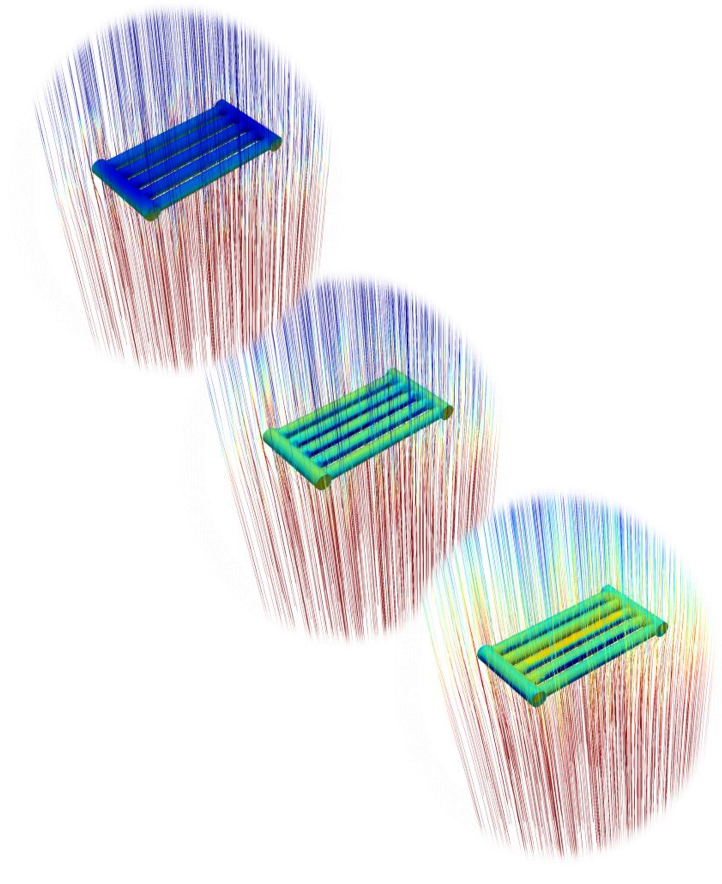
University of
South-Eastern Norway

www.usn.no

FMH606 Master's Thesis 2021

Process Technology

Hydrodynamics – calculation of added mass on complex objects



Sondre Kaasa

Faculty of Technology, Natural sciences and Maritime Sciences
Campus Porsgrunn

University of South-Eastern Norway

www.usn.no

Course: FMH606 Master's Thesis, 2021

Title: Hydrodynamics – calculation of added mass on complex objects

Number of pages: 139

Keywords: CFD, hydrodynamics, marine technology, added mass, drag force, numerical simulation.

Student: Sondre Kaasa

Supervisor: Prof. Knut Vågsæther

External partner: Stressman Engineering AS, Sondre Luca Helgesen

Summary:

Added mass is a complex hydrodynamic effect that is poorly understood in many fields of engineering. Thus, the motivation for the present work was to gain knowledge in hydrodynamics and numerical modeling of hydrodynamic effects. The present work began with a literature study on hydrodynamics and the most conventional estimation methods.

The k-ω SST model is used in the CFD analyses performed. This model behaves considerably better than the standard k-ε model in adverse pressure gradient flows, which is a significant drawback.

Two geometries are analyzed – the submerged cylinder analysis is to verify the methodology, as added mass can be easily compared against DNV-RP-C205. The result for the submerged cylinder verifies the methodology with an error of less than 3 % compared to DNV-RP-C205. The second geometry, the cylindrical frame, is based on the conclusions drawn from the submerged cylinder analysis. There is no easy method of comparing the results regarding the cylindrical frame, as the analyses show a considerable flow interaction between the cylinders. DNV-RP-C205 thus falls short of providing correct estimates for the complex geometry investigated.

Preface

I am presenting my master’s thesis about the calculation of added mass on complex objects using CFD. The work presented in this report is the result of work performed during the spring semester of 2021. I have gained considerable knowledge about hydrodynamics and CFD analyses compared to before starting the present work.

Firstly, I would like to thank my supervisor Prof. Knut Vågsæther for sharing valuable knowledge and excellent supervision along the way. I would also like to thank Mr. Sondre Luca Helgesen, CEO of Stressman Engineering AS, for providing me with such an exciting and challenging project and provide me with the resources necessary to complete my thesis.

Porsgrunn, 18.05.21

Sondre Kaasa

Contents

Nomenclature

Symbol	Description, unit
μ	Friction factor, dynamic viscosity, [-], [Pas]
A	Projected area normal to flow, coefficient matrix [m^2], [-]
A_{33}	Heave added mass for a perforated plate, [kg]
$A_{33,s}$	Heave added mass for a solid plate, [kg]
A_{33}^{2D}	Heave added mass in 2D, [kg]
A_{33}^{3D}	Heave added mass in 3D, [kg]
$ANSYS$	Analysis systems, [-]
A_R	Reference area, [m^2]
b	Source vector, [-]
C_μ	Model constant equal to 0.09, [-]
$C_A(v)$	Coriolis added mass matrix, [-]
C_A^{2D}	2D added mass coefficient, [-]
C_A^{3D}	3D added mass coefficient, [-]
C_d	Drag coefficient, [-]
$C_d(Re)$	Drag coefficient as a function of Reynolds number, [-]
$CD_{k\omega}$	Positive part of the cross-diffusion term in Equation (3.8), [kg/(m^{3}s^3)]
CFD	Computational fluid dynamics, [-]
C_M	Added mass and inertia coefficient, [-]
Co	Courant number, [-]
CPU	Central Processing Unit, [-]
D_{33}^{2D}	Heave damping coefficient in 2D, [-]
D_{33}^{3D}	Heave damping coefficient in 3D, [-]
$DILU$	Simplified diagonal-based incomplete LU preconditioner, [-]
DNS	Direct numerical simulations, [-]
DNV	Det Norske Veritas, [-]
DOF	Degree-of-freedom, [-]
$D_P(v)$	Potential damping matrix, [kg/s, kgm/s, kgm^2/s]
f	Total hydrodynamic force, [N]
$f(U)$	Drag force as a function of velocity, [N]

F_1, F_2	Blending functions, [-]
FEA	Finite Element Analysis, [-]
F_P	Pressure force, [N]
$g(\eta)$	Restoring forces, [N]
$GAMG$	Geometric-algebraic multigrid solver, [-]
h	Width of vortex street, [m]
I	Turbulence intensity, [%]
$\bar{I}_x, \bar{I}_y, \bar{I}_z$	Inertia around X-, Y-, and Z- axes, [kgm]
$\bar{I}_{xy}, \bar{I}_{xz}, \bar{I}_{yz}$	Product of inertia, [kgm^2]
k	Turbulent kinetic energy, [m^2/s^2]
K	Moment about X-axis, [Nm]
KC	Keulegan-Carpenter number, [-]
$K_{\dot{p}}, m_{44}$	Added mass component about X-axis due to acceleration about X-axis, [kgm^2]
$K_{\dot{q}}, m_{45}$	Added mass component about X-axis due to acceleration about Y-axis, [kgm^2]
$K_r(A, b)$	Krylov subspace vector, [-]
$K_{\dot{r}}, m_{46}$	Added mass component about X-axis due to acceleration about Z-axis, [kgm^2]
$K_{\dot{u}}, m_{41}$	Added mass component about X-axis due to acceleration in X-direction, [kgm]
$K_{\dot{v}}, m_{42}$	Added mass component about X-axis due to acceleration in Y-direction, [kgm]
$K_{\dot{w}}, m_{43}$	Added mass component about X-axis due to acceleration in Z-direction, [kgm]
l	Distance between adjacent vortices, [m]
L	Characteristic length, [m]
LES	Large-eddy simulation, [-]
$\overline{m}$	Mass of displaced water, [kg]
M	Moment about Y-axis, [Nm]
M_A	Added mass, [kg]
M_{FK}	Froude-Kriloff mass matrix, [kg, kgm, kgm^2]
$M_{\dot{p}}, m_{54}$	Added mass component about Y-axis due to acceleration about X-axis, [kgm^2]
$M_{\dot{q}}, m_{55}$	Added mass component about Y-axis due to acceleration about Y-axis, [kgm^2]
$M_{\dot{r}}, m_{56}$	Added mass component about Y-axis due to acceleration about Z-axis, [kgm^2]
$M_{\dot{u}}, m_{51}$	Added mass component about Y-axis due to acceleration in X-direction, [kgm]
$M_{\dot{v}}, m_{52}$	Added mass component about Y-axis due to acceleration in Y-direction, [kgm]
$M_{\dot{w}}, m_{53}$	Added mass component about Y-axis due to acceleration in Z-direction, [kgm]

N	Moment about Z-axis, [Nm]
$N_{\dot{p}}, m_{64}$	Added mass component about Z-axis due to acceleration about X-axis, [kgm^2]
N_P, N_V	Constant matrices, [-]
$N_{\dot{q}}, m_{65}$	Added mass component about Z-axis due to acceleration about Y-axis, [kgm^2]
$N_{\dot{r}}, m_{66}$	Added mass component about Z-axis due to acceleration about Z-axis, [kgm^2]
$N_{\dot{u}}, m_{61}$	Added mass component about Z-axis due to acceleration in X-direction, [kgm]
$N_{\dot{v}}, m_{62}$	Added mass component about Z-axis due to acceleration in Y-direction, [kgm]
$N_{\dot{w}}, m_{63}$	Added mass component about Z-axis due to acceleration in Z-direction, [kgm]
$OpenFOAM$	Open Field Operation And Manipulation, [-]
p	Rotational velocity around X-axis, [rad/s], perforation ratio, [%]
p^*	Intermediate pressure value in PISO, [m^2/s^2]
p^{**}	Corrected pressure in PISO, [m^2/s^2]
p'	Value of pressure correction in PISO, [m^2/s^2]
$PBiCG$	Preconditioned bi-conjugate gradient, [-]
$PISO$	Pressure-implicit with splitting of operators, [-]
P_k	Production of turbulence, [kg/(ms^3)]
$\tilde{P}_k$	Turbulence production limiter in stagnation points, [kg/(ms^3)]]
PMM	Planar motion mechanism, [-]
q	Rotational velocity around Y-axis, [rad/s]
r	Rotational velocity around Z-axis, [rad/s]
RAM	Random Access Memory, [-]
$RANS$	Reynolds-averaged Navier-Stokes, [-]
Re	Reynolds number, [-]
ROV	Remotely operated vehicle, [-]
RP	Recommended Practice, [-]
S	Surface (wetted), [m^2]
S_f	Surface normal vector, [-]
$SNAME$	Society of Naval Architects and Marine Engineers, [-]
SST	Shear stress transport, [-]
T	Oscillation period, [s]
T_A	Added mass kinetic energy, [J]
u	Freestream velocity, [m/s]

u^*	Intermediate velocity value in PISO, [m/s]
u^{**}	Corrected velocity in PISO, [m/s]
u, v, w	Linear velocity components in X-, Y-, and Z-direction, [m/s]
$\bar{U}$	Mean flow velocity, [m/s]
U'	Fluctuating velocity component, [m/s]
u'	Value of velocity correction in PISO, [m/s]
u^+	Dimensionless velocity, [-]
U_m	Velocity amplitude, [m/s]
$URANS$	Unsteady Reynolds-averaged Navier-Stokes, [-]
U_{ref}	Reference velocity, [m/s]
u_w	Near-wall velocity, [m/s]
u_τ	Friction velocity, [m/s]
v	Velocity vector, [m/s], [rad/s]
V_R	Reference volume, [m^3]
X	Force in X-direction, [N]
$X_{\dot{p}}, m_{14}$	Added mass component in X-direction due to acceleration about X-axis, [kgm]
$X_{\dot{q}}, m_{15}$	Added mass component in X-direction due to acceleration about Y-axis, [kgm]
$X_{\dot{r}}, m_{16}$	Added mass component in X-direction due to acceleration about Z-axis, [kgm]
$X_{\dot{u}}, m_{11}$	Added mass component in X-direction due to acceleration in X-direction, [kg]
$X_{\dot{v}}, m_{12}$	Added mass component in X-direction due to acceleration in Y-direction, [kg]
$X_{\dot{w}}, m_{13}$	Added mass component in X-direction due to acceleration in Z-direction, [kg]
y	Distance from the wall to the node, [m]
Y	Force in Y-direction, [N]
y^+	Dimensionless distance from the wall to the first node, [-]
$Y_{\dot{p}}, m_{24}$	Added mass component in Y-direction due to acceleration about X-axis, [kgm]
$Y_{\dot{q}}, m_{25}$	Added mass component in Y-direction due to acceleration about Y-axis, [kgm]
$Y_{\dot{r}}, m_{26}$	Added mass component in Y-direction due to acceleration about Z-axis, [kgm]
$Y_{\dot{u}}, m_{21}$	Added mass component in Y-direction due to acceleration in X-direction, [kg]
$Y_{\dot{v}}, m_{22}$	Added mass component in Y-direction due to acceleration in Y-direction, [kg]
$Y_{\dot{w}}, m_{23}$	Added mass component in Y-direction due to acceleration in Z-direction, [kg]
Z	Force in Z-direction, [N]
$Z_{\dot{p}}, m_{34}$	Added mass component in Z-direction due to acceleration about X-axis, [kgm]

$Z_{\dot{q}}, m_{35}$	Added mass component in Z-direction due to acceleration about Y-axis, [kgm]
$Z_{\dot{r}}, m_{36}$	Added mass component in Z-direction due to acceleration about Z-axis, [kgm]
$Z_{\dot{u}}, m_{31}$	Added mass component in Z-direction due to acceleration in X-direction, [kg]
$Z_{\dot{v}}, m_{32}$	Added mass component in Z-direction due to acceleration in Y-direction, [kg]
$Z_{\dot{w}}, m_{33}$	Added mass component in Z-direction due to acceleration in Z-direction, [kg]
α_1	Model constant for the k-ω SST turbulence model, [-]
α_2	Model constant for the k-ω SST turbulence model, [-]
β^*	Model constant for the k-ω SST turbulence model, [-]
β_1	Model constant for the k-ω SST turbulence model, [-]
β_2	Model constant for the k-ω SST turbulence model, [-]
Γ	Diffusion coefficient, [-]
Δt	Time-step, [s]
Δx	Grid spacing, [m]
δx	Distance between neighboring nodes, [m]
ε	Dissipation of turbulent kinetic energy, [m^2/s^3]
ν	Kinematic viscosity, [m^2/s]
ν_t	Turbulent eddy-viscosity, [m^2/s]
ρ	Fluid density, [kg/m^3]
σ_{k1}	Model constant for the k-ω SST turbulence model, [-]
σ_{k2}	Model constant for the k-ω SST turbulence model, [-]
$\sigma_{\omega 1}$	Model constant for the k-ω SST turbulence model, [-]
$\sigma_{\omega 2}$	Model constant for the k-ω SST turbulence model, [-]
$\tau_{current}$	Current forces and moments, [N], [Nm]
τ_R	Radiation forces and moments, [N], [Nm]
τ_w	Wall shear stress, [Pa]
τ_{xy}	Shear stress in the XY-plane, [Pa]
φ	General property, [-]
φ_e	Value of property at east interface, [-]
φ_P	Value of property at node P, [-]
φ_W	Value of property at node W, [-]
$\nabla\phi$	Gradient of property, [-]
ω	Specific dissipation rate, [1/s]

1 Introduction

This section of the report gives an overview of the importance of investigating hydrodynamic loads on subsea constructions. The signed task description for this thesis is in Appendix [A]. Also, a WBS and Gantt diagram is created to generate a sensible workflow throughout writing this thesis. WBS and Gantt diagrams can be found in Appendix [B] and [C], respectively.

There is a high degree of uncertainty in the offshore and subsea industries regarding the added mass of structures. DNVGL and other authorities provide estimation methods through their recommended practices and standards. These methods are, however, limited to simple geometrical shapes. In some situations, this results in over-dimensioning of the structures. This is because the systems become heavier and more costly than necessary since more steel needs to be used to achieve code compliance in accordance with applicable design codes. Also, since the structures become heavier, larger crane ships are required for the offshore transportation of such modules.

There are many opinions amongst engineers, but few are grounded in experimental procedures or numerical analyses. Hopefully, this master's thesis will shed more light on the added mass of more complex shapes and geometries.

During offshore lifting operations, the module to be lifted is carried by a crane ship. These tasks come with multiple challenges, such as uncertainty of weather conditions, placement of the center of mass, and the total hydrodynamic loads acting on the module during the lifting operation. An illustration of offshore lifting is shown in Figure 1.1.

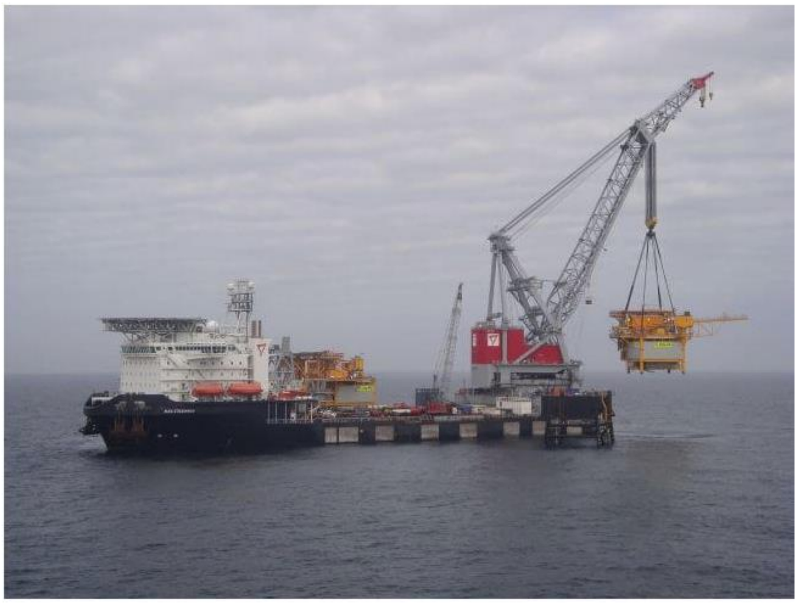


Figure 1.1: Typical offshore lifting operation [1]

Due to the complexity of subsea modules, accurate estimations of added mass are not easily obtained. In lifting operations, added mass is important concerning the size of the ship required and when considering crane tip motions. If the wave period coincides with the crane tip motion period, resonance can occur, causing unstable operation of the crane ship. This highlights some of the importance of quantifying added mass on subsea modules. [1]

Chapter 2 summarizes the literature study on hydrodynamics.

Chapter 3 gives an insight into computational fluid dynamics with a focus on turbulence modeling techniques.

Chapter 4 describes the chosen solver and appropriate solver settings.

Chapter 5 gives the general setup procedure for the analyses.

Chapter 6 describes the numerical setup procedure for the geometries investigated. One simple 3D geometry is investigated, and a more complex 3D geometry, based on the simple analysis. The 3D geometry illustrates a typical part of a protection structure in subsea applications.

Chapter 7 provides the results from the CFD analyses.

Chapter 8 discusses the findings from the present work.

Chapter 9 is the conclusion with recommendations for further work.

2 Theory

The purpose of this section of the report is to give an insight into the field of hydrodynamics. Added mass formulations and simplifications are presented, and methods of determining the total hydrodynamic force on an object based on empirical, experimental, and theoretical procedures.

2.1 Definition of terms

In marine engineering, it is common practice to use the SNAME notation for the motion of crafts and objects modeled in a six degrees of freedom (6 DOF) system. The SNAME notation describes linear and angular motions with the corresponding forces, moments, and positions. The SNAME notation is described in Table 2.1. These directions are illustrated in Figure 2.1 concerning the orientation of the body.

Table 2.1: SNAME notation [2]

DOF	**Motion**	**Forces and moments**	**Velocities**	**Positions and Euler angles**
1	x-direction (surge)	X	u	x
2	y-direction (sway)	Y	v	y
3	z-direction (heave)	Z	w	z
4	Rotation about the x-axis (roll)	K	p	ϕ
5	Rotation about the y-axis (pitch)	M	q	Θ
6	Rotation about the z-axis (yaw)	N	r	ψ

The body-fixed frame in Figure 2.1 follows the COG of the moving object and describes the linear and angular velocities, while the inertial frame is used to describe the relative position of the object. [2]

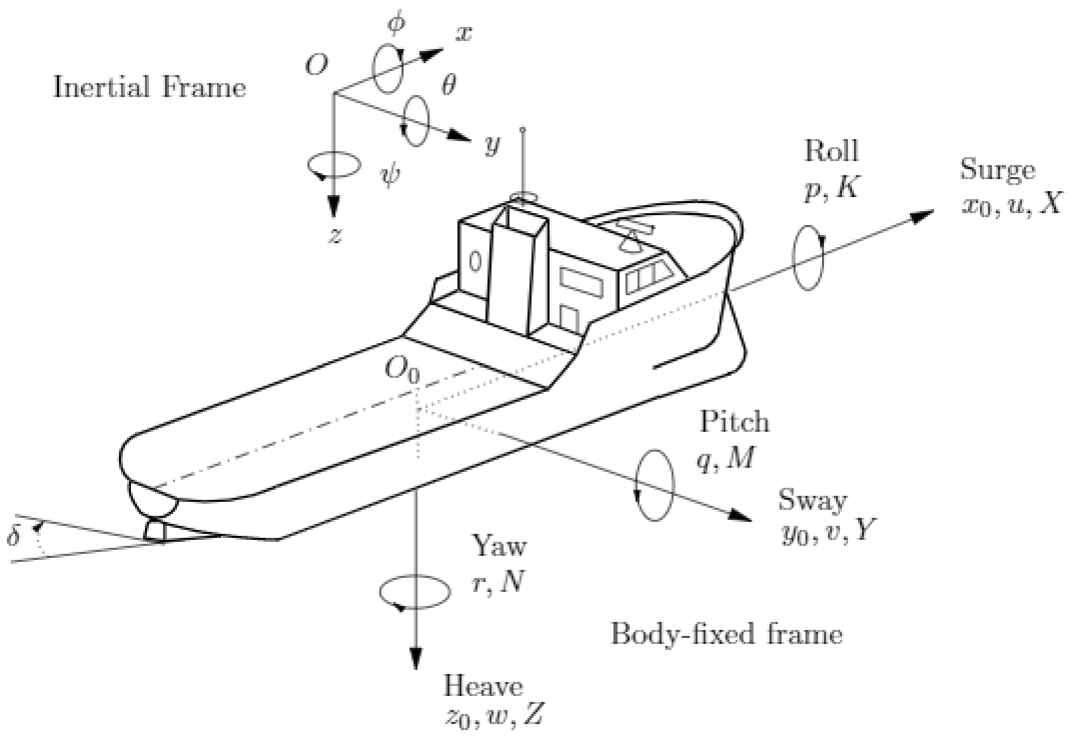


Figure 2.1: Direction of motion on a fixed body frame [3]

2.2 Added mass and kinetic energy

Added mass or virtual mass can be regarded as additional inertia experienced by a body subjected to acceleration through a fluid. Added mass are pressure loads caused by forced body-motion, which are proportional to the acceleration. The equations presented in this chapter assume rigid-body dynamics, i.e., the mechanical flexibility of the object is not considered. [2]

As an object, i.e., ROV or subsea structure, moves through a stationary fluid, the fluid is continuously displaced before closing in behind the object in motion. Thus, the fluid exerts kinetic energy that would not be present if the body were stationary. The kinetic energy is expressed in vector form as in Equation (2.1). According to Newton's 2nd law, the total force acting on a body is $F = m \cdot a$. In a steady, rectilinear motion, the kinetic energy is constant, the acceleration and the force due to acceleration is zero. [4]

$$T_A = \frac{1}{2} v^T M_A v \tag{2.1}$$

Where T_A is the kinetic energy, v is the velocity vector, and M_A is the added mass matrix. The added mass matrix is defined for a 6 DOF system in Equation (2.2). [2]

$$\boldsymbol{M}_A = \begin{bmatrix} X_{\dot{u}} & X_{\dot{v}} & X_{\dot{w}} & X_{\dot{p}} & X_{\dot{q}} & X_{\dot{r}} \\ Y_{\dot{u}} & Y_{\dot{v}} & Y_{\dot{w}} & Y_{\dot{p}} & Y_{\dot{q}} & Y_{\dot{r}} \\ Z_{\dot{u}} & Z_{\dot{v}} & Z_{\dot{w}} & Z_{\dot{p}} & Z_{\dot{q}} & Z_{\dot{r}} \\ K_{\dot{u}} & K_{\dot{v}} & K_{\dot{w}} & K_{\dot{p}} & K_{\dot{q}} & K_{\dot{r}} \\ M_{\dot{u}} & M_{\dot{v}} & M_{\dot{w}} & M_{\dot{p}} & M_{\dot{q}} & M_{\dot{r}} \\ N_{\dot{u}} & N_{\dot{v}} & N_{\dot{w}} & N_{\dot{p}} & N_{\dot{q}} & N_{\dot{r}} \end{bmatrix} \tag{2.2}$$

The added mass matrix can also be described as m_{ij} where the force is in the i direction due to acceleration in the j direction, this notation is described for a 6 DOF system in Equation (2.3).

$$\boldsymbol{M}_A = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} & m_{15} & m_{16} \\ m_{21} & m_{22} & m_{23} & m_{24} & m_{25} & m_{26} \\ m_{31} & m_{32} & m_{33} & m_{34} & m_{35} & m_{36} \\ m_{41} & m_{42} & m_{43} & m_{44} & m_{45} & m_{46} \\ m_{51} & m_{52} & m_{53} & m_{54} & m_{55} & m_{56} \\ m_{61} & m_{62} & m_{63} & m_{64} & m_{65} & m_{66} \end{bmatrix} \tag{2.3}$$

Thus, the external forces and moments due to added mass are described in Equation (2.4).

$$\begin{Bmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \\ \boldsymbol{Z} \\ \boldsymbol{K} \\ \boldsymbol{M} \\ \boldsymbol{N} \end{Bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} & m_{15} & m_{16} \\ m_{21} & m_{22} & m_{23} & m_{24} & m_{25} & m_{26} \\ m_{31} & m_{32} & m_{33} & m_{34} & m_{35} & m_{36} \\ m_{41} & m_{42} & m_{43} & m_{44} & m_{45} & m_{46} \\ m_{51} & m_{52} & m_{53} & m_{54} & m_{55} & m_{56} \\ m_{61} & m_{62} & m_{63} & m_{64} & m_{65} & m_{66} \end{bmatrix} \cdot \begin{Bmatrix} \dot{\boldsymbol{u}} \\ \dot{\boldsymbol{v}} \\ \dot{\boldsymbol{w}} \\ \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{r}} \end{Bmatrix} \tag{2.4}$$

As previously mentioned, the added mass matrix represents loads imposed on an object due to acceleration in any of the defined directions. The diagonal elements of the added mass matrix ($i = j$) define the primary elements, where the added mass in one direction is caused by an acceleration in that same direction. The off-diagonal elements are the coupled elements. Added mass of the 36 elements depends entirely on the geometry to be analyzed, fluid density, etc. However, the diagonal elements are generally dominating, and it is often assumed that the coupled elements are negligible. [2], [5]

2.3 Reduction of the added mass matrix

Evaluating all 36 components of the matrix can be a time-consuming and complicated process; hence, simplifying the added mass matrix is helpful in practical problems.

2.3.1 Reduction by symmetry

Symmetry conditions can be applied for many subsea applications, whether the object is an underwater vehicle or a protection structure. Symmetry conditions for subsea structures can vary a lot for structures of similar applications. It can be shown that applying symmetry in each of the primary planes, i.e., XY-, XZ-, and YZ-plane, can be described by Equation (2.5) – Equation (2.7), respectively. [2]

$$M_{A,XY} = \begin{bmatrix} m_{11} & m_{12} & 0 & 0 & 0 & m_{16} \\ m_{21} & m_{22} & 0 & 0 & 0 & m_{26} \\ 0 & 0 & m_{33} & m_{34} & m_{35} & 0 \\ 0 & 0 & m_{43} & m_{44} & m_{45} & 0 \\ 0 & 0 & m_{53} & m_{54} & m_{55} & 0 \\ m_{61} & m_{62} & 0 & 0 & 0 & m_{66} \end{bmatrix} \tag{2.5}$$

$$M_{A,XZ} = \begin{bmatrix} m_{11} & 0 & m_{13} & 0 & m_{15} & m_{16} \\ 0 & m_{22} & 0 & m_{24} & 0 & m_{26} \\ m_{31} & 0 & m_{33} & 0 & m_{35} & 0 \\ 0 & m_{42} & 0 & m_{44} & 0 & m_{46} \\ m_{51} & 0 & m_{53} & 0 & m_{55} & 0 \\ 0 & m_{62} & 0 & m_{64} & 0 & m_{66} \end{bmatrix} \tag{2.6}$$

$$M_{A,YZ} = \begin{bmatrix} m_{11} & 0 & 0 & 0 & m_{15} & 0 \\ 0 & m_{22} & 0 & m_{24} & 0 & 0 \\ 0 & 0 & m_{33} & 0 & 0 & 0 \\ 0 & m_{42} & 0 & m_{44} & 0 & 0 \\ m_{51} & 0 & 0 & 0 & m_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & m_{66} \end{bmatrix} \tag{2.7}$$

For underwater vehicles, symmetry in two planes can often be assumed. Symmetry in only one plane can cause poor maneuvering capabilities for such vehicles. For symmetry in the XZ and YZ planes, the added mass matrix is reduced to that described in Equation (2.8). [6]

$$\boldsymbol{M}_A = \begin{bmatrix} m_{11} & 0 & 0 & 0 & m_{15} & m_{16} \\ 0 & m_{22} & 0 & m_{24} & 0 & 0 \\ 0 & 0 & m_{33} & 0 & 0 & 0 \\ 0 & m_{42} & 0 & m_{44} & 0 & 0 \\ m_{51} & 0 & 0 & 0 & m_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & m_{66} \end{bmatrix} \tag{2.8}$$

If symmetry in all three primary planes can be applied, only the diagonal components of the matrix remain.

2.3.2 Reduction by slender body theory

Slender body theory/strip theory applies to objects where the length is much greater than any of the other two dimensions. An illustration of an arbitrary slender body is shown in Figure 2.2. The basic idea of slender body theory is that the three-dimensional added mass coefficient can be estimated as the sum of two-dimensional coefficients along the object's length. See Equation (2.9) for the mathematical formulation. For slender body theory to be applicable, the flow variation in the cross-sectional planes must be much greater than the flow variation in the longitudinal direction. [7]

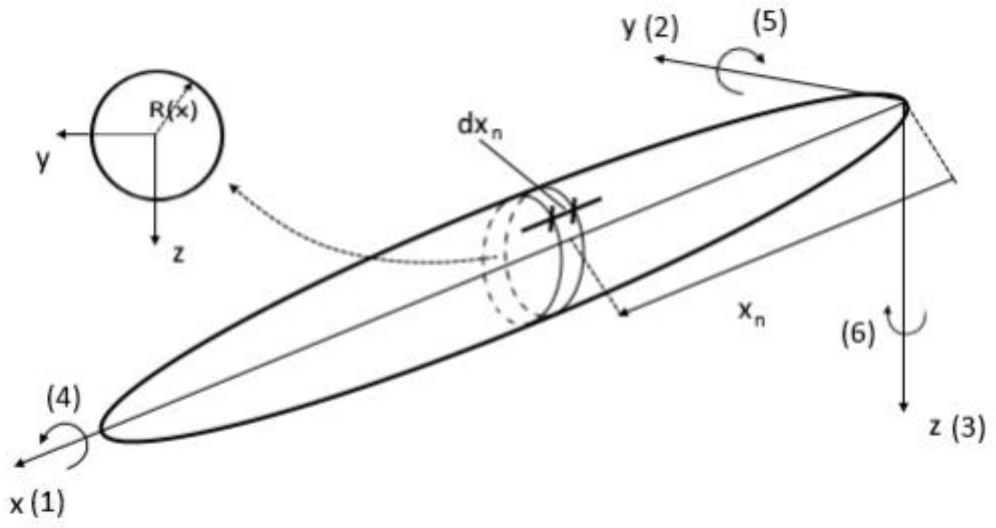


Figure 2.2: Principal sketch of the slender body theory application [7]

$$C_A^{3D} = \int_0^L C_A^{2D} dx \tag{2.9}$$

2.4 Hydrodynamic damping

Frictional and drag forces are commonly termed hydrodynamic damping forces. These forces consist of two components – a linear term and a quadratic term. The main contributions to hydrodynamic damping forces occur from skin friction and damping due to vortex shedding. [5], [8]

2.4.1 Skin friction damping

Skin friction is a boundary layer problem caused by a velocity gradient on the surface of a body. The velocity gradient generates shear stresses, which in turn creates damping forces. Shear stress is described in Equation (2.10). [9]

$$\tau_{xy} = \mu \left[\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right] \tag{2.10}$$

Where μ is the friction factor, $\frac{\partial u}{\partial y}$ is the velocity gradient of the horizontal component in the vertical direction, and $\frac{\partial v}{\partial x}$ is the velocity gradient of the vertical component in the horizontal direction. The boundary layer is shown in Figure 2.3.

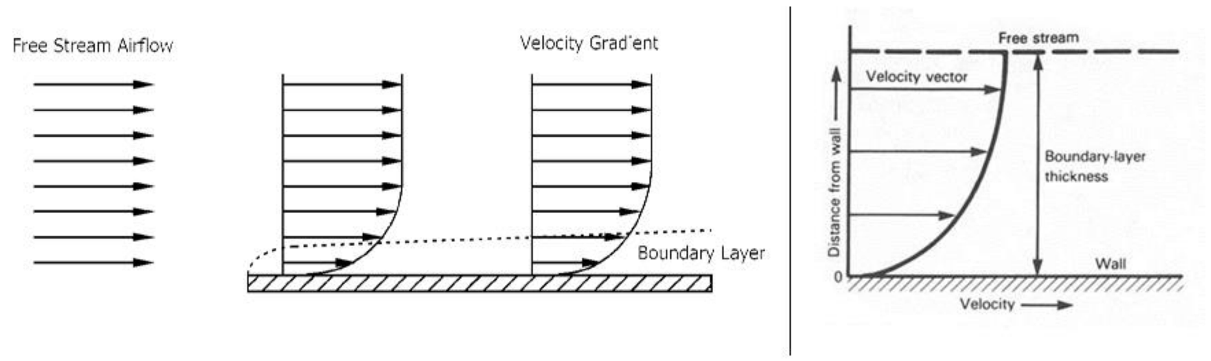


Figure 2.3: Velocity gradient and boundary layer thickness on a flat plate [10]

2.4.2 Vortex shedding

Vortex shedding occurs when the flow separates after flowing across an object. Von Karman studied the stability of these vortex shedding streets on two-dimensional geometries. These eddies are formed in rows, where only two arrangements are possible – the eddies are either placed directly opposite of those in the next row or a symmetrically staggered configuration. The former is unstable, while the latter becomes stable for the ratio described by Equation (2.11). [11]

$$\frac{h}{l} = \frac{1}{\pi}\cosh^{-1}\left(\sqrt{2}\right) \approx 0.28 \tag{2.11}$$

Where h is the vortex street width and l is the distance between two adjacent vortices in the same row.

Viscous damping occur due to vortex shedding on a completely submerged body in a fluid moving at constant velocity. The viscous damping forces are modeled by Equation (2.12). [5]

$$f(U) = -\frac{1}{2}\rho C_d(Re)A|U|U \tag{2.12}$$

Where $C_d(Re)$ is the viscous drag coefficient, A is the projected area, and U is the body velocity.

2.5 Radiation forces

Radiation forces are defined as forces occurring when an object oscillates with the wave excitation frequency. Three components contribute to radiation forces and moments: added mass, radiation-induced damping, and restoring forces. The sum of these components defines the total radiation load on a body and is defined in Equation (2.13). [2]

$$\tau_R = -\boldsymbol{M}_A v - \boldsymbol{C}_A(\boldsymbol{v})\boldsymbol{v} - \boldsymbol{D}_P(\boldsymbol{v})\boldsymbol{v} - \boldsymbol{g}(\boldsymbol{\eta}) \tag{2.13}$$

Where $\boldsymbol{C}_A(\boldsymbol{v})\boldsymbol{v}$ is the Coriolis and centripetal added mass matrix, $\boldsymbol{D}_P$ is the potential damping, and $\boldsymbol{g}(\boldsymbol{\eta})$ are restoring forces due to weight and buoyancy.

For a completely submerged body far away from a free surface, radiation forces can typically be neglected. This assumption is used throughout the present work.

2.6 Froude-Kriloff and diffraction forces

Suppose a structure or a marine vehicle is restrained from moving. In that case, two hydrodynamic effects are present due to the unsteady pressure caused by the presence of an object – Froude-Kriloff and diffraction forces. The former are hydrodynamic loads imposed on the body due to the undisturbed pressure field, while diffraction forces and moments are caused by the changed pressure field due to the presence of the body. These forces and moments are described in Equation (2.14). [12]

$$\boldsymbol{\tau}_{current} = \boldsymbol{M}_{FK}\dot{\boldsymbol{v}}_c + \boldsymbol{M}_A\boldsymbol{v}_c + \boldsymbol{N}_P\boldsymbol{v}_c + \boldsymbol{N}_V\boldsymbol{v}_c \tag{2.14}$$

Where $\boldsymbol{M}_{FK}$ is the Froude-Kriloff inertia matrix, $\dot{\boldsymbol{v}}_c$ is the time-derivative of the velocity vector, $\boldsymbol{v}_c$ is the velocity vector, $\boldsymbol{N}_P$ and $\boldsymbol{N}_V$ are constant matrices given by the partial derivative of radiation damping and viscous damping, respectively.

$\boldsymbol{M}_{FK}$ is defined as the inertia of the fluid displaced by the submerged object. It can be shown that the inertia forces and moments can be described by Equation (2.15). [12]

$$\boldsymbol{M}_{FK} = \begin{bmatrix} \bar{m} & 0 & 0 & 0 & \bar{m}z_B & -\bar{m}y_B \\ 0 & \bar{m} & 0 & -\bar{m}z_B & 0 & \bar{m}x_B \\ 0 & 0 & \bar{m} & \bar{m}y_B & -\bar{m}x_B & 0 \\ 0 & -\bar{m}z_B & \bar{m}y_B & \bar{I}_x & -\bar{I}_{xy} & -\bar{I}_{xz} \\ \bar{m}z_B & 0 & -\bar{m}x_B & -\bar{I}_{xy} & \bar{I}_y & -\bar{I}_{yz} \\ -\bar{m}y_B & \bar{m}x_B & 0 & -\bar{I}_{xz} & -\bar{I}_{yz} & \bar{I}_z \end{bmatrix} \tag{2.15}$$

Where $\bar{m}$ is the mass of the displaced fluid, x_B, y_B and z_B center of buoyancy coordinates and $\bar{I}_j$ and $\bar{I}_{kj}$ are inertia and products of inertia, respectively.

For the computations performed throughout the present work, diffraction forces are neglected since an infinite fluid is assumed, i.e., interaction with surface waves is negligible.

2.7 Morison Equation

The Morison equation was first proposed by Morison et al. [13]. The purpose of the equation is that the total hydrodynamic force on a cylinder can be expressed as the sum of added mass and drag forces. This relation is expressed in Equation (2.16).

$$f = \rho C_M V \dot{u} + \frac{1}{2} \rho A C_d |u| u; \quad C_M = 1 + C_A \tag{2.16}$$

Where f is the total hydrodynamic force, C_M and C_d are the mass and drag coefficients, respectively, V is the volume displaced by the object, and u is the velocity. The drag coefficient can be represented with the same DOFs as the added mass coefficient.

It is stated by Newman [14] that the validity of the Morison equation is limited to the cases where the ratio A/L is respectively small or large and should be verified with experimental procedures. For submerged bodies in engineering applications, the equation appears to give satisfactory results. However, experimental procedures with appropriate values of the Reynolds number should be performed to validate theoretically determined quantities.

2.8 Estimation methods for added mass

There are numerous different methods available for determining the hydrodynamic parameters on marine and subsea objects. The hydrodynamic coefficients can be determined by, e.g., simplified empirical estimates, experimental procedures with force transducers, or numerical estimation methods.

2.8.1 Empirical estimates

As previously described, slender body theory is applicable when the length of the structure is large compared to the other dimensions. The hydrodynamic coefficients are considered in a 2D plane parallel to the cross-section along the length of the structure. For the theory to be valid, the flow variation in the cross-sectional plane must be large compared to flow variation in the longitudinal direction. [6], [8]

The hydrodynamic coefficients are found by integrating the 2D coefficients in the structure's longitudinal direction. These are described for added mass and damping by Equation (2.17) and Equation (2.18), respectively.[6], [8]

$$A_{33}^{3D} = \int_L A_{33}^{2D}(x) dx \tag{2.17}$$

$$D_{33}^{3D} = \int_L D_{33}^{2D}(x) dx \tag{2.18}$$

Where A_{33}^{2D} and D_{33}^{2D} are the two-dimensional added mass and damping coefficients, respectively, L is the length of the object.

2.8.2 Experimental procedures

Free decay or pendulum tests are classified by connecting the object, typically a scaled-down version of an ROV or part of a subsea structure, to a string connected to a pivot point. The ROV or subsea structure is then released from a starting position and oscillates freely for some time until the motion stops. This is a simple 1 DOF system described by the angular position. The hydrodynamic parameters are calculated from the time history of the motion. A free-decay pendulum test setup is shown in Figure 2.4. [15]

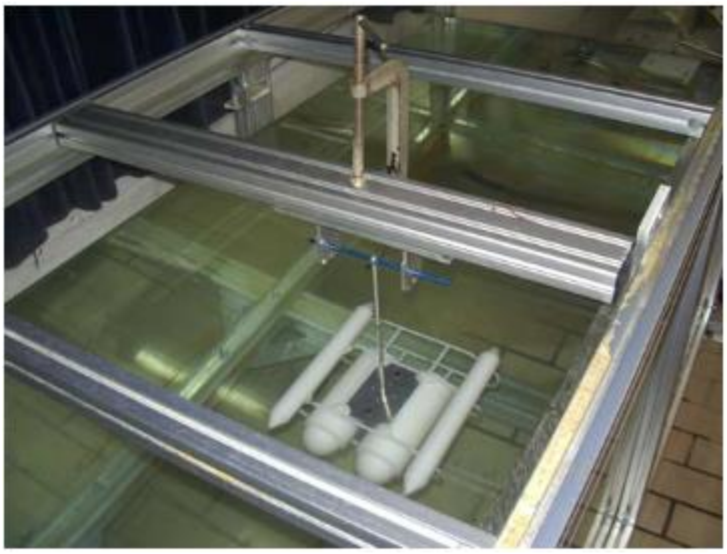


Figure 2.4: Setup of free-decay pendulum test for a scaled ROV model [15]

Tow tank tests are another experimental procedure commonly used to determine added mass. The test rigs are typically quite large and expensive to build due to the amount of space and equipment needed for a full-scale model experiment. The towing tests can be configured to apply a PMM, where the object (ship, ROV, subsea structure, etc.) is forced to oscillate harmonically while being towed along the length of the tank. Typical tow tank test configurations are illustrated in Figure 2.5 and Figure 2.6. [15], [16]

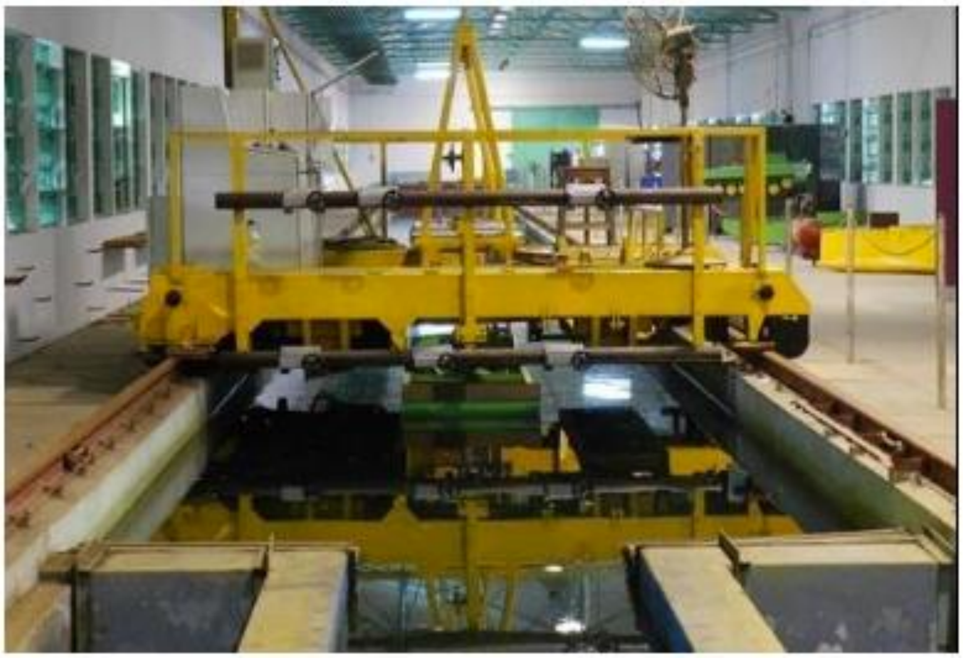


Figure 2.5: Illustration of a tow tank with test carriage [16]

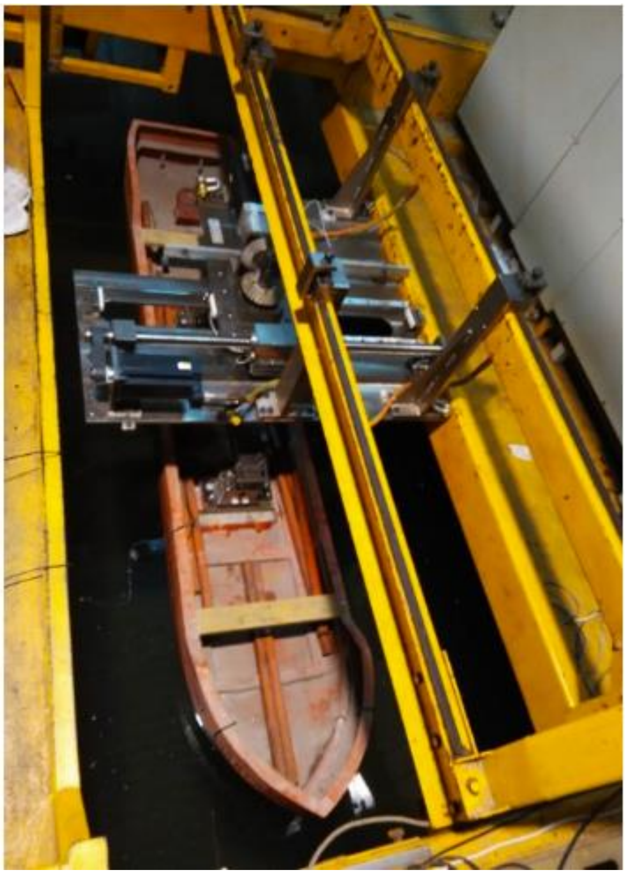


Figure 2.6: Tow tank test setup for a ship [16]

2.8.3 Numerical estimations

Recently there has been an increasing interest in determining the hydrodynamic parameters using numerical methods, such as CFD. This is likely related to the advanced developments of available computational resources.

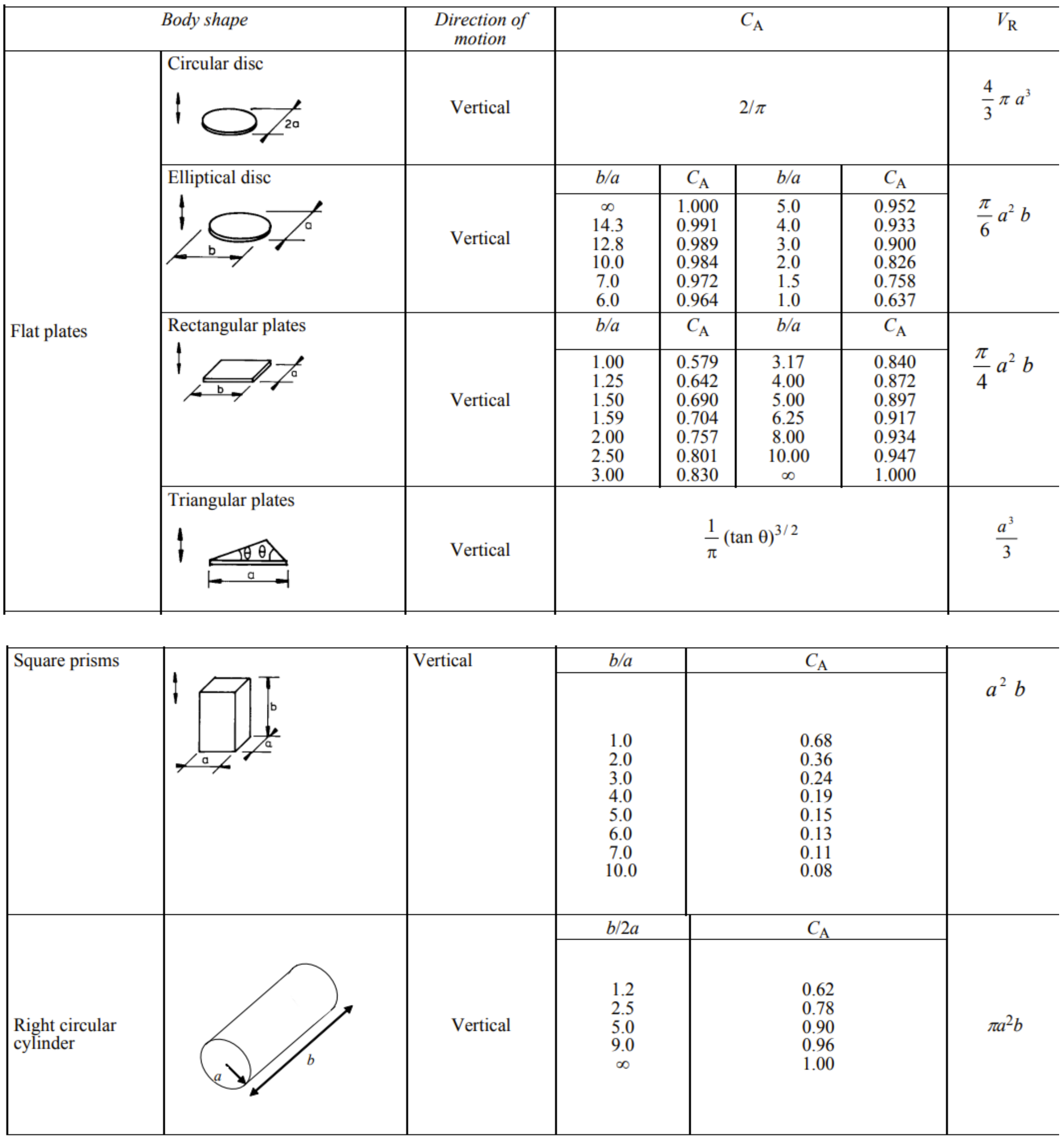
Noteworthy is that there is little research being conducted for rectilinear acceleration. Most research papers, to the author's knowledge, are being conducted for oscillating conditions.

Since the analyses are numerically solved, it is important to validate the methods chosen. The authors of [17] use experimental procedures to verify the numerical model. The CFD analysis is conducted using Reynolds-averaged Navier-Stokes (RANS) numerical method with the $k-\omega\ SST$ turbulence model for square cylinders subjected to oscillating flow conditions. Based on the results, the hydrodynamic coefficients are generally compliant with the experimental procedures.

2.8.4 DNV-RP-C205

The recommended practice developed by DNV provides some guidelines for hydrodynamic computation methods in different flow conditions, e.g., rectilinear acceleration or oscillating conditions identified using KC numbers.

Analytical values of added mass are given for simple objects in appendix D of the RP. Some of them are summarized in Figure 2.7. The arrows indicate the direction of motion, C_A is the added mass coefficient, and V_R is the reference volume. For complex constructions, i.e., subsea protection structures, the RP does not provide any estimation procedures. The RP does not provide acceptance criteria for added mass but refers to relevant design codes for structural evaluation.



Body shape		*Direction of motion*	C_A				V_R
Flat plates	Circular disc	Vertical	$2/\pi$				$\frac{4}{3}\pi a^3$
	Elliptical disc	Vertical	*b/a*	C_A	*b/a*	C_A	$\frac{\pi}{6} a^2 b$
			∞ 14.3 12.8 10.0 7.0 6.0	1.000 0.991 0.989 0.984 0.972 0.964	5.0 4.0 3.0 2.0 1.5 1.0	0.952 0.933 0.900 0.826 0.758 0.637	
	Rectangular plates	Vertical	*b/a*	C_A	*b/a*	C_A	$\frac{\pi}{4} a^2 b$
			1.00 1.25 1.50 1.59 2.00 2.50 3.00	0.579 0.642 0.690 0.704 0.757 0.801 0.830	3.17 4.00 5.00 6.25 8.00 10.00 ∞	0.840 0.872 0.897 0.917 0.934 0.947 1.000	
	Triangular plates	Vertical	$\frac{1}{\pi}(\tan\theta)^{3/2}$				$\frac{a^3}{3}$

Square prisms		Vertical	*b/a*	C_A	$a^2 b$
			1.0 2.0 3.0 4.0 5.0 6.0 7.0 10.0	0.68 0.36 0.24 0.19 0.15 0.13 0.11 0.08	
Right circular cylinder		Vertical	*b/2a*	C_A	$\pi a^2 b$
			1.2 2.5 5.0 9.0 ∞	0.62 0.78 0.90 0.96 1.00	

Figure 2.7: Added mass coefficients for some simple geometries provided by DNV-RP-C205 [18]

It is stated in the RP that if numerical estimation procedures are employed, these results must be validated with experimental data.

2.9 Fluid-structure interactions

The previously described equations are based on rigid-body dynamics; hence, it is assumed that the structure subjected to movement retains its initial shape. During the motion period, forces are exerted on the structure by the fluid, thus generating stresses and deforming the body. FSI aims at predicting a coupling of fluid flow behavior and mechanical response of the moving structure.

In general, there are two ways of approaching an FSI problem – the monolithic or the partitioned system coupling approach. When using the monolithic approach, the structural and fluid systems are solved as a single mathematical system. An ill-conditioned matrix can occur from this method due to differences in stiffnesses. The degree of ill-conditioning is related to the magnitude of the condition number, which depends on how singular the system matrix is. [19], [20]

The partitioned system coupling approach solves one iteration of the computational fluid domain. It transfers the forces onto the mechanical model, which is then used in the next iteration of the fluid flow computation. [21]

The partitioned method solves the fluid domain and structural model in an alternating manner – one iteration is performed within the fluid domain. The results are mapped onto the mechanical model used during the next iteration of the fluid domain computation. System coupling is achieved either through a one-way or two-way coupling configuration. One-way system coupling is used for weak FSI, while if the structure is subject to large deflections, a two-way system coupling is recommended. [21]

In a one-way coupling approach, the fluid domain is solved until convergence is achieved. The forces at the interface are extracted and used in the structural computation. The structural solver (typically FEA) is solved until convergence is achieved. This process is repeated for the specified coupled system timeframe. A schematic of one-way coupling is given in Figure 2.8. [21]

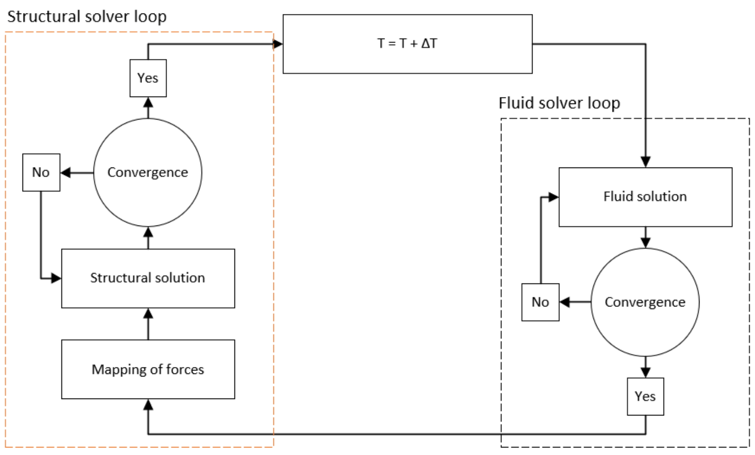


Figure 2.8: Schematic of a one-way system coupling procedure

The two-way system coupling is far more time-consuming than one-way system coupling. Firstly, the fluid domain is solved for one time-step until convergence. The forces at the interface are transferred to the structural model. The forces exerted on the object generate structural displacements. The deformation of the structural component is applied to the fluid domain before solving the fluid model until convergence once more. This process is repeated until the difference in forces calculated between iterations falls below a convergence criterion. A schematic of the two-way system coupling is given in Figure 2.9. [21]

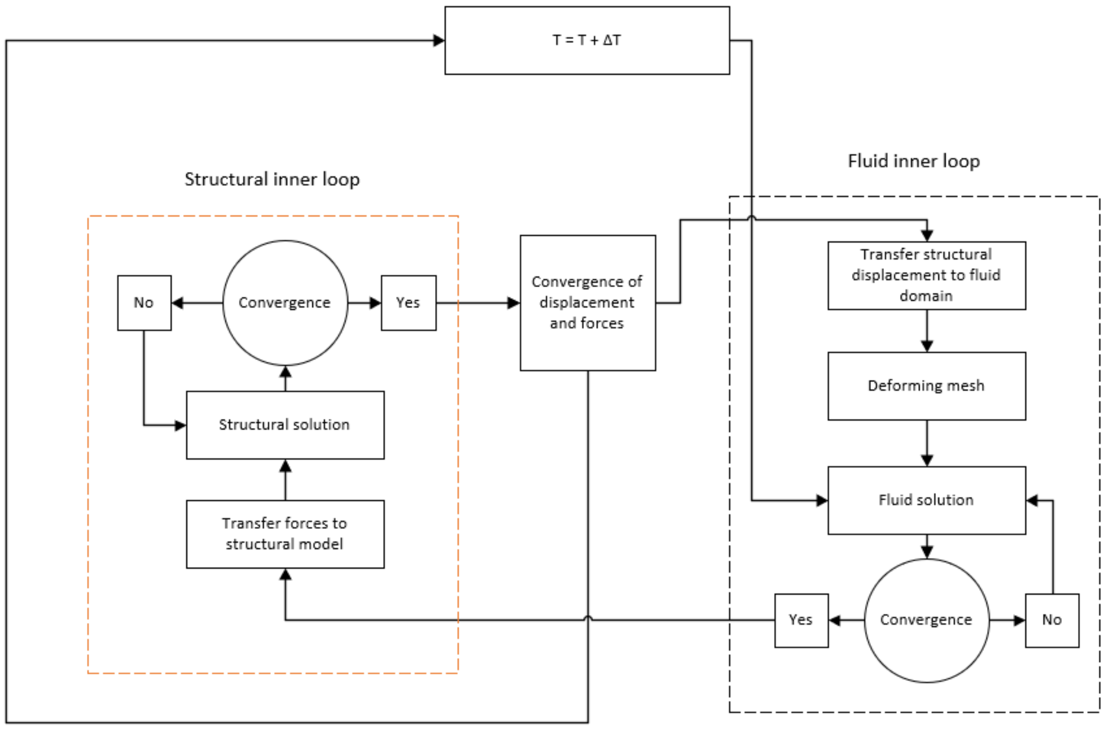


Figure 2.9: Schematic of a two-way system coupling

2.10Oscillating object motion

Several subsea installations are subject to the occurrence of earthquakes, which generate oscillating movement of the structure. Oscillating movement is represented by a dimensionless quantity, called the Keulegan-Carpenter number, defined in Equation (2.12). [22]

$$KC = \frac{U_m T}{D} \tag{2.15}$$

Where U_m is the flow velocity amplitude, T is the period, and D is the characteristic dimension of the object.

The two force components added mass, and drag, are effects of acceleration and velocity, respectively, which for oscillating movement, means that the two forces are 90^o out of phase. In a paper presented by Keulegan and Carpenter [22], it is stated that added mass and drag coefficients can be modeled following Equation (2.13) and (2.14), respectively.

$$C_m = \frac{2}{\pi^2} \cdot \frac{U_m T}{D} \int_0^{2\pi} \frac{F sin(\theta) d\theta}{\rho U_m^2 D} \tag{2.16}$$

$$C_d = -\frac{3}{4} \int_0^{2\pi} \frac{F cos(\theta) d\theta}{\rho U_m^2 D} \tag{2.17}$$

The experiments were conducted on cylinders and flat plates. It was found that the added mass and drag coefficients have opposite trends – the added mass coefficient decreases from its initial value to a minimum value at $KC = 15$, while the drag coefficient increases and reaches a maximum value at this KC number. For flat plates, the most noticeable of the results was the trend of the drag coefficient. The drag coefficient is unusually high for low KC numbers, while it decays to a value resembling the steady-state drag coefficient for higher KC numbers. [22]

3 CFD and turbulence

Computational fluid dynamics solves the characteristic equations of the fluid on a discretized fluid domain. The characteristic equations are typically conservation and transport equations, i.e., conservation of energy/mass/momentum and transport of turbulent flow properties. The discretization is achieved by dividing the domain into blocks, called a mesh. Each block consists of one node in the cell center where the desired quantities are computed. These quantities are then approximated to the cell faces using discretization schemes. Hence, the characteristic equations are solved at discrete points. Sufficiently refining the mesh is highly important to get accurate representation of the quantities computed. In some instances, i.e., for the near-wall treatment of turbulent flows, additional mesh refinement is required.

Numerical errors from analyses are expected. The numerical errors can be substantially reduced by critically evaluating the resolution of the mesh and the benefits and downsides of numerical schemes. A general CFD analysis procedure can be broken down into the steps shown in Figure 3.1.

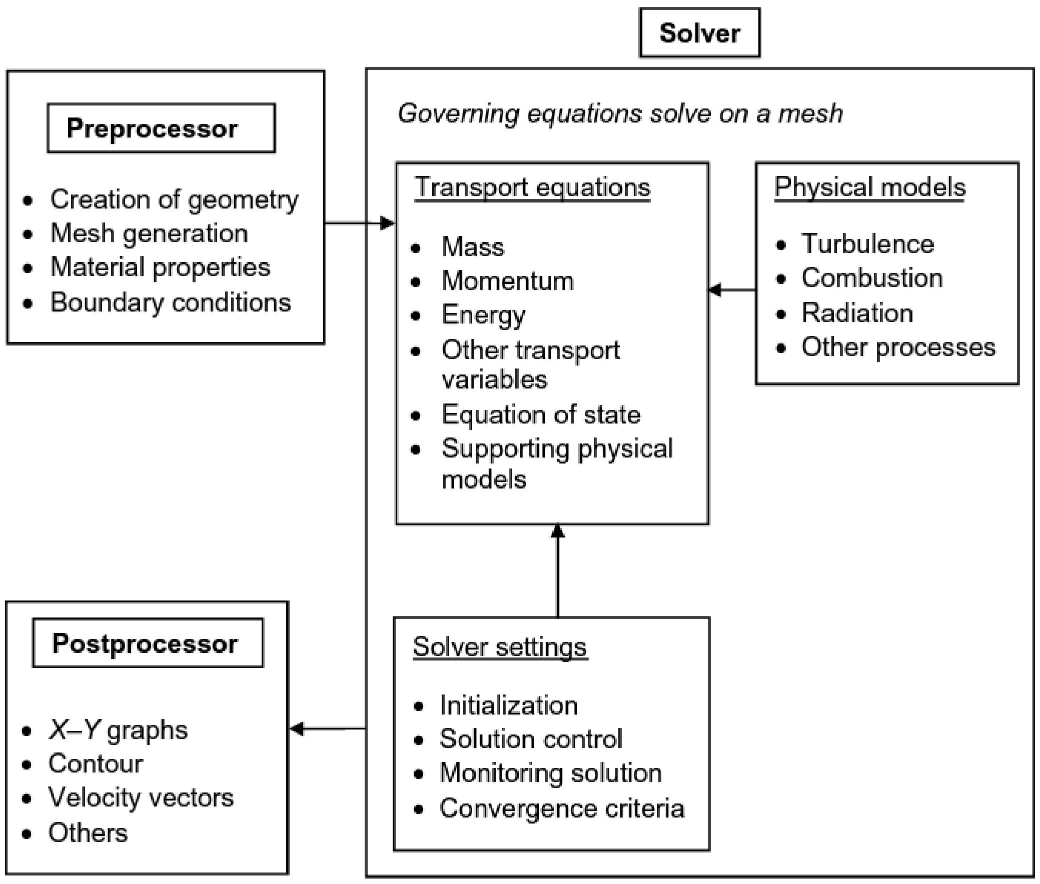


Figure 3.1: General CFD workflow [23]

3.1 Overview of turbulence

The flow regime is characterized by the Reynolds number – a dimensionless number relating inertia forces and viscous forces. At low Reynolds numbers, the flow is laminar, and the fluid moves from a point A to a point B in smooth adjacent layers. At high Reynolds numbers, the flow becomes turbulent and presents numerous complex effects. This flow regime is characterized by the random and chaotic motion of the fluid. Typical streamlines for laminar and turbulent flows are illustrated in Figure 3.2. [24]

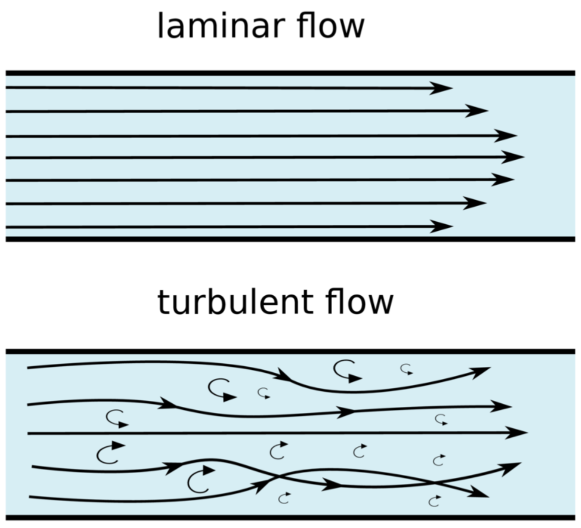


Figure 3.2: Laminar and turbulent streamlines in a pipe [25]

Turbulent eddies represent turbulent flows' rotational behavior, which enhances the fluid mixing, resulting in more effective transport of heat, mass, and momentum. The enhanced momentum transfer effect occurs due to convective transport of the eddies in an acceleration-deceleration process – the faster-moving fluid is decelerated by accelerating slower-moving fluid and vice versa. [24]

Turbulent eddies can further be described by two subcategories – large eddies and small eddies. The large eddies are of the same length and velocity as the mean flow and are classified as inviscid, meaning inertia effects dominate over viscous effects. Large eddies are transported through vortex stretching, during which energy is extracted from the mean flow. The stretching process is caused by the mean/bulk flow, causing one end of the eddies to move faster than the other. As the stretching process continues, the larger eddies produce smaller eddies. This is a repeated process, where viscous effects eventually become important for the smallest eddies. Hence, large eddies are products of mean flow characteristics, while small eddies are products of large eddies. [26]

3.2 Turbulence modeling

The three main methods of analyzing turbulent flows are DNS, LES, and RANS/URANS. A brief description of the former two will be provided; however, RANS/URANS description is focused, as this will be used for the remainder of the thesis.

3.2.1 Direct numerical simulations

DNS calculates all fluid motions in the flow field by evaluating the governing equations directly, without approximations and averaging techniques, except for those required for the numerical discretization of the fluid domain. For all flow properties to be accurately represented by this method, the calculations need to be performed on a sufficiently fine grid to capture the behavior of the turbulent eddies. [24]

3.2.2 Large-eddy simulations

An alternative to DNS is to evaluate turbulent flows as distinct transport of large-scale and small-scale motions. This method is called large-eddy simulations (LES), where the large-scale motions are modeled exactly while calculating approximations for the small-scale motions. LES simulations are sensible in that the large-scale motions are more effective for transporting the conserved properties. It is much less expensive than DNS but still requires a lot of computational resources to capture the large eddies. [24]

3.2.3 Reynolds-averaged Navier-Stokes equations

The third method, namely the URANS method, considers turbulent flow properties on the mean flow rather than eddy-motions. URANS models employ Reynolds decomposition, where the velocity is decomposed into an average and a statistically fluctuating component. The velocity at some point in a turbulent flow might have the structure illustrated in Figure 3.3. [24]

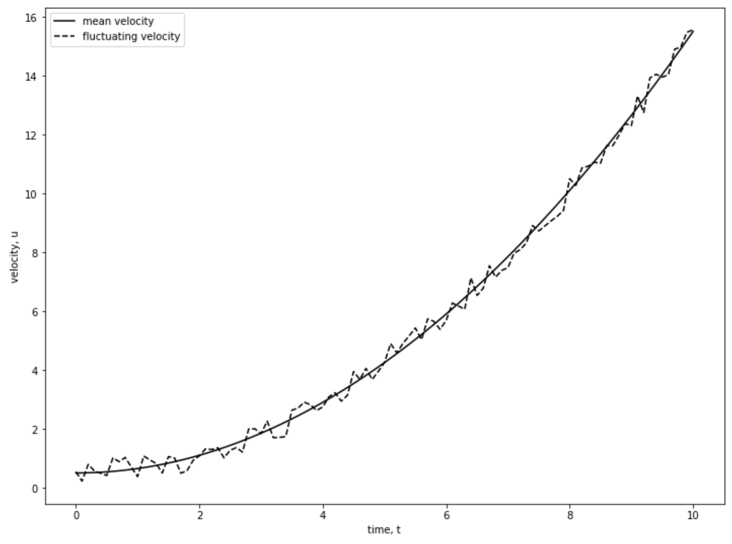


Figure 3.3: Instantaneous point-velocity of an unsteady flow

The Reynolds decomposition is described in Equation (3.1). For simplicity, the remaining part of this section is based on one-dimensional assumptions. The description of two- and three-dimensional RANS follow the same procedure as for the one-dimensional descriptions. [24]

$$u(x,t) = \bar{u}(x) + u'(x,t) \tag{3.1}$$

Where $u(x,t)$ is the instantaneous point-velocity, $\bar{u}(x)$ is the mean velocity, and $u'(x,t)$ is the fluctuating statistical component.

Since added mass is a concept of acceleration (unsteady motion), URANS will be considered. It is worth mentioning that the variables are spatial functions and functions of time, as described in Equation (3.2). [24]

$$\bar{u} = \bar{u}(x,y,z,t) \tag{3.2}$$

The URANS equations for an incompressible fluid are described in Equation (3.3) and Equation (3.4) for mass and momentum transport. [24], [27]

$$div\,\boldsymbol{U} = 0 \tag{3.3}$$

$$\frac{\partial U}{\partial t} + div(U\mathbf{U}) = -\frac{1}{\rho}\frac{\partial P}{\partial x} + v div\big(grad(U)\big) + \frac{1}{\rho}\left[\frac{\partial\left(-\rho\overline{u'^2}\right)}{\partial x} + \frac{\partial(-\rho\overline{u'v'})}{\partial y} + \frac{\partial\left(-\rho\overline{u'w'}\right)}{\partial z}\right] \tag{3.4}$$

The fluctuating components in the last term of Equation (3.4) represent the Reynolds stresses, denoted τ_{ij}. Reynolds stresses consist of nine stress components – three normal stresses ($i = j$) and six shear stresses ($i \neq j$), giving rise to nine independent stress components. However, velocity fluctuations can be interchanged between directions i and j. The stress-tensor described in Equation (3.5) is reduced to six independent components. [24], [27]

$$\tau_{ij} = \rho\overline{u_i'u_j'} = \begin{bmatrix} \bar{\tau}_{11} & \bar{\tau}_{12} & \bar{\tau}_{13} \\ \bar{\tau}_{21} & \bar{\tau}_{22} & \bar{\tau}_{23} \\ \bar{\tau}_{31} & \bar{\tau}_{32} & \bar{\tau}_{33} \end{bmatrix} = \begin{bmatrix} \rho\overline{u_1'^2} & \rho\overline{u_1'u_2'} & \rho\overline{u_1'u_3'} \\ \rho\overline{u_2'u_1'} & \rho\overline{u_2'^2} & \rho\overline{u_2'u_3'} \\ \rho\overline{u_3'u_1'} & \rho\overline{u_3'u_2'} & \rho\overline{u_3'^2} \end{bmatrix} \tag{3.5}$$

A visual illustration of the direction of the stress components is shown in Figure 3.4.

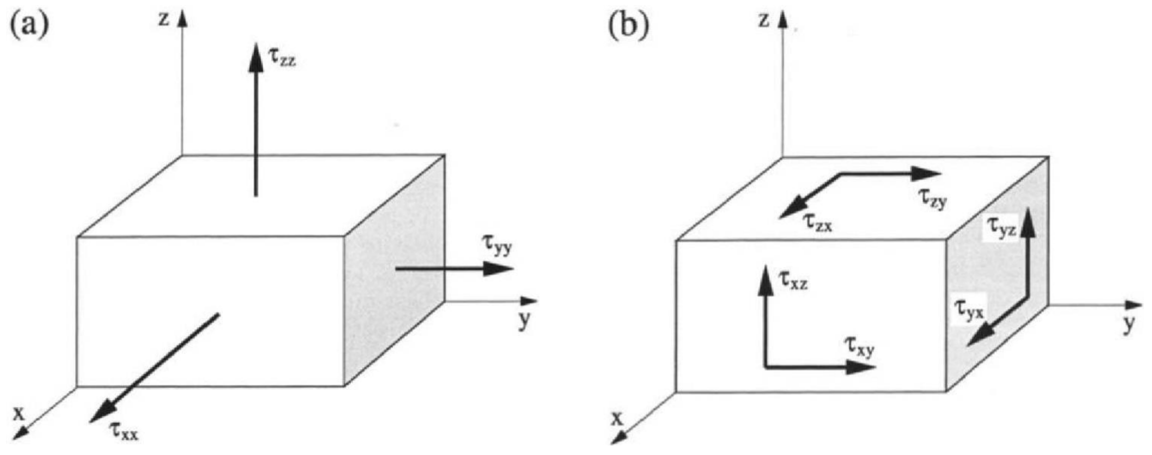


Figure 3.4: Direction of Reynolds stresses on a control volume [27]

3.3 The k-ω SST turbulence model

"*The starting point for the development of the SST model was the need for the accurate prediction of aeronautics flows with strong adverse pressure gradients and separation. Over decades, the available turbulence models had consistently failed to compute these flows."* [28] Since the development of the SST model, it has been shown that it applies to a wide range of numerical flow problems.

The purpose of the $k-\omega$ SST model is to implement the best features of the standard $k-\omega$ and $k-\varepsilon$ models. The former model can more accurately predict shear stresses within the boundary layer compared to the $k-\varepsilon$ model, though it is generally quite sensitive to the freestream value of the specific dissipation rate, ω_f. A major weakness of the $k-\varepsilon$ model is the overestimation of shear stresses in adverse pressure gradient conditions. [27]

Hence, the SST model uses the $k-\omega$ in the near-wall regions and switches to $k-\varepsilon$ outside the boundary layer for increased stability. The transport equation for turbulence kinetic energy is defined by Equation (3.6). [28]

$$\rho\left[\frac{\partial k}{\partial t}+\frac{\partial U_i k}{\partial x_i}\right]=\tilde{P}_k-\beta^*\rho k\omega+\frac{\partial}{\partial x_i}\left[(\mu+\sigma_k\mu_t)\frac{\partial k}{\partial x_i}\right] \tag{3.6}$$

Where $\frac{\partial k}{\partial t}$ is the rate of change of turbulent kinetic energy, $\frac{\partial U_i k}{\partial x_i}$ is the convective transport of k, $\beta^*\rho k\omega$ is the rate of dissipation of k and $\frac{\partial}{\partial x_i}\left[(\mu+\sigma_k\mu_t)\frac{\partial k}{\partial x_i}\right]$ is the diffusive transport of k. $\tilde{P}_k$ is a production limiter used to prevent turbulence from building up in stagnation regions, defined by Equation (3.7).

$$P_k=\mu_t\frac{\partial U_i}{\partial x_j}\left(\frac{\partial U_i}{\partial x_j}+\frac{\partial U_j}{\partial x_i}\right)\rightarrow\tilde{P}_k=\min(P_k,10\cdot\beta^*k\omega) \tag{3.7}$$

The equation for dissipation of turbulence kinetic energy, ε, is transformed into an ω equation. The transformation is done by substituting $\varepsilon = k\omega$, which yields Equation (3.8). [24]

$$\frac{\partial \omega}{\partial t} + \frac{\partial U_i \omega}{\partial x_i} = \alpha S^2 - \beta \omega^2 + \frac{\partial}{\partial x_i}\left[(\nu + \sigma_\omega \nu_t)\frac{\partial \omega}{\partial x_i}\right] + 2(1 - F_1)\rho\sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i} \tag{3.8}$$

S is the invariant measure of strain rate, and F_1 is a blending function defined by Equation (3.9). F_1 is one inside the boundary layer and goes to zero far away from the wall. [24], [28]

$$F_1 = \tanh\left\{\left\{min\left[max\left(\frac{\sqrt{k}}{\beta^* \omega y}, \frac{500\nu}{y^2 \omega}\right), \frac{4\rho\sigma_{\omega 2} k}{CD_{k\omega} y^2}\right]\right\}^4\right\} \tag{3.9}$$

Where y is the distance from the cell center to the surface and $CD_{k\omega}$ is the positive portion of the cross-diffusion term in Equation (3.8) (last term of the equation). $CD_{k\omega}$ is defined in Equation (3.10). [24], [28]

$$CD_{k\omega} = \max\left(2\rho\sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i}, 10^{-10}\right) \tag{3.10}$$

The turbulent eddy viscosity is defined as:

$$\nu_t = \frac{a_1 k}{\max(a_1 \omega, SF_2)} \tag{3.11}$$

Where F_2 is a second blending function defined in Equation (3.12). [24]

$$F_2 = \tanh\left[\left[max\left(\frac{2\sqrt{k}}{\beta^* \omega y}, \frac{500\nu}{y^2 \omega}\right)\right]^2\right] \tag{3.12}$$

The constants for the $k - \omega\ SST$ model are given in Table 3.1.

Table 3.1: Model constants for k-ω SST [28]

Model constant notation	**Model constant value**
β^*	0.09
α_1	5/9
β_1	3/40
σ_{k1}	0.85
$\sigma_{\omega 1}$	0.5
α_2	0.44
β_2	0.0828
σ_{k2}	1
$\sigma_{\omega 2}$	0.856

3.4 Near-wall turbulence

The boundary layer can be divided into the inner layer and the outer layer. The inner layer can further be described by three sublayers – the viscous sublayer, the buffer zone, and the turbulent zone (logarithmic layer). Dimensionless quantities describe these sublayers u^+ and y^+ for velocity and distance from the wall, respectively. The evolution of u^+ as a function of y^+ is shown in Figure 3.5. [24]

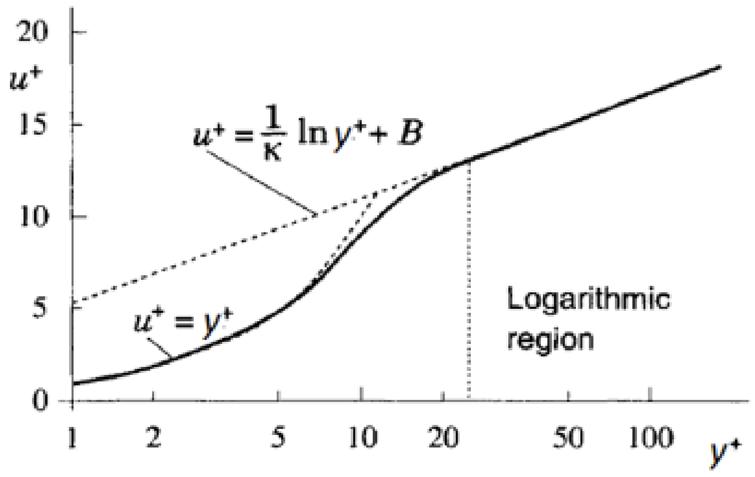


Figure 3.5: Law of the wall highlighted with the onset of the logarithmic region [29]

Viscous forces are dominating within the viscous sublayer, which exists for $y^+ < 5$, where it is assumed that $u^+ = y^+$. The shear stresses are assumed equal to the wall shear stress within this region. The transitional buffer zone is characterized by competing viscous and turbulent stresses, leading to complex flow structures. Finally, in the logarithmic zone ($30 < y^+ < 200$), turbulent stresses are dominating. [24]

Calculation of y^+ can be performed following Equation (3.13) – (3.15) below.

$$y^+ = \frac{y \cdot u_\tau}{\nu} \tag{3.13}$$

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}} \tag{3.14}$$

$$\tau_w = \frac{\mu \cdot u_w}{y} \tag{3.15}$$

4 Solver and solver settings

The numerical methods presented in this section of the report include discretization schemes used to resolve the partial derivatives. The general functionality of the choice of the solver is also given, as well as matrix solvers. These can be found in *controlDict*, *fvSchemes*, and *fvSolutions* directories in the *system* folder.

4.1 The PISO algorithm

The PISO algorithm (Pressure Implicit with Splitting of Operators) is a procedure for pressure-velocity coupling and was initially developed for the non-iterative calculation of transient and compressible flows. PISO can be regarded as an extension of the SIMPLE algorithm, with one predictor step and two corrector steps. [30]

Firstly, the discretized momentum equations are solved for a guessed or intermediate pressure field, p^*, to yield intermediate calculations for the velocity u^*. Then comes the first corrector step. As the first calculations are performed based on guessed values, the velocity fields will not satisfy continuity. This correction step corresponds to that of the SIMPLE algorithm. The first correction step is described in Equation (4.1). [30]

$$\begin{aligned} p^{**} &= p^* + p' \\ u^{**} &= u^* + u' \end{aligned} \tag{4.1}$$

Where quantities with “ ** ” superscript describe the corrected values after the first correction step, “ * ” is the intermediate values, and the “ ‘ “ notation defines the correction value.

As previously mentioned, the PISO algorithm performs two corrector steps. The momentum equation is solved using the value of u^* and p^{**}. Then, the twice-corrected solution for the velocity field can be calculated by solving the momentum equation once more. A second pressure-correction equation is introduced to estimate the pressure field better. [30]

The PISO algorithm for one time-step is illustrated in Figure 4.1. The block called *nCorrectors* describes how many times the pressure and velocity fields are solved for each time-step. For this thesis, *nCorrectors* has been set to 2, which has proven to be stable during preliminary analyses.

The *nNonOrthogonalCorrectors* block indicates how often the pressure correction equation is solved for each time-step. Unstructured meshes, such as those used in this thesis (and for most practical engineering problems), can cause calculation errors; hence, *nNonOrthogonalCorrectors* should be introduced. For simple geometrical shapes, it has been found that setting this value equal to 3 yields accurate results for added mass. A comparison with *nCorrectors* set to 5 and *nNonOrthogonalCorrectors* set to 7 has been performed with no difference in the results.

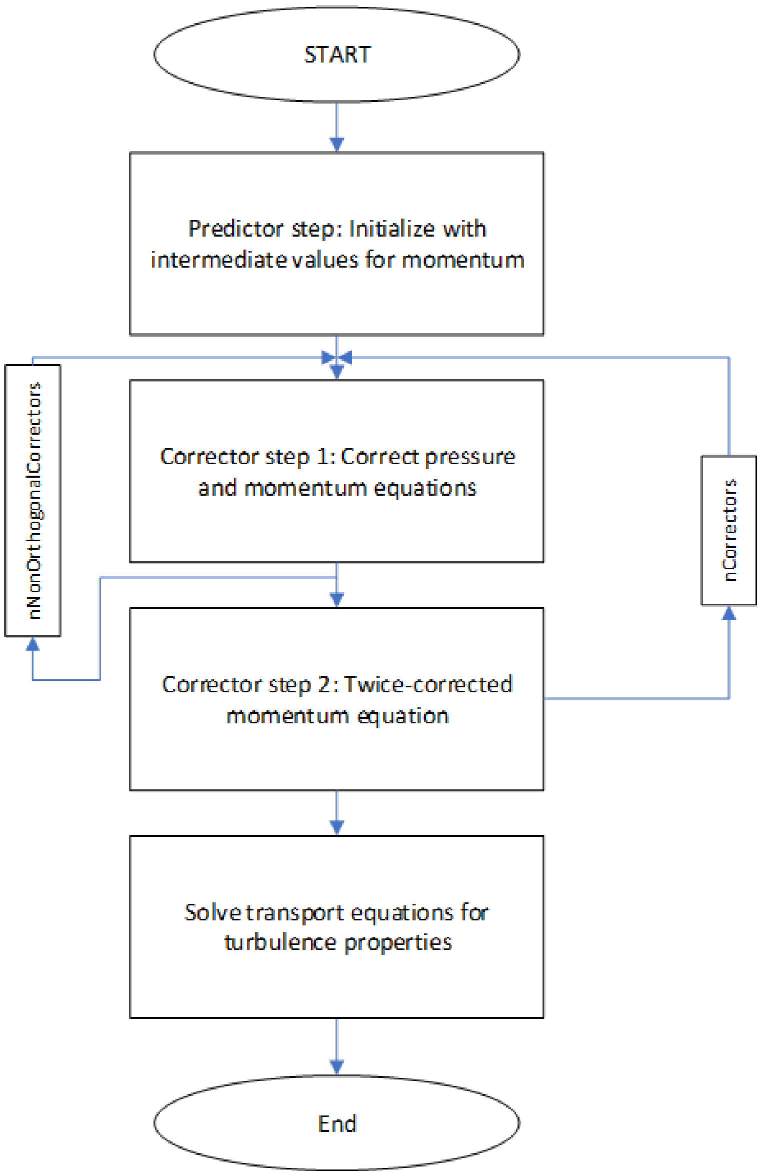

START
Predictor step: Initialize with intermediate values for momentum
Corrector step 1: Correct pressure and momentum equations
Corrector step 2: Twice-corrected momentum equation
Solve transport equations for turbulence properties
End
nNonOrthogonalCorrectors
nCorrectors

Figure 4.1: Flowchart of the PISO algorithm

4.2 Discretization methods

The numerical discretization methods are important to assess when performing a CFD analysis. If the numerical schemes are not valid for the analysis in question, the results will not yield representable results. In general, there are three criteria for the numerical schemes – boundedness, transportiveness, and conservativeness. Discretization methods are found in the *fvSchemes* dictionary, while the linear solvers are in the *fvSolutions* dictionary.

The temporal schemes define how a general property, ϕ, is integrated in time. The implicit Euler time integration method is used in the analyses. Implicit Euler is a first-order scheme guaranteeing boundedness, whereas some higher-order schemes, such as Crank-Nicolson, do not guarantee boundedness. Equation (4.2) defines the implicit Euler scheme. [31]

$$\frac{d\varphi}{dt} = \frac{\varphi_{t+\Delta t} - \varphi_t}{\Delta t} \tag{4.2}$$

The convective terms are discretized using linear upwind differencing, a second-order scheme, which is less prone to false diffusion than first-order upwind differencing. The first-order upwind scheme utilizes one upstream value for extrapolating values from the node to the neighboring face; the linear upwind discretization scheme utilizes two nodal upstream values. Equation (4.3) considers a general property φ on a one-dimensional grid with the flow in the positive x-direction. The linear upwind differencing scheme is a second-order extension of first-order upwind differencing due to the presence of the gradient in the last term of the equation. [32]

Capital notations, i.e., W, P, and E, represent the location of the nodes, while lower-case notations represent interfaces. See Figure 4.2 for the illustration of the one-dimensional grid.

$$\varphi_e = \varphi_P + \frac{\varphi_P - \varphi_W}{\delta x} \cdot \frac{\delta x}{2} = \varphi_P + \frac{\varphi_P - \varphi_W}{2} = \frac{3}{2}\varphi_P - \frac{1}{2}\varphi_W \tag{4.3}$$

Where φ_e is the value at the east interface, φ_P and φ_W are values at node P and W, respectively, and δx is the distance between the nodes.

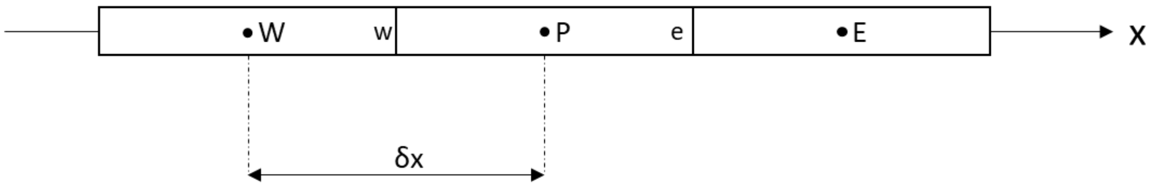


Figure 4.2: One-dimensional grid

Using upwind differencing schemes ensures that the schemes are stable and that the transportiveness requirement is obeyed, unlike central differencing. [33]

The divergence terms are discretized using linear interpolation, i.e., central differencing. Referring to the one-dimensional grid in Figure 4.2, the value of a property at interfaces can be expressed by Equation (4.4). [30]

$$\varphi_w = \frac{\varphi_W + \varphi_P}{2}, \qquad \varphi_e = \frac{\varphi_P + \varphi_E}{2} \tag{4.4}$$

The non-orthogonality of a mesh is defined as the angle between a line connecting the cell-centers of two cells and a normal face vector. Figure 4.3 shows this for cells P and N. For a non-orthogonal mesh, the vectors $\boldsymbol{d}$ and $\boldsymbol{S_f}$ are not parallel. [35]

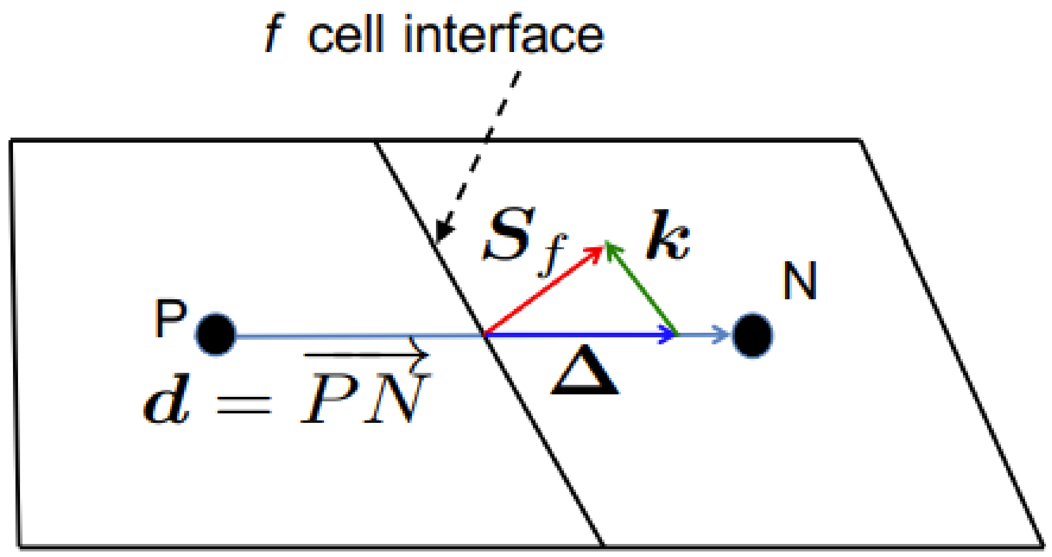


Figure 4.3: Principal sketch showing non-orthogonality in a mesh [35]

The non-orthogonality correction is accounted for by the second term on the right-hand side of Equation (4.5). [35]

$$\nabla\varphi_f \cdot \boldsymbol{S_f} = \frac{\phi_N - \phi_P}{|\boldsymbol{d}|}|\boldsymbol{\Delta}| + \nabla\phi_f \cdot \boldsymbol{k} \tag{4.5}$$

$\nabla\phi_f$ is calculated with Equation (4.6) by interpolating gradient values at P and N. [35]

$$\nabla\phi_f = f_x\nabla\phi_P + (1 - f_x)\nabla\phi_N \tag{4.6}$$

Where $\nabla\phi_P$ and $\nabla\phi_N$ are the gradients at node P and N, respectively, and f_x is an interpolation factor.

The normal surface gradients, *snGradSchemes* in the *fvSchemes* dictionary, are corrected for non-orthogonality of the mesh. Non-orthogonal corrections are applied by specifying *corrected* for *snGradSchemes*. The correction value depends on the non-orthogonality of the mesh. Corrections for non-orthogonality are also used for the Laplacian terms. For highly non-orthogonal meshes, it can be necessary to introduce limiters to achieve convergence. [36], [37]

To summarize the discretization methods used, refer to Table 4.1.

Table 4.1: Summary of discretization methods used in the CFD analyses

Description	**Symbol**	**Scheme**
Temporal	$\frac{\partial \phi}{\partial t}$	Euler
Convective	$\nabla \phi$	Linear upwind
Diffusive	$\nabla \cdot \phi$	Gauss linear
Surface-normal gradient	$\vec{n} \cdot (\nabla \phi)_f$	Corrected
Laplacian	$\nabla^2 \phi$	Gauss linear corrected

4.3 Linear solvers

The linear solvers are defined in the *fvSolution* file. Here, the differential equations are transformed to a set of linear equations, which are solved in the form described by Equation (4.7).

$$Ax = b \tag{4.7}$$

Where A is the coefficient matrix, x is the vector of unknowns, and b is the source vector.

OpenFOAM provides a variety of linear solvers that the user can implement. GAMG coarsens the grid to get fast solutions. The coarse-grid solutions are then mapped to a finer grid to initiate fine-grid solving of the equations. This is called geometric agglomeration and is illustrated in Figure 4.4 below. This procedure is used to solve the pressure equations. [38], [39], [40]

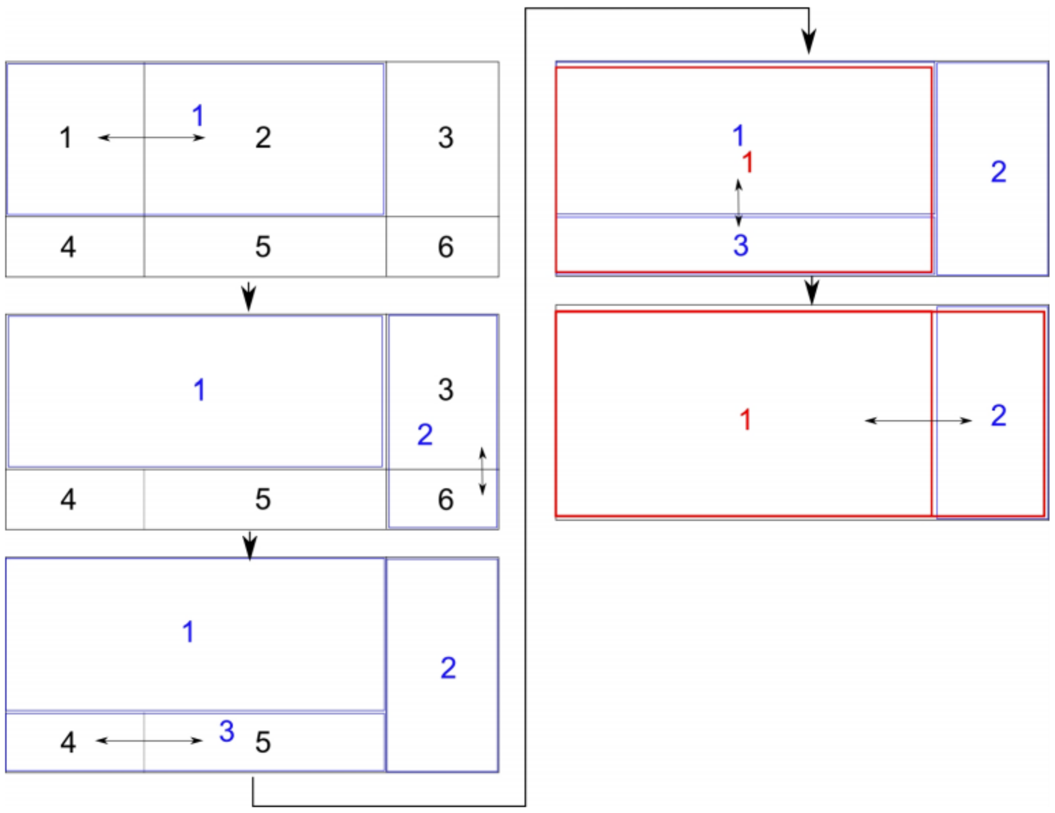


Figure 4.4: Geometric agglomeration process on a 2D mesh [40]

PBiCG is used in the analyses to solve the momentum and turbulence equations along with the DILU preconditioner. PBiCG is a Krylov subspace solver, where the squared matrix A (n-by-n dimensions) is multiplied by a vector b for the first r powers of A. [40]

The method can be computed as follows:

1. Multiply the matrix A by the vector b to get a new vector Ab
2. Multiply the matrix A with the previously calculated result to get A^2b
3. Continue this process for the r powers of A

The analyst controls the behavior of the matrix solvers by adjusting tolerances. Higher tolerances generate less accurate solutions but take less time to solve. Hence, the analyst is responsible for setting the tolerances small enough not to compromise the accuracy of the solution. For this thesis, a tolerance of $1 \cdot 10^{-6}$ is used for the pressure equation, while $1 \cdot 10^{-8}$ is used for the momentum and turbulence equations.

5 Analysis configuration

This section gives insight into the methodology applied in setting up the computational models for this thesis. The analyses performed for the present work use OpenFOAM v-2012.

5.1 Fluid properties

The fluid properties are stated for seawater. The seawater properties are evaluated at $5^{o}C$, which is a typical design temperature for subsea installations. Fluid properties are taken from DNV-RP-C205, Appendix F, and presented in Table 5.1.

Table 5.1: Properties of seawater at 5 ºC [18]

Description	Symbol	Value	Unit
Density	ρ	1027.6	kg/m^3
Kinematic viscosity	ν	1.6E-06	m/s^2

5.2 Time-stepping

The Courant number (also called the Courant-Friedrich-Lewy number) is an important parameter to assess when performing a CFD analysis. If the number becomes too high, the solver can become unstable and eventually diverge. The Courant number is defined in Equation (5.1). [41]

$$Co = u \cdot \frac{\Delta t}{\Delta x} \tag{5.1}$$

Where u is the flow velocity, Δt is the time-step, and Δx is the grid spacing. As a rule, the time-stepping should be fine enough so that the Co value does not increase beyond unity. For $Co > 1$, the solver cannot capture all the details regarding the flow field.

It is worth mentioning that during transient analyses, some high transient start-up values are expected. If this situation is extended beyond a few start-up iterations, the analyst can use adjustable time-stepping in the *controlDict* dictionary.

5.3 Acceleration and velocities

Since the added mass force is proportional to the acceleration, the numerical value of the mass should be constant, although the accelerations differ. For this thesis, three accelerations are investigated – 0.5 m/s^2, 1.0 m/s^2 and 2.0 m/s^2.

Based on experience, typical velocities range from 0.2 m/s – 1.0 m/s when lowering subsea structures. All analyses with constant acceleration are initiated at 0.2 m/s freestream velocity to ease convergence and stability concerning the initial values of the turbulence parameters.

5.4 Meshing

All meshes are created using the meshing module in ANSYS 19.2 with physics preference set to CFD and solver preference to Fluent to create a valid mesh format when exporting to OpenFOAM polymesh.

ANSYS provides numerous methods of creating a mesh with different cell types and surface refinement options. The simplest of the cell types is the tetrahedral mesh, composed of triangles. These cell types can be used where the curvature is high, and other methods fail to generate a mesh, or the quality of using other methods is too poor for analysis purposes. Hexahedron cells are more efficient and generally give more accurate results than tetrahedron cells. The reason is that hexahedron cells have more faces per cell compared to tetrahedrons. One possible downside of using hexahedron cells is the lack of flexibility. For highly complex geometries, the software might not be able to generate a mesh using hexahedron cells.

A third cell type is the polyhedral cells, which the cells are composed of agglomerated tetrahedral cells. It has been shown that polyhedral cells are far more efficient than tetrahedrons while being more accurate for a smaller overall cell count. An illustration of the three mentioned cell types is presented in Figure 5.1. Hexahedrons are to the left, tetrahedrons in the middle and polyhedrons to the right. [42], [43]

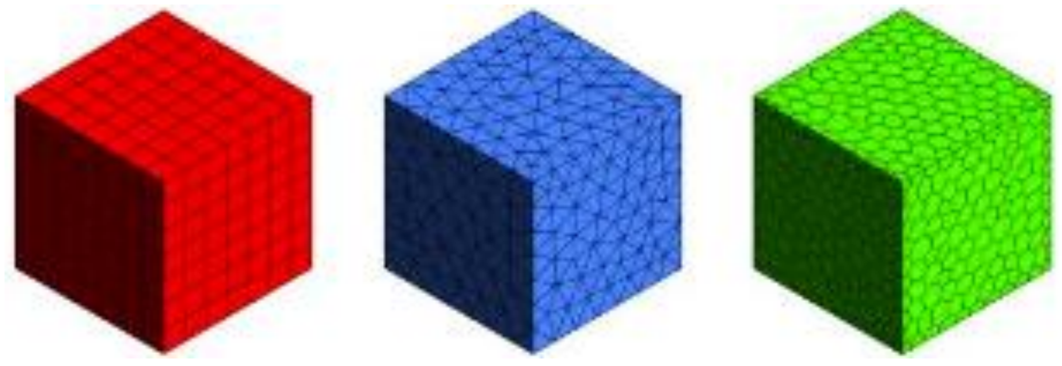


Figure 5.1: Basic cell types for CFD. Left is hexahedral, the middle is tetrahedral, and right is polyhedral cells [43]

With the current licensing configuration at Stressman Engineering AS, polyhedral cells cannot be created. Because of this, hexahedral cells are used with the "CutCell" method in ANSYS Meshing, which generates an unstructured grid. The CutCell method creates a refinement zone around the object in the fluid domain, where hanging nodes are located at the interface of each refinement layer. [44]

5.5 Post-processing

The pressure forces can be calculated in accordance with Equation (5.2). [45] To simplify the post-processing procedure, automatic force calculations are performed using *forcelibs* in OpenFOAM. A python script is developed to ease the post-processing procedure both for added mass and drag coefficient. The python script is available in Appendix [D] and a user manual for Stressman Engineering AS in Appendix [E].

$$F_p = \sum_{i=1}^{N} \rho s_{f,i} \cdot (p_i - p_{ref}) \tag{5.2}$$

Where p is the kinematic pressure, $s_{f,i}$ is the face area vector, p_i is the pressure and p_{ref} is the reference pressure.

The submerged cylinder described in section 6.1 is compared to Equation (5.3), which is the analytical added mass for a 3D cylinder. It is more difficult to validate the output from cylindrical frame analyses in section 6.2, as no theoretical data is available for complex objects. Likely, the simplified procedures in accordance with DNV-RP-C205 and DNV-RP-H103 do not give very accurate results. However, the added mass from OpenFOAM is compared to two calculation methods in accordance with the before-mentioned codes:

1. Calculating added mass for each cylinder and adding the results to give a total estimate.
2. Estimate added mass for a perforated plate.

The former calculation can be done in accordance with Equation (5.3), while the latter is performed following Equation (5.4). Method 1 is also used to validate the methodology for the submerged cylinder described in section 6.1. [46]

$$m_A = \rho \cdot C_A \cdot V_R \tag{5.3}$$

$$A_{33} = \begin{cases} A_{33,S} \; if \; p \leq 5 \\ A_{33,S} \cdot \left(0.7 + 0.3 \cdot \cos\left[\dfrac{\pi(p-5)}{34}\right]\right) \; if \; 5 < p \leq 34 \\ A_{33,S} \cdot e^{\frac{10-p}{28}} \; if \; 34 \leq p < 50 \end{cases} \tag{5.4}$$

Where V_R is the reference volume, A_{33} is added mass, $A_{33,S}$ is added mass without perforation, and p is the perforation ratio in percent. An illustration of evaluating Equation (5.8) is shown in Figure 5.2 for a flat plate with different length-to-width ratios.

For a flat plate without perforation, added mass can be calculated using Equation (5.5), taken from Table A-2 in DNV-RP-H103. [46]

$$m_A = \rho \cdot C_A \cdot \frac{\pi}{4} \cdot a^2 \cdot b \tag{5.5}$$

Where a and b are the width and length of the plate, respectively.

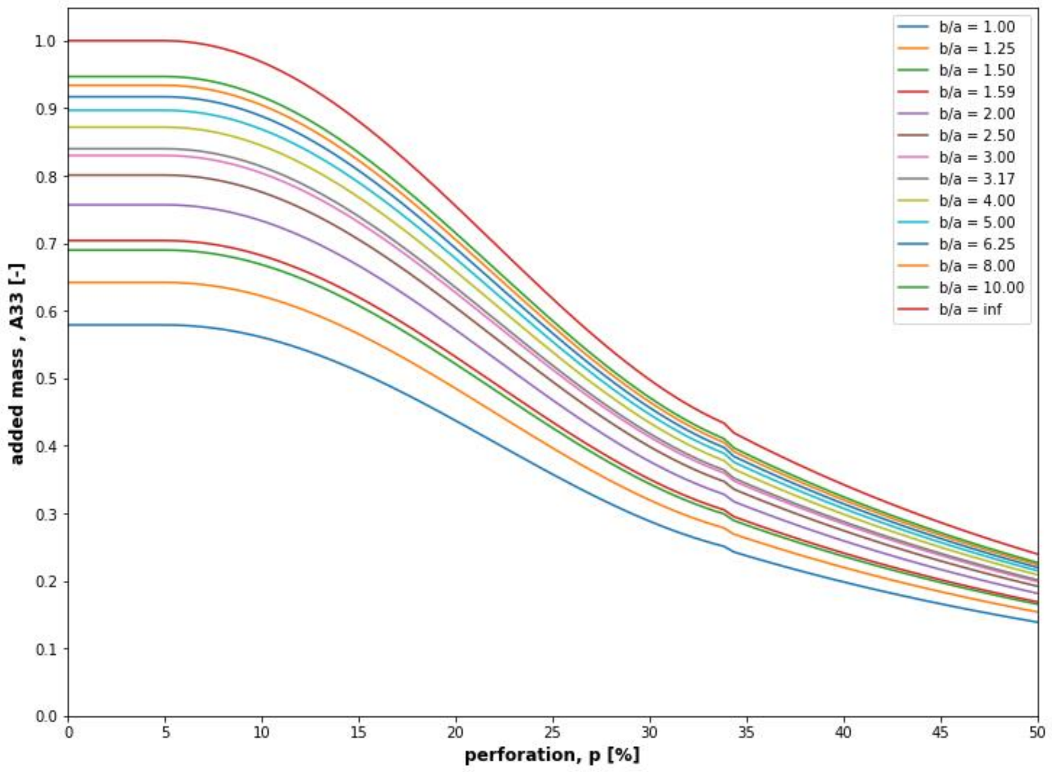


Figure 5.2: Effect of perforation for a flat plate in accordance with DNV-RP-H103. b is the length, and a is the width of the flat plate [46]

6 Case descriptions

This section describes the computational models with the boundary conditions applied. The added mass analyses are initiated at the minimum velocity of 0.2 m/s for 0.2 s. This is to overcome the large transient overshoots during the first few iterations and give initial estimates for the turbulence properties.

6.1 Case description – the submerged cylinder

The submerged cylinder analysis aims to establish a methodology to analyze more complex geometries further. Details regarding the fluid domain and geometry of the cylinder can be found in Appendix [F]. Figure 6.1 shows the fluid domain on the left and the cylinder on the right. The cylinder is symmetric about the XY-, XZ-, and YZ-planes. From section 2.3.1, it was demonstrated that this reduces the added mass matrix only to be composed of diagonal elements.

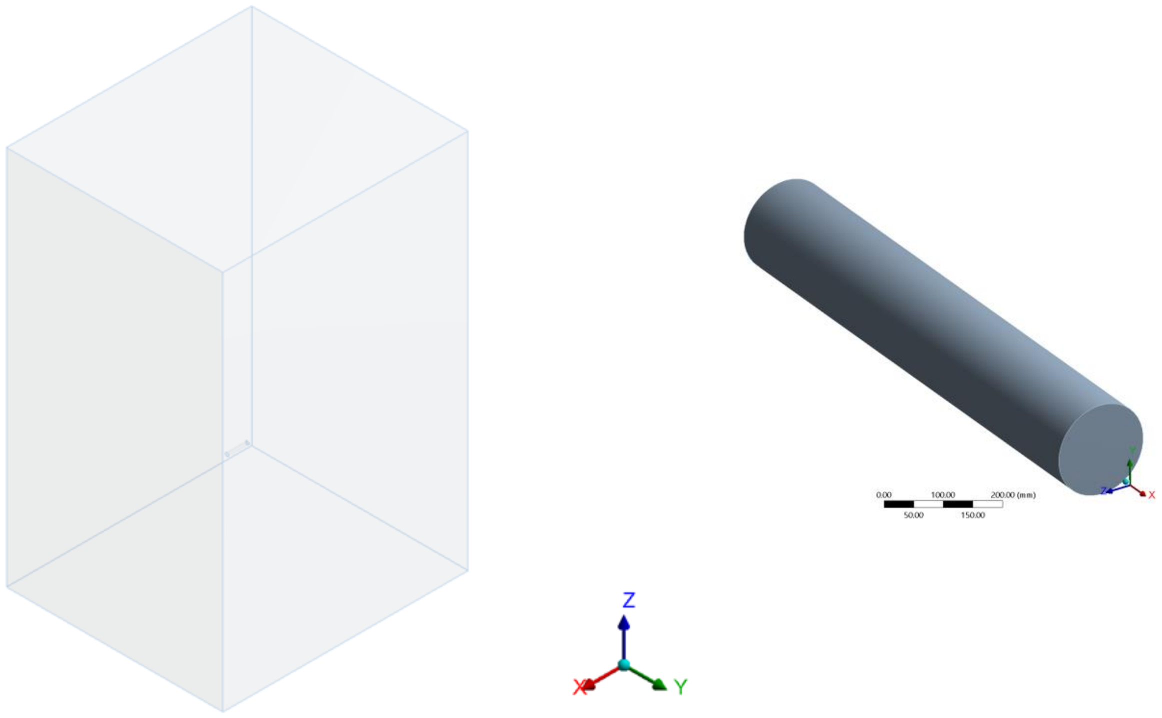


Figure 6.1: Isometric view of the fluid domain (left) and the cylinder (right)

Detailed views of the local mesh sizing are shown in Figure 6.2 and Figure 6.3 in the XY-plane and YZ-plane, respectively. The mesh is configured with a global sizing of 200 mm and 10 mm local surface refinement around the cylinder.

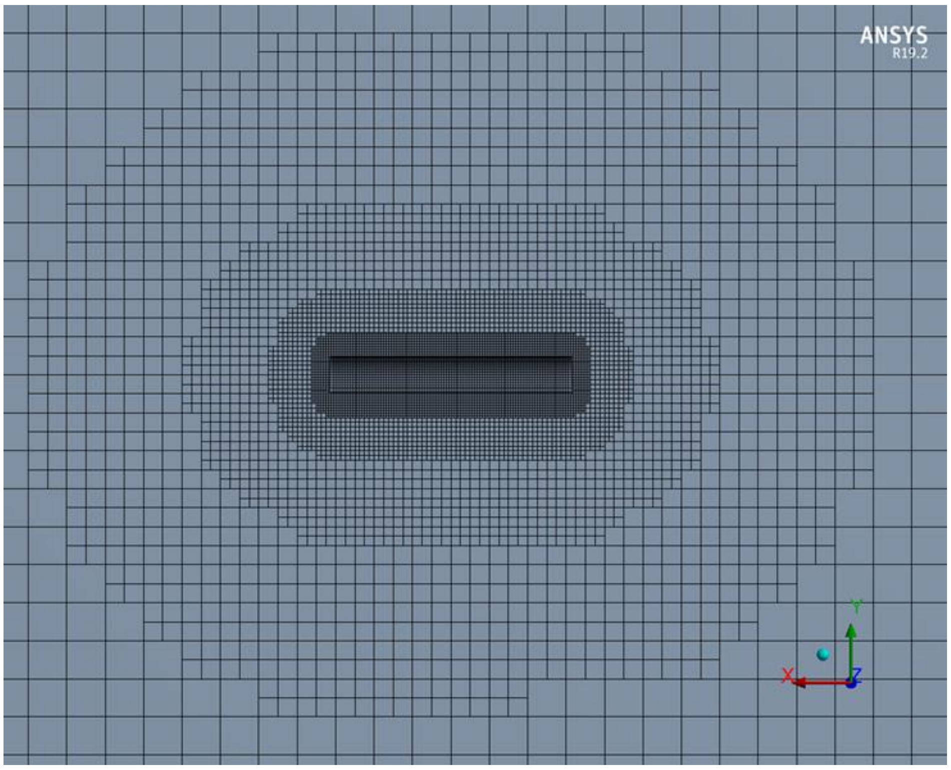


Figure 6.2: Detailed view of the local mesh sizing in the XY-plane

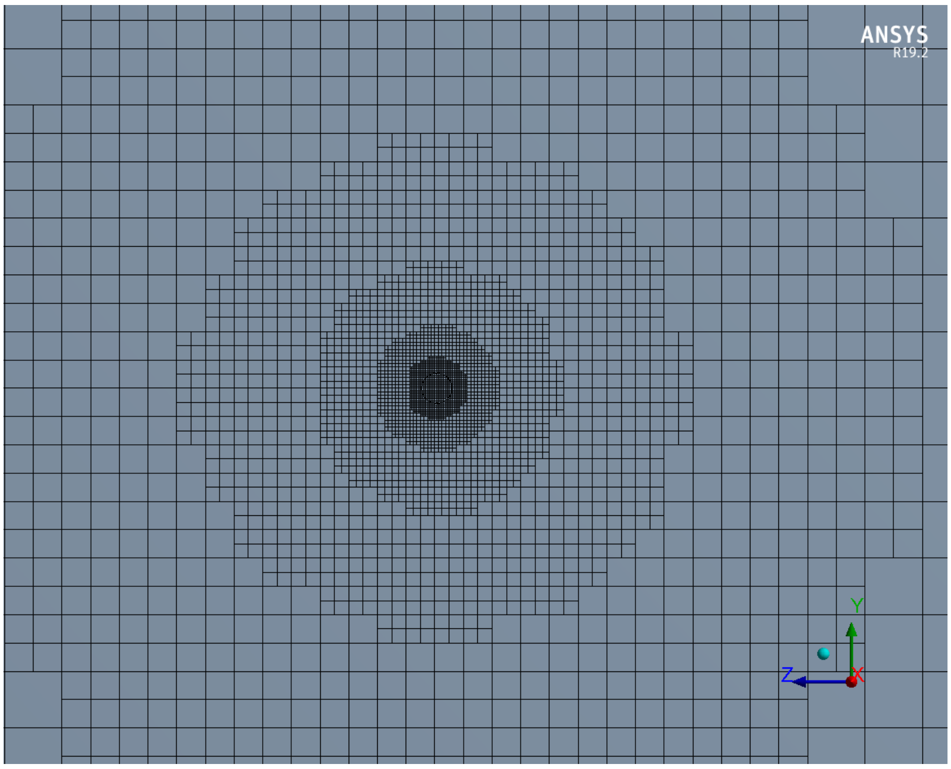


Figure 6.3: Detailed view of the local mesh sizing in the YZ-plane

6.1.1 Boundary conditions

The dimensions of the fluid domain should be large enough to simulate an infinite fluid. Hence, the symmetry boundaries should be far away from the cylinder so that the boundaries are of negligible influence on the results. The boundary conditions are summarized in Table 6.1. See Appendix [G] for the OpenFOAM case-setup for the submerged cylinder.

Symmetry conditions imply that the normal velocity component and pressure gradient should be equal to zero at the boundary.

Further, the *zeroGradient* pressure condition is used both at the inlet and outlet. This is assumed to be a realistic condition since gravitational and buoyancy forces are neglected in the analyses.

The *noSlip* condition is employed at the "walls" patch, which means that all velocity components are equal to zero, i.e., $u_{w,x} = u_{w,y} = u_{w,z} = 0$. Finally, the turbulence properties are treated with wall-functions at the "walls" patch to allow a coarser local mesh refinement.

Table 6.1: Boundary conditions for the submerged cylinder

Field	**inlet**	**outlet**	**walls**
$\boldsymbol{U}$ **[m/s]**	*uniformFixedValue*[1]	*zeroGradient*	*noSlip*
$\boldsymbol{p}$ **[m^2/s^2]**	*zeroGradient*	*zeroGradient*	*zeroGradient*
$\boldsymbol{k}$ **[m^2/s^2]**	*fixedValue*[2]	*zeroGradient*	*kqRWallFunction*
$\boldsymbol{\omega}$ **[1/s]**	*fixedValue*	*zeroGradient*	*omegaWallFunction*
$\boldsymbol{\nu_t}$ **[m^2/s]**	*calculated*	*calculated*	*nutkWallFunction*

The turbulence properties, i.e., k and ω, need specified initial values. Therefore, the constant acceleration cases are started at a constant velocity of 0.2 m/s for 0.2 s to initiate the flow field and to overcome transient overshoots. Initial values can be computed from Equations (6.1) and (6.2) for turbulent kinetic energy and specific dissipation rate, respectively. [47]

$$k = \frac{3}{2} \cdot \left(U_{ref} \cdot I\right)^2 \tag{6.1}$$

$$\omega = \frac{k^{0.5}}{C_\mu^{0.25} \cdot L} \tag{6.2}$$

[1] Time-dependent velocity vector is written as (time (Ux Uy Uz)) in the U dictionary.

[2] In cases where reverse flow can occur, such as for oscillating flow velocity, *turbulentIntensityKineticEnergy* should be used instead of *fixedValue*. This constraint switches from *fixedValue* to *zeroGradient* when reverse flow is detected.

Where U_{ref} is the freestream fluid velocity, I is the turbulence intensity, L is the characteristic length, and $C_{\mu} = 0.09$. The turbulence intensity is assumed to be at 4.5 % for the submerged cylinder and the cylindrical frame described in the next section. During preliminary analyses, it was found that the turbulence intensity was of negligible influence on the results, with a difference of 10^{-5} when comparing added mass at 4.5 % and 1.0 % turbulence intensity.

Figure 6.4 shows the location of the patches with keywords to indicate the boundary conditions.

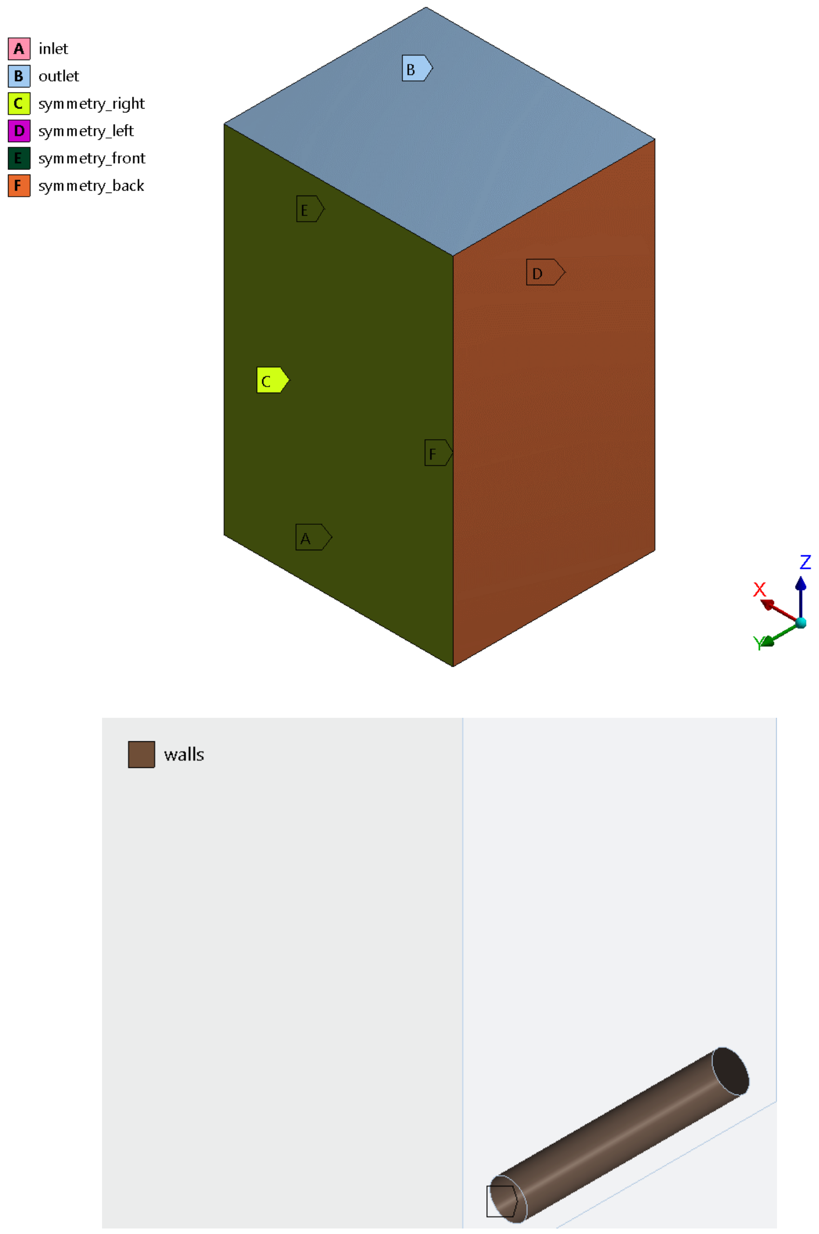


Figure 6.4: Location of patches for the submerged cylinder

The added mass obtained from the analysis is easily verified with DNV-RP-C205, as the RP contains tabulated data for the added mass of simple geometries. Equation (5.3) is used to validate the methodology.

6.2 Case description – the cylindrical frame

The cylindrical frame is the focus of this thesis, as it illustrates a typical protection roof assembled on a subsea structure, though the geometrical dimensions are smaller. Added mass is considered only in the heave direction (Z-direction), as this would be the governing direction for a typical subsea structure.

The structure is composed of five CHS Ø168.3 mm in parallel with one CHS Ø219.1 mm cylinder in each end. See Appendix [H] for the dimensions of the cylindrical frame. Figure 6.5 shows the geometry of the cylindrical frame. From the figure, it can be seen that the cylindrical frame is symmetric about the three primary planes.

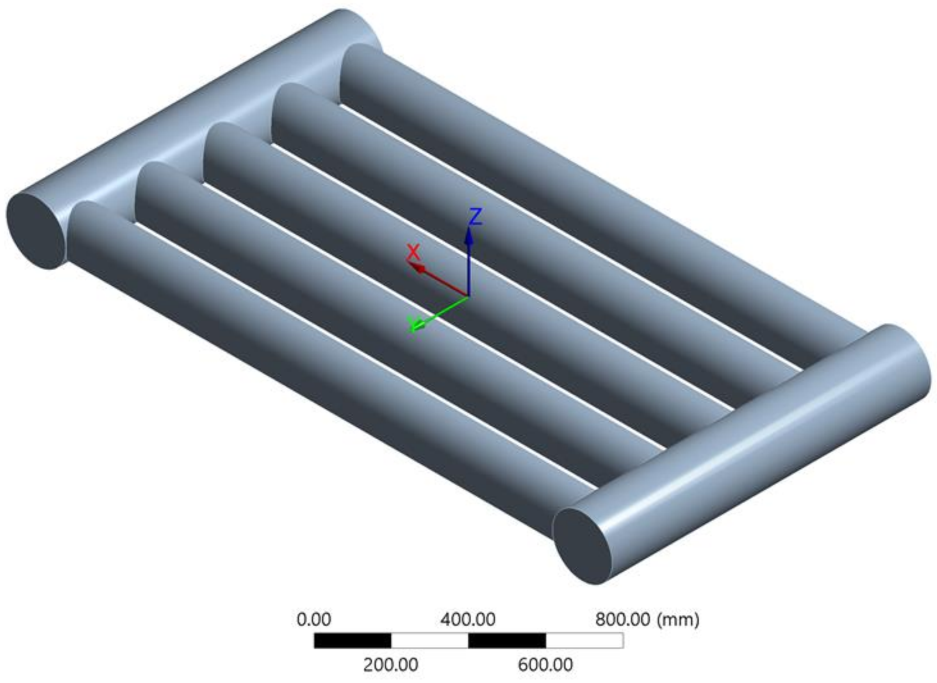


Figure 6.5: Isometric view of the cylindrical frame

Three independence studies are performed for the model. The first is concerning the domain size required, followed by time-step and mesh size independence studies. These studies are done to find a sufficient compromise between accuracy and efficiency in the computations, as is the main goal of practical engineering problems.

The two remaining independence analyses are performed at constant acceleration. The reasoning behind this is that the drag force analyses are performed over a long time to see how the drag coefficient changes over time.

A second configuration of the cylindrical frame will be analyzed, where two of the inner cylinders are removed, and is shown in Figure 6.6. The purpose of this configuration is to demonstrate the effect of the cylinder spacing and comparing the results to DNV-RP-C205 and DNV-RP-H103.

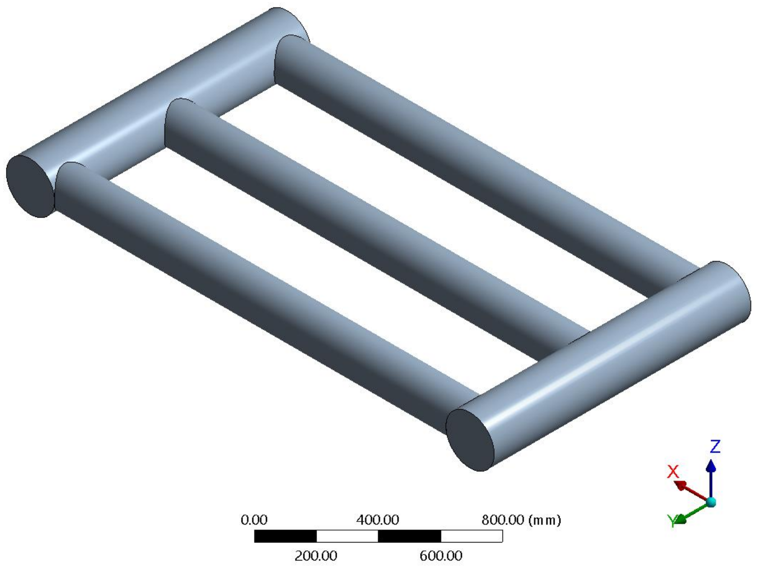


Figure 6.6: Isometric view of the second cylinder frame configuration. The two inner cylinders are removed.

6.2.1 Boundary conditions

The location of the applied boundaries can be seen in Figure 6.7. Appendix [I] and [J] are the setup files for constant acceleration and constant velocity, respectively.

Table 6.2: Boundary conditions for the cylindrical frame

Field	**inlet**	**outlet**	**walls**
$\boldsymbol{U}$ **[m/s]**	*uniformFixedValue*	*zeroGradient*	*noSlip*
$\boldsymbol{p}$ **[m^2/s^2]**	*zeroGradient*	*zeroGradient*	*zeroGradient*
$\boldsymbol{k}$ **[m^2/s^2]**	*fixedValue*	*zeroGradient*	*kqRWallFunction*
$\boldsymbol{\omega}$ **[1/s]**	*fixedValue*	*zeroGradient*	*omegaWallFunction*
$\boldsymbol{\nu_t}$ **[m^2/s]**	*calculated*	*calculated*	*nutkWallFunction*

The *kqRWallFunction* is a *zeroGradient* condition employed for the wall boundaries at high Reynolds numbers, while *omegaWallFunction* is a *fixedValue* condition, and is applicable for both low and high Reynolds numbers. [48], [49]

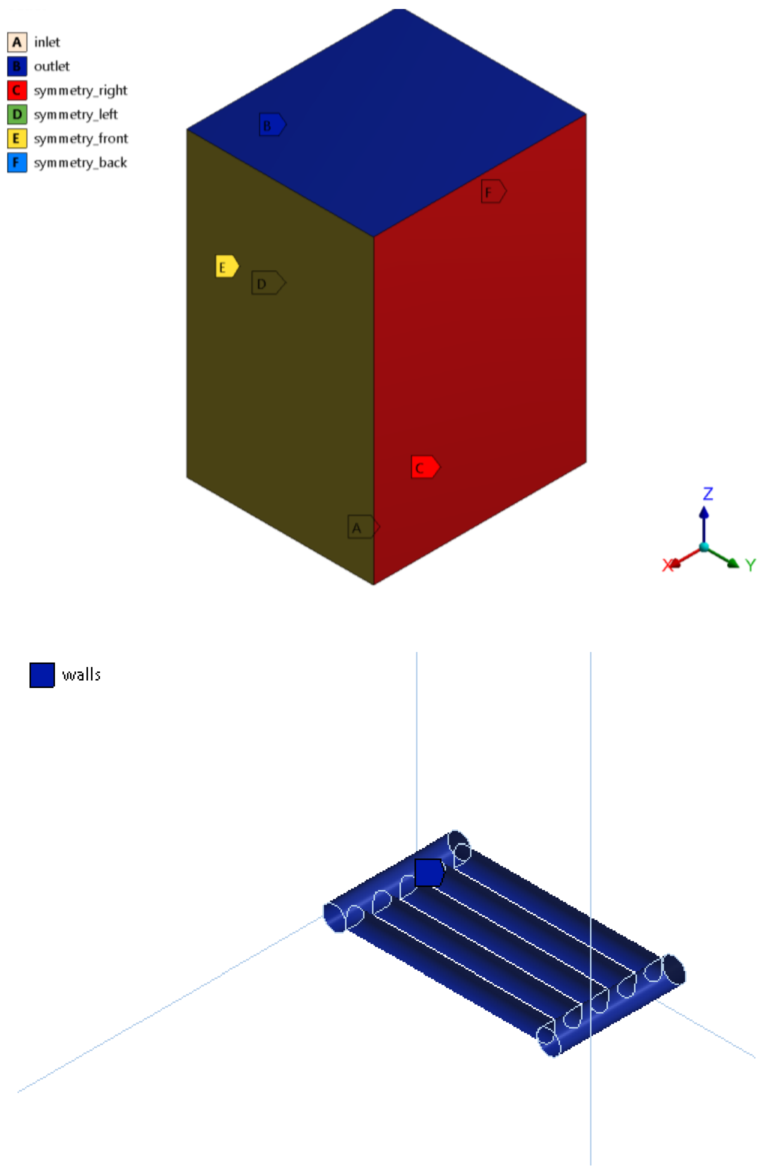


Figure 6.7: Location of boundaries for the cylindrical frame

A characteristic length is needed to calculate initial values for the specific dissipation rate in Equation (6.2). However, for a complex structure, the characteristic length is not easily obtained. It is common to assume the smallest dimension of an object to be the characteristic length. The smallest geometrical dimension (the minimum cylinder diameter) is used to calculate initial values for ω. The analyses appear to be stable with this assumption.

6.2.2 Domain size independence study

The purpose of the domain size independence study is to evaluate the necessary distance from the object to the symmetry boundary conditions to simulate an infinite fluid. A total of seven fluid domains are investigated, where the size of the domain increases each iteration. An illustration of the first and final domain sizes is shown in Figure 6.8.

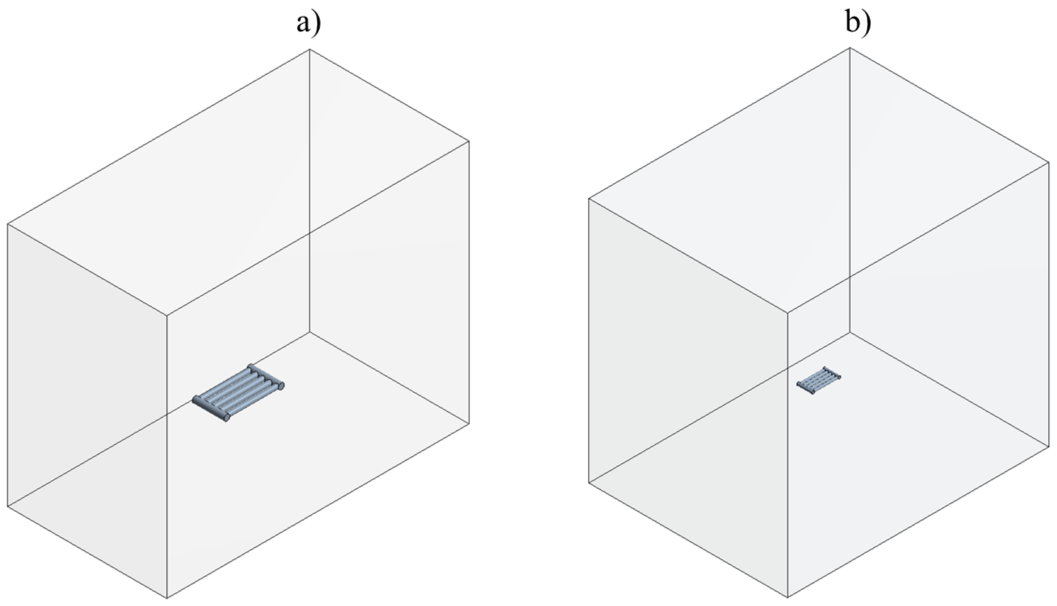


Figure 6.8: a) initial fluid domain size, b) final fluid domain size

The investigated fluid domains are compared in Figure 6.9. As shown from the figure, there is little variation between fluid domains 1 – 3, which are the smallest. However, when comparing domain 4 to domain 3, the change is more significant. When comparing domain 6 to domain 7, there is a marginal difference in the drag coefficient throughout the analysis time. Hence, domain 6 has achieved size independence and is used for the remaining analyses. The complete dimensions of fluid domain 6 can be found in Appendix [K].

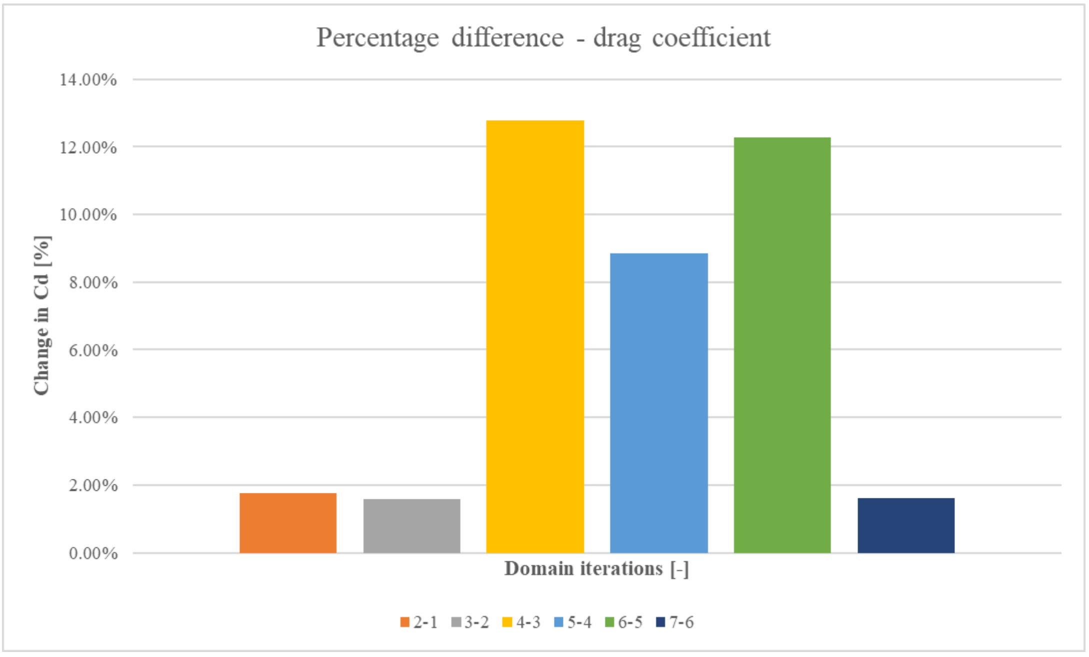


Figure 6.9: Percentage difference in the drag coefficient for the investigated fluid domains

6.2.3 Time-step independence study

It is important to ensure that sufficiently fine time-stepping is used in the analyses. The time-step independence study is performed to find an optimal trade-off between accuracy and efficiency. Figure 6.10 shows the time-history of the hydrodynamic force with different time-stepping. As can be seen, the difference between 10 μs and 0.1 ms is relatively small. Maximum and average differences are shown in Table 6.3.

Table 6.3: Comparison of hydrodynamic force for the investigated time-step settings

Time-stepping comparison	Maximum difference [%]	Avg. difference [%]
10 ms and 1.0 ms	25.6	6.1
1.0 ms and 0.1 ms	14.2	4.6
10 μs and 0.1 ms	11.1	3

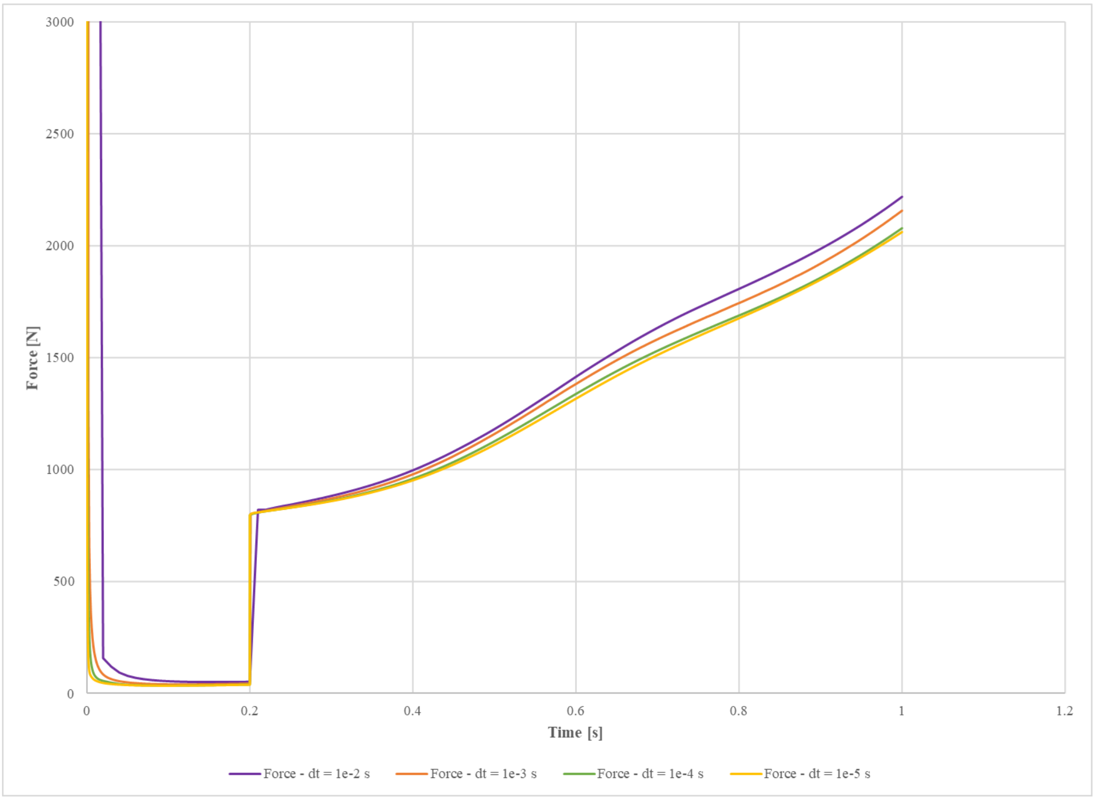


Figure 6.10: Comparison of the force history with different temporal resolutions for 1.0 m/s^2 flow acceleration. Acceleration is initiated at t = 0.2 s

The added mass analyses are run with a time-step of 0.1 ms due to negligible deviations compared to 10 μs time-stepping. On the other hand, the constant velocity analyses are performed using 1.0 ms in time-stepping because of the negligible difference compared to 0.1 ms.

6.2.4 Mesh size independence study

Evaluating the mesh is highly important when performing numerical analyses, as it describes the resolution of the fluid domain discretization. When using wall functions, the analyst can use a coarser mesh resolution in these regions compared to omitting wall functions.

The investigated mesh resolutions, with mesh metrics, are presented in Table 6.4. A finer local resolution could not be achieved due to the high CPU and RAM usage. The different resolutions presented are concerning the local sizing around the cylindrical frame. 200 mm cells are used for the mesh shown in Figure 6.11. Detailed views of the local mesh sizing can be seen in Figure 6.12 and Figure 6.13.

Table 6.4: Mesh metrics for the mesh independence study

Mesh	**Local surface refinement [mm]**	**Number of cells**	**Aspect ratio (max)**	**Non-orthogonality (max/avg.)**	**Skewness (max)**
M1	30	403 976	6.8	51.5	6.2
M2	20	485 859	6.8	51.5/5.8	0.7
M3	15	849 860	7.4	57.8/6.8	0.9
M4	10	1 245 153	7.4	57.8/5.8	0.9

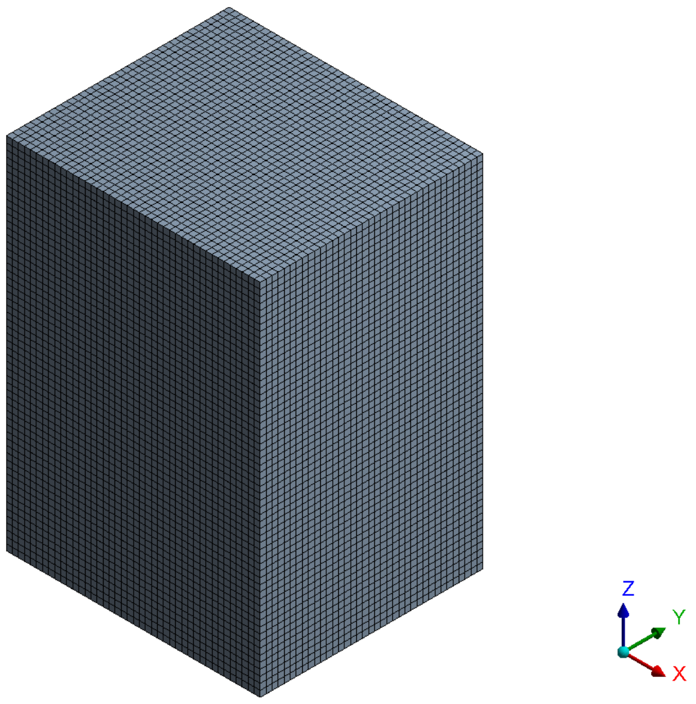


Figure 6.11: Isometric view of the global mesh size. The global sizing is 200 mm

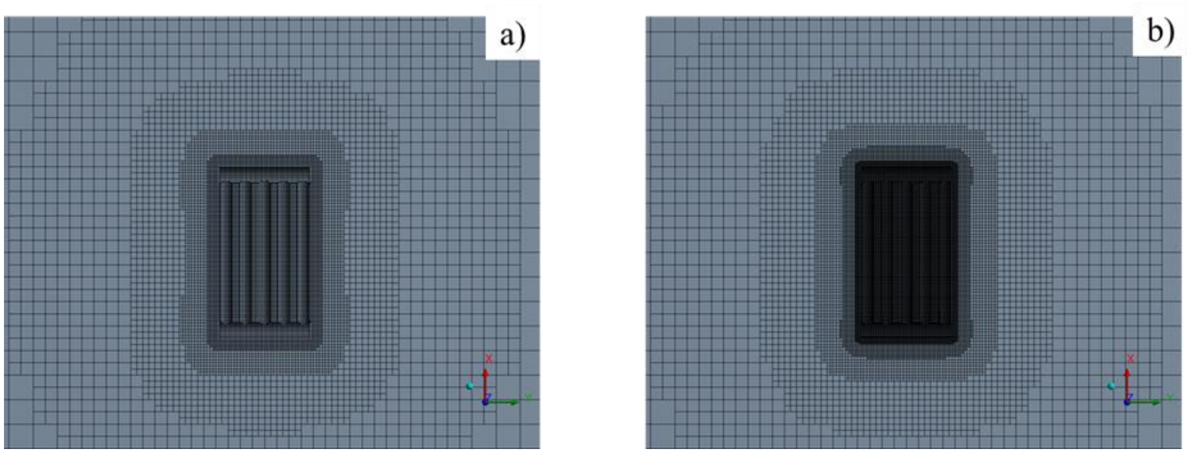


Figure 6.12: Detailed view in the XY-plane of a) initial mesh size with 30 mm local sizing and b) final mesh size with 10 mm local sizing

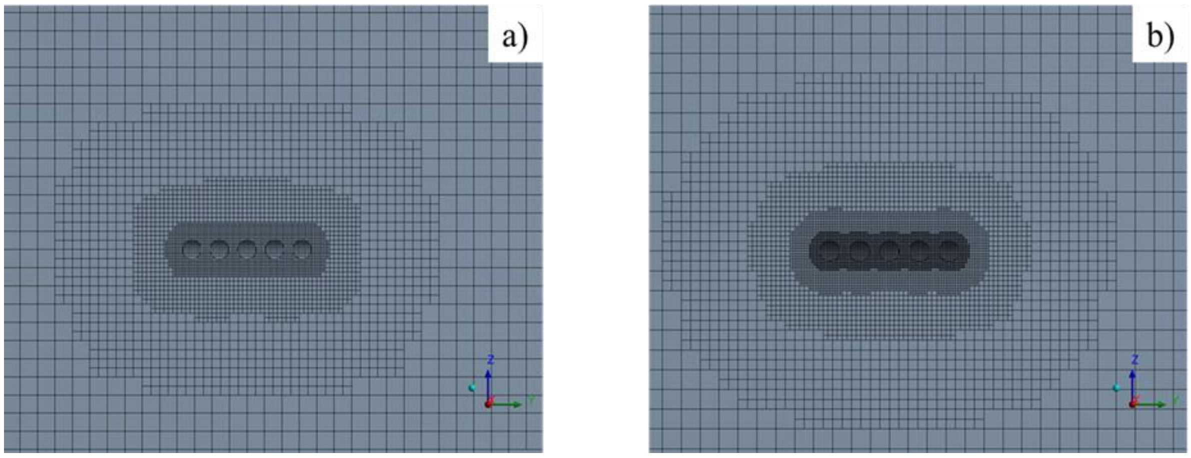


Figure 6.13: Detailed view in the YZ-plane of a) initial mesh size with 30 mm local sizing and b) final mesh size with 10 mm local sizing

Comparisons of the meshes are given in Table 6.5. M3 is chosen for this work because of negligible differences between M3 and M4., which is proven by the differences in Table 6.5 and the time-history shown in Figure 6.14.

Table 6.5: Comparison of computed hydrodynamic force for the different resolutions

Mesh comparison	**Maximum difference [%]**	**Avg. difference [%]**
M1 and M2	1.1	0.2
M2 and M3	53.3	13.8
M3 and M4	2.5	0.2

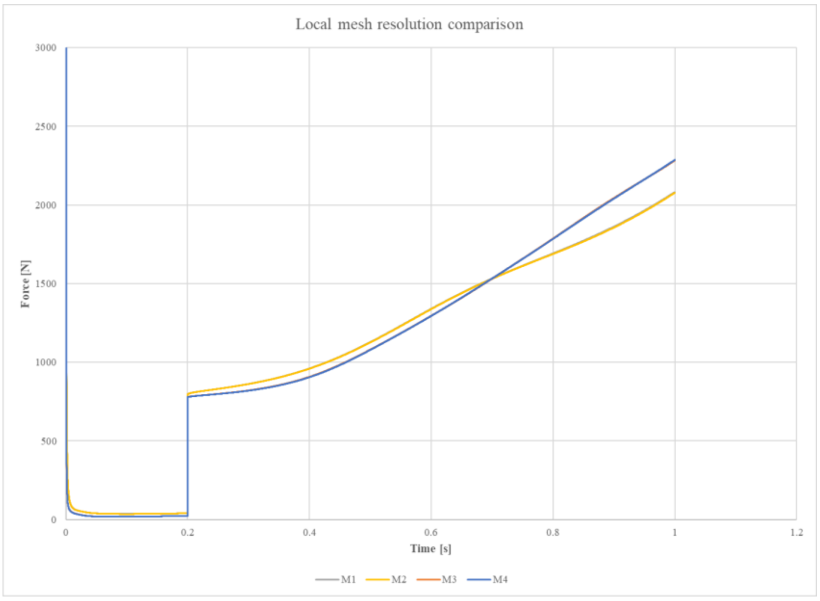


Figure 6.14: Time-history of the hydrodynamic force with different mesh resolutions. Acceleration is 1.0 m/s2 and initiated at t = 0.2 s

7 Results

This section describes the results of the analyses performed. As mentioned in section 5, added mass is evaluated for three different accelerations – 0.5 m/s^2, 1.0 m/s^2 and 2.0 m/s^2. $Co < 1$ is achieved for all analyses performed.

7.1 The submerged cylinder

The results obtained from the analyses on the submerged cylinder are used to benchmark the established methodology in accordance with DNV-RP-C205. The direction of the flow is shown in Figure 7.1. As shown in Table 7.1, the results are quite accurate compared to the DNV reference.

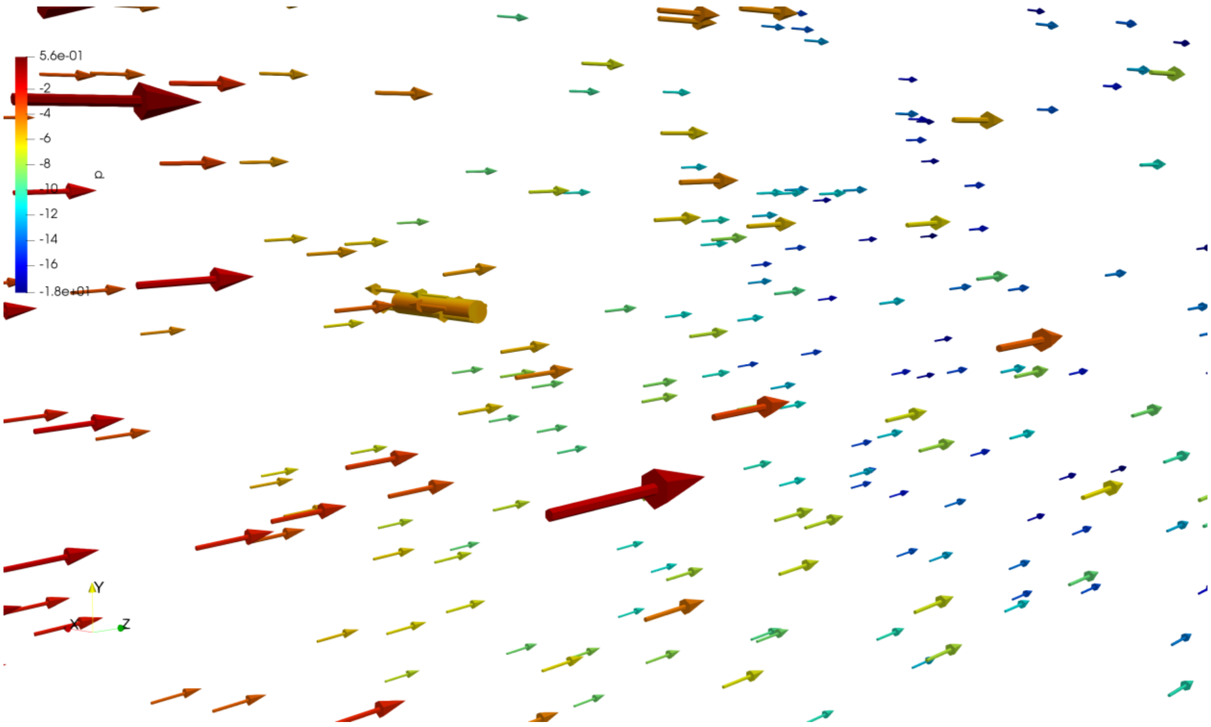


Figure 7.1: Velocity vector glyphs with kinematic pressure contours applied to the submerged cylinder at t = 1.0 s for 1.0 m/s^2 flow acceleration

Table 7.1: Computed added mass for the submerged cylinder

Acceleration [m/s^2]	Hydrodynamic mass [kg]	Displaced mass [kg]	Added mass	DNV-RP-C205	Deviation [%]
0.5	43.2	22.9	20.4	20.9	2.6
1.0	43.2		20.4		2.6
2.0	43.2		20.4		2.6

The results indicate that the methodology used can accurately determine added mass using CFD. All force components are shown in Figure 7.2, with each of the components plotted separately in Figure 7.3 to Figure 7.5. The plots of the X- and Y-components indicate that the geometric symmetry condition is satisfied for the submerged cylinder.

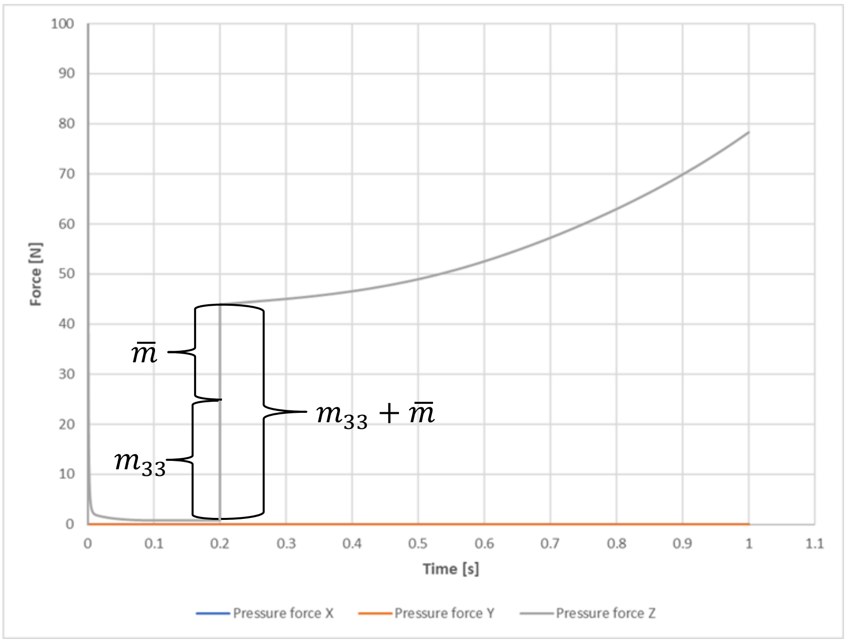


Figure 7.2: Evolution of the pressure force [N] at 1.0 m/s^2 flow acceleration. Acceleration initiated at t = 0.2 s. Added mass and Froude-Kriloff forces are marked

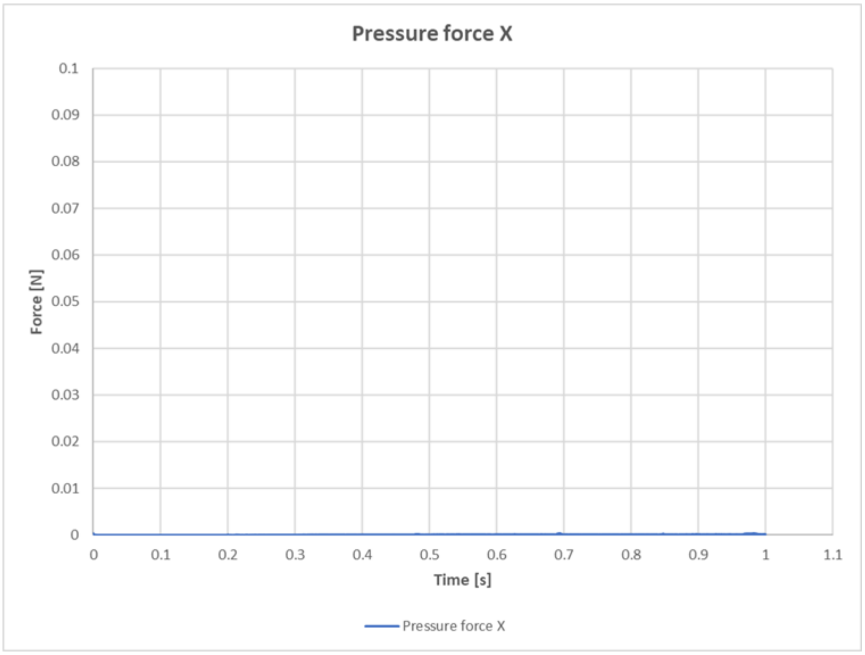


Figure 7.3: X-component of the pressure force [N] at 1.0 m/s^2 flow acceleration. Acceleration initiated at t = 0.2 s

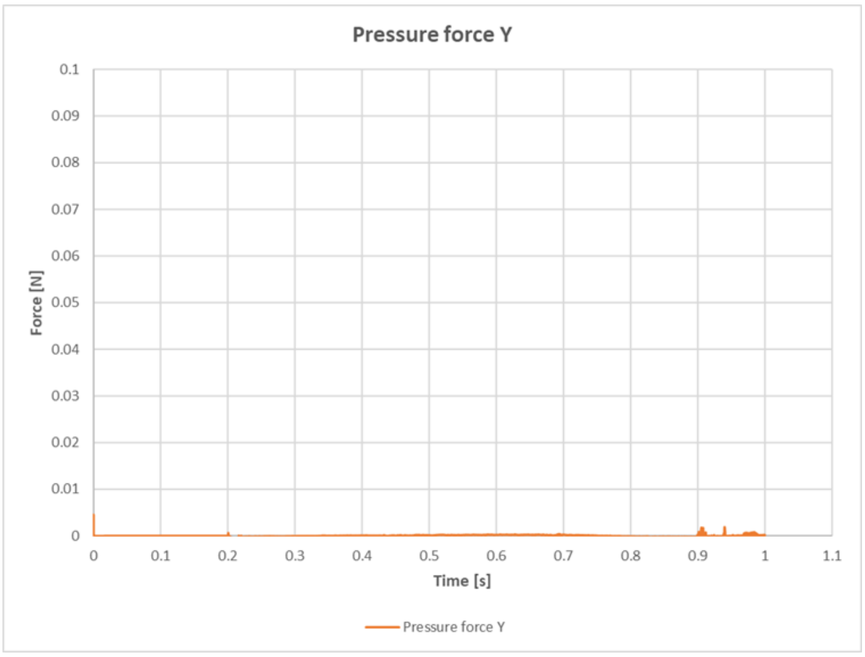


Figure 7.4: Y-component of the pressure force [N] at 1.0 m/s^2 flow acceleration. Acceleration initiated at t = 0.2 s

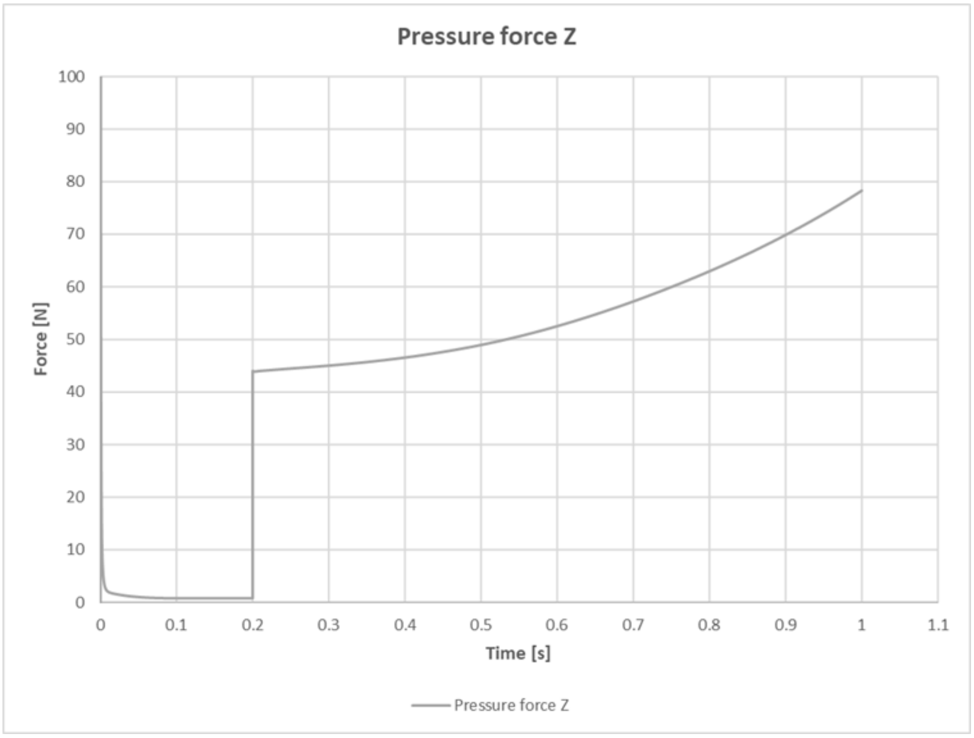


Figure 7.5: Z-component of the pressure force [N] at 1.0 m/s^2 flow acceleration. Acceleration initiated at t = 0.2 s

The kinematic pressure contours for flow acceleration of 1.0 m/s^2 are shown in Figure 7.6 at three different time-steps.

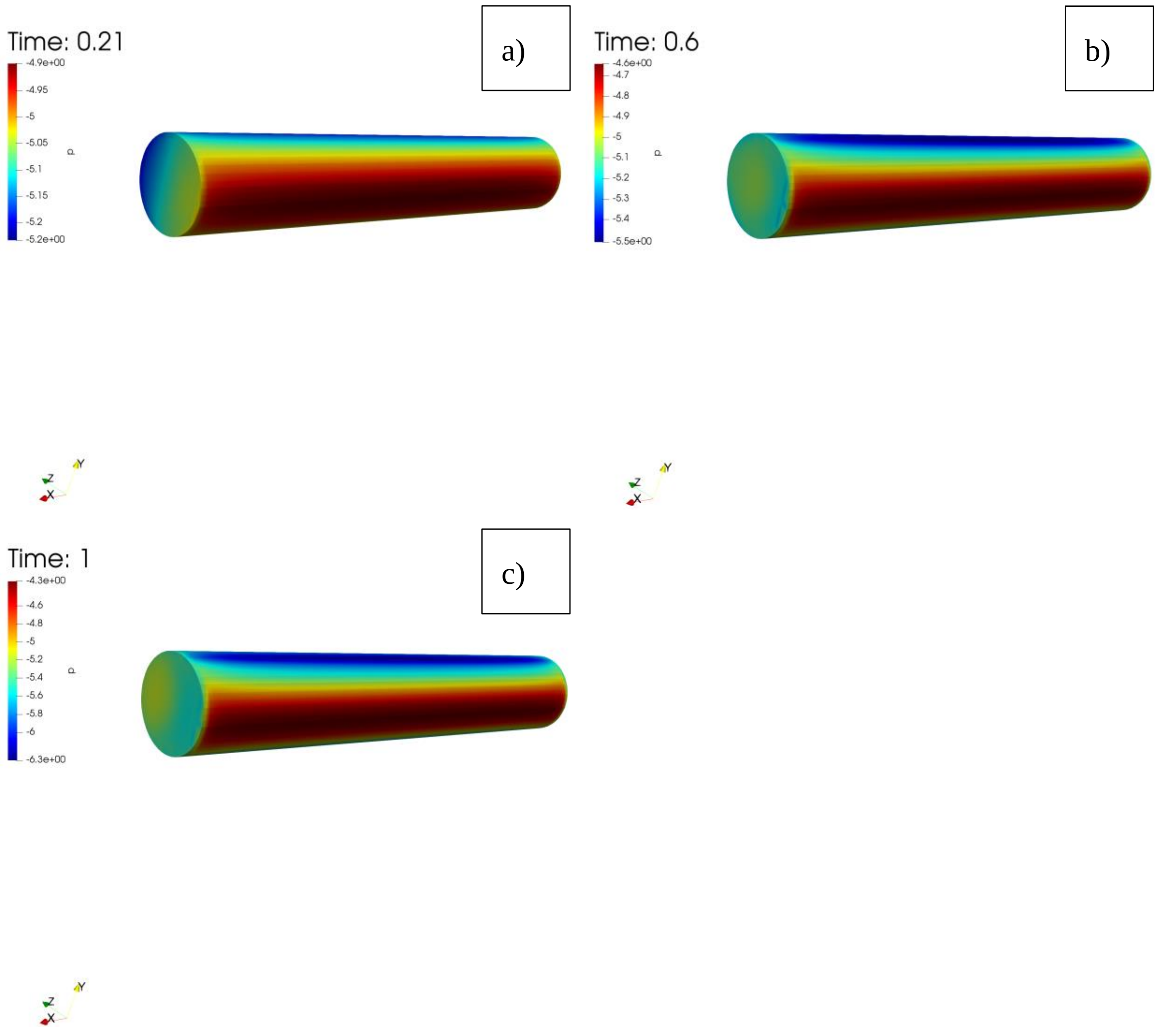


Figure 7.6: Kinematic pressure contours [m^2/s^2] over submerged cylinder at a) t = 0.21 s, b) = 0.6 s, and c) t = 1.0 s. Acceleration is 1.0 m/s^2

7.2 The cylindrical frame

This section presents the results for the cylindrical frame, which consists of the size-independent fluid domain, time-stepping, and mesh resolution. The direction of the flow is indicated by Figure 7.7.

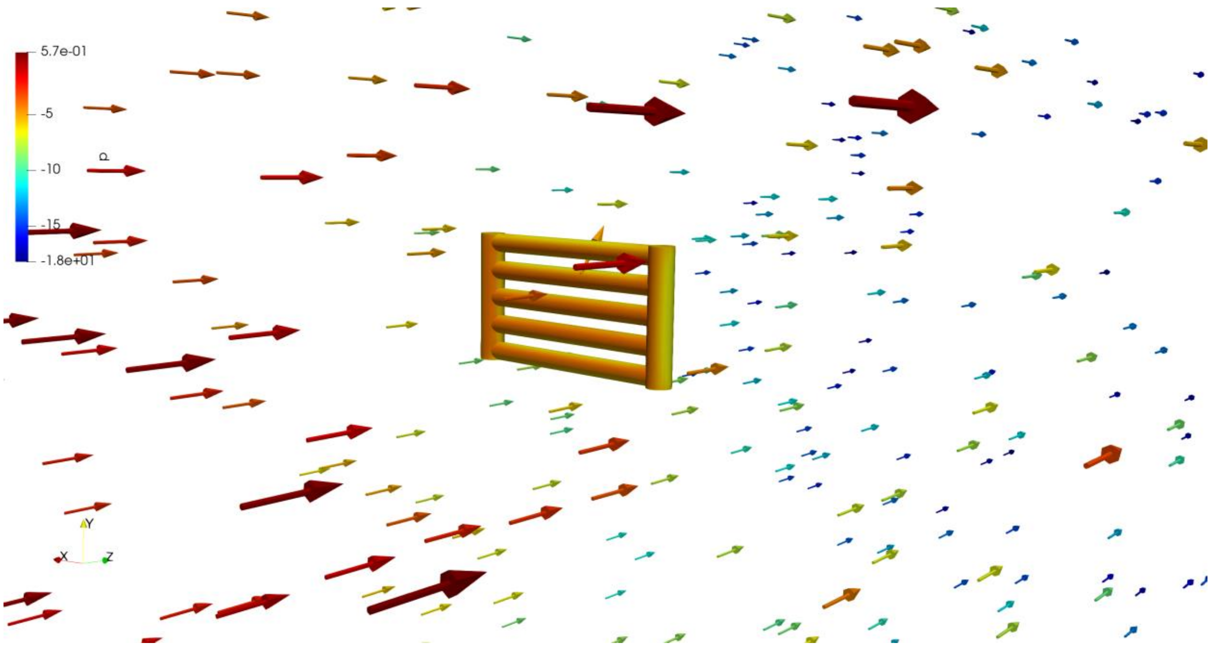


Figure 7.7: Velocity vector glyphs with kinematic pressure contours applied to the cylindrical frame at t = 1.0 s for 1.0 m/s^2 flow acceleration

7.2.1 Added mass

Newton's 2nd law can compute added mass and inertia forces; hence, there should be close to zero difference in the results of the computed added mass by varying the acceleration. Also, since the time-stepping is small, the relative contribution from the drag force after initiating acceleration is neglected. Table 7.2 gives the computed added mass for the three accelerations. As can be seen from the table, the difference in the computed results is relatively small.

Table 7.2: Computed added mass for the cylindrical frame

Acceleration [m/s^2]	Mass force [N]	Hydrodynamic mass [kg]	Displaced mass [kg]	Added mass [kg]
0.5	379.1	758.2	294.1	464.1
1	758.2	758.2		464.1
2	1516.3	758.2		464.1

These results are compared to the two methods described in section 5.5. The first method evaluates the total added mass as the sum of the added mass contributions from each cylinder in accordance with Table D-2 in DNV-RP-C205. The essential input for Equation (5.3) is listed in Table 7.3.

Table 7.3: Required input for Equation (5.3). Subscript "1" denotes the Ø 168.3 mm cylinders, and subscript "2" denote the Ø 219.1 mm cylinders

Description	Symbol	Value	Unit
Diameter of smallest cylinder	D_1	0.2	m
Length of smallest cylinder	L_1	1.78	m
Aspect ratio	L_1/D_1	10.6	-
Added mass coefficient of smallest cylinder	C_{a1}	1.0	-
Diameter of largest cylinder	D_2	0.2	m
Length of largest cylinder	L_2	1.2	m
Aspect ratio	L_2/D_2	5.3	-
Added mass coefficient of largest cylinder	C_{a2}	0.9	-

Now, for the Ø 168.3 mm diameter cylinders:

$$m_a = 1027.6[\frac{kg}{m^3}] \cdot 1 \cdot \frac{\pi}{4} \cdot (0.2[m])^2 \cdot 1.8[m] \cdot 5 = 195[kg]$$

And for the Ø 219.1 mm diameter cylinders:

$$m_a = 1027.6[\frac{kg}{m^3}] \cdot 0.9 \cdot \frac{\pi}{4} \cdot (0.21[m])^2 \cdot 1.2[m] \cdot 2 = 82[kg]$$

This gives a total added mass of 277 kg, which is not very close to the estimated added mass from OpenFOAM.

The second method approximates the added mass for the cylindrical frame as a perforated plate in accordance with DNV-RP-H103. All required input for the calculations is given in Table 7.4.

Table 7.4: Required input for evaluating Equation (5.4) and (5.5)

Description	Symbol	Value	Unit
Length of the cylindrical frame	b	2	m
Width of the cylindrical frame	a	1.2	m
Length-to-width ratio	b/a	1.7	-
Projected area	A_p	2.0	m2
Projected area for a solid flat plate	A_s	2.3	m2
Perforation ratio	p	14	%

Firstly, Equation (5.5) is evaluated for a flat plate without perforation:

$$A_{33,s} = 1027.6\left[\frac{kg}{m^3}\right] \cdot 0.7 \cdot \frac{\pi}{4} \cdot (1.2[m])^2 \cdot 2[m] = 1586[kg]$$

Now, evaluating Equation (5.4) to give added mass for a perforated plate gives:

$$A_{33} = 1586[kg] \cdot \left(0.7 + 0.3 \cdot \cos\left[\frac{\pi(14-5)}{34}\right]\right) = 1432[kg]$$

Which, compared to OpenFOAM, is a severe overestimation of the added mass. Neither of the proposed methods comply well with the numerical results.

Figure 7.8 shows the hydrodynamic force components at 1.0 m/s2 flow acceleration. It is clear from the figure that the force in the Z-direction is dominating compared to the X- and Y-components.

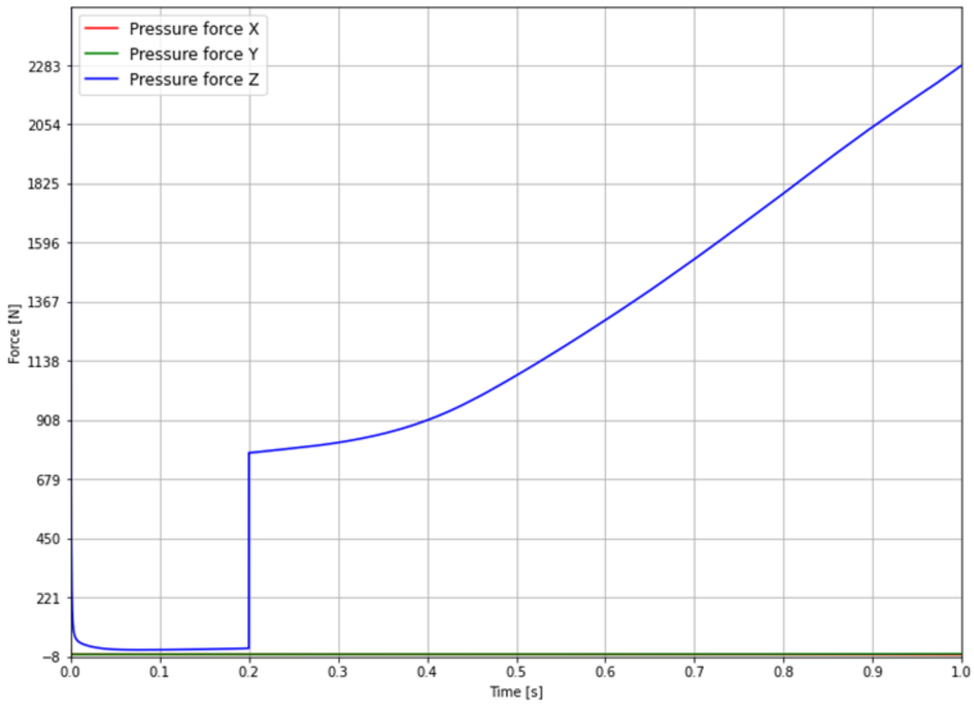


Figure 7.8: Evolution of the pressure force [N] at 1.0 m/s^2 flow acceleration. Acceleration initiated at t = 0.2 s

Each of the force components shown in Figure 7.6 to Figure 7.8 indicate the significance of the X- and Y-components compared to the Z-component of the pressure force.

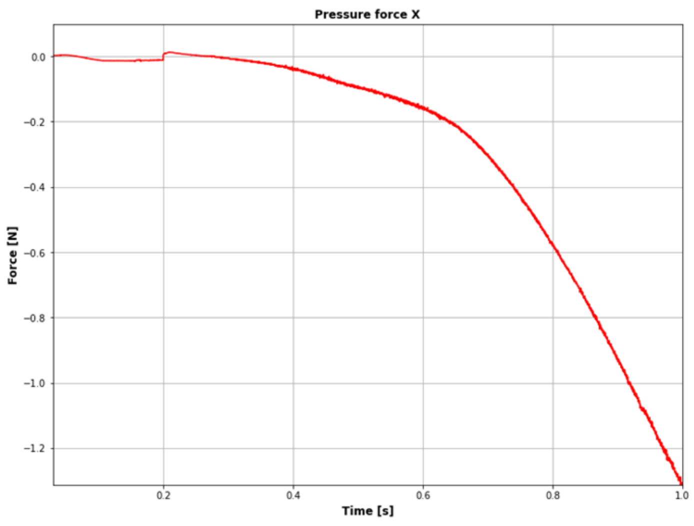


Figure 7.9: X-component of the pressure force [N] at 1.0 m/s^2 flow acceleration. Acceleration initiated at t = 0.2 s

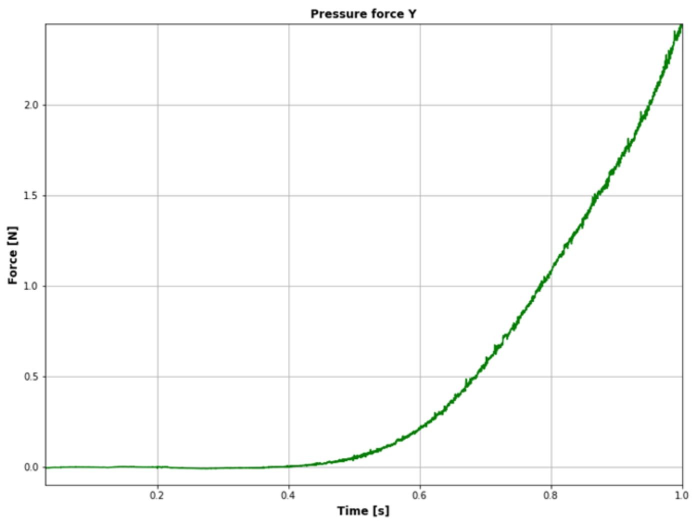


Figure 7.10: Y-component of the pressure force [N] at 1.0 m/s^2 flow acceleration. Acceleration initiated at t = 0.2 s

The pressure forces shown in Figure 7.9 and Figure 7.10 are quite small compared to the force in the Z-direction. At a final velocity of 1.0 m/s and 1.0 m/s2 flow acceleration, the values of the forces in the X- and Y-directions are -1.3 N and 2.5 N, respectively.

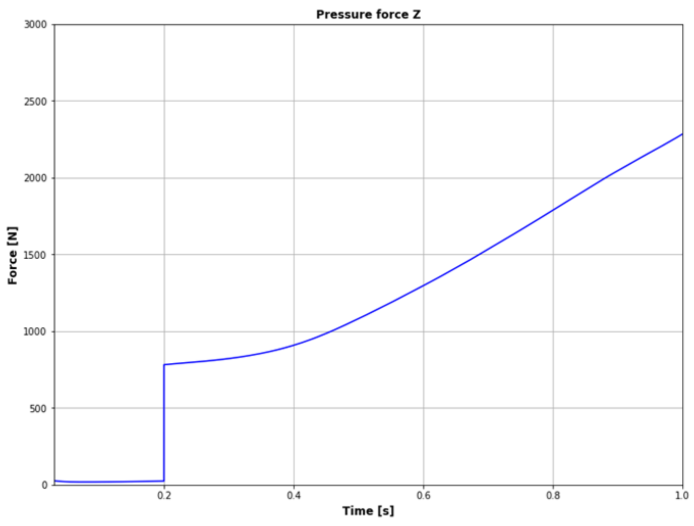


Figure 7.11: Z-component of the pressure force [N] at 1.0 m/s^2 flow acceleration. Acceleration initiated at t = 0.2 s

Pressure contours around the cylindrical frame are displayed in Figure 7.12 at three points in time for a constant acceleration of 1.0 m/s^2. The first time is before the initiation of acceleration, the second is halfway, and the last contour plot is when the velocity has reached 1.0 m/s.

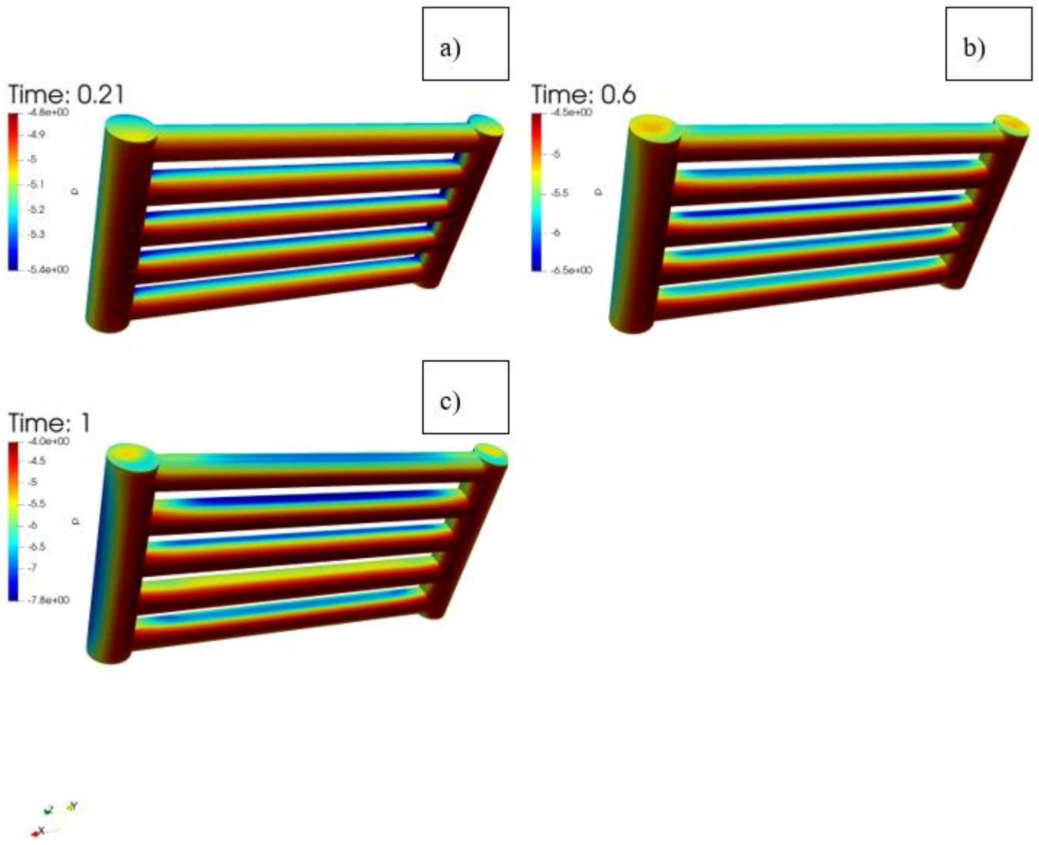


Figure 7.12: Kinematic pressure contours [m^2/s^2] over the cylindrical frame at a) t = 0.21 s, b) = 0.6 s, and c) t= 1.0 s. Acceleration is 1.0 m/s^2

The same three accelerations are analyzed for the second configuration of the cylindrical frame. The results from OpenFOAM are listed in Table 7.5.

Table 7.5: Numerical added mass results for the second configuration of the cylindrical frame

Acceleration [m/s^2]	Mass force [N]	Hydrodynamic mass [kg]	Displaced mass [kg]	Added mass [kg]
0.5	228.7	457.5	213.9	243.6
1	457.5	457.5		243.6
2	915.0	457.5		243.6

Further, evaluating Equation (5.3) gives a total added mass of 199 kg, whereas Equation (5.4) estimates 551 kg in accordance with the two manual calculation methods described in section 5.5.

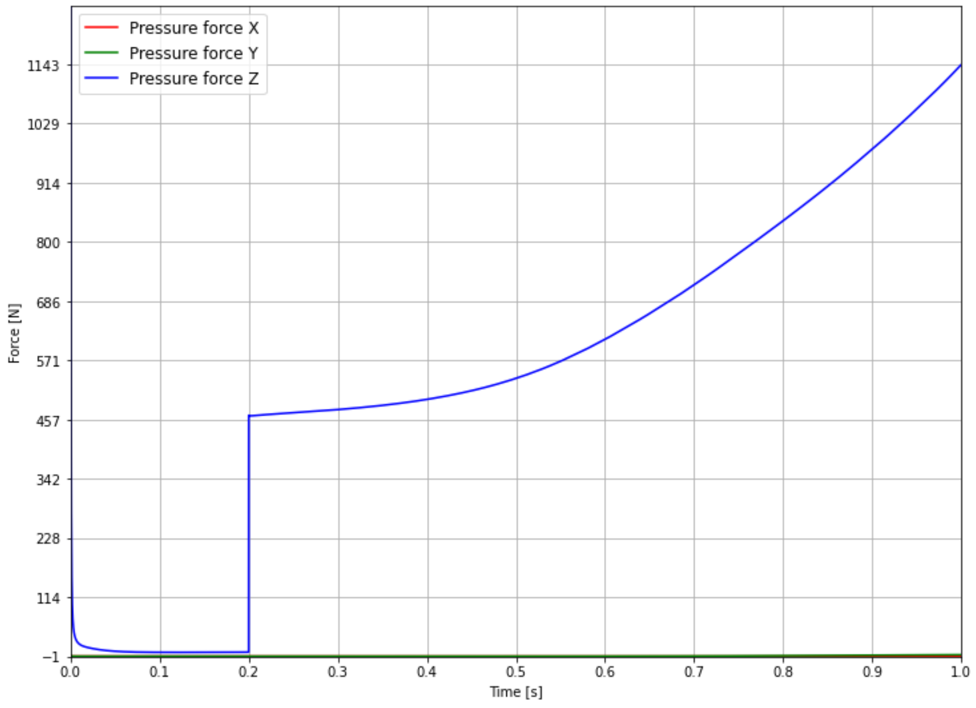


Figure 7.13: Evolution of the pressure force [N] for the second configuration of the cylindrical frame at 1.0 m/s^2 flow acceleration. Acceleration initiated at t = 0.2 s

7.2.2 Drag coefficient

Figure 7.14 shows how the drag coefficient changes with increasing freestream velocity. The drag coefficients approach a steady value towards the end of the analyses. Only minor differences in the drag coefficients are detected for the investigated freestream velocities.

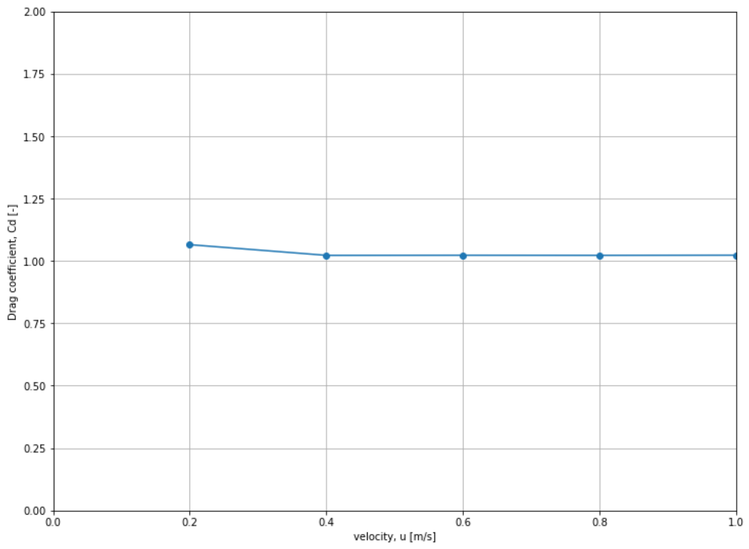
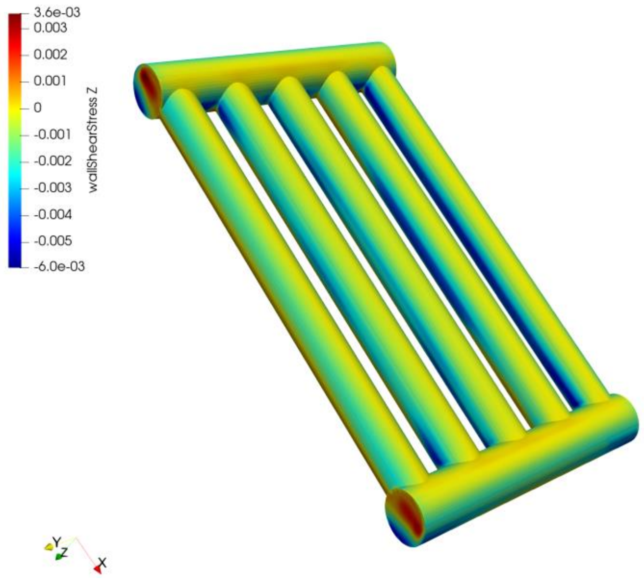


Figure 7.14: Drag coefficient evaluated at different freestream velocities

Figure 7.15 shows the velocity field (in the downstream direction) at a freestream velocity of 1.0 m/s, and the wall shear stress. As seen from the figure, there is a significant increase in the velocity in the open area between the cylinders.

a)



b)

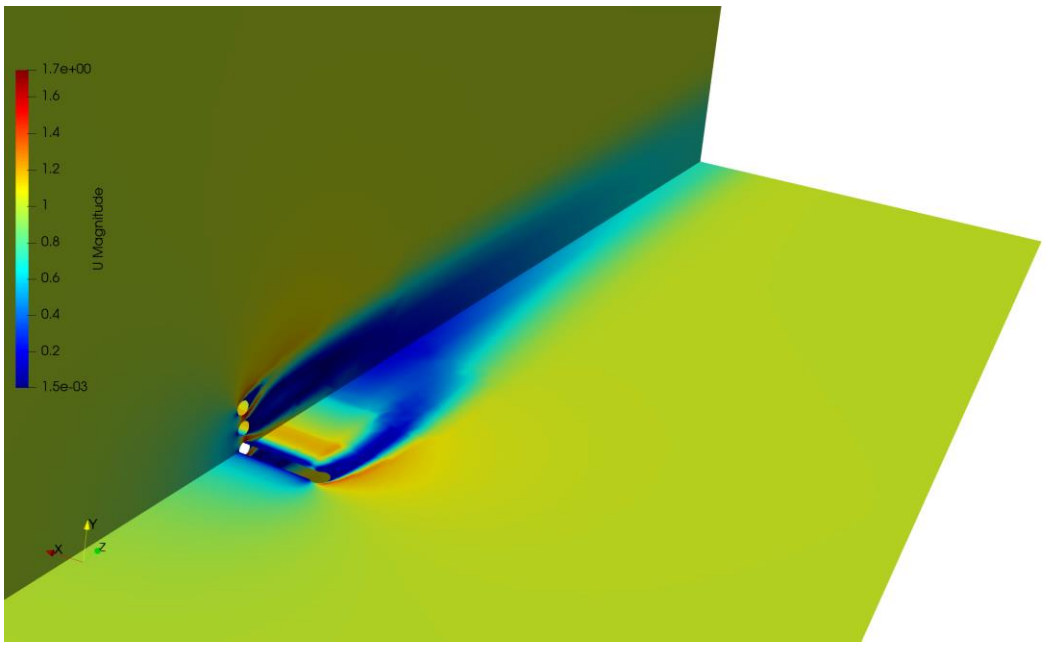


Figure 7.15: a) wall shear stress [m^2/s^2] in the flow direction b) velocity field [m/s] in the XZ- and YZ-planes at a constant freestream velocity of 1.0 m/s

8 Discussion

Added mass occurs from acceleration, either of a body submerged in water or by an acceleration of the fluid. The effect is related to the kinetic energy of the fluid required to accelerate the body through the fluid. The energy required to displace the fluid is dependent on the geometry and complexity of the object. By comparison with DNV-RP-C205 for an infinite circular cylinder, the added mass equals the mass of the displaced fluid, while for an infinite squared cylinder, added mass is 51 % higher.

If slender body/strip theory can be applied, added mass in the sway and heave directions can be computed as the sum of added mass for thin slices over the length of the body. However, this method lacks an approximation for the computation of added mass in the longitudinal direction of the body. Also, it cannot be applied at the start and end positions for many applications, as the usage criteria are not satisfied.

By studying the object's geometry, the added mass matrix can be considerably reduced if the object is symmetric about either of the primary planes.

Several methods are applicable to estimate the added mass of an object. To the author’s knowledge, experimental procedures are more common than, i.e., numerical estimates among researchers. If numerical estimations are applied, the results are typically verified with experiments as well. If performed correctly, experimental procedures give quite accurate results, which can verify any numerical simulations performed. It can be challenging to achieve accurate results during a numerical simulation, as the results are highly dependent on the discretization methods and resolution of the computational grid. However, by performing experiments in context with numerical methods, the required grid resolution and assumptions regarding the analysis can be mapped.

Fluid-structure interactions are implemented to predict the hydrodynamic forces on an object more accurately, as this accounts for the mechanical flexibility of the object. In conjunction with experimental procedures, FSI can be used for the numerical comparison to minimize possible deviations between results. The applicability of FSI depends on the object to be analyzed, and the analyst should evaluate the necessity of using the analysis coupling method. For a flat plate, where the thickness is much smaller than any other dimension, some noticeable deflection likely occurs. FSI can be employed in such scenarios to predict the hydrodynamic forces with greater accuracy when compared with experiments. If, on the other hand, the plate is thick, employing FSI would likely result in increased computational time and resources.

DNV provides, through DNV-RP-C205 and DNV-RP-H103, some rough estimates of added mass for simple constructions. Typical subsea constructions are composed of several different components of different sizes. Neither of the RPs accounts for this issue, and it is up to the analyzing engineer to determine satisfactory estimations.

As mentioned in section 2.8.4, there are no definitive acceptance criteria when evaluating added mass for a construction. The limiting value is dependent on the design criteria defined in the respective design codes for the application.

The $k - \omega\, SST$ turbulence model was developed by Menter to better predict wall shear stresses in adverse pressure gradient flows. While the original $k - \omega$ model behaves better close to the wall, it is quite sensitive to the freestream value of ω, and can become unstable if the freestream value is too high. The standard $k - \varepsilon$ model is better at correctly estimating freestream flow behavior but can give poor estimates in near-wall regions. Menter suggested a new turbulence model that employs the best features of both turbulence models. There are, of course, other turbulence models that can be applied in adverse pressure gradient flows. However, the $k - \omega\, SST$ model is extensively used in the industry due to its high applicability. Moderate computational resources are also required compared to more complex turbulence models or even other turbulence modeling techniques, such as LES or DNS.

The near-wall treatment of turbulence is essential for accurate predictions of the properties in these regions. Employing wall functions reduces the mesh resolution required. In general, it is difficult to achieve a uniform value of y^+, but performing some iterations with different resolutions can make significant changes in the y^+ value and thereby the results.

The discretization schemes give accurate and stable results from the analyses. Linear upwind differencing can become unbounded for highly convective flows, but at low freestream velocities, this does not occur. If the flow was highly convective, one could use, e.g., first-order upwind differencing to enforce a boundedness. Noteworthy is that the first-order upwind scheme is less accurate than linear upwind differncing. The surface normal gradient schemes apply corrections for non-orthogonality. Non-orthogonality has been investigated in OpenFOAM for the created meshes but is relatively low but higher than the recommended minimum limit to neglect non-orthogonal corrections. For a poorly constructed mesh with high non-orthogonality, convergence can be difficult, if not impossible, to achieve. In such scenarios, the analyst should reconsider the meshing techniques used and evaluate other methods of constructing the mesh.

The mesh for the cylindrical frame uses hexahedral cells using the “CutCell” method in ANSYS Meshing. With the current licensing setup available at Stressman Engineering AS CutCell is the most efficient method currently available. If additional licenses become available, polyhedral cells with inflation layers around the object could increase the analyses' computational efficiency.

8.1 The submerged cylinder

The results for the simple geometry investigated estimates added mass within an error margin of 3 % when comparing the results to DNV-RP-C205, which indicates that the established method is well-suited for further calculations. Ideally, for a simple geometry, there should be zero error. The error can be the result of the local surface refinement used. Increasing the local mesh resolution could increase the accuracy of the model.

The difference in the calculated added mass is of the order of 10^{-5} for the investigated accelerations. Noteworthy is that only small acceleration values are investigated, as subsea operations likely accelerate the structures within this range to ensure a safe and stable operation. Higher acceleration values could have been studied to verify the general stability and applicability of the methodology.

Increasing the numerical efficiency can be achieved by changing the cell type, e.g., to polyhedral cells. Polyhedral cells would have been a preferable solution for meshing the numerical models, as the total cell count decreases while the number of faces/interfaces per cell increases.

8.2 The cylindrical frame

Three independence studies are performed for the cylindrical frame. The choices made are based on marginal differences between refinement iterations. It could be argued that the models need higher accuracy; however, there are significant differences in computational time between the most refined analysis setups and the chosen parameters.

Only minor differences in added mass are observed for the accelerations investigated. On the one hand, this indicates that the methodology is stable and accurately predicts added mass for the cylindrical frame. On the other hand, the results cannot be verified without performing experiments that evaluate the hydrodynamic forces on the structure. While verification has been established through different analysis parameters, these results are yet to be verified with experimental data.

The added mass analyses took only hours to finish. The most time-consuming part of the analyses arises from drag coefficient estimation. Each of the analyses took more than 30 hours to complete with the resources available.

It can be argued that the turbulence intensity is set too high for the cylindrical frame. As described previously, a change of 4 % in the intensity does not give any significant difference in estimated added mass. However, for evaluation of the drag coefficient, the results can be somewhat different. The sensitivity can also be geometrically dependent and is yet to be determined for other complex geometrical constructions.

The computed forces in the X- and Y-directions deviate from zero. The force values are not of comparable magnitude with the Z-component; however, these deviations can be caused by, e.g., the resolution of the grid since the fluid domain is not perfectly symmetric with the meshing parameters used. The numerical discretization of the domain can be of some influence on the results. Also, the Y-force component is not perfectly smooth throughout the analysis. The cause of this can occur from flow interaction between the cylinders due to little spacing between the components.

Comparing the results with DNV-RP-C205 and DNV-RP-H103 shows that the structure is not well represented by either method 1 or method 2 described in section 5.5. Hence, it is likely that simplified estimates cannot predict the hydrodynamic loads for complex constructions, as one approximation underestimates added mass, while the other method severely overestimates the effect.

However, when removing two of the inner cylinders, the numerical results comply better with DNV-RP-C205 when computing the added mass for each individual cylinder and summing the contributions. It is evident that the spacing of the cylinders is of significant influence when evaluating added mass for such geometries.

The pressure and velocity field contour plots indicate that the flow is separating from the surface of the object, in which case the $k - \omega\ SST$ model is better suited than the $k - \varepsilon$ model.

9 Conclusion

Analyses on simple geometries show that numerical added mass estimations with CFD can be used with high accuracy. From a practical engineering perspective, the observed differences between CFD and DNV-RP-C205 are negligible.

It was shown that recommended procedures fail at predicting the hydrodynamic loads of complex objects. Recommended procedures by DNV are unable to account for the interaction of different components in construction assemblies. In such scenarios, the engineer should use available resources to achieve more accurate estimates of the hydrodynamic loads on the constructions.

The sensitivity to turbulence intensity has been found negligible for constant acceleration analyses. This conclusion is based on a comparison between 1 % and 4.5 % turbulence intensity.

While the added mass results for the cylindrical frame are quite similar for different accelerations, the values cannot be verified until experiments are performed for comparison. If performed correctly, the CFD analyses should yield results that are comparable with experiments.

From the analyses on the cylindrical frame, it was observed that neither of the RPs by DNV provides accurate results for added mass. The codes do not account for flow interaction caused by the components, which was seen to be of significant influence.

9.1 Way forward

There are many exciting aspects in the field of hydrodynamics. The recommendations for further work are summarized from discussions with Stressman Engineering AS and the supervisor.

- Verify numerical results for the cylindrical frame with experimental procedures if possible. Stressman Engineering AS will be in contact with industry partners and arrange a meeting regarding this issue.
- When the numerical model has been validated, establish a computational methodology to evaluate the entire added mass matrix for complex objects. Evaluation of the entire added mass matrix includes both rectilinear and angular accelerations.
- Extend the computational model to evaluate added mass for oscillating motions.
- Perform CFD analyses to compute wave slamming loads on the structure when the structure is lowered through the surface of the water.
- When possible, include fluid-structure interactions in the models.

References

[1] V. Balasubramaniyan, *Understanding offshore lifting operations and engineering analysis*, 2020. [Online]. Available at: https://www.marineinsight.com/offshore/offshore-lifting-operations-and-engineering-analysis/. Accessed: 24.04.21.

[2] T. I. Fosse, “Modelling of Marine Vehicles,” in *Guidance and control of ocean vehicles*. University of Trondheim, Norway: John Wiley & Sons, 1994, pp. 5-57.

[3] T. Perez, M. Blanke, “Simulation of Ship Motion in Seaway,” EE02037, 19th Jan 2015.

[4] F. H. Imlay, “The complete expressions for added mass of a rigid body moving in an ideal fluid,” Armed Services Technical Information Agency, Virginia, USA, 1528, July. 1961.

[5] T. I. Fossen, “Maneuvering Theory,” in *Handbook of marine craft hydrodynamics and motion control,* Norwegian University of Science and Technology, Norway: John Wiley & Sons, 2014, pp. 109-133.

[6] O. A. Eidsvik, “Identification of hydrodynamic parameters for remotely operated vehicles,” Master’s thesis, Institute of marine technology, Norwegian University of Science and Technology, Trondheim, 2015.

[7] J. Severholt, “Generic 6-DOF added mass formulation for arbitrary underwater vehicles based on existing semi-empirical methods,” Master’s thesis, Naval architecture, KTH Royal Institute of Technology, Sweden, 2017.

[8] F. Grønstad, “Modelling and simulation of underwater vehicle-manipulator system,” Master’s thesis, Marine Cybernetics, Norwegian University of Science and Technology, Trondheim, 2019.

[9] F. P. Incropera, D. P. Dewitt, T. L. Bergman, A. S. Lavine, “Introduction to convection,” in *Principles of heat and mass transfer,* seventh edition, Singapore: John Wiley & Sons, 2013, pp. 377-433.

[10] B. Reisfeld, *Fluids and fluid flow,* 2021. Available at: https://www.engr.colostate.edu/CBE101/topics/fluids_and_fluid_flow.html. Accessed: 09.04.21.

[11] O. M. Faltinsen, “Current and wind loads,” in *Sea loads on ships and offshore structures,* United Kingdom: Cambridge University Press, 1990, pp. 174-223.

[12] T. I. Fossen, “Environmental disturbances,” in *Guidance and control of ocean vehicles*. University of Trondheim, Norway: John Wiley & Sons, 1994, pp. 57-93.

[13] J. R. Morison, M. P. O’Brien, J. W. Johnson, S. A. Chaaf, ”The force exerted by surface waves on piles,” in *J Pet Technol 2, AIME,* vol. 189, 149-154, May, 1950. Doi: https://doi.org/10.2118/950149-G

[14] J. N. Newman, “Model testing,” in *Marine Hydrodynamics,* 40th anniversary edition, Cambridge, MA: The MIT Press, 2017, pp. 9-55.

[15] Y. H. Eng, G. Seet, M. Lau, C. S. Chin, "Estimation of Hydrodynamic Coefficients of an ROV using Free Decay Pendulum Motion," *Engineering Letters*, 16, 329-342, Aug.

2008, EL_16_3_09. URL: https://www.researchgate.net/publication/224282758_Estimation_of_the_Hydrodynamics_Coefficients_of_an_ROV_using_Free_Decay_Pendulum_Motion

[16] A. Balagopalan, K. N. Tiwari, P. Krishnankutty, ”Horizontal planar motion mechanism (HPMM) incorporated to the existing towing carriage for ship maneuvering studies,” RINA 5th International Conference on Ship and Offshore Technology, vol. 1, Dec. 2017. URL: https://www.researchgate.net/publication/322147054_HORIZONTAL_PLANAR_MOTION_MECHANISM_HPMM_INCORPORATED_TO_THE_EXISTING_TOWING_CARRIAGE_FOR_SHIP_MANOEUVRING_STUDIES/stats

[17] A. S. Bjørke et al., ”Study of hydrodynamic forces on complex structures,” *Proceedings of the 37th International Conference on Ocean, Offshore & Arctic Engineering*, Jun. 17-22, 2018. URL: https://ntnuopen.ntnu.no/ntnu-xmlui/bitstream/handle/11250/2591304/Bj%25C3%25B8rke%2bet%2bal.%2b-%2b2018%2b-%2bStudy%2bof%2bhydrodynamic%2bforces%2bon%2bcomplex%2bstructures.pdf?sequence=2&isAllowed=y

[18] *Environmental conditions and environmental loads*, DNV-RP-C205, Oct. 2010.

[19] Wikipedia, *Condition number*, 2021. Available at: https://en.wikipedia.org/wiki/Condition_number. Accessed: 09.04.21.

[20] Y. Grøstad, “Fluid-structure interaction study of vortex-induced vibration in thermowells,” Master’s thesis, Energy and Process Engineering, Norwegian University of Science and Technology, Trondheim, 2019.

[21] F. K. Benra, H. J. Dohmen, J. Pei, S. Schuster, B. Wan, ”A comparison of one-way and two-way coupling methods for numerical analysis of fluid-structure interactions,” *Journal of Applied Mathematics*, vol. 2011, 2011, 853560. Doi: https://doi.org/10.1155/2011/853560.

[22] G. H. Keulegan, L. H. Carpenter, "Forces on cylinders and plates in an oscillating fluid," *Journal of Research of the National Bureau of Standards*, vol. 60, no. 5, 1958, 2857. URL: https://nvlpubs.nist.gov/nistpubs/jres/60/jresv60n5p423_A1b.pdf.

[23] J. Tu. G. H. Yeoh, C. Liu, “CFD solution procedure: a beginning,” in *Computational fluid dynamics: a practical approach*, third edition, Oxford, United Kingdom: Elsevier Ltd., 2018, pp. 33-65.

[24] H. K. Versteeg, W. Malalasekera, "Turbulence and its modeling," in *An introduction to computational fluid dynamics*, second edition. Harlow, England: Pearson Education Limited, 2007, pp. 40-115.

[25] CFD Support, *Laminar vs. Turbulent Flow*. Available at: https://www.cfdsupport.com/OpenFOAM-Training-by-CFD-Support/node334.html. Accessed: 02.03.21.

[26] J. Tu, G. H. Yeoh, C. Liu, ”Governing equations for CFD fundamentals,” in *Computational fluid dynamics: a practical approach*, third edition, Oxford, United Kingdom: Elsevier Ltd., 2018, pp. 65-125.

[27] J. Blazek, “Governing equations,” in *Computational fluid dynamics: principles and applications*, second edition, Oxford, United Kingdom: Elsevier Ltd, 2007, pp. 5-29.

[28] F. Menter, M. Kuntz, RB. Langtry, “Ten years of industrial experience with the SST turbulence model,” *Heat and mass transfer*, 4, 2003. URL: https://www.researchgate.net/publication/228742295_Ten_years_of_industrial_experience_with_the_SST_turbulence_model#fullTextFileContent.

[29] O. Gulinsky, A. Rogatkin, V. Vorobyov, ”Singularity MAXI: a comparison of CFD and tank test results,” presented at 5th High Performance Yacht Design Conference, Auckland, USA, 2015.

[30] H. K. Versteeg, W. Malalasekera, “Solution algorithms for pressure-velocity coupling in steady flows,” in *An introduction to computational fluid dynamics*, second edition. Harlow, England: Pearson Education Limited, 2007, pp. 179–212.

[31] OpenCFD Ltd, Euler implicit time scheme, 2016. Accessed: 24.03.21. Available at: https://www.openfoam.com/documentation/guides/latest/doc/guide-schemes-time-euler.html.

[32] H. K. Versteeg, W. Malalasekera, “The finite volume method for convection-diffusion problems,” in An introduction to computational fluid dynamics, second edition. Harlow, England: Pearson Education Limited, 2007, pp. 134-179.

[33] J. Tu, G. H. Yeoh, C. Liu, ”CFD techniques: the basics,” in Computational fluid dynamics: a practical approach, third edition. Oxford, United Kingdom: Elsevier Ltd, 2018, pp. 155-211.

[34] Wolf dynamics, “Finite volume method: a crash introduction,” course. [Online]. Accessed: 01.04.21. Available at: http://www.wolfdynamics.com/wiki/fvm_crash_intro.pdf.

[35] M. Ishigaki, S. Abe, Y. Sibamoto, T. Yonomoto, “Influence of non-orthogonality on numerical simulation of buoyant jet flows,” Nuclear Engineering and Design, 314, pp. 326-337, Feb. 7, 2017. Doi: https://doi.org/10.1016/j.nucengdes.2017.02.010.

[36] C. Hirsch, “General properties and high-resolution numerical schemes,” in *Numerical computation of internal and external flows*, second edition. Oxford, United Kingdom: Butterworth-Heinemann Ltd, 2007, pp. 337-411.

[37] C. Greenshields, OpenFOAM v6 user guide: 4.4 Numerical schemes, 2018. [Online]. Available at: https://cfd.direct/openfoam/user-guide/v6-fvschemes/. Accessed: 01.04.21.

[38] Wolf dynamics, “Tips and tricks in OpenFOAM,” course. [Online]. Available at: http://www.wolfdynamics.com/wiki/tipsandtricks.pdf. Accessed: 01.04.21.

[39] High-performance computing group, OpenFOAM – performance on Vilje. [Online]. Available at: https://www.hpc.ntnu.no/ntnu-hpc-group/vilje/user-guide/software/openfoam/openfoam-performance-on-vilje. Accessed: 01.04.21.

[40] T. Behrens, “OpenFOAM’s basic solvers for linear systems of equations,” DTU, Denmark, Feb. 18, 2009.

[41] E. F. Toro, “Notions on Numerical Methods,” in *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*, third edition, Berlin: Springer-Verlag, 2014, pp. 163-213.’

[42] J. Tu, G. H. Yeoh, C. Liu, ”CFD mesh generation: a practical guide,” in *Computational fluid dynamics: a practical approach*, third edition. Oxford, United Kingdom: Elsevier Ltd, 2018, pp. 125-155.

[43] Marcin Sosnowski et al 2018 J. Phys.: Conf. Ser. 1101 012036

[44] ANSYS INC., *ANSYS Meshing user's guide version 13.0*. 2010.

[45] OpenFOAM: User Guide v2012, *Post-processing: Forces*, 2017. [Online]. Available at: https://www.openfoam.com/documentation/guides/latest/doc/guide-fos-forces-forces.html. Accessed: 26.04.21.

[46] *Modelling and analysis of marine operations*, DNV-RP-H103, April 2009.

[47] OpenFOAM: User Guide v2012, *k-omega Shear Stress Transport (SST)*. [Online]. Available at: https://www.openfoam.com/documentation/guides/latest/doc/guide-turbulence-ras-k-omega-sst.html. Accessed: 18.05.21.

[48] OpenFOAM: User Guide v2012, *kqRWallFunction*. [Online]. Available at: https://www.openfoam.com/documentation/guides/latest/doc/guide-bcs-wall-turbulence-kqRWallFunction.html. Accessed: 18.05.21.

[49] OpenFOAM: User Guide v2012, *omegaWallFunction*. [Online]. Available at: https://www.openfoam.com/documentation/guides/latest/doc/guide-bcs-wall-turbulence-omegaWallFunction.html. Accessed: 18.05.21.

Appendices

University of South-Eastern Norway

Faculty of Technology, Natural Sciences and Maritime Sciences, Campus Porsgrunn

FMH606 Master's Thesis

Title: Hydrodynamics – Effects of added mass on complex objects

USN supervisor: Knut Vågsæther

External partner: Stressman Engineering AS, Sondre Luca Helgesen

Task background:
In the offshore and subsea industry there is a lot of uncertainty with regards to added mass of structures. DNVGL and other authorities gives rough estimates through their recommended practices and standards, but these are normally limited to simple shapes and geometries. This is thought to greatly over-dimension structures since the estimates needs to be on the conservative side. Side effect of this is that structures become heavier and more costly than necessary. One side-effect is that often bigger crane ships needs to be used.

There are many opinions among the engineers, but they are not grounded in tests nor analyses. This master thesis will, hopefully, shed more light on added mass of more complex shapes and geometries.

Task description:
The task will be focusing on understanding the effect of added mass. First through a literature search to gather already existing knowledge. Then CFD analyses based on already known results will be set up and used as benchmarks. When the CFD shows good correlation with the observed values and analysis values, more complex shapes will be investigated.

The first part will focus on constant acceleration and deceleration of objects submerged in seawater. Constant velocities will be checked to evaluate the drag of the object.

Second part, if time allows, will be to accelerate objects up and down in the water domain. This will be similar to what is happening during an earthquake event.

The main objective is to obtain a procedure of how to calculate added mass for any given object using OpenFoam.

Student category: Reserved for Sondre Kaasa

The task is suitable for online students (not present at the campus): Yes

Practical arrangements:
Sondre Kaasa will have access to Stressman Engineering's computers and licenses.

Supervision:
As a general rule, the student is entitled to 15-20 hours of supervision. This includes necessary time for the supervisor to prepare for supervision meetings (reading material to be discussed, etc).

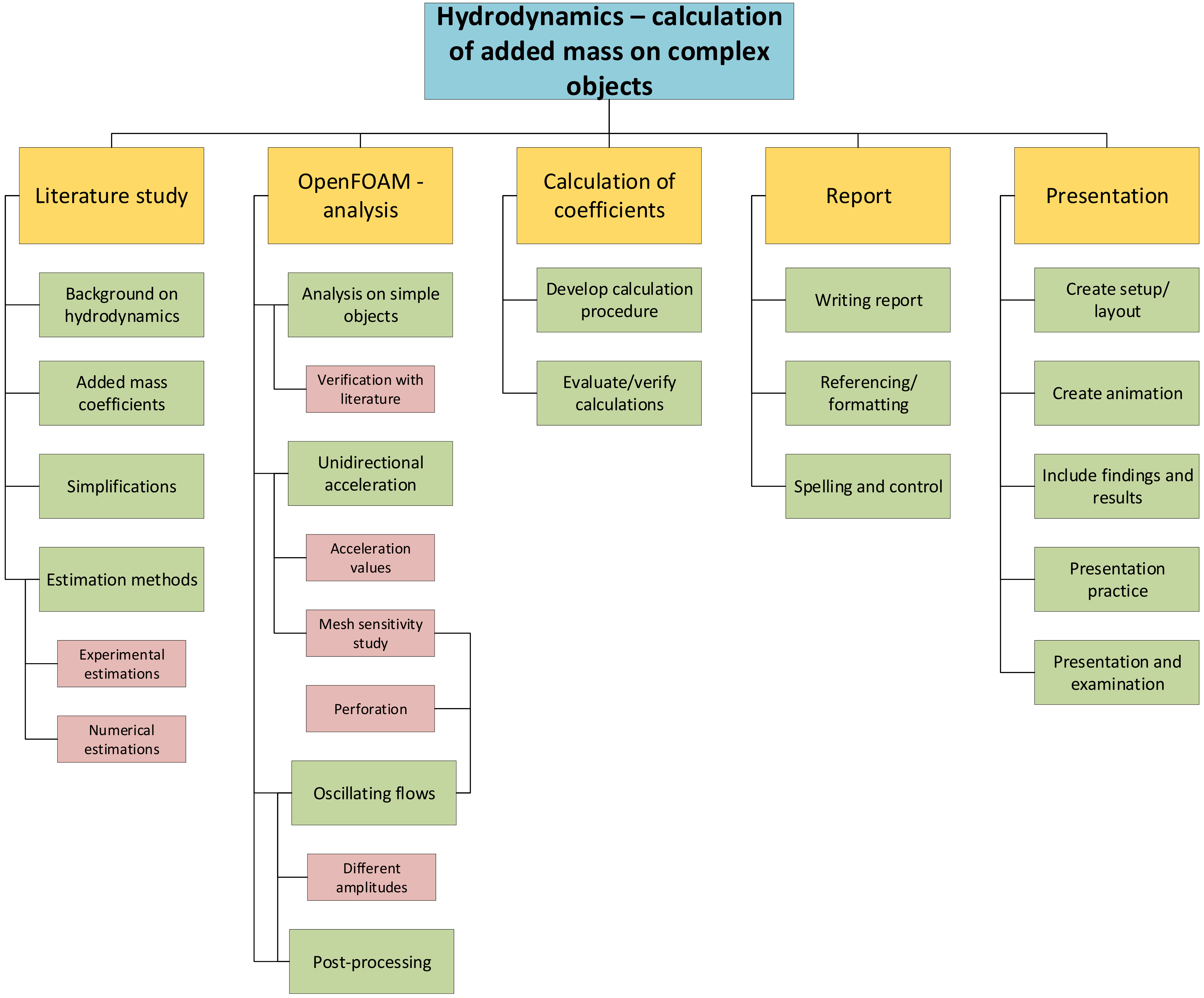
Signatures:

Supervisor (date and signature): [illegible]

Student (write clearly in all capitalized letters): SONDRE KAASA

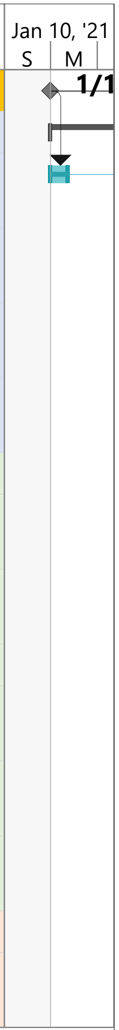
Student (date and signature): 19.03.2021 Sondre Kaasa

Appendix B - Work breakdown structure



Appendix C - Gantt diagram

ID	ℹ	Task Mode	WBS	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1	✓	➡	1	Start	0 hrs	Mon 1/11/21	Mon 1/11/21		
2	✓	➡	**2**	**Meetings**	**7.5 days**	**Mon 1/11/21**	**Wed 1/20/21**		
3	✓	📌	2.1	Meeting no. 1 with supervisor	1 hr	Mon 1/11/21	Mon 1/11/21	1	
4	✓	➡	2.2	Meeting no. 2 with supervisor	1 hr	Wed 1/20/21	Wed 1/20/21	3	
5	✓	📌	2.3	Meeting no. 3 with supervisor	1 hr	Wed 1/20/21	Wed 1/20/21	4	
6	✓	📌	2.4	1st formal meeting	2 hrs	Wed 1/20/21	Wed 1/20/21	5	
7	✓	📌	**3**	**Literature study**	**1 mon?**	**Wed 1/13/21**	**Tue 2/9/21**	**1,3**	
8	✓	➡	3.1	Background on hydrodynamics	1 wk	Wed 1/13/21	Tue 1/19/21	1,3	
9	✓	➡	3.2	Added mass coefficients	2 days	Wed 1/20/21	Thu 1/21/21	8	
10	✓	➡	3.3	Simplifications	3 days	Fri 1/22/21	Tue 1/26/21	9	
11	✓	📌	**3.4**	**Estimation methods**	**2 wks**	**Wed 1/27/21**	**Tue 2/9/21**	**8,9,10**	
12	✓	➡	3.4.1	Experimental estimates	1 wk	Thu 1/28/21	Wed 2/3/21	8,9,10	
13	✓	➡	3.4.2	Numerical estimates	1 wk	Wed 2/3/21	Tue 2/9/21	8,9,10	
14	✓	📌	**4**	**OpenFOAM analysis**	**2 mons**	**Tue 2/9/21**	**Mon 4/5/21**	**7,1**	
15	✓	📌	**4.1**	**Analysis on simple objects**	**2 wks**	**Tue 2/9/21**	**Mon 2/22/21**	**13**	



Project: Hydrodynamics - calcu
Date: Tue 5/18/21

Task	▬	Inactive Summary	▭	External Tasks	▬
Split	┈	Manual Task	▬	External Milestone	◇
Milestone	◆	Duration-only	▬	Deadline	⬇
Summary	┏━┓	Manual Summary Rollup	▬	Progress	─
Project Summary	┏━┓	Manual Summary	┏━┓	Manual Progress	─
Inactive Task	▭	Start-only	[		
Inactive Milestone	◇	Finish-only	]		

ID	ℹ	Task Mode	WBS	Task Name	Duration	Start	Finish	Predecessors	Resource Names	Jan 10, '21 S M
16	✓	📌	**4.1.1**	**Become familiar with dynamic mesh**	**1 wk**	**Tue 2/9/21**	**Mon 2/15/21**	**7**		
17	✓	➡	4.1.1.1	Accelerate object	1 wk	Tue 2/9/21	Mon 2/15/21			
18	✓	➡	4.1.2	Verification with literature	1 wk	Tue 2/9/21	Mon 2/15/21			
19	✓	📌	**4.2**	**Unidirectional acceleration**	**2 wks**	**Tue 2/9/21**	**Mon 2/22/21**	**15**		
23	✓	📌	**4.3**	**Oscillating motions**	**26 days**	**Mon 3/1/21**	**Mon 4/5/21**	**19**		
27	✓	📌	**5**	**Report**	**92 days**	**Tue 1/12/21**	**Wed 5/19/21**	**14**		
28	✓	📌	5.1	Writing report	80 days	Wed 1/13/21	Tue 5/4/21	14		
29	✓	➡	5.2	Referencing and formatting	2 wks	Thu 5/6/21	Wed 5/19/21	28		
30	✓	➡	5.3	Spelling and control	2 wks	Wed 5/5/21	Tue 5/18/21	28		
31	✓	📌	5.4	Hand-in	0 days	Wed 5/19/21	Wed 5/19/21	28,29,30		
32	✓	📌	**6**	**Presentation**	**6 days?**	**Mon 5/31/21**	**Mon 6/7/21**	**31**		
33	✓	➡	6.1	Create setup/layout	1 day	Mon 5/31/21	Mon 5/31/21	31		
34	✓	➡	6.2	Create animation	1 day	Tue 6/1/21	Tue 6/1/21	33		
35	✓	➡	6.3	Include findings and results	1 day	Wed 6/2/21	Wed 6/2/21	34		
36	✓	➡	6.4	Presentation practice	2 days	Thu 6/3/21	Sun 6/6/21	35		

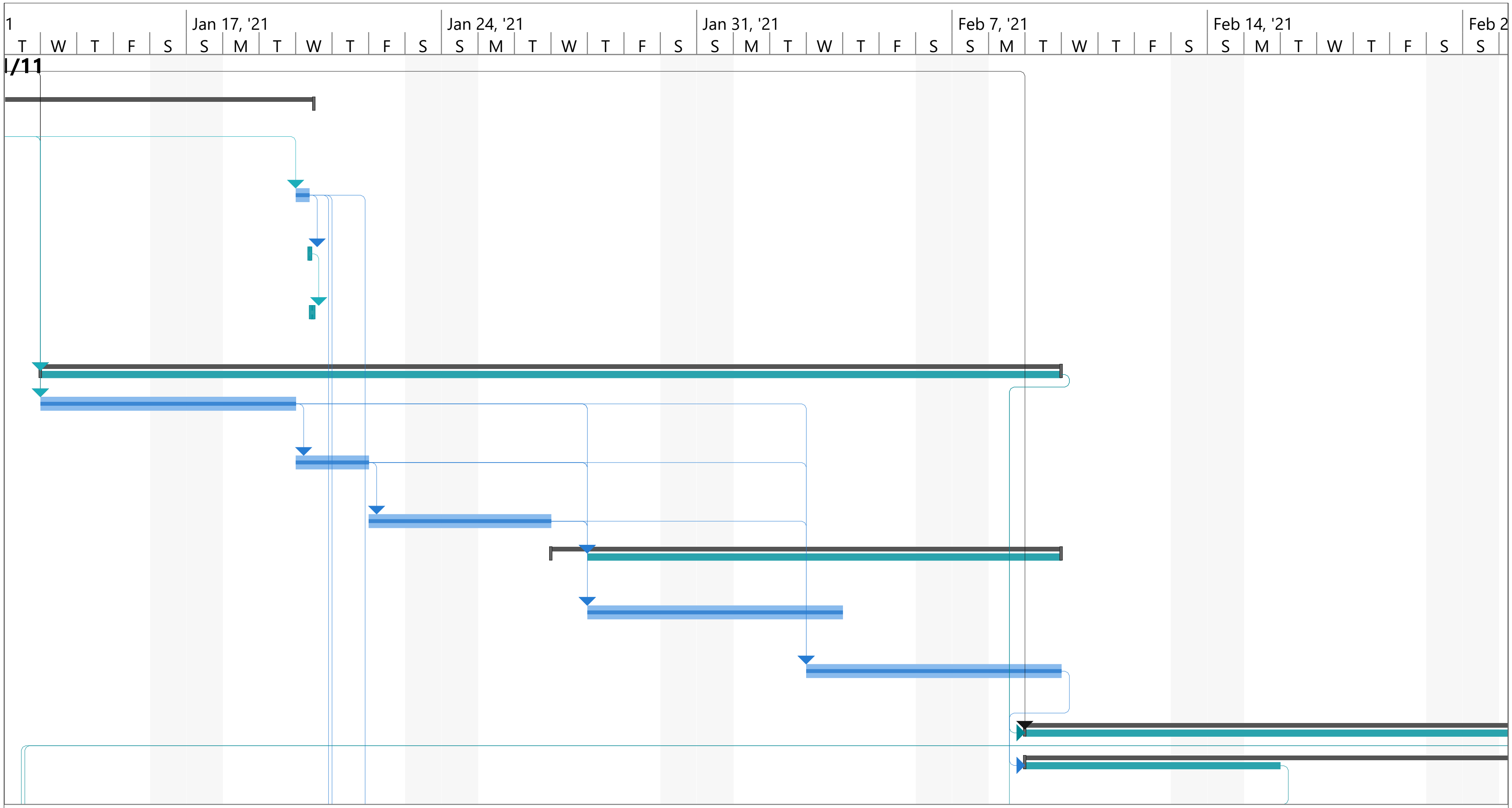
Project: Hydrodynamics - calcu
Date: Tue 5/18/21

Task	▬	Inactive Summary	▭	External Tasks	▬
Split	┈	Manual Task	▬	External Milestone	◇
Milestone	◆	Duration-only	▬	Deadline	⬇
Summary	┏━┓	Manual Summary Rollup	▬	Progress	─
Project Summary	┏━┓	Manual Summary	┏━┓	Manual Progress	─
Inactive Task	▭	Start-only	[		
Inactive Milestone	◇	Finish-only	]		

ID	ℹ	Task Mode	WBS	Task Name	Duration	Start	Finish	Predecessors	Resource Names	Jan 10, '21 S M
37	✓	➡	6.5	Presentation and examination	1 day	Mon 6/7/21	Mon 6/7/21	36		
38	✓	➡	**7**	**Important dates**	**101 days**	**Mon 2/1/21**	**Tue 6/22/21**			
39	✓	📌	7.1	Deadline for signing the final (adjusted) project topic description	0 days	Mon 2/1/21	Mon 2/1/21			
40	✓	📌	7.2	Deadline for the supervisor to find external assessor	0 days	Mon 4/5/21	Mon 4/5/21			
41	✓	📌	7.3	Deadline for deciding date for oral presentation and examination	0 days	Mon 4/26/21	Mon 4/26/21			
42	✓	📌	7.4	Deadline for oral presentation and examination	0 days	Tue 6/22/21	Tue 6/22/21			
43	✓	📌	7.5	Deadline for grading the thesis	0 days	Tue 6/22/21	Tue 6/22/21			
44	✓	📌	**8**	**Project steering**	**3 days**	**Thu 1/21/21**	**Mon 1/25/21**	**4**		
45	✓	📌	8.1	Create Gantt diagram	4 hrs	Thu 1/21/21	Thu 1/21/21	4		
46	✓	📌	8.2	Create work breakdown structure	4 hrs	Fri 1/22/21	Fri 1/22/21	4		

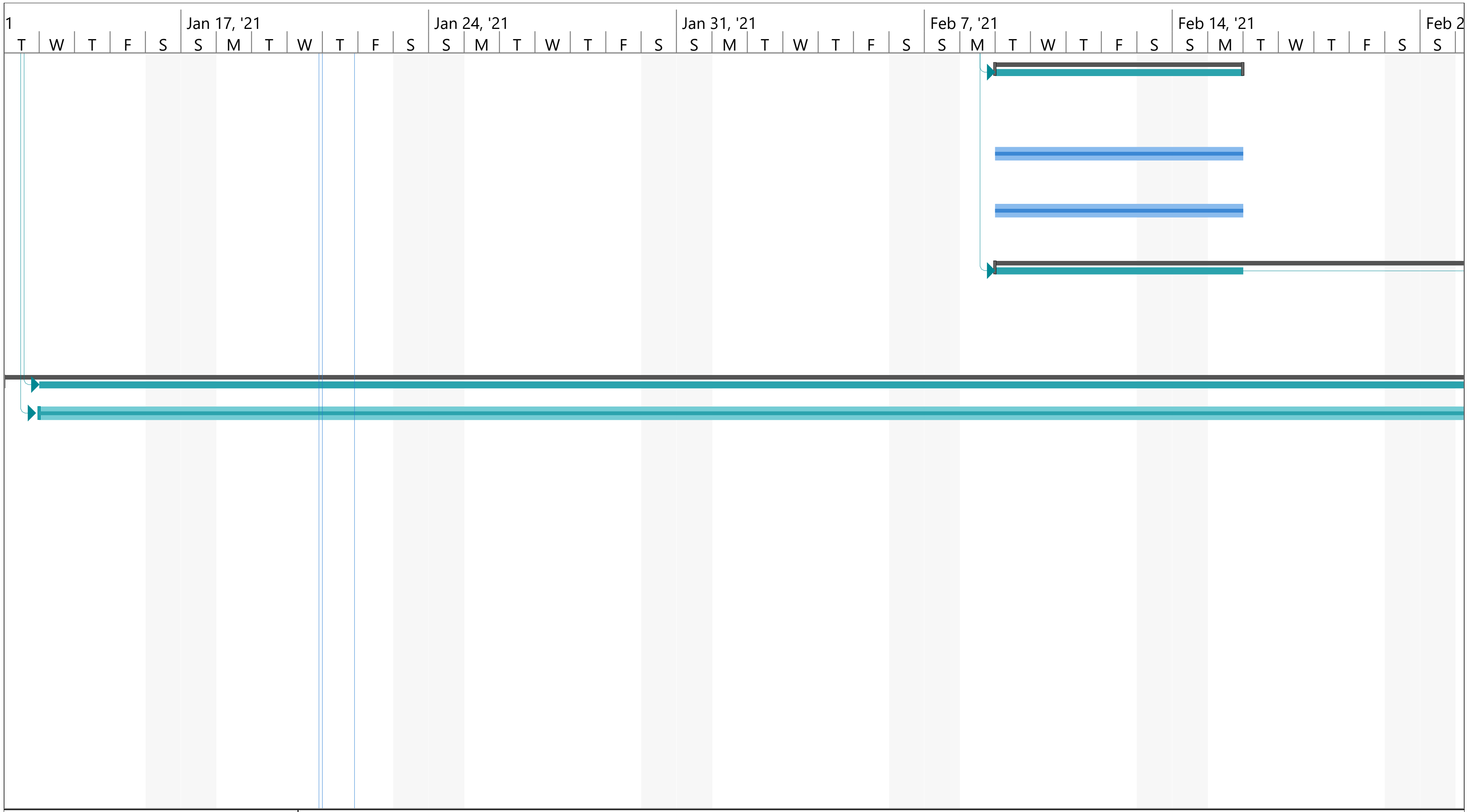
Project: Hydrodynamics - calcu
Date: Tue 5/18/21

Task	▬	Inactive Summary	▭	External Tasks	▬
Split	┈	Manual Task	▬	External Milestone	◇
Milestone	◆	Duration-only	▬	Deadline	⬇
Summary	┏━┓	Manual Summary Rollup	▬	Progress	━
Project Summary	┏━┓	Manual Summary	┏━┓	Manual Progress	━
Inactive Task	▭	Start-only	[		
Inactive Milestone	◇	Finish-only	]		



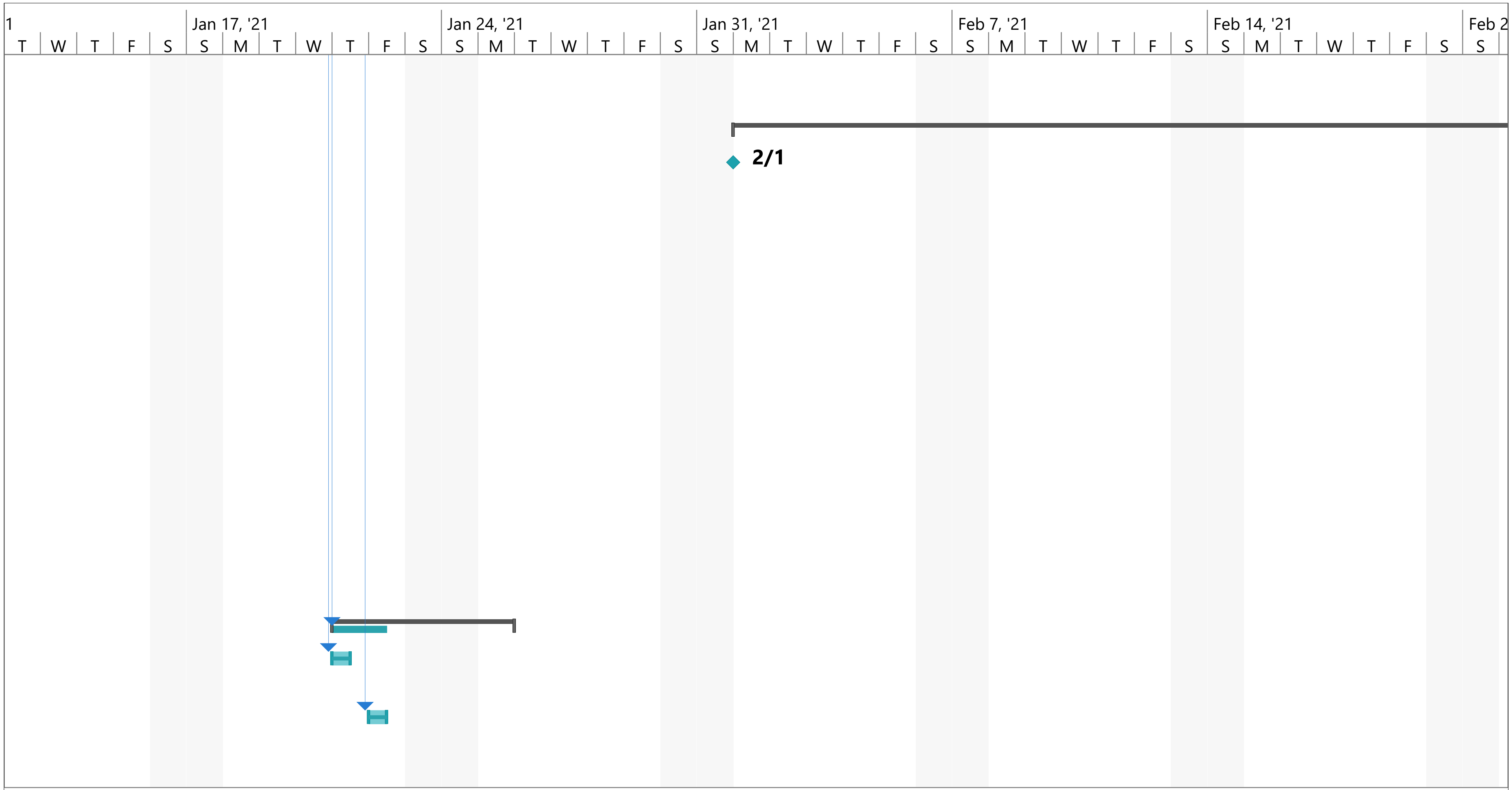
Task	🟦	Inactive Summary	▭	External Tasks	▬
Split	┅	Manual Task	▬	External Milestone	◇
Milestone	◆	Duration-only	▬	Deadline	⇩
Summary	┏━┓	Manual Summary Rollup	▬	Progress	━
Project Summary	┏━┓	Manual Summary	┏━┓	Manual Progress	━
Inactive Task	▭	Start-only	[		
Inactive Milestone	◇	Finish-only	]		

Project: Hydrodynamics - calcu
Date: Tue 5/18/21



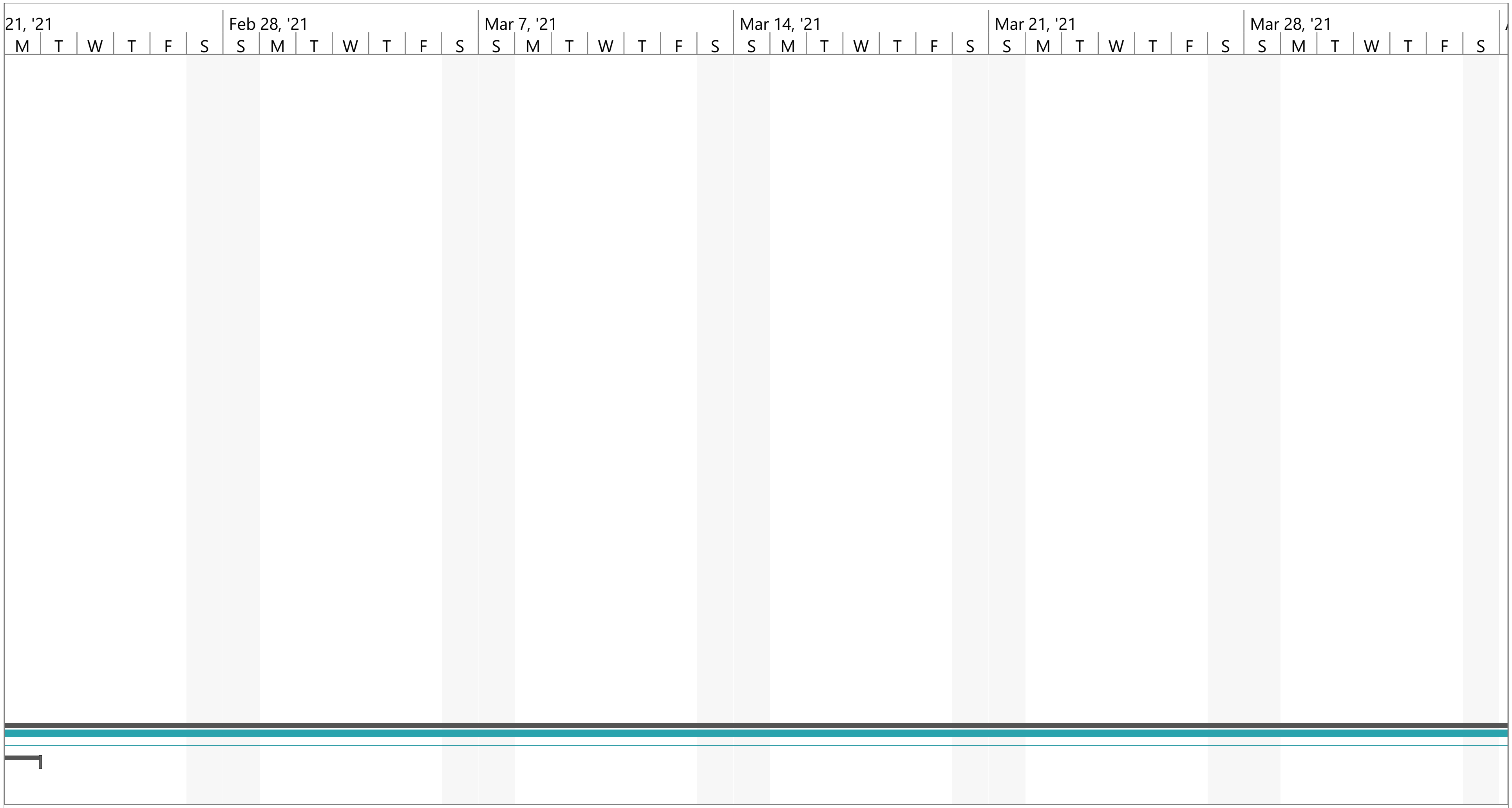
Project: Hydrodynamics - calcu
Date: Tue 5/18/21

Task	▬	Inactive Summary	▭	External Tasks	▬
Split	┈	Manual Task	▬	External Milestone	◇
Milestone	◆	Duration-only	▬	Deadline	⬇
Summary	┏━┓	Manual Summary Rollup	▬	Progress	─
Project Summary	┏━┓	Manual Summary	┏━┓	Manual Progress	─
Inactive Task	▭	Start-only	[		
Inactive Milestone	◇	Finish-only	]		



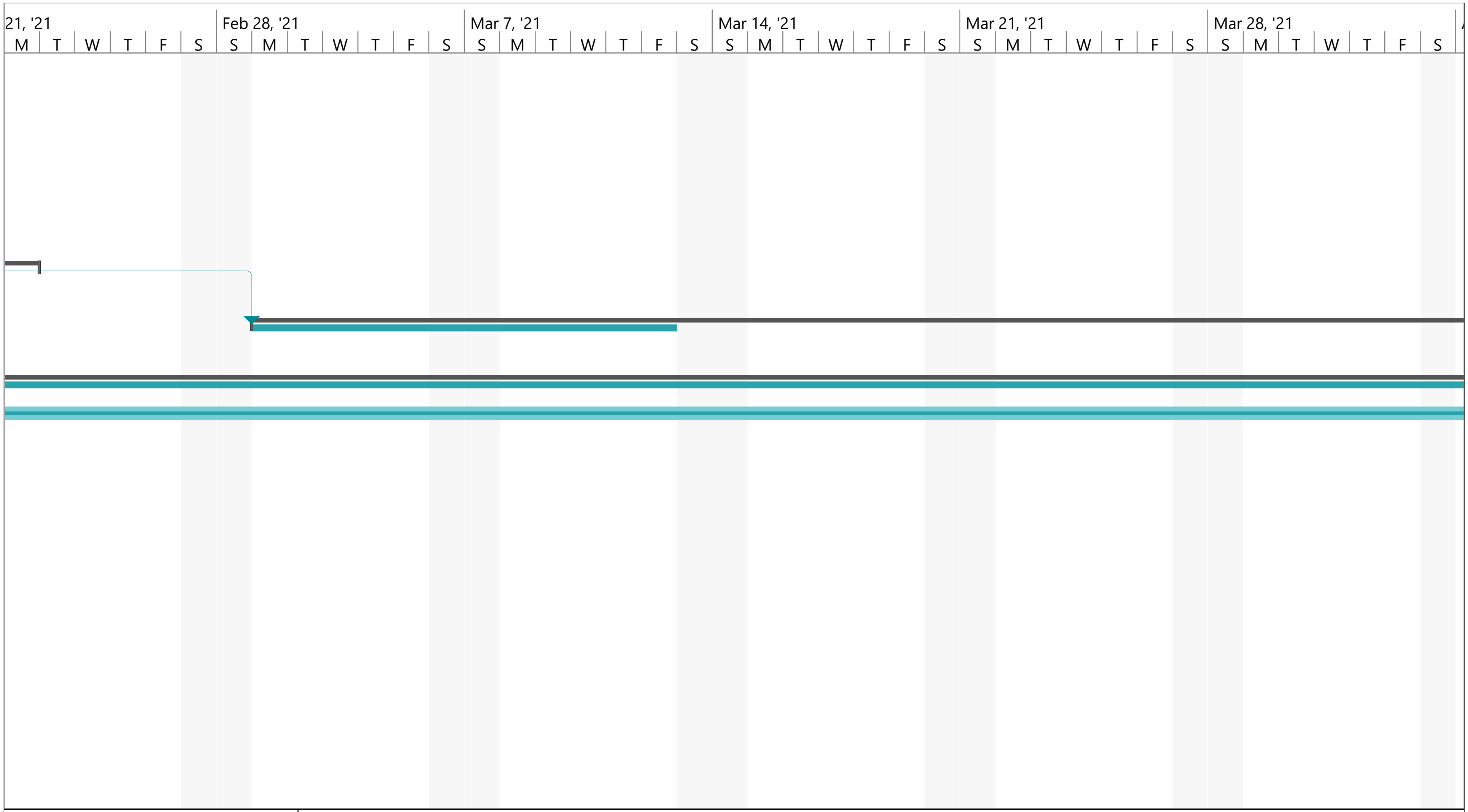
Project: Hydrodynamics - calcu
Date: Tue 5/18/21

Task	▬	Inactive Summary	▭	External Tasks	▬
Split	┈	Manual Task	▬	External Milestone	◇
Milestone	◆	Duration-only	▬	Deadline	⬇
Summary	┏━┓	Manual Summary Rollup	▬	Progress	━
Project Summary	┏━┓	Manual Summary	┏━┓	Manual Progress	━
Inactive Task	▭	Start-only	[		
Inactive Milestone	◇	Finish-only	]		



Project: Hydrodynamics - calcu
Date: Tue 5/18/21

Task	▬	Inactive Summary	▬	External Tasks	▬
Split	┈	Manual Task	▬	External Milestone	◇
Milestone	◆	Duration-only	▬	Deadline	⬇
Summary	▬	Manual Summary Rollup	▬	Progress	▬
Project Summary	▬	Manual Summary	▬	Manual Progress	▬
Inactive Task	▭	Start-only	[		
Inactive Milestone	◇	Finish-only	]		



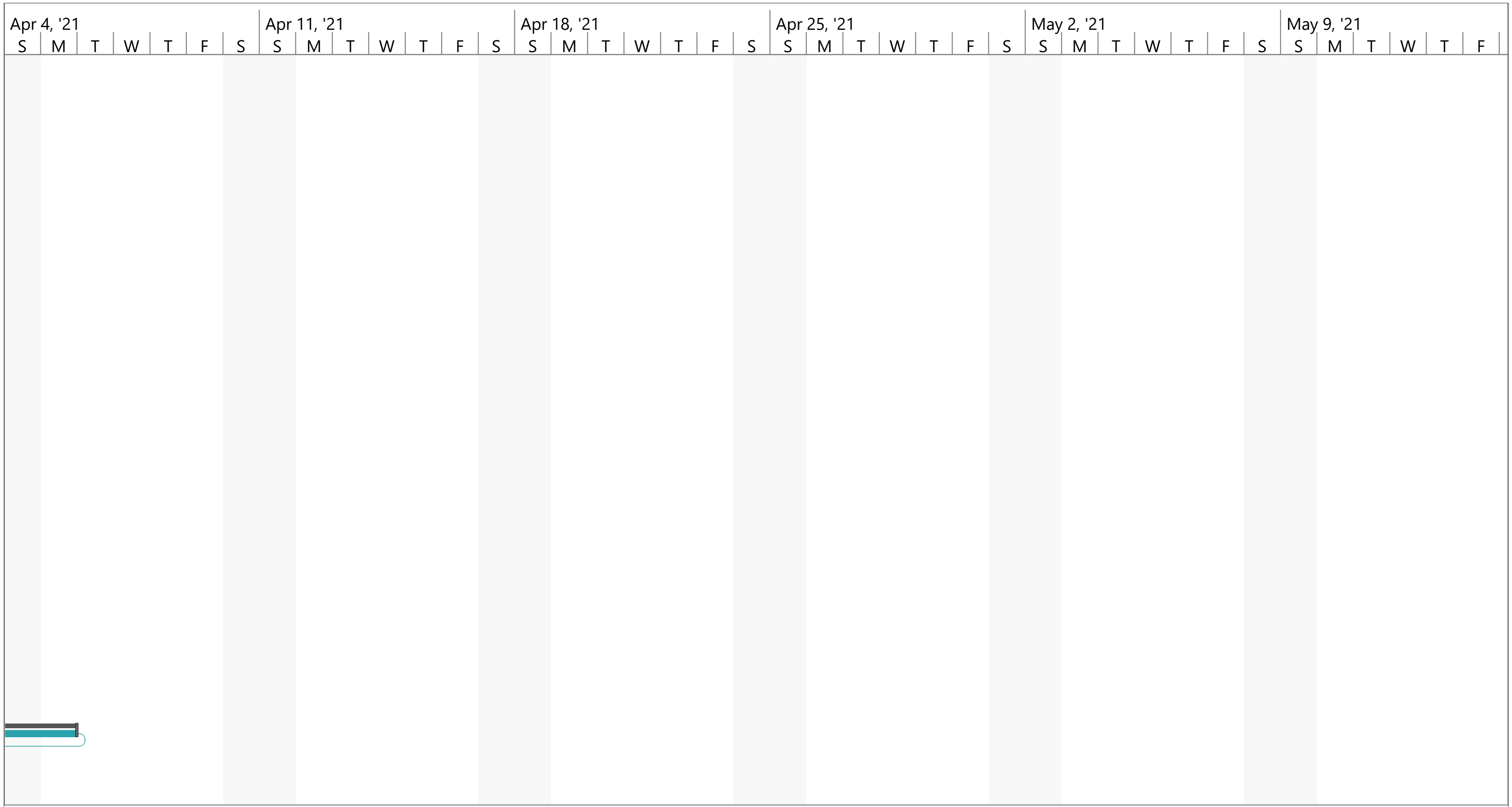
Project: Hydrodynamics - calcu
Date: Tue 5/18/21

Task	▬	Inactive Summary	▭	External Tasks	▬
Split	┅	Manual Task	▬	External Milestone	◇
Milestone	◆	Duration-only	▬	Deadline	⬇
Summary	┏━┓	Manual Summary Rollup	▬	Progress	━
Project Summary	┏━┓	Manual Summary	┏━┓	Manual Progress	━
Inactive Task	▭	Start-only	[		
Inactive Milestone	◇	Finish-only	]		

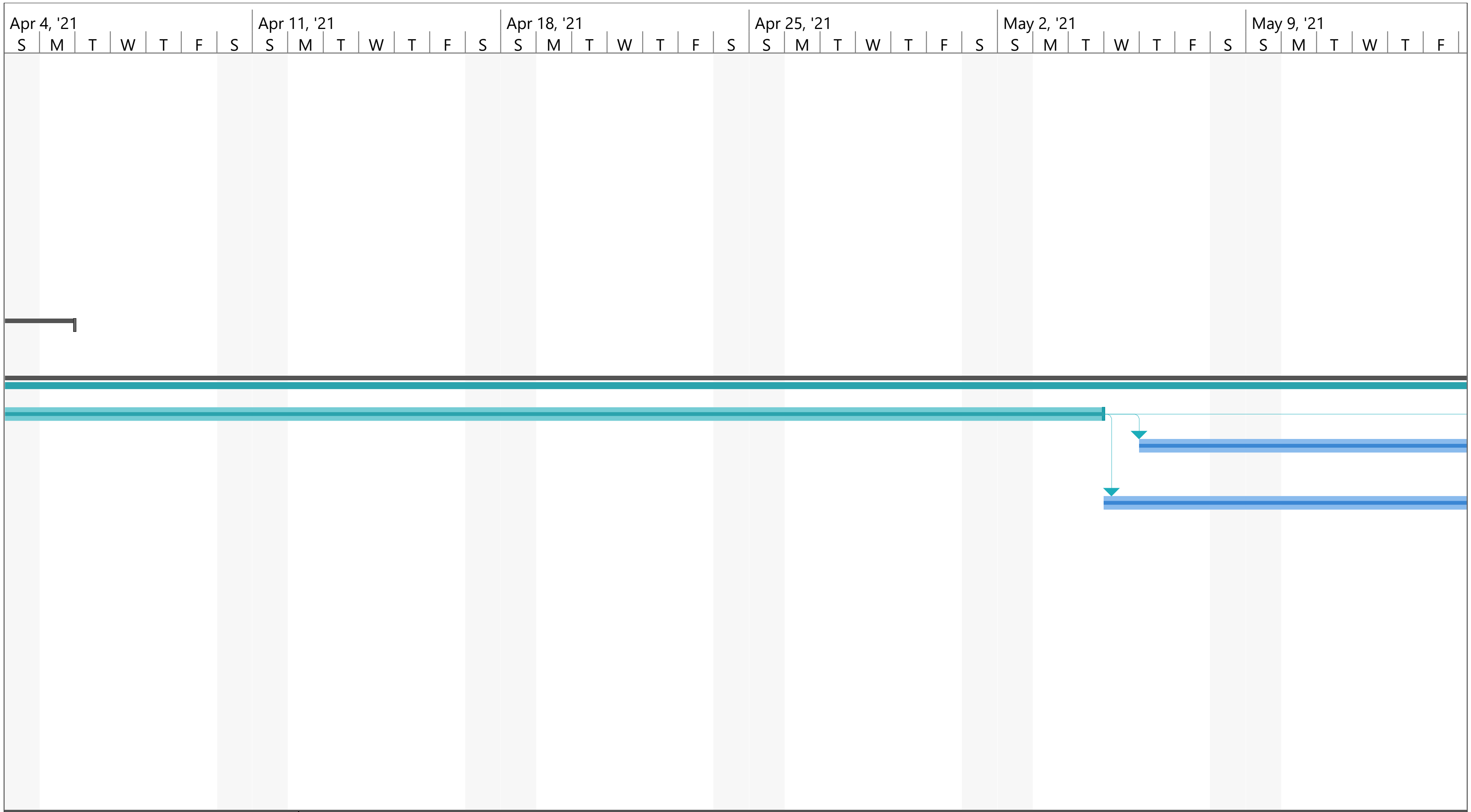
21, '21
Feb 28, '21
Mar 7, '21
Mar 14, '21
Mar 21, '21
Mar 28, '21
M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S

Project: Hydrodynamics - calcu
Date: Tue 5/18/21

Task	▬	Inactive Summary	⌐¬	External Tasks	▬
Split	┅	Manual Task	▬	External Milestone	◇
Milestone	◆	Duration-only	▬	Deadline	⬇
Summary	⌐¬	Manual Summary Rollup	▬	Progress	─
Project Summary	⌐¬	Manual Summary	⌐¬	Manual Progress	─
Inactive Task	▭	Start-only	[		
Inactive Milestone	◇	Finish-only	]		

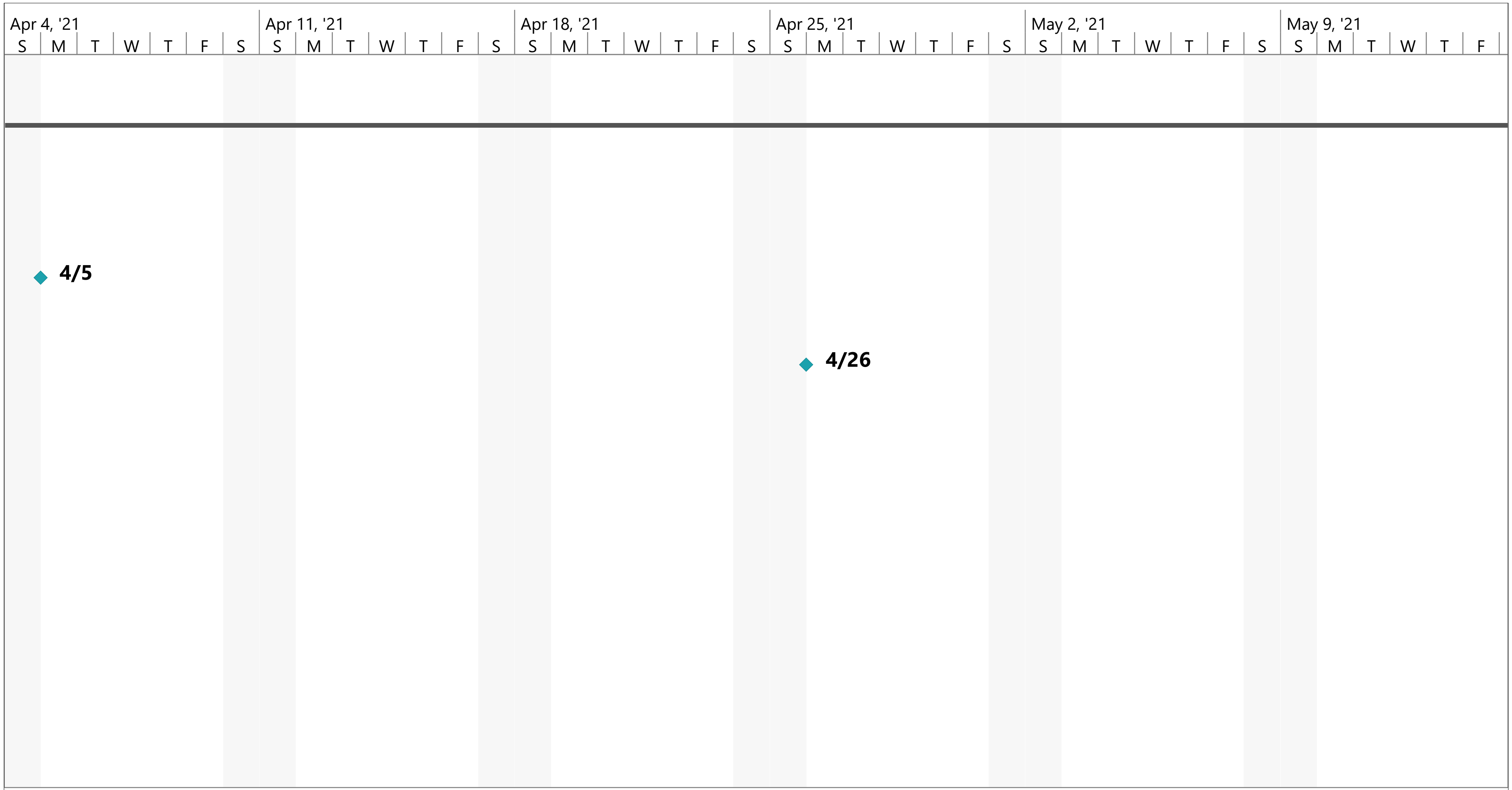


Project: Hydrodynamics - calcu Date: Tue 5/18/21	Task	▬	Inactive Summary	▭	External Tasks	▬
	Split	┅	Manual Task	▬	External Milestone	◇
	Milestone	◆	Duration-only	▬	Deadline	⬇
	Summary	┏━┓	Manual Summary Rollup	▬	Progress	━
	Project Summary	┏━┓	Manual Summary	┏━┓	Manual Progress	━
	Inactive Task	▭	Start-only	[		
	Inactive Milestone	◇	Finish-only	]		



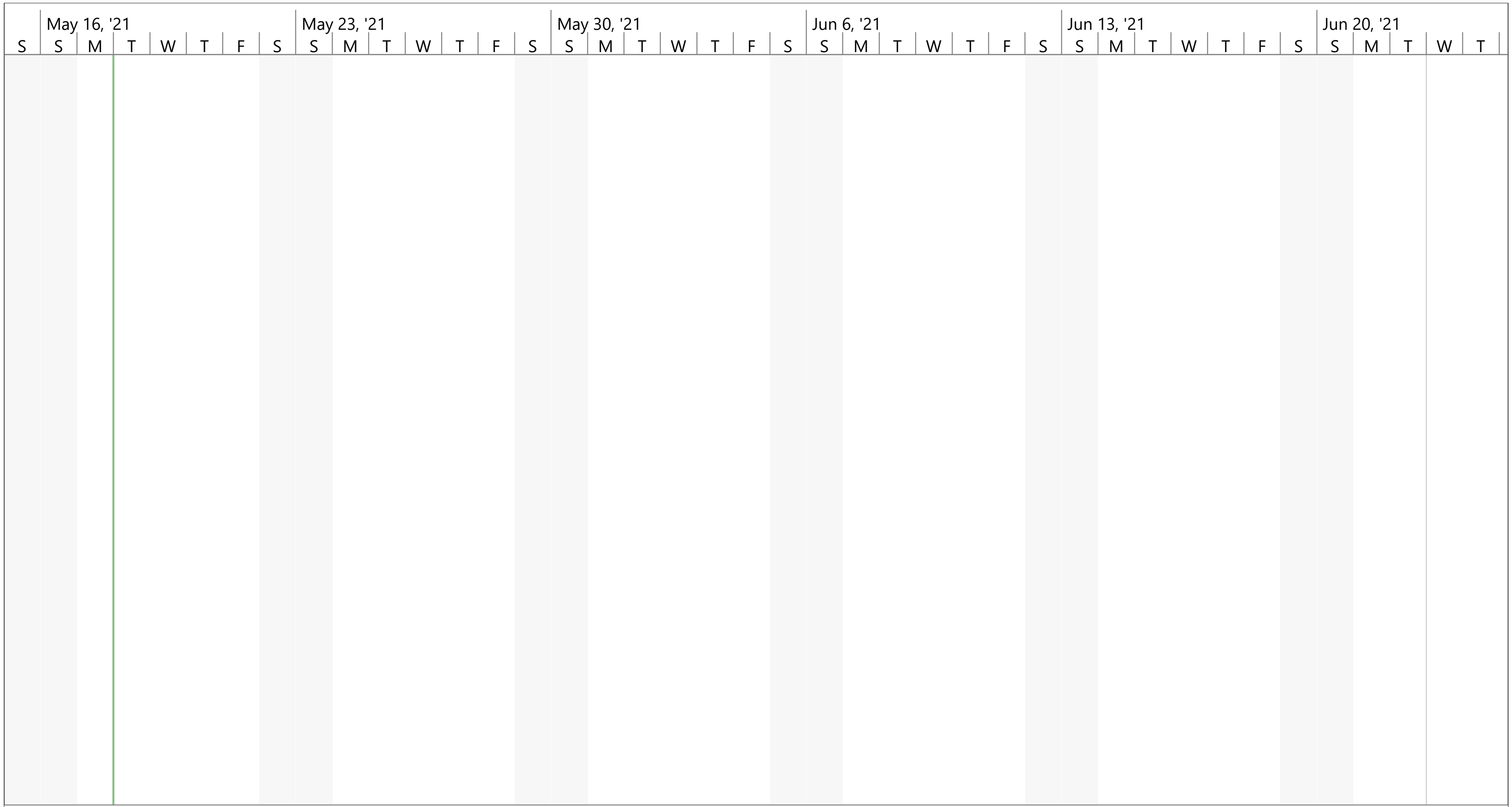
Project: Hydrodynamics - calcu
Date: Tue 5/18/21

Task	▬	Inactive Summary	▭	External Tasks	▬
Split	┅	Manual Task	▬	External Milestone	◇
Milestone	◆	Duration-only	▬	Deadline	⬇
Summary	┏━┓	Manual Summary Rollup	▬	Progress	─
Project Summary	┏━┓	Manual Summary	┏━┓	Manual Progress	─
Inactive Task	▭	Start-only	[		
Inactive Milestone	◇	Finish-only	]		



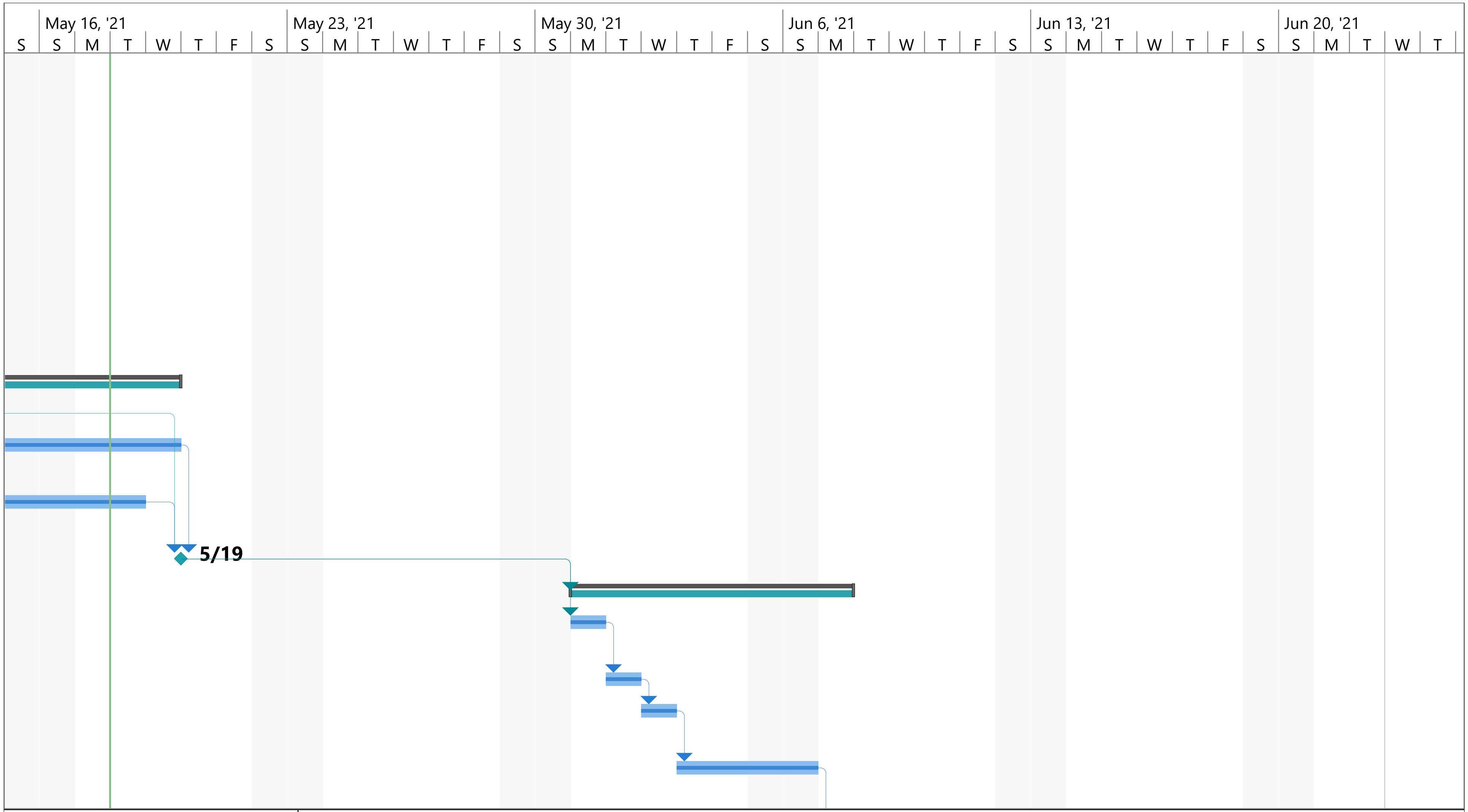
Project: Hydrodynamics - calcu
Date: Tue 5/18/21

Task	▬	Inactive Summary	▬	External Tasks	▬
Split	┈	Manual Task	▬	External Milestone	◆
Milestone	◆	Duration-only	▬	Deadline	⬇
Summary	▬	Manual Summary Rollup	▬	Progress	▬
Project Summary	▬	Manual Summary	▬	Manual Progress	▬
Inactive Task	▭	Start-only	[		
Inactive Milestone	◇	Finish-only	]		



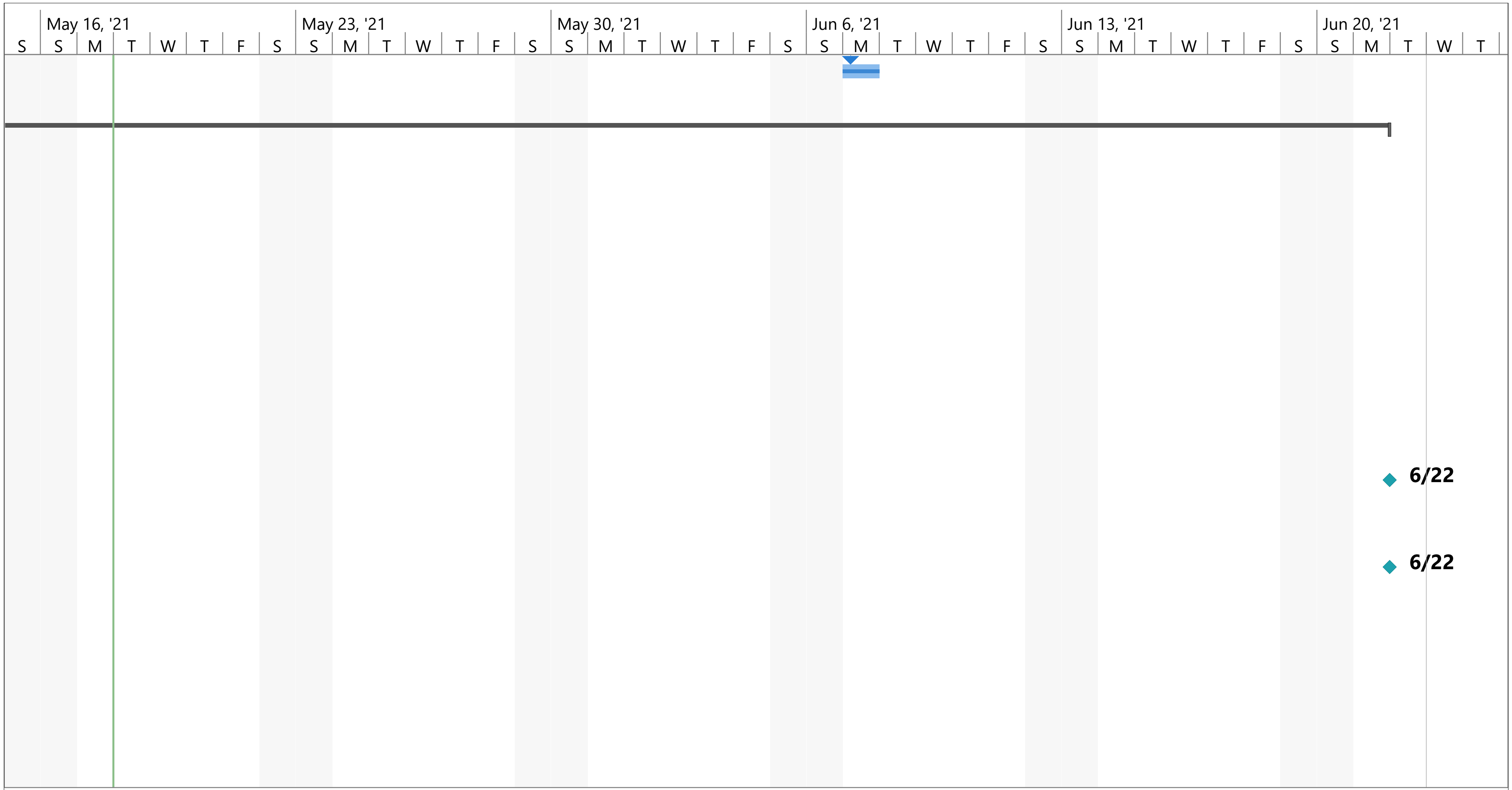
Project: Hydrodynamics - calcu
Date: Tue 5/18/21

Task	▬	Inactive Summary	▭	External Tasks	▬
Split	┈	Manual Task	▬	External Milestone	◇
Milestone	◆	Duration-only	▬	Deadline	⬇
Summary	┏━┓	Manual Summary Rollup	▬	Progress	━
Project Summary	┏━┓	Manual Summary	┏━┓	Manual Progress	━
Inactive Task	▭	Start-only	[		
Inactive Milestone	◇	Finish-only	]		



Project: Hydrodynamics - calcu
Date: Tue 5/18/21

Task	▬	Inactive Summary	▭	External Tasks	▬
Split	┅	Manual Task	▬	External Milestone	◇
Milestone	◆	Duration-only	▬	Deadline	⬇
Summary	▬	Manual Summary Rollup	▬	Progress	―
Project Summary	▬	Manual Summary	▬	Manual Progress	―
Inactive Task	▭	Start-only	[		
Inactive Milestone	◇	Finish-only	]		



Task	▬	Inactive Summary	▭	External Tasks	▬
Split	┈	Manual Task	▬	External Milestone	◇
Milestone	◆	Duration-only	▬	Deadline	⬇
Summary	┏━┓	Manual Summary Rollup	▬	Progress	━
Project Summary	┏━┓	Manual Summary	┏━┓	Manual Progress	━
Inactive Task	▭	Start-only	[		
Inactive Milestone	◇	Finish-only	]		

Project: Hydrodynamics - calcu
Date: Tue 5/18/21

Appendix D - Python-script for post-processing

```python
#%% Importing packages

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

#%% Loading data from constant velocity text files

# Abbreviation df = dataframe

dfCd1 = pd.read_table("coeffs-u1.dat", sep = "\t",
                      skiprows= (12), usecols = [0,1])
dfCd2 = pd.read_table("coeffs-u2.dat", sep = "\t",
                      skiprows = (12), usecols = [0,1])
dfCd3 = pd.read_table("coeffs-u3.dat", sep = "\t",
                      skiprows = (12), usecols = [0,1])
dfCd4 = pd.read_table("coeffs-u4.dat", sep = "\t",
                      skiprows = (12), usecols = [0,1])
dfCd5 = pd.read_table("coeffs-u5.dat", sep = "\t",
                      skiprows = (12), usecols = [0,1])

#%% Converting dataframe to numpy array

Cd1_arr = dfCd1.to_numpy()
Cd2_arr = dfCd2.to_numpy()
Cd3_arr = dfCd3.to_numpy()
Cd4_arr = dfCd4.to_numpy()
Cd5_arr = dfCd5.to_numpy()

"""The drag coefficients should approach a steady state value towards the
end of the simulation. The average drag coefficient is also computed. """

Cd1_final = Cd1_arr[-1,1]; Cd2_final = Cd2_arr[-1,1];
Cd3_final = Cd3_arr[-1,1]; Cd4_final = Cd4_arr[-1,1];
Cd5_final = Cd5_arr[-1,1]
Cd1_avg = (1.0)/len(Cd1_arr)*sum(Cd1_arr[:, 1])
Cd2_avg = (1.0)/len(Cd2_arr)*sum(Cd2_arr[:, 1])
Cd3_avg = (1.0)/len(Cd3_arr)*sum(Cd3_arr[:, 1])
Cd4_avg = (1.0)/len(Cd4_arr)*sum(Cd4_arr[:, 1])
Cd5_avg = (1.0)/len(Cd5_arr)*sum(Cd5_arr[:, 1])

# Array collecting the final drag coefficients

Cd_coll_fin = np.array([Cd1_final, Cd2_final, Cd3_final, Cd4_final,
                        Cd5_final])
Cd_coll_avg = np.array([Cd1_avg, Cd2_avg, Cd3_avg, Cd4_avg, Cd5_avg])
```

```
#%% Assigning an array to velocity for drag coeff. evaluation

u_min = 0.2                # minimum velocity
u_max = 1.0                # maximum velocity
N = len(Cd_coll_fin)           # number of analyses at constant velocity
u_arr = np.linspace(u_min, u_max, N)

#%% Plotting the drag coefficient as a function of velocity

plt.figure(1, figsize = (12, 9))
plt.plot(u_arr, Cd_coll_fin, '-o', label = 'Final Cd')
plt.plot(u_arr, Cd_coll_avg, '-o', label = 'Average Cd')
plt.legend()
plt.xlabel('velocity, u [m/s]')
plt.ylabel('Drag coefficient, Cd [-]')
plt.xlim(0, u_max); plt.ylim(0, 2)
plt.show()

#%% Loading data from constant acceleration text file

dftime = pd.read_table("force.dat", sep = "\t",
                        skiprows = (3), usecols = [0])

dfTotForce = pd.read_table("force.dat", sep = "\t",
                           skiprows = (3), usecols = [1])
dfPressForce = pd.read_table("force.dat", sep = "\t",
                             skiprows = (3), usecols = [2])
time = dftime.to_numpy()

# Removing parentheses from the .dat file

dfTotForce = dfTotForce.replace(to_replace='\(', value="", regex = True)
dfTotForce = dfTotForce.replace(to_replace='\)', value="", regex = True)

dfPressForce = dfPressForce.replace(to_replace='\(', value="", regex = True)
dfPressForce = dfPressForce.replace(to_replace='\)', value="", regex = True)

# Splitting force vectors into three columns

dfSplitTotForce = dfTotForce['(total_x total_y total_z)'].apply(
    lambda x: pd.Series(x.split(' ')))

dfSplitPressForce = dfPressForce['(pressure_x pressure_y pressure_z)'].apply(
    lambda x: pd.Series(x.split(' ')))

""" The forces are loaded as object arrays initially, due to the
combination of string and float elements. They need to be converted to
```

```python
numpy float arrays for post-processing."""


# Converting dtaframes to numpy array
arrTF = dfSplitTotForce.to_numpy()  # initially object array
totForce = arrTF.astype(np.float)   # converted to float array

arrPF = dfSplitPressForce.to_numpy()
pressForce = arrPF.astype(np.float)

# Locating the smallest numeric value of the forces

min_elemXPF = 0.0
min_elemYPF = 0.0
min_elemZPF = 0.0

for i in range(0, len(pressForce)):
    if min_elemXPF < pressForce[i, 0]:
        min_elemXPF = min_elemXPF
    else:
        min_elemXPF = pressForce[i, 0]

    if min_elemYPF < pressForce[i, 1]:
        min_elemYPF = min_elemYPF
    else:
        min_elemYPF = pressForce[i, 1]

    if min_elemZPF < pressForce[i, 2]:
        min_elemZPF = min_elemZPF
    else:
        min_elemZPF = pressForce[i, 2]
min_elem = min(min_elemXPF, min_elemYPF, min_elemZPF)

#%% Plotting the forces

# Plotting total forces
plt.figure(2, figsize = (12, 9))
plt.plot(time, totForce[:, 0], 'r', label = 'Total force X')
plt.plot(time, totForce[:, 1], 'g', label = 'Total force Y')
plt.plot(time, totForce[:, 2], 'b', label = 'Total force Z')
plt.xlim(0.01, time[-1]); plt.ylim(1.1*min_elem, 1.1*totForce[-1, 2])
plt.legend()
plt.xlabel('Time [s]'); plt.ylabel('Force [N]')
plt.show()

# Plotting pressure forces
```

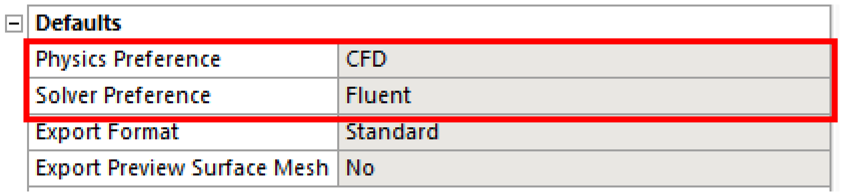
```
plt.figure(3, figsize = (12, 9))
plt.plot(time, pressForce[:, 0], 'r', label = 'Pressure force X')
plt.plot(time, pressForce[:, 1], 'g', label = 'Pressure force Y')
plt.plot(time, pressForce[:, 2], 'b', label = 'Pressure force Z')
plt.xlim(0.01, time[-1]); plt.ylim(1.1*min_elem, 1.1*pressForce[-1, 2])
plt.legend()
plt.xlabel('Time [s]'); plt.ylabel('Force [N]')
plt.show()

#%% Properties of the analysis

rho = 1027.6                    # sea water density at 5 C [kg/m3]
V_ref = 0.28815542              # volume of solid [m3]
m_w_disp = V_ref * rho          # mass of displaced water [kg]

acc = 1.0                       # fluid acceleration [m/s2]
time_acc_onset = 0.2           # time of initiating fluid acceleration [s]

#%% Extracting the mass force from .dat file and printing to console

t0 = np.where(time == time_acc_onset)  # location at time of acceleration
t1 = t0[0] + 1                         # next point in time

# Calculating mass and inertia force in acceleration direction

# computing the mass force
F_m = float(pressForce[t1[0], 2] - pressForce[t0[0], 2] )
F_FK = float(m_w_disp * acc)    # computing the Froude-Kriloff force
print("Fluid acceleration, a = {:.3f} m/s2".format(acc))
print("Mass force, F_m = {:.3f} N".format(F_m))
print("Froude-Kriloff force, F_FK = {:.3f}".format(F_FK))
print("Mass of displaced water, M_w = {:.3f}".format(m_w_disp))

F_a = float(F_m - m_w_disp)
M_a = float(F_a / acc)
print("Added mass force, F_a = {:.3f} N".format(F_a))
print("Added mass, M_a = {:.3f} kg".format(M_a))
```

Appendix E – user manual

Firstly, make sure that ANSYS 19.2 is installed on your computer. This is because the meshing methods to be used do not work in newer versions of ANSYS.

Part 1 – preparing the geometry

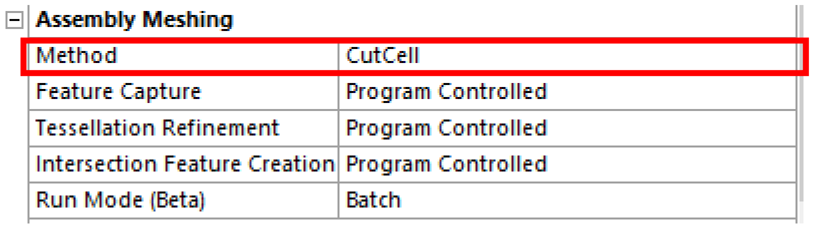
1. Open your geometry in SpaceClaim and make the required changes. Remove unnecessary details and correct modeling errors.
2. Merge all the different parts into one body. If all parts are not merged, it means that there are some gaps in the model that needs to be corrected.
3. Go to the prepare tab and select "Enclosure". Uncheck the "Symmetric dimensions" option and click the object you modified.
4. Make the domain sufficiently large. A few different fluid domains should be investigated for comparison to verify the accuracy of the created fluid domain.
5. Next, open the fluid domain in ANSYS Meshing. Go to the Mesh branch in the project tree. Select "CFD" for physics preference and "Fluent" for solver preference, like in Figure 1.



Defaults	
Physics Preference	CFD
Solver Preference	Fluent
Export Format	Standard
Export Preview Surface Mesh	No

Figure 1: Physics and solver preferences in ANSYS Meshing

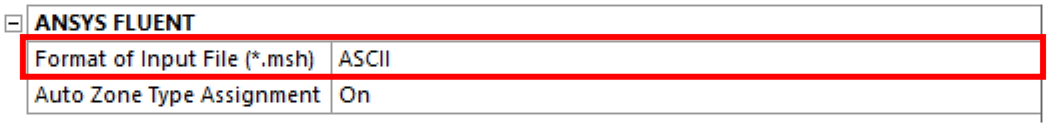
6. Under assembly meshing, select "CutCell" for method. This is the most computationally efficient meshing method available with the licenses currently available.



Assembly Meshing	
Method	CutCell
Feature Capture	Program Controlled
Tessellation Refinement	Program Controlled
Intersection Feature Creation	Program Controlled
Run Mode (Beta)	Batch

Figure 2: Assembly meshing preferences

7. Create named selections for each of the external and internal faces in the model. It is beneficial to name the faces in accordance with the boundary type to be applied, e.g. "symmetry_top" for symmetry boundaries, "walls" for wall boundaries. Inlets and outlets can be named "inlet" and "outlet", respectively. Do not have any spaces in the named selections created.
8. Apply sufficient global and local face sizing for your analysis.
9. Before exporting the created mesh, go to Tools -> Options -> Meshing -> Export. Select ASCII under "Format of input file", like in Figure 3. Do not have any spaces in the name of the .msh file.



ANSYS FLUENT	
Format of Input File (*.msh)	ASCII
Auto Zone Type Assignment	On

Figure 3: Format of input file preference

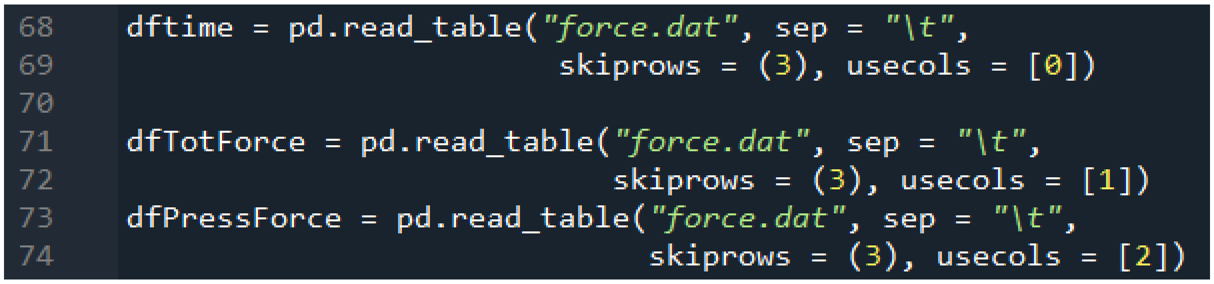
Part 2 – setting up the analysis

1. Upload the .msh file to your cloud storing system. Go to the virtal machine and download the file.
2. A compressed, predefined analysis setup folder called "ADDED_MASS_TEMPLATE" is in the OpenFOAM folder on the virtual machine. Copy this to your desired directory and extract the files. Place your .msh file along the three working folders (0, constant and system).
3. Check that the boundaries are of the correct type. If not, go to system -> createPatchDict. Make some changes to the boundary names that need correction and apply the correct boundary type.
    a. If you do not want to use the default patch names provided in the template, change all patch names in the 0 folder to match your named selections.
4. RMB in the working directory and select "Open in terminal". Type *"of2012"* in the terminal.
5. Open *run_mesh.*sh and change the name of the .msh file on line **6**. Convert the .msh file to polyMesh by typing *"sh run_mesh.sh"* in the terminal.
    a. If it is not necessary to run createPatchDict go to the file *"run_mesh.sh"* and comment out line **8** by placing a "#" in front of the code line.
6. Go to the 0 folder, and check if you have the exact same boundary names as those in the files here. If not, change the boundary names in each file. PS: OpenFOAM is case sensitive.
7. In the "U" file, make changes to the velocity vector for both internalField and the inlet patch. Time-dependent velocity is written as "(time (U_x U_y U_z)). Make changes to fit your application here.
8. Adjust the turbulence parameters k and ω to fit your application. Rough initial estimations for the turbulence properties can be calculated by editing the excel spreadsheet "Turbulence". This file is placed among Stressman Engineering's templates. While you are at it, copy the "postProcess.py" script to the working directory on your physical computer.
9. Go to constant -> transportProperties. Change the kinematic viscosity if necessary.
10. Go to system -> controlDict. Make sure the "endTime" corresponds with the final time you inserted in the "U" directory, previously.
11. While in the system folder, open "decomposeParDict" and adjust the number of cores you wish to use. Note: the computer has 16 cores, any number higher than this will not work.
12. Open the script *run_solver.sh* and change the number 12 to the desired number of cores in createPatchDict.
13. Run the analysis by following typing *"sh run_solver.sh"* in the terminal.

Part 3 – Post-processing

1. During the solution procedure another folder called "postProcessing" was created along with your other working folders. Click through the subfolders here and upload the .dat file to your cloud storage system.

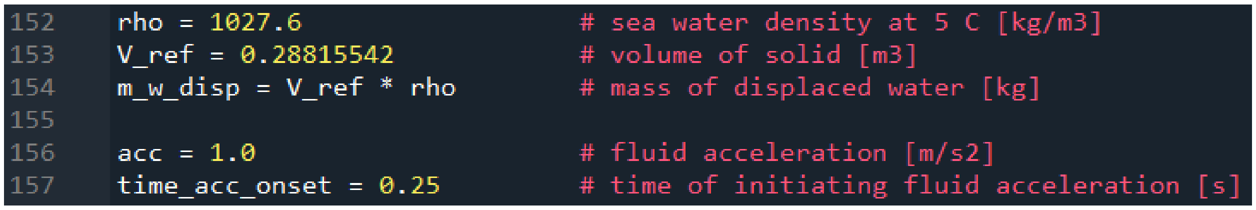
2. Download the file to the working directory of your physical computer.
3. If you want to evaluate added mass before having solved the drag coefficient issue, simply make a string from line 6 to line 65 in the script. Strings are created using “”” at the start and end.
4. If you renamed the .dat file from OpenFOAM, change “force on lines **68**, **71** and **73** shown in Figure 4.



```
68    dftime = pd.read_table("force.dat", sep = "\t",
69                                  skiprows = (3), usecols = [0])
70
71    dfTotForce = pd.read_table("force.dat", sep = "\t",
72                                  skiprows = (3), usecols = [1])
73    dfPressForce = pd.read_table("force.dat", sep = "\t",
74                                  skiprows = (3), usecols = [2])
```

Figure 4: Loading data from .dat file

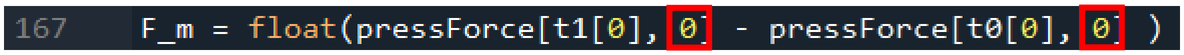
5. Scroll down to lines **152**, **153**, **156** and **157**. Change the values in accordance with your specifications. This section of the code is shown in Figure 5.



```
152   rho = 1027.6                    # sea water density at 5 C [kg/m3]
153   V_ref = 0.28815542              # volume of solid [m3]
154   m_w_disp = V_ref * rho          # mass of displaced water [kg]
155
156   acc = 1.0                       # fluid acceleration [m/s2]
157   time_acc_onset = 0.25           # time of initiating fluid acceleration [s]
```

Figure 5: Analysis parameters

6. Finally, on line **167** of the script, change the column number to extract the added mass from. 0 is the column for force in the x-direction, 1 is the column for force in y-direction, and 2 is the column for force in the z-direction. The indices to be changed are marked with a red square in Figure 6.

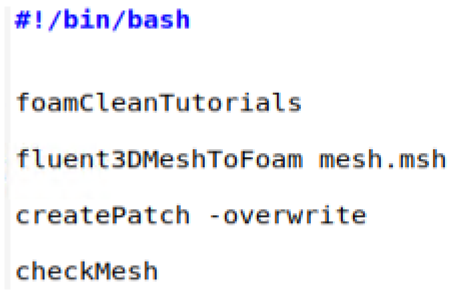


```
167    F_m = float(pressForce[t1[0], 0] - pressForce[t0[0], 0] )
```

Figure 6: Direction of force

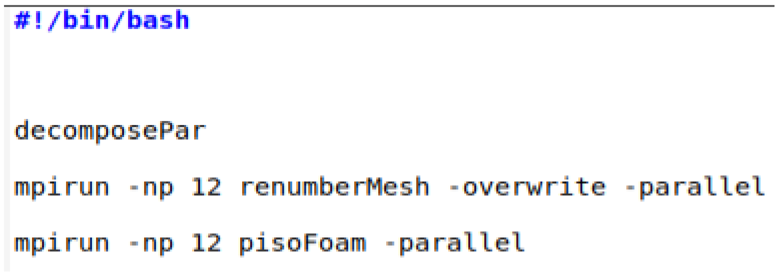
7. Run the script. The plots are available in the “Plots” tab, while the computed added mass force is printed in the Python console.

Shell scripts



```
#!/bin/bash

foamCleanTutorials
fluent3DMeshToFoam mesh.msh
createPatch -overwrite
checkMesh
```

Figure 7: run_mesh.sh script



```
#!/bin/bash


decomposePar
mpirun -np 12 renumberMesh -overwrite -parallel
mpirun -np 12 pisoFoam -parallel
```

Figure 8: run_solver.sh script

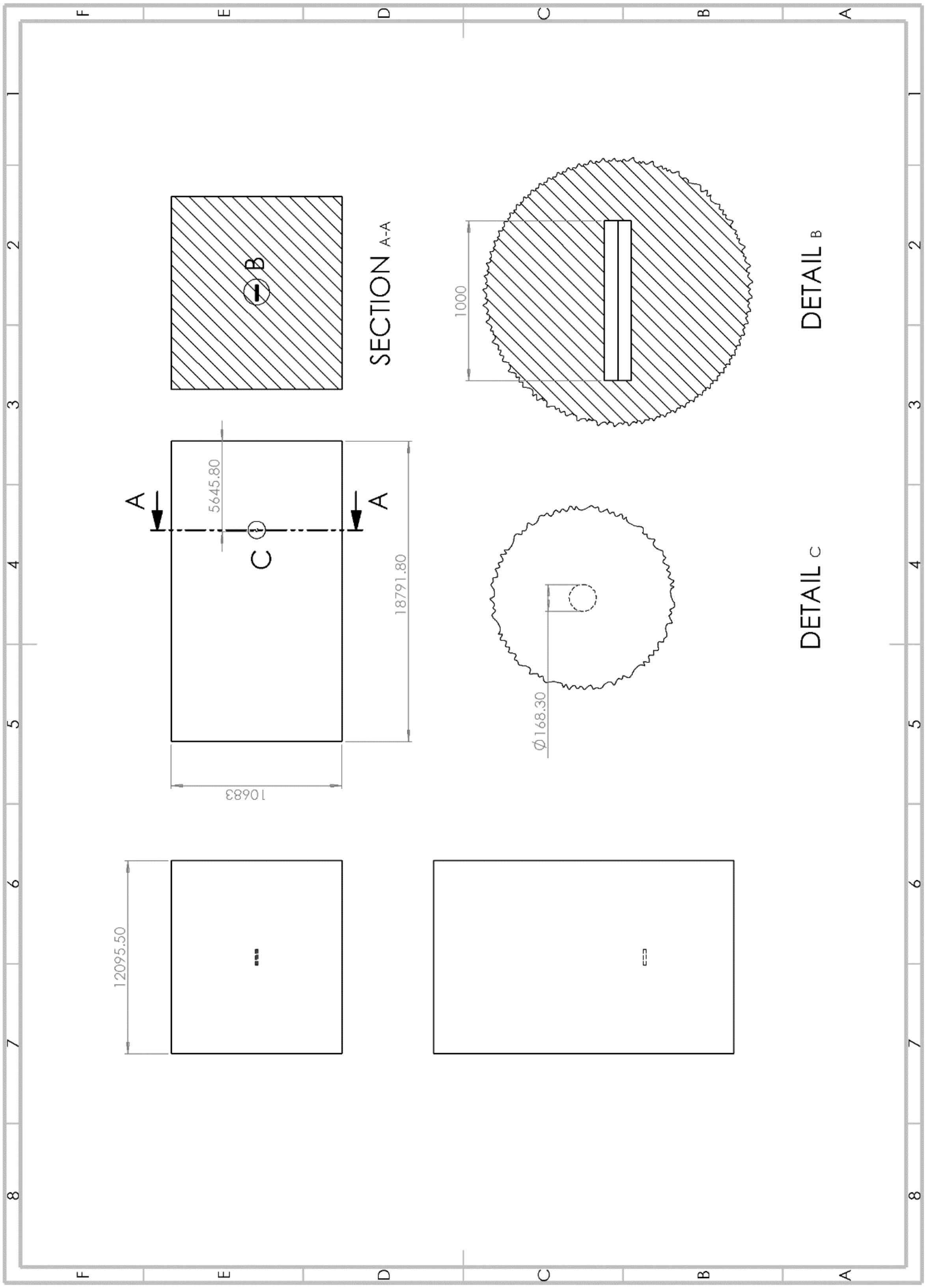
createPatchDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      createPatchDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


pointSync false;

// Patches to create.
patches
(
    {
        // Name of new patch
        name inlet;

        // Dictionary to construct new patch from
        patchInfo
        {
            type patch;
        }

        // How to construct: either from 'patches' or 'set'
        constructFrom patches;

        // Current patch name
        patches (inFlow);

    }

    {
        // Name of new patch
        name outlet;

        // Dictionary to construct new patch from
        patchInfo
        {
            type patch;
        }

        // How to construct: either from 'patches' or 'set'
        constructFrom patches;

        // Current patch name
        patches (outFlow);

    }
```

```
{
    // Name of new patch
    name walls;

    // Dictionary to construct new patch from
    patchInfo
    {
        type wall;
    }

    // How to construct: either from 'patches' or 'set'
    constructFrom patches;

    // Current patch name
    patches (object);

}

{
    // Name of new patch
    name symmetry_right;

    // Dictionary to construct new patch from
    patchInfo
    {
        type symmetry;
    }

    // How to construct: either from 'patches' or 'set'
    constructFrom patches;

    // Current patch name
    patches (sym-0);

}

{
    // Name of new patch
    name symmetry_left;

    // Dictionary to construct new patch from
    patchInfo
    {
        type symmetry;
    }

    // How to construct: either from 'patches' or 'set'
    constructFrom patches;

    // Current patch name
    patches (sym-1);

}
```

```
{
    // Name of new patch
    name symmetry_front;

    // Dictionary to construct new patch from
    patchInfo
    {
        type symmetry;
    }

    // How to construct: either from 'patches' or 'set'
    constructFrom patches;

    // Current patch name
    patches (sym-2);

}

{
    // Name of new patch
    name symmetry_back;

    // Dictionary to construct new patch from
    patchInfo
    {
        type symmetry;
    }

    // How to construct: either from 'patches' or 'set'
    constructFrom patches;

    // Current patch name
    patches (sym-3);

}
```

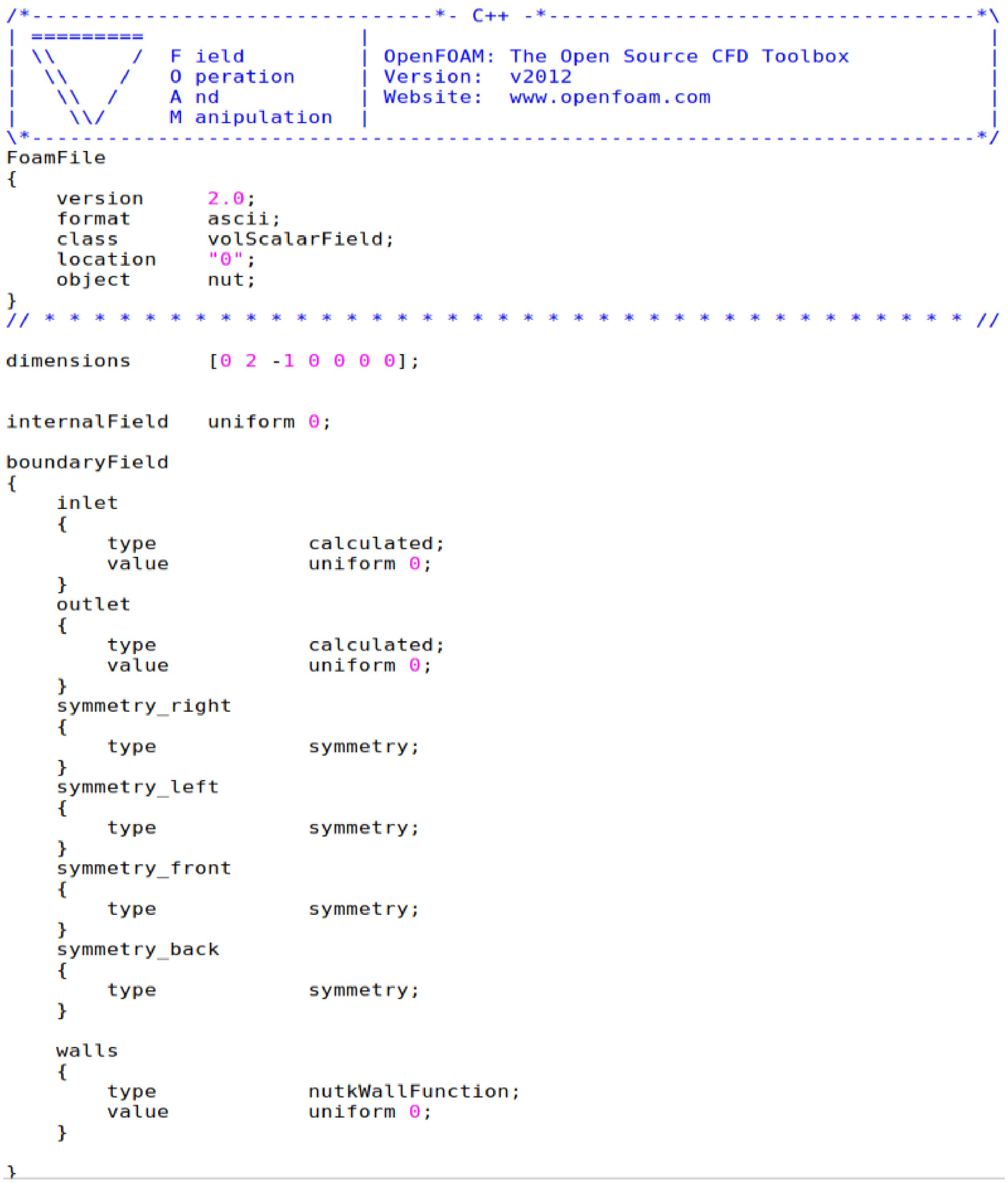
Appendix F – Technical drawing of the fluid domain for the submerged cylinder



Appendix G – Case-setup for the submerged cylinder at constant acceleration

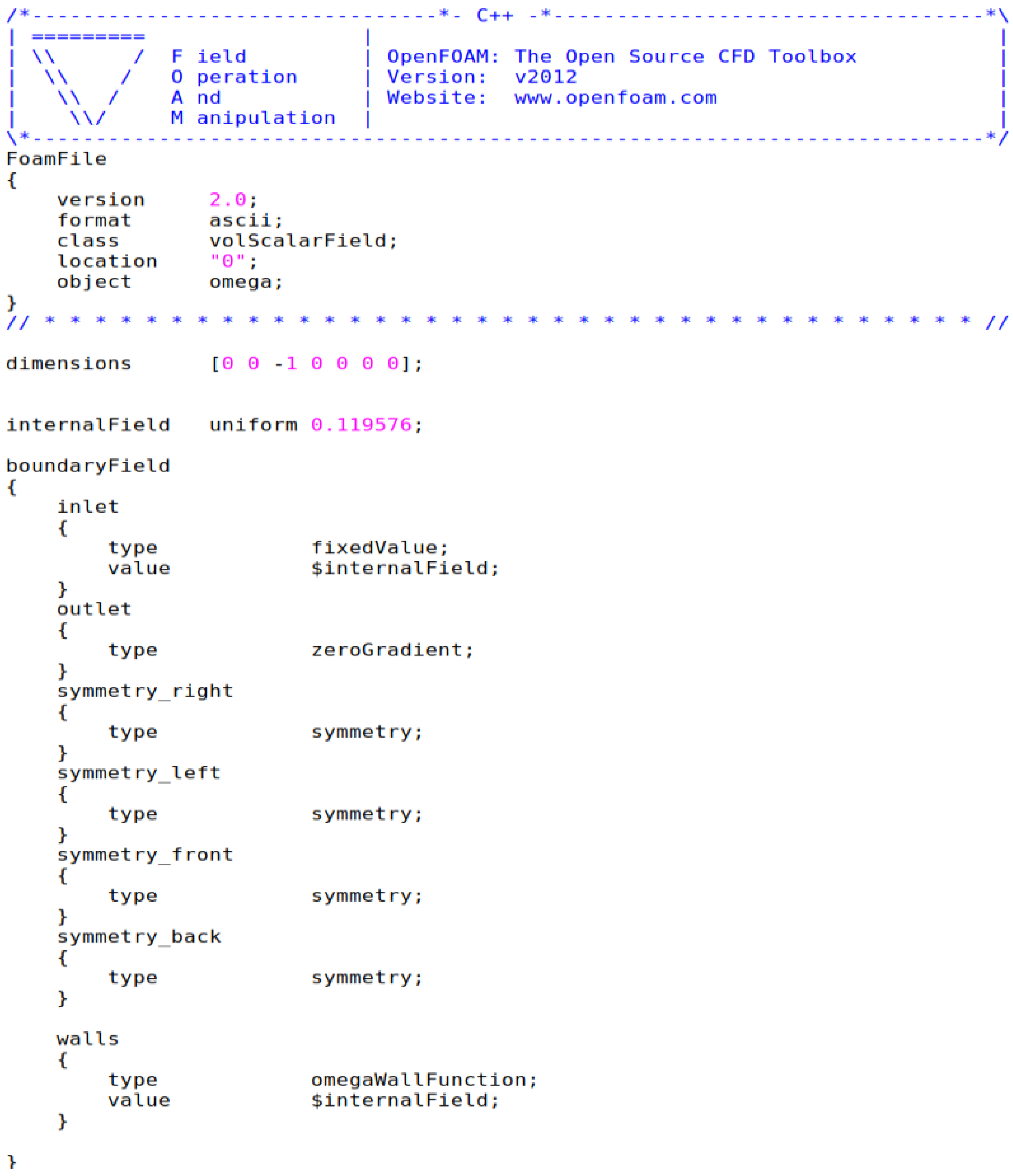
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];


internalField   uniform 1.215e-04;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           $internalField;
    }
    outlet
    {
        type            zeroGradient;
    }
    symmetry_right
    {
        type            symmetry;
    }
    symmetry_left
    {
        type            symmetry;
    }
    symmetry_front
    {
        type            symmetry;
    }
    symmetry_back
    {
        type            symmetry;
    }

    walls
    {
        type            kqRWallFunction;
        value           $internalField;
    }

}
```

Figure 1: 0 directory: turbulent kinetic energy



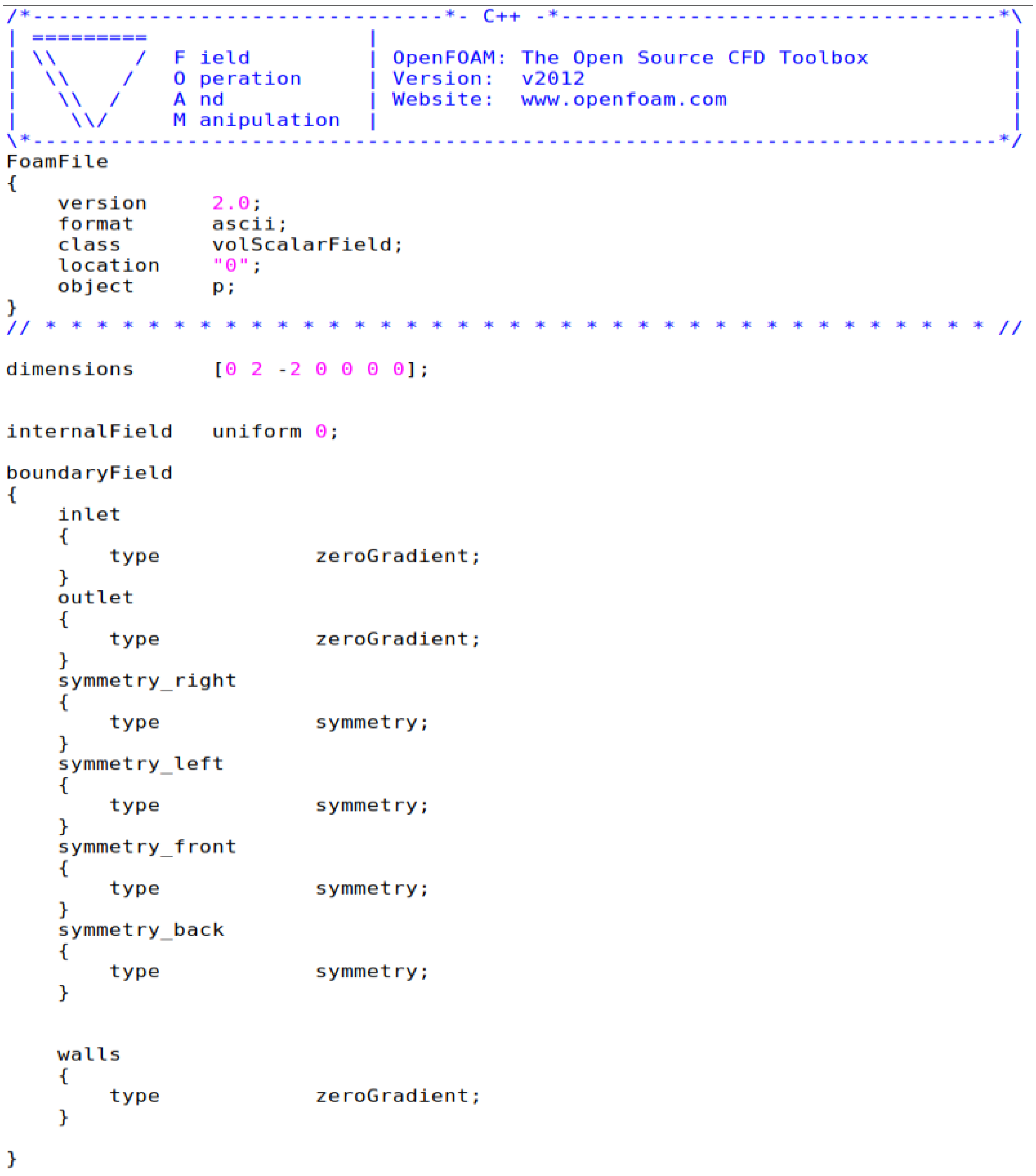
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];


internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            calculated;
        value           uniform 0;
    }
    outlet
    {
        type            calculated;
        value           uniform 0;
    }
    symmetry_right
    {
        type            symmetry;
    }
    symmetry_left
    {
        type            symmetry;
    }
    symmetry_front
    {
        type            symmetry;
    }
    symmetry_back
    {
        type            symmetry;
    }

    walls
    {
        type            nutkWallFunction;
        value           uniform 0;
    }

}
```

Figure 2: 0 directory: eddy-viscosity



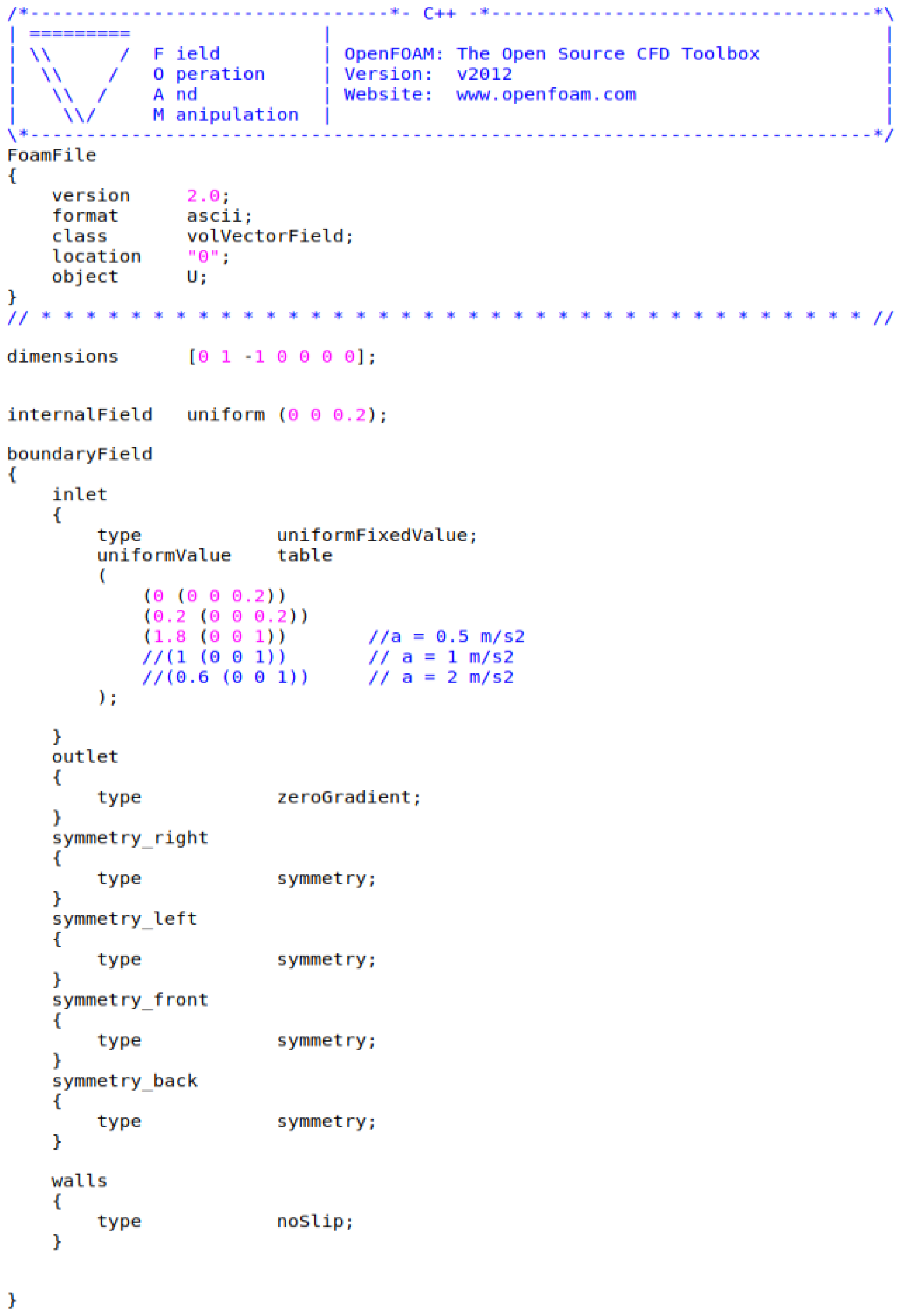
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      omega;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 -1 0 0 0 0];


internalField   uniform 0.119576;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           $internalField;
    }
    outlet
    {
        type            zeroGradient;
    }
    symmetry_right
    {
        type            symmetry;
    }
    symmetry_left
    {
        type            symmetry;
    }
    symmetry_front
    {
        type            symmetry;
    }
    symmetry_back
    {
        type            symmetry;
    }

    walls
    {
        type            omegaWallFunction;
        value           $internalField;
    }

}
```

Figure 3: 0 directory: turbulent frequency



```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];


internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            zeroGradient;
    }
    outlet
    {
        type            zeroGradient;
    }
    symmetry_right
    {
        type            symmetry;
    }
    symmetry_left
    {
        type            symmetry;
    }
    symmetry_front
    {
        type            symmetry;
    }
    symmetry_back
    {
        type            symmetry;
    }


    walls
    {
        type            zeroGradient;
    }

}
```

Figure 4: 0 directory: kinematic pressure



```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];


internalField   uniform (0 0 0.2);

boundaryField
{
    inlet
    {
        type            uniformFixedValue;
        uniformValue    table
        (
            (0 (0 0 0.2))
            (0.2 (0 0 0.2))
            (1.8 (0 0 1))       //a = 0.5 m/s2
            //(1 (0 0 1))       // a = 1 m/s2
            //(0.6 (0 0 1))     // a = 2 m/s2
        );

    }
    outlet
    {
        type            zeroGradient;
    }
    symmetry_right
    {
        type            symmetry;
    }
    symmetry_left
    {
        type            symmetry;
    }
    symmetry_front
    {
        type            symmetry;
    }
    symmetry_back
    {
        type            symmetry;
    }

    walls
    {
        type            noSlip;
    }


}
```

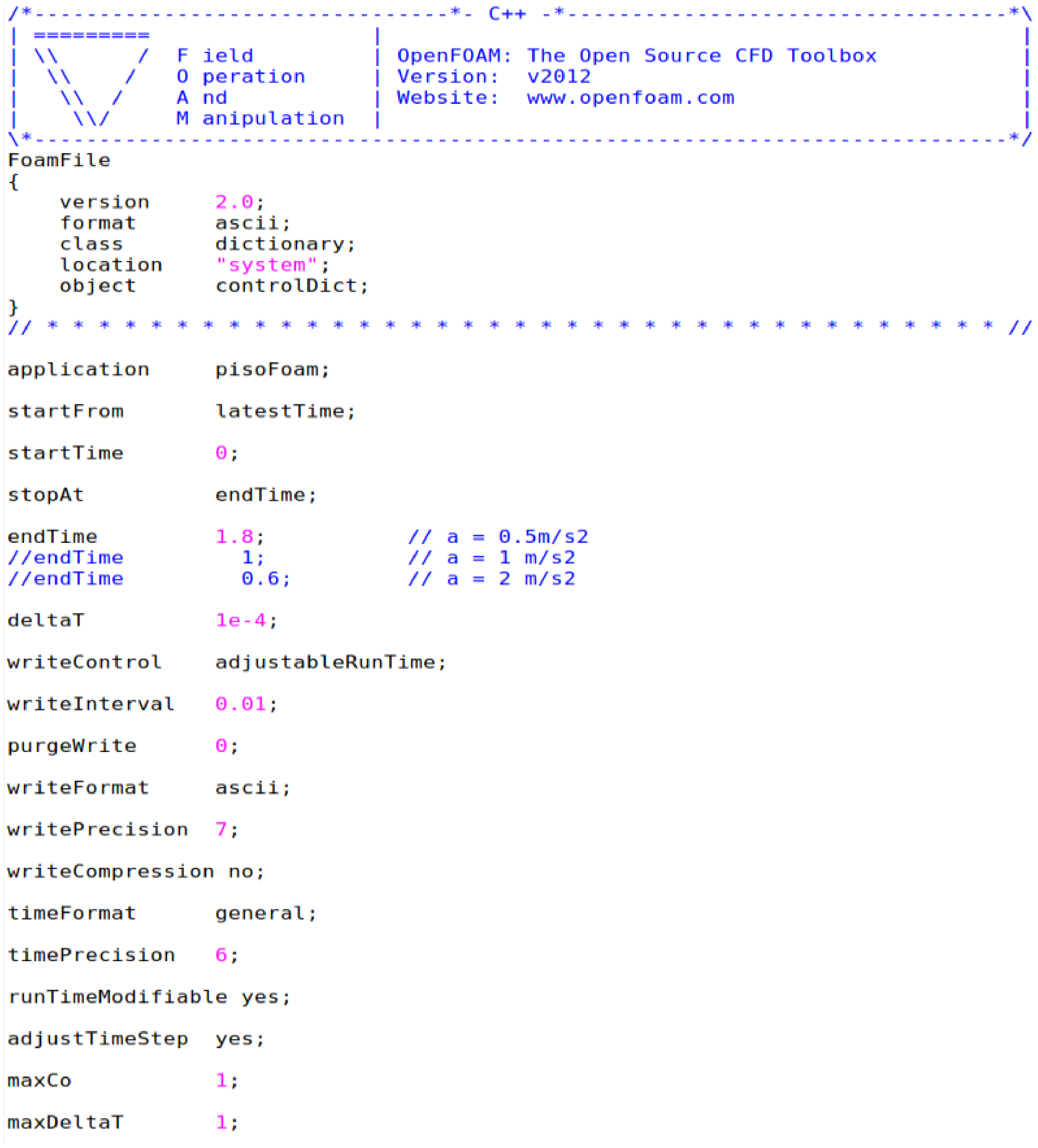
Figure 5: 0 directory: velocity

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

transportModel  Newtonian;

nu              nu [ 0 2 -1 0 0 0 0 ] 1.56e-06;

// ************************************************************************* //
```

Figure 6: constant directory: transportProperties

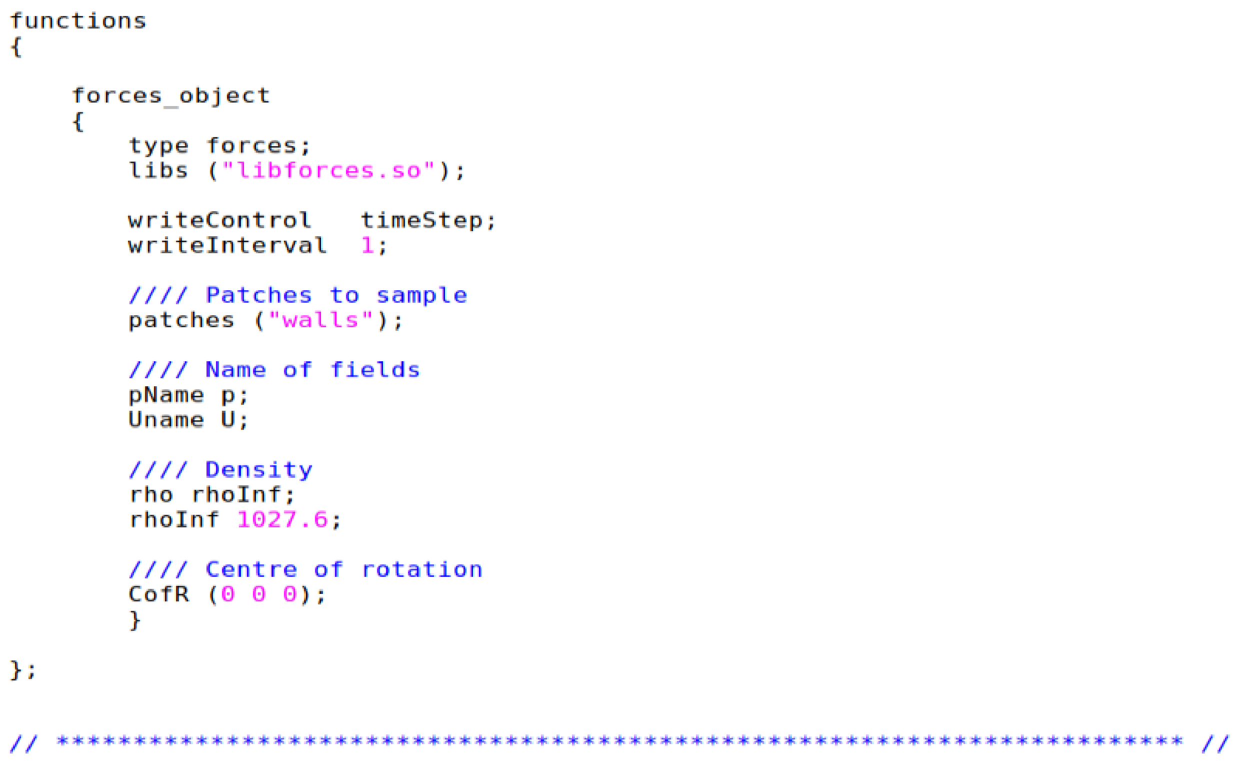
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


simulationType  RAS;

RAS
{
    RASModel        kOmegaSST;

    turbulence      on;

    printCoeffs     on;
}

// ************************************************************************* //
```

Figure 7: constant directory: turbulenceProperties



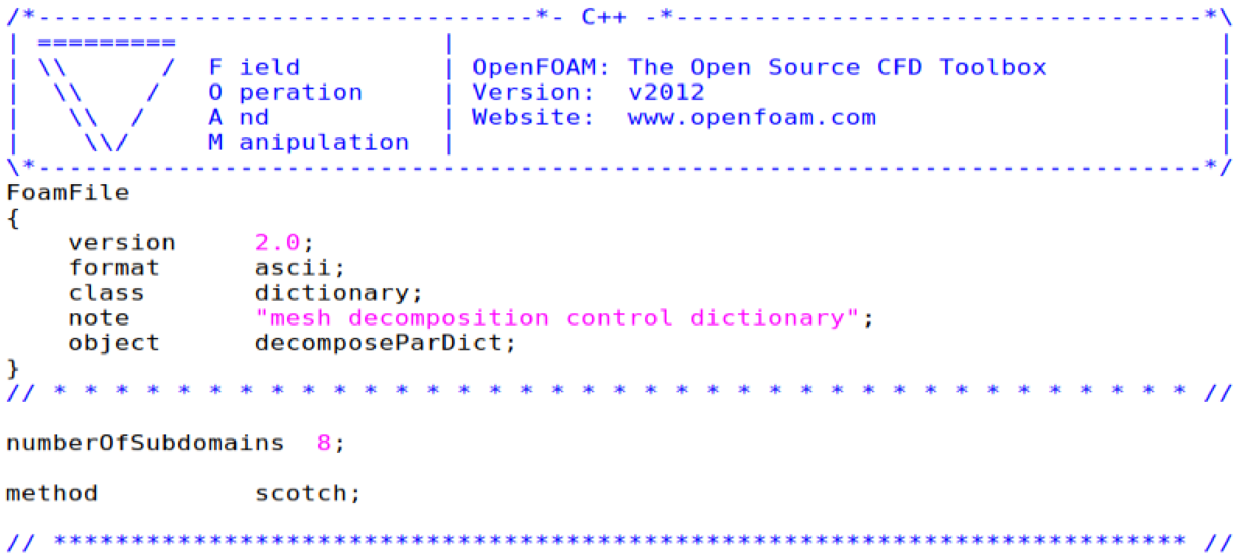
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     pisoFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         1.8;            // a = 0.5m/s2
//endTime         1;            // a = 1 m/s2
//endTime         0.6;          // a = 2 m/s2

deltaT          1e-4;

writeControl    adjustableRunTime;

writeInterval   0.01;

purgeWrite      0;

writeFormat     ascii;

writePrecision  7;

writeCompression no;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;

adjustTimeStep  yes;

maxCo           1;

maxDeltaT       1;
```

Figure 8: system directory: controlDict



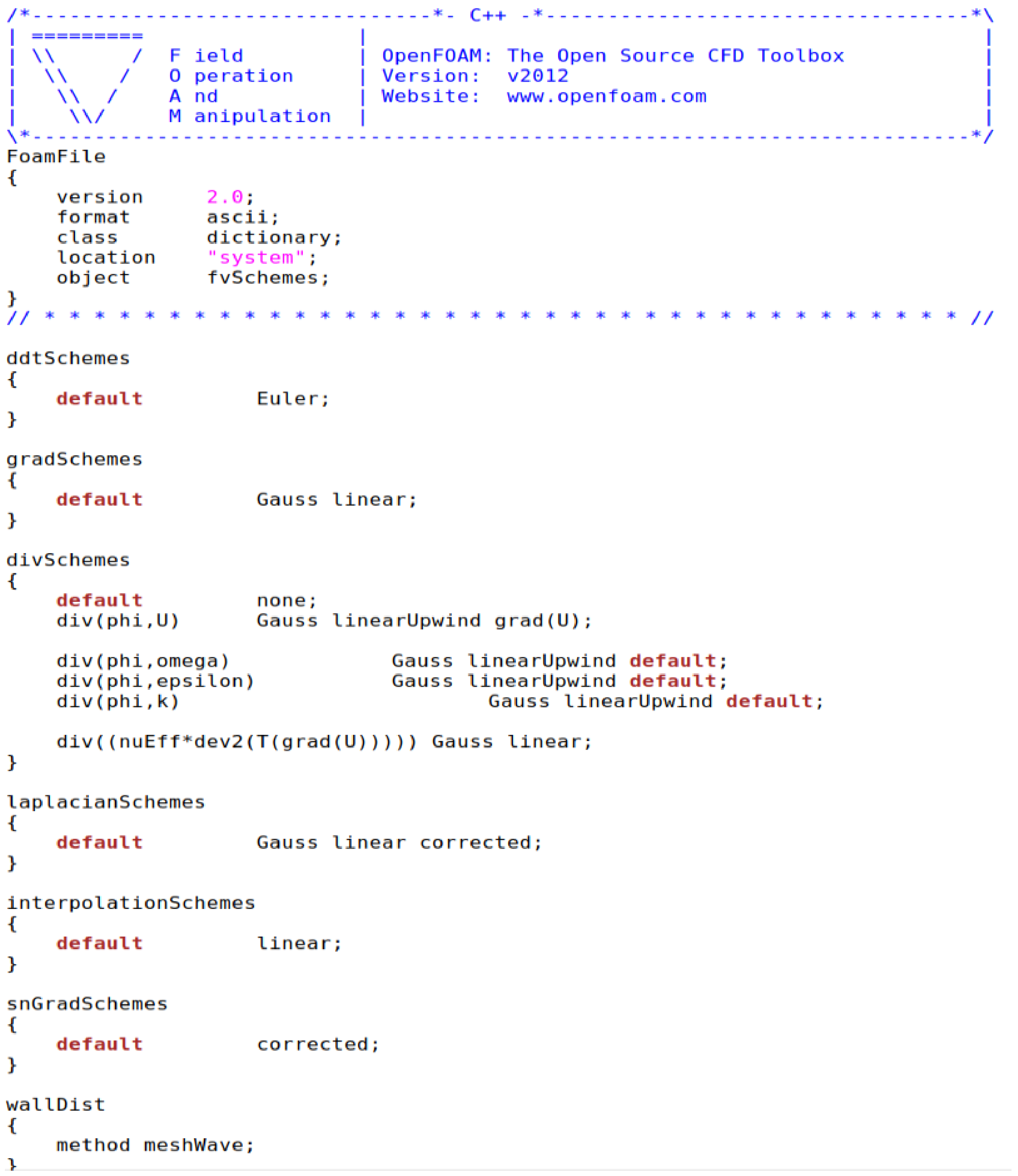
```
functions
{

    forces_object
    {
        type forces;
        libs ("libforces.so");

        writeControl   timeStep;
        writeInterval  1;

        //// Patches to sample
        patches ("walls");

        //// Name of fields
        pName p;
        Uname U;

        //// Density
        rho rhoInf;
        rhoInf 1027.6;

        //// Centre of rotation
        CofR (0 0 0);
        }

};

// ************************************************************************* //
```

Figure 9: system directory: controlDict (forcelibs)



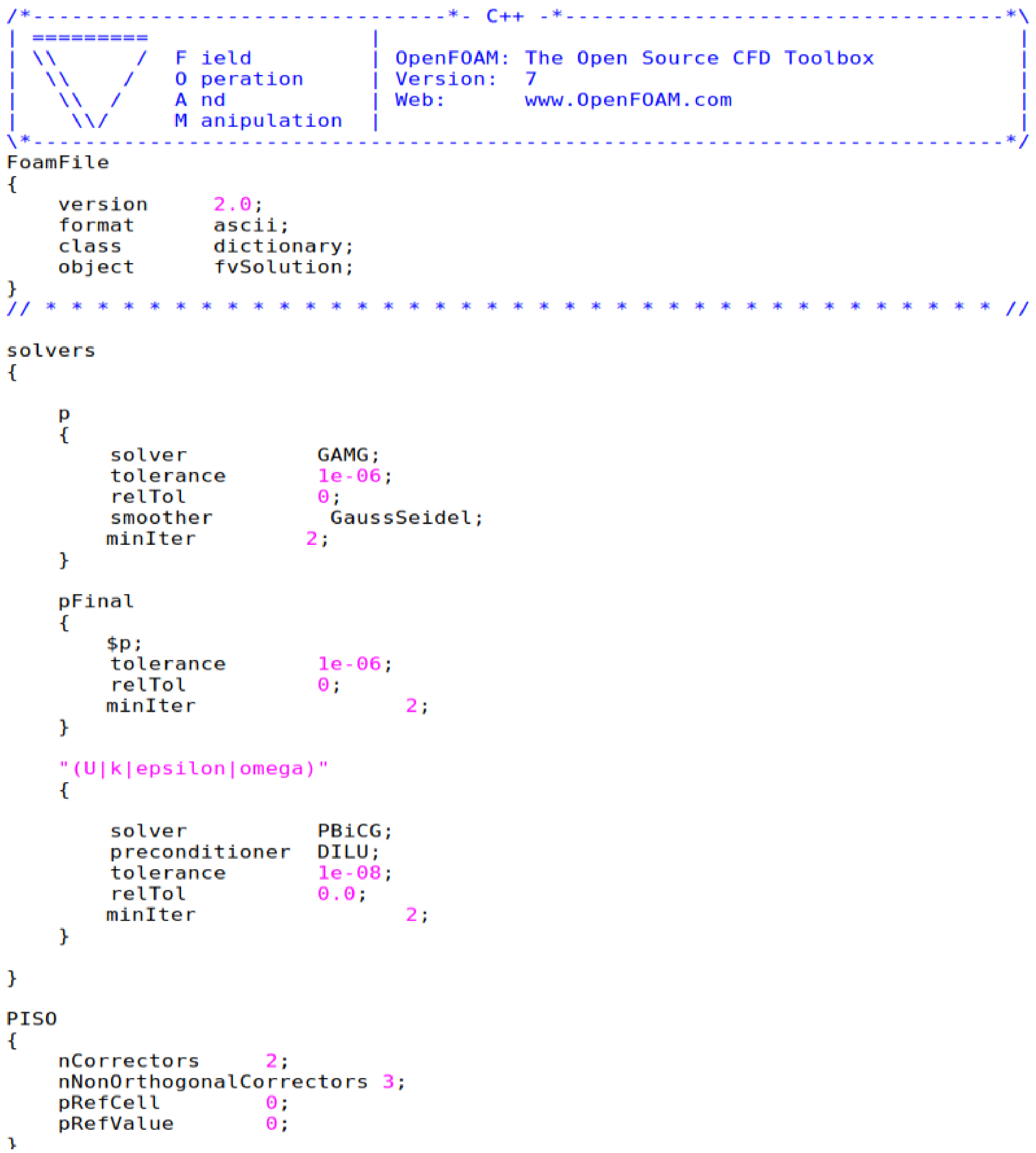
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    note        "mesh decomposition control dictionary";
    object      decomposeParDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains  8;

method          scotch;

// ************************************************************************* //
```

Figure 10: system directory: decomposeParDict



```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      Gauss linearUpwind grad(U);

    div(phi,omega)             Gauss linearUpwind default;
    div(phi,epsilon)           Gauss linearUpwind default;
    div(phi,k)                         Gauss linearUpwind default;

    div((nuEff*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

wallDist
{
    method meshWave;
}
```

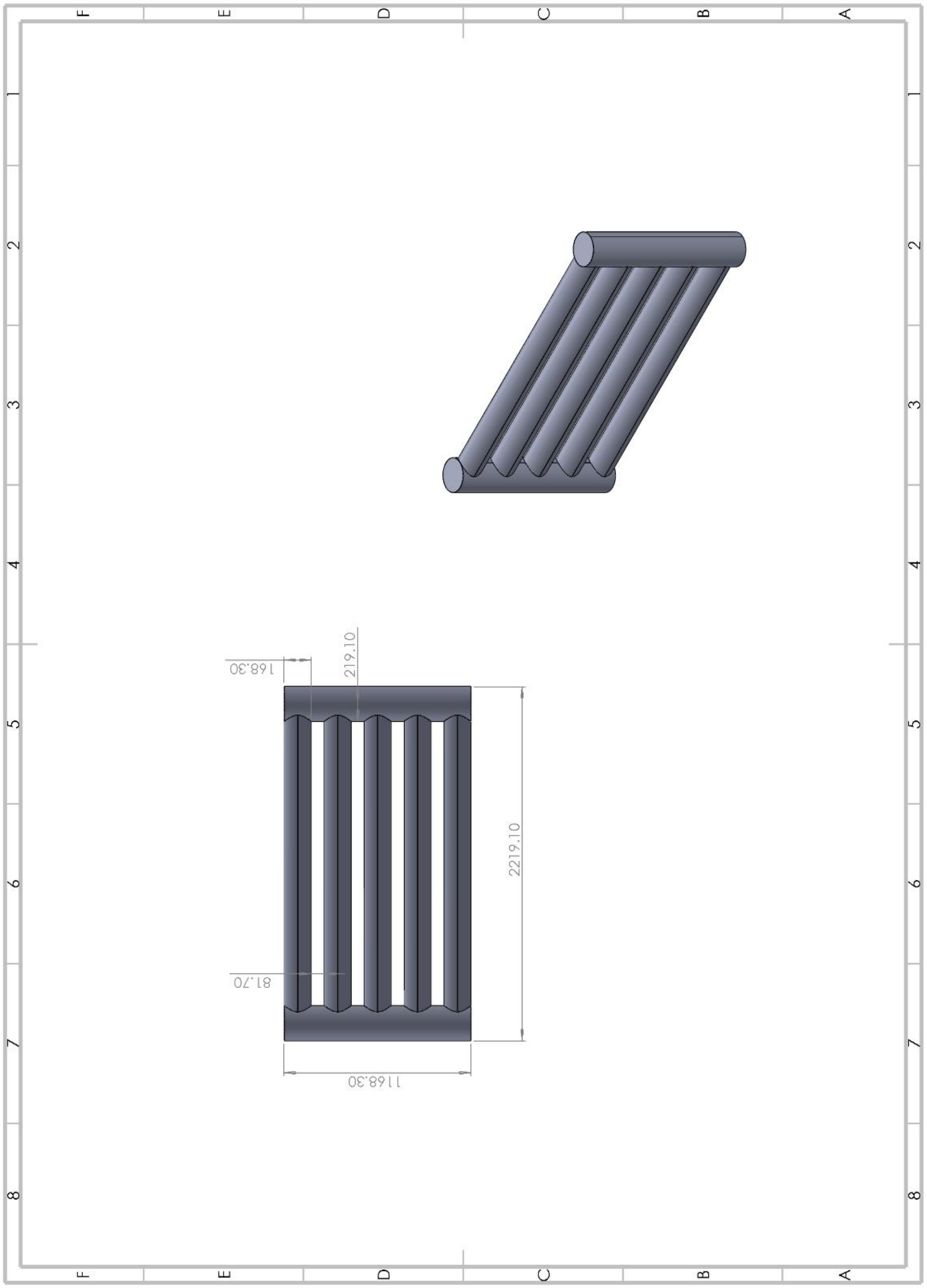
Figure 11: system directory: fvSchemes



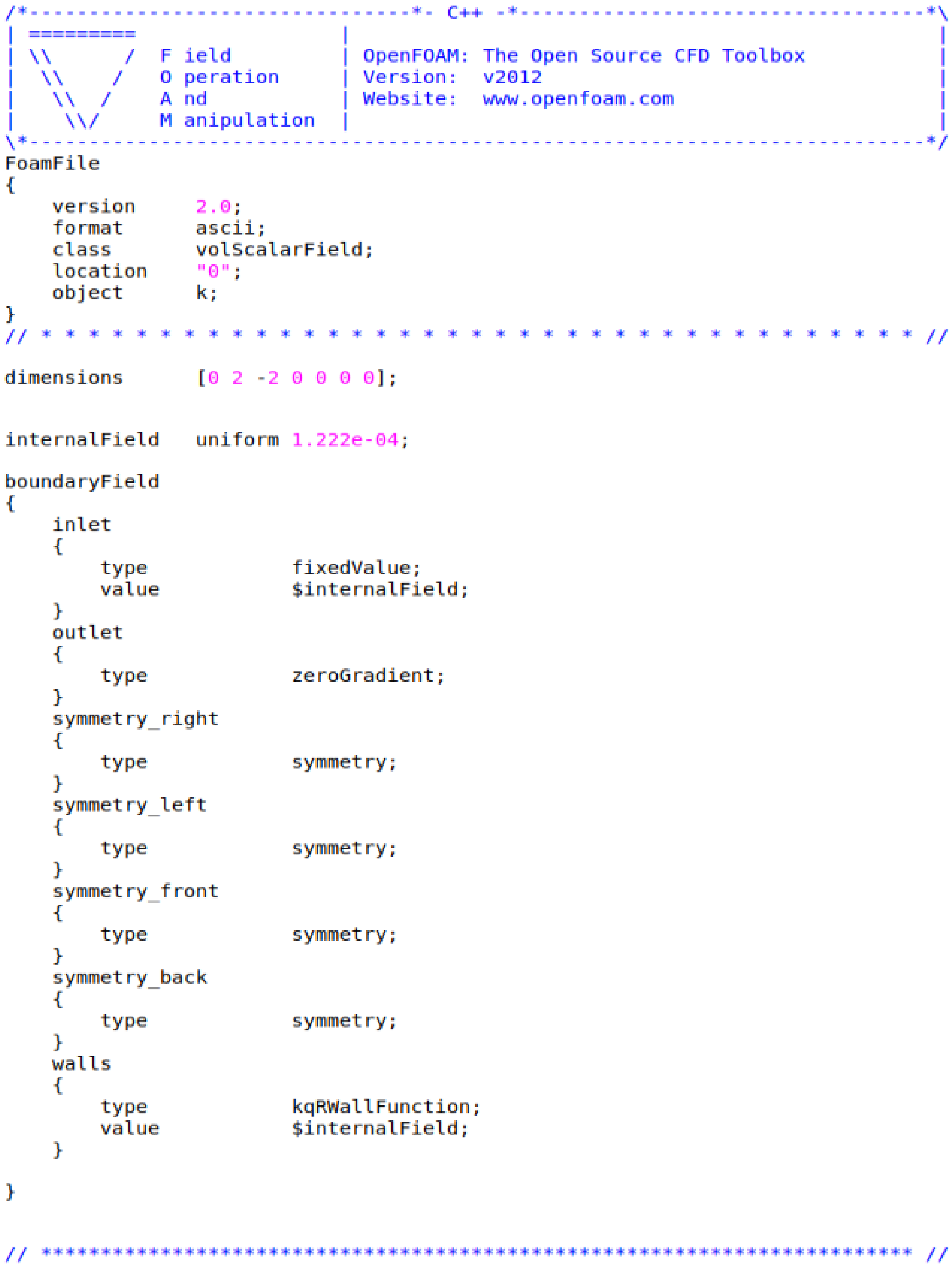
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  7                                     |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{

    p
    {
        solver          GAMG;
        tolerance       1e-06;
        relTol          0;
        smoother         GaussSeidel;
       minIter         2;
    }

    pFinal
    {
       $p;
        tolerance       1e-06;
        relTol          0;
       minIter                 2;
    }

    "(U|k|epsilon|omega)"
    {

        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-08;
        relTol          0.0;
       minIter                 2;
    }

}

PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 3;
    pRefCell        0;
    pRefValue       0;
}
```

Figure 12: system directory: fvSolution

Appendix H – Technical drawing of the cylinder frame



Appendix I – case-setup for the cylindrical frame at constant acceleration

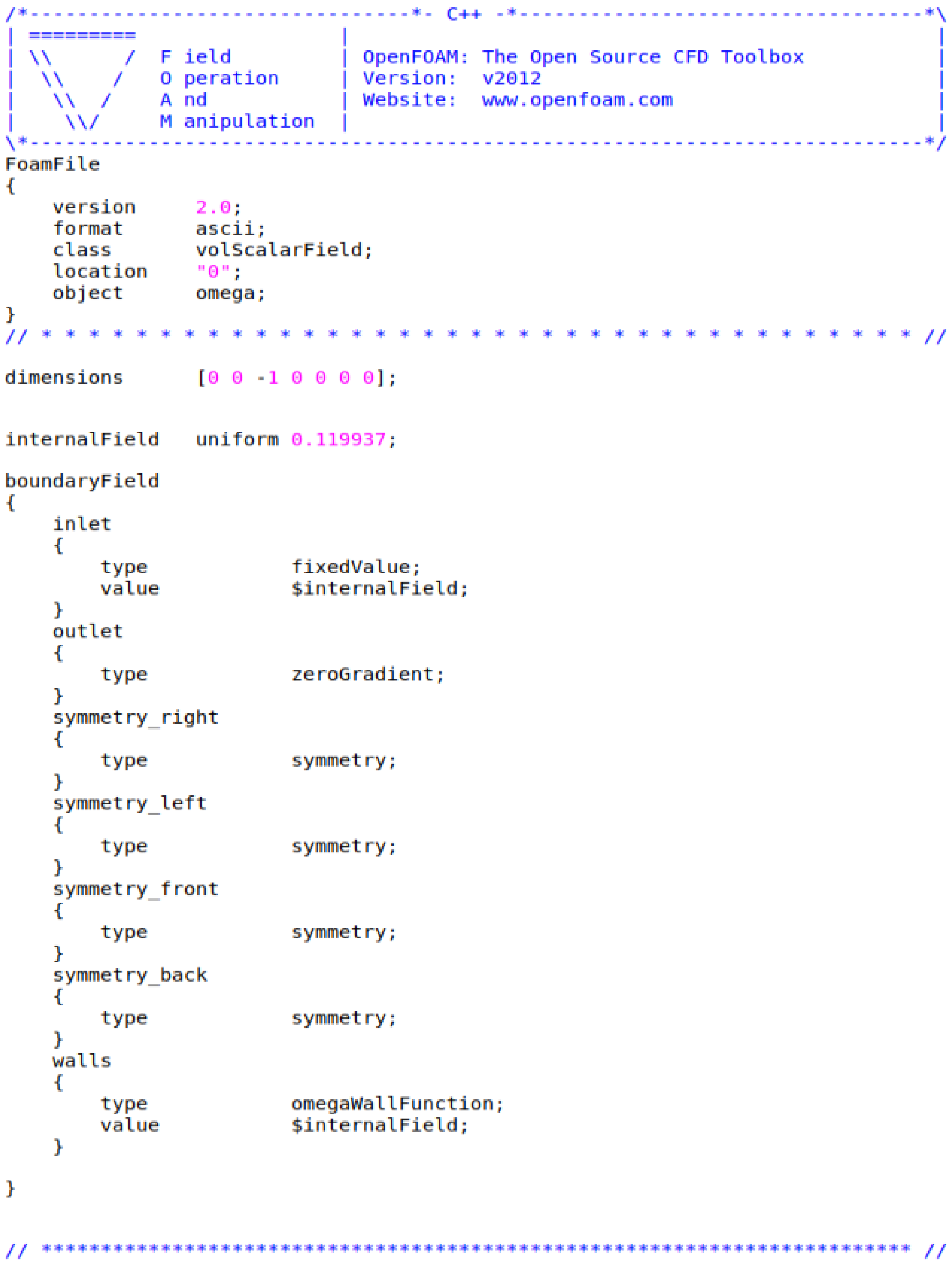


```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];


internalField   uniform 1.222e-04;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           $internalField;
    }
    outlet
    {
        type            zeroGradient;
    }
    symmetry_right
    {
        type            symmetry;
    }
    symmetry_left
    {
        type            symmetry;
    }
    symmetry_front
    {
        type            symmetry;
    }
    symmetry_back
    {
        type            symmetry;
    }
    walls
    {
        type            kqRWallFunction;
        value           $internalField;
    }

}


// ************************************************************************* //
```

Figure 1: 0 directory: turbulent kinetic energy

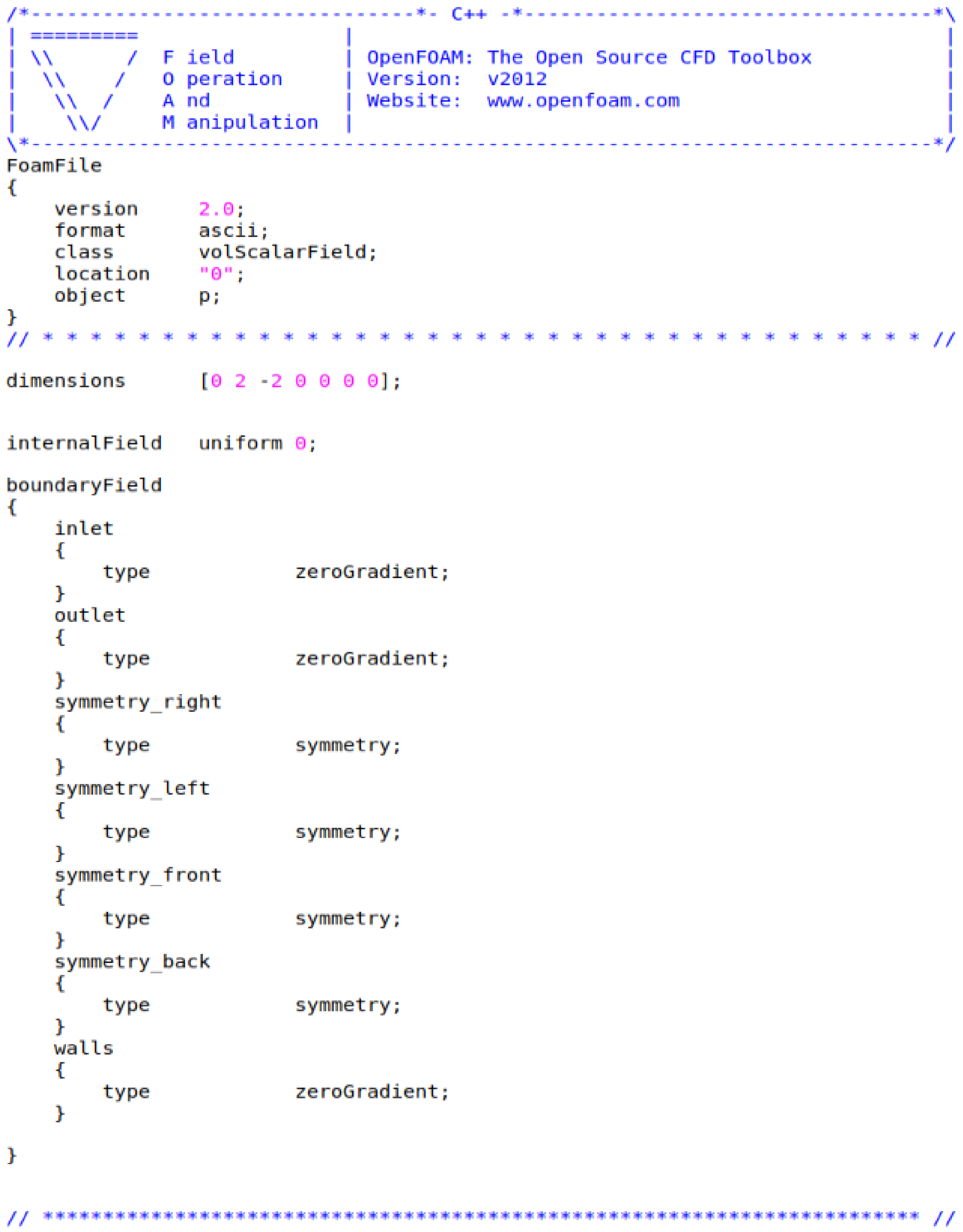
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];


internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            calculated;
        value           uniform 0;
    }
    outlet
    {
        type            calculated;
        value           uniform 0;
    }
    symmetry_right
    {
        type            symmetry;
    }
    symmetry_left
    {
        type            symmetry;
    }
    symmetry_front
    {
        type            symmetry;
    }
    symmetry_back
    {
        type            symmetry;
    }
    walls
    {
        type            nutkWallFunction;
        value           uniform 0;
    }
}


// ************************************************************************* //
```

Figure 2: 0 directory: eddy-viscosity



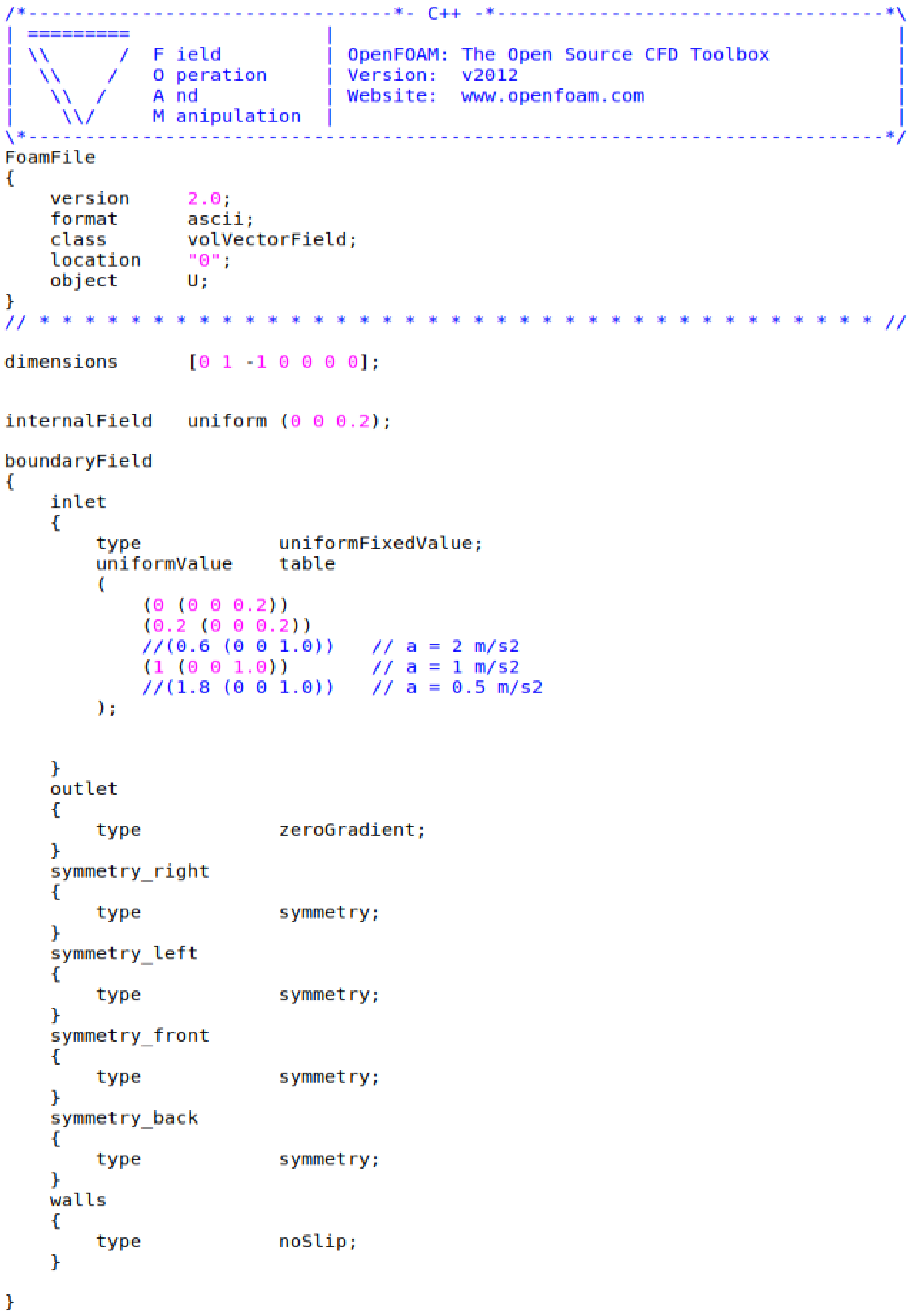
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      omega;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 -1 0 0 0 0];


internalField   uniform 0.119937;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           $internalField;
    }
    outlet
    {
        type            zeroGradient;
    }
    symmetry_right
    {
        type            symmetry;
    }
    symmetry_left
    {
        type            symmetry;
    }
    symmetry_front
    {
        type            symmetry;
    }
    symmetry_back
    {
        type            symmetry;
    }
    walls
    {
        type            omegaWallFunction;
        value           $internalField;
    }

}


// ************************************************************************* //
```

Figure 3: 0 directory: turbulent frequency



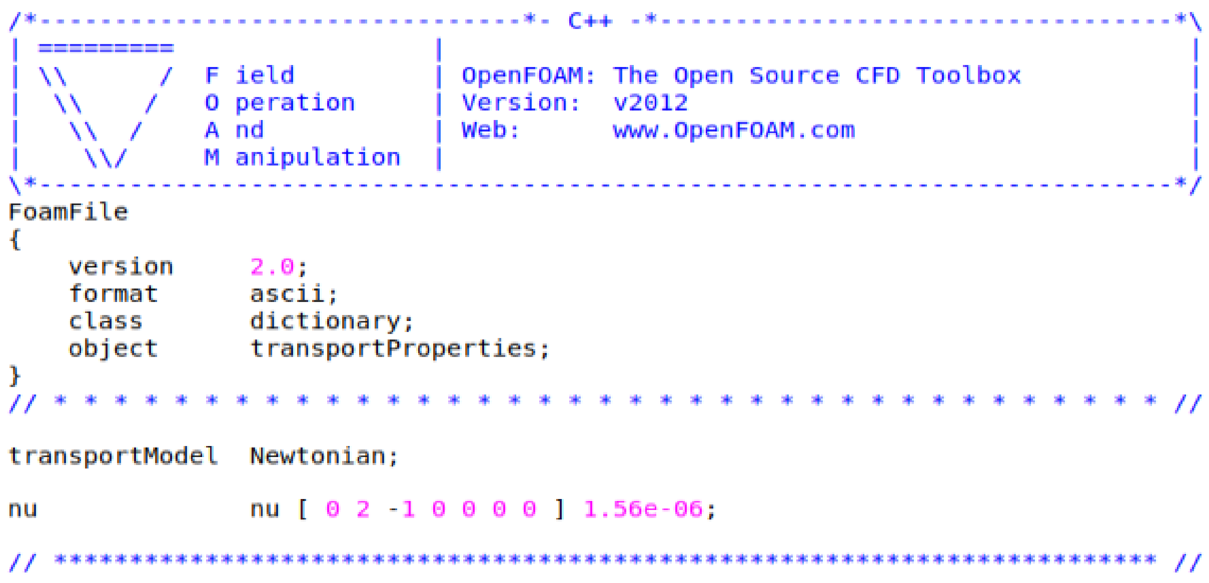
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];


internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            zeroGradient;
    }
    outlet
    {
        type            zeroGradient;
    }
    symmetry_right
    {
        type            symmetry;
    }
    symmetry_left
    {
        type            symmetry;
    }
    symmetry_front
    {
        type            symmetry;
    }
    symmetry_back
    {
        type            symmetry;
    }
    walls
    {
        type            zeroGradient;
    }

}

// ************************************************************************* //
```

Figure 4: 0 directory: kinematic pressure



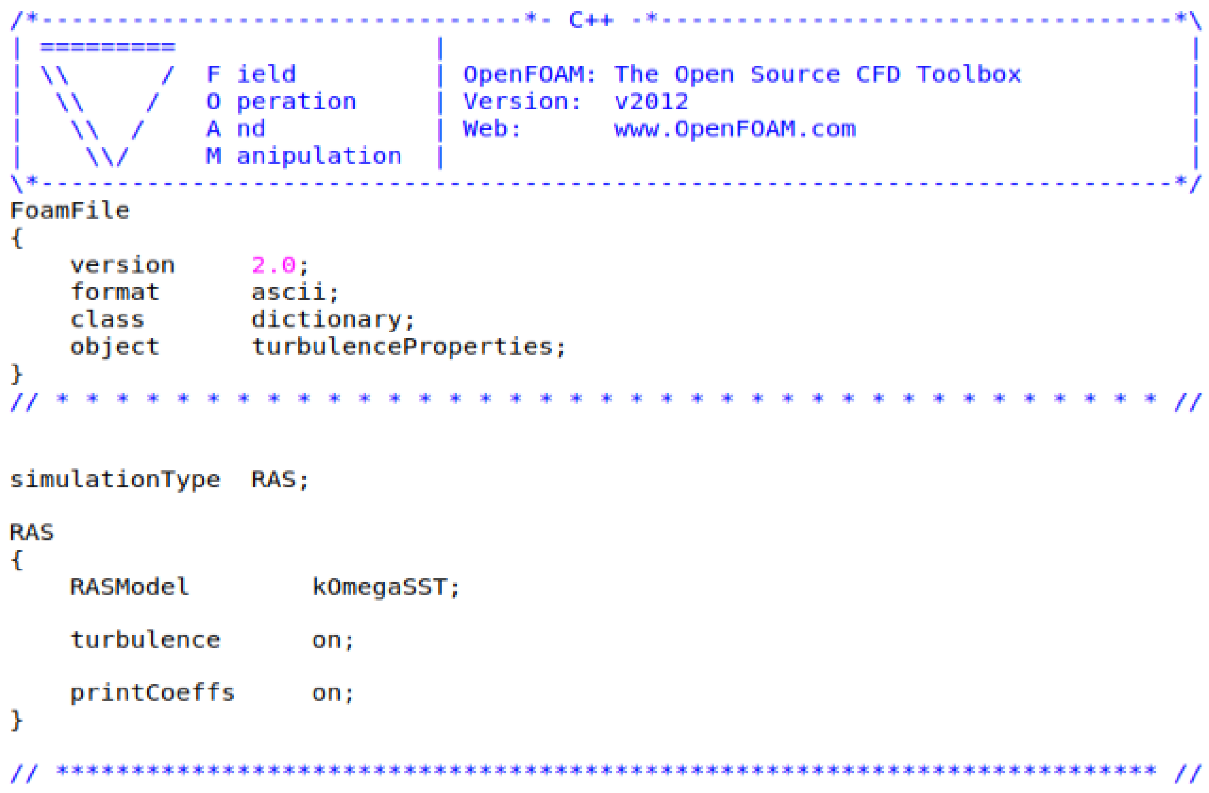
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];


internalField   uniform (0 0 0.2);

boundaryField
{
    inlet
    {
        type            uniformFixedValue;
        uniformValue    table
        (
            (0 (0 0 0.2))
            (0.2 (0 0 0.2))
            //(0.6 (0 0 1.0))   // a = 2 m/s2
            (1 (0 0 1.0))       // a = 1 m/s2
            //(1.8 (0 0 1.0))   // a = 0.5 m/s2
        );


    }
    outlet
    {
        type            zeroGradient;
    }
    symmetry_right
    {
        type            symmetry;
    }
    symmetry_left
    {
        type            symmetry;
    }
    symmetry_front
    {
        type            symmetry;
    }
    symmetry_back
    {
        type            symmetry;
    }
    walls
    {
        type            noSlip;
    }

}
```

Figure 5: 0 directory: velocity



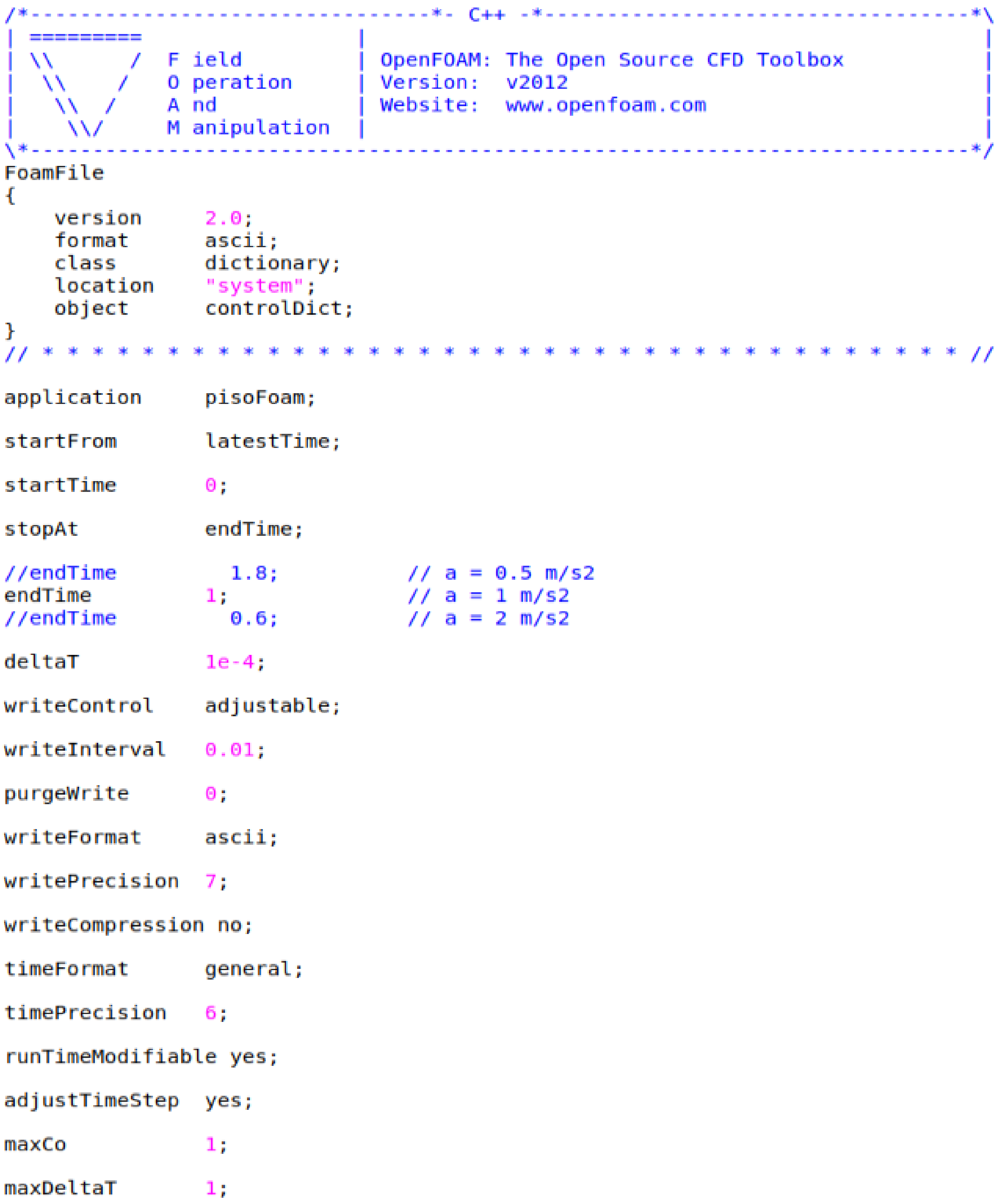
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

transportModel  Newtonian;

nu              nu [ 0 2 -1 0 0 0 0 ] 1.56e-06;

// ************************************************************************* //
```

Figure 6: constant directory: transportProperties



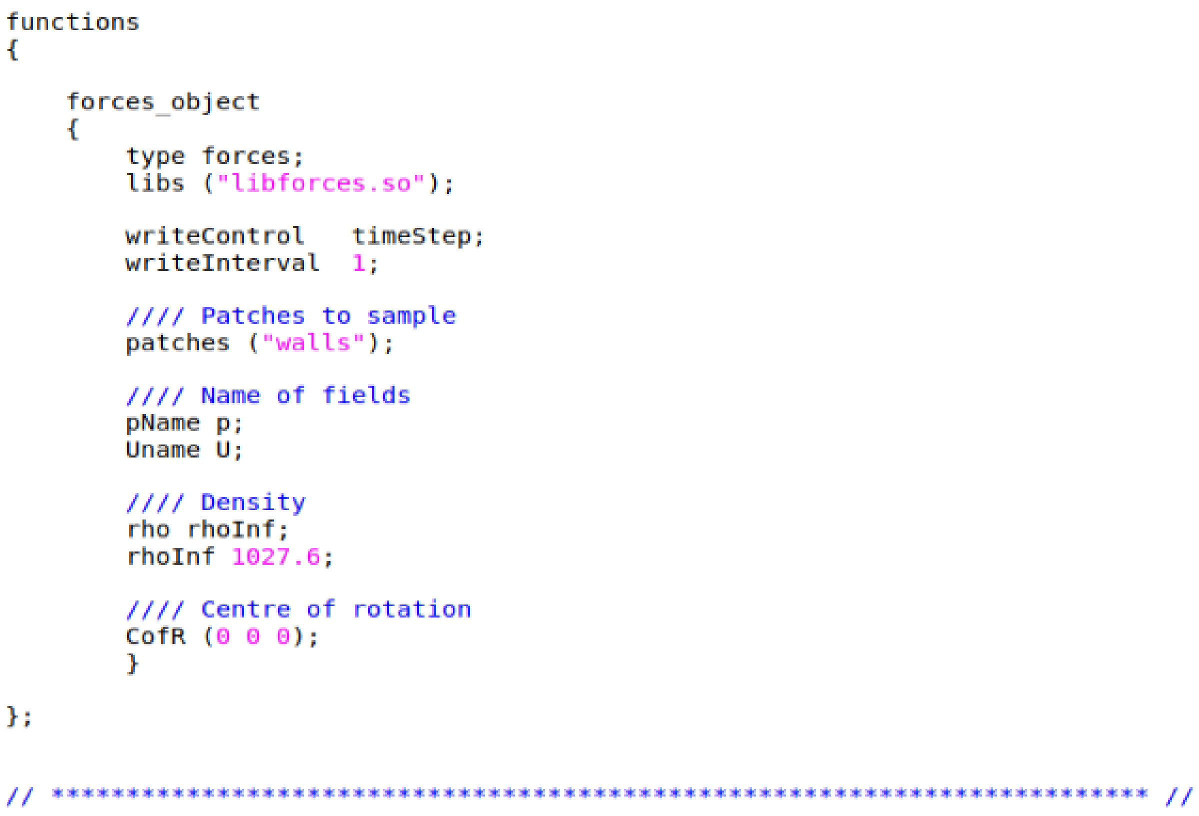
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


simulationType  RAS;

RAS
{
    RASModel        kOmegaSST;

    turbulence      on;

    printCoeffs     on;
}

// ************************************************************************* //
```

Figure 7: constant directory: turbulenceProperties



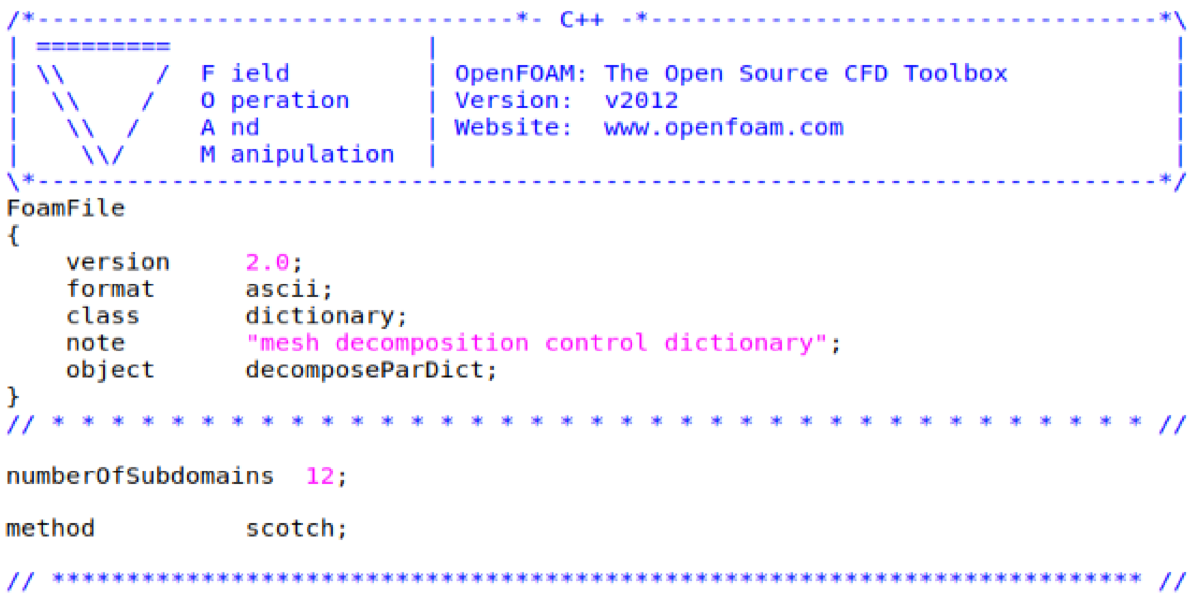
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     pisoFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

//endTime         1.8;          // a = 0.5 m/s2
endTime         1;              // a = 1 m/s2
//endTime         0.6;          // a = 2 m/s2

deltaT          1e-4;

writeControl    adjustable;

writeInterval   0.01;

purgeWrite      0;

writeFormat     ascii;

writePrecision  7;

writeCompression no;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;

adjustTimeStep  yes;

maxCo           1;

maxDeltaT       1;
```

Figure 8: system directory: controlDict



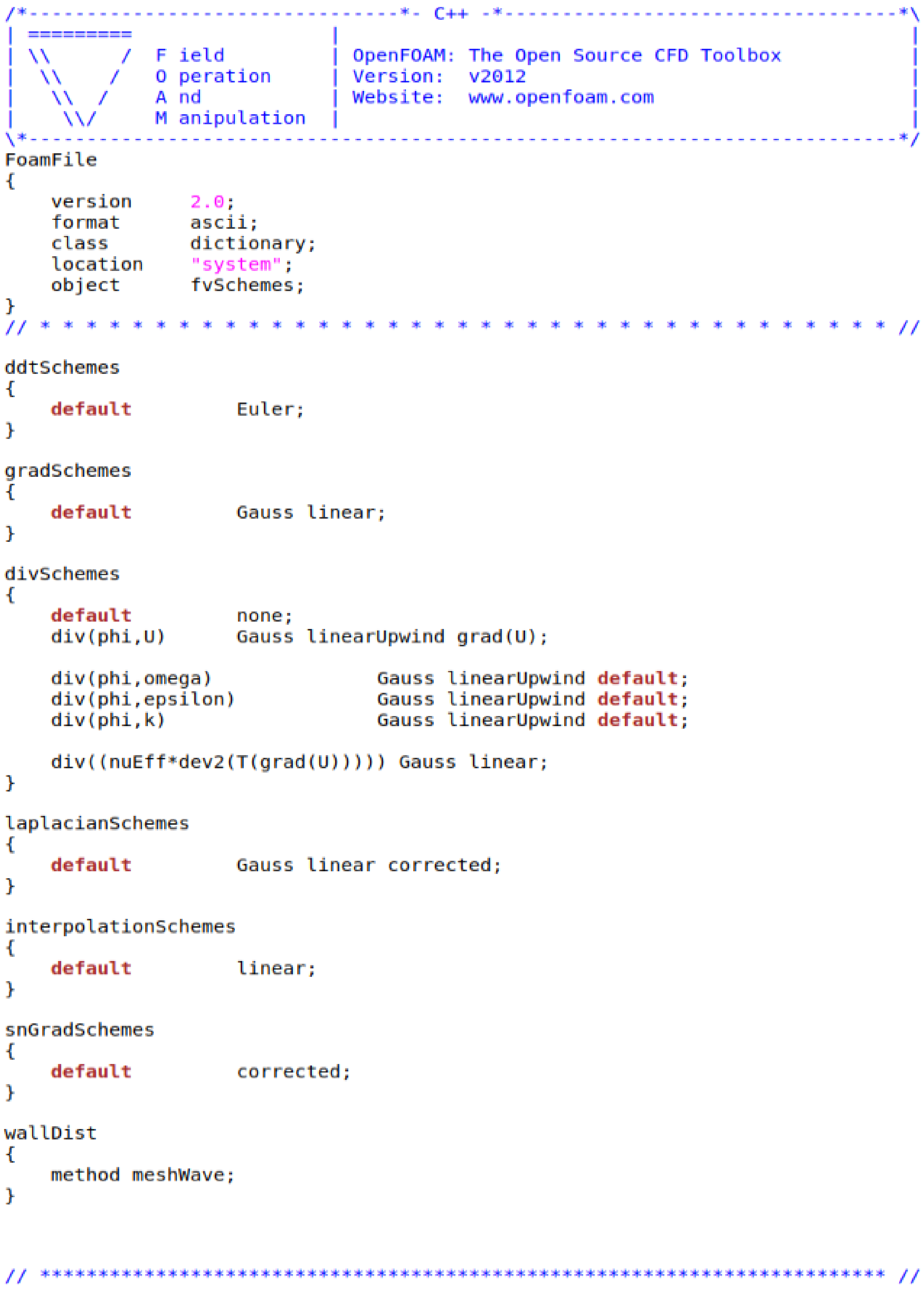
```
functions
{

    forces_object
    {
        type forces;
        libs ("libforces.so");

        writeControl   timeStep;
        writeInterval  1;

        //// Patches to sample
        patches ("walls");

        //// Name of fields
        pName p;
        Uname U;

        //// Density
        rho rhoInf;
        rhoInf 1027.6;

        //// Centre of rotation
        CofR (0 0 0);
        }

};

// ************************************************************************* //
```

Figure 9: system directory: controlDict (forcelibs)



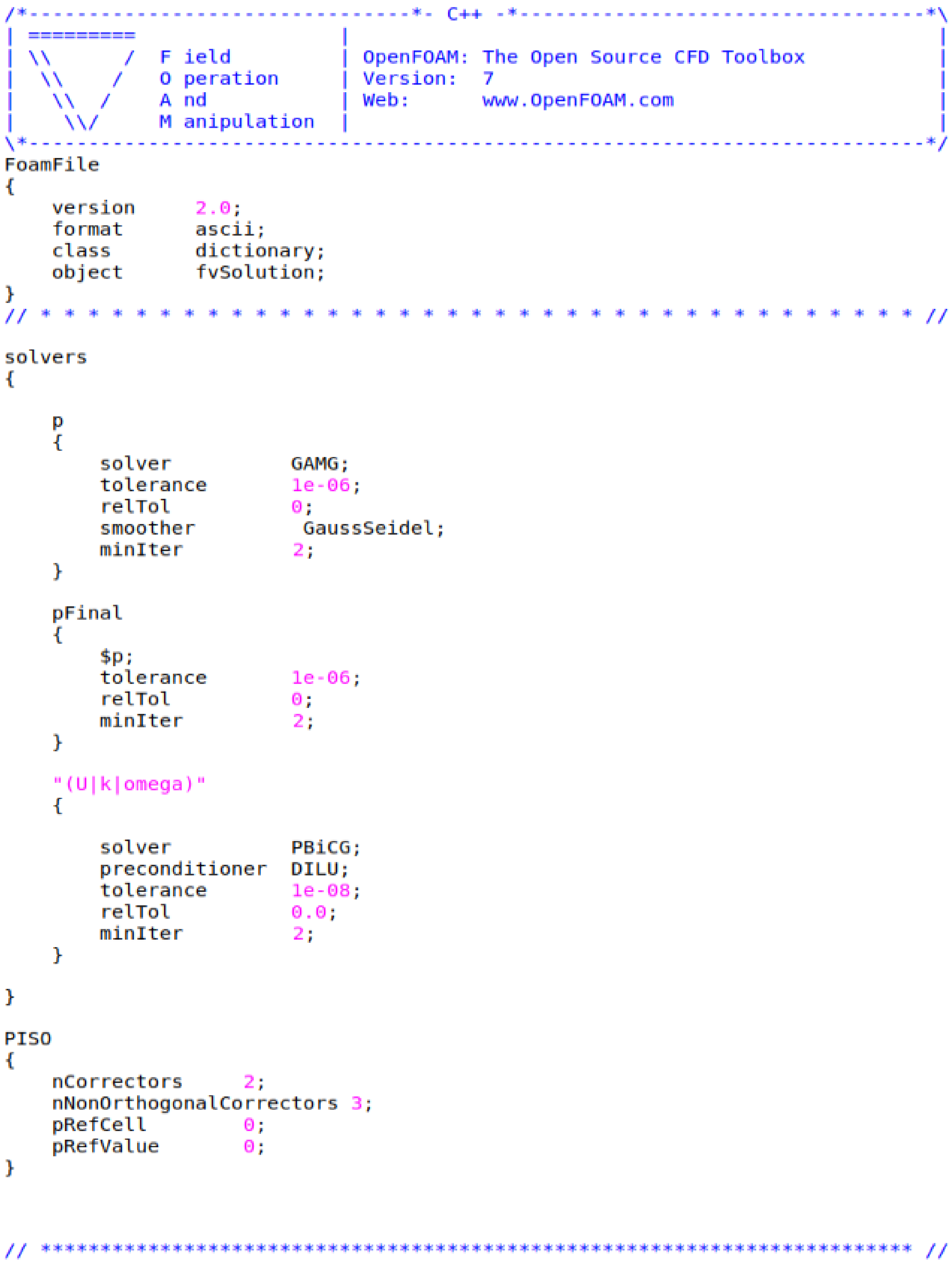
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    note        "mesh decomposition control dictionary";
    object      decomposeParDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains  12;

method          scotch;

// ************************************************************************* //
```

Figure 10: system directory: decomposeParDict



```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      Gauss linearUpwind grad(U);

    div(phi,omega)              Gauss linearUpwind default;
    div(phi,epsilon)            Gauss linearUpwind default;
    div(phi,k)                  Gauss linearUpwind default;

    div((nuEff*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

wallDist
{
    method meshWave;
}


// ************************************************************************* //
```

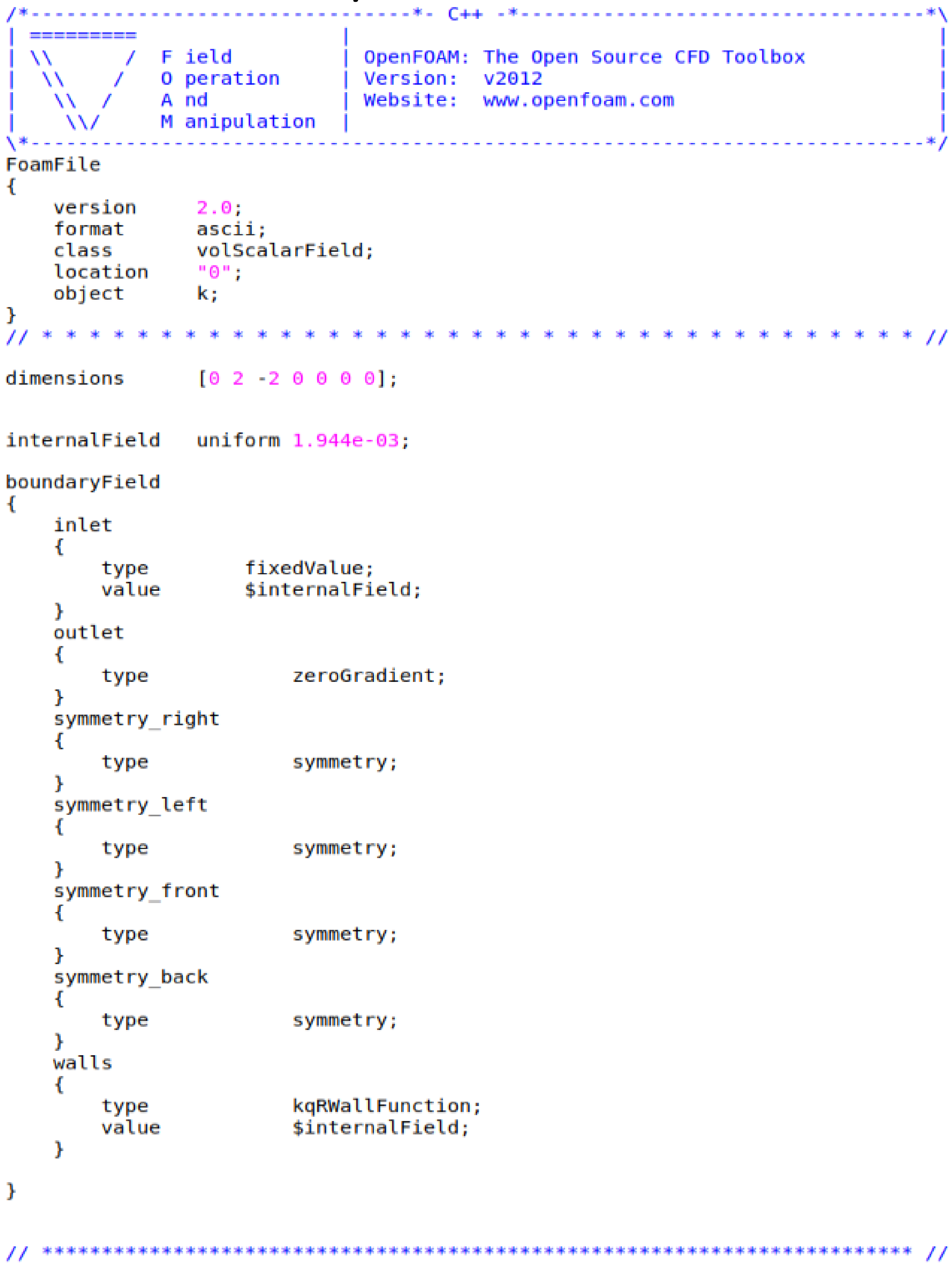
Figure 11: system directory: fvSchemes



```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  7                                     |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{

    p
    {
        solver          GAMG;
        tolerance       1e-06;
        relTol          0;
        smoother         GaussSeidel;
        minIter         2;
    }

    pFinal
    {
        $p;
        tolerance       1e-06;
        relTol          0;
        minIter         2;
    }

    "(U|k|omega)"
    {

        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-08;
        relTol          0.0;
        minIter         2;
    }

}

PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 3;
    pRefCell        0;
    pRefValue       0;
}


// ************************************************************************* //
```

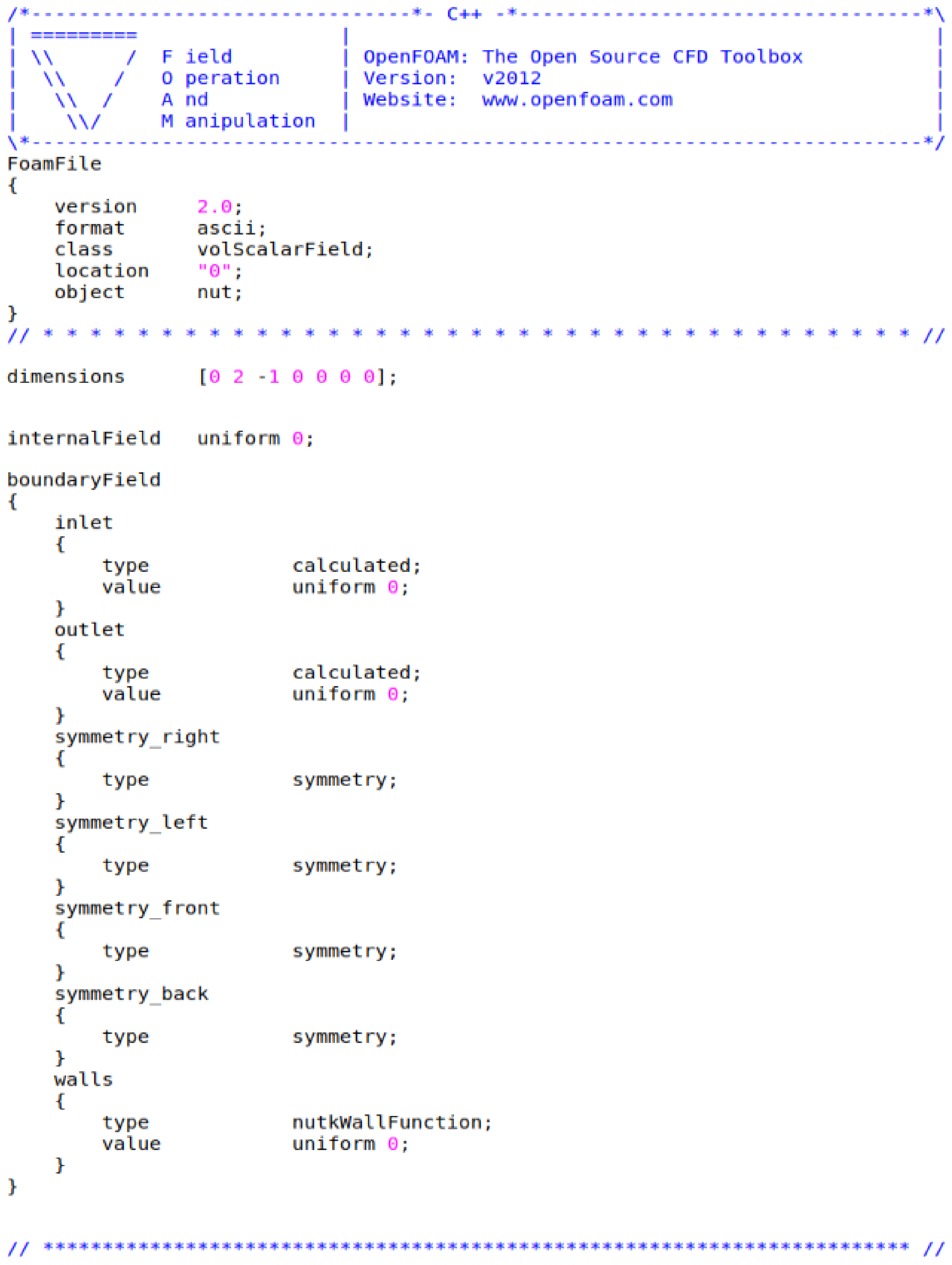
Figure 12: system directory: fvSolution

Appendix J – case-setup for the cylindrical frame at constant velocity



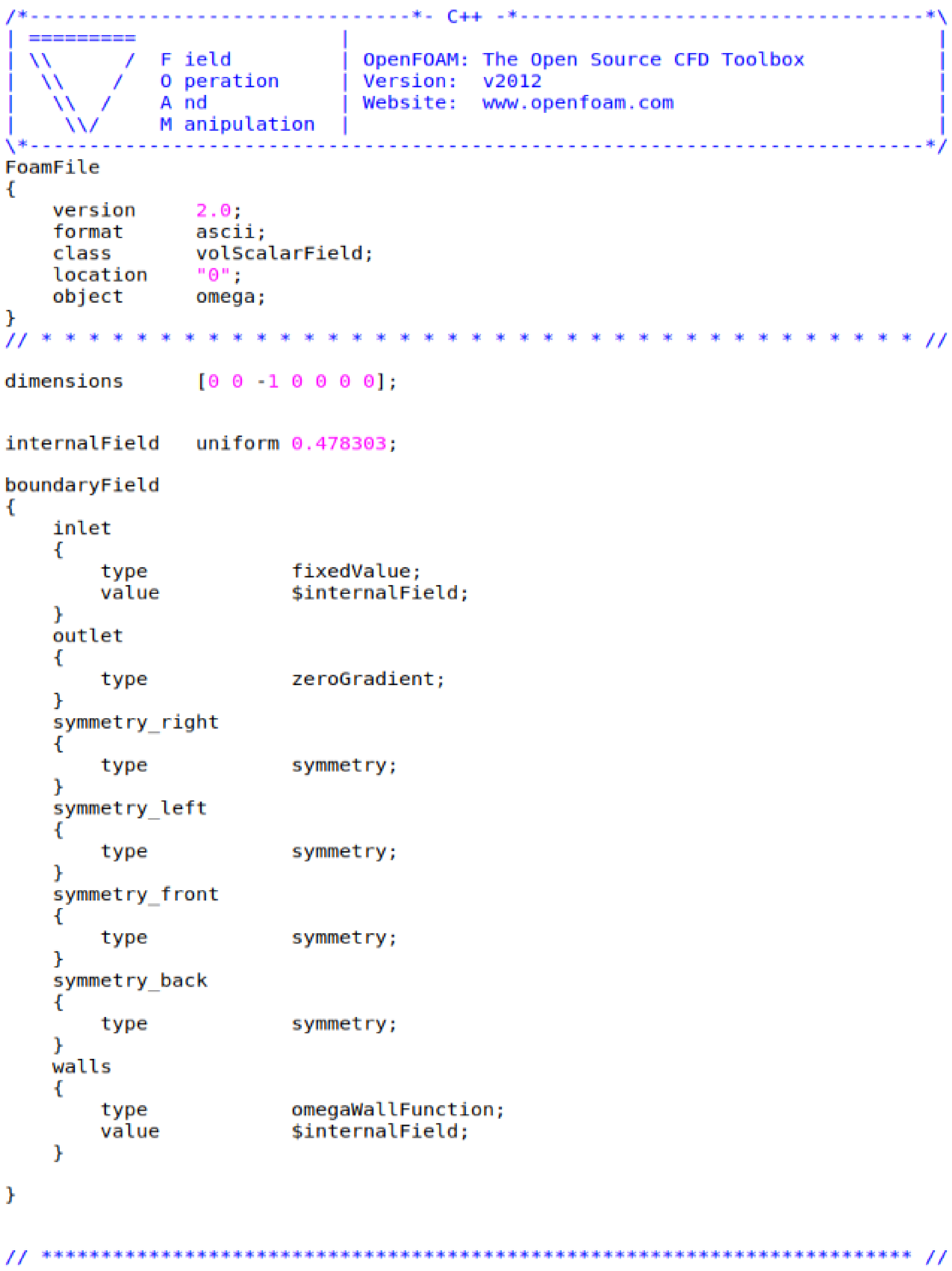
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];


internalField   uniform 1.944e-03;

boundaryField
{
    inlet
    {
        type        fixedValue;
        value       $internalField;
    }
    outlet
    {
        type            zeroGradient;
    }
    symmetry_right
    {
        type            symmetry;
    }
    symmetry_left
    {
        type            symmetry;
    }
    symmetry_front
    {
        type            symmetry;
    }
    symmetry_back
    {
        type            symmetry;
    }
    walls
    {
        type            kqRWallFunction;
        value           $internalField;
    }

}


// ************************************************************************* //
```

Figure 1: 0 directory: turbulent kinetic energy



```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];


internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            calculated;
        value           uniform 0;
    }
    outlet
    {
        type            calculated;
        value           uniform 0;
    }
    symmetry_right
    {
        type            symmetry;
    }
    symmetry_left
    {
        type            symmetry;
    }
    symmetry_front
    {
        type            symmetry;
    }
    symmetry_back
    {
        type            symmetry;
    }
    walls
    {
        type            nutkWallFunction;
        value           uniform 0;
    }
}


// ************************************************************************* //
```

Figure 2: 0 directory: eddy-viscosity



```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      omega;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 -1 0 0 0 0];


internalField   uniform 0.478303;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           $internalField;
    }
    outlet
    {
        type            zeroGradient;
    }
    symmetry_right
    {
        type            symmetry;
    }
    symmetry_left
    {
        type            symmetry;
    }
    symmetry_front
    {
        type            symmetry;
    }
    symmetry_back
    {
        type            symmetry;
    }
    walls
    {
        type            omegaWallFunction;
        value           $internalField;
    }

}


// ************************************************************************* //
```

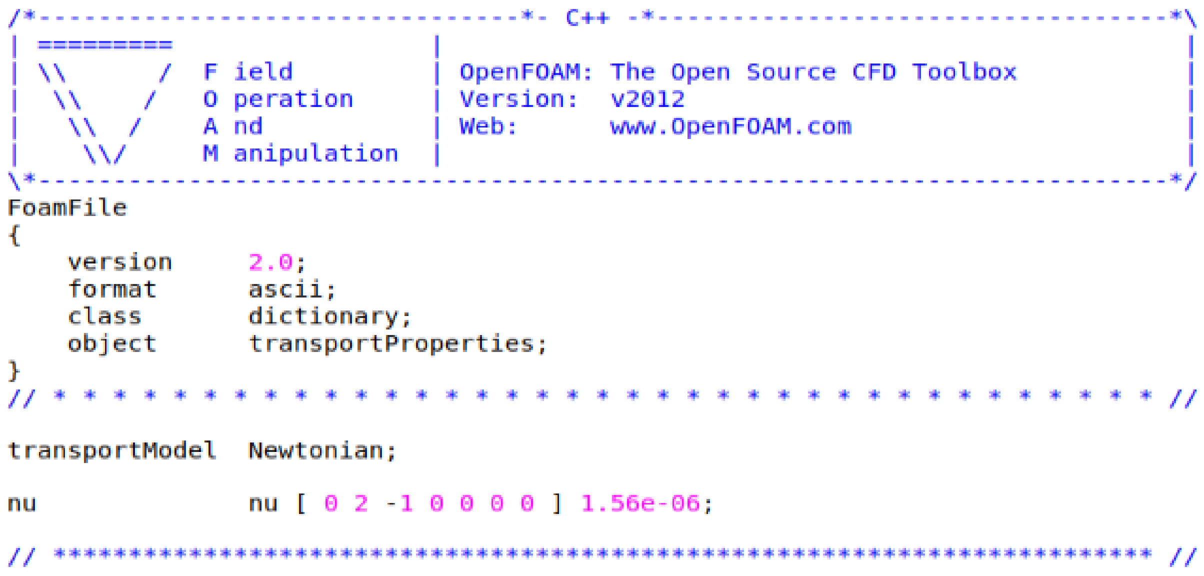
Figure 3: 0 directory: turbulent frequency

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];


internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            zeroGradient;
    }
    outlet
    {
        type            zeroGradient;
    }
    symmetry_right
    {
        type            symmetry;
    }
    symmetry_left
    {
        type            symmetry;
    }
    symmetry_front
    {
        type            symmetry;
    }
    symmetry_back
    {
        type            symmetry;
    }
    walls
    {
        type            zeroGradient;
    }

}


// ************************************************************************* //
```

Figure 4: 0 directory: kinematic pressure

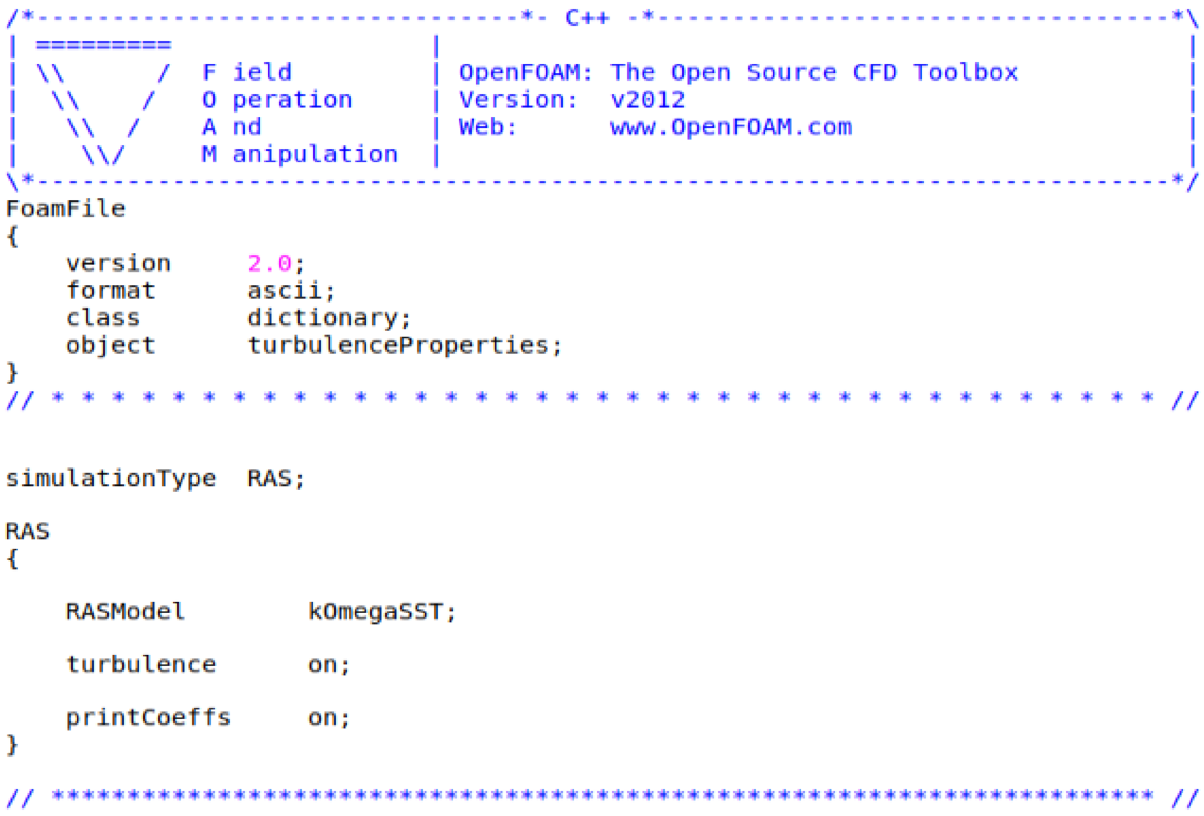
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];


internalField   uniform (0 0 0.8);

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform (0 0 0.8);
    }
    outlet
    {
        type            zeroGradient;
    }
    symmetry_right
    {
        type            symmetry;
    }
    symmetry_left
    {
        type            symmetry;
    }
    symmetry_front
    {
        type            symmetry;
    }
    symmetry_back
    {
        type            symmetry;
    }
    walls
    {
        type            noSlip;
    }

}


// ************************************************************************* //
```

Figure 5: 0 directory: velocity



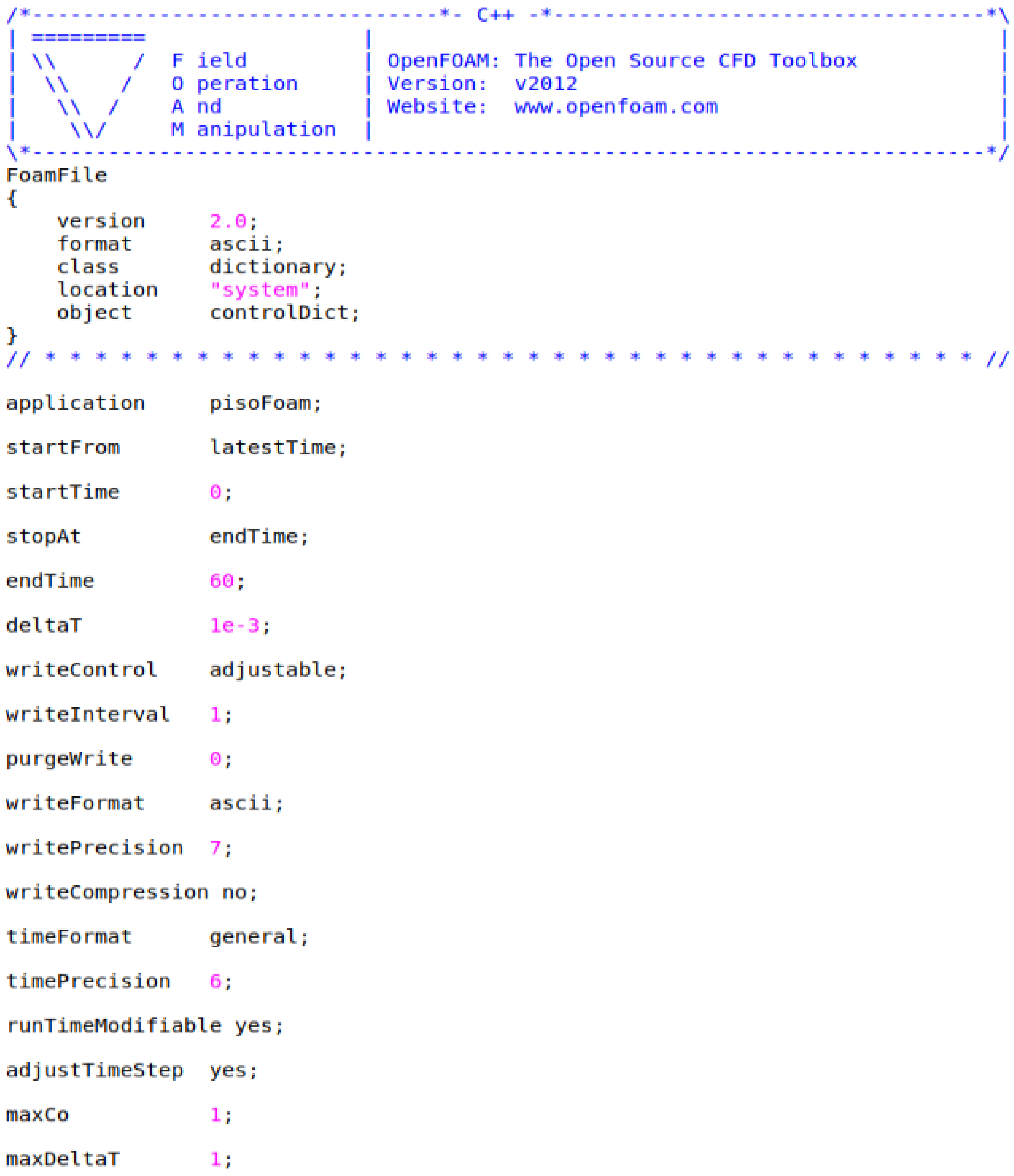
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

transportModel  Newtonian;

nu              nu [ 0 2 -1 0 0 0 0 ] 1.56e-06;

// ************************************************************************* //
```

Figure 6: constant directory: transportProperties



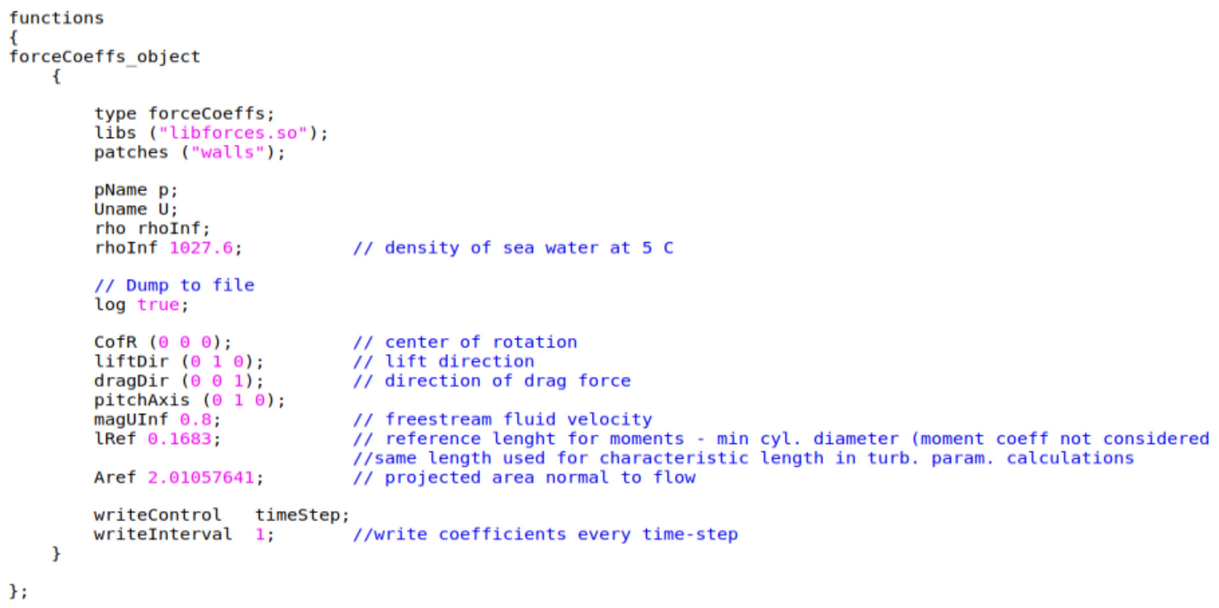
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


simulationType  RAS;

RAS
{

    RASModel        kOmegaSST;

    turbulence      on;

    printCoeffs     on;
}

// ************************************************************************* //
```

Figure 7: constant directory: turbulenceProperties



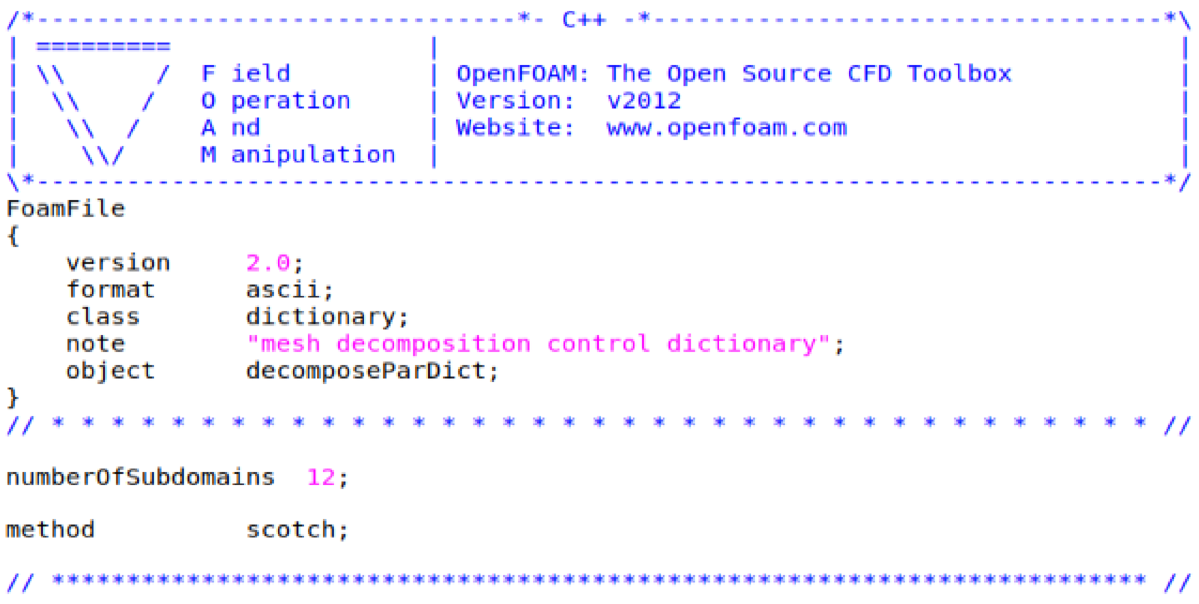
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     pisoFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         60;

deltaT          1e-3;

writeControl    adjustable;

writeInterval   1;

purgeWrite      0;

writeFormat     ascii;

writePrecision  7;

writeCompression no;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;

adjustTimeStep  yes;

maxCo           1;

maxDeltaT       1;
```

Figure 8: system directory: controlDict



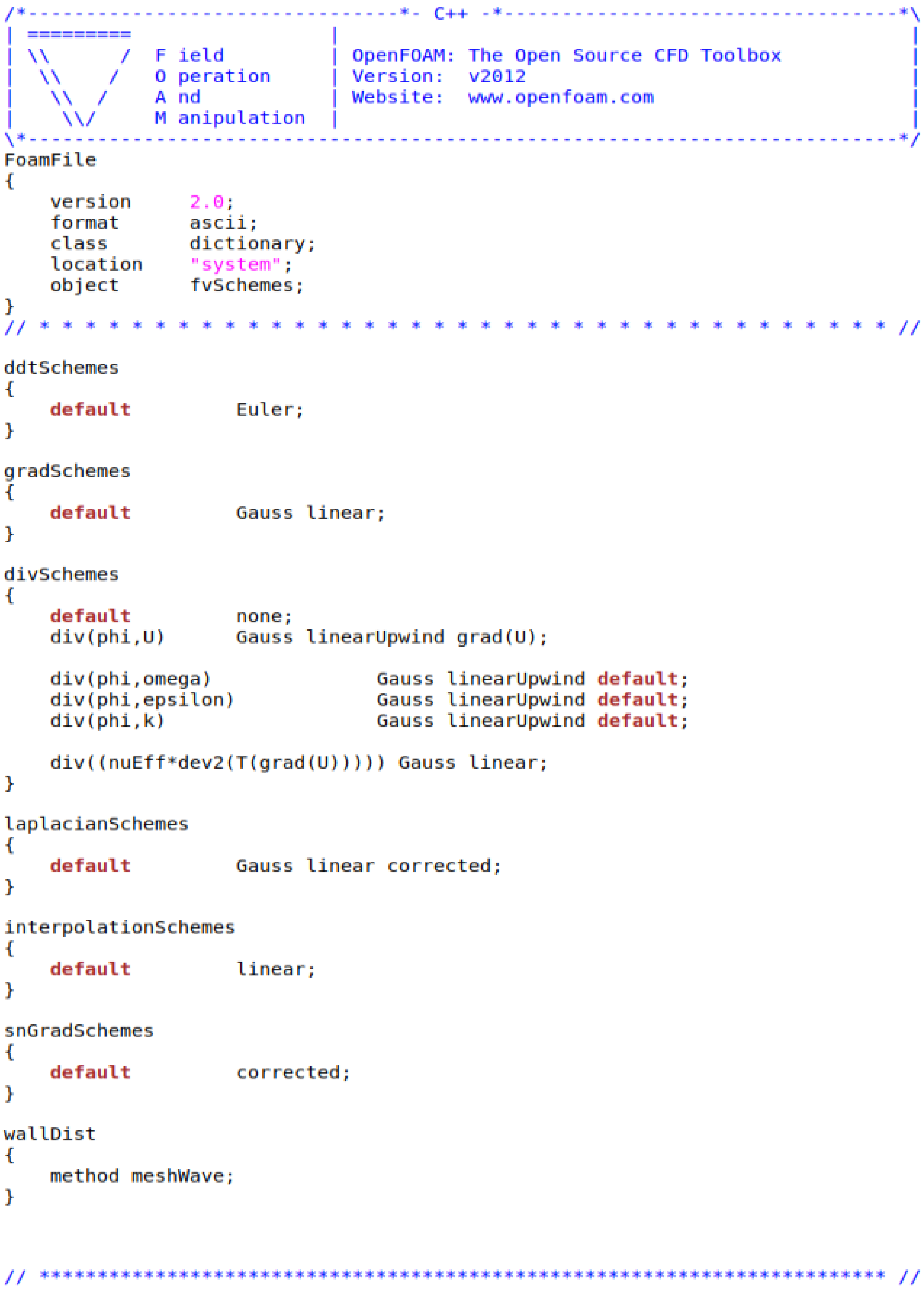
```
functions
{
forceCoeffs_object
    {

        type forceCoeffs;
        libs ("libforces.so");
        patches ("walls");

        pName p;
        Uname U;
        rho rhoInf;
        rhoInf 1027.6;          // density of sea water at 5 C

        // Dump to file
        log true;

        CofR (0 0 0);           // center of rotation
        liftDir (0 1 0);        // lift direction
        dragDir (0 0 1);        // direction of drag force
        pitchAxis (0 1 0);
        magUInf 0.8;            // freestream fluid velocity
        lRef 0.1683;            // reference lenght for moments - min cyl. diameter (moment coeff not considered
                                //same length used for characteristic length in turb. param. calculations
        Aref 2.01057641;        // projected area normal to flow

        writeControl   timeStep;
        writeInterval  1;       //write coefficients every time-step
    }

};
```

Figure 9: system directory: controlDict (libforces)



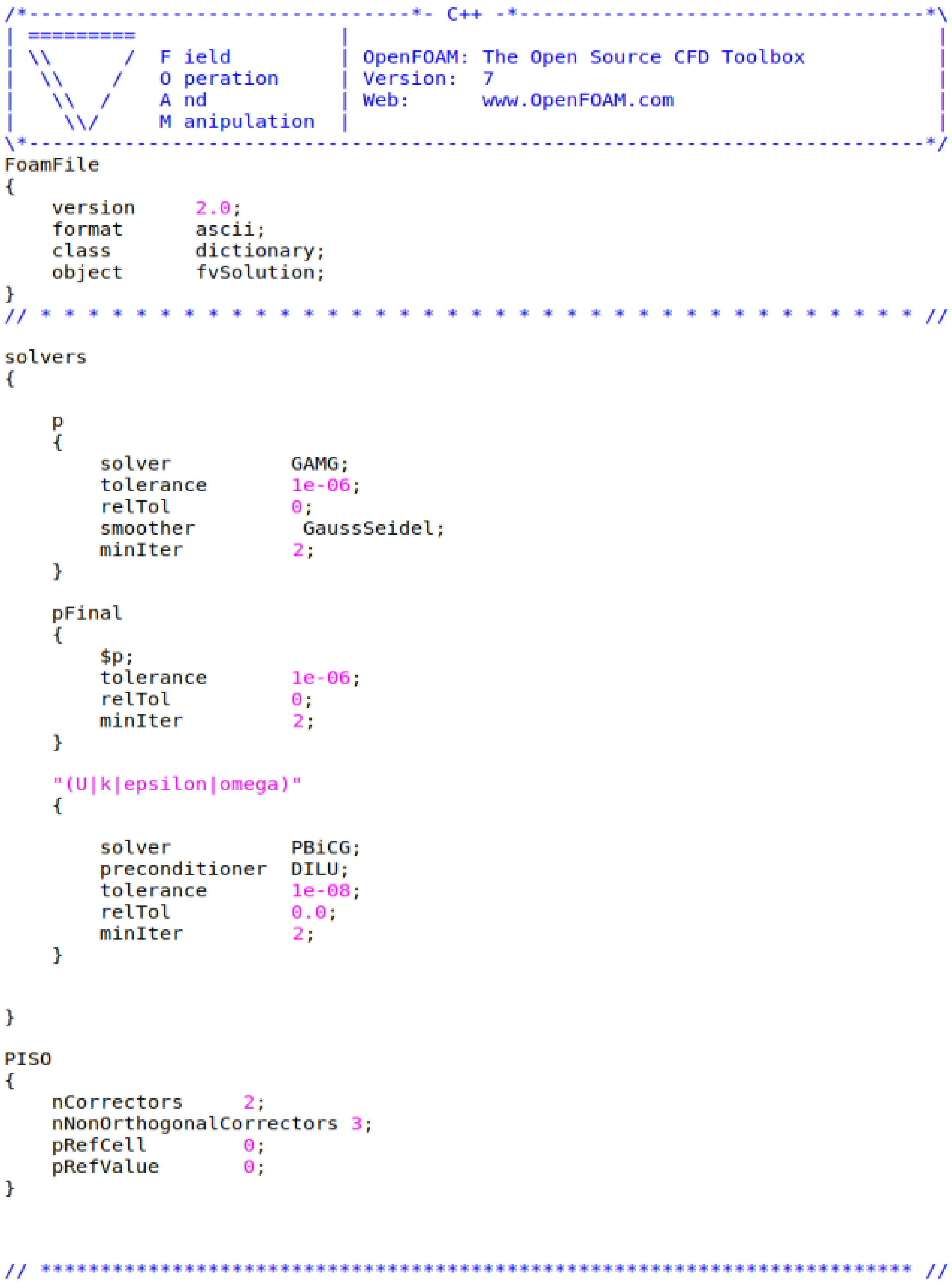
```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    note        "mesh decomposition control dictionary";
    object      decomposeParDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains  12;

method          scotch;

// ************************************************************************* //
```

Figure 10: system directory: decomposeParDict



```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      Gauss linearUpwind grad(U);

    div(phi,omega)              Gauss linearUpwind default;
    div(phi,epsilon)            Gauss linearUpwind default;
    div(phi,k)                  Gauss linearUpwind default;

    div((nuEff*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

wallDist
{
    method meshWave;
}


// ************************************************************************* //
```

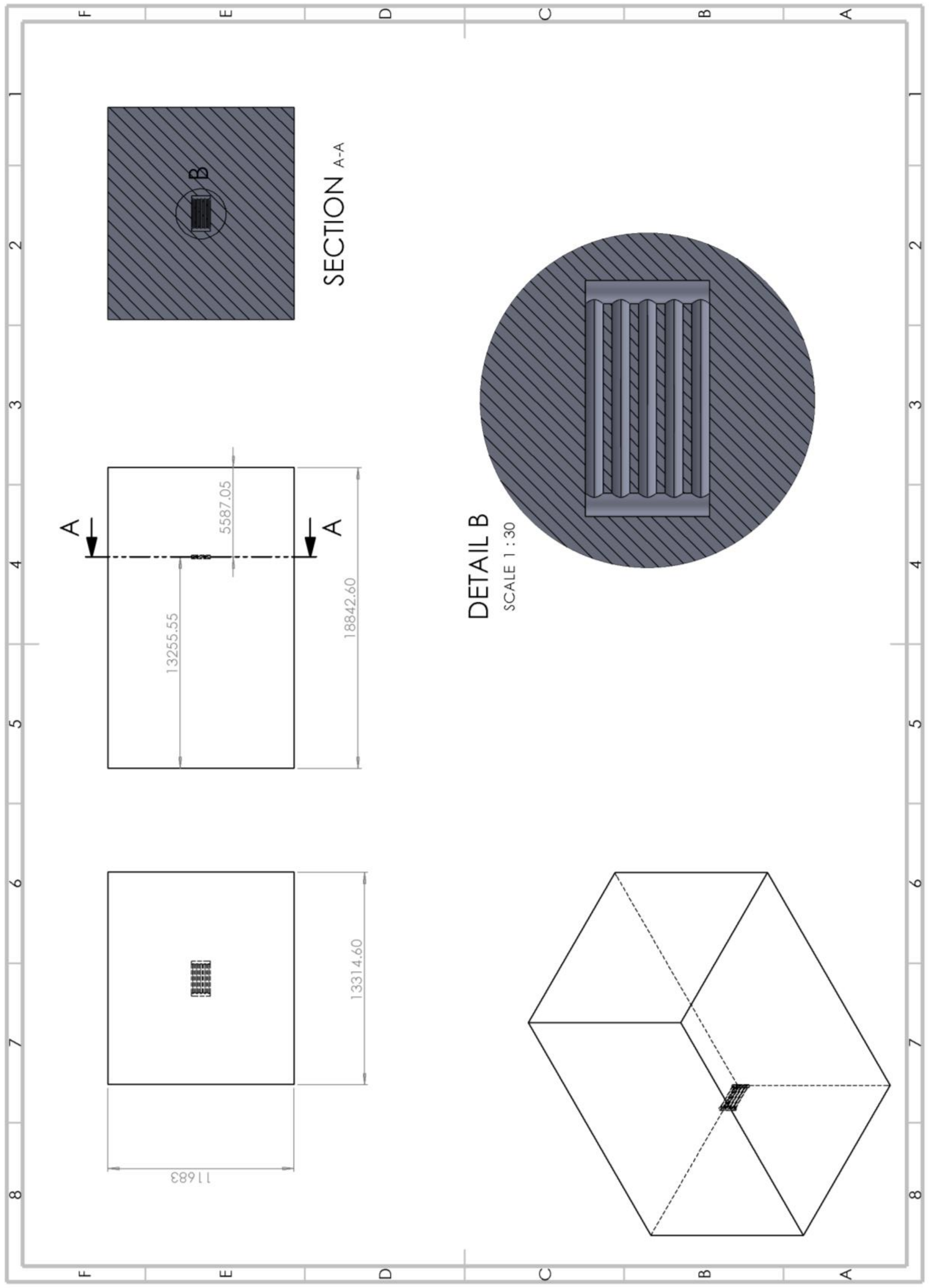
Figure 11: system directory: fvSchemes



```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  7                                     |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{

    p
    {
        solver          GAMG;
        tolerance       1e-06;
        relTol          0;
        smoother         GaussSeidel;
        minIter         2;
    }

    pFinal
    {
        $p;
        tolerance       1e-06;
        relTol          0;
        minIter         2;
    }

    "(U|k|epsilon|omega)"
    {

        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-08;
        relTol          0.0;
        minIter         2;
    }


}

PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 3;
    pRefCell        0;
    pRefValue       0;
}


// ************************************************************************* //
```

Figure 12: system directory: fvSolution

Appendix K– technical drawing of the size independent fluid domain


SECTION A-A
B
A
A
5587.05
13255.55
18842.60
DETAIL B
SCALE 1 : 30
13314.60
11683