SINTEF Energi AS
SINTEF Energy Research
Address:
Postboks 4761 Torgarden
NO-7465 Trondheim
NORWAY

Switchboard: +47 45456000

energy.research@sintef.no

Deliverable for project

Compact Offshore Steam Bottoming Cycles Phase 2: COMPACTS2

Number	D2.2_2020_2
Title	Numerical simulation of fluid-elastic instability in normal triangular tube bundles

Approved by

Name	Role	Signature
Marius Andersen	Quality assurance	*Marius Andersen* Marius Andersen (25. Nov. 2020 14:11 GMT+1)
Kristin Jordal	Research manager	*Kristin Jordal* Kristin Jordal (25. Nov. 2020 16:06 GMT+1)

SINTEF

2020:00788 - Unrestricted

Report

Numerical simulation of fluid-elastic instability in normal triangular tube bundles

Author(s)
Ole Meyer

SINTEF Energy Research
Gas Technology
August 21, 2020

SINTEF Energi AS
SINTEF Energy Research

Address:
Postboks 4761 Torgarden
7465 Trondheim
NORWAY

www.sintef.no

Enterprise Number: NO 939 350 675 MVA

KEYWORDS:
Fluid-elastic instability
CFD
Fluid-structure interaction
Heat-exchanger vibration

Report

Numerical simulation of fluid-elastic instability in normal triangular tube bundles

VERSION
1.0

DATE
August 21, 2020

AUTHOR(S)
Ole Meyer

CLIENT(S)
Andreas Q. Nielsen

CLIENT'S REFERENCE
280713

PROJECT
COMPACTS2 (502001904)

NUMBER OF PAGES AND ATTACHMENTS
32 0

ABSTRACT
This document gives an overview of the numerical modelling activity conducted within work package 2 of the COMPACTS2 project. Various attempts to construct a flexible modelling tool to predict the fluid-elastic instability threshold in normal triangular tube bundles are presented. Numerical predictions are compared against experimental measurements. Qualitative validation on force and amplitude predictions is documented. Guidelines and recommendations for further research are offered.

PREPARED BY
Ole Meyer

SIGNATURE

REPORT NUMBER
2020:00788

ISBN
978-82-14-06581-7

CLASSIFICATION
Unrestricted

CLASSIFICATION THIS PAGE
Unrestricted

Document History

VERSION	DATE	VERSION DESCRIPTION
0.1	2020-06-04	First draft version sent for internal QA
1.0	2020-08-20	Final version for approval

SINTEF

Contents

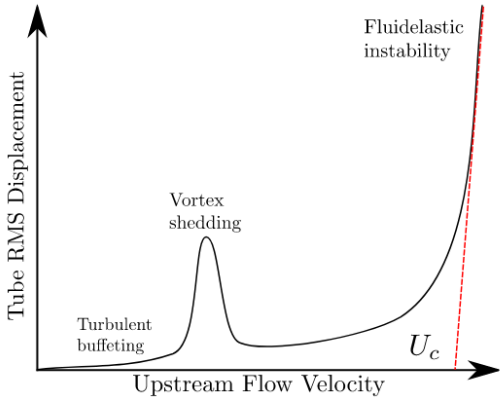

Figure 1: Vibrational response, as sketched in [3].

1 Introduction

Assessment of fluid-elastic instability threshold is of vital importance in heat-exchanger design, as induced vibration run-away can lead to short term failure [1]. Critical flow velocities are among other effects influenced by array setup, pitch ratio, forcing fluid, tube mass, damping and material stiffness. Of specific research focus in the COMPACTS2 project, is the heat exchanger performance in conjunction with the waste-heat-recovery unit. Powerful optimisation routines specifiy constraints on e.g. tube spacing, diameter, pitch ratio and flow rate, but presently rely on correlations for limiting operation velocities. It is therefore of interest to predict the limiting *critical* velocity for the onset of fluid-elastic instability by means of computation methods. To this end, a flexible modeling tool is required to provide confidence in and possibly extend presently used correlations for limiting operating velocities. The modelling tool should also be generic enough to allow for later inclusion of surface-altered tubes, such as finned-tubes.

This work builds on the findings of Lindqvist and Næss [2], who have provided correlations for pressure drop and heat transfer rate for a range of pitch ratios and array arrangements by means of computational fluid dynamics (CFD) simulations. The work in [2] is concerned with static tube bundles, i.e. the coupling of fluid forces and structure motion is neglected. Fig. 1 shows a typical response curve for tube displacement due to fluid-induced forces: The incoming cross-stream induces lift forces proportional to the shedding frequency of vortical structures, f_s. For modest flow velocities, the structure response is largely controlled by turbulent interactions (”Turbulent buffeting” regime). As the upstream velocity is increased, the frequency of vortex shedding behind the tubes approaches the natural frequency, f_n, of the structure. The fluid-induced lift force, $F_L \sim \sin 2\pi f_s$, then gives rise to increased vibration amplitudes perpendicular to the flow direction (”Vortex shedding” regime). Beyond the lock-in ($f_s \sim f_n$) the amplitudes are reduced and increase until the critical velocity, U_c is reached. At this velocity, the tube vibration increase without bounds and can potentially lead to critical failure of the structure (”Fluidelastic instability” regime). The details of the onset of fluid-elastic instability are an active field of research with the complex fluid-structure interplay involving turbulence, boundary layer separation, flow periodicity, and structure stiffness and damping. In depth review studies, numerical and experimental work can be found e.g. in [1–10].

The employed method to assess the fluid-elastic instability in normal triangular tube bundles is outlined in Sec. 2. Evaluation of the method against experiments is presented in Sec. 3, with conclusions and recommendations for further work offered in Sec. 4. Relevant code listings are attached in the appendix.

2 Method

The goal of this activity is to put forward a flexible tool, which can be applied to different array geometries and pitch ratios. Further, quantification of vibration amplitudes is required to be computationally practical, as

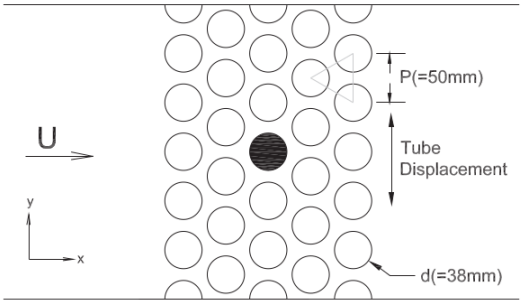

Figure 2: Generic array setup for flow over normal triangular tube bundle, as sketched in [9].

large parameter studies are envisioned based on the successfull construction of the method. As prediction of the critical velocity will involve solution of some form of the (incompressible) Navier-Stokes equations, this endeavour is clearly impossible. Trade-offs between numerical accuracy, physical validity and computational effort as well as comprehensibility of the method must consequently be tolerated.

Fluid motion is described by the incompressible Navier-Stokes equations,

$$\nabla \cdot \boldsymbol{u} = 0, \tag{1}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) = -\nabla p + \nabla \cdot \left[\nu \left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^{\mathrm{tr}}\right)\right] + \boldsymbol{f}. \tag{2}$$

Here, the velocity field $\boldsymbol{u}$ is coupled to the incompressible pressure, $p \rightarrow p/\rho$, with the fluid density ρ, and a generic driving force density $\boldsymbol{f}$. The kinematic viscosity, $\nu = \mu/\rho$, with the dynamic viscosity μ, is assumed constant. An overview of the flow domain is presented in Fig. 2, introducing the free-stream velocity U_0, tube-pitch, P, and cylinder diameter, D. For incompressible flow, the mass-averaged gap-velocity is given by

$$U_g = U_0 \frac{P}{P - D}, \tag{3}$$

Reynolds-numbers based on the gap velocity are expected to be on the order of $\mathcal{O}(10^4)$, consequently the flow in the tube array will be turbulent. Lindqvist and Næss [2] have successfully employed the unsteady-Reynolds-averaged-Navier-Stokes (URANS) formulation to take into account turbulent fluctuations. Here, the $k-\omega$-SST model ([11]) will be employed. This turbulence model has proven practical and accurate in describing separated boundary layer phenomena and is further described in [12].

2.1 OpenFoam framework

We use the open-source framework OpenFOAM to solve the governing equations. OpenFOAM provides a generic framework for finite-volume discretization of partial differential equations. It is written as a set of C++ libraries, and its object-oriented structure allows for close top-level representation of the mathematical formulations. This enables intuitive custom development and modification [13]. The flexibility of OpenFOAM for tailor-made applications has received increasing attention recently [14].

A typical workflow consists of specifying initial and boundary conditions for the field variables at hand in separate files, as well as mesh files that contain the discretization domain and configuration files to specify the solver with numerical schemes and solution/convergence criteria. This work employs the PIMPLE algorithm for pressure-velocity coupling. The PIMPLE algorithm is a hybrid SIMPLE–PISO iteration scheme that allows larger time steps. Adaptive time steps limited by a user defined Courant number[1] may be chosen. Summarized, the SIMPLE algorithm [15] contains the following steps:

[1]The Courant number gives a necessary stability condition that relates the time step and spatial discretization. It can be understood as a constraint on the minimum allowable propagation velocity of numerical waves. We also refer to the Courant number as the CFL number.

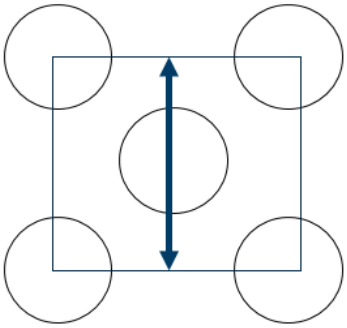

Figure 3: Periodic subdomain in relation to array geometry.

1. solve for the velocity vector from the momentum equation with an initial guess of the pressure
2. add corrections to the velocity and pressure
3. solve for the pressure corrections
4. solve for the velocity corrections
5. repeat until the convergence criterion is reached

The PISO algorithm [16] adds a second corrector stage to obtain better convergence. Time integration is performed by an implicit, first-order Euler scheme or second-order Crank Nicolson scheme. Interpolation of the face fluxes to the cell values is achieved by combinations of second- and first-order central-differencing schemes. At each time step, the convergence of velocity, pressure, and enthalpy is monitored. The algorithm is considered converged upon reaching user defined values of residuals.

The incompressible, transient *pimpleFoam* solver is employed to solve the system of URANS equations. If applicable, the iterative *simpleFoam* solver is used to compute appropriate initial conditions.

2.2 Turbulence model

In this work, the "k-ω SST" model has been employed. Within this framework, equations for turbulent kinetic energy

$$k = \frac{3}{2}\boldsymbol{u}\boldsymbol{I} \cdot \boldsymbol{u}\boldsymbol{I}, \tag{4}$$

and turbulence specific dissipation rate

$$\omega = \frac{\nu_t}{k}, \tag{5}$$

with the turbulence intensity, I and turbulence viscosity, ν_t, are coupled to the incompressible Reynolds-averaged Navier-Stokes equations. The model is documented in [17], and references therein.

2.3 Meshing strategy

In this contribution, three different meshing strategies are pursued. Their general characteristics are outlined here, with different validation approaches provided in Sec. 3. It should be noted that all approaches are 2-dimensional, in the sense that the flow is computed in the downstream- and cross-stream directions. The spanwise direction is not solved for. Presently, only bare tubes are considered.

2.3.1 Periodic mesh

The *periodic* mesh, is a direct application of the meshing in [2], adapted into a dynamic mesh motion framework. Fig. 3 illustrates the idea: The underlying array periodicity allows construction of a periodic subdomain of the

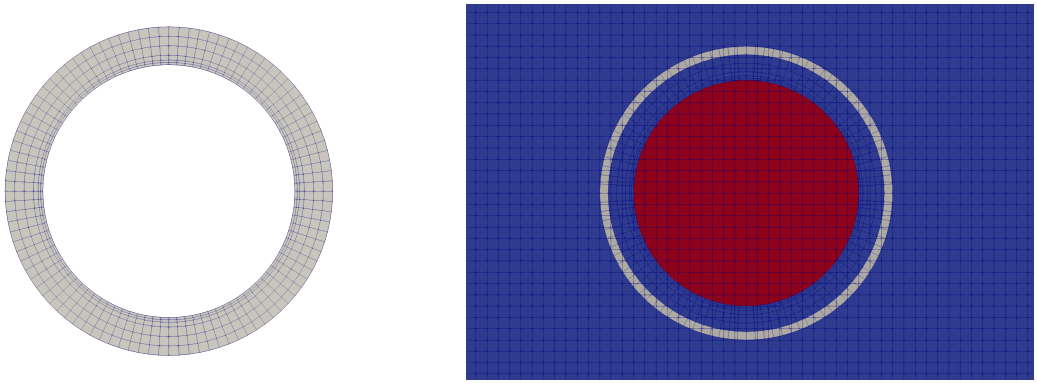

Figure 4: Cylinder mesh (left) merged onto a structured background mesh (right).

entire array. The correspondence to a single oscillating tube is achieved by fixing the static corner quarter-cylinders and allowing the central tube to move in accordance to the fluid forcing. This meshing approach has resulted in good agreement for pressure drop and heat transfer rates in static tube bundles [2]. Mesh motion is achieved by conventional cell deformation.

2.3.2 Overset mesh

An interesting, state-of-the art dynamic mesh approach is the *overset mesh* (”Chimera mesh motion”) setup. In this framework, the computational domain consists of a number of separate meshes, with dynamic meshing appearing through interpolation of the fluid field onto the separate overset meshes. Fig. 4 illustrates the resulting setup of a single cylinder, body fitted mesh (left), overset onto the structured background mesh (right). Fig. 4 (right) illustrates the overset method: The solver distinguishes between calculated (blue), interpolated (grey) and blocked (red) cells. The flow field is solved for on the calculated cells in the background mesh and interpolated onto the overset mesh. Inside the overset mesh, blocked cells are identified, with the flow field not being calculated there. Further information is offered in [18]. Fluid induced motion of the entire overset mesh in each time step is then interpolated back onto the background mesh. This approach has a number of advantages for FSI situations, as meshing resources can be effectively focused where necessary. A good mesh features smooth transitions across domains of different degree of resolvement, which can be challenging when multiple bodies are present in close proximity to each other. The overset grid approach conveniently bypasses this challenge by allowing multiple body fitted meshes (at the respective desired resolution) to be merged onto the background mesh. The smoothness constraint translates to a cell size-ratio constraint at the overset patch (located at the cell interpolation regions across meshes), with ratios of 1-1.5 recommended. A major advantage of the overset grid approach is the self-consistent handling of large mesh displacements. Conventional dynamic mesh motion is complicated through necessary motion induced cell deformation. In the overset grid setup, the overset mesh is simply displaced according the fluid motion, with the interpolation region following. There is no need to deform any cells. However, recommended practice is that at least 4 cells should be present between the moving overset patch boundary and confining walls, i.e. the motion should be far enough away from any boundaries.

2.3.3 Wind-tunnel mesh

Finally, a mesh that directly replicates the entire wind-tunnel experiment is referred to as *wind-tunnel mesh*. This setup features 27 cylinders placed in 5 rows, as well as six half-cylinders, as sketched in Fig. 2. A related mesh of the same height and length as well as row numbers, but fewer cylinders per column has also been tested. This mesh corresponds to the experimental setup and numerical simulations performed in [4]. In both cases, mesh motion is achieved by conventional cell deformation.

Boundary layer resolution

In the following, two approaches for the near-wall region resolution are applied. The most physically accurate framework, seeks to fully resolve the boundary layer, i.e. $y+ < 1$, where

$$y+ = \frac{yu_\tau}{\nu}, \tag{6}$$

is the normalised wall distance exressed through the normal wall distance y and the friction velocity u_τ. The friction velocity is related to the wall shear stress τ_w via

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}}, \tag{7}$$

with

$$\tau_w = \rho\nu \left(\frac{\mathrm{d}u}{\mathrm{d}y}\right)_{y=0}, \tag{8}$$

where u is the velocity component parallel to the wall. The friction velocity sets the scale for the non-dimensional near-wall velocity

$$u+ = \frac{u}{u_\tau}. \tag{9}$$

Mesh generation for fully resolved boundary layers must respect a smooth transition in cell size from the high resolution near-wall region to the bulk mesh size. In addition, the exact $y+$ is not known a priori, hence an iterative approach is necessary to assure that $y+ < 1$ for a given velocity. Further, in velocity-scan type studies like this one, the mesh setup should respect the cell size constraint for all velocities. The fully resolved boundary layer approach is costly, as typically 10-15 layers of small cell size needs to be fitted to the body. The maximum time step is constrained by the smallest cell size, consequently fully resolved boundary layer simulations are costly, especially for high Reynolds number flows.

Another meshing approach, typically applied in industrial settings and for high Reynolds number flows, is the wall function setup. In this case, the near wall region profile for velocity and shear stresses is modeled. The wall function approach is valid for $y+ > 11$, which is particulary challenging for the low end of the velocity scan, as cell sizes can become impractically large. The main argument against using wall functions, is that the details of the boundary layer are prescribed by the functional form of the wall functions. Typically, wall functions should not be employed in situations where flow separation is expected. Nonetheless, wall functions are commonly employed also in these settings.

2.4 Wall functions

Wall functions impose relations for turbulence quantities in the near wall region and set boundary conditions for turbulence quantities at the wall. Specifically, the turbulence viscosity is related to the nondimensionalised wall distance

$$\nu_t = \nu \left[\frac{\kappa y+}{\log(Ey+)} - 1\right], \tag{10}$$

with the von Karman constant $\kappa = 0.41$, the model coefficient $E = 9.8$, and the (laminar) kinematic viscosity ν. The relation among $y+$ and the nondimensional velocity $u+$ (Spalding's law) couples the near-wall turbulence viscosity profile to the near-wall velocity profile,

$$y+ = (u+) + \frac{1}{E}\left[\exp(\kappa u+) - 1 - \kappa u + -\frac{1}{2}(\kappa u+)^2 - \frac{1}{6}(\kappa u+)^3\right]. \tag{11}$$

The turbulence specific dissipation rate is prescribed the profile

$$\omega = \sqrt{\omega_{\mathrm{vis}}^2 + \omega_{\mathrm{log}}^2}, \tag{12}$$

with

$$\omega_{\text{vis}} = \frac{6\nu}{0.075 y^2}, \tag{13}$$

and

$$\omega_{\log} = \frac{k^{1/2}}{C_\mu^{1/4} \kappa y}, \quad C_\mu = 0.09. \tag{14}$$

The turbulent kinetic energy is assigned a zero-gradient boundary condition at the wall. In OpenFoam, these wall functions are called nutUSpaldingWallFunction, omegaWallFunction, and kqRWallFunction, respectively.

2.5 Boundary conditions

Boundary conditions for the velocity, pressure, turbulent kinetic energy, turbulent dissipation rate and turbulent viscosity need to be provided within the $k - \omega$ SST turbulence model. We distinguish between inlet driven and source-term driven flow, with the source-term in Eq. (2) zero in the former and nonzero in the latter case.

Periodic mesh

At the static cylinder walls, a no-slip velocity boundary condition is applied. The flexible central cylinder needs to be specified a moving-wall velocity, which allows translation of the boundary points as part of the background mesh. The pressure boundary conditions on all walls are of type zero-gradient. Fully resolved boundary layer simulations are setup with approximate zero-value boundary conditions at cylinder walls for turbulent kinetic energy and turbulent viscosity, and $\omega = 6\nu/0.075(\Delta y)^2$, with Δy the cell-center distance from the first cell off the wall. The left, right, up and down patches feature periodic boundary conditions. Spanwise sides are specified as empty, as the flow is not solved for in that direction. Specifying the forcing term in Eq. (2) gives rise to the flow.

Overset mesh

The overset mesh features conventional incompressible inflow-outflow boundary conditions at the left and right boundaries: A fixed velocity is specified with zero pressure gradient at the inflow side (left) and a zero-gradient velocity condition together with specified zero pressure is applied at the outflow side (right). Turbulence fields are specified a fixed value and zero gradient at the inlet and outlet patch, respectively. At the up and down side, as well as the cylinder walls, wall functions are applied. Velocity and pressure are assigned no-slip and zero-gradient conditions on lateral walls, respectively. At the moving cylinder walls, the moving-wall condition is used for velocity, with wall functions for turbulence fields and zero-gradient for pressure. Spanwise sides are specified as empty, as the flow is not solved for in that direction. At the outer boundary of the overset meshes, the so-called oversetPatch boundary condition is applied. This boundary condition forces the solver to interpolate the flow field across the meshes as illustrated in Fig. 4.

Wind-tunnel mesh

The boundary conditions for the wind-tunnel mesh are similar to the overset mesh, with the exception that oversetPatches do not need to be specified.

2.6 Fluid-structure coupling

Fluid-structure interactions are computed by the *sixDoFRigidBodyMotionSolver* under the *linearSpring* restraint. Fig. 5 shows the two-dimensional restraints applied on a flexible cylinder. The fluid-structure coupling

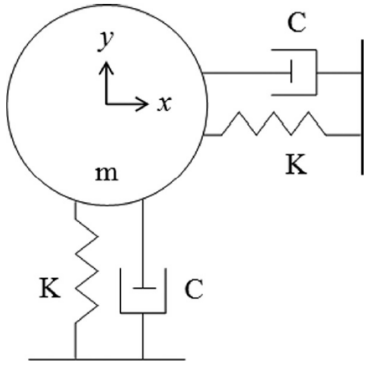

Figure 5: Illustration of a mass-spring-damper system, as sketched in [19].

is consequently modeled as a spring-damper system:

$$m\frac{\mathrm{d}^2x}{\mathrm{d}t^2} + C\frac{\mathrm{d}x}{\mathrm{d}t} + Kx = F_D, \tag{15}$$

$$m\frac{\mathrm{d}^2y}{\mathrm{d}t^2} + C\frac{\mathrm{d}y}{\mathrm{d}t} + Ky = F_L, \tag{16}$$

with the flow aligned with the x-direction. Here the fluid dynamics provides the forcing in terms of drag $F_D = 0.5\rho U^2 DC_D$ and lift $F_L = 0.5\rho U^2 DC_L$. It should be noted that the drag- and lift forces are evaluated at run-time by intgrating the pressure gradients around the surface, producing the drag- (C_D) and lift- (C_L) coefficients respectively. The spring-damper system is setup to resemble the actual response of a free tube in experiments, i.e. single-degree of freedom motion perpendicular to the flow is achieved by eliminating the linear spring in the x-direction and applying a line constraint in the y-direction (no x-motion allowed). The damping coefficient C is related to the intrinsic (material) damping-ratio ζ of the tubes via

$$C = 4\pi m\zeta f_n, \tag{17}$$

with the damping connected to the tubes' natural vibration frequency f_n and the logarithmic decrement δ:

$$\zeta = \frac{1}{\sqrt{1 + \left(\frac{2\pi}{\delta}\right)^2}}. \tag{18}$$

A representative spring-stiffness is provided by means of

$$K = 4\pi^2 m\zeta f_n^2. \tag{19}$$

The structure response to the fluid forces computed in each time step relies on an algorithm to evaluate the structure displacement. For details on the coupling algorithm, cf. [20]. The updated position vector of the structure is fed into the dynamic mesh solver, which updates the mesh accordingly.

3 Validation procedure

A validated method should reproduce experimental findings to a reasonable degree. The experiments in [8] are chosen as a reference. Their setup for fluid induced motion of the central tube is identical to the pressure distribution studies on static arrays, performed in [9], giving another reference data set. Numerous numerical studies on fluid induced tube motion further compare results against the single cylinder motion experiments in [21].

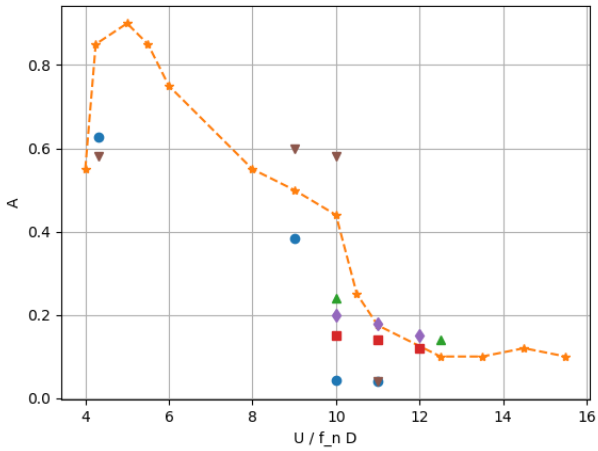

Figure 6: Vibration amplitudes computed on a simple boundary layer resolved mesh (blue dots). Experimental values from [21] are marked by orange stars. Remaining markers correspond to numerical studies performed by [22] (downward triangles), [23] (upward triangles), [19] (diamonds), and [24] (squares).

3.1 A single dynamic cylinder in water

A good starting point to validate the cell deforming dynamic mesh framework implemented in OpenFoam, is provided by the experiment performed in [21]. In the experiment, vibration amplitudes for a rigid cylinder, subject to cross-flow of water are measured. Here, the corresponding spring-constant and damping parameter are computed from the respective provided values of natural frequency, mass ratio and damping ratio:

$$\begin{aligned} f_n &= 0.543\,\text{Hz}, \\ m^* &= 2.4, \\ m^*\zeta &= 0.013, \end{aligned}$$

The mass ratio is defined as the ratio of system mass m_{sys} to displaced fluid mass m_{d}, with the displaced fluid mass related to the tube diameter and length, L: $m_{\text{d}} = \pi\rho D^2 L/4$. Further, the damping ratio is given as a measure of the system damping coefficient, c_{sys} to critical damping, $\zeta = c_{\text{sys}}/2\sqrt{k m_{\text{sys}}}$. The set of fluid-structure parameters employed is

$$\begin{aligned} D &= 40 \times 10^{-3}\,\text{m}, \\ L &= 1\,\text{m}, \\ m_{\text{sys}} &= 3\,\text{kg}, \\ K &= 35\,\text{N/m}, \\ C &= 0.1\,\text{N s/m}, \end{aligned}$$

with the normalised fluid velocity $U^* = U_0/f_n D$ in the range of 4-16. Initial values for turbulent kinetic energy and turbulence dissipation rate are similar to the numerical study in [22]. A simple boundary-resolved body-fitted mesh is obtained from the snappyHexMesh toolbox. As the purpose is to gain quick insight, and qualitatively compare computed vibration amplitudes to experimental observations, appropriate wake refinement and grid convergence studies have not been performed at this stage. The domain is loaded with a fixed free-stream velocity, corresponding to the Reynolds number in the experiment, with the tube free to respond to variations in the lift force. Fig. 6 (orange stars) shows resulting amplitudes in statistically stationary steady-state, i.e. measured when regular tube oscillation has set in. It can be observed, that the overall trend is captured even with the relatively simple mesh. Specifically, the results from [22], who have performed a dedicated comparison against [21], for $U/f\,D = 4.33$ compare favorably. At higher relative velocity, the vibration amplitude

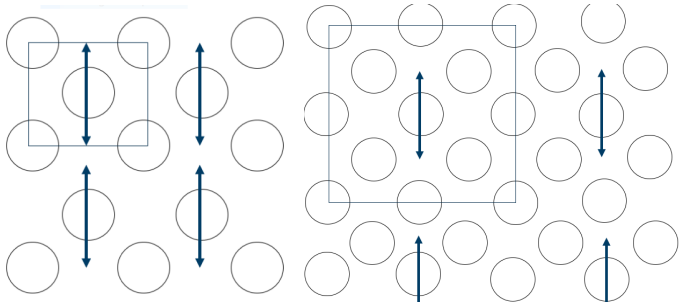

Figure 7: Illustration of the physical implication of invoking periodic boundary conditions for a single flexible tube in the center of the computational domain. Left: Small mesh unit. Right: Large mesh unit.

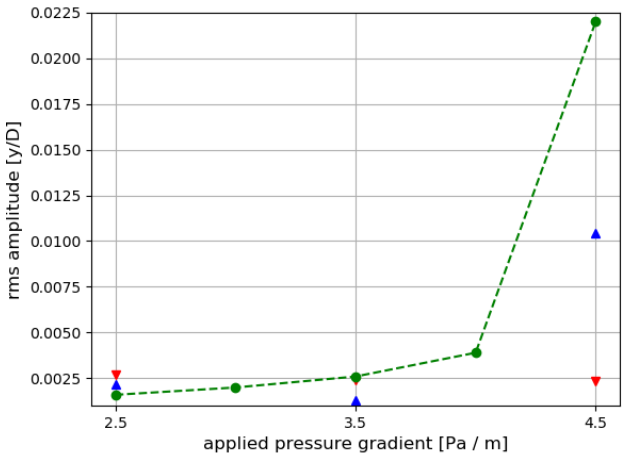

Figure 8: Vibration amplitudes as function of applied pressure gradient for periodic small- (red) and larger mesh (blue), and compared to experimental measurements [8] (green).

is underpredicted, however, the trend of a "flat tail", seems to be present also in our simulations. A decent agreement with experimental ([21]) and numerical results [19, 22–24] can be observed, even for a simple mesh. The results give confidence, that vortex-induced structure motion can be modelled within the existing dynamic mesh coupling implemented in OpenFoam version 1812.

3.2 Periodic mesh

The first mesh to be tested on the experimental results in [8] is the periodic mesh, as the concept for static bundle flow had been established in [2]. Fully resolved boundary layer meshes of two sizes have been considered: The smallest consists of a single central cylinder, with four quarter cylinders at the edges (cf. rectangular domain shown in Fig. 7 (left)), and a larger one with 5 cylinders, 4 half-cylinders at the domain sides, and 4 quarter cylinders at the edges, cf. Fig. 7 (right).

A cumbersome detail with the periodic mesh is that direct comparison with experiment is an iterative process: As mentioned in Sec. 2.5, the flow is driven by the applied force term in the momentum equation. The flow response to the applied "artificial pressure gradient" is *a priori* not clear. Only after the flow has established a response to a specified force term, can the corresponding free-stream velocity of the experiment be computed. Fig. 8 shows the resulting vibration amplitudes for the two periodic meshes compared to experimental values from [8] as a function of applied pressure gradient source-term.

The source-term range compares approximately to free-stream values between 3.1 m/s and 4.1 m/s. Fig. 8 illustrates an interesting point concerning this meshing approach. The amplitudes are fairly constant as a function

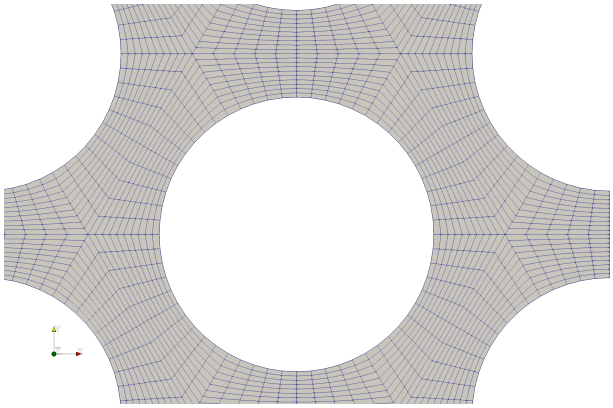

Figure 9: Honey-comb type mesh. Observe the occurence of skewed cells at the nodes connecting the honey combs around the cylinders.

of driving force (and hence free-stream velocity) for the small mesh (red). Some qualitative agreement with the experiment for the larger mesh (blue) is present. Clearly, the vibration amplitude is strongly dependent on the chosen size of the periodic simulation domain. This has led to the realisation, that the periodic boundary condition (while offering sensible results for the static case) are unfit for the present dynamic mesh study. In any case, their physical consistency with the experiment is not correct as the net flow direction associated with the specified applied pressure *gradient* necessarily violates the assumption of periodicity in the flow direction. Further, for motion of the central tube, the actual periodic extension in cross-flow direction of the small mesh would resemble Fig. 7 (left): What is actually simulated is not the vibration of a single cylinder, but the motion of the entire column that cylinder belongs to. This is clearly not the case in the experiment [8]. As the mesh size is increased to the large rectangular domain in Fig. 7 (right), with the central cylinder allowed to move, there are additional static cylinders that could break the periodic forcing of the flow, yielding more realistic results.

The periodic approach was deemed impractical for the present purpose due the complexity of flow initialisation, grid size constraint and lacking physical correspondence for boundary conditions.

3.3 Overset mesh

Preliminary studies have shown promising results for single cylinder motion. A full array setup, however, has revealed that the minimum spacing of 4 cells cannot consistently be maintaned for all relevant flow conditions. In addition, the OpenMPI parallelisation in the latest version of OpenFoam-v1912 (also OpenFoam-v1812) resulted in problems: Single core simulations did not provide identical results as multi-core simulations. This technique was not further investigated for that reason. Future updates in the dynamic motion solvers of OpenFoam are likely to resolve this issue. The overset mesh approach should be considered in the future.

3.4 Wind-tunnel mesh

A conventional meshing approach has resulted in the most consistent and promising results thus far. The entire tube array is modelled with confining walls and inflow outflow patches. Based on the large computational resources necessary to fully resolve the boundary layer, a coarse-mesh with-wall-function approach has been followed. Promising results for prediction of the critical velocity have been achieved by [25] by use of a coarse "honey-comb" type mesh. Hexagonal blocks are constructed around each cylinder, resulting in an overall resemblence to a honey-comb. The starting point for the present approach was to employ the blockMesh routine of OpenFoam to design a script that computes the cell distribution for a 5 row normal triangular tube array as studied in [4]. This array features at most 3 cylinders per row, reducing the computational cost. A detail of the honey-comb structure is shown in Fig. 9. It is not well documented how [25] resolved the boundary layer and what boundary condition they applied at the cylinder walls. To validate the meshing approach, data of [9] for

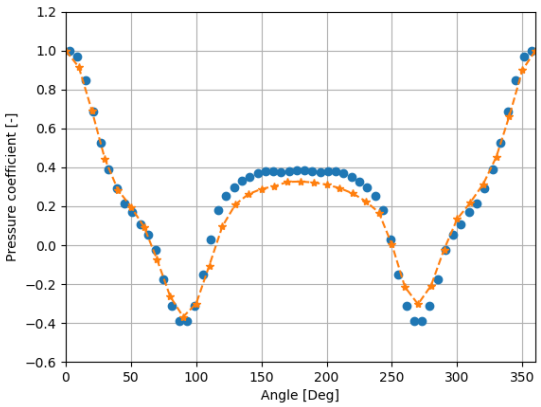

Figure 10: Angular pressure coefficient distribution for $U_0 = 7\,\text{m/s}$. Numerical simulations (dots) compared with experimental values from [8] (dotted line).

the pressure coefficient distribution in static normal triangular arrays have been used for comparison. A normal triangular tube bundle of pitch ratio $P_r = 1.32$, with $D = 38\,\text{mm}$ has been studied in a wind tunnel with 5 rows, 11 columns and 2 half-cylinder columns for the flow of air. Measurements in [9] are obtained around the circumference of the central cylinder of row number 3. Cases considered are for free-stream velocities in the range $3 - 12\,\text{m/s}$. The pressure coefficient in [9] reads

$$C_p = 1 - \frac{p_{\theta\text{max}} - p_\theta}{\frac{1}{2}\rho U_g}, \tag{20}$$

with the azimuthal angle increasing clock-wise and of zero value at the upstream stagnation point of the cylinder. Fig. 10 shows the azimuthal variation of the pressure coefficient on the central tube for an upstream velocity of $U_0 = 7\,\text{m/s}$. A remarkable agreement with experimental values is observed until $\theta \approx 110^\circ$ and beyond $\theta \approx 260^\circ$. These findings should be compared to [4], who have tested different turbulence models on a fully resolved boundary layer mesh, with deviations of the same order in the recirculation zone ($\theta \in [110^\circ, 260^\circ]$). This gives confidence that pressure fluctuations in the boundary layer can to some extent be modeled with the wall function approach, however, also small deviations in the pressure coefficient will yield deviations in the resulting lift force

$$F_L \sim \int C_p \sin\theta \mathrm{d}\theta. \tag{21}$$

Setting the single tube free to respond to the flow has resulted in inconclusive amplitudes (not shown here). It was concluded that the employed bulk resolution, cell size transition, presence of acute angle cells at hexagonal nodes (cf. Fig. 9) and possibly overall mesh size were contributing factors to the inconsistency in tube amplitudes. Various refinement approaches of the near-wall region, as presented in [25], could be pursued in the future, but the resulting procedure would at present be too cumbersome to be classified as a flexible tool. We therefore decided on a different meshing strategy.

In light of the failure of the honey-comb mesh to produce reasonable vibration amplitudes, the whole wind tunnel was meshed with the snappyHexMesh routine. The snappyHexMesh routine allows to easily modify the diameter or the positions of the tubes to change the pitch ratio or the tubes' relative positions. Inclusion of 3-dimensional features such as tube fins can be added consistently.

The resulting overall mesh is shown in Fig. 11 (left), with details of the boundary layer resolution in Fig. 11 (right). A smooth cell size transition from the boundary region to the bulk region, cf. Fig. 11 (right), can be observed. Further, the issue of acute angle cells is circumvented in this mesh. Necessary meshing files are attached in Listing 4 and Listing 3.

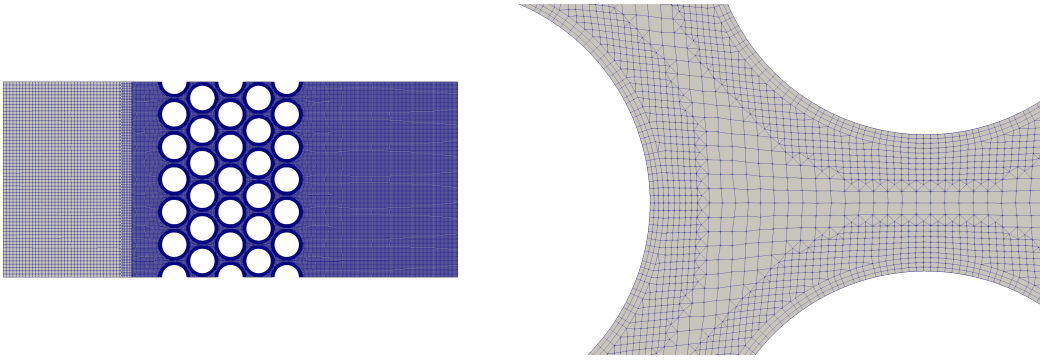

Figure 11: Close-up view of the wind-tunnel mesh. Entire mesh (left) and close-up view of the near cylinder region (right). The fluid enters the domain on the left boundary.

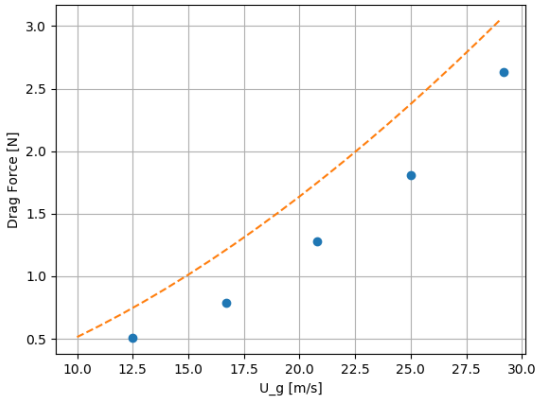

Figure 12: Drag force on the center tube for numerical simulations (blue dots) and experimental measurements by [8] (orange dotted line).

As in Sec. 3.2, we set out to model the $\delta = 0.093$ experiment of [8]. Corresponding fluid and structure parameters are

$$\begin{aligned} f_n &= 6.6\,\text{Hz}, \\ \zeta &= 0.014, \\ \frac{m_{\text{sys}}\delta}{\rho L D^2} &= 175.3, \\ D &= 38 \times 10^{-3}\,\text{m}, \\ L &= 3 \times 10^{-1}\,\text{m}, \\ m_{\text{sys}} &= 1.05\,\text{kg}, \\ K &= 1805.66\,\text{N/m}, \\ C &= 1.289\,\text{N s/m}, \end{aligned}$$

An initial solution is obtained by running the steady-state simpleFoam solver on a static array. The simulations are then continued by the transient pimpleFoam solver, with the central tube free to respond to the fluid loading.

In Fig. 12, the computed values for the drag force on the central tube are shown as a function of gap velocity and compared to a scaling derived by [8]. The force values from the experiment are reported for the central tube

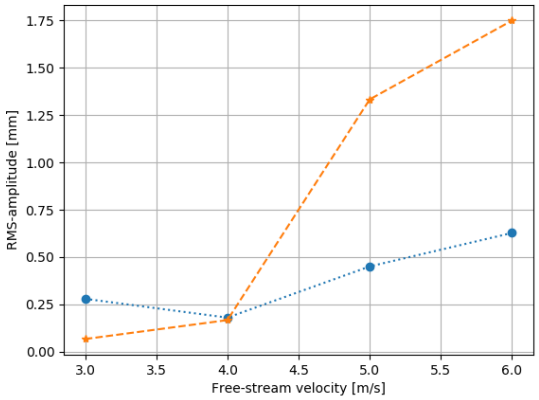

Figure 13: Vibrational response of the flexible center tube to imposed air cross-flow. Orange stars are experimental values due [8] and blue dots are numerical results.

prior to release, with the numerical values corresponding to the results after the simpleFoam computation has ended. A decent agreement in the scaling of drag-force with flow velocity is observed.

Fig. 13 shows the dynamic response of the center tube to the imposed air cross-flow. A trend of increasing vibration amplitude with flow velocity can be seen for $U > 4$ m/s, with a steep increase between $U = 4$ m/s and $U = 5$ m/s. The experimental values are up to three times higher than the numerical results, however, the fluid-elastic instability threshold identified by the numerical method agrees favorably to the reported experimental value of $U = 4.1$ m/s. Clearly, a finer spacing in velocity increase should be considered to estimate a more precise value of the critical velocity. It should be noted that the excess vibration amplitude for low flow velocity ($U < 4$ m/s) is likely associated with the breakdown of the wall function validity ($y+ < 11$). The vibrational amplitudes derived from the wind-tunnel mesh are reasonable, taking into account the drastic modelling descision to prescripe boundary layer properties via wall functions.

4 Conclusions

This activity has considered various approaches to model the onset of fluid-elastic instability in normal triangular tube arrays. Fluid-structure coupling within the OpenFoam open-source CFD toolbox has been applied to experiments for tube oscillation in water and air cross-flow. A flexible modelling tool was developed by means of the snappyHexMesh routine in OpenFoam. Meshing the entire wind-tunnel setup in alignment with the experiments has proved to yield the most accurate structure response to the impose air flow. Experimental results have been qualitatively reproduced in a high Reynolds number boundary layer formulation, where wall functions are applied at confining walls. The necessity for additional research to lift the performance of the tool in terms of quantitative predictions has been elucidated.

5 References

[1] D. S. Weaver, S. Ziada, M. K. Au-Yang, S. S. Chen, M. P. Paidoussis, and M. J. Pettigrew. “Flow-induced vibrations in power and process plant components - progress and prospect.” In: *Journal of Pressure Vessel Technology* 122 (2000), pp. 339–348.

[2] K. Lindqvist and E. Næss. “Numerical Modeling of Vortex Shedding in Helically Wound Finned Tube Bundles in Cross Flow”. In: *Proceedings of the 16th International Heat Transfer Conference, Beijing*. 2018, pp. xx–xx.

[3] S. E. Bouzidi. “Numerical Modelling of Fluidelastic Instability in a Normal Triangle Tube Array”. MA thesis. Guelph, Ontario, Canada: University of Guelph, 2014.

[4] A. Khalifa. “Fluidelastic Instability in Heat Exchanger Tube Arrays”. PhD thesis. Hamilton, Ontario, Canada: McMaster University, 2011.

[5] V. Shinde, E. Longatte, F. Baj, and M. Braza. “A theoretical model of fluidelastic instability in tube arrays”. In: *Nuclear Engineering and Design* 337 (0029-5493 2018), pp. 406–418. DOI: 10.1016/j.nucengdes.2018.07.011.

[6] D. S. Weaver and J. H. Lever. “Tube frequency effect on cross flow induced vibrations in tube array.” In: *Proceedings of the 5th Biennial Symposium on Turbulence* 1 (1977), pp. 323–331.

[7] J. Wang and D. S. Weaver. “Fluidelastic Instability in Normal and Parallel Triangular Arrays of Finned Tubes”. In: *Journal of Pressure Vessel Technology* 134 (2012). ISSN: 021302. DOI: 10.1115/1.4004621.

[8] C. Meskell and J. A. Fitzpatrick. “Investigation of the Nonlinear Behavior of Damping Controlled Fluid-Elastic Instability in a Normal Triangular Tube Array”. In: *Journal of Fluids and Structures* 18 (2003), pp. 573–593. ISSN: x. DOI: 10.1016/j.fluidstructs.2003.08.013.

[9] J. Mahon and C. Meskell. “Surface pressure distribution survey in normal triangular tube arrays”. In: *Journal of Fluids and Structures* 25 (2009), pp. 1348–1368. DOI: 10.1016/j.fluidstructs.2009.07.006.

[10] D. R. Polak and D. S. Weaver. “Vortex Shedding in Normal Triangular Tube Arrays”. In: *Journal of Fluids and Structures* 9 (1995), pp. 1–17. ISSN: 0889-9746.

[11] F. Menter, M. Kuntz, and R. Langtry. “Ten years of industrial experience with the SST turbulence model.” In: *Proceedings of the fourth international symposium on turbulence, heat and mass transfer* (2003), pp. 625–632. DOI: 10.2478/IJNAOE-2013-0011.

[12] N. L. R. Center. *The Menter Shear Stress Transport Turbulence Model*. https://turbmodels.larc.nasa.gov/sst.html. 2015.

[13] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby. “A tensorial approach to computational continuum mechanics using object-oriented techniques”. In: *Computers in Physics* 12 (1998), pp. 620–631. ISSN: 6. DOI: 10.1063/1.168744.

[14] H. Jasak. “Openfoam: open source CFD in research and industry”. In: *International Journal of Naval Architecture and Ocean Engineering* 1 (2009), pp. 89–94. ISSN: 2. DOI: 10.2478/IJNAOE-2013-0011.

[15] S. V. Patankar and D. B. Spalding. “A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows”. In: *International Journal of Heat and Mass Transfer* 15 (1972), pp. 1787–1806. ISSN: 10. DOI: 10.1016/0017-9310(72)90054-3. (Visited on 03/01/2017).

[16] R. I. Issa. “Solution of the implicitly discretized fluid flow equations by operator splitting”. In: *Journal of Computational Physics* 62 (1985), pp. 40–65. ISSN: 1. DOI: 10.1016/0021-9991(86)90099-9.

[17] OpenFOAM. *Turbulence model documentation*. https://www.openfoam.com/documentation/guides/latest/doc/guide-turbulence-ras-k-omega-sst.html. 2020.

[18] OpenFOAM. *Overset grid documentation*. https://www.openfoam.com/documentation/guides/latest/doc/guide-overset.html. 2020.

[19] H. H. Jafari and B. G. Dehkordi. “Numerical Prediction of Fluid-Elastic Instability in Normal Triangular Tube Bundles With Multiple Flexible Circular Cylinders”. In: *Journal of Fluids Engineering* 135 (2013). Transactions of the ASME. ISSN: 031102. DOI: 10.1115/1.4023298.

[20] A. Placzek, J.-F. Sigrist, and A. Hamdouni. “Numerical Simulation of an Oscillating Cylinder in a Cross-Flow at Low Reynolds Number: Forced and Free Oscillations”. In: *Computers and Fluids* 38 (2009), pp. 80–100. ISSN: 1. DOI: 10.1016/j.compfluid.2008.01.007.

[21] A. Khalak and C. Williamson. “Dynamics of hydroelastic cylinder with very low mass and damping”. In: *Journal of Fluids and Structures* 10 (1996), pp. 455–472.

[22] E. Guilmineau and P. Queutey. “Numerical simulation of vortex-induced vibration of a circular cylinder with low mass-damping in turbulent flow”. In: *Journal of Fluids and Structures* 19 (2004), pp. 449–466. DOI: 10.1016/j.fluidstructs.2004.02.004.

[23] J. Meneghini, F. Sahara, and P. Bearman. “Numerical Simulation of Vortex Shedding From an Oscillating Circular Cylinder.” In: *WIT Transactions on Modelling and Simulation* 17 (1997). DOI: 10.2495/CMEM970401.

[24] J. Wanderley and C. Levi. “Vortex induced loads on marine risers.” In: *Ocean Engineering* 32 (2005), pp. 1281–1295. ISSN: 11-12. DOI: 10.1016/j.oceaneng.2004.12.007.

[25] K. Schröder and H. Gelbe. “Two- and three-dimensional CFD-simulation of flow-induced vibration excitation in tube bundles”. In: *Chemical Engineering and Processing* 38 (1999), pp. 621–629.

A Code listings

A frequently encountered issue is the lack of documentation of numerical solver settings in the literature. Listing 1 gives the recommended numerical schemes for the wind-tunnel setup. The numerical diffusion introduced via the first order upwind discretisation of the convective terms in the turbulent kinetic energy and turbulent dissipation rate acts stabilising. The linear solvers and correponding solution criteria are specificed in Listing 2. 10 PIMPLE iterations with 2 pressure-velocity corrections are employed. In the final PIMPLE iteration, the convergence criteria for velocity and turbulent fields are reduced. It is important to note the specifiation of “pRefValue 0” and “pRefCell 0” in the Listing 2 file. Further, at least 10 “moveMeshOuterCorrectors” should be applied to obtain converged force calculations in dynamic mesh simulations.

Finally, Listing 3 gives the mesh creation file for the wind-tunnel setup. It should be used on the background mesh produced by the file under Listing 4.

Listing 1: fvSchemes

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  v1812
|
|   \\  /    A nd           | Web:      www.OpenFOAM.com
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
```

```
{
    default Euler;
    //default CrankNicolson 0.9;
    //default steadyState; // simpleFoam
}

gradSchemes
{
    default          Gauss linear;
    grad(p)          Gauss linear;
    grad(U)          Gauss linear;
}

divSchemes
{
    default          none;
    div(phi,U)       Gauss linearUpwind grad(U);
    div(phi,k)       Gauss upwind;
    div(phi,omega)   Gauss upwind;
    div((nuEff*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default          Gauss linear corrected;
}

interpolationSchemes
{
    default          linear;
}

snGradSchemes
{
    default          corrected;
}

wallDist
{
    method meshWave;
}


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

Listing 2: fvSolution

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
```

```
|
|  \\      /   O peration      | Version:  v1812
|
|   \\    /    A nd            | Web:      www.OpenFOAM.com
|
|    \\/      M anipulation   |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    "pcorr.*"
    {
        solver          GAMG;
        tolerance       0.001;
        relTol          0;
        smoother        GaussSeidel;
    }

    p
    {
        $pcorr;
        tolerance       1e-7;
        relTol          0.001;
    }

    pFinal
    {
        $p;
        tolerance       1e-7;
        relTol          0;
    }

    "(U|k|omega)"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-07;
        relTol          0.1;
    }

    "(U|k|omega)Final"
```

```
    {
        $U;
        tolerance       1e-08;
        relTol          0;
    }

    cellDisplacement
    {
        solver          GAMG;
        tolerance       1e-7;
        relTol          0;
        smoother        GaussSeidel;
    }
}

PIMPLE
{
    correctPhi          yes;
    nOuterCorrectors    10;
    nCorrectors         2;
    nNonOrthogonalCorrectors 0;
    moveMeshOuterCorrectors yes;
    pRefCell            0;
    pRefValue           0;
}

relaxationFactors
{
}

cache
{
    grad(U);
}

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

Listing 3: snappyHexMeshDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  v1812
|
|   \\  /    A nd           | Web:      www.OpenFOAM.com
|
|    \\/     M anipulation  |
|
```

```
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      snappyHexMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

castellatedMesh true;
snap            true;
addLayers       true;

geometry
{
    c
    {
        type searchableCylinder;
        point1 (0 0 -20e-3);
        point2 (0 0 20e-3);
        radius 19e-3;
    }

    c1
    {
        type searchableCylinder;
        point1 (-86.6e-3 150e-3 -20e-3);
        point2 (-86.6e-3 150e-3 20e-3);
        radius 19e-3;
    }
    c2
    {
        type searchableCylinder;
        point1 (0 150e-3 -20e-3);
        point2 (0 150e-3 20e-3);
        radius 19e-3;
    }
    c3
    {
        type searchableCylinder;
        point1 (86.6e-3 150e-3 -20e-3);
        point2 (86.6e-3 150e-3 20e-3);
        radius 19e-3;
    }
    c4
    {
        type searchableCylinder;
        point1 (-43.3e-3 125e-3 -20e-3);
        point2 (-43.3e-3 125e-3 20e-3);
```

```
    radius 19e-3;
}
c5
{
    type searchableCylinder;
    point1 (43.3e-3 125e-3 -20e-3);
    point2 (43.3e-3 125e-3 20e-3);
    radius 19e-3;
}
c6
{
    type searchableCylinder;
    point1 (-86.6e-3 100e-3 -20e-3);
    point2 (-86.6e-3 100e-3 20e-3);
    radius 19e-3;
}
c7
{
    type searchableCylinder;
    point1 (0 100e-3 -20e-3);
    point2 (0 100e-3 20e-3);
    radius 19e-3;
}
c8
{
    type searchableCylinder;
    point1 (86.6e-3 100e-3 -20e-3);
    point2 (86.6e-3 100e-3 20e-3);
    radius 19e-3;
}

c9
{
    type searchableCylinder;
    point1 (-43.3e-3 75e-3 -20e-3);
    point2 (-43.3e-3 75e-3 20e-3);
    radius 19e-3;
}
c10
{
    type searchableCylinder;
    point1 (43.3e-3 75e-3 -20e-3);
    point2 (43.3e-3 75e-3 20e-3);
    radius 19e-3;
}
c11
{
    type searchableCylinder;
    point1 (-86.6e-3 50e-3 -20e-3);
    point2 (-86.6e-3 50e-3 20e-3);
```

```
    radius 19e-3;
}
c12
{
    type searchableCylinder;
    point1 (0 50e-3 -20e-3);
    point2 (0 50e-3 20e-3);
    radius 19e-3;
}
c13
{
    type searchableCylinder;
    point1 (86.6e-3 50e-3 -20e-3);
    point2 (86.6e-3 50e-3 20e-3);
    radius 19e-3;
}
c14
{
    type searchableCylinder;
    point1 (-43.3e-3 25e-3 -20e-3);
    point2 (-43.3e-3 25e-3 20e-3);
    radius 19e-3;
}
c15
{
    type searchableCylinder;
    point1 (43.3e-3 25e-3 -20e-3);
    point2 (43.3e-3 25e-3 20e-3);
    radius 19e-3;
}
c16
{
    type searchableCylinder;
    point1 (-86.6e-3 0 -20e-3);
    point2 (-86.6e-3 0 20e-3);
    radius 19e-3;
}
c18
{
    type searchableCylinder;
    point1 (86.6e-3 0 -20e-3);
    point2 (86.6e-3 0 20e-3);
    radius 19e-3;
}
c19
{
    type searchableCylinder;
    point1 (-43.3e-3 -25e-3 -20e-3);
    point2 (-43.3e-3 -25e-3 20e-3);
    radius 19e-3;
```

```
}
c20
{
    type searchableCylinder;
    point1 (43.3e-3 -25e-3 -20e-3);
    point2 (43.3e-3 -25e-3 20e-3);
    radius 19e-3;
}
c21
{
    type searchableCylinder;
    point1 (-86.6e-3 -50e-3 -20e-3);
    point2 (-86.6e-3 -50e-3 20e-3);
    radius 19e-3;
}
c22
{
    type searchableCylinder;
    point1 (0 -50e-3 -20e-3);
    point2 (0 -50e-3 20e-3);
    radius 19e-3;
}
c23
{
    type searchableCylinder;
    point1 (86.6e-3 -50e-3 -20e-3);
    point2 (86.6e-3 -50e-3 20e-3);
    radius 19e-3;
}
c24
{
    type searchableCylinder;
    point1 (-43.3e-3 -75e-3 -20e-3);
    point2 (-43.3e-3 -75e-3 20e-3);
    radius 19e-3;
}
c25
{
    type searchableCylinder;
    point1 (43.3e-3 -75e-3 -20e-3);
    point2 (43.3e-3 -75e-3 20e-3);
    radius 19e-3;
}
c26
{
    type searchableCylinder;
    point1 (-86.6e-3 -100e-3 -20e-3);
    point2 (-86.6e-3 -100e-3 20e-3);
    radius 19e-3;
}
```

```
c27
{
    type searchableCylinder;
    point1 (0 -100e-3 -20e-3);
    point2 (0 -100e-3 20e-3);
    radius 19e-3;
}
c28
{
    type searchableCylinder;
    point1 (86.6e-3 -100e-3 -20e-3);
    point2 (86.6e-3 -100e-3 20e-3);
    radius 19e-3;
}
c29
{
    type searchableCylinder;
    point1 (-43.3e-3 -125e-3 -20e-3);
    point2 (-43.3e-3 -125e-3 20e-3);
    radius 19e-3;
}
c30
{
    type searchableCylinder;
    point1 (43.3e-3 -125e-3 -20e-3);
    point2 (43.3e-3 -125e-3 20e-3);
    radius 19e-3;
}
c31
{
    type searchableCylinder;
    point1 (-86.6e-3 -150e-3 -20e-3);
    point2 (-86.6e-3 -150e-3 20e-3);
    radius 19e-3;
}
c32
{
    type searchableCylinder;
    point1 (0 -150e-3 -20e-3);
    point2 (0 -150e-3 20e-3);
    radius 19e-3;
}
c33
{
    type searchableCylinder;
    point1 (86.6e-3 -150e-3 -20e-3);
    point2 (86.6e-3 -150e-3 20e-3);
    radius 19e-3;
}
```

```
    box_horizontal
    {
        type searchableBox;
        min     (-0.15 -150e-3 -20e-3);
        max     (0.35   150e-3 20e-3);
    }
}

castellatedMeshControls
{
    maxLocalCells 3000000;
    maxGlobalCells 10000000;
    minRefinementCells 10;
    maxLoadUnbalance 0;
    nCellsBetweenLevels 6;

    features
    (
    );

    refinementSurfaces
    {

        "(c)"
        {
            level (3 3 );
            patchInfo
            {
                type wall;
            }

        }

        "(c.*)"
        {
            level (3 3 );
            patchInfo
            {
                type wall;
            }

        }


     }

    resolveFeatureAngle 30;

    refinementRegions
    {
```

```
        box_horizontal
        {
            mode inside;
            levels ((1e15 2));
        }
    }

    locationInMesh (5.01326e-3 25.5698e-3 0.14536e-3);
    allowFreeStandingZoneFaces true;
}


snapControls
{
    nSmoothPatch 3;
    tolerance 4.0;
    nSolveIter 30;
    nRelaxIter 5;

    implicitFeatureSnap false;
    explicitFeatureSnap true;
    multiRegionFeatureSnap false;
}

addLayersControls
{
    relativeSizes true;
    layers
    {
        "(c.*)"
        {
            nSurfaceLayers 3;
        }
    }

    expansionRatio 1.1;

    finalLayerThickness 0.7;
    minThickness 0.25;
    nGrow 0;

    featureAngle 160;
    nRelaxIter 5;
    nSmoothSurfaceNormals 1;
    nSmoothNormals 3;
    nSmoothThickness 10;
    maxFaceThicknessRatio 0.5;
    maxThicknessToMedialRatio 0.3;
    minMedialAxisAngle 90;
    nBufferCellsNoExtrude 0;
```

```
    nLayerIter 50;
}

meshQualityControls
{
    maxNonOrtho 65;

    maxBoundarySkewness 20;
    maxInternalSkewness 4;

    maxConcave 80;
    minVol 1e-13;

    minTetQuality 1e-30;
    minArea -1;

    minTwist 0.05;
    minDeterminant 0.001;

    minFaceWeight 0.05;

    minVolRatio 0.01;

    minTriangleTwist -1;

    nSmoothScale 4;
    errorReduction 0.75;
}

writeFlags
(
    scalarLevels
    layerSets
    layerFields
);

mergeTolerance 1e-6;


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

Listing 4: blockMeshDict

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  plus
|
|   \\  /    A nd           | Web:      www.OpenFOAM.com
```

```
|
|     \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
scale   0.001;  // mm scale

vertices
(

  (-350 -150 -10)    // Vertex p0 = 0
  (350 -150 -10)    // Vertex p1 = 1
  (350 150 -10)    // Vertex p2 = 2
  (-350 150 -10)    // Vertex p3 = 3

  (-350 -150 10)    // Vertex p4 = 4
  (350 -150 10)    // Vertex p5 = 5
  (350 150 10)    // Vertex p6 = 6
  (-350 150 10)    // Vertex p7 = 7

);

blocks
(

    hex (0 1 2 3 4 5 6 7) (140 60 1)
    simpleGrading (1 1 1)


);

edges
(

);

boundary
(
    left
    {
      type patch;
      faces
      (
```

```
            (4 7 3 0)
        );
    }
    right
    {
        type patch;
        faces
        (
            (5 1 2 6)
        );
    }
    front
    {
        type wall;
        faces
        (
            (4 0 1 5)
        );
    }
    back
    {
        type wall;
        faces
        (
            (7 6 2 3)
        );
    }
    top
    {
        type wall;
        faces
        (
            (4 5 6 7)
        );
    }
    bottom
    {
        type wall;
        faces
        (
            (0 3 2 1)

        );
    }

);

mergePatchPairs
(
);
```

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

Technology for a better society

www.sintef.no