ICHE 2014, Hamburg - Lehfeldt & Kopmann (eds) - © 2014 Bundesanstalt für Wasserbau ISBN 978-3-939230-32-8

The Multiphase Capabilities of the CFD Toolbox OpenFOAM for Hydraulic Engineering Applications

L. Schulze & C. Thorenz
Federal Waterways Engineering and Research Institute, Karlsruhe, Germany

ABSTRACT: The widely known CFD-toolbox "OpenFOAM" is a well-designed C++ library that allows the numerical simulation of various engineering applications. Through its object-orientated structure and the open source code concept it is very flexible and can be adjusted to very specific problems. However, little documentation and the lack of a graphical user interface make the usage in the beginning more difficult than most commercial software. This contribution is aiming to show the functionalities and capabilities of the toolbox for hydraulic engineering applications, including a short description of the meshing process, the numerics of the solver as well as a short overview of the applicability and the limitations of the solver "interFoam", which is most commonly used in hydraulic engineering applications. Most information described in the paper stems from years of experience with OpenFOAM at the Federal Waterways Engineering and Research Institute (BAW), where the tool is used to investigate complex questions concerning waterway structures like locks, weirs, fish passages or the interaction between ship and waterway.

Keywords: CFD, OpenFOAM, interFoam, Multiphase, Hydraulic engineering

1 THE TOOLBOX OPENFOAM

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 div[1], 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. The official version of OpenFOAM is distributed under the GNU

[1] Remark: All terms, adapted directly from the OpenFOAM code terminology are written in monospace font.

license by ESI/OpenCFD (www.openfoam.org). 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 programme 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 programmes, 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.

2 FINITE VOLUME DISCRETISATION

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 divided into two parts: the geometric and the equation discretization.

2.1 *Geometric discretization*

For discretizing the space, the computational domain is divided into a finite number of volumes, where the solution is to be calculated. The amount of volumes (=computational cells) influences the calculation effort and is therefore limited by the available resources. With decreasing cell size, the accuracy of the results and the computational effort is increased. Therefore, it is necessary to choose the grid size very carefully.

The mesh in OpenFOAM is specified through cells with an arbitrary number of faces, which are defined by a number of vertices (points). For each face an owner and a neighbour cell is specified. In the input data, lists of the points, faces and cells are stored either as plain text or as binary files. In an additional file, the boundary faces and their boundary specification are assigned. With this simple but effective concept of saving the mesh data in combination with the applied numerical concept for equation discretization, the computational cells can be of arbitrary shape, which is a big advantage for modelling complex geometries. The mesh should fulfil typical quality considerations: The cells should not overlap, the faces of the cells should be flat and neighbouring cells should fulfil orthogonality, i.e. the face normal vectors should be on the connection line between the centres of the neighbouring cells. Since OpenFOAM uses a collocated variable arrangement, all primary variables are stored at the cell-centres.

2.2 *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:

$$\frac{\partial \rho\phi}{\partial t} + \nabla \cdot (\rho U\phi) - \nabla \cdot \left(\rho\Gamma_\phi \nabla\phi\right) = S_\phi(\phi) \tag{1.}$$

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

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

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:

$$\int_t^{t+\Delta t}\left[\left(\frac{\partial \rho \phi}{\partial t}\right)_P + \sum F\phi_f - \sum(\rho\Gamma_\phi)_f \boldsymbol{S}\cdot(\nabla\phi)_f\right]dt = \int_t^{t+\Delta t}\left[S_u v_p + S_p V \phi_P\right]dt \quad (3.)$$

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 mid-point 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.

Apart from the choice of the spatial discretization schemes, the user has to define the temporal discretization method. OpenFOAM offers the choice between Euler implicit, Euler explicit or Crank-Nicolson discretization. With the explicit discretization, all variables in the equation system stem from the previous time-step. With this approach, the result is of first order accuracy and the time step has to be limited strongly to ensure stability. In OpenFOAM, an automatic restriction based on the CFL-criterion is available. In contrast, the implicit method implies the usage of all variables from the next time-step, resulting in a large linear system. 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 OpenFOAM 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.

The discretized equation results in a sparse matrix system with the form: [A] [ϕ] = [R], which can be solved using iterative solution techniques. The sparse matrix [A] contains all coefficients of the dependent variable, which are stored in the column vector [ϕ]. [R] holds all right-hand-side terms (Jasak 1996). For the solution of the sparse matrix system, OpenFOAM provides a library of several iterative, linear solvers. This includes e.g. the "Conjugate Gradients" class of solvers (i.e. PCG, BiCG, BiCGStab) or, as state of the art, an "Algebraic Multigrid" solver (e.g. GAMG). For reducing the number of necessary iterations preconditioners of different type can be used. For more detailed information the interested reader is referred to standard references about linear solvers as Saad (2003). The choice of the matrix solver and the usage of preconditioners and smoothers has a significant influence on the calculation time, where the applicability of the solvers is mainly dependent on the matrix size. The results however should be comparable, when the same residual sizes are set.

3 MESHING

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, body-fitted 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):

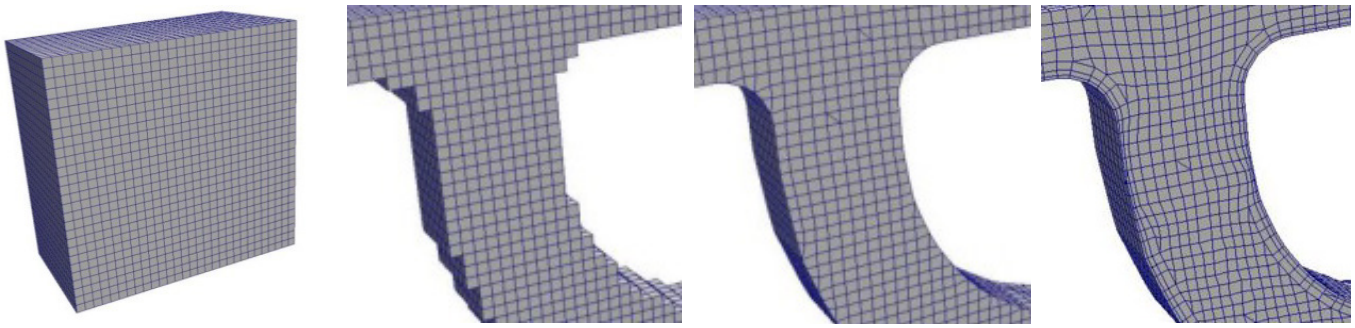

Figure 1. Grid generation steps with the native OpenFOAM meshing tools: a) blockMesh grid, b) snappyHexMesh castellated, c) snappyHexMesh snapped, d) snappyHexMesh addLayers

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 fulfils 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.). To the experience of the authors, the usage of externally created grids can be difficult, since they often do not fulfil the required quality standards of OpenFOAM.

For hydraulic engineering applications like the modelling of stretched river parts it is often useful to create anisotropic cells which are stretched in the flow direction and small in vertical direction. For that the `blockMesh` can be created in an anisotropic manner. Yet this is not always advisable because this affects the complete basic grid. After the grid generation the quality of the mesh can be reviewed with the utility `checkMesh`. This tool generates statistics for the mesh (amount of points, cells, outer bounds etc.) and checks the quality of the cells. Thereby the following quality parameters are considered: orthogonality, skewness, cell aspect ratio. This check compares all cells with an optimal hexahedral cell, with an orthogonality of 1, no skewness and a cell aspect ratio of 1. If cells do not meet the cell quality criteria, predefined in the `meshQualityDict`, the mesh check fails. It is recommendable to carefully examine the critical cells, since they can severely influence the stability and accuracy of the simulation results. Since OpenFOAM only offers restricted methods for the manipulation of single cells after the mesh generation, grids with severe quality restraints should be generated again. Bad quality cells mostly evolve in the range of complex external geometries as well as in the boundary layer region. Experience has proven that avoidance of boundary layers or addition of refinement regions in the bad mesh quality zones helps improving the mesh quality. Since quality controls and layer settings in `snappyHexMesh` can only be set for the whole domain, it can sometimes be useful to generate multiple grids separately and combine them afterwards. For merging multiple meshes into one, the `mergeMesh` and `stitchMesh` utilities can be used. When a mesh is to be transformed, scaled or rotated, the tool `transformPoints` can be used. A more detailed description of the meshing process with OpenFOAM's native utilities can be found in the paper of Gisen (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 con-

tains all constant parameters like gravity, surface tension and the mesh data. In the `system` folder, the chosen discretisation 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 discretisation 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, multiphaseInterFoam: Solvers that are based on the Eulerian-Eulerian approach (for detailed information refer 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 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.

5.1 *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:

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

$$\frac{\partial \rho U}{\partial t} + \nabla \cdot (\rho U U) = -\nabla \mathrm{p}_{-rgh} + [\nabla \cdot (\mu \nabla \mathrm{U}) + \nabla \mathrm{U} \cdot \nabla \mu] + \rho \cdot g + \int_{\mathrm{S}} \sigma \kappa \delta(x - x_s) n d\mathrm{S}(x_s) \tag{5.}$$

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

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 describes 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 $\alpha = 0$ or $\alpha = 1$.

5.2 *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 of the authors 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 discretisation on the convection term `div (alpha, phi)` and the `interfaceCompression` 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.

5.3 *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.

5.4 *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 workarounds it is possible to set a fixed water level or a specific water inflow condition, when simulating with the VoF-solver interFoam. 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

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 programme structure of OpenFOAM, all available turbulence modelling 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.

For parallelization the mesh is divided into parts by domain decomposition, for which the computations can then be performed simultaneously. A 0-halo layer approach is used for the information exchange at the domain boundaries (Jasak 2006). When choosing the number of domains, it is necessary to consider, that the information exchange between the domains can be costly compared to the actual computation time if the domains are chosen too small. An optimal number of cells per computation unit is amongst other things dependent on the hardware and has to be found via trial and error for every high performance computer. At the BAW typically decompositions that result in a workload of 50 000 to 100 000 cells per core are used for standard cases.

In general, the computation time and the stability of the interFoam simulations are strongly dependent on the mesh size and quality, the chosen numerical schemes and matrix solvers. For simulations, where parts of the structure or the grid have to be moved during runtime, the interDyMFoam solver can be used. interDyMFoam solves the same set of equations as the interFoam solver but additionally includes all necessary features for dynamic simulations. However, the setup of a case with moving parts is not trivial and there is hardly any official documentation available.

7 CONCLUSIONS

Most information described in this paper stems from years of experience with OpenFOAM at the BAW, where the tool is used to investigate complex questions concerning the flow in and around waterway structures like locks, weirs, fish passages or the interaction between ship and waterway. 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.

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 ex-

tent, 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 OpenFOAM might be not as easy as with commercial CFD software. However, once the concept of setting up cases is understood, the handling of the OpenFOAM is not more costly than other CFD tools. Further, the open code concept allows the introduction of new boundary conditions and the adaption of the code.

REFERENCES

Aguerre, H. J.; Damián, S. M.; Gimenez, J. M.; Nigro, N. M. (2013): Modelling of compressible fluid problems with OpenFOAM 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

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.

Ferziger, J. H.; Perić, M. (2002): Computational methods for fluid dynamics. 3rd, rev. ed. Berlin, New York: Springer.

Gisen, D. (2014): Generation of a 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.

Hinkelmann, R.-P. (2003): Efficient Numerical Methods and Information-Processing Techniques in Environment Water. Habilitation. University of Stuttgart, Stuttgart. Institute of Hydraulic Engineering.

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.

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.

Jasak, H. (2006): Numerical Solution Algorithms for Compressible Flows. Lecture Notes. Zagreb.

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.

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.

Saad, Y. (2003): Iterative methods for sparse linear systems. 2nd ed. Philadelphia: SIAM.

Thorenz, C.; Strybny, J. (2012): On the numerical modelling of filling-emptying systems for locks. In. Hinkelmann R.-P, Liong, 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.

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 http://www.foamcfd.org/Nabla/main/PDFdocs/CompInPhys98.pdf, checked on 5/22/2014.