each variable (for example, GAMG - geometric-algebraic multigrid for pressure), relative and absolute tolerances at each iteration, number of sub-iterations within in each iteration, residual control for convergence criteria, under-relaxation factors for specific variables, the number of non-orthogonal correctors etc. When working with only Dirichlet type conditions for the velocity on every boundaries it is necessary to define the value of the pressure at a specific point of the mesh. In OpenFOAM this must be written in the *fvSolution* dictionary. The file *fvSchemes* contains information of the numerical scheme that is used to approximate the variables, such as the numerical schemes for the time derivative, for the divergence terms, or for the gradients of each variable. In *controlDict* one can define the solver controls, such as setting the solver itself (p.e. simpleFoam), the time step, maximum number of iterations, the initial time step, writing criteria for saving the results after determined steps. In the *controlDict* file one can set the definition of functions used for post-processing as well. In this folder one should include other dictionaries, such as blockMeshDict - when the built-in mesh generator is desired to be used, decomposeParDict - to specify the domain decomposition method, or other dictionaries for post-processing.

2.3.2 Parallelisation in OpenFOAM

OpenFOAM gives the possibility to its users to run cases in parallel, and it counts with several built-in partitioners to decompose the domain. The decomposition of the domain is made based on the specifications given in the dictionary called *decomposeParDict*, located in the folder *system*. This file contains the specifications about the number of subdomains, decomposition method with different partitioner specific parameters. Some of the most frequently used partitioners will be briefly presented below:

- **simple**
  It decomposes the domain into subdomains along the defined directions. The decomposition is done with uniform weight, which means, the resulting mesh count per CPU is approximately the same for all partitions.

coefficients	description
n	number of subdivisions in each direction
order	directional order in which the decomposition is made
delta	value for the rotation matrix
transform	transformation of the cartesian coordinates

Table 2.2: Coefficients of the simple partitioner

- **hierarchical**
  It is based on the *simple* method, however, it sets a hierarchical order between the different directions before doing the partitioning. So, it splits the domain first in one direction, and then in the second and third directions. This method provides very uniform size for the created subdomains.

coefficients	description
n	number of subdivisions in each direction
order	directional order in which the decomposition is made
delta	value for the rotation matrix
transform	transformation of the cartesian coordinates

Table 2.3: Coefficients of the hierarchical partitioner

- **metis**
  Metis is a very fast and efficient partitioner. It is based on graph partitioning. Due to the Fill-reducing Matrix Ordering techniques applied in this decomposition method, it can reduce the requirements for the computation and storage of sparse matrix factorization up to an order of magnitude [Lab15]. This means that metis can create high quality partitions in a very time efficient way.

  The coefficients that can be applied are: *method* - recursive or k-way, depending on the algorithm which is of the preference of the user, *processorWeights* - useful if the machine on which the calculations will be made consist of CPUs with different speed and cache.

coefficients	property
method	recursive or k-way (users preference)
processorWeights	different mesh count at each CPU

Table 2.4: Coefficients of the metis partitioner

- **scotch**
  This method is considered to be a metis-like method, as it is also based on graphs. It creates partitions by minimising the number of boundaries between the processors. It creates good quality partitions and it is implemented in a very user-friendly manner into OpenFOAM. It only requires the number of subdomains given by the user. As Metis is not part of the basic third party license of OpenFOAM, generally, this method is recommended to use instead.

- **kahip:**
  This method is called the Karlruhe High Quality Partitioning (KaHiP) algorithm. It is also a metis-like algorithm based on graphs. This method minimise the number of edges between the subdomains while creating equally sized partitions.

- **manual**:
  This method decompose the domain following the manually written script of the user, specifying the cells corresponding to a defined processor.

In addition, **multi-region** method allows to assign different partitioners for specific regions of the mesh, and **multi-level** allows to decompose a mesh level-by-level with different partitioners. One example could be the decomposition of a domain using metis among the different CPUs, whereas the resulting meshes could be further divided using hierarchical.

Chapter 3

Numerical validation

3.1 2D cavity flow

The first benchmark that has been tested was a 2D problem, a lid-driven cavity flow. The problem consists of a cavity with three fixed walls and the top surface of the hole is moving with a constant velocity in the horizontal direction. This case is compared with the results presented in *Tezduyar and Mittal* [TMRS92] and *Donea and Huerta* [DH03]. The domain is [0 1] $\times$ [0 1] and the Reynolds number is set to 400. The second reference [DH03] is used to compare the performance of the finite volume method using hexahedral and tetrahedral meshes, and compare them with a Finite Element solution. From the practical viewpoint a 3D domain is considered by OpenFOAM with only one element in the third direction, and an "empty" boundary condition is set on the redundant surfaces.

Summing up, the boundary conditions of the case are the following for the fluid variables $\mathbf{u}$ and p, also see in Figure 3.1:

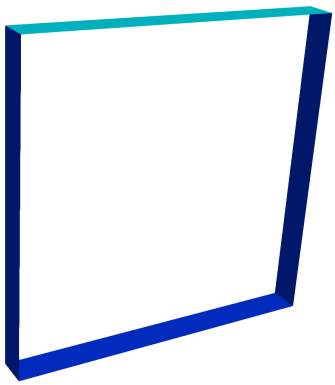
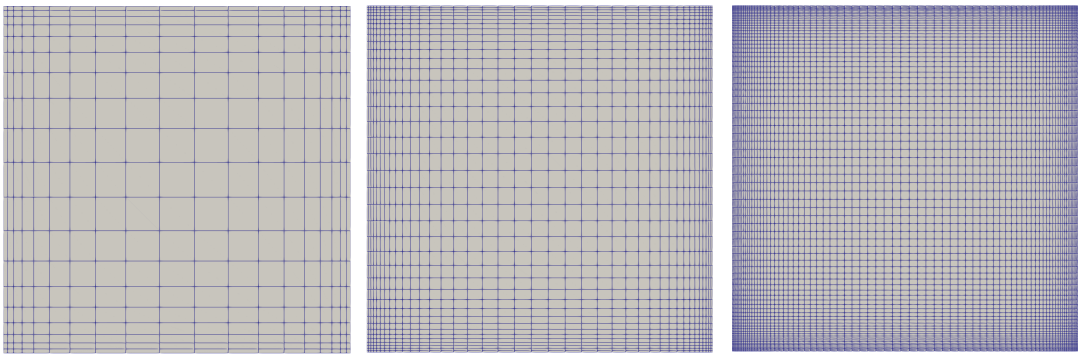


Figure 3.1: Boundary conditions applied in the case of the 2D cavity flow. Dark blue surfaces represent fixed no-slip walls, while light blue stands for the moving lid

- Lid (light blue): velocity vector ($\mathbf{u}$) = [1 0 0], and zero pressure gradient

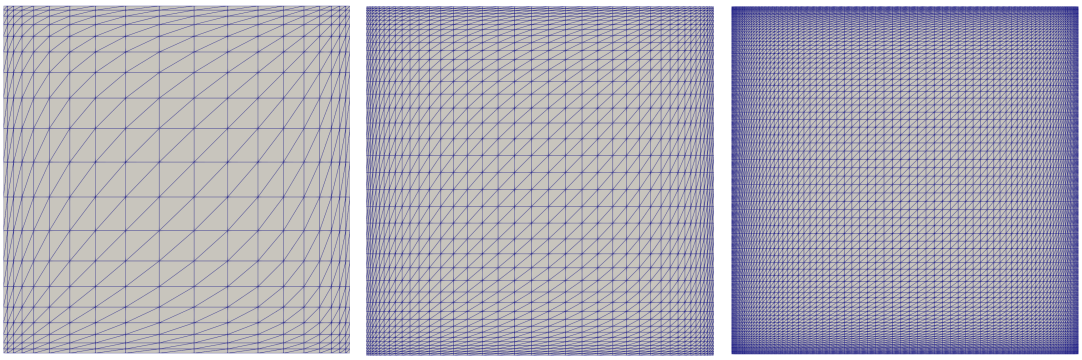
- Fixed walls (dark blue): No-slip condition for velocity, and zero gradient for pressure
- Front and back surfaces: as the problem is 2D they are set as "empty" surfaces
- At the lower left corner the value of the pressure is set to be zero. This is a necessary condition as on the walls only Dirichlet conditions have been defined for velocity, so pressure is not uniquely defined.

Three different meshes have been created for each element types (triangular and quadrilateral) with the exact same number of divisions per length. The six meshes are represented in Figure 3.2 and 3.3.



(a) Mesh Q1 (b) Mesh Q2 (c) Mesh Q3

Figure 3.2: Quadrilateral meshes of the cavity domain



(a) Mesh P1 (b) Mesh P2 (c) Mesh P3

Figure 3.3: Triangular meshes of the cavity domain

In Table 3.1 one can also find some quantitative details of the six meshes.

The results of the OpenFOAM solver is compared with a Taylor-Hood Finite ELement solution [DH03] computed on the most refined mesh. The obtained data is compared in the following order: for each element type and mesh refinement, the horizontal component of the velocity and the pressure value in the vertical symmetry plane are shown for the FEM and FV cases.

- Mesh Q1
  In this case there is a clear difference between the target, FEM and FV solutions. The relative L^2 error for the horizontal velocity component U_x has been found to be 7.05 % between the FEM and FV solutions for the same mesh.

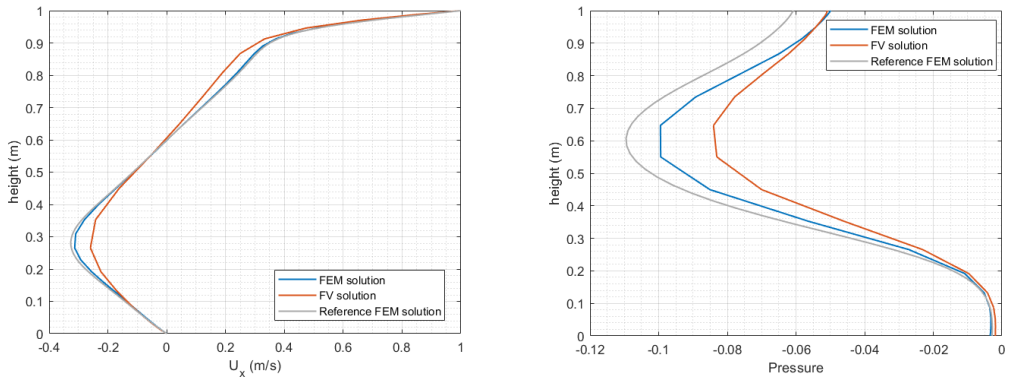


Figure 3.4: Comparison of the horizontal velocity component (left) and pressure (right) in the vertical symmetry plane of the cavity between FV and FEM solutions for Mesh Q1.

- Mesh Q2
  The error between the FEM and FV results has been reduced considerably, having an L^2 error for U_x equal to 2.7 %.

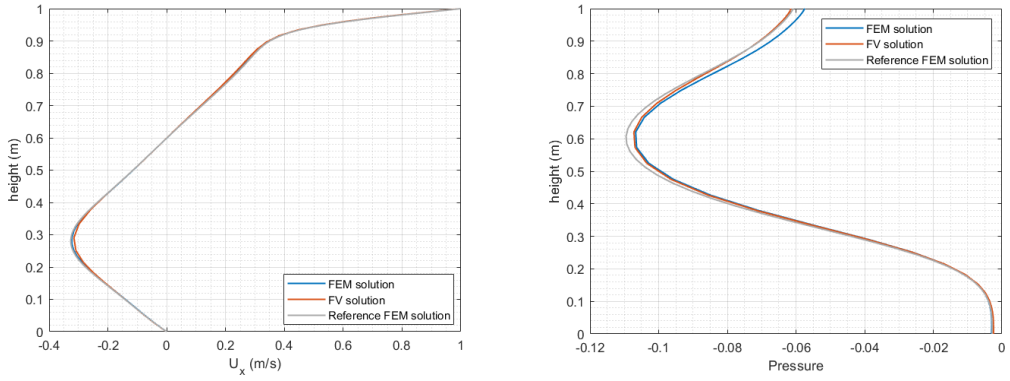


Figure 3.5: Comparison of the horizontal velocity component (left) and pressure (right) in the vertical symmetry plane of the cavity between FV and FEM solutions for Mesh Q2.

- Mesh Q3
  This is the finest mesh, where the FEM and FV solutions appears to be almost identical, showing a relative L^2 error for U_x of 1.38 %. The above results are confirmed also in the case of triangular elements, see Figures 3.73.83.9.

	Mesh Q1	**Mesh Q2**	**Mesh Q3**	**Mesh P1**	**Mesh P2**	**Mesh P3**
# Elements	361	1,521	6,241	722	3,041	12,482
# Nodes	800	3,200	12,800	800	3,200	12,800

Table 3.1: Mesh details

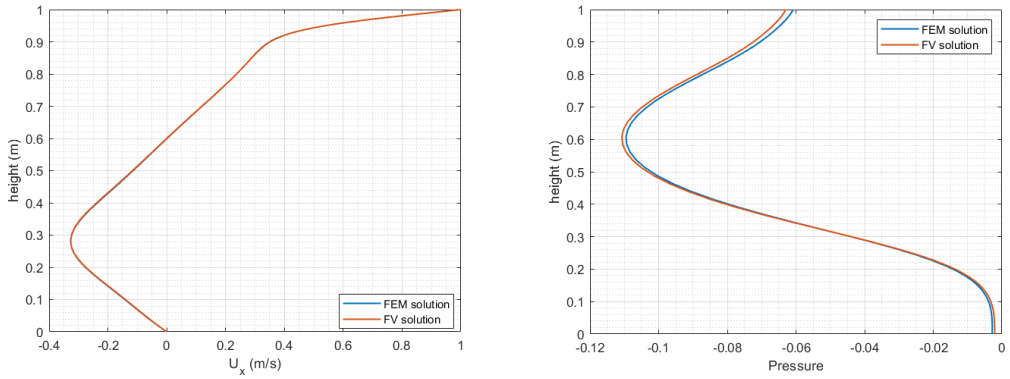


Figure 3.6: Comparison of the horizontal velocity component (left) and pressure (right) in the vertical symmetry plane of the cavity between FV and FEM solutions for Mesh Q3.

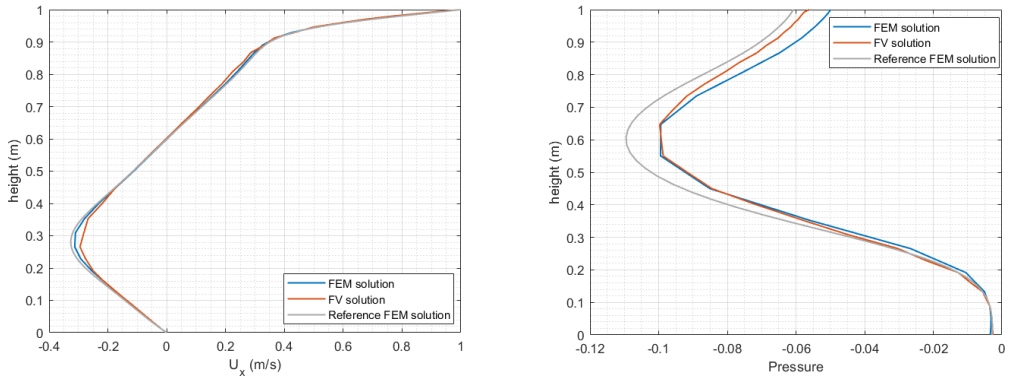


Figure 3.7: Comparison of the horizontal velocity component (left) and pressure (right) in the vertical symmetry plane of the cavity between FV and FEM solutions for Mesh P1.

Concerning the computational cost, Table 3.2 the number of iterations and elapsed clock time to achieve convergence for all 6 scenarios. It is worth noting that OpenFOAM performs better with quadrilateral meshes than with triangular ones, as both the number of iterations needed to reach convergence and the elapsed time are less for quadrilateral meshes. This is due to the well-known sensitivity of CCFV to mesh orthogonality, see [SGH18].

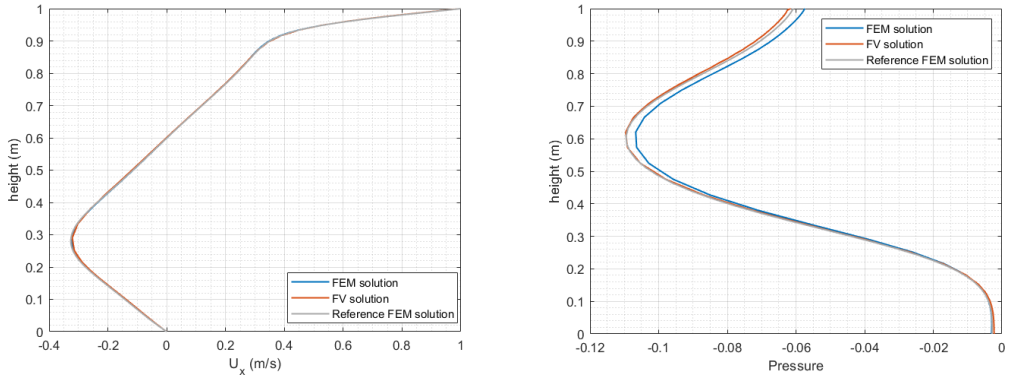


Figure 3.8: Comparison of the horizontal velocity component (left) and pressure (right) in the vertical symmetry plane of the cavity between FV and FEM solutions for Mesh P2.

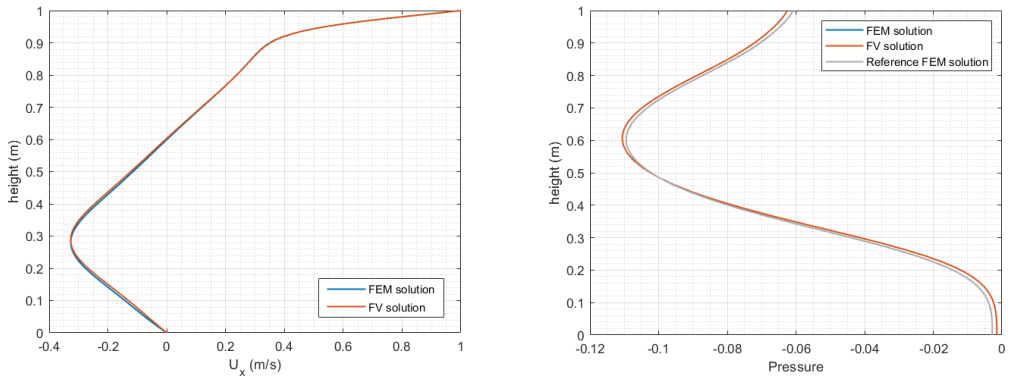


Figure 3.9: Comparison of the horizontal velocity component (left) and pressure (right) in the vertical symmetry plane of the cavity between FV and FEM solutions for Mesh P3.

	Mesh Q1	**Mesh Q2**	**Mesh Q3**	**Mesh P1**	**Mesh P2**	**Mesh P3**
# iterations	108	97	213	109	184	531
Time (s)	0.89	1.2	5.41	0.82	2.12	17.08
# cells	361	1,521	6,241	722	3,041	12,482

Table 3.2: Calculation time and iterations needed to reach convergence in all meshing scenarios

3.2 Flow past a backward facing step

This benchmark studies the flow over a backward facing step, for different Reynolds numbers. The problem is 2D, steady state, and it is generally used to study the capability of numerical schemes to capture the recirculation zone behind the step. The domain is normalized in terms of the height of the step (h), so the length of the channel is set to 80 h downstream from the step, 5 h upstream the step, and its inner height is

2 times h. The following boundary conditions are applied:

- Inlet: Fixed velocity $\mathbf{u}$=[1 0 0], zero pressure gradient
- Outlet: zero normal velocity gradient , fixed pressure p=0
- Fixed wall: no-slip condition for velocity, zero pressure gradient
- Back and front surfaces: "empty" (according to 2D problem)

The Inlet (red), Outlet (green), and Fixed wall (grey) conditions are presented in Figure 3.10.

Figure 3.10: Boundary conditions applied to the problem of the Backward facing step, inlet (red), outlet(green), and fixed walls (grey)

The solution was compared at three Reynolds numbers, namely Re=100, Re=400, and Re=700. These results can be seen in Figure 3.11.

The results represent the growth of the first recirculation zone with the increment of the *Reynolds number*, however, the length of this zone increases non-linearly with the *Reynolds number*. For a Re higher than 400, a second recirculation zone appears. The dimensionless position of the reattachment point is studied and compared to the literature, see Figure 3.12. The reattachment occurs at X_r/h= 2.84, 8.65, and 11.6 in the three cases, respectively, where X_r is the position of the reattachment along the x-axis, and h is the height of the backward facing step. These results present a relative error of 1.3 %, 5.8 %, and 2.9 % with respect to the data presented by Erturk [Ert07] and Giacomini [GSH19].

3.3 Flow past a cylinder

This benchmark is used to test the simulation of laminar unsteady incompressible flows. The problem is given by a 2D cylinder immersed in a fluid. The flow has a uniform upstream velocity given by a specific Reynolds number, namely Re=100, calculated with the diameter of the cylinder.

The physical domain is set up according to the parameters in [TMRS92], the length of the domain is 30,5 times the diameter (D), while the cross sectional length is 16 D. As in this case the diameter was set to be 2 m, and it is placed in the position P = [0,0,0], the domain has the following dimensions: [-12 49] × [-16 16]. Its boundaries are

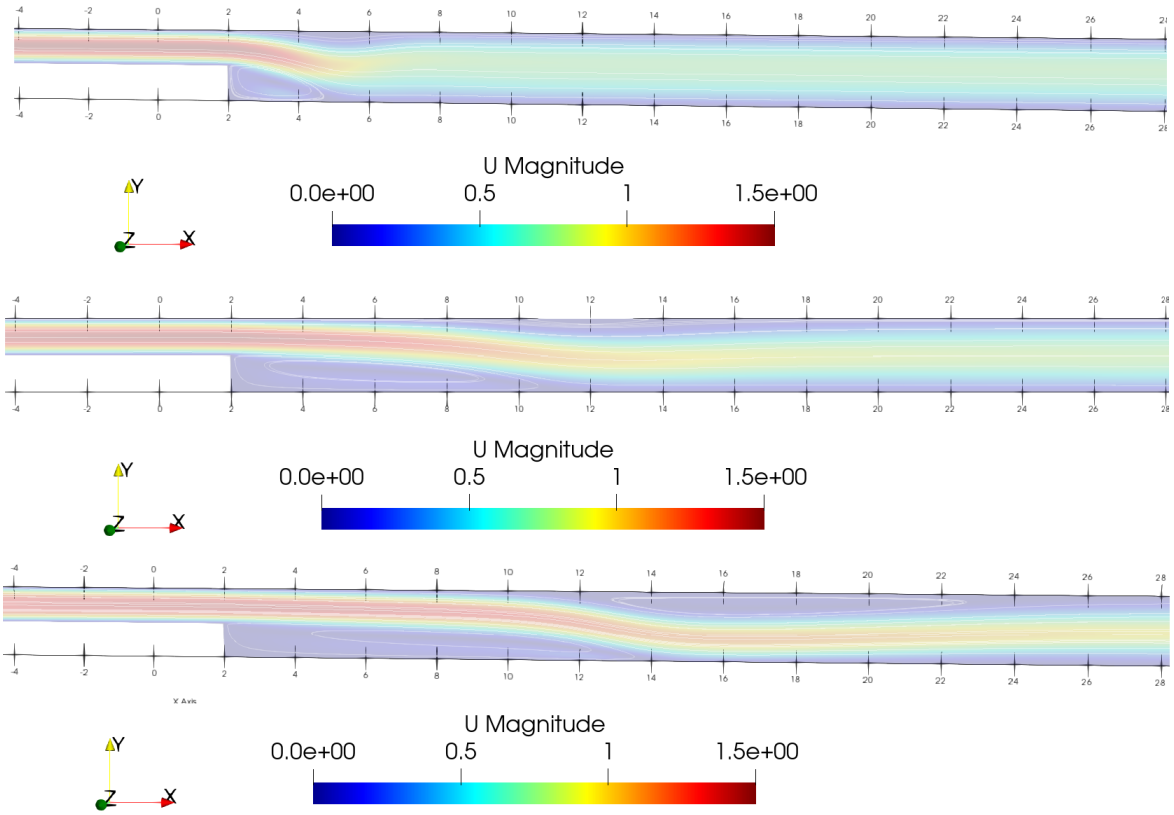


Figure 3.11: Results for the backward facing step at Re=100, Re=400, and Re=700.

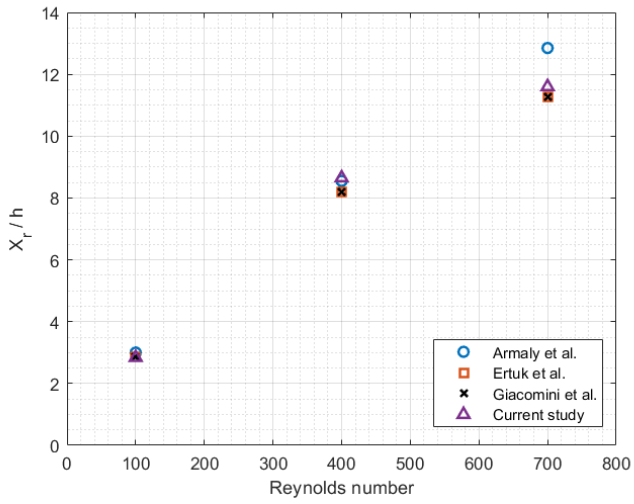


Figure 3.12: Comparison of the dimensionless reattachment position with literature data, see Armaly [ADPS83], Erturk [Ert07], and Giacomini [GSH19]

represented in Figure 3.13, where the yellow boundary represents the face where inlet boundary is defined, green, and blue faces stands for symmetry boundary conditions, while on the black boundary outlet condition is applied. The cylinder is represented by

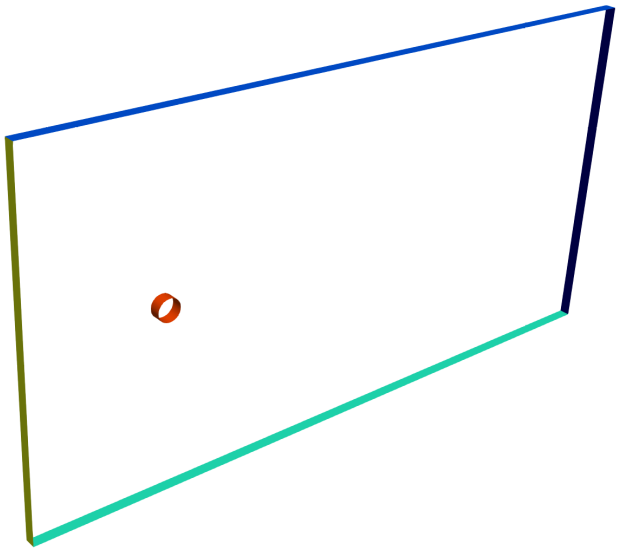


Figure 3.13: Boundary conditions of the domain of the flow past a cylinder problem

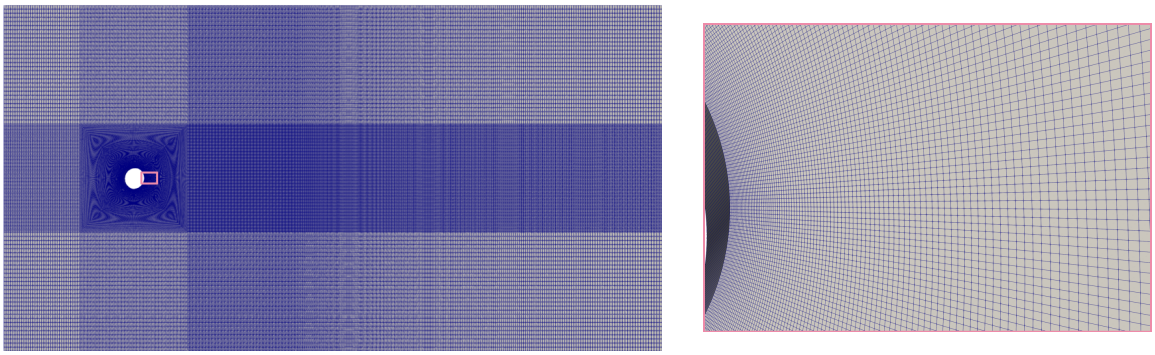


Figure 3.14: Mapped mesh used to discretise the domain around the cylinder, a closer view of the mesh near the cylinder walls can be seen in the figure on the right.

the red faces, where no-slip wall conditions are set. The previous conditions mean the following restrictions on the flow parameters:

- **Inlet boundary condition:** $\mathbf{u}$=[0,05 0 0] m/s, zero pressure gradient;
- **Outlet boundary condition:** zero normal velocity gradient, p=0;
- **Symmetry boundary conditions:** $\mathbf{u}$: $\mathbf{u}\cdot\mathbf{n}$=0, $\mathbf{t}(\nabla\mathbf{u})\cdot\mathbf{n}$=0, zero pressure gradient;
- **No-slip wall:** $\mathbf{u}$=[0 0 0], zero pressure gradient.

As the problem is 2D the front and back faces are defined to be "empty". The domain has been discretised using the mesh presented in Figure 3.14. The previously described problem is solved using pisoFoam having the temporal domain defined between 0 and 6500 s. The velocity and pressure fields at the last instant are presented in Figure 3.15

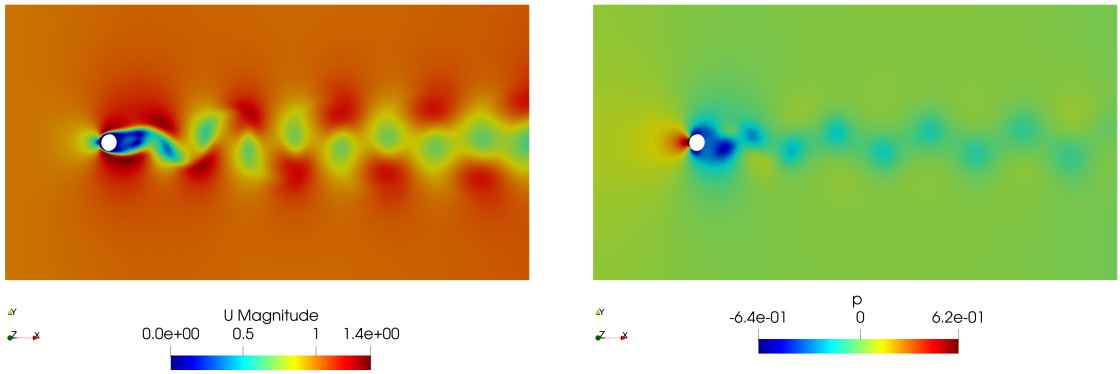


Figure 3.15: Velocity (left) and pressure field (right) showed at time 6500s

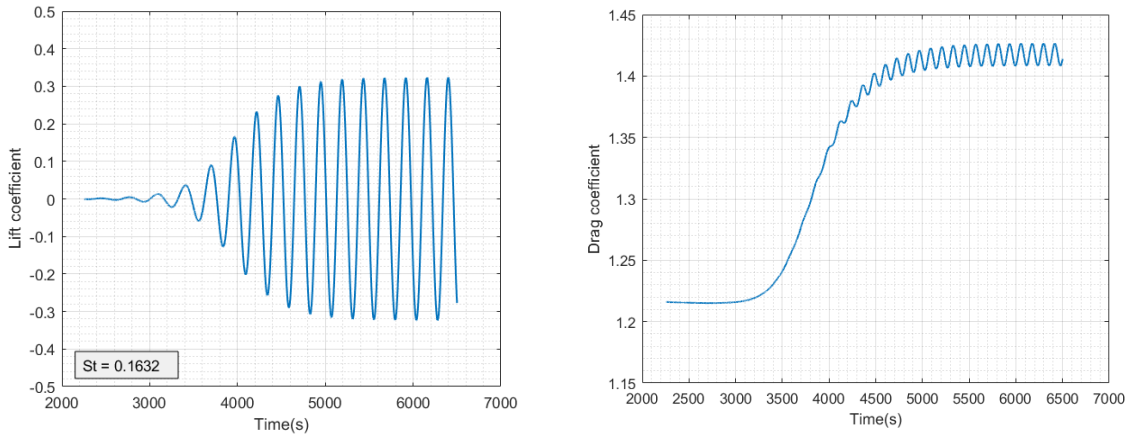


Figure 3.16: Lift (left) and Drag (right) coefficients calculated in OpenFOAM

From the obtained results the lift, drag coefficients are computed and the Strouhal number is determined. The obtained data are shown in Figure 3.16, and compared with the data in the literature, in Table 3.3..

Cases	Lift coefficient	Drag coefficient	Strouhal number
Current case	± 0.321	1.42 ±0.007	0.163
Tezduyar [TMRS92]	±0.350	1.40±0.01	0.170
Kim[JDH01]	±0.320	1.33	0.165
Calhoun[D02]	±0.300	1.35±0.014	0.175
Russel[DJ03]	±0.322	1.38±0.007	0.169
Choi[ICRcw07]	±0.315	1.34±0.011	0.164
Ming [PW09]	±0.313	1.34±0.008	0.165

Table 3.3: Literature comparison of the calculated lift, drag coefficients and Strouhal number at Re=100.

The computed solution presents relative errors of 4,42 % in the drag, 0,31 % in the

lift, and 2,98 % in the Strouhal number, compared with the reference values in the literature.

Chapter 4

Parallel simulations using OpenFOAM

4.1 Description of the computational infrastructure

All the results presented in this study have been computed on the cluster of the Laboratori de Càlcul Numéric - LaCàN de la Universitat Politècnica de Catalunya. The cluster consist of several nodes and machines that belongs to different queues that can be used according to the size of the problem at hand. Table 4.1 summarises the relevant information on the LaCàN cluster.

The HPC cluster consists of 33 computing nodes, one master node and one storage node. These nodes are connected using an Infiniband network. The queue standard top

Partition Name	Number of nodes	Number of cores	Memory Limit
standard	1	1-8	64 GB
mpi16c	1-2	1-32	192 GB
mpi28c	2-10	56-280	192 GB

Table 4.1: Partitions of the LaCàN HPC cluster

is composed of 6 x Bull Sequana X440-E5 2 x 12-core Xeon Gold 6246 (3.30 GHz/25MB cache, 2933Mhz FSB) with 384 GB RAM - EDR Infiniband Network or 3 x Dell Power Edge R630 2 x Octa-Core Xeon E5-2667 v3 (3.2 GHz/20MB cache, 2133Mhz FSB) with 128 GB RAM - QDR Infiniband Network. This queue is used for serial computation. The other two partitions are for parallel computing only. The smaller one is the mpi16c partition formed by 10 x Bull Sequana X440-E5 2 x Octa-Core Xeon Gold 6134 (3.20 Ghz/25MB cache, 2666Mhz FSB) with 192 GB RAM connected with EDR Infiniband Network. The largest partition for HPC is the mpi28c, which is formed by 10 x Bull Sequana X440-E5 2 x 14-Core Xeon Gold 6132 (2.60 Ghz/19MB cache, 2666Mhz FSB) with 192 GB RAM using EDR Infiniband Network.

4.2 Parallel performance metrics

In order to evaluate OpenFOAM performance using parallel computing architectures, the following metrics will be used:

- **Relative maximum discrepancy**
  This metric can be written as:

$$D_{max_r} = \frac{max|n_{el}(i) - \bar{n}_{el}|}{\bar{n}_{el}} \cdot 100,$$

  where $n_{el}(i)$ is the number of cells per processor at CPU i and $\bar{n}_{el}$ is the average number of cells per processor. This metric measures the ratio in percentage of the average number of cells per processor with respect to difference between the number of cells at the worst processor (where highest or lowest number of cells are assigned) and the average number of cells. This is a measure of the largest inhomogeneity of the workload across processors.

- **Mean discrepancy**
  This metric is similar to the previous one, but it measures an average discrepancy, so it expresses mean inhomogeneity of the workload subdivision:

$$\bar{D}_r = \frac{1}{n_{proc}} \sum \frac{|n_{el}(i) - \bar{n}_{el}|}{\bar{n}_{el}} \cdot 100,$$

  where n_{proc} is the total number of processors used.

- **Speed up**
  This metric gives the ratio between the clock time of running the case in serial (1 processor only), and the clock time of the parallel computation, giving the user the information of the speed up of the computation if the domain is split up into several subdomains.

$$Sp = \frac{T_s}{T_p},$$

  where T_s is the clock time of the serial, and T_p is the clock time of the parallel computation. This value can be defined on two different scaling basis. When the term strong scaling is used, it refers to the case when the computation time is evaluated while increasing the number of CPUs and the global problem size is maintained fix (Amdahl's law). Weak scaling evaluates the computational time while increasing the problem size for a fixed size of processors (Gustafson's law) [Wol]. In this study, strong scaling have been analysed.

- **Relative efficiency**
  This metric compares the efficiency of the parallel computation by evaluating the ratio of the total run time needed to solve the problem in serial (using only 1 processor) T_s, and the total CPU run time, which is the sum of the clock time of each processor T_t. Sometimes T_t is approximated as $n_{proc} \cdot T_p$. However, as the load of each processor is not the same there may be important differences.

$$\eta_{rel} = \frac{T_s}{T_t} \approx \frac{T_s}{n_{proc} \cdot T_p},$$

If this value is greater than one, the overall time spent by the processors in parallel is smaller than the clock time of the serial run, so the algorithm is accelerated and it converges faster.

It can be mentioned that there are other efficiency metrics that are used in scalability analsysis, however, they are not used in this study. One is the *isoefficiency*, which gives the relation between the problem size and the number of machines to maintain a constant efficiency value. The other metrics is the *CMP efficiency*, which gives an efficiency while maintaining the local problem size constant (the increment in the number of processors is linear with the global problem size). For more details, interested readers are referred to [Fie95].

4.3 Internal laminar flow in Sbend

The order in which the scalability testing is done follows the principle of graduality, which means that the starting point of this study is a reasonably small mesh, that gives the opportunity to study several parameters, such as the influence of the partitioner on the convergence, and to set the lower limit for an optimal number of cells per processors.

For this reason, the first case that has been parallelised is an internal laminar flow in a three dimensional S-bend geometry. This problem allows a discretisation with a relatively small mesh, without loosing important details, or physical phenomena.

4.3.1 Case setup and simulation results

The geometry is created in GiD, and it is composed of a rectangular tube with a cross section of 0,04 $\times$ 0,04 meters, bended in an S-shape in the horizontal and vertical directions as well. The bend starts at 0,335 meters from the inlet and its total length is about 0,18 meters, reaching a displacement of 0,048 meters in the two directions with respect to the initial cross section.

The boundary conditions of the problem are the following ones:

- **Inlet**: fixed velocity of $\mathbf{u}$=[1 0 0] m/s, zero pressure gradient
- **Outlet**: zero gradient for the velocity, and fixed pressure p=0 Pa.
- **Fixed walls**: no-slip wall condition for the velocity, zero pressure gradient

The boundary conditions can be observed in Figure 4.1

To guarantee a laminar flow inside the pipe, the viscosity of the fluid has been determined in order to give a Reynolds number Re=1000. The problem has been solved using the steady state algorithm, simpleFoam. The problem has been computed on the mpi16c queue of the LaCàN cluster.

The solution have been obtained on two different meshes, a coarser mesh and a refined mesh. The first one consists of approximately 900,000 cells, while the latter is made of approximately 3.6 millions of elements. The streamlines of the flow at the S-bend are presented in Figure 4.2 on the finer mesh.

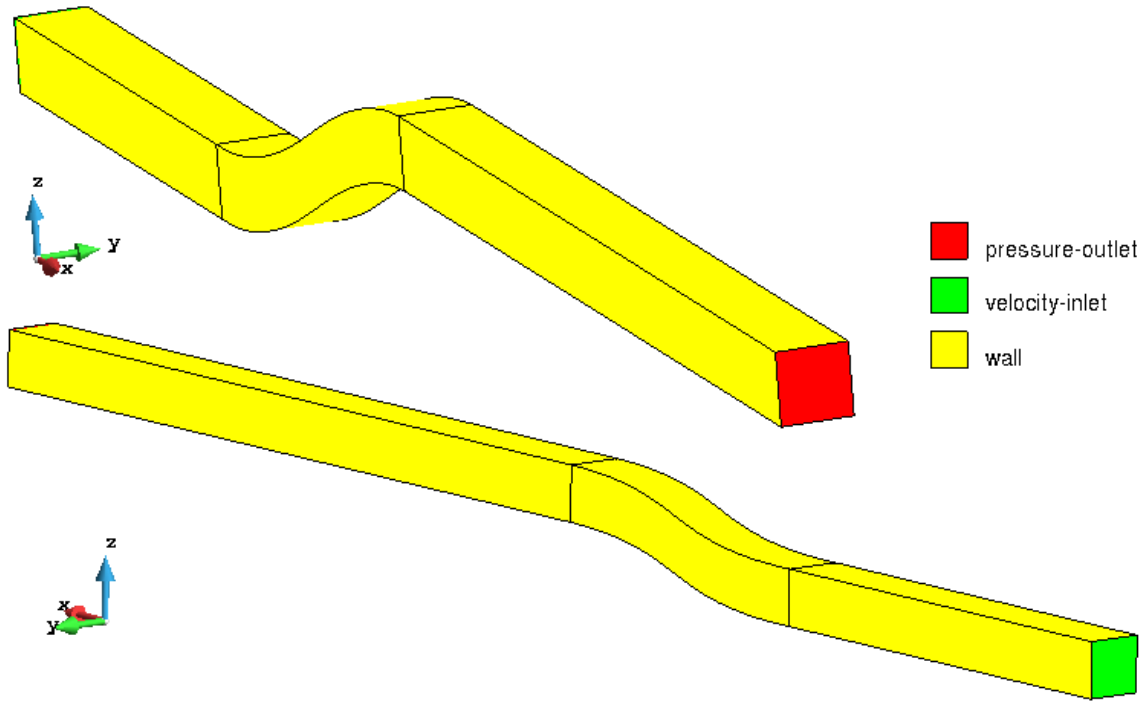


Figure 4.1: Boundary conditions of the S-bend geometry

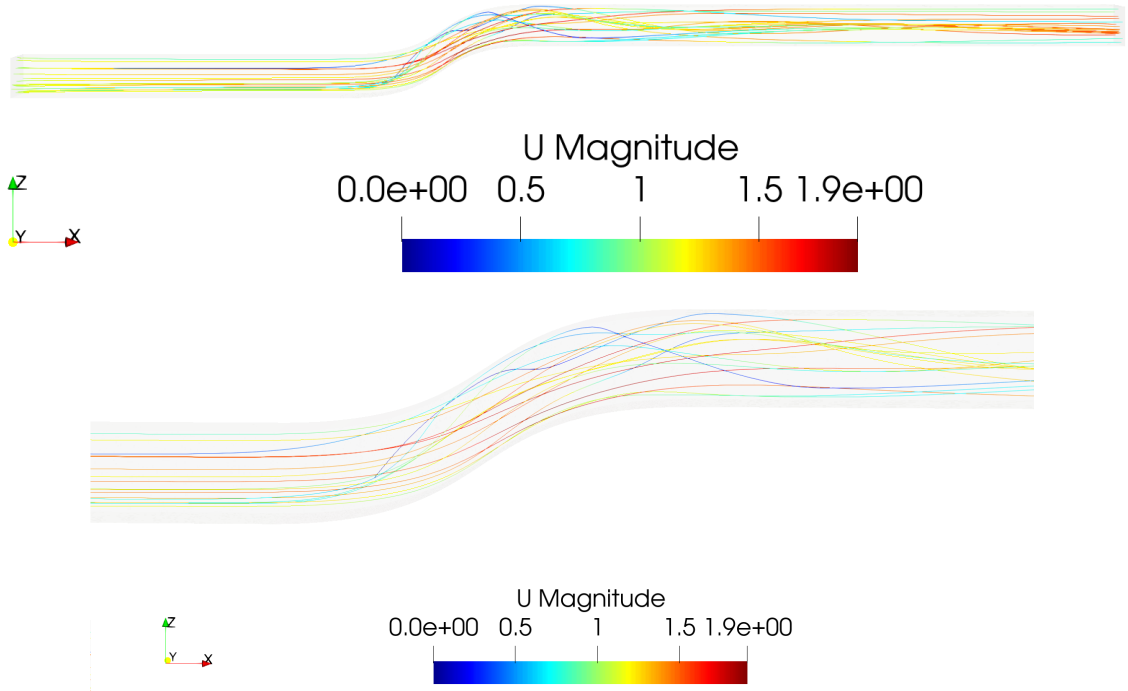


Figure 4.2: Fluid flow at the Sbend at Re=1000, view of the whole domain above, and a closer look to the bending section below

The results of the coarser and the refined mesh are compared at three different sections, at 0.5, 0.76, and 1.12 meters from the inlet. These results can be seen in Figure 4.3.

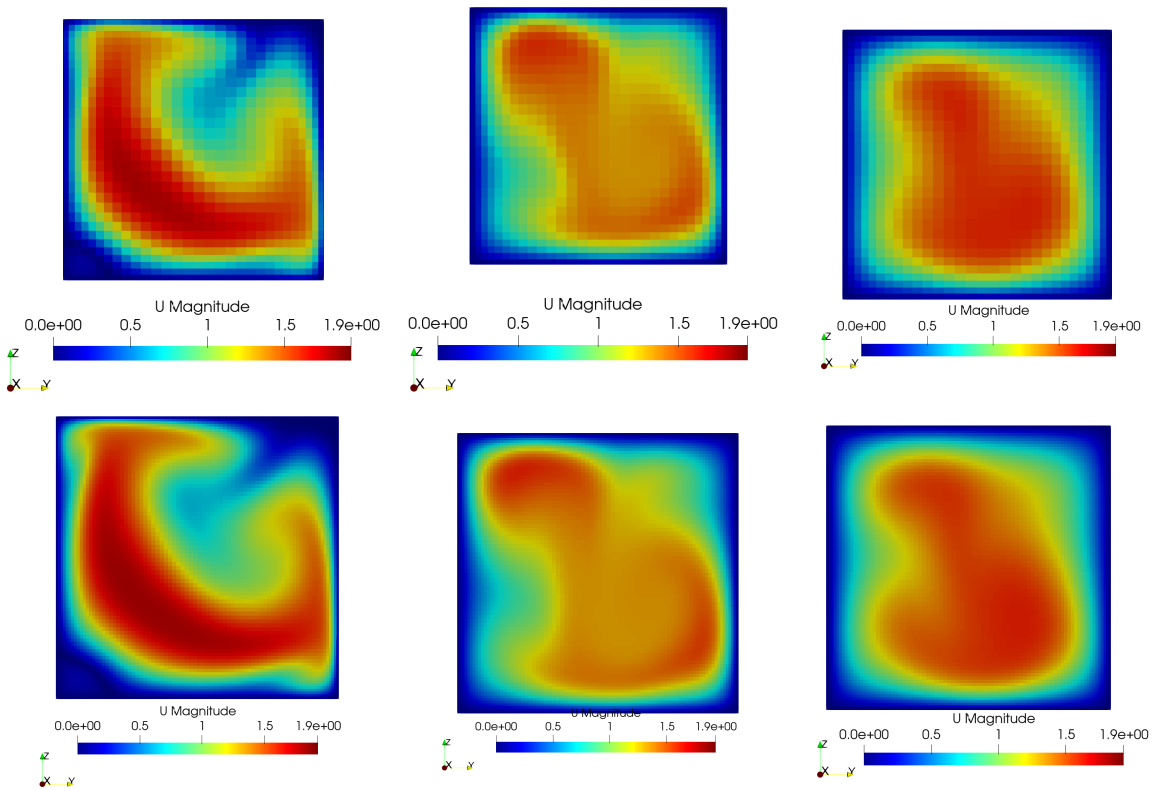


Figure 4.3: Velocity map at the cross-section of the tube at 0.5m, 0.76, and 1.12 meters from the inlet, being the coarser mesh presented above, and the refined below.

In these figures the stabilisation of the cross flows in the tube can be observed after the Sbend. It can be seen that the coarser mesh can already give reasonable results about the velocity pattern and the flow in the tube. Only some smaller details are lost, but the main behaviour of the flow is reproduced.

4.3.2 Scalability results

The testing is made using two different meshes, a coarser and a refined one. The objective of these tests is to determine an optimal partitioner to achieve better performance, and to study the behaviour of the scalability. The latter is evaluated when the test is done using one node of processors only, and also when two nodes are involved, and there is a important communication effort between the nodes as well. For the one node study a mesh of 900,000 hexahedral cells is used, while for the multiple nodes study, mesh of 3.6 millions of cells is considered.

Case 1: coarse mesh

In this test the problem is computed using 4 different partition sizes. It is split into 2, 4, 8, and 16 subdomains that are computed in parallel, and the results are compared with the results of the serial computation. The partitioning is done by three different partitioners, namely, simple, hierarchical, and scotch.

2 CPUs	Simple	Hierarchical	Scotch
# cells	900,000	900,000	900,000
# cells per CPU	450,000	450,000	450,000
D_{max_r} (%)	0	0	0
$\bar{D}_r$ (%)	0	0	0
Clock Time (s)	661.2	659.7	658.3
Total CPU time (s)	1,325	1,323	1,325
η_{rel}	1.056	1.057	1.056
Speed Up	2.116	2.121	2.125

Table 4.2: Running the Sbend case with the coarser mesh on 2 CPUs

4 CPUs	Simple	Hierarchical	Scotch
# cells	900,000	900,000	900,000
# cells per CPU	225,000	225,000	225,000
D_{max_r} (%)	0	0	0.400
$\bar{D}_r$ (%)	0	0	0.200
Clock Time (s)	314.3	313.3	317.2
Total CPU time (s)	1,284	1,267	1,284
η_{rel}	1.090	1.104	1.090
Speed Up	4.452	4.465	4.410

Table 4.3: Running the Sbend case with the coarser mesh on 4 CPUs

8 CPUs	Simple	Hierarchical	Scotch
# cells	900,000	900,000	900,000
# cells per CPU	112,500	112,500	112,500
D_{max_r} (%)	100	0	0.56
$\bar{D}_r$ (%)	50	0	0.296
Clock Time (s)	298.0	167.0	160.6
Total CPU time (s)	2,420	1,353	1,302
η_{rel}	0.578	1.034	1.075
Speed Up	4.694	8.375	8.710

Table 4.4: Running the Sbend case with the coarser mesh on 8 CPUs

As it can be observed in Tables 4.2-4.5, simple and hierarchical partitioners create subdomains with the exact same number of cells in the first two scenarios. Scotch only

16 CPUs	Simple	Hierarchical	Scotch
# cells	900,000	900,000	900,000
# cells per CPU	56,250	56,250	56,250
D_{max_r} (%)	294	0	0.999
$\bar{D}_r$ (%)	70.632	0.000	0.530
Clock Time (s)	303.5	99.6	93.5
Total CPU time (s)	4,823	1,615	1,517
η_{rel}	0.290	0.866	0.922
Speed Up	4.610	14.050	14.964

Table 4.5: Running the Sbend case with the coarser mesh on 16 CPUs

decomposes the domain with the exact weight for two subdomains, and the relative mean discrepancy and relative maximum discrepancy start to increase with the number of subdomains. This is not true for the hierarchical decomposer, who maintain the same number of cells for all subdomains for all cases. It can be seen that the simple partitioner starts to suffer with the decomposition above 4 CPUs, provoking an imbalance in the number of cells per CPU. This leads to relative efficiencies of only 0,290 when the domain is decomposed in 16 partitions. The Speed Up does not change effectively above 4 CPUs with this partitioner. Henceforth, the simple partitioner will thus be neglected.

The remaining two alternatives scale well in every case. It is worth noticing that they both present their maximum efficiency when the number of cells per CPU is around 225,000, that is for 4 processors. The relative efficiency and Speed Up data are displayed in Figure 4.4.

In order to estimate the effect of communication costs among the nodes, the previous mesh is now split into 8, 16, and 32 subdomains distributed over two computing nodes.

8 CPUs	Hierarchical	Scotch
# cells	900,000	900,000
# cells per CPU	112,500	112,500
D_{max_r} (%)	0.000	0.56
$\bar{D}_r$ (%)	0	0.296
Clock Time (s)	174.4	163.7
Total CPU time (s)	1,412	1,336
η_{rel}	0.991	1.047
Speed Up	8.020	8.547

Table 4.6: Running the Sbend case with the coarser mesh on 8 CPUs with 2 nodes

The results of this case suggest that the minimum weight per CPU should be above 100,000 cells. It is worth noticing that the efficiency of the parallel computing slightly decays by increasing the number of partitions when using hierarchical decomposition. On the other hand, scotch slightly decays for 16 CPUs but it increases when 32 subdomains are created, so in the case when both nodes are working at maximum capacity.

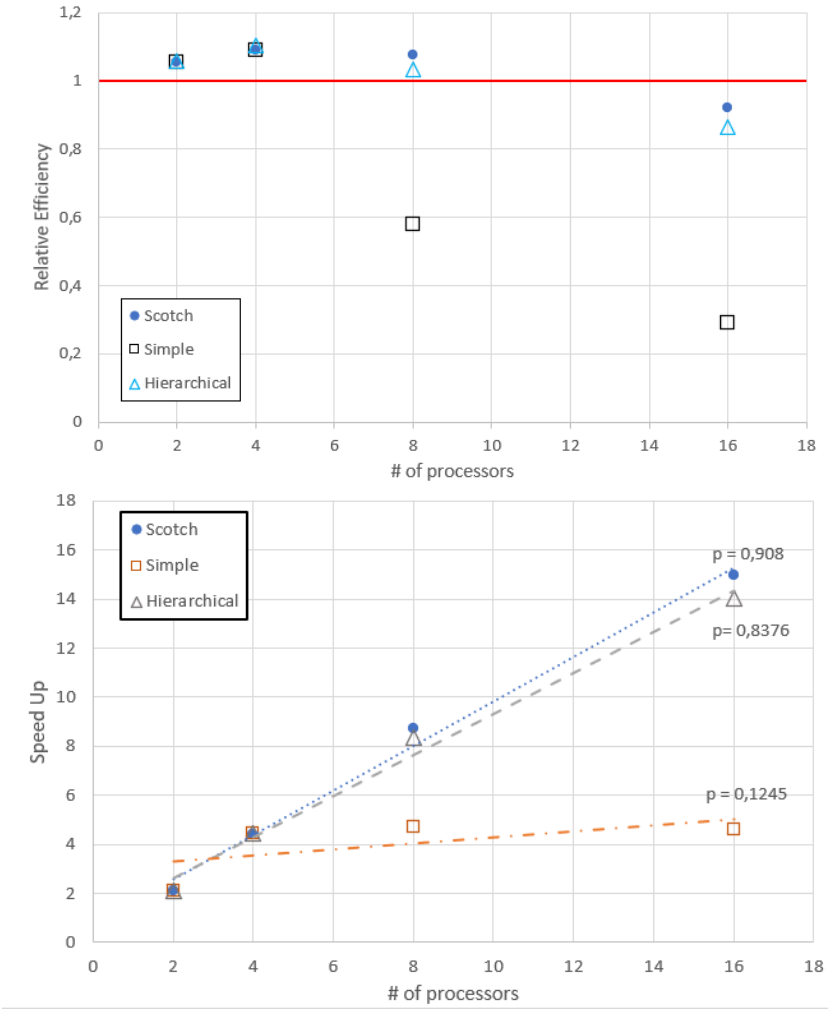


Figure 4.4: Relative efficiency (above) and Speed Up (below) obtained with the coarse mesh used on the S-bend geometry. The value of the parameter p represents the estimated pendent for the linear regression curve

16 CPUs	Hierarchical	Scotch
# cells	900,000	900,000
# cells per CPU	56,250	56,250
D_{max_r} (%)	0.000	0.999
$\bar{D}_r$ (%)	0	0.530
Clock Time (s)	99.0	95.8
Total CPU time (s)	1,594	1,554
η_{rel}	0.878	0.900
Speed Up	14.132	14.597

Table 4.7: Running the Sbend case with the coarser mesh on 16 CPUs on two nodes

Case 2: fine mesh

In order to study the effect of employing more than 2 nodes (and more than 32 processors), the second mesh is used. It is split up until 3 nodes and 84 processors. This case

32 CPUs	Hierarchical	Scotch
# cells	900,000	900,000
# cells per CPU	28,125	28,125
D_{max_r} (%)	0.000	0.999
$\bar{D}_r$ (%)	0	0.603
Clock Time (s)	49.4	46.7
Total CPU time (s)	1,611	1,526
η_{rel}	0.868	0.917
Speed Up	28.331	29.983

Table 4.8: Running the Sbend case with the coarser mesh on 32 CPUs on two nodes

has been run on the partition mpi28c of the LaCàN cluster.

14 CPUs	Scotch	Hierarchical
# cells	3,600,000	3,600,000
# cells per CPU	257,142.9	257,142.9
D_{max_r} (%)	0.728	8.333e-4
$\bar{D}_r$ (%)	0.260	2.857e-4
Clock Time (s)	842.1	876.3
Total CPU time (s)	11,810	11,579
η_{rel}	0.883	0.901
Speed Up	12.377	11.893

Table 4.9: Running the Sbend case with the refined mesh on 14 CPUs

28 CPUs	Scotch	Hierarchical
# cells	3,600,000	3,600,000
# cells per CPU	128,571.4	128,571.4
D_{max_r} (%)	0.900	1.222e-3
$\bar{D}_r$ (%)	0.460	4.286e-4
Clock Time (s)	642.0	667.8
Total CPU time (s)	18,058	18,768
η_{rel}	0.578	0.556
Speed Up	16.233	15.606

Table 4.10: Running the Sbend case with the refined mesh on 28 CPUs

It can be seen that when increasing the number of subdomains and nodes the scotch algorithm outperforms the hierarchical partitioner. However, both decomposers provide the best results for the smallest number of processors. This can be seen as the Relative Efficiency drops considerably when dealing with less than 130,000 cells per CPU. Nonetheless, it is also observed the scotch method increases its relative efficiency and Speed Up when two nodes are working at their maximum capacity (56 CPUs).

56 CPUs	Scotch	Hierarchical
# cells	3,600,000	3,600,000
# cells per CPU	64,285.7	64,285.7
D_{max_r} (%)	0.998	3.556e-3
$\bar{D}_r$ (%)	0.83	0.0014
Clock Time (s)	286.3	353.1
Total CPU time (s)	16,142	19,546
η_{rel}	0.646	0.534
Speed Up	36.400	29.517

Table 4.11: Running the Sbend case with the refined mesh on 56 CPUs

84 CPUs	Scotch	Hierarchical
# cells	3,600,000	3,600,000
# cells per CPU	42,857.1	42,857.1
D_{max_r} (%)	0.984	4.333e-3
$\bar{D}_r$ (%)	0.524	5.952e-4
Clock Time (s)	217.2	238.4
Total CPU time (s)	18,396	20,129
η_{rel}	0.567	0.518
Speed Up	47.986	43.713

Table 4.12: Running the Sbend case with the refined mesh on 84 CPUs

These results are represented in Figure 4.5, whereas the detailed description of the performance of the two approaches is reported in Tables 4.9-4.12.

4.4 External turbulent flow around a vehicle

4.4.1 Description of DriveAer model

The DriveAer model is a generic car CAD model created at the Technical University of Munich in collaboration with AUDI AG and BMW Group. This model is widely used in the scientific community, as generic bodies like the Ahmed body do not provide physically significant results to be employed for the shape optimisation or evaluation of the flow field around realistic car geometries [Mun22]. The dimensions of the model are reported in Figure 4.6.

The domain length is 18 L_c, with 7 L_c before the car, 10 L_c after the car, 48 W_c width, and 13 H_c height, where L_c, W_c, and H_c are the length, width, and height of the car, respectively.

4.4.2 Case setup and simulation results

Mesh generation of complex geometries is extremely expensive in terms of man-hours of specialized technicians. Fot the current study, the mesh available from [IJ19] was

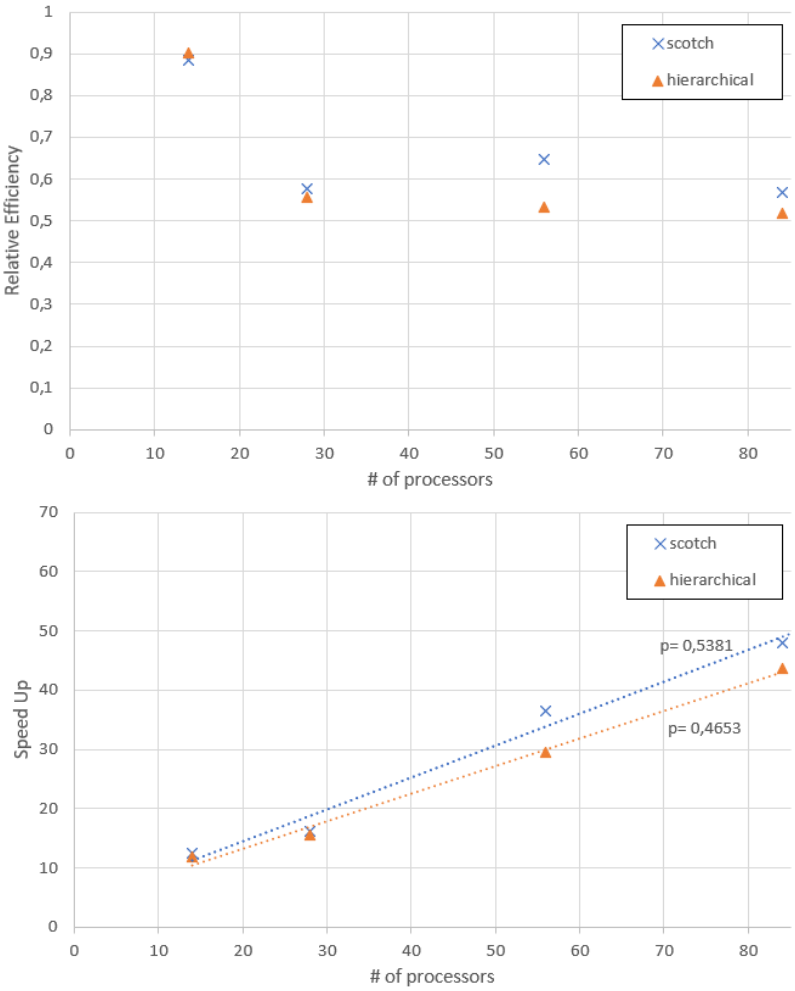


Figure 4.5: Relative efficiency (above) and Speed Up (below) obtained on the fine mesh used on the S-bend geometry. The value of the parameter p represents the estimated pendent for the linear regression curve

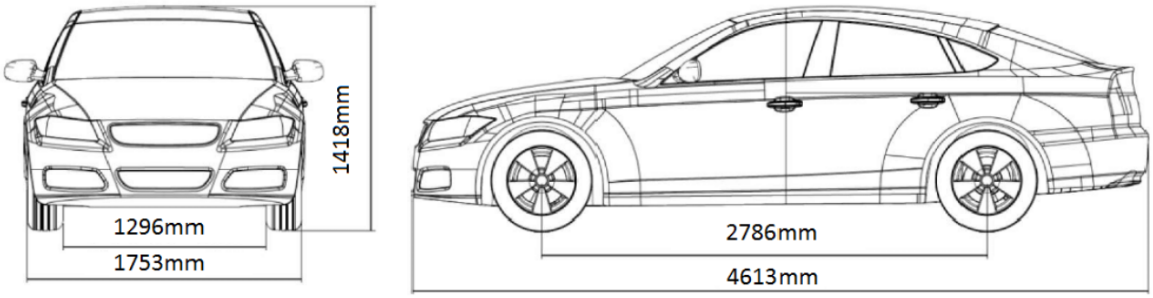


Figure 4.6: Representation and dimensions of the DriveAer car model[AJSM18]

employed. The mesh contains approximately 25 million cells, 96.4% of which are hexahedral elements, 2.9 % polyhedral, and 0.45 % prismatic elements. Tetrahedral, wedges, and piramids are only 0.25 % of the total number of cells. The mesh is non-uniform,

with local refinement in regions such as the mirrors, and the wheels. The coarsest elements have dimensions of 1 meter (far from the car surface), while on the car surface this data varies between 15.8 and 4 mm. The average mesh size on Some sketches of the mesh are displayed in Figure 4.7.

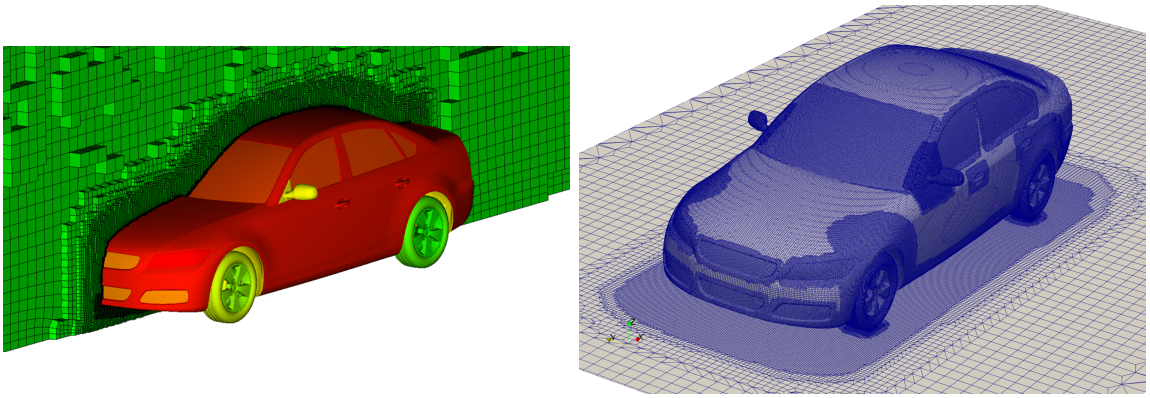


Figure 4.7: The development of several mesh layers in the domain, obtained from [IJ19] on the left, and the different mesh refinement zones on the right shown in Paraview

The boundary conditions are the following ones:

- Floor: Moving wall (constant velocity of $\mathbf{u}$=[40 0 0]m/s).
- Car surface: No-slip walls (steady)
- Inlet: velocity inlet ($\mathbf{u}$ constant, 40 m/s in the direction of the motion of the vehicle
- Outlet: pressure outlet, mean pressure is set to be 0, the normal gradient of the velocity is also zero
- Top and lateral surfaces (corresponding to far field): Perfectly slip walls.

The free-stream velocity is of 40 m/s, which corresponds to a *Reynolds number* equal to 4,87 millions. The relative pressure has been set to zero. The problem has been modelled with the k-ω SST model. As the mesh layers around the body are not sufficiently small ($y^+ > 1$) wall functions are needed to estimate the boundary condition for ω, ν_t, and k.

After the problem converged, the results are compared to Heft [HIA12]. First, the pressure coefficient map on the driver's window is compared in Figure 4.8.

The obtained results reproduce with good resemblance the ones presented in the literature. The pressure coefficient distribution on the windshield is also compared, see Figure 4.9. Important differences can be seen here, with a high pressure region in the lower central part of the windshield. Whereas, the pressure coefficient distribution elsewhere shows a similar pattern presented by Heft in [HIA12].

To verify quantitatively the difference between the results of the literature and the model used in the present study, the pressure coefficient has been plotted in the

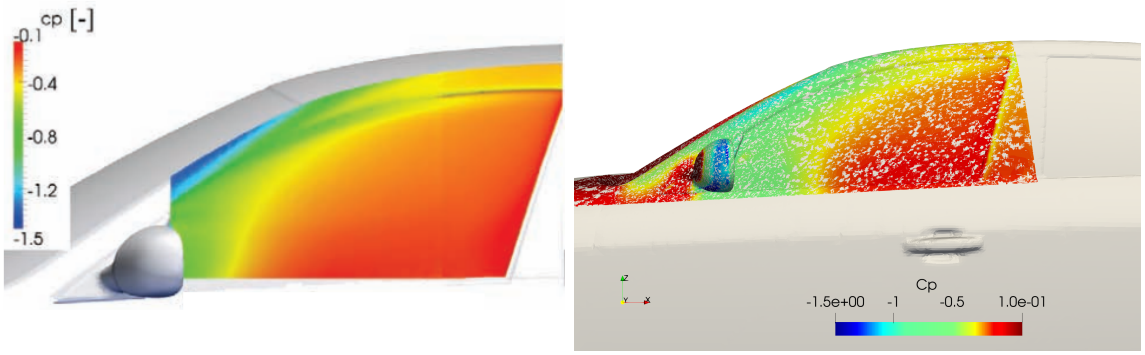


Figure 4.8: Comparison of the pressure coefficient map on the driver's window, Heft [HIA12] on the left, and current case on the right

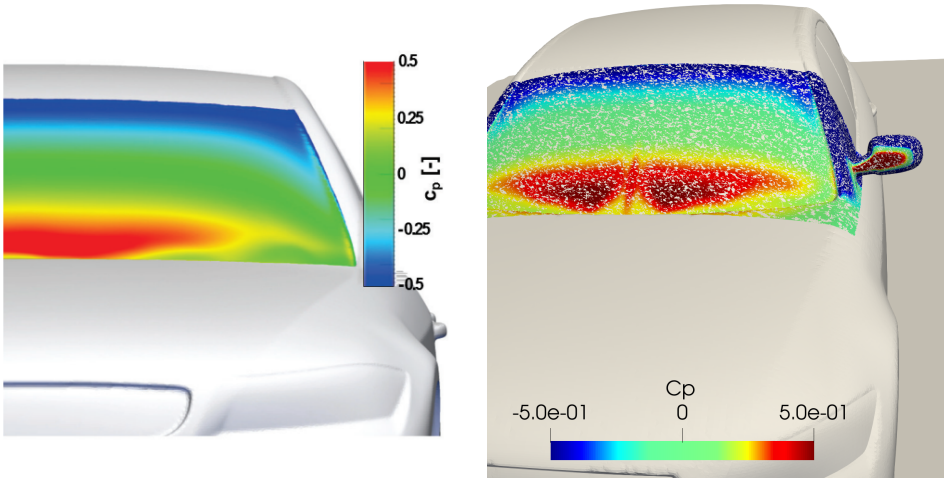


Figure 4.9: Comparison of the pressure coefficient map on the windshield of the DriveAer model, Heft [HIA12] on the left, and current case on the right

midplane of the vehicle, in the first part of the model. This suggests that the mesh should be further refined in that region (Figure 4.10).

From the numerical results it can be seen that the obtained pressure distribution represents with good resemblance the ones from the literature, however, there are some critical points where deviations can be observed. One is in the middle of the hood, where a pressure drop can be seen. Another is the already mentioned region in the windshield where the pressure shows a maximum value. In the current study the maximum value of the pressure coefficient at 220 mm is below the reference values. On the other hand, the pressure drop around 475 mm is over estimated, with respect to the bibliographical data.

Finally, the drag coefficient of the current model and the one in [ATN12] are compared. The drag coefficient of this study gives a value of 0,284, compared with the reference value of 0,246. This gives a relative error of around 15 %, confirming that a refinement of the mesh around the vehicle is required to obtain more accurate results.

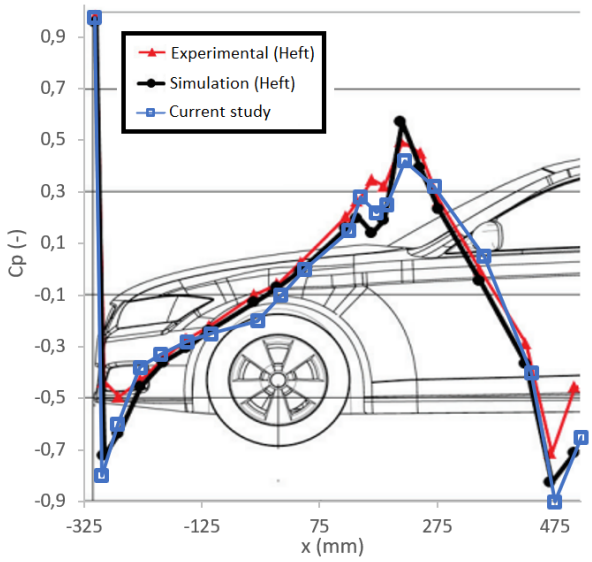


Figure 4.10: Pressure coefficient distribution in the symmetry plane of the DriveAer model

4.4.3 Scalability results

The scalability testing of this geometry consists of studying the relative efficiency and Speed up while increasing the number of subdomains. The partitioner that was used for this purpose is *scotch* due to its better performance with respect to the hierarchical approach, as seen in Section 4.3.2. The domain was partitioned into 28, 42, 56, 84, 112, and 224 subdomains, using mpi28c on the cluster. The results are summed up in Table 4.13. It is worth mentioning that all tests are performed considering a perfectly linear scalability using 28 processors, that is, the baseline value of computing time using one processor is estimated as 28 the time required by 28 processors.

Scotch	28	56	84	112	224
# cells	25.33 M	25.33 M	25.33 M	25.33 M	25.33 M
# cells per CPU	904 k	452 k	301 k	226 k	113 k
D_{max_r} (%)	0.799	0.100	0.998	0.999	0.999
Clock Time (s)	49,250	22,492	13,934	8,912	4,643
Total CPU time (s)	1,379,160	1,259,400	1,171,080	997,980	1,040,100
η_{rel}	1	1.0951	1.178	1.382	1.326
Speed Up	28	61.31	98.97	154.75	297.00

Table 4.13: Scalability results with the DriveAer model

The number of cells per CPU varies between almost 1 million elements and 100,000. In 4.11 it is seen that using the full capacity of the nodes, the relative efficiency grows until reaching a maximum value at 112 CPUs. Also, for less than 200,000 cells per CPU, the efficiency starts to decrease slowly. The variation of the relative efficiency with the increment of the number of machines is displayed in Figure 4.11.

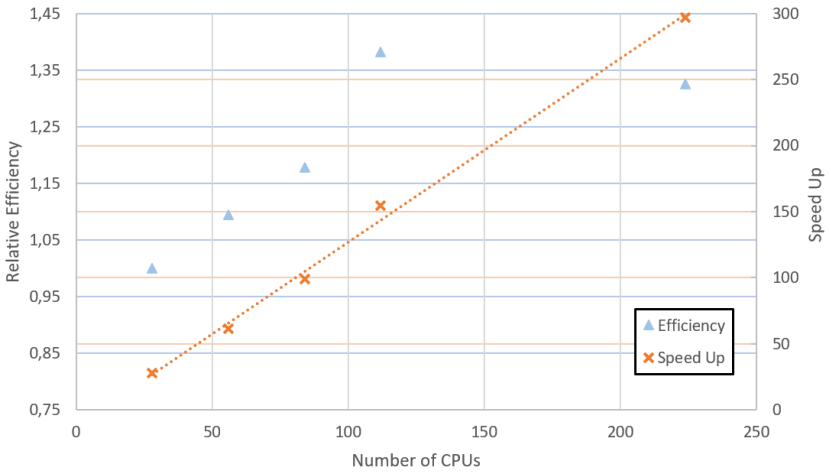


Figure 4.11: Scalability results of the DriveAer model, presenting the Relative Efficiency and Speed Up at a range between 28 and 224 CPUs

Figure 4.11 also reports the Speed Up which scales linearly with the number of CPUs. The maximum relative efficiency is reached for 112 CPUs (225,000 cells per CPU). The value of the Speed Up is higher than the number of CPUs, e.g. the Speed Up is around 154 when computing with 112 CPUs.

Chapter 5

Conclusions

The objectives of this study was twofold: to get acquainted with OpenFOAM, and to define a set of best practices to run it using parallel architectures.

To achieve the first goal, several benchmarks have been tested, namely the lid-driven cavity, flow over a backward facing step, and flow past a circular cylinder. The first benchmark has been used to compare the influence of triangular and quadrilateral mesh on the accuracy of the results and computational clock time. It has been observed that the number of iterations needed to reach convergence with triangular elements is more than twice the value corresponding to quadrilateral elements, with the finest mesh. Comparing the results with a Finite Element solution, the error between the two solutions reduces down to 1.38% with the most refined mesh. This verifies that the Finite Volume Method can give results with high accuracy when the mesh is sufficiently fine. In the case of the second benchmark the capability of catching recirculation regions of OpenFOAM has been studied, obtaining results that correspond to the values found in the literature. Studying the third benchmark, the capability of OpenFOAM to treat unsteady problems were verified with errors below 5% with respect to the reference solution.

In order to define some guidelines for HPC, two physical problems were studied, a laminar internal flow in an S-bend and the turbulent external flow around a car geometry. For the S-bend problem, a coarse and a fine mesh have been created. With the coarse mesh, three partitioners have been analysed, namely the simple, hierarchical, and scotch decomposers. The case has been run on 2, 4, 8, and 16 processors. It has been observed, that up to 4 partitions simple and hierarchical split the domain in equal size subdomains, while scotch starts to create non-equal size partitions. Simple partitioner demonstrated to be inefficient for a decomposition of 8 CPUs or above, whereas the hierarchical and scotch decomposers provided comparable results.

The same case has also been computed using 2 nodes with up to 32 CPUs. In that case the similar behaviour between the two studied partitioners is confirmed. Analysing the HPC metrics, it is clear that the case is too small for 2 node computation, so cases below 1 million elements should not be run on two nodes.

The next study uses the same geometry of the S-bend, but increasing the mesh count up to 3.6 millions. In this case, an important difference of performance between the hierarchical and scotch methods was seen when creating 56 partitions or more. For

this reason, it has been decided that the hierarchical method is not recommended for more than 28-32 partitions.

The last model under analysis is the external turbulent flow around a car. For this purpose the DriveAer model has been studied using the mesh created from [IJ19]. This mesh consists of 25.33 million elements, and it was run on 28-224 CPUs. Scotch partitioner was employed. The maximum efficiency has been found for 112 CPUs, which corresponds to approximately 226,000 cells per CPU in average. It has been observed, that the relative efficiency increases linearly with the number of processors until 84 CPUs. This generates a supposition that the processors might suffer memory limitations. The Speed Up for all cases scaled up linearly with the number of CPUs. In general it can be said that the scotch partitioner is the good choice for large-scale problems, and a recommended value of cells per CPU can be found around 225,000.

5.1 Future work

Future work will explore new partitioners in the HPC study, for example the metis algorithm which is widely used in commercial softwares and in the industry as well.

Moreover, it would be interesting to extend the current study to transient simulations.

Bibliography

[ADPS83] B. F. Armaly, F. Dursts, J. C. F. Pereira, and B. Schonung. Experimental and theoretical investigation of backward-facing step flow. *Journal of Fluid Mechanics*, 1983.

[AJSM18] Mohamed Sukri Mat Ali, Jafirdaus Jalasabri, Anwar Mohd Sood, and Sallehuddin Muhamad. Wind noise from a-pillar and side view mirror of a realistic generic car model, driveaer. *International Journal of Vehicle Noise and Vibration 14*, 2018.

[ATN12] Heft A, Indinger T, and Adams N. Introduction of a new realistic generic car model for aerodynamic investigations. *SAE Technical paper*, 2012.

[Bha12] Sailaja Bhanduvula. Finite difference method in computational fluid dynamics. *International Journal of Education and applied research*, 2012.

[BP19] Jakub Broniszewski and Janusz Piechna. A fully coupled analysis of unsteady aerodynamics impact on vehicle dynamics during braking. *Engineering Applications of Computational Fluid Mechanics*, 2019.

[BRG13] S. K. Birwa, N. Rathi, and R. Gupta. Aerodynamic analysis of audi a4 sedan using cfd. *Journal of the Institution of Engineers of India*, 2013.

[Cha22] FIA Formula 1 World Championship. Race maximum speeds at formula 1 crypto.com miami grand prix 2022 - miami. `https://www.fia.com/sites/default/files/2022_05_usa_f1_r0_timing_racemaximumspeeds_v01.pdf`, 2022. Accessed: 2022-06-15.

[CKS00] B. Cockburn, G. E. Karniadakis, and C.-W. Shu. Discontinuous galerkin methods. theory, computation and applications. *Lecture Notes in Computational Science and Engineering 11*, 2000.

[Com21] OpenFOAM CFD Community. Openfoam user guide v2112. `https://www.openfoam.com/documentation/user-guide`, 2021. Accessed: 2022-02-13.

[D02] Calhoun D. A cartesian grid method for solving the two-dimensional stream function-vorticity equations in irregular region. *Journal of Computational Physics*, 2002.

[DH03] Jean Donea and Antonio Huerta. Finite element methods for flow problems. http://ww2.lacan.upc.edu/huerta/FEM4FLOW/exercises/Exercises.htm, 2003. Accessed: 2022-01-18.

[DJ03] Russell D. and Wang Z. J. A cartesian grid method for modeling multiple moving objects in 2d incompressible viscous flow. *Journal of Computational Physics*, 2003.

[Ert07] Ercan Erturk. Numerical solutions of 2-d steady incompressible flow over a backward-facing step, part i: High reynolds number solutions. *Computers & Fluids, 37*, 2007.

[Fie95] Mark Alan Fienup. *Scalability study in parallel computing.* Department of Computer Science, Iowa State University, 1995.

[Fon18] Ed Fontes. Fem vs. fvm. https://www.comsol.com/blogs/fem-vs-fvm/, 2018. Accessed: 2022-06-21.

[GC20] Alex Guerrero and Robert Castilla. Aerodynamic study of the wake effects on a formula 1 car. *Energies*, 2020.

[GSH19] M. Giacomini, R. Sevilla, and A. Huerta. Tutorial on hybridizable discontinuous galerkin (hdg) formulation for incompressible flow problems. *CISM International Centre for Mechanical Sciences, Springer International Publishing.*, 2019.

[Hai17] Jibran Haider. An upwind cell centred finite volume method for large strain explicit solid dynamics in openfoam. *PhD thesis at Swansea University*, 2017.

[HIA12] Angelina I. Heft, Thomas Indinger, and Nikolaus A. Adams. Experimental and numerical investigation of the drivaer model. *ASME 2012 - Fluids Engineering Summer Meeting*, 2012.

[ICRcw07] Choi J I, Oberoi R C, Edwards J R, and co workers. An immersed boundary method for complex incompressible flows. *Journal of Computational Physics*, 2007.

[IJ19] CfMesh Irena Juretic. Turbulent flow simulation around drivaer vehicle solved by using cf-mesh+ and openfoam. https://cfmesh.com/turbulent-flow-simulation-around-drivaer-vehicle-solved-by-using-cf-mesh-and
2019. Accessed: 2022-06-10.

[JDH01] Kim J, Kim D, and Choi H. An immersed boundary finite volume method for simulations of flow in complex geometries. *Journal of Computational Physics*, 2001.

[JL72] W. P. Jones and B. E Launder. The prediction of laminarization with a two-equation model of turbulence. *International Journal of Heat and Mass Transfer*, 1972.

[KK11] Dochan Kwak and Cetin C. Kiris. *Computation of Viscous Incompressible Flows*. Springer, 2011.

[Lab15] George Karypis Karypis Lab. Overview - metis: Serial graph partitioning and fill-reducing matrix ordering. `http://glaros.dtc.umn.edu/gkhome/metis/metis/overview`, 2015. Accessed: 2022-06-10.

[LS74] 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, vol. 1*, 1974.

[MCQ16] C. Mingham, D.M. Causon, and Ling Qian. *Numerical Modelling of Wave Energy Converters - Chapter 6: Computational Fluid Dynamics (CFD) Models*. Academic Press, 2016.

[Men93] Florian R. Menter. Zonal two equation k-ω, turbulence models for aerodynamic flows. *24th Fluid Dynamics Conference*, 1993.

[MHB15] Justin A. Morden, Hassan Hemida, and Chris. J. Baker. Comparison of rans and detached eddy simulation results to wind-tunnel data for the surface pressures upon a class 43 high-speed train. *J. Fluids Eng*, 2015.

[MKL03] Florian R. Menter, Martin Kuntz, and Robin Langtry. Ten years of industrial experience with the sst turbulence model. *Turbulence, Heat and Mass Transfer 4*, 2003.

[Mun22] T.U. Munchen. Drivaer model. `https://www.epc.ed.tum.de/en/aer/research-groups/automotive/drivaer/`, 2022. Accessed: 2022-04-23.

[NPC09] N.C. Nguyen, J. Peraire, and B. Cockburn. An implicit high-order hybridizable discontinuous galerkin method for the incompressible navier-stokes equations. *Journal of Computational Physics*, 2009.

[Pop04] Stephen B Pope. Ten questions concerning the large-eddy simulation of turbulent flows. *New Journal of Physics*, 2004.

[PS72] S.V Patankar and D.B Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *Int. J. Heat Mass Transfer, 15*, 1972.

[PW09] Ming Pingjian and Zhang Wenping. Numerical simulation of low reynolds number fluid-structure interaction with immersed boundary method. *Chinese Journal of Aeronautics*, 2009.

[RL21] Arthur Rizzi and James M. Luckring. Historical development and use of cfd for separated flow simulations relevant to military aircraft. *Aerospace Science and Technology*, 2021.

[RS18] Umberto Ravelli and Marco Savini. Aerodynamic simulation of a 2017 f1 car with open-source cfd code. *Journal of Traffic and Transportation Engineering*, 2018.

[Rum21] Christopher Rumsey. Turbulence modelling resource: The menter shear stress transport turbulence model. `https://turbmodels.larc.nasa.gov/sst.html`, 2021. Accessed: 2022-06-11.

[Rum22] Christopher Rumsey. Turbulence modelling resource: The spalart-allmaras turbulence model. `https://turbmodels.larc.nasa.gov/spalart.html`, 2022. Accessed: 2022-06-11.

[SA92] P. R. Spalart and S. R. Allmaras. A one-equation turbulence model for aerodynamic flows. *AIAA Paper*, 1992.

[SB13] R. B. Sharma and Ram Bansal. Cfd simulation for flow over passenger car using tail plates for aerodynamic drag reduction. *IOSR Journal of Mechanical and Civil Engineering*, 2013.

[SGH18] Ruben Sevilla, Matteo Giacomini, and Antonio Huerta. A face-centred finite volume method for second-order elliptic problems. *International Journal for Numerical Methods in Engineering*, 2018.

[Sha16] Dr. Atul Sharma. *Introduction to Computational Fluid Dynamics: Development, Application and Analysis - Chap: Essentials of Numerical-Methods for CFD*. Springer, 2016.

[Sjo16] Bjorn Sjodin. What's the difference between fem, fdm, and fvm? `https://www.machinedesign.com/3d-printing-cad/fea-and-simulation/article/21832072/whats-the-difference-between-fem-fdm-and-fvm`, 2016. Accessed: 2022-05-25.

[TMRS92] T.E. Tezduyar, S. Mittal, S.E. Ray, and R. Shih. Incompressible flow computations with stabilized bilinear and linear equal-order-interpolation velocity-pressure elements. *Computer Methods in Applied Mechanics and Engineering, 95*, 1992.

[WFA+13] Z. J. Wang, Krzysztof Fidkowski, Rémi Abgrall, Francesco Bassi, et al. High-order cfd methods: current status and perspective. *International Journal For Numerical Methods in Fluids*, 2013.

[Wil88] David C. Wilcox. Re-assessment of the scale-determining equation for advanced turbulence models. *AIAA Journal, vol. 26*, 1988.

[Wil06] David C. Wilcox. *Turbulence Modelling for CFD*. DCW Industries, Inc., 3rd edition, 2006.

[Wol] Wolfdynamics. Running in parallel - teaching slides. `http://www.wolfdynamics.com/wiki/parallel.pdf`. Accessed: 2022-03-18.

[ZBFU19] Chunhui Zhang, Charles Patrick Bounds, Lee Foster, and Mesbah Uddin. Turbulence modeling effects on the cfd predictions of flow over a detailed full-scale sedan vehicle. *Fluids*, 2019.