Annex

A Wall Functions

In this part of the Annex, a small introduction to wall functions will be illustrated and his principles.

A.1 What are wall functions

As said during the Thesis, wall functions are empirical equations used to satisfy the physics of the flow in the near wall region. To know in which conditions wall functions approach can be used, it has to be remembered some theory from [1]. The boundary layer is divided in three parts:

1. **The wall layer** (with $y^+ < 5$)
   In this layer the fluid is dominated by the viscous effects. The velocity profile is given by

   $$u^+ = y^+$$

   At the figure **??** it can be seen a graphical representation.

2. **The Overlap layer** ($30 < y^+ < 200$)
   In this layer, the turbulence and viscous shear stress are present and dimensional analysis indicates that the velocity in the overlap layer is proportional to the logarithm of distance, and the velocity profile can be expressed as:

   $$u^+ = 2.5 \ln y^+ + 5.0$$

3. **The Outer layer**
   The turbulent shear dominates. The normalized velocity profile in the core region of turbulent flow in a pipe depends on the distance from the center-line and is independent of the viscosity of the fluid.

There is also a fourth layer called the **Buffer layer**, between the Wall layer and the Overlap layer. Here viscous and turbulent stresses are of similar magnitude.

Some turbulence models such as k-ε are only valid in the area of turbulence fully developed, and do not perform well in the area close to the wall. In order to deal with the near wall region, two ways are usually proposed [2]:

- One way is to **integrate the turbulence to the wall**. Turbulence models are modified to enable the viscosity-affected region to be resolved with all the mesh down to the wall, including the Viscous sublayer. This approach is used when we are interested in the forces of the wall and with high Reynolds numbers (figure 2).

- Another way is to use the so-called **wall functions**, which can model the near wall region (figure 1). Normaly, they are used not that interested in the forces of the wall and with high Reynolds numbers.

Using this wall functions, helps us to reduce computation time since it is not needed a very fine mesh near the wall to capture the viscous effects. So this functions can be applied in the turbulence model, which is actually the describing function of the boundary layer (logarithmic). Nevertheless, to ensure the accuracy of the result, the fist cell center needs to be placed in the log-law regions. If the cell center lies in the Viscous sub-layer, the results from this approach are very inaccurate. Something similar happens in the Buffer layer, since it is complex velocity profile is not well defined and the original wall functions avoid the first cell center located in this region [3]. Here is a visual difference between using wall functions or not:

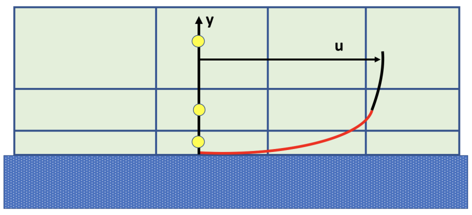

Figure 1: Wall function approach [2]

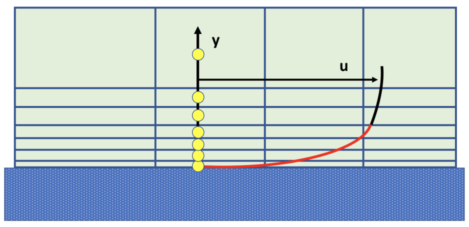

Figure 2: Full resolution approach.

B Solvers and CFL Condition

B.1 CFL Condition

In many books has been explained detailed the physical meaning of this condition ([4]), so here it will be developed a relatively simplified explanation of this condition.

The convergence condition by Courant–Friedrichs–Lewy (CFL) can be interpreted simply as one of the basic rules that should be satisfied for convergence while solving certain partial differential equations numerically. This condition expresses that the distance that any information travels during the time-step length within the mesh must be lower than the distance between mesh cells. The Courant number is:

$$CFL = \frac{u \cdot \Delta t}{\Delta x}$$

Therefore, to achieve numerical stability the velocity times the time step ($u \cdot \Delta t$) must be smaller than the size of the cell (Δx), and, then:

$$CFL \leq 1$$

To lower this value it is possible to refine the mesh or reduce the time-step. Note that the CFL condition is necessary, but not sufficient, condition for the stability of a numerical scheme. The reality is that this is more complex than this and there is an extensive book about this issue [5].Nevertheless, this theory is going to be hugely simplified in order to introduce the learner to it.

The coupled set of governing equations is discretized in time for both steady and transient calculations, so the CFL number is used to compute the time step in both cases. This temporal discretization is accomplished by either implicit or explicit algorithms. From [4]:

"An explicit numerical method is one in which the dependent variables are computed directly via already known values. In this case any discretization operator can be directly evaluated based on the actual variable values. On the other hand, a numerical method is said to be implicit when the dependent variables are treated as unknowns and assembled to form a coupled set of equations which are then solved via special numerical tools using either a direct or an iterative solution algorithm."

In the explicit formulation the time step is computed from the CFL condition. Nevertheless, with implicit formulation is the Courant number can be higher than one. To see why this is true consider an ODE system with backward Euler (or implicit Euler method):

$$\vec{U}^{n+1} = \vec{U}^n + \Delta t A \vec{U}^{n+1} \Rightarrow (I - \Delta t A)\vec{U}^{n+1} = \vec{U}^n$$

Solving for the solution at t^{n+1} requires knowledge of all values at t^n, meaning that no matter what time step it is used we are pulling in the entire physical domain of dependence[6].

SimScale uses implicit schemes, so is possible to run at high Courant number. Though stability does not imply accuracy, that depends on the time step of simulation and the variation over time expected in your flow. This affects specially the transient simulations, with larger time steps it can be missed some transient features or other fluctuations.

B.2 Simscale Solvers

It is necessary to differentiate between the solver and the turbulence model. The models supported nowadays are the Reynolds-averaged Navier-Stokes (RANS) and the Large eddy simulation (LES). The solvers available according to the chosen turbulence model and time-dependency are presented in the tables 1 and 2, for incompressible and compressible flows, respectively.

In order to understand the solvers, it is presented an simple explanation extracted form [7]:

In a **steady state SIMPLE** (Semi-Implicit Method for Pressure-Linked Equations) loop, velocities and pressures are reduced every iteration until all parameters reached values small enough for convergence. The reduction of these values per iteration may or may not be a very smaller number. As long as the overall reduction meets convergence requirement, the simulation is over.

In a **time dependent SIMPLE loop**, all the parameters (velocities and pressure etc) are required to reduce to values small enough for convergence in each time step. It is easier to view transient SIMPLE as a lot of steady state SIMPLE loops over multiple time steps. Now, because within each time step, there are multiple steady state SIMPLE loops, it is called *'iterative time advancement'*.

The difference between **PISO** (Pressure Implicit with Splitting of Operator) and SIMPLE in terms of time advancement is that PISO does not require an iterative process, and is therefore called *'non-iterative time advancement'*. Here by iterative I mean multiple PISO loops each time step (just like transient SIMPLE). However, for a solution to converge when solving with PISO, it would be needed to do iterations within the PISO loop itself. Note that there is a difference between iterations within each PISO loop, and in between PISO loops. For PISO to yield promising results, it is generally required to go through two pressure corrector loops (each of these two loops will then go through multiple iterations to reduce residuals) and one loop for each velocity component with multiple iterations.

Finally, the **PIMPLE algorithm** is a combination of PISO and SIMPLE, and is very similar to transient SIMPLE with PISO replacing steady state SIMPLE. The PISO loops are repeated until the convergence requirement is reached. All these algorithms are iterative solvers but PISO and PIMPLE are both used for transient cases whereas SIMPLE is used for steady-state cases.

In conclusion, transient SIMPLE is a lot of steady-state SIMPLE loops per time step; PIMPLE is a lot of PISO loops per time step. For transient SIMPLE, solutions converge when the initial residuals at the start of each steady-state SIMPLE loop fall below pre-defined values; for PIMPLE to converge, the initial residuals at the start of each PISO loop will fall below pre-defined values.

Turbulence Mode	Time dependency	Solver	OpenFOAM solver
Laminar	Transient	PIMPLE	pimpleFoam
		PISO	pisoFoam
		ICO	icoFoam
	Steady-state	SIMPLE	simpleFoam
		ICO	icoFoam
RANS	Transient	PIMPLE	pimpleFoam
		PISO	pisoFoam
	Steady-state	SIMPLE	simpleFoam
LES	Transient	PIMPLE	pimpleFoam
	Transient	PISO	pisoFoam

Table 1: SimScale's available solvers for incompressible flow. [8]

Turbulence Mode	**Time dependency**	**Solver**	**OpenFOAM solver**
Laminar	Transient	pressure-based	rhoPimpleFoam
	Transient	density-based	rhoCentralFoam[1]
	Steady-state	-	rhoSimpleFoam
RANS	Transient	pressure-based	rhoPimpleFoam
	Steady-state	-	rhoSimpleFoam
LES	Transient	pressure-based	rhoPimpleFoam

Table 2: SimScale's available solvers for compressible flow. [9]

References

Books

1. ÇENGEL YUNUS A.; CIMBALA JOHN M. *Fluid Mechanics: Fundamental applications*. ISBN 0–07–247236–7.
4. MOUKALLED, F; MANGANI, L; DARWISH, M. *The Finite Volume Method in Computational Fluid Dynamics. An Advanced Introduction with OpenFOAM and Matlab.* 2016. ISBN 978-3-319-16873-9. Available from DOI: `10.1007/978-3-319-16874-6`.
5. MOURA, Carlos A. de.; KUBRUSLY, Carlos S. *The Courant-Friedrichs-Lewy (CFL) condition : 80 years after its discovery*. ISBN 9780817683931.

Reports

3. FANGQING LIU. *A Thorough Description Of How Wall Functions Are Implemented In OpenFOAM*. 2017. Available also from: `http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2016`. Technical report. Chalmers university of techology.

Websites

2. *What is y+ (yplus)? - SimWiki - SimScale CAE Forum*. Available also from: `https://www.simscale.com/forum/t/what-is-y-yplus/82394`.
6. *pde - Understanding the Courant–Friedrichs–Lewy condition - Computational Science Stack Exchange*. Available also from: `https://scicomp.stackexchange.com/questions/25398/understanding-the-courant-friedrichs-lewy-condition`.
7. *PIMPLE pressure residual - Using SimScale / Fluid Flow / CFD - SimScale CAE Forum*. Available also from: `https://www.simscale.com/forum/t/pimple-pressure-residual/443/5`.
8. *Incompressible Fluid Flow Analysis I SimScale Documentation*. Available also from: `https://www.simscale.com/docs/analysis-types/incompressible-fluid-flow-analysis/`.
9. *Compressible Fluid Flow Analysis I SimScale Documentation*. Available also from: `https://www.simscale.com/docs/analysis-types/compressible-fluid-flow-analysis/`.

This references are also included in the report. They have been written in the Annex to help the reader to see the sources. Some numbers may not be the same.