



Summer Internship

Indian Institute of Technology, IIT Bombay

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY IN MECHANICAL ENGINEERING

Submitted by:
Nihar Shah (19BME068)



**MECHANICAL ENGINEERING DEPARTMENT
INSTITUTE OF TECHNOLOGY
NIRMA UNIVERSITY
Year: 2020-21**

Declaration

This is certify to that

- The report comprises my learning during the summer internship at IIT Bombay towards the degree of Bachelor of Technology in Mechanical Engineering at Nirma University and has not been submitted elsewhere for degree.
- Due acknowledgement has been made in the text to all other materials used.

Sign: 

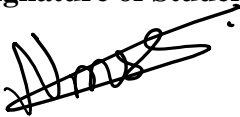
Name: Nihar Shah

Roll No: 19BME068

Undertaking for Originality of the Work

I, Nihar Shah (19BME068) give undertaking that the summer internship report submitted by me, towards the partial fulfillment of the requirements for the degree of Bachelor of Technology in Mechanical Engineering of Nirma University, Ahmedabad, is the original work carried out by me. I give assurance that no attempt of plagiarism has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

Signature of Student

A handwritten signature in black ink, appearing to read 'Nihar Shah', written over a horizontal line.

Date: 9th July, 2022

Place: Ahmedabad



भारतीय प्रौद्योगिकी संस्थान मुंबई
पवई, मुंबई-400 076, भारत
Indian Institute of Technology Bombay
Powai, Mumbai-400 076, India



दूरभाष/Phone (+91-22) 2572 2545
फैक्स/Fax (+91-22) 2572 3480
वेबसाइट/Website www.iitb.ac.in

IIT Bombay

Date: 30th June, 2022

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Mr. Nihar Mayur Shah**, B.Tech. Degree in Mechanical Engineering, student from Institute of Technology, Nirma University, Ahmedabad, has undertaken project work entitled : **Computational Fluid Dynamics : Application and Analysis** under the supervision of Prof. Atul Sharma, Department of Mechanical Engineering, Indian Institute of Technology, Bombay, during the Period from 01st June, 2022 to 30th June, 2022 and has successfully completed the same.

Date of issue: 30th June, 2022

Place: IIT Bombay



N. Lalwalekar
30/06/22
Offcy. Asst. Registrar
(Academic)

Certificate

TO WHOMSOEVER IT MAY CONCERN

This is to certify that, Mr. Nihar Shah student of B.Tech (Mechanical engineering), VIIIth Semester of, Institute of Technology, Nirma University has satisfactorily completed the project report.

Date:

Prof.
Mentor, Assistant Professor/Associate Professor/Professor,
Department of Mechanical Engineering,
Institute of Technology
Nirma University, Ahmedabad

Industry Trainer/Supervisor: Prof. Atul Sharma
Designation: Professor
Industry Name: Indian Institute of Technology, Bombay

Dr. K M Patel,
Head and Professor,
Department of Mechanical Engineering,
Institute of Technology
Nirma University, Ahmedabad

Approval Sheet

The Report by **Nihar Shah (19BME068)** is approved for the degree of Bachelor of Technology in Mechanical Engineering

Examiner(s)

Date: _____

Place: _____

Abstract

Computational Fluid Dynamics (CFD) is a branch of Fluid Mechanics which deals with Computational Analysis of Fluid. Some of the common examples are flow over a plate, flow over a circular cylinder. In this summer Internship I got a chance for learning about applications and analysis using CFD. The problem primarily performed in this internship is on Flow over a heated square cylinder. Two cases have been considered 1st when the cylinder was maintained at a constant Temperature and 2nd case when the cylinder was maintained with constant heat flux through Cylinder walls. A non-dimensional grid independent study has been carried out for the same problems, along with simulations. This work is based on Thesis by Professor Atul Sharma IIT Bombay under their guidance and with their permission. The software used in the making is OpenFOAM v2112 and paraview for creating simulations. The graphs were made using Tecplot360x. This analysis has not only been done computationally but also theoretically verified and verified through experiments. To verify simulations results with thesis graphs of Cl and C vs time were plotted and the error was calculated and found to be within 5% margin. There were further two cases one for steady periodic state and the second for transient state performed for both constant heat flux and constant wall temperature condition.

Table of contents

Declaration	I
Undertaking for Originality of the Report	II
Certificate	
Approval Sheet	
Abstract	
 CHAPTER 1	
INTRODUCTION	01 – 09
 CHAPTER 2	
DETAILS OF THE TRAINING	10 – 25
 CHAPTER 3	
SUMMARY AND CONCLUSION	26 – 30
 REFERENCES (If any)	56
 APPENDIX (If any)	57 – 80

Introduction

CHAPTER 1

Problem Statement: To simulate non dimensional grid independent flow over a heated square cylinder.(for $Re=100$) Horizontal and Vertical Flow Configuration. (with considering Forced and Natural convective heat transfer).

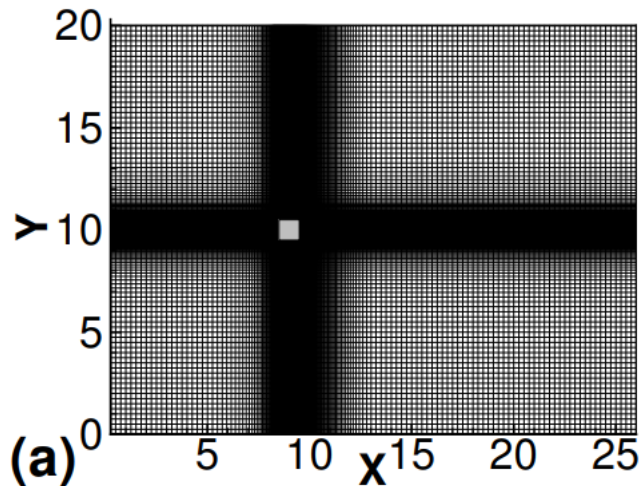
Case 1: Constant Wall temperature/isothermal boundary condition

- 1) Steady periodic state
- 2) Transient state

Case 2: Constant Heat Flux Boundary condition

- 1) Steady Periodic State
- 2) Transient State

Computational Domain: As given in thesis. 4 blocks with $100 * 100 * 1$ cells each. Hence total cells: 40000 with appropriate grading (finer near the walls and coarser away from walls) of 100.

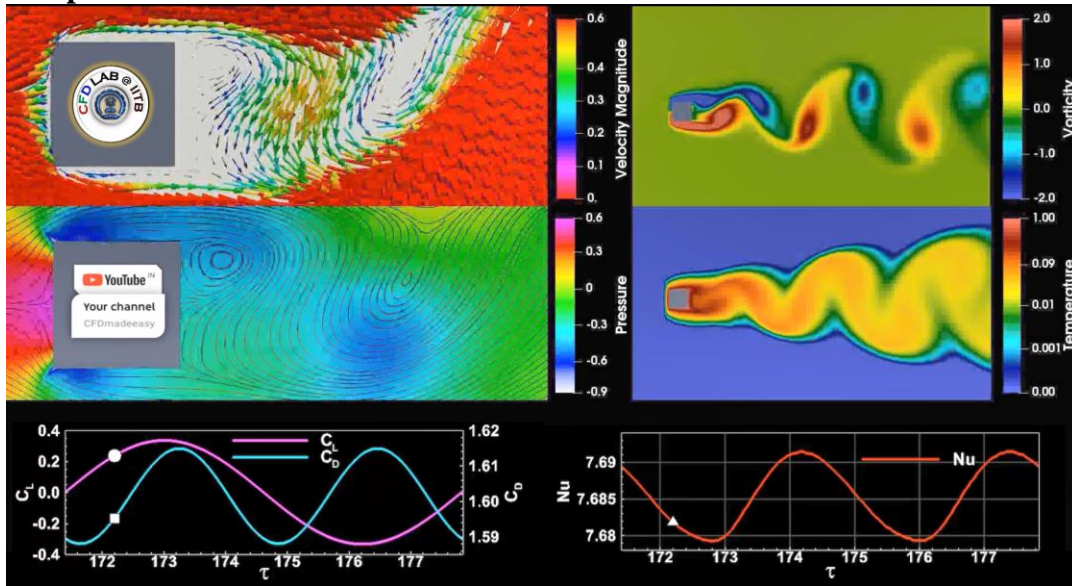


Software used:

- OpenFOAM v2112 – icoFOAM (modified with Temperature – transient Solver)
- Paraview – for all required postprocessing.
- Tecplot – for graphs

Reynolds number= 100 and hence from non dimensional analysis we get (assuming density =1, velocity =1 , $D=1$) hence dynamic viscosity =0.01.

Setup:



DETAILS OF THE TRAINING

CHAPTER 2

After applying all the required boundary conditions the simulation was done on a 4GB Ram i5 Intel Core workstation using parallel threading. The simulation took 36 hours + 6 hours for reconstructing.

Initial/Boundary Conditions:

PATCH	CYLINDER WALLS	INLET	OUTLET	TOP/BOTTOM
PRESSURE	ZERO GRADIENT	ZERO GRADIENT	FIXED VALUE ZERO	SYMMETRY PLANE
VELOCITY	NO SLIP	UNIFORM (1 0 0)	ZERO GRADIENT	SYMMETRY PLANE
TEMPERATURE	UNIFORM 1	UNIFORM 0	ZERO GRADIENT	SYMMETRY PLANE

PROCEDURE:

icoFOAM is a transient solver but does not have temperature as input. Hence we were required to integrate Temperature and hence build a new Solver named my_icoFOAM. The steps to do so are shown below.

1 HOW TO ADD TEMPERATURE TRANSPORT TO ICOFOAM

This HOWTO will cover rudimentary methods for altering an existing solver (icoFoam) to solve thermal transport. The main items to be accomplished are first, to copy and test that your installation of OpenFOAM can compile the existing solver correctly. Once that is accomplished, various small changes are necessary to the solver files themselves and these

will be detailed below. Finally, new fields have to be added to the initial and boundary conditions and alterations to the fvSchemes file.

2 COPY AND RECOMPILE ICOFOAM

This will go through the steps of creating a personal version of icoFoam in the user's subdirectory. The first step is to insure that your installation of OpenFOAM works properly and compiles the unedited solver. First, bring up a console and move to your OpenFOAM installation folder.

```
cd OpenFOAM
```

Your particular OpenFOAM installation folder will have a version number following it such as:

```
cd OpenFOAM-1.7.1
```

OpenFOAM has organized the solvers separate from the source code of OpenFOAM calling them 'applications'. Inside the 'applications' folder, there are subdirectories including one for solvers.

```
cd applications/solvers/
```

A shortcut you can use is stored in the environment variable FOAM_SOLVERS:

```
cd $FOAM_SOLVERS
```

Have a look around, but this tutorial is based on icoFoam, which we will copy to our own location:

```
cd incompressible
```

If you don't have a solver directory in \$WM_PROJECT_USER_DIR do

```
mkdir -p $WM_PROJECT_USER_DIR/applications/solvers  
cp -r icoFoam $WM_PROJECT_USER_DIR/applications/solvers/my_icoFoam  
cd $WM_PROJECT_USER_DIR/applications/solvers/my_icoFoam
```

Now, a couple of alterations need to be made to the make files in order for everything to compile and not overwrite the original solver. First, rename the primary file to your new solver name and delete the old dependency file:

```
mv icoFoam.C my_icoFoam.C
rm icoFoam.dep
```

Now go into the Make subdirectory and open the 'files' file with your favorite editor. Change it to read:

```
my_icoFoam.C

EXE = $(FOAM_USER_APPBIN)/my_icoFoam
```

No changes are necessary for the 'options' file. Delete the old binaries subdirectory:

```
rm -rf linuxGccDP0pt
cd ..
```

Now, test that the renamed solver (and your installation of OpenFOAM) works:

```
wmake
```

If everything worked correctly, your new solver binary should appear in the FOAM_USER_APPBIN directory. Check this with:

```
ls $FOAM_USER_APPBIN
```

3 ADDING THE TEMPERATURE FIELD

This will show the steps to add another field variable to a solver. Open the my_icoFoam.C (or whatever you have named it) with your favorite text editor.

First, edit the 'Application' to reflect the new name.

Following the flow of the program, one notices that the header file 'createField.H' is called prior to the solution loop. This file was copied with the solver and has the specific information pertaining to what variables will be solved.

Open createFields.H in your favorite editor.

The first items loaded is the kinematic viscosity from the transportProperties dictionary file. We will add a new transport property related to the thermal diffusion which will be denoted as DT. Make the following edits:

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

```
IOdictionary transportProperties
```

```
(  
    IObject  
    (  
        "transportProperties",  
        runTime.constant(),  
        mesh,  
        IOobject::MUST_READ,  
        IOobject::NO_WRITE  
    )  
);
```

```
dimensionedScalar nu
```

```
(  
    transportProperties.lookup("nu")  
);
```

```
//Add here...
```

```
dimensionedScalar DT
```

```
(  
    transportProperties.lookup("DT")  
);
```

```
//Done for now...
```

Aside: What does this code do and how do we learn about it?

Later on, we will need to edit the dictionary file to reflect this change.

Following this in the file there are lines which pertain to the creation of the pressure (p) and velocity (U) fields. We will add a new field for temperature (T). The fastest way to do this is to copy and paste the pressure lines and then edit them appropriately like so:

```
Info<< "Reading field T\n" <<endl;
```

```
volScalarField T
```

```
(  
    IObject
```

```
(
    "T",
    runTime.timeName(),
    mesh,
    IOobject::MUST_READ,
    IOobject::AUTO_WRITE
),
mesh
);
```

Save these changes. You've completed adding a new field variable, which will be called 'T' to the solver.

4 ADDING A NEW EQUATION TO SOLVE

The next step is to add a new equation describing the transport of the temperature. Return to editing the my_icoFoam.C file (or whatever you named it).

Because the temperature transport depends on the velocity field, we will add the equation after the momentum equation is solved (after the PISO loop), but before the time step is written. Edit your file so it looks like this:

```
#           include "continuityErrs.H"

           U -= rUA*fvc::grad(p);
           U.correctBoundaryConditions();
       }

//add these lines...
       fvScalarMatrix TEqn
       (
           fvm::ddt(T)
           + fvm::div(phi, T)
           - fvm::laplacian(DT, T)
       );

       TEqn.solve();
//done adding lines...
```

```
runTime.write();
```

These lines add a new equation for the temperature and make use of the face flux variable, phi, which is already used in the momentum equation solution.

Save your changes and run wmake:

```
wmake
```

Your computer should then produce a newly compiled binary.

5 ADD A NEW FILE FOR INITIAL AND BOUNDARY CONDITIONS

The next step in this process is to test the new solver. This will first be accomplished by modifying the existing cavity tutorial. Later, we will benchmark the solver against an analytical solution.

First, copy the cavity tutorial files to a new folder:

```
cd $FOAM_RUN/tutorials/incompressible/icoFoam
cp -r cavity $FOAM_RUN/my_icoFoam_cavity
cd $FOAM_RUN/my_icoFoam_cavity
```

We will start by editing the transportProperties dictionary:

```
cd constant
```

Open the transportProperties dictionary in your favorite editor and add the following line under the definition of nu:

```
DT          DT [0 2 -1 0 0 0 0] 0.002;
```

The value doesn't really matter at this point.

Next, we will create a new initial and boundary conditions file for the temperature field:

```
cd ../0
cp p T
```

Open the 'T' file in your favorite editor.

For this initial test, we will simply make the moving wall and the fixed walls different temperatures. Edit the file so it looks like the following:

```
class          volScalarField;
object         T;
}
//*****//

dimensions      [0 0 0 1 0 0 0];

internalField    uniform 300;

boundaryField
{
    movingWall
    {
        type      fixedValue;
        value      uniform 350;
    }

    fixedWalls
    {
        type      fixedValue;
        value      uniform 300;
    }

    frontAndBack
    {
        type      empty;
    }
}
```

Save those changes. There is one more change to be made before we can run the solver.

6 WHAT TO ADD IN FVSCHEMES AND FVSOLUTION

By adding a new equation to solve, we need to tell OpenFOAM what discretization schemes to apply to the equations. This is done in the fvSchemes dictionary. Enter the 'system' subdirectory for this case:

```
cd ../system
```

Open the 'fvSchemes' dictionary with your favorite editor.

Now, there are two main items added in the thermal transport equation above: a divergence term and a laplacian. Under each of these sections in the dictionary, we need to add terms which match what was added in the solver source code. Edit your file so it includes the following:

```
divSchemes
{
    default          none;
    div(phi,U)       Gauss linear; //NOTICE: there is no space between the comma and the
variables
    div(phi,T)       Gauss upwind;
}

laplacianSchemes
{
    default          none;
    laplacian(nu,U)   Gauss linear corrected;
    laplacian((1|A(U)),p) Gauss linear corrected;
    laplacian(DT,T)   Gauss linear corrected;
}
```

Alternatively, a scheme can be added to the 'default' field instead of adding specific ones below.

Save these changes.

Next, open the fvSolution dictionary in your favorite editor. It should look like this:

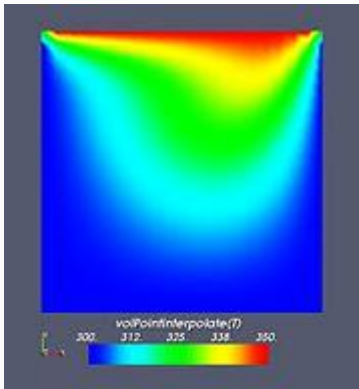
```
solvers
{
```

```

p PCG
{
    //information about the pressure solver
};
//add this...
T
{
    solver          BICCG;
    preconditioner  DILU;
    tolerance       1e-7;
    relTol          0;
};
//done editing...

```

Leave the velocity information intact and save these changes.



Temperature solution at 0.5s for the test case described.

Review the controlDict file to remind yourself what it looks like. You are now ready to run the case and test your solver.

```

cd ..
my_icoFoam

```

Your case should run and display the usual information in the window.

The solution for time-step 0.5 should look like this:

[Media:CaseFile.tar.gz](#) --[Media:My_icoFoam_cavity_OF-1.7.0.tar.gz](#) case according OpenFoam-1.7.0

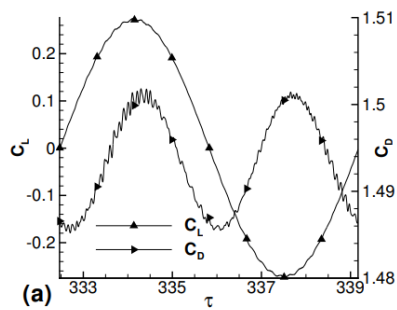
[Media:SolverSource.tar.gz](#) --[Mike Jaworski](#) 06:49, 25 February 2008 (CET)

7 BENCHMARKING YOUR NEW SOLVER

A case study of the Blasius flat-plate flow problem is shown using this solver on another wiki-page. [Blasius Flat-Plate Flow Benchmark](#)

Validation:

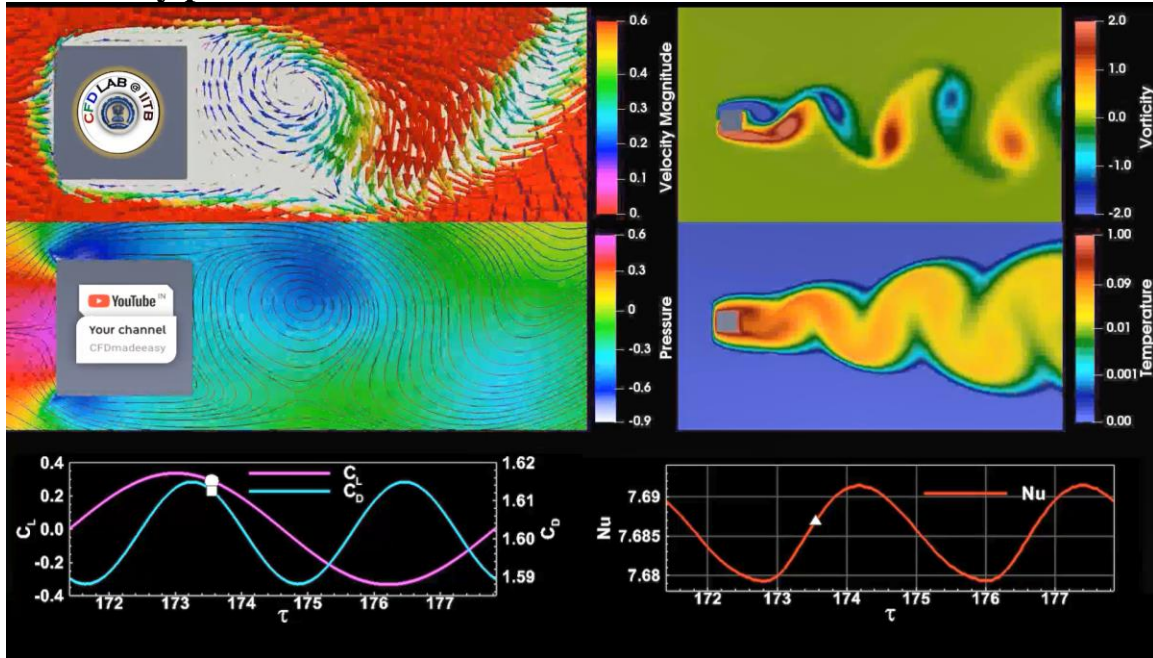
After the simulation the verification has been done by plotting c_l c_d vs time and checking the Strouhal number for results. The Strouhal number had error less than 5% and hence the results were validated.



SUMMARY AND CONCLUSION

CHAPTER 3

For steady periodic state:



The result is as shown above built using paraview.

Topmost left: It shows velocity vector plot with magnitude being represented by color and arrow size and the direction of flow as pointed by arrows.

Middle left: It is a pressure contour plot with magnitude represented by color map and streamlines shown in black.

Topmost Right: It shows vorticity contour with red representing anticlockwise and blue representing clockwise.

Middle Right: It shows Temperature contour with magnitude represented in logarithmic color map.

The two graphs represent how C_l , C_d and Nu varies with time shown by white dot changing its position with time.

The simulation has been done for one periodic cycle of C_l which has two cycles of C_d and Nu as shown above.

Similar Simulations have been performed for Transient state where the flow develops and has vortex shedding till the time when the system become steady periodic.