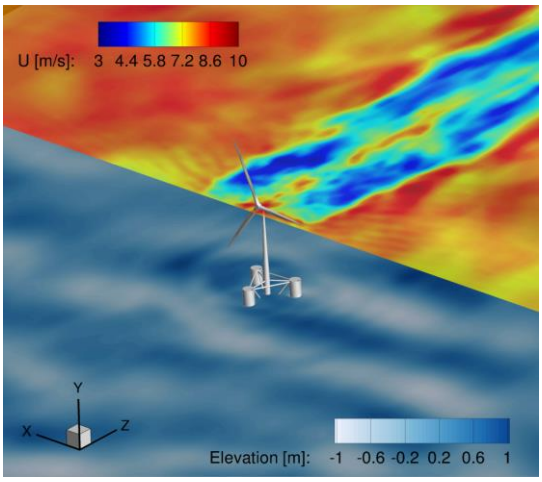
VFS

User Manual

Wind Version 1.0

March 3, 2016



University of Minnesota
Driven to Discover℠

Sandia National Laboratories

Acknowledgements

VFS 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) and the University of Minnesota Initiative for Renewable Energy and the Environment (RM-0005-12).

Current Code Developers

Dennis Angelidis

Antoni Calderer

Anvar Gilmanov

Ali Khosronejad

Trung Le

Xiaolei Yang

Former Code Developers

Iman Borazjani

Liang Ge

Seokkoo Kang

Contributors to this manual

Antoni Calderer (Saint Anthony Falls Lab.)

Ann Dallman (Sandia National Laboratories)

D. Todd Griffith (Sandia National Laboratories)

Thomas Herges (Sandia National Laboratories)

Kelley Ruehl (Sandia National Laboratories)

License

Contents

Chapter 1

Introduction

Virtual Flow Simulator (VFS) is a three-dimensional (3D) incompressible Navier-Stokes solver based on the Curvilinear Immersed Boundary (CURVIB) method developed by Ge and Sotiropulos [1]. The CURVIB is a sharp interface type of immersed boundary (IB) method that enables the simulation of fluid flows with 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 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 wind energy applications, one could take advantage of this feature when simulating the site specific geometry of a wind farm. The fluid mesh can follow the actual topography of the terrain while treating the turbines as immersed bodies.

VFS also integrates a two-phase flow solver module based on the level set method that allows simulation of coupled free surface flows with water waves, winds, and floating structures [2, 3]. This module was designed to simulate offshore floating wind turbines considering the non-linear effects of the free surface with a two-phase flow solver, the coupled 6 degrees of freedom (DoF) dynamics of the floating structure, and the ability to incorporate turbulent wind and wave conditions representative of realistic offshore environments.

The CURVIB method has been applied to a broad range of applications, such as cardiovascular flows [4, 5, 6], river bed morphodynamics [7, 8, 9], and wind and hydro-kinetic turbine simulations [10, 11]. For wind energy applications, a turbine can be resolved by immersing the geometry with the IB method or by using one of the different rotor parametrization models implemented.

The current version of the manual is for the VFS-Wind version of VFS. This version of the code includes the base solver and all the necessary libraries for simulating land based and offshore wind farms. The parts that are not included in this version

are the modules for sediment transport, bubbly flows, and elastic body deformations.

The organization of this user manual is as follows: in Chapter 2, the main governing equations and numerical methods employed by VFS are briefly described. In Chapter 3, the user is introduced to the basic steps to start using VFS. In Chapter 4, the source code organization is introduced with a description of the main functions in each module. In Chapter 5, the code input parameters are described. Finally, in Chapter 6, a series of application cases are documented.

Chapter 2

Overview of the Numerical Algorithms

2.1 The Flow Solver

The code solves the spatially-filtered Navier-Stokes equations governing incompressible flows of two immiscible fluids. The equations adopt a two-fluid formulation based on the level set method and are expressed in generalized curvilinear coordinates as follows ($i, j, k, l = 1, 2, 3$):

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

$$\begin{aligned}\frac{1}{J}\frac{\partial U^j}{\partial t} &= \frac{\xi_l^i}{J}\left(-\frac{\partial}{\partial \xi_j}\left(U^j u_l\right) + \frac{1}{\rho\left(\phi\right)Re}\frac{\partial}{\partial \xi^j}\left(\mu\left(\phi\right)\frac{\xi_l^j \xi_l^k}{J}\frac{\partial u_l}{\partial \xi^k}\right) - \right. \\ &\left. -\frac{1}{\rho\left(\phi\right)}\frac{\partial}{\partial \xi^j}\left(\frac{\xi_l^j p}{J}\right) - \frac{1}{\rho\left(\phi\right)}\frac{\partial \tau_{lj}}{\partial \xi^j} - \frac{\kappa}{\rho(\phi)We^2}\frac{\partial h(\phi)}{\partial x_j} + \frac{\delta_{i2}}{Fr^2}\right),\end{aligned} \tag{2.2}$$

where ϕ is the level set function defined below, 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, κ is the curvature of the interface, δ_{ij} is the Kronecker delta, h is the smoothed Heaviside function, and Re, Fr, and We are the dimensionless Reynolds, Froude, and Weber numbers, respectively, which can be defined as:

$$Re = \frac{UL\rho_{water}}{\mu_{water}}, Fr = \frac{U}{\sqrt{gL}}, We = U\sqrt{\frac{\rho_{water}L}{\sigma}} \tag{2.3}$$

where U and L are the characteristic velocity and linear dimension, ρ_{water} and μ_{water}, the density and dynamic viscosity of the water phase, g the gravitational acceleration, and σ the surface tension.

The level set function ϕ is a signed distance function, adopting positive values on the water phase and negative values on the air phase. The density and viscosity are taken to be constant within each phase, and transition smoothly across the interface, which is smeared over a distance 2ϵ, as follows:

$$\rho(\phi) = \rho_{air} + (\rho_{water} - \rho_{air})\, h(\phi), \tag{2.4}$$

$$\mu(\phi) = \mu_{air} + (\mu_{water} - \mu_{air})\, h(\phi), \tag{2.5}$$

where $h(\phi)$ is the smoothed Heaviside function given in [12] and defined as:

$$h(\phi) = \begin{cases} 0 & \phi < -\epsilon, \\ \frac{1}{2} + \frac{\phi}{2\epsilon} + \frac{1}{2\pi}\sin(\frac{\pi\phi}{\epsilon}) & -\epsilon \leq \phi \leq \epsilon, \\ 1 & \epsilon < \phi. \end{cases} \tag{2.6}$$

Note that using the above equations, one can recover the single fluid formulation, by setting the density and viscosity to a constant value throughout the whole computational domain.

The free surface interface, given by the zero level of the distance function ϕ, can be modeled by solving the following level set equation proposed by Osher and Sethian [13]:

$$\frac{1}{J}\frac{\partial \phi}{\partial t} + U^j \frac{\partial \phi}{\partial \xi^j} = 0. \tag{2.7}$$

After solving the above level set advection equation, the distance function no longer maintains a unit gradient $|\nabla\phi| = 1$, which is a requirement to ensure conservation of mass between the two phases. To remedy this situation, the code solves the mass conserving re-initialization equation proposed by Sussman and Fatemi [14]. A detailed description of the method in the context of curvilinear coordinates can be found in Kang and Sotiropoulos [15].

The flow equations (2.1) and (2.2) are solved using the fractional step method of Ge and Sotiropoulos [1]. The momentum equations are discretized in space and time with a second-order central differencing scheme for the viscous, pressure gradient, and SGS terms, a second-order central differencing or a third-order WENO scheme for the advective terms, and the second-order Crank-Nicholson method for time advancement as follows:

$$\frac{1}{J}\frac{\mathbf{U}^* - \mathbf{U}^\mathbf{n}}{\triangle t} = \mathbf{P}(p^n, \phi^n) + \frac{1}{2}(\mathbf{F}(\mathbf{U}^*, \mathbf{u}^*, \phi^{n+1}) + (\mathbf{F}(\mathbf{U}^\mathbf{n}, \mathbf{u}^\mathbf{n}, \phi^n)), \tag{2.8}$$

where n indicates the previous time step, $\triangle t$ the time step size, $\mathbf{F}$ the right hand side of Eq.(2.2) excluding the pressure term, and $\mathbf{P}$ the pressure term. The continuity condition, discretized with three-point central differencing scheme, is enforced in the second stage of the fractional step method with the following pressure Poisson equation:

$$-J\frac{\partial}{\partial \xi^i}\left(\frac{1}{\rho(\phi)}\frac{\xi^i_l}{J}\frac{\partial}{\partial \xi^j}\left(\frac{\xi^j_l \Pi}{J}\right)\right) = \frac{1}{\triangle t} J \frac{\partial U^{j,*}}{\partial \xi^j}, \tag{2.9}$$

where Π is the pressure correction, used as follows, to update the pressure and the velocity field resulting from the first step of the fractional step method:

$$p^{n+1} = p^n + \Pi, \tag{2.10}$$

$$U^{i,n+1} = U^{i,*} - J\Delta t \frac{1}{\rho(\phi)} \frac{\xi_l^i}{J} \frac{\partial}{\partial \xi^j} \left(\frac{\xi_l^j \Pi}{J} \right), \tag{2.11}$$

The momentum equations are solved using an efficient matrix-free Newton-Krylov solver, and the Poisson equation with a generalized minimal residual (GMRES) method preconditioned with multi-grid.

The level set equations (2.7) are discretized with a third-order WENO scheme in space, and second-order Runge-Kutta in time. The re-initialization step uses a second-order ENO scheme. A detailed description of the method can be found in [15].

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 [16, 17], river bed morphodynamics [7, 8, 9], and cardiovascular flows [4, 5, 6].

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 nodes of the computational domain 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. While, the velocity reconstruction uses the wall models described by [18, 19, 20] 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 $\Delta\phi$ 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 single phase flows is described in Borazjani, Ge, and Sotiropoulos [16], and the extension to two-phase free surface flows in Calderer, Kang, and Sotiropoulos [2].

The code solves the rigid body equations of motion (EoM) in 6 DoF, which can be written in Lagragian form and in principle axis as follows (i=1,2,..6),

$$M\frac{\partial^2 Y^i}{\partial t^2} + C\frac{\partial Y^i}{\partial t} + KY^i = F_f^i + F_e^i \tag{2.12}$$

where Y^i represents the coordinates of the Lagrangian vector describing the motion of the structure. 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, induced by the fluid and by the external forces, respectively.

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

$$F_f = \int_\Gamma -pnd\Gamma + \int_\Gamma \tau_{ij} n_j d\Gamma \tag{2.13}$$

$$M_f = \int_\Gamma -\epsilon_{ijk} r_j p n_k d\Gamma + \int_\Gamma \epsilon_{ijk} r_j \tau_{kl} n_l d\Gamma \tag{2.14}$$

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 (Eqs. 2.12) 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 [21] 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 [16].

2.4 Turbine Parameterizations

The actuator disk and actuator line models implemented in the code for parameterizing turbine rotors are given, respectively, in Yang, Kang, and Sotiropoulos [10] and in Yang et al. [22]. 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 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

$$F_{AD} = -\frac{F_T}{\pi D^2/4}, \tag{2.15}$$

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

$$F_T = \frac{1}{2}\rho C_T \frac{\pi}{4} D^2 U_\infty^2, \tag{2.16}$$

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

$$U_\infty = \frac{u_d}{1-a}, \tag{2.17}$$

where u_d is the disk-averaged stream-wise velocity computed as

$$u_d = \frac{4}{\pi D^2} \sum_{N_t} u(X) A(X), \tag{2.18}$$

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

$$u(X) = \sum_{N_D} u(x) \delta_h(x - X) V(x), \tag{2.19}$$

where δ_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:

$$f_{AD}(x) = \sum_{N_D} F_{AD}(X) \delta_h(x - X) A(x). \tag{2.20}$$

2.4.2 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 (L) and drag (D) forces are computed using the following expressions:

$$L = \frac{1}{2}\rho C_L C V_{rel}^2, \tag{2.21}$$

$$D = \frac{1}{2}\rho C_D C V_{rel}^2, \tag{2.22}$$

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

$$V_{rel} = (u_z, u_\theta - \Omega r) \tag{2.23}$$

where u_z and $u_\theta - \Omega r$ are the components of the velocity in the axial and azimuthal directions, respectively, Ω the angular velocity of the rotor, and r 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 (2.19). 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:

$$f_{AL}(x) = \sum_{N_L} F(X)\delta_h(x - X)A(x). \tag{2.24}$$

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.

2.5 Large-Eddy Simulation

The description of the Large-eddy simulation (LES) model implemented in the code is extensively described in Kang et al. [23]. 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 [24] which reads as follows

$$\tau_{ij} - \frac{1}{3}\tau_{kk}\delta_{ij} = -2\mu_t \overline{S}_{ij}, \tag{2.25}$$

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

$$\mu_t = C_s \Delta^2 |\overline{S}|, \tag{2.26}$$

where Δ is the filter width taken from the box filter, $|\overline{S}| = (2\overline{S}_{ij}\overline{S}_{ij})^{1/2}$ is the magnitude of strain-rate tensor, and C_s is the Smagorinsky constant computed dynamically with the Smagorinsky model of Germano et al. [25].

2.6 Wave Generation

The code uses an internal generation method as described in [3]. As illustrated in Figure 2.1, a surface force is applied at the so called source region, generating waves that propagate symmetrically to both stream-wise directions. A sponge layer method is used at the lateral boundaries to dissipate the waves and prevent reflections. The area between the source region and the sponge layer can be used to study body-wave interactions. Both the sponge layer force and the wave generation force are introduced in the code using a source term in the right hand side of the momentum equations.

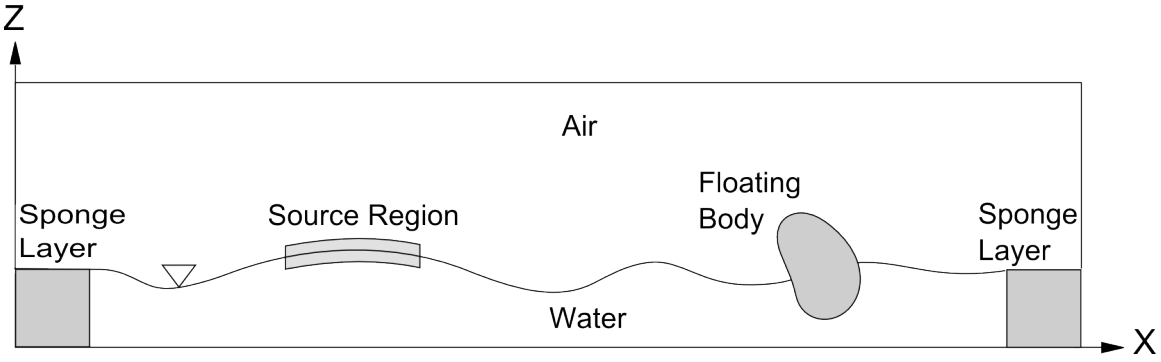


Figure 2.1: Schematic description of the wave generation method using the free surface forcing method [3].

To generate the following surface elevation,

$$\eta(x, y, t) = A\cos(k_x x + k_y y - \omega t + \theta), \tag{2.27}$$

where k_x and k_y are the components of the wavenumber vector K, θ is the wave phase, A is the wave amplitude, and ω the wave frequency, the forcing term reads as follows

$$S_i(x, y, t) = n_i(\phi) P_0 \delta(x, \epsilon_x)\delta(\phi, \epsilon_\phi) sin(\omega t - k_y y - \theta), \tag{2.28}$$

where P_0 is a coefficient that depends on the wave and fluid characteristics,

$$P_0 = A\frac{g^2}{\omega^2}\frac{\epsilon}{f(\epsilon_x, k_x)}\frac{2\rho_w}{\rho_a + \rho_w}\frac{k_x}{\left(k_x^2 + k_y^2\right)^{1/2}}, \tag{2.29}$$

δ is a distribution function defined as:

$$\delta(\alpha, \beta) = \begin{cases} \frac{1}{2\beta}\left[1 + \cos\left(\frac{\pi\alpha}{\beta}\right)\right] & \text{if } -\beta < t < \beta \\ 0 & \text{otherwise.} \end{cases}, \tag{2.30}$$

and $f(\epsilon_x, k_x)$ is

$$f(\epsilon_x, k_x) = \frac{\pi^2}{k_x\left(\pi^2 - \epsilon_x^2 k_x^2\right)}\sin(k_x \epsilon_x). \tag{2.31}$$

With the present forcing method, wave fields with multiple components, such as a broadband wave spectrum, can be incorporated into the fluid domain. This method enables to simulate the interaction of floating structures with complex wave fields. The wave fields can either be originated in a far-field precursor simulation or taken from theoretical or measured data.

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

$$S_i(x,y,t) = -\left[\mu C_0 u_i + \rho C_1 u_i \left|u_i\right|\right] \frac{\exp\left[\left(\frac{x_s - x}{x_s}\right)^{n_s}\right] - 1}{\exp(1) - 1} \text{ for } (x_0 - x_s) \leq x \leq x_0, \tag{2.32}$$

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. In [3], n_s is 10, C_0 is 200000, and C_1 is 1.0.

Chapter 3

Getting Started

In this chapter, we describe how to install the libraries required for VFS to work and how to run a simulation case.

3.1 Installing PETSC and Required Libraries

VFS is implemented in C and is parallelised 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 need to be properly installed.

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

Note that when installing the PETSC libraries there is the option to automatically install the other required libraries in the case that they are not already in the computer. The PETSC web page [26] (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-p6.tar.gz

4. Unzip the PETSC source code in the source folder:
   tar xvfz petsc-3.1-p6.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-mpich=1 --download-hypre=1 --prefix=installation_folder --with-shared=0 --with-debugging=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 is installed without debugging because it will be used in production mode. The PETSC installation with debugging may be needed for developing purpose 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 is general and 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 in every computer the compiler libraries are located in different directories, the user has to create a new text file with the name: "makefile.local". This file is read by makefile and should contain the following user dependent information:

```
1 MPICXX       = mpicxx
2 HOME    = /Your_Home_Folder/
3 ACML    = /Your_ACML_Folder/
4 PETSC   = /Your_PETSC_Folder/
5 HYPRE   = /Your_HYPRE_Folder/
6 USE_TECPLOT=1 ##1 if tecplot is intalled, otherwise set to 0
7 TEC360HOME=/
```

If all libraries have been successfully installed and properly referenced in the file "makefile.local" 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

Either of the last two commands will generate the executable source file "**vwis**".

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) and Paraview format (vtk). The advantage of Paraview over Tecplot is that it is open source and can be download from the Paraview webpage.

3.2.1 Compiling the Post-Processing File for Paraview Only

This may be useful when Tecplot 360 is not installed in the computer. For creating the post-processing file which is only suitable for generating Paraview output files, first set the tecplot option in the file "makfile.local" to inactive as follows "USE_TECPLOT=0".

Then type the following command in the linux shell:

make data

3.2.2 Compiling the Post-Processing File for Tecplot and Paraview

If this is the case, in the previously described file "makfile.local", the tecplot option has to be active as follows "USE_TECPLOT=1".

In addition, one needs to download the library file "libtecio.a" from the tecplot library webpage (http://www.tecplot.com/downloads/tecio-library/) and 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

3.3.1 File Structure Overview

All the files that are necessary for running VFS should be located 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.

vwis Executable file obtained upon code compilation.

Submission script The linux script for submitting the job in a linux based cluster.

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

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

The remainder of this chapter describes the compiling process for obtaining the executable vwis 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,walltime=4:00:00
6  #PBS -j oe
7
8  cd $PBS_O_WORKDIR
9
10 /openmpi_directory/mpirun --bind-to-core  ./vwis>& 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 "vwis", and the on screen information will be stored in "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" times steps, 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
vfield'x'.dat	Ucont	Contravarian velocity components (fluxes)
ufield'x'.dat	Ucat	Cartesian velocity components
pfield'x'.dat	P	Pressure field
nvfield'x'.dat	Nvert	If IB is used, it indicates the classification of nodes. 3 is structure node, 1 is IB node, and 0 is fluid node.
lfield'x'.dat	level	The distance function used in the levelset 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 (momenum, poisson, levelset, IBMSERA0)
  - momentum: Momentum equation solver.
  - poisson: Poisson solver for the second step of the fractional step method.
  - levelset: If using the level set method it refers to the Reinitialization equation.
  - IBMSERA0: Refers to the searching algorithm for node classification when at least one immersed body is present.
- The third column is the computational time in seconds that it takes the algorithm to complete.
- 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, 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 is 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:

$$ti, Y_x, \dot{Y_x}, F_x, Y_y, \dot{Y_y}, F_y, Y_z, \dot{Y_z}, F_z, \tag{3.2}$$

where ti is the time step number, 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:

$$ti, \Theta_x, \dot{\Theta}_x, \Theta_y, \dot{\Theta}_y, \Theta_z, \dot{\Theta}_z, \tag{3.3}$$

where ti is the time step number, 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_Coeff_SI00

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:

$$ti, F_x, F_y, F_z, Cp_x, Cp_y, Cp_z, Ap_x, Ap_y, Ap_z, \tag{3.4}$$

where ti is the time step number, F denotes the fluid force applied to the body, Cp the normalized force coefficient, and Ap the area of the body projected in

the corresponding direction which has been used for computing Cp. Again, a different file is exported for each additional immersed body.

Momt_Coeff_SI00
This file is equivalent to Force_Coeff_S00 but in the rotational degrees of freedom. The information exported is the following:

$$ti, M_x, M_y, M_z, \tag{3.5}$$

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

surface000_00_nf.dat, surface000_01_nf.dat, ...
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.

sloshing.dat
Data file with information about the free surface elevation in the center of the tank for the sloshing case. The first column is the simulation time [sec], the second is the computed surface elevation [m] at the tank center, the third is the expected theoretical solution at the tank center, and the final column is the error.

3.3.4 Post-Processing

Once VFS 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 (ufieldxx.dat, vfieldxx.dat, pfieldxx.dat, nvfieldxx.dat, lfieldxx.dat, lfieldxx.dat, etc), which are in binary form, to a format which is readable for a visualization software such as TecPlot360 or Paraview.

Create plt Files for Tecplot360

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

mpiexec ./data -tis 0 -tie 50000 -ts 100

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

The above step will generate n number of result files (for each timestep) compatible with tecplot with names similar to: Resultxx.plt where xx 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

Create vtk Files for Paraview

If the user wants to visualize the files using Paraview, the option -vtk 1 should be included as follows

mpiexec ./data -tis 0 -tie 50000 -ts 100 -vtk 1

Averaged results

By default, the plt and vtk 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; 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 the options -avg 2 and 3, the code should have been executed using the option -averaging with a value equal or larger than the -avg value. An example on how to process averaged results is as follows:

mpiexec ./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, regardless of being Tecplot or Paraview formatted, 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 structure node, 1 is IB node, and 0 is fluid node.
level	The distance function used in the levelset method to track the interface

Averaged Results

The variables of the post-processed results file, regardless of being Tecplot or Paraview formatted, 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	Averaged velocity components at the grid cell centers
uu, vv, ww, uv, uw, vw	Turbulence intensities
P	Averaged pressure field
Nvert	If the IB method is used, it indicates the classification of nodes. 3 is structure node, 1 is IB node, and 0 is fluid node.
level	The distance function used in the levelset method to track the interface

Chapter 4

Code Input Parameters

4.1 Units

In terms of the units that the code uses, the user needs to differentiate between two cases, when the level set method is active (levelset option set to 1) and when it is not active (levelset option set to 0). In the case that the level set method is not active, the Navier-Stokes equations are solved in the dimensionless form. In this case, it is recommended that the reference length of the domain and the reference velocity of the flow are both equal to one.

In the case that the level set method is in use, the equations solved have real dimensions, and the units are in MKS (meters-kilograms-seconds system).

4.2 VFS Input Files

The VFS code requires several input files that have to be located by default at the cases directory. The input files are the following:

control.dat

bcs.dat

grid.dat

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 categories as proposed below.

The control options start with a dash "-" symbol. If for any case the same option is introduced twice the code will take the value introduced latest in the file. If a control option wants to be kept in the control file for later use it can be commented by introducing a "!" sign in the start of the line.

Time Related Options

dt (double)

Time step size for the solution of the Navier-Stokes equations. The CFL number has to be smaller than 1 although values less than 0.5 are recommended. When using the level set method, the CFL number is more restrictive and in that case the CFL should be lower than 0.05.

tio (int)

The code exports the complete flow field data every time step that is a multiple of the value of this parameter.

totalsteps (int)

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

rstart (int)

This option is for restarting a simulation. The value given to this parameter is the time step number for which the simulation will restart. This option can only be used if results from a previous simulation are present in the folder. Note that even if the value is set to zero, the simulation will restart from a previous run, in that case, from a zero time step.

rs_fsi (int 0, 1)

This parameter can be used when restarting a simulation (rstart active) and FSI module is in use. If rs_fsi is set to 1, the code will read the file FSI_DATA corresponding to time step rstart. This file contains information regarding the body motion such as body position, velocity, forces, etc.

delete (int 0, 1)

If this option is set to 1, the code keeps in the hard drive only the result files from the two last exported time step, 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 which 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 in 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 need to be renamed is because of the way that the code performs averaging. The code uses the current time step for dividing the velocity sum and obtaining the averaged results. So 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, there is no problem with restarting the simulation during stage 2. The different values for this parameter, 1, 2, and 3, 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 ones from option 2 plus 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" needs to be activated. The value given to "avg" should be lower or equal to the value given to the option "averaging".

Options for Boundary Contitions

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) has to be 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)

Use wall model at the walls of the immersed bodies. It basically 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 need to 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. The units are m^3/s or non-dimensional depending on whether level set is used.

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 reinitialitzation equation for mass conservation. 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.

Modelling Options and Solver Parameters

les (int 0,1,2)

Activating the LES model.

1: Smagorinsky - Lilly model.

2: Dynamic Smagorinsky model (recommended).

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 smaller than 1.0×10^{-5} would be 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 iteration 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 maximum divergence of the flow.

ren (double)

This parameter defines the Reynolds Number in the case of non-dimensional simulations. When using the level set method, the equations solved have dimensions and "ren" is not in use.

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 sufficient to resolve the depth.

x_c, y_c, z_c (double)

Initial translation of the immersed body position.

angle_x0, angle_y0, angle_z0 (double)

Initial rotation of the immersed body position with respect to the center of rotation defined by x_r, y_r, and z_r. Note that the initial rotation is applied after the translation for which x_r, y_r, and z_r are defined.

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.

forced_motion (int 0, 1)

When "fsi" is active, the parameter controls whether to use prescribed motion or FSI motion.

0: The motion of the structure is computed in a coupled manner with the flow.

1: The motion of the structure is prescribed. Both the position of the structure in time as well as the velocity in time are specified in the function Forced_Motion which is located in the code source file fsi_move.c. In this function the motion is defined with an analytic expression. If a user needs

to set a specific motion which can be defined through a mathematical expression it needs to be implemented by editing this function. Obviously, if the code is edited it also needs to be recompiled.

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 1,2,3)

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

1: Rotation along the x axis.

2: Rotation along the y axis.

3: Rotation along the z axis.

fsi_6dof (int 0,1)

1: Activates the ability to move the structure in up to six DoF. Each of the six DoF can be activated independently of each other with the following control options: dgf_x, dgf_y, dgf_z, dgf_ax, dgf_ay, dgf_az, (described below). Note that any combination of the six DoF is valid regardless of the translational DoF or rotational DoF. One can obtain the same results as using the "fsi" option or the "rfsi" option by activating only one of the 6 DoF.

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 are active, the parameter uses the loose coupling algorithm.

1: When either fsi or rfsi are 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".

body_mass (double)

Mass of the structure.

body_inertia_x, body_inertia_y, body_inertia_z (double)

Moment of inertia with respect to the center of rotation defined by x_r, y_r, and z_r.

body_alpha_rot_x, body_alpha_rot_y, body_alpha_rot_z (double)

Damping coefficient for the rotational DoFs.

body_alpha_lin_x, body_alpha_lin_y, body_alpha_lin_z (double)

Damping coefficient for the translational DoFs.

body_beta_rot_x, body_beta_rot_y, body_beta_rot_z (double)

Elastic spring constant for the rotational DoFs.

body_beta_lin_x, body_beta_lin_y, body_beta_lin_z (double)

Elastic spring constant for the translational DoFs.

x_r, y_r, z_r (double)

Center of rotation in the rotational DoFs.

fall_cyll_case (int 0, 1)

Option for the falling cylinder test case.

Wave Generation Options

wave_momentum_source (int 0, 1, 2)

When using the level set method, the parameter activates the wave generation module, based on the method of Guo and Shen (2009).

0: Waves are not generated.

1: Waves are read from an external file.

2: A single monochromatic wave is generated.

wave_angle_single (double)

In the case that wave_momentum_source is equal to 2, the parameter sets the wave initial phase.

wave_K_single (double)

In the case that wave_momentum_source is equal to 2, the parameter sets the wave number.

wave_depth (double)

In the case that wave_momentum_source is active, the parameter sets the water depth. This parameter is used in the dispersion relation to compute the wave frequency.

wave_a_single (double)

In the case that wave_momentum_source is equal to 2, the parameter sets the wave amplitude.

wave_ti_start (int)

Time step to start applying the wave generation method.

wave_skip (int)

This option is used in the case that wave_momentum_source is equal to 1 (waves are imported from external files). The external data file to be imported, which has been generated using an external code, may have a time step size not equivalent to the time step of the present code simulation. Usually the time step size in the wave data is significantly larger than that from the present simulation ($\Delta t_{wave-data} = \Delta t_{simulation} \times b$, where b is an integer). By setting wave_skip to b, the code will import a new wave data file every wave_skip time steps, and since wave_skip=b, the time of the simulation will match the time of the wave data.

wind_skip (int)

This option is equivalent to wave_skip but for importing the inlet wind profile when the option air_flow_levelset is equal to 2.

wave_start_read (int)

This parameter is useful for restarting the simulation in the case that wave_momentum_source is equal to 1 (waves are imported from external files). The code will import the wave file corresponding to the wave time step wave_start_read, instead of importing the starting wave data file (WAVE_info000000.dat).

wind_start_read (int)

This parameter is useful for restarting the simulation in the case that air_flow_levelset is equal to 2 (inlet wind profile is imported from external files). The code will import the wind data file corresponding to the wind time step wind_start_read, instead of importing the first wind data file (WAVE_wind000000.dat), .

wave_recicle (int)

In the case that wave_momentum_source is equal to 1 (waves are imported from external files) and the wave time step reaches this number, the wave time step is recycled to 0, which means that the code will import the wave data corresponding to time step 0 (WAVE_info000000.dat).

wind_recicle (int)

In the case that air_flow_levelset is equal to 2 (wind is imported from external files) and the wave time step reaches this number, the wind data time step is recycled to 0, which means that the code will import the wind data corresponding to time step 0 (WAVE_wind000000.dat).

wave_sponge_layer (int 0, 1, 2)

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

2: Sponge layer method is applied at the four lateral boundaries.

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_ys (double)

Length of the sponge layer applied at the y boundaries.

wave_sponge_y01 (double)

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

wave_sponge_y02 (double)

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

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 a input parameter.

5: Actuator surface model (under development).

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

turbine (int)

Number of wind/hydro-kinetic turbines to be modeled.

reflength (double)

Reference length of the turbine. The code will divide the imported turbine diameter (from the mesh file and Turbine.inp) by this amount.

rotatewt (int 1,2,3)

Direction to which the turbines point to.

1: i direction

2: j direction

3: k direction

r_nacelle (double)

This parameter represent the radius of the turbine nacelle. The code will ignore the rotor effect within this radius.

num_foiltype (int)

Number of foil types used along the turbine blade. VFS requires a description file named FOIL00, FOIL01, ..., for each foil type as described in Section $4.2.6.

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 and normalize when plotting the data.

loc_refvel (int)

Distance upstream of the turbine in rotor diameters where the turbine incoming velocity or reference velocity is computed.

deltafunction (int)

Type of smoothing function within which the pressure due to the rotor is applied.

halfwidth_dfunc (double)

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

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. Lets 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, respectively.

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
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 has to be set to 1, otherwise the code expects the ASCII form.

Any grid generator software that is able to export PLOT3D grids may be suitable for VFS. 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 are referred to 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.

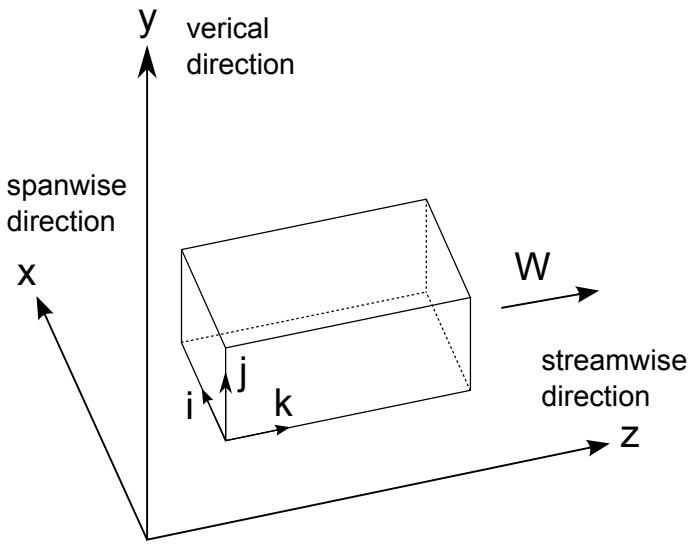


Figure 4.1: Axis orientation

A third format type that VFS 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 are the grid points of the three axis. 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 axis N_x, N_y, and N_z. The values are followed by the three coordinate 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}

...
X_{xN_x} Y_{xN_x} Z_{xX_x}
X_{y1} Y_{y1} Z_{y1}
X_{y2} Y_{y2} Z_{y2}
...
X_{yN_y} Y_{yN_y} Z_{yX_y}
X_{z1} Y_{z1} Z_{z1}
X_{z2} Y_{z2} Z_{z2}
...
X_{zN_z} Y_{zN_z} Z_{zX_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 normal direction of the triangular elements 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 Wave Data External File

The wave generation module allows the incorporation of broadband wave fields with large number of frequency components (wave_momentum_source set to 1). The origin of the wave data can be from a precursor simulation (far-field simulation) using an external wave code or from real measurements. A broadband wave field composed of $NZMOD/2 \times (NYMOD + 1)$ wave frequencies, where $NZMOD/2$ is the number of frequencies in the Z direction and $2 \times (NYMOD + 1)$ is the number of frequencies in the X direction) can be given by the following expression

$$\eta(z,x) = \sum_{k_z=1}^{k_z=NZMOD/2} \sum_{k_x=-NXMOD/2}^{k_x=NXMOD/2} a_{k_z,k_x} cos(k_z{*}PEZ{*}z + k_x{*}PEX{*}x + \theta_{k_z,k_x}) \quad (4.1)$$

where η is the free surface elevation, k_z and k_x indicate the directional wave numbers, a is the wave amplitude, and (θ) is the wave phase. PEZ and PEX are coefficients to scale the wave numbers, which are usually set to 1.

The code will read the wave data file named WAVE_infoXXXXXX.dat, where XXXXXX represents the time step of the wave data. The first line of the wave data file contains the time of the wave, NZMOD, NXMOD, PEZ, and PEX. The second line is where the actual wave data starts. The wave file is as follows:
$time_{wave}$ $NZMOD$ $NXMOD$ PEZ PEX
$a_{k_z=1,k_x=0}$ $\theta_{k_z=1,k_x=0}$
$a_{k_z=2,k_x=0}$ $\theta_{k_z=2,k_x=0}$
...
$a_{k_z=NZMOD/2,k_x=0}$ $\theta_{k_z=NZMOD/2,k_x=0}$
$a_{k_z=1,k_x=1}$ $\theta_{k_z=1,k_x=1}$
$a_{k_z=2,k_x=1}$ $\theta_{k_z=2,k_x=1}$
...
$a_{k_z=NZMOD/2,k_x=1}$ $\theta_{k_z=NZMOD/2,k_x=1}$
$a_{k_z=1,k_x=-1}$ $\theta_{k_z=1,k_x=-1}$
$a_{k_z=2,k_x=-1}$ $\theta_{k_z=2,k_x=-1}$
...
$a_{k_z=NZMOD/2,k_x=-1}$ $\theta_{k_z=NZMOD/2,k_x=-1}$
$a_{k_z=1,k_x=2}$ $\theta_{k_z=1,k_x=2}$
$a_{k_z=2,k_x=2}$ $\theta_{k_z=2,k_x=2}$
...
$a_{k_z=NZMOD/2,k_x=2}$ $\theta_{k_z=NZMOD/2,k_x=2}$
$a_{k_z=1,k_x=-2}$ $\theta_{k_z=1,k_x=-2}$
$a_{k_z=2,k_x=-2}$ $\theta_{k_z=2,k_x=-2}$
...
$a_{k_z=NZMOD/2,k_x=-2}$ $\theta_{k_z=NZMOD/2,k_x=-2}$
...
$a_{k_z=1,k_x=NXMOD/2}$ $\theta_{k_z=1,k_x=NXMOD/2}$
$a_{k_z=2,k_x=NXMOD/2}$ $\theta_{k_z=2,k_x=NXMOD/2}$
...
$a_{k_z=NZMOD/2,k_x=NXMOD/2}$ $\theta_{k_z=NZMOD/2,k_x=NXMOD/2}$
$a_{k_z=1,k_x=-NXMOD/2}$ $\theta_{k_z=1,k_x=-NXMOD/2}$
$a_{k_z=2,k_x=-NXMOD/2}$ $\theta_{k_z=2,k_x=-NXMOD/2}$
...
$a_{k_z=NZMOD/2,k_x=-NXMOD/2}$ $\theta_{k_z=NZMOD/2,k_x=-NXMOD/2}$

As discussed in the control option for the wave module, the wave time step size may no be equal to the time step of the present code simulation. Usually the time step in the wave data is significantly larger than that from the present simulation ($\Delta t_{wave-data} = \Delta t_{simulation} \times b$, where b is an integer). By setting wave_skip to b, the code will import a new wave data file every wave_skip time steps, and since wave_skip=b, the time of the simulation will match the time of the wave data.

If the wave frequencies do not vary in time, one could use a single wave data file by setting the option wave_skip to a very large value.

The wave module also allows the wind field associated with a wave field pre-computed simulation to be incorporated by setting air_flow_levelset to 2. In such a case, the code will read the wave data file named WAVE_windXXXXXX.dat, with XXXXXX representing the time step of the wind data. The first line of the wind data file contains the time of the far-field simulation, the number of grid points in the vertical direction NY, and the number of grid points in the horizontal direction NX. The actual wave data starts in line two. The overall structure of the file is as follows:
$time_{far-field}$ NY NX
$X_{0,0}$ $Y_{0,0}$ $Z_{0,0}$ $U_{0,0}$ $V_{0,0}$ $W_{0,0}$
$X_{0,1}$ $Y_{0,1}$ $Z_{0,1}$ $U_{0,1}$ $V_{0,1}$ $W_{0,1}$
...
$X_{0,NX}$ $Y_{0,NX}$ $Z_{0,NX}$ $U_{0,NX}$ $V_{0,NX}$ $W_{0,NX}$
$X_{1,0}$ $Y_{1,0}$ $Z_{1,0}$ $U_{1,0}$ $V_{1,0}$ $W_{1,0}$
$X_{1,1}$ $Y_{1,1}$ $Z_{1,1}$ $U_{1,1}$ $V_{1,1}$ $W_{1,1}$
...
$X_{1,NX}$ $Y_{1,NX}$ $Z_{1,NX}$ $U_{1,NX}$ $V_{1,NX}$ $W_{1,NX}$
...
$X_{NY,0}$ $Y_{NY,0}$ $Z_{NY,0}$ $U_{NY,0}$ $V_{NY,0}$ $W_{NY,0}$
$X_{NY,1}$ $Y_{NY,1}$ $Z_{NY,1}$ $U_{NY,1}$ $V_{NY,1}$ $W_{NY,1}$
...
$X_{NY,NX}$ $Y_{NY,NX}$ $Z_{NY,NX}$ $U_{NY,NX}$ $V_{NY,NX}$ $W_{NY,NX}$

4.2.6 The Files Required for the Turbine Rotor Model

The Turbine.inp Control File

The Turbine.inp file is a text file containing input parameters used by the rotor model. The file has two lines, the first line is ignored by VFS 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.

```
1 nx_tb-ny_tb-nz_tb-x_c-y_c-z_c-indf_axis-Tipspeedratio-J_rot ...
2 0.0 0.0 1.0 0.0 0.104 0.256 0.36 4.5 14380000 ...
```

nx_tb, ny_tb, nz_tb (double)

Normal direction of the turbine rotor plane.

x_c, y_c, z_c (double)

The turbine rotor initial translation.

indf_axis (double)

Induction factor when using the actuator disk model (rotor_model=1).

Tipspeedratio (double)

Tip speed ratio when using the actuator line model (rotor_model=3,6). The tip-speed ratio (TSR) can adopt negative values which indicate that the turbine is rotating counterclockwise with respect to the stream-wise axis.

J_rotation (double)

Rotor moment of inertia used only when the option "turbinetorquecontrol" is active.

r_rotor (double)

Radius of the turbine rotor.

CP_max (double)

Maximum power coefficient of the turbine; used only when the option "turbinetorquecontrol" is active.

TSR_max (double)

Refers to the maximum TSR of the turbine; used only when the option "turbinetorquecontrol" is active.

angvel_fixed (double)

Rotor rotational speed when the option "fixturbineangvel" is active. The variable angvel_fixed can adopt negative values which indicate that the turbine is rotating counterclockwise with respect to the stream-wise axis.

Torque_generator (double)

Turbine torque, used only when the option "turbinetorquecontrol" is active.

pitch (double)

Pitch angle of the turbine blades when using actuator line or actuator surface models.

CT (double)

Thrust coefficient used with rotor_modelled 4.

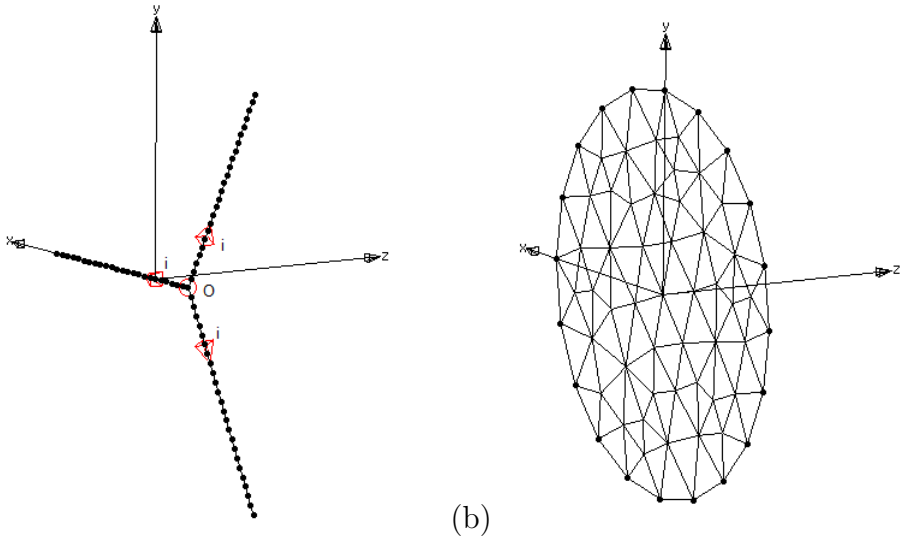
The acldata000 Mesh File

The acldata000 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, as shown in Figure 4.2(a). The ASCII data file uses the SEGMENT format.

In the case of the actuator disc models, 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 turbine center, o, of this mesh could be located directly at the actual position of the turbine, or alternatively, 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 has to point towards the tip of the blade. In the actuator disk model, the direction normal to the rotor has to point towards the direction of the flow.



(a) (b)

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

The Urefdata000 Mesh File

The Urefdata000 file is a UCD formatted triangular mesh equivalent to the actuator disk acldata000 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 Urefdata000 have to match the dimensions of the turbine rotor defined in "turbine.inp".

The code positions the disk upstream of the actual turbine location. At every time step, the velocity of the flow is transferred to each triangular element of this mesh. By adding the velocity at all triangular elements and dividing by the surface of the disk, the code computes the turbine reference velocity (disk average velocity). As an

example, when using the actuator line, the reference velocity is used for determining the TSR of the simulation.

The CD00, CD01, CD02, ..., and CL00, CL01, CL02, ... Files

These files contain the lift and drag coefficients at each profile along the turbine blades when using the actuator line model (rotor_model 3 or 6). 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 Lift/Drag coefficient (column 2) are listed as shown in the example below.

```
1 Airfoil type: DU97-W-300
2 Drag coefficients
3 31
4 7.50925697358677e-001    1.41002221673661e-002
5 2.25610960256727e+000    2.10614663046161e-002
6 4.32615403604048e+000    2.79782769686497e-002
7 6.20902246358924e+000    2.78301653912614e-002
8 ...
9 8.98528141199704e+001    2.13806467538879e+000
```

The FOIL00, FOIL01, FOIL02, ... Files

These files contain the angle of attack and chord length for each profile used along the turbine blades when using the actuator line model (rotor_model 3 or 6). 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 blade section angle of attack in degrees (column 2), and the profile chord length in non-dimensional units or meters (column 3) are listed as shown in the example below.

```
1  Turbine type: Clipper 2.5 MW
2  Airfoil type: Circular
3  7
4  0.000  9.5  2.400
5  2.800  9.5  2.400
6  3.825  9.5  2.385
7  4.950  9.5  2.259
8  6.950  9.5  2.338
9  8.950  9.5  2.339
10 10.800 9.5  2.848
```

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

level.c, distance.c
: Subroutines for simulating two-phase free surface flows with the levelset method

wave.c
: Subroutines for the wave module (also to simulate wind over waves, in which the wind is imported from a far-field simulation)

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 turbine or a hydro-kinetic turbine using the actuator disk or actuator line models

les.c
Subroutines for the turbulent models

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

5.2 The Flow Solver

To describe the basic elements of the flow solver we present a code flow chart of VFS, which displays the order in which the relevant functions of the code are called. This code flow chart corresponds to the most simple case that VFS 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 stream-wise 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 Convergence_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-processesing purposes, then, a second set of functions are 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 are 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 are 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 Level Set Method Module

- **Subroutines for pre-processing**. In this module, the pre-processing basically consists of initializing the level set main variables and setting the free-surface initial condition.

    - **MG_Initial** Initializes the levelset variables. The levelset main variable is named "level[k][j][i]", which is defined as the distance from the current cell center to the closest point of the free surface interface. It adopts a positive value in the water phase and negative value in the air phase. The free surface interface coincides with the zero level.

- **Levelset_Function_IC** Sets the initial position of the free-surface.

- **Subroutines for time advancing**. Here the time-advancing involves solving an advection equation to find the new location of the free surface interface and a mass conserving reinitialization to ensure that the mass within the two phases is conserved.

  - **Advect_Levelset** Solves the levelset advection equation.
    - **Levelset_Advect_RHS** Forms the right hand side terms of the advection equation.
  - **Reinit_Levelset** Solves the mass conserving reinitizlization equation.
    - **Init_Levelset_Vectors** Creates temporary arrays for solving the reinitialization equation.
    - **Solve_Reinit_explicit** Solves the equation in an explicit form.
      - **Distance_Function_RHS** Forms the right hand side terms of the reinitialization equation.
    - **Destroy_Levelset_Vectors** Deletes the temporary arrays.
  - **Compute_Density** Updates the density and viscosity values of the fluid with the values corresponding to the new location of the free surface. The function executes the functions Advect_Levelset and Reinit_Levelsetfree, which update the free surface location.
  - **Compute_Surface_Tension** Applies the surface tension at the free surface.
  - **Levelset_BC** Sets the boundary conditions of the free surface. The function is called both before and after solving the advective and the reinitialization equations.
  - **Calc_free_surface_location** Exports the free surface elevation to the external file FreeSurfaceElev_XXXXXX.dat (XXXXXX refers to the time step) at every "tiout" time steps.

5.3.2 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 μ_t by applying equation (2.26).
- **Formfunction_2** Adds the eddy viscosity term to the right hand side of the momentum equation.

5.3.3 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 its 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 are 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 has 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. Otherwise, this function needs to be called at every time step, if the body is moving, 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.

5.3.4 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_xyz** This function applies a rotation to a given point with respect to a center of rotation in a given direction.
  - **Elmt_Move_FSI_ROT_TRANS** This function applies the structural motion in any of the six DoF to the body mesh. The function is called when the six DoF module is in use. During pre-processing, the function is used to apply an initial motion 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.
  - **Calc_forces_SI_levelset** Computes the force and moments that the fluid imparts to the body. It replaces **Calc_forces_SI** when the level set method is in use.
  - **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_pos_SC_levelset** Solves the EoM in a single translational DoF when the levelset method is active. Needs to be followed by a call to **Elmt_Move_FSI_TRANS**.
  - **Calc_FSI_pos_6dof_levelset** Solves the six DoF EoM. Needs to be followed by a call to **Elmt_Move_FSI_ROT_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).

5.3.5 The Wave Generation Module

- **Subroutines for pre-processing**. In this module the pre-processing consists of initializing the variables for the wave generation method and for specifying the inlet wind from the far-field precursor simulation.

- **Initialize_wave** Initializes the variables for the wave generation method.
- **Initialize_wind** Initializes the variables for importing the wind field from the far field precursor simulation.

- **Subroutines for time advancing**. In this module the code needs to import the wave data and, if necessary, the wind data, at the time steps for which it needs to be updated (every wave_skip and wind_skip time steps, respectively). Then the imported wave/wind information is applied.
  - **WAVE_DATA_input** Sets the water wave field information (amplitude, frequencies, direction angle, ...) to be simulated.
    If the option wave_momentum_source is 1, the function imports the wave information from an external file.
    If wave_momentum_source is equal to 2, the function uses the information given in the control file.
  - **WAVE_Formfuction2** Adds the pressure force in the right hand side of the momentum equation in the form of a source term.
  - **WAVE_SL_Formfuction2** Applies the sponge layer method at the side wall boundaries that are specified in the control file.
  - **WIND_DATA_input** Reads the external file containing information of the wind field of the far-field simulation to be applied at the inlet of the present simulation.
    - **WIND_vel_interpolate** Function to perform bi-linear interpolation to obtain the velocity values at the present fluid mesh which generally differs from the far-field fluid mesh.

5.3.6 The Rotor Turbine Modeling Module

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 modelling module the pre-processing 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 functions 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.

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.

- **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.
  - **rotor_Rot_6dof_fsi**
    If the 6 DoF FSI module is active, this function applies the same motion to the actuator line as was applied to the floating platform.
  - **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.

Chapter 6

Applications

6.1 3D Sloshing in a Rectangular Tank

6.1.1 Case Definition

This test aims to validate the implementation of the level set method which is used in the code to track the motion of the free surface. The test consists of a 3D sloshing of liquid in a tank with the dimension of $L \times L$ and a mean flow depth of D (see Figure 6.1). The initial free surface elevation is of Gaussian shape and is given by

$$\varrho(x, z) = D + \eta_0(x, z), \tag{6.1}$$

where

$$\eta_0(x, z) = H_0 \exp\left\{\left(x - \frac{L}{2}\right)^2 + \left(z - \frac{L}{2}\right)^2\right\}, \tag{6.2}$$

H_0 is the initial hump height, and κ is the peak enhancement factor.

In the computation, the following parameters are used: $L = 20$, $\kappa = 0.25$, and $H0 = 0.1$. The free-slip boundary conditions are applied at all boundaries. This condition is signified by the value 10 in bcs.dat. The number of grid points in the x, y, and z directions are 201, 41, and 201, respectively. Uniform grid spacing of $\Delta x = \Delta z = 0.1$ is employed in the x and z directions, while stretched grid is used in the y direction. The initial hump height (0.1 m) is resolved by approximately five vertical grid nodes. The time step used for the computation is $\Delta t = 0.001s$ and the value of ϵ (free surface thickness) is set to $0.03m$. The solution of the free surface elevation at the center of the tank is given in Figure 6.3.

Further details about the simulation as well as the analytical solution of the problem can be found in Kang and Sotiropoulos [15].

6.1.2 Main Parameters

The main parameters in the control file for setting this test case are listed in Table 6.1.2.

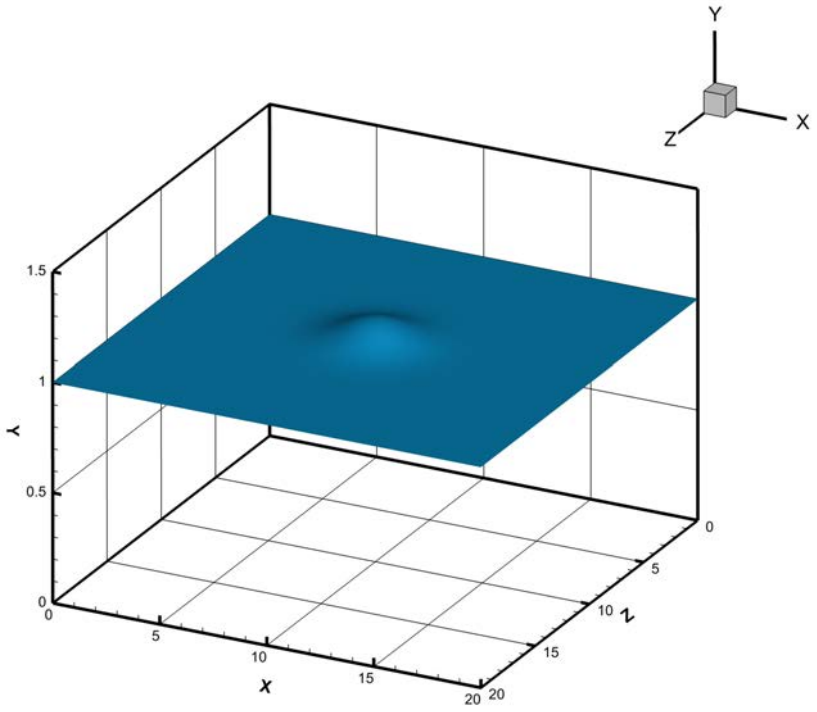


Figure 6.1: Schematic description of the domain and the initial condition of the free surface.

Table 6.1: Parameters in control.dat file for the sloshing case.

Parameter	Option in control file	Value
Time step size	dt [sec]	0.001
Read xyz.dat type mesh	xyz	1
Activate level set method	levelset	1
Set density of water and air	rho0, rho1 [kg/m3]	1000, 1
The viscosity is neglected	inv	1
Set gravity in y direction	gy [m2/s]	-9.81
Activate 3D Sloshing test case option	sloshing	2
Initialize flat free surface at elevation set by level_in_height	level_in	2
Set level set interface thickness	dthick [m]	0.03
Number of level set reinitializations	levelset_it	15
Reinitialization time step size	levelset_tau [sec]	0.05

By activating the sloshing option, two actions are implemented in the code. First, the initial condition of the distance function, and thus the free surface elevation, is set to the one corresponding to the present case. Then, at every time step after the flow solution is updated, it computes the analytical solution of the free surface position at the tank center and exports it along with the computed solution in a file named sloshing.dat described below.

Also note that because the level set method is active, the equations are solved with dimensions.

6.1.3 Validation

The output value for comparison in this test case is the time evolution of the free surface elevation at the tank center. This value can be checked at the post-processed data file containing the whole domain solution, however, this information is only available for the few exported time steps defined by the input option -tio. Another approach that is more convenient for evaluating the surface elevation in the tank center is by checking the output data file (sloshing.dat). This file exports information at every time step and consists of an ASCII data file with four columns. The first column is the simulation time [sec], the second is the computed surface elevation [m] at the tank center, the third is the expected theoretical solution also at the tank center, and the last is the error. If the purpose of running this test case is for validation only, it is not necessary to post-process any flow-field data. The surface elevation at the tank center is plotted in Figure 6.3.

Although Figure 6.3 shows the solution for 60000 time steps (60s), for validation purpose, it is not necessary to run the simulation for that long. With the proposed time step size, between 6000 and 8000 iterations (equivalent to 6 to 8 sec) should be sufficient to demonstrate that the solution is valid. As a reminder, the total number of time steps for which the code runs is specified at the control.dat file with the variable "totalsteps".

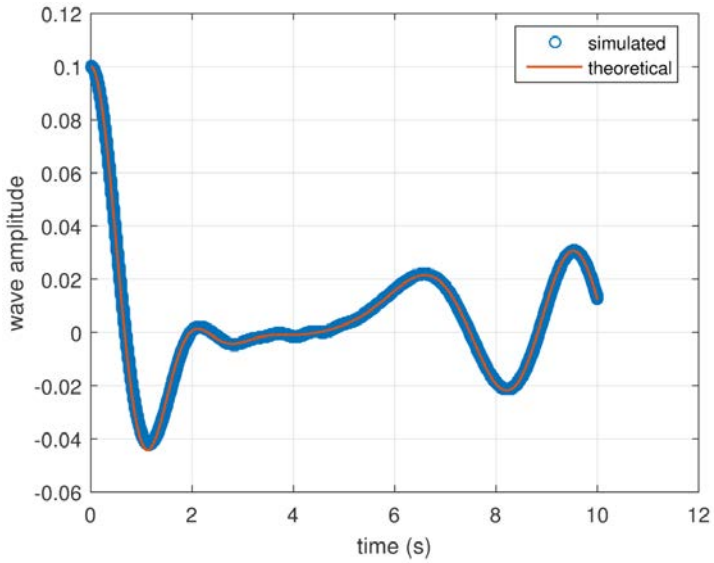


Figure 6.2: Comparison of the computed and analytic free surface elevation at the center of the tank. (symbol: computed solution, solid line: analytical solution).

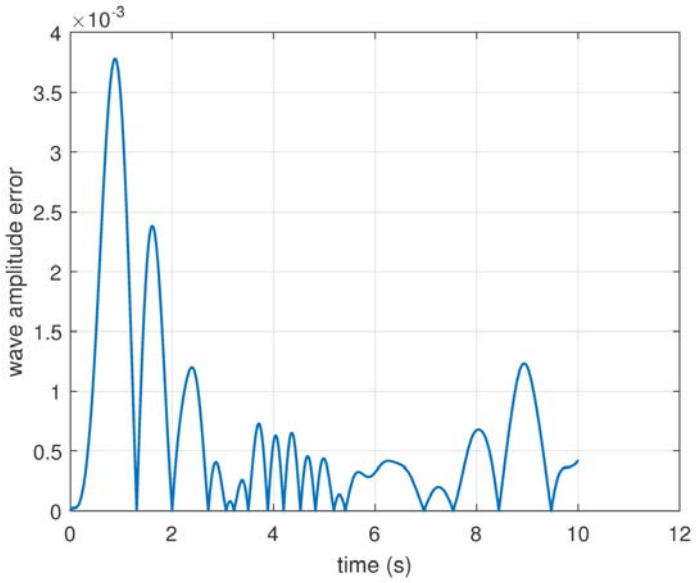


Figure 6.3: Error in the computed free surface elevation.

6.2 2D VIV of an Elastically Mounted Rigid Cylinder

6.2.1 Case Definition

Vortex induced vibration (VIV) of an elastically mounted rigid cylinder is a well-known benchmark case for validating FSI codes. The schematic description of the problem is shown in Figure 6.4. The problem consists of a 2D rigid cylinder of diameter D that is elastically mounted in a uniform flow of velocity U. The cylinder is allowed to move in the direction perpendicular to the flow with one degree of freedom. Additional details can be found in Borazjani, Ge, and Sotiropoulos [16].

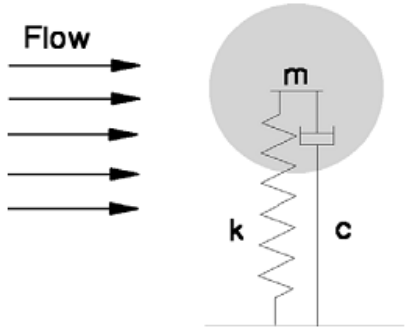


Figure 6.4: Schematic description of the elastically mounted rigid cylinder in the free stream. This figure was reproduced from [16].

A rectangular computational domain with dimensions $32D \times 16D$ is considered.

The cylinder is initially positioned at $8D$ from the inlet and centered. The boundary conditions for the side walls are slip wall (defined by 10 in bcs.dat), uniform flow is prescribed at the inlet (5 in bcs.dat), and a convective boundary condition is applied at the outlet (4 in bcs.dat). A non-uniform grid is used with 281×241 nodes in the streamwise and span-wise directions, respectively. The code requires at least 5 grid nodes (current test case employs 6) in the span-wise direction to carry out a two-dimensional simulation since the code is fully three-dimensional in addition to slip-wall boundary conditions along the span-wise walls. A square box with constant grid spacing of $0.02D$ and 50×50 nodes centered on the cylinder is used. Outside of the box the grid is gradually stretched towards the boundaries.

6.2.2 Main Parameters

The main parameters in the control file for setting this case are listed in Table 6.2. See Borazjani, Ge, and Sotiropoulos [16] for the definition and details of the parameters. In contrast to the previous case the level set method is not employed and the solved equations are all non-dimensional.

Table 6.2: Parameters in control.dat file for the VIV case.

Parameter	**Option in control file**	**Value**
Time step size	dt	0.01
Reynolds number ($RE = U * D/$)	ren	150
Reduced velocity of 6	red_vel	1.0472
Raduced mass of 2	mu_s	0.25
Cylinder damping	damp	0.0
Activate IB method	imm	1
Use of one body	body	1
Activate FSI module	fsi	1
Activate proper degree of freedom	dgf_y	1
Apply an initial translation of the cylinder to the actual position. The cylinder mesh was initially defined at the origin and needs to be translated to the desired location.	y_c, z_c	8.0, 8.0

6.2.3 Validation

The VIV case can be validated by comparing the position of the cylinder in time. Similar to the previous case, there is no need to post-process the flow field data in order to get the cylinder position. The cylinder position is provided at every time step in the FSI_position00 file. The FSI_position00 file 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. The file consists of 10 columns as follows:

$$ti, Y_x, \dot{Y}_x, F_x, Y_y, \dot{Y}_y, F_y, Y_z, \dot{Y}_z, F_z, \tag{6.3}$$

where ti is the time step number, Y is the body position with respect to its initial position Y/D, $\dot{Y}$ is the body velocity, and F is the force that the fluid imparts to the immersed body. For this test case, 6000 time steps should be sufficient.

In the test case folder, a successful run file name _FSI_position00_good_run is provided to quickly validate the case. Figure 6.5 provides a plot showing a comparison with the provided successful run file. The maximum amplitude of the cylinder is approximately 0.49.

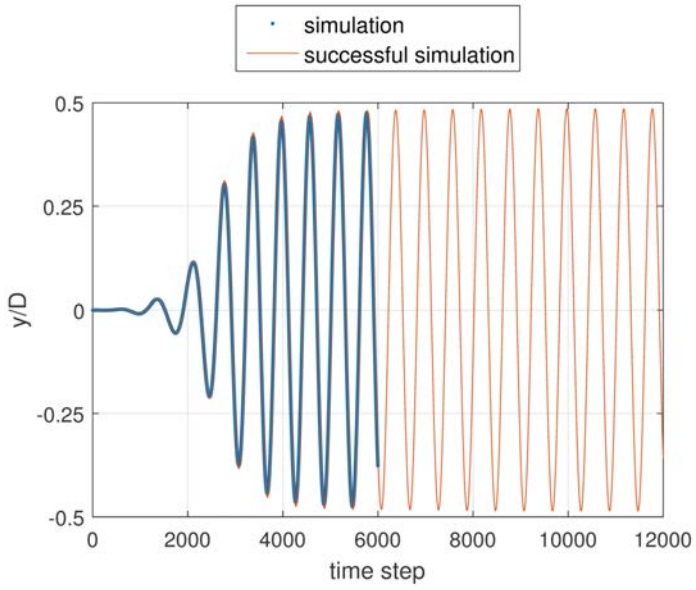


Figure 6.5: Displacement of the cylinder for a successful simulation.

6.3 2D Falling Cylinder (Prescribed Motion)

6.3.1 Case Definition

This test case is to validate the integration of the CURVIB and level set methods. The case involves a body moving across the air/water interface with prescribed motion. Note that the FSI algorithm is not used currently. This test case considers an infinitely long 2D cylinder of radius $R = 1m$ moving with constant downward speed in an infinitely wide domain and crossing the free surface from a gas to liquid phase.

The configuration provided currently is identical to that simulated by Yang and Stern [27] where an identical cylinder, initially positioned above the free surface at a distance $h = 1.25m$, moves downwards with constant velocity of $u = -1m/s$. The liquid and gas densities are $\rho_0 = 1kg/m^3$ and $\rho_1 = 1 \times 10^3 kg/m3$, respectively. Both

the liquid and gas are considered inviscid, while the gravity is set to $gz = -1m/s^2$ and the time is normalized as $T = ut/h$.

The 2D computational domain is $40R \times 24R$ in the horizontal and vertical directions, respectively. A non-uniform grid consisting of an inner region, centered on the cylinder, within which the mesh is uniform and an outer region where the grid is gradually stretched. The inner region is the rectangular domain defined by $[-5, 5]$ and $[-4, 2.6]$ in the horizontal and vertical directions, respectively. Within this inner domain uniform grid spacing is employed along both directions, which is equal to $0.05R$. Outside of this inner domain the mesh is stretched gradually away from the cylinder using the hyperbolic stretching function with a stretching ratio kept below 1.05. The total number of nodes is $360 \times 255 \times 6$. A time step of $0.01s$ and a free surface thickness ϵ of $0.04m$ are used. For more detail about this test case refer to Calderer et al. [2]

6.3.2 Main Parameters

The main parameters in the control file for setting this case are listed in Table 6.3.

Table 6.3: Parameters in control.dat file for the 2D falling cylinder with prescibed motion test case.

Parameter	**Option in control file**	**Value**
Time step size	dt [s]	0.01
Activate level set method	levelset	1
Set density of liquid and gas	rho0, rho1 [kg/m3]	1, 0.001
Set gravity in z direction	gz [m/s2]	-1.0
Thickness of the free surface interface	dthick [m]	0.04
Initialize flat free surface at elevation set by level_in_height	level_in	1
Initial free surface elevation	level_in_height [m]	0.0
Set levelset interface thickness	levelset_it	10
Number of level set reinitializations	levelset_tau [s]	0.05
Activate IB method	imm	1
Use of one body	body	1
Activate fluid structure interaction module	fsi	1
Activate prescribed motion function	forced_motion	1
Activate Falling cylinder case. This option basically prescribes the velocity of the body to be constant and downward.	fall_cyll_case	1
Activate proper degree of freedom	dgf_ay	1
Initial translation of the ib at the right position	z_c	1.25

6.3.3 Validation

The falling cylinder case can be validated by observing the free surface elevation at four instances in time as shown in Figure 6.6. These plots are from the post processed cylinder position (Nv) and free surface (level = 0) at time step T = 125, 250, 375, and 500.

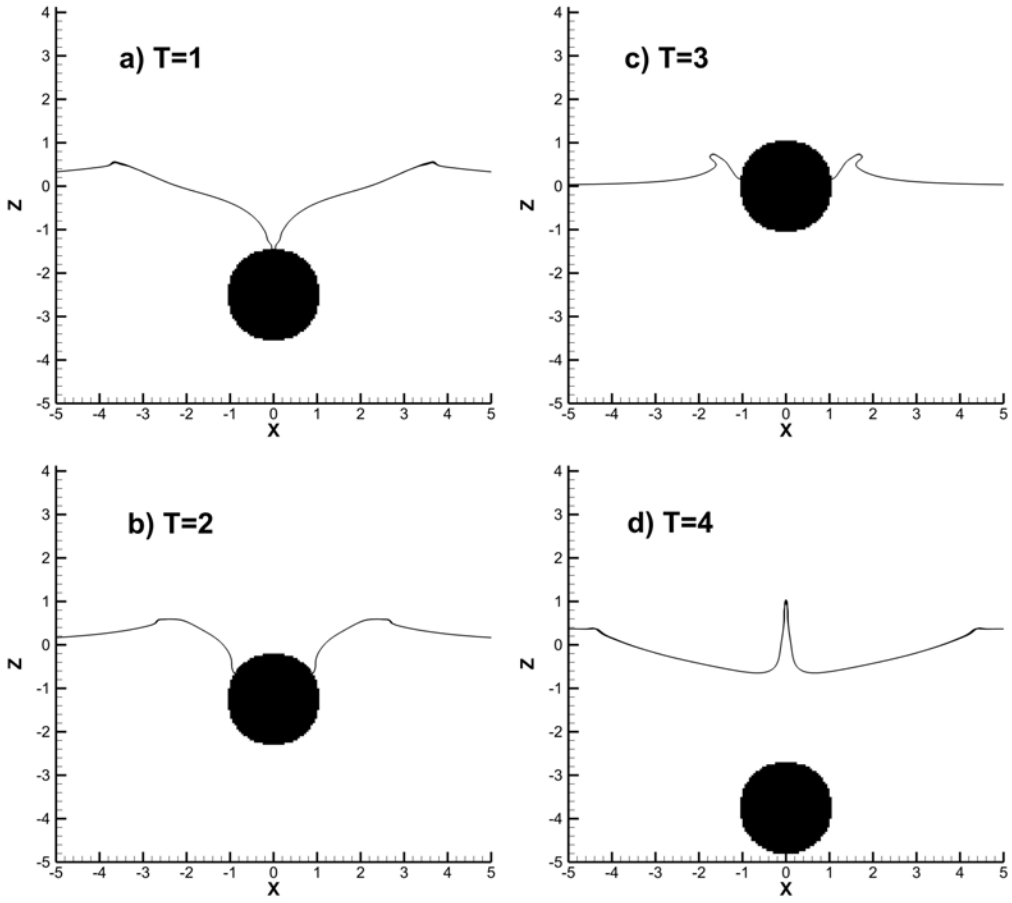


Figure 6.6: Water entry of a horizontal circular cylinder moving with prescribed velocity. Free surface position at different non-dimensional times T calculated by the present method.

6.4 3D Heave Decay Test of a Circular Cylinder

6.4.1 Case Definition

To validate the coupled FSI algorithm for simulating complex floating structures a free heave decay test of a horizontal cylinder is provided. This same test case was studied experimentally by Ito [28]. A horizontal circular cylinder of diameter $D = 0.1524m$ and density $\rho = 500kg/m^3$ is partially submerged with its center

positioned $0.0254m$ above the free surface of a rectangular channel (see Figure 6.7 for a schematic representation).

The computational domain is a $27.4m$ long, $2.59m$ wide, and $1.22m$ deep channel with initially stagnant water. The cylinder movement is restricted to the vertical degree of freedom while allowed to oscillate freely. A non-uniform grid is employed with $436 \times 8 \times 261$ nodes in the horizontal, vertical, and span-wise directions, respectively.

The grid is uniform with spacing equal to $0.02D$ in a rectangular region centered around and containing the body defined by $[0.3, 0.3]$ in the horizontal direction and $[0.2, 0.2]$ in the vertical direction. The grid is stretched using a hyperbolic function, where the ratio never exceeds 1.05, in the domain outside of the uniform grid region. The interface thickness used is $0.065D$ and the simulation was carried out employing the loose coupling FSI algorithm and a time step size of $0.0005s$.

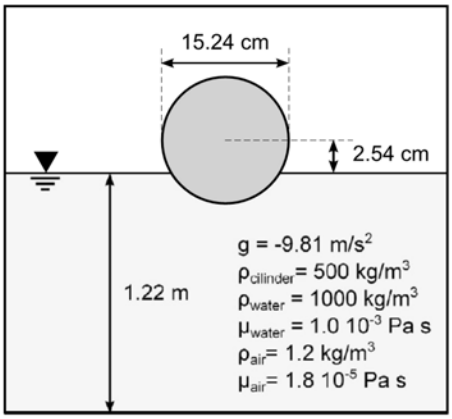


Figure 6.7: Schematic description of the cylinder case. This figure was reproduced from [2].

6.4.2 Main Parameters

The main parameters in the control file for setting this case are listed in table 6.4.

6.4.3 Validation

In the heave decay test case of a horizontal cylinder it is simplest to compare the vertical position of the cylinder in time as shown in Figure 6.8 using the FSI_position00 file's vertical position column as described in the VFS Manual Section $3.3.3 and in the test case of the 2D VIV of an elastically mounted rigid cylinder in section $6.2. An FSI_position00_good_run file is provided for comparison and validation.

Table 6.4: Parameters in control.dat file for the 3D heave decay of a circular cylinder test case.

Parameter	**Option in control file**	**Value**
Time step size	dt [s]	0.0005
Activate Dynamic Smagorinski LES model	les	2
Activate level set method	levelset	1
Set density of water and air	rho0, rho1 [kg/m3]	1000, 1.0
Set viscosity of water and air	mu0, mu1 [Pa s]	1e-3, 1.8e-5
Set gravity in z direction	gy [m/s2]	-9.81
Thickness of the free surface interface	dthick [m]	0.006
Initialize flat free surface at elevation set by level_in_height	level_in	2
Initial free surface elevation	level_in_height [m]	0.0
Set level set interface thickness	levelset_it	20
Number of level set reinitializations	levelset_tau [s]	0.01
Activate IB method	imm	1
Use of one body	body	1
Activate fluid structure interaction module	fsi	1
Activate proper degree of freedom	dgf_y	1
Mass of the cylinder	body_mass [kg]	23.63
Initial translation of the ib at the right position	y_c [m]	0.0254

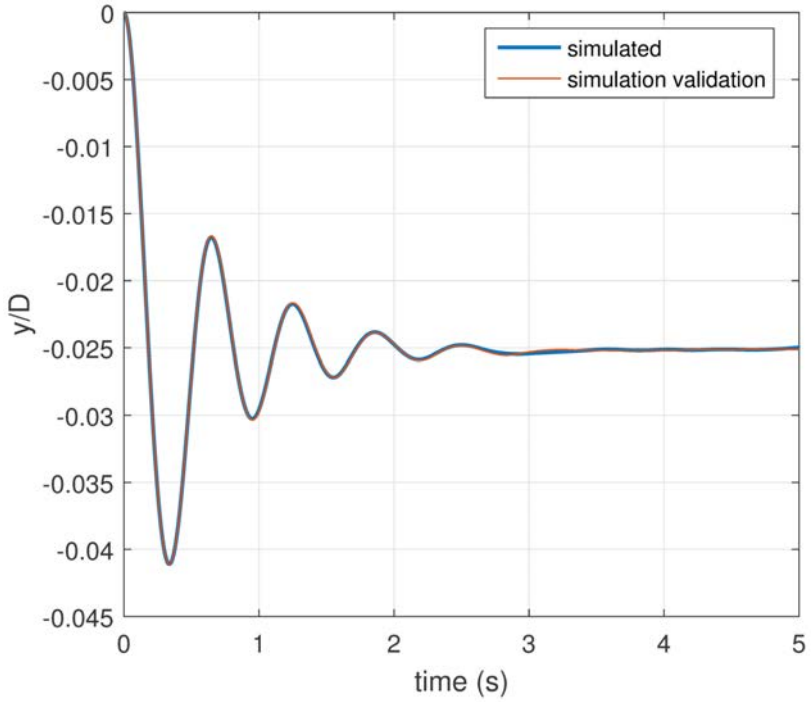


Figure 6.8: Normalized position of the cylinder in time computed with the present code.

6.5 2D Monochromatic Waves

6.5.1 Case Definition

The simulated generation and propagation of a progressive monochromatic wave in a 2D rectangular channel is used to validate the wave generation method of Guo and Shen [29] implemented in this code. A linear wave of amplitude $0.01m$ and wavelength of $1.2m$, for which the analytic solution is known from linear wave theory, is considered. A two-dimensional domain of length equal to $24m$, water depth of $2m$, air column above the water of $1m$, and gravity equal to $gy = -9.81m/s^2$ is simulated using a non-uniform grid size of 40×177 in the longitudinal and vertical direction, respectively. The grid is uniform in a rectangular region centered on $z = 0$ and spanning $24m$ along the z direction ($[12, 12]$), and containing the free surface along the vertical direction y ($[0.15, 0.15]$). Within this region the grid spacing is 0.06 and 0.005 in the horizontal and vertical directions, respectively, while outside of this region the grid spacing increases progressively with a stretching ratio limited to 1.05.

The time step of the simulation is $0.002s$ with the thickness of the interface set to $0.02m$. The source region is centered on the origin. Sponge layers with length equal to $3m$ are defined at the two ends of the computational domain and slip boundary conditions are implemented at the bottom and top boundaries (10 in bcs.dat).

Details of the method implementation are given in Calderer et al. [3]. The analytical solution is

$$\eta(z,t) = Acos(\omega t - kx), \tag{6.4}$$

where η is the surface elevation, A is the wave amplitude, $k = 2\pi/L$ is the wave number, d is the water depth, and ω is the angular frequency solved for with the dispersion relation as follows:

$$\omega = \sqrt{kgtanh(kd)}. \tag{6.5}$$

6.5.2 Main Parameters

The main parameters in the control file for setting this case are listed in Table 6.5.

6.5.3 Validation

The free surface elevation (height of level = 0 in the post-processed data files) and its comparison with the analytical solution are presented in Figure 6.12. Note that in the source region the computed results are not expected to follow the analytical free surface pattern.

Table 6.5: Parameters in control.dat file for the 2D monochromatic wave test case.

Parameter	**Option in control file**	**Value**
Time step size	dt [s]	0.002
Activate level set method	levelset	1
Set density of water and air	rho0, rho1 [kg/m3]	1000, 1.0
Set viscosity of water and air	mu0, mu1 [Pa s]	1e-3, 1.8e-5
Set gravity in z direction	gy [m/s2]	-9.81
Thickness of the free surface interface	dthick [m]	0.02
Initialize flat free surface at elevation set by level_in_height	level_in	1
Initial free surface elevation	level_in_height [m]	0.0
Set levelset interface thickness	levelset_it	15
Number of level set reinitializations	levelset_tau [s]	0.05
Activate the wave generation method for single wave frequency	wave_momentum_source	2
Time step for which wave generation method begins	wave_ti_start	1
Wave Direction. If 0rad waves travel in x direction	wave_angle_single [rad.]	0
Wavenumber	wave_K_single [1/m]	5.23599
Water depth	wave_depth [m]	2
Wave amplitude	wave_a_single [m]	0.01
Activate wave sponge layer method at the x boundaries for wave suppression.	wave_sponge_layer	1
Length of the sponge layer	wave_sponge_xs [m]	3
Starting coordinate of the first x sponge layer.	wave_sponge_x01 [m]	-12
Starting coordinate of the second sponge layer.	wave_sponge_x02 [m]	9

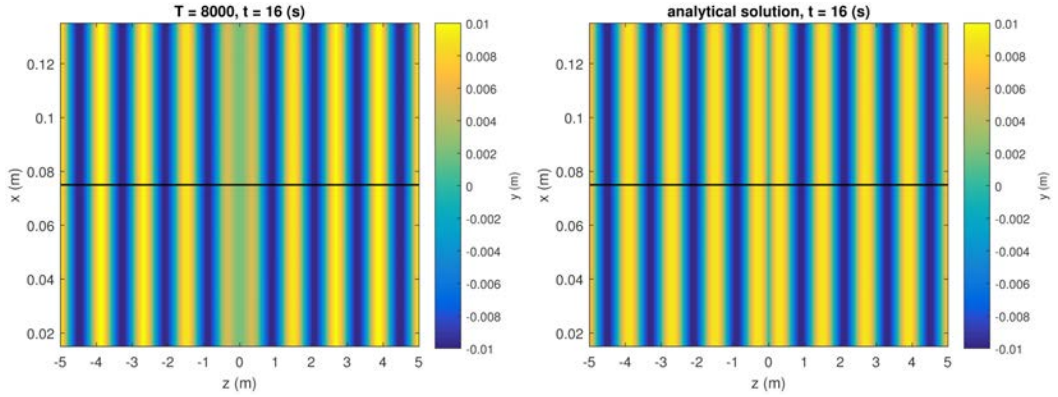


Figure 6.9: Generation of monochromatic waves. Contours of free surface elevation, computed (left) and analytical solution (right).

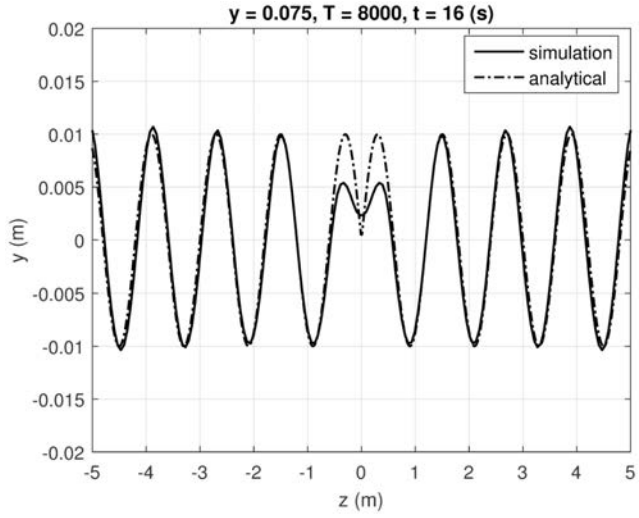


Figure 6.10: Generation of monochromatic waves. Computed and analytical free surface elevation.

6.6 3D Directional Waves

6.6.1 Case Definition

This test case validates the implemented forcing method for wave generation by generating a linear directional wave field in a 3D basin of constant depth. The wave amplitude is $A = 0.01m$, the wavelength is $L = 1.2m$, and the wave direction is $\beta = 25deg$. The domain length is $24m$ ($-12m \leq z \leq 12m$) in the longitudinal direction, $12m$ ($-6m \leq x \leq 6m$) in the span-wise direction, and the depth of water and air is $2m$ and $1m$, respectively. A non-uniform grid size of $121 \times 139 \times 201$ in the x, y, and z directions, respectively, is employed consisting of an inner rectangular region with uniform grid spacing and an outer region within which the mesh is gradually stretched towards the boundaries. The inner region ($-6m \leq z \leq 6m$, $-6m \leq x \leq 6m$, $-0.1m \leq y \leq 0.1m$), which contains the source region and part of the propagated waves, has a constant grid spacing of $0.1m$, $0.1m$, and $0.005m$, in the x, y and z directions, respectively. The source region is centered on $z = 0$ and spans the entire domain along the x direction. The time step is $0.005s$, the interface thickness is $0.02m$, and the gravity is $g = -9.81m/s^2$. The sponge layer method with length $3m$ is applied at the Z boundaries and periodic boundary conditions is applied at the X boundaries. Details of the method implementation are given in Calderer et al. [3]. The analytical solution is

$$\eta(z, x, t) = Acos(\omega t - k_x x - k_z z) \tag{6.6}$$

where η is the surface elevation, A is the wave amplitude, $k = 2\pi/L$ is the wave number, $k_x = kcos(\beta)$, $k_y = ksin(\beta)$, d is the water depth, and ω is the angular frequency solved for with the dispersion relation as follows

$$\omega = \sqrt{kgtanh(kd)}. \tag{6.7}$$

6.6.2 Main Parameters

The main parameters in the control file for setting this case are listed in Table 6.6.

6.6.3 Validation

The free surface elevation (height of level = 0) and its comparison with the analytical solution is presented in Figure **??**.

Table 6.6: Parameters in control.dat file for the 3D directional wave test case.

Parameter	**Option in control file**	**Value**
Time step size	dt [s]	0.002
Activate level set method	levelset	1
Set density of water and air	rho0, rho1 [kg/m3]	1000, 1.0
Set viscosity of water and air	mu0, mu1 [Pa s]	1e-3, 1.8e-5
Set gravity in z direction	gy [m/s2]	-9.81
Thickness of the free surface interface	dthick [m]	0.04
Initialize flat free surface at elevation set by level_in_height	level_in	2
Initial free surface elevation	level_in_height [m]	0.0
Set number of pseudo time steps to solve the reinitialization equation for the level set method	levelset_it	15
Number of level set reinitializations	levelset_tau [s]	0.05
Activate the wave generation method for single wave frequency	wave_momentum_source	2
Wave Direction. If set to $0rad$ waves travel in z direction	wave_angle_single [rad.]	0.5235988
Wavenumber	wave_K_single [1/m]	5.23599
Water depth	wave_depth [m]	2.0
Wave amplitude	wave_a_single [m]	0.01
Activate wave sponge layer method at the x boundaries for wave suppression.	wave_sponge_layer	1
Length of the sponge layer	wave_sponge_zs [m]	1.2
Starting coordinate of the first z sponge layer.	wave_sponge_z01 [m]	-12.0
Starting coordinate of the second sponge layer.	wave_sponge_z02 [m]	10.6

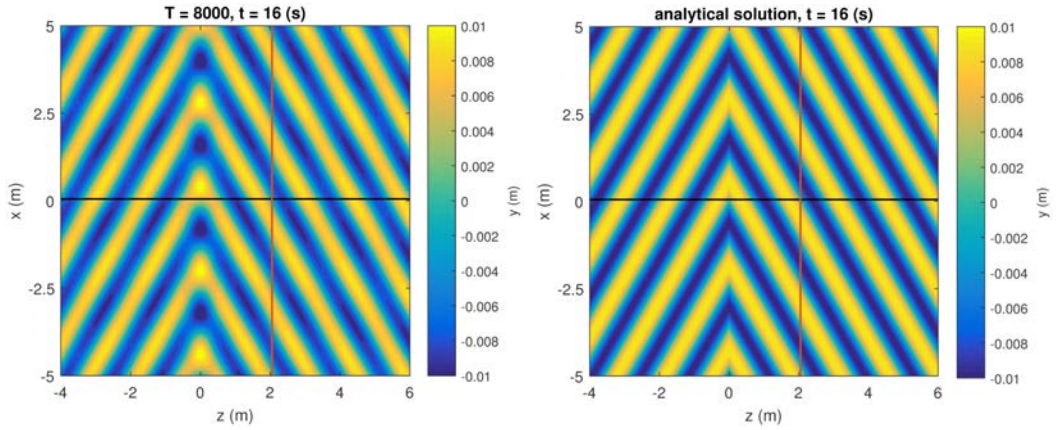


Figure 6.11: Generation of directional progressive waves in a 3D tank. Contours of computed (left) and analytical (right) free surface elevation.

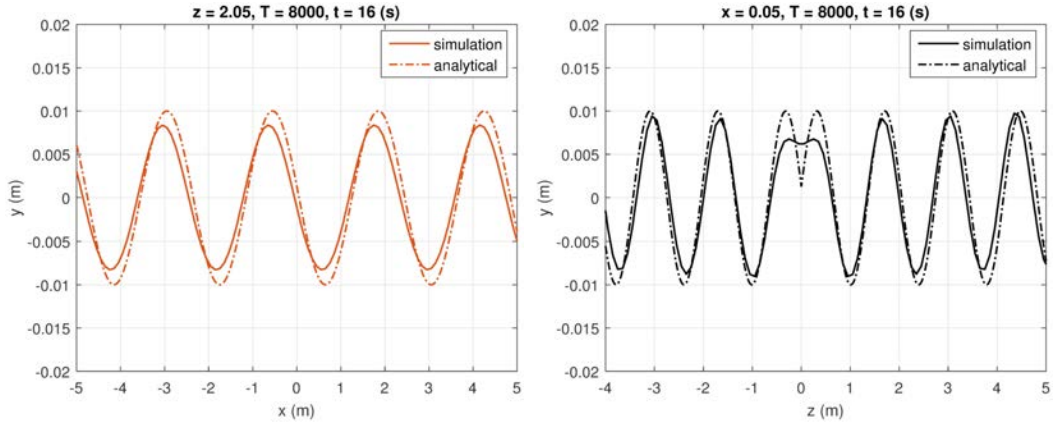


Figure 6.12: Generation of directional progressive waves in a 3D tank. Profiles of surface elevation.

6.7 Floating Platform Interacting with Waves

This test case involves a barge style floating platform model that is installed in the Saint Anthony Falls Laboratory main channel. The channel acts as a wave basin.

The floating platform has a cylindrical shape and is allowed to move in two degrees of freedom, pitch and heave. Rotations are in respect to the center of gravity of the structure. The platform has two small cylindrical masses located underneath which are also considered in this simulation.

In this particular case we study the response of the structure under a given incoming wave frequency. Monochromatic waves of amplitude $0.00075m$ and wave-number 0.8039 are generated at the source region located at $z = 0$. The platform is located at $z = 30m$ and oscillates as a response of the forces induced by waves both in the vertical direction Y and rotating along the X axis. Using the dispersion relation and considering that the water depth is $1.37m$ the wave period is $T = 2.5s$.

The value that we are interested for validating this simulation is the maximum amplitude of the oscillation in both the pitch and heave directions. These can be seen at the files FSI_angle00 and FSI_position00. Figure 6.13 shows the experimental results for several incoming wave frequencies known as Response amplitude operator (RAO). In Figure 6.13 the values are normalized by the incident wave height as follows

$$RAO_{heave} = Y_{max}/H_{wave} \tag{6.8}$$

and

$$RAO_{pitch} = \phi_{max}/H_{wave} \tag{6.9}$$

where Y_{max} is the maximum vertical displacement measured from peak to peak, H_{wave} is the incident wave height, and ϕ_{max} is the maximum rotation angle between two consecutive peaks.

Note that for this case the computed wave amplitude may be slightly inferior to that specified at the control file. This result is due to the fact the waves are in a shallow water regime. This fact will not alter the results as long as the actual simulated wave height is considered in the normalization.

6.8 Clipper Wind Turbine

6.8.1 Case Definition

This test case is for introducing and validating the actuator line model for turbine parameterization. The test case involves the simulation of the Clipper Liberty 2.5MW wind turbine which was built during the EOLOS project and is located in UMore Park, Rosemount, MN. The turbine has a diameter of $96m$ and a hub height of $80m$. Details can be found in [11]. For validation purpose we propose the case with uniform inflow which makes the case simple as it does not require any precursor simulation and the boundary conditions at the top wall, bottom wall and side walls are free slip. Two cases with different TSR, 5.0 and 8.0, are tested.

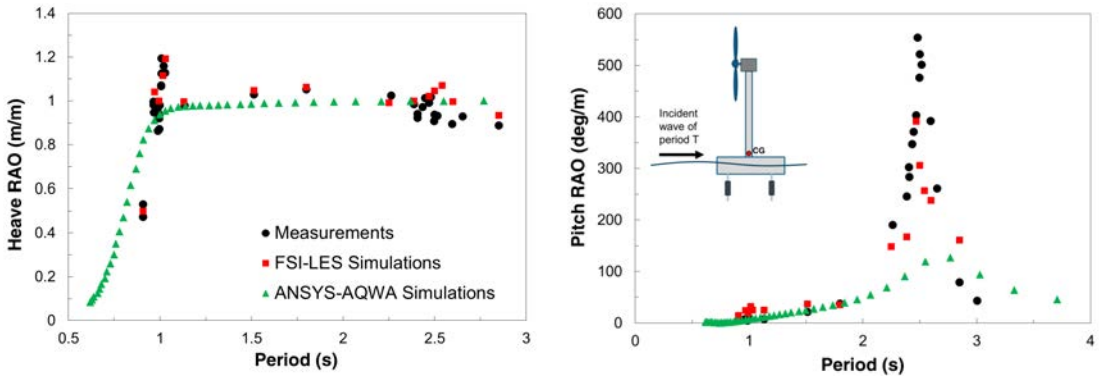


Figure 6.13: Response amplitude operator for the floating turbine.

The simulation is carried in a non-dimensional form being the reference length the turbine diameter. The grid dimensions are 2 units in the y and x directions, and 4 units in the z direction, which corresponds to a dimensional domain of 192m and 384m, respectively. The grid is uniform and the spacing is 0.05 units in all three directions which is equivalent to a grid size of $41 \times 41 \times 81$. Note that the grid file (grid.dat or xyz.dat) has already non-dimensionalized size. The simulation is set with a unit non-dimensional velocity equivalent to a real velocity of 8m/s.

In contrast to the fluid mesh, the actuator line mesh (acldata000) can be constructed with the real turbine dimenions ($96m$) and non-dimensionalized by setting the turbine reference length option "reflength_wt" in control.dat to 96.0. This will divide the actuator line mesh dimensions by "reflength_wt". Alternatively one could generate a rotor mesh already with the non-dimensionalised units and choose "reflength_wt" equals to 1.0.

6.8.2 Main Case Parameters

Since the main solver parameters have already been discussed in previous test cases, only the parameters related to the wind turbine rotor model will be addressed. When using the rotor model the control options for setting the case are located not only in control.c but also in Turbine.inp. The former parameters are summarized in Table 6.7, with the latter in Table 6.8. The parameters in Turbine.inp that are not discussed in the Table 6.8 are not used in the simulation.

This test case requires averaging, and therefore has to be executed in two stages as described in Section $4.2.1. In the first stage the flow is fully developed, while in the second stage the time averaging is performed.

For developing the flow field, it is sufficient to perform 5000 time steps. For time-averaging the flow field, 2000 time steps are enough. 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 (ufield005000.dat, vfield005000.dat, pfield005000.dat, nvfield005000.dat, and cs_field005000.dat) to the value corresponding to time zero (

Table 6.7: Parameters in control.dat file for the Clipper wind turbine.

Parameter	**Option in control file**	**Value**
Time step size	dt [s]	0.0025
Activate the actuator line model	rotor_modeled	6
Number if blades in the rotor	num_blade	3
Number of foil types along the blade	num_foiltype	5
Number of turbines	turbine	1
Reference length of the turbine	reflength_wt	96.
Nacelle diameter	r_nacelle	1.3
Reference velocity of the case. It is only used for dimensionalizing the turbine model output files	refvel_wt	8.0
Distance upstream of the turbine in rotor diameters where the turbine incoming velocity or reference velocity is computed	loc_refvel	0.5
Direction to which the turbine points to	rotatewt	1
Type of smoothing delta function	deltafunc	10
Delta function width in cell units	halfwidth_dfunc	2.0
Activate constant turbine rotation mode	fixturbineangvel	1

Table 6.8: Parameters in Turbine.inp file for the Clipper wind turbine.

Parameter	**Option in Turbine.inp file**	**Value**
Normal direction of the turbine rotor plane.	nx_tb, ny_tb, nz_tb	0.0 0.0 1.0
The turbine rotor initial translation.	x_c, y_c, z_c	0.0 0.0 0.0
Tip speed ration	Tipspeedratio	5
Radius of the rotor corresponding to the acldata000 mesh	r_rotor	48.0
Tip speed ratio when FixTipSpeedRatio option in control.dat is active, otherwise is not used	TSR_max	8.3
Rotor angular velocity when fixturbineangvel option in control.dat is active, otherwise is not used	angvel_fixed	10.0
Angle of pitch of the blades in degrees	pitch[0]	1.0

ufield000000.dat, vfield000000.dat, ...).

Also, when restarting the flow field for averaging, set rstart_turbinerotation as 1 and rename TurbineTorqueControl005001_000.dat as TurbineTorqueControl000001_000.dat.

6.8.3 Validation

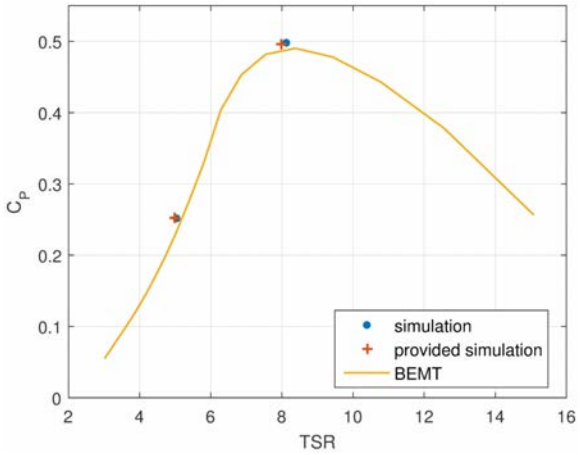


Figure 6.14: Cp coefficient of the Clipper turbine as a function of the TSR.

For validation the case the power coefficient (Cp) of the turbine is compared with the theoretical power coefficient obtained using blade element momentum theory.

A folder named "Tecplotfiles" is provided to assist in generating the Cp comparison. Executing the tecplot macro file ReadSaveCp.mcr a file CP.txt is created with the averaged Cp coefficient. The CP value for the TSR 5 and TSR 8 cases is 0.25 and 0.50, respectively, as shown in figure 6.14.

6.9 Model Wind Turbine Case

6.9.1 Case Definition

This test case is to further demonstrate the validity of the actuator line model by analyzing the turbulence statistics in the turbine wake. The test case consists of a miniature wind turbine simulation with geometry equivalent to the model turbine tested experimentally in the Saint Anthony Falls wind tunnel by [30]. The diameter of the turbine is $0.15m$ and the turbine hub height is $0.125m$. The rotor blade cross section is approximated as a NACA0012, although the real blade geometry is unknown. The tower is neglected and the nacelle is modeled by extending the actuator line effect to the center point of the rotor. The inflow velocity at hub height is $2.2m/s$ and the TSR is 4.1. Reynolds number based on rotor diameter D and the incident velocity at the hub height U_{hub} is 4.2×10^4. For more details about the case and the turbine geometry see [22].

The computational domain dimensions are $30D$ in the streamwise direction, $12D$ in the spanwise direction, and $3D$ in the vertical direction. The turbine is positioned $2D$ after the inlet plane and centered on the transverse direction. The grid is considered uniform along the spanwise and vertical directions. In the streamwise direction,

the mesh is uniform from the inlet to 12D downstream and stretched towards the outlet. Grid size is $201 \times 121 \times 31$, which is equivalent to 10 grid points per turbine diameter.

The velocity profile specified at the inlet is from a pre-computed simulation consisting of a channel flow. An example of such channel flow simulation is provided in the test case in Section $6.10. The channel flow case presented in $6.10 is illustrative of how to prepare fully developed inlet flow conditions, although the case parameters may not exactly coincide with the precursor simulation used for the present simulation.

The bottom wall cannot be resolved with the current grid resolution and is treated with a wall model.

6.9.2 Main Case Parameters

Table 6.9: Parameters in control.dat file for the wind turbine model simulation.

Parameter	**Option in control file**	**Value**
Time step size	dt [s]	0.001
Activate the actuator line model	rotor_modeled	6
Number if blades in the rotor	num_blade	3
Number of foil types along the blade	num_foiltype	2
Number of turbines	turbine	1
Reference length of the turbine	reflength_wt	1.5
Nacelle diameter	r_nacelle	0.0
Reference velocity of the case. It is only used for dimensionalizing the turbine model output files	refvel_wt	8.0
Distance upstream of the turbine in rotor diameters where the turbine incoming velocity or reference velocity is computed	loc_refvel	1.0
Direction to which the turbine points to	rotatewt	1
Type of smoothing delta function	deltafunc	0
Delta function width in cell units	halfwidth_dfunc	2.0
Activate constant turbine rotation mode	fixturbineangvel	1

As in the previous test case, time averaging is required (see Section $4.2.1). For developing the flow field, it is sufficient to perform 5000 time steps. For time-averaging the flow field, 2000 time steps are sufficient. 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 (ufield005000.dat, vfield005000.dat, pfield005000.dat, nvfield005000.dat, and cs_field005000.dat) to the value corresponding to time zero (ufield000000.dat, vfield000000.dat, etc.).

Table 6.10: Parameters in Turbine.inp file for the wins turbine model simulation.

Parameter	**Option in Turbine.inp file**	**Value**
Normal direction of the turbine rotor plane.	nx_tb, ny_tb, nz_tb	0.0 0.0 1.0
The turbine rotor initial translation.	x_c, y_c, z_c	1.0 0.0833 0.2
Tip speed ration	Tipspeedratio	4.0
Radius of the rotor corresponding to the acldata000 mesh	r_rotor	0.025
Tip speed ratio when FixTipSpeedRatio option in control.dat is active, otherwise is not used	TSR_max	8.3
Angle of pitch of the blades in degrees	pitch[0]	1.0

Also, when restarting the flow field for averaging, set rstart_turbinerotation as 1 and rename TurbineTorqueControl005001_000.dat as TurbineTorqueControl000001_000.dat.

6.9.3 Validation

In this test case, vertical profiles of velocity and turbulence statistics at different downstream locations are compared with measured data from the experiment of [30]. To facilitate the creation of these figures a Tecplot macro file named "ExtractLines_3D.mcr", that automatically generates the comparison figures, is provided in the sub-folder "Tecplotfiles". The macro file calls the averaged results file "ResultsXXXXXX.plt" (XXXXXX refers to the time step), which may have to be edited based on the results available. As an example, figure 6.15 shows the vertical profiles of averaged velocity, Reynolds shear stresses, and turbulence intensity $5D$ behind the turbine.

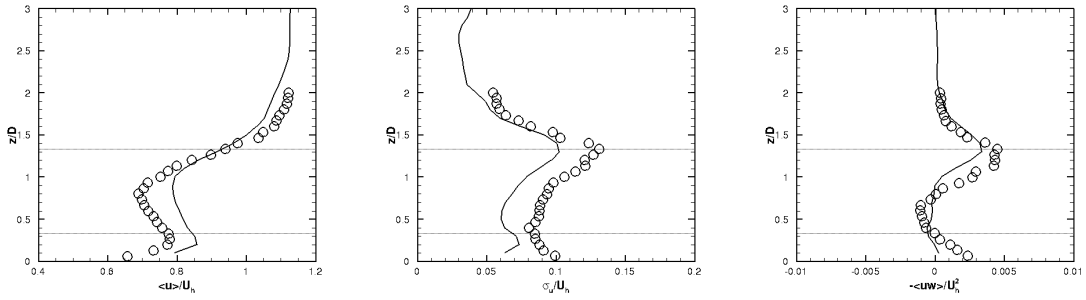


Figure 6.15: Vertical profiles of averaged stream-wise velocity (left), Reynolds stresses (center), and Turbulence intensity (right) at a distance $5D$ behind the turbine.

6.10 Channel Flow

6.10.1 Case Definition

This is the classical channel flow case as extensively described in [31]. A rectangular duct of height $H = 2h$ with uniform flow is considered. This test case can have a dual purpose: (1) to test the wall model at the bottom and/or top walls, or (2) as a precursor simulation to generate fully developed turbulent flow conditions to be fed at the inlet of other cases such as the wind turbine model case in section \$6.9.

The domain is $1.2m$ in the spanwise direction (x), $0.4m$ in the vertical direction (y), and $2m$ in the streamwise direction. A uniform grid is used with size $121 \times 41 \times 61$. Note that the height of $0.4m$ corresponds to the channel half height and slip-wall boundary conditions are used at the top boundary. Fully developed turbulent boundary conditions can be achieved using periodic boundary conditions in the streamwise and spanwise directions and no slip wall boundary condition at the bottom boundaries.

The flow can be characterized with the Reynolds number defined as

$$Re = 2\delta U/\nu, \tag{6.10}$$

where δ is the channel half height and U is the bulk velocity. The flow case can also be defined with the friction Reynolds number defined as

$$Re_\tau = u_\tau \delta/\nu, \tag{6.11}$$

with $u_\tau = \sqrt{\tau_w/\rho}$, and τ_w the shear stress at the wall. The Reynolds number flow can be related to the friction Reynolds number with the following expression

$$Re_\tau \approx 0.09 Re^{.88}. \tag{6.12}$$

In the present simulation the friction Reynolds number is $Re_\tau = 3000$ which using (6.12) corresponds to a Reynolds number flow of $Re = 137900$. With the kinematic viscosity of air taken as $\nu = 1.6 \times 10^{-5}$, using equation (6.10) the flow bulk velocity is 2.7584.

6.10.2 Main Case Parameters

The main parameters used in the present simulation control file are summarized in Table 6.11.

6.10.3 Validation

This test case is validated by comparing the vertical time-averaged profiles of streamwise velocity and turbulence intensity. Time averaging is performed as described in Section \$4.2.1 or as shown in previous test cases. The flow is first executed for about

Table 6.11: Parameters in control.dat file for the channel flow simulation.

Parameter	**Option in control file**	**Value**
Time step size	dt [s]	0.001
This parameter corresponds to $1/\nu$	ren	62500
Dynamic LES modelling active	les	2
Periodic boundary conditions in the streamwise and spanwise directions	kk_periodic, ii_periodic	1, 1
Initial condition set to uniform flow	inlet	1
The inlet flux to obtain a bulk velocity of 2.785. This takes in account the domain cross section area which is $0.48m^2$	flux	1.324
Introduces a random perturbation to the initial velocity field	perturb	1
Activate the wall model at the bottom boundary	viscosity_wallmodel	1
When active it exports to an external file the velocity field at the inlet plane at every time step. First it is set to 0 while the flow is developed. In the second stage the case is reinitialized with this option active.	save_inflow	0, 1
This parameter defines the number of times steps that are stored in an individual file	save_inflow_period	500
Folder where to store the exported velocity files. The code will create in the specified directory a folder named inflow.	path_inflow	"./"

4000 times steps to achieved developed conditions (check the file Kinetic_Energy.dat to ensure that the kinematic energy is stabilized). Then rename the flow field files to the corresponding to time step zero and restart the simulation with the time averaging option active. Also, when restarting, activate the option save_inflow to export the velocity field into the inlet.

A tecplot macro file is provided in the test case folder (ExtractLines_3D.mcr) which extracts vertical profiles of the data from the post-processed data file (Result010000-avg.plt). About 5000 times steps may be sufficient for time averaging although 10000 may provide smoother results. One can use the option ikavg equal to 1 to do space averaging in the streamwise and spanwise directions, when post-processing the average results in addition to the avg equals to 1 option.

Also note that the code output file provides the actual computed values of Re_{tau}

and u_τ. These values are printed at every time step and can be found, at the output file, by searching for the word "Bottom". The two values, are indicated as u* and Re*, respectively. However, the Re* value printed by the code for this case is not the real Re_{tau}. The code assumes that the vertical dimension is 1. So, to obtain the real value of Re_{tau}, the given Re* has to be multiplied by the channel half height δ, equals to 0.4 for this case. These two values are not exact and tend to oscillate in time. Errors of about 10% are within an admissible range.

The averaged stream-wise velocity profile can be compared with the logarithmic law of the wall (see figure 6.16) given by

$$u^+ = \frac{1}{\kappa} \ln(y^+) + 5.0 \tag{6.13}$$

where $\kappa \approx 0.41$ is the Von Karman constant, $u^+ = u/u_\tau$, and $y^+ = yu_\tau/\nu$.

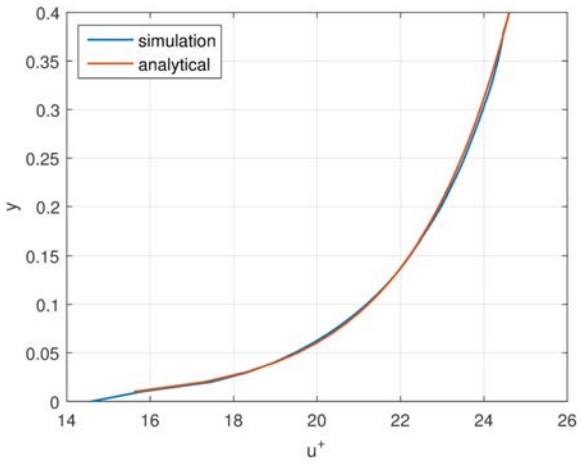


Figure 6.16: Vertical profile of averaged stream-wise velocity.

Bibliography

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

[2] A. Calderer, S. Kang, and F. Sotiropoulos, "Level set immersed boundary method for coupled simulation of air/water interaction with complex floating structures," *Journal of Computational Physics*, vol. 277, pp. 201–227, 2014.

[3] 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," in *Journal of Physics: Conference Series*, vol. 524, p. 012091, IOP Publishing, 2014.

[4] 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.

[5] T. B. Le, *A computational framework for simulating cardiovascular flows in patient-specific anatomies*. PhD thesis, UNIVERSITY OF MINNESOTA, 2011.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] S. J. Osher and R. P. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2003 ed., Oct. 2002.

[13] S. Osher and J. A. Sethian, "Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations," *J. Comput. Phys.*, vol. 79, pp. 12–49, Nov. 1988.

[14] M. Sussman and E. Fatemi, "An efficient, interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow," *SIAM J. Sci. Comput.*, vol. 20, pp. 1165–1191, Feb. 1999.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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 Virtual Wind Simulator (VWiS): LES of turbulent flow past wind turbines/farms: VWiS," *Wind Energy*, pp. n/a–n/a, Aug. 2014.

[23] 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.

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

[25] 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.

[26] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, V. Eijkhout, 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.

[27] J. Yang and F. Stern, "Sharp interface immersed-boundary/level-set method for wave-body interactions," *Journal of Computational Physics*, vol. 228, pp. 6590–6616, Sept. 2009.

[28] S. Ito, *Study of the transient heave oscillation of a floating cylinder*. PhD thesis, MIT, 1971.

[29] X. Guo and L. Shen, "On the generation and maintenance of waves and turbulence in simulations of free-surface turbulence," *Journal of Computational Physics*, vol. 228, pp. 7313–7332, Oct. 2009.

[30] L. P. Chamorro and F. Porté-Agel, "A wind-tunnel investigation of wind-turbine wakes: boundary-layer turbulence effects," *Boundary-layer meteorology*, vol. 132, no. 1, pp. 129–149, 2009.

[31] S. B. Pope, *Turbulent flows*. Cambridge university press, 2000.