VFS-Geophysics
User Manual

Prologue

VFS-Geophysics is the result of many years of research work by several graduate students, post-docs, and research associates that have been involved in the Computational Hydrodynamics and Biofluids Laboratory directed by Professor Fotis Sotiropoulos. The preparation of the present manual has been supported by the U.S. Department of Energy (DE-EE 0005482).

Current Code Developers

Ali Khosronejad
Xiaolei Yang
Christian Santoni

Former Code Developers

Iman Borazjani
Liang Ge
Seokkoo Kang
Dennis Angelidis
Antoni Calderer
Anvar Gilmanov
Trung Le

License

Contents

Chapter 1

Introduction

Virtual Flow Solver - Geophysics (VFS-Geophysics) is a three-dimensional (3D) incompressible Navier-Stokes solver based on the Curvilinear Immersed Boundary (CURVIB) method developed by Ge and Sotiropoulos [1]. The CURVIB is a sharp interface type of immersed boundary (IB) method that enables the simulation of fluid flows in the presence of geometrically complex moving bodies. In IB approaches, the structural body mesh is superposed on the underlying Eulerian fluid mesh that is kept fixed. This approach circumvents the limitation of classical body fitted methods in which the fluid mesh adapts to the body, and thus is limited to relatively simple geometries and small amplitude motions.

A particularity of the CURVIB method with respect to most sharp interface IB methods is that it is formulated in generalized curvilinear coordinates. This feature allows application of a body-fitted approach for the simpler boundaries while maintaining the ability to incorporate complex and moving geometries. For instance, in marine and hydrokinetic (MHK) energy applications, one could take advantage of this feature when simulating the site-specific geometry of a MHK farm. The fluid mesh can follow the actual topography of the terrain while treating the turbines as immersed bodies.

The CURVIB method has been applied to a broad range of applications, such as cardiovascular flows [2, 3, 4], riverbed morphodynamics [5, 6, 7], and wind/ MHK turbine simulations [8, 9]. For wind/ MHK energy applications, a turbine can be resolved by immersing the geometry with the IB method or by using one of the different rotor parameterization models implemented.

Chapter 2

Overview of the Numerical Algorithms

2.1 The Flow Solver

The code solves the spatially filtered Navier-Stokes equations governing incompressible flows.

$$J\frac{\partial U^i}{\partial \xi^i} = 0 \tag{2.1}$$

$$\frac{\partial U^j}{\partial t} = \xi_l^i\left[-\frac{\partial U^j u_l}{\partial \xi_j} + \frac{1}{Re}\frac{\partial}{\partial \xi^j}\left(\frac{\xi_l^j \xi_l^k}{J}\frac{\partial u_l}{\partial \xi^k}\right) - \frac{\partial}{\partial \xi^j}\left(\frac{\xi_l^j p}{J}\right) - \frac{\partial \tau_{lj}}{\partial \xi^j} + \frac{\delta_{i2}}{Fr^2}\right] \tag{2.2}$$

where ξ^i are the curvilinear components, ξ_l^i are the transformation metrics, J is the Jacobian of the transformation, U^i are the contravariant volume fluxes, u_i are the Cartesian velocity components, ρ is the density, μ is the dynamic viscosity, p is the pressure, τ_{lj} is the sub-grid scale (SGS) tensor, δ_{ij} is the Kronecker delta, and Re, and Fr are respectively the dimensionless Reynolds and Froude numbers. These numbers are defined as:

$$Re = \frac{UL\rho}{\mu}, Fr = \frac{U}{\sqrt{gL}} \tag{2.3}$$

where U is the characteristic velocity, L is the linear dimension, ρ is the density, μ is the dynamic viscosity, and g is the gravitational acceleration.

2.2 The CURVIB Method

The code can simulate flows around geometrically complex moving bodies with the sharp interface CURVIB method developed by Ge and Sotiropoulos [1]. The method has been thoroughly validated in many applications, such as fluid-structure interaction (FSI) problems [10, 11], riverbed morphodynamics [5, 6, 7], and cardiovascular flows [2, 3, 4].

In the CURVIB method, the bodies are represented by an unstructured triangular mesh that is superposed on the background curvilinear or Cartesian fluid grid. First, the computational domain's nodes are classified depending on their location with respect to the position of the body. The nodes that fall inside the body are considered structural nodes and are blanked out from the computational domain. The nodes that are located in the fluid but in the immediate vicinity of the structure are denoted as IB nodes, where the boundary condition of the velocity field is reconstructed. The remaining nodes are the fluid nodes where the governing equations are solved.

The velocity reconstruction is performed in the wall normal direction with either linear or quadratic interpolation in the case of low Reynolds number flows when the IB nodes are located in the viscous sub-layer. otherwise, the velocity reconstruction uses the wall models described by [12, 13, 14] in high Reynolds number flows when the grid resolution is not sufficient to accurately resolve the viscous sub-layer.

The distance function φ also needs to be reconstructed at the body-fluid interface. This is done by setting gradient Δφ to be zero at the cell faces that are located between the fluid and IB nodes as described in [15].

2.3 The Structural Solver and the Fluid-Structure Interaction Algorithm

The original FSI algorithm implementation for incompressible flows is described in Borazjani, Ge, and Sotiropoulos [10].

The code solves the rigid body equations of motion (EoM) in 6 degrees of freedom (DoF), which can be written in Lagrangian form for the principal axis as follows (i=1, 2, ..., 6):

$$M\frac{\partial Y^2}{\partial t^2} + C\frac{\partial Y^i}{\partial t} + KY^i = F_f^i + F_e^i \tag{2.4}$$

where Y^i represents the coordinates of the Lagrangian vector describing the structure's motion. For the translational DoFs, Y^i are the Cartesian components of the body position, M is the mass matrix, C is the damping coefficients matrix, K is the spring stiffness coefficient matrix, F_f^i are the forces exerted by the fluid, and F_e^i are the components of the external force vector. For the rotational DoFs, Y^i are relative angle components of the body, M represents the moment of inertia, and F_f^i and F_e^i are the moments around the rotation axis, respectively induced by the fluid and by the external forces.

The forces and moments that the fluid exerts on the rigid body are computed by integrating the pressure and the viscous stresses along the surface Γ of the body as follows:

$$F_f = \int_\Gamma -pn\, d\Gamma + \int_\Gamma \tau_{ij}\, n_j d\Gamma \tag{2.5}$$

$$M_f = \int_\Gamma -\epsilon_{ijk}\, r_j p n_k d\Gamma + \int_\Gamma \epsilon_{ijk}\, r_j \tau_{kl} n_l d\Gamma \tag{2.6}$$

where p denotes the pressure, τ the viscous stress, ϵ_{ijk} the permutation symbol, r the position vector, and n the normal vector.

The EoM (equation (2.4)) are coupled with the fluid domain equations through a partitioned FSI approach. The time integration can be done explicitly with loose coupling (LC-FSI), or implicitly with strong coupling (SC-FSI). The Aitken acceleration technique of [16] allows for significant reduction in the number of sub-iterations when the SC-FSI algorithm is used. A detailed description of both time-integration algorithms is given in [10].

2.4 Turbine Parameterizations

The actuator disk, actuator line, and actuator surface models implemented in the code for parameterizing turbine rotors are given in Yang, Kang, and Sotiropoulos [8] and in Yang et al. [17]. The basic idea of these models is to extract from the flow field the kinetic energy that is estimated to be equivalent to that from a turbine rotor, without the need to resolve the flow around its geometry. To introduce such kinetic energy reduction into the flow, a sink term, affecting the fluid nodes located at the vicinity of the turbine, is considered in the right-hand side of the momentum equations.

2.4.1 Turbine Resolving

In applications such as simulations of marine hydrokinetic turbines, it is computationally impractical to impose a no-slip boundary for flows with high Reynolds numbers. Thus, the CURVIB wall model was adopted by Kang et al. [31] to resolve the simplified turbulent boundary layer (TBL) equations. The solution of the TBL equations is based on the equilibrium stress balance model expounded by Wang and Moin [32]. The user can read a more detailed description of the wall model in Kang et al. [33].

The turbine resolution utilizes a hybrid staggered/non-staggered grid fractional step method as proposed by Ge and Sotiropoulos [34] and Kang et al. [31]. In the first step, the Navier-Stokes momentum equations are discretized in time by the second-order Crank-Nicolson method and are solved by the matrix-free Newton-Krylov method [33]. In the next step, the Poisson equation for pressure correction is solved, which is the following:

$$-J\frac{\partial}{\partial\xi^i}\left(\frac{1}{\rho}\frac{\xi_l^i}{J}\frac{\partial}{\partial\xi^j}\left(\frac{\xi_l^j\Pi}{J}\right)\right)=\frac{1}{\Delta t}J\frac{\partial U^{j,*}}{\partial\xi^j} \tag{2.7}$$

where $\Pi = p^{n+1} - p^n$ and the superscript * denotes the variables computed in the first step. In the last step, the solution of equation (2.7) is taken to update the velocity and pressure fields, and so the updated velocity becomes divergence free. The user can read more details about this method in Kang et al. [33].

When an immersed boundary (e.g., blade) moves, a local non-zero velocity flux arises at the interface between the immersed boundary and the internal nodes. This non-zero velocity flux is incorporated into the Poisson equation's right-hand side as a boundary condition to account the moving boundaries' inertial effects on the flow field.

2.4.2 Actuator Disk Model

In the actuator disk model, the turbine rotor is represented by a circular disk that is discretized with an unstructured triangular mesh. The body force of the disk per unit area is the following:

$$F_{AD} = -\frac{F_T}{\pi D^2/4} \tag{2.8}$$

where D is the rotor diameter and F_T is the thrust force computed as:

$$F_T = \frac{1}{2}\rho C_T\frac{\pi}{4}D^2U_\infty^2 \tag{2.9}$$

where U_∞ is the turbine incoming velocity, $C_T = 4a(1-a)$ is the thrust coefficient taken from the one-dimensional momentum theory, and a is the induction factor. The incoming velocity U_∞ is also computed from the one-dimensional momentum theory as

$$U_\infty = \frac{u_d}{1-a} \tag{2.10}$$

where u_d is the disk-averaged streamwise velocity computed as:

$$u_d = \frac{4}{\pi D^2}\sum_{N_t} u(X)A(X) \tag{2.11}$$

where N_t is the number of triangular elements composing the disk mesh, $A(X)$ is the area of each element, and $u(X)$ is the velocity at the element centers. The fluid velocity at the disk ($u(X)$) requires interpolation from the velocity values at the surrounding fluid mesh points as the nodes from the fluid and disk meshes do not necessarily coincide. If we consider *X* to be the coordinates of the actuator disk nodes and *x* the coordinates of the fluid mesh nodes, the interpolation, which uses a discrete delta function, reads as follows:

$$u(X) = \sum_{N_D} u(x)\delta_h(x - X)V(x) \tag{2.12}$$

where h is a 3D discrete delta function, *V (x)* is the volume of the corresponding fluid cell, and N_D is the number of fluid cells involved in the interpolation. Finally, the body force f_{AD}, which is computed at the disk mesh nodes, needs to be distributed over the fluid cells located in the immediate vicinity using the following expression:

$$f_{AD}(x) = \sum_{N_D} F_{AD}(X)\delta_h(x - X)A(x) \tag{2.13}$$

2.4.3 Actuator Line Model

In the actuator line method, each blade of the rotor is modeled with a straight line, divided in several elements along the radial direction. In each of the elements, the lift C_L and drag C_D forces are computed using the following expressions:

$$L = \frac{1}{2}\rho C_L C V_{rel}^2, \tag{2.14}$$

$$D = \frac{1}{2}\rho C_D C V_{rel}^2, \tag{2.15}$$

where C_L and C_D are respectively the lift and drag coefficients taken from tabulated two-dimensional (2D) airfoil profile data, C is the chord length, and V_{rel} is the incoming reference velocity computed as:

$$V_{rel} = (u_z, u_\theta - \Omega r) \tag{2.16}$$

where u_z and $u_\theta - \Omega r$ are the velocity components respectively in the axial and azimuthal directions, Ω the rotor's angular velocity, and r is the distance to the center of the rotor.

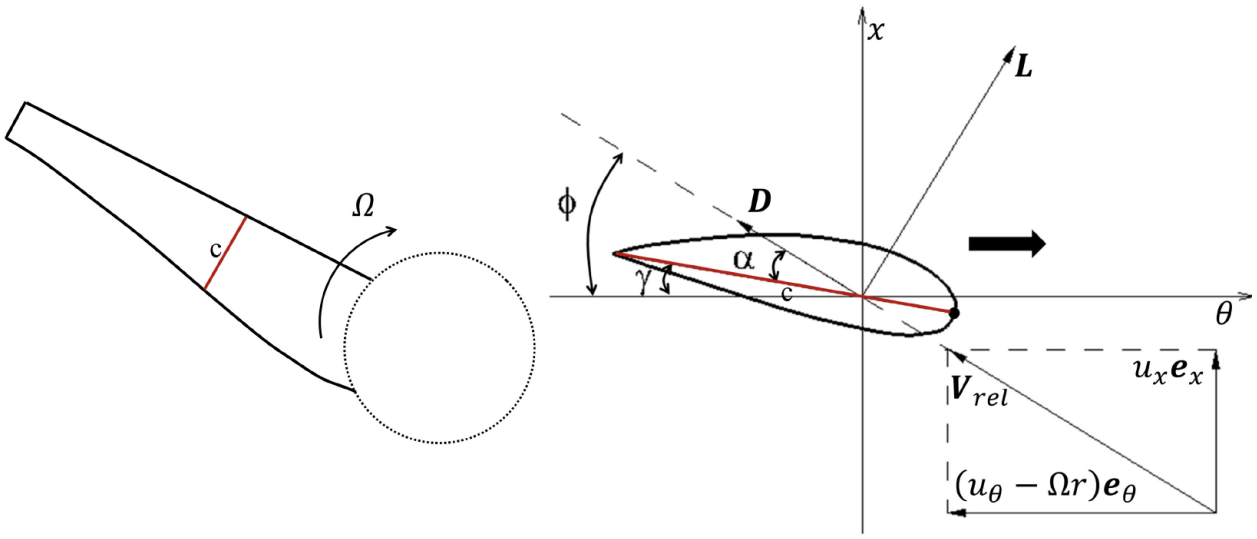
To compute the reference velocity at the line elements, similarly to the actuator disk method, the velocity at the fluid mesh is transferred to the line elements using a discrete delta function as given by equation (2.12). Once the lift and drag forces are computed at each of the line elements, the distributed body force in the fluid mesh can be computed using the following equation:

$$f_{AL}(x) = \sum_{N_L} F(X)\delta_h(x - X)A(x) \tag{2.17}$$

where N_L is the number of segments composing one of the actuator lines, $F(X)$ is the projection of *L* and *D*, expressed in actuator line local coordinates, into Cartesian coordinates.

2.4.4 Actuator Surface model

In both actuator surface models, we have two sets of independent meshes: i.e., the background Cartesian grid for the flow with its coordinate denoted by x (x, y, z or x_1, x_2, x_3), and the Lagrangian grid following the actuator surfaces with its coordinate denoted by X (X, Y, Z or X_1, X_2, X_3). For both models, the smoothed discrete delta function in [18] is employed as the kernel function for transferring information between the two meshes. The major difference between the blade actuator surface model and the nacelle actuator surface model is the way in which the forces on the actuator surfaces are calculated.



(a) Actuator surface of a blade (b) Cross section of a blade

Figure 2.1: A schematic of the actuator surface model for blade. The lift and drag forces calculated using the blade element method are distributed over the actuator surface formed by chord lines of a blade.

2.4.4.1 Actuator surface model for blade

In the blade actuator surface model, the blade geometry is represented by a surface formed by the chord lines at different radial locations of a blade as shown in Figure 2.1(a). The forces are calculated in the same way as in the actuator line model. The lift (L) and drag (D) at each radial location are calculated by:

$$L = \frac{1}{2}\rho C_L \mathrm{c}|\mathbf{V_{rel}}|^2 \mathbf{e}_L \tag{2.18}$$

$$D = \frac{1}{2}\rho C_D \mathrm{c}|\mathbf{V_{rel}}|^2 \mathbf{e}_D \tag{2.19}$$

where C_L and C_D are the lift and drag coefficients, c is the chord length, $\mathbf{V_{rel}}$ is the relative incoming velocity, and $\mathbf{e}_L$ and $\mathbf{e}_D$ are the unit vectors in the directions of lift and drag, respectively. The C_L and C_D are functions of the Reynolds number and the angle of attack. The angle of attack α shown in Figure 2.1(b) is computed at each radial location by:

$$\alpha = \phi - \gamma \tag{2.30}$$

where $\phi = -\tan^{-1}\left(u_x/(u_\theta - \Omega r)\right)$, and γ is the angle including the blade twist and the blade pitch, the latter of which is specified to a given value instead of from a turbine control algorithm. The relative incoming velocity $\mathbf{V_{rel}}$ at each radial location is computed by:

$$\mathrm{V_{rel}} = \mathrm{u_x}\mathbf{e_x} + (\mathrm{u_\theta} - \Omega\,\mathrm{r})\mathbf{e}_\theta \tag{2.31}$$

where Ω is the rotor's rotational speed, and $\mathbf{e}_x$ and $\mathbf{e}_\theta$ are respectively the unit vectors in the axial and azimuthal directions. The u_x and u_θ are the axial and azimuthal components of the flow velocity averaged over the chord for each radial location, which are computed using:

$$\mathrm{u_x} = \frac{1}{\mathrm{c}}\int_{\mathrm{c}} \mathrm{u(X)} \cdot \mathrm{e_x ds} \tag{2.32}$$

$$\mathrm{u_\theta} = \frac{1}{\mathrm{c}}\int_{\mathrm{c}} \mathrm{u(X)} \cdot \mathrm{e_\theta ds} \tag{2.33}$$

where X denote the coordinates of the grid points on the actuator surfaces.

Generally, the grid points on the actuator surfaces do not coincide with any background nodes. We employ a smoothed discrete delta function (i.e., the smoothed four-point cosine function) proposed by Yang et al. [18] to interpolate $u(X)$ from the values on the background grid nodes as follows:

$$\mathrm{u(X)} = \sum_{\mathrm{x}\in \mathrm{g_x}} \mathrm{u(x)}\delta_\mathrm{h}(\mathrm{x} - \mathrm{X})\mathrm{V(x)} \tag{2.34}$$

where $\boldsymbol{x}$ are the coordinates of the background grid nodes, $g_{\boldsymbol{x}}$ is the set of the background grid cells, $V = h_x h_y h_z$ (h_x, h_y and h_z are respectively the grid spacings in the x, y and z directions) is the volume of the background grid cell, $\delta_h(\boldsymbol{x} - \boldsymbol{X}) = \frac{1}{V}\phi\left(\frac{x-X}{h_x}\right)\phi\left(\frac{y-Y}{h_y}\right)\phi\left(\frac{z-Z}{h_z}\right)$ is the discrete delta function, and ϕ is the smoothed four-point cosine function [18], which is expressed as:

$$\begin{cases} \frac{1}{4} + \frac{sin(\pi(2|r|+1)/4)}{2\pi} - \frac{sin(\pi(2|r|-1)/4)}{2\pi}, & |r| \le 1.5, \\ \frac{5}{8} - \frac{|r|}{4} - \frac{sin(\pi(2|r|-1)/4)}{2\pi}, & 1.5 \le |r| \le 2.5, \\ 0, & 2.5 \le |r| \end{cases} \tag{2.35}$$

in which $r_i = (x_i - X_i)/h_i$ (i=1, 2, 3).

The blade rotation causes the stall delay phenomenon at the blade's inboard sections, which increases the lift coefficients and decreases the drag coefficients as compared with the corresponding two-dimensional airfoil data. To account for such a three-dimensional rotational effect, the stall delay model developed by Du and Selig [19] is employed to correct the lift and drag coefficients from two-dimensional experiments or numerical simulations. In Du and Selig's model, the corrected lift and drag coefficients ($C_{L,3D}$ and $C_{D,3D}$) are calculated as follows:

$$\mathrm{C_{L,3D}} = \mathrm{C_{L,2D}} + \mathrm{f_L}\left(\mathrm{C_{L,p}} - \mathrm{C_{L,2D}}\right) \tag{2.36}$$

$$C_{D,3D} = C_{D,2D} - f_D\left(C_{D,2D} - C_{D,0}\right) \tag{2.37}$$

where $C_{L,p} = 2\pi(\alpha - \alpha_0)$, $C_{D,0}$ is the two-dimensional drag coefficient at zero angle of attack, and the correction functions f_L and f_D are respectively determined by:

$$f_L = \frac{1}{2\pi}\left(\frac{1.6(c/r)a - (c/r)^{\frac{dR}{\Lambda r}}}{0.1267b + (c/r)^{\frac{dR}{\Lambda r}}} - 1\right) \tag{2.38}$$

$$f_D = \frac{1}{2\pi}\left(\frac{1.6(c/r)a - (c/r)^{\frac{d\,R}{2\Lambda r}}}{0.1267b + (c/r)^{\frac{d\,R}{2\Lambda r}}} - 1\right) \tag{2.39}$$

where $\Lambda = \Omega R/\sqrt{U^2 + (\Omega R)^2}$, U is the incoming wind/ water speed, R is the rotor radius, and a, b, and d are the empirical correction factors. In this work, a, b and d are equal to 1 as in Du and Selig's paper [19].

Non-zero force can exist at the blade tip when the pitch angle is nonzero, or when a chambered foil is used [20]. This is in contradiction with the physical understanding that the force should tend to zero at the tip due to pressure equalization as discussed by Shen et al. [20]. To correct this non-physical force behavior, the tip-loss correction proposed by Shen et al. [20, 21] is applied to the drag and lift coefficients computed from equation (2.36) and (2.37). With the tip loss correction, the employed C_D and C_L are calculated as:

$$C_L = F_1 C_{L,3D} \tag{2.40}$$

$$C_D = F_1 C_{D,3D} \tag{2.41}$$

where:

$$F_1 = \frac{2}{\pi}\cos^{-1}\left(\exp\left(-g\,\frac{B(R-r)}{2r\sin\phi}\right)\right) \tag{2.42}$$

in which B is the number of blades, and g is computed by:

$$g = \exp\left(-c_1(B\Omega R/U_\infty - c_2)\right) + c_3 \tag{2.43}$$

where c_1, c_2, and c_3 are the correction coefficients, which are respectively equal to 0.125, 21, and 0.1 as in [21].

After calculating the lift (**L**) and drag (**D**), the force per unit area on the actuator surface at each radial location is then calculated as:

$$f(X) = (L + D)/c \tag{2.44}$$

Note that the above expression essentially means that the lift and drag on the blade are uniformly distributed in the chordwise direction. To calculate the forces on the background mesh for the flow field, the computed forces on the actuator surfaces are then distributed to the background grid nodes as follows:

$$f(x) = -\sum_{X \in g_X} f(X)\delta_h(x - X)A(X) \tag{2.45}$$

where $g_{\mathbf{X}}$ is the set of the actuator surface grid cells and A is the area of the actuator surface grid cell. The same discrete delta function as in equation (2.34) is employed. It is noted the negative sign is because $\mathbf{f}(\mathbf{x})$ represents the forces of the actuator surfaces acting on the flow while $\mathbf{f}(\mathbf{X})$ denotes the forces of the flow acting on the actuator surfaces.

2.4.4.2 Actuator surface model for Nacelle

Resolving the boundary layer over a nacelle of a utility scale wind/MHK turbine, which becomes almost impossible for wind/MHK farm scale simulations, requires a much finer mesh as compared with the mesh for resolving the thickness of an atmospheric boundary layer. To take into account the nacelle effects on a relatively coarse mesh, in this section we present an actuator surface model for parameterizing the nacelle effect without directly simulating the boundary layer flows over the nacelle.

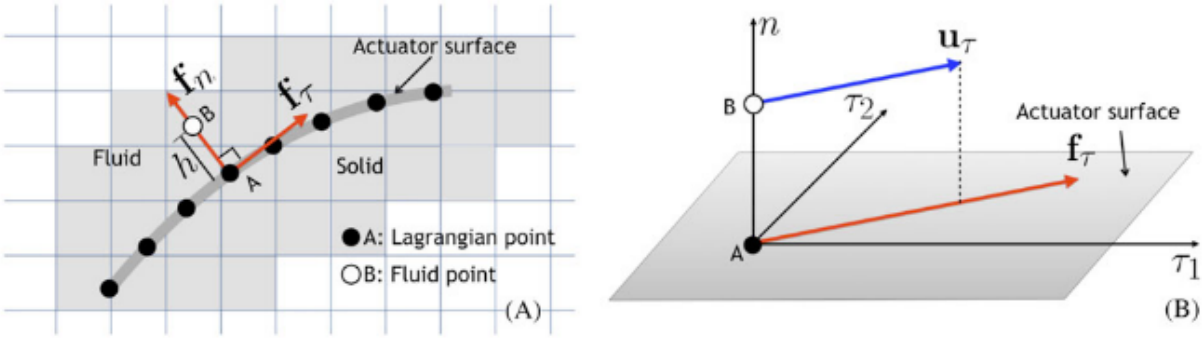


Figure 2.2: A schematic of the actuator surface model for nacelle. As shown in (a), the normal force f_n at the Lagrangian point A is computed by equation (2.46) with the estimated velocity $\tilde{u}(X)$ at point A interpolated from the surrounding Cartesian grid nodes using equation (2.34). The magnitude of the tangential force f_τ is computed by equation (2.48) using the incoming streamwise velocity U and a friction coefficient given by equation (2.49). The tangential force's direction is assumed to be the same as that of the tangential velocity at point B, which is interpolated from the surrounding Cartesian grid nodes using equation (2.34). The computed forces are distributed to the surrounding Cartesian grid nodes (the gray area shown in (a)) using equation (2.45).

In this model, the nacelle geometry is represented by the actual surface of the nacelle with distributed forces. As shown in Figure 2.2, the force acting on the actuator surface is decomposed into the normal component and the tangential component. From a physical point of view, the actuator surface of the nacelle may not act as a sink or a source of mass however the boundary layer over the nacelle is modeled. Based on this consideration, the normal component of the force on the actuator surface is computed by satisfying the non-penetration constraint. The non-penetration constraint can be met by directly reconstructing the wall-normal velocity near the actuator surface as in the sharp interface immersed boundary method. However, a physically reasonable interpolation scheme is difficult to

implement because of the very coarse grid employed in the actuator type simulations. Moreover, direct velocity reconstruction cannot ensure a smooth representation of the nacelle geometry because of the very coarse grid. In this work we represent the nacelle geometry as a diffused interface the same as in the direct forcing immersed boundary method. In this actuator surface model for nacelle, the normal component of the force acting on the surface per unit area is calculated in the same way as in the direct forcing immersed boundary method, which is expressed as follows:

$$\boldsymbol{f_n}(\boldsymbol{X}) = \frac{h\left(-\boldsymbol{u}^d(\boldsymbol{X}) + \tilde{\boldsymbol{u}}(\boldsymbol{X})\right) \cdot e_n(\boldsymbol{X})}{\Delta t} e_n(\boldsymbol{X}) \tag{2.46}$$

where $\mathbf{u^d(X)}$ is the desired velocity on the nacelle surface, $\mathbf{e_n(X)}$ is the unit vector in the normal direction of the nacelle surface, $h = \left(h_x h_y h_z\right)^{1/3}$ is the length scale of the local background grid spacing (different values of h have been tested without noticing any significant differences), $\tilde{\mathbf{u}}\mathbf{(X)}$ is the estimated velocity on the actuator surface, which is interpolated from the corresponding estimated velocity on the background mesh using equation (2.34) with the $\tilde{\mathbf{u}}\mathbf{(x)}$ on the background mesh computed as:

$$\tilde{\boldsymbol{u}}(\boldsymbol{x}) = \boldsymbol{u}^n(\mathbf{x}) + \boldsymbol{rhs}^{\mathrm{n}}(\mathbf{x})\boldsymbol{\Delta} t \tag{2.47}$$

where Δt is the size of the time step, the right-hand-side term $\mathbf{rhs^n}$ includes the convection, pressure gradient and diffusion terms computed from the quantities of previous time step n.

The tangential force acting on the surface depends on the incoming velocity, the surface geometry, and the complex near-wall turbulence. Neither of the last two can be directly captured in the actuator type simulations by using the no-slip boundary conditions or the shear stress boundary conditions for wall models [22, 23]. In the present actuator surface model, we assume that the tangential force is proportional to the local incoming velocity, and that the effects of surface geometry and near-wall turbulence can be parameterized using a single parameter, the friction coefficient c_f. Based on this assumption, the tangential force acting on the surface per unit area is computed as:

$$\boldsymbol{f_\tau}(\boldsymbol{X}) = \frac{1}{2} c_f U^2 \boldsymbol{e_\tau}(\boldsymbol{X}) \tag{2.48}$$

where c_f is calculated from the empirical relation proposed by F. Schultz-Grunow [24] for turbulent boundary layers with zero pressure gradient, which is expressed as follows:

$$c_f = 0.37(\log R\, e_x)^{-2.584} \tag{2.49}$$

where Re_x is the Reynolds number based on the incoming velocity and the distance from the upstream edge of the immersed body. It is noticed that this expression is invalid for the region with a large pressure gradient. Yet, for the present nacelle simulation, a zero-pressure gradient can be regarded as a reasonable assumption for most of the turbine nacelle. The framework presented in this work is applicable to more complex geometries if the corresponding distribution of c_f is available from experiments or high-fidelity

simulations. In equation (2.48), U is the reference velocity of the incoming flow. In the present work, it is selected as the magnitude of the incoming streamwise velocity. The direction of the tangential force $\mathbf{e}_\tau(\mathbf{X})$, on the other hand, is determined by the local tangential velocity relative to the velocity at the nacelle surface at points located at h away from the wall, i.e., point B in Figure 2.2, as follows:

$$e_\tau(X) = \frac{u\big(X + he_n(X)\big)}{\big|u\big(X + he_n(X)\big)\big|} \tag{2.50}$$

After computing the forces on the actuator surface of the nacelle, the forces on the background mesh are distributed from the actuator surface mesh in the same way as in equation (2.45). It is noted that the forces are distributed over the closest five cells defined by the width of the employed smoothed cosine discrete delta function. This force distribution width will be very small if we have a sufficiently high grid resolution to resolve the boundary layer. For the grids used in the actuator type simulations, however, this force distribution width is so large that it may cause non-physical velocity field near the nacelle, affecting the calculation of the blade's relative incoming velocity. To remedy this problem, we simply let the relative incoming velocity ($\mathbf{V_{rel}}$ in equation (2.31)) of the two closest radial locations to the nacelle surface be equal to that of the third closest location away from the nacelle.

2.5 Large-Eddy Simulation

The description of the Large-eddy simulation (LES) model implemented in the code is extensively described in Kang et al. [25]. The sub-grid stress term in the right-hand side of the momentum equation resulting from the the filtering operation is modeled with the Smagorinsky SGS model of [26] which reads as follows:

$$\tau_{ij} - \frac{1}{3}\tau_{kk}\delta_{ij} = -2\mu_t\overline{S}_{ij} \tag{2.51}$$

where μ_t is the eddy viscosity, the overline indicates the grid filtering operation, and $\overline{S_{ij}}$ is the large-scale strain-rate tensor. The eddy viscosity can be written as

$$\mu_t = C_s\Delta^2\big|\overline{S}\big| \tag{2.52}$$

where Δ is the filter width taken from the box filter, $\big|\overline{S}\big| = \left(2\overline{S}_{ij}\overline{S}_{ij}\right)^{1/2}$ is the magnitude of strain-rate tensor, and C_s is the Smagorinsky constant computed dynamically with the Smagorinsky model of Germano et al. [27].

2.6 Wave generation

2.6.1 Linear wave theory

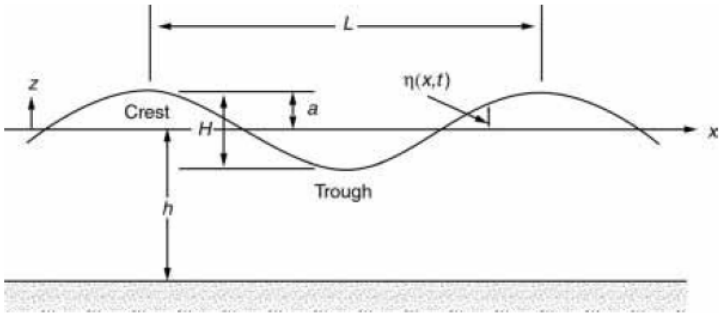


Figure 2.3: Schematic of a water wave in linear wave theory [40]

The equation governing the displacement of the water surface η(x, t) from the mean water level is:

$$\eta(x,t) = \frac{H}{2} cosk(x - Ct) = \frac{H}{2} cosk(kx - \sigma t) \tag{2.53}$$

Because the wave motion is assumed to be periodic (repeating identically every wavelength) in the wave direction (+x), the factor k, the wave number, is used to ensure that the cosine (a periodic mathematical function) repeats itself over a distance L, the wavelength. This forces the definition of k to be $k = 2\pi/L$.

For periodicity in time, which requires that the wave repeat itself every T seconds, we have $\sigma = 2\pi/T$, where T denotes the wave period. We refer to σ as the angular frequency of the waves. Finally, C is the speed at which the wave form travels, $C = L/T = \sigma/k$. The wavelength and period of the wave are related to the water depth by the dispersion relationship:

$$\sigma^2 = gktanhkh \tag{2.54}$$

Under the progressive wave, Eq. (2.53), the water particles move in elliptical orbits, which can be decomposed into the horizontal and vertical velocity components u and w as follows:

$$u(x,z,t) = \frac{H\sigma}{2} \frac{cosh(h+z)}{sinhkh} cos(kx - \sigma t) \tag{2.55}$$

$$w(x,z,t) = \frac{H\sigma}{2} \frac{sinh(h+z)}{sinhkh} sin(kx - \sigma t) \tag{2.56}$$

The code imposes incident solitary or linear waves by prescribing the associated free-surface elevation and velocity components as inlet boundary conditions based on the solitary wave equations or linear wave theory [28]. Herein, either the Boussinesq or the third-order Grimshaw equations are utilized to prescribe the wave characteristics at the inlet section. Each of these equations, the Boussinesq and the third-order Grimshaw equations, is valid for a certain range of normalized wave heights, i.e., $\varepsilon = H/h$, where H is the wave height and h is the still water depth. More specifically, if $\varepsilon \leq 0.25$, the Boussinesq equation is applied; while for $0.25 < \varepsilon < 0.5$, the third order Grimshaw equation is applied.

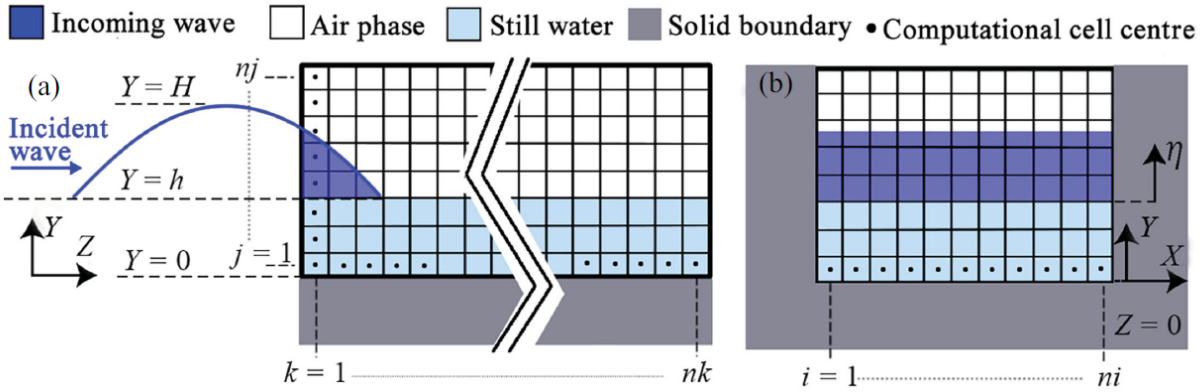


Figure 2.4: Schematic of an incident solitary wave entering the computational domain from inlet cross-section. (a) and (b) illustrate the longitudinal and cross-sectional views of the channel, respectively. h and H are the still water depth and wave height, respectively. The incident wave travels from left to right [28].

Using the third order Grimshaw equation, the free-surface elevation (η) and the velocity components in spanwise (X), vertical (Y), and streamwise (Z) directions are:

$$s = \operatorname{sech}\left(\frac{\alpha z}{h}\right) \tag{2.57}$$

$$q = tanh\left(\frac{\alpha z}{h}\right) \tag{2.58}$$

$$z = Z - Ct \tag{2.59}$$

$$\alpha = \frac{\sqrt{3\epsilon}}{2}\left(1 - \frac{5}{8}\epsilon + \frac{71\epsilon^2}{128}\right) \tag{2.60}$$

$$C = \sqrt{gh}\left(1 + \epsilon - \frac{\epsilon^2}{20} - \frac{3\epsilon^3}{70}\right) \tag{2.61}$$

$$\eta(Z,t) = \left[\epsilon s^2 - \frac{3}{4(\epsilon s q)^2} + \epsilon^3\left(\frac{5}{8(sq)^2} - \frac{101}{58}s^4q^2\right)\right] \tag{2.62}$$

$$\frac{w}{\sqrt{gh}} = \epsilon s^2 - \epsilon^2\left[-\frac{s^2}{4} + s^4 + \left(\frac{Y}{h}\right)^2\left(\frac{3s^2}{2} - \frac{9s^4}{4}\right)\right] +$$

$$\epsilon^3\left[\frac{19s^2}{40} + \frac{s^4}{5} - \frac{6s^6}{5} + \left(\frac{Y}{h}\right)^2\left(-\frac{3s^2}{8} - \frac{15s^4}{16} + \frac{15s^6}{2}\right) + \left(\frac{Y}{h}\right)^2\left(-\frac{3s^2}{8} + \frac{45s^4}{16} - \frac{45s^6}{16}\right)\right] \tag{2.63}$$

$$\frac{V}{\sqrt{gh}} = \frac{Yq\sqrt{3\epsilon}}{h}\left\{-\epsilon s^2 + \epsilon^2\left[\frac{3s^2}{8} + 2s^4 + \left(\frac{Y}{h}\right)^2\left(\frac{s^2}{2} - \frac{3s^4}{2}\right)\right] +\right.$$

$$\left.\epsilon^3\left[\frac{49s^2}{640} - \frac{17s^4}{29} - \frac{18s^6}{5} + \left(\frac{Y}{h}\right)^2\left(-\frac{13s^2}{6} - \frac{25s^4}{16} + \frac{15s^6}{2}\right) + \left(\frac{Y}{h}\right)^4\left(-\frac{3s^2}{8} + \frac{9s^4}{8} + \frac{27s^6}{16}\right)\right]\right\} \tag{2.64}$$

The Boussinesq equation for small amplitude solitary wave:

$$\eta(Z,t) = H[\operatorname{sech}\left(\sqrt{\frac{3}{4}\frac{H}{h}}\frac{z}{h}\right)]^2 \tag{2.65}$$

$$\frac{w}{\sqrt{gh}} = \left\{\epsilon + 3\epsilon^2 + \left[\frac{1}{6} - \frac{1}{2}\left(\frac{Y}{h}\right)^2\right]\right\}\frac{\eta}{H} - \epsilon^2\left[\frac{7}{4} - \frac{9}{4}\left(\frac{Y}{h}\right)^2\right]\left(\frac{\eta}{H}\right)^2 \tag{2.66}$$

$$\frac{v}{\sqrt{gh}} = \frac{Y\eta\sqrt{3\epsilon}}{h^2}\tanh\left(\sqrt{\frac{3\epsilon}{4}}\frac{z}{h}\right) \times \left\{1 + \frac{1}{2}\epsilon\left[1 - \frac{7\eta}{H} - \left(\frac{Y}{h}\right)^2\left(1 - \frac{3\eta}{H}\right)\right]\right\} \tag{2.67}$$

$$u = 0 \tag{2.68}$$

where C is the wave speed, u, v, and w are the velocity components in the spanwise (X), vertical (Y), and streamwise (Z) directions.

The sponge layer method for dissipating the waves at the boundaries reads as follows [29]:

$$S_i(x, y, t) = -[\mu C_0 u_i + \mu C_1 u_i |u_i|]\frac{exp\left[\left(\frac{x_s - x}{x_s}\right)^{n_s}\right]}{exp(1) - 1} \text{ for } (x_0 - x_s) \leq x \leq x_0 \tag{2.69}$$

where x_0 denotes the starting coordinate of the source region, x_s is the length of the source region, and C_0, C_1, and n_s are coefficients to be determined empirically.

Chapter 3

Getting Started

In this chapter, we describe how to install the libraries required for VFS-Geophysics, how to run a simulation case and how to post-process the results.

3.1 Installing PETSC and Required Libraries

VFS-Geophysics is implemented in C and is parallelized using the Message Passing Interface (MPI). The Portable, Extensible Toolkit for Scientific Computation (PETSC) libraries are used for the code organization and to facilitate its parallel implementation. Also, we use the library HYPRE for solving the Poisson equation. Before the code can be compiled, the following libraries must be properly installed:

- PETSC version 3.1-p8
- Blas and Lapack
- openmpi
- HYPRE

When installing the PETSC libraries there is the option to install the other required libraries automatically in the case that they are not already on the computer. The PETSC web page [30] (http://www.mcs.anl.gov/petsc/documentation/installation.html) gives a detailed description on how to install all of these libraries. We briefly outline the steps for installing PETSC in the command line of a Linux machine in the case that none of the aforementioned libraries have been previously installed:

1. Create a directory where to download the PETSC source files:
   mkdir source
2. Create the installation directory:
   mkdir system
3. Download the PETSC source code from the PETSC server in the source folder:
   wget http://ftp.mcs.anl.gov/pub/petsc/release-snapshots/petsc-3.1-p8.tar.gz
4. Unzip the PETSC source code in the source folder:
   tar xvfz petsc-3.1-p8.tar.gz
5. Execute the PETSC configuration script:
   ./config/configure.py --with-cc=mpicc --with-cxx=mpicxx --with-fc=mpif90 --download-f-blas-lapack=1 --download-f2f2cblaslapack=1 --download-hypre=1 --with-shared-libraries=0 --with-debugging=0 --with-x=0
6. If the previous action executes successfully, PETSC will print on the screen the subsequent steps.

Note that PETSC can be installed with the debugging option either active or inactive. It is recommended that PETSC be installed without debugging because it will be used in production mode.

The PETSC installation with debugging may be needed for developing purposes, but it compromises the speed of the code.

3.2 Compiling the Code

To compile the code and generate an executable file we include a file named "makefile". This file can work for any Linux-based computer without being modified. It basically links all the source code files (*.c, *.h) and the necessary libraries (PETSC, HYPRE, etc.). Given that on every computer the compiler libraries are located in different directories, the user has to designate the directory location as with the following user dependent information:

```
1  CC = mpixx
2  PETSC = /Your_Petsc_Library/
3  HYPRE = /Your_Hypre_Library/
4  TEC360HOME = /Your_Tecplot_Library/
5
6  Required_Library_Flags = -lpthread -lrt -ldl -lstdc++ -lgfortran -l:libpetsc.a -llapack -lblas
7  Source Code = /Source_Code_VFS-Geophysics/
```

If all libraries have been successfully installed and properly referenced with appropriate flags in the "makefile", the code should compile by typing the following command in the Linux shell:

make

Alternatively, one can add the option -j to increase the computation speed by taking advantage of the several processors as follows:

make -j #of Processors involving in the compilation

Either of the last two commands will generate the executable source file "**vfs-geophysics"**.

In addition to the executable file for running the code, it is also necessary to compile the executable file for post-processing the resulting output data. The post-processing file can generate result files in both Tecplot format (.plt).

Compiling the Post-Processing File

In addition, one needs to download the library file "libtecio.a" from the tecplot library webpage (http://www.tecplot.com/downloads/tecio-library/) and to add it to the code directory. The post-processing file named "data" should then be created with the following command:

make data

3.3 Running VFS-Geophysics

3.3.1 File Structure Overview

All the files that are necessary for running VFS-Geophysics should be in a user-defined folder. The required input files are the following:

grid.dat or xyz.dat The structured mesh for the fluid domain.

bcs.dat The option file for setting the BCs of the fluid boundaries.

vfs-geophysics Executable file obtained upon code compilation.

Submission script The Linux script for submitting the job in a Linux based cluster. If the executable file "vfs-geophysics" is in another folder, the user has to define the executable file path.

control.dat The file containing most of the control options.

ibmdata00, ibmdata01, etc. The mesh files for the immersed bodies, if any.

data The post-processing executable file

When the fluid-structure interaction is involved in the simulation, the following files should be added:

fsi-rot.dat allows users to define the position and angular velocity of the immersed bodies.

If the user would like to implement the actuator models, the following files are necessary:

acldataX file is an ASCII data file containing the mesh of the turbine model. When using the actuator line model, the file consists of n segments, where n is the number of rotor blades.

acddataX file is a UCD formatted triangular mesh of the actuator disk. The file is the same with the Urefdata file.

acsdataX file is an ASCII data file containing the mesh of the turbine model. When using the actuator surface model, the surface for the blades can be imported into the simulation.

Urefdata is a UCD formatted triangular mesh of disk. The purpose of this file is to compute the inflow reference velocity required for both the actuator models.

Turbine.inp is a text file containing input parameters used by the rotor model.

Nacelle.inp is a text file containing input parameters used by the rotor model with nacelle.

nacelle000_ is a UCD formatted triangular mesh of nacelle geometry.

Aerodym000_00, etc. These files contain the lift and drag coefficients at each profile along the turbine blades when using the actuator models.

FOIL000_00, FOIL000_01, etc. These files contain the angle of attack and chord length for each profile used along the turbine blades when using the actuator models.

When the actuator models are used, the following folder should be created:

ForceAOA Forces over the blade will be stored into this folder as files **ForceAOA_TimeStep_XX,** where XX is the index of each turbine.

The fluid grid file, the immersed bodies grid files, the boundary conditions file, the control file and the files containing fluid-structure information are described extensively in Section 4.2.

The remainder of this chapter describes the compiling process for obtaining the executable vfs-geophysics file, and the submission script for running the code.

3.3.2 Execute the Code

Each cluster may have different job resource manager systems although the most common is PBS. If it is not PBS, your system manager may provide instructions about the resource manager in use. There is ample documentation online as well.

In the case that your system uses the PBS system, the user can submit a job in the cue with the example script shown below:

```
1  #!/bin/bash
2  ### Job name
3  ### Mail to user
4  #PBS -k o
5  #PBS -l nodes=1:ppn=16
6  #PBS -l walltime=4:00:00
7  #PBS -j oe
8
9  cd $PBS_O_WORKDIR
10 mpirun –bind-to core /Your_Executable_File_Path/vfs-geophysics &> err
```

In the script above, the job will use 1 node of 16 cpus per node (ppn). The job maximum duration will be 4 hours (walltime). The name of the file executed is “vfs-geophysics”, and the on-screen information will be stored in the “err” file.

The command for submitting this script is

qsub script_name.sh

One can check the status of the job

showq

To finalize the job type

qdel job_id

3.3.3 Simulation Outputs

vfieldX.dat, ufieldX.dat, pfieldX.dat, nvfieldX.dat, lfieldX.dat, cs_X.dat

These are binary files containing the flow variables at the whole computational domain at a given time step indicated by "X". These files will be exported at every "tio" time step, where "tio" is a control option. The content in each of these files is summarized in Table 3.1.

Table 3. 1: Description of the instantaneous output results

File Name	Containing variable	Description of the Variable
vfieldX.dat	Ucont	Contravariant velocity components (fluxes)
ufieldX.dat	Ucat	Cartesian velocity components
PfieldX.dat	P	Pressure field
nvfieldX.dat	Nvert	If IB is used, it indicates the classification of nodes. 3 is the structure node, 1 is the IB node, and 0 is the fluid node.
lfieldX.dat	level	The distance function used in the level-set method to track the interface.
Cs_X.dat	Cs	The eddy viscosity coefficient when using LES.

Converge_dU

This text file contains the following information:

- The first number displayed in each of the lines is the time step number.
- Second column is the algorithm that the line refers to (momentum, poisson, IBMSEARCH0X)
  - Momentum: Momentum equation solver.
  - Poisson: Poisson solver for the second step of the fractional step method.
  - IBMSEARCH0X: Refers to the searching algorithm for node classification when at least one immersed body is present, and X is the immersed body's number.
- The third column is the computational time in seconds needed to complete the algorithm.
- The fourth column, if any, is the convergence of the corresponding solver. In the case of the Poisson solver the convergence is the maximum divergence and is indicated with "Maxdiv=".

Kinetic_Energy.dat

This file exports a text file with two columns. The first column displays the time step, and the second column the total kinetic energy of the whole computational domain calculated as follows.

$$KE = \sum_{I=0}^{N_I} \sum_{j=0}^{N_j} \sum_{k=0}^{N_k} u_{ijk}^2 + v_{ijk}^2 + w_{ijk}^2 \,, \tag{3.1}$$

Where N_i, N_j and N_k are the number of grid nodes in the I, j and k directions, respectively, and u_{ijk}, v_{ijk} and w_{ijk}, are the Cartesian velocity components computed at the cell centers. The function **KE_Output** is responsible for creating this file.

FSI_position00, FSI_position01, etc

This is a text file containing information about the immersed body location, velocity, and forces for the linear degrees of freedom in the x, y, and z direction. It consists of 10 columns as follows:

$$tio, Y_x, \dot{Y}_x, F_x, Y_y, \dot{Y}_y, F_y, Y_z, \dot{Y}_z, F_z \tag{3.2}$$

where tio is the output time step defined by the user, Y is the body position with respect to its initial position, $\dot{Y}$ is the body velocity, and F the force that the fluid imparts to the immersed body. If multiple immersed bodies are used, the code will print a file for each of the bodies, labeled with the body number.

FSI_Angle00, FSI_ Angle01, etc

This output file is similar to the FSI_position file, but for the rotational degrees of freedom. It contains information about the immersed body rotation angle and angular velocity of the body. It consists of 7 columns as follows:

$$tio, \Theta_x, \dot{\Theta}_x, M_x, \Theta_y, \dot{\Theta}_y, M_y, \Theta_z, \dot{\Theta}_z, M_z \tag{3.3}$$

where tio is the output time step, Θ is the body rotation with respect to its starting position and $\dot{\Theta}$ is the body angular velocity. If multiple immersed bodies are used, the code will print a file for each of the bodies, labeled with the body number.

Force_Coefficient_SI00, Force_Coefficient_SI01, etc.

This text file contains information about the forces that the fluid imparts to the immersed body in the three linear degrees of freedom, x, y, and z. The file has 10 columns with the following data:

$$ti, \mathrm{F}_x, \mathrm{F}_y, \mathrm{F}_z, C_{P_x}, C_{P_y}, C_{P_z}, \mathrm{A}_x, \mathrm{A}_y, \mathrm{A}_z \tag{3.4}$$

where ti is the time step number, F denotes the fluid force applied to the body, C_P is the normalized force coefficient, and Ap is the area of the body projected in the corresponding direction which has been used for computing C_P. Again, a different file is exported for each additional immersed body.

Momt_Coefficient_SI00, Momt_Coefficient_SI01, etc.

This file is equivalent to Force_Coeff_SI00 but in the rotational degrees of freedom. The information exported is the following:

$$ti, \mathrm{M}_x, \mathrm{M}_y, \mathrm{M}_z \tag{3.3}$$

where ti is the time step number and M denotes the moments applied to the structure.

Power_SI00, Power_SI01, etc.

This file can be used to obtain the power production over the immersed body. The following parameters are exported:

$$ti, \eta, \mathrm{P}_x, \mathrm{P}_y, \mathrm{P}_z, \mathrm{F}_z, \mathrm{A}_{Total} \tag{3.4}$$

where ti is the time step number, η is the efficiency, P denotes the power produced by the fluid over the immersed body and A_{Total} is the total area.

surfaceX_0.dat, surface_1.dat. etc.

These are tecplot ASCII files containing the surface of the corresponding IB body at the time step indicated at the first number of the file name. It basically contains the x, y, and z coordinates of the nodal points of the triangular mesh.

surface000_X.vtk, surface001_X.vtk, etc.

The surface of the corresponding IB body at the output time step can be recreated in the files which are available to read by Paraview with vtk extension.

DATA_FSIX_00.dat, DATA_FSIX_01.dat, etc.

At every "tiout" time step, it exports the immersed body motion information in the file. It is especially important while restarting the simulation.

3.3.4 Post-Processing

Once VFS-Geophysics reaches a time step at which data are output (multiple of "tio", or output time step), the output file can be post-processed. Post-processing consists of converting the output files (ufieldX.dat, vfieldX.dat, pfieldX.dat, nvfieldX.dat, lfieldX.dat, etc), which are in binary form, to a format which is readable for visualization software such as TecPlot360.

Create plt Files

To post-process the data, the executable "data" should be used with the following command:

$$mpirun\ /\text{Your\_Executable\_File\_Path}/data\ -tis\ 0\ -tie\ 50000\ -ts\ 100$$

where -tis is the start time step, -tie is the end time step and -ts is the time step interval at which the files were generated.

The above step will generate some number of result files for each timestep, compatible with tecplot and with names similar to ResultX.plt where X indicates the timestep. The .plt file can be opened in tecplot and worked upon.

The following webpage contains several tutorials in flash video on how to use tecplot: http://www.genias-graphics.de/cms/tp-360-tutorials.html.

Averaged results

By default, the plt files contain instantaneous data results even if the code has performed averaging of the results (-averaging set to 1, 2, or 3 at the time that the code is executed). To include the averaged results in the post-processed file, the option -avg needs to be activated when executing the file ``data". The option -avg can be set to 1, 2, and 3, depending on the amount of information to be included in the post-processed file, having an impact on the overall file size. If -avg is set to 1, the post-processed file

contains only averaged velocity and turbulence intensities (U, V, W, uu, vv, ww, wv, vw, uw); if it is set to 2, the post-processed file contains the same as the case for -avg 2 plus the averaged pressure and pressure fluctuations; and if it is set to 3, the file contains the same variables as in -avg 2 plus averaged vorticity. Note that for post-processing the data using options -avg 2 and 3, the code should be executed using the option -averaging with a value equal to or greater than the -avg value. An example on how to process averaged results is as follows:

$$mpirun\ /\text{Your\_Executable\_File\_Path}/data\ \ -tis\ 0\ -tie\ 50000\ -ts\ 100\ -avg\ 1$$

3.3.5 The Post-Processed File

Instantaneous Results

The variables of the post-processed results file are summarized in Table 3.2.

Table 3. 2: Description of the instantaneous output results in the postprocessed file

Variable	Description
X, Y, Z	Coordinates of the fluid grid
U, V, W	Velocity components at the grid cell centers
UU	Velocity magnitude
P	Pressure field
Nvert	If IB is used, it indicates the classification of nodes. 3 is the structure node, 1 is the IB node, and 0 is the fluid node.

Averaged results

The variables of the post-processed results file are summarized in Table 3.3.

Table 3. 3: Description of the time averaged output results in the post-processed file

Variable	Description
X, Y, Z	Coordinates of the fluid grid
U, V, W	Velocity components at the grid cell centers
uu, vv, ww, uv, uw, vw	Turbulence intensities
P	Pressure field
Nvert	If IB is used, it indicates the classification of nodes. 3 is the structure node, 1 is the IB node, and 0 is the fluid node.

Chapter 4

Code Input Parameters

4.1 Units

The Navier-Stokes equations are solved in the dimensionless form. In this case, it is recommended to set both the domain's reference length and the flow's reference velocity equal to one.

4.2 VFS code Input Files

The VFS code requires several input files that must be stored by default at the case's directory. The input files are the following:

control.dat
bcs.dat
grid.dat
Turbine.inp
Nacelle.inp (if nacelle model is in use)
ibmdata00, ibmdata01, ibmdata02, etc. (if IB method is in use)

4.2.1 The control.dat File

The file control.dat is a text file that is read by the code upon initiation. It contains most of the input variables for the different modules of the code. The order in which the different options are included in the control file is not relevant; however, it is recommended to group the options in the listed categories. The control options start with a dash "-" symbol. If for any case the same option is stated twice, the code takes the last value in the file. If the user wants to keep an option in the control file for later use, it can be commented by typing a "!" sign at the start of the line.

Time Related Options

dt (double)

Time step size for the solution of the Navier-Stokes equations. The CFL number must be smaller than 1. Values less than 0.5 are recommended.

tio (int)

The code exports the complete flow field data at each time step that is a multiple of this parameter's value.

totalsteps (int)

Total number of time steps to run before ending the simulation.

rstart (int)

This option is activated to restart a simulation. The value given to this parameter is the time step number at which the simulation is restarted. This option can only be used if a previous

simulation's results are in the same folder. Note that even if the value is set to zero, the simulation is restarted from a previous run; in that case, from a zero time step.

rstart_fsi (int 0, 1)

This parameter can be used when restarting a simulation with rstart active and when using the FSI module. If rstart_fsi is set to 1, the code reads the file FSI_DATA corresponding to time step rstart. This file contains information regarding body motion such as body position, velocity, forces, etc.

delete (int 0, 1)

If this option is set to 1, the code only keeps the result files from the two last exported time steps in the hard drive, deleting the files from previously exported time steps.

averaging (int 0, 1, 2, 3)

If this parameter is set to 1, 2, or 3, the code performs time averaging of the flow field. Time averaging is typically started when the flow field is fully developed; this occurs when the kinetic energy of the flow is stabilized. Therefore, averaging is a two-stage process. In the first stage, the simulation is started with the averaging option set to zero and advanced to a point at which the flow is developed. In the second stage, the flow results from stage 1 are used to restart the simulation and start the averaging. The first step to do in stage 2 is to rename the flow field files to be used from stage 1 (ufieldXXXXXX.dat, vfieldXXXXXX.dat, ...) to the file name corresponding to time step 0 files (ufield000000.dat, vfield000000.dat, ...). Then, the simulation can be restarted by activating the averaging option and setting the rstart option to 0 (restart from time step 0). The reason that the files must be renamed is because of the way in which the code performs averaging. The code uses the current time step for dividing the velocity sum and obtaining the averaged results. Thus, if averaging is started at a non-zero time step, the number of velocity summands will not correspond to the current time step number and the average will not be correct. As long as the averaging has been started at time step zero, restarting the simulation during stage 2 poses no problem. The non-zero values for this parameter refer to the amount of information that is averaged and exported to the results files. If average is set to 1, only averaged velocities (U, V, and W) and turbulence intensities (uu, vv, ww, uw, vw, uv) are processed. If the parameter is set to 2, in addition to the averaged velocities and turbulence intensities, the averaged pressure and pressure fluctuations are computed. Finally, if the parameter is set to 3, the processed variables include the variables from option 2 and the averaged vorticity vector. As already indicated in section 3.3.4, to post-process the results and include the averaged results to the output file, the option "avg" must be activated. The value given to "avg" should be lower or equal to the value given to the option "averaging".

Options for Boundary Conditions

inlet (int)

The inlet option defines the inflow profile type when the inlet plane is set to inflow mode (see bcs.dat description). It also sets the velocity initial conditions to the one corresponding to the inlet profile.

1: Uniform inflow with velocity value determined by the option flux.

13: This option is used for performing channel flow simulation. It sets the velocity initial condition to follow a log law. The domain height (channel half height) must be set to 1.

100: Imports inflow from external file. The external files must have a cross-plane grid geometry equivalent to the one of the current simulation grid.

perturb (int 0, 1)

If a non-zero initial condition is given, this option perturbs the initial velocity by adding random velocity values which are proportional to the local streamwise velocity component.

wallfunction (int)

Apply wall model at the walls of the immersed bodies. Basically, it interpolates the velocity at the IB nodes using a wallfunction.

ii_periodic, jj_periodic, kk_periodic (int 0,1)

Consider periodic boundary conditions in the corresponding direction. When this option is chosen in the control file, the corresponding boundaries in bcs.dat must be set to any non-defined value such as 100.

flux (double)

0: Sets the velocity at the inlet boundary to 1.

Non-zero value: Sets the flux at the inlet boundary. The flux is defined as the bulk inlet velocity divided by the area of the inlet cross-section.

-1: Sets the velocity at the inlet boundary as 1, which is non-dimensional

Modelling Options and Solver Parameters

les (int 0,1,2)

A nonzero value activates the LES model.

1: Smagorinsky - Lilly model.

2: Dynamic Smagorinsky model (recommended).

rans (int 1,2,3)

1: Using low Reynolds number version of Wilcox [38] $k-\omega$ model

2: Using high Reynolds number version of Wilcox [38] $k-\omega$ model

3: Using Menter [39] $k-\omega$ SST model.

imp (int 1,2,3,4)

Type of solver for the momentum equation. The only value supported is 4 which corresponds to the Implicit solver. Other values correspond to obsolete approaches and are not guaranteed to work.

imp_tol (double)

Tolerance for the momentum equation. A value less than or equal to 1.0e−5 is recommended.

poisson (int -1,0,1)

Selection of the Poisson solver. The only value supported is 1. Other values correspond to obsolete approaches and are not guaranteed to work.

poisson_it (int)

Maximum number of iterations for solving the Poisson equation. If the tolerance set by the option poisson_tol is reached, the Poisson solver is completed before reaching poisson_it iterations.

poisson_tol (double)

Tolerance for the flow's maximum divergence.

ren (double)

This parameter defines the Reynolds Number in the case of non-dimensional simulations.

inv (int 0,1)

1: Neglects the viscous terms in the RHS of the momentum equation, and thus the flow is considered inviscid.

Immersed Boundary Method Options

imm (int 0,1)

Activate the immersed boundary method. The code will expect a structural mesh (ibmdata00).

body (int)

If using the immersed boundary method, this parameter determines the number of bodies considered. There must be the same number of IB meshes (e.g., ibmdata00, ibmdata01,...,ibmdataXX, where XX is the number of bodies).

thin (int)

Option for simulating bodies with very sharp geometries where the resolution is not fine enough to resolve the depth.

x_c, y_c, z_c (double)

Initial translation of the immersed body position.

Fluid-Structure Interaction Options

fsi (int 0,1)

1: Activates the ability to move the structure in a single translational DoF. Select the desired DoF by setting one of the following options to 1: dgf_ax, dgf_ay, or dgf_az.

rfsi (int 0,1)

1: Activates the ability to move the structure in a single rotational DoF. Select the desired rotational DoF by setting rotdir.

rotdir (int 0,1,2,3)

When rfsi is active, the parameter selects the axis of rotation.

0: Rotation along the x axis.

1: Rotation along the y axis.

2: Rotation along the z axis.

3: Skewed in the XY plane.

dgf_ax, dgf_ay, dgf_az (int 0, 1)

In the case of a single translational DoF (fsi 1), the desired translational DoF is specified by setting one of these to 1.

When using fsi_6dof, multiple DoF can be activated.

dgf_x, dgf_y, dgf_z (int 0, 1)

In the case of a single rotational DoF (rfsi 1), the desired rotational DoF is specified by setting one of these to 1.

str (int 0, 1)

0: When either fsi or rfsi is active, the parameter uses the loose coupling algorithm.

1: When either fsi or rfsi is active, the parameter uses the strong coupling algorithm.

red_vel, damp, mu_s (double)

Parameters to be used in the test case "VIV of an elastically mounted rigid cylinder".

x_r, y_r, z_r (double)

Center of rotation in the rotational DoFs.

Turbine Parameterization Options

rotor_modeled (int 0, 1, 2, ..., 6)

Activate the turbine modeling option with the following parameterization approach:

1: Actuator disk model using the induction factor as input parameter.

2: Option for development purpose. This option is currently obsolete.

3: Actuator line model.

4: Actuator disk model using thrust coefficient as an input parameter.

5: Actuator surface model

6: Actuator line model with an additional actuator line for computing the reference velocity.

nacelle_model (int 0, 1, 2, ..., 5)

Activate the nacelle modeling option with the following parameterization approach:

1: The direct forcing immersed boundary method.

2: The normal force is calculated by the direct forcing immersed boundary method as equation (2.46) whereas the tangential force is zero.

3: The actuator surface model for nacelle. The tangential force is the desired shear.

4: The actuator surface model for nacelle. The friction factor is calculated locally using shear stress from the last time-step.

5: The actuator surface model for nacelle presented in section 2.4.4.2.

turbine (int)

Number of wind/ MHK turbines to be modeled.

reflength_wt (double)

The turbine's reference length. The code will divide the imported turbine diameter from the mesh file and Turbine.inp by this amount.

r_nacelle (double)

This parameter represents the turbine nacelle's radius. The code will ignore the rotor effect within this radius.

num_foiltype (int)

Number of foil types used along the turbine blade. VFS code requires a description file named FOIL00, FOIL01, ..., for each foil type as described in Section $4.2.5.

num_blade (int)

Number of blades in the turbine rotor.

refvel_wt, refvel_cfd (double)

These parameters do not have any effect in the simulation and are only for normalizing the output profiles. One can set them to 1 for normalization when plotting the data.

loc_refvel (int)

Distance upstream of the turbine in terms of rotor diameter, where the turbine's incoming velocity or reference velocity is computed.

deltafunc (int)

Type of smoothing function in which the pressure due to the rotor is applied [18].

0: A 2-point hat function

6: A 2-point linear function

7: A 2-point exponential function

8: A smoothed 4-point piecewise function

10: A smoothed 4-point cosine function

halfwidth_dfunc (double)

Half the distance for which the turbine effect is smoothed. The value is expressed in the number of grid nodes.

NumberOfNacelle (Int)

Number of nacelles in your simulation

NumNacellePerLoc (Int)

Number of nacelles in the same location

reflength_nacelle (double)

The nacelle's reference length. The code will divide the imported nacelle diameter from the mesh file and Nacelle.inp by this amount.

FixTipSpeedRatio (Int)

Fixed tip-speed ratio in rotor model.

dhi_fixed, dhj_fixed, dhk_fixed (double)

Width of the discrete delta function.

fixturbineangvel (Int)

Fixed angular velocity

rstart_turbinerotation (Int)

If activated, it reads the turbine rotation restart file. Restart turbine rotation (should restart change to one upon sims. restart)

Level Set Method Options

levelset (int 0,1)

Activates the levelset method. If used, the solved Navier-Stokes equations are in dimensional form.

dthick (double)

If using the level set method, this parameter defines half the thickness of the air/water interface. The fluid properties adopt their corresponding value in each phase and vary smoothly across this interface. Typical values adopted by “dthick" are on the order of 2 times the vertical grid spacing. Larger values may be necessary in extreme cases.

sloshing (int 0, 1, 2)

Sets the initial condition of the sloshing problem in a tank and exports to a file (sloshing.dat) the free surface elevation at the center of the tank.

1: Sets the initial condition for the 2D sloshing problem in a tank.

2: Sets the initial condition for the 3D sloshing problem in a tank.

0: Sloshing problem is not considered.

level_in (int 0, 1, 2)

Flat initial free surface with elevation defined by level in height.

1: The free surface normal is in the z direction.

2: The free surface normal is in the y direction.

level_in_height (double)

When the level in option is active this parameter determines the free surface vertical coordinate which is uniform.

fix_level (int 0,1)

1: The free surface is considered but kept fixed. In this case, the level set equation is not solved.

fix_outlet, fix_inlet (int 0,1)

When using inlet and outlet boundary conditions, activating any of these parameters will keep constant the free surface elevation at the corresponding boundary.

levelset_it (int)

Number of times to solve the reinitialization equation for mass conservation. A higher number may be useful in cases involving high curvature free surface patterns.

levelset_tau (double)

Parameter to define the pseudo-time step size used in the reinitialization equation. The pseudo time step is levelset tau times the minimum grid spacing.

rho0, rho1 (double)

Density of the water and the air respectively.

mu0, mu1 (double)

Dynamic viscosity of the water and the air respectively.

stension (int 0,1)

If active it considers the surface tension at the free surface interface.

inlet_y (double)

depth of the background grid at the inlet

outlet_y (double)

depth of the background grid the outlet.

Wave Generation Options

solitary_wave (Int 0,1,2)

Activates the wave generation at the inlet boundary condition.

0: wave generation is not activated.

1: solitary wave is generated at the inlet.

2: linear wave is generated at the inlet

ti_start_solitary_wave (Int)

Time that solitary wave starts propagating.

inlet_bed_elevation (double)

height of the bed elevation at the inlet. To calculate the water depth, this parameter should be subtracted from the inlet_y.

inlet_z_for_solitary_wave (double)

the z location of the solitary wave to start.

solitary_wave_amplitude (double)

solitary wave height (H).

ti_start_linear_wave_single (Int)

Time that linear wave starts propagating.

inlet_z_for_linear_wave_single (double)

the z location of the solitary wave to start.

linear_wave_single_amplitude (double)

Linear wave height (H).

linear_wave_single_number (double)

wave number (k).

wave_sponge_layer (int 0, 1, 2, 3, 4, 5)

0: Sponge layer is not used

1: Sponge layer method is only applied at the z boundaries.

2: Sponge layer method is applied at both z and x boundaries.

3: Sponge layer method is applied to water and air.

4: Sponge layer method is applied to the water.

5: Sponge layer method is applied at the water depth.

wave_sponge_xs (double)

Length of the sponge layer applied at the x boundaries.

wave_sponge_x01 (double)

Starting x coordinate of the first sponge layer applied at the x boundary.

wave_sponge_x02 (double)

Starting x coordinate of the second sponge layer applied at the x boundary.

wave_sponge_zs (double)

Length of the sponge layer applied at the z boundaries.

wave_sponge_z01 (double)

Starting z coordinate of the first sponge layer applied at the z boundary.

wave_sponge_z02 (double)

Starting z coordinate of the second sponge layer applied at the z boundary.

4.2.2 The bcs.dat File

The bcs.dat file is another text file with information about the boundary conditions of the fluid domain boundaries. We denote the six boundaries as Imin, Imax, Jmin, Jmax, Kmin, and Kmax, corresponding to the starting and ending boundary in the i-, j- and k- directions.
The format of the bcs.dat file is a single line with the 6 integers corresponding to each of the boundaries. This number can adopt the following values:

Table 4.1: Options for the bcs.dat file

Boundary condition type	Value
Slip Wall	10
No slip wall	1
No slip with wall modelling, smooth wall	-1
No slip with wall modelling, rough wall	-2
[1]Periodic boundary conditions	100
[1]Inlet	5 [1]
Outlet	4

The bcs.dat file has the following aspect: Imin-value, Imax-value, Jmin-value, Jmax-value, Kmin-value, Kmax-value.

Example. Simulation case with slip wall at the Imin, Imax, Jmin, Jmax boundaries, and inlet and outlet along the k direction: 10 10 10 10 5 4

[1] *Require additional information in the control file. Further details can be found in the corresponding section.*

4.2.3 The Grid File

The grid file (grid.dat) is formatted with the standard PLOT3D. The file can be imported in binary form or in ASCII form. For importing the grid in binary form, the option "binary" in the control.dat must be set to 1; otherwise, the code expects the ASCII form.

Any grid generator software that can export PLOT3D grids may be suitable for VFS code. However, Pointwise is recommended.

When creating the mesh, the user needs to pay attention to the orientation of two different sets of coordinate systems. The Cartesian components which are indicated in Figure 4.1 with x, y, and z, and the curvilinear components which are attached to the mesh and denoted as i, j, and k. Both coordinate systems should be right-hand oriented.

The recommended axis combination between Cartesian components and grid coordinates is depicted in Figure 4.1.

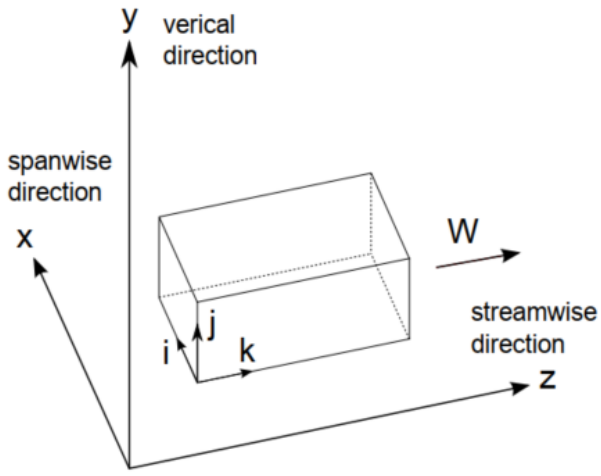


Figure 4.1: Axis orientation

A third format type that VFS code can handle is "SEGMENT". This format is suitable only for cases where the fluid mesh is Cartesian as the only information that the mesh file stores is the grid points of the three axes. An obvious advantage of this approach is that the mesh file size is much smaller than the "PLOT3D" formatted files.

To use "SEGMENT" format, the grid file should be named "xyz.dat" and the option "xyz" in the control file should be set to 1. In the first three lines, the "xyz.dat" file contains the number of grid nodes of the mesh for each of the three axes N_x, N_y, and N_z. The values are followed by the three coordinates of the points in the X axis, then the coordinates of the points in the Y axis, and finally the coordinates of the points in the Z axis as follows:

$$X_{x1}, Y_{x1}, Z_{x1}$$
$$X_{x2}, Y_{x2}, Z_{x2}$$
$$\dots$$
$$X_{xN_x}, Y_{xN_x}, Z_{xN_x}$$

$$X_{y1}, Y_{y1}, Z_{y1}$$
$$X_{y2}, Y_{y2}, Z_{y2}$$
$$\dots$$
$$X_{yN_y}, Y_{yN_y}, Z_{yN_y}$$
$$X_{z1}, Y_{z1}, Z_{z1}$$
$$X_{z2}, Y_{z2}, Z_{z2}$$
$$\dots$$
$$X_{zN_z}, Y_{zN_z}, Z_{zN_z}$$

4.2.4 The Immersed Boundary Grid File

The immersed boundary method allows one or more immersed bodies to be incorporated into the computational domain. If more than one body is considered, by default, each body has its own body mesh and its name is ibmdata00 for the first body, ibmdata01 for the second body, etc.

The body mesh is an unstructured surface mesh with triangular nodes. The format is the standard UCD. When generating an immersed boundary mesh, one needs to consider the following:

- The triangular's elements' normal direction must point towards the flow.
- In general, a triangular mesh with triangles of similar sizes as the fluid background mesh is recommended. If the immersed boundary is a flat wall, the triangular mesh may be coarser than the fluid mesh without loss of accuracy.

4.2.5 The Files Required for the Turbine Rotor Model

The Turbine.inp Control File

The Turbine.inp file is a text file that contains the input parameters used by the rotor model. The file comprises two major columns: the first column contains the numbers for the parameters, and the other column contains the corresponding parameter names. For example, in one line, the integer is given as 3 in the first column, and the parameter name is given as "num_blade" in the other column. Multiple parameters may be specified in one line for each column. For example, three double variables can be written in the first column with the corresponding translation parameters "x_c, y_c, and z_c" in the other column. For more information, a detailed example of the file is shown below:

```
----Turbine 1 ------
3 1                     num_blade num_foiltype
8.0 2.5 0.8             x_c y_c z_c
1.12087e5  0.25         J_rotation r_rotor
0 0.0425                r_nearhubinflowcorrection r_nacelle
0                       TPCntrl
0 -2.5                  indf_axis Tipspeedratio
0                       CT
0 0 0                   CP_max TSR_max angvel_fixed
0 0 0                   Torque_generator_max GeneratorSpeed_desired GearBoxRatio
0 0 0                   K_proportional K_integral K_derivative
0 0 0                   Ki_IPC angvel_axis_err_relax Kratio_torque
0 0 0                   WindSpeed_rate WindSpeed_cutin WindSpeed_cutout
0                       YawController
1.0 0.0 0.0                 Yaw1 Yaw2 Yaw3
0.0 0.0 0.0 1.0             Pitch1 Pitch2 Pitch3 Pitch_Min
```

num_blade (integer)

Number of blades.

num_foiltype (integer)

Number of airfoils description files (FOIL000_∗) for the blade.

nx_tb, ny_tb, nz_tb (double)

The turbine rotor plane's normal direction.

x_c, y_c, z_c (double)

The turbine rotor's initial translation.

indf_axis (double)

Induction factor for the actuator disk model (rotor model=1).

Tipspeedratio (double)

Tip speed ratio (TSR) for the actuator line model (rotor model=3,6). Negative TSR indicates that the turbine is rotating counterclockwise with respect to the streamwise axis.

J_rotation (double)

The rotor's moment of inertia. It is used only when the option "turbinetorquecontrol" is active.

r_rotor (double)

The turbine rotor's radius.

CP_max (double)

The turbine's maximum power coefficient. It is used only when the option "turbinetorquecontrol" is active.

TSR_max (double)

The turbine's maximum TSR of the turbine. It is used only when the option.

"turbinetorquecontrol" is active.

angvel_fixed (double)

The rotor's rotational speed when the option "fixturbineangvel" is active. When the variable angvel_fixed is a negative value, the turbine is rotating counterclockwise with respect to the streamwise axis.

Torque_generator (double)

Turbine torque. Used only when the option "turbinetorquecontrol" is active.

GeneratorSpeed_desired (double)

Rated generator angular velocity.

GearBoxRatio (double)

Gear box ratio from the aerodynamic shaft to the generator shaft.

K_proportional (double)

PID controller proportional term constant.

K_integral (double)

PID controller integral term constant.

K_derivative (double)

PID controller derivative term constant.

angvel_axis_err_relax (double)

Angular velocity error relaxation constant.

Kratio_torque (double)

Proportional constant of the torque's PID controller to the pitch control.

WindSpeed_rated (double)

The turbine's rated wind/water speed at which it produces its rated power.

WindSpeed_cutin (double)

Minimum wind/water speed at which the turbine produces power.

WindSpeed_cutout (double)

Maximum wind/water speed at which the turbine produces power.

CT (double)

Thrust coefficient used with rotor_modeled 4.

r_nearhubinflowcorrection (double)

$= r_{nacelle} + 2(grid\ spacing)$. If $r < rnearhubcorrct$, the relative incoming velocity at the near radius location will be employed instead of locally computed. This is to avoid the effect of the actuator surface nacelle model on computing the relative incoming velocity in the actuator surface model for turbine blades.

Pitch1, Pitch2, Pitch3 (double)

Optimum pitch angle (region 2).

Pitch_Min (double)

Minimal pitch angle for any of the pitches

The Nacelle.inp Control File

The Nacelle.inp file is a text file containing input parameters for the rotor model. The file has two lines. The first line is ignored by VFS code and only used for informative purposes by listing the input variable names. The second line is the control value corresponding to the variable listed in line 1 as shown in the example below:

```
nx_tb - ny_tb - nz_tb - x_c - y_c - z_c - angvel_axis - rotate_alongaxis - frictionfactor - dh …
1     0     0     8     2.5     0.8     0     0     0.4     0.02 …
xnacelle_upstreamend - ynacelle_upstreamend - znacelle_upstreamend - r_nacelle
7.4     2.5     0.0     0.0425
```

nx_tb, ny_tb, nz_tb (double)

The nacelle plane's normal direction. It is activated by setting the parameter from zero to one

x_c, y_c, z_c (double)

The nacelle's initial translation.

angvel_axis (double)

The blade's angular velocity.

rotate_alongaxis (int)

When activated, the nacelle is rotated about its axis.

frictionfactor (double)

Nacelle friction factor formula to parameterize the effects of surface geometry and near-wall turbulence from the empirical relation proposed by F. Schultz-Grunow [1]

dh (double)

The wall-normal thickness of the nacelle mesh.

xnacelle_upstreamend, ynacelle_upstreamend, znacelle_upstreamend (double)

The x- ,y- , and z-coordinates of nacelle's upstream end

r_nacelle (double)

Nacelle radius.

The acldata000 Mesh File

The acldata000 file is an ASCII data file that contains the turbine model's mesh. In the case of the actuator line model, the file consists of n segments where n is the number of rotor blades as shown in Figure 4.2(a). The ASCII data file uses the SEGMENT format.

In the case of the actuator disk model, the mesh is a UCD formatted unstructured triangular mesh, and the rotor is represented with a circle as shown in Figure 4.2(b). The actuator surface uses a file named acsdata000. The file is a triangular mesh in either Facet or UCD format and contains a two-dimensional drawing of the blades with the corresponding chord variation along the radial direction (Figure 4.2c).

The turbine center, *o*, of this mesh can be located directly at the actual position of the turbine or centered at the origin and later translated with the control options x_c, y_c, and z_c defined in the rotor model control file "turbine.inp".

In the actuator line model, the coordinate attached to the segment *i* must point towards the tip of the blade. In the actuator disk model, the direction normal to the rotor must point towards the direction of the flow.

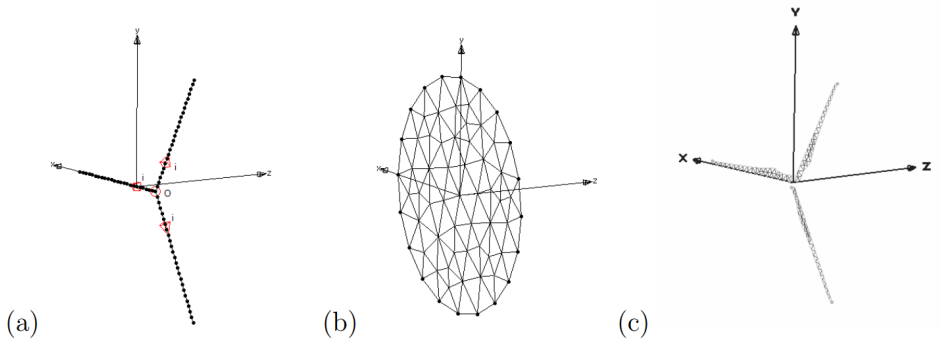


Figure 4.2: Representation of the "acldata000" mesh used to represent the blades in the (a) actuator line model, (b) actuator disk model and (c) actuator surface model.

The Urefdata Mesh File

The Urefdata file is a UCD formatted triangular mesh equivalent to the actuator disk acddata file. The purpose of this file is to compute the inflow reference velocity required for both the actuator disk and actuator line models.

The disk dimensions and normal direction in Urefdata must match the turbine rotor's dimensions defined in "turbine.inp".

The code places the disk upstream of the actual turbine location. At each time step, the flow velocity is transferred to each triangular element of this mesh. By adding the velocity at all triangular elements and dividing by the disk's surface, the code computes the turbine reference velocity (disk average velocity). For example, with the actuator line, the reference velocity is used for determining the simulation's TSR.

The Aerodym000_00, …, Aerodym000_XX Files

These files contain the lift coefficients, drag coefficients, and torque coefficients at each profile along the turbine blades for the actuator line (rotor_model = 3,6) and for the actuator surface model (rotor_model=5). The first two lines in the file are descriptive and the third line defines the number of data points in the file. Starting at line 4, the angle of attack (column 1) in degrees and the corresponding coefficients are listed as shown in the example below.

```
# Airfoil type: NACA0012  / NACA4412 HydroFoil From 0-20
#       Alpha(deg)     Cl       Cd       Cm
145
0       0.1853  0.03338 0
1       0.3271  0.03668 0
2       0.4248  0.04047 0
3       0.5199  0.04474 0
4       0.6076  0.04962 0
5       0.6932  0.05489 0
6       0.7625  0.06126 0
8       0.8612  0.0769  0
```

The FOIL00, FOIL01, FOIL02, ... Files

These files contain the angle of attack and chord length for each profile used along the turbine blades for the actuator line (rotor_model = 3,6) and for the actuator surface (rotor_model=5). The first two lines in the file are descriptive and the third line defines the number of data points in the file. Starting at line 4, the distance from the blade section to the turbine hub in non-dimensional units or in meters (column 1), the profile chord length in non-dimensional units or meters (column 2), and the blade section angle of attack in degrees (column 3) are listed as shown in the example below.

Cylinder

#	r(m)	chord (m)	twist (deg)
12			
0	0.18	19.9	
0.06	0.18	19.9	
0.07	0.1725	17.45	
0.09	0.1575	13.44	
0.11	0.1425	10.32	
0.13	0.1275	7.85	
0.15	0.1125	5.82	
0.17	0.105	4.12	
0.19	0.09	2.65	
0.21	0.0825	1.35	
0.23	0.075	0.21	
0.25	0.0675	0	

Chapter 5

Library Structure

5.1 The Source Code Files

The source code is structured in several files with extension ".c" and one file with extension ".h". The header file (variables.h) is included at the beginning of any other ".c" file and contains all the function prototypes, global variable definitions, and structure definitions. The ".c" files contain the subroutines which are generally grouped by code module. A brief description of the ".c" files is presented as follows:

main.c

Main code file where the code is initialized and finalized.

bcs.c

Subroutines for specifying boundary conditions.

compgeom.c, ibm.c, ibm io.c, variables.c

Subroutines for the IB method

fsi.c, fsi move.c

Subroutines for the FSI module

data.c

Subroutines for post-processing and visualizing the results.

wallfunction.c

Subroutines for the wall modeling

rotor_model.c

Contains all the subroutines that are necessary for simulating a wind/MHK turbine using the actuator disk, actuator line and actuator surface models.

Turbinecontrol.c

Contains subroutines for initializing wind/MHK turbine controller, calculating the rotational velocity of the turbine, etc.

les.c

Subroutines for the turbulent models using LES approach.

k-omega.c

Subroutines for the turbulent models using $k-\omega$ method within RANS framework.

solvers.c, implicitsolver.c, momentum.c, poisson.c, poisson hypre.c, rhs.c, rhs2.c, timeadvancing.c, timeadvancing1.c

Contains all the subroutines used by the flow solver, including the momentum and the Poisson equations.

init.c

Subroutines for initializing the code variables.

metrics.c

The subroutines for computing the grid Jacobian and metrics.

level.c, distance.c

Subroutines for simulating two-phase free surface flows with the levelset method.

wave.c

Subroutines for the wave module based on the inlet boundary condition.

5.2 The Flow Solver

To describe the basic elements of the flow solver we present a code flow chart of VFS-Geophysics, which displays the order in which the relevant functions of the code are called. This code flow chart corresponds to the simplest case that VFS-Geophysics can simulate, and no additional module is considered. An example would be to perform Direct Numerical Simulation of the channel flow case.
As in any "c" code, the so-called main function is the entry point or where the software starts the execution. In the code flow chart presented below, the functions, emphasized in bold, are indented such that the functions from a lower level are called by the function of the above level.

- **main** (pre-processing)
  In the first part of this main function, the code pre-processing is performed as follows:

  – MG_Initial
  Reads the structured grid file (grid.dat), partitions the domain within the CPUs, allocates memory for the main variables in a partitioned form. Also reads the boundary conditions file (bcs.dat).

  ∗ **FormMetrics**
  Computes the metrics and Jacobians of the transformation given by equation (2.2).

  – Calc_Inlet_Area
  Computes the inlet area corresponding to section k=0. The code was designed such that the streamwise direction is k.

  – SetInitialGuessToOne
  Sets the initial velocity of the whole computational domain at time=0 to a specific profile defined by the variable "inlet".

  – Contra2Cart
  Using the Cartesian velocity components at the cell centers, the contravariant velocity components at the cell faces are calculated through interpolation.

- **main** (time iteration)
  At this point of the main function, the time-stepping loop starts.

  – **Flow_Solver**
  This function solves for the velocity and pressure fields to advance to time step ti+1.

  ∗ **Calc_Minimum_dt**
  Calculates and prints the minimum time step size (dt) required such that the CFL number is equal to 1.

  ∗ **Pressure_Gradient**
  Reads the pressure field and computes the pressure gradient.

  ∗ **Formfunction_2**
  Forms the right-hand side of the momentum equation.

  ∗ **Implicit_MatrixFree**
  Solves the momentum equation.

  ∗ **PoissonSolver_Hypre**
  Solves the Poisson equation in the second step of the fractional step method to obtain the pressure correction.

  ∗ **UpdatePressure**
  The pressure correction is applied to obtain the pressure field.

  ∗ **Projection**
  Corrects the velocity to make it divergence free.

  ∗ **IB_BC**
  Sets most of the boundary conditions. Note, however, that other functions such as "Implicit MatrixFree" and "Contratocart" also deal with a part of the boundary conditions.

  ∗ **Divergence**
  Checks and prints the maximum divergence to the output file Converge_du.

  ∗ **Contra2Cart**
  Using the Cartesian velocity components at the cell centers, the contravariant velocity components at the cell faces are calculated through interpolation.

  ∗ **Calc_ShearStress**
  Computes and outputs the shear stress.

  ∗ **KE_Output**
  Exports the total kinetic energy of the whole computational domain to the file Kinetic Energy.dat.

∗ **Ucont_P_Binary_Output**

Writes the flow field results to files provided that the time step is a multiple of the control option “tiout”.

- End of time-stepping loop

  – **MG_Finalize**
  This function is called right before ending the code to di-allocate all the memory created during the execution of the code.

5.3 Code Modules

In the present section the main functions used by the different modules of the code are reviewed. All modules follow a common structural pattern. First, a group of functions are called for pre-processing purposes. Then, a second set of functions is called with the purpose of advancing the solution in time.

- **Subroutines for pre-processing.** Upon initiation of the program and before starting the time iteration, a set of functions is called to: (1) import the module specific input files (if any); and (2) initialize and allocate memory for the necessary variables. This process happens only once in the beginning of the main function located in the file “main.c”.
- **Subroutines for time advancing.** After the initial pre-processing part is completed, the code is ready to start advancing in time. Then a second set of functions is used to compute, at every time step, the necessary elements involved in the corresponding module. This part is generally executed from the function Flow Solver located in “solvers.c”.

5.3.1 The Large-Eddy Simulation (LES) Method Module

- **Subroutines for pre-processing.** In this module the pre-processing basically consists of initializing the LES main variables.

  – **MG_Initial** Initializes the LES model main variables.

- **Subroutines for time advancing.** Here the time-advancing involves computing the new eddy viscosity which is added to the diffusion term of the momentum equation.

  – **Compute_Smagorinsky_Constant_1** Computes the Smagorinsky constant C_s

  – **Compute eddy viscosity LES** Computes the eddy viscosity μt by applying equation (2.41).

  – **Formfunction_2** Adds the eddy viscosity term to the right-hand side of the momentum equation.

5.3.2 The Immersed Boundary (IB) Method Module

- **Subroutines for pre-processing.** In this module the pre-processing consists of initializing the IB method variables and importing the IB mesh.

  – **main** Initializes the primary variables for the IB method.

  – **ibm_read_ucd** Reads and imports the body triangular mesh (ibmdata00, ibmdata01, ...).

  – **ibm_search_advanced** Performs a classification of the fluid nodes depending on their position with respect to the structure. This classification is stored in the variable "nvert". If nvert is 0, the node belongs to the fluid domain and the equations are solved; if nvert is 3, the node belongs inside the structural domain and the node is blanked from the computational domain; if nvert is 1, the node is an IB node, which belongs in the fluid domain but is located at the immediate vicinity of the structure. IB nodes are where the velocity boundary condition of the body is specified.

  – **ibm_interpolation_advanced** Computes the velocity boundary conditions at the IB nodes. This computation can be done using linear interpolation or using a wall model.

    ∗ **noslip** Applies the no-slip-wall boundary condition using linear interpolation.

    ∗ **freeslip** Applies the slip-wall boundary condition using linear interpolation. Used when the inviscid option is active.

    ∗ **wall_function** Applies a wall model assuming a smooth wall. Used when the option wallfunction is active.

    ∗ **wall_function_roughness** Applies a wall model assuming a rough wall. Used when the wallfunction option is active and rough set is specified.

- **Subroutines for time advancing.** The time-advancing part depends on whether the body is moving or not. While the velocity boundary condition at the IB nodes has to be recomputed at every time step, the classifications of nodes have to be recomputed only if the body is moving.

  – **ibm_search_advanced** This function does not need to be called if the body is not moving. If the body is moving, this function needs to be called at every time step to update the node classification once the body position has been updated.

  – **ibm_interpolation_advanced** The velocity at the IB nodes has to be updated at every time step.

5.3.3 The Fluid-Structure Interaction (FSI) Algorithm Module

- **Subroutines for pre-processing.** In this module the pre-processing consists of initializing the FSI variables and applying an initial motion to the body.

  – **FsiInitialize** Initializes the variables for the body motion; either the motion is prescribed or determined using FSI.

  – **FSI_DATA_Input** Reads the external file "DATA FSIXXXXXX YY.dat". (XXXXXX refers to the time step and YY to the body number). This process is necessary when the simulation is restarted. The option rstart fsi needs to be active.

  – **Elmt_Move_FSI_TRANS** This function applies a linear translation to the body mesh in a single DoF. The function is called when the single translational DoF module is in use. In the pre-processing, the function is used to apply an initial translation to the body either when starting the simulation or when restarting.

  – **Elmt_Move_FSI_ROT** This function applies a rotation to the body mesh in a single DoF. The function is called when the single rotational DoF module is in use. In pre-processing, the function is used to apply an initial rotation to the body, either when starting the simulation or when restarting.

    ∗ **rotate_xyz6dof** This function applies a rotation to a given point with respect to a center of rotation in the three axial directions.

- **Subroutines for time advancing.** The time-advancing part depends on whether the body is moving or not. As already discussed for the IB method module, the velocity boundary condition at the IB nodes has to be recomputed at every time step, and the classifications of nodes has to be recomputed only if the body is moving.

  – **Struc_Solver** This function computes and updates the new position of the body. The function is called within the main function at every time step.

    ∗ **Calc_forces_SI** Computes the force and moments that the fluid imparts to the body.

    ∗ **Forced_Motion** Computes the position and velocity of the structure using the prescribed motion mode. Both the position and the velocity are specified through an analytic expression. Needs to be followed by a call to either the function **Elmt_Move_FSI_TRANS** or **Elmt_Move_FSI_ROT_TRANS**.

    ∗ **Calc_FSI_pos_SC** Solves the EoM in a single translational DoF. Needs to be followed by a call to **Elmt_Move_FSI_TRANS.**

∗ **Calc_FSI_Ang** Solves the EoM in a single rotational DoF. Needs to be followed by a call to **Elmt_Move_FSI_ROT**.

∗ **Forced_Rotation** Computes the rotation and angular velocity of the structure using the prescribed motion mode through an analytic expression. Needs to be followed by a call to **Elmt_Move_FSI_ROT**.

∗ Note that after the motion has been applied to the body mesh, the function **ibm_search_advanced** needs to be applied to update the fluid mesh node classification.

– **FSI_DATA_Output** At every "tiout" time step, it exports the body motion information in the file "DATA_FSIXXXXXX_YY.dat". (XXXXXX refers to the time step and YY to the body number).

5.3.4 The Rotor Turbine Modeling Module

5.3.4.1 Actuator Disk Model

The actuator disk model is activated by setting rotor modeled to 1 (the model input is the induction factor) or to 4 (the input is the thrust coefficient).

- **Subroutines for pre-processing.** In the turbine modeling module, the preprocessing subroutines import the turbine model input file, and initialize the corresponding variables, allocating memory if necessary. Again, this process happens only once in the beginning of the main function located in the file "main.c".

  – **main** Initializes variables and imports the turbine control file "Turbine.inp".

  ∗ **disk_read_ucd** Imports the disk mesh. The function is called first to import the actual turbine mesh, named acddata000, and then to import the disk mesh for the reference length, named Urefdata000.

  ∗ **Pre_process** This function searches the fluid cells that are at the vicinity of the disk mesh, and it is called every time step provided that the disk changes its position.

- **Subroutines for time advancing.** After the previous part is completed and the code starts advancing in time, the code computes the necessary elements involved in the turbine models at every time step, such as the interaction forces between the fluid and the turbine rotor or updates the new position of the rotor. These subroutines are called in the function Flow_Solver located in "solvers.c".

  – **Uref_ACL**

Calculates the reference velocity (U ref). This value corresponds to the space averaged velocity along a disk of the same diameter as the rotor and located some distance upstream of the turbine. The value is multiplied by the disk normal that points downstream.

– **Calc_U_lagr**
Interpolates the velocity from the fluid mesh to the Lagrangian points at the rotor model mesh.

– **Calc_F_lagr**
Computes the actuator line forces at each element of the Lagrangian mesh.

– **Calc_forces_rotor**
Computes the overall turbine forces.

– **Calc_F_eul**
Transfers the forces from the Lagrangian mesh to the fluid mesh.

5.3.4.2 Actuator Line Model

The actuator line model is activated by setting rotor modeled to 3 (the reference velocity is computed within a disk located upstream of the turbine) or to 6 (the reference velocity is computed within a line mesh instead of a disk).

- **Subroutines for pre-processing.** Equivalent to the actuator disk model with the difference that the turbine blades are represented with a one-dimensional mesh and the blade profile information is required.

  – **main** Initializes variables and imports the turbine control file “Turbine.inp”.

    ∗ **ACL_read_ucd** Imports the actuator line mesh file named “acldata000”.

    ∗ **disk_read_ucd** Imports the disk mesh file for computing the reference velocity named “Urefdata000”.

    ∗ **Pre_process** This function searches the fluid cells that are at the vicinity of the actuator line mesh or the reference velocity disk/line mesh. The function is called every time that the disk/line mesh changes its position.

    ∗ **airfoil_ACL** Imports the airfoil information.

    ∗ **TurbineTorqueControlInitialization** Initialize wind/MHK turbine controller.

- **Subroutines for time advancing.**

  – **Uref_ACL**
  Calculates the reference velocity (U ref) for the actuator line model. This value corresponds to the space averaged velocity along a disk of the same diameter as the rotor

and located some distance upstream of the turbine. The value is multiplied by the disk normal that points downstream.

– **Calc_turbineangvel**
Calculates the rotational velocity of the turbine based on the U ref velocity value.

– **rotor_Rot**
Applies a rotation to the turbine equivalent to the rotation velocity times the time step dt.

– **Pre_process**
Updates the new location of the turbine.

– **refAL_Rot**
Applies a constant rotation to the reference line located upstream of the turbine.

– **Export_ForceOnBlade**
Compute force at Lagrangian points using actuator line model.

– **Calc_U_lagr**
Interpolates the velocity from the fluid mesh to the Lagrangian points at the rotor model mesh.

– **Calc_F_lagr_ACL**
Computes the actuator line forces at each element of the Lagrangian mesh.

– **Calc_forces_ACL**
Computes the overall turbine forces.

– **Calc_F_eul**
Transfers the forces from the Lagrangian mesh to the fluid mesh.

5.3.4.3 Actuator Surface Model

The actuator surface model is activated by setting rotor_modeled to 5.

- **Subroutines for pre-processing.** Equivalent to the actuator line model with the difference that the turbine blades are represented with a two-dimensional mesh and the blade profile information is required.

  – **main** Initializes variables and imports the turbine control file “Turbine.inp”.

    ∗ **surface_read_xpatch** Imports the actuator surface mesh file named “acsdata000”.

    ∗ **ACL_read_ucd** Imports the actuator line mesh file named “acldata000”.

∗ **disk_read_ucd** Imports the disk mesh file for computing the reference velocity named “Urefdata000”.

∗ **Pre-process** This function searches the fluid cells that are at the vicinity of the actuator line mesh or the reference velocity disk/line/surface mesh. The function is called every time that the disk/line/surface mesh changes its position.

∗ **airfoil_ACL** Imports the airfoil information.

∗ **calc_s2l** Project surface elements to center line.

∗ **TurbineTorqueControlInitialization** Initialize wind/MHK turbine controller.

- **Subroutines for time advancing**

– **Uref_ACL**
Calculates the reference velocity (U_ref) for the actuator line model. This value corresponds to the space averaged velocity along a disk of the same diameter as the rotor and located some distance upstream of the turbine. The value is multiplied by the disk normal that points downstream.

– **Calc_turbineangvel**
Calculates the rotational velocity of the turbine based on the U_ref velocity value.

– **rotor_Rot**
Applies a rotation to the turbine equivalent to the rotation velocity times the time step dt.

– **bladepitch_Rot**
Computes blade pitch of the actuator surface.

– **Pre_process**
Updates the new location of the turbine.

– **Calc_U_lagr**
Interpolates the velocity from the fluid mesh to the Lagrangian points at the rotor model mesh.

– **Calc_F_lagr_ACL**
Computes the actuator line forces at each element of the Lagrangian mesh.

– **Export_ForceOnBlade**
Export force at Lagrangian points using actuator line model.

– **ForceProjection_l2s**
Project force from actuator line to surface.

– **Calc_forces_ACL**
Computes the overall turbine forces.

– **Calc_F_eul**
Transfers the forces from the Lagrangian mesh to the fluid mesh.

5.3.5 The Wave Generation Module

The incoming incident wave at the computational domain's inlet cross-section is implemented by prescribing the velocity components (u, v and w) and free surface elevation at the inlet.

- **Subroutines for time advancing.**

  - solitary_wave_inlet_velocity_profile_boussinesq
  The velocity components based on the Boussinesq or the third-order Grimshaw equations are calculated for each time at the inlet boundary.

  - solitary_wave_inlet_elevation_profile_boussinesq
  The free surface elevation based on the Boussinesq or the third-order Grimshaw equations is calculated for each time at the inlet boundary.

  - linear_wave_inlet_velocity_profile
  The velocity components based on the linear wave theory equations are calculated for each time at the inlet boundary.

  - linear_wave_inlet_elevation_profile
  The free surface elevation based on the linear wave theory equations is calculated for each time at the inlet boundary.

Chapter 6

Applications

These test cases are for introducing and simulating turbine parameterization which is used in the code. First, the actuator model cases for turbine parametrization are introduced, and then turbine resolving simulation is implemented. The test case involves the simulation of the Kinetic Hydro-Power System (KHPS) turbines that will be used to harvest kinetic energy from the channel developed by Verdant Power. These are 3-blade axial turbines 5m in diameter. Details can be found in [35]. We propose the case with uniform inflow which makes the case simple as it does not require any precursor simulation. The boundary conditions at the top wall, is free slip, and at the bottom wall and side walls is no-slip. The bottom wall cannot be resolved with the current grid resolution and is treated with a wall model.

The tip speed ratio (TSR) is considered as 2.5 for actuator models and angular velocity is considered as 1 rad/s for turbine resolving case. The grid is uniform and the spacing is 0.02 units in all three directions which is equivalent to a grid size of 257×85×881. In contrast to the fluid mesh, the actuator models mesh (acddata for actuator disk, acldata000 for actuator line and acsdata000 for actuator surface) can be constructed with the real turbine dimenions and non-dimensionalized by setting the turbine reference length option “reflength_wt" in control.dat. This will divide the actuator model mesh dimensions by “reflength_wt". Alternatively, one could generate a rotor mesh already with the non-dimensionalised units and choose “reflength_wt" equals to 1. The simulation is set with a unit non-dimensional velocity as well, so that we can use “refvel_wt” in order to make the velocity non-dimensionalized.

6.1 Actuator disk model for blades with IB nacelle

6.1.1 Case Definition

This test case is for introducing and simulating the actuator disk model for turbine parameterization. A computationally intensive approach for studying the fluid dynamics of wind and MHK farms is to solve numerically the full Navier-Stokes equations with turbines parameterized with actuator disk models [8]. In the actuator disk model, a turbine rotor is represented by a permeable circular disk that is discretized using an unstructured triangular grid, as can be seen in the figure below.

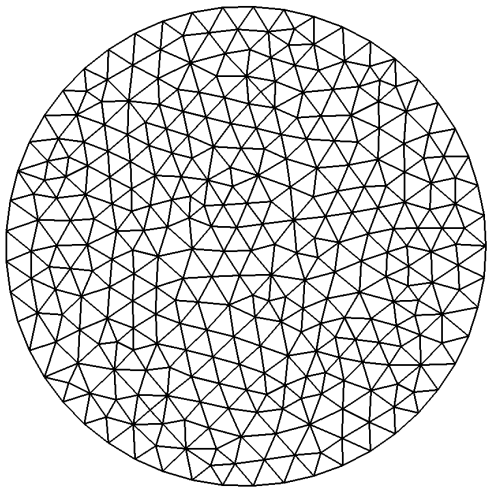


Figure 6.1: Schematic of the actuator disk used in the simulation

A UCD formatted unstructured triangular mesh for the disk with the name of "acddata" should be in the simulation directory along with "Urefdata", both of which are the same file but with different names. The tower and nacelle geometry are treated as sharp interface immersed boundaries, so that we can use CURVIB method, expressed in Ref. [1]. The tower and nacelle geometry can be seen in the figure below.

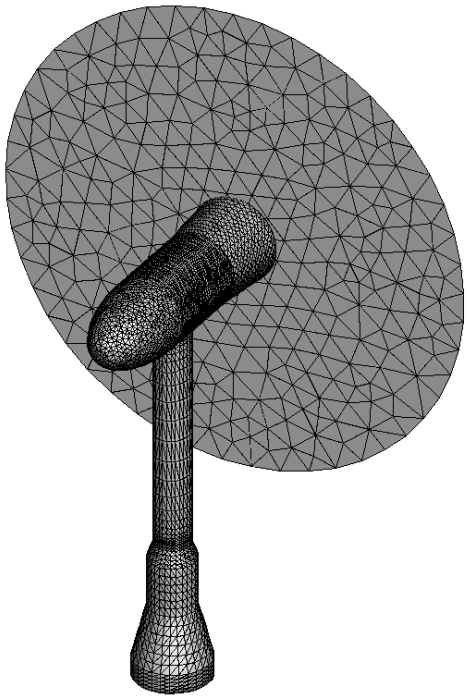


Figure 6.2: Schematic of the tower and nacelle with the actuator disk used in the simulation

6.1.2 Main Case Parameters

Since the main solver parameters have already been discussed in previous test cases [37], only the parameters related to the MKH turbine rotor model will be addressed.

The main parameters in the control file for setting this case are listed in Table 6.1. The parameters related to the MHK turbine rotor and nacelle model will be also addressed. When using the rotor model, the control options for setting the case are located not only in control.dat but also in Turbine.inp. The flow parameters are summarized in Table 6.1, with the rotor parameters in Table 6.2. The parameters in Turbine.inp that are not discussed in Table 6.2 are not used in the simulation.

Table 6.1 Parameters in control.dat file for the MHK turbine

Parameter	**Option in control file**	**Value**
Time step size	dt [s]	0.001
Activate the actuator disk model	rotor_modeled	1.0
Number of turbines	turbine	1.0
Reference length of the turbine	reflength_wt	0.5
Distance upstream of the turbine in rotor diameters where the turbine incoming velocity or reference velocity is computed	loc_refvel	1.0
Type of smoothing delta function	deltafunc	10
Delta function width in cell units	halfwidth_dfunc	4.0

Table 6.2 Parameters in Turbine.inp file for the MHK turbine

Parameter	**Option in Turbine.inp file**	**Value**
Number of blades in the rotor	num_blade	3
Number of foil types along the blade	num_foiltype	1
Normal direction of the turbine rotor plane.	nx_tb, ny_tb, nz_tb or Yaw1, Yaw2, Yaw3	1.0 0.0 0.0
The turbine rotor initial translation.	x_c, y_c, z_c	8.0 2.5 0.8
Tip speed ratio	Tipspeedratio	-2.5
Radius of the rotor corresponding to the acddata mesh	r_rotor	0.5

For this test case, time averaging was done so that it has to be executed in two stages as described in Section 6. In the first stage, the flow is fully developed; while in the second stage, the time averaging is performed. For developing the flow field, we performed 10000-time steps. For time averaging the flow field, 10000-time steps are enough, in order to make the flow fully developed based on the current

time step. For time averaging, set the option in the control file “averaging” to 3, “rstart” to 0, and rename the result files from the last instantaneous time step (ufield010000.dat, v_field010000.dat, pfield010000.dat, nvfield010000.dat, and cs_010000.dat) to the value corresponding to time zero (ufield000000.dat, vfield000000.dat, ...). Also, when restarting the flow field for averaging, set “rstart_turbinerotation” as 1 and rename TurbineTorqueControl010001_000.dat as TurbineTorqueControl000001_000.dat.

6.1.3 Results

Figure 6. and Figure 6. shows the contour plots of the instantaneous streamwise velocity in two different planes. As can be seen, actuator disk simulation can visualize the area which turbulence dominates in the flow field with approximately good resolution; but not that much in detail in comparison with turbine resolving method. This method is the least computationally expensive method in comparison with turbine resolving and can be a good choice when the details of the vortical structures and turbulence is not that much important.

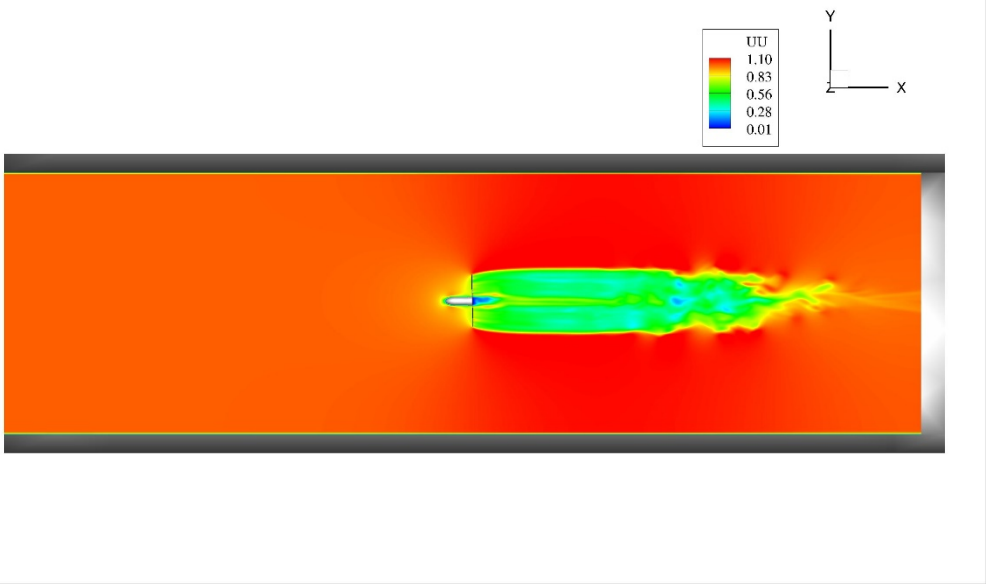


Figure 6.3: Contours of instantaneous streamwise velocity on the X-Y plane at the hub height

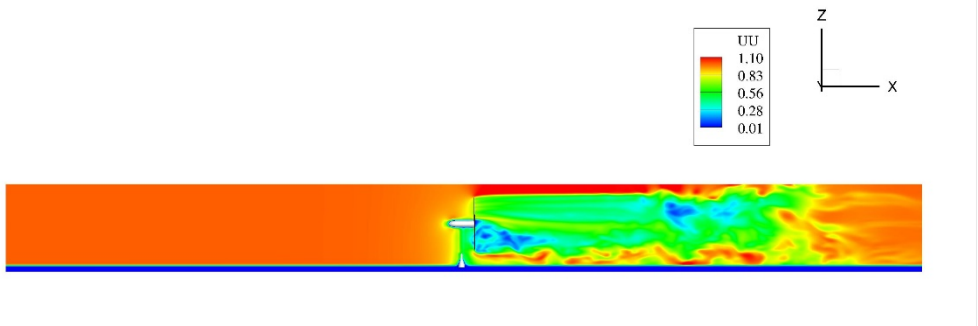


Figure 6.4: Contours of instantaneous streamwise velocity on the Z-X plane at the hub height

An animation of the instantaneous streamwise velocity on the X_Y plane at the hub height can also be found at this link.

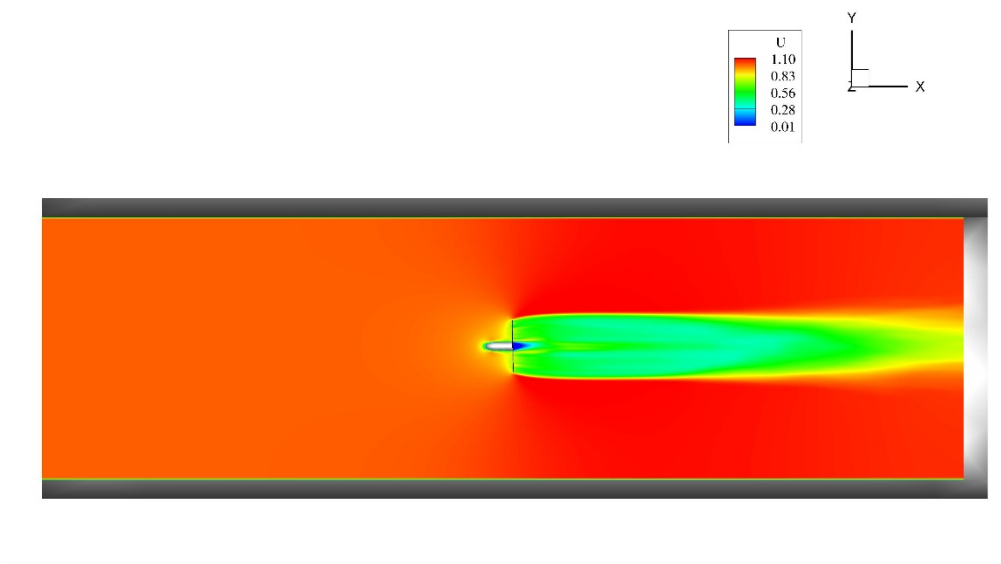


Figure 6.5: Time-averaged streamwise velocity on the Z-X plane at the hub height

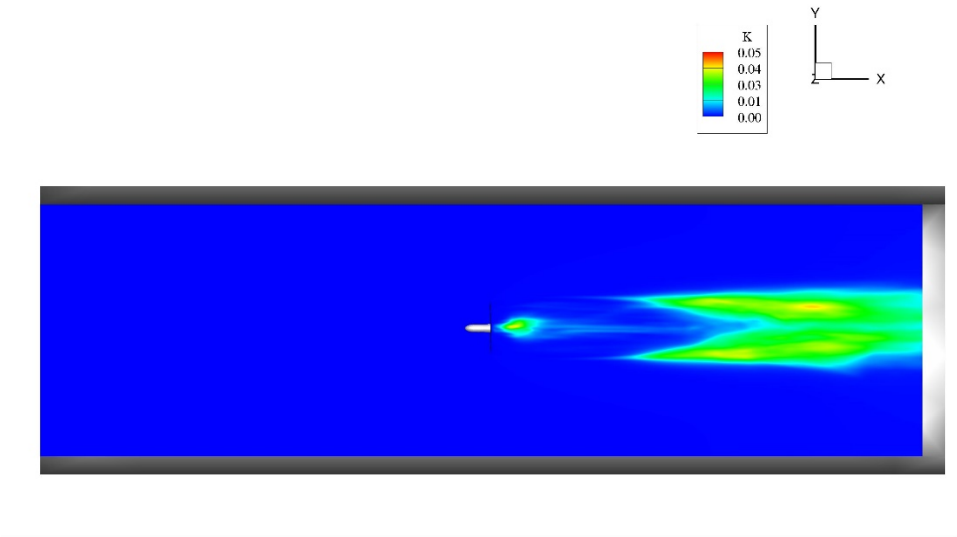


Figure 6.6: Turbulent Kinetic Energy on the Z-X plane at the hub height

6.2 Actuator line model for blades and actuator surface model for nacelle

6.2.1 Case Definition

This test case is for introducing and simulating the actuator line model for turbine parameterization. A computationally intensive approach for studying the fluid dynamics of wind and MHK farms is to solve numerically the full Navier-Stokes equations with turbines parameterized with actuator line models [8].

In the actuator line approach, the turbine blade is represented by a rotating line with distributed forces, which are calculated from a blade element approach combined with tabulated 2D airfoil drag and lift coefficients [17]. The nacelle geometry is represented by the actual surface of the nacelle with distributed forces, as can be seen in the figure bellow.

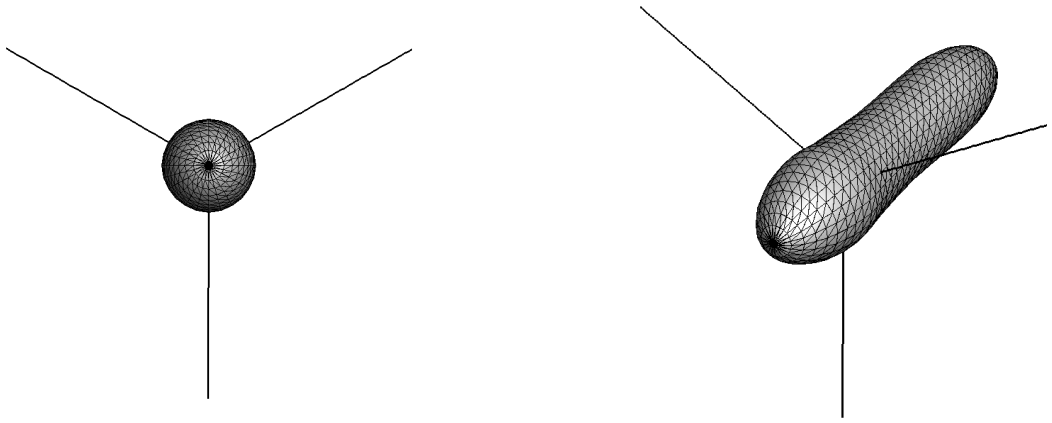


Figure 6.7: Schematic of the actuator line and nacelle used in the simulation

An ASCII data file that contains the turbine model's mesh with the name of "acldata000" should be in the simulation directory, along with the "Urefdata" which is a UCD formatted unstructured triangular mesh. A UCD formatted unstructured triangular mesh for the nacelle with the name of "nacelle000_" should also be in the simulation directory. The forces on each element are calculated based on the local inflow velocity and the drag and lift coefficients of the 2D airfoil. The rotor blades. cross section is approximated as a NACA0012 So, for airfoil data, we should have text files including "Aerodym000_00" for lift and drag coefficients and "FOIL000_00" for cord length of the airfoil. These text files should be inside the project directory.

6.2.2 Main Case Parameters

The main parameters in the control file for setting this case are listed in Table 6.3. The parameters related to the MHK turbine rotor and nacelle model will be also addressed. When using the rotor model, the control options for setting the case are located not only in control.dat but also in Turbine.inp and Nacelle.inp. The flow parameters are summarized in Table 6.3, with the rotor parameters in Table 6.4 and nacelle parameters in Table 6.5. The parameters in Turbine.inp and Nacelle.inp that are not discussed in the Table 6.4 and Table 6.5 are not used in the simulation.

Table 6.3 Parameters in control.dat file for the MHK turbine

Parameter	Option in control file	Value
Time step size	dt [s]	0.001
Activate the actuator disk model	rotor_modeled	6.0

Number of turbines	turbine	1.0
Reference length of the turbine	reflength_wt	0.5
Reference velocity of the case. It is only used for dimensionalizing the turbine model output files	refvel_wt	1.0
Distance upstream of the turbine in rotor diameters where the turbine incoming velocity or reference velocity is computed	loc_refvel	1.0
Shen tip loss correction model	Shen_AL	1.0
Type of smoothing delta function	deltafunc	10
Delta function width in cell units	halfwidth_dfunc	4.0
Activate the actuator surface model for nacelle	nacelle_model	5.0
Number of nacelles in the simulation	NumberOfNacelle	1.0
Number of nacelles in the same location	NumNacellePerLoc	1.0
Reference length of the nacelle	reflength_nacelle	0.5

Table 6.4 Parameters in Turbine.inp file for the MHK turbine

Parameter	**Option in Turbine.inp file**	**Value**
Number of blades in the rotor	num_blade	3
Number of foil types along the blade	num_foiltype	1
Normal direction of the turbine rotor plane.	nx_tb, ny_tb, nz_tb or Yaw1, Yaw2, Yaw3	1.0 0.0 0.0
The turbine rotor initial translation.	x_c, y_c, z_c	8.0 2.5 0.8
Tip speed ratio	Tipspeedratio	-2.5
Radius of the rotor corresponding to the acddata mesh	r_rotor	0.5

Table 6.5 Parameters in Nacelle.inp file for the MHK turbine

Parameter	**Option in Turbine.inp file**	**Value**
Normal direction of the turbine nacelle plane	nx_tb, ny_tb, nz_tb	1.0 0.0 0.0
The turbine nacelle initial translation	x_c, y_c, z_c	8.0 2.5 0.8
friction factor which is used in the nacelle model calculation	frictionfactor	0.4
Radius of the nacelle corresponding to the nacelle000_ mesh	r_nacelle	0.0425
length scale of the local background grid spacing	dh	0.02

The upstream end x-coordinates of nacelle for friction factor calculation	xnacelle_upstreamend	2.5
The upstream end y-coordinates of nacelle for friction factor calculation	ynacelle_upstreamend	0.0

For this test case, time averaging was done, so that it has to be executed in two stages as described in Section 6. In the first stage the flow is fully developed, while in the second stage the time averaging is performed. For developing the flow field, we performed 10000-time steps. For time averaging the flow field, 10000-time steps are enough, in order to make the flow fully developed based on the current time step. For time averaging, set the option in the control file "averaging" to 3, "rstart" to 0, and rename the result files from the last instantaneous time step (ufield010000.dat, v_field010000.dat, pfield010000.dat, nvfield010000.dat, and cs _010000.dat) to the value corresponding to time zero (ufield000000.dat, vfield000000.dat, ...). Also, when restarting the flow field for averaging, set "rstart_turbinerotation" as 1 and rename 010001_ 000.dat as TurbineTorqueControl000001_000.dat.

6.2.3 Results

Figure 6. and Figure 6. shows the contour plots of the instantaneous streamwise velocity in two different planes. As can be seen, actuator line simulation can visualize the vortical structures and turbulence in the flow field with a good resolution in comparison with turbine resolving method. Also, a wake region can be found in these contours which is due to the effect of nacelle. This method is not as computationally expensive as turbine resolving and can be a good choice in case of tidal farms.

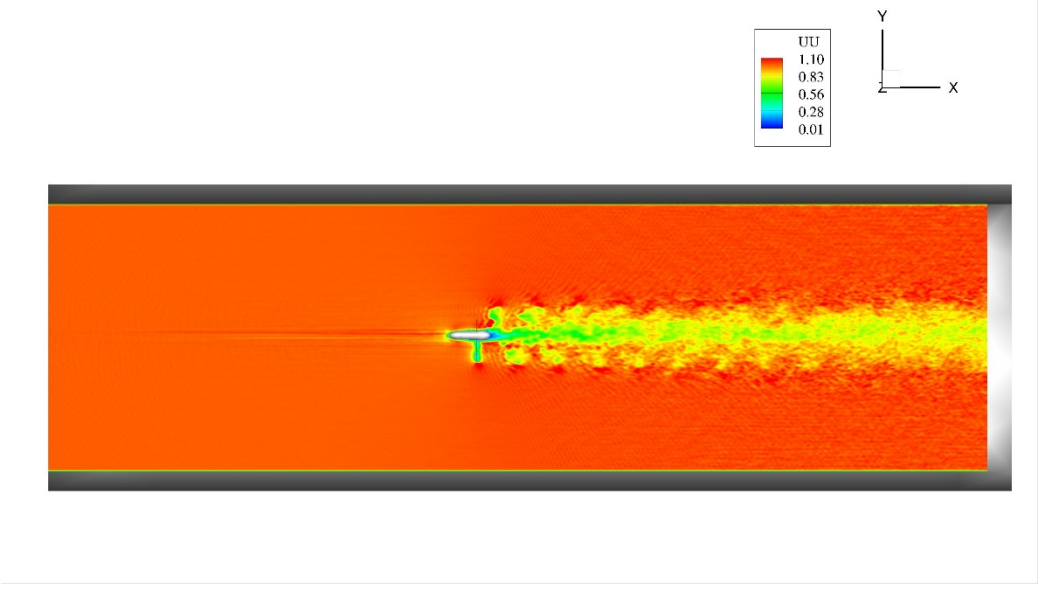


Figure 6.8: Contours of instantaneous streamwise velocity on the X-Y plane at the hub height

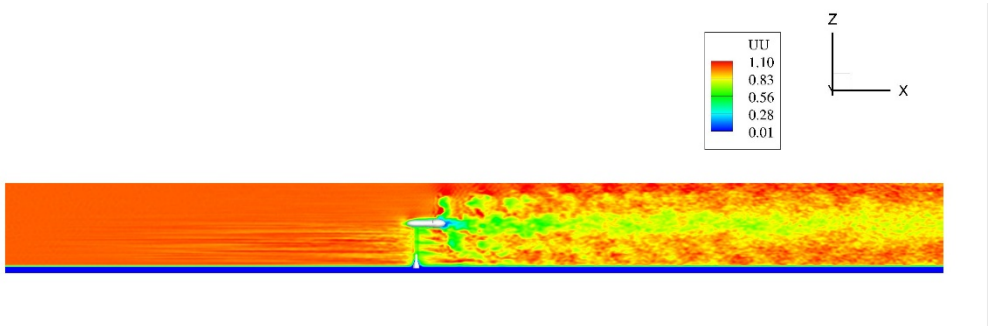


Figure 6.9: Contours of instantaneous streamwise velocity on the Z-X plane at the hub height

An animation of the instantaneous streamwise velocity on the X_Y plane at the hub height can also be found at this link.

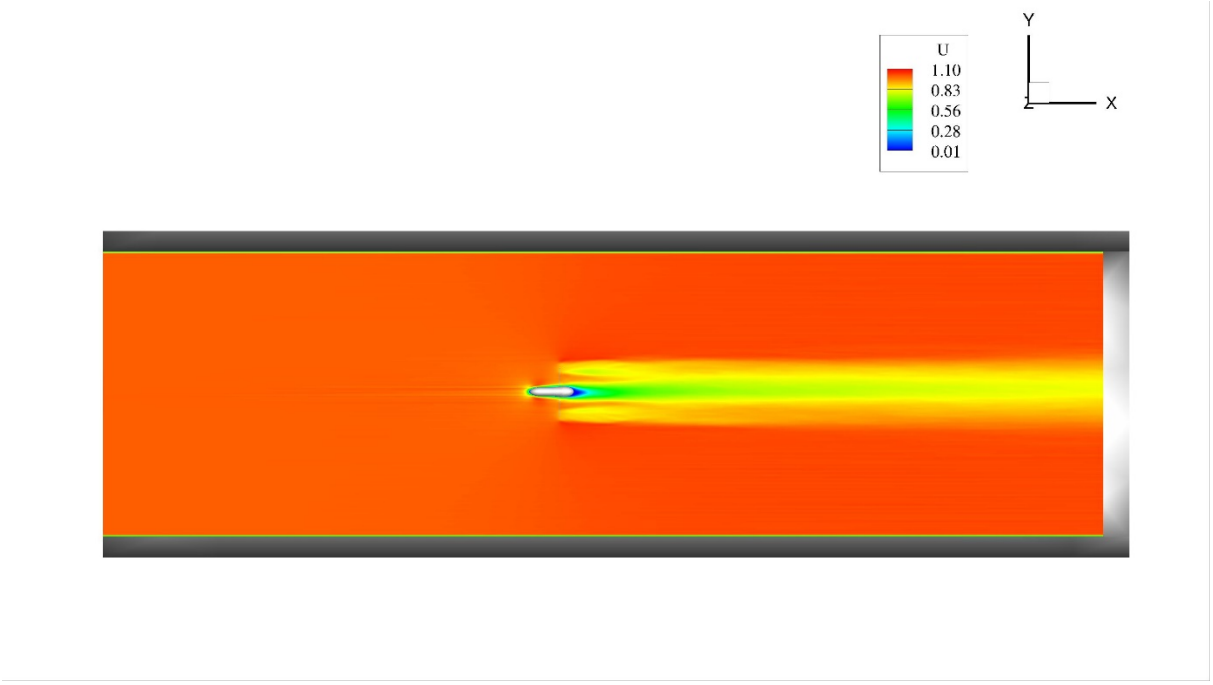


Figure 6.10: Time-averaged streamwise velocity on the Z-X plane at the hub height

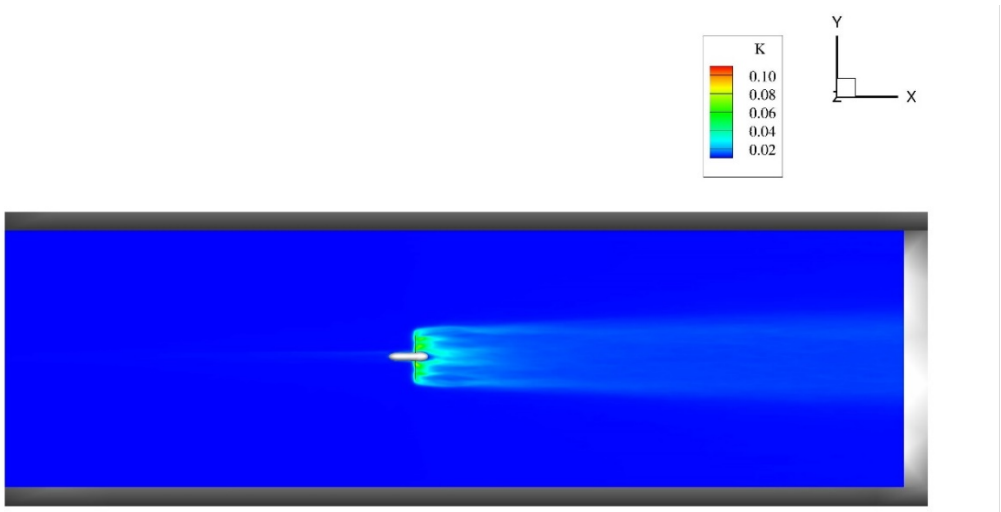


Figure 6.11: Turbulent Kinetic Energy on the Z-X plane at the hub height

6.3 Actuator surface model for both blades and nacelle

6.3.1 Case Definition

This test case is for introducing and simulating the actuator surface model for turbine parameterization. MHK turbine blades and nacelle were modeled using the actuator surface model. The actuator surface models the turbine blade by computing the lift and drag forces using the blade element theory [36]. The geometries represented by the actuator surface of the blades and nacelle can be seen in the figure below.

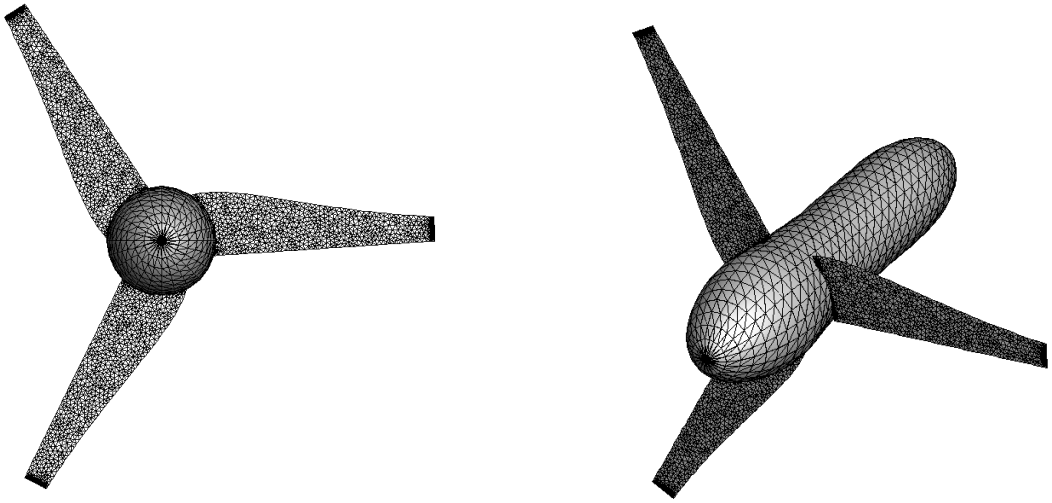


Figure 6.12: Schematic of the actuator line and nacelle used in the simulation

An ASCII data file that contains the turbine model's mesh with the name of "acldata000" should be in the simulation directory, along with the "Urefdata" which is a UCD formatted unstructured triangular mesh. A UCD formatted unstructured triangular mesh for the blades with the name of "acsdata000" and for the nacelle with the name of "nacelle000_" should also be in the simulation directory. The forces on each element are calculated based on the local inflow velocity and the drag and lift coefficients of the 2D airfoil. So, for airfoil data, we should have text files including "Aerodym000_00" for lift and drag coefficients and "FOIL000_00" for cord length of the airfoil. These text files should be inside the project directory.

6.3.2 Main Case Parameters

The main parameters in the control file for setting this case are listed in Table 6.6. The parameters related to the MHK turbine rotor and nacelle model will be also addressed. When using the rotor model, the control options for setting the case are located not only in control.dat but also in Turbine.inp and Nacelle.inp. The flow parameters are summarized in Table 6.6, with the rotor parameters in Table 6.7 and nacelle parameters in Table 6.8. The parameters in Turbine.inp and Nacelle.inp that are not discussed in the Table 6.7 and Table 6.8 are not used in the simulation.

Table 6.6 Parameters in control.dat file for the MHK turbine

Parameter	Option in control file	Value
Time step size	dt [s]	0.001
Activate the actuator disk model	rotor_modeled	5
Number of turbines	turbine	1
Reference length of the turbine	reflength_wt	0.5
Reference velocity of the case. It is only used for dimensionalizing the turbine model output files	refvel_wt	1.0
Distance upstream of the turbine in rotor diameters where the turbine incoming velocity or reference velocity is computed	loc_refvel	1.0
Type of smoothing delta function	deltafunc	10
Delta function width in cell units	halfwidth_dfunc	4.0
Activate the actuator surface model for nacelle	nacelle_model	5.0
Number of nacelles in the simulation	NumberOfNacelle	1.0
Number of nacelles in the same location	NumNacellePerLoc	1.0
Reference length of the nacelle	reflength_nacelle	0.5

Table 6.7 Parameters in Turbine.inp file for the MHK turbine

Parameter	Option in Turbine.inp file	Value

Number of blades in the rotor	num_blade	3
Number of foil types along the blade	num_foiltype	1
Normal direction of the turbine rotor plane.	nx_tb, ny_tb, nz_tb or Yaw1, Yaw2, Yaw3	1.0 0.0 0.0
The turbine rotor initial translation.	x_c, y_c, z_c	8.0 2.5 0.8
Tip speed ratio	Tipspeedratio	-2.5
Radius of the rotor corresponding to the acddata mesh	r_rotor	0.5

Table 6.8 Parameters in Nacelle.inp file for the MHK turbine

Parameter	**Option in Turbine.inp file**	**Value**
Normal direction of the turbine nacelle plane	nx_tb, ny_tb, nz_tb	1.0 0.0 0.0
The turbine nacelle initial translation	x_c, y_c, z_c	8.0 2.5 0.8
friction factor which is used in the nacelle model calculation	frictionfactor	0.4
Radius of the nacelle corresponding to the nacelle000_ mesh	r_nacelle	0.0425
length scale of the local background grid spacing	dh	0.02
The upstream end x-coordinates of nacelle for friction factor calculation	xnacelle_upstreamend	2.5
The upstream end y-coordinates of nacelle for friction factor calculation	ynacelle_upstreamend	0.0

For this test case, time averaging was done, so that it has to be executed in two stages as described in Section 6. In the first stage the flow is fully developed, while in the second stage the time averaging is performed. For developing the flow field, we performed 10000-time steps. For time averaging the flow field, 10000-time steps are enough in order to have fully developed flow. For time averaging, set the option in the control file "averaging" to 3, "rstart" to 0, and rename the result files from the last instantaneous time step (ufield010000.dat, v_field010000.dat, pfield010000.dat, nvfield010000.dat, and cs_010000.dat) to the value corresponding to time zero (ufield000000.dat, vfield000000.dat, ...). Also, when restarting the flow field for averaging, set "rstart_turbinerotation" as 1 and rename TurbineTorqueControl010001_000.dat as TurbineTorqueControl000001_000.dat.

6.3.3 Results

Figure 6.13 and Figure 6.14 shows the contour plots of the instantaneous streamwise velocity in two different planes. As can be seen, actuator surface simulation can visualize the vortical structures and

turbulence in the flow field with a good resolution in comparison with the turbine resolving method. Also, a wake region can be found in these contours which is due to the effect of nacelle. This method is not as computationally expensive as turbine resolving and can be a good choice in case of tidal farms.

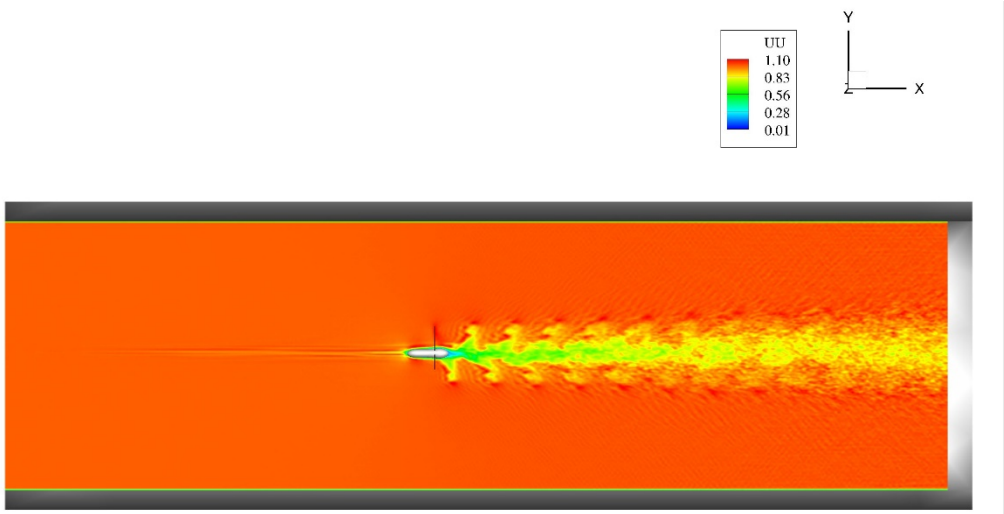


Figure 6.13: Contours of instantaneous streamwise velocity on the Y-X plane at the hub height

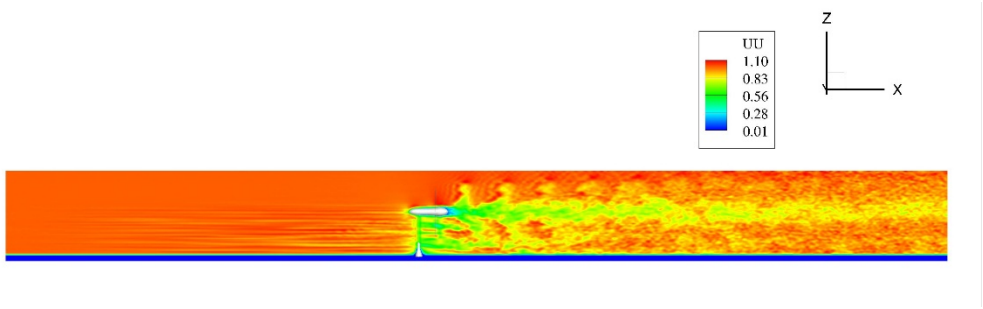


Figure 6.14: Contours of instantaneous streamwise velocity on the Z-X plane at the hub height

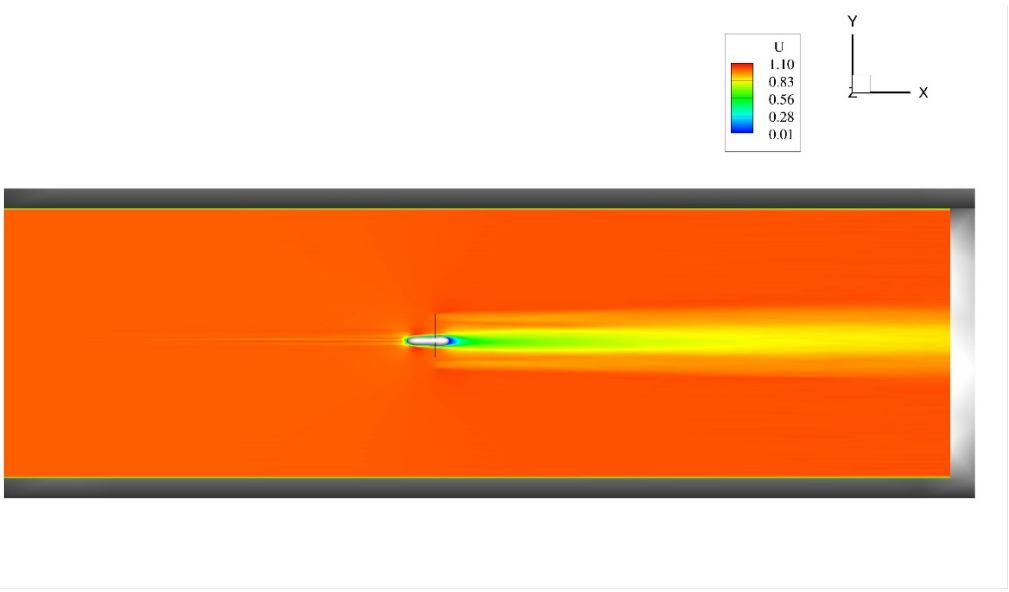


Figure 6.15: Time-averaged streamwise velocity on the Z-X plane at the hub height

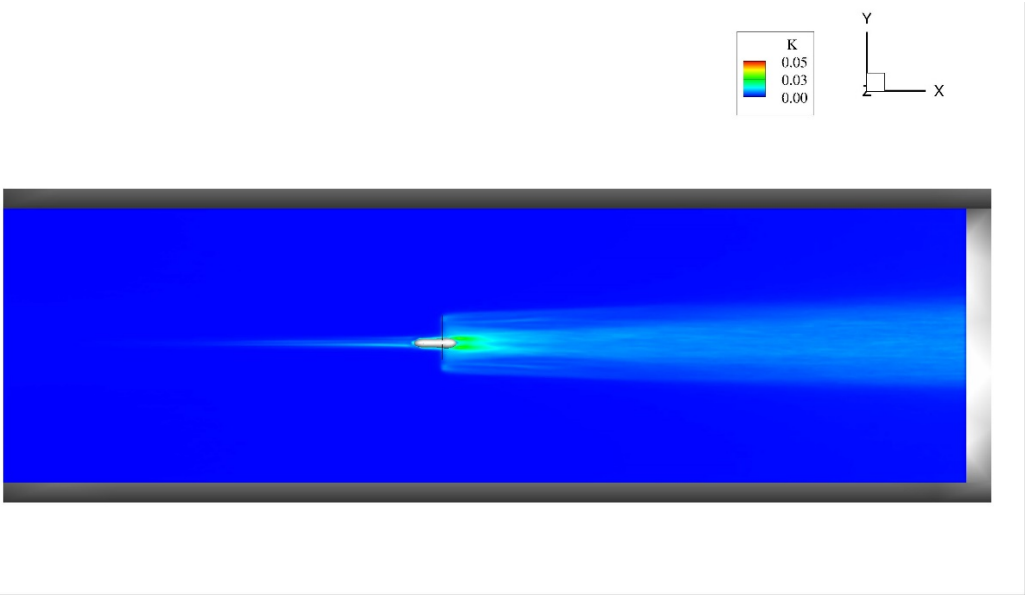


Figure 6.16: Turbulent Kinetic Energy on the Z-X plane at the hub height

6.4 Turbine resolving simulations with IB nacelle

6.4.1 Case Definition

This test case is for introducing and simulating the coupled FSI algorithm for simulating the flow past real-life MHK turbines consisting of the moving rotor and stationary nacelle, pylon and foundation. To resolve flow-blade interactions directly, we used a turbine-resolving approach. In the turbine-resolving approach, flow around MKH turbine blades is directly resolved in numerical simulation by employing the grid that is sufficiently fine enough to resolve them. The turbine-resolving approach would be more appropriate for studying the detailed flow physics around individual turbines. Furthermore, flows past

MHK turbines in natural or man-made waterways occur at high Reynolds numbers (10^6 to 10^7 based on the mean flow depth and velocity) and are dominated by energetic coherent structures induced by the interaction of moving and stationary turbine components with the complex waterway bathymetry and the approaching turbulent flow. Therefore, one must employ turbulence closure models that can to capture highly 3D and dynamic flow environment and resolve turbulent flows dominated by energetic coherent structures [31].

In the turbine resolving approach, the complete MHK turbine geometry, including the rotor and all stationary parts, is treated as a sharp interface immersed boundary and embedded in a background curvilinear grid discretizing the channel or natural waterway. The nacelle geometry is represented by the actual surface of the nacelle with distributed forces, as can be seen in the figure below.

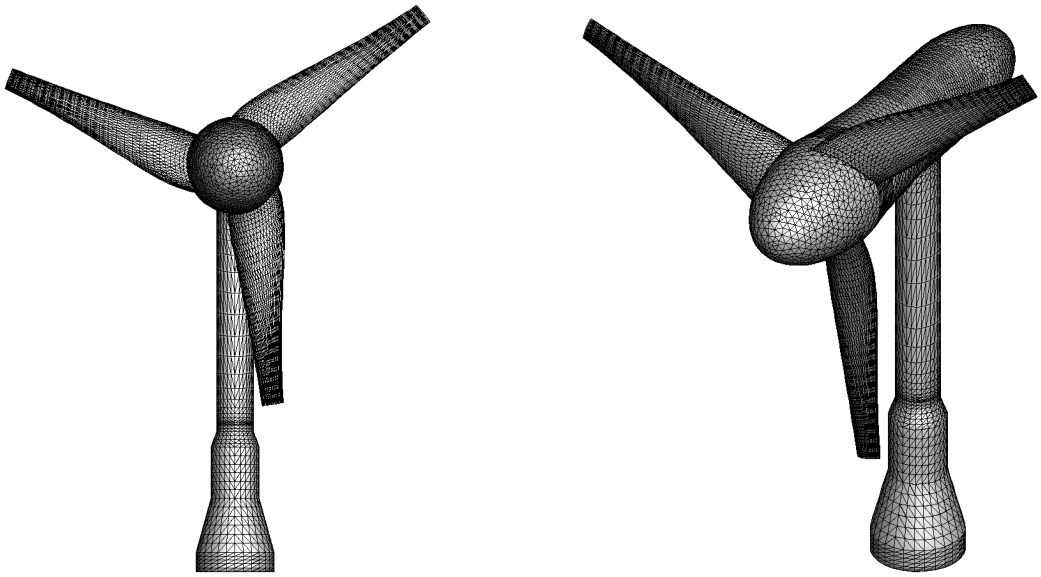


Figure 6.17: Schematic of the tower and nacelle as IBs used in the turbine resolving simulation

A text file that contains the needed parameters for the FSI simulation with the name of "fsi-rot.dat" should be also in the simulation directory. In this simulation, we have 3 IBs including the channel bed and the turbine's blade, tower, and nacelle. For the channel bed all the parameters are zero, so we will not talk about them. The rest can be found in Table 6.10. The bottom wall cannot be resolved with the current grid resolution and is treated with a wall model.

6.4.2 Main Case Parameters

The main parameters in the control file for setting this case are listed in Table 6.9. When using the rotate FSI, the control options for setting the case are located not only in control.dat but also in fsi-rot.dat. The flow and FSI parameters are summarized in Table 6.9, with the rotate FSI parameters in Table 6.10. The parameters in control.dat that are not discussed in Table 6.9 are not used in the simulation.

Table 6.9 Parameters in control.dat file for the MHK turbine

Parameter	Option in control file	Value
Time step size	dt [s]	0.0005
Move the structure in a single translational DoF	fsi	0
Move the structure in a single rotational DoF	rfsi	1
restarting a simulation when using FSI	rstart_fsi	0
Use loose coupling in FSI simulation	str	0
Reduced velocity	red_vel	0.8975
Reduced mass	mu_s	0.25

Table 6.10 Parameters in fsi-rot.dat file for the MHK turbine

Parameter	Option in fsi-rot.dat file	Value for IB2	Value for IB3
Number of IB	IBNum	1	2
axis of rotation	Rot_Dir	0	0
Angle of rotor	XYAngle	0	0
angular velocity	Ang_V	1	0
center of rotation	x_r, y_r, z_r	0.036, 0, 0.8	0, 0, 0.8
Center of Mass for translating the IBs	CMx_c, CMy_c, CMz_c	6.964 2.5 0	6.964 2.5 0

6.4.3 Results

Figure 6.18 and Figure 6.19 show the contour plots of the instantaneous streamwise velocity in two different planes. As can be seen, turbine resolving simulation can visualize the vortical structures and turbulence in the flow field with a high resolution. Also, a wake region can be found in these contours. Although this method is computationally expensive, it has a high accuracy in capturing most of the eddies in the LES simulation.

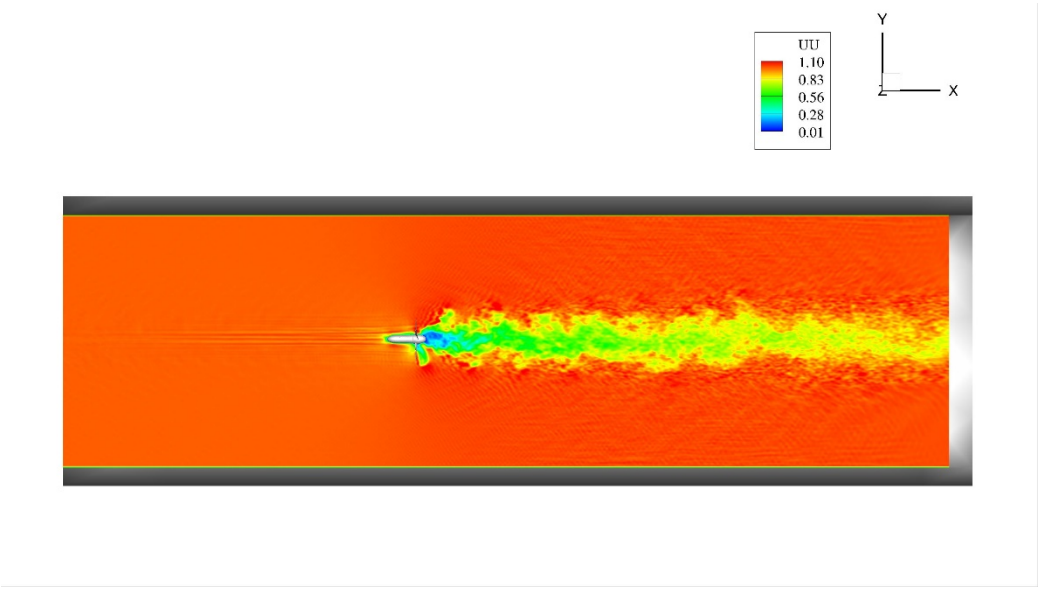


Figure 6.18: Contours of instantaneous streamwise velocity on the X_Y plane at the hub height

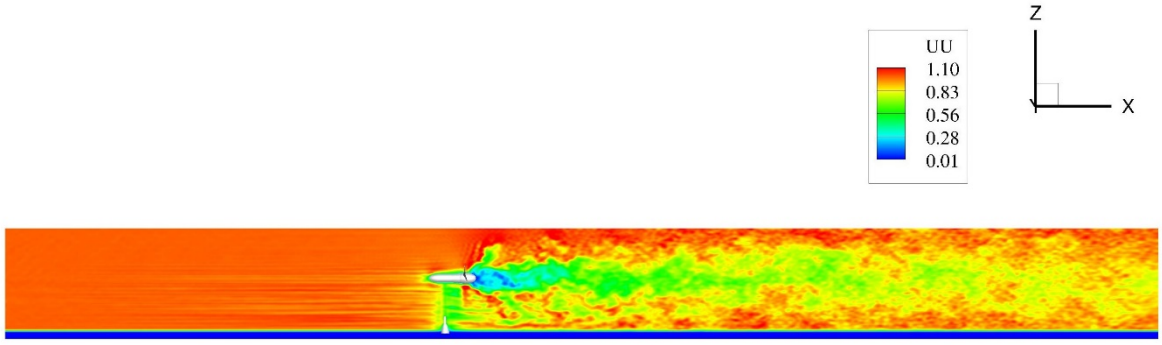


Figure 6.19: Contours of instantaneous streamwise velocity on the Z-X plane at the hub height

An animation of the instantaneous streamwise velocity on the X_Y plane at the hub height can also be found at this link.

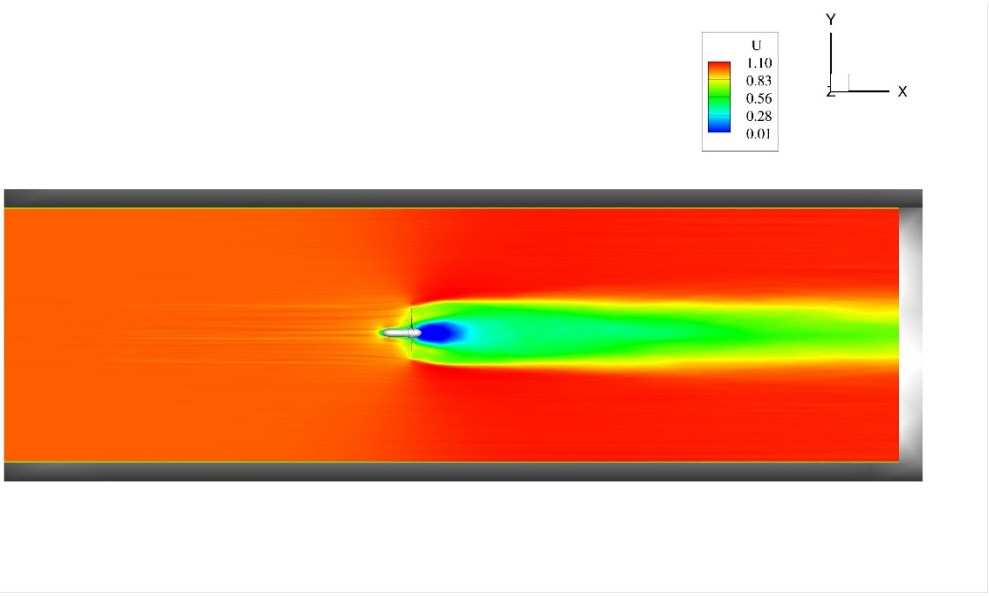


Figure 6.20: Time-averaged streamwise velocity on the Z-X plane at the hub height

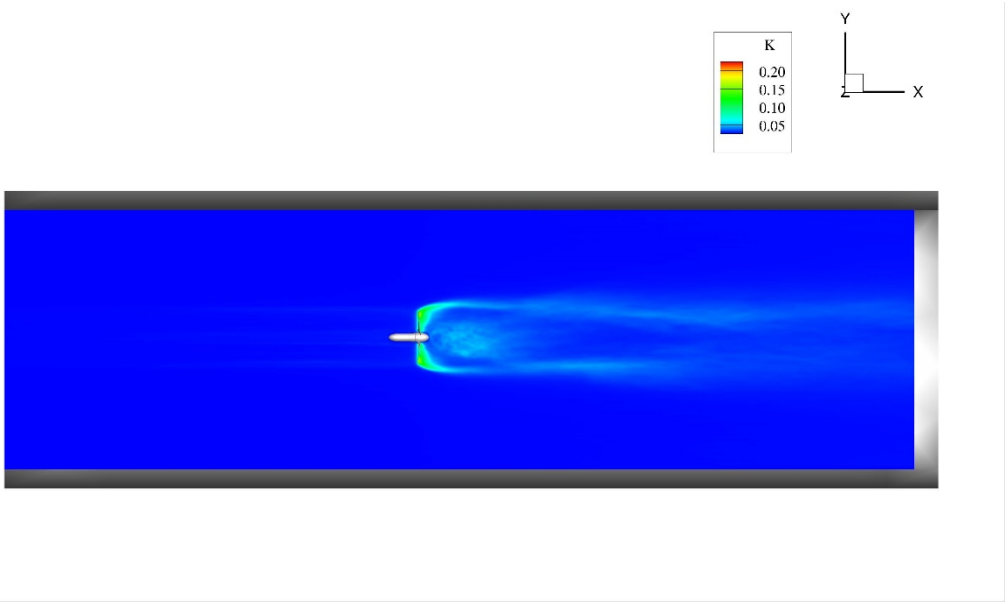


Figure 6.21: Turbulent Kinetic Energy on the Z-X plane at the hub height

6.5 Turbine Resolving Simulations with IB Nacelle Integrating Solitary Wave

6.5.1 Case Definition

This test case is for introducing and simulating the coupled FSI algorithm for simulating the wave and the flow past real-life MHK turbines consisting of the moving rotor and stationary nacelle, pylon and foundation. To directly resolve wave and flow-blade interactions, we employed a turbine-resolving case with the solitary wave designation. It is performed integrating the solitary wave designation into the

domain which resolves flow-blade interactions with a turbine-resolving approach. We used solitary wave herein to predict the impacts of waves on infrastructures.

In the turbine-resolving approach, flow around MHK turbine blades is directly resolved in numerical simulation by employing the grid that is sufficiently fine enough to resolve them. The turbine-resolving approach would be more appropriate for studying the detailed flow physics around individual turbines. Furthermore, flows past MHK turbines in natural or manmade waterways occur at high Reynolds numbers (10^6 to 10^7 based on the mean flow depth and velocity) and are dominated by energetic coherent structures induced by the interaction of moving and stationary turbine components with the complex waterway bathymetry and the approaching turbulent flow. Therefore, turbulence closure models that are able to capture such highly 3D and dynamic flow environment and resolve turbulent flows dominated by energetic coherent structures need to be employed [29].

In the turbine resolving approach, the complete MHK turbine geometry, including the rotor and all stationary parts, is treated as a sharp interface immersed boundary and embedded in a background curvilinear grid discretizing the channel or natural waterway. The nacelle geometry is represented by the actual surface of the nacelle with distributed forces, as can be seen in the figure below.

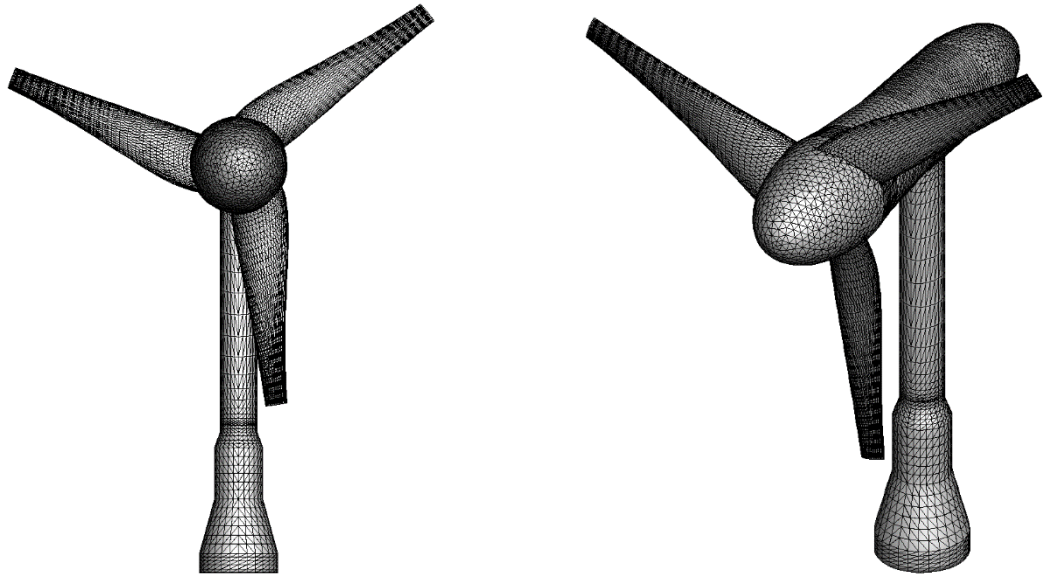


Figure 6. 22: Schematic of the tower and nacelle as IBs used in the turbine resolving simulation

A text file that contains the needed parameters for the FSI simulation with the name of "fsi-rot.dat" should be also in the simulation directory. In this simulation, we have 3 IBs, including channel bed, turbine's blade, and tower and nacelle. For the channel bed, all the parameters are zero except 'Rot_Dir' parameter which should be defined as 2. The rest can be found in Table 6.12. The bottom wall cannot be resolved with the current grid resolution and is treated with a wall model.

6.5.2 Main Case Parameters

The main parameters in the control file for setting this case are listed in Table 6.11 & 6.12. Parameters required for levelset method and wave characteristics are tabulated in Table 6.12. When using the rotate FSI, the control options for setting the case are located not only in control.dat but also in fsi-rot.dat. The flow and FSI parameters are summarized in Table 6.11, with the rotating FSI parameters in Table 6.13. The parameters in control.dat that are not discussed in Table 6.11 and 6.12 are not used in the simulation.

Table 6. 11: Parameters for the level-set method and wave characteristics in control.dat file for the MHK turbine

Parameter	Option in control file	Value
Activates the levelset method	levelset	1
Density of the water in kg/m^3	rho0	1000
Density of the air in kg/m^3	rho1	1.2
Dynamic viscosity of water in Ns/m^2	mu0	1.e-3
Dynamic viscosity of air in Ns/m^2	mu1	1.8e-5
The thickness of the air/water interface	dthick	0.02
These parameters will keep constant the free surface elevation at the corresponding boundary.	inlet_y, outlet_y	0.78
Number of times to solve the reinitialization equation for mass conservation.	level_it	30
Defines the pseudo-time step size used in the reinitialization equation.	levelset_tau	0.033
Activates the solitary wave	solitary_wave	1
It sets the wave amplitude	solitary_wave_amplitude	0.1
Allows to define the source of wave in streamwise direction	inlet_z_for_solitary_wave	0.0
Time when the wave is introduced into the domain	ti_start_solitary_wave	0.0

Table 6. 12: Parameters in control.dat file for the MHK turbine

Parameter	Option in control file	Value
Time step size	dt [s]	0.001
Move the structure in a single translational DoF	fsi	0
Move the structure in a single rotational DoF	rfsi	1
restarting a simulation when using FSI	rstart_fsi	0
Use loose coupling in FSI simulation	str	0
Reduced velocity	red_vel	0.8975
Reduced mass	mu_s	0.25

Table 6. 13: Parameters in fsi-rot.dat file for the MHK turbine

Parameter	Option in fsi-rot.dat file	Value for IB2	Value for IB3
Number of IB	IBNum	1	2
axis of rotation	Rot_Dir	2	2
Angle of rotor	XYAngle	0	0
angular velocity	Ang_V	-1	0
center of rotation	x_r, y_r, z_r	0.036, 0, 0.8	0, 0, 0.764
Center of Mass for translating the IBs	CMx_c, CMy_c, CMz_c	2.5 0 8.964	2.5 0 8.964

6.5.3 Results

Figure 6.23 and Figure 6.24 show the contour plots of the instantaneous streamwise velocity in two different planes. As can be seen in Figure 6.23, turbine resolving simulation can visualize the vortical structures and turbulence in the flow field with a high resolution. Also, a wake region can be found in these contours. Although this method is computationally expensive, it has a high accuracy in capturing most of the eddies in the LES simulation.

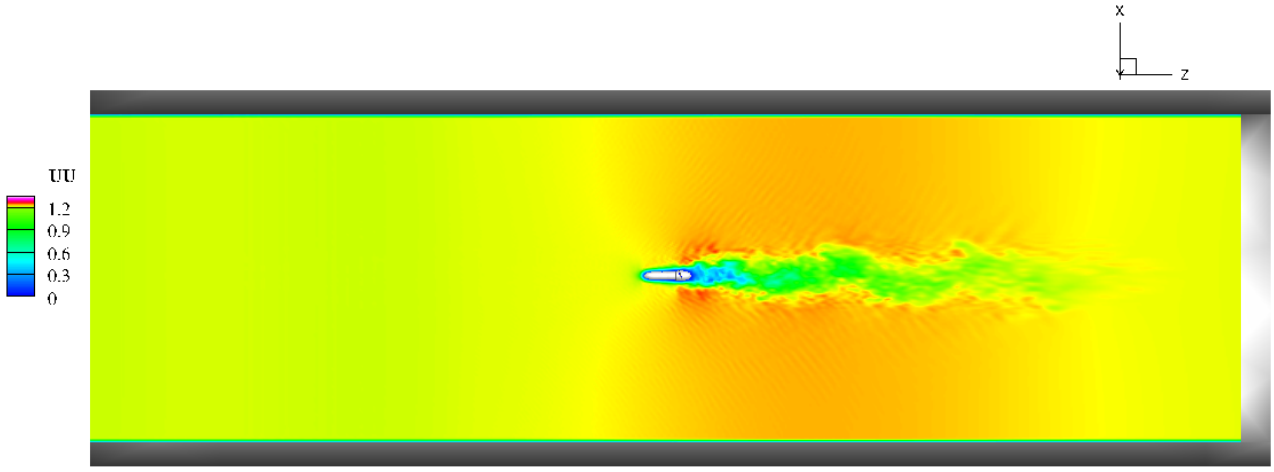


Figure 6. 23: Contours of instantaneous streamwise velocity on the X_Y plane at the hub height
Besides, in Figure 6.24, the effect of the wave is observed with the red line which demonstrates the free surface with the solitary wave effects.

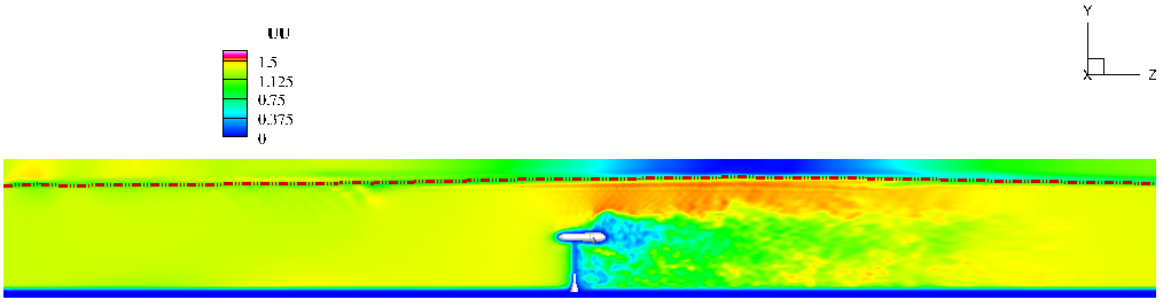


Figure 6. 24: Contours of instantaneous streamwise velocity on the Z-X plane at the hub height

An animation of the instantaneous streamwise velocity on the X_Y plane at the hub height can also be found at this link.

Bibliography

[1] L. Ge and F. Sotiropoulos, "A numerical method for solving the 3D unsteady incompressible Navier–Stokes equations in curvilinear domains with complex immersed boundaries," *Journal of Computational Physics*, vol. 225, pp. 1782–1809, Aug. 2007.

[2] T. B. Le and F. Sotiropoulos, "Fluid–structure interaction of an aortic heart valve prosthesis driven by an animated anatomic left ventricle," *Journal of computational physics*, vol. 244, pp. 41–62, 2013.

[3] T. B. Le, *A computational framework for simulating cardiovascular flows in patient-specific anatomies*. PhD thesis, University of Minnesota, 2011.

[4] I. Borazjani and F. Sotiropoulos, "The effect of implantation orientation of a bileaflet mechanical heart valve on kinematics and hemodynamics in an anatomic aorta," *Journal of biomechanical engineering*, vol. 132, no. 11, p. 111005, 2010.

[5] A. Khosronejad, S. Kang, I. Borazjani, and F. Sotiropoulos, "Curvilinear immersed boundary method for simulating coupled flow and bed morphodynamic interactions due to sediment transport phenomena," *Advances in water resources*, vol. 34, no. 7, pp. 829–843, 2011.

[6] A. Khosronejad, S. Kang, and F. Sotiropoulos, "Experimental and computational investigation of local scour around bridge piers," *Advances in Water Resources*, vol. 37, pp. 73–85, 2012.

[7] A. Khosronejad, C. Hill, S. Kang, and F. Sotiropoulos, "Computational and experimental investigation of scour past laboratory models of stream restoration rock structures," *Advances in Water Resources*, vol. 54, pp. 191–207, 2013.

[8] X. Yang, S. Kang, and F. Sotiropoulos, "Computational study and modeling of turbine spacing effects in infinite aligned wind farms," *Physics of Fluids (1994-present)*, vol. 24, no. 11, p. 115107, 2012.

[9] X. Yang, J. Annoni, P. Seiler, and F. Sotiropoulos, "Modeling the effect of control on the wake of a utility-scale turbine via large-eddy simulation," in *Journal of Physics: Conference Series*, vol. 524, p. 012180, IOP Publishing, 2014.

[10] I. Borazjani, L. Ge, and F. Sotiropoulos, "Curvilinear immersed boundary method for simulating fluid structure interaction with complex 3D rigid bod- ies," *Journal of Computational Physics*, vol. 227, pp. 7587–7620, Aug. 2008.

[11] I. Borazjani and F. Sotiropoulos, "Vortex induced vibrations of two cylinders in tandem arrangement in the proximity-wake interference region," *Journal of fluid mechanics*, vol. 621, pp. 321–364, 2009. PMID: 19693281.

[12] W. Cabot and P. Moin, "Approximate wall boundary conditions in the large-eddy simulation of high reynolds number flow," *Flow, Turbulence and Combustion*, vol. 63, no. 1-4, pp. 269—-291, 2000.

[13] M. Wang and P. Moin, "Dynamic wall modeling for large-eddy simulation of complex turbulent flows," *Physics of Fluids*, vol. 14, no. 7, p. 2043, 2002.

[14] J.-I. Choi, R. C. Oberoi, J. R. Edwards, and J. A. Rosati, "An immersed boundary method for complex incompressible flows," *Journal of Computational Physics*, vol. 224, pp. 757–784, June 2007.

[15] S. Kang and F. Sotiropoulos, "Numerical modeling of 3D turbulent free surface flow in natural waterways," *Advances in Water Resources*, vol. 40, pp. 23–36, May 2012.

[16] B. M. Irons and R. C. Tuck, "A version of the aitken accelerator for computer iteration," *International Journal for Numerical Methods in Engineering*, vol. 1, no. 3, pp. 275—-277, 1969.

[17] X. Yang, F. Sotiropoulos, R. J. Conzemius, J. N. Wachtler, and M. B. Strong, "Large-eddy simulation of turbulent flow past wind turbines/farms: the Vir- tual Wind Simulator (VWiS): LES of turbulent flow past wind turbines/farms: VWiS," *Wind Energy*, pp. n/a–n/a, Aug. 2014.

[18] X. Yang, X. Zhang, Z. Li, and G. W. He, "A smoothing technique for discrete delta functions with application to immersed boundary method in moving bound- ary simulations," *Journal of Computational Physics*, vol. 228, no. 20, pp. 7821 – 7836, 2009.

[19] Z. Du and M. Selig, *A 3-D stall-delay model for horizontal axis wind turbine performance prediction*.

[20] W. Z. Shen, R. Mikkelsen, J. N. Sørensen, and C. Bak, "Tip loss corrections forwind turbine computations," *Wind Energy*, vol. 8, no. 4, pp. 457–475, 2005.

[21] W. Z. Shen, J. N. Sørensen, and R. Mikkelsen, "Tip loss correction for actua- tor/navier–stokes computations," *Journal of Solar Energy Engineering*, vol. 127, pp. 209–213, 04 2005.

[22] W. Cabot and P. Moin, “Approximate wall boundary conditions in the large-eddy simulation of high reynolds number flow,” *Flow, Turbulence and Combustion*, vol. 63, pp. 269–291, Jan 2000.

[23] U. Piomelli and E. Balaras, “Wall-layer models for large-eddy simulations,” *An- nual Review of Fluid Mechanics*, vol. 34, no. 1, pp. 349–374, 2002.

[24] F. Schultz-Grunow, “Neues reibungswiderstandsgesetz für glatte platten,” *Luft- fahrtforschung*, vol. 17, no. 8, pp. 239–246, 1940.

[25] S. Kang, A. Lightbody, C. Hill, and F. Sotiropoulos, “High-resolution numerical simulation of turbulence in natural waterways,” *Advances in Water Resources*, vol. 34, pp. 98–113, Jan. 2011.

[26] J. Smagorinsky, “General circulation experiments with the primitive equations: I. the basic experiment*,” *Monthly weather review*, vol. 91, no. 3, pp. 99–164,1963.

[27] M. Germano, U. Piomelli, P. Moin, and W. H. Cabot, “A dynamic subgrid-scale eddy viscosity model,” *Physics of Fluids A: Fluid Dynamics (1989-1993)*, vol. 3,no. 7, pp. 1760–1765, 1991.

[28] E. Bagherizadeh, Z. Zhang, A. Farhadzadeh, D. Angelidis, M. Ghazian Arabi, S. Moghimi, & A, Khosronejad, “Numerical modelling of solitary wave and structure interactions using level-set and immersed boundary methods by adopting adequate inlet boundary conditions”. *Journal of Hydraulic Research*, Vol. 59(4), pp. 559-585, 2021.

[29] A. Calderer, X. Guo, L. Shen, and F. Sotiropoulos, “Coupled fluid-structure interaction simulation of floating offshore wind turbines and waves: a large eddy simulation approach," *Journal of Physics: Conference Series*, vol. 524, p. 012091, IOP Publishing, 2014.

[30] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, V. Ei-jkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, and H. Zhang, “Petsc web page,” 2014. http://www.mcs.anl.gov/petsc.

[31] S. Kang, I. Borazjani, J. A. Colby, F. Sotiropoulos, “Numerical simulation of 3D flow past a real-life marine hydrokinetic turbine”, *Advances in Water Resources*, Volume 39, Pages 33-43, 2012

[32] M. Wang, P. Moin, “Dynamic wall modeling for large-eddy simulation of complex turbulent flows”. *Phys Fluids*, 14:2043–51, 2012.

[33] S. Kang, A. Lightbody, C. Hill, F. Sotiropoulos, “High-resolution numerical simulation of turbulence in natural waterways”, *Advances in Water Resources*, Volume 34, Issue 1, Pages 98-113, 2011.

[34] L. Ge, F. Sotiropoulos, “A numerical method for solving the 3D unsteady

incompressible Navier–Stokes equations in curvilinear domains with complex immersed boundaries", *Journal of Computational Physics*, Volume 225, Issue 2, Pages 1782-1809, 2007.

[35] S. Chawdhary, D. Angelidis, J. Colby, D. Corren, L. Shen, and F. Sotiropoulos, "Multiresolution large-Eddy simulation of an array of hydrokinetic turbines in a field-scale River: The Roosevelt Island Tidal Energy Project in New York City". *Water Resources Research*, 54(12), 10-188, 2018

[36] C. Santoni, A. Khosronejad, P. Seiler, and F. Sotiropoulos, "Toward control co-design of utility-scale wind turbines: Collective vs. individual blade pitch control". *Energy Reports*, 9, 793-806, 2023.

[37] A. Calderer, X. Yang, D. Angelidis, A. Khosronejad, T. Le, S. Kang, A. Gilmanov, L. Ge, I. Borazjani, Virtual Flow Simulator (Version 00) [Computer software], May 2015. https://www.osti.gov//servlets/purl/1312901.

[38] D.C. Wilcox, "Turbulence modeling for CFD". La Canada, CA: DCW Industries." Inc, November 34, 2006.

[39] F.R. Menter, M Kuntz, and R Langtry, "Ten years of industrial experience with the SST turbulence model." *Turbulence, heat and mass transfer*, 4(1), 625-632, 2003.

[40] R. G. Dean, & R. A. Dalrymple, "Coastal processes with engineering applications". *Cambridge University Press*, (2004).