

FLUID FLOW ANALYSIS OVER A SYMMETRIC AIRFOIL PITCHING AT LOW FREQUENCIES.

Project report submitted to



Visvesvaraya Technological University, Belagavi

in partial fulfillment of requirements of VIII semester Project work (Phase II)- **16ME8DCPW2**

of

BACHELOR OF ENGINEERING in MECHANICAL ENGINEERING

Submitted by-

AJIT.H.A 1BM16ME026

ROHAN.S 1BM16ME124

VISHAL RAJ 1BM16ME186

AASHA.G.C 1BM16ME194

Under the guidance of:

Internal Guide:

Mr. RAMESH.M.CHALKAPURE, Assistant Professor, BMS College of Engineering.

External Guide:

Dr.K.RAMESH, CFD Lab, Department of Aerospace Engineering, IISc.



Department of Mechanical Engineering (Accredited by NBA, under Tier 1, 2014-2019)

BMS COLLEGE OF ENGINEERING

(Autonomous Institution Affiliated to Visvesvaraya Technological University, Belgavi) PB 1908,

Bull Temple Road, Bangalore-560 019

BMS COLLEGE OF ENGINEERING

(Autonomous Institution Affiliated to Visvesvaraya Technological University, Belgavi) PB 1908,
Bull Temple Road, Bangalore-560 019

Certificate

Certified that the project entitled : **Fluid Flow analysis over a symmetric airfoil pitching at low frequencies** is a bonafide work carried out

by

Submitted by-

A.JIT.H.A 1BM16ME026

ROHAN.S 1BM16ME124

VISHAL RAJ 1BM16ME186

AASHA.G.C 1BM16ME194

in partial fulfillment for the award of Bachelor of Engineering in Mechanical Engineering of the Visvesvaraya Technological University, Belgaum, during the year 2019– 20. It is certified that all corrections / suggestions indicated in the internal assessment will be incorporated in the report to be deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project Work Phase – II (16ME7DCPW2) prescribed for the said degree.

Signature of Internal Guide

Mr. Ramesh.M.Chalkapure

Signature of External Guide

Dr. K. Ramesh

Signature of HOD

Signature of Principal

Semester End Examination

Name of the Examiners

1-

2-

Signature with date



B.M.S. College of Engineering, Bull Temple Road, Bengaluru 560019
Autonomous College under VTU

CERTIFICATE FROM GUIDE

This is to certify that the project report titled, "**FLUID FLOW ANALYSIS OVER A SYMMETRIC AIRFOIL PITCHING AT LOW FREQUENCIES**" is a bonafide record of the project work done by

Ajit H A :1BM16ME026
Rohan S :1BM16ME124
Vishal Raj :1BM16ME186
Aasha G C :1BM16ME194

during the academic year 2019-2020 in Bachelor of Engineering in MECHANICAL ENGINEERING. This is certified that this project work is true work and was carried out under my supervision. Further, it is certified that I have gone through the complete project report. The students have incorporated all the correction and suggestions made by me.

Guide


Signature: 

Name: Ramesh M Chalkapure.

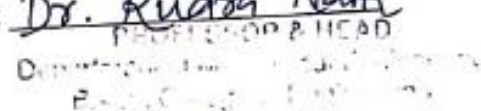
Seal:

Date: 27.07.2020

HOD

Signature: 

Name: Dr. Rudra Naik
PROFESSOR & HEAD

Seal: 

Date: 27.07.2020



Declaration

We hereby declare that the project work entitled : **Fluid Flow analysis over a symmetric airfoil pitching at low frequencies** has been independently carried out by us at Department of Mechanical Engineering, under the guidance of Mr. Ramesh.M.Chalkapure, Assistant Professor, Department of Mechanical Engineering, B. M. S. College of Engineering, Bengaluru, and Dr. K.Ramesh, CFD Lab, Department of Aerospace Engineering, IISC, in partial fulfilment of the requirements of the degree of Bachelor of Engineering in Mechanical Engineering of Visvesvaraya Technological University, Belgavi. We further declare that we have not submitted this report either in part or in full to any other university for the award of any degree.

Submitted by-

AJIT.H.A	1BM16ME026
ROHAN.S	1BM16ME124
VISHAL RAJ	1BM16ME186
AASHA.G.C	1BM16ME194

Place: Wind-Tunnel Lab, BMSCE.

Date: / /2020

Acknowledgement

We would like to express our sincere gratitude to our external guide Dr.K. Ramesh, IISc and our internal guide Mr. Ramesh.M. Chalkapure, Assistant Professor, Department of Mechanical Engineering, B. M. S. College of Engineering, Bengaluru for their invaluable guidance, comments and suggestions throughout the course of the project. Such suggestions ignited multiple thoughts and ideas that were essential in developing, understanding, and motivating us closer to the end goal.

We would like to thank the Project Co-ordinator Dr. R. N. Ravikumar, Assistant Professor, Department of Mechanical Engineering, B. M. S. College of Engineering, Bengaluru and Dr. Rudra Naik, Professor Head, Department of Mechanical Engineering, B. M. S. College of Engineering, Bengaluru for giving us an opportunity to work on the project.

We would also like to thank Dr. B. V. Ravishankar, Principal, B. M. S. College of Engineering, Bengaluru for his consistent support to our endeavors.

Contents

0.1	Abstract	2
0.2	Introduction	3
0.3	Literature Review	5
0.3.1	A Review on analysis in OpenFOAM	5
0.3.2	A Review of analysis in SU2	11
0.3.3	A Review on Experimental methods and Theoretical studies	15
0.4	Methodology: OpenFOAM	22
0.4.1	Static Analysis Case	22
0.4.2	Dynamic Analysis case	25
0.5	Methodology: SU2	26
0.5.1	Solver: Space Integration	26
0.5.2	Time Integration	27
0.5.3	Linear Solver and Preconditioner	28
0.5.4	Convergence Algorithm	29
0.5.5	Geometry and Mesh Movement	29
0.5.6	Simulation and Modeling Parameters	30
0.6	Methodology: Experimental Investigation	32
0.6.1	Manufacturing and Fabrication	32
0.6.2	Experiment	33
0.7	Results and Discussion	34
0.7.1	Results: OpenFOAM	34
0.7.2	Results: SU2	37
0.7.3	Plots on SU2 and OpenFOAM	40
0.7.4	Results: Experimental Investigation	42

0.7.5	Results: Dynamic Case Simulation	45
0.7.6	Flow Coefficients	46
0.7.7	Pressure Coefficient	46
0.8	Concluding Remarks	47
0.9	References	48

List of Figures

1	The stages of dynamic stall	4
2	A single block by BlockMesh	6
3	Vertices in BlockMesh	6
4	Boundary Conditions in OpenFOAM	8
5	Dynamic Mesh Solvers in OpenFOAM	10
6	Mach number-02 flow characteristics	11
7	The class structure for SU2_CFD	14
8	The pitching of an aircraft schematic diagram	16
9	The experimental setup	17
10	The flow pattern emerging for different pitching angles	18
11	Plots of coefficients of drag and pressure	19
12	Vortex Generation from a Pitching airfoil	21
13	Mean and Unsteady vortex patterns	21
14	The mesh of NACA-0012 airfoil	24
15	Boundary Conditions for the case.	24
16	Space Integration in SU2	26
17	Mesh from volumetric deformation	30
18	Convergence study using Tecplot for 8 degree AOA	35
19	Convergence study using Tecplot for 0 and 8 degree AOA	35
20	Convergence study using Tecplot for 0 degree AOA	36
21	Paraview simulations for static cases	36
22	Convergence of Cl and Cd at 0 degree AOA	37
23	Convergence of Cm at 0 degree AOA	38
24	Contours of Pressure and Velocity at -5 degree AOA	39

25	Convergence of C_l at -8 degree AOA	39
26	Convergence of C_m and C_d at -8 degree AOA	40
27	Drag and Lift coefficients vs Alpha	41
28	C_p vs x/c for upper surface	42
29	C_p vs x/c for lower surface	42
30	C_p vs x/c for upper surface	43
31	C_p vs x/c for lower surface	44
32	C_l vs x/c for lower surface	44
33	C_d vs x/c for lower surface	44
34	Pressure Contours at 9 degree AOA for the Pitching Simulation	45
35	Velocity Contours at 9 degree AOA	45
36	Pressure coefficient (C_p) vs x/c for Pitching simulation	47

0.1 Abstract

Most of the automobiles and aerospace vehicles maneuvering in fluids like airplanes, ships and submarines use components which deflect from the original position to produce certain desired change in the process of motion. For example rudder, flaps, ailerons etc to mention a few are used to get pitching, rolling and yawing moments. These components are designed and shaped aerodynamically to produce the necessary effects in the process of flight.

The pitching moment is one such action of a component and is the moment produced by the aerodynamic force on the component acting at the center of pressure. These components move relative to the air and flight motion, so the aerodynamic characteristics of these components change in time and space. In the present work, we analyze the pitching moment of airfoil to understand the time dependent aerodynamic characteristics, such as dynamic pressure.

We aim to study and analyze the pitching motion of a symmetric airfoil both experimentally and numerically. Numerical studies are carried out using two open source software. The experimental verification of static distribution of pressure is done in wind-tunnel lab for various angles of attack and flow visualization studies are carried out using smoke. Our aim is to determine the dynamic pressure distribution on the airfoil when pitching at low frequencies. This will be achieved by mounting piezoelectric pressure sensors on the airfoil and by measuring the pressure as a function of time. Once the pressure data is obtained the corresponding characteristics of C_p , C_d are plotted and compared to the numerical simulations carried out using CFD in OPENFOAM and SU2 Software.

The experimental results are obtained for static case at the wind-tunnel lab and the pitching mechanism setup incorporated with the Piezoelectric Pressure Transducers for the pitching mechanism is tested. The CFD simulations are carried out on oscillating cylinder, and on stationary NACA-0012 airfoil, for different angles of attack on both the software and currently working on the parameters for the pitching airfoil. The dynamic pitching mechanism is incorporated in the CFD and comparisons were made with experimental and theoretical results. The observed variations in the results between the two software are analyzed and reasoned out.

0.2 Introduction

Most of the autonomous and non-autonomous aerospace and under water vehicles like, submarines, flapping foil underwater vehicle and their components; like flaps, ailerons, propellers, undergo pitching (oscillatory motion about an axis perpendicular to the plane of the body) at very low frequency where the flow can be considered to be quasi-steady. The temporal variations at any spatial location for a quasi steady flow is marginally observable than that of steady flow, this perturbs the pressure along the body, giving rise to a varying pressure distribution.

The knowledge about variation is essential in accurately predicting and designing the performance of such bodies as they form a crucial part in the stability of the vehicle and are affected by the pitching motion.

Pitching motion can be of the form:

$$\alpha(t) = \alpha_o + \alpha_1 \sin \omega t \quad (1)$$

where α is the angle of attack in degrees varying with time, α_o is the mean angle of attack about which the airfoil will oscillate in degrees, α_1 is the amplitude of oscillation in degrees, ω is the frequency of oscillation in radians per second, and t is time in seconds.

In general, the underwater vehicles in shallow water regions move surrounded by low amplitude waves striking the vehicle at different angles. By changing the reference frame, we simulate the mechanism by pitching the vehicle (airfoil in this case) and allowing the flow to impinge at a particular fixed angle of attack. The characteristics of the dynamic pressure distribution in both the cases are similar, as only relative velocity is being changed, keeping other conditions constant.

The quasi-steady flow and the pressure distribution is analyzed using CFD in two open-source software OpenFOAM and SU2. These allow for dynamic mesh setup which is required for the current case-study. We initially simulate the flow over NACA-0012 airfoil under static, steady state conditions at different angles of attack, plotting the values of C_p , C_d , and C_l . These characteristic curves are overlapped and corresponded to the dynamic pressure distribution for the variation presented in pitching motion simulation. The results obtained from both the software are compared and the validation is provided by the experiment. The purpose for such a simulation is to compare algorithms, the behavior of solvers and analyse the differences in the solutions obtained from both the software, while the test-case utilized being airfoil pitching mechanism, validated further through the experiment.

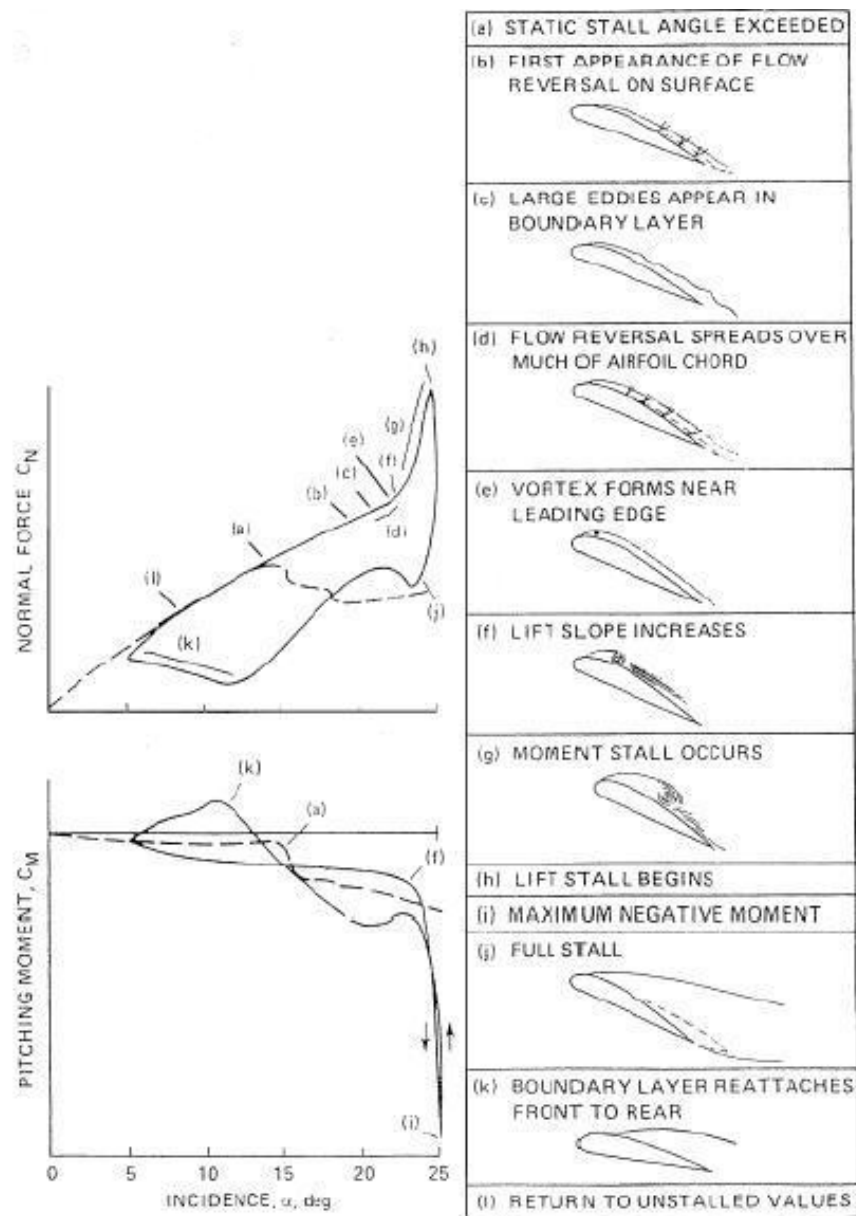


Figure 1: The stages of dynamic stall

0.3 Literature Review

0.3.1 A Review on analysis in OpenFOAM

OpenFOAM is an abbreviation of Open Field of Operation And Manipulation (FOAM). It is an open source CFD software package and the GNU public license provides complete freedom to contribute to any or all projects. It's based on C++ programming language with a set of libraries that solve any problem on generalized continuum mechanics. The solvers are generated based on cell-centred Finite Volume Method.

As in any CFD software, the process involved in simulation is based on three main steps:

- 1- Pre-processing
- 2- Running Simulation
- 3- Post-processing

The review by Park and Kang [1] suggested possible ways in which the above three steps are been carried out in OpenFOAM.

Pre-Processing

Mesh generation is one of the starting points in the pre-processing step. The first step involves defining the geometry of the region of interest, OpenFOAM is primarily used for structured mesh, hence moderately complex geometries can be simulated with ease.

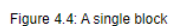
Next step involves grid generation- the computational domain sub-division of the flow region into a number of smaller, non-overlapping sub-domains: a grid (or mesh) of cells (or control volumes or elements). Although mesh could be imported from other software - either analysis software like Ansys (using the command- `fluentToMeshFoam`) or CAD modeling software like SolidWorks (using the file- `SnappyHexMesh`), OpenFOAM provides an inbuilt tool known as `BlockMesh` which is convenient for simple to moderately complex geometries.

The mesh can be found in the 'constant' directory \rightarrow *PolyMesh* folder \rightarrow *BlockMeshDict* file.

The `blockMeshDict` file is a dictionary using keywords as described in Figure [2]. The `scale` keyword specifies a scaling factor by which all vertex coordinates in the mesh description are multiplied.

scale 0.001;

means that all coordinates are multiplied by 0.001, i.e. the values quoted in the blockMeshDict file are in *mm*



Keyword	Description	Example/selection
scale	Scaling factor for the vertex coordinates	0.001 scales to mm
vertices	List of vertex coordinates	(0 0 0)
edges	Used to describe arc or spline edges	arc 1 4 (0.939 0.342 -0.5)
block	Ordered list of vertex labels and mesh size	hex (0 1 2 3 4 5 6 7) (10 10 1) simpleGrading (1.0 1.0 1.0)
patches	List of patches	symmetryPlane base ((0 1 2 3))

The vertices of the blocks of the mesh are given next as a standard list named vertices, e.g. for the example block in Figure, the vertices for example can be denoted as:

```
vertices
(
    ( 0 0 0 ) // vertex number 0
    ( 1 0 0.1 ) // vertex number 1
    ( 1.1 1 0.1 ) // vertex number 2
    ( 0 1 0.1 ) // vertex number 3
    ( -0.1 -0.1 1 ) // vertex number 4
    ( 1.3 0 1.2 ) // vertex number 5
    ( 1.4 1.1 1.3 ) // vertex number 6
    ( 0 1 1.1 ) // vertex number 7
);
```

Figure 3: Vertices in BlockMesh

Edges in blockMesh can be coded based on the shape and structural outlook of the geometry. Each edge joining 2 vertex points is assumed to be straight line. However any edge may be specified to be curved by entries in a list named edges. The list is optional; if the geometry contains no curved edges, it may be omitted.

A set of edges collectively form blocks. The block definitions are contained in a list named blocks. Each block definition is a compound entry consisting of a list of vertex labels, a vector giving the number of cells required in each direction, the type and list of cell expansion ratio in each direction. The boundary of the mesh is given in a list named *boundary*. The boundary is broken into patches (regions), where each patch in the list has its name as the keyword, which is the choice of the user. The patch information is then contained in sub-dictionary with:

type: the patch type, either a generic patch on which some boundary conditions are applied or a particular geometric condition, as listed in the figure;

faces: a list of block faces that make up the patch and whose name is the choice of the user, like patch faces.

The mesh for this case too is made using BlockMeshDict, with the *spline* function used to join vertices along arcs that is essential for a curved geometry as in case of an airfoil.

Boundary Conditions and Physical Properties

Incorrect boundary conditions will lead to physically incorrect predictions, and in many cases solver failure. We must specify the boundary conditions for each solved field and in OpenFOAM's 0 folder, files for the conditions are stored.

Thermophysical models are used to describe cases where the thermal energy, compressibility or mass transfer is important. OpenFOAM allows thermophysical properties to be constant, or functions of temperature, pressure and composition. Thermal energy can be described in form of enthalpy or internal energy. The $p - v - T$ relation can be described with various equations of state or as isobaric system.

The thermophysicalProperties dictionary is read by any solver that uses the thermophysical model library. A thermophysical model is constructed in OpenFOAM as a pressure-temperature $p - T$ system from which other properties are computed. But since the current simulation doesn't need thermophysical models, we restrict our path along this and focus on turbulence models and solvers.

```

basic
  basic types
    ■ fixedValue
    ■ fixedGradient
    ■ mixed
    ■ ...

constraint
  geometrical constraints
    ■ symmetry
    ■ wedge
    ■ empty
    ■ cyclic
    ■ ...

derived
  specialised conditions
    ■ fixedProfile: to specify a profile of a variable
    ■ swirlFlowRateInletVelocity: to specify velocity inlet for a swirling flow providing flow rate
    ■ inletOutlet: outlet condition with handling of reverse flow
    ■ codedFixedValue: fixed value set by user coding

```

Figure 4: Boundary Conditions in OpenFOAM

The turbulenceProperties dictionary is read by any solver that includes turbulence modelling. Within that file is the *simulationType* keyword that controls the type of turbulence modelling to be used, either:

laminar uses no turbulence models;

RAS uses Reynolds-averaged stress (RAS) modelling;

LES uses large-eddy simulation (LES) or detached-eddy simulation (DES) modelling. Currently, the flow over a stationary airfoil uses Omega-SST turbulence modeling under RAS, just like the pitching mechanism.

Simulation: Numerical Solvers and Schemes

: Numerical schemes are controlled by the *fvSchemes* file in the system directory. An Euler scheme is used for time derivatives. The Euler scheme is a first-order explicit scheme and places a limit (Courant number) on how fast a given simulation may be run.

Schemes in OpenFOAM

The *fvSchemes* dictionary in the system directory sets the numerical schemes for terms, such as derivatives in equations, that are calculated during a simulation. The terms that must typically be assigned a numerical scheme in *fvSchemes* range from derivatives, e.g. gradient ∇ , to interpolations of values from one set of points to another. The aim in OpenFOAM is to allow choices like discretisation which is generally standard Gaussian finite volume integration. Gaussian integration is based on summing values on cell faces, which must be interpolated from cell centres. There are wide range of options for interpolation scheme, with certain schemes being specifically designed for particular derivative terms, especially the advection divergence *nabla·* terms.

The set of terms, for which numerical schemes must be specified, are subdivided within the *fvSchemes* dictionary into the categories below.

- 1- timeScheme: first and second time derivatives, e.g. $\frac{\partial}{\partial t}$
- 2- gradSchemes: gradient ∇
- 3- divSchemes: divergence $\nabla \cdot$
- 4- laplacianSchemes: Laplacian ∇^2
- 5- interpolationSchemes: cell to face interpolations of values.
- 6 - snGradSchemes: component of gradient normal to a cell face.
- 7 - wallDist: distance to wall calculation, where required.

A linear scheme is used for gradient terms and for interpolation. This linear interpolation scheme uses central differencing. A total-variation diminishing (TVD) scheme with a Sweby flux limiter. is used to calculate the divergence terms in the *kT*, *kL*, and *omega* equations. Flux limiters allow a higher-order scheme to be used where gradients are small and enforce first-order scheme where gradients are large; a switching function (the flux limiter) dictates when to use each.

This allows a solution to have the accuracy of a higher-order numerical scheme while maintaining the benefit of zero overshoot that first-order schemes provide. A linear upwind scheme is used for the divergence terms in the momentum equation. This scheme is similar to the linear scheme previously described, but differs since it uses upwind differencing.

Solvers in OpenFOAM

Equation solvers are controlled by the fvSolution file in the system directory. A multi-grid solver with a Gauss-Seidel smoother is used for the pressure and cell displacement equation in the pitching airfoil mechanism.

The solvers used in the simulations of our interest are: PISOFOAM for static, steady flow analysis and PimpleDymFOAM for pitching airfoil simulation.

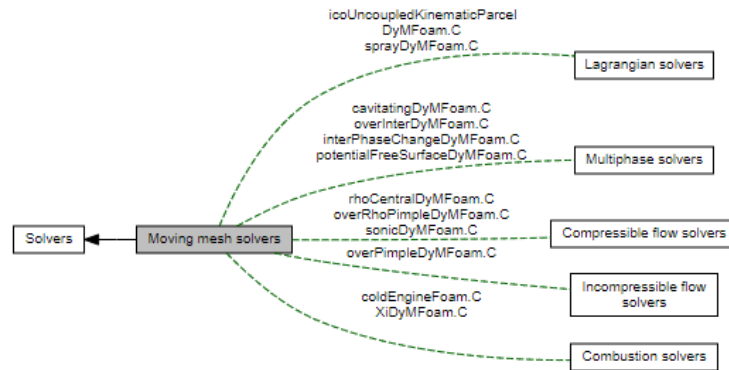


Figure 5: Dynamic Mesh Solvers in OpenFOAM

The simulation for pitching mechanism was done by various researchers [5], [6], the work by Ridley, and Vlcek [6] focus on pitching mechanism of NACA-0015 on OpenFOAM.

The simulation time was for 5 periods of vibration (256.5 ms), solutions was compared for fifth period and for average for all periods. The movement of the wing in numerical solution is prescribed according to the measured data. For comparison of the experimental data with the numerical solution it was necessary to choose the points at which the calculated data are compared with experimental data.

Points were to be close to wing surface, because evaluation experiment is made using by interferographic method on the surface of the wing. For the numerical solution is prescribed zero flow velocity on the wing. For this reason mesh nodes in the first and second layer were chosen. The experimental data is compared against the fifth period of vibration and against an average over all five periods.

The results obtained showed significantly lower Mach numbers near the stagnation zone, which they believe is due to poor reflection of the interferography method.

The results further showed that the top surface of wing the numerical results match the experimen-

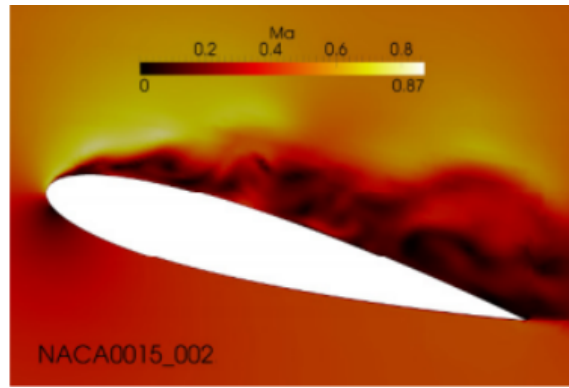


Figure 6: Mach number-02 flow characteristics

tal data much worse. This were deduced to have been caused by two effects: first, the flow patterns in the separated region are very complex and the interferographic images and the subsequent evaluation procedure might not capture the separated turbulent boundary layer and the real velocity field with sufficient resolution.

Second, the current simulation is run without a turbulence model and with insufficiently fine mesh in the boundary layer.

Post-Processing: Paraview

Post-processing is done on the in-built software called Paraview. It is advantageous due to its innate multi-functional features such as contour plots, global line plots, color coding/schemes. Visualizing of the geometry, mesh, simulation and plots will be carried out in the said software.

0.3.2 A Review of analysis in SU2

The Stanford University Unstructured (SU2) is a computational analysis and design suite for performing Partial Differential Equation (PDE) analysis to solve multiphysics analysis and optimization tasks using unstructured mesh topologies . SU2 is an open-source collection of C++ software tools to discretize and solve problems put together by python scripts. The core of the tool is a Reynolds-averaged Navier-Stokes (RANS) solver capable of simulating simple to complex multiphysics engineering systems by providing gradient information. Finite Volume Method (FVM) and Finite Element Method (FEM) solvers, their corresponding adjoint systems, and, combination of both are available in SU2.

This study was based on the review by Palacios, Economon, Lonkar, and Alonso [2], along with papers by Copeland, and Likaczyk [3].

Governing equations and solvers in SU2

The solvers within SU2 enables us to solve cases like Compressible Navier-Stokes, Incompressible Navier-Stokes, Compressible Euler, Incompressible Euler, Turbulence Modeling, Elasticity and Heat Conduction equations. We are interested in the Incompressible Euler and Navier-Stokes equations for the problem definition. The governing equation in SU2 for in-compressible Navier Stokes is of the form:

$$\partial_t V + \nabla \cdot \bar{F}^c(V) - \nabla \cdot \bar{F}^v(V, \nabla V) = Q \quad (2)$$

in the domain $\Omega, t < 0$.

$V = \{p, \bar{v}, T\}^T$, is the vector of working variables within the solver, $\bar{F}^c(V) = \{\rho \bar{v}, \rho \bar{v} \otimes \bar{v} + \bar{\bar{I}}p, \rho c_p T\}^T$, is the vector of convective fluxes, $\bar{F}^v(V) = \{\cdot, \bar{\bar{\tau}}, \kappa \nabla T\}^T$, is the vector of viscous fluxes, and Q is the source term.

Pressure-velocity coupling is obtained by artificial compressibility developed by Chorin and the incompressible equations are solved in a fully coupled manner. The constant density model is chosen for the problem and hence constant density, as given by the user, is used for calculation throughout.

The incompressible solver in SU2 supports the following boundary condition types:

- 1- Euler (flow tangency) wall,
- 2- no-slip wall (adiabatic), far-field and symmetry boundaries, inlet boundaries (total conditions or mass flow prescribed),
- and
- 3- outlet boundaries (back pressure prescribed).

Turbulence Modeling in SU2

As per the Boussinesq hypothesis, viscosity(μ) is dissolved into dynamic(μ_d) and turbulent (μ_t) components as $\mu = \mu_d + \mu_t$. μ_d is chosen constant for the current case as it is constant temperature flow analysis and μ_t is computed using the Menter Shear Stress Transport (SST) Model. SU2 also offers a one-equation SA model; however, previous studies (like villalpando et al, 2011) have conclusively

shown SST to be a better suited turbulence model than SA model. The Menter SST turbulence model is a two equation eddy-viscosity turbulence model that combines $k - \omega$ and $k - \epsilon$ turbulence models which overcomes the sensitivity of $k - \omega$ model to the values of ω in the free stream while having better boundary layer modeling capability.

Pre-processing in SU2

SU2 handles pre-processing with the help of three input files:

1. **Configuration file:** It is a text file with a .cfg extension (optional) containing user's inputs for a particular problem. It has three elements: *Options*: consists of the desired options and their values in the scalar or array data type or even a complicated structure. Syntax: $option_{name} = value$

Comments: any text appearing after % is considered a comment and the % sign should be typed for each new line.

White spaces: empty lines are insignificant in SU2 Upon execution of C++ modules, SU2 checks and throws errors for unknown options specified, duplicated options, syntax error, etc.

2. **Mesh file:** SU2 supports native mesh file format with an extension of .su2 and has converter for CGNS data format for when imported from another software. The native files are in a readable ASCII format consisting the information about dimensionality of the problem, node location and connectivity, the type of element forming the volume and the nodes forming the elements in a specific structure. We have used GMSH open source software to import mesh in the native format.

'NDIME=' , 'NELEM=' and "NPOIN=" are the first three keywords in the mesh file which will be read by SU2 to set the dimension and store total number of elements and total number of nodes respectively. Finally, markers are set for boundary conditions, specified by 'NMARK-TAG=' string. The two markers used for this analysis are farfield and airfoil.

3. **Restart file:** By default, SU2 generates solution binary format files with .dat extension. Restart files are used to run the adjoint simulations and restart the code from a previous solution. The adjoint simulations are carried out by setting the restart _ flow.dat to solution_flow.dat.

In the configuration RESTART-SOL flag needs to be set to YES to perform restart simulation and UNST_RESTART_ITER to the required iteration number which need to be restarted from for performing unsteady restart. We intend to use dual time stepping for unsteady setting for the defined problem.

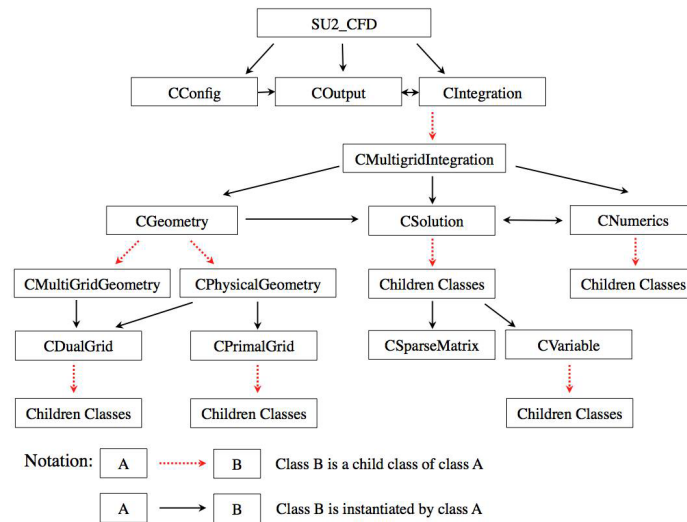


Figure 7: The class structure for SU2_CFD

Execution in SU2

After specifying the problem parameters in the configuration file, C++ programs can be executed. SU2 currently has several C++ executables namely; SU2_CFD, SU2_DOT, SU2_DEF, SU2_MSH, SU2_SOL, SU2_GEO in \$SU2_HOME/<MODULE_NAME>/bin directory which can be coupled to utilize advanced features like shape optimization or adaptive mesh refinement among others.

These modules can be executed individually or coupled using python scripts. We have exploited SU2_CFD which is capable of solving direct, adjoint, and linearized problems for the Euler, Navier-Stokes, and Reynolds-Averaged Navier-Stokes (RANS) equation sets, etc using a Finite Volume Method (FVM), and an edge-based structure. Explicit and implicit time integration methods are available with centered or upwinding spatial integration schemes. Advanced features to improve robustness and convergence, including residual smoothing, preconditioners, and agglomeration multigrid are available in this module. Other advanced features will be implemented as per the need with the progress of the project. The class structure for SU2_CFD is as shown:

Numerical algorithms of SU2

The main numerical algorithms of SU2 for fluid dynamics solver are: Space Integration, Time Integration, Convergence acceleration technique, Mesh manipulation, Adjoint formulation and sensitivity computation, Goal-oriented grid adaptation, and Design variable definition.

SU2 discretize PDEs using finite volume method with standard edge based structure on dual grid

with control volumes created using a medial-dual, vertex-based schemes. The convective and viscous fluxes are evaluated at the midpoint of an edge and the numerical solver loops through all of the edges in the primal mesh in order to calculate these fluxes and then integrates them to evaluate the residual at every node in the numerical grid.

Time integration can be carried out using Euler and Runge-Kutta methods with explicit and implicit schemes using local time stepping technology for steady simulations and dual time stepping and for unsteady simulations.

Post-Processing in SU2

SU2 is capable of exporting the solution information for post-processing of results, visualization, and a restart in ParaView (.vtk), Tecplot (.dat for ASCII, .plt for binary), CGNS (.cgns), comma-separated values (.csv), or STL (.stl) depending on the instructions given in the configuration file and/or the type of problem.

0.3.3 A Review on Experimental methods and Theoretical studies

Theoretical framework

In aerodynamics, the pitching moment on an airfoil is the moment produced by the aerodynamic force on the airfoil if that aerodynamic force is considered to be applied, not at the center of pressure, but at the aerodynamic center of the airfoil. The pitching moment on the wing of an airplane is part of the total moment that must be balanced using the lift on the horizontal stabilizer. More generally, a pitching moment is any moment acting on the pitch axis of a moving body.

The lift on an airfoil is a distributed force that can be said to act at a point called the center of pressure. However, as angle of attack changes on an airfoil, there is movement of the center of pressure forward and aft. This makes analysis difficult when attempting to use the concept of the center of pressure. One of the remarkable properties of an airfoil is that, even though the center of pressure moves forward and aft, if the lift is imagined to act at a point called the aerodynamic center the moment of the lift force changes in proportion to the square of the airspeed. If the moment is divided by the dynamic pressure, the area and chord of the airfoil, the result is known as the pitching moment coefficient.

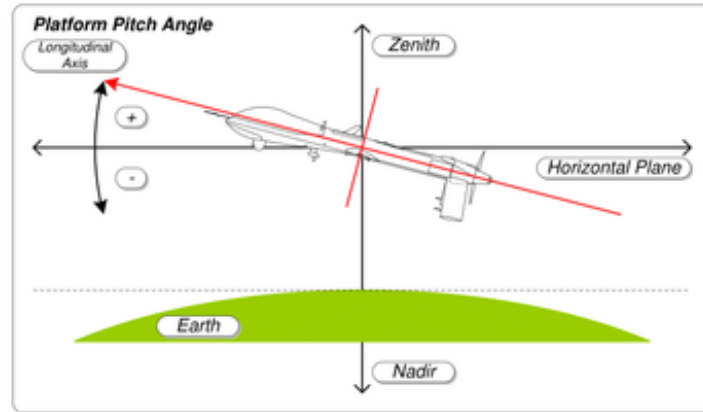


Figure 8: The pitching of an aircraft schematic diagram

This coefficient changes only a little over the operating range of angle of attack of the airfoil. The combination of the two concepts of aerodynamic center and pitching moment coefficient make it relatively simple to analyse some of the flight characteristics of an aircraft.

The aerodynamic forces on an airfoil section may be represented by lift, drag, and pitching moment. At each value of the lift coefficient there will be one particular point about which the pitching moment coefficient is zero, and the aerodynamic effects on the airfoil section may be represented by the lift and the drag alone acting at that point. This special point is termed the center of pressure. The aerodynamic center is a fixed point that always lies within the profile of a normal airfoil section, the center of pressure moves with change in lift coefficient and is not necessarily within the airfoil profile.

The paper by Dong-Ha Kim, Jo-won Chang [4] elaborated on the experimental setup and analysis for low pitching angles of an airfoil.

The effect of low Reynolds number ranging from 20000 to 50000 on the aerodynamic characteristics of a pitching NACA 0012 airfoil was observed.

Experimental Setup

Experiments were conducted in a low-speed wind tunnel, which is an open suction type tunnel with a test-section size of 0.5m (H) 0.5m (W) 1.4m(L). The cross-section of the airfoil model was a NACA0012 with a chord length of 0.18m.

The air foil was pitched at the quarter chord, and the instantaneous angle of attack was written as $\alpha(t)=0 +/ - 6$ degrees.

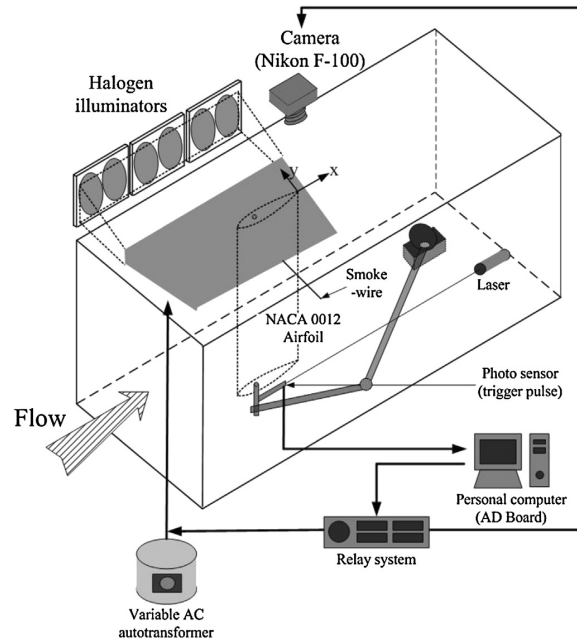


Figure 9: The experimental setup

Reverse flow is watched distinctly in the upstroke movement during a cycle, beginning from about $\alpha=3^\circ$, and the intriguing vortical structures are seen in the boundary layer and approach wake regions. At $\alpha=5^\circ$, the main trailing-edge vortex that turns in the clock-wise direction is framed in the region of the trailing edge. In this way, the subsequent trailing-edge vortex that rotates in counter clockwise direction is seen in the near wake region close to the trailing edge.

The area where the pressure is practically steady is commonly seen in the rear part of the airfoil over the positive angle of attack range. The start and end points of this zone are non-linearly varied with the Reynolds number, and this region is likewise extended with the increment in Reynolds number. That is, this zone starts at a lower phase angle and finishes at a higher phase angle as the Reynolds number expands on the grounds that this Reynolds number expanding advances the progress from laminar to turbulent inside the boundary layer.

The area where the pressure is almost constant is generally observed in the rear part of the airfoil over the positive angle-of-attack range. The starting and ending points of the area are non-linearly varied with the Reynolds number, and this area is also expanded with the increase in Reynolds number. That is, this area initiates at a lower phase angle and ends at a higher phase angle as the Reynolds number increases because this Reynolds number increasing promotes the transition from laminar to turbulent inside the boundary layer.

Through flow visualization, the clockwise rotation of the first trailing-edge vortex and the counter-

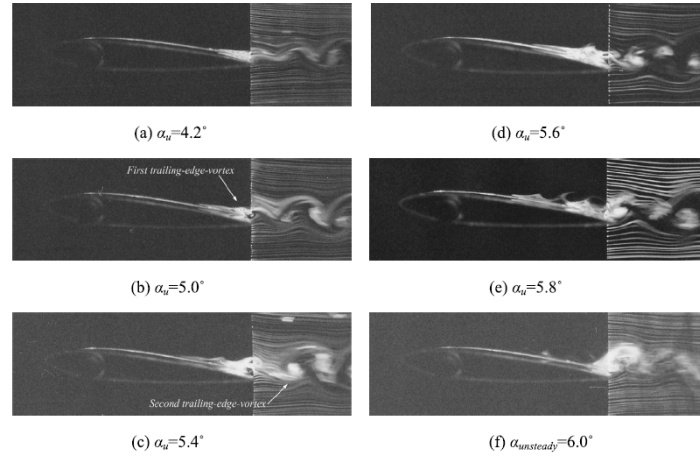


Figure 10: The flow pattern emerging for different pitching angles

clockwise rotation of the second trailing-edge vortex were observed. The interaction of the two vortices forms the mushroom vertical structure in the near-wake region. Lift and pressure drag coefficients, which are nonlinearly varied, were observed due to the nonlinear behaviour of unsteady boundary-layer events.

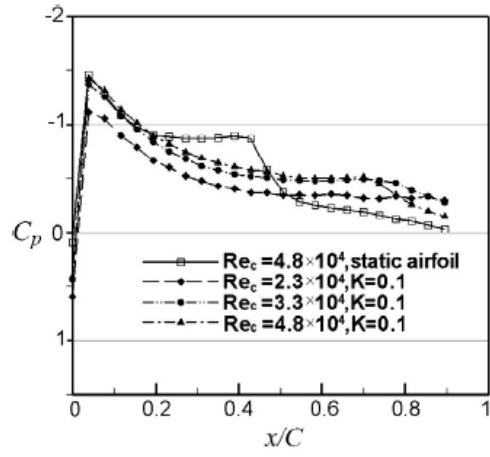
This result implied that the increase in Reynolds number promotes the occurrence of boundary-layer events such as laminar separation and transition.

A paper by Kurtules[5] suggested possible algorithms and methods for pitching mechanism at low frequencies.

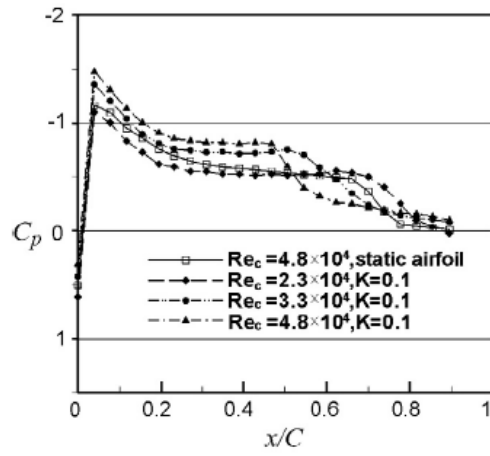
In this study, the unsteady behavior of the flow around a symmetric airfoil was considered as incidence angle increases. The vortex pattern generated was analyzed numerically for different angles of attack from 5 to 90 degrees at $Re=1000$ around NACA 0012 airfoil. At this Reynolds number, the flow was laminar and boundary layers were quite thick. Flow separation and unsteady vortex shedding is observed even at low angles of attack. ANSYS software was used for mesh simulation and post-processing.

Governing equations used were:

- The governing equations were Navier-Stokes equations for incompressible, laminar and two dimensional flow:
- Flow Condition: $Re = 1000$ steady external conditions
- ANSYS Fluent implements the finite-volume method to solve conservation equations
- Pressure-velocity coupling is done by means of the SIMPLE-type fully implicit algorithm
- Transient solution has been approximated using second order implicit method

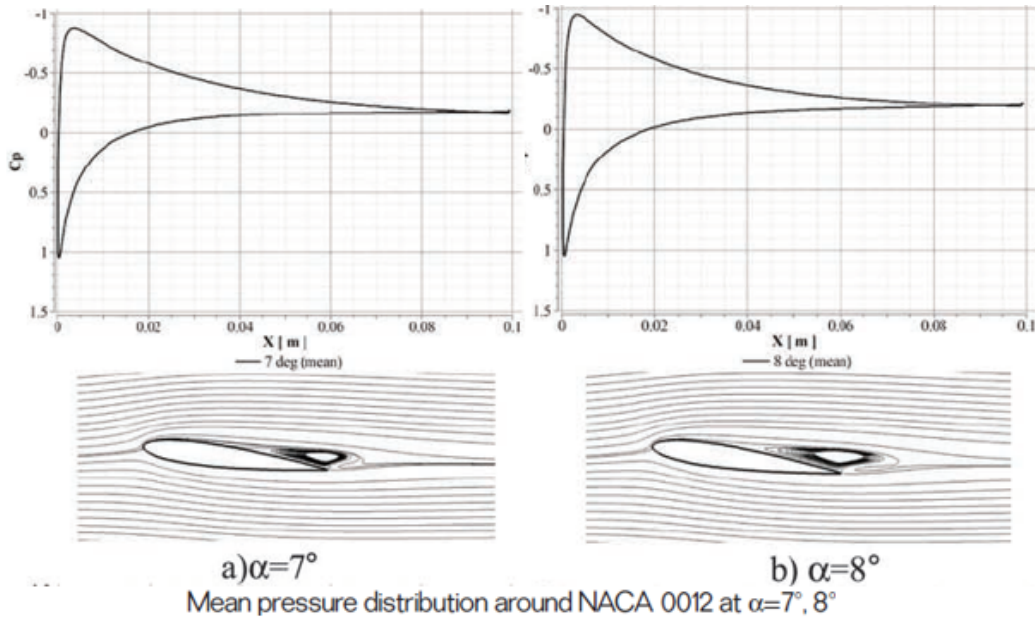


(a) $\alpha_u=5.0^\circ$



(c) $\alpha_d=3.0^\circ$

Figure 11: Plots of coefficients of drag and pressure



- The solution is second-order accurate in space and time
- Velocity inlet boundary condition is used at the semi-circular region of the outer domain
- Pressure condition is used at the outlet boundary of the outer domain.

The results predicted the solutions for different angles of attack are obtained from 0° to 90° . They were obtained with 1° increments from 0° to 41° and with 10° increments from 40° until 90° . The mean values for the current study are obtained in the time interval of $t^*[73 \ 146]$ in order not to take into account initial computational effects of the results.

The unsteady aerodynamic coefficients and instantaneous vorticity distribution of NACA 0012 airfoil at $Re=1000$ are presented for $0^\circ < \alpha < 8^\circ$. The analysis shows that the periodic behaviors of the aerodynamic coefficients are not observed for angles of attack below about 8° . The two trailing edge vortices that are generated on the upper surface of the airfoil are not shed downstream of the airfoil for less than 7° . The aerodynamic coefficients are constant after the computational starting effects disappears.

The vortex shedding pattern is observed to be constant for angles of attack lower than 8° and alternating at the far downstream of the airfoil for $\alpha = 8^\circ$. The flow at $\alpha < 3^\circ$ corresponds to an attached flow, where the flow was completely attached on the entire upper surface of the airfoil. Skin friction coefficient distribution on NACA 0012 below $\alpha = 3^\circ$ do not show any separation point on the upper surface. The shedding phenomena is observed at the farfield of the airfoil at 8° . Hence, $\alpha = 8^\circ$ is also

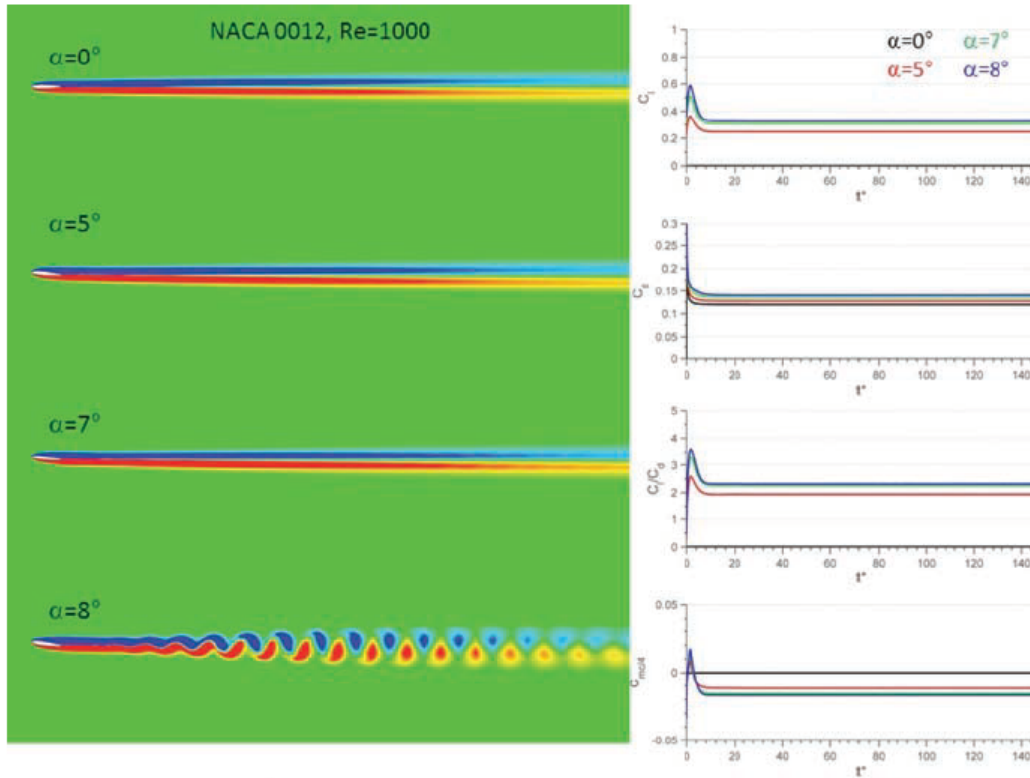


Figure 12: Vortex Generation from a Pitching airfoil

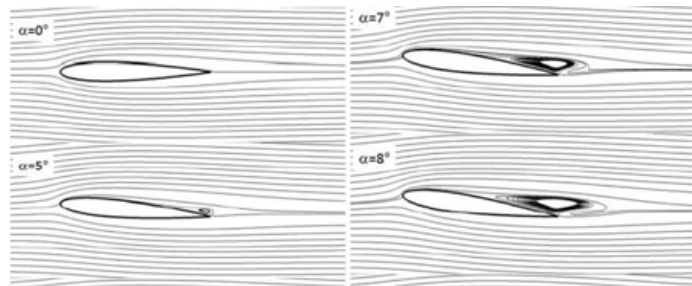


Figure 13: Mean and Unsteady vortex patterns

alternating vortex sheet formation angle.

Above 8° , the alternating vortices are observed just behind the airfoil. As angle of attack increases to the point where the flow separation is initiated, a shear layer forms near the separation point. The vortex created by this shear layer is in clockwise direction, however the vortex created by the shear layer formed from the trailing edge has a counterclockwise direction. These two shear layers roll up in opposite directions and an alternating wake rollup pattern is obtained.

0.4 Methodology: OpenFOAM

0.4.1 Static Analysis Case

Simulation: Solver

Static analysis is done with the help of the solver known as PISOFOAM. PISO stands for “Pressure Implicit with Splitting of Operators”.

PisoFOAM- It is strictly an incompressible flow solver coupling between density and pressure is removed, as well as the coupling between the energy equation and the rest of the system. It solves the coupled continuity equation:

$$\nabla \cdot \mathbf{U} = 0 \quad (3)$$

and the momentum equation:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = \nu \nabla^2 \mathbf{u} - \nabla p + \mathbf{g} \quad (4)$$

There is no pressure equation, but the continuity equation imposes a scalar constraint on the momentum equation (since $\nabla \cdot \mathbf{u}$ is a scalar).

The idea of PISO is as follows: Pressure-velocity systems contain two complex coupling terms: Non-linear convection term, containing u-u coupling.

Linear pressure-velocity coupling: On low Courant numbers (small time-step), the pressure-velocity coupling is much stronger than the non-linear coupling.

It is therefore possible to repeat a number of pressure correctors without updating the discretization of the momentum equation.

In such a setup, the first pressure corrector will create a conservative velocity field, while the second and following will establish the pressure distribution.

On the negative side, the derivation of PISO is based on the assumption that the momentum discretization may be safely frozen through a series of pressure correctors, which is true only at small

time-steps. Experience also shows that the PISO algorithm is more sensitive to mesh quality than the SIMPLE algorithm.

In case of PISOFOAM: Pressure-velocity systems contain two complex coupling terms: Non-linear convection term, containing u-u coupling.

Linear pressure-velocity coupling: On low Courant numbers (small time-step), the pressure-velocity coupling is much stronger than the non-linear coupling.

The non-linearity in the convection term ($\nabla \cdot (uu)$) is handled using an iterative solution technique, where $\nabla \cdot (uu) = \nabla \cdot (u^o u^n)$ where u^o is the currently available solution and u^n is the new solution. The algorithm cycles until $u^0 = u^n$. Use the conservative fluxes, ϕ , derived from the previous time step, to discretize the momentum equation. Now, ϕ represents the 'old' velocity, u^o , in the convective term.

Reasons for using PISOFOAM: pisoFoam uses the non-constant effective viscosity from the non-newtonian model and turbulence model, pisoFoam solves the turbulence equations after the PISO loop:

turbulence - correct();

Simulation: Geometry and Boundary conditions

The geometry is created using the OpenFOAM tool called the BlockMesh utility. It as previously mentioned, has provision for vertices, blocks, edges and splines to create curved geometry.

In this case we decided to make a C-Bordered mesh that coincides with the pitching axis of the airfoil.

The mesh contains:

nPoints: 161500

nCells: 80000

nFaces: 320750

nInternalFaces: 159250

Cells are dense more towards the airfoil and coarser towards the far-field.

The main idea here was to divide the domain into four separate regions, those are inlet, outlet, top, and bottom. To accomplish this, five vertices were defined on the airfoil. Vertex 6 was defined on the leading edge of the airfoil followed by vertices 5 and 7 which were defined on the point of maximum

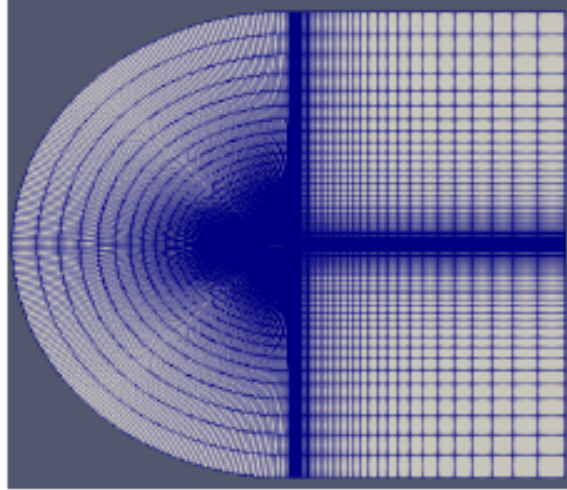


Figure 14: The mesh of NACA-0012 airfoil

Parameter	Inlet	Outlet	Top and Bottom	Airfoil
u	fixedValue	inletOutlet	inletOutlet	noSlip
p	fixedValue	inletOutlet	inletOutlet	zeroGradient
ν_t	fixedValue	inletOutlet	inletOutlet	nutUSpaldingWallFunction
k	fixedValue	inletOutlet	inletOutlet	kqRWallfunction
ω	fixedValue	inletOutlet	inletOutlet	omegaWallfunction

Figure 15: Boundary Conditions for the case.

thickness of the airfoil. Since the flap has to be introduced at a distance of 0.1m(to the left) from the trailing edge of the airfoil, vertices 8 and 9 were defined at the hinge point of the flap.

The variables of particular interest here are the velocity, pressure, and the parameters required for the two equation Omega-SST model, viz, turbulent viscosity, kinematic energy and dissipation rate. The velocity to be given at the inlet can be determined by the Reynolds Number which is given by,

$$Re = \rho v c / \mu \quad (5)$$

where, ρ is the fluid density, v is the inlet velocity, c is the chord length of the airfoil, μ is the dynamic viscosity.

The conditions used for the current case include:

$$\rho = 1\text{g/cm}^3$$

$$c = 1\text{m}$$

$$\nu = 1\text{e-}06\text{m}^2/\text{s}$$

0.4.2 Dynamic Analysis case

Simulation: Solver

PimpledyMFOAM allows us to work with dynamic meshes and simulation regarding the same. The solver works on PIMPLE mechanism with an allowance to motion along 6-DOF. The method utilized in OpenFOAM is known as 6-DOF Solid Body rotation.

PimpleDyMFoam is an implementation of the PimpleFoam solver that allows for dynamic meshes. Like pimpleFoam, the solver is transient, allowing for relatively large time steps due to the hybrid PISO-SIMPLE (PIMPLE) algorithm. This solver is been utilized for the current test-case of pitching airfoil. It allows for oscillatory or moving (dynamic) meshes and it allows for generic turbulent models such as laminar, LES or RAS.

6-DOF Solid Body Rotation

Parameters required for the pitching mechanism can be found in the sixDoFRigidBodyMotionCoeffs. These parameters seemed confusing because they combine to define several items relating to the body motion. The parameters can be applied in any order. To clarify the application of each parameter, they were further divided into the following main categories.

- 1-Mesh Morphing Control
- 2- 6DoF Solver Control
- 3- Body Definition
- 4- Forces Definitions
- 5- Motion Definitions
- 6- Output Control

All documentation from this point on assumes that any information is applied within the sixDoFRigidBodyMotionCoeffs subdictionary.

Boundary conditions

The simulation would be done for -8 degrees to +8 degrees varying every 2 degrees. It would have a velocity value of 40m/s as the same in wind-tunnel lab with very low pitching frequency: 0.00467 Hz.

The plots obtained is compared with the ones overlapped in static-analysis case.

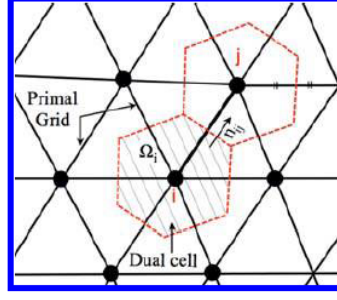


Figure 16: Space Integration in SU2

0.5 Methodology: SU2

0.5.1 Solver: Space Integration

SU2 discretize PDEs using a finite volume method determined on a standard edge-based structure on a dual grid, with control volumes built using a median-dual, vertex-based scheme. The centroids, face, and edge-midpoints of all cells sharing the particular node are connected to form median-dual control volumes as shown for Ω_i is the volume of control volume with i and j vectors denoting edges. The convective and viscous fluxes are then evaluated at the midpoint of an edge by the numerical solver first through all of the edges in the primal mesh in order to calculate the fluxes evaluated at the mid-points and then integrating them to evaluate the residual at every node in the numerical grid.

Integration of Convective Fluxes

Central or upwind methods can be used to discretize the convective fluxes using several numerical schemes like JST, Lax-Friedrich, Roe, AUSM, HLLC, Roe-Turkel have been implemented in SU2. Flux Difference Splitting (FDS), an incompressible upwind scheme, is used with a low-speed preconditioning, as upwind schemes tend to give higher degree of accuracy than central schemes. Monotone Upstream-centered Schemes for Conservation Laws (MUSCL) approach is used to achieve second-order accuracy.

Integration of Viscous Fluxes

To evaluate the viscous fluxes using FVM, flow quantities and their first derivatives need to be evaluated at the faces of the control volume. The values of the flow variables involved are then averaged at the cell faces in SU2 using Green-Gauss (GG) or least-squares method. The truncation error is

taken care of while calculating the gradients of the flow variables at all grid nodes and then averaged to obtain the gradients at the cell faces.

In the current study, the gradients are calculated using Green-Gauss method, a “divergence theorem” based gradient calculation method that is believed to outperform the least-squares (LS) gradient method, in the simulation of aerodynamic boundary layer flows over curved surfaces (syrakos et al., 2017).

0.5.2 Time Integration

Steady Simulations

SU2 uses a local-time-stepping technique to accelerate convergence to a steady state using Euler (first-order) and Runge kutta (second and fourth order) implicit and explicit methods. Local-time-stepping allows each cell in the mesh to advance at a different local time step, calculated by the estimation of the eigenvalues and first-order approximations to the Jacobians at every node. In RK method the room for errors is very small, leading to rapid changing behavior and potential non-convergence. Implicit Euler Scheme is used to discretize the system which offers stability at higher CFL numbers. Local-time-stepping is used to accelerate the convergence where each cell in the mesh advances at a different local time step. The options ITER or INNER_ITER are used to control the number of iterations where INNER_ITER has the precedence.

Unsteady Simulations

To enable a time-dependent simulation the TIME_DOMAIN option is set to YES and DUAL_TIME_STEPPING-2ND_ORDER is selected of the many time marching options available. The dual-time stepping approach implemented in the SU2 solves the following problem:

$$\frac{\partial V}{\partial \tau} + R^*V = 0, \quad (6)$$

where $R^*V = \frac{3}{2t}V + \frac{1}{|\Omega|^{n+1}}(R(V) - \frac{2}{t}V^n|\Omega|^n + \frac{1}{2t}V^{n-1}|\Omega|^{n-1})$ is the residual for real time t and fictitious time τ .

The unsteady problem is transformed into a steady problem at each physical time step which can then be solved as a steady problem. INNER_ITER controls the number of iterations for the pseudo time

or internal iteration and TIM_ITER controls the physical time iterations. The options TIME_ITER or MAX_TIME are used to control the physical time iterations depending on the first occurrence.

0.5.3 Linear Solver and Preconditioner

Linear Solver

Linear algebra (solution of the matrix equation $AX=B$) lies at the very heart of any cfd solver and ensuring efficiency of the solver is extremely crucial in making the entire simulation procedure quick and efficient. As the selection of a solver depends on the simulation conditions and requirements, SU2 offers a number of solvers, of which, Flexible Generalised Minimum Residual or FGMRES (SU2 default option) was chosen for the current study. FGMRES, a more “flexible” and faster version of the GMRES, is a Krylov-space iterative solver capable of handling non-symmetric systems of linear equations. It implements two iteration phases: an inner iteration that uses GMRES to precondition the system, and an outer iteration that minimizes the system residual to generate the solution (DeVries et al.,2013)

Preconditioner

Preconditioning is the application of a preconditioner (a transformation) which transforms the original system into a form that is more suitable for numerical solution by improving the convergence criteria.

The preconditioning is obtained by construction of lines in the mesh normal to grid stretching in a matrix constructed with the diagonal entries of the system matrix and the non-diagonal entries of the edges that belong to linelets (grid lines). A system $Ax = b$ can be solved using preconditioning by solving a system of equations of the form $Pz = r$, where P is the linelet preconditioner.

The algorithm involves:

Linelet building

Preconditioning matrix construction

Nodal renumbering to obtain tri-diagonal structure for preconditioner

Use of the formulation of the iterative scheme

Solution of the tridiagonal system for the linelets using direct method

0.5.4 Convergence Algorithm

Convergence acceleration techniques are the innovation in cfd which speed up the iterative solvers tremendously, allowing realistic CFd simulations. High-frequency errors are usually damped for most iterative relaxation schemes, however low frequency errors(global errors spanning the larger solution domain) aren't well damped.

Steady-state residual field is chosen for controlling the convergence. The residual looks at the density residual as CONV_FIELD is specified as RESIDUAL and continues the simulation until the same decreases to the order of -6 as mentioned under the option CONV_RESIDUAL_MINVAL.

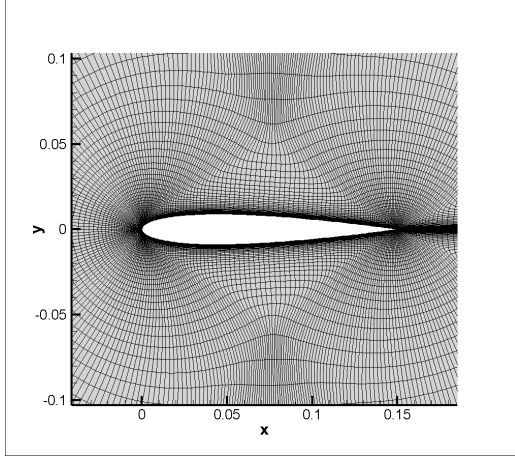
0.5.5 Geometry and Mesh Movement

Geometry

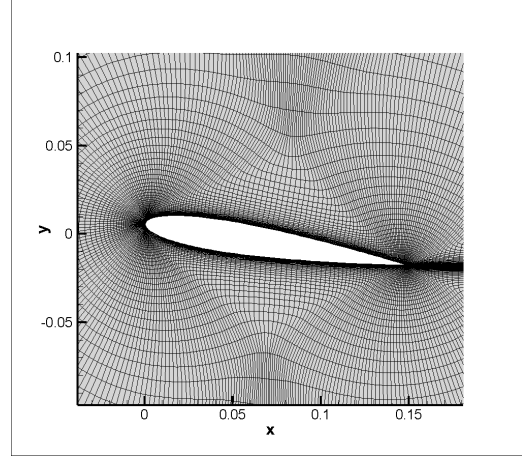
A 2D C-grid structured mesh is used after applying SU2_DEF function to reduce the unit chord of the mesh to 0.15m. The selected grid has a far-field extension of 500c with 57824 points overall and 256 points on the airfoil. The grid is stretched in the direction normal to wall with dense meshing around the airfoil to accurately model the parameters and their gradients in the proximity of airfoil surface. The boundaries of the mesh, recognized by SU2 as “markers”, are tagged as “airfoil” and “farfield” for the airfoil surface and farfield boundaries respectively.

Mesh Movement

The volumetric deformation procedure is used for moving the mesh. The surface boundary, airfoil, is moved followed by deformation of the volume mesh to accommodate the new surface position with each time step. The key element of this model is the definition of a stiffness matrix k_{ij} which connects two ends of a single mesh edge. The system of equations is solved iteratively using the same linear



(a) Mesh at 0° AOA



(b) Mesh at 9° AOA

Figure 17: Mesh from volumetric deformation

solvers used for the simulation. Equilibrium of forces is then imposed at each mesh node

$$\left(\sum_{j \in N(i)} k_{ij} \vec{e}_{ij} \vec{e}_{ij}^T \right) \vec{u}_i = \sum_{j \in N(i)} k_{ij} \vec{e}_{ij} \vec{e}_{ij}^T \vec{u}_j, \quad (7)$$

where, the displacement \vec{u}_i is the unknown and is computed as a function of the known surface displacements \vec{u}_j . $N(i)$ is the set of neighboring points to node i and \vec{e}_{ij} the unit vector in the direction connecting both points. The captures at 0° and 9° AOA are as shown in figure 17.

0.5.6 Simulation and Modeling Parameters

Static analysis using SU2 was done on NACA0012 airfoil for Angles of Attack 0, +5, -5, +8 and -8, based on the Navier Stokes Equation that consisting of the following two constituent equations:

The continuity equation:

$$\nabla \cdot \mathbf{U} = 0 \quad (8)$$

and the momentum equation:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = \nu \nabla^2 \mathbf{u} - \nabla p + \mathbf{g} \quad (9)$$

The fluid was assumed to be incompressible and viscous, with an unchanging viscosity throughout the flow domain. SST (Shear Stress Transport) model, which is a two-equation eddy-viscosity turbulence model, was used for turbulence modelling. It combines the turbulence models k-omega, used in inner region, and k-epsilon, used in outer free-shear-flow region. To compute the gradient of scalars, Green- Gauss Gradient Scheme was used.

Green Gauss Scheme is based on Gauss' Theorem – the surface integral of a scalar function is equal to the volume integral (over the volume bounded by the surface).

Time-integration is very key to any cfd simulation and central to the discussion on time-integration is the concept of Courant Number. It is a measure of how much information traverses a computational grid cell in a given time step. The Courant-Fredrichs-Lewy Condition is the maximum allowable Courant Number that a time-integrator can have.

A Courant Number of 25 was used here for static analysis. The convective numerical method used for flow definition was FDS (Fire Dynamics Simulator), a fire driven fluid-flow method that solves numerically an LES form of Navier Stokes Equation appropriate for low-speed thermally-driven flow, and that for turbulent method definition was Scalar Upwind. An Euler Implicit Time Discretization Scheme was implemented for both flow and turbulent numerical methods, as implicit schemes provide greater stability than their explicit counterparts. The method used for numerical solution of system of linear equations was Flexible Generalized Minimal Residual Method (FGMRES).

This method approximates the solution by the vector in a Krylov-Subspace with minimal residual. ILU (incomplete LU factorization), which is a sparse approximation of the LU factorization, was used as the preconditioner for the Krylov linear solver. The convergence criteria used was Residual, with Drag being the variable on which the convergence criteria was applied.

Physical Parameters

Density = 1.1839 kg/m³ Freestream velocity (x,y,z) = (40, 0, 0) Viscosity = 1.849×10^{-5}
Freestream temperature (for density and viscosity values) = 300 K Pitching Axis Location (relative to chord length 'c') = 0.25c from leading edge Angle of Attack = 0, +5, -5, +8, -8

Geometry

Grid type – A C-grid was used as the domain of fluid flow Chord Length = 1 unit of length

Mesh

Number of Nodes = 57824 Number of Elements = 57344

Boundary Conditions

FARFIELD – It refers to the outermost boundary of the domain and physically represents the impermeable walls within which the flow flows applied on the farfield.

AIRFOIL – It signifies the physical boundary of the airfoil. Constant heatflux boundary condition is used with zero heat flux , to enforce the adiabatic no-slip wall condition on the airfoil surface.

0.6 Methodology: Experimental Investigation

0.6.1 Manufacturing and Fabrication

NACA-0012 airfoil is manufactured using the Laser-cutting machine at the Propel Lab-3 in BMS College of Engineering. The material used is Balsa wood for the fabrication.

The details of the airfoil:

chord length= $150mm$

thickness= $18mm$

density of Balsa wood= $75kg/m^3$

The fabricated parts are put together using 743-glue. The smooth surface is obtained by sanding it over sand paper of fine grit of 350 grit. The finishing is then done with resin.

The fabricated airfoil is later inserted in the wind-tunnel, connected to manometer tubes outside. The tubes are used to measure the pressure distribution for various angles of attack.

Wind-tunnel specifications: The open circuit wind tunnel is set up and the experiment was conducted on the said NACA-0012 airfoil model.

Wind-Tunnel Type	Open circuit, Suction Type
Test section size	390 * 302 * 1140mm
Jet Type	Honey Comb Stream-lining
Maximum Velocity	30m/s
Motor Power	15KW
Fan speed	2300rpm

0.6.2 Experiment

Static Case

The wing is symmetric and has a passage through its transverse cross section at 25% of the chord for the rod. The surface of the wing at the central cross-section has 16 ports, at known distances from the leading edge of the airfoil, relative to its chord length.

The ports are connected to manometric extensions, which relay the pressure head to a manometer having 30 ports – 22 of which are connected to the extensions from the wing, 2 are connected to pitot tube for measuring free stream pressure and 6 are open to the atmosphere, they denote a reference for the measurement of change in head in the ports connected to the extensions from the wing. The wing is assembled with the rod, which itself is attached to the circular board with locking for the wind-tunnel wall and scaling for different angles of attack.

The model is then mounted in the test section of the wind-tunnel and the pressure tapings are connected to the manometer as per the numbered labels provided on the extension tubes. The readings of the manometer are adjusted to a particular height and made zero. The wind-tunnel speed is set to 30 m/s by adjusting the rpm of the motor. At the desired speed and angles of attack, the difference in pressure heads in respective manometric tubes are recorded and static pressure is measured at the corresponding points. From the pressure data, C_p is calculated as follows:

$$C_p = \frac{p - p_\infty}{\frac{1}{2}\rho v^2} \quad (10)$$

Where, $p - p_\infty$ is the pressure difference, ρ is the density of air, $v = 40$ is the velocity of flow. The

calculation of static pressure is based on the principle of measurement of change in manometric head when the corresponding ports are subjected to flow in the wind tunnel.

Dynamic Case

The pitching mechanism is carried out using multiple components such as stepper motor, arduino circuit, spur gears. The arduino is connected to a 240V, 5A DC power supply, and the stepper motor is connected to a spur gear with module 1.5 and gear ratio 1:1. The spur gear is meshed with another spur gear that's connected to the shaft that pitches the airfoil.

Motor Type	Stepper Motor
Torque	$8Kg/cm$
Speed of the motor	$500rpm$
Minimum Step Angle	1.8

The gear rotates at the step rate required and to be coded. For a pitching of 8 degrees, for one oscillation, the desired time was 60 seconds, hence the frequency of oscillations turned out to be 0.0467Hz.

0.7 Results and Discussion

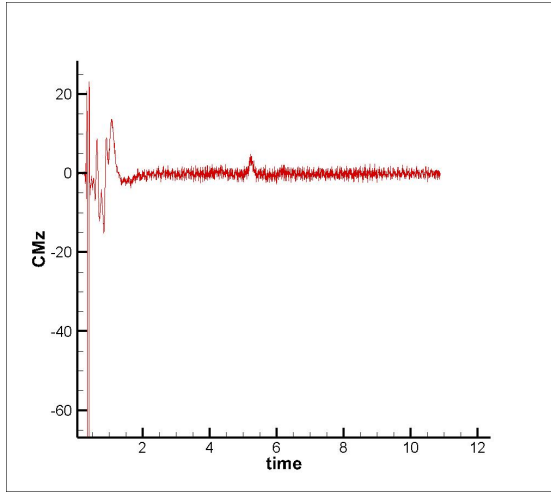
0.7.1 Results: OpenFOAM

Static Case Analysis

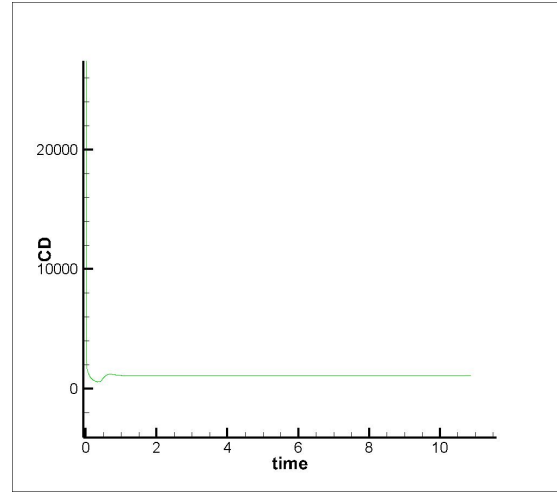
Static case analysis was done for various angles of attack: -9° , -7° , -5° , -2° , 0° , 2° , 5° , 7° , 9° .

The plots were compared theoretically and analyzed for differences. 1 unit chord length was considered and the geometry was made using BlockMeshDict utility, with a timeStep= 0.0001s and writeInterval= 1e-05, the data was written and analyzed.

The convergence was plotted using TecPlot software for all the aerodynamic coefficients. The figures show the convergence rate of the aerodynamic coefficients with time. It can be observed that the moment coefficient steadily converges after multiple oscillations, damping with time. On the other hand, drag coefficient has a rather steep and narrow range of convergence. Both are observed for 0 degree angle of attack. Coefficient of lift convergence is rather intermediate and that is due to zero

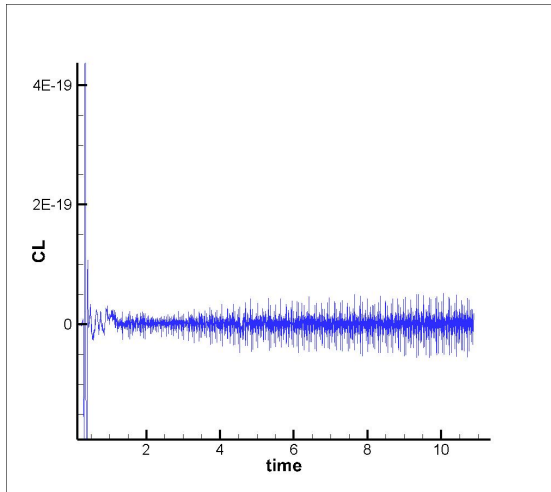


(a) Convergence of C_m for 8deg

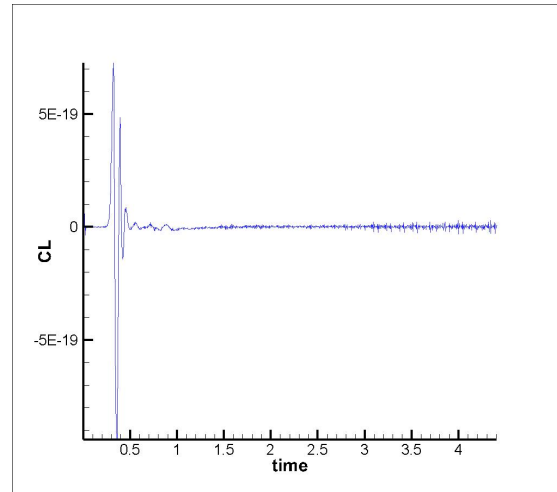


(b) Convergence of C_d for 8deg

Figure 18: Convergence study using Tecplot for 8 degree AOA



(a) Convergence of C_l for 8deg

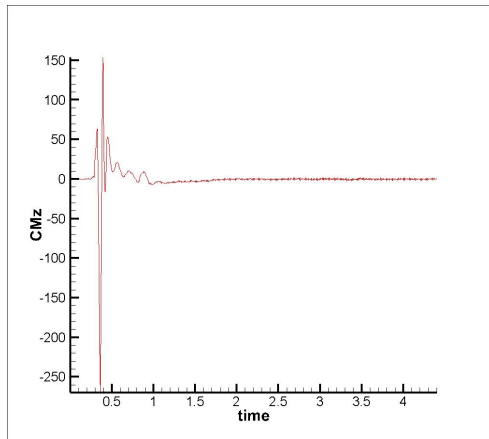


(b) Convergence of C_l for 0deg

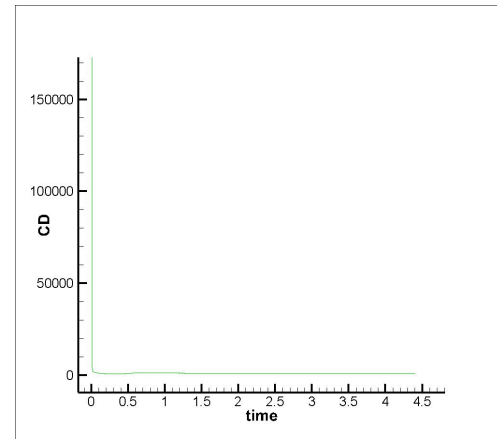
Figure 19: Convergence study using Tecplot for 0 and 8 degree AOA

angle of attack. There is considerable oscillation and damping effect but lower than that of moment coefficient.

We then simulate for -8 degree angle of attack and analyze the difference in the rate of convergence. We observe that the rate of oscillations during convergence is significantly higher at -8 degrees AOA. This may be due to the angle change and might fluctuate more if we increase the angle consistently. The same is shown for moment coefficient at 8 degree AOA, the amplitude greatly varies initially and dampens with fluctuations and oscillations, higher than the one found for 0 degree angle of attack.

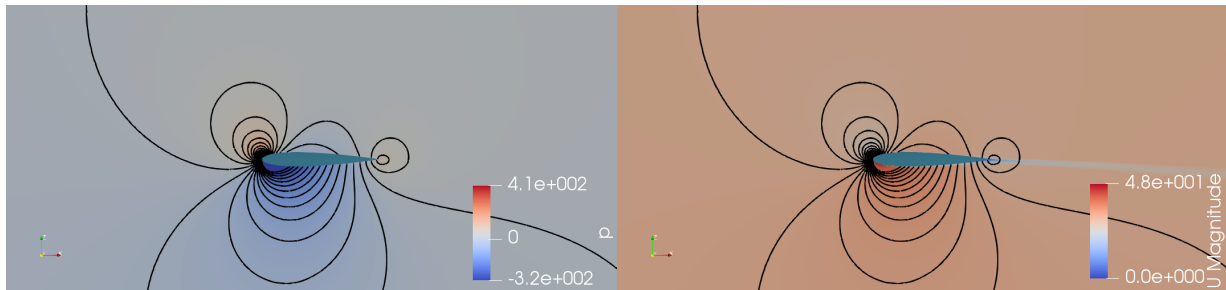


(a) Convergence of Cm for 0deg



(b) Convergence of Cd for 0deg

Figure 20: Convergence study using Tecplot for 0 degree AOA



(a) Pressure contour for -5 degrees AOA

(b) Velocity contour for -5 degrees AOA

Figure 21: Paraview simulations for static cases

Coefficient of drag has a similar trend to that of 0 degree in the 8 degree convergence study, with slight variations near zero and then a steady convergence.

The figures above show the characteristics of contour in case of pressure and velocity for -5 degrees angle of attack. The airfoil's image is exaggerated for the analysis purpose. We observe higher pressure and a lower velocity (corresponding to bernoulli's equation) at the rear end while the reverse happens at the tail end.

There was no vortices formed, probably due to the Reynolds number being high- around 2×10^5 and considering a low velocity of about 40m/s.

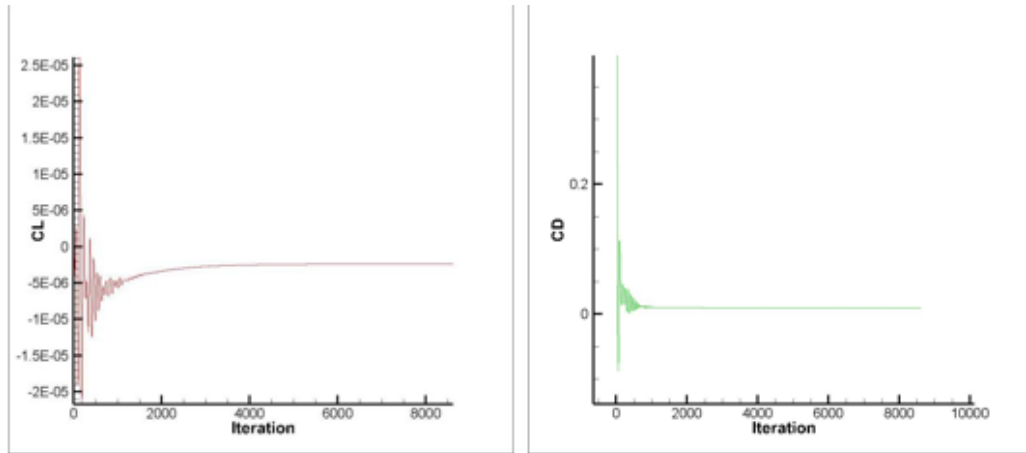


Figure 22: Convergence of Cl and Cd at 0 degree AOA

0.7.2 Results: SU2

Static case analysis

Static case analysis was carried out for angles of attack: -9, -7, -5, -2, 0, 2, 5, 7, 9. The plots were compared and analyzed for differences with the OpenFOAM results, theoretical data, and experimental values. The geometrical parameters and the free stream parameters were identical to the OpenFOAM for comparative study. The convergence was plotted using TecPlot software for all the aerodynamic coefficients.

The figures show the convergence rate of the aerodynamic coefficient with time.

It can be observed the lift coefficient has general damping pattern with the iterations, where the oscillation in the values of the coefficient is relatively large during the first few hundred oscillations and dampen out with further iterations. The amplitude of initial oscillations are greater for -9 and 9 degrees of angle of attack compared to the -5, 0 and 5 degrees. Drag coefficient when plotted against iterations for the above mentioned angles of attack. The coefficient of drag values were observed to oscillate initially and dampen gradually. However, the values were seen to exhibit less amplitude of oscillation which were also less frequent. The oscillations tend to shorten out to a constant value much earlier than that for lift coefficient for corresponding angles of attack.

The moment coefficient was calculated about the z-axis, and the calculated values were plotted against the iterations to uncover large initial oscillations at the beginning which gradually die out to a constant value. Compared to the lift and drag coefficients plots, the plots for moment coefficient converged faster.

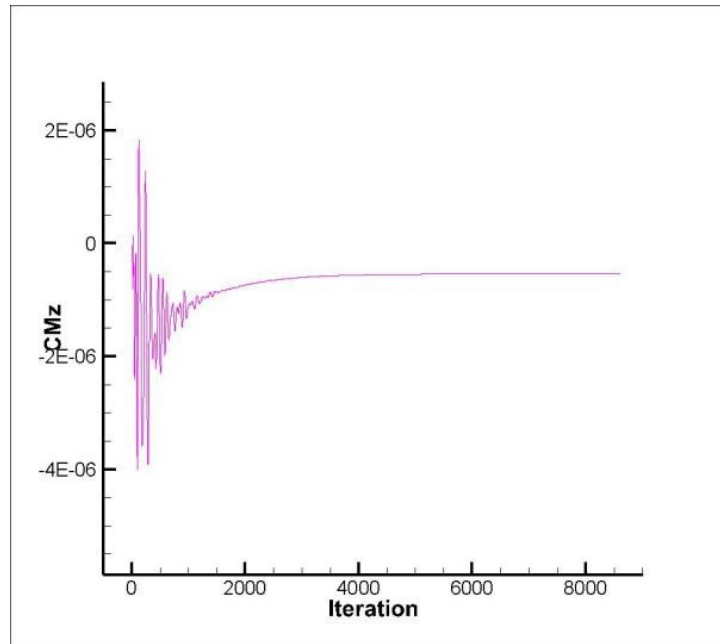


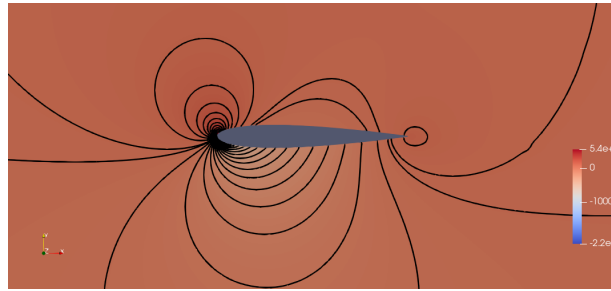
Figure 23: Convergence of C_m at 0 degree AOA

The negative slope for positive angles of attack and positive slope for negative angles of attack was observed in the plots for lift and drag coefficients against iterations with an infinitesimal value but with very less variation when compared to the initial oscillations and hence the results have much converged.

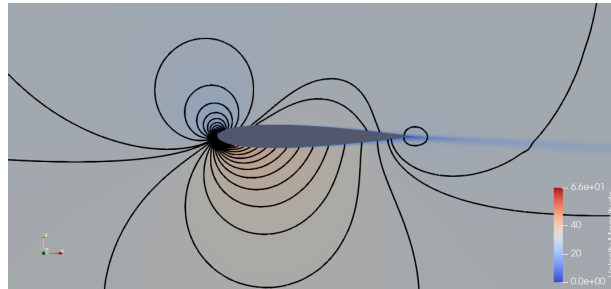
The figures above show the characteristics of contour in case of pressure and velocity for -5 degrees angle of attack and results were comparable with OpenFOAM.

The contours and streamline plots suggest that for a flow speed of 40 m/s for NACA 0012, the flow remains attached and laminar throughout the surface without any vortices formed. The pressure contours are observed to be curved contours encircling each other with a bump of high pressure at the stagnation point and low pressure at the alternate surface of airfoil while they are symmetrical for 0 angle of attack.

The leading edge is observed at the leading edge for no angle of attack and shifts right on the upper and lower surface for increase in negative and positive angles of attack respectively. No wake is observed for any of the flows. However, flows with non-zero angle of attack share contour characteristic tails of low velocity flow extending beyond the trailing edge.



(a) Pressure contour for -5 degrees AOA



(b) Velocity contour for -5 degrees AOA

Figure 24: Contours of Pressure and Velocity at -5 degree AOA

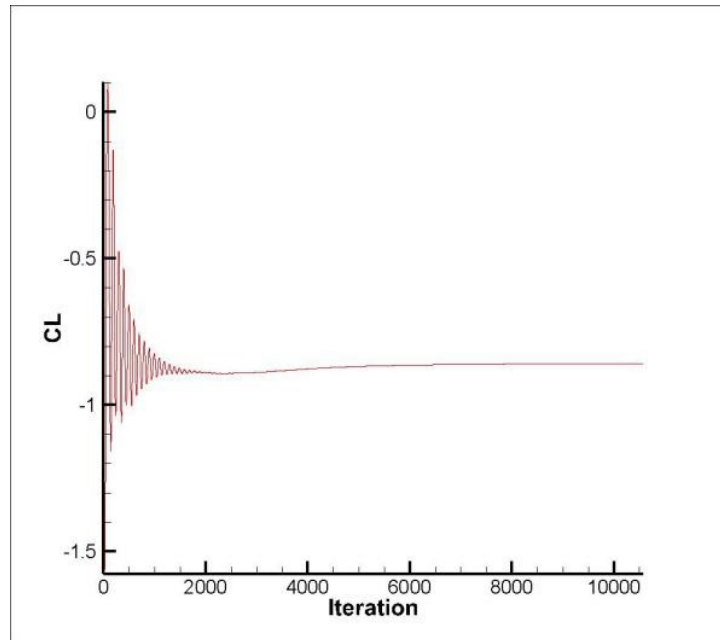


Figure 25: Convergence of CL at -8 degree AOA

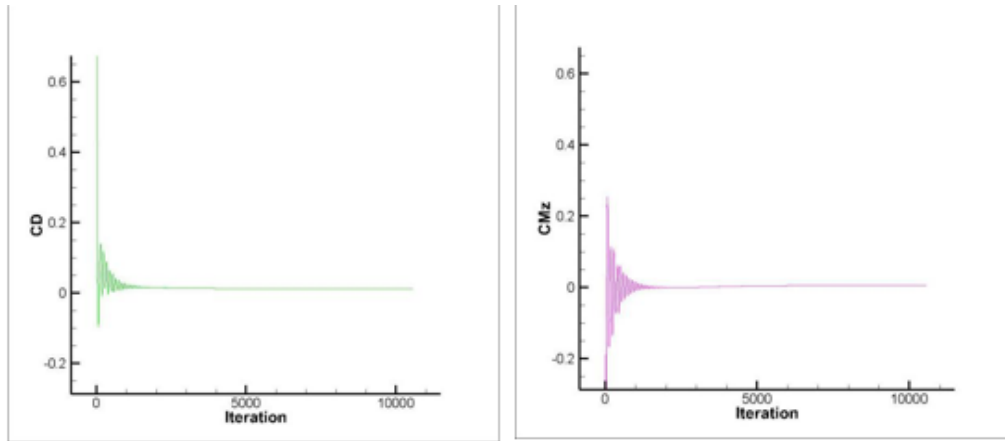


Figure 26: Convergence of Cm and Cd at -8 degree AOA

0.7.3 Plots on SU2 and OpenFOAM

Aerodynamic coefficients such as coefficient of lift (C_l), coefficient of drag (C_d) and coefficient of moment (C_m) were plotted in both SU2 and OpenFOAM. The plots were done for -8,-5,0,5,8 degree angles of attack, and compared based on their differences. The figures show the plots of all three coefficients with respect to the angle of attack.

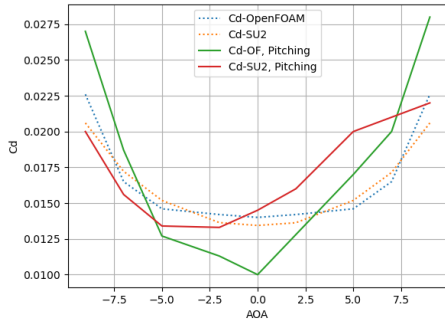
The difference in case of moment coefficient initially for a larger angle of attack is higher and progressively reduces towards 0 degree angle of attack. Then the difference slightly increases beyond that.

In case of lift coefficient, the same trend is observed but the difference is lesser than that of moment. The reason is probably that the angle of attack is less and velocity is also subsonic. They coincide at almost all angles.

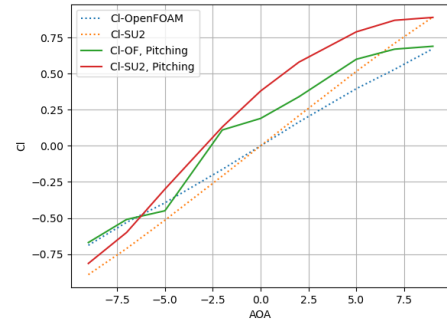
Drag coefficient shows an even more closer correspondance between openFOAM and SU2 with insignificant difference throughout.

The reason for difference in the first place is the solvers algorithms. PISOFOAM uses pressure-velocity coupling while SU2 uses a wide range of parameters and solvers' procedures that are compiled consistent with the flow simulation.

Figure- 26 shows the plots of flow coefficients vs alpha (the angle of attack) of both the software for both, static and dynamic cases. Static case results are explained here, dynamic simulation results are discussed in the further section. For the static case analysis, the drag coefficients were deviating as the angle of attack increased, this is possibly due to the fact that wakes at the boundary of the



(a) Drag Coefficient- C_d



(b) Lift Coefficient- C_l

Figure 27: Drag and Lift coefficients vs Alpha

airfoil were observed at an earlier stage in OpenFOAM than on SU2, the resolving of boundary-layer interaction was enhanced and vortices were observable sooner on OpenFOAM as the angle of attack increased. The lift coefficient was slightly deviating even at lower angles of attack, it was observed that the values obtained by OpenFOAM were under-predicted as the angle increased. The possible reason for this would be the difference in the pressure range as observed in the contour, and the trailing edge flow variation differences observed in both the software. The variation reduced considerably when C_l and C_d were calculated using PMean and UMean, indicating the difference in the methodology adopted by the algorithm in computing the flow-field parameters.

The theoretical results by [7] and Suzuki's experimental verification were used to compare the nature of the plots obtained. The case-study was done on a conventional NACA-0012 airfoil and with no flap modification. We observe that SU2 plot for lift coefficient is rather linear compared to the theoretical graph. The reason might be due to the fact that the solver approximates at near wing section and plots accordingly, while OpenFOAM provides a modified linear curve.

The drag coefficient for all three graphs seem to coincide in terms of the nature, yet there is certain smoothness in the simulation results compared to that of experiment.

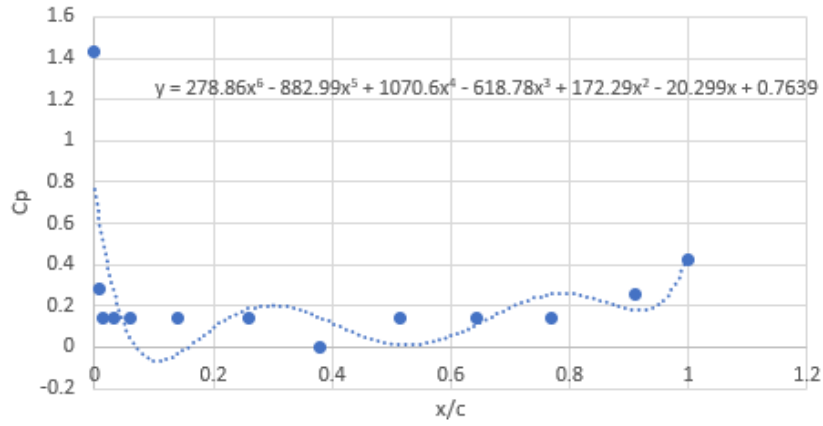


Figure 28: Cp vs x/c for upper surface

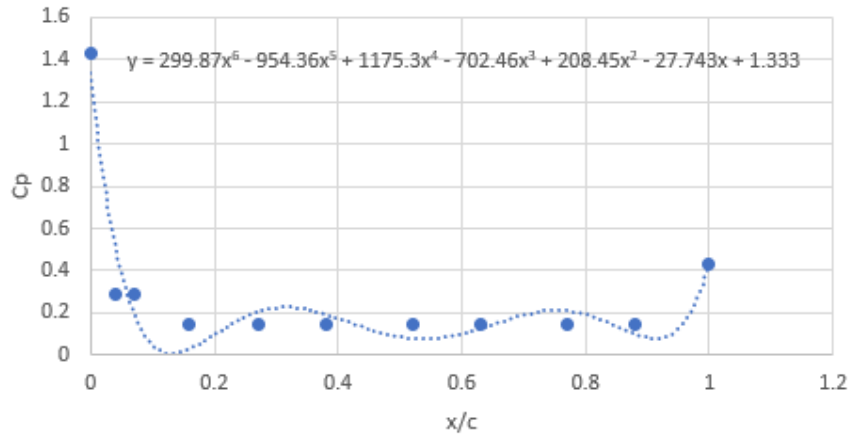


Figure 29: Cp vs x/c for lower surface

0.7.4 Results: Experimental Investigation

Cp for 0 degree AOA

The Cp values obtained were plotted against the corresponding x/c values, separately for the upper and lower surfaces. A best fitting curve of 6th-order polynomial (shown as y in each plot) was obtained for each of the curves, and integrated accordingly, with limits from x=0 to x=1, to find the Lift Coefficient.

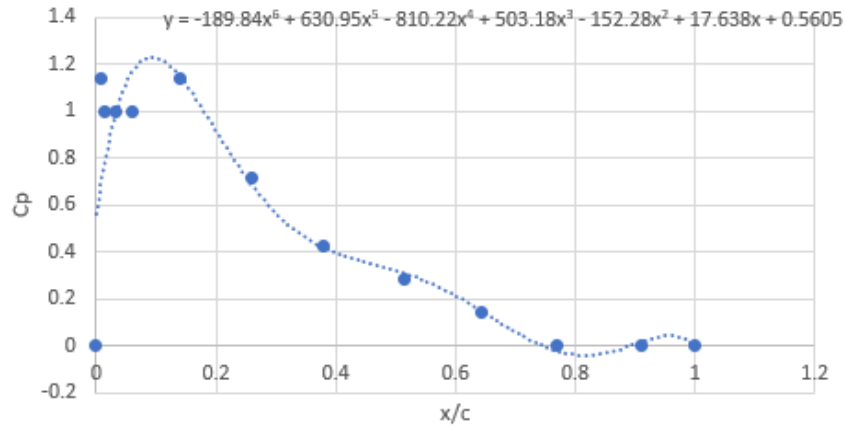


Figure 30: Cp vs x/c for upper surface

Integrated Equation for Cl:

$$\begin{aligned}
 Cl = & (-189.84 * x^7/7 + 630.95 * x^6/6 - 810.22 * x^5/5 + 503.18 * x^4/4 \\
 & - 152.28 * x^3/3 + 17.638 * x^2/2 + 0.5605 * x) \\
 & - (394.98 * x^7/7 - 1224.3 * x^6/6 + 1450.3 * x^5/5 \\
 & - 819.18 * x^4/4 + 221.65 * x^3/3 - 23.444 * x^2/2 - 0.0744 * x)
 \end{aligned} \tag{11}$$

Putting limits from x=0 to x=1, $Cl_{\alpha=0} = -0.0268$

Cp at 10 degree AOA

After integrating and applying limits, $Cl = 0.687215$

Cd for AoA = 10

$$Cd = -0.06 \cos(10) + 0.0268 \sin(10) = 0.06$$

Cp, Cd for AoA = -10

Similarly, Cp and Cd values were obtained for Angle Of Attack = -10 degrees and were found to be:

$$Cp = -0.69107$$

$$Cd = 0.06$$

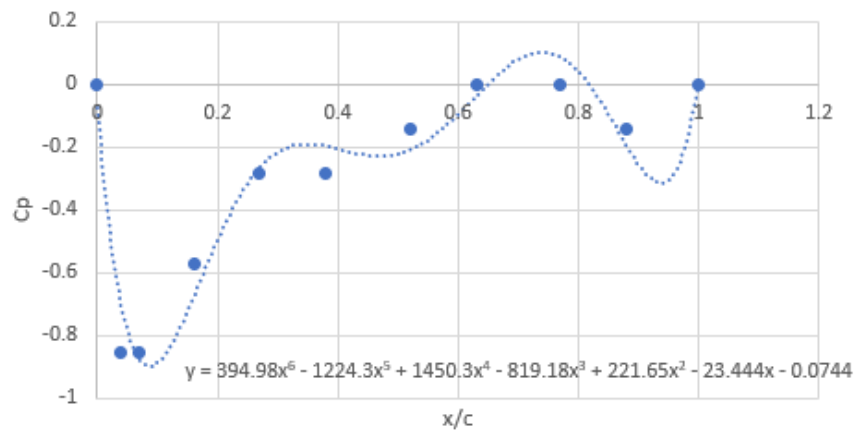


Figure 31: C_p vs x/c for lower surface

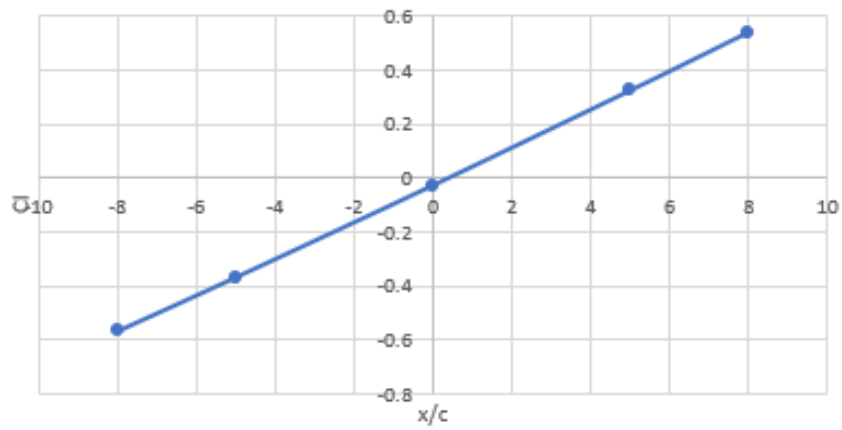


Figure 32: C_l vs x/c for lower surface

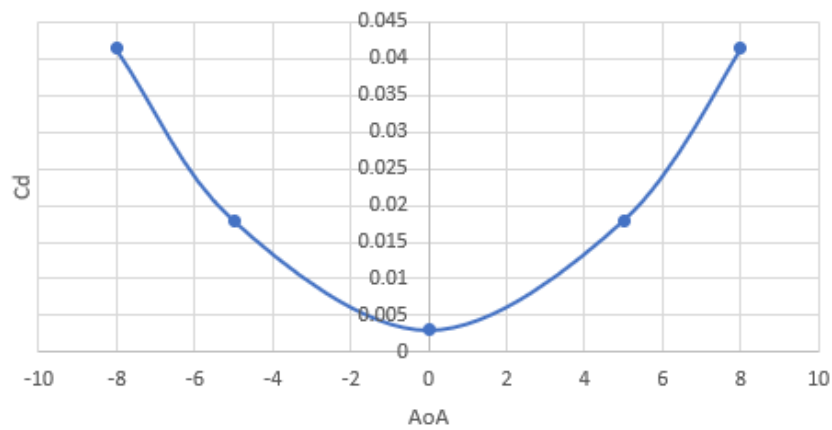
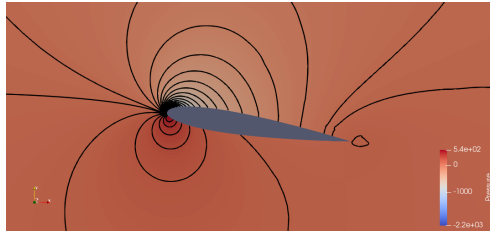
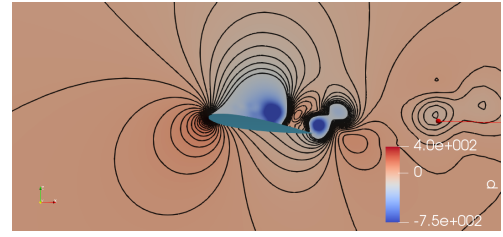


Figure 33: C_d vs x/c for lower surface

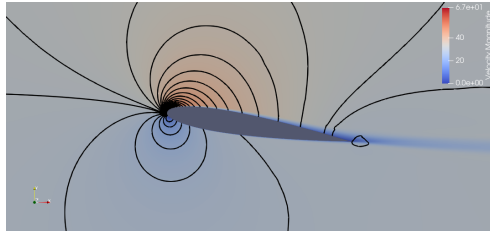


(a) SU2

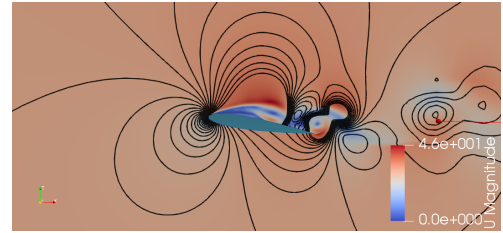


(b) OpenFOAM- pMean

Figure 34: Pressure Contours at 9 degree AOA for the Pitching Simulation



(a) SU2



(b) OpenFOAM- Instantaneous

Figure 35: Velocity Contours at 9 degree AOA

0.7.5 Results: Dynamic Case Simulation

OpenFOAM

Pressure and Velocity Contours

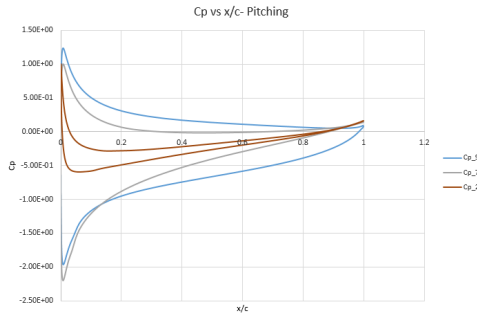
In case of OpenFOAM, both time-averaged (mean) and the instantaneous fields were computed during the simulation. Figures- 36 and 37 depict the contours for the flow parameters. The pressure contour for the OpenFOAM was pMean- time-averaged over the iterations, the values differ in the lower range indicating the variation in the methodology of computing averaged values between the two software. Figure- 5 shows the velocity contours. In case of OpenFOAM, instantaneous flow-field contour is shown to illustrate the formation of wakes and boundary-layer. These were missing from the averaged contours, as in the case of SU2 as it diminishes the effect of the 'fluctuating' component. The variation in wakes formation can also be attributed to the dynamic meshing methodology, as by observations, the 6- DOF RigidBodyMotion solver which works analogous to the spring method on SU2, demonstrated a comparatively weaker wake formation and development of boundary layer, and the formation seemed to enhance as the angle of attack (amplitude) of the airfoil increased.

0.7.6 Flow Coefficients

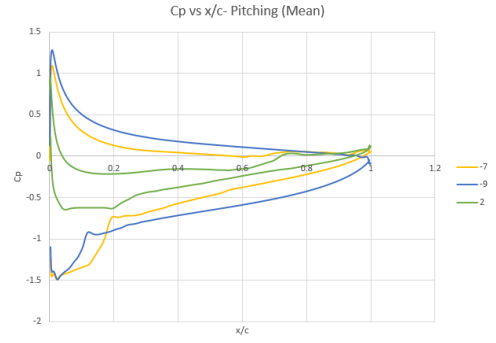
Figure- 26 shows the variation of averaged (computed based on averaged fields) drag and lift coefficients for the pitching simulation. Drag coefficient of OpenFOAM were over-predicted at higher angles of attack, and had a significantly higher range than SU2, showing a jump in obtained values as the airfoil pitches from 0 AOA to +9. While in case of lift coefficient, the values did not show a linear increase in either of the software, even at low angles of attack, due to fluctuation of pressure distribution over the surface causing a continuous variation in pressure difference as the airfoil pitched. The differences could be based on the methodology behind the calculation of flow coefficients, in SU2, the averaged flow-parameters tend to have fluctuations “smoothened” out, while on OpenFOAM, p_{Mean} and U_{Mean} are calculated from the instantaneous parameters, “absorbing” the fluctuations, hence leading to such discrepancies as noted. Since there is no flow attachment and reattachment, there was no indication of dynamic stalling, although a significantly higher range was observed compared to static simulation. As the angle of attack was low, the maximum averaged lift coefficient was around 0.8-0.9 in both the software.

0.7.7 Pressure Coefficient

Figure -35 shows the pressure coefficient plot for angles of attack: 2, -7 and -9. The pressure coefficient for OpenFOAM was calculated based on Mean pressure and velocity values, to eliminate oscillatory nature of the distribution and to focus on the differences based on the values and gradients. As the magnitude of angle of attack decreased, (pitching from -9 degree AOA towards 0), the distribution around the leading edge is smoother in nature, due to less deviation of flow, less wakes around the leading edge, and as the boundary layer at the leading edge seemed to appear after an angle of attack of + 5-6 degrees. The difference between static and quasi-steady simulation solely exists on the enhanced boundary layer, and trailing edge vortices, hence, change in the skin-friction and eventually the drag coefficient. In case of SU2, the algorithm based on averaging being different in comparison to the computation of p_{Mean} and U_{Mean} , and hence, causing a difference in distribution could explain the discrepancy in the lower range of the pressure values even for pitching simulation.



(a) SU2



(b) OpenFOAM

Figure 36: Pressure coefficient (C_p) vs x/c for Pitching simulation

0.8 Concluding Remarks

A symmetric airfoil was pitched at low reduced frequencies on OpenFOAM and SU2, having a total amplitude of 18 degrees. The simulation for static case illustrated the differences in pressure distribution between a static airfoil and a quasi-steady pitching simulation. The flow coefficients variation provided a clearer understanding of the differences in both: static and dynamic case, as well as the algorithm (methodology) implemented by both the software. The time-averaged method of calculation as opposed to instantaneous, and the artificial compressibility procedure to calculate pressure in SU2, could possibly explain differences that arose for the current simulation of interest. A study based on differences in moving mesh algorithms, considering a complete pitching cycle as opposed to only downstroke, analysis of lift coefficients over the cycle as well as maximum obtained, and further, an experimental validation in the future, would provide sufficient understanding in the flow physics behind a low reduced frequency pitching oscillation.

0.9 References

- 1- REVIEW ON OPENFOAM - AN OPEN SOURCE SOFTWARE, J.K. Park, K.H.Kang.
- 2- Stanford university unstructured (SU2): an open source integrated computational environment for multi-physics simulation and design, Francisco Palacios, Thomas D. Economon, Amrita K. Lonkar, Juan J. Alonso.
- 3- SU2: an open source suite for multi-physics simulation and design, Thomas D economon , Francisco Palacios, Sean R. Copeland, Trent W. Likaczyk, Juan J Alonso.
- 4- Low-Reynolds number effect on the aerodynamic characteristics of a pitching NACA0012 airfoil, Dong-Ha Kim, Jo-won Chang
- 5- On the Unsteady Behavior of the Flow around NACA 0012 Airfoil with Steady External Conditions at $Re=1000$, Dilek Funda Kurtulus.
- 6- Numerical simulations of the flow with the prescribed displacement of the airfoil and comparison with experiment, V .Ridky, P. Sidlof and V. Vlcek
- 7- Numerical Investigation of Fluid Flow around a NACA0012 Airfoil in OpenFOAM, Subham Kant Das.
- 8- Investigation of drag and lift force on an airfoil shaped body at different angles, Bhattacharya, Shannnigrahi, Chakraborty, Chattarjee.
- 9- Stanford university unstructured (SU2): an open source analysis and design technology for turbulent flows, Francisco Palacios, Thomas D. Economon, Amrita K. Lonkar, Juan J. Alonso, Copeland.
- 10- Fluid Mechanics, Frank.M.White
- 11- Guide to Version v7, su2foundation
- 12- A numerical method for solving incompressible viscous flow problems, A. J. Chorin.
- 13- Ten years of industrial experience with the SST turbulence model, Florian R Menter, Martin Kuntz, Robin Langtry.