III International Symposium
Contemporary Problems of Fluid Mechanics
May 12-13th, 2011.
University of Belgrade,
Faculty of Mechanical Engineering, Serbia

An Overview of OpenFOAM CFD Software

A. Ćoćić[1] and I. Guranov[1]

[1]Department of Fluid Mechanics, Faculty of Mechanical Engineering, University of Belgrade, Serbia
Corresponding author: acocic@mas.bg.ac.rs

Abstract

Computational fluid dynamics (CFD) is nowadays a vital part of engineering practice and research in fluid mechanics. Application areas of CFD are vast - from fluid flows in various engineering devices, to flow of air in Earth atmosphere to flow of blood in human vessels. These are all very complex phenomena, but with adequate modeling and afterwards numerical solving, they can be predicted with appropriate accuracy. There are a lot of softwares specially designed for solving problems in fluid dynamics. Most of them are commercial ones, licensed with very expensive price and from the viewpoint of user, it is impossible to fully access and modify the numerical codes. The open source CFD is one of the solutions to overcome these obstacles. One of the open-source CFD softwares is OpenFOAM. In this paper a brief overview of it's usage and capabilities are presented. Essentially, OpenFOAM is large C++ library from which users can use precompiled applications and utilities, or they can use the library to create their own applications and utilities. The code for creation libraries and utilities is open.

Keywords: open-source, CFD, OpenFOAM

1. Introduction

CFD software has developed in outstanding way is past decades. Nowadays it is possible to simulate numerically complex fluid behavior in various engineering devices and in nature. It's interesting to make a short historical development of CFD. Numerical methods itself were known since Newton's time in 1700s. The solution methods of ordinary and partial differential equations were conceptually established, but it was only on paper. One of first CFD calculations were performed by Lewis Fry Richardson (1881-1953) at the beginning of 20th century. He did the first numerical weather prediction system by dividing physical space into grid cells and using the finite difference approximations of the equations for flow in atmosphere, developed by Vilhelm Bjerknes, [1]. In 1910, Richardson published 50 pages paper to Royal Society. From our point of view it sounds like unbelievable thing, but Richardson performed hand calculations, while he was serving with the Quaker ambulance unit in northern France. Richardson attempted to use a mathematical model of the principal features of the atmosphere, and he used data taken at a specific time (7 a.m.) to calculate the weather six hours later. By calculations, he predicted a huge 14.5 kPa rise in pressure over six hours when the pressure actually stayed more or less static. However, detailed analysis by Lynch [2] has shown that the cause was a failure to apply smoothing techniques to the data, which rule out unphysical surges in pressure. When these are applied, Richardson's forecast turns out to be essentially accurate which is a remarkable achievement considering the calculations were done by hand! Later on, in 1943 Finite element analysis (FEA) was first developed by R. Courant, who utilized the Ritz method of numerical analysis and minimization of variation calculus to obtain approximate solutions to vibration systems. Around 1960 first Scientific American articles on CFD were published, in 1965 CFD became significant research tools 1970 finite difference method for Navier-Stokes equations were derived. Based on the work of Harlow and Yakayma, [3] Launder and Spalding from Imperial College London proposed famous $k-\varepsilon$ turbulent model. The team from Imperial College also established finite volume method, which is now the most common standard approach in CFD. Another key event in CFD industry was in 1980 when Suhas V. Patankar published "Numerical Heat Transfer and Fluid Flow". It is possibly the most influential book on CFD

to date, and the one that spawned a thousand CFD codes. During 1980s, based on work at Imperial College in 1970s, CFD software appeared on the market, and from that moment their continuous developments started till the present times. These are some milestones in that sense:

- 1981 - PHOENICS was launched as first commercial CFD software
- 1983 - Fluent was launched also as commercial CFD software, afterwards a number of commercial CFD software growth and fulfilled the competition of CFD market.
- 1985 - use of CFD software in aeronautical industry (Boeing, General Electric, . . .)
- 1995 - use in other fields of industry (General Motors, Mercedes, Audi, BMW, Ford, Astra, Ericsson, . . .)
- 2004 - an open-source CFD software FOAM was released
- 2006 - Fluent was acquired by ANSYS, Inc. This acquisition makes ANSYS as a strongest computer aided engineering player in numerical FEA and CFD simulation

This was a brief overview of significant moments in development of CFD principles and accompanied software. In terms of software development in general, during 1980s, besides the commercial software, due to pioneering work of Richard Matthew Stalman so called "free software" also appeared on the market. According to Stallman, [4] it is essential for computer user that software he or she is using must be "free". The key point is not on free as "free beer" (zero prize), but on four essential freedoms: (0) The freedom to run the program, for any purpose; (1) The freedom to study how the program works, and adapt it to your needs (access to the source code is a precondition for this); (2) The freedom to redistribute copies so you can help your neighbor; (3) The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (access to the source code is a precondition for this). We think that these principles are really essential, specially in research and education at the University and we have to keep them in mind in our everyday work - not only from viewpoint of software we use, but on more broad contents. During 1990s Open Source Movement was formed from inside the Free Software Movement. Open Source is more or less based on free software principles - open source software mostly use some variants of GNU General Public Licenses. But it's accent is on practical point of software use
rather than philosophical one. Open source community don't reject use of non-free software (it is think mostly on non-open code) for some of the applications, but most of software has an open code which can be studied and extended.

One of the open source CFD software is OpenFOAM, developed at the beginning of 21st century at the Imperial College. In following sections we'll make a review of OpenFOAM, and it's capabilities in modern CFD calculations.

2. OpenFOAM as open source tool for CFD

Development of tool what later become OpenFOAM started at the beginning of 1990's at Imperial College London by group led by prof. David Gossman and Dr. Raad Issa. Two of principal developers were Henry Weller and Hrvoje Jasak. Large number of PhD thesis also contributed to development of the code. The code itself was written in C++, because the main idea was to create a C++ class library for computational continuum mechanics. This object-oriented approach has numerous advantages. According to Brajne Stroustrup, creator of C++, this approach

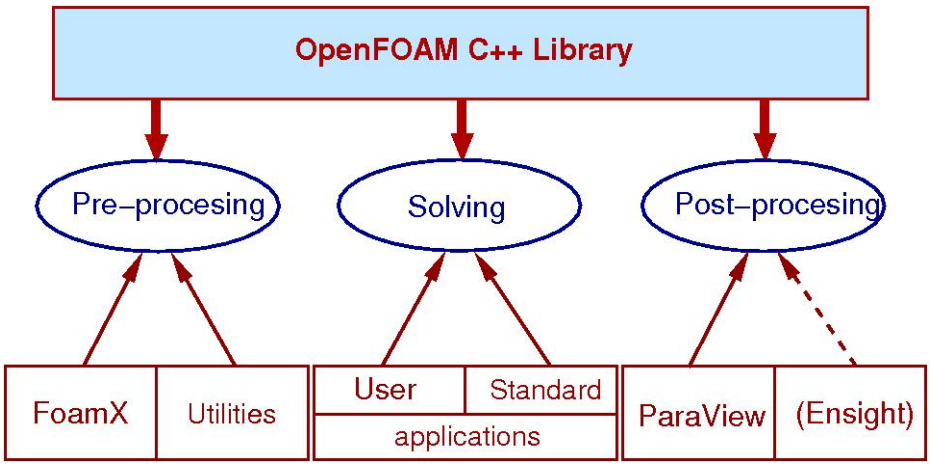

Figure 1: Overall structure of OpenFOAM.

involves three main things: abstraction, inheritance and polymorphism. Abstraction enables that conceptual objects can be represented as classes. These classes are encapsulated, i.e. they contain and protect data that make up the object. For each class, member functions are provided. These member functions permit limited, well-defined access to the encapsulated data. Thus it is possible to create data types that represent tensor fields and typical terms in the equations constructed to behave like their mathematical counterparts, hiding the numerical details of the implementation by encapsulation. Inheritance enables relationships between the various classes to be expressed, representing commonality between different classes of objects by class extension. By doing this existing classes can be given new behavior without the necessity of modifying the existing class. For example, this can be used to construct a complicated mathematical class by extending base classes that express simpler mathematical objects. On the higher level in OpenFOAM code this is used in making conceptual links between turbulence models, [5]. Polymorphism is the ability to provide the same interface to objects with different implementations, thus representing a conceptual equivalence between classes that in practical terms have to be coded differently. Examples of this in OpenFOAM include the implementation of boundary conditions. At the end, C++ also involving operator overloading, which means that it is possible to construct an interface that resembles standard mathematical notation. For example, sign `*` is ordinary use for multiplication of two scalars, but it is also used as sign for outer product of two second order tensors.

Comparing this approach with the approach in procedural languages (FORTRAN or C) we see numerous advantages. In procedural languages accent is on low-levels of coding, i.e. on manipulation between numbers organized in arrays. Highest level of data is vector or matrix. In OpenFOAM the viewpoint is at higher level, from physical objects - tensors which are describing various physical quantities in continuum mechanics. The result is C++ library in which is possible to implement a basically every continuum - mechanics modeling techniques in modern CFD. In fact any system of time-dependent PDEs including convection, diffusion, and source terms can be handled.

This high level of abstraction makes it possible to represent complicated mathematical and physical models in code as high-level mathematical expressions, [5]. For example, if we consider laminar, unsteady flow of incompressible Newtonian fluid equation that describes the motion of that fluid is Navier-Stokes equation

$$\frac{\partial U}{\partial t} + \nabla \cdot \left(UU \right) = -\nabla p + \nabla \cdot \left(\nu \nabla U \right). \tag{1}$$

Numerical solving of this equation in OpenFOAM code is represented with following lines

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

solve(UEqn == -fvc::grad(p));
```

It can be seen that representation of equation in code is very clear, and it corresponds to the equation written in mathematical notation in invariant form.

At the top of OpenFOAM code are solvers, which are designed for solving for particular problems in CFD. For example, `simpleFoam` solvers is designed for steady flow of incompressible fluids. There are numerous solvers available in OpenFOAM, from basic solvers like `scalarTransportFoam` or `potentialFoam` to solvers the can handle turbulence modelling (LES, DES, URANS), multi-phase flows, heat transfer and multi-physics simulation like fluid-structure interaction, or conjugate heat transfer problems.

For creation new solvers and application OpenFOAM is supplied with `wmake` compilation script that is based on make but is considerably more versatile and easier to use, [6]. A class is defined through a set of instructions such as object construction, data storage and class member functions. The file containing the class definition takes a .C extension, e.g. a class nc would be written in the file nc.C. This file can be compiled independently of other code into a binary executable library file known as a shared object library with the .so file extension, i.e.nc.so. When compiling a piece of code, say newApp.C, that uses the nc class, nc.C need not be recompiled, rather newApp.C calls nc.so at runtime. This is known as dynamic linking, [7].

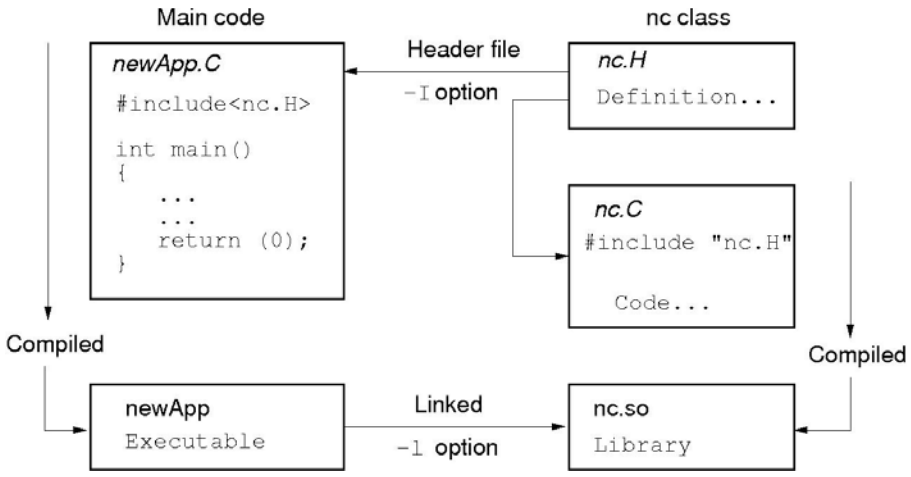

Figure 2: Header files, source files, compilation and linking

3. Examples of OpenFOAM use

We'll now show results of simulation in OpenFOAM on one simple, but always interesting example in fluid mechanics - a vortex shedding behind circular cylinder.

3.1 Laminar flow around circular cylinder in a channel

We'll take the geometry given in [8], which is shown in Figure 3. We are considering the flow as 2D. At the inlet, a parabolic velocity inlet is prescribed

$$u_x = \frac{6U}{H^2}\left[(y+2D)H-(y+2D)^2\right], \qquad u_y = 0,$$

where U is the mean velocity, H = 4.1D is the channel height.

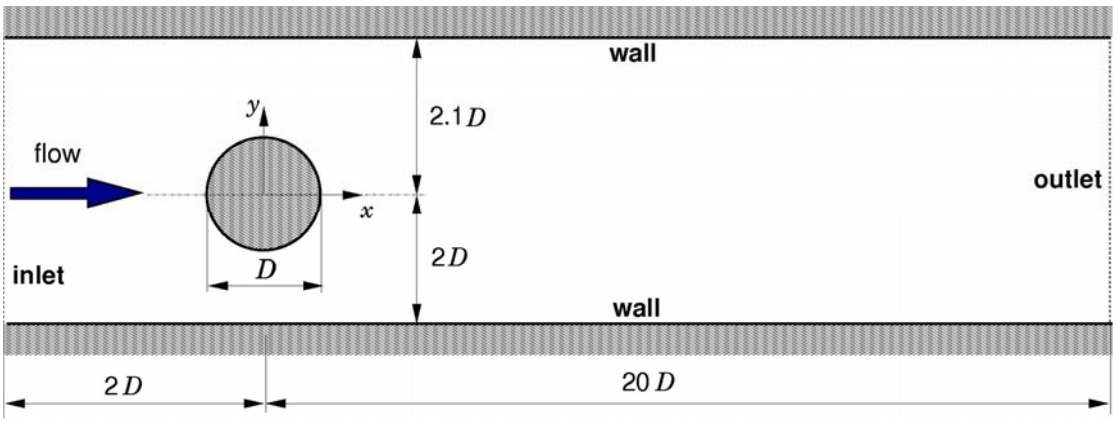

Figure 3: Geometry of the problem

The main task here is to determine drag and lift forces, which are present due to asymmetrical location of cylinder in the channel. For that purpose we'll use OpenFOAM library `libForces.so`. We choose value of Reynolds number $\mathrm{Re} = 100$, for which we have vortex shedding behind circular cylinder. This vortex shedding will imply that lift and drag force will also oscillate.

So, we are considering laminar, unsteady flow of incompressible Newtonian fluid. It is described with Navier-Stokes equations which can be written in the strong conservation form

$$\frac{\partial \vec{U}}{\partial t} + \nabla \cdot \vec{U}\vec{U} = -\nabla p^* + \nabla \cdot \left(\nu \, \nabla \vec{U}\right) \qquad (2)$$

suitable for discretization by finite volume method. Physical quantity p^* is so called kinematic pressure

$\left(p^* = p/\rho\right)$, and ν is kinematic viscosity. For specified problem we are using the following boundary condition:

$$\text{inlet}: \quad u_x \text{- fully developed profile}, \; u_y = 0; \qquad \frac{\partial p}{\partial x} = 0$$

$$\text{outlet}: \quad \frac{\partial u_x}{\partial x} = \frac{\partial u_y}{\partial x} = 0, \qquad p^* = p_0^*$$

$$\text{walls}: \quad \vec{U} = 0; \qquad \frac{\partial p}{\partial n} = 0,$$

where p_0^* is constant value of kinematic pressure at the outlet, and index n in wall boundary condition for pressure designates a direction perpendicular to wall.

Solver for this type of flow (laminar, unsteady incompressible flow) in OpenFOAMis `icoFoam`. For mesh generation we used application `blockMesh` and we performed calculations on four systematically refined block-structured grid. The coarsest grid had 5000 cells, and the finest 100000 cells. We've achieved the grid independence test on finest grid. We used Euler differencing schemes for unsteady term, upwind differencing
scheme for convective therm, and central differencing scheme for gradient and laplacian therm in equation (2). For solution of Navier-Stokes equation (2) characteristic thing is that there is no independent equation for pressure, whose gradients contribute to each of three momentum equations, [8]. These difficulties are overcomed by use of numerical procedure called SIMPLE algorithm, [9], which is implemented in OpenFOAM. Preconditioned conjugate gradient methods are used for iterative solutions for system of linear equations.

Flow is impulsively starts from rest, and after the development period it's reaching the final state, which is periodic flow with shedding of vortices behind cylinder.

Some results are shown in figures 4 and 5. In figure 5 changes of lift and drag force coefficient are shown. Because of asymmetrical location of cylinder lift coefficient oscillates between values $C_{l,\min} = -1.054$ and $C_{l,\max} = 0.95$ and the drag coefficient oscillates between $C_{d,\min} = 3.17$ and $C_{d,\max} = 3.23$. The drag and lift forces oscillates at different frequencies - approximately, the drag has the twice the frequency as lift. Physical explanation is that drag force has one maximum and one minimum during growth and shedding of each vortex, while the sign of lift force depends on the location of vortex (above or below cylinder axis). The oscillation has also shifting in phase. That shifting is about 10% of the drag oscillation period.

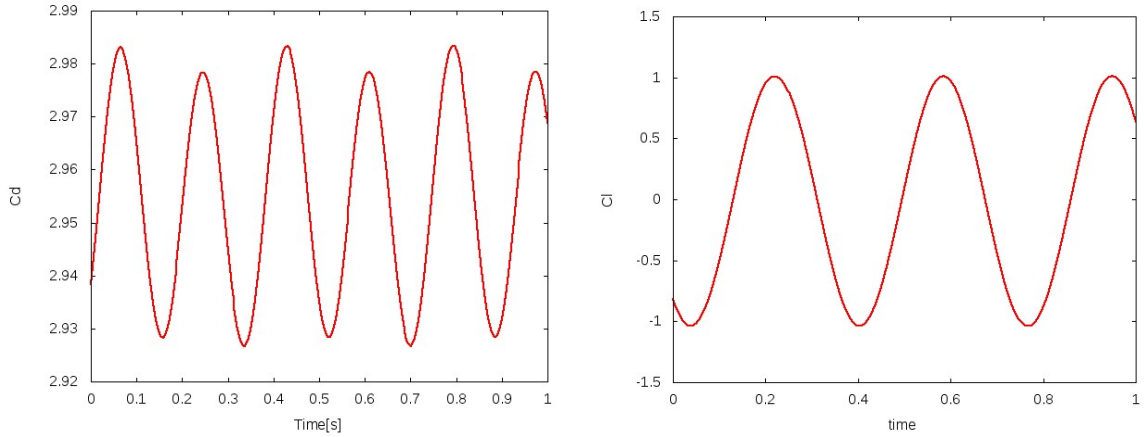

Figure 4: Time variations of drag and lift coeffients on cylinder (Re = 100).

Instantaneous contour lines of pressure and vorticity are shown in figure 5. We can see that some of the isobars are closed curved lines which indicates the location of vortex centers in which pressure has a local minimum.

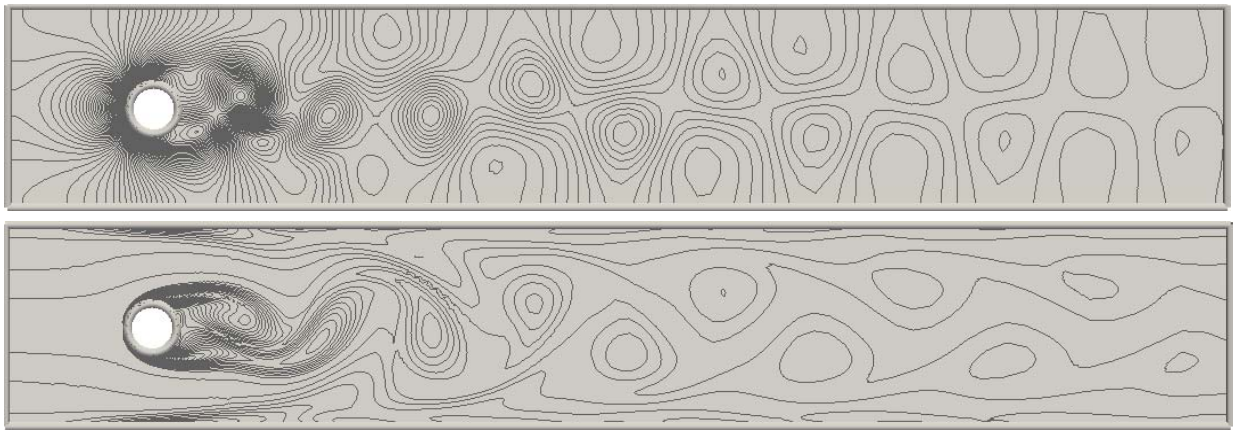

Figure 5: Instantaneous contours of pressure and vorticity in laminar flow around the cylinder in channel (Re = 100).

3.2 Modification of existing solver and creation of new one

We now want to solve temperature field in the same problem as in proceeding subsection. We are still solving laminar, incompressible and unsteady flow and we want to solve the energy equation for this type of flow. The energy equation, with neglecting the heat flux to the surrounding, has the form

$$\rho c_p \left[\frac{\partial T}{\partial t} + \vec{U} \cdot \nabla T \right] = \alpha \nabla^2 T + \mathbf{T} : \nabla \vec{U}, \tag{3}$$

where cp specific heat and _ is thermal conductivity of fluid. The last term in the equation represents viscous dissipation rate, a term which describes irreversible process of conversion of one part of fluid mechanical energy to internal energy. Usually, this term could be neglected in type of flow considered, but we'll keep it here, because we want to show how it is implemented in OpenFOAM code. T designates the viscous stress tensor, and for incompressible Newtonian fluid it has the form

$$\mathbf{T} = 2\mu \mathbf{S} = \mu \left[\nabla \vec{U} + \left(\nabla \vec{U} \right)^T \right] \tag{4}$$

where μ is fluid viscosity, $\nabla \vec{U}$ is velocity gradient tensor, and $\left(\nabla \vec{U} \right)^T$ is it's transpose. Before implementation in the code, equation (3) will be written in the form

$$\frac{\partial T}{\partial t} + \vec{U} \cdot \nabla T = k \nabla^2 T + \frac{\nu}{c_p} \left[\nabla \vec{U} + \left(\nabla \vec{U} \right)^T \right] : \nabla \vec{U} \tag{5}$$

where $k = \alpha / \rho c_p$ diffusion coefficient, and it's dimensions in SI units is $\left[\mathrm{m}^2/\mathrm{s} \right]$. Dimension of constant $C^* = \nu / c_p$ is Ks (Kelvin-second) and we have also to implement these facts in the code. First, let us compare values of α i C^* for water at $t = 20^\circ \mathrm{C}$. We get $\alpha = 1.364 \cdot 10^{-7}\ \mathrm{m}^2/\mathrm{s}$ and $C^* = 2.3889 \cdot 10^{-10}\ \mathrm{Ks}$, so in flows where there is no large velocity gradients the viscous dissipation rate could be neglected. But, as we said before, we'll keep this term in the code. It's interesting to see the results of temperature field for various values of parameter C^*.

It's not recommended to make the changes in precompiled OpenFOAM solvers. Instead, users should do changes in their local directories. We copied the solver `icoFoam` in local directory and create a new solver with `wmake` script. The name of the solver can be arbitrary - we called it `lamTempFoam`. The new lines of the code which are solving the equation (5) are the following:

```
// Odredi gradijent brzine
volTensorField gradU = fvc::grad(U);


// vrednosti nu, k i Cnu=nu/Cp se citaju u datoteci transportProperties

// Tenzor brzine deformisanja
volTensorField S = (gradU + gradU.T());

// Jednacina energije
fvScalarMatrix TEqn
(
  fvm::ddt(T)
  + fvm::div(phi, T)
  - fvm::laplacian(k, T)
  - Cnu * ( S && gradU ) // viskozna disipacija!
);

// Resi jednacinu!
    TEqn.solve();
```

We used new solver on same problem like in subsection 3.1. Now we have additional boundary condition for temperature: constant, fixed value of $300\ \mathrm{K}$ at the inlet and zero gradient at the outlet; we prescribed adiabatic channel walls and temperature of $350\ \mathrm{K}$ at the cylinder.

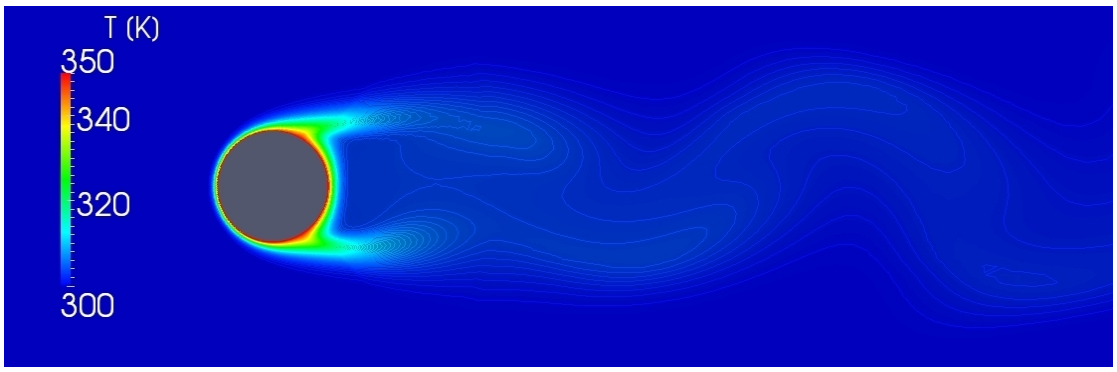

Figure 6: Instantaneous contours of temperature in cylinder wake with neglected viscous dissipation $(C^* = 0)$ and $\mathrm{Re} = 100$.

In figure 7 instantenous conture of temperature is shown in case of neclecting viscous dissipation $(C^* = 0)$. We used values of $\nu = 10^{-6}\ \mathrm{m^2/s}$ and $\alpha = 1.364 \cdot 10^{-7}\ \mathrm{m^2/s}$ (for water) and it can be seen that convection dominates in temperature evolution and distribution.

In next simulation we used the value of $C^* = 2.3889 \cdot 10^{-10}\ \mathrm{Ks}$ and we obtained more or less the same temperature distribution, from which we concluded that viscous dissipation rate has no significant effect to temperature distribution in this flow.

At the end we used physically unrealistic situation - we kept values of ν and α the same like in previous simulation but we significantly increased value of C^*, which we take to be $C^* = 0.02\ \mathrm{Ks}$. With this value we significantly increased the effect of viscous dissipation term which gives additional rise in temperature in flow field. This increasing is most significant near the walls and near the cylinder. This is expected, because we have larger velocity gradients near the walls.

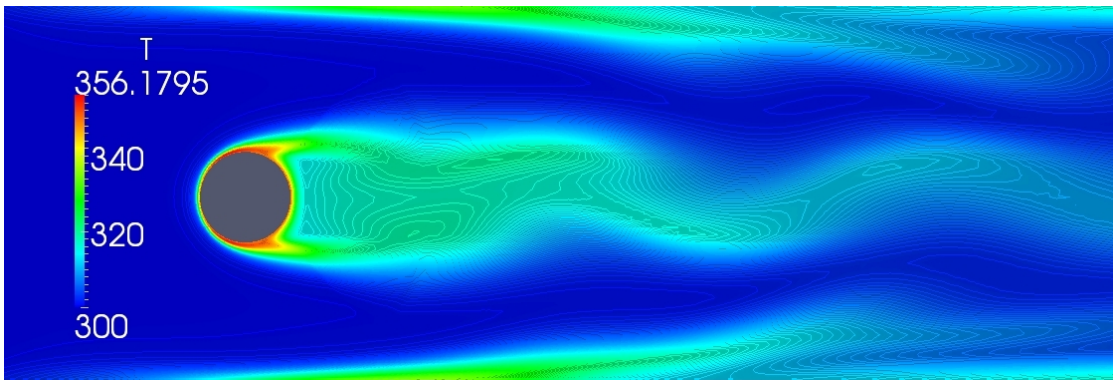

Figure 7: Instantaneous contours of temperature in cylinder wake with high viscous dissipation $(C^* = 0.02)$ and $\mathrm{Re} = 100$.

Real application of this solver could be in simulation of highly viscous flows, like flows of crude oil in pipelines. We are planning to additinally improve the solver by adding the temperature dependent fluid properties.

4. Concluding remarks

In this paper open-source CFD software OpenFOAM was used for calculations of vortex shedding which is formed in laminar flow around circular cylinder mounted in a channel. Obtained results have very good agreement with the results from the literature. Creation of new solver in OpenFOAM, which can handle energy equation in laminar flow of incompressible fluid is also presented. That was just a glimpse of capabilities of OpenFOAM. Extensive research in comparison between OpenFOAM and commercial CFD software by other authors showed that OpenFOAM can handle basically every problem like commercial software. Certanlly, commercial software have some utilities which OpenFOAM doesn't; also working environment is more user-friendly (well, it depends from viewpoint). But, the fact that OpenFOAM can not be used as "black-box", in our opinion is it's advantage. If you're a researcher in CFD you have to master three important things - mathematics, fluid dynamics and computer science. And OpenFOAM is great tool and companion in that road.

References

[1] Hunt J.C.R (1998) Lewis Fry Richardson and His Contributions to Mathematics, Meteorology and Models of Conflict, Annual Review of Fluid Mechanics, Vol. 30,
[2] Lynch P. (2008) The Origins of Weather Prediction and Climate Modeling, Journal of Computational Physics, vol. 227, pp. 3431-3444
[3] Harlow, F.H., and Nakayama P.I (1967) Turbulence Transport Equations, Phys. Fluids, Vol. 10, pp. 2323-2332
[4] Stallman R.M. (2002) Free Software, Free Society - Selected Essays of Richard. M Stallman, GNU press
[5] Weller, H.G.; Tabor G.; Jasak, H. and Fureby, C. (1998) A Tensorial Approach to CFD using Object Orientated Techniques, *Computers in Physics*, Vol. 12 No. 6, pp 620 - 631
[6] OpenFOAM User Guide, 2010, http://www.openfoam.com/docs/
[7] OpenFOAM Programming Guide, 2010, http://www.openfoam.com/docs/
[8] Ferziger J.H., Peri' (2002) Computational Methods for Fluid Dynamics, 3rd Edition, Springer, 2002
[9] Patankar S.V. and Spalding D. B. (1972) A Calculation Procedure for Heat, Mass and Momentum Transfer in 3-Dimensional Parabolic Flows, Int. J. Heat Mass Transfer, Vol. 15, pp. 1787-1806.