Chair of Petroleum and Geothermal Energy Recovery

Master Thesis

Numerical Study of Steady State and Transient Heat Conduction using OpenFOAM

Written by:

Mariia MIRONOVA.

01535695

Advisor:

Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Wilhelm Brandstätter

Leoben, 2nd.September 2018

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Masterarbeit selbstständig und ohne fremde Hilfe verfasst, anders als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

Leoben, am 2 Oktober.2018

Mariia MIRONOVA

Affidavit

I hereby declare that the content of this work is my own composition and has not been submitted previously for any higher degree. All extracts have been distinguished using quoted references and all information sources have been acknowledged.

Acknowledgements

Foremost, I would like to express my deepest appreciation to my supervisor, Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Wilhelm Brandstätter for sharing his time and knowledge and for keeping his office door open day and night, to discuss the challenges and results of my ongoing research.

I am very grateful to and Head of the Chair of Petroleum and Geothermal Energy Recovery, Univ.-Prof. Dr. Herbert Hofstätter for his continuous support, motivation and for sharing his broad practical and theoretical knowledge throughout my studies at Montanuniversität Leoben.

I am very thankful for the close cooperation, interesting discussions and most important the valuable support during the research for this thesis with Dr. Mont. Petr Vita.

I may take this opportunity to express a deep sense of gratitude to my family, David and all my friends. They are all truly wonderful people, and I am looking forward to a bright and shared future together.

Kurzfassung

Diese Diplomarbeit wurde als unterstützende Literatur für die Computational Continuum Mechanics Lehrveranstaltung an der Montanuniversität Leoben verfasst. Sie soll vor allem Bachelor- und Masterstudenten den Umgang mit der Software Open FOAM erleichtern. In dieser Arbeit werden die Grundlagen von Wärmeübertragung, sowie die analytischen und numerischen Herangehensweisen für Probleme der Wärmeleitung behandelt.

Das erste Problem gibt eine Einführung in die Wärmeleitung mit Hilfe einer zweidimensionalen (2D) Platte im Gleichgewichtszustand. Um eine numerische Lösung zu erhalten, wird eine Schritt für Schritt Anleitung für dieses Beispiel in OpenFOAM gegeben. Desweiteren werden die Ergebnisse dieses OpenFOAM Falles mit den theoretischen Berechnungen verglichen und mögliche Ursachen für die vorhandenen Fehler erläutert.

Das zweite Beispiel behandelt den eindimensionalen (1D) Gleichgewichtszustand eines Wärmeleitungsproblems im Falle eines isolierten Zylinders. Dieser Teil der Diplomarbeit beschreibt den Effekt des kritischen Isolationsradius einer nicht planaren Geometrie, im Falle einer Abkühlung eines heißen dünnen Rohres auf die Umgebungstemperatur. Die detailierte Beschreibung mit allen Programmiercodes werden ebenfalls angeführt. Am Ende dieses Kapitels werden die analytischen und numerischen Lösungen miteinander verglichen.

Als drittes Beispiel wurde ein instationäres eindimensionales (1D) Wärmeleitungsproblem, der Abschreckungsprozesses einer Stahlplatte gewählt. Wie bei den vorherigen Fällen werden die theoretischen und rechnerischen Lösungen präsentiert. Weitere Recherchen dieses Problems geben eine Erklärung für die auftretenden Fehler zwischen den Lösungen.

Im letzten Teil dieser Diplomarbeit wird die Wärmeübertragung in einem komplexen dreidimensionalen (3D) Gebilde simuliert. Dies soll zeigen, dass eine komplexe Geometrie durch die gleiche Herangehensweise behandelt wird, wie ein selbiges Problem bei einfacheren Körpern.

Abstract

This master thesis serves as a support for the Computational Continuum Mechanics course for advanced undergraduates and first-year graduate students at Montanuniversitaet Leoben. It discusses the fundamentals of heat transfer phenomenon, and the analytical and numerical approaches to heat conduction problems are reviewed.

The first problem introduces steady state heat conduction in two-dimensional (2D) slab. In order to obtain a numerical solution, the step-by-step description of this case in OpenFOAM is given. Then the results of the OpenFOAM case are compared to theoretical calculations, and possible reasons for the existing errors are noticed.

The second example examines the one dimensional (1D) steady state heat conduction problem in terms of cylinder insulation. This part describes the effect of critical insulation radius in case of non-planar geometries when a hot thin pipe is cooled down to an ambient temperature. The detailed set up with all codes to run a successful case are presented. At the end of this chapter the comparison between the analytical and numerical solution is performed.

Dealing with 1D transient heat conduction problem, quenching a steel plate process is taken as the third case. As with the previous cases, the theoretical and computational solutions are shown. Further investigation of the problem offers the explanation of certain errors in the numerical solution.

The last part, heat transfer in a complex three-dimensional (3D) figure, is simulated in order to show that in spite of the complicated geometry a problem shall be approached in the same manner as a problem in simpler forms.

List of Figures

List of Tables

Table of Content

1 INTRODUCTION

While designing a wide range of industrial processes and equipment petroleum engineers face the issues of heat transfer, such as convection and conduction heat transfer, thermal conductivity, phase transformations and others. Nowadays the science uses different approaches to analyse and enhance heat transfer for single and multiphase systems by basic heat transfer concepts, the fundamental modes of heat transfer, implementing thermophysical properties of materials, numerical methods, computational methodologies, modelling and simulation. Nevertheless, some methods like analytical solutions due to its limitations are not applicable for cases that are more complex.

For example, in the case of materials that have thermal conductivity, which varies slightly with temperature, constant thermal conductivity is generally assumed. However, if temperature change is substantial or the thermal conductivity varies greatly with temperature, the assumption of constant thermal conductivity may lead to significant errors in the solution. Therefore, when modelling and simulating temperature distribution for such problems, non-linearities caused by temperature-dependent thermal conductivity have to be accounted for by the numerical computation.

In the present thesis work, both analytical and numerical solutions are observed for steady-state and transient heat conduction problems, and simple 1 and 2-dimensional examples are investigated using the computational fluid dynamics software OpenFOAM.

1.1 Problem Definition

During the past decades the importance of new methodology for attacking the complex problems caused by heat transfer has increased, and this technology is called Computational Fluid Dynamics (CFD). In this computational approach, the equations that govern a process of interest are solved numerically. Although experimentation and theoretical methods continue to be important, especially when the flows involved are very complex, the trend is clearly toward greater precision using computer-based predictions. Due to the enhanced computer power, the general understanding of the capabilities and limitations of algorithms has increased.

1.2 Objective

This thesis is intended to serve as a text for a part of Computational Continuum Mechanics (CCM) course for advanced undergraduates and/or first-year graduate students at Montanuniversitaet Leoben. This guide, even though without warranty that all the details of the subject are covered, will help a learner to understand basics of the topic and explain how to set up the first cases of heat conduction problems by him/herself.

Therefore, the material has been divided into two parts. The first part presents basic concepts and introduces the reader to the fundamentals of finite volume method and OpenFOAM general working process. The second part consists out of three different heat conduction cases, for which numerical method is applied, and the results are further compared with a known analytical solution.

For a better understanding of the numerical approach to solve a problem in fluid mechanics and heat transfer the more complex 3D model is constructed, and based on this geometry heat transfer due to conduction in the body is simulated.

In the conclusion part of the thesis, all the results are summarized. Furthermore, advantages and disadvantages of computational method are revised with its limitations and constraints.

2 FUNDAMENTALS

As this thesis focuses on development of reliable numerical solutions for heat conduction problems, the basics of these topics are described first. Furthermore, the analytical solution approach for addressed problems of heat transfer is discussed. For a better understanding of the workflow and results of the solutions, the OpenFOAM software's general structure is observed.

2.1 Heat Conduction

Where temperature differences exist between two or more objects, and they are not isolated from each other, then heat will flow from the point of higher temperature to the point of lower temperature until the temperatures are equalized. This phenomenon in science is called heat transfer. There are different modes of heat transfer marked:

- Conduction is consequence of electron vibrations between the rigidly fixed molecules in the lattice of a solid substance or of direct molecular interaction in liquid and gases.
- Thermal convection is a heat transfer from one point to another by a fluid movement.
- Thermal radiation is electromagnetic radiation generated by the thermal motion of charged particles in body. [1]

Conduction, as it was mentioned above, is the transmission of heat through a substance without noticeable motion of a substance itself. Heat could be conducted through gases, liquids, and solids. When talking about liquids, conduction is a primary heat transfer mechanism whilst there is zero bulk velocity in the fluid. In non-transparent solids, conduction is the only possible mode of heat transfer. In the gases, the kinetic energy of the molecules is associated with the property called *temperature*. In higher temperature regions, gas molecules have higher velocities than same molecules in lower temperature zones. The random motion of molecules leads to collisions and an exchange of energy and momentum. If temperature gradient is present in the gas, those molecules through random motions and collision transfer some of energy to molecules in the low temperature region. [2]

The situation in liquids with heat conduction is more complex than in gases due to the tighter spacing between molecules, however, the mechanism of heat transfer itself is the same. Hence, molecular force and bonds have an effect on the energy transmission between molecules. [2]

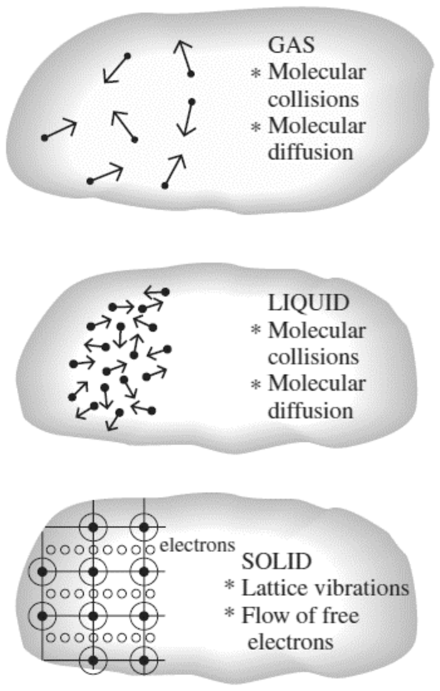


Figure 1: The mechanism of heat conduction in different substances [3]

Heat conduction in solids, on the other hand, is believed to happen due to motion of free electrons, lattice waves, and magnetic disturbance. The motion of free electrons can take place only in those substances, which are considered to be good electrical conductors, such as silver, gold, and cooper. The theory below this statement is that heat can be transported only by electrons, or so-called electron gas, which can easily move through the metal lattice (similar process to electricity conduction). Figure 1 shows the different mechanisms of heat conduction in solids, gases and liquids.

In a substance, the molecular energy of vibration is transmitted between nearby atoms or molecules from a region with high to low temperature. These physical phenomena can be seen from lattice waves or energy being transmitted be a gas composed of particles, known as phonons. Phonon motion happens due to diffusing through the lattice in the same way as the electron gas. This is another considerable factor in conduction through non-metals. However, it is at the least importance for heat conduction in metals.

Another important factor for heat conduction in solids can be effects of magnetic dipoles between adjacent atoms, which can also result in energy transmitting. This effect in physics is called electromagnetic radiation, and when the material has little capacity of absorbing energy, the conduction of heat is significantly dependent on the mentioned physical phenomenon.

Even though it is very important to understand all the factors influencing heat conduction, however, the objective of this master thesis is to show how mathematically describe macroscopic effects of heat transfer via conduction mode rather than microscopic activities of molecules associated with this mode. Different mathematical methods will be considered further in this work, such as numerical and analytical solutions.

2.1.1 Fourier’s Law of Heat Conduction

In order to describe the phenomena of conduction on the macroscopic level with the disregard to molecular structure of matter, the experiments were run and then the relationship between the heat flux and temperature gradient was derived. [4]

The experimental setup is shown in Figure 2. A slab of material is in contact with a heat source with temperature T1. The upper surface of the slab is in touch with cooling water system, which has the lower temperature T2. The contact area between hot plate and sink is A, and heat conducted through the slab towards the cooling water from the heat source perpendicular to the area. If the assumption is that the system is at steady state and temperature is measured at each point of the slab and then plotted, this slope can be described as dT/dx. The same experiment with different materials of the slab would show the same trend of the slope, thus empirical correlation between these parameters can be derived.

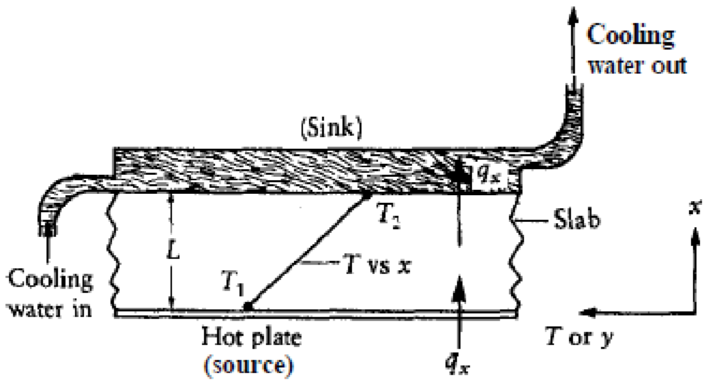


Figure 2: Experiment for observing Fourier’s Law of Heat Conduction [3]

The empirical Fourier’s law of conduction describes the relationship between the heat flow and the temperature field. This law has the following vector form:

$$q'' = -k\nabla T \tag{1}$$

where the temperature gradient is a vector normal to the isothermal surface and the heat flux vector q'' shows the heat flow per unit time and per unit area of the surface and minus represents the direction of flux, which is from the warmer to colder object. The proportional constant *k* is the thermal conductivity of the media in $[\frac{W}{m*K}]$. [5]

Thermal conductivity is a property of a material, whose number indicates how fast heat can be conducted through the material and includes all the molecular effects contributing to the conduction mode of heat transfer. For most substances, thermal conductivity varies with temperature.

2.1.2 The General Heat Conduction Equation

The general differential equation of conservation of energy can be applied to describe the physical process of heat conduction

$$\frac{\partial}{\partial t}(\rho c T) + \vec{\nabla} \cdot \vec{q} - Q = 0 \tag{2}$$

where T – temperature at any point and any time t, c – specific heat $[\frac{J}{K}]$ or $[\frac{kg \cdot m^2}{K \cdot s^2}]$, ρ – mass density $[\frac{kg}{m^3}]$, $\vec{q}$ – heat flux, Q – heat energy generated per unit volume per unit time. More can be done, if one applies Fourier's Law mentioned in the chapter 2.1.1 for the heat flux $\vec{q}$ in terms of the temperature. The equation of heat energy then becomes

$$\frac{\partial}{\partial t}(\rho c T) = \vec{\nabla} \cdot \left(k \vec{\nabla} T\right) + Q \tag{3}$$

In addition, this differential equation can be written in one, two or more dimensions. Most heat transfer problems encountered in practice can be written as one dimensional situations due to its simplicity, however, that is not always the case. The governing equation mentioned above can be developed in different coordinate systems. Figure 3 shows the three-dimensional heat conduction problem in the Cartesian coordinate system.

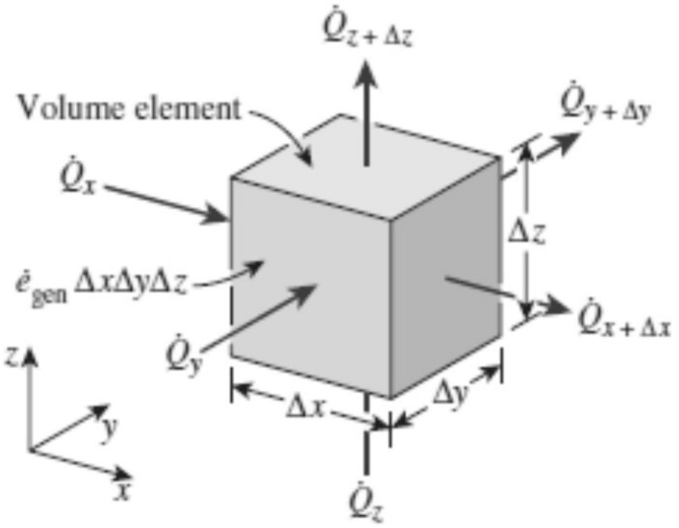


Figure 3: Three-dimensional heat conduction through a rectangular volume element [4]

The small rectangular element with density of the body ρ, specific heat c, length, width, and height is assumed and shown in Figure 3. A conservation of energy for a small time interval Δt then will be expressed as

$$\begin{pmatrix} Rate\ of\ heat \\ conduction \\ at\ x, y, z \end{pmatrix} - \begin{pmatrix} Rate\ of \\ heat\ conduction \\ at\ x+\Delta x, \\ y+\Delta y, \\ z+\Delta z \end{pmatrix} + \begin{pmatrix} Rate\ of\ heat \\ generation \\ in\ the \\ element \end{pmatrix} = \begin{pmatrix} Rate\ of\ change \\ of\ energy \\ in\ the\ element \end{pmatrix} \tag{4}$$

When assumption is made that thermal conductivity is constant in time, this equation will look in the differential form, as shown

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} + \frac{Q}{k} = \frac{1}{\alpha}\frac{\partial T}{\partial t} \tag{5}$$

where α is the thermal diffusivity of the material and can be found as $\alpha = \frac{k}{\rho c}$.

Note that in case of 1-dimensional heat conduction, the derivatives with respect to y and z drop out and the eq. (5) will be reduced to the equation for a plane wall. The same procedure can be applied for 2-dimensional problems.

2.2 Boundary and Initial Conditions

The heat conduction equations above were developed using an energy balance inside the medium, and they remain the same regardless of the thermal conditions on the surfaces of the medium. Thus, the differential equations do not bring any information related to the surface conditions of this media, such as the surface temperature or a specific heat flux. However, it is known that conditions on the surfaces influence the heat flux and the temperature distribution in a medium, and in order to complete the definition of a heat conduction problem the thermal conditions at the boundaries shall be set up, defined as the boundary conditions.

To describe a heat conduction problem completely, two boundary conditions for each direction shall be specified. Therefore, two boundary conditions are necessary for 1D problems, and four boundary conditions for 2D problems.

In addition, to solve a differential heat conduction equation and see temperature distribution along the medium the temperature at the beginning of the process shall be defined. Usually it is specified at time t=0 and called the initial condition. Nevertheless, only one initial condition is required regardless of the dimension to define a heat conduction problem.

In Cartesian coordinates, the initial condition can be written in its general form

$$T(x, y, z, 0) = f(x, y, z) \tag{6}$$

where the function $f(x, y, z)$ shows the temperature distribution in the media at the beginning. Under steady state conditions, the heat conduction equation does not include time derivatives, and consequently, no need to specify initial conditions. [5]

There are three general types of boundary conditions existing: Dirichlet, Neumann and Mixed conditions.

- Dirichlet Boundary Conditions. It sets up the temperature at the boundary. For one-dimensional case it will be in form of [6]:

$$T(x = 0, t) = T_{bc1}(t) \tag{7}$$

This means that at the left side boundary of the system the temperature is a specified function of time. If it is constant over time, then it will take form:

$$T(x = 0, t) = T_{bc1} \tag{8}$$

This, on the other hand, means that the system is touching the infinite heat source with constant temperature.

1D systems requires two boundary conditions specified. The right-hand-side boundary for the 1D problem with the system of the length L will look like:

$$T(x = L, t) = T_{bc2}(t) \tag{9}$$

- Neumann Boundary conditions. They say that at one of the boundary sides there is a heat flux instead of temperature. It will take the form like:

$$\frac{dT}{dx}(x = o, t) = \frac{dT(t)}{dx}\ at\ bc1 \tag{10}$$

If the heat flux is constant over time, the boundary condition will be:

$$\frac{dT}{dx}(x = o, t) = \frac{dT}{dx}\ at\ bc1 \tag{11}$$

In this particular case the system is touching an infinite heat source that maintains a constant flux of heat into the system regardless of the temperature.

The second boundary condition [5] for this case be of the form:

$$\frac{dT}{dx}(x = L, t) = \frac{dT}{dx}\ at\ bc2 \tag{12}$$

- Mixed Boundary conditions. It is a mixture of the two boundary conditions described above. [6] For 1D case it will take a form of:

$$\frac{dT}{dx}(x = o, t) + T(x = 0, t) = (T(t) + \frac{dT}{dx})\ at\ bc1 \tag{13}$$

2.3 Steady State and Transient Regimes

Another important factor to mention while investigating heat conduction is the situation one deals with. Generally, a heat conduction study may be divided into two groups, steady state models and transient models. A steady state model is simpler to solve, since the solution varies only along spatial coordinates, but not along the time axis as the time derivative is equal to zero. On the other hand, transient models are built to investigate the time dependency of heat conduction problems. [7]

2.4 Analytical Solutions

Analytical solution is generally the best approach to receive the outcome of a problem with stricter formulation, where all the parameters explicitly influence the results.

Analytical solutions to heat conduction problems are mainly solving the PDE eq. (2), namely the heat equation within a homogeneous solid, under appropriate initial and boundary conditions (this may include convective and radiative terms). The simplest analytical solutions refer just to the simple one-dimensional planar problem ignoring the dissipation and convection [5]:

$$\frac{\partial^2 T}{\partial x^2} - \frac{1}{\alpha}\frac{\partial T}{\partial t} = 0 \tag{14}$$

However, more practical analytical solutions refer to enriched two-dimensional PDE with heat sources and a possible coordinate-motion:

$$\frac{1}{r^n}\frac{\partial}{\partial r}\left(r^n \frac{\partial T}{\partial r}\right) + \frac{\partial^2 T}{\partial z^2} - \frac{\rho c \vartheta}{k}\frac{\partial T}{\partial z} + \frac{\emptyset(r,z,t)}{k} - \frac{1}{\alpha}\frac{\partial T}{\partial t} = 0 \tag{15}$$

where *n* describes the geometry (*n=1* for cylindrical, *n=2* for spherical) and ϑ is the velocity of solid material relative to a z-moving reference frame.

There are several different approaches to find analytical solution of PDE: dimension reduction, reduction by similarity, separation of variables, Green's function integral, Laplace transform and so on. [8] Even though it can be mathematically difficult to solve such PDEs, however, it is very important for general understanding of thermal problems, set-ups of correct boundary and initial conditions and to cross check using more practical numerical solutions.

2.5 Numerical Solutions

The one of the objectives of this thesis is to develop a numerical method for solving heat conduction problems at steady state and transient regimes. The basis of numerical methods is the idea of discretization. An analytical solution to a partial differential equation gives us the value of T as a function of the independent variables *(x, y, z, t)*. The result of the numerical solution, on the other hand, is a discrete number of T values in the domain. These points are called *grid points,* or nodes, or cell centroids. The process, where governing general equation is converted into an equation for each discrete number of T, is the discretization process and the methods applied in this process are called discretization methods [9].

The development of numerical methods focuses on both the derivation of the discrete set of algebraic equations, as well as a method of their solution. Another point is to make profile assumptions, so how T changes from node to node. Most common method for those assumptions are local neighborhood surrounding assumptions, but not over entire domain.

To convert one differential equation into a set of discrete algebraic equations requires the discretization of space, and that can be done by mesh generation. Mesh generation divides up the entire domain into the cells, and each of those cells would be bound to discrete value T, which are computed by algebraic equation [10].

The most popular and wide-spread numerical methods, from simplest to complex, may be divided into two major groups:

- Integral methods, when the integral energy equation $\frac{\partial E}{\partial t} = Q + W$ is solved, instead of differential heat equation.
- Differential methods, when the heat equation is solved, but not the integral energy equation. [11]

2.5.1 Finite Volume Method

The computational procedure in this master thesis was chosen to be Finite Volume Method (FVM). In order to understand this method, first, the differential form of the general transport equation is observed. A general variable ∅ is introduced for all fluid flow equations. The equation will take then form:

$$\frac{\partial(\rho\emptyset)}{\partial t} + div(\rho\emptyset u) = div(\Gamma\ grad\emptyset) + S_{\emptyset} \tag{16}$$

which can also be written as:

$$\begin{pmatrix} Rate\ of\ increase \\ of\ \emptyset\ of\ fluid \\ element \end{pmatrix} + \begin{pmatrix} Net\ rate\ of\ flow \\ \emptyset\ out\ of \\ fluid \\ element \end{pmatrix} = \begin{pmatrix} Rate\ of\ increase \\ of\ \emptyset\ due\ to \\ diffusion \end{pmatrix} + \begin{pmatrix} Rate\ of\ increase \\ of\ \emptyset\ due\ to \\ sources \end{pmatrix} \tag{17}$$

The equation (16) can be also called transport equation of property ∅. It describes different transport processes in the system: on the right side of the equation are diffusive($\Gamma = diffusion\ term$) and source (S) terms, on the left – temporal and convective terms.

So the equation (16) is the beginning point for computational procedures in the FVM. The next step is to integrate this equation over a three-dimensional control volume CV

$$\int_0^{CV} \frac{\partial(\rho\emptyset)}{\partial t} dV + \int_0^{CV} div(\rho\emptyset u)dV = \int_0^{CV} div(\Gamma\ grad\emptyset)dV + \int_0^{CV} S_\emptyset dV \tag{18}$$

Applying the Gauss' divergence theorem and integrating over a small time interval Δt with respect to time t the most general integral form can be written: [12]

$$\int_t^{t+\Delta t} \left(\frac{\partial}{\partial t}\int_0^{V_m(t)} \rho\chi dV\right) dt + \int_t^{t+\Delta t} \left[\oint_0^{\partial V_m(t)} (\rho u\chi) n dS\right] dt = \int_t^{t+\Delta t} \left[\oint_0^{\partial V_m(t)} (\rho\Gamma_\chi \nabla\chi) n dS\right] dt + \int_t^{t+\Delta t} \left[\int_0^{V_m(t)} (S_\chi(\chi)) dV\right] dt \tag{19}$$

2.5.2 Mesh Terminology and Types

The domain of interest is divided into the cells by meshing. The certain terminology exists describing the mesh as it is shown in the Figure 4.

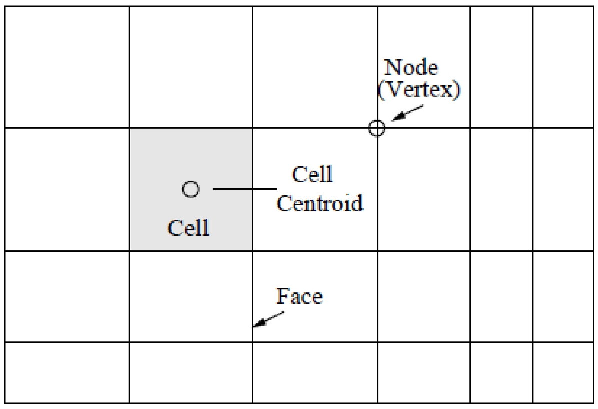


Figure 4: Mesh Terminology [13]

The fundamental unit of the mesh is the cell (sometimes called the element). Associated with each cell is the cell centroid. A cell is surrounded by faces, which meet at nodes or vertices. In three dimensions, the face is a surface surrounded by edges. In two dimensions, faces and edges are the same. A variety of mesh types are utilized in practice.

The fundamental unit of the mesh is the cell (sometimes called the element). Associated with each cell is the cell centroid. A cell is surrounded by faces, which meet at nodes or vertices. In three dimensions, the face is a surface surrounded by edges. In two dimensions, faces and edges are the same. A variety of mesh types are utilized in practice. [13]

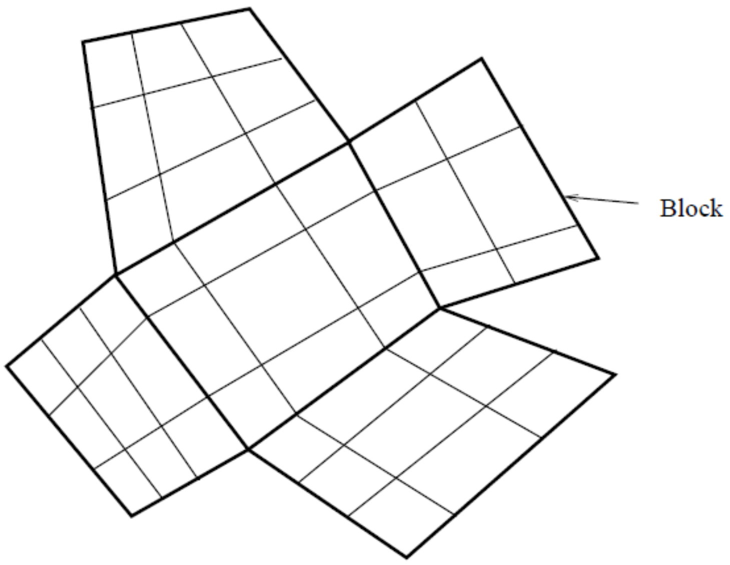


Figure 5: Block-structured Mesh [13]

In this master thesis the block-structured mesh will be used, which is showed in Figure 7. Here, the mesh is divided into blocks, and the mesh within each block is structured. This will be done using the mesh generation utility *blockMesh* supplied with OpenFOAM. The principle behind *blockMesh* is to decompose the domain geometry into a set of 1, 2 or 3 dimensional, hexahedral blocks. Edges of the blocks can be straight lines, arcs or splines. The mesh is defined as a number of cells in each direction of the block. [14]

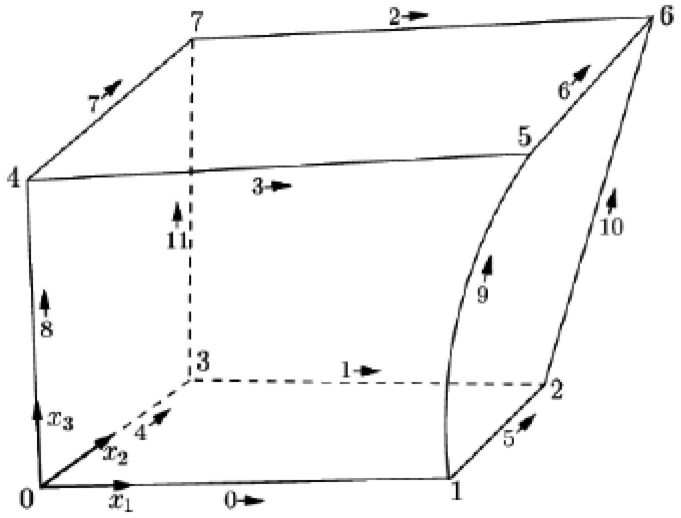


Figure 6: Diagram of a block [14]

2.5.3 Accuracy, Consistency, Stability and Convergence

It is also important to mention the certain properties of all numerical methods.

Convergence describes how the numerical solution approaches the exact solution as the grid spacing, control volume size or element size is reduced to zero.

Theoretically it can be difficult to achieve, and so Lax's equivalence theorem is used. This theorem states that if the method is both consistent and stable, the numerical method is convergent. However, this applies for linear problems only.

If the numerical solution includes a set of algebraic equations, which are equivalent to the original governing equation as the grid spacing tends to be zero, then numerical method is *consistent.* [12]

Stability, on the other hand, is a property of the path to solution. To solve a problem, steady state for example, initially a set of algebraic equations is obtained. Then the method to solve these equations is chosen between iterative or direct. Depending on the properties of the method, solution errors may increase or decrease. An iterative solution method is unstable if it doesn't show a solution as a discrete set.

Accuracy compares a numerical solution to an exact solution of equation. However, in most cases, the exact solution is unknown, and then truncation error of a discretization method is introduced. "The truncation error of a discretization scheme is the largest truncation error of each of the individual terms in the equation being discretized." [13] The order of discretization is described by *n*. Nevertheless, the truncation error does not show how high is the error of each certain mesh, but it presents a decrease of the error with mesh refinement. This means, that the methods of high orders still may show inaccurate results in the mesh. [13]

2.5.4 OpenFOAM

Computational fluid dynamics software OpenFOAM (Open Source Field Operation and Manipulation) is a C++ library developed by OpenCFD.ltd. The working process in OpenFOAM is divided into three parts as shown in Figure 7. [14]

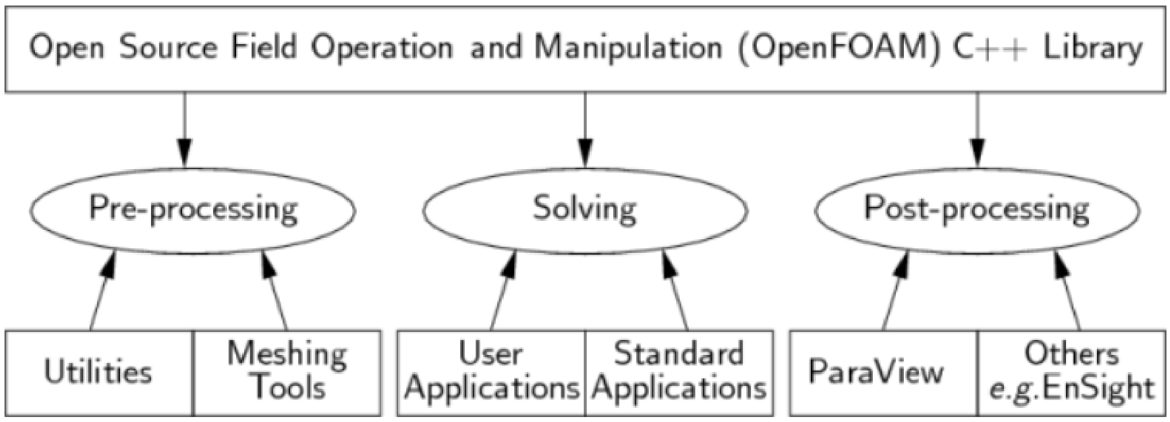


Figure 7: Overview of OpenFOAM environment [14]

The OpenFOAM dictionary files are used to define a specified case. Those files describe all the necessary, physical and numerical, conditions to solve the problem [14]. The file structure is explained in Figure 8. Before the running the case, the file includes three basic directories. In the time directory with the folder notation *0*, the boundary and initial conditions are specified. In the *constant* folder the physical properties shall be specified. The mesh properties, such as boundaries, points, cells, faces are saved in the folder *polyMesh*. The *system* folder describes the solution procedure with the three files in it:

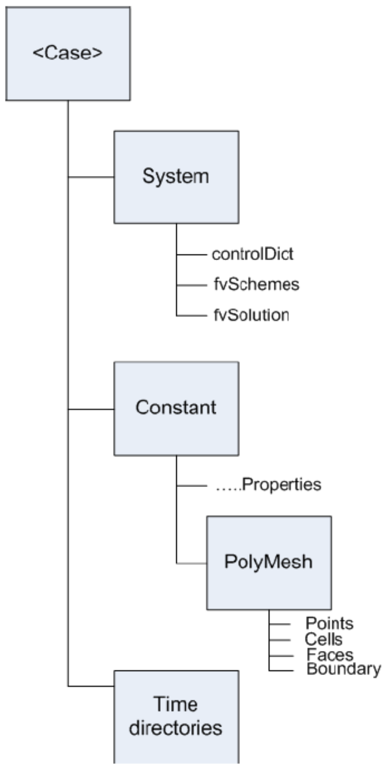


Figure 8: The file structure of a case in the OpenFOAM [14]

fvSchemes describes the discretization method, which is chosen for each particular case.

fvSolution includes the solver and equations, which are to be solved. As well the tolerances and relaxation factors can be stated here.

controlDict contains the time set-up. All the parameters, which will be changing over time, will be stated here. Here is the description of either it is steady state or transient regime being observed. [14]

Later, the step-by-step description of setting a case in OpenFOAM will be given.

2.5.4.1 Numerical Schemes

The *fvSchemes* dictionary (shown in Figure 8) sets up the numerical schemes for derivatives appearing in the equation to be run. OpenFOAM offers not only linear interpolation, even though it is most effective in many cases, but also other interpolation schemes for all the terms in an application. [14]

The first choice of discretization is standard Gaussian finite volume integration which is based on summing values of cell faces. Those values, in turn, are interpolated from cell centers. Then there is once again a choice of interpolation scheme, with special schemes designed for particular derivatives.

The set of terms with specified numerical schemes is subdivided into the categories listed in Table 1. Further explanation of setting up a case in OpenFOAM will be given in chapter 3.1.4.5.

Table 1: Main keywords in *fvSchemes*

Keyword	**Category of mathematical terms**
interpolationSchemes	Point-to-point interpolations
snGradSchemes	Component of gradient normal to a cell face
gradSchemes	Gradient
divSchemes	Divergence
laplacianSchemes	Laplacian
timeSchemes	First and second time derivatives

2.5.4.2 Solution and Algorithm Control

The equation solvers, tolerances and algorithms are specified in the *fvSolution* directory in the system folder of any OpenFOAM case. This directory includes a code describing a set of subdirectories, such as: *solvers*, *relaxationFactors*, *PISO*, *SIMPLE.*

The *solvers* is the first subdirectory to be specified. The code starts with a keyword of each variable being solved in the particular equation. In this master thesis *laplacianFoam* application is used, thus, *T* is the only entry in this file directory. The solver shall be selected through the *solver* keyword from the options presented in Table 2.

Table 2: Types of solvers in OpenFOAM directory *fvSolution*

Solver code	**Type of solver**
PCG/PBiCGStab	Stabilized preconditioned (bi-)conjugate gradient, for both symmetric and asymmetric matrices.
PCG/PBiCG	Preconditioned (bi-)conjugate gradient, with PCG for symmetric matrices, PBiCG for asymmetric matrices
smoothSolver	solver that uses a smoother

GAMG	generalized geometric-algebraic multi-grid
diagonal	diagonal solver for explicit systems

Generalized geometric algebraic multi-grid is used in the cases observed in this master thesis. The principle behind this type of solver is that it generates a quick solution on the mesh with a small number of cells; maps this solution onto a finer mesh; uses it as initial guess to obtain an accurate solution on the fine mesh. This method is faster than standard methods, especially in solving the Laplace equation.

3 NUMERICAL FORMULATION

In this chapter, the three different cases of heat conduction, both in steady state and transient conditions, are observed. Firstly, the description of a case is given, followed by the analytical solution of such a case. As a second part of the problem analysis, the detailed set up of the case is presented in order to show how to receive the numerical solution of a problem in OpenFOAM and further post-processing. To sum up, the comparison between the analytical and numerical solutions are provided with the discussion of possible reasons of deviations in results.

3.1 Steady State Conduction in 2D

The first example of solving heat transfer equation will be steady-state heat conduction in 2D Slab, shown in Figure 9.

3.1.1 Description of the Case

A uniform square section with length and height of 1 meter and width of 0.1 meter is observed. This slab has constant temperature at the side 4 ($T = T_1$) of 1, the sides 1 and 2 are assumed to be adiabatic. The condition at the side 3 can be described as $\frac{\partial T}{\partial x} + hT = 0$, where $h = \frac{h_f}{k}$ with h_f as heat transfer coefficient over *x=1.*

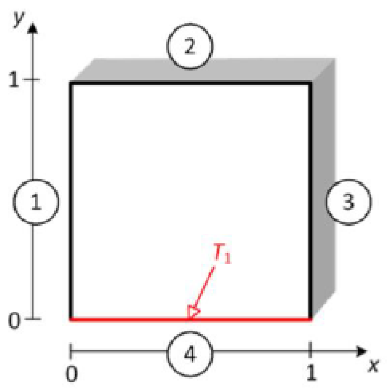


Figure 9: Steady State problem in 2D slab [15]

3.1.2 Governing Equation

For steady state conduction the general form of heat transfer equation can be written as:

$$\nabla.(\alpha \nabla T) = 0 \tag{20}$$

In case of steady state conduction in a slab, the equation (5) will take a form of the Laplace equation. This equation applies with assumptions, that there is no heat generated and the thermal conductivity remains constant [16]

$$\frac{\partial^2 \emptyset}{\partial x^2} = 0 \qquad (21)$$

Boundary conditions are described in the chapter above.

3.1.3 Analytical Solution

The analytical solution of eq. (3) for the given case was solved in [16] as:

$$T = 2h \sum_{i=1}^{\infty} \frac{\cos(p_i x) cosh[p_i(1-y)]}{[(p_i^2+h^2)+h]\cos(p_i)\cosh(p_i)}, \qquad (22)$$

where “the p_i are the positive roots of the transcendental equation” [17]:

$$f(p) = p\ tanp - h = 0 \qquad (23)$$

If to take h=10, the analytical solution will take a form as presented in Figure 10. The comparison of analytical solutions and numerical solutions, which are given in the source [16], are given later.

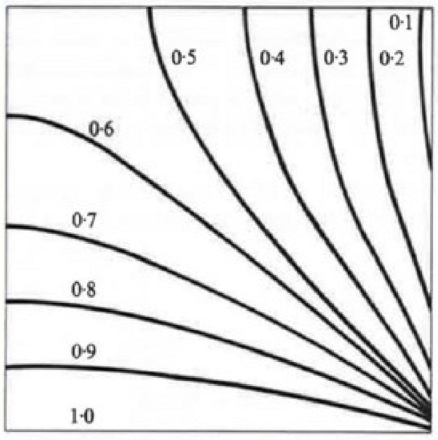


Figure 10: Analytical solution for the 2D slab. Temperature distribution [17]

3.1.4 Preprocessing

As it was mentioned before in chapter 2.5.4, all the data required to simulate a case in OpenFOAM is stored in a case directory.

The first step in the simulation process is to create the case directory, named *Slab* for example. *Hint: there must not be any spaces in the file directory names.*

The constant directory contains physical properties of the object and a *polyMesh* folder where a full description of the mesh will be stored. In the *system* directory are the files with setting parameters associated with the solution procedure. The *0* directory contains the initial and boundary conditions set up. After running the case the time-directories will appear containing the solutions for each time step.

To set up the files inside the directories, one can use a text editor, for example *NotePad*, and write the files from the scratch. However, it is recommended for this case to copy similar case from tutorials folder and modify it according to the particular simulation parameters. For this problem a case *tutorials/basic/laplacianFoam/flange* can be copied and modified.

3.1.4.1 Mesh Generation

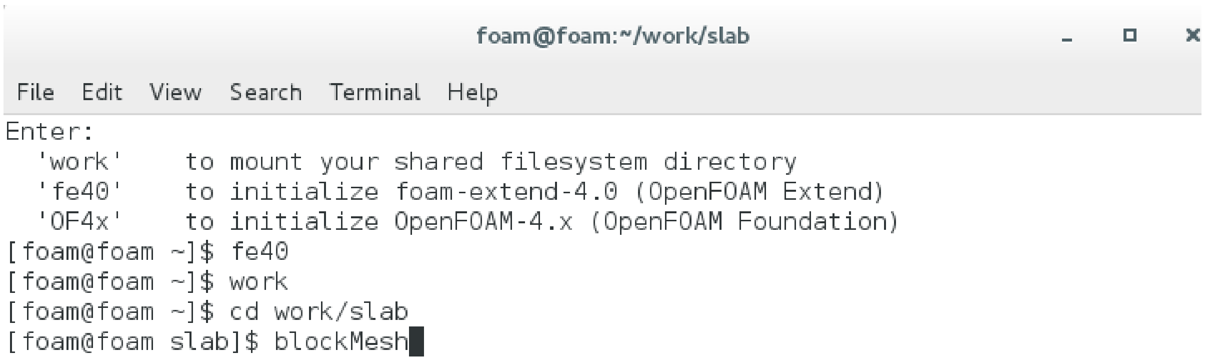
OpenFOAM always uses 3D meshes and solves the case in 3 dimensions by default. To solve 2D problem, 3D mesh still must be generated with two dimensions matching the geometry of the Slab and the third one is arbitrary. So the heat will transfer only through x- and y-axis. To create a mesh, the file in system directory named *blockMeshDict* should be created.

The entries, which are shown in Appendix 1, mean:

- ***convertToMeters***. The default unit for the coordinates are meters. If other coordinates are required, for example centimeters, 1 shall be changed to 0.01.
- ***vertices*.** Here the vertices of the mesh are defined, starting from 0. There are 8 vertices in this case making up together a hexadron block.
- ***blocks*.** This part has several entries, which define the mesh block and its division. The order of the vertices, which define a single block, must be written by the rules of right-handed coordinate system. The first vertex defines the origin of the local coordinate system. In the second entry of blocks the number of cells in each direction is specified. The third entry defines cell expansion ratio along a direction or an edge, so it is a ratio between the first and the last cell. There are different codes possible, like *simpleGrading* or *edgeGrading.* However, in this example only *simpleGrading* is used, which specifies expansion ratios in x, y and z directions.
- ***edges*.** This entry is used to describe the edges joining to vertex points. If nothing is specified, straight lines are assumed as default.
- ***boundary**.* Here the boundary of the mesh is broken into patches. These are the regions where boundary conditions will be applied. After each name of the patch, the boundary type is specified. The *empty* type instructs OpenFOAM to solve the problem in other two dimensions. The generic type *patch* does not state any special information. Face entry specifies one of several vectors, which contains vertices of the faces assigned to the corresponding patch. [14]

Once the *blockMeshDict* is set up, the terminal of OpenFOAM should be open. Then the case directory shall be browsed with the command *cd work/Slab/* and then command *blockMesh* can be entered, as shown in Figure 11.

This command will create the file *blockMesh* in constant directory. Also the result can be observed in the Paraview software, which is used for post-processing. Later, the detailed description how to use this software will be given, in chapter 3.1.6.



```
Enter:
  'work'    to mount your shared filesystem directory
  'fe40'    to initialize foam-extend-4.0 (OpenFOAM Extend)
  'OF4x'    to initialize OpenFOAM-4.x (OpenFOAM Foundation)
[foam@foam ~]$ fe40
[foam@foam ~]$ work
[foam@foam ~]$ cd work/slab
[foam@foam slab]$ blockMesh
```

Figure 11: Terminal in OpenFOAM

3.1.4.2 Boundary and Initial Conditions Set Up

The next step of the simulation is to introduce boundary and initial conditions of temperature field. For each field of interest there is a corresponding file inside the *0* folder.

There are three main entries in this file:

- ***dimensions.*** This entry specifies the dimensions of all the variables. Each position is assigned to a specific unit.

  Table 3: Properties assigned to their position in the entry

Position	Property
1	Mass [kg]
2	Length [m]
3	Time [s]
4	Temperature [K]
5	Quantity of substance [mol]
6	Current [A]
7	Luminous intensity [cd]

- ***internalField****.* This entry defines the initial internal field. *uniform* sets the specified value (0) to all the internal elements. For steady-state cases this doesn't affect the final solution, but may have an impact on the stability and resolution speed.

- ***boundaryField.*** Introduces the boundary conditions at each of the patches, which were defined in *blockMeshDict.*
    - *frontAndBack. empty* is used in case of 1D and 2D cases. Since the mesh OpenFOAM produces is always 3D, this entry type means that the z-axis is chosen as arbitrary.
    - *bottom. fixedValue* applies a single value (273) to the boundary.
    - *top* and *left. zeroGradient* sets the normal gradient ($\pm\frac{\partial T}{\partial y}$) to 0, establishing the adiabatic condition at sides 1 and 2.
    - *right.* Type *groovyBC* is introduced in the library *swak4Foam* [18]. It allows to set non-uniform boundary conditions without programming. *gradientExpression* strings with the gradient to be used if a Neumann condition is needed. Defaults to zero. *fractiontExpression* says if the face is at Dirichlet (1) or Neumann (0) condition.

The temperature field file is shown in Appendix 2.

3.1.4.3 Physical Properties

The only physical property OpenFOAM needs for this case is the thermal diffusivity of the material (the parameter α from the heat equation), which is defined within *transportProperties* and named DT. The unit, which could be depicted in this code as well, is $\left[\frac{m^2}{s}\right]$. The code is the following:

```
DT          DT [ 0 2 -1 0 0 0 0 ] 0.1;
```

3.1.4.4 Time Settings

The parameters, which control the time of simulation and the input/output of the data are written in the *controlDict* file. In this case the time controls relate to the number of iterations. The used code is shown in Appendix 3

Some of the important entries here:

- ***startFrom.*** Controls the start of the simulation. The keyword *startTime* instructs the solver to begin the simulation always from scratch. Also there is the possibility to enter *latestTime* keyword, which would begin the simulation from the latest stored time-folder.
- ***stopAt***. Controls the end-point of the simulation. *endTime* entry here says to stop after 30 sec of simulation.
- ***deltaT***. Defines the time-step. In steady-state problems every step is an iteration.
- ***writeControl.*** Controls the timing of the write output file.
- ***purgeWrite.*** Specifies a limit on the number of output time-directories stored. 0 is for no limitations.

- ***libs.*** Allows OpenFOAM to use a specific library mentioned. This library was introduced in chapter 3.1.4.2 to describe special boundary conditions.

3.1.4.5 Discretization and Linear Solver Set Up

The finite volume, as it was mentioned in chapter 2.5.1, was chosen to be discretization method. This method will be specified in the *fvSchemes* file. The *fvSolution* file specifies the linear equation solvers and tolerances and other algorithm controls.

The code in the file *fvSchemes* is shown in Appendix 4

In this file it is important to change the ***ddtSchemes*** entry to *Euler*. This will mean, that the result will represent a stable solution achieved after some transient behavior.

The code in the file *fvSolution* is shown in Appendix 5

3.1.5 Running the Simulation

Having prepared the case files, the solver can be run by typing in the case directory:

laplacianFoam

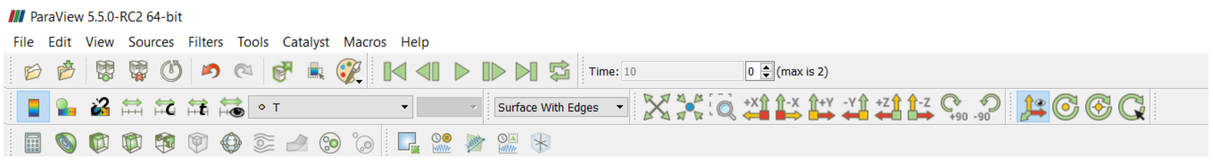
This will generate files in the time directory for each time-step.

In order to observe the results in visualization software, one more step should be done in the same case directory:*touch slab.foam*

This will generate a blank file in the case directory with the file extension *.foam*.

3.1.6 Post-processing

Having the case solved in OpenFOAM, it is a good idea to visualize results in Paraview software. This software is an open-source, multi-platform data analysis and visualization application. The file with the *.foam* extension can be opened in Paraview and then the button *apply* can be clicked. Now Paraview is ready to display the results. In Figure 12 are shown some of the basic controls, which should be changed: *T*-field to be colored and *Surface With Edges.*



Time: 10
0 (max is 2)
T
Surface With Edges

Figure 12: Some basic post-processing controls

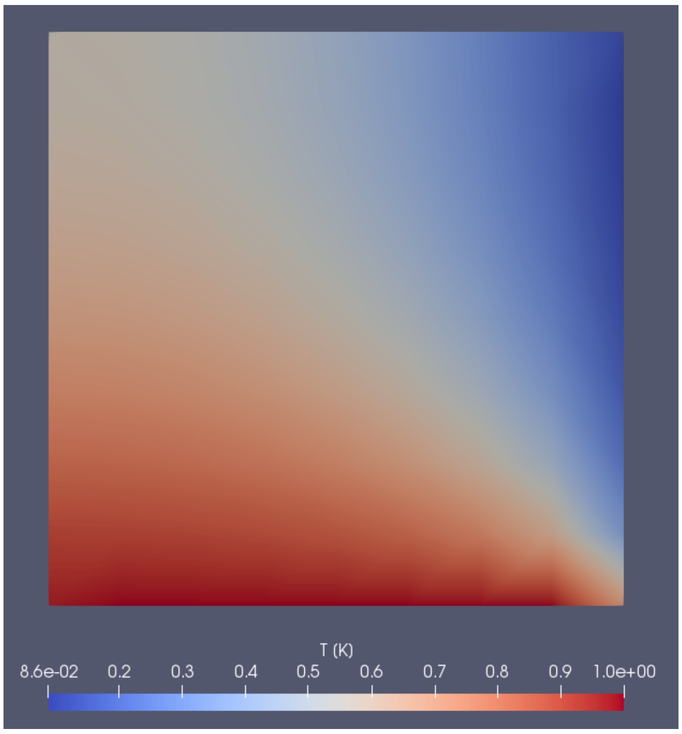


Figure 13: Temperature distribution in the Slab

Figure 13 shows the result of the solution. However, in order to compare the analytical solution results, given in [17], with results of OpenFOAM solution, the data should be interpolated. To do so, the solution as cell values are presented by choosing in coloring control button *T*-field in cells. Figure 14 presents the cell values of the solution calculated with the FVM.

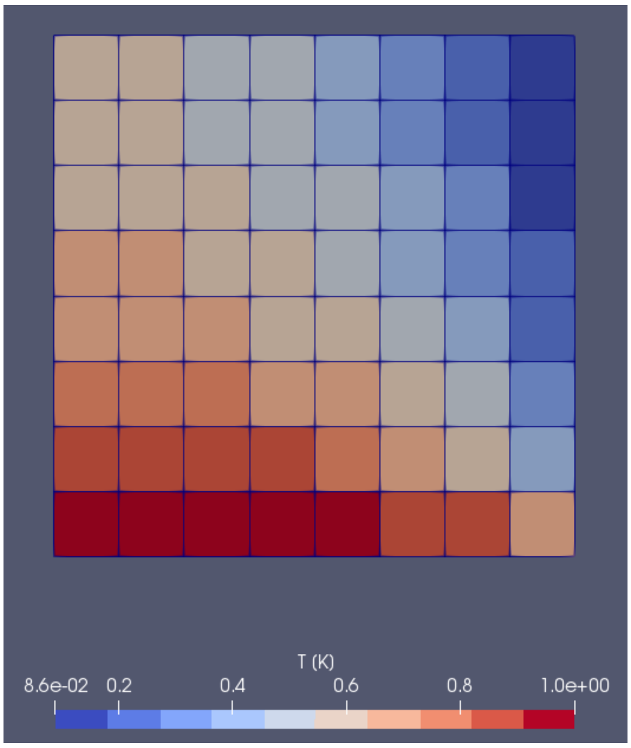


Figure 14: The cells values calculated by the FVM

Then the uppermost line in Paraview the *Filters* tool is chosen, *Point interpolation → Point Volume Interpolator.* In order to produce isolines corresponding with the analytical solution, the 10 color sets are chosen in *Edit color map →Number of Table Values.* Figure 15 presents the final result, which now can be compared to the analytical solution.

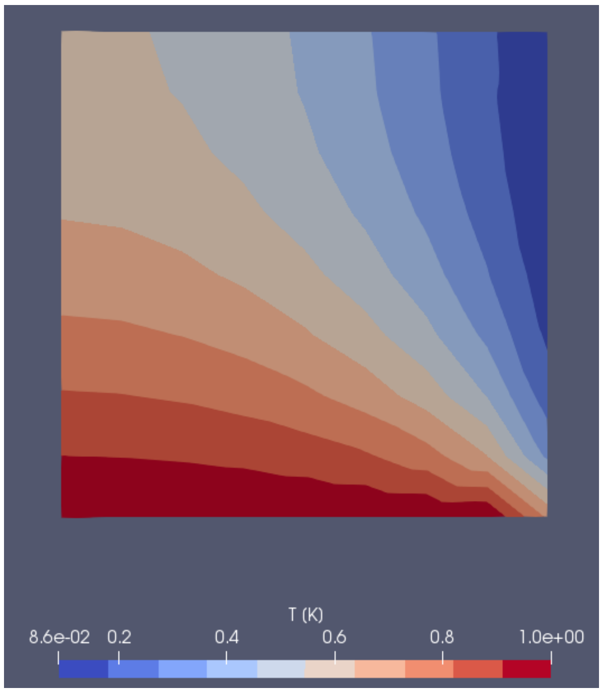


Figure 15: Solution with interpolated point values with 10 color sets

3.1.7 Result and Discussion

The result shows a relatively low percentage error between the analytical and the numerical solution, although the chosen mesh is coarse. However, an error of roughly 2% is observed near the sides 3 and 4. The rapid temperature change occurs near these regions, thus, the discretization of the problem is rather crude. Bayley, et al.[17] mentions that the convergence of analytical solution, similar to the numerical solution, is also slow in this region.

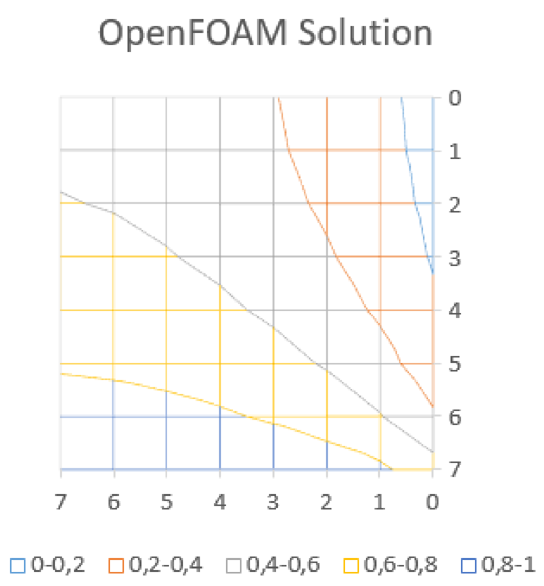


Figure 16: Temperature distribution in the Slab calculated in OpenFOAM

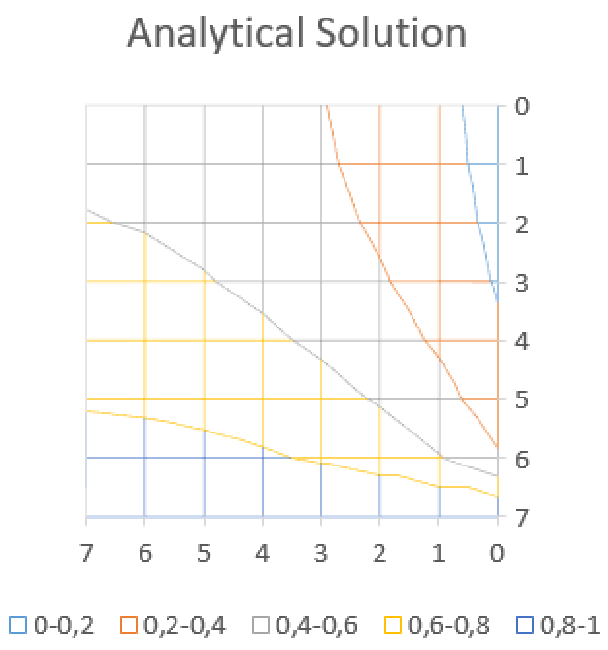


Figure 17: Temperature distribution in the Slab calculated analytically

Figure 16 and Figure 17 present the OpenFOAM and analytical solutions, where the error of 2% for the sides 3 and 4 can be depicted.

3.2 Critical Radius of Insulation

The second example will describe a possible effect in case of non-planar geometries when adding an insulating layer over a hot thin wire or hot sphere, cooled to an ambient fluid, which can be described by critical radius. This effect means that the increase in heat transfer area by the additional layer overcomes the effect of a small thermal conductivity of the cover, as shown in Figure 19.

3.2.1 Description of the Case

A cylinder with the radius of r_1=15 mm has an insulation layer with thermal conductivity of $k = 0{,}155\ \frac{W}{m*K}$ of the radius r_2=30 mm [19]. At the surface of the cylinder the temperature is T_1=*100 °C*. The heat flows from the hot cylinder through the insulation to the ambient medium, consequently the steady state regime of heat conduction was assumed and all the heat is uniformly distributed. All air gaps were neglected. The condition at the surface of insulation is assumed as natural convection. Radiation and forced convection were not observed in this case, since it would reduce significantly the critical radius. [20]

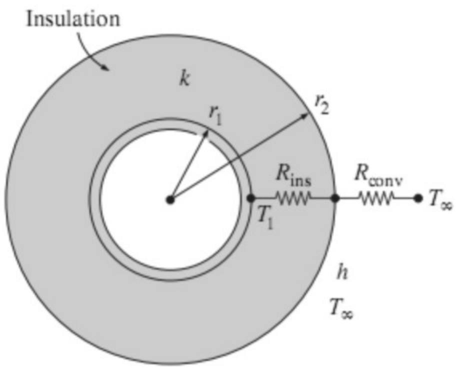


Figure 18: Critical radius of insulation material, 1D Steady state conduction problem [2]

The temperature of the ambient medium is T_2=*29 °C,* convective heat transfer coefficient *h=10* [$\frac{W}{m^2*K}$] [21].

3.2.2 Analytical Solution

As an illustration, the heat flow from a hot wire of radius r_1 with a fixed T_R, exposed to an ambient fluid of T_∞ and convective coefficient of h=*const,* through an insulating layer of conductivity k between r_1 and r_2 is calculated as:

$$Q(r_2 \geq r_1) = k2\pi L\frac{T_{r_1} - T(r)}{\ln\frac{r_2}{r_1}} = h2\pi r_2 L(T(r) - T_\infty) = 2\pi L\frac{T_{r_1} - T_\infty}{\frac{1}{k}\ln\frac{r_2}{r_1} + \frac{1}{r_2 h}} \rightarrow \tag{24}$$

$$\frac{\partial Q}{\partial r} = 0,\ \frac{1}{rk} - \frac{1}{r^2 h} = 0 \rightarrow r|_{Q_{min}} = \frac{k}{h}$$

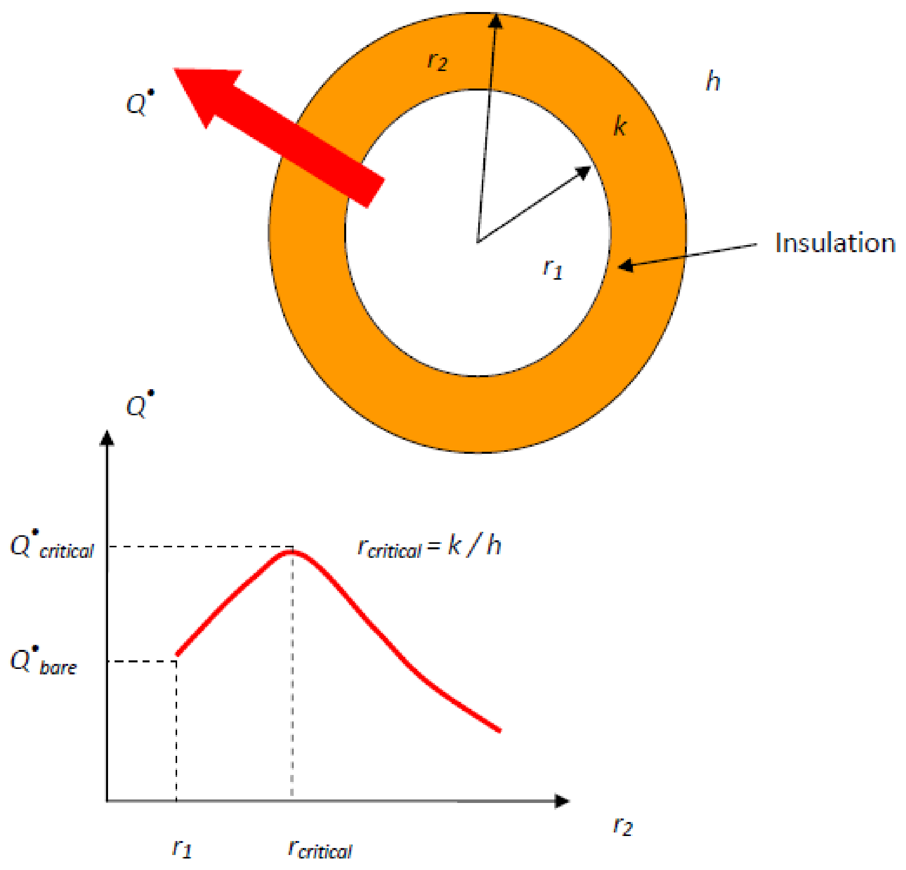


Figure 19: Critical radius of insulation for non-planar geometry [22]

For the spherical geometry the critical radius will be $r|_{Q_{min}} = \frac{2k}{h}$. However, this critical radius matter only in case of small dimensions of cylinders or spheres, for instance, for wires and pipes with diameter smaller than 20 mm [22]. Based on the parameters described in the case description, the $r|_{Q_{min}} = 0{,}0155\ mm$.

3.2.3 Preprocessing

Same as with the example above (Chapter 3.1), all the data required to simulate a case in OpenFOAM is stored in a case directory.

The first step in the simulation process can be to copy the case directory, named *Radius* for example. The *constant* directory contains physical properties of the object and *polyMesh* folder where a full description of the mesh will be stored. In the *system* directory are the files with setting parameters associated with the solution procedure. The *0* directory contains the initial and boundary conditions set up. After running the case, the time-directories will appear containing the solutions for each time step.

3.2.3.1 Mesh Generation

This case will be assumed as 1D steady state heat conduction because of the same temperature distribution in x and y directions due to cylinder symmetry. The z-axis is taken as arbitrary, such as in the case *Slab.*

To create a mesh, the file in system directory named *blockMeshDict* should be created.

The codes, which are entered in the file, mean the same as described in chapter 3.1.4.1. However, since it is a cylinder, the special entry should be defined as:

- ***edges.*** This entry is used to describe the edges joining to vertex points. If nothing is specified, straight lines are assumed as default. In the case of a cylinder, the edges have to be specified. Each edge must be described with at least one interpolation point, which the circular arc will intersect. For hollow cylinder example, 16 arcs are assigned.[14]

The *blockMeshDict* code for this case is shown in Appendix 6, Appendix 7 and Appendix 8.

Once the *blockMeshDict* is set up, the terminal of OpenFOAM should be opened. Then the case directory shall be browsed with the command *cd work/Radius/* and then command *blockMesh* can be run.

This command will create the file *blockMesh* in the *constant* directory. Also the result can be observed in Paraview software, which is used for post-processing.

3.2.3.2 Boundary and Initial Conditions Set Up

The next step of the simulation is to introduce boundary and initial conditions of temperature field. For each field of interest there is a corresponding file inside with *0* folder.

The first two entries (*dimensions* and *internalField*) in this file shall be the same as in the case *Slab.*

However, the third entry *boundaryField* shall be changed.

- ***boundaryField.*** Introduces the boundary conditions at each of the patches, which were defined in *blockMeshDict.*
  - *in. fixedValue* is used to specify the hot temperature at the surface of the cylinder. This temperature was assumed to be constant over time.
  - *out. groovyBC* type of boundary is used in order to describe the natural convection at the surface of insulation
  - *Left and right. zeroGradient* sets the normal gradient ($\pm\frac{\partial T}{\partial y}$) to 0, establishing the adiabatic condition at the outermost sides of cylinder.

The temperature field file is shown in Appendix 9.

3.2.3.3 Physical Properties

The only physical property OpenFOAM needs for this case is the thermal diffusivity of the material (the parameter α from the heat equation), which is defined within *transportProperties* and named DT. For the case of critical radius of insulation, the thermal diffusivity of insulation layer is specified in this file. The unit, which could be depicted in this code as well, is $\left[\frac{m^2}{s}\right]$. The code is the following:

```
DT          DT [ 0 2 -1 0 0 0 0 ] 0.00015;
```

3.2.3.4 Time Settings

The parameters, which control the time of simulation and the input/output of the data are written in the *controlDict* file. In this case the time controls relate to the number of iterations. Here all the entries have the same role as in the case with slab. The used code is shown in Appendix 10.

3.2.3.5 Discretization and Linear Solver Set Up

The finite volume, as it was done also for the case *Slab*, was chosen to be discretization schemes. This method will be specified in the *fvSchemes* file. The *fvSolution* file specifies the linear equation solvers and tolerances and other algorithm controls.

The code is not shown, since it looks the same as in the case before and the file can be simply copied.

In the file *fvSchemes* it is important to change the ***ddtSchemes*** entry to *steadyState* This will present the stable solution over all run time.

The code in the file *fvSolution* is the same as in chapter 3.1.

3.2.4 Running the Simulation

Having prepared the case files, the solver can be run by typing in the case directory: *laplacianFoam*

In order to observe the results in visualization software, one more step should be done in the same case directory: *touch radius.foam*

3.2.5 Post-processing

The file with *.foam* extension can be opened in Paraview and then the button *apply* can be clicked.

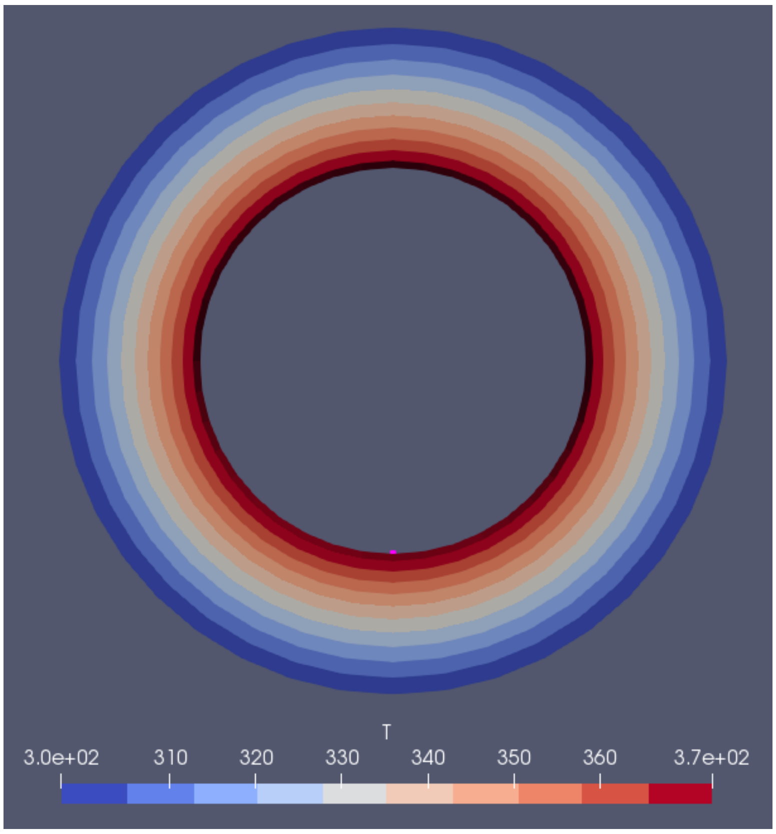


Figure 20: Temperature distribution in the insulation layer of the cylinder

The settings of Paraview may not allow to view the temperature distribution in full color. In order to obtain the colorful picture, as in Figure 20, the left mouse button should be clicked on the mesh, and then the option *Edit color* is chosen. On the right side of *Color map editor* the button *Rescale to visible range* can be selected.

3.2.6 Result and Discussion

Plotting the heat transfer rate *Q [W]* as the function of the insulating material *r [m]* gives the graph (Figure 21). The numerical solution gives the same value for the critical radius of insulation of the cylinder, however, there is a slight error in the heat energy between OpenFOAM and analytical results due to the temperature differences.

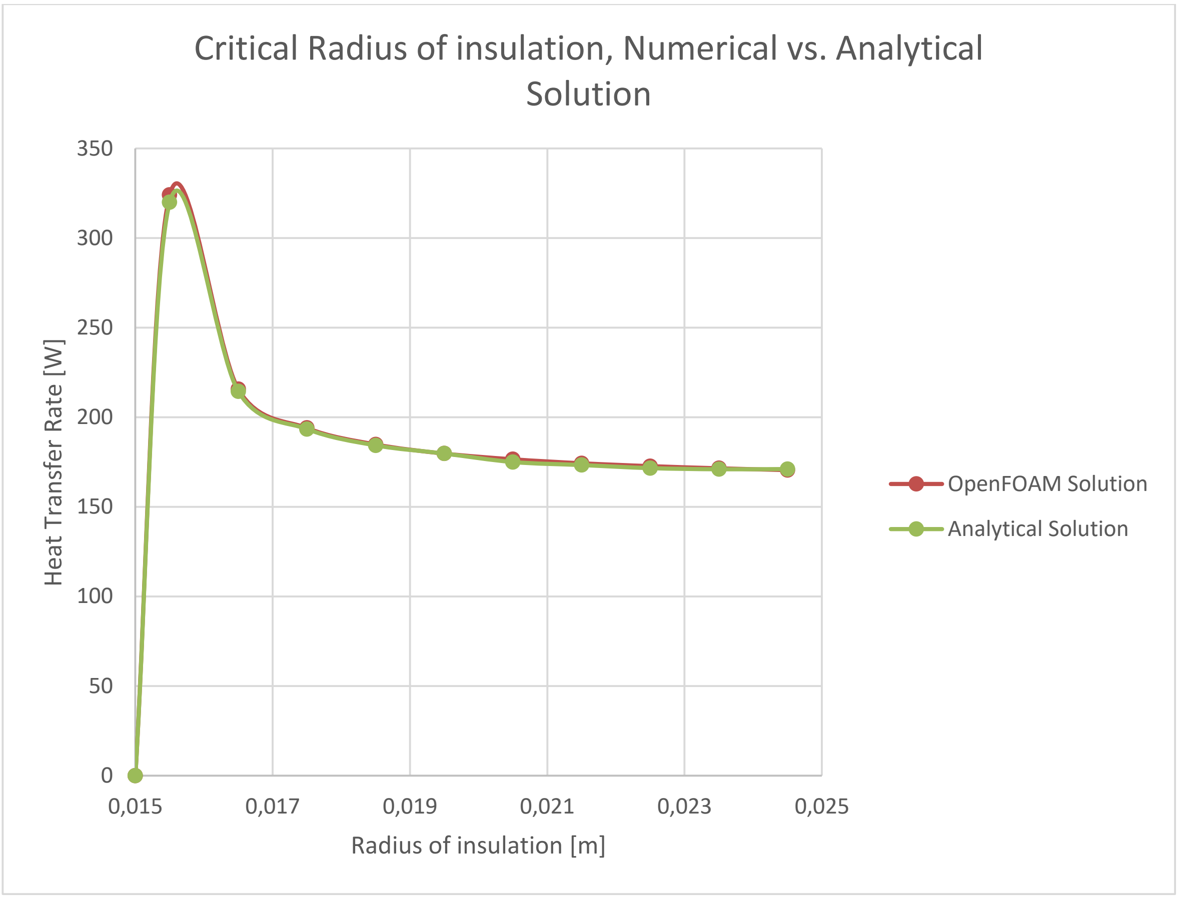


Figure 21: Comparison of OpenFOAM and analytical results

The error in the temperature distribution may be decreased by certain grid refinement techniques. On the one hand, a finer grid would give the very precise solution with further accurate heat energy calculations. Nevertheless, the objective of this problem was rather to study the radius of insulation than to be precise in the calculation, which due to a refined mesh would increase the computation time. The coarse grid was still relevant to be applied in this study.

3.3 Transient Heat Conduction in 1D

Heat conduction problems involving time mostly lead to parabolic PDEs. In this chapter, the 1D transient heat conduction problem will be observed with further comparison of analytical and numerical solutions.

3.3.1 Description of the Case

An infinite steel plate of infinite size in y-axis with thermal diffusivity $\alpha = 0{,}00003\, {m^2}/{s}$ and thermal conductivity $k = 36\frac{W}{m\,°K}$ [23] of thickness L=5 cm, initially at an uniform temperature of $T = T_i = 1\ K$, is suddenly immersed in an fluid bath at $T_\infty = 0\ K$. Convection heat transfer coefficient between the fluid and the surfaces is $h = 36\frac{W}{m^2\,°K}$.This problem is also called in the literature as “quenching” [17]. Since the initial temperature is uniform, for the analytical solution only one half of the plate will be considered, and the coordinate origin will be the center-line. The ratio of h/k for the simplification is taken as 1.

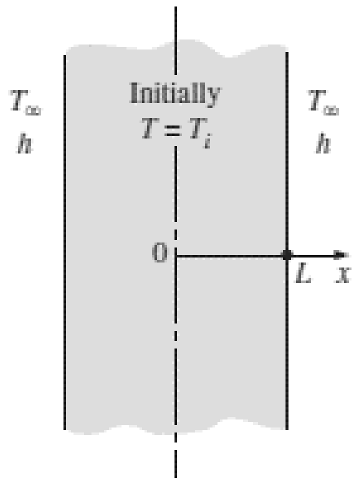


Figure 22: An infinite steel plate [22]

3.3.2 Governing Equation

This case can be described by Fourier’s 1D transient heat conduction equation [24]:

$$\frac{\partial T'}{\partial t'} = \alpha \frac{\partial^2 T'}{\partial x'^2} \tag{25}$$

where $t',\ x',\ T'$are dimensional time, distance and temperature,

with the initial condition:

$$T = 1\ for\ 0 \le x \le 1, t = 0 \tag{26}$$

And the boundary conditions:

$$k\frac{\partial T}{\partial x} = hT \; at \; x = 0, t > 0 \tag{27}$$

$$k\frac{\partial T}{\partial x} = -hT \; at \; x = 1, t > 0$$

3.3.3 Analytical Solution

Using the variables in dimensionless form is one of common techniques to solve parabolic equations, in case of transient heat conduction for instance. One solution can often be used for similar problems with very different linear dimensions, thermal conductivities, and temperatures. The transient equation (25) applied to an uniform rod of length L with T_i at zero time can become dimensionless with following variables [25]:

$$x = \frac{x'}{L}, \; T = \frac{T'}{T_i}, t = \frac{\alpha t'}{L^2} \tag{28}$$

$$\frac{\partial T}{\partial t} = \alpha\frac{\partial^2 T}{\partial x^2} \tag{29}$$

The next step to analyze the system is to calculate the *Biot* number (*Bi*). This will characterize the problem as lumped or not lumped system. The numerical value of *Bi* is a criterion which indicates the importance of conduction and convection in temperature distribution of an object being cooled or heated by convection at its surface [26]. Figure 23 shows the physical meaning of Bi, when $Bi \ll 1$ the heat transfer due to conduction can be neglected, and on the other hand, when $Bi \gg 1$, the heat conduction is primary mechanism of heat transfer.

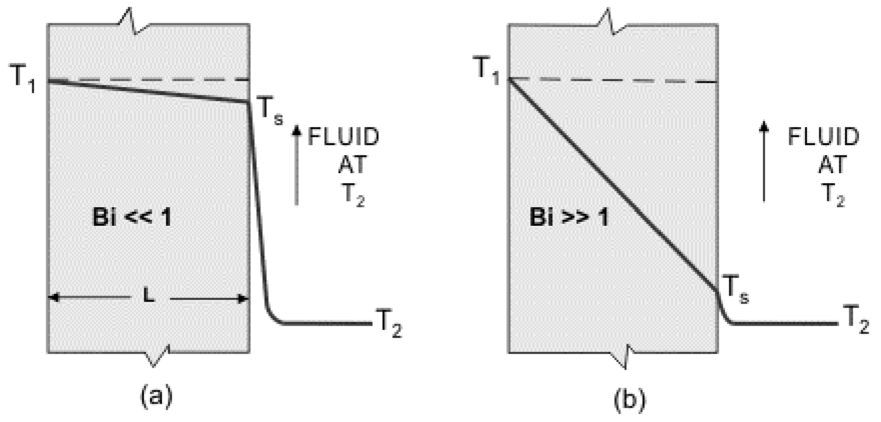


Figure 23: Comparison of temperature profiles in two plates cooled by the same fluid [27]

$$Bi = \frac{R_{cond}}{R_{conv}} = \frac{h\,L_c}{k} \tag{30}$$

where $L_c = \frac{V}{A} = \frac{A*2L}{2\,A} = L$ is a characteristic length defined as the ratio of the volume of the body to its surface area and L is a half thickness of the plate in this case. For the problem observed in this master thesis, $Bi \gg 0{,}1$ is true. Thus, lumped system analysis is not applicable and the temperature gradient in the body is not negligible. [22]

The analytical solution to this problem, given in [17], represented in form of series, which, on its turn, is rapidly convergent.

$$T = 4\sum_{i=1}^{\infty} \frac{cos\left[2p_i\left(x - \frac{1}{2}\right)\right]}{e^{4p_i^2 t}(3 + 4p_i^2)\cos(p_i)}, \quad 0 \le x \le 1 \tag{31}$$

where p_i are the positive roots of the equation $p\tan(p) = \frac{1}{2}$.

3.3.4 Preprocessing

Repeating the first step of the previous simulation, the case directory shall be created named *Transient*.

Furthermore, this case directory will be modified in order to show the 1D transient heat conduction problem, described in chapter 3.3.1.

3.3.4.1 Mesh Generation

Since the problem described is 1D, which is the same as the previous case, a 3D mesh will be created by *blockMesh* utility, and then the dimensions with no solutions will be depicted as arbitrary. To create a mesh, the file in system directory named *blockMeshDict* should be created.

The entries in the *blockMeshDict* code have the same meaning as in chapter 3.1.4.1. The code (Appendix 13*)* is modified in order to obtain the plate representation with the z-axis as arbitrary. The solution will be done in x-axis, however, the y-axis is constructed in order to show the correct domain.

Once the *blockMeshDict* is set up, the mesh can be run and, then stored in the *constant* directory.

3.3.4.2 Boundary and Initial Conditions Set Up

The next step of the simulation is to introduce boundary and initial conditions of temperature field. For each field of interest there is a corresponding file inside with *0* folder.

- ***internalField***. This entry defines the initial internal field. *uniform* sets the specified value (1) to all the internal elements, since initially the plate at the constant hot temperature.
- ***boundaryField.*** Introduces the boundary conditions at each of the patches, which were defined in *blockMeshDict.*
    - *frontAndBack, top and Bottom. empty* is used in case of 1D and 2D cases. Since the mesh OpenFOAM produces is always 3D, this entry type means that the z-axis is chosen as arbitrary.
    - *right* and *left* Type *groovyBC* is used on the left and right sides of the plate in order to set up the condition of sudden cooling process of an object by the surrounded fluid (natural convection only is taken into account).

The temperature field file is shown in Appendix 17.

3.3.4.3 Physical Properties

The only physical property OpenFOAM needs for this case is the thermal diffusivity of the material (the parameter α from the heat equation), which is defined within *transportProperties* and named DT. The unit, which could be depicted in this code as well, is $\left[\frac{m^2}{s}\right]$. The code is the following:

```
DT          DT [ 0 2 -1 0 0 0 0 ] 0.03;
```

3.3.4.4 Time Settings

The parameters, which control the time of simulation and the input/output of the data are written in the *controlDict* file. In this case the time controls shall be set up carefully, because the solution is heavily dependent on time. The used code is shown in Appendix 16.

The entries in the *controlDict* code have the same meaning as in the Chapter 3.1.4.4. The code is modified in order to achieve the solution after 2 minutes of sudden cooling of the plate. The time intervals were chosen in such a manner that the analytical and numerical results could be compared.

3.3.4.5 Discretization and Linear Solver Set Up

The finite volume, as it was mentioned in chapter 2.5.1, was chosen to be discretization schemes. This method will be specified in the *fvSchemes* file. *fvSolution* file specifies the linear equation solvers and tolerances and other algorithm controls.

The code in the file *fvSchemes* is shown in Appendix 15.

In this file it is important to change the ***ddtSchemes*** entry to *Euler*. This will mean, that the result will represent a transient solution over the run-time.

The code in the file *fvSolution* is shown in Appendix 16.

3.3.5 Running the Simulation

Having prepared the case files, the solver can be run by typing in the case directory:

laplacianFoam

This will generate files in the time directory for each time-step.

In order to observe the results in visualization software, one more step should be done in the same case directory:

touch Transient.foam

This will generate a blank file in the case directory with the file extension *.foam*.

3.3.6 Post-processing

The file with the *.foam* extension can be opened in Paraview and then the button *apply* can be clicked. Figure 24 presents the solution for the first time step (whereas Figure 25 shows the solution for the last step) with the temperature distribution over the whole domain. In case the domain is colored only in red or blue, it is a good idea to *Rescale to visible range* (in Chapter 3.2.5 it was discussed in more detail).

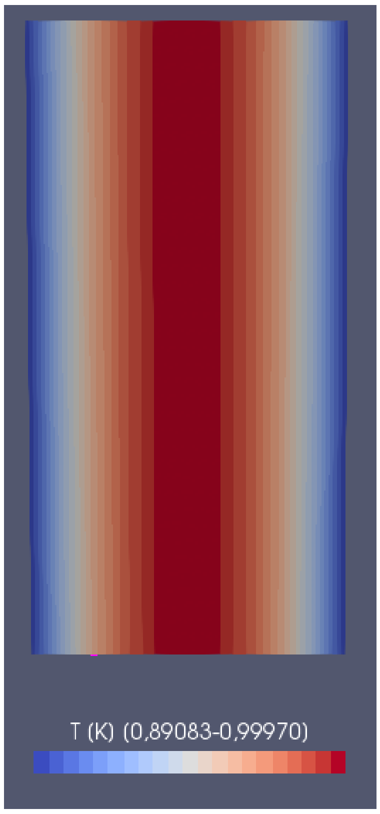


Figure 24: The 1D transient problem OpenFOAM solution of the first time step

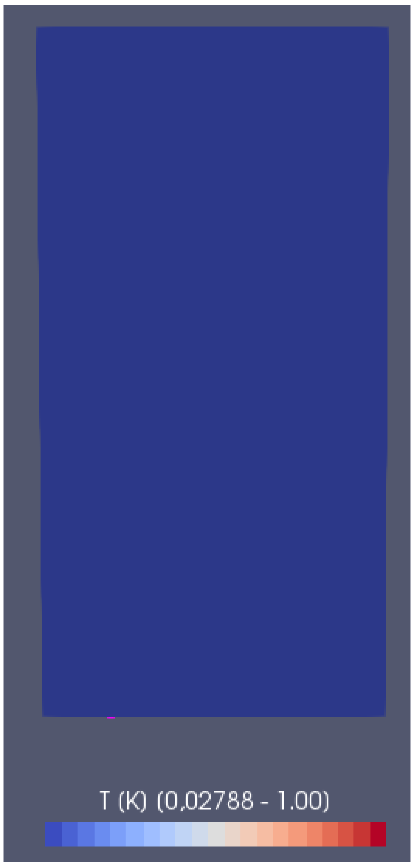


Figure 25: The 1D transient problem OpenFOAM solution of the last time step

3.3.7 Result and Discussion

Figure 26shows the comparison between the analytical and OpenFOAM solution, where the x-axis is the dimensionless distance x/L and y-axis is the temperature in [K]. Seven time intervals are constructed in this graph with the base line representing analytical solutions and the dashed line for numerical solutions.

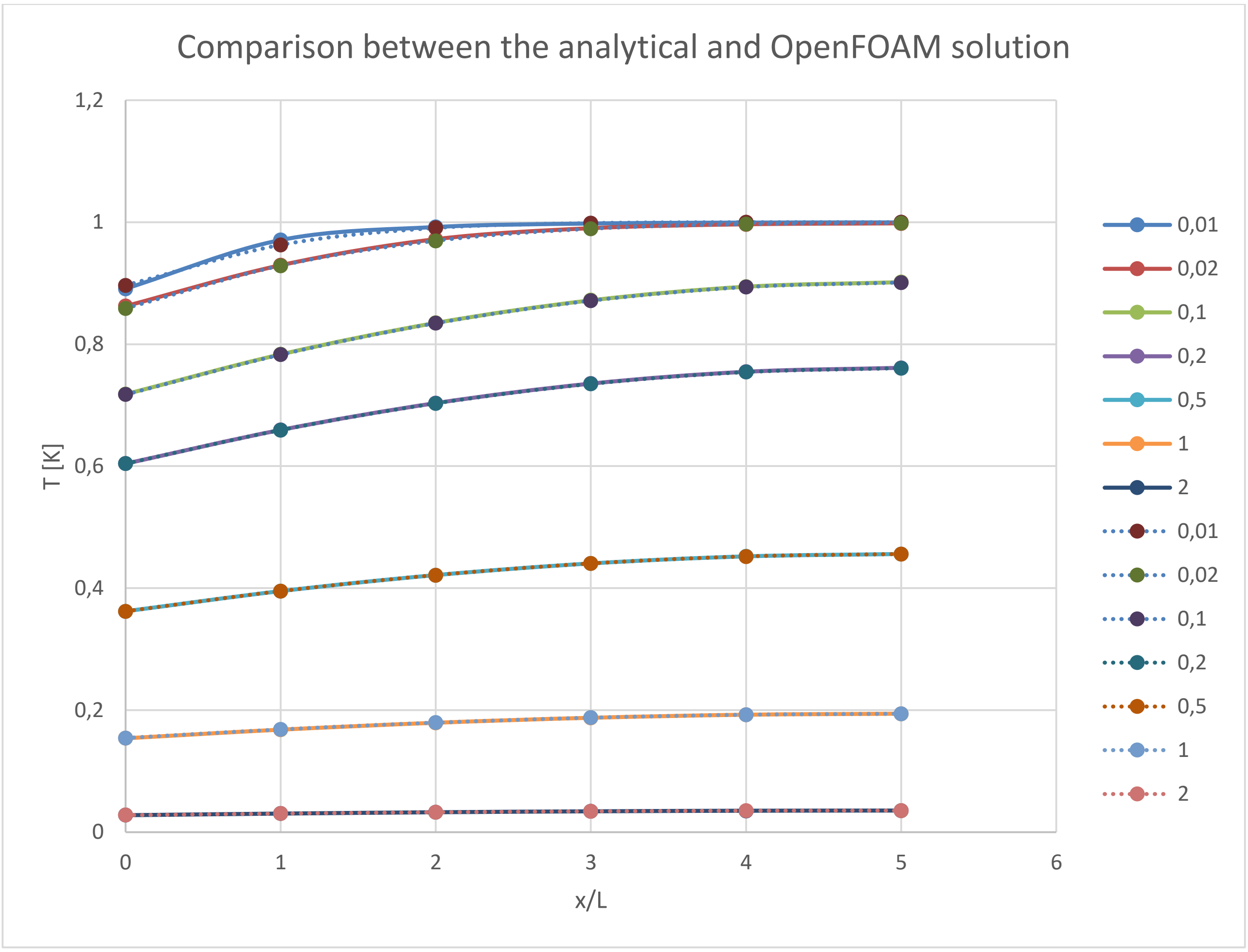


Figure 26: The graphical comparison between the analytical and OpenFOAM solution

The high accuracy of the results is achieved throughout, and the greatest error is less than 1%. This means that this implicit calculation gives stable, convergent and accurate solution [28].

3.4 Transient Heat Conduction in 3D

The last case is performed in order to show the capability of a numerical approach to solve heat conduction problems in more complex 3D geometries.

3.4.1 Description of the Case

A flange, initially at an uniform temperature of $T_1 = 273{,}15\ K,$ is heated up from the inner part of raised face with constant temperature of $T_1 = 573{,}15\ K,$ The geometry is represented in Figure 27: Paraview visualization of *Flange* mesh.

3.4.2 Preprocessing

Repeating the first step of the previous simulation, the case directory shall be created named *Flange*.

Furthermore, this case directory will be modified in order to show the transient heat conduction problem in the 3D complex geometry.

3.4.2.1 Mesh Generation

The 3D mesh for the Flange problem is not possible to construct with the same utility as in the previous cases. Due to its complexity the flange mesh should be constructed using either *snappyHexMesh* or *cfMesh* utility, or open source software to generate a mesh file, which can be later on opened in OpenFOAM. The tutorials how to use those utilities are given in the Bahram, et. al [29], however, for this master thesis the toolbox Ansys is used to develop a mesh representing the Flange problem (Figure 27).

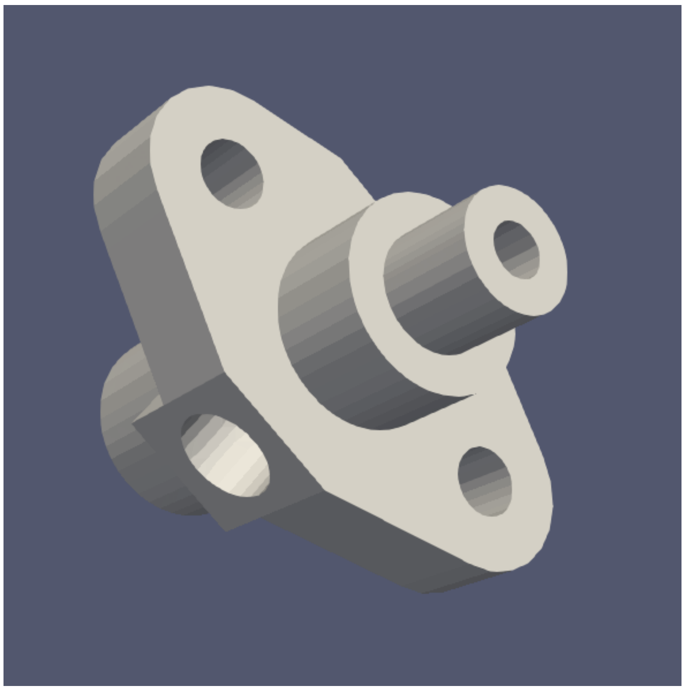


Figure 27: Paraview visualization of *Flange* mesh

3.4.2.2 Boundary and Initial Conditions Set Up

The boundary and initial conditions of temperature field are introduced in Appendix 18.

- ***internalField.*** This entry defines the initial internal field. *uniform* sets the specified value (273,5) to all the internal elements, since initially the plate at the constant hot temperature.
- ***boundaryField.*** Introduces the boundary conditions at each of the patches.
    - *patch 1 and patch 2. zeroGradient* type sets the normal gradient *($\pm\frac{\partial T}{\partial y}$)* to 0, establishing the adiabatic condition at outermost part of a flange.
    - *patch2 and patch 4. fixedValue* (for 273,15 and 573,15 values respectively) type is used to specify the hot temperature in the inner part of the flange. This temperature was assumed to be constant over time.

3.4.2.3 Physical Properties

The thermal diffusivity of the material is set up for the case (the parameter α from the heat equation), which is defined within *transportProperties* and named DT. The unit, which could be depicted in this code as well, is $\left[\frac{m^2}{s}\right]$. The code is the following:

```
DT          DT [ 0 2 -1 0 0 0 0 ] 4e-05;
```

3.4.2.4 Time Settings

The parameters, which control the time of simulation and the input/output of the data are written in the *controlDict* file. In this case the time controls shall be set up carefully, because the solution is heavily dependent on time. The used code is shown in Appendix 19.

3.4.2.5 Discretization and Linear Solver Set Up

The finite volume, as it was mentioned in chapter 2.5.1, was chosen to be discretization schemes. This method will be specified in the *fvSchemes* file. In this file it is important to change the ***ddtSchemes*** entry to *Euler*. This will mean, that the result will represent a transient solution over the run-time. *fvSolution* file specifies the linear equation solvers and tolerances and other algorithm controls. The codes are presented in Appendix 20 and Appendix 21

3.4.3 Running the Simulation

The solver can be run by entering in the case directory: *laplacianFoam*. This will generate files in the time directory for each time-step.

In order to observe the results in visualization software, one more step should be done in the same case directory: *touch Flange.foam*. This will generate a blank file in the case directory with the file extension *.foam*.

3.4.4 Post-processing

The file with the *.foam* extension can be opened in Paraview and then the button *apply* can be clicked.

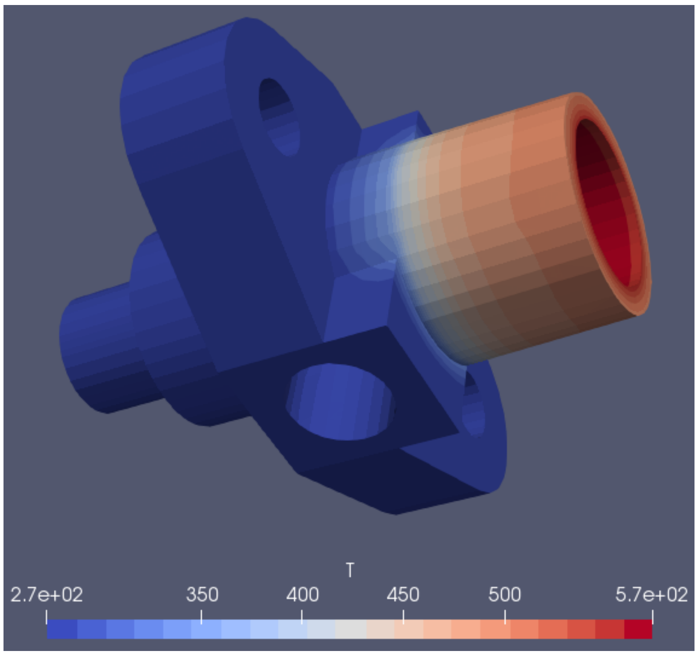


Figure 28: Temperature distribution in flange, the first time step

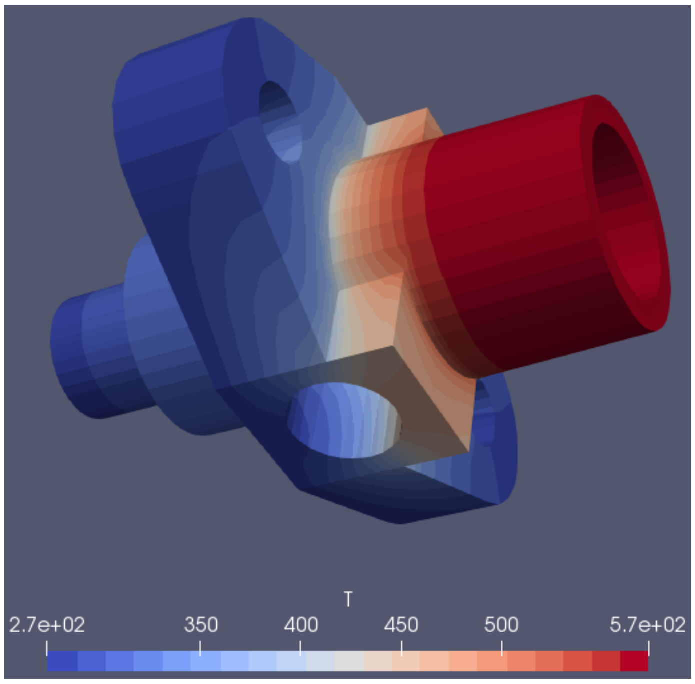


Figure 29: Temperature distribution in flange, the last time step

4 CONCLUSION

This section concludes the results and evaluates a computer simulation utilization for solving fluid dynamics and heat transfer problems in comparison to a theoretical approach. A computational method to solve heat transfer problems is free of some of constraints imposed on other methods, such as analytical and experimental, thus, it might provide information that is not available by any other means. On the other hand, this approach also has limitations from computer storage and computation speed through to the inability to understand and mathematically model certain complex phenomena.

In the computational method, a limited number of assumptions are made and a high-speed computer shall be used to solve the resulting governing fluid dynamic equations. In order to achieve an accurate numerical solution, certain parameters should be described: governing equation of a problem, the functional dependence of temperature on space and time coordinates, boundary conditions, initial conditions, material properties and geometry of the body. When all information important for a numerical method has been defined, the temperature field can be calculated, and moreover, the heat flux, thermal stresses, expansion, deflection, design insulation thickness, heat treatment methods are possible to compute and analyze.

The solution to a heat conduction problem is defined at nodes inside each cell. The accuracy of a CFD solution is governed by the number of cells in the grid. In general, the larger the number of cells the better the solution accuracy. The level of detail of the grid affects both the accuracy of a solution and its cost in terms of calculation time. To verify the results obtained by the computational method, it is important to compare them with the existing analytical solution. In case the conditions were set up correctly and the error between analytical and computed results is rather considerable, the decision shall be taken either to change a grid and increase computational time, otherwise these results are sufficient for the solution.

As it has been seen in the case of the critical insulation radius, the certain refinement was necessary in order to obtain an accurate solution. Mesh refinement is a common practice to get a higher resolution in regions of greater interest, however, initially the optimal CFD result for a system shall be determined.

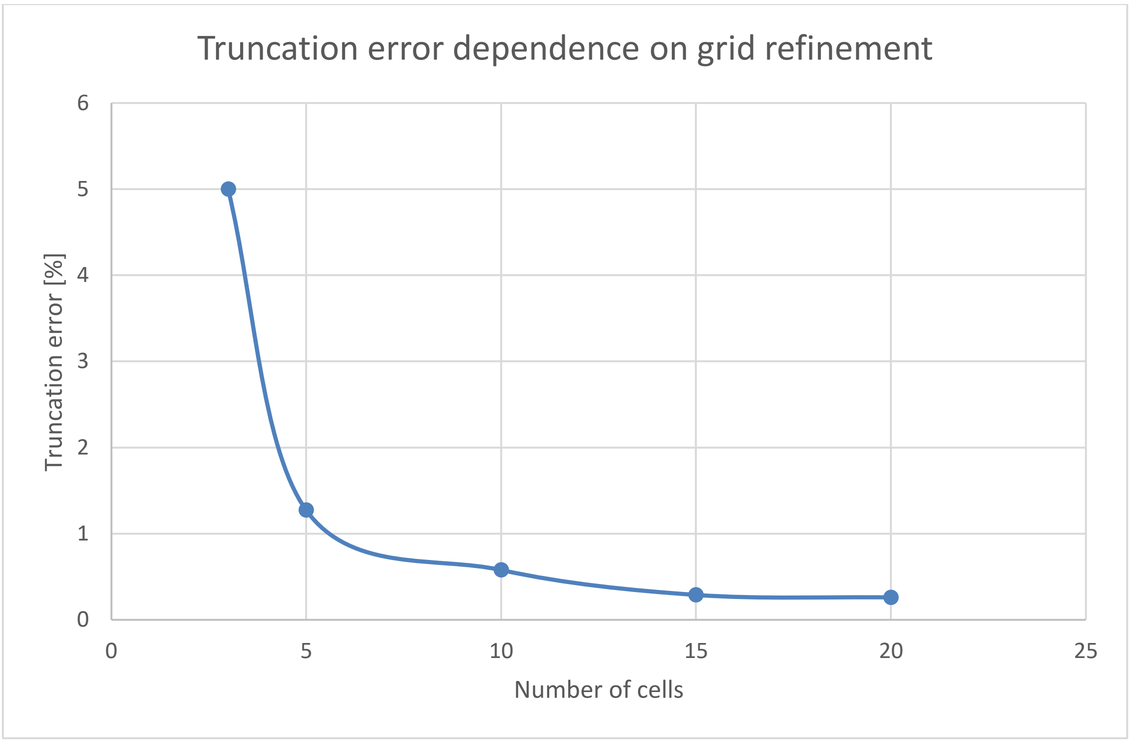


Figure 30: The accuracy of numerical solution dependence on grid refinement

Figure 30 shows the dependence on the total number of cells in the grid of truncation error between the analytical precise solution and numerical solution calculated by OpenFOAM for the case of the critical insulation radius. The conclusion can be made that after certain degree of refinement the error does not considerably change, and that 10 cells in this case is the optimum choice.

The ongoing advantages of computer analysis comparing to theoretical calculation is sufficiently impressive that the industry looks more closely in this direction. A computational approach works out the implications of an analytical method, and the user receives an end result that depends on both the mathematical model and the numerical method.

Finally, the toolbox used in this thesis has several advantages in contrast to any proprietary software and is widely utilized in academic and industrial researches. First of all, it works with the accessible and modifiable source code and there are no licence costs, which is a great virtue for students. Likewise, OpenFOAM has massive capabilities to solve variety of problems in CFD and a wide range of applications and models ready to use.

5 References

[1] J. H. Lienhard, *A heat transfer textbook,* 3rd ed. Cambridge Mass.: Phlogiston Press, 2001.

[2] F. P. Incropera, D. P. DeWitt, T. L. Bergman, and A. Lavine, *Foundations to heat transfer,* 6th ed. Singapore: John Wiley & Sons, 2013.

[3] Y. A. Çengel, *Introduction to thermodynamics and heat transfer: Second edition*: McGraw−Hill Primis, 2003.

[4] W. Beitz and K.-H. Küttner, *Dubbel Handbook of Mechanical Engineering*. London: Springer London, 1994.

[5] W. S. Janna, *Engineering heat transfer,* 2nd ed. Boca Raton, Fla.: CRC Press, 2000.

[6] W. H. Press, *Numerical recipes: The art of scientific computing,* 3rd ed. Cambridge, UK, New York: Cambridge University Press, 2007.

[7] R. J. LeVeque, *Finite difference methods for ordinary and partial differential equations: Steady-state and time-dependent problems / Randall J. LeVeque*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2007.

[8] J. Kevorkian, *Partial differential equations: Analytical solution techniques / J. Kevorkian,* 2nd ed. New York, London: Springer, 2000.

[9] S. Mazumder, *Numerical methods for partial differential equations: Finite difference and finite volume methods / Sandip Mazumder*. Amsterdam: Academic Press, 2015.

[10] V. D. Liseĭkin, *Grid generation methods*. Cham, Switzerland: Springer, 2017.

[11] S. V. Patankar, *Numerical heat transfer and fluid flow*. Washington, London: Hemisphere, 1980.

[12] H. K. Versteeg and W. Malalasekera, *An introduction to computational fluid dynamics: The finite volume method / H.K. Versteeg and W. Malalasekera*. Harlow: Longman Scientific & Technical, 1995.

[13] W. J. Minkowycz, E. M. Sparrow, and J. Murthy, *Handbook of numerical heat transfer,* 2nd ed. Hoboken, N.J.: John Wiley; Chichester : John Wiley [distributor], 2006.

[14] *OpenFOAM User Guide: CFD Direct, Architects of OpenFOAM.* [Online] Available: https://cfd.direct/openfoam/user-guide/. Accessed on: Jun. 18 2018.

[15] I. D.m. Vita P., *CCM Lecture Notes: Basic of Heat transfer*.

[16] H. S. Carslaw and J. C. Jaeger, *Conduction of heat in solids,* 2nd ed. Oxford: Clarendon, 1973.

[17] F. J. Bayley, J. M. Owen, and A. B. Turner, *Heat transfer*. London: Nelson, 1972.

[18] *Contrib/groovyBC - OpenFOAMWiki.* [Online] Available: https://openfoamwiki.net/index.php/Contrib_groovyBC. Accessed on: Jun. 18 2018.

[19] *Insulation materials and their thermal properties.* [Online] Available: http://www.greenspec.co.uk/building-design/insulation-materials-thermal-properties/. Accessed on: Aug. 05 2018.

[20] E. M. Sparrow and S. S. Kang, “Two-dimensional heat transfer and critical radius results for natural convection about an insulated horizontal cylinder,”

International Journal of Heat and Mass Transfer, vol. 28, no. 11, pp. 2049–2060, 1985.

[21] Prof.N.B.Totala, Anant A.Paralkar,Nikhil R.Kakade, Kunal R.Kawthekar, Shashank R.Borde, “Analysis for critical radius of insulation for a cylinder,” *IOSR Journal of Engineering*, vol. 3, no. 9, 2013.

[22] Y. A. Çengel, *Heat and mass transfer: A practical approach / Yunus A. Çengel,* 3rd ed. Boston, London: McGraw-Hill, 2007.

[23] *Thermal Conductivity of Metals.* [Online] Available: https://www.engineeringtoolbox.com/thermal-conductivity-metals-d_858.html. Accessed on: Aug. 27 2018.

[24] D. A. Anderson, R. H. Pletcher, and J. C. Tannehill, *Computational fluid mechanics and heat transfer*. Boca Raton: CRC Press Taylor & Francis Group, 2013.

[25] University of Nebraska-Lincoln | Web Developer Network, *Exact Analytical Conduction Toolbox.* [Online] Available: http://exact.unl.edu/exact/home/home.php. Accessed on: Aug. 20 2018.

[26] T. W. Davies, “BIOT NUMBER,” in *A-to-Z Guide to Thermodynamics, Heat and Mass Transfer, and Fluids Engineering*: Begellhouse, 2006.

[27] W. Brandstaetter, *Thermodynamics and Heat Transfer: Lecture Nr.: 306029.* Accessed on: Aug. 06 2018.

[28] G. D. Smith, *Numerical solution of partial differential equations: Finite difference methods / G.D. Smith,* 3rd ed. Oxford: Clarendon, 1985.

[29] Bahram Haddadi, Christian Jordan, Michael Harasek, *OpenFOAM Basic Training. Tutorial Twelve. snappyHexMesh - Single Region*. 4th edition.

ABBREVIATIONS

CFD	Computational fluid dynamics
CCM	Computational continuum mechanics
1D	One-dimensional
2D	Two-dimensional
3D	Three-dimensional
PDE	Partial differential equation
FVM	Finite volume method
CV	Control volume

NOMENCLATURE

Symbol	Unit	Definition
$q"$	$[\frac{W}{m^2*s}]$	heat flux vector
k	$[\frac{W}{m*K}]$	thermal conductivity
ρ	$[\frac{kg}{m^3}]$	mass density
c	$[\frac{J}{K}]$	specific heat
Q	$[J]$	heat energy
α	$[\frac{m}{s^2}]$	thermal diffusivity
ϑ	$[\frac{m}{s}]$	velocity of solid material
L	$[m]$	length
t	$[s]$	time
h	$[\frac{W}{m^2*K}]$	convective heat transfer coefficient
r_1	$[m]$	radius of cylinder
r_2	$[m]$	radius of insulation
Bi	-	Biot number
R_{cond}		heat transfer by conduction
L_c	$[m]$	characteristic length
R_{conv}		heat transfer by convection
x, T, t	-	dimensionless distance, temperature and time
V	$[m^3]$	volume
A	$[m^2]$	area

APPENDICES

```
convertToMeters 1;
vertices
(    (0 0 0)
    (1 0 0)
    (1 1 0)
    (0 1 0)
    (0 0 0.1)
    (1 0 0.1)
    (1 1 0.1)
    (0 1 0.1)  );
blocks
(    hex (0 1 2 3 4 5 6 7) (10 10 1) simpleGrading (1 1 1)  );
edges
(
);
boundary
(    left
    {   type patch;
        faces
        (           (0 4 7 3)        );    }
    right
    {   type patch;
        faces
        (           (2 6 5 1)        );    }
    top
    {   type patch;
        faces
        (           (3 7 6 2)              );    }
```

```
  bottom
  {   type patch;
      faces
      (                   (1 5 4 0)       );    }
  frontAndBack
  {    type empty;
      faces
      (          (0 3 2 1)
                 (4 5 6 7)       );    }
);
```

Appendix 1 *blockMeshDict* file, case “SS conduction in 2D Slab”

```
dimensions      [0 0 0 1 0 0 0];
internalField   uniform 0;
boundaryField
{    frontAndBack
  {       type    empty;    }
  bottom
  {       type    fixedValue;
      value   uniform 1;    }
  top
  {       type    zeroGradient;    }
  right
  {       type    groovyBC;
      gradientExpression "-10.*T";
      fractionExpression "0";
      value   uniform 0;    }
  left
  {       type    zeroGradient;    }
}
```

Appendix 2 Temperature field file *0*, case “SS conduction in 2D Slab”

```
application     laplacianFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         30;

deltaT          0.1;

writeControl    runTime;

writeInterval   10;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression uncompressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;

libs            ("libgroovyBC.so");
```

Appendix 3 *controlDict* file, case “SS conduction in 2D Slab”

```
ddtSchemes
{   default    Euler; }
gradSchemes
{   default    Gauss linear; }
divSchemes
{   default    none; }
laplacianSchemes
{   default    none;
   laplacian(DT,T) Gauss linear corrected;  }
interpolationSchemes
{   default    linear;   }
snGradSchemes
{   default    corrected;   }
fluxRequired
{   default    no;
   T;
}
```

Appendix 4 *fvSchemes* file, case “SS conduction in 2D Slab”

```
solvers
{   T
   {       solver     GAMG;
          tolerance   1e-7;
          relTol     0;


      smoother   GaussSeidel;


      cacheAgglomeration true;
      nCellsInCoarsestLevel 8;
      agglomerator faceAreaPair;
      mergeLevels 1;    };
}
SIMPLE
{   nNonOrthogonalCorrectors 0;  }
```

Appendix 5 *fvSolutions* file, case “SS conduction in 2D Slab”

```
convertToMeters 0.1;
vertices
(  (-5.000000 -0.000000 0.000000)
(-2.500000 -4.330127 0.000000)
(0.000000 -5.000000 0.000000)
(2.500000 -4.330127 0.000000)
(5.000000 -0.000000 0.000000)
(2.500000 4.330127 0.000000)
(0.000000 5.000000 0.000000)
(-2.500000 4.330127 0.000000)
(-3.000000 -0.000000 0.000000)
(-1.500000 -2.598078 0.000000)
(0.000000 -3.000000 0.000000)
(1.500000 -2.598078 0.000000)
(3.000000 -0.000000 0.000000)
(1.500000 2.598078 0.000000)
(0.000000 3.000000 0.000000)
(-1.500000 2.598078 0.000000)
(-5.000000 -0.000000 1.000000)
(-2.500000 -4.330127 1.000000)
(0.000000 -5.000000 1.000000)
(2.500000 -4.330127 1.000000)
(5.000000 -0.000000 1.000000)
(2.500000 4.330127 1.000000)
(0.000000 5.000000 1.000000)
(-2.500000 4.330127 1.000000)
(-3.000000 -0.000000 1.000000)
(-1.500000 -2.598078 1.000000)
(0.000000 -3.000000 1.000000)
(1.500000 -2.598078 1.000000)
(3.000000 -0.000000 1.000000)
(1.500000 2.598078 1.000000)
(0.000000 3.000000 1.000000)
(-1.500000 2.598078 1.000000)  );
```

Appendix 6 *blockMeshDict* file, case “Radius”

```
blocks
(
hex (2 10 8 0 18 26 24 16) (10 10 1) simpleGrading (1 1 1)
hex (2 4 12 10 18 20 28 26) (10 10 1) simpleGrading (1 1 1)
hex (12 4 6 14 28 20 22 30) (10 10 1) simpleGrading (1 1 1)
hex (0 8 14 6 16 24 30 22) (10 10 1) simpleGrading (1 1 1)
);
edges
(  arc 0 2 (-3.750000 -3.307189 0.000000)
arc 2 4 (1.250000 -4.841229 0.000000)
arc 4 6 (3.750000 3.307189 0.000000)
arc 6 0 (-1.250000 4.841229 0.000000)
arc 8 10 (-2.250000 -1.984314 0.000000)
arc 10 12 (0.750000 -2.904744 0.000000)
arc 12 14 (2.250000 1.984314 0.000000)
arc 14 8 (-0.750000 2.904744 0.000000)
arc 16 18 (-3.750000 -3.307189 1.000000)
arc 18 20 (1.250000 -4.841229 1.000000)
arc 20 22 (3.750000 3.307189 1.000000)
arc 22 16 (-1.250000 4.841229 1.000000)
arc 24 26 (-2.250000 -1.984314 1.000000)
arc 26 28 (0.750000 -2.904744 1.000000)
arc 28 30 (2.250000 1.984314 1.000000)
arc 30 24 (-0.750000 2.904744 1.000000)
);
```

Appendix 7 *blockMeshDict* file, case “Radius”

```
boundary
(
   out
   {   type wall;
       faces
       (           (0 2 18 16)
                   (2 4 20 18)
                   (4 6 22 20)
                   (6 0 16 22)    );
   }
   in
   {    type wall;
       faces
       (         (10 8 24 26)
                 (12 10 26 28)
                 (14 12 28 30)
                 (8 14 30 24)     );
   }
);
//
```

Appendix 8 *blockMeshDict* file, case “Radius”

```
dimensions      [0 0 0 1 0 0 0];
internalField   uniform 0;
boundaryField
{   in
   {      type    fixedValue;
          value   uniform 373;    }
   out
   {      type    groovyBC;
          gradientExpression "-1.*T";
          fractionExpression "0";
          value   uniform 273;    }
   left
   {      type    zeroGradient;    }
}
```

Appendix 9 Temperature field file *0*, case “Radius”

```
application     laplacianFoam;
startFrom       startTime;
startTime       0;
stopAt          endTime;
endTime         100;
deltaT          10;
writeControl    runTime;
writeInterval   20;
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression uncompressed;
timeFormat      general;
timePrecision   6;
runTimeModifiable yes;
libs            ("libgroovyBC.so");
```

Appendix 10 *controlDict* file, case “Radius”

```
ddtSchemes
{   default    steadyState; }
gradSchemes
{   default    Gauss linear; }
divSchemes
{   default    none
   div(phi,T) Gauss limitedLinear 1;
}
laplacianSchemes
{   default    none;
   laplacian(DT,T) Gauss linear corrected; }
interpolationSchemes
{   default    linear; }
snGradSchemes
{   default    corrected; }
fluxRequired
{   default    no;
   T;
}
```

Appendix 11 *fvSchemes file*, case “Radius

```
convertToMeters 0.01;
vertices
(    (0 0 0)
    (5 0 0)
    (5 10 0)
    (0 10 0)
    (0 0 0.1)
    (5 0 0.1)
    (5 10 0.1)
    (0 10 0.1)
);
blocks
(hex (0 1 2 3 4 5 6 7) (10 10 1) simpleGrading (1 1 1) );
edges
( );
boundary
(
    left
    {       type patch;
        faces
        (           (0 4 7 3)       );    }
    right
    {       type patch;
        faces       ((2 6 5 1)       );    }
```

Appendix 12 blockMeshDict file, “Transient” case

```
  top
  {     type empty;
     faces
     (        (3 7 6 2)          );   }
  bottom
  {     type empty;
     faces
     (                 (1 5 4 0)     );   }
  frontAndBack
  {     type empty;
     faces
     (        (0 3 2 1)
        (4 5 6 7)      );   } );
mergePatchPairs
(
);
```

Appendix 13 *blockMeshDict* file, case “Transient”

```
application    laplacianFoam;
startFrom      startTime;
startTime      0;
stopAt         endTime;
endTime        100;
deltaT         10;
writeControl   runTime;
writeInterval  20;
purgeWrite     0;
writeFormat    ascii;
writePrecision  6;
writeCompression uncompressed;
timeFormat     general;
timePrecision   6;
runTimeModifiable yes;
libs           ("libgroovyBC.so");
```

Appendix 14 *controlDict* file, case “Transient”

```
ddtSchemes
{   default     Euler; }
gradSchemes
{   default     Gauss linear; }
divSchemes
{   default     none;
   div(phi,T)  Gauss limitedLinear 1; }
laplacianSchemes
{   default     none;
   laplacian(DT,T) Gauss linear corrected; }
interpolationSchemes
{   default     linear; }
snGradSchemes
{   default     corrected; }
fluxRequired
{   default     no;
   T;  }
```

Appendix 15 *fvSchemes* file, case “Transient”

```
solvers
{   T
  {      solver     GAMG;
      tolerance   1e-7;
      relTol     0;

      smoother    GaussSeidel;

      cacheAgglomeration true;
      nCellsInCoarsestLevel 8;
      agglomerator faceAreaPair;
      mergeLevels 1;
  };  }
SIMPLE
{   nNonOrthogonalCorrectors 0;  }
```

Appendix 16 *fvSolution* file, case “Transient”

```
dimensions      [0 0 0 1 0 0 0];
internalField   uniform 1;
boundaryField
{   frontAndBack
   {       type    empty;
   }
   top
   {       type    empty;
   }
   bottom
   {       type    empty;
   }
   left
   {  type    groovyBC;
      gradientExpression "-1.*T";
      fractionExpression "0";
      value   uniform 0;
   }
   right
   {       type    groovyBC;
      gradientExpression "-1.*T";
      fractionExpression "0";
      value   uniform 0;
   }
}
```

Appendix 17 Temperature file *0*, case “Transient”

```
dimensions      [0 0 0 1 0 0 0];
internalField   uniform 273.15;
boundaryField
{   patch1
   {       type    zeroGradient;    }
   patch2
   {       type    fixedValue;
      value   uniform 273.15;    }
   patch3
   {       type    zeroGradient;    }
   patch4
   {       type    fixedValue;
      value   uniform 573.15;    }
}
// ************************************************************************* //
```

Appendix 18 Temperature file 0, “Flange” case

```
application     laplacianFoam;
startFrom       latestTime;
startTime       0;
stopAt          endTime;
endTime         3;
deltaT          0.005;
writeControl    runTime;
writeInterval   0.1;
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression uncompressed;
timeFormat      general;
timePrecision   6;
runTimeModifiable yes;
// ************************************************************************* //
```

Appendix 19 *ControlDict* file, “Flange” case

```
ddtSchemes
{   default       Euler; }
gradSchemes
{   default       none;
   grad(T)       Gauss linear; }
divSchemes
{   default       none; }
laplacianSchemes
{   default       none;
   laplacian(DT,T) Gauss linear corrected; }
interpolationSchemes
{   default       linear; }
snGradSchemes
{   default       corrected; }
fluxRequired
{   default       no;
   T            ; }
```

Appendix 20 fvSchemes file, “Flange” case

```
solvers
{   T
   {      solver     GAMG;
      tolerance   1e-7;
      relTol     0;


      smoother    GaussSeidel;


      cacheAgglomeration true;
      nCellsInCoarsestLevel 10;
      agglomerator faceAreaPair;
      mergeLevels 2;    };  }
SIMPLE
{   nNonOrthogonalCorrectors 2; }
```

Appendix 21 fvSolution file, “Flange” case