



# OPEN-SOURCE CFD ANALYSIS OF MULTI-DOMAIN UNSTEADY HEATING WITH NATURAL CONVECTION

PAWEŁ SOSNOWSKI AND JACEK POZORSKI

*Institute of Fluid-Flow Machinery,  
Polish Academy of Sciences,  
Fiszera 14, 80-952 Gdansk, Poland  
psosnows@gmail.com, jp@imp.gda.pl*

(Received 15 September 2009)

**Abstract:** The paper presents the capabilities of open-source programs used for numerical simulation of fluid flow and heat transfer phenomena. These tools are applied for unsteady problems with natural convection and, in particular, the heating of an automotive headlight. To perform these studies, a solver was created using an open-source C++ library `OpenFOAM`. The implementation problems encountered were, inter alia, dealing with multi-domain computation and coupling of sub-domains. The presented simulation results clearly show that open-source software can readily be applied also to engineering problems and can compete with commercial packages.

**Keywords:** CFD, open-source software, unsteady heating, natural convection, multi-domain heat transfer

## 1. Introduction

Numerical simulations have become an important part of research and development (R&D) industrial activities over the past years. Nowadays, computational fluid dynamics (CFD) and computational heat transfer (CHT) methods have reached a certain degree of maturity, *cf.* [1]. They are often the first choice when dealing with problems involving thermomechanics of fluids in the initial phase of a design process. This has resulted in the development of numerous commercial software packages which have made it possible to perform numerical simulations. As an alternative, open-source programs started to be created as well. This approach allows the cost of performing such studies to be significantly reduced, since the price of an industrial license for a commercial CFD package on a single workstation can sometimes reach several thousand euros per month. Furthermore, the engineer has access to basically the same functionality as far as mathematical models and ease-of-use are concerned.



The main aim of the work presented in this article was to create a numerical solver for unsteady simulations of heating of an automotive headlight and to perform such a simulation. Natural convection and heat transfer inside automotive headlights are well understood [2]. Nowadays, the main attention is focused on improving the geometry of headlights using the available experimental and simulation tools. Most of the work in the R&D sector of the automotive industry is done with commercial software. At the same time, a great amount of effort has been put by the open-source community to develop reliable and accurate tools for CFD calculations. One of such projects is `OpenFOAM` which is an open-source C++ library. It has been applied and further developed in the present work.

In Section 2 the basic theory used in the present study and the most important features of the created solver are introduced. In Section 3 open-source CFD tools: the `OpenFOAM` library as well as pre- and post-processing programs are presented. In Section 4 the process of creation of the main solver, the actual calculation performed and the obtained results are described.

## 2. Physical problem considered

This Section introduces physical aspects of the studied case with heat transfer [3]. It also presents the mathematical model and the most important features of the created solver.

### 2.1. Governing equations

Firstly, the general equations and theory used in the work are introduced. All the equations can be written in the form of a generic transport equation (further discretised using `OpenFOAM`) for variable  $\psi$ :

$$\frac{\partial}{\partial t}(\rho\psi) + \nabla \cdot (\rho \mathbf{U}\psi) = \nabla \cdot (\gamma \nabla \psi) + S, \quad (1)$$

where  $\rho$  is density,  $\mathbf{U}$  is the velocity vector,  $\gamma$  is the diffusion coefficient and  $S$  is the source term. Using this general formula, the mass conservation equation ( $\psi \equiv 1$ ) can be written:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{U}) = 0. \quad (2)$$

Next, the momentum equation ( $\psi \equiv \mathbf{U}$ ) is recalled:

$$\frac{\partial}{\partial t}(\rho \mathbf{U}) + \nabla \cdot (\rho \mathbf{U}\mathbf{U}) = \mu \nabla^2 \mathbf{U} - \nabla p + \mathbf{B}, \quad (3)$$

where  $\mu$  is the dynamic viscosity,  $p$  is the pressure and  $\mathbf{B}$  are the mass forces acting on a unit mass of fluid (in the following: the buoyancy force). At last, the energy equation is taken into account, which after several modifications can be written as the equation describing the behaviour of temperature ( $\psi \equiv T$ ):

$$\frac{\partial T}{\partial t} + \nabla \cdot (\mathbf{U}T) = \frac{\kappa}{\rho c_p} \nabla^2 T + \frac{S_h}{\rho c_p}, \quad (4)$$

where  $\kappa$  is the thermal conductivity,  $c_p$  is the specific heat and  $S_h$  is the heat generation rate per unit volume.

As in most numerical applications, the Eulerian discretisation scheme was applied in the form of the finite volume method [1]. This involved a division of the computational domain into small volumes (cells), writing a set of conservation equations for each cell and solving the resulting system of non-linear equations.

### 2.2. Multi-domain approach

The studied case consisted of nine sub-domains (solid or fluid regions) which differed in physical properties. The selected solution algorithm was based on the assumption that each region was dealt with sequentially. The coupling between them was modelled by specific consistency conditions at the common boundaries (interfaces). Two physical models were required to simulate the conjugate heat transfer. For solid sub-domains, it was enough to consider the temperature field and its evolution in time. The following equation was solved:

$$\rho_s c_{p,s} \frac{\partial T_s}{\partial t} = \nabla \cdot (\kappa_s \nabla T_s) + S_s, \quad (5)$$

where  $\rho_s$  was the density,  $c_{p,s}$  was the specific heat,  $T_s$  was the temperature,  $\kappa_s$  was the conduction coefficient, and  $S_s$  was the heat source in the solid element. In the fluid sub-domains, the flow of a viscous, thermodynamically perfect gas was considered. Also, the Boussinesq approximation was used to account for the natural convection effects. It assumes that the thermal properties of fluid do not greatly influence its density except for the buoyancy force term. This results in a possibility of approximating it by a constant,  $\rho \cong \rho_0$ . In the end, the density itself can be calculated by an independent relation,  $\rho = \rho(T)$ . These relationships state that the difference in temperature can generate movement inside fluid domains. The final set of equations could be written as:

$$\nabla \cdot \mathbf{U}_f = 0, \quad (6)$$

$$\frac{\partial \mathbf{U}_f}{\partial t} + \nabla \cdot (\mathbf{U}_f \mathbf{U}_f) - \nabla \cdot \left( \frac{\mu}{\rho_f} \nabla \mathbf{U}_f \right) = -\epsilon (T_f - T_0) \mathbf{g} - \nabla p, \quad (7)$$

$$\frac{\partial T_f}{\partial t} + \nabla \cdot (\mathbf{U}_f T_f) - \nabla \cdot \left( \frac{\kappa_f}{\rho_f c_{p,f}} \nabla T_f \right) = 0, \quad (8)$$

where  $\mathbf{U}_f$  is the velocity of the fluid,  $\mu$  is the dynamic viscosity,  $\rho_f$  is the density,  $\epsilon$  is the thermal expansion coefficient,  $T_f$  is the fluid temperature,  $T_0$  is a reference temperature,  $\mathbf{g}$  is the gravitational acceleration,  $p$  is the pressure, and  $\kappa_f$  is the thermal conductivity.

### 2.3. Coupling condition, boundary conditions

In most multi-domain, conjugate heat transfer computations, solid and fluid domains are in contact. In the selected approach, each domain was solved separately at each time step. A difficulty appeared in the appropriate formulation of the consistency conditions at the solid-fluid interfaces, as well as the numerical implementation of these conditions to couple temperature fields in neighbouring regions. The main problem was due to significant differences in heat conduction coefficients of neighbouring regions (like air and glass). It appeared that (contrary

to the first idea) it was not enough to copy the temperature value from the next-to-interface cell centre to the boundary of a studied region. It was necessary to create a coupling condition which was based on matching the heat fluxes at both sides of the interface. The formula developed during the studies of the problem could be written as:

$$T_{\text{int}} = \frac{T_{\text{own}}\lambda_{\text{own}}D_{\text{nei}} + T_{\text{nei}}\lambda_{\text{nei}}D_{\text{own}}}{\lambda_{\text{own}}D_{\text{nei}} + \lambda_{\text{nei}}D_{\text{own}}}, \quad (9)$$

where  $T_{\text{int}}$  is the modelled temperature at the interface,  $T_{\text{own}}$  is the temperature inside the studied region's cells next to the interface,  $T_{\text{nei}}$  is the temperature inside the next-to-interface cells in the neighbour region;  $\lambda_{\text{own}}$  and  $\lambda_{\text{nei}}$  are thermal conduction coefficients,  $D_{\text{own}}$  and  $D_{\text{nei}}$  are the distances between cell centres and the interface. As can be seen, the temperature at the interface was not only dependent on the temperature of connected regions and their conduction properties, but also on the distance between the interface and the centres of the next-to-interface cells. Although this condition proved itself to work and to provide a convergent solution, it resulted in significant dependency of the temperature distribution on the mesh structure close to the interface. These effects will be described later on in the paper.

Another aspect of the studied case was the external boundary condition. In the presented simulation it was assumed that the whole case was situated inside a container (thermostat) which could absorb pre-defined amounts of heat.

#### 2.4. Numerical and computational improvements

In problems concerning heat transfer from a specific source, regions which are far away from the heat source are not affected by the temperature rise at the beginning of the simulation. Therefore, the program does not have to compute those regions (sub-domains). The created solver has been provided with such a functionality. A particular region (sub-domain) starts to be computed when at least one of the following conditions becomes true:

- there is a heat source in the region;
- there is a flow source in the region (*e.g.*, inflow, moving wall, *etc.*);
- there is a pressure or temperature gradient inside the region;
- temperature at the boundary differs from the temperature inside the region.

The presented feature allows us to save computational resources during the initial phase of the simulation.

Another specific ability of the solver is the way it handles the material properties. For example, the density of a solid or fluid may change with the rising temperature. The nature of those changes can be described by a pre-defined function or by interpolation from experimental data. The solver has been extended to have the possibility to modify the properties in both ways. The material properties can be inserted as a polynomial function or by a set of experimental data which is to be interpolated.

### 3. Open-source CFD software

This section is dedicated to introducing the implementation of algorithms used in computational fluid dynamics and heat transfer. The commercial market for such programs is very large. The available products provide accurate and reliable tools. On the other hand, two main flaws of such solutions are noticed. The first one is the price of commercial programs. A great complexity and variety of functionalities reflect the cost of such a tool. At the same time, it often happens that most functions offered by such a product are not used. The second main problem is that the source code of commercial applications is not available. Engineers working with such programs cannot include their own modifications. This prevents them from implementing newly developed algorithms. Usually, if a new functionality is to be included into a commercial code, it has to be purchased from the developer. And once again the money problem is encountered.

These flaws are not present if the developing team uses open-source codes. Studies done for this paper were performed with **OpenFOAM** (Open Field Operation And Manipulation) software [4, 5] which is released under the Lesser General Public License (LGPL).

#### 3.1. Open-source pre- and post-processing

The open-source community have created many professional, free tools for working in all phases of numerical analysis of field problems: pre-processing, solving and post-processing. Some of them, used for the present purpose, are presented in this paper.

##### 3.1.1. OpenFOAM pre-processing

The first contact with **OpenFOAM** always involves working with provided pre-processing tools: **blockMesh** and **snappyHex**. The first tool allows to create simple geometries and structural grids. Unfortunately, it lacks a graphical interface, and all input is done by creating a specific file called **blockMeshDict**. It is helpful to get familiar with **blockMesh** since with it comes the knowledge on how the data and the mesh are taken care of in **OpenFOAM**. On the other hand, while preparing complex or unstructured grids, **blockMesh** proves to be insufficient. The second mentioned tool, **snappyHex**, is a mesh generating program. It makes it possible to create a hexagonal, unstructured mesh using STL formatted geometry files as input.

##### 3.1.2. Pre-processor CalculiX

Next, an open-source pre-processing tool that should be mentioned is **cgx**. It is an environment which comes together with the **CalculiX** package; **cgx** allows geometries and meshes to be created. It has advanced structural meshing algorithms implemented. The program interface is very basic and requires some practice to be efficiently used, however, work with **cgx** becomes rewarding when some experience has been gained. Unfortunately, up to now, only hexagonal-element meshes can be saved in the **OpenFOAM** format. Thus, **cgx** can generate STL surface meshes which can be used by other meshing algorithms.

### 3.1.3. Meshers Netgen and enGrid

Open-source codes allow unstructured grids of good quality to be created. In particular, the **Netgen** program provides a very efficient meshing algorithm. The program itself provides sufficient functionality. On the other hand, it does not give access to several helpful tools (such as near-wall boundary layer mesh creators). Another program, **enGrid**, supports those features, while using the same, efficient algorithm within itself. It also provides full export compatibility with **OpenFOAM**. An example of a mesh created using **cgx** and **enGrid** can be seen in Figure 1.

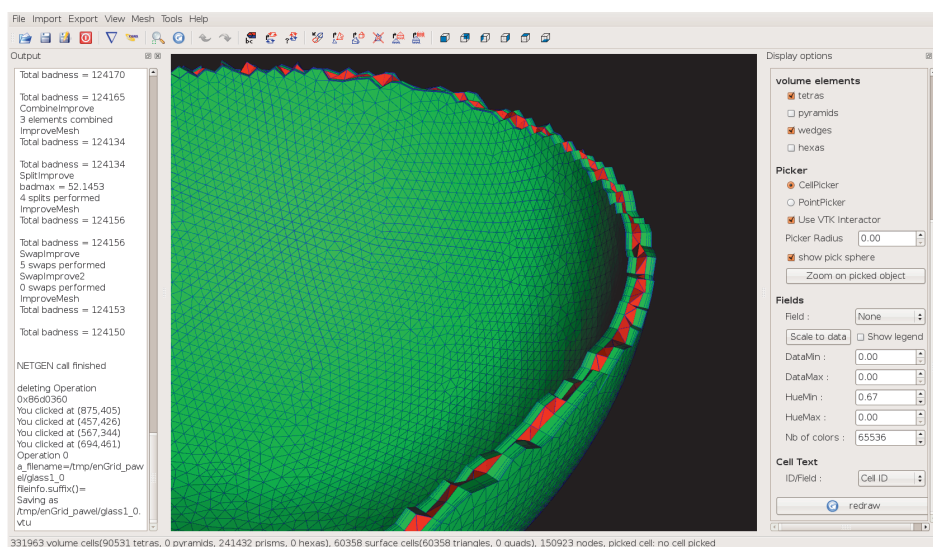


Figure 1. Mesh created using **cgx** and **enGrid**

### 3.1.4. Post-processing tool ParaView

One of the important steps while performing numerical simulations is the post-processing part. The open-source community has produced tools which can be as good (or even better) as many commercial solutions. One of them is the **ParaView** visualisation software. The program makes it possible to visualize multiple formats of data, create animations, screenshots, *etc.* At the same time **ParaView** has an easy to operate and intuitive interface and no special training is required to fully use its capabilities.

### 3.1.5. Advantages of OpenFOAM

**OpenFOAM** differs from most of the numerical libraries by the way it takes care of implementation of differential transport equations. Thanks to its high-level, object oriented C++ structure, the equations can be put into code in a very clear and easy way [4, 5]. For example, a transport equation:

$$\frac{\partial}{\partial t} (\rho_s c_{p,s} T_s) = \nabla \cdot (\kappa \nabla T_s) + \nabla \cdot \mathbf{Q}_{\text{rad}} + P$$

will be represented in **OpenFOAM** by such an object as in Figure 2.

```
fvMatrix Eqn {
    fvm::ddt(rhos*Cps, Ts)
    ==
    fvm::laplacian(K, Ts)
    + fvc::div(Qrad)
    + P
};
```

Figure 2. OpenFOAM's representation of a transport equation

It can be easily seen that the representation is very intuitive. Furthermore, an engineer does not have to know exactly how the `fvMatrix` class works. Instead, it is enough for him to understand and remember its functionality. This substantially accelerates the development process.

## 4. Results for test cases

The following part describes the process of testing the OpenFOAM environment and creation of a solver which was used in the main simulation.

### 4.1. Solver *icoFoam*

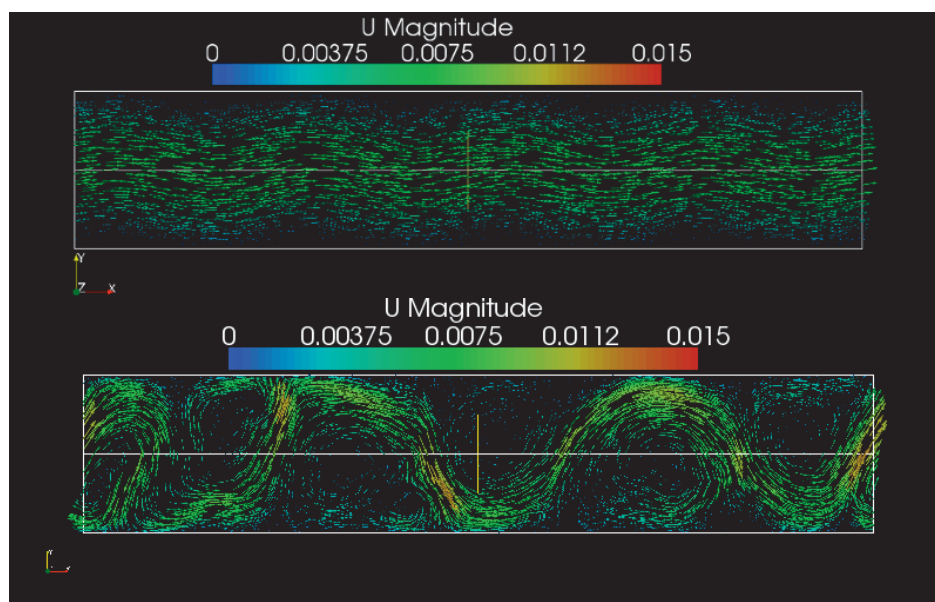
The first test involved checking *icoFoam*. It is a solver included in OpenFOAM releases which allows incompressible flow cases to be stimulated. It does not include an energy equation. The studied domain was a two-dimensional square cavity with a moving top wall. Three tests were performed, each with a different Reynolds number. The results were compared with the paper of Ghia *et al.* [6]. The data obtained from *icoFoam* agreed well with those from the reference. This allows us to assume that the algorithms implemented in OpenFOAM work correctly. For the next test, a third dimension was added. The newly created walls were chosen to be periodic. That way an infinite cavity with a moving top wall was modelled. The test was aimed at checking whether the same solver would be able to recognize and properly handle such an extended case. The results confirmed the expectations.

### 4.2. Modified *icoFoam* solver

The third test involved a modification of *icoFoam* by adding mass force to the momentum equation and including the temperature equation as a passive scalar. The studied case consisted of an infinite, two-dimensional, vertical channel. The initial temperature distribution across the channel was set to be parabolic. The channel walls were held at different, fixed temperatures (isothermal boundary condition). The calculations were performed until a steady state laminar flow was reached. As expected, the resulting velocity field had a parabolic profile, and the temperature ended up with a constant gradient, linear profile.

### 4.3. Rayleigh-Bénard convection

This analysis tested the equation set which was used in the main program. The studied problem was the Rayleigh-Bénard natural convection process [3]. The case consisted of two infinite, horizontal plates of which the lower one was hotter than the upper one. To simplify the model, the Boussinesq approximation was used, hence the fluid density was considered to be constant. For this setup, two characteristic states could be observed. In the former, the fluid stopped moving due to its viscosity and the heat was transported by molecular diffusion. The latter state was observed when the so-called “convective cells” appeared inside the domain. The fluid circulated between hotter (lower) and colder (top) walls, so that the heat was transferred mostly by convection. The tests were performed for stable, unstable, and marginal-state configurations. The final results are presented in Figure 3. Here again, the results matched the expectations. This allowed us to start working on the final, multi-region solver.



**Figure 3.** Rayleigh-Bénard convection; final simulation state: (top) stable configuration, (bottom) unstable configuration

## 5. Results for generic automotive headlight

The open-source solver developed and tested for heat transfer due to natural convection was used for simulation of heating of a headlight with a P21W bulb, commonly used in the automotive industry. To simplify the model, calculations were made in two dimensions. The case consisted of nine sub-domains: filament, gas inside the bulb, glass, gas inside the bulb base, bulb thread, socket, box, gas inside the box, and gas surrounding the box. The mesh used is presented in

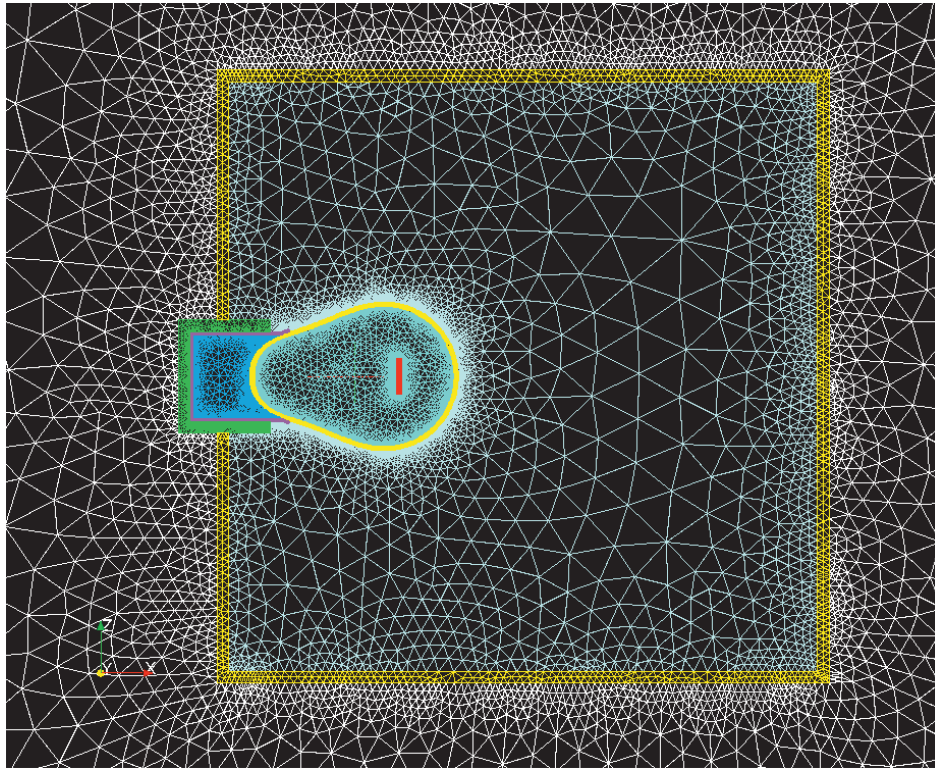


Figure 4. Unstructured mesh used in the main simulation

Figure 4. It consisted of around 100000 cells. The time step was chosen to be  $\Delta t = 10^{-4}$ s.

The first simulations were performed on a coarse mesh of 5500 cells. They were aimed at testing the solver and producing first results. An analysis of those results led us to remove the filament and replace it with a fixed value boundary condition. It appeared that the model could not fully simulate the real process, since it lacked radiation effects.

### 5.1. Velocity and temperature fields

The main simulation lasted for over three weeks of the CPU time. 160 seconds of full computational time (temperature + velocity) were generated and followed by another 2400 seconds of temperature simulation. The results obtained from the full simulation can be divided into two parts. The first one lasted up to 30 seconds. By that time, the gas inside the bulb was nearly in a steady state. In addition, the glass of the bulb started to heat the gas inside the box. This resulted in creating a heat flow which was pointed directly upwards from the bulb. The results are illustrated in Figure 5 (left).

After  $t = 30$ s, the flow started to move leftwards, towards the wall of the box. A great vortex was generated inside the box. This fact was a result of the simplification to two dimensions. The gas which was heated by the bottom of the

bulb tended to move upwards. Since it had no other way out, it moved around the bulb. At the same time it powered the big vortex, and moved the horizontal flow to the left. The results till that time are shown in Figure 5 (right).

At  $t = 160\text{s}$  the velocity field was frozen and averaged (Figure 6). After that the simulation was performed only for the temperature field. Without this action, the calculations would have had to continue for several months (the simulation was running on a personal computer with a 2.1GHz dual-core processor, 3GB RAM). At  $t = 3000\text{s}$ , the temperature reached a steady state in all control points, and the simulation was stopped (Figure 7).

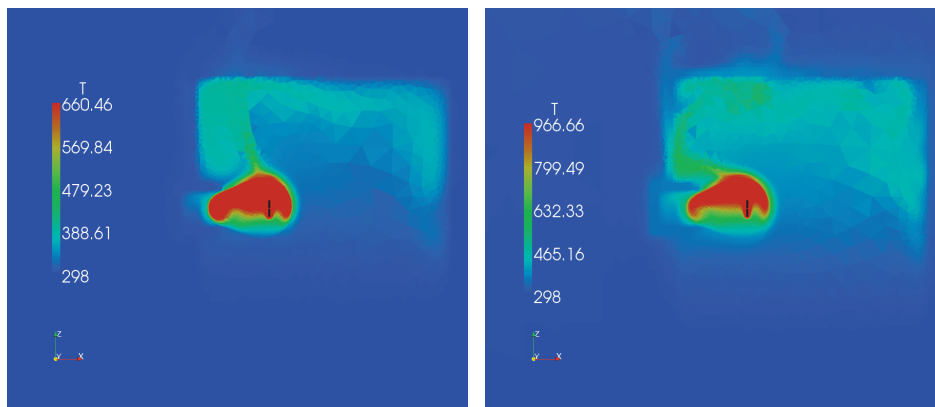


Figure 5. Transient temperature fields at: (left)  $t = 30\text{s}$ , (right)  $t = 160\text{s}$

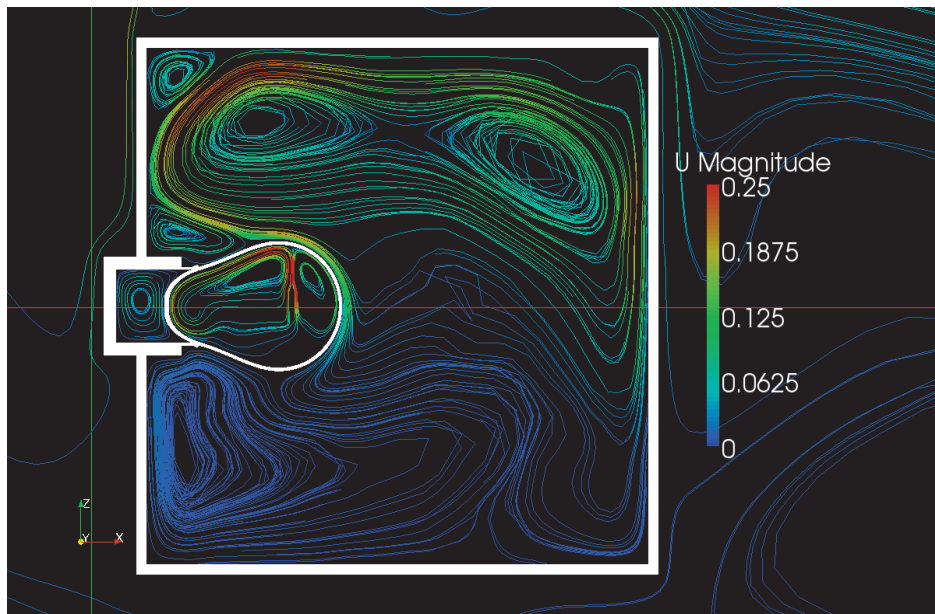


Figure 6. Stream lines of averaged velocity field

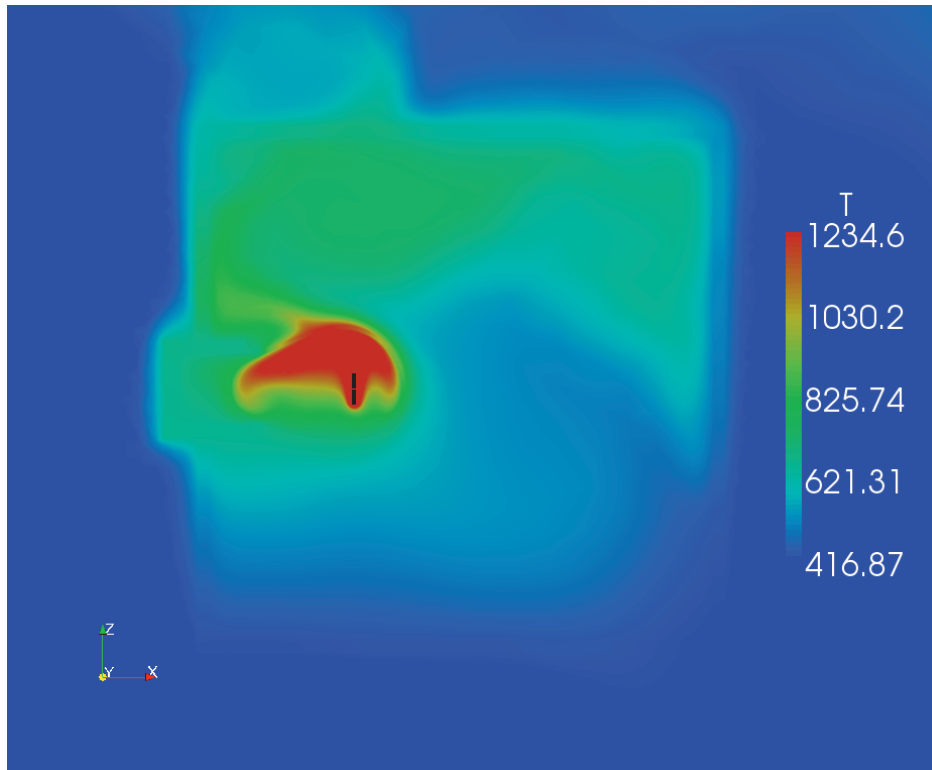


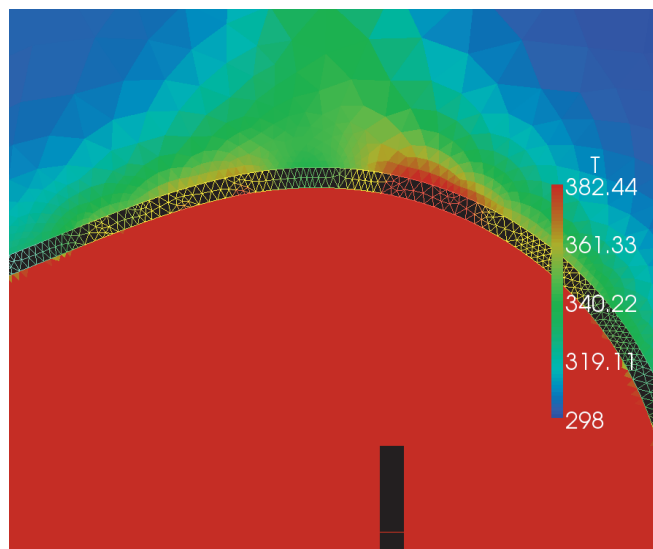
Figure 7. Steady-state temperature field

### 5.2. Discussion

The obtained results matched the expectations qualitatively. A so called “hot spot” was observed in the upper part on the glass of the bulb, *cf.* Figure 7, in agreement with the experimental evidence [2]. On the other hand, a simplification of the physical model (by not including radiation) led to a higher value of temperature at the “hot spot” than the temperature obtained during the experiment. This opens a field for further development of physical models used in the solver. At the same time, a significant dependency of the temperature distribution on the mesh quality was observed near the interfaces of the sub-domains. It was caused by the properties of the selected coupling model. This problem occurred in the initial state of the simulation (around  $t = 4$ s). It can be seen in Figure 8. This fact points out to the need of a specific near-wall mesh handling while using the model, and indicates the direction of future studies on the coupling problem.

## 6. Conclusions

The paper has shown that it is possible to simulate advanced heat and fluid flow phenomena using open-source programs and environments. The `OpenFOAM` solver was further developed by adding the consistency conditions for conjugate heat transfer. The described solver made it possible to simulate the temperature



**Figure 8.** “Temperature bubbles” caused by unstructured mesh effects at solid-fluid interface

field behaviour for a multi-domain computation where thermal convection and conduction were the main heat transfer factors. In addition, several free, advanced numerical programs for pre-processing and post-processing were used for the purpose, as described in the text.

### ***Acknowledgements***

The paper presents some of the results obtained in the Master Thesis of P. S., partly prepared at the University of Trieste (Italy) in the group of Prof. Vincenzo Armenio, in association with Eng. Antonio Filipuzzi from Centro Ricerche Plasta-Optica (Udine, Italy). The authors also wish to thank the OpenFOAM community for their continuing efforts in developing open-source CFD software.

### ***References***

- [1] Fletcher C A J 1988 *Computational Techniques for Fluid Dynamics*, Springer
- [2] Filipuzzi A and Zanoletti F 2007 *EnginSoft CAE User’s Meeting 2007*
- [3] Atkinson J and Rubin H 2001 *Environmental Fluid Mechanics*, Marcel Dekker Inc.
- [4] 2008 *OpenFOAM Programmers Guide*, <http://www.open CFD.co.uk/openfoam/>, OpenCFD Limited
- [5] 2008 *OpenFOAM User Guide*, <http://www.open CFD.co.uk/openfoam/>, OpenCFD Limited
- [6] Ghia U, Ghia K N and Shin C T 1982 *J. Comput. Phys.* **48** 387