V European Conference on Computational Fluid Dynamics
ECCOMAS CFD 2010
J. C. F. Pereira and A. Sequeira (Eds)
Lisbon, Portugal,14-17 June 2010

OPENFOAM® : AN EXETER PERSPECTIVE

Gavin R. Tabor*

*CEMPS, University of Exeter,
Harrison Building,
North Park Road,
Exeter EX4 4QF, UK.
e-mail: g.r.tabor@ex.ac.uk

Key words: Computational Fluid Dynamics, Open Source Software, Code Development

Abstract. *OpenFOAM is an open source CCM code with capabilities which rival and exceed available commercial codes. In this paper I discuss some of the strengths of OpenFOAM and present a review of some of the work carried out at the University of Exeter using it; with particular reference to constitutive equation modelling, LES, turbulence modelling in swirling flows and free surface modelling for Urban Drainage applications.*

1 INTRODUCTION

OpenFOAM is an open source computational continuum mechanics (CCM) code. Originally developed at Imperial College as FOAM (Field Operation And Manipulation) by Henry Weller and various associates [1] and then as a commercial product by the company Nabla Ltd, it is now released open source under the Gnu General Public License by the company OpenCFD Ltd [2]. [1] It makes use of the Finite Volume technique, in which the computational domain is divided into a collection of small non-overlapping but packing polyhedral cells. The equations of motion are integrated over the volume of each cell and Gauss' theorem utilised to convert divergence terms in the equations into fluxes across cell faces. This converts the governing partial differential equations into a set of difference equations, representing spatial derivatives in terms of differences in quantities between neighbouring cells, and these difference equations can be solved by matrix inversion. The finite volume method is commonly used in a range of commercial and academic codes.

Unlike most CFD codes (and all commercial codes), OpenFOAM is not properly a CFD *code* but is a C++ library of classes and routines of use for writing CFD codes. C++ is an Object Orientated language, which means that it contains the facilities for extending the types available in the basic language (C, in the case of C++) by defining new classes. Classes are units of code which contain data and functions which operate on that data; once a new class has been declared, variables of that type can be created (in the jargon, instances of the class can be instantiated) and manipulated using the associated functions. In the case of C++, the associated functions can be textual or symbolic; it allows the redefinition of operators such as + and % (operator overloading). This facility is immensely useful for defining mathematical classes for which these operators have obvious meaning – if we wish to define a vector class, for example, the concepts of addition and multiplication are well defined and obviously implied by the symbols + and *. Object orientated languages also include the facility to define relations between classes – inheritance. This can be used to promote code reuse, for example by constructing new classes using data classes already defined (encapsulation) or by extending existing classes (derivation). It also provides ways of grouping types of object together. For instance, in OpenFOAM all RANS turbulence models are variants of one common base class; amongst other things all turbulence models provide a value for the divergence of the Reynolds Stress. In programming terms all OpenFOAM RANS models are implemented as being derived from a common base class. This makes instances of the RANS models interchangeable (so they can be selected at runtime) and enforces a common set of functions on any new class to be added (promoting a standard interface to the RANS models). Object orientation is the best way that has yet been found to organise data in a large computer program. It has the advantage of allowing multiple programmers to work on different parts of the programming task independently. Hence; OpenFOAM classes operate at different levels;

[1] The OpenFOAM name is a registered trademark of OpenCFD Ltd.

- Basic variable types (`vector`, `dimensionedScalar` etc.)
- Container classes such as arrays of basic types
- Mesh classes and field classes, allowing the definition of for instance a field of vectors defined at particular points in space
- Explicit evaluation of derivatives by differencing between cells
- Matrix classes for implicit evaluation of derivatives and equations by matrix inversion
- Modelling classes representing mathematical models, eg. of turbulence.

The separation of class implementation from class interface means that it is perfectly possible for one programmer to work on the lower level classes (eg. introducing new matrix inversion routines) independently of another working on the higher level classes (eg. the turbulence models). One of the design aims of the OpenFOAM code is to provide a top level class syntax that is as close as possible to standard mathematical notation. The operator overloading mentioned above is one aspect of this. Additionally, OpenFOAM tracks the physical dimensions of all variables and fields through a class `dimensionSet`, which registers all 7 SI dimensions; it is thus impossible to implement a model which is dimensionally incorrect.

Most CFD calculations involve implicit rather than explicit evaluation; that is to say, the solution of any PDE is represented by a matrix equation of the form

$$\mathcal{M}x = g \tag{1}$$

where x is a vector of unknown values representing the state of the dependent variable field to be determined, and $\mathcal{M}$ and g are a known matrix and vector respectively. Evaluation of the equation is accomplished by inverting this equation. Matrix equations in OpenFOAM are represented by a group of classes based on a template, `fvMatrix` which stores the various coefficients making up $\mathcal{M}$ and g. Two properties of objects of this class are noted;

1. Particular patterns of entries in $\mathcal{M}$ and g represent particular differential operators. Functions in the `fvm::` namespace (finite volume method) return variables of type `fvMatrix` representing particular vector operators.
2. `fvMatrix` objects can be manipulated algebraically; the standard operators + etc. having been appropriately overloaded to permit this.

The result of this is that matrix equations can be built up operator by operator in a way which is very similar to mathematical notation and which is therefore intuitively simple to read. At the top level indeed, OpenFOAM could be regarded as a high-level procedural language for writing CCM codes.

As an example, let us examine the Burgers equation. In 1-d this takes the form

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \tag{2}$$

This is very close to the 1-d Navier-Stokes equation, without the complication of the pressure term. The term $u\frac{\partial u}{\partial x}$ is non-linear; we can rewrite this in our standard conservation form as

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{1}{2}u^2\right) = \nu\frac{\partial^2 u}{\partial x^2} \tag{3}$$

We can extend this to 3d, writing

$$\frac{\partial \vec{u}}{\partial t} + \vec{u}.\nabla\vec{u} = \nu\nabla^2\vec{u} \tag{4}$$

Again, assuming the fluid to be incompressible ($\nabla.\vec{u} = 0$), we can rewrite this in conservative form as

$$\frac{\partial \vec{u}}{\partial t} + \nabla.(\frac{1}{2}\vec{u}\,\vec{u}) = \nu\nabla^2\vec{u} \tag{5}$$

This can be implemented in OpenFOAM as

```
fvVectorMatrix UEqn
(
    fvm::ddt(U)
  + 0.5*fvm::div(phi, U)
  - fvm::laplacian(nu, U)
);

UEqn.solve();
```

where the matrix equation `UEqn` is first constructed from the various individual terms `fvm::ddt(U)` etc. and then solved by the call `UEqn.solve()`. Various specific differencing schemes (for the different derivatives) and matrix inversion algorithms (for `UEqn.solve()`) are included and are selected at run-time through entries in dictionary files `fvSchemes` and `fvSolution` respectively. To complete the solution of the Burgers equation, we enclose this in an appropriate loop, iterating through timesteps until the end of the simulation, and writing the results out as necessary. This has been applied to the 1d case of a sine wave, with results shown in figure 1; the expected sharpening of the initial sinusoidal condition due to the equation's nonlinearity is clearly demonstrated.

2 LID-DRIVEN CAVITY CASE

In this section I present a case study demonstrating the use of OpenFOAM for sophisticated CFD modelling. The test case selected for this is a variant of the archetypal lid

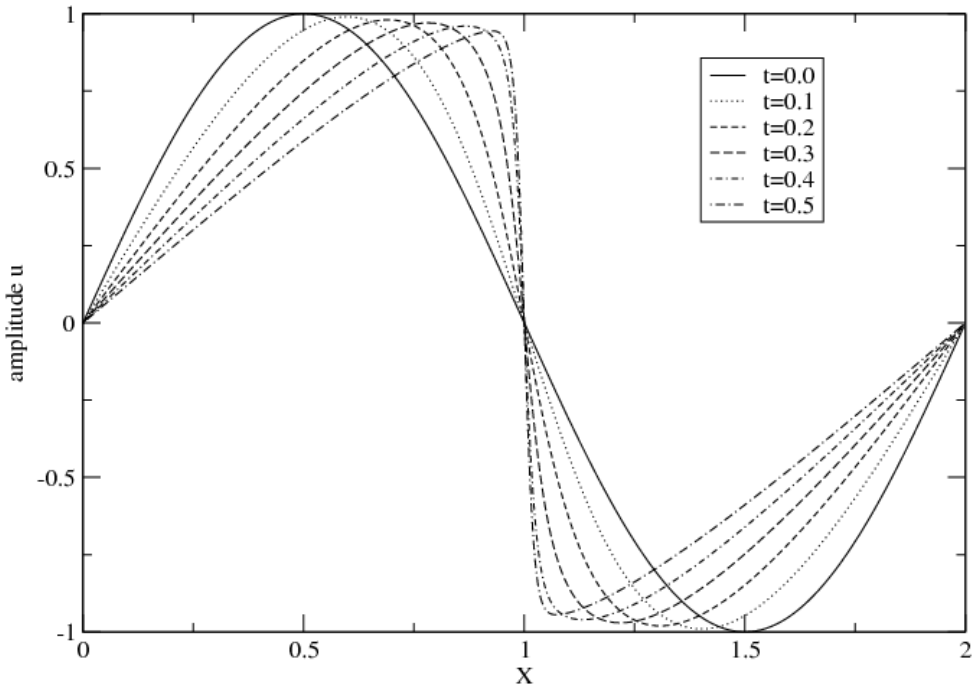


Figure 1: Results from solution of Burgers equation for 1d case. The initial conditions of a sinusoidal wave sharpen with time as the propagation speed depends on amplitude.

driven cavity problem. In the standard lid driven cavity problem, one wall (conventionally the top) moves parallel to itself at a fixed velocity U_0 whilst the others are fixed. The wall motion drives a main vortex roughly centred on the middle of the domain; subsidiary vortices may occur in the corners, depending on flow regime, and obviously the flow may be laminar or turbulent depending on the Reynolds number. There are two issues with this as a test case; the wall motion at the very corners of the moving wall is ill-defined, and there is no analytical solution to compare with. Nevertheless there is a substantial body of experimental and DNS data in the literature to compare with, and this does represent a standard test case for CFD codes. Numerous variants of the basic case have also been investigated, of these of particular interest is the modified lid driven cavity case introduced by Shih *et al* [3], in which a spatially varying lid velocity and a complicated body force are introduced into the 2d problem. The result of these alterations is that the problem now has an analytical solution. Shih *et al* give the problem in dimensionless coordinates as the solution of the equations

$$\nabla.\vec{u_*} = 0 \tag{6}$$

$$\vec{u_*}.\nabla\vec{u_*} = \frac{1}{Re}\nabla^2\vec{u_*} - \nabla p_* - \vec{j}B(x, y, Re) \tag{7}$$

Across the top surface the moving wall velocity is given as

$$u_{*x}(x, 1) = 16(x^4 - 2x^3 + x^2) \tag{8}$$

and the body force is given by the expression

$$B(x,y,Re) = -\frac{8}{Re}\left[24F(x) + 2f'(x)g''(y) + f'''(x)g(y)\right] - 64\left[F_2(x)G_1(y) - g(y)g'(y)F_1(x)\right] \tag{9}$$

where

$$\begin{aligned}
f(x) &= x^4 - 2x^3 + x^2 \\
g(y) &= y^4 - y^2 \\
F(x) &= \int f(x)dx = 0.2x^5 - 0.5x^4 + x^3/3 \\
F_1(x) &= f(x)f''(x) - 0.5\left[f'(x)\right]^2 = -4x^6 + 12x^5 - 14x^4 + 8x^3 - 2x^2 \\
F_2(x) &= \int f(x)f'(x) = 0.5\left[f(x)\right]^2 \\
G_1(y) &= g(y)g'''(y) - g'(y)g''(y) = -24y^5 + 8y^3 - 4y
\end{aligned}$$

where the primes represent derivatives w.r.t. x and y as appropriate. The exact solution to this problem is

$$\begin{aligned}
u_{*x}(x,y) &= 8f(x)g'(y) = 8(x^4 - 2x^3 + x^2)(4y^3 - 2y) \\
u_{*y}(x,y) &= -8f'(x)g(y) = 8(4x^3 - 6x^2 + 2x)(y^4 - y^2) \\
p_*(x,y,Re) &= \frac{8}{Re}\left[F(x)g'''(y) + f'(x)g'(y)\right] + 64F_2(x)\left[g(y)g''(y) - \left[g'(y)\right]^2\right]
\end{aligned}$$

Helpfully, Shih *et al* note the fact that $B(0.5, 0.5, 1) = -3.356250$. Implementation of this into OpenFOAM is of course fairly straightforward once the conversion back to dimensioned quantities has been achieved. Results are shown in figure 2; which shows velocity vectors for the flow solution together with contours of the magnitude of the solution error.

3 RESEARCH

In this section I present a short overview of a number of research projects undertaken in OpenFOAM at Exeter.

3.1 Polymer Rheology

The development of constitutive models to describe polymer flow has made significant progress in recent years. However models that accurately describe polymer behaviour are computationally very expensive, particularly in practical industrial situations where high shear rates and polydisperse polymers are important. Thus there is an incentive to develop constitutive equations that adequately predict the correct dynamics but are computationally tractable. One established model is the Remmelgas-Harrison-Leal (RHL) model. This is a vector-based implementation of the DEMG reptation model in which the

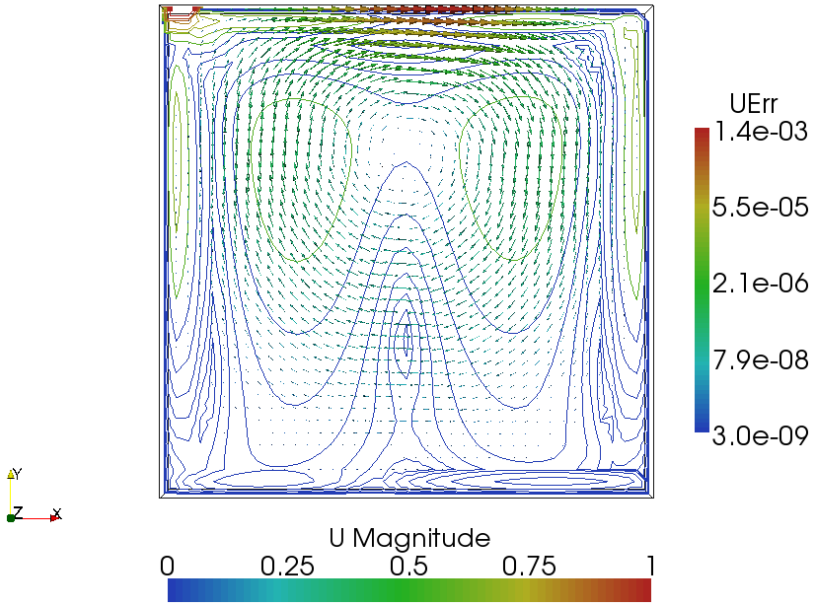


Figure 2: Results from modified lid driven cavity case. The solution is represented by the velocity vectors (coloured according to the lower scale) and the magnitude of the solution error shown by the contours (right hand scale).

flow-induced orientation and stretch of polymer chains in the flow occurs on timescales which are sufficiently widely separated that they can be effectively decoupled from one another. The second moment of the orientation distribution for a unit vector $\vec{n}$ is given by

$$\frac{\delta < \vec{n}\vec{n} >}{\delta t} = -2\nabla\vec{u} :< \vec{n}\vec{n} >< \vec{n}\vec{n} > -\frac{1}{\tau_d < R^2 >}\left[< \vec{n}\vec{n} > -\frac{1}{3}\vec{I}\right] \tag{10}$$

where $\delta/\delta t$ is the incompressible Co-deformational time derivative :

$$\frac{\delta < \vec{n}\vec{n} >}{\delta t} = \frac{\partial < \vec{n}\vec{n} >}{\partial t} + \vec{u}.\nabla < \vec{n}\vec{n} > -\nabla.(< \vec{n}\vec{n} > \vec{u}) - \nabla\vec{u}^T. < \vec{n}\vec{n} > \tag{11}$$

The second factor describing the molecular dynamics of the polymer molecules is the stretch relaxation, which is described in terms of the vector stretch factor $\vec{R}$ which is the end-to-end length of the polymer scaled by the equilibrium chain length. The evolution of this quantity is described in terms of the evolution of $< R^2 >$:

$$\frac{D < R^2 >}{Dt} = 2\nabla\vec{u} :< \vec{n}\vec{n} >< R^2 > -\frac{1}{\tau_r}\left[f(< R^2 >) < R^2 > -1\right] \tag{12}$$

Here τ_r is the longest Rouse timescale and $f(< R^2 >)$ is a nonlinear spring function representing the molecular extensibility. τ_r and τ_d are related through the number of

entanglements per chain :

$$\tau_d = 3N_e\tau_r \tag{13}$$

The governing equations of motion of the polymer are given by the Navier-Stokes equations with the stress term

$$\vec{\sigma} = \frac{G_N^0}{\tau_d} f(< R^2 >) < R^2 >< \vec{n}\vec{n} > \tag{14}$$

This was implemented as a laminar solver based on PISO using OpenFOAM. The resulting code was named polyFoam and was applied to the case of flow in a sudden contraction. This case was chosen to be similar to the die of a capillary rheometer. As part of the project, an experimental test rig was also developed giving optical access to this die [4], allowing comparison on various flow and orientation parameters. More details and results are presented in [5]; representative results are shown in figure 3

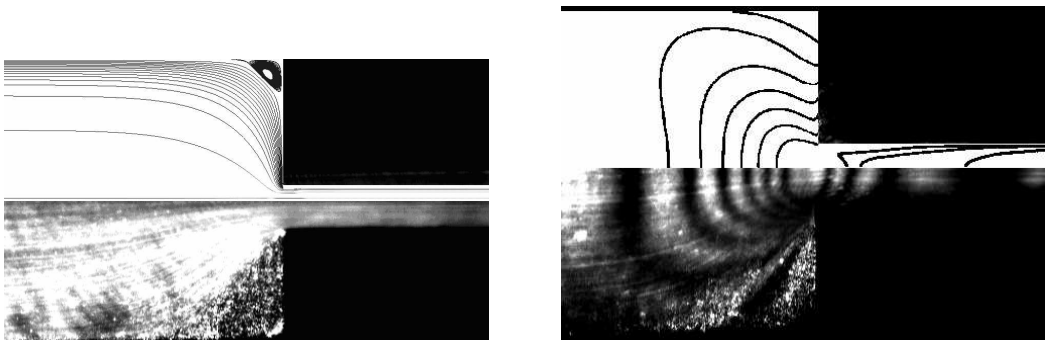


Figure 3: Polymer flow in extrusion die. Left; comparison of streamlines (top) calculated from polyFoam and visualised experimentally (bottom). Right; comparison of birefringence patterns calculated (top) and visualised experimentally (bottom).

3.2 Large Eddy Simulation

Large Eddy Simulation (LES) has been extensively investigated as an alternative to RANS simulation for turbulent flow. Although considerably more expensive than RANS, its explicit simulation of the large scale eddies in the flow produces results which are more accurate, particularly for flow cases which are strongly influenced by these large scales, such as combustion [6]. In particular, swirling flows are difficult to model with Reynolds Averaged Navier Stokes (RANS) methods due to the effect of the mean flow streamline curvature [7], so this is one example where LES methods have come to the fore. Research in LES in Exeter has concentrated on the implementation of inlet boundary conditions. Implementing inlet conditions for LES is significantly more challenging than is the case for RANS models; the inlet flow has to include the grid scale (GS) turbulence, and so has to include a stochastically fluctuating component which satisfies a range of conditions (such as the correct temporal and spatial correlation) [8]. OpenFOAM includes a remapping

technique implemented by de Villiers [9] in which a region of the domain is designated as the inlet domain, at the end of which the flow conditions are sampled and reintroduced at the start of the domain. This recirculating of the flow can be shown to generate very good inlet turbulence [10]. Further work at Exeter has focused on modifying this mapping process to allow the exact properties of the turbulent flow at the inlet to be tuned to meet specific targets, in particular target mean velocity profiles including swirl, and target turbulent profiles. A combination of techniques have been investigated using feedback control; coupled with velocity correction in the mapping section to generate the desired turbulent profile, and with a variable body force to generate the desired swirl profile. Results for a sudden expansion case are shown in figure 4 [11, 12].

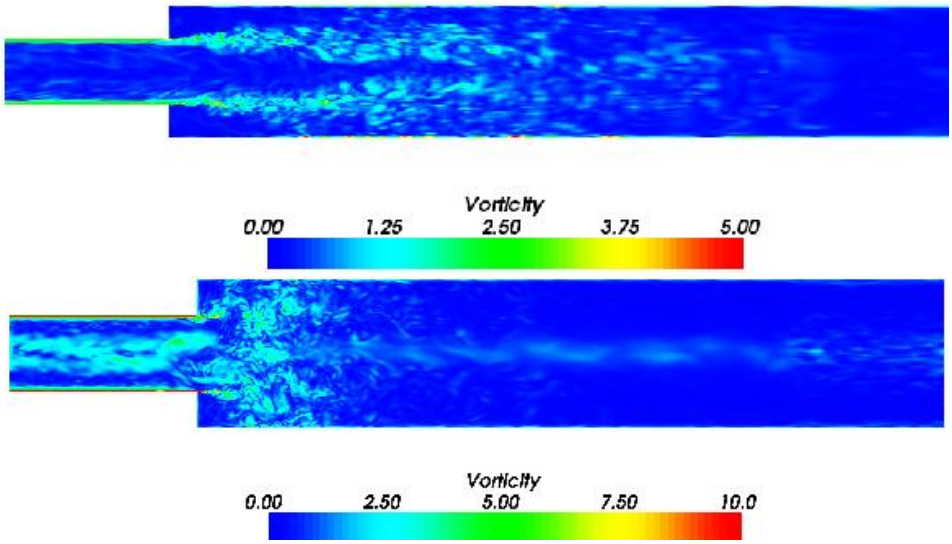


Figure 4: LES of flow in sudden expansion; without swirl (top) and with swirl (bottom); Swirl number 0.6.

3.3 Urban Drainage

OpenFOAM is also being used in Exeter in relation to a number of projects with industrial impact in the urban drainage area. One is a KTP project with the company Hydro International Ltd, which is a company involved in the design and manufacture of devices for flow control and wastewater separation for urban drainage. Their products are based on control of water through vortex motion, and they have a longstanding interest in CFD processes as an investigative tool for understanding and improving their products. The primary aim of the project has been to use CFD to investigate the behaviour of one specific product; a Vortex Flow Control (VFC) used to regulate flow of stormwater runoff through a drainage network. This device operates in two states; at low flow it behaves as a simple orifice plate with low head loss, whilst at higher flow rates a vortex forms choking off the flow and substantially increasing the head loss. The behaviour of this

vortex and its development is a highly complex process which is challenging to simulate computationally. The project has successfully validated the use of CFD to investigate this behaviour using Reynolds Stress models, with and without free surface effects being included; see figure 5a,b. for more details. Results from the modelling have been used to develop a functional model for the unit behaviour which has already proved of significant commercial value to the sponsoring company.

The second project being pursued in this area is the simulation of surface water runoff from road surfaces into the drainage system. This is one part of a substantial multi-partner research programme, FRMRC-II (Flood Risk Management Research Consortium) led by Heriot-Watt University. Modelling of the runoff is complex, having to deal with both shallow depth effects (on the roadway) and deep flow (in the gully) through a geometrically complex trap. OpenFOAM provides 3 computational models for free surface flow; the Volume of Fluid (VOF), G-equation and Finite Area [13] approaches. Results from VOF simulations have been compared with results from experiments being carried out at Sheffield University, and have shown very promising results (figure 5c,d.). Simulations using the other techniques are currently in progress.

4 DISCUSSION

OpenFOAM represents a significant departure from the traditional approaches to engineering CFD. Previously, most CFD was performed using commercial 'grey-box' codes which provided good technical support for a product with limited ability to examine or change the solution methodologies. License costs have tended to be high and based on number of concurrent jobs, resulting in a financial constraint on the use of CFD within any organisation. Illustrative of this is the experience of Hydro International, where the introduction of OpenFOAM has leveraged a very substantial expansion of CFD within the company. CFD has always been carried out using non-commercial codes in the academic sector, but such codes have tended to be limited, both in abilities (often by choice; written to investigate specific issues) and in geographical extent (used within one individual research group).

The Open Source model for providing software which has demonstrated advantages in other industries, changes this paradigm in CFD as well. From the financial perspective it divorces the direct cost of the CFD simulation itself from other incurred costs, such as support and mesh generation. Thus the number of CFD simulations undertaken is not constrained by license costs but by other factors such as availability of computing hardware, which is significantly less costly today. From the scientific perspective it provides a common code platform for researchers in CFD to exchange practical code examples. Hence, in the traditional research model, researcher A would develop some new methodology, and would publish the technical details in a journal. Researcher B, wanting to make use of this new methodology, would then have to work out how to implement it in her own code. With OpenFOAM as a common modelling platform, A can just send B the code section to examine and if required, compile into B's own program. This enhances

code sharing. As with many open source projects, an extensive community of users has built up who are able to exchange code in this way, as well as mutual support; including through the CFD-online bulletin board [14], and separate community activities such as the OpenFOAM Workshop series. Individuals can use OpenFOAM at several levels. At the most basic level, it can be used in a similar way to more traditional CFD codes; using preexisting code examples to solve particular modelling problems (eg. `turbFoam` for turbulence simulation problems). At a somewhat higher level, the basic library structure makes it easy to implement new modelling and develop new stand-alone codes. In the most in-depth cases, the whole code is available under the terms of the GNU license and can be extended, as with the '-dev' version created by Prof Jasak [15, 16], or pyFoam created by B.Gschaider [17].

5 CONCLUSIONS

OpenFOAM is a fully-featured Open Source CCM code with capabilities which rival and in many cases exceed those of traditional commercial CFD codes. It represents an alternative paradigm in CFD modelling which is beneficial both for software users (zero license fee means potential for a greater use of CFD) and for modellers, with innovations in scientific software development which are being adopted in other scientific codes [18]. In this paper I have tried to showcase some of the research being carried out at the University of Exeter using OpenFOAM in the fields of constitutive modelling, LES, and urban drainage, at the same time demonstrating some of the diverse possibilities for using OpenFOAM's simulation capabilities.

6 ACKNOWLEDGEMENTS

This paper has been conceived as a review of work undertaken at the University of Exeter by various individuals whose contributions I gratefully acknowledge. These include PhD students Mohammad Baba-Ahmadi, Dan Jarman, Istvan Galambos and Mark Beard; colleagues Dr Slobodan Djordjevic, Prof David Butler, Prof Ken Evans, and Dr Mike Faram (Hydro International). Prof Hrvoje Jasak's contributions to the work are gratefully acknowledged. I would also like to thank OpenCFD Ltd for supporting and releasing OpenFOAM.

REFERENCES

[1] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby. A tensorial approach to computational continuum mechanics using object orientated techniques. *Computers in Physics*, 12(6):620 – 631, 1998.

[2] OpenCFD Ltd. website:. http://www.openfoam.com/.

[3] T. M. Shih, C. H. Tan, and B. C. Hwang. Effects of grid staggering on numerical schemes. *int.J.Num.Methods in Fluids*, 9:193 – 212, 1989.

[4] M. A. Beard, G. R. Tabor, S. J. K. Ritchie, K. E. Evans, and R. I. Walton. Development of freeze flow apparatus to study filled polymer melts. *Polymer Engineering and Science*, 47(11):1937 – 1942, 2007.

[5] M.A. Beard, G. R. Tabor, S. J. K. Ritchie, and K. E. Evans. Numerical investigation and visualisation of polymer behaviour within a capillary die. *Polymer Engineering and Science*, 47(10):1688 – 1694, 2007.

[6] G. Tabor and H. G. Weller. Large Eddy Simulation of premixed turbulent combustion using ξ flame surface wrinkling model. *Flow, Turbulence and Combustion*, 72:1 – 28, 2004.

[7] S. Jakirlić, K. Hanjalić, and C Tropea. Modelling rotating and swirling turbulent flows: a perpetual challenge. *AIAA J.*, 40(10):1984 – 1996, 2002.

[8] G. R. Tabor and M. H. Baba-Ahmadi. Inlet conditions for large eddy simulation: A review. *Computers and Fluids*, 39:553 – 567, 2010.

[9] E. de Villiers. *The Potential for Large Eddy simulation for the modelling of wall bounded flows*. PhD thesis, Imperial College, 2006.

[10] G. Tabor, M. H. Baba-Ahmadi, E. de Villiers, and H. G. Weller. Construction of inlet conditions for LES of turbulent channel flow. In *Proceedings of the ECCOMAS Congress, Jyväskylä, Finland*, 2004.

[11] M. H. Baba-Ahmadi and G. Tabor. Inlet conditions for LES of gas-turbine swirl injectors. *AIAA.J.*, 46(7):1782 – 1790, 2008.

[12] M. H. Baba-Ahmadi and G. Tabor. Inlet conditions for LES using mapping and feedback control. *Computers and Fluids*, 38(6):1299 – 1311, 2009.

[13] Ž. Tukovic and H Jasak. Simulation of free-rising bubble with soluble surfactant using moving mesh finite volume/area method. In *6th International Conference on CFD in Oil & Gas, Metallurgical and Process Industries*. SINTEF/NTNU, June 2008.

[14] CFD-Online. Openfoam bulletin board. http://www.cfd-online.com/Forums/openfoam/.

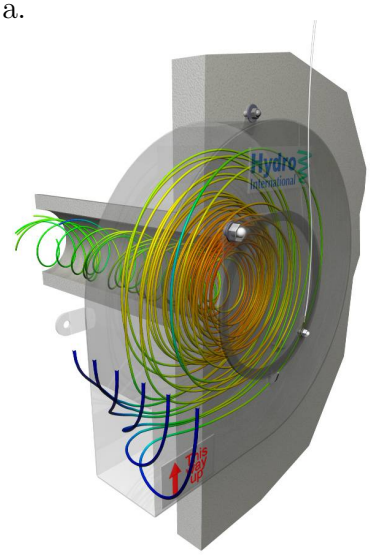
[15] Wikki Ltd. website. http://www.wikki.co.uk/.

[16] H. Jasak. Openfoam: Open source cfd in research and industry. *International Journal of Naval Architecture and Ocean Engineering*, 1(2):89–94, December 2009.

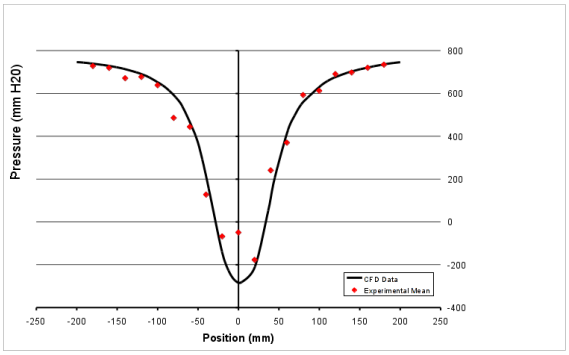
[17] B. Gschaider. Pyfoam:. http://openfoamwiki.net/index.php/Contrib_PyFoam.

[18] B.D. Dudson, M.V. Umansky, X.Q. Xu amd P.B. Snyder, and H.R. Wilson. BOUT++: A framework for parallel plasma fluid simulations. *Computer Physics Communications*, 180(9):1467 – 1480, 2009.

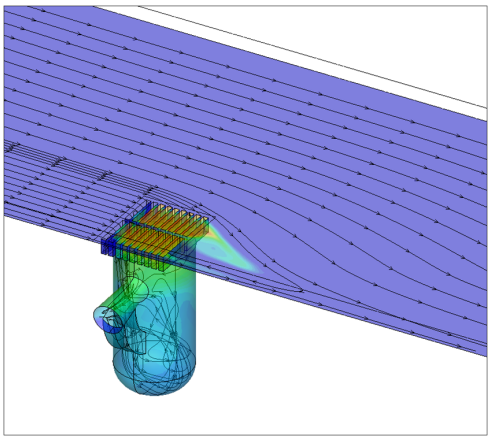
a.



b.



c.



d.

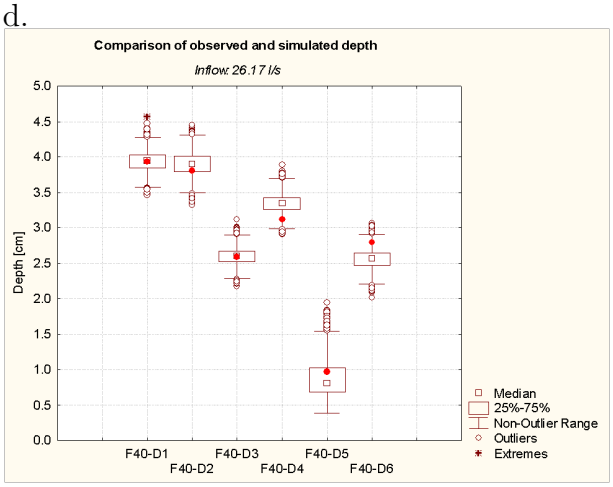


Figure 5: Urban drainage-related examples of OpenFOAM use. a. Flow simulation in Vortex Flow Control; b. comparison of experimental and computed pressures across VFC unit. c. Streamlines of flow for gully pot simulation. d. Comparison of observed and simulated depth.