Summer Fellowship Report

On

CONJUGATE HEAT TRANSFER STUDIES ON BLUFF BODIES USING OPENFOAM

Submitted by

VIPINKUMAR C P

Under the guidance of

Dr. Samarjeet Chanda

Discipline of Mechanical Engineering

INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

July 2020

Acknowledgment

I would like to express my sincere gratitude to Dr. Samarjeet Chanda, Assistant Professor, Discipline of Mechanical Engineering**,** Indian Institute of Technology Palakkad for his guidance. I deeply thank FOSSEE, IIT Bombay for starting the fellowship program and providing me a work opportunity at Indian Institute of Technology Palakkad. In addition, I am thankful to all the FOSSEE members for their mentorship and help during my fellowship.

CONTENTS

Abstract

This study aims to modify existing OpenFOAM solvers in order to perform conjugate heat transfer studies on bluff bodies. In order to get introduced to the problem and obtain familiarity with the solver codes in OpenFOAM, the task of modifying an existing incompressible flow solver viz. *icoFoam to* solve thermal transport is carried out first. Following this, the conjugate heat transfer problem involving natural convection over a heated sphere is solved using *chtMultiregionSimpleFoam.* Finally the effects of temperature dependent thermo-physical properties are brought in and the results are validated with predictions obtained using existing correlations.

CHAPTER 1

MODIFYING EXISTING ICOFOAM SOLVER TO SOLVE THERMAL TRANSPORT

1.1 Introduction and problem description

Each solver included in OpenFOAM is a stand-alone application that solves a set of equations on a specified grid. Different solvers may share existing code but they can vary significantly among one another. In this work the existing solver *icoFoam*[1] solves for the pressure and velocity field in laminar incompressible flows is modified by adding a scalar equation for solving transient heat transfer.

The *icoFoam* solver solves the incompressible laminar Navier-Stokes equations using the PISO algorithm. The solver is inherently transient and uses the PISO algorithm to solve the continuity and momentum equations. The modification in the *icoFoam* solver is done by adding the scalar energy transport equation to calculate the temperature [2,3]. The transport equation for calculating temperature as follows.

$$\nabla.(\vec{u}T) - \alpha\nabla^2 T = -\frac{\partial T}{\partial t} \tag{1.1}$$

The modified solver is then tested with a two inlet elbow, in which air enters at two different velocities and temperature. The code for modified *icoFoam* solver is provided in the appendix of this report.

1.2 Elbow case study using modified solver

1.2.1 Background

This case study demonstrates the simulation of incompressible air flow through an elbow with two different inlets in which air enters at different temperatures, using modified *icoFoam* solver by adding the energy transport equation. The geometric model is created using Ansys Design Modeler and the meshing is done using Ansys meshing Academic R3. The simulations are performed using OpenFOAMv7. The velocity profiles, pressure and temperature distributions are obtained from the performed simulations.

1.2.2 Problem statement

Incompressible, flow of air entering at two different temperatures and mixing in an elbow as shown in Fig.1 is analysed. Air at a temperature 303 K enters into elbow through “inlet 1”with a velocity of 0.01 m/s, simultaneously air at temperature of 373 K enters into the elbow through “inlet 2” with a velocity of 0.04 m/s and the air mixture flows out through the “outlet”. The temperature and velocity distributions are calculated for this case. The steps followed for the analysis are:

1. Creating 2D Geometry by using Ansys Design Modeler utility [4].
2. Creating 2D mesh by using Ansys Fluent meshing utility [4].
3. Converting mesh to OpenFOAM compatible format using the utility *fluentMeshToFoam*[1].
4. Setting boundary/initial conditions (BC/IC).
5. Solving using newly modified solver.

The dimensions and problem description for the scenario mentioned above is depicted in Fig.1.

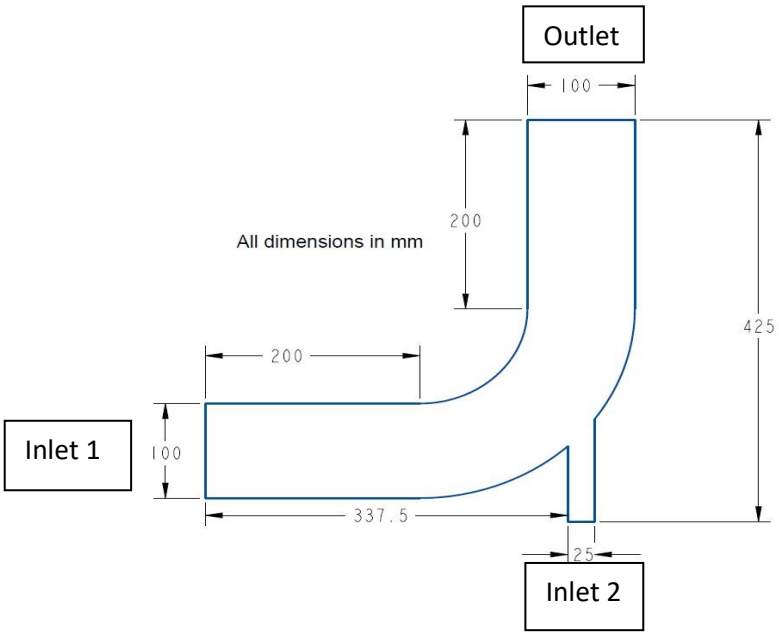

Fig. 1: Problem description

1.2.3 Meshing

The 2D meshing of the geometry (elbow) is generated using Ansys meshing utility [4]. Then the conversion of Ansys Fluent mesh to OpenFOAM compatible format is done by using the utility *fluentMeshToFoam* in the OpenFOAM. Fig. 2 shows the refined 2 D mesh used for the simulation generated using Ansys Fluent meshing utility.

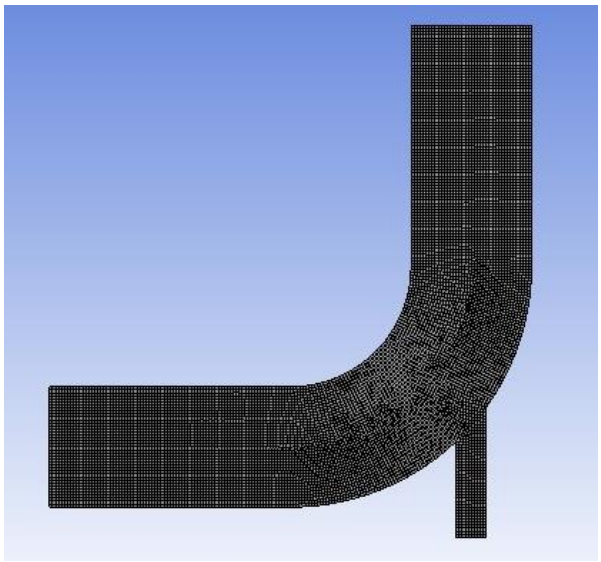

Fig. 2: Mesh for elbow generated using Ansys meshing

1.2.4 Simulation details (Initial and boundary conditions)

- Air is taken as the fluid that flows through the elbow. Kinematic viscosity of air [4] at 30℃ is $16\times 10^{-6}\,\frac{m^2}{s}$ and the thermal diffusivity [4] is $22.561\times 10^{-6}\,\frac{m^2}{s}$.
- At “inlet 1” air at a temperature 303 K enters into the elbow with a velocity of 0.01 m/s.
- At “inlet 2” air at a temperature 373 K enters into the elbow with a velocity of 0.04 m/s.
- The “outlet” is set as pressure outlet boundary condition with the gauge kinematic pressure prescribed as zero.
- The temperature of all the walls are prescribed as 303 K along with non-slip at the surface.
- The simulation is run for a total flow time of 125 s.

1.2.5 Results and discussion

Distributions of temperature, kinematic pressure and velocity in the elbow as obtained from the simulations is depicted in Figures 3, 4 and 5 respectively. Fig. 3 shows the distribution of temperature inside the elbow. The cold air from “inlet 1” (303K) and the hot air from “inlet 2” (373 K) combines and flows through the “outlet”.

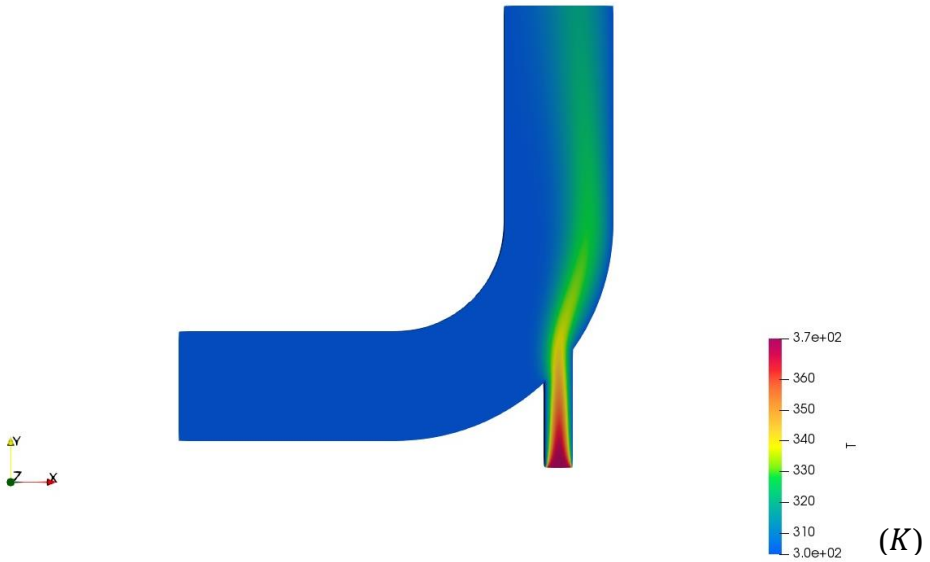

Fig. 3: Temperature distributions in the elbow obtained using modified solver in OpenFOAM at time (t) = 125s

Fig. 4: Kinematic pressure distributions in the elbow obtained using modified solver in OpenFOAM at time (t) = 125 s

Figure 4 shows the distribution of kinematic pressure inside the elbow. The hot air is initially at atmospheric pressure which flows through the “inlet 2” , while reaching the bend of the elbow, kinematic pressure decreases at the point where the two streams intersect that is because of the increased velocity at that region caused due to temperature effects.

Fig. 5: Velocity distribution in the elbow obtained using modified solver in OpenFOAM at time (t) = 125 s.

By referring to the above Fig. 5 it is observed that the velocity is maximum at “ inlet 2 ” and as the flow progress through the elbow the velocity starts decreasing because of the mixing of the two streams as a consequence of momentum exchange between them.

The contour plots show how the kinematic pressure gets decreased at the connecting “inlet 2” due to which kinematic pressure loss occur in the pipe and the velocity and temperature of the flow gets increased due to mixing. The flow velocity near the wall approaches zero because of the no-slip condition imposed at the walls. The inertia at the curve causes bend in the flow and at the entry of the “inlet 2” where the maximum mixing occurs and the temperature gradually decreases towards “outlet”. The results obtained were also compared with Ansys 19 Academic version and were found to be in good agreement.

1.3 Closure

Based on the simulation of incompressible air flow through an elbow with two different inlets in which air enters at different temperatures using modified *icoFoam* solver, the velocity profiles, kinematic pressure and temperature distributions were obtained. The results were also compared with the commercial software package and were found to be in good agreement.

CHAPTER 2

SIMULATION OF STEADY NATURAL CONVECTION AROUND A HOT SPHERE IN AIR

2.1 Introduction

Natural convection over a heated sphere is numerically solved as a conjugate heat transfer problem and the same is described in this chapter. The term conjugate heat transfer (CHT) is used to describe processes which involve variations of temperature within solids and fluids, due to the thermal interaction between them.

In OpenFOAM v1912 the steady Conjugate heat transfer problem involving solid and fluid regions are solved using the solver *chtMultiRegionSimpleFoam* [6] and it also facilitates a way to model variable thermo-physical properties to take into account the effect of temperature variation in the flow.

A steady case of natural convection in air surrounding a hot sphere is simulated using the solver *chtMultiRegionSimpleFoam* (OpenFOAMv1912). A sphere of 30 mm diameter with a wall temperature of 500 K is surrounded by an open volume of dry air at 1atm, initially quiescent and maintained at a uniform temperature of 300 K. The schematic below (Fig. 6) depicts the case being studied. The thermo-physical properties are evaluated at the mean film temperature.

Assumptions made for this simulation are:

- Steady state
- Laminar flow (Turbulent effects are not considered)
- Newtonian fluid
- Perfect gas
- Negligible radiation effect
- The thermo-physical properties of the fluids are functions of the temperature

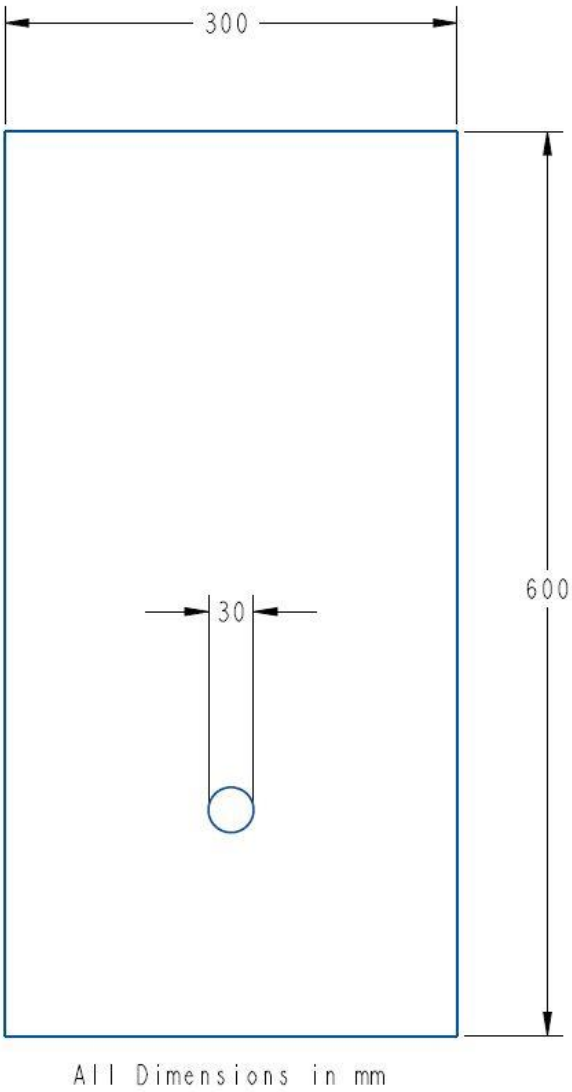

Fig. 6: Description of the problem

2.2 Meshing

SnappyHexMesh, a powerful 3D meshing tool which can generate meshes based on STL or OBJ geometry files is used [6,7]. The sphere of 30 mm diameter is created using a 3D modelling software and meshed using *snappyHexMesh*. Before using *snappyHexMesh* it is first necessary to generate a background mesh covering all the domain inside the air volume. This is usually done with *blockMesh*. The mesh is generated with *blockMesh* using the *blockMeshDict,* The *surfaceFeatures* for *snappyHexMesh* is generated using *surfaceFeatureExtractDict* and mesh regions (soild,air) is defined using *snappyHexMeshDict*. The mesh used in the study is depicted in Fig. 7.

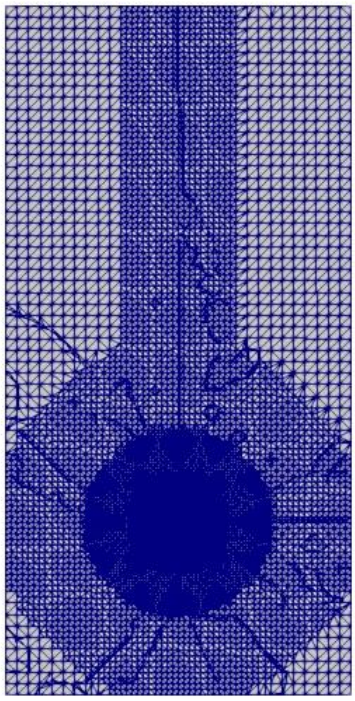

Fig. 7: Mesh and zones generated using *snappyHexMesh*

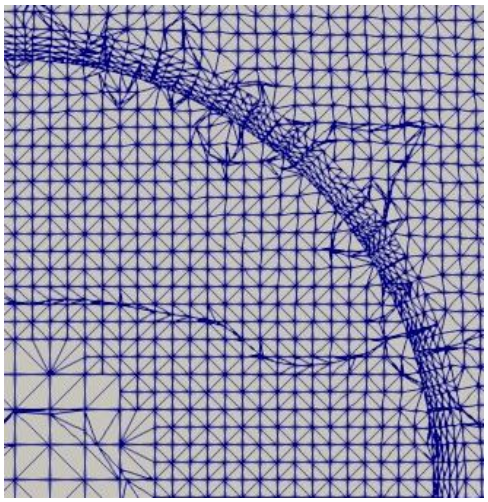

Fig. 8: Layer of cells over sphere created using add layer function in *snappyHexMesh*

To capture the physics associated with the fluid near the boundaries, a high quality layered mesh is used. The add layer function in *snappyHexMesh* defines parameters for adding a boundary layer mesh. Fig. 8 shows the layer of cells over sphere created using add layer function in *snappyHexMesh.*

2.3 Initial and boundary conditions

Setting the proper initial and boundary conditions for natural convection cases can be a delicate matter. The boundary patches are placed far from the sphere in order to minimize their effect around the sphere. The following conditions are set for the simulation

Air

- Velocity of air is set to 0 m/s in all direction.
- Pressure is assumed to be constant over the entire fluid domain and assigned a value of 1 atm.
- The temperature of air is assigned a value of 300 K over the entire domain.
- The outer boundary of air domain is assigned adiabatic wall boundary condition.
- Air to sphere Temperature coupling between air and the sphere is assigned.

Sphere (copper)

- The wall temperature of copper sphere is as set 500 K.
- Sphere to air Temperature coupling assigned.

2.4 Thermo-physical properties

In order to model the temperature dependence of thermo-physical properties, the thermo-physical properties dictionary is used for both air and the copper sphere regions. The properties of air and the copper sphere are defined using polynomial functions for wide range of temperatures [8].The solver solves each region separately according to the applicable equations and couples them with the appropriate prescribed boundary conditions. As far as the dependence of thermo-physical properties of air and copper sphere on temperature is concerned, it is modelled using the following expressions [5,9]:

For air (valid from 300-500 K)

$$\rho(T) = 2.7558 - 6.974 \times 10^{-3}T + 5.84 \times 10^{-6}T^2 \tag{2.1}$$

$$C_p(T) = 1.0406571429 \times 10^{3} - 2.4971428571 \times 10^{-1}T + 4.5714285714 \times 10^{-4}T^2 \tag{2.2}$$

$$\mu(T) = 1.4982857143 \times 10^{-6} + 6.4871428571 \times 10^{-8}T - 2.7714285714 \times 10^{-11}T^2 \quad (2.3)$$

$$k(T) = 7.5428571429 \times 10^{-4} \times 9.2771428571 \times 10^{-5}T - 2.5714285714 \times 10^{-8}T^2 \quad (2.4)$$

For copper sphere (valid from 300-600 K)

$$k(T) = 4.5555533587 \times 10^{2} - 3.0223468118 \times 10^{-1}T + 5.0530909016 \times 10^{-4}T^2 - 3.5717443591 \times 10^{-7}T^3 \quad (2.5)$$

$$C_p(T) = 1.8658483803 \times 10^{2} + 1.4920474673\mathrm{T} - 4.1752372544 \times 10^{-3}\mathrm{T}^2 + 5.5111636355 \times 10^{-6}\mathrm{T}^3 - 2.7194694879 \times 10^{-9}\mathrm{T}^4 \quad (2.6)$$

2.5 Results and discussion

2.5.1 Solution convergence

Residuals are the most fundamental measures of an iterative solution's convergence, as they directly quantify the error in the solution of the system of equations. In a CFD analysis, the residual measures the local imbalance of a conserved variable in each control volume. In an iterative numerical solution, the residual will never be exactly zero. However, the lower the residual value is, the more numerically accurate the solution. Here from Fig. 9 it shows that the residuals of the simulation carried out for around the 2600 iterations.

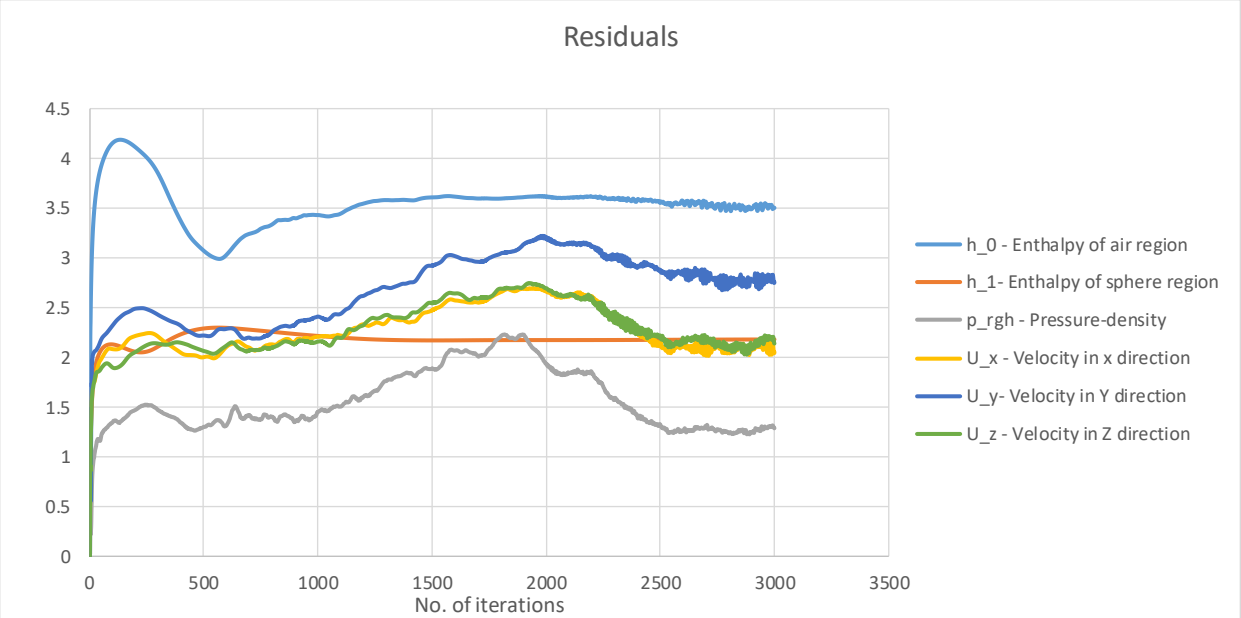

Fig. 9: Residual plot

2.5.2 Temperature and velocity profiles

Free convection fluid motion is due to buoyancy forces with in the fluid, while in forced convection it is externally imposed. Here we consider the situation for which there is no forced velocity, yet convection currents exists within the fluid. Such situations are referred to as free or natural convection and they originate due to the presence of density gradients in the medium. Here density gradient is due to a temperature gradient and the body force is due to the gravitational field. In absence of adjoining surface, free boundary flows may occur in the form of a plume or a buoyant jet. A plume is associated with the fluid rising from the submerged heated object [10].

Fig. 10 shows the heated plume discharged vertically quiescent medium that has air at a temperature.

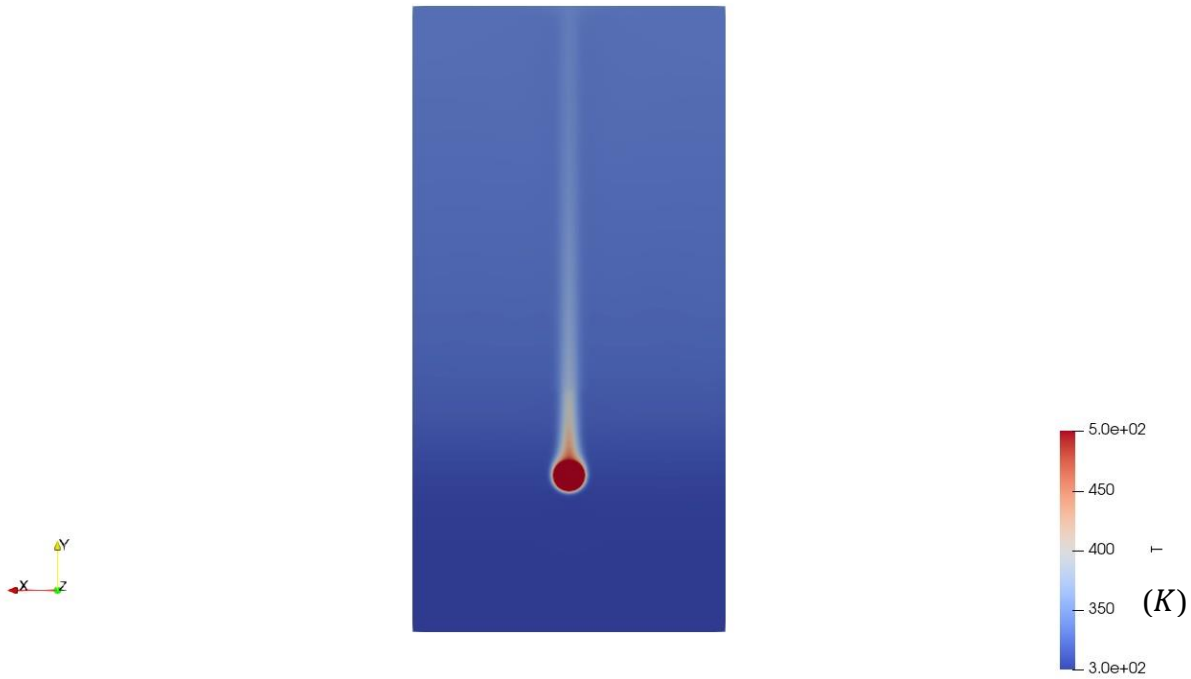

Fig. 10: Temperature profile around the hot sphere

Fig.11 depicts the velocity profile around the hot sphere formed as a consequence of natural convection over it.

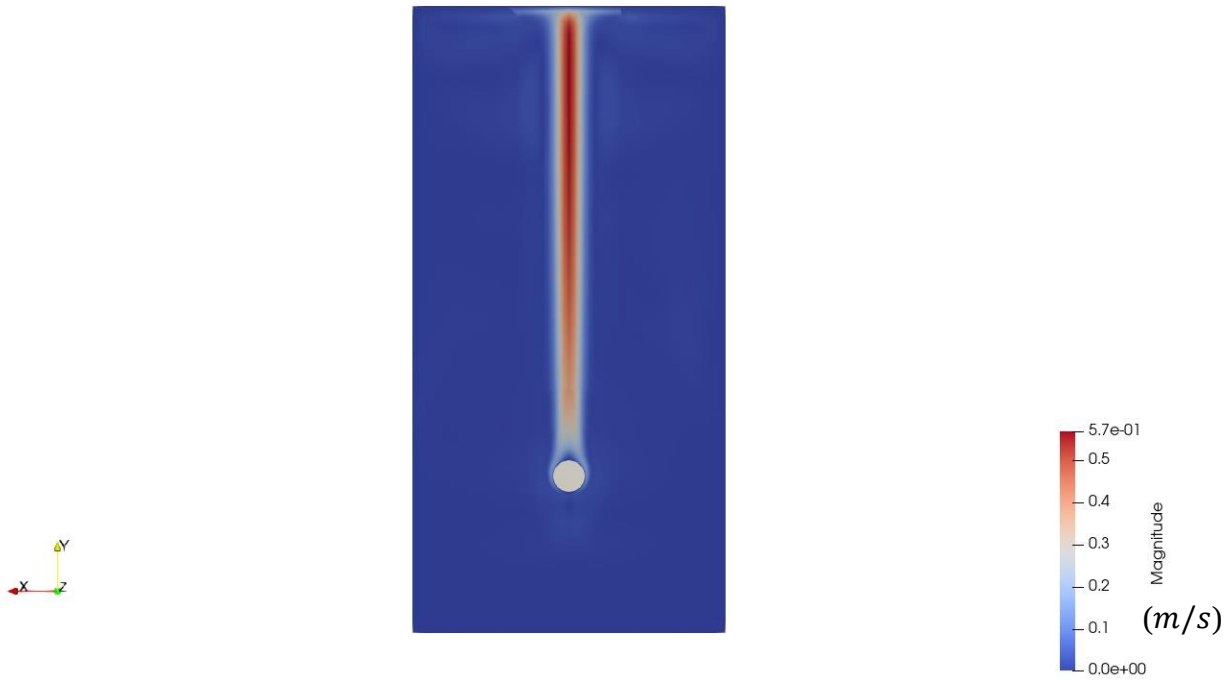

Fig. 11: Velocity profile around the hot sphere

In the case of sphere the width of the plume increases along the diameter of the sphere. From the Fig. 10 it is observed that the plume itself will eventually dissipate as a result of viscous effects and a reduction in the buoyancy force caused by cooling the fluid in the plume.

2.6 Validation

A validation exercise is carried out to demonstrate the accuracy of the CFD codes used for this simulation. Validation exercise determines if the computational simulation agrees with other established solution or measurements. For the simulation of conjugate heat transfer problem on bluff bodies the Nusselt number is calculated from a well-established correlation and is compared with that obtained from the CFD simulation.

2.6.1 Nusselt number calculation from numerical simulation

The Nusselt number is defined as the dimensionless temperature gradient at surface and is given as $Nu = \frac{hD}{K}$, here h is the heat transfer co-efficient or convection co-efficient, k is the thermal conductivity of the fluid (air) and D is the characteristic length (here the diameter of the sphere)[10]. In the OpenFOAM v1912 there is a special function called $surfaceFieldValue$ which calculates the average surface heat transfer coefficient in each iteration [5,11].

Heat transfer co-efficient obtained from CFD simulation using special function $surfaceFieldValue$ is h_{CFD} =12.32 W/m^2K

- Surface temperature of sphere at last iteration $T_{surface}$= 499.23 K
- The bulk temperature of air T_{bulk}= 300 K
- The thermo-physical properties are evaluated at the film temperature T_{film}

$$T_{film} = \frac{T_{bulk} + T_{surface}}{2} = 399.61 \text{ K}$$

- For the given film temperature the thermal conductivity k = 0.0337 W/m-k
- From the above relation for Nusselt number Nu_{CFD}= 10.97

2.6.2 Nusselt number calculation from correlation

For natural convection over a sphere, Churchill (1983) suggested the following correlation which gave the best fit with the available experimental data [10,12].

$$\overline{Nu} = 2 + \frac{0.589\, Ra_D^{\frac{1}{4}}}{\left[1 + \left(\frac{0.469}{pr}\right)^{\frac{9}{16}}\right]^{\frac{4}{9}}} \tag{2.7}$$

The correlation is valid for the $Pr \geq 0.7$ and the $Ra < 10^{11}$

Rayleigh number *Ra*, is a dimensionless number as defined [10,13].

$$\text{Rayleigh number } Ra = Gr \times Pr \tag{2.8}$$

where *Gr* is the Grashof number and *Pr* is the Prandtl number.

Prandtl number is defined as the ratio of momentum to thermal diffusivities [10]. Prandtl number is given as

$$Pr = \frac{\mu C_p}{K} \tag{2.9}$$

The Grashof number indicates the ratio of the buoyancy force to the viscous force acting on the fluid [10]. The Grashof number for bluff bodies calculated as

$$Gr = g\frac{\beta(T_s - T_f)D^3}{\nu^2} = g\frac{\beta(T_s - T_f)D^3}{\mu^2}\rho^2 \tag{2.10}$$

If density variations are due to only temperature variations, volumetric thermal expansion coefficient β can be calculated as[10]

$$\beta = -\frac{1}{\rho}\left(\frac{\partial p}{\partial T}\right)_p \tag{2.11}$$

Where all the thermos-physical properties are calculated at the film temperature (399.61 K) using the polynomial functions of respective property and are mentioned below.

$$\rho(T_{film}) = 0.8712665590\ kg/m^3$$

$$C_P(T_{film}) = 1013.8697224304\ \text{j/kg-k}$$

$$\mu(T_{film}) = 0.0000274108228093\ \text{kg/m}^3$$

$$k(\text{T}_{\text{film}}) = 0.033721\ \text{W/m-K}$$

From the polynomial function for density

$$\left(\frac{\partial \rho}{\partial T}\right)_p = -6.974\times 10^{-3} + 11.68\times 10^{-6}\text{T}$$

$$\text{Then } \left(\frac{\partial \rho}{\partial T}\right)_p = -0.00464\ kg/m^3$$

The Prandtl number(*Pr*) , Volumetric thermal expansion coefficient β, Grashof Number *Gr*, Rayleigh number *Ra* are calculated using respective equations stated above. The calculated values are given below.

Prandtl number $Pr = 0.824\ \text{m}^2\ \text{s}^{-1}$

$$\beta = 0.005326$$

Grashof Number $Gr = 141973.3$

Rayleigh number $Ra = 117007.3$

Nusselt number thus obtained from the correlation is

$$Nu_{correlation} = 2 + \frac{0.589\, Ra_D^{\frac{1}{4}}}{\left[1+\left(\frac{0.469}{pr}\right)^{\frac{9}{16}}\right]^{\frac{4}{9}}} = 10.54$$

2.6.3 Error estimation

Percentage error between Nusselt number Nu_{CFD} and $Nu_{correlation}$ is calculated as follows

$$\text{Percentage error} = \frac{Nu_{CFD} - Nu_{correlation}}{Nu_{CFD}} \times 100 = 3.91\ \%\ \text{error}$$

2.7 Closure

Natural convection over a heated sphere was solved as a conjugate heat transfer problem using *chtMultiRegionSimpleFoam* (OpenFOAMv1912) and the results were presented in this chapter. The percentage error in the Nusselt number calculated from the correlation and the Nusselt number obtained from the CFD simulation is about 3.91 % which is acceptable since the turbulent and the radiations effects are not considered. Results obtained from the computational simulation and correlations give a good match which shows that the physics of the problem under consideration were captured well by numerical simulations.

References

[1] Greenshields, C. J. "The OpenFOAM Foundation User Guide 7.0." *The OpenFOAM Foundation Ltd: London, United Kingdom, 10th July* (2019).

[2] https://openfoamwiki.net/index.php/How_to_add_temperature_to_icoFoam

[3] https://openfoamwiki.net/index.php/ScalarTransportFoam

[4] ANSYS Fluent User's Guide, 2019R1

[5] https://www.openfoam.com/documentation/user-guide/

[6] https://www.engineeringtoolbox.com/material-properties-t_24.html

[7] https://sites.google.com/site/snappywiki/snappyhexmesh

[8] https://cfd.direct/openfoam/user-guide/v6-thermophysical/

[9] https://www.thermalfluidscentral.org/

[10] Incropera, Frank P. "David P De witt, "." *Fundamentals of Heat & Mass Transfer", 5th Edition, John Wiley& Sons* (2007).

[11] https://cpp.openfoam.org/v6/classFoam_1_1functionObjects_1_1fieldValues_1_1surfaceFieldValue.html

[12] Churchill SW. Free convection around immersed bodies. Section 2.5.7, New York: Hemisphere, 1983. In: Schlender EU, editor. Chief Heat Exchanger Design Handbook.

[13] Kitamura, K., Mitsuishi, A., Suzuki, T., & Misumi, T. (2015). Fluid flow and heat transfer of high-Rayleigh-number natural convection around heated spheres. *International Journal of Heat and Mass Transfer*, *86*, 149-157.

Appendix

OpenFOAM Code for modified icoFoam solver to solve thermal transport

- **Make/files**

```
icoFoamT.C

EXE = $(FOAM_APPBIN)/icoFoamT
```

- **Make/Options**

```
EXE_INC = \
    -I$(LIB_SRC)/finiteVolume/lnInclude \
    -I$(LIB_SRC)/meshTools/lnInclude

EXE_LIBS = \
    -lfiniteVolume \
    -lmeshTools
```

- **createfiled.H**

```
    Info<< "Reading transportProperties\n" << endl;

    IOdictionary transportProperties
    (
        IOobject
        (
            "transportProperties",
            runTime.constant(),
            mesh,
            IOobject::MUST_READ_IF_MODIFIED,
            IOobject::NO_WRITE
        )
    );

    dimensionedScalar nu
    (
        "nu",
        dimViscosity,
        transportProperties.lookup("nu")
);

//Thermal diffusivity Alpha(α)

//*****************************************

    dimensionedScalar alpha
    (
        "alpha",
        dimensionSet(0,2,-1,0,0,0,0),
        transportProperties.lookup("alpha")
    );
//*****************************************
```

```
      Info<< "Reading field p\n" << endl;
      volScalarField p
      (
          IOobject
          (
              "p",
              runTime.timeName(),
              mesh,
              IOobject::MUST_READ,
              IOobject::AUTO_WRITE
          ),
          mesh
      );
//Temperature scalar
//*******************************
      volScalarField T
      (
          IOobject
          (
              "T",
              runTime.timeName(),
              mesh,
              IOobject::MUST_READ,
              IOobject::AUTO_WRITE
          ),
          mesh
      );

//*********************************
      Info<< "Reading field U\n" << endl;
      volVectorField U
      (
          IOobject
          (
              "U",
              runTime.timeName(),
              mesh,
              IOobject::MUST_READ,
              IOobject::AUTO_WRITE
          ),
          mesh
      );


      #include "createPhi.H"


      label pRefCell = 0;
      scalar pRefValue = 0.0;
      setRefCell(p, mesh.solutionDict().subDict("PISO"), pRefCell, pRefValue);
      mesh.setFluxRequired(p.name());
```

- **Application**

```
#include "fvCFD.H"
#include "pisoControl.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//

int main(int argc, char *argv[])
{
    #include "setRootCaseLists.H"
    #include "createTime.H"
    #include "createMesh.H"

    pisoControl piso(mesh);

    #include "createFields.H"
    #include "initContinuityErrs.H"

    // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//

    Info<< "\nStarting time loop\n" << endl;

    while (runTime.loop())
    {
        Info<< "Time = " << runTime.timeName() << nl << endl;

        #include "CourantNo.H"

        // Momentum predictor

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

        if (piso.momentumPredictor())
        {
            solve(UEqn == -fvc::grad(p));
        }
```

```
    // --- PISO loop
        while (piso.correct())
        {
            volScalarField rAU(1.0/UEqn.A());
            volVectorField HbyA(constrainHbyA(rAU*UEqn.H(), U, p));
            surfaceScalarField phiHbyA
            (
                "phiHbyA",
                fvc::flux(HbyA)
              + fvc::interpolate(rAU)*fvc::ddtCorr(U, phi)
            );
            adjustPhi(phiHbyA, U, p);

            // Update the pressure BCs to ensure flux consistency
            constrainPressure(p, U, phiHbyA, rAU);

            // Non-orthogonal pressure corrector loop
            while (piso.correctNonOrthogonal())
            {
                // Pressure corrector
                fvScalarMatrix pEqn
                (
                    fvm::laplacian(rAU, p) == fvc::div(phiHbyA)
                );

                pEqn.setReference(pRefCell, pRefValue);

                pEqn.solve();

                if (piso.finalNonOrthogonalIter())
                {
                    phi = phiHbyA - pEqn.flux();
                }
            }
#include "continuityErrs.H"

            U = HbyA - rAU*fvc::grad(p);
            U.correctBoundaryConditions();
        }
//Transport equation
//*****************************
        fvScalarMatrix TEqn
        (
                fvm::ddt(T)
                +fvm::div(phi,T)
                -fvm::laplacian(alpha,T)
        );
        TEqn.solve();
        runTime.write();
//*****************************

        Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
            << "  ClockTime = " << runTime.elapsedClockTime() << " s"
            << nl << endl;
    }

    Info<< "End\n" << endl;

    return 0;
}
```

- **fvSchemes**

```
ddtSchemes
{
    default         Euler;
    ddt(T)      Euler;
}
gradSchemes
{
    default         Gauss linear;
}
divSchemes
{
    default         none;
    div(phi,U)      Gauss limitedLinearV 1;
    div(phi,T)       Gauss linear;
}
laplacianSchemes
{
    default         Gauss linear corrected;
    laplacian(alpha,T)  Gauss linear corrected;//Laplacian term for α,T
}
    interpolationSchemes
{
    default         linear;
}
snGradSchemes
{
    default         corrected;
}
```

- **fvSolution**

```
solvers
{
    p
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0.05;
    }
    pFinal
    {
        $p;
        relTol          0;
    }
    U
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-05;
        relTol          0;
    }
    T
 {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-05;
        relTol          0;
    }
}
PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 2;
}
```