The Multiphase Capability of Openfoam CFD Toolbox in Solving Flow Field in Hydraulic Structure

Mohammad Manafpour[1], Hamzeh Ebrahimnezhadian[2]

1- Assistance Prof. in Civil Eng., Water & Hydraulic Structures, Dept. of Civil Eng., Faculty of Eng., Urmia University, Urmia, Iran. Email: m.manafpour@urmia.ac.ir

2- Ph.D. Candidate in Water & Hydraulic Structures, Dept. of Civil Eng., Faculty of Eng., Urmia University, Urmia, Iran. Corresponding Author

Email: h.ebrahimnezhadian@gmail.com

Abstract

The widely known CFD-toolbox "OpenFOAM" is a well-designed C++ library that allows the numerical simulation of various Engineering problems. Owing to its object-orientated structure and the open source code concept, it is very flexible and can be adjusted to very specific problems. Therefore, the code analysis and its manipulation are possible. In general, the library is designed for tackling complex physical problems which can be described with the means of Partial Differential Equations (PDEs). These PDEs are then discretized on the basis of the Finite Volume Method (FVM) in space and with a Finite Differences Scheme in time. However, inappropriate documentation and the lack of a graphical user interface make the usage in the beginning more difficult than most commercial software. The aim of this study is to show the functionalities and capabilities of the toolbox for hydraulic engineering applications, including a short description of the meshing process, boundary condition and the numeric of the solvers. Therefore, a short overview of the applicability and the limitations of the solver "TwoPhaseEulerFoam and InterFoam" which is most commonly used in hydraulic Engineering are presented.

Keywords: OpenFOAM, InterFoam, TwoPhaseEulerFoam, Multiphase, Hydraulic Engineering.

1. INTRODUCTION

Computational Fluid Dynamics is essentially a method for solving a set of partial differential equations that represent a fluid system. These typically include equations representing the principles of conservation of mass, momentum and energy, as well as auxiliary equations to represent other physical phenomena or sources e.g. porous medium, heat exchangers, actuator discs, magnetic fields, etc. Additional transport equations can also be included within a CFD solution to model the transport of a given property such as species concentration in the case of combustion modelling, or turbulent quantities such as the turbulent kinetic energy k and its dissipation rate ε when modelling turbulence using the standard k-ε model.

The three most common methods for numerically solving partial differential equations are the Finite Difference (FDM), Finite Element (FEM) and Finite Volume (FVM) Methods. Common to all of these is that the computational domain is divided into smaller regions with a computational grid, and the differential equations are approximated at discrete points using algebraic equations. Different schemes can be used for approximation and interpolation, usually trading complexity and computational costs for accuracy. For CFD the standard method is FVM, which is rarely used for other purposes. In FVM the domain is divided into control volumes (CV) and the integral form of the conservation equations are applied to each of them. The variables are defined at the centres of the CVs and are interpolated to the CV boundaries. Importantly for CFD, conservation is built into the method. Other reasons for its popularity are that it can be applied to any kind of computational mesh, as the mesh only defines the boundaries of the CVs – instead of the computational nodes as in FDM – and all the variables have a clear physical meaning (Ferziger and Peric, 2002). Open source software is an attractive CFD tool for academic and research purposes. Unrestricted access allows detailed insight into the algorithms used and limitless customization for specific purposes. In CFD applications, the lack of licensing fees makes massively parallel computations economically feasible, provided that the parallelization of the solver is efficient. The widely known CFD-toolbox "OpenFOAM" (Open Field Operation and Manipulation) is a well designed C++ library that allows the numerical simulation of various engineering applications. Through its object-orientated structure it is very flexible and can be adjusted to very specific problems. Since the code is open source, code analysis and manipulation are possible. In general, the library is designed for tackling complex physical problems, which can be described with the means of partial differential equations (PDEs). These PDEs are then discretized on the basis of the Finite-Volume-Method (FVM) in space and with a Finite-

Differences-Scheme in time. With its specific data types for describing the PDEs and the usage of operator overloading, OpenFOAM allows formulating the equations in a way that resembles the mathematical formulation (Weller et al. 1998). Thus, operators like divergence, gradient or laplacian can be simply written as div1, grad and laplacian. A Message-Passing-Interface based parallelization concept is embedded seamlessly which enables highly effective massive parallel computing. As the code is open source, parallel computing with OpenFOAM is limited by the hardware resources available and not by the number of licenses available. But, as the parallelization is based on a domain decomposition approach, the efficiency of parallelization is only given, if the problem size is large enough (Hinkelmann 2003).The class based structure divides the software into the smallest possible units, where each is designed for performing one specific task. Through the object orientated structure the maintenance of the code and development of extensions are generally made easier, as it is possible to add functionality at the outer layers of the code without the necessity to know everything about the inner layers of the libraries. Furthermore, code duplication is avoided, since all parts of the library can be used at multiple positions. With its ingenious concept for the discretization, which is described below, the software allows the usage of arbitrarily shaped cells in the mesh. Besides the official release some forks and adaptions are available. One noteworthy release is the community-driven distribution by the “extend- project”, which aims to “open the OpenFOAM CFD toolbox to community contributed extensions in the spirit of the Open Source development” (www.extend-project.de). Containing various valuable user-developed extensions, it is widely used by many researchers. With the ongoing developments the differences between the two main release branches are growing, therefore switching between different versions is not recommended. The following description refers to the official version 2.2.2. The program package can be installed or compiled for most Linux distributions; versions for Mac OS are available. Running OpenFOAM on Windows is possible but entails several restraints. For post-processing results of OpenFOAM simulations, the open source software ParaView or other common post-processing tools like Tecplot or Gnuplot can be used. With ParaView, even domain decomposed cases (that were calculated in parallel on several CPUs and are stored in separate directories for each domain), can be post-processed without reconstructing the case. In contrary to most CFD programs, OpenFOAM is not delivered with a graphical user interface for performing the pre- and post-processing of the simulations. Settings and data are saved in ASCII text files, where the names of the files and folders have to correspond to a predefined structure. Simulation results are saved in folders named according to the time-step or iteration (Schulze & Thorenz.,2014).

2. FINITE VOLUME DISCRETION IN OPEN FOAM

Since an analytical solution of the PDEs is rarely possible, the solution has to be approximated. For this, the FVM is used here. The discretization process can be can be done as follows:

2.1. EQUATION DISCRETIZATION

To solve the equations that describe the flow transport, a transformation from partial differential equation to linearized algebraic equation is to be performed. For a generic transport equation this can be done as follows (Jasak 1996).

The generic transport equation for the field variable ϕ in integral form can be formulated as:

$$\frac{\partial \rho\varphi}{\partial t} + \nabla.(\rho U\varphi) - \nabla.(\rho\Gamma_\varphi \nabla_\varphi) = S_\varphi(\varphi) \tag{1}$$

All terms are integrated over the time step ranging from t to $t + \Delta t$ and the control volume V_p :

$$\int_t^{t+\Delta t}\left[\frac{\partial}{\partial t}\int_{V_p}\rho\varphi dV + \int_{V_p}\nabla.(\rho U\varphi)dV - \int_{V_p}\nabla.(\rho\Gamma_\varphi\nabla\varphi)\right]dt = \int_t^{t+\Delta t}\left[\int_{V_p}S(\varphi)dV\right]dt \tag{2}$$

By using the Gauss theorem, volume integrals can be converted into surface integrals, which can then be written as sums over the regarded control volume:

$$\int_t^{t+\Delta t}\left[\left(\frac{\partial\rho\varphi}{\partial t}\right)_P + \sum F\varphi_f - \sum(\rho\Gamma_\varphi)_f S.(\nabla\varphi)_f\right]dt = \int_t^{t+\Delta t}\left[S_u v_p + S_p V\varphi_p\right]dt$$

(3)

ρ represents the density, U is the velocity field, through which the variable ϕ is transported through the domain, Γ describes the diffusion coefficient and S includes all source terms. Index P denotes the midpoint of the control volume, index f indicates the value at the surface of the control volume.

The first term accounts for the temporal variation of the generic variable ϕ, the second term describes the convective transport, the third term quantifies the diffusive transport and the right hand side in the equations specifies sources and sinks. The exact way of discretization is defined through the chosen discretization scheme. Since the discretization in OpenFOAM works on a "per operator basis", different schemes (e.g. upwind or different TVD schemes are available) for each operator can be chosen during runtime. As described below, this choice has a large influence on the accuracy of the results and must therefore be handled with care.

The order of accuracy is also of first order, but the time-step restrictions are much less severe. For achieving second order accuracy, the discretization can be blended between the implicit and the explicit scheme. In standard literature an equally weighted blending between implicit and explicit calculation is labelled as Crank Nicolson Method (Ferziger und Perić 2002), however in Open- FOAM the user can blend between a 50:50-weighting and the fully implicit method. That means, the entry ddtSchemes {default CrankNicolson 0;} refers to a fully implicit temporal discretization, whereas ddtSchemes {default CrankNicolson 1;}implies the standard Crank Nicolson scheme with 50 % implicit and 50 % explicit discretization(Schulze & Thorenz.,2014).

3. MESHING

Mesh generation is a mandatory process phase in typical CFD, structural and acoustic simulations and analyses. Although being just requirement for performing calculation it has an important impact on efficiency and accuracy of computation; element or cell shape and size does matter on both computation speed and numerical accuracy.

The mesh is an integral part of the numerical solution and must satisfy certain criteria to ensure a valid, and hence accurate, solution. During any run, OpenFOAM checks that the mesh satisfies a fairly stringent set of validity constraints and will cease running if the constraints are not satisfied. By default OpenFOAM defines a mesh of arbitrary polyhedral cells in 3-D, bounded by arbitrary polygonal faces, i.e. the cells can have an unlimited number of faces where, for each face, there is no limit on the number of edges nor any restriction on its alignment. A mesh with this general structure is known in OpenFOAM as a polyMesh. This type of mesh offers great freedom in mesh generation and manipulation in particular when the geometry of the domain is complex or changes over time. The OpenFOAM toolbox includes a meshing toolbox that allows the generation and manipulation of structured and unstructured meshes. The meshing generation is performed with two main utilities: blockMesh and snappyHexMesh. blockMesh allows the generation of block structured, bodyfitted meshes. On the basis of coordinates, the boundaries of the domain are defined. In the further settings, the names of the boundaries and the size of the cells can be specified. In general, all meshes are created in three dimensions. For a two-dimensional mesh, the mesh gets only one cell in the third dimension and the faces normal to the third dimension get a specific boundary condition ("empty"). With the snappyHexMesh utility the mesh can be adapted to complex external geometries. The mesh generation is based on a blockMesh grid and consists of three successive steps (see Figure 1):

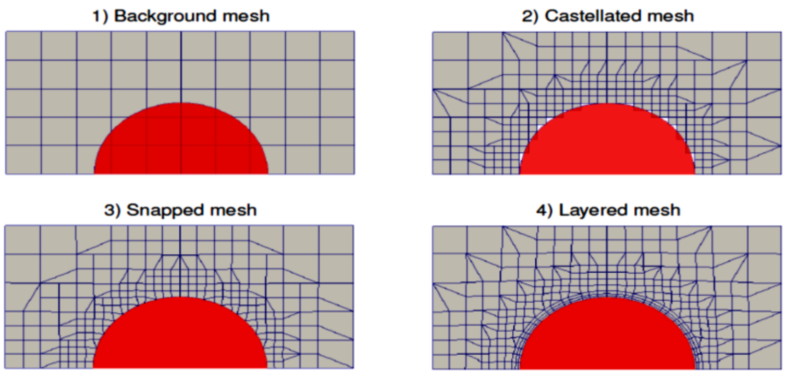

Figure 1. Grid generation steps with the native OpenFOAM meshing tools: a) blockMesh grid, b) snappyHexMesh castellated, c) snappyHexMesh snapped, d) snappyHexMesh addLayers

Castellated mesh generation: In the first step, the input mesh is locally refined according to the predefined settings. Cells close to the surface of the external geometries and cells in predefined regions are refined by orthogonal division of the block structured cells. Afterwards, all cells that overlap the external geometry are deleted from the mesh. This results in the so called castellated mesh.

Snapping: In the second step, cells that are intersecting with the geometry surface are deformed such that the mesh fits the external geometry. This process is performed in an iterative manner to assure that the shape of the surface resembles the external geometry's surface and fulfills the required mesh quality parameters. Cells on the inside are deformed, too, in order to avoid too distorted cells.

Addition of boundary layers: The third step adds boundary layers to the mesh. This is done by first shrinking the existing mesh and then inserting stretched block structured cells at the surface of the external geometry. These layers have the purpose of improving the modelling of boundary layer flow. If local head losses are dominating the flow and friction losses can be neglected, the creation of boundary layers can be avoided. As the creation of boundary layers in snappyHexMesh often results in a decreased mesh quality, the necessity of boundary layers is to be thought over before the simulation setup. The snappyHexMesh grid generation can be performed in parallel. This is advantageous, if large meshes are to be created. With OpenFOAM it is also possible to use meshes that are not created with the native tools. For the conversion of these meshes, several tools are available (e.g. fluentToFoam, starToFoam etc. (Schulze & Thorenz.,2014).

4. CASE SETUP

After the meshing, solver settings, calculation settings and boundary conditions have to be defined. For that, a case must contain at least the following subfolders: 0, constant, system. In the 0 folder, files with the boundary and initial conditions of all primary variables are stored. The constant folder contains all constant parameters like gravity, surface tension and the mesh data. In the system folder, the chosen discretization schemes, iterative solving methods and parameters that control the solution process like the time-step size or the maximum Courant number are defined. The system folder will be read during runtime, which means that a change of settings in this folder is immediately effective. Since appropriate choice of boundary conditions and discretization schemes are crucial for a successful simulation setup, more details on these topics are given below.

4.1. BOUNDARY CONDITIONS

In OpenFOAM boundary conditions are defined per variable at each boundary patch. It is possible to define generic type boundary conditions like fixed values (Dirichlet) or fixed normal gradients (Neumann), additionally derived boundary condition types are available that combine several generic conditions with additional restrictions. It is necessary that the conditions for various variables at one patch match, so that the boundary conditions result in a physically sound combination.

4.2. DISCRETIZATION SCHEME AND ITERATIVE SOLVERS

In the fvSchemes file the user has to define, which discretization schemes are to be used. As explained before, the discretization of the equation is based on a per operator basis. This means that one discretization scheme can be chosen for each operator. With the choice of the schemes, stability and accuracy of the calculation are strongly influenced. Therefore, a lot of effort should be put into the choice of the schemes. The chosen schemes in the official tutorials of the toolbox are mostly chosen such that the simulation runs fast and stable (i.e. using upwind schemes) whereas for real world applications higher accuracy (higher order schemes) is needed in most cases.

The fvSolution file specifies the iterative solvers and limiters that should be used for solving the PDE systems. The choice for the iterative solvers strongly affects the simulation time but only has small influence on the actual results, if the error tolerances of the different solvers are set to the same order.

5. MULTIPHASE SOLVERS

In OpenFOAM, multiple multiphase solvers are available. These are namely:
- InterFoam, LTSinterfoam InterDyMFoam: Solvers that are based on the Volume-of-Fluid Method (as explained below). These are useful for simulations, where a sharp and well-defined interface between the fluid phases exists.
- TwoPhaseEulerFoam, multiphaseEulerFoam, and multiphaseInterFoam: Solvers that are based on the Eulerian-Eulerian approach (for detailed information refers to Rusche, 2002).

For hydraulic engineering applications, the first three are of most relevance, whereas the last three are rather used for applications where small-scale flow regions (i.e. as in chemical engineering) are considered. In the following the interFoam and TwoPhaseEulerFoam solver is analysed in detail. The interFoam solver is made for simulating flow of two inmiscible fluids, which share an interface that is significantly larger than the cell size. The continuous fluid regions should contain a multitude of cells. In this approach, only one mass and one momentum conservation equation is solved for both fluids. For that, density and viscosity of both fluids are averaged according to the volume fractions in the cell. Mass and momentum transfer between the phases is neglected (Schulze & Thorenz., 2014). On the other hand, two incompressible fluid phases with one phase dispersed are solved using TwoPhaseEulerFoam solver. Both the phases are described using the Eulerian conservation equations and thus it is referred as Euler-Euler model. Each of the phases is treated as a continuum in this approach.

5.1. INTERFOAM SOLVER BASIC EQUATIONS

The Volume of Fluid (VoF) method is used for tracking the position and shape of the interface through solving an additional advection equation for the volume fraction in each cell. Together with the Navier- Stokes equations this results in the following set of equations that has to be solved for each cell during each time step:

$$\nabla.U = 0 \tag{4}$$

$$\frac{\partial \rho U}{\partial t} + \nabla.\left(\rho U U\right) = -\nabla p_{-rgh} + \left[\nabla.\left(\mu \nabla U\right) + \nabla U.\nabla \mu\right] + \rho.g + \int_{s} \sigma k \delta\left(x - x_s\right) n dS\left(x_s\right) \tag{5}$$

$$\frac{\partial \alpha}{\partial t} + \nabla.\left(U\alpha\right) + \nabla.\left(U_r \alpha\left(1-\alpha\right)\right) = 0 \tag{6}$$

With ρ = density; U = velocity; t = time; $p_{-rgh} = p - \rho g \cdot x$ = modified pressure, obtained by subtraction of the hydrostatic pressure from the pressure; x =special position vector ; μ= dynamic viscosity; g=gravity; S = interface between the phases; σ = surface tension coefficient; κ = curvature of the surface; δ = dirac delta; $(x - x_s)$ = distance from the considered point to the surface; n = normal vector on the interface; α = volume fraction of the first phase (water); U_r = compressive velocity counteracting numerical diffusion. The first equation accounts for the conservation of mass, the second represents the momentum Conservation equation and the third describe the transport of the volume fraction α. For counteracting numerical diffusion in the VoF equation, an artificial compression velocity U_r is introduced. This term creates a flux in the direction of the gradient of the volume fraction $\nabla\alpha$, e. g. the smeared interface is artificially compressed. It only acts within the zone of the interface as it becomes 0 where α = 0 or α = 1.

5.2. TWOPHASEEULERFOAM SOLVER BASIC EQUATIONS

TwoPhaseEulerFoam is a two-fluid, Euler-Euler method solver for incompressible two- phase turbulent flows. It has been included in OpenFOAM releases since version 1.3 with small variations. The twoPhaseEulerFoam is based on a solver called bubbleFoam, which is a result of Henrik Rusche's work for his PhD thesis "Computational Fluid Dynamics of Dispersed Two-Phase Flows at High Phase Fractions" (2002) and on further development (Weller, 2002, 2005) of the algorithm developed for the BRITE II project at Imperial College. TwoPhaseEulerFoam differs from bubbleFoam by the addition of models for particle- particle interaction. Two alternative approaches are included. Firstly, with a particle normal force, i.e. a powder modulus model as suggested by Gidaspow et al. (1983; 1985) and Bouillard et al. (1989) and secondly, using the kinetic theory for granular flow (KTGF) model.

Two incompressible fluid phases with one phase dispersed are solved using this solver. Both the phases are described using the Eulerian conservation equations and thus it is referred as Euler-Euler model. Each of the phases is treated as a continuum in this approach.

The Eulerian conservation equations are used to describe both the phases in the two-fluid model. Each of the phase is treated as continuum and inter-penetrating each other and is represented by averaged equations. The equations implemented in OpenFOAM solver are given here. The equations for two fluid modeling approaches in OpenFOAM are implemented from "Computational Fluid Dynamics of Dispersed Two-Phase flows at high phase fractions" by Henrik Rusche. The averaged inter-phase momentum transfers term accounts for the transfer of momentum between the two phases. The averaged momentum and continuity equations for each phase φ can be written as:

$$\frac{\partial \alpha_\varphi \bar{U}_\varphi}{\partial t} + \nabla.\left(\alpha_\varphi \bar{U}_\varphi \bar{U}_\varphi\right) + \nabla.\left(\alpha_\varphi \bar{R}_\varphi^{eff}\right) = -\frac{\alpha_\varphi}{\rho_\varphi}\nabla \bar{p} + \alpha_\varphi g + \frac{\bar{M}_\varphi}{\rho_\varphi} \quad (7)$$

$$\frac{\partial \alpha_\varphi}{\partial t} + \nabla.\left(\alpha_\varphi \bar{U}_\varphi\right) = 0 \quad (8)$$

Where the subscript φ denotes the phase, α is the phase fraction, $\bar{R}_\varphi^{eff}$ the combined Reynolds (turbulent) and viscous stress, $\bar{M}_\varphi$ is the averaged inter-phase momentum transfer term. Combining the second equation for the two phases when φ = a and φ = b yields the volumetric continuity equation and can be formulated as an implicit equation for pressure.
The inter-phase momentum transfer can be calculated by adding the forces acting on the Dispersed Phase particles. The drag, lift and the virtual mass forces are considered as the main contribution. The other forces such as Basset or history forces are neglected.
The volumetric continuity equation is:

$$\nabla.\bar{U} = 0 \quad \text{Where: } \bar{U} = \alpha_a \bar{U}_a + \alpha_b \bar{U}_b \quad (9)$$

This equation is recast into a pressure equation:

$$\bar{U} = \alpha_a \bar{U}_a + \alpha_b \bar{U}_b \left[\left[\nabla\left(\left(\alpha_{af}\left(\frac{1}{\rho_a (A_a)_D}\right)_f + \alpha_{bf}\left(\frac{1}{\rho_b (A_b)_D}\right)_f\right)\nabla[\bar{p}]\right)\right]\right] = \nabla.\left(\alpha_{af}\varphi_a^* + \alpha_{bf}\varphi_b^*\right) \quad (10)$$

The phase continuity equation solved is:

$$\frac{\partial \alpha_\varphi}{\partial t} + \nabla.\left(\alpha_a \bar{U}\right) + \nabla.\left(\bar{U}_r \alpha_a (1-\alpha_a)\right)0 \quad (11)$$

where $\bar{U}$ is as given above and U_r is the relative velocity between the phases.

5.3. DISCRETIZATION WITH APPROPRIATE SCHEMES

Since the numerical solution of advection equations tends to produce numerical diffusion and thereby smear discontinuities, a special solution technique for the VoF equation is available in OpenFOAM. To guarantee a bounded solution with sharp interface between the phases, the total variation dimishing scheme "interGamma" (Jasak 1996) is, used mostly in combination with the flux corrected transport approach "MULES" (MUltidimensional Limiter for Explicit Solutions) (Damian 2013). However, other advection schemes for the flux calculation can also be used. The experience showed that for free-surface hydraulic engineering simulations, the choice of the divergence schemes for the VoF equation has significant impact on the quality of the results. In particular, the usage of the Minmod scheme for the discretization on the convection term div (alpha, phi) and the interface Compression for the artificial compression term div (alpha, phir) showed good results. For the discretization of the momentum transport div (U, rho) it is absolutely necessary to use discretization schemes of higher order, as first order upwind discretization smears the results (Schulze & Thorenz.,2014).

5.4. PRESSURE-VELOCITY COUPLING

For the mass and the momentum conservation equation incompressibility is assumed, therefore the densities of the phases do not change over time. The equations of the system are strongly coupled; therefore, a special solution algorithm is needed. In the interFoam solver, a segregated approach is adopted for the pressure velocity coupling. For this, the PISO (Pressure Implicit Splitting of Operators (Issa 1986)) algorithm is applied. To avoid the "checkerboarding" phenomena, an interpolation method "in the spirit of the Rhie Chow method" is used (Peng Kärrholm 2006). In particular, the complete solution procedure of the interFoam solver consists of the following steps (Damian 2013), when the standard PISO algorithm is set: 1 Solve VoF equation on basis of the old velocity field from the previous time step. This gives new values for the volumetric phase fraction and the dependent density in each cell. 2 Perform the momentum predictor step, where the new momentum is calculated on basis of the previous velocities which are interpolated as fluxes from the cell midpoints to the cell faces, the old pressure values and the new density distribution from step 1. 3 The predicted (2.) or the old velocities from the previous timestep are used to set up a linear equation system for solving the new pressure values. 4 The new pressure is calculated. 5 In the last step the predicted velocities are corrected, so that continuity is fulfilled. The momentum predictor step (2.) is not mandatory, but it can reduce the calculation time in some cases. The last two steps are performed several times within one time-step; the number of cycles is user-defined and can be set in the fvSolution file for each case. The fact, that the volume fraction is solved on the basis of the old velocity field, results in a solution where the variables are "temporally staggered". This could be avoided when an additional correction of the volume fraction variable would be performed after the PISO algorithm. Since the PISO algorithm is based on the assumption, that the time-step size is small (Co<1) (Jasak 2006), the temporal offset can be neglected. Alternatively, the SIMPLE or PIMPLE algorithm can be selected instead. PIMPLE (in other software this is called SIMPISO) is an extension of the SIMPLE algorithm which performs only one momentum corrector step but applies a more detailed treatment for the pressure gradient arising from non-orthogonality similar to the PISO algorithm (Aguerre et al. 2013). When the non-orthogonal correctors are set to unity, PIMPLE reduces to the PISO algorithm.

5.5. Boundary conditions for hydraulic engineering applications

In the standard toolbox of OpenFOAM, the available generic boundary conditions are often not practical for hydraulic engineering investigations. Only with some work around it is possible to set a fixed water level or a specific water inflow condition, when simulating with the VoF-solver interFoam and Eulerian-Eulerian solver TwoPhaseEulerFoam. This was the motivation to develop a set of boundary conditions for hydraulic engineering purposes. In particular, a boundary condition for a fixed water level (to be used primarily at the downstream side of a model) and one for a fixed flow rate of water independent from the water level (to be used at the upstream side) were developed amongst others at the Federal Institute for Waterway Engineering and Research. A more detailed description of this code extension can be found in Thorenz und Strybny (2012). A release of the code to the public is planned in the near future.

6. Applicability and Limitations of InterFoam and TwoPhaseEulerFoam

As with every CFD simulation, the accuracy and credibility of the results is highly dependent on the grid resolution. With a too coarse grid important effects of the flow can get lost. For some aspects, models can be applied, which compensate the lost information. In hydraulic engineering turbulence and free-surface modelling is essential. Due to the program structure of OpenFOAM, all available turbulence modeling approaches can be combined with almost every solver. OpenFOAM's VoF-solver interFoam is a valuable tool for many hydraulic engineering investigations. Through the volume of fluid approach it is suitable, when the free surface between water and air is of interest. However, the user must be aware, that the interface between the fluids can only be represented with a limited accuracy that is mainly dependent on the size of the cells. Bubbles or droplets, which are smaller than the control volumes, cannot be represented appropriately. Therefore, air entrainment or bubble transport and detrainment cannot be modelled in most hydraulic engineering simulations (Schulze & Thorenz.,2014).

TwoPhaseEulerFoam is Solver for a system of 2 incompressible fluid phases with one phase dispersed, e.g. gas bubbles in a liquid including heat-transfer, Therefore, air entrainment or bubble transport and detrainment can be modelled in most hydraulic engineering simulations. In general, the computation time and the stability of the interFoam and TwoPhaseEulerFoam simulations are strongly dependent on the mesh size and quality, the chosen numerical schemes and matrix solvers.

7. Conclusions

In this paper, the CFD software OpenFOAM is introduced. It has been highlighted that OpenFOAM is a powerful tool which offers an extensive range of features to model fluid flow, from incompressible to compressible flows, as well as multi-phase flows. OpenFOAM is distributed with a large number of models, including laminar and turbulence models within the RANS, LES and DNS simulation/modelling frameworks.

The included meshing tool and the solvers allow the modelling of complex systems, which can be post processed with tools like ParaView, Gnuplot or similar software. Due to the sophisticated structure of the library massive parallel computing is possible, which is almost only limited to the available hardware resources. The experience shows, that the above described interFoam solver is a suitable tool for typical hydraulic engineering questions based on the investigation of water levels, velocities, pressures etc. The named solver is capable of simulating turbulent two-phase flow, with long, stretched water-air interfaces. On the other hand, TwoPhaseEulerFoam solver is a suitable for a system of two incompressible fluid phases with one phase dispersed; therefore, air entrainment or bubble transport and detrainment can be modelled in most hydraulic engineering simulations. The quality of the results is mainly dependent on the grid quality and the chosen discretization schemes. In comparison to many commercial CFD software packages OpenFOAM is very sensitive concerning the grid quality; it is therefore advisable to put effort into the grid generation. Further, the user should be aware, that the chosen discretization schemes have a great influence on the stability of the calculation and the quality of the results. As usual in numerical simulations, the definition of the domain extent, the definition of the boundary condition as well as the adjustment of all other settings is also crucial for getting plausible results. As only little user-friendly documentation and no graphical user interface exist, the start with Open-FOAM might be not as easy as with commercial CFD software.

8. REFERENCES

1. Aguerre, H. J.; Damián, S. M.; Gimenez, J. M.; Nigro, N. M. (2013): Modelling of compressible fluid problems with Open- FOAM using dynamic mesh technology XXXII, pp. 995–1011. Available online at http://www.cimec.org.ar/ojs/index.php/mc/article/viewFile/4404/, checked on 5/29/2014
2. Damian, S. M. (2013): An Extended Mixture Model for the Simultaneous Treatment of Short and Long Scale Interfaces. Doctoral Thesis. Universidad Nacional del Litoral, Santa Fe, Argentinia. Facultad de Ingeneria y Ciencias Hidricas. Available online at https://docs.google.com/file/d/0B2lpdhG-Zh05Y0ZzOHVLb3lGekk/edit?pli=1, checked on 5/22/2014.
3. Ferziger, J. H.; Perić, M. (2002): Computational methods for fluid dynamics. 3rd, rev. ed. Berlin, New York: Springer.
4. Gisen, D. (2014): Generation of 3D mesh using snappyHexMesh featuring anisotropic refinement and near-wall layers for hydro power dam tailwater. In: Proceedings of the 11th International Conference on Hydroscience & Engineering (ICHE) 2014. Hamburg.
5. Hinkelmann, R.-P. (2003): Efficient Numerical Methods and Information-Processing Techniques in Environment Water. Habilitation. University of Stuttgart, Stuttgart. Institute of Hydraulic Engineering.
6. Issa, R.I (1986): Solution of the implicitly discretised fluid flow equations by operator-splitting. In Journal of Computational Physics 62 (1), pp. 40–65. DOI: 10.1016/0021-9991(86)90099-9.
7. Jasak, H. (1996): Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows. Doctoral Thesis. Imperial College of Science, Technology and Medicine, London, Great Britain. Department of Mechanical Engineering. Available online at http://powerlab.fsb.hr/ped/kturbo/OpenFOAM/docs/HrvojeJasakPhD.pdf, checked on 5/22/2014.
8. Jasak, H. (2006): Numerical Solution Algorithms for Compressible Flows. Lecture Notes. Zagreb.
9. Peng Kärrholm, F. (2006): Rhie-Chow interpolation in OpenFOAM. Chalmers University of Technology. Available online at http://www.tfd.chalmers.se/hani/kurser/OS_CFD_2007/rhiechow.pdf, checked on 5/22/2014.
10. Rusche, H. (2002): Computational fluid dynamics of dispersed two-phase flows at high phase fractions. Doctoral Thesis. Imperial College of Science, Technology and Medicine, London, Great Britain. Department of Mechanical Engineering. Available online at http://powerlab.fsb.hr/ped/kturbo/OpenFOAM/docs/HenrikRuschePhD2002.pdf, checked on 5/22/2014.
11. Saad, Y. (2003): Iterative methods for sparse linear systems. 2nd ed. Philadelphia: SIAM.
12. Thorenz, C.; Strybny, J. (2012): On the numerical modelling of filling-emptying systems for locks. In. Hinkelmann R.-P, Liong,
13. Schulze, L; Thorenz, C.(2014): The Multiphase Capabilities of the CFD Toolbox OpenFOAM for Hydraulic Engineering Applications. ICHE Conference, , Hamburg , Lehfeldt & Kopmann
14. Y. Savic, D., Nasermoaddeli, M., Daemrich, K.-F., Fröhle, P., Jacob, D. (Eds.): Proceedings, 10th International Conference on Hydroinformatics HIC 2012. International Conference on Hydroinformatics. Hamburg, July 14-18 2012. Hamburg: TuTech Innovation.
15. Weller, H. G.; Jasak, H.; Fureby, C. (1998): A tensorial approach to computational continuum mechanics using object-oriented techniques. In Journal of Computational Physics 12 (6), pp. 620–631. Available online at
16. http://www.foamcfd.org/Nabla/main/PDFdocs/CompInPhys98.pdf, checked on 5/22/2014.