NUMERICAL COMPUTATION OF WIND TURBINE FLOWS AND FLUID PROBLEM BY OPENFOAM AND ANSYS

A Dissertation

by

YI-CHING WANG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2015

Major Subject: Mathematics

ABSTRACT

Wind energy is the mainstream source of clean and renewable energy and it is also the fastest-growing source of sustainable energy in the world. In the Global Wind Energy Council's report in 2014, wind industry grew 44% worldwide. In order to optimize the efficiency of wind farms, it is important to observe wake interactions among wind turbines. Computational mathematics and mechanics provide fundamental methods and tools for simulating physical processes. Numerical computation can offer important insights and data that are either difficult or expensive to measure or to perform tests experimentally. In this dissertation, we use Computational Fluid Dynamics (CFD) software OpenFOAM and ANSYS FLUENT to simulate the wake effect of Horizontal Axis Wind Turbines (HAWT) and related problems. Numerical simulation can also help us comprehend and control man-made disasters. Air craft crashworthiness and human survivability are of utmost concerns in any emergency landing situation. Motivated by the air incidents lately, the disappearance of Malaysia Airlines Flight MH370 in March 2014 and Germanwings Flight 9525 crash in March 2015, we use Computational Structural Dynamics (CSD) software ANSYS Explicit Dynamics and LS-DYNA to try different numerical simulations of Airbus A320 crashing into a wall and compare the results to the reality.

We calculate three CFD problems in this dissertation: lid-driven problems, one turbine wake problem, and two serial turbines wake problem. We simulate a lid-driven flow in both two- (2D) and three-dimension (3D) to compare the simulation capability of the three turbulence modelings, i.e., Direct Numerical Simulation (DNS), Large Eddy Simulation (LES), and Reynolds-Averaged Navier-Stokes Equations Simulation (RANS) by OpenFOAM. Among these three turbulence models, we can find that LES is capable of capturing more details of turbulence flow. We simulate the airflow effect of one wind turbine with both fixed angular velocity and wind-driven case, run benchmark tests based on NRELs reports, and compare the numerical results under the same condition by OpenFOAM and FLUENT. For the fixed angular velocity case, we use wind speed 8 m/s and angular velocity of the wind turbine 75 deg/s. For the wind-driven case, we use wind speed 8 m/s and 16 m/s and the angular velocity of the wind turbine calculated by FLUENT converges faster than OpenFOAM case. We simulate the interactions of wake flow for two serial wind turbines by FLUENT. We use wind speed 8 m/s and angular velocity of the wind turbine 75 deg/s. The wake of former turbine affects the rear one and the diffusion of flow caused by two turbines can be seen clearly. For both one and two serial turbines problems, the turbulence model RANS $k\varepsilon$ is used. We calculate and simulate Airbus A320 crashing into a wall by ANSYS Explicit Dynamics and LS-DYNA. For ANSYS Explicit Dynamics, we use the angle of approach $0°$, $15°$, and $30°$. For LS-DYNA, we only test the pitch angles $0°$. For all cases, we use the speed of aircarft 200 m/s. The deformation of both aircraft and wall can be seen clearly.

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my gratitude to all those who gave me the possibility to complete this dissertation. First, I am deeply indebted to my adviser Dr. Goong Chen for all the guidance, advice, and support that he has provided during this research. I am especially thankful for his valuable insights and patience to train and make me a better researcher.

I would also like to thank all people in my group and Dr. Alain Perronnet for all of their help. I especially thank Dr. Alain Perronnet for providing his professional perception to help me with my aircraft crash simulations patiently.

I would also like to thank the staff of TAMU's Supercomputing Center for allocation and assistance. I really appreciate all technical help from Help Desk staff.

Lastly, I would like to truly thank my family and all dearest friends for all of their love, encouragement, and support during my study. Thank you to keep me energetic and enthusiastic all the time and I can complete this research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION [*]

Wind energy is the mainstream source of clean and renewable energy. It could mitigate and reduce the effect of global warming since wind energy resource doesn't produce the greenhouse gas which mainly results from burning of fossil fuels. Of all renewable energy resources, wind power is the fastest growing one in the world. It is growing at the rate of 30% annually, and it reported that at the end of 2012, the worldwide installed capacity has attained 282,482 megawatts (MW). Wind power is widely used and developed in Europe, Asia, and the United States. In the Global Wind Energy Council's report in 2013 [10], 44.7 Gigawatts of new wind power was added to worldwide capacity in 2012 and and it is a 19% increase over the preceding year. This amount brings overall total global capacity to 282.5 Gigawatts. This represents a tenfold increase in wind power capacity over the last decade. The U.S. Department of Energy aims to grow wind energy to supply 20% of U.S. electricity demand by 2030 [14]. Hence, efficiency of wind farm power production has to be advanced. In a big wind farm, wakes from the upwind turbine can decrease the mean velocity of the downstream turbine and this gives rise to power production loss. It also increases fluctuations and turbulence which results in wind turbines' structural fatigue. Thence, for designing and optimizing wind farms' setup, local turbine wake interactions is important and essential.

Wind energy resource is the energy extracted from wind by using wind turbines

[*]Reprinted with permission from "OpenFOAM for Computational Fluid Dynamics" by Goong Chen and Qingang Xiong and Philip J Morris and Eric G Paterson and Alexey Sergeev and Yi-Ching Wang, 2014. Notices of the AMS, 61(4): 354-363, 2014, Copyright [2014] by American Mathematical Society, and Reprinted with permission from "Malaysia Airlines Flight MH370: Water Entry of an Airline" by Goong Chen, Cong Gu, Philip J Morris, Eric G Paterson, Alexey Sergeev, Yi- Ching Wang, and Tomasz Wierzbicki, 2015. Notices of the AMS, 62(4):330-344, 2015. Copyright [2015] by American Mathematical Society.

to produce electrical power. The normal 1.5 MW wind turbine for commercial production of electrical power has a tower 260 feet high and the length of the blades is between 110 to 124 feet. The rotor assembly (blades and hub) weighs 48,000 pounds and the nacelle, which contains the generator component, weighs 115,000 pounds. It is money and time consuming to do the experiment.

Computational mathematics and mechanics provide fundamental methods and tools for simulating physical processes. Numerical computation can offer important insights and data that are either difficult or expensive to measure or to perform tests experimentally. It has been recognized for at least 30 years that *computational science* constitutes a third and independent branch of science, on equal footing with *theoretical and experimental sciences.* Cutting across disciplines at the center of computational science is *computational fluid dynamics (CFD)*, which makes up the core of FLUENT and OpenFOAM. Using CFD software, such as Fluent and OpenFOAM, for many questions in wind energy is an effective way for modeling and problem-solving, as it saves expensive experimental cost.

In the first part of this dissertation, first, we investigate the applicability of three different turbulence models, *Direct Numerical Simulation (DNS)*, *Large Eddy Simulation (LES)*, and *Reynolds-averaged Navier-Stokes (RANS)* on 2D and 3D lid-driven flow [6, 18] by using OpenFOAM. Second, we use turbulence model RANS k-ε to calculate and compare the wake development of one rotating turbine with fixed angular velocity by FLUENT. Third, we compare the resulting angular velocity of one wind-driven rotating turbine by using OpenFOAM and FLUENT. Last, we examine the wake interaction of two in-line rotating turbines by using FLUENT.

In addition to natural resources issues, numerical simulation can also help us comprehend and control man-made disasters. The problem was motivated by the air incident, in March 2014, Malaysia Airlines Flight MH370 disappeared less than an hour after take-off on a flight from Kuala Lumpur to Beijing. The Boeing 777-200ER carried 12 crew members and 227 passengers. On March 24 the Malaysian Prime Minister announced that "It is therefore with deep sadness and regret that I must inform you that ... Flight MH370 ended in the Southern Indian Ocean." Though the exact fate of Flight MH370 remains undetermined, the available evidence indicates a crash into the ocean. However, disturbing as this is, not all emergency water landings, referred to as *"ditching"* when they are controlled, end in tragedy. In the "Miracle on the Hudson", on January 15, 2009, Capt. Chelsey B. "Sully" Sullenberger and his crew successfully ditched US Airways Flight 1549, an Airbus A320-200, in the Hudson River after a loss of power due to a bird strike on take-off from La Guardia Airport. There was no loss of life. Another terrible tragic air incident happened in March 24, 2015. Germanwings Flight 9525, an Airbus A320-200, crashed into the French Alps during the flight from Barcelona to D*ü*sseldorf. The crash was intentionally caused by the co-pilot who locked the captain out of the cockpit during the flight and began a descent that caused this tragedy. All 144 passengers and 6 crew members were killed in this accident.

Aircraft crashworthiness and human survivability are of utmost concerns in any emergency landing situation. The earth is covered 71% by water and many major airports are situated oceanside. Assume that an aircraft did not have a mid-air explosion and all available signs indicate that it crashed somewhere in the ocean, this is an aircraft water-entry problem. Concerned with Germanwings Flight 9525 crash incident in March 2015, numerical simulations of the crashing of the Airbus A320 into a wall by using Computational Structural Dynamics (CSD) software ANSYS Explicit Dynamics and LS-DYNA can help us understand the physical mechanisms at work and also to improve passenger safety. It also shows how computational mathematics and mechanics can help us understand the physical nature of an aircraft crash, how to model and compute it, and how this knowledge is helping safe civil aviation and other aerospace related undertakings.

In the second part of this dissertation, first, we discuss the water entry problem. Second, we simulate the impact and damage of an aircraft, Airbus A320, crashing into a wall by using both ANSYS Explicit Dynamics and LS-DYNA.

2. INTRODUCTION TO OPENFOAM SOFTWARE AND USAGE

2.1 Introduction

In this Section, we introduce the *OpenFOAM* software. There is a revolution going on, impacting and transforming how computational mechanics and the associated design and optimization are done: the emergence, availability, and large-scale use of *OpenFOAM* [20]. It belongs to the contemporary *open-source trend* not unlike the roles played by the Linux operating system or the Internet encyclopedia Wikipedia. OpenFOAM is free and is used by thousands of people worldwide in both academic and industrial settings. The acronym OpenFOAM stands for *Open Source Field Operation and Manipulation.*

OpenFOAM was born in the strong British tradition of fluid dynamics research, specifically at The Imperial College, London, which has been a center of CFD research since the 1960's. The original development of OpenFOAM was begun by Prof. David Gosman and Dr. Radd Issa, with principal developers Mr. Henry Weller and Dr. Hrvoje Jasak. It was based on the *finite volume method (FVM)* [28], an idea to use C++ and object-oriented programming to develop a syntactical model of *equation mimicking* (see Table 2.2), and scalar-vector-tensor operations. A large number of PhD students and their theses have contributed to the project. Weller and Jasak founded the company Nabla Ltd., but it was not successful in marketing its product FOAM (the predecessor of OpenFOAM) and folded in 2004. Weller founded OpenCFD Ltd. in 2004 and released the GNU general public license of OpenFOAM software. OpenFOAM constitutes a C^{++} CFD toolbox for customized numerical solvers (over 60 of them) that can perform simulations of *basic CFD*, *combustion*, *turbulence modeling*, *electromagnetics*, *heat transfer*, *multiphase flow*, *stress analysis*, and even *financial mathematics* modeled by the *Black-Scholes equation.* In August 2011, OpenCFD was acquired by Silicon Graphics International (SGI). In September 2012, SGI sold OpenCFD Ltd to the ESI Group.

The revenue and survival strategy of the company, OpenCFD Ltd. (which has been absorbed into ESI Group), is a "Redhat model" [40] by providing support, training, and consulting services. While OpenFOAM is open-source, the development model is a "*cathedral*" style [41] where code contributions from researchers are not accepted back into the main distribution due to strict control of the code base. For researchers who want to distribute their developments, and find other online documentation, there are a community-oriented discussion forum [19], a wiki [1], and an international summer workshop [22].

Now, with the open-source libraries in OpenFOAM, one does not have to spend one's whole career writing CFD codes, or be forced to buy commercial softwares. Many other users of OpenFOAM have developed relevant libraries and solvers that are either posted online or may be requested for free. The number of OpenFOAM users has been steadily increasing. It is now estimated to be of the order of many thousands, with the majority of them being engineers in Europe. But the U.S. is catching up.

2.2 A Sketch of How to Use OpenFOAM

For beginners who are enthusiastic to learn how to use OpenFOAM to obtain CFD solutions, the best way is to study the many *tutorial examples* available in [20]. One such tutorial is the lid-driven cavity case [6]. (Such a case will also be computed in Cases 1 and 2 in Section 5.) It provides nearly all information *from start to finish* as to how to use OpenFOAM, including *pre-processing, solving* (i.e., how to run the codes) and *post-processing*. The structure of OpenFOAM could be seen in Chart. 2.1. The tutorial has a couple of dozen pages. If the beginner can get some help from an experienced OpenFOAM user, then it usually takes only a few weeks to run a simple OpenFOAM computer program for this problem.

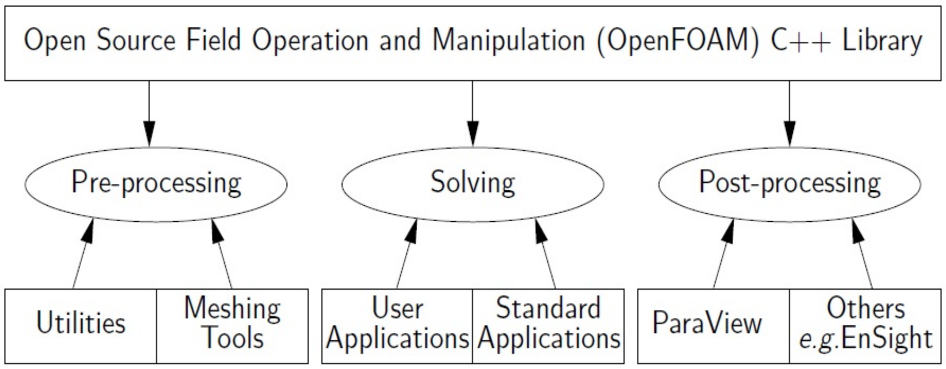


Chart 2.1: The structure of OpenFOAM.

As most of the readers may not necessarily be interested in running OpenFOAM codes for now, in this section we will mainly give some brief sketch. We first illustrate this for a simple elliptic boundary value problem

$$\begin{cases} \nabla^2 u(x,y,z) = f(x,y,z), \quad \text{on } \Omega \subseteq \mathbb{R}^3, \\ u(x,y,z) = g(x,y,z), \quad \text{on the boundary } \partial\Omega. \end{cases} \tag{2.1}$$

In OpenFOAM, one can use a Laplacian solver in the heat transfer library to obtain numerical solutions. By using the C^{++} language, in OpenFOAM Eq. $(2.1)_1$ is written

Table 2.1: OpenFOAM code for the potential equation (2.1).

```
// define field scalar u and f
volVectorField u, f;
// construct the Laplacian equation and solve it
solve
(
fvm::laplacian(u) == f
);
```

as in Table 2.1.

Note that the inhomogeneous Dirichlet boundary condition, given in $(2.1)_2$, will be prescribed elsewhere, in the "time directories" in the Case Directory Structure as shown in Chart 2.2. If, instead of $(2.1)_2$, we have inhomogeneous Neumann or Robin boundary conditions such as

$$\begin{aligned}\frac{\partial u(x,y,z)}{\partial n} &= g(x,y,z),\\ \frac{\partial u(x,y,z)}{\partial n} + \alpha u(x,y,z) &= g(x,y,z), \ \text{on } \partial\Omega,\end{aligned} \tag{2.2}$$

they can be specified similarly in the time directories.

To numerically solve a PDE by using OpenFOAM, a user needs to create a Case Directory Structure as shown in Chart 2.2. Normally, it contains three sub-directories. The user first gives a name for the <case>. The compositions of the various sub-directories are indicated in Chart 2.2.

Now, we look at the core case of this section, the incompressible Navier-Stokes

Chart 2.2: Case directory structure (adapted from [20]).

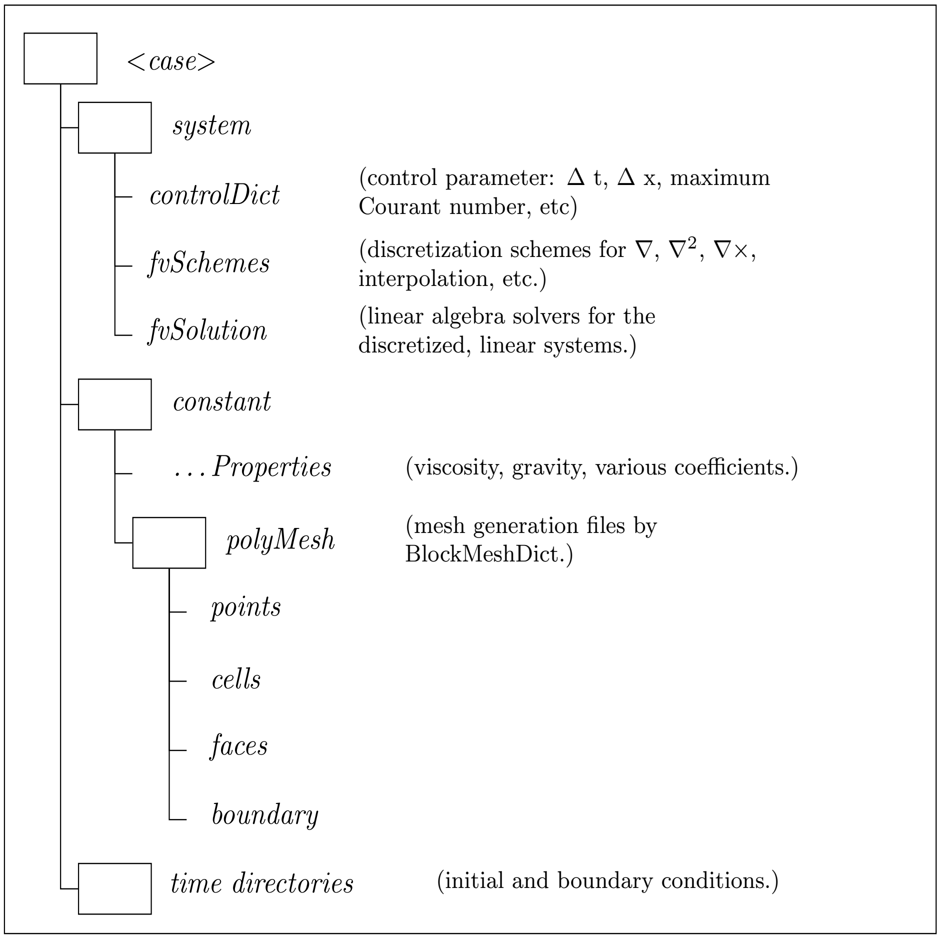

<case>
system
controlDict
(control parameter: Δ t, Δ x, maximum Courant number, etc)
fvSchemes
(discretization schemes for ∇, ∇^2, $\nabla\times$, interpolation, etc.)
fvSolution
(linear algebra solvers for the discretized, linear systems.)
constant
. . . Properties
(viscosity, gravity, various coefficients.)
polyMesh
(mesh generation files by BlockMeshDict.)
points
cells
faces
boundary
time directories
(initial and boundary conditions.)

Table 2.2: OpenFOAM code for the N-S equation (2.3). Here as well as in Table 2.1, *equation mimicking* is quite obvious. Note that the specifications fvm and fvc are selected by the user from the fvSchemes dictionary in the *system* dictionary, cf. Chart 2.1. Here fvm::laplacian means an *implicit* finite volume discretization for the Laplacian operator, and similarly for fvm:: div for the divergence operator. On the other hand, fvc::grad means an *explicit* finite volume discretization for the gradient operator. The parentheses (,) means a product of the enclosed quantities, including tensor products.

```
Solve
(
fvm::ddt(rho,U)
+ fvm::div(U,U)
- fvm::laplacian(mu,U)
==
- fvc::grad(p)
+ f
);
```

(N-S) equations in CFD. The governing equations are

$$\frac{\partial(\rho\mathbf{u})}{\partial t} + \nabla\cdot(\rho\mathbf{u}\mathbf{u}) - \mu\nabla^2\mathbf{u} = -\nabla p + \mathbf{f}(x,y,z,t), \tag{2.3}$$

$$\nabla\cdot\mathbf{u} = 0. \tag{2.4}$$

Note that in (2.2), $\mathbf{uu}$ is defined to be the 3×3 matrix

$$\mathbf{uu} = [u_i u_j]_{3\times 3}.$$

One can specify the given initial and boundary conditions on $\mathbf{u}$ in the "time directories" of Chart 2.2.

An effective algorithm for solving the coupled system (2.3) and (2.4) is the PISO (pressure-implicit with splitting of operators) algorithm of Issa [13]; see also [39]. In

OpenFOAM, basically, Eq. (2.3) is written as shown in Table 2.2.

With some details, the PISO algorithm is implemented in OpenFOAM as shown in Table 2.3. This (largely) takes care of the *equation solving* step.

For *pre-processing* involving *mesh generation*, one can use the utility *blockMesh*, supplied in OpenFOAM, to first generate a rectangular mesh for a cubic domain. The input data consists of coordinates of 8 vortices of the cube and numbers of cells in each direction, (n_x, n_y, n_z). The output is a rectangular mesh containing $n_x \times n_y \times n_z$ cells. In case of more complicated geometry, one can use either the *snappyHexMesh* utility or third-party packages such as Gambit meshing software [17], with subsequent conversion into OpenFOAM format.

Finally, for *post-processing*, to produce graphical output [23], OpenFOAM uses an open-source, multi-platform data analysis and visualization application called *ParaView* [38]. Alternatively, one can also use third party commercial products such as *EnSight* [12].

As opposed to a monolithic solver as is typically seen in commercial software, *pisoFoam* is one of 76 *standard solvers* that are included in the OpenFOAM distribution. These solvers are tailored to specific physics in the broad categories of combustion, compressible flow, discrete methods, electromagnetics, financial, heat transfer, incompressible flow, Lagrangian particle dynamics, multiphase flow, and stress analysis. There are also 80+ *standard utilities* for pre- and post-processing of data, parallel computing, and mesh creation and manipulation. For all of these different programs, the burden is on the user to verify that the implemented physics and models match their needs and intended application.

Table 2.3: The OpenFOAM code to solve the N-S equation of incompressible fluid. Note that the codes from lines (a) to (b) implement the PISO algorithm [13, 39].

```
//define field vector fluid velocity u and f, face flux phi, and pressure p
volVectorField u, f;
volScalarField p;
surfaceScalarField phi;
//define constant parameter fluid dynamical viscosity nu
scalarField nu;
//construct the fluid velocity equation
fvVectorMatrix UEqn (
fvm::ddt(u) + fvm::div(phi, u) - fvm::laplacian(nu, u) - f )                (a)
//solve the momentum equation using explicit pressure
solve (
UEqn == -fvc::grad(p) )
//predict the intermediate fluid velocity to calculate face flux
volVectorField rUA = 1.0/UEqn.A();
u = rUA *UEqn.H();
phi = fvc::interpolate(u) & mesh.Sf();
//construct the pressure equation using the constraint from continuity equation
fvScalarMatrix pEqn (
fvm::laplacian(rUA,p) == fvc::div(phi) )
pEqn.solve();
//correct the fluid velocity by the post-solve pressure and update face flux
u = u - rUA*fvc::grad(p);
phi = phi - pEqn.flux();                                                    (b)
```

2.3 Problem Description

This CFD model contains the geometry of the Horizontal axis wind turbine (HAWT) where a wind turbine consisting of three blades with 50 m radius. The physical setup is provided by Dr. Alain Perronnet. The computational domain contains the rotor region and an outer cylinder. It extends from approximately 1.87 times blades radius upwind to 7.22 times blades radius downwind of this wind turbine. A uniform incoming velocity 8 m/s is prescribed for the wind and a uniform angular velocity of the wind turbine is set 75 deg/s. The wind direction is exactly perpendicular to the plane of the turbine blades. The main grids were unstructured with polyhedral mesh elements. The total grid consists of approximately 0.7 million cells for the computational domain. The geometry of our model could be seen in Fig. 2.1.

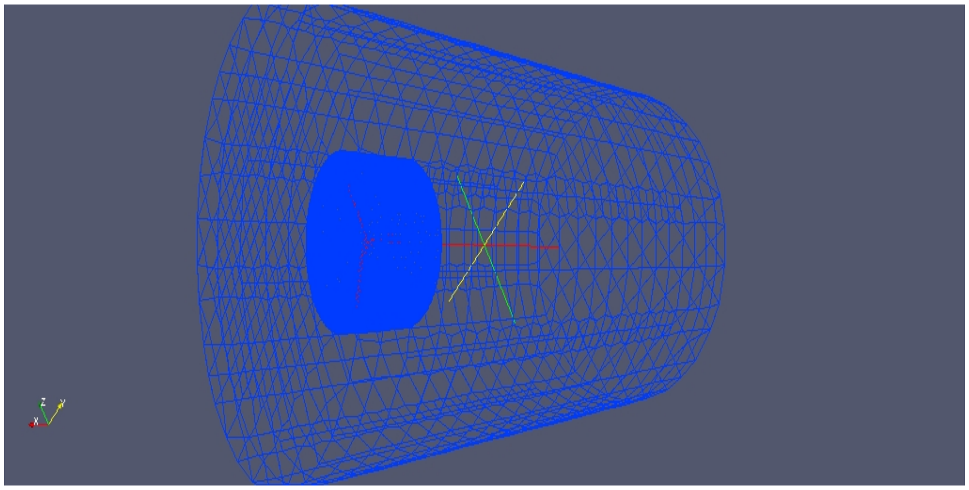


Figure 2.1: Geometry of the wind turbine.

2.3.1 Problem Setup and Configurations

In the following, we will describe how to pre-process, run, and post-process our case by OpenFOAM. For our case, we use **pimpleDyMFoam** solver which is a transient solver for incompressible flow of Newtonian fluids on a moving mesh by using the *PIMPLE* (*merged PISO-SIMPLE*) *algorithm* to solve it.

2.3.2 Pre-Processing

2.3.3 Mesh Generation

In OpenFOAM, there are two utilities which are used to create meshes. One is the **blockMesh** utility which can be used to generate simple meshes of blocks containing hexahedral cells and the other one is the **snappyHexMesh** utility which is for generating complex 3-D meshes consisting of hexahedral and split-hexahedral cells from triangulated surface geometries in Stereolithography (STL) format. The domain of one wind turbine case consists of one block, one outer cylinder, and one propeller surrounded by the smaller cylinder. We are going to use **blockMesh** utility to make the background mesh for the block and **snappyHexMesh** utility to make the mesh for the wind turbine and cylinders. We must build the external boundary to create the background mesh by using **blockMesh** before executing **snappyHexMesh**.

The mesh of outside block in our domain is made by the **blockMesh** utility. The steps of the **blockMesh** utility creating the mesh are reading a file called ***blockMeshDict*** which is in the folder *polyMesh* under the directory *constant*, generating the mesh, and writing out the data ***points***, ***pointZones***, ***pointLevel***, ***faces***, ***faceZones***, ***cellLevel***, ***cellZones***, ***neighbour***, ***owner***, and ***boundary*** under the same folder.

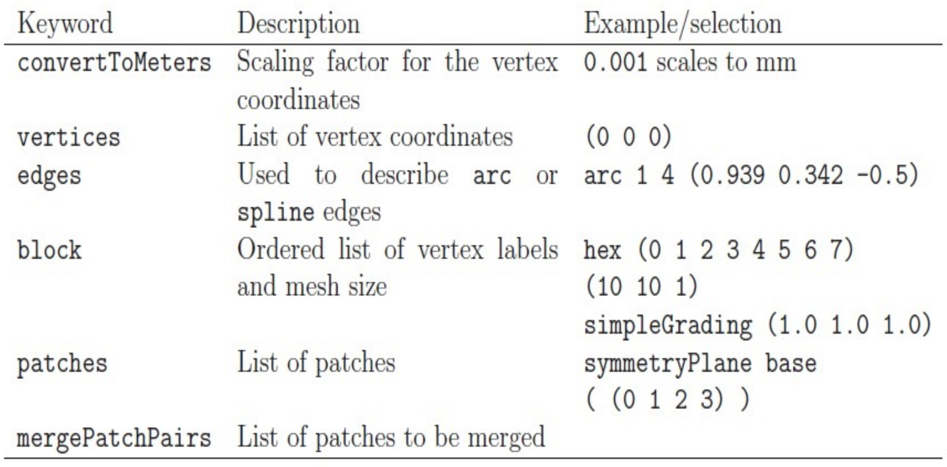
The ***blockMeshDict*** entries for this case can be seen as follows.

```
convertToMeters 445.455;
vertices((-0.81 -0.3 -0.3)
         (-0.81 0.3 -0.3)
         (0.21 0.3 -0.3)
         (0.21 -0.3 -0.3)
         (-0.81 -0.3  0.3)
         ( -0.81 0.3  0.3)
         ( 0.21 0.3   0.3)
         (0.21 -0.3   0.3));
blocks( hex ( 4 5 6 7 0 1 2 3) (12 20 12) simpleGrading (1 1 1));
edges ( );
boundary ( walls{type wall;
        faces((1 5 6 2)
              (1 2 3 0)
              (3 7 4 0)
             (7 6 5 4));}
inlet{ type patch;
        faces
        ((2 6 7 3));}
    outlet
    {type patch;
faces((0 4 5 1));});
```

The description of important keywords in the ***blockMeshDict*** file can be seen in Chart 2.3.

In our case, we generate the background mesh by subdividing a rectangular par-

Chart 2.3: The description of keywords in the ***blockMeshDict*** file (adapted from [21]).



Keyword	Description	Example/selection
`convertToMeters`	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) )`
`mergePatchPairs`	List of patches to be merged	

allelepiped 449.9.955 × 267.273 × 267.273 in 108 × 44 × 44 parts. It has 209088 hexahedra cells.

To run the **blockMesh** utility, we type **blockMesh** in the terminal within the case directory. The running status of the case will be reported in the terminal. The **checkMesh** utility could be used to check the validity of the mesh after we generate it.

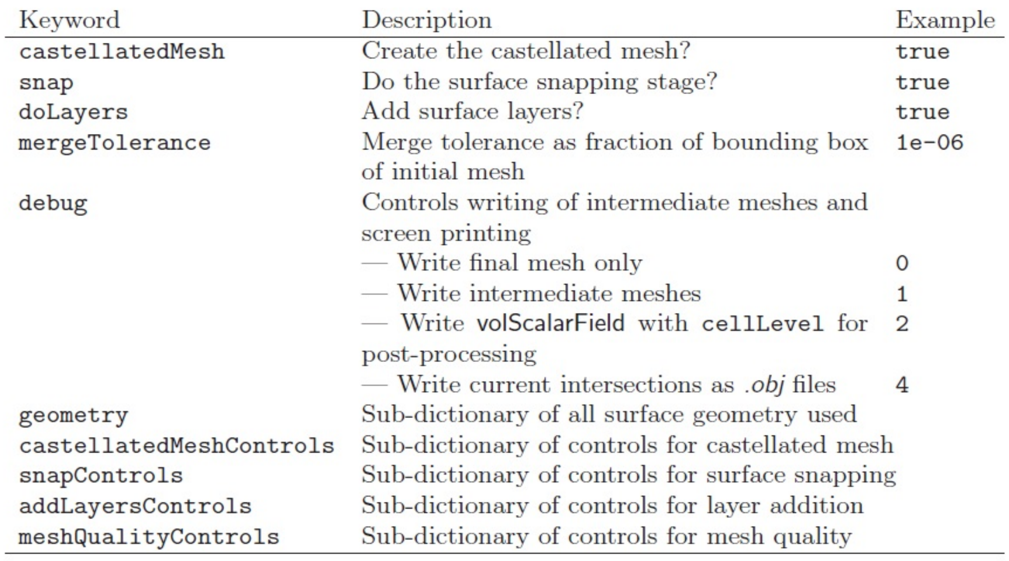
For our case, the geometry of the outer cylinder and one wind turbine surrounded by a smaller cylinder are specified through *obj* files in the folder *triSurface* under the directory *constant*. The mesh of the wind turbine system is made by the **snappyHexMesh** utility. The steps of the **snappyHexMesh** utility creating the mesh are generating initial mesh, splitting cells by feature edges, splitting cells by surface, refining regions, removing cells, splitting cells by regions, snapping surface, and adding layers.

The ***snappyHexMeshDict*** entries for this case can be seen as follows.

```
castellatedMesh true;
snap            true;
addLayers       false;

geometry
{ innerCylinder.obj
    { type        triSurfaceMesh;
      name        innerCylinder;
      regions
        {ascii { name       innerCylinder; }}}
    innerCylinderSmall.obj
    { type        triSurfaceMesh;
      name        innerCylinderSmall;
      regions
        {ascii { name       innerCylinderSmall;}}}
    outerCylinder.obj
    { type        triSurfaceMesh;
      name        outerCylinder;
      regions
        {ascii{ name       outerCylinder;}}}
    propellerTip.obj
    { type        triSurfaceMesh;
      name        propellerTip;
      regions
        {ascii{ name       propellerTip;}}}};
```

```
castellatedMeshControls
{ features
    (   { file        "innerCylinderSmall.eMesh";
          level       4;}
        { file        "outerCylinder.eMesh";
          level       0;}
        { file        "propellerTip.eMesh";
          level       4;});
  refinementSurfaces
    { innerCylinder
    {  level       (2 3);
           cellZone    innerCylinder;
           faceZone    innerCylinder;
           cellZoneInside  inside;}
      innerCylinderSmall
        {  level       (4 4);
           cellZone    innerCylinderSmall;
           faceZone    innerCylinderSmall;
           cellZoneInside  inside;}
        outerCylinder
        {  level       (0 0);}
        propellerTip
        {  level       (4 6);} }

    resolveFeatureAngle 30;
    refinementRegions
    { innerCylinder
        { mode        inside;
          levels      ((1E15 3));}
        innerCylinderSmall
        { mode        inside;
          levels      ((1E15 4));}
        outerCylinder
        { mode        inside;
          levels      ((1E15 0));} }

    locationInMesh (-350 0 0);
    allowFreeStandingZoneFaces true;}

snapControls
    {nSmoothPatch 3;
     tolerance 4.0;
     nSolveIter 300;
     nRelaxIter 5;
     nFeatureSnapIter 20; }

addLayersControls
{ relativeSizes true;
  layers { }
  expansionRatio 1.0;
  finalLayerThickness 0.3;
  minThickness 0.1;

  nGrow 0;

  featureAngle 30;

  nRelaxIter 3;

  nSmoothSurfaceNormals 1;

  nSmoothNormals 3;

  nSmoothThickness 10;

  Stop layer growth on highly warped cells

  maxFaceThicknessRatio 0.5;

  maxThicknessToMedialRatio 0.3;

  minMedianAxisAngle 90;

  nBufferCellsNoExtrude 0;

  nLayerIter 50;}


meshQualityControls

{ maxNonOrtho 65;

  maxBoundarySkewness 20;

  maxInternalSkewness 4;

  maxConcave 80;

  minVol 1e-13;

  minTetQuality -1;

  minArea -1;

  minTwist 0.01;

  minDeterminant 0.001;

  minFaceWeight 0.05;

  minVolRatio 0.01;
```

Chart 2.4: The description of keywords in the ***snappyHexMeshDict*** file (adapted from [21]).



Keyword	Description	Example
`castellatedMesh`	Create the castellated mesh?	`true`
`snap`	Do the surface snapping stage?	`true`
`doLayers`	Add surface layers?	`true`
`mergeTolerance`	Merge tolerance as fraction of bounding box of initial mesh	`1e-06`
`debug`	Controls writing of intermediate meshes and screen printing	
	— Write final mesh only	`0`
	— Write intermediate meshes	`1`
	— Write volScalarField with `cellLevel` for post-processing	`2`
	— Write current intersections as *.obj* files	`4`
`geometry`	Sub-dictionary of all surface geometry used	
`castellatedMeshControls`	Sub-dictionary of controls for castellated mesh	
`snapControls`	Sub-dictionary of controls for surface snapping	
`addLayersControls`	Sub-dictionary of controls for layer addition	
`meshQualityControls`	Sub-dictionary of controls for mesh quality	

```
minTriangleTwist -1;

nSmoothScale 4;

errorReduction 0.75;

relaxed { maxNonOrtho 75;}}


debug 0;

mergeTolerance 1e-6;
```

The description of important keywords in the ***snappyHexMeshDict*** file can be seen in Chart 2.4.

We could run **snappyHexMesh** on this ***snappyHexMeshDict*** file by typing **snappyHexMesh** in the terminal within the case directory.

2.3.4 Boundary and Initial Conditions

After **blockMesh** and **snappyHexMesh** complete mesh generation, the details of boundary geometry can be viewed by the ***boundary*** file in the *polyMesh* folder under the *constant* directory. Hence, we can examine the type and number of faces for the geometry. For our case, the boundary conditions can be seen as follows:

```
(inlet
       { type             patch;
         nFaces           268;
         startFace        1658757;}
 outlet
    {   type             patch;
        nFaces           112;
        startFace        1659025;}
 outerCylinder
    {   type             wall;
        nFaces           976;
        startFace        1659137;}
    propellerTip
    {   type             wall;
        nFaces           5741;
        startFace        1660113;}
    AMI1
    {   type             cyclicAMI;
        nFaces           22632;
        startFace        1667522;
        matchTolerance  0.0001;
        neighbourPatch  AMI2;
        transform       noOrdering;}
    AMI2
    {   type            cyclicAMI;
        nFaces          22632;
        startFace       1690154;
        matchTolerance  0.0001;
        neighbourPatch  AMI1;
        transform       noOrdering;})
```

The initial field data is saved in a ***0*** directory under the case directory since our case is set up to start at the beginning time $t = 0\ s$. There are two files in ***0*** directory, pressure ($\boldsymbol{p}$) and velocity ($\boldsymbol{U}$). The initial values and boundary conditions for pressure and velocity fields of regions can be set in these two files.

The $\boldsymbol{p}$ entries for this case are as follows:

```
dimensions      [0 2 -2 0 0 0 0];
internalField   uniform 0;
boundaryField
{inlet
    {type           zeroGradient;}
 outlet
    {type           fixedValue;
     value          uniform 0;}
outerCylinder
    {type           zeroGradient;}
"propeller.*"
    {type           zeroGradient;}
AMI1
    {type           cyclicAMI;
     value          uniform 0;}
AMI2
    {type           cyclicAMI;
     value          uniform 0;}}
```

At the beginning of these two files, there are dimensional units. The format for a *dimensionSet* is 7 scalars with square brackets. The base units for the Système International (SI) and the United States Customary System (USCS) can be seen in Chart 2.5. The description of important keywords in the $\boldsymbol{p}$ and $\boldsymbol{U}$ field data files can be seen in Chart 2.6. For boundary field, a boundary is normally divided into a set of *patches*, and one or more area of the boundary surface can be included in one patch, even those areas are not connected to each other. There are three main type of patches: *Base type* which type of patch only describes the geometry, *Primitive type* which type of patch is assigned the numerical condition to a filed variable on, and *Derived type* which type of patch is a complicated patch condition and derived from Primitive type. The examples of these three type of patches can be seen in Chart **??**. The description of the basic, primitive, and derived patch field types can be viewed in Chart 2.8, Chart 2.9, and Chart 2.10, respectively.

For our case, the boundary consists of inlet, outlet, outer cylinder, and propeller. Both intern fields are set to be uniform. For inlet, propellers, and camels, all are given a **zeroGradient** boundary condition in $\boldsymbol{p}$, and this means that the normal gradient of pressure is zero. The **fixedValue** condition with a value of uniform 0 is

Chart 2.5: Base units for Système International SI and the United States Customary System USCS (adapted from [21]).

No.	Property	SI unit	USCS unit
1	Mass	kilogram (kg)	pound-mass (lbm)
2	Length	metre (m)	foot (ft)
3	Time	— — — — second	(s) — — — —
4	Temperature	Kelvin (K)	degree Rankine (°R)
5	Quantity	kilogram-mole (kgmol)	pound-mole (lbmol)
6	Current	— — — — ampere	(A) — — — —
7	Luminous intensity	— — — — candela	(cd) — — — —

Chart 2.6: The description of keywords in the ***p*** and ***U*** field data files (adapted from [21]).

Keyword	Description	Example
`dimensions`	Dimensions of field	`[1 1 -2 0 0 0 0]`
`internalField`	Value of internal field	`uniform (1 0 0)`
`boundaryField`	Boundary field	

Chart 2.7: The examples of three type of patches (adapted from [21]).

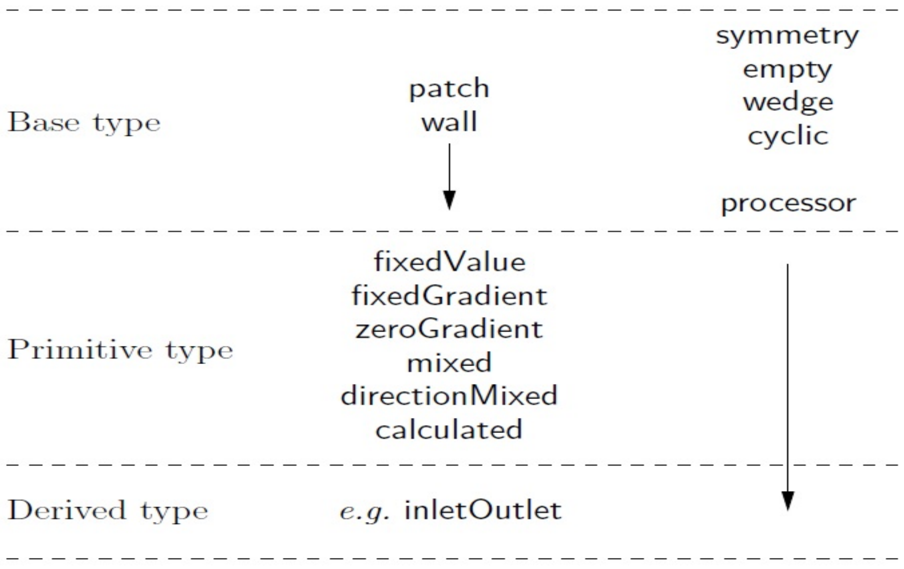


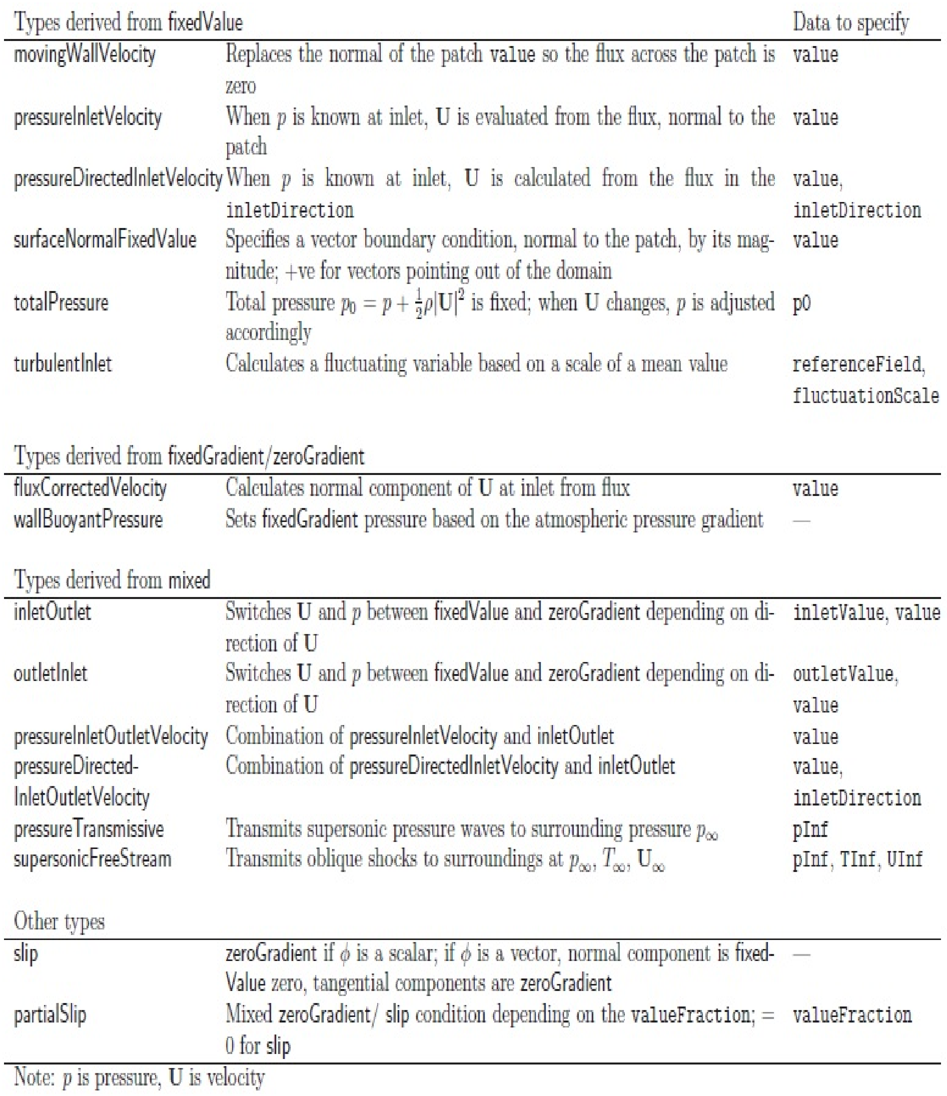
Chart 2.8: The description of the basic patch field types (adapted from [21]).

Selection Key	Description
`patch`	generic patch
`symmetryPlane`	plane of symmetry
`empty`	front and back planes of a 2D geometry
`wedge`	wedge front and back for an axi-symmetric geometry
`cyclic`	cyclic plane
`wall`	wall — used for wall functions in turbulent flows
`processor`	inter-processor boundary

Chart 2.9: The description of the primitive patch field types (adapted from [21]).

Type	Description of condition for patch field ϕ	Data to specify
fixedValue	Value of ϕ is specified	`value`
fixedGradient	Normal gradient of ϕ is specified	`gradient`
zeroGradient	Normal gradient of ϕ is zero	—
calculated	Boundary field ϕ derived from other fields	—
mixed	Mixed fixedValue/ fixedGradient condition depending on the value in `valueFraction`	`refValue`, `refGradient`, `valueFraction`, `value`
directionMixed	A mixed condition with tensorial `valueFraction`, *e.g.* for different levels of mixing in normal and tangential directions	`refValue`, `refGradient`, `valueFraction`, `value`

Chart 2.10: The description of the derived patch field types (adapted from [21]).



Types derived from fixedValue		Data to specify
movingWallVelocity	Replaces the normal of the patch `value` so the flux across the patch is zero	`value`
pressureInletVelocity	When p is known at inlet, $\mathbf{U}$ is evaluated from the flux, normal to the patch	`value`
pressureDirectedInletVelocity	When p is known at inlet, $\mathbf{U}$ is calculated from the flux in the `inletDirection`	`value`, `inletDirection`
surfaceNormalFixedValue	Specifies a vector boundary condition, normal to the patch, by its magnitude; +ve for vectors pointing out of the domain	`value`
totalPressure	Total pressure $p_0 = p + \frac{1}{2}\rho\lvert\mathbf{U}\rvert^2$ is fixed; when $\mathbf{U}$ changes, p is adjusted accordingly	`p0`
turbulentInlet	Calculates a fluctuating variable based on a scale of a mean value	`referenceField`, `fluctuationScale`
Types derived from fixedGradient/zeroGradient		
fluxCorrectedVelocity	Calculates normal component of $\mathbf{U}$ at inlet from flux	`value`
wallBuoyantPressure	Sets fixedGradient pressure based on the atmospheric pressure gradient	—
Types derived from mixed		
inletOutlet	Switches $\mathbf{U}$ and p between fixedValue and zeroGradient depending on direction of $\mathbf{U}$	`inletValue`, `value`
outletInlet	Switches $\mathbf{U}$ and p between fixedValue and zeroGradient depending on direction of $\mathbf{U}$	`outletValue`, `value`
pressureInletOutletVelocity	Combination of pressureInletVelocity and inletOutlet	`value`
pressureDirected-InletOutletVelocity	Combination of pressureDirectedInletVelocity and inletOutlet	`value`, `inletDirection`
pressureTransmissive	Transmits supersonic pressure waves to surrounding pressure p_∞	`pInf`
supersonicFreeStream	Transmits oblique shocks to surroundings at p_∞, T_∞, $\mathbf{U}_\infty$	`pInf`, `TInf`, `UInf`
Other types		
slip	zeroGradient if ϕ is a scalar; if ϕ is a vector, normal component is fixedValue zero, tangential components are zeroGradient	—
partialSlip	Mixed zeroGradient/ slip condition depending on the `valueFraction`; = 0 for slip	`valueFraction`

Note: p is pressure, $\mathbf{U}$ is velocity assigned to outlet in $\boldsymbol{p}$. The wind comes from the inlet in the negative x-direction so we assign a **fixedValue** condition with a value of uniform (-8 0 0) to inlet and outer cylinder in $\boldsymbol{U}$. The outlet is given a **inletOutlet** boundary condition with a value of uniform (-8 0 0) in $\boldsymbol{U}$. The **movingWallVelocity** boundary condition is assigned to propeller with a value of uniform (0,0,0) in $\boldsymbol{U}$.

2.3.5 Setup of the Model

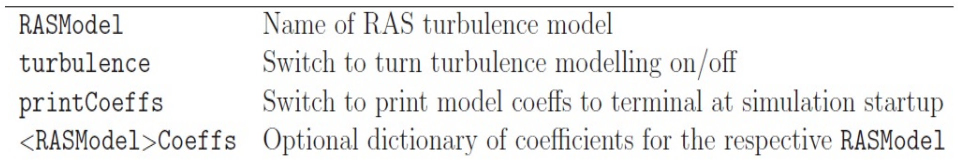
The properties of rotating motion, center of rotation and angular velocity for the propeller can be specified in the ***dynamicMeshDict*** file in *constant* directory.

The ***dynamicMeshDict*** entries for this case are as follows:

```
dynamicFvMesh   solidBodyMotionFvMesh;
motionSolverLibs ( "libfvMotionSolvers.so" );
solidBodyMotionFvMeshCoeffs
{
    cellZone        innerCylinderSmall;
    solidBodyMotionFunction  rotatingMotion;
    rotatingMotionCoeffs
    {
        CofG        (0 0 0);
        radialVelocity (-75 0 0); // deg/s
    }
}
```

For our case, we set the center of rotation to be (0,0,0) and the angular velocity for the propeller in the negative x-direction (-75,0,0).

Chart 2.11: The description of keywords in the ***RASProperties*** file (adapted from [21]).



`RASModel`	Name of RAS turbulence model
`turbulence`	Switch to turn turbulence modelling on/off
`printCoeffs`	Switch to print model coeffs to terminal at simulation startup
`<RASModel>Coeffs`	Optional dictionary of coefficients for the respective `RASModel`

2.3.6 Physical Properties

The physical properties, such as the viscosity of air and type of turbulence models can be set up in the ***transportProperties*** and ***turbulenceProperties*** files in the *constant* directory.

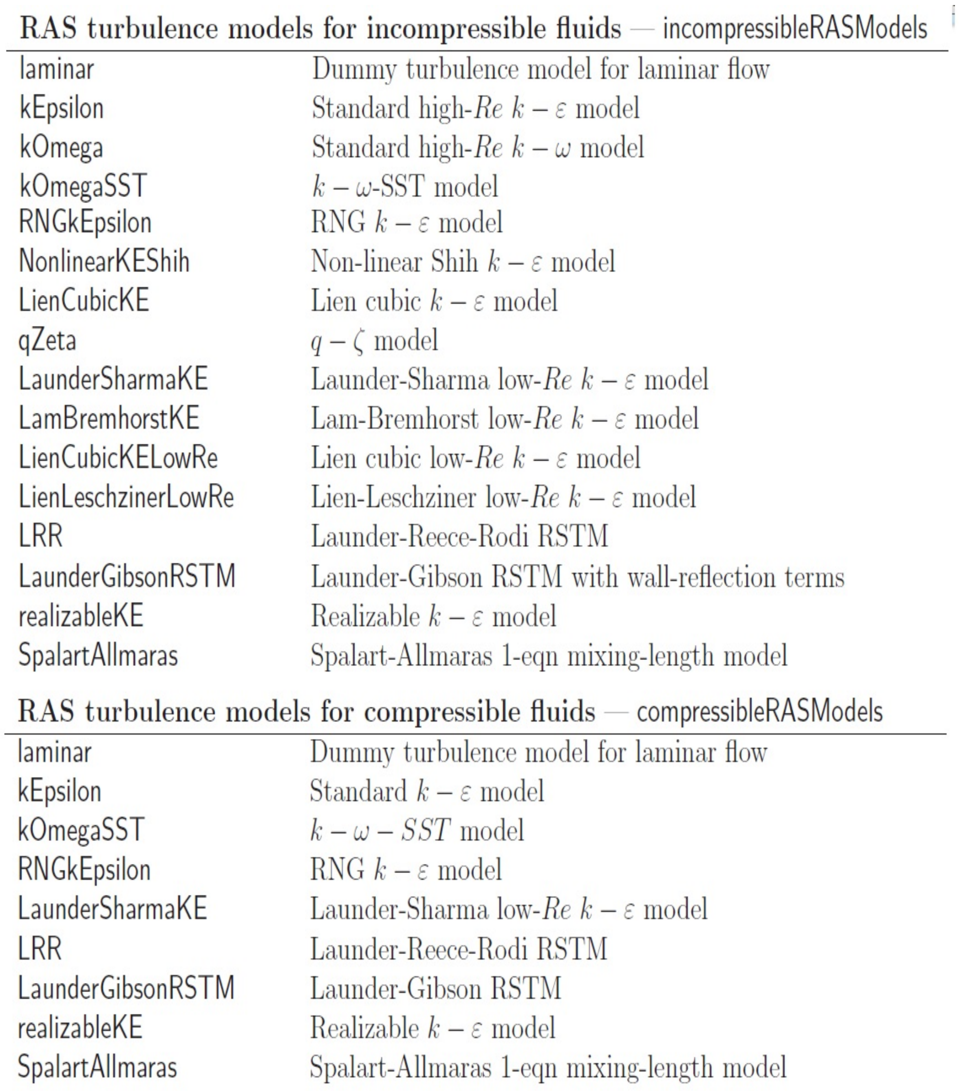
The ***transportProperties*** entries for this case are as follows:

```
transportModel  Newtonian;
nu              nu [ 0 2 -1 0 0 0 0 ] 1e-6;
```

ν is the kinematic viscosity of air and we use the default value 10^{-6} m^2/t.

In the ***turbulenceProperties*** file, there are three turbulence models we can use, *laminar* which does not use any turbulence models, *RASModel* which uses Reynolds-averaged stress (RAS) modeling, and *LESModel* which uses large eddy simulation (LES) modeling. If RASModel is selected, the options of RAS modeling is specified in the ***RASProperties*** file under the *constant* directory. The description of keywords in the ***RASProperties*** file can be seen in Chart 2.11. The library of all RAS models in OpenFOAM could be seen in Chart 2.12. If LESModel is selected, the choices of LES modeling is stored in the ***LESProperties*** file under the *constant* directory. The description of keywords in the ***LESProperties*** file can be seen in Chart 2.13. The library of all LES models in OpenFOAM could be seen in Chart **??**.

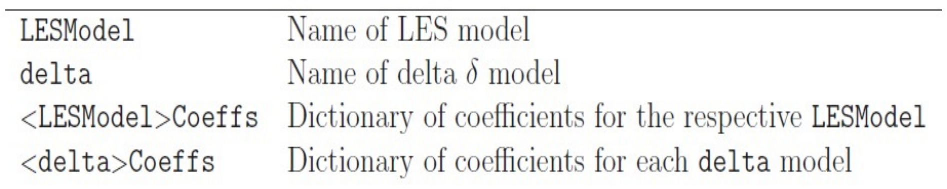
Chart 2.12: Libraries of RAS turbulence model (adapted from [21]).



RAS turbulence models for incompressible fluids — incompressibleRASModels	
laminar	Dummy turbulence model for laminar flow
kEpsilon	Standard high-*Re* $k-\varepsilon$ model
kOmega	Standard high-*Re* $k-\omega$ model
kOmegaSST	$k-\omega$-SST model
RNGkEpsilon	RNG $k-\varepsilon$ model
NonlinearKEShih	Non-linear Shih $k-\varepsilon$ model
LienCubicKE	Lien cubic $k-\varepsilon$ model
qZeta	$q-\zeta$ model
LaunderSharmaKE	Launder-Sharma low-*Re* $k-\varepsilon$ model
LamBremhorstKE	Lam-Bremhorst low-*Re* $k-\varepsilon$ model
LienCubicKELowRe	Lien cubic low-*Re* $k-\varepsilon$ model
LienLeschzinerLowRe	Lien-Leschziner low-*Re* $k-\varepsilon$ model
LRR	Launder-Reece-Rodi RSTM
LaunderGibsonRSTM	Launder-Gibson RSTM with wall-reflection terms
realizableKE	Realizable $k-\varepsilon$ model
SpalartAllmaras	Spalart-Allmaras 1-eqn mixing-length model

RAS turbulence models for compressible fluids — compressibleRASModels	
laminar	Dummy turbulence model for laminar flow
kEpsilon	Standard $k-\varepsilon$ model
kOmegaSST	$k-\omega-SST$ model
RNGkEpsilon	RNG $k-\varepsilon$ model
LaunderSharmaKE	Launder-Sharma low-*Re* $k-\varepsilon$ model
LRR	Launder-Reece-Rodi RSTM
LaunderGibsonRSTM	Launder-Gibson RSTM
realizableKE	Realizable $k-\varepsilon$ model
SpalartAllmaras	Spalart-Allmaras 1-eqn mixing-length model

Chart 2.13: The description of keywords in the ***LESProperties*** file (adapted from [21]).



`LESModel`	Name of LES model
`delta`	Name of delta δ model
`<LESModel>Coeffs`	Dictionary of coefficients for the respective `LESModel`
`<delta>Coeffs`	Dictionary of coefficients for each `delta` model

Chart 2.14: Libraries of LES turbulence model (adapted from [21]).

Large-eddy simulation (LES) filters — LESfilters	
laplaceFilter	Laplace filters
simpleFilter	Simple filter
anisotropicFilter	Anisotropic filter
Large-eddy simulation deltas — LESdeltas	
PrandtlDelta	Prandtl delta
cubeRootVolDelta	Cube root of cell volume delta
maxDeltaxyz	Maximum of x, y and z; for structured hex cells only
smoothDelta	Smoothing of delta
Incompressible LES turbulence models — incompressibleLESModels	
Smagorinsky	Smagorinsky model
Smagorinsky2	Smagorinsky model with 3-D filter
dynSmagorinsky	Dynamic Smagorinsky
homogenousDynSmag-orinsky	Homogeneous dynamic Smagorinsky model
dynLagrangian	Lagrangian two equation eddy-viscosity model
scaleSimilarity	Scale similarity model
mixedSmagorinsky	Mixed Smagorinsky/scale similarity model
dynMixedSmagorinsky	Dynamic mixed Smagorinsky/scale similarity model
kOmegaSSTSAS	$k-\omega$-SST scale adaptive simulation (SAS) model
oneEqEddy	k-equation eddy-viscosity model
dynOneEqEddy	Dynamic k-equation eddy-viscosity model
locDynOneEqEddy	Localised dynamic k-equation eddy-viscosity model
spectEddyVisc	Spectral eddy viscosity model
LRDDiffStress	LRR differential stress model
DeardorffDiffStress	Deardorff differential stress model
SpalartAllmaras	Spalart-Allmaras model
SpalartAllmarasDDES	Spalart-Allmaras delayed detached eddy simulation (DDES) model
SpalartAllmarasIDDES	Spalart-Allmaras improved DDES (IDDES) model
Compressible LES turbulence models — compressibleLESModels	
Smagorinsky	Smagorinsky model
oneEqEddy	k-equation eddy-viscosity model
dynOneEqEddy	Dynamic k-equation eddy-viscosity model
lowReOneEqEddy	Low-Re k-equation eddy-viscosity model
DeardorffDiffStress	Deardorff differential stress model
SpalartAllmaras	Spalart-Allmaras 1-eqn mixing-length model

Chart 2.15: The description of keywords in the ***fvSchemes*** file (adapted from [21]).

Keyword	Category of mathematical terms
`interpolationSchemes`	Point-to-point interpolations of values
`snGradSchemes`	Component of gradient normal to a cell face
`gradSchemes`	Gradient ∇
`divSchemes`	Divergence $\nabla\bullet$
`laplacianSchemes`	Laplacian ∇^2
`timeScheme`	First and second time derivatives $\partial/\partial t, \partial^2/\partial^2 t$
`fluxRequired`	Fields which require the generation of a flux

The ***turbulenceProperties*** entries for this case are as follows:

```
simulationType  RASModel;
```

The ***RASProperties*** entries for this case are as follows:

```
RASModel        kEpsilon;

turbulence      on;

printCoeffs     on;
```

For our incompressible case, we choose *kEpsilon* which is one of RAS turbulence model. We turn on turbulence model and printCoeffs switch in our case.

2.3.7 Discretisation Schemes and Linear-Solver Settings

The option of finite volume numerical discretisation schemes can be specified in the ***fvSchemes*** file under the *system* directory. The choice of linear equation solvers, tolerances and algorithms used in the solution can be made in the ***fvSolution*** file under the *system* directory. The description of main keywords in the ***fvSchemes*** file can be viewed in Chart 2.15.

The ***fvSchemes*** entries for this case are as follows:

```
ddtSchemes

{   default         Euler;}

gradSchemes

{   default         Gauss linear;

    grad(p)         Gauss linear;

    grad(U)         cellLimited Gauss linear 1;}

divSchemes

{   default         none;

    div(phi,U)      Gauss linearUpwind grad(U);

    div(phi,k)      Gauss upwind;

    div(phi,epsilon) Gauss upwind;

    div((nuEff*dev(T(grad(U))))) Gauss linear;}

laplacianSchemes

{   default         Gauss linear corrected;}

interpolationSchemes

{   default         linear;}

snGradSchemes

{   default         corrected;}

fluxRequired

{   default         no;

    pcorr           ;

    p               ;}
```

The ***fvSolution*** entries for this case are as follows:

```
solvers

{   pcorr
    {   solver          GAMG;
        tolerance       1e-2;
        relTol          0;
        smoother        DICGaussSeidel;
        cacheAgglomeration no;
        nCellsInCoarsestLevel 10;
        agglomerator    faceAreaPair;
        mergeLevels     1;
        maxIter         50;}
    p
    {   $pcorr;
        tolerance       1e-5;
        relTol          0.01;}
    pFinal
    {   $p;
        tolerance       1e-6;
        relTol          0;}
    "(U|k|epsion)"
    {   solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-6;
        relTol          0.1;}
    "(U|k|epsilon)Final"
    {   solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-6;
```

```
        relTol          0;}}
PIMPLE
{   correctPhi          no;
    nOuterCorrectors    3;
    nCorrectors         1;
    nNonOrthogonalCorrectors 0;}
relaxationFactors
{   "(U|k|epsilon).*"   1;}
cache
{   grad(U);}
```

2.3.8 Control

The time settings, such as start/end time and fixed/adjusted time step, and the form of data output can be set up in the ***controlDict*** file under the *system* directory. The description of main keywords in the ***fvSchemes*** file can be viewed in Chart 2.16.

The ***controlDict*** entries for this case are shown below:

```
application     pimpleDyMFoam;
startFrom       startTime;
startTime       0;
stopAt          endTime;
endTime         200;
deltaT          0.001;
writeControl    adjustableRunTime;
writeInterval   0.1;
purgeWrite      0;
```

Chart 2.16: The description of keywords in the ***controlDict*** file (adapted from [21]).

Time control	
`startFrom`	Controls the start time of the simulation.
- `firstTime`	Earliest time step from the set of time directories.
- `startTime`	Time specified by the `startTime` keyword entry.
- `latestTime`	Most recent time step from the set of time directories.
`startTime`	Start time for the simulation with `startFrom startTime;`
`stopAt`	Controls the end time of the simulation.
- `endTime`	Time specified by the `endTime` keyword entry.
- `writeNow`	Stops simulation on completion of current time step and writes data.
- `noWriteNow`	Stops simulation on completion of current time step and does not write out data.
- `nextWrite`	Stops simulation on completion of next scheduled write time, specified by `writeControl`.
`endTime`	End time for the simulation when `stopAt endTime;` is specified.
`deltaT`	Time step of the simulation.
Data writing	
`writeControl`	Controls the timing of write output to file.
- `timeStep`†	Writes data every `writeInterval` time steps.
- `runTime`	Writes data every `writeInterval` seconds of simulated time.
- `adjustableRunTime`	Writes data every `writeInterval` seconds of simulated time, adjusting the time steps to coincide with the `writeInterval` if necessary — used in cases with automatic time step adjustment.
- `cpuTime`	Writes data every `writeInterval` seconds of CPU time.
- `clockTime`	Writes data out every `writeInterval` seconds of real time.
`writeInterval`	Scalar used in conjunction with `writeControl` described above.
`purgeWrite`	Integer representing a limit on the number of time directories that are stored by overwriting time directories on a cyclic basis. Example of $t_0 = 5$s, $\Delta t = 1$s and `purgeWrite 2;`: data written into 2 directories, *6* and *7*, before returning to write the data at 8 s in *6*, data at 9 s into *7*, *etc.* *To disable the time directory limit, specify* `purgeWrite 0;`† For steady-state solutions, results from previous iterations can be continuously overwritten by specifying `purgeWrite 1;`
`writeFormat`	Specifies the format of the data files.
- `ascii`†	ASCII format, written to `writePrecision` significant figures.
- `binary`	Binary format.
`writePrecision`	Integer used in conjunction with `writeFormat` described above, 6† by default
`writeCompression`	Specifies the compression of the data files.
- `uncompressed`	No compression.†
- `compressed`	`gzip` compression.
`timeFormat`	Choice of format of the naming of the time directories.
- `fixed`	$\pm m.dddddd$ where the number of ds is set by `timePrecision`.
- `scientific`	$\pm m.dddddde{\pm}xx$ where the number of ds is set by `timePrecision`.
- `general`†	Specifies `scientific` format if the exponent is less than -4 or greater than or equal to that specified by `timePrecision`.
`timePrecision`	Integer used in conjunction with `timeFormat` described above, 6† by default
`graphFormat`	Format for graph data written by an application.
- `raw`†	Raw ASCII format in columns.
- `gnuplot`	Data in `gnuplot` format.
- `xmgr`	Data in `Grace/xmgr` format.
- `jplot`	Data in `jPlot` format.
Data reading	
`runTimeModifiable`	`yes`†/`no` switch for whether dictionaries, *e.g.controlDict*, are re-read by OpenFOAM at the beginning of each time step.
Run-time loadable functionality	
`libs`	List of additional libraries (on `$LD_LIBRARY_PATH`) to be loaded at run-time, *e.g.*`( "libUser1.so" "libUser2.so" )`
`functions`	List of functions, *e.g.* `probes` to be loaded at run-time; see examples in *$FOAM_TUTORIALS*

† denotes default entry if associated keyword is omitted.

```
writeFormat     binary;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

adjustTimeStep  yes;

maxCo           2;
```

2.4 Running the Code

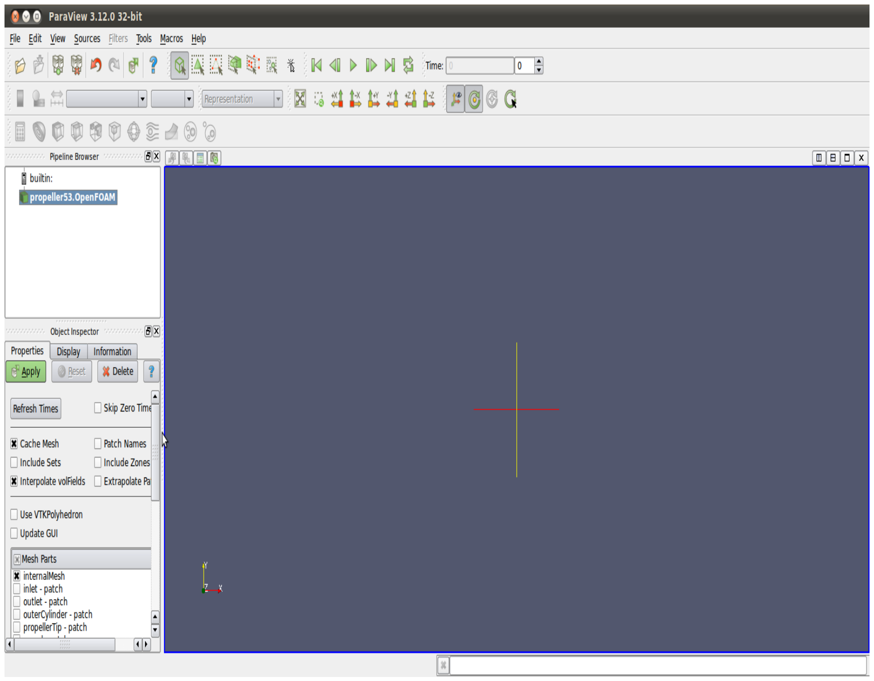
For our case, we use **pimpleDyMFoam** solver. To run the **pimpleDyMFoam** solver, we type **pimpleDyMFoam** in the terminal within the case directory. The progress of the case, such as the current time, initial and final residuals for all fields, and maximum Courant number, will be showed on the terminal window during the calculation.

2.5 Post-Processing

2.5.1 Settings of ParaView

The main post-processing utility used by OpenFOAM is *paraFoam* and it uses the software *ParaView* which is an open source visualisation application. To run the **paraFoam** utility, we type **paraFoam** in the terminal within the case directory. The panel of ParaView for our case will be showed (Fig. 2.2). After opening the panel of ParaView, we click our case under *Pipeline Browser* to make it hightlighted in blue and switch the eye button alongside our case on to show the graphs.

The **Properties** panel controls the input settings for the case, such as time



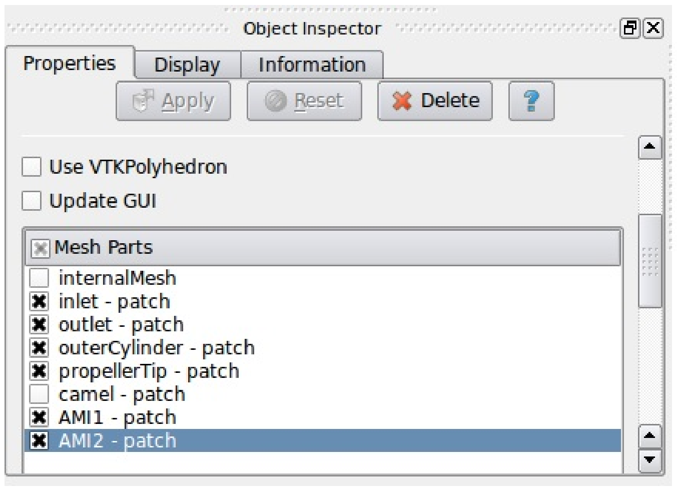
☐ Skip Zero Time
☒ Cache Mesh
☐ Patch Names
☐ Include Sets
☐ Include Zones
☒ Interpolate volFields
☐ Extrapolate Pa
☐ Use VTKPolyhedron
☐ Update GUI

Mesh Parts
☒ internalMesh
☐ inlet - patch
☐ outlet - patch
☐ outerCylinder - patch
☐ propellerTip - patch

Figure 2.2: The panel of ParaView.

steps, region status, and filed status. We enable the patches, **inlet**, **outlet**, **outer cylinder**, and **propeller**, in **Mesh Parts** which we want to show in the geometry and mesh graphs in the **Properties** panel (Fig. 2.3). We also enable the **p** and **U** fields in **Volume Fields** to display filed plots (Fig. 2.4). After the setups are done, we click **Apply** to finish the settings and display the results.

The **Display** panel (Fig. 2.5) controls the visual representation of the case, such as colors. In the **Display** panel, we select **U** and **Magnitude** from the **Color by** menu in the **Color** panel to make the plot of velocity. We could change the color scale of velocity by the settings in **Edit Color Map...**. In the **Color Scale** panel of the **Color Scale Editor** window (Fig. 2.6), we click **Choose Preset**. In the **Choose Preset** panel (Fig. 2.7), we could choose common color scale **Blue to Red**



☐ Use VTKPolyhedron
☐ Update GUI

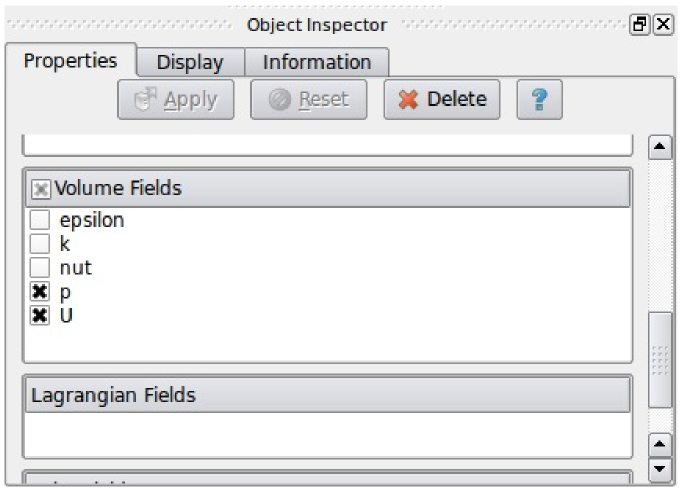
Mesh Parts
☐ internalMesh
☒ inlet - patch
☒ outlet - patch
☒ outerCylinder - patch
☒ propellerTip - patch
☐ camel - patch
☒ AMI1 - patch
☒ AMI2 - patch

Figure 2.3: The properties panel 1.

Rainbow instead of the default one, **CIELab Blue to Red**, to make our plot look clearer and prettier. After clicking the **OK** confirmation button, we could also click the **Make Default** button so that **ParaView** will always use this type of color bar for our case. If the data range is not automatically updated to the max/min limits of a field, we select **Rescale to Data Range** and the data will be at appropriate intervals. We can show color bar for the field by enabling **Show Color Legend** in the **Color Legend** panel of the **Color Scale Editor** window. We can also justify the color bar, such as text size, font selection, and numbering format for the scale. In the **Style** panel (Fig. 2.8), the mesh can be represented by select **Wireframe** from **Representation** menu.

2.5.2 Plots

At first, we select **Clip** (Fig. 2.9) from the **Filter** menu at the top menu bar to create the hemi-cylinder. We input $(0, 0, 0)$ for **Origin** and $(0, 0, 1)$ for **Normal** in the **Display** panel under **Clip** to make the center and normal vector of this



Volume Fields
- [ ] epsilon
- [ ] k
- [ ] nut
- [x] p
- [x] U

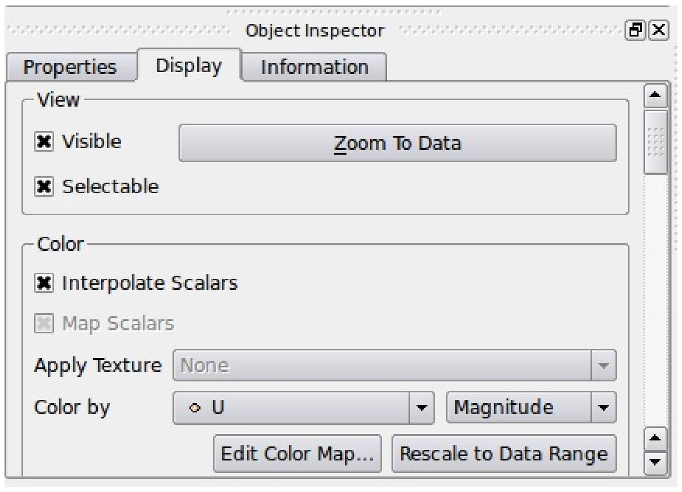
Figure 2.4: The properties panel 2.

hemi-cylinder to be $(0, 0, 0)$ and $(0, 0, 1)$ (Z axis).

We can make a contour plot by selecting **Contour** from the **Filter** menu. If the module is the 3D case, the contours filter will be a set of 2D surfaces that represent a constant value, i.e. isosurfaces. The contour plot can be seen in (Fig. 2.10).

Second, We select **Slice** (Fig. 2.11) from the **Filter** menu to create the cutting plane. We use the same settings as **Clip** to make the center and normal vector of this cutting plane to be (0; 0; 0) and (0; 0; 1) (Z axis).

Vector plots are made by using the **Glyph** filter. In **ParaView**, by default, vectors are plotted on cell vertices but we would like to plot them at cell centers. This could be done by applying the **Cell Centers** filter to the case, and then applying the **Glyph** filter to the resulting cell center data. In the **Properties** panel of **Glyph**, since it is the only vector field present, the velocity field, **U**, is automatically selected in the vectors menu. In the **Scale Mode** menu, we choose **off** which means each glyph has the same length, instead of the default value **Vector**, where the glyph



View

[✓] Visible — Zoom To Data

[✓] Selectable

Color

[✓] Interpolate Scalars

[✓] Map Scalars

Apply Texture: None

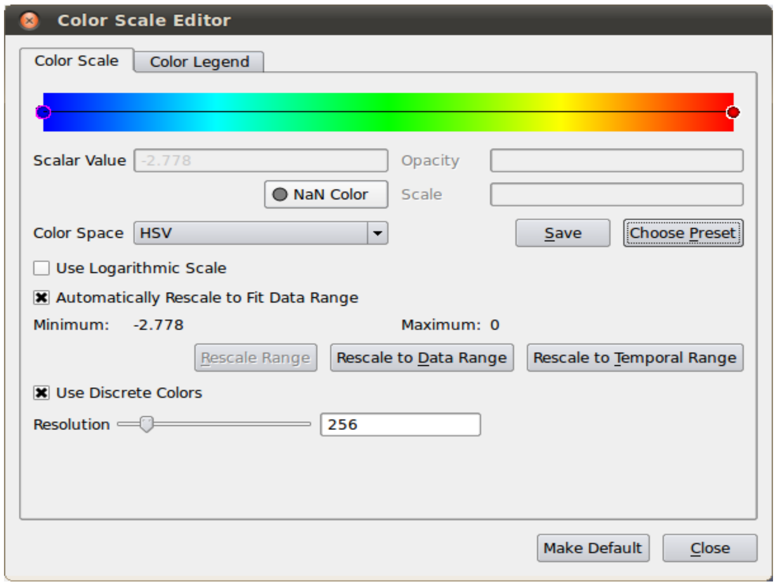
Color by: U, Magnitude

Edit Color Map... — Rescale to Data Range

Figure 2.5: The display panel 1.

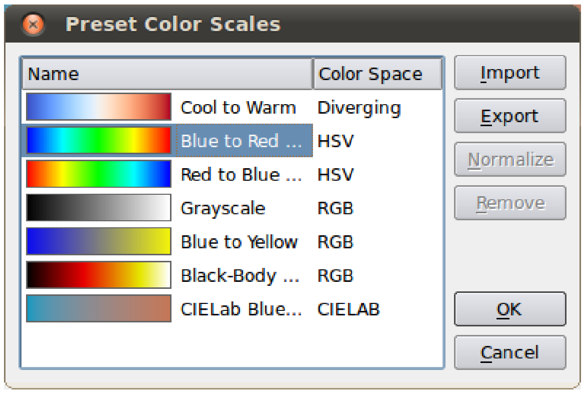
length is proportional to the vector magnitude. The **Set Scale Factor** parameter controls the base length of the glyphs. The vector plot can be seen in (Fig. 2.12).

Streamlines plots are made by creating tracer lines using the **Stream Tracer** filter. We could adjust the number and range of tracer lines by changing the **Number of Points** and **Radius** in the **Properties** panel. The streamlines plot can be seen in (Fig. 2.13).



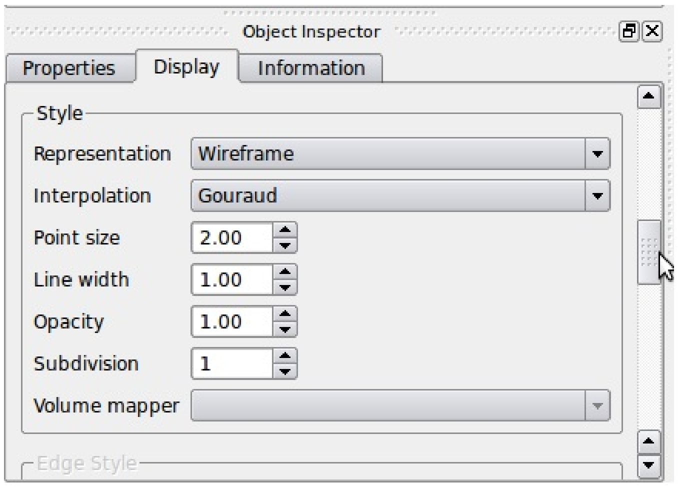
Scalar Value: -2.778
Opacity:
Scale:
Color Space: HSV
☐ Use Logarithmic Scale
☒ Automatically Rescale to Fit Data Range
Minimum: -2.778 Maximum: 0
☒ Use Discrete Colors
Resolution: 256

Figure 2.6: The color scale edit.



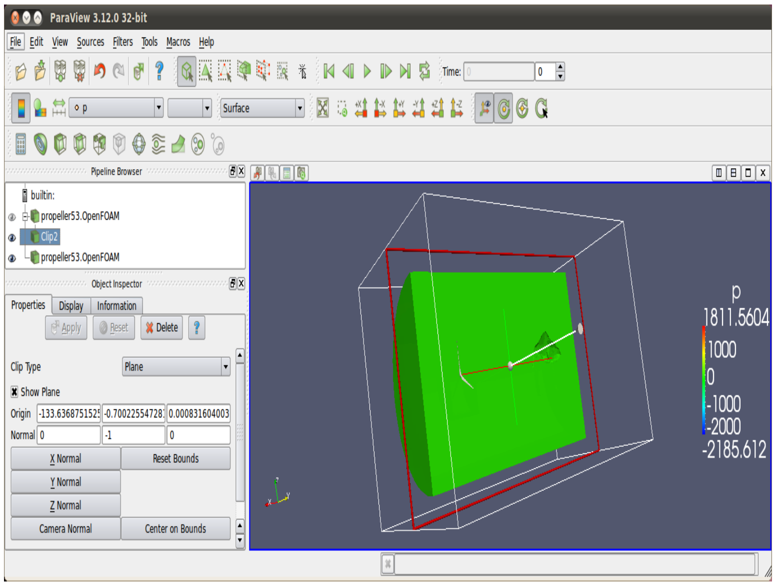
Name	Color Space
Cool to Warm	Diverging
Blue to Red ...	HSV
Red to Blue ...	HSV
Grayscale	RGB
Blue to Yellow	RGB
Black-Body ...	RGB
CIELab Blue...	CIELAB

Figure 2.7: Preset color scales.



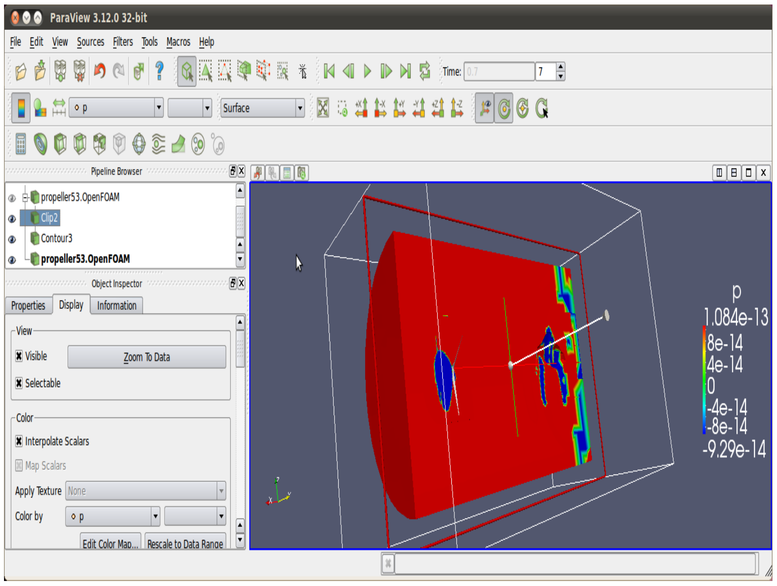
Representation: Wireframe
Interpolation: Gouraud
Point size: 2.00
Line width: 1.00
Opacity: 1.00
Subdivision: 1
Volume mapper:

Figure 2.8: The display panel 2.



Clip Type: Plane
Show Plane: ✓
Origin: -133.6368751525 | -0.70022554728 | 0.000831604003
Normal: 0 | -1 | 0

Figure 2.9: The clip filter.



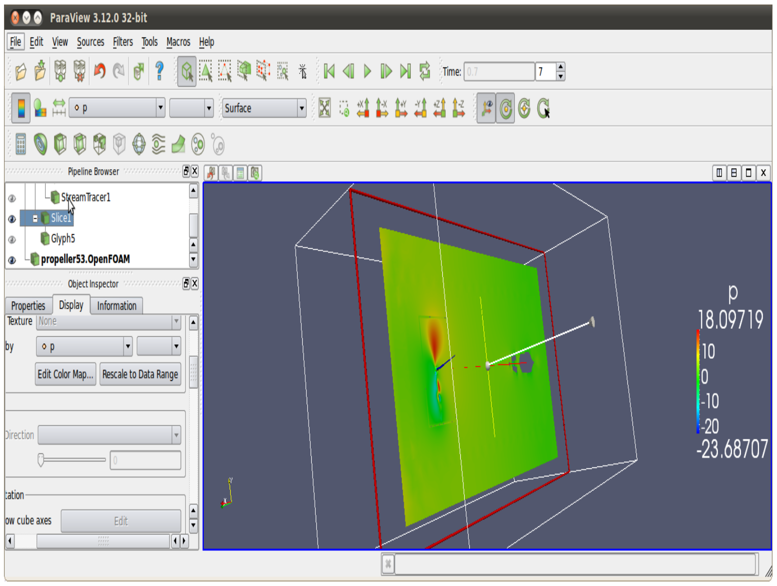
View: [✓ Visible] [✓ Selectable]

Color: [✓ Interpolate Scalars] Map Scalars

Apply Texture: None

Color by: p

Figure 2.10: The contour plot.



Texture: None

by: p

Figure 2.11: The slice filter.

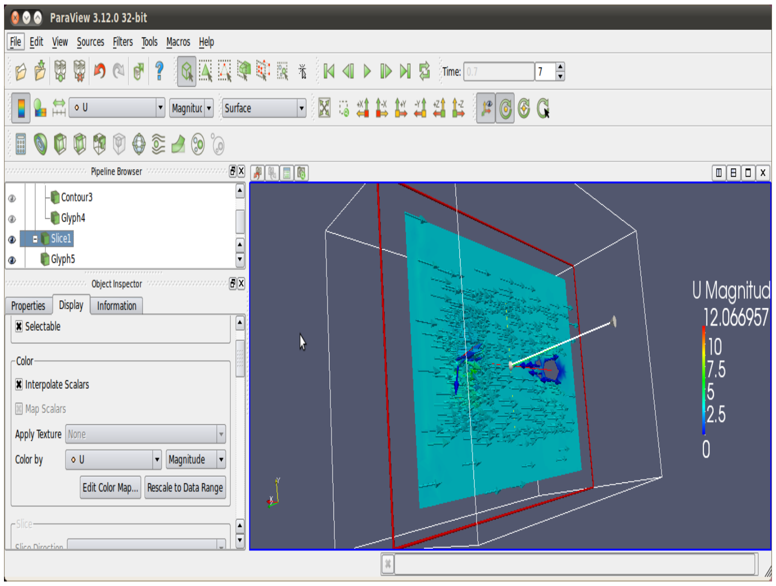


Figure 2.12: The glyph plot.

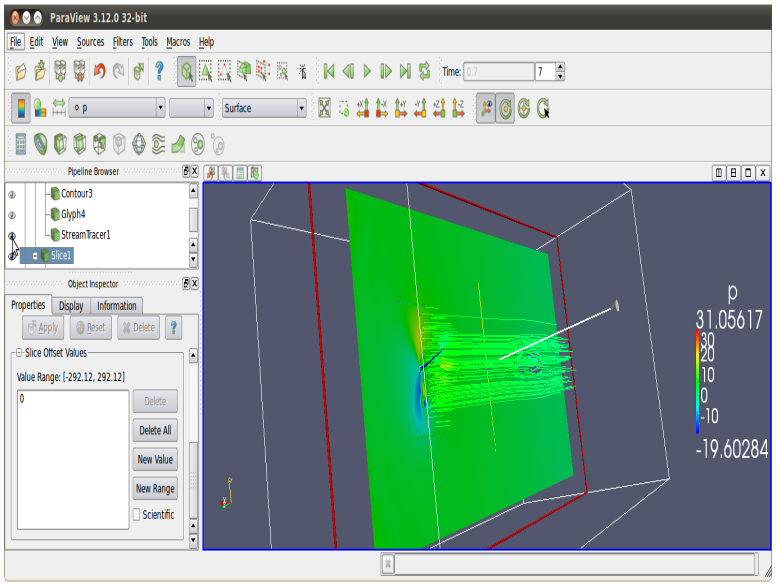


Figure 2.13: The streamtracer plot.

3. INTRODUCTION TO ANSYS FLUENT SOFTWARE AND USAGE

3.1 Introduction

In this Section, we introduce the *ANSYS FLUENT* software [11]. This software is a product of ANSYS, Inc., considered to be one of world's largest computational fluid dynamics (CFD) developers. FLUENT software could be used to compute and simulate a wide range of complex physical problems, especially fluid flow problems. Therefore, scientists and research engineers could build virtual models to visualize and predict the performance of their design problems. This gives them an intuitive and efficient way for understanding and problem-solving, as it saves expensive experimental cost. It also makes FLUENT useful tool of the computer-aided engineering (CAE) process for the manufacturing industry.

FLUENT has excellent parallel processing capability so it could deal with large-scale problems well by using multiple processors to run one single simulation. FLUENT uses the *finite volume method* to discretize the domain and the governing equations of the problem. Hence, by employing the control-volume-based technique, the governing equations of the problem could be converted into algebraic equations and then solved numerically. Creating user-defined functions in FLUENT would allow our design and solution process more flexibly. It can implement the new user models and customize the existing user models extensively for specific applications, e.g., wind turbines.

3.2 Problem Description

This CFD model contains similar geometry of the horizontal axis wind turbine (HAWT) where a wind turbine consisting of three blades with 50 m radius to the one in Section 2. The physical setup is provided by Dr. Alain Perronnet, too. The computational domain contains the rotor region which is surrounded by a smaller cylinder and an outer cylinder inside the block. It extends from approximately 1.87 times blades radius upwind to 7.22 times blades radius downwind of this wind turbine. The wind comes from the left hand side, the inlet, and goes out of the right hand side, the outlet. The uniform incoming velocity of the wind is 8 m/s. The uniform angular velocity of the wind turbine is 75 deg/s. The geometry of our model is given in Fig. 3.1.

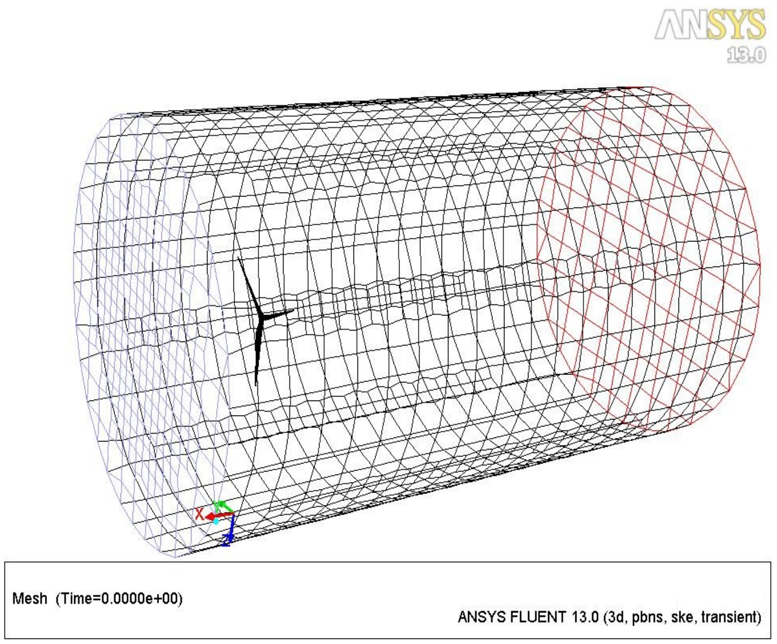


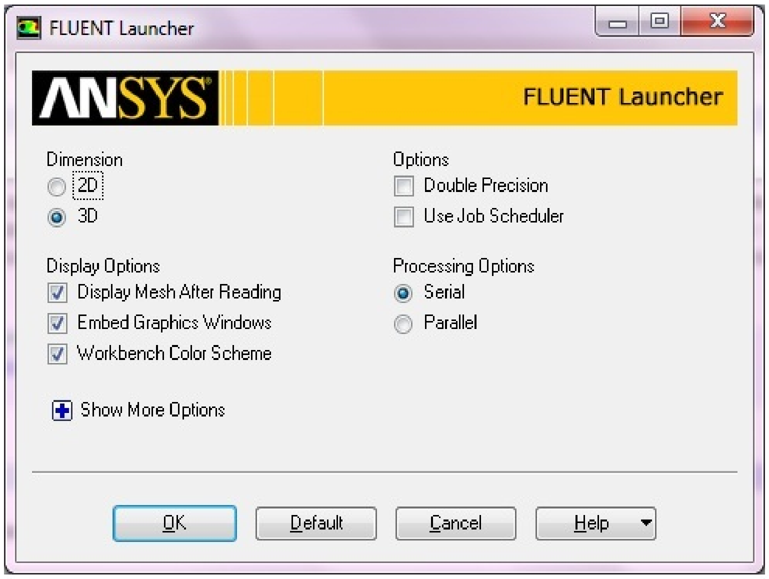
Figure 3.1: Geometry of our model of the wind turbine in an outer cylinderical domain.

3.2.1 Problem Setup and Configurations

In the following sections, we show how to set up the problem in some details.

3.2.2 Preparation

We start ANSYS FLUENT by using FLUENT launcher (Fig. 3.2). In FLUENT launcher, we can specify the options that fit our models. For our case, we select **3D** for the three-dimensional solver under **Dimension**. Under **Display Option**, we can make decisions about the graphic windows within the FLUENT application. We enable **Display Mesh After Reading**, which asks FLUENT to automatically display all of the boundary zones of the mesh immediately after it reads a mesh or case file, **Embed Graphics Windows**, which has the graphics windows embedded within FLUENT instead of floating graphics windows, and **Workbench Color Scheme**, which shows a blue background in the FLUENT graphics windows instead of the classic black one (Fig. 3.3). Under **Options**, **Double Precision** can be used to run the double-precision solver in FLUENT instead of the default single-precision solver. The double-precision solver uses 64 bits to represent each floating point number, compared with the single-precision solver which uses 32 bits. In addition to precision, the extra bits also increase the range of the numbers which can be represented, nevertheless at the cost of much more memory space (for using double precision). The **Use Job Scheduler** can run FLUENT with various job schedulers such as, the Microsoft Job Scheduler for Windows, or LSF, SGE, and PBS Pro on Linux. We did not enable these two options in our case. Under **Processing Options**, we can select one from two options, *serial or parallel*, to run the solver. After enabling **Parallel**, we can specify the number of processors which we would like to use. We use the serial solver to run our case.


FLUENT Launcher

Dimension
- 2D
- [✓ 3D]

Options
- Double Precision
- Use Job Scheduler

Display Options
- ✓ Display Mesh After Reading
- ✓ Embed Graphics Windows
- ✓ Workbench Color Scheme

Processing Options
- [✓ Serial]
- Parallel

Figure 3.2: The FLUENT launcher.

3.2.3 Mesh

We read our mesh file from the folder.

File → **Read** → **Mesh**. . .

3.2.4 General

The options of mesh and solver can be specified under **General** (Fig. 3.4).

First, we display our mesh.

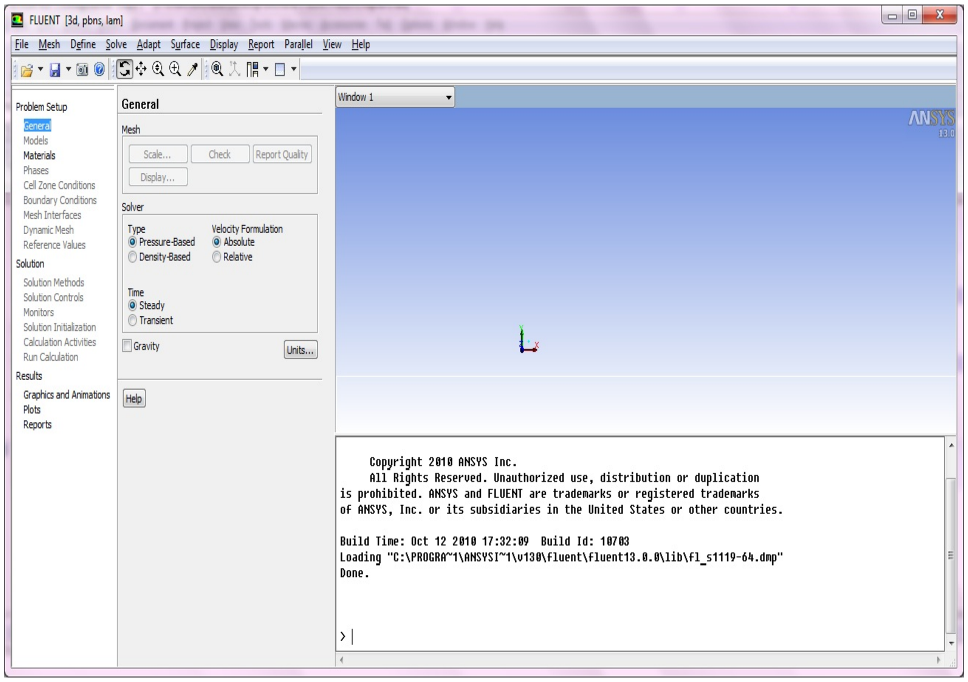
General → **Display** (Fig. 3.5)

Mesh can be displayed in different ways such as nodes, edges, faces, or partitions; meanwhile, we can select the surface areas that we would like to see.

The mesh can be scaled in case of needs.

General → **Scale** (Fig. 3.6)

There is information about dimensions of the mesh on the mesh scale panel. We



```
     Copyright 2010 ANSYS Inc.
     All Rights Reserved. Unauthorized use, distribution or duplication
is prohibited. ANSYS and FLUENT are trademarks or registered trademarks
of ANSYS, Inc. or its subsidiaries in the United States or other countries.

Build Time: Oct 12 2010 17:32:09  Build Id: 10703
Loading "C:\PROGRA~1\ANSYSI~1\v130\fluent\fluent13.0.0\lib\fl_s1119-64.dmp"
Done.


> |
```

Figure 3.3: The FLUENT display panel.

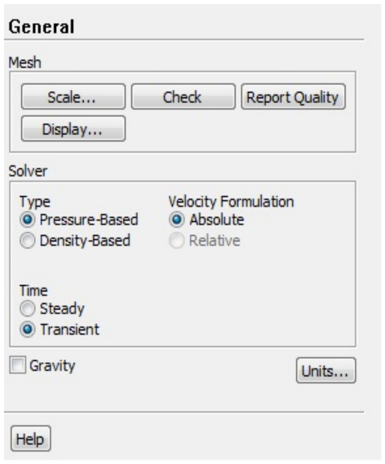
can scale the mesh by enabling **Convert Units** under **Scaling**, selecting the unit we need under **Mesh Was Created In**, and clicking **Scale**. The working unit for the length could be changed by selecting it under **View Length Unit In**. For our case, we use the default unit m.

We define the units of the quantities for our model. The panel of units settings can be seen in Fig. 3.7. We select **angular-velocity** and **pressure** under **Quantities**, and **deg/s** and **pascal** under **Units** respectively.

After manipulating our mesh, we check the mesh to see the information of its statistics.

General → **Check** (Fig. 3.8)

Various checks will be done on the mesh by FLUENT and the progress will be



Type: [✓ Pressure-Based] Density-Based

Velocity Formulation: [✓ Absolute] Relative

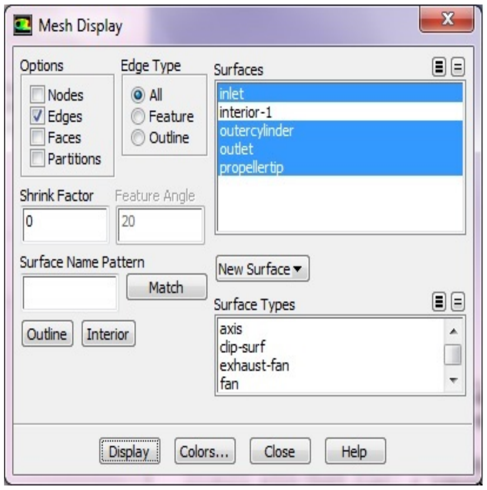
Time: Steady [✓ Transient]

☐ Gravity

Figure 3.4: The general panel.

reported in the console window. There are the statistics for Domain Extents, Volume, and Face area. Errors in the mesh will be reported here. Make sure that the reported minimum volume under Volume statistics is a positive number or FLUENT could not run the calculation. A negative minimum volume value means that one or more cells in the solution domain have improper connectivity so this discretization area needs to be repaired or removed.

There are two numerical methods we can choose in FLUENT. One is *pressure-based solver* and the other is *density-based solver*. Originally, the pressure-based solver is designed for low-speed incompressible flows and mildly compressible flows, and the density-based solver is used for high-speed compressible flows. Nevertheless, lately both solvers have been remodeled and extended to solve for a broad range of flow conditions (from incompressible to highly compressible). Based on the origin of the density-based solver's formulation, the density-based solver may be more accurate than the pressure-based solver for high-speed compressible flows.



Options
- [ ] Nodes
- [✓] Edges
- [ ] Faces
- [ ] Partitions

Edge Type: [✓ All] Feature Outline

Surfaces: [✓ inlet] interior-1 [✓ outercylinder] [✓ outlet] [✓ propellertip]

Shrink Factor: 0
Feature Angle: 20

Surface Name Pattern:

Surface Types: axis, clip-surf, exhaust-fan, fan

Figure 3.5: The mesh display panel.

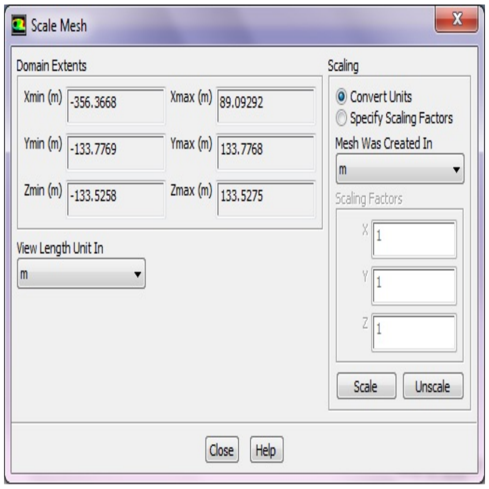
In both approaches, the velocity field is derived from the momentum equation:

$$\oint \rho \vec{v}\vec{v} \cdot d\vec{A} = -\oint pI \cdot d\vec{A} + \oint \bar{\bar{\tau}} \cdot d\vec{A} + \int_V \vec{F} dV \tag{3.1}$$

In the pressure-based solver, the pressure field is solved by a pressure (or pressure correction) equation which is derived from the momentum equation (3.1) and the continuity equation:

$$\oint \rho \vec{v} \cdot d\vec{A} = 0 \tag{3.2}$$

There are two pressure-based solver algorithms in FLUENT. One is the *segregated algorithm*, which solves the governing equations one after another since each governing equation is segregated from other equations, and the other one is the *coupled algorithm*, which solves a coupled system of governing equations. In the density-based solver, the density field is obtained from the continuity equation; meanwhile, the pressure filed is solved by the equation of state. The governing equations are non-



Domain Extents

Xmin (m) -356.3668 Xmax (m) 89.09292

Ymin (m) -133.7769 Ymax (m) 133.7768

Zmin (m) -133.5258 Zmax (m) 133.5275

View Length Unit In: m

Scaling: [✓ Convert Units] Specify Scaling Factors

Mesh Was Created In: m

Scaling Factors: X 1, Y 1, Z 1

Figure 3.6: The mesh scale panel.

linear and coupled so the process of finding a solution must be calculated iteratively until the numerical solution converges.

The iteration steps of pressure-based and density-based solution methods are illustrated in Chart 3.1 and Chart 3.2, respectively.

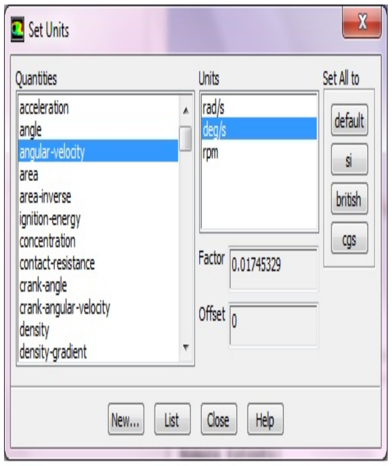
For our case, we select **Pressured-Based**, **Transient**, and **Absolute** under **Type**, **Time**, and **Velocity Formulation**, respectively to define the settings of our solver.

3.2.5 Models

Since our case is about the wind turbine model, the viscous model is the only model that we deal with.

Models → **Viscous** → **Edit...** (Fig. 3.9)

In the panel of viscous model (Fig. 3.10), we select **k-epsilon(2 eqns)**, **Realizable**, and **Enhanced Wall Treatment** under **Model**, **k-epsilon Model**, and **Near-Wall Treatment**, respectively, for our case while keeping the variables under



Factor 0.01745329
Offset 0

Figure 3.7: The setup of units.

Model Constants as default values.

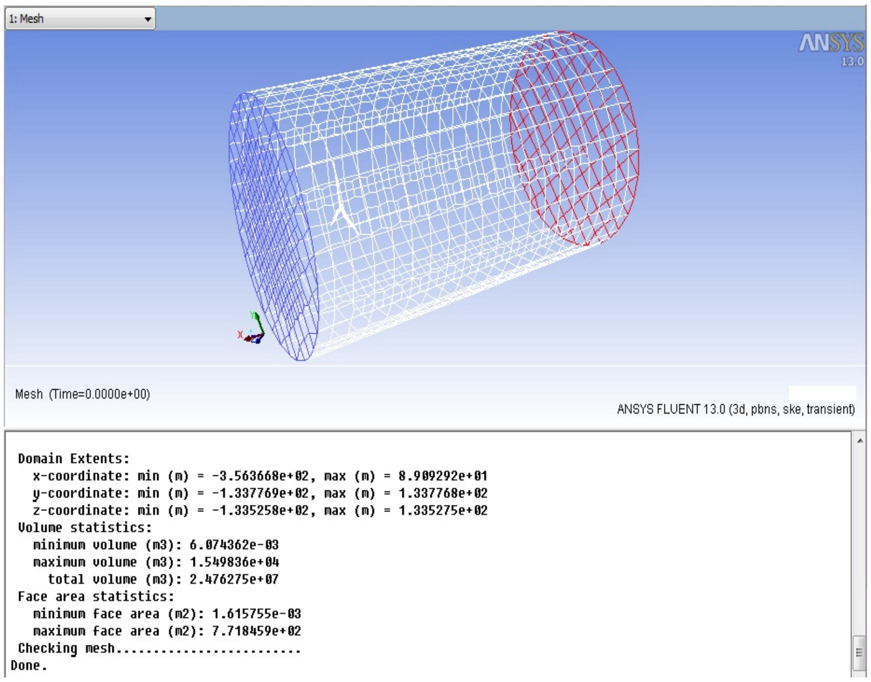
We will introduce and explain turbulence models and wall functions in details in Section 4.

3.2.6 Materials

The only fluid material in our case is air so here we specify the properties of air in the settings of materials.

Materials → **air** → **Create/Edit. . .** (Fig. 3.11)

The density and viscosity of air depend on the temperature, humidity, and so on. According to International Standard Atmosphere (ISA), air has a density of 1.225 kg/m^3 and a viscosity of 1.81×10$^-$5 kg/m·s approximately. We select **constant** under **Density(kg/m^3)** and **Viscosity(kg/m·s)**, and set the number to be **1.225** and **1.81e − 05**, respectively, in the panel of air properties settings. (Fig. 3.12)



```
 Domain Extents:
   x-coordinate: min (m) = -3.563668e+02, max (m) = 8.909292e+01
   y-coordinate: min (m) = -1.337769e+02, max (m) = 1.337768e+02
   z-coordinate: min (m) = -1.335258e+02, max (m) = 1.335275e+02
 Volume statistics:
   minimum volume (m3): 6.074362e-03
   maximum volume (m3): 1.549836e+04
     total volume (m3): 2.476275e+07
 Face area statistics:
   minimum face area (m2): 1.615755e-03
   maximum face area (m2): 7.718459e+02
 Checking mesh.........................
Done.
```

Figure 3.8: The statistics of the mesh.

3.2.7 Cell Zone Conditions

In FLUENT, the fluid flow problems which involve moving parts (such as rotating turbine blades) can be solved in a moving reference frame instead of the default stationary reference frame. When a moving reference frame is used, an additional acceleration term would be considered in the original motion equation since a stationary reference frame is transformed to a moving reference frame. The flow features could be modeled by solving those equations. For some problems, the entire computational domain can be considered as a single moving reference domain. This is called the *single reference frame (SRF)* approach. For some other problems with more complicated geometries or multiple moving parts, the model must be broken up into multiple fluid/solid cell zones and the interfaces between these cell zones

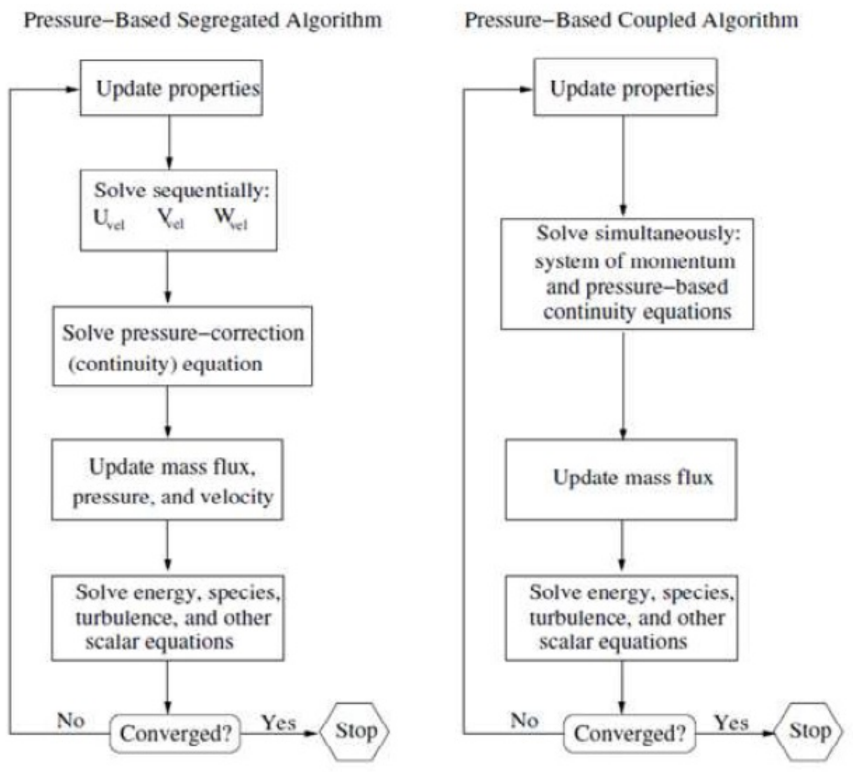

Pressure–Based Segregated Algorithm
Update properties
Solve sequentially: U_{vel} V_{vel} W_{vel}
Solve pressure–correction (continuity) equation
Update mass flux, pressure, and velocity
Solve energy, species, turbulence, and other scalar equations
No
Converged?
Yes
Stop
Pressure–Based Coupled Algorithm
Update properties
Solve simultaneously: system of momentum and pressure–based continuity equations
Update mass flux
Solve energy, species, turbulence, and other scalar equations
No
Converged?
Yes
Stop

Chart 3.1: Iteration steps of Pressure-Based solution methods.

should be well-defined. There are two approaches to solve this type of problems in FLUENT:

- Multiple Moving Reference Frames
    - Multiple Reference Frame (MRF) Model
    - Mixing Plane Model
- Sliding Mesh Model
- Dynamic Mesh Model

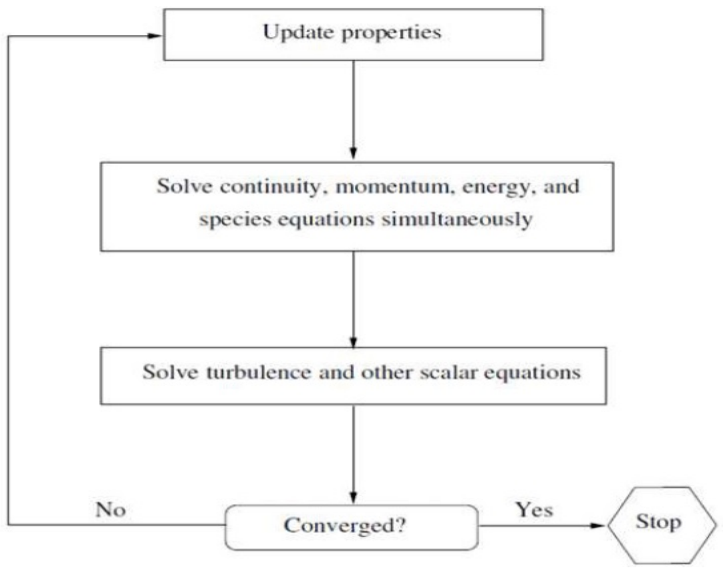
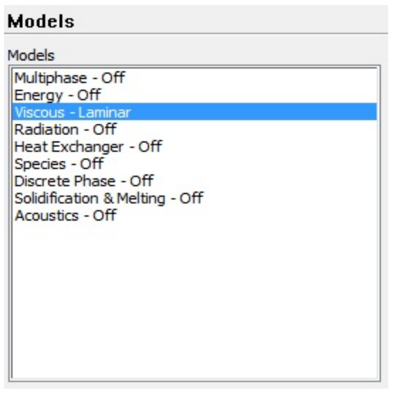

Update properties
Solve continuity, momentum, energy, and species equations simultaneously
Solve turbulence and other scalar equations
No
Converged?
Yes
Stop

Chart 3.2: Iteration steps of Density-Based solution methods.

The MRF and mixing plane approaches are both approximations for steady states and the main difference between these two approaches is that they use different ways to treat the interfaces' conditions. When the interactions between the stationary and moving parts are unsteady, the sliding mesh approach could be used to capture the transient behaviors of the fluid flow around the moving parts. Both sliding mesh and dynamic mesh approaches allow us to move zones relative to each other and the dynamic mesh approach can even adjust the mesh according to the changes. The primary determining factor of using the sliding mesh approach or the dynamic mesh approach is whether the problem involves the *mesh deformation*. If the mesh in the problem is deforming, the dynamic mesh approach must be used. Otherwise, either approach could be used. The sliding mesh approach would be recommended if the mesh is not deforming in the problem since this approach is simpler and more efficient than the dynamic approach.


Models

Models

Multiphase - Off
Energy - Off
Viscous - Laminar
Radiation - Off
Heat Exchanger - Off
Species - Off
Discrete Phase - Off
Solidification & Melting - Off
Acoustics - Off

Figure 3.9: The panel of the models.

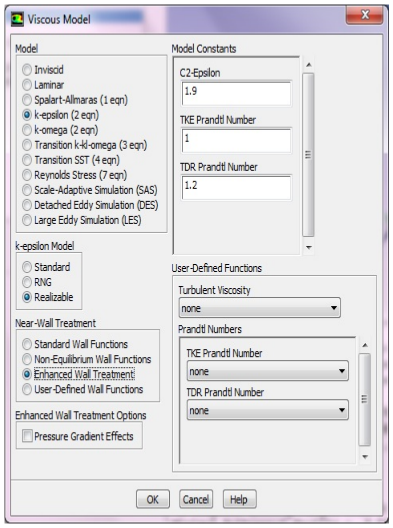
To simulate rotating blades in our case, we use the sliding mesh approach. We start to set the cell zone conditions for the fluid inside the cylinder.

Cell Zone Conditions $\rightarrow$ **fluid-1** $\rightarrow$ **Edit. . .** (Fig. 3.13)

In the panel of fluid settings (Fig. 3.14), we enable **Mesh Motion** in order to use the sliding mesh approach and then click on **Mesh Motion** tab to set up the conditions for the fluid. We use the default number **0**, **0**, **0** for **X(m)**, **Y(m)**, **Z(m)** under **Rotation-Axis Origin**, respectively. For our case, we set the number to be **−75** for **Speed(deg/s)** under **Rotational Velocity**. We set the direction **1**, **0**, **0** for **X**, **Y**, **Z** under **Rotation-Axis Direction**. We do not need the translational behavior so we use the default number **0**, **0**, **0** for **X(m/s)**, **Y(m/s)**, **Z(m/s)** under **Translation Velocity**.

3.2.8 Boundary Conditions

After setting up the conditions for the interior fluid, we can start to set the boundary conditions for other parts, *inlet*, *outlet*, *outer cylinder*, and *turbine blades*, in our case.



Viscous Model

Model
- Inviscid
- Laminar
- Spalart-Allmaras (1 eqn)
- [✓ k-epsilon (2 eqn)]
- k-omega (2 eqn)
- Transition k-kl-omega (3 eqn)
- Transition SST (4 eqn)
- Reynolds Stress (7 eqn)
- Scale-Adaptive Simulation (SAS)
- Detached Eddy Simulation (DES)
- Large Eddy Simulation (LES)

k-epsilon Model
- Standard
- RNG
- [✓ Realizable]

Near-Wall Treatment
- Standard Wall Functions
- Non-Equilibrium Wall Functions
- [✓ Enhanced Wall Treatment]
- User-Defined Wall Functions

Enhanced Wall Treatment Options
- Pressure Gradient Effects

Model Constants

C2-Epsilon: 1.9

TKE Prandtl Number: 1

TDR Prandtl Number: 1.2

User-Defined Functions

Turbulent Viscosity: none

Prandtl Numbers

TKE Prandtl Number: none

TDR Prandtl Number: none

Figure 3.10: The setup of the viscous model.

First, we choose the type for the inlet and then start setting up the boundary conditions.

Boundary Conditions → **inlet** → **velocity-inlet** under **Type** → **Edit. . .** (Fig. 3.15)

In the panel of the inlet settings (Fig. 3.16), we select **Components**, **Absolute**, and **Cartesian (X,Y,Z)** under **Velocity Specification Method**, **Reference Frame**, and **Coordinate System**. We set the number $(\mathbf{-8, 0, 0})$ for **X-Velocity (m/s)**, **Y-Velocity (m/s)**, and **Z-Velocity (m/s)**. For **Supersonic/Initial Gauge Pressure (pascal)** and **Turbulence** part, we use the default values.

Second, we choose the type for the outlet and then start setting up the boundary conditions.

Boundary Conditions → **outlet** → **pressure-outlet** under **Type** → **Edit. . .**

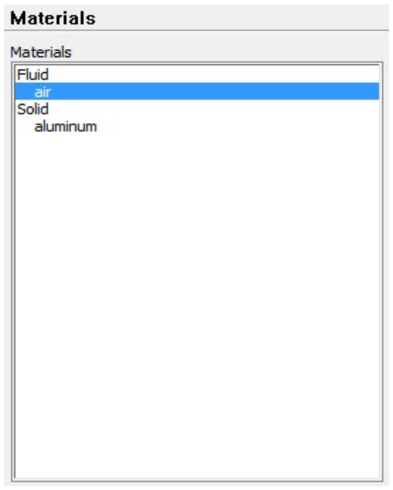


Figure 3.11: The panel of the materials.

We retain the default boundary conditions for the outlet in the panel of the outlet settings (Fig. 3.17).

Third, we choose the type for the outer cylinder and then start setting up the boundary conditions.

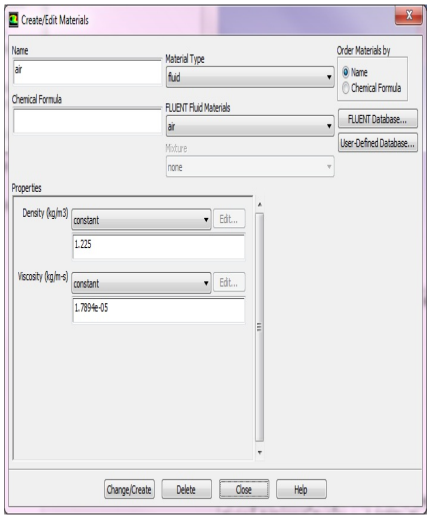
Boundary Conditions → **outer cylinder** → **wall** under **Type** → **Edit...**

In the panel of the outer cylinder settings (Fig. 3.18), we select **Moving Wall**, **Absolute**, **Components**, and **No Slip** under **Wall Motion**, **Motion**, and **Shear Condition**, respectively. We change the number $(-\mathbf{8}, \mathbf{0}, \mathbf{0})$ for **X-Velocity (m/s)**, **Y-Velocity (m/s)**, and **Z-Velocity (m/s)** under **Velocity Components**.

Last, we choose the type for the turbine blades and then we start setting up the boundary conditions.

Boundary Conditions → **turbine blades** → **wall** under **Type** → **Edit...**

In the panel of the turbine blades settings (Fig. 3.19), we select **Moving Wall**, **Relative to Adjacent Cell Zone**, **Rotational** and **No Slip** under **Wall Motion**,


Create/Edit Materials

Name: air

Chemical Formula:

Material Type: fluid

FLUENT Fluid Materials: air

Mixture: none

Order Materials by: [✓ Name] Chemical Formula

Properties

Density (kg/m3): constant — 1.225

Viscosity (kg/m-s): constant — 1.7894e-05

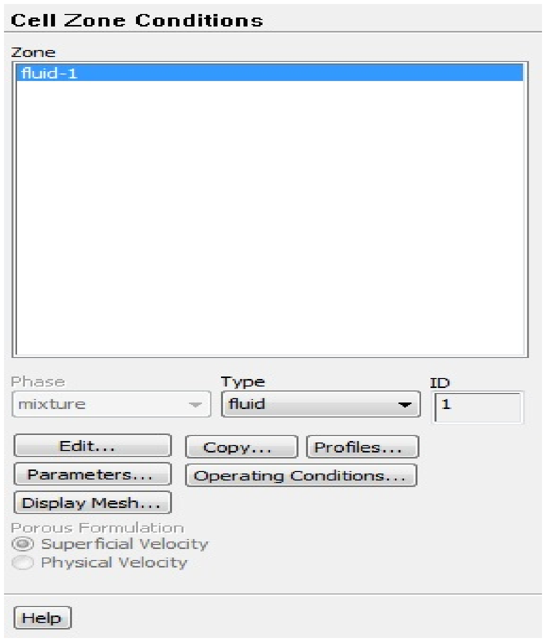
Figure 3.12: The setup of air.

Motion, and **Shear Condition**, respectively. We set the number **0** under **Speed (deg/s)**. For the rotational properties, we change the value $(\mathbf{0}, \mathbf{0}, \mathbf{0})$ for **X (m)**, **Y (m)**, and **Z (m)** under **Rotation-Axis Origin**, and set the direction $(\mathbf{1}, \mathbf{0}, \mathbf{0})$ for **X**, **Y**, and **Z** under **Rotation-Axis Direction**.

3.2.9 Solution Methods

Since we choose the pressure-based solver to be the solver type for our case, we can choose from various pressure-velocity coupling algorithms to attain the results we want. There are two pressure-based solver in FLUENT. One is the segregated algorithm and the other is the coupled algorithm. FLUENT provides five pressure-velocity coupling algorithms: *SIMPLE* , *SIMPLEC (SIMPLE-Consistent)*, *PISO*, *Fractional Step (FSM)*, and *Coupled*. The first four schemes use the pressure-based segregated solver and the last one uses the pressure-based coupled solver.

SIMPLE stands for *Semi-Implicit Method for Pressure-Linked Equations* and it


Cell Zone Conditions

Zone

fluid-1

Phase: mixture

Type: fluid

ID: 1

Porous Formulation

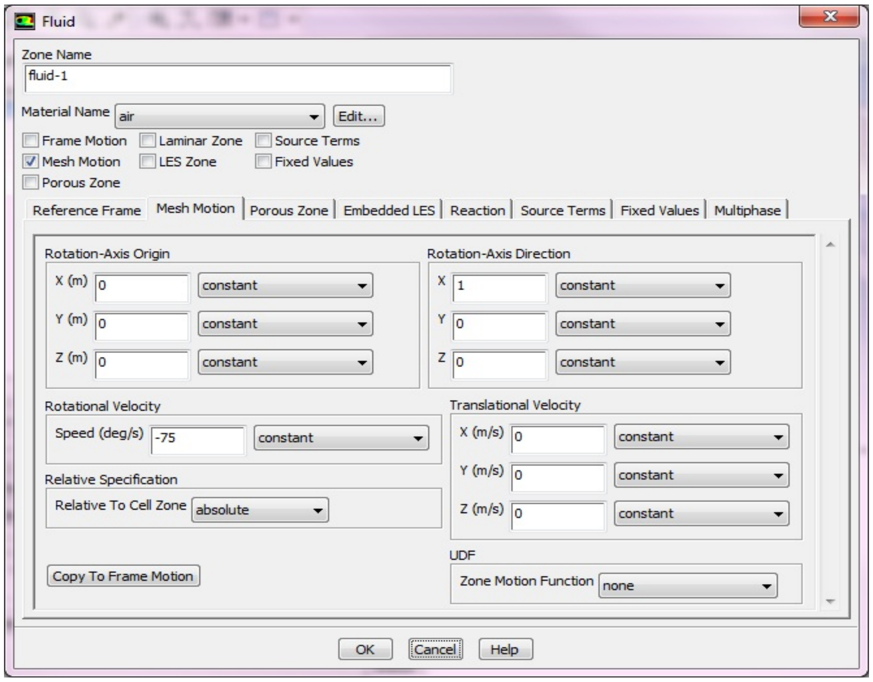
(● Superficial Velocity) Physical Velocity

Figure 3.13: The panel of the cell zone conditions.

is normally used for *steady-state problems*. The SIMPLE algorithm uses the velocity and pressure correction equations to make the mass conservation equation to solve for the pressure field. ([?])

The SIMPLEC algorithm is similar to the SIMPLE algorithm. The only difference between these two algorithms is the expression for the face flux correction equation.

PISO is the acronym for *Pressure Implicit with Split Operator*. The PISO algorithm is the extension of the SIMPLE algorithm and normally is used for the time-dependent fluid flow problems. The primary differences between these two algorithms is that the PISO algorithm does not apply under-relaxation and could repeat the momentum corrector step until the condition is satisfied. In FLUENT, the PISO algorithm offers two additional corrections: *neighbor correction* and *skewness correction*, to improve the efficiency of the calculation. ([?])


Fluid

Zone Name: fluid-1

Material Name: air

Frame Motion [✓ Mesh Motion] Porous Zone Laminar Zone LES Zone Source Terms Fixed Values

Reference Frame | Mesh Motion | Porous Zone | Embedded LES | Reaction | Source Terms | Fixed Values | Multiphase

Rotation-Axis Origin

X (m)	0	constant
Y (m)	0	constant
Z (m)	0	constant

Rotation-Axis Direction

X	1	constant
Y	0	constant
Z	0	constant

Rotational Velocity

Speed (deg/s): -75 constant

Translational Velocity

X (m/s)	0	constant
Y (m/s)	0	constant
Z (m/s)	0	constant

Relative Specification

Relative To Cell Zone: absolute

UDF

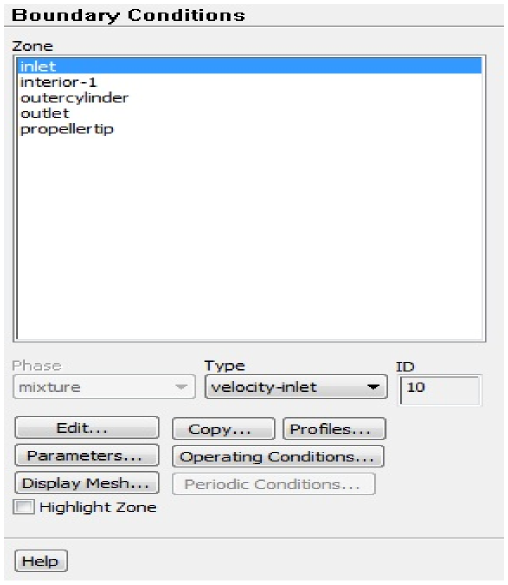
Zone Motion Function: none

Figure 3.14: The setup of the cell zone conditions.

The Fractional Step Method is normally used for unsteady flow problems. FSM uses operator-splitting (is also called approximate factorization) to decouple the momentum equations from the continuity equations to solve the problem and this way is similar to the segregated solution algorithms. FSM could be enabled by using the non-iterative time advancement (NITA) scheme.

The coupled algorithm can be used both in the steady-state flows problems and the transient flows problems especially with poor mesh quality or large time steps. This algorithm solves a coupled system of governing equations, i.e., the momentum and continuity equations together.

To compare with the results in OpenFOAM, we use the SIMPLE algorithm for the pressure-velocity coupling scheme. The solution parameters selection is as fol-


Boundary Conditions

Zone

inlet
interior-1
outercylinder
outlet
propellertip

Phase: mixture
Type: velocity-inlet
ID: 10

Figure 3.15: The panel of the boundary conditions.

lows: we select **SIMPLE** for **Scheme** under **Pressure-Velocity Coupling**; meanwhile, we select **Green-Gauss Cell Based**, **PRESTO!**, **Second Order Upwind**, **Second Order Upwind**, and **Second Order Upwind**, for **Gradient**, **Pressure**, **Momentum**, **Turbulent Kinetic Energy**, and **Turbulent Dissipation Rate** under **Spatial Discretization**, respectively; finally, we select **First Order Implicit** under **Transient Formulation**. (Fig. 3.20)

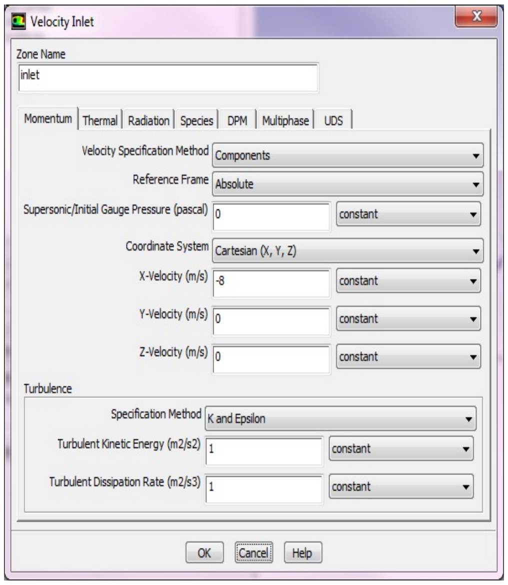
3.2.10 Monitors

FLUENT can record residuals, drag coefficient, lift coefficient, and moment, and report the results of these options. (Fig. 3.21)

We can make the plots of residuals during the calculation.

Monitors → Residuals → Edit...

In the panel of the residuals settings (Fig. 3.22), we select **Print to Console** and


Velocity Inlet

Zone Name: inlet

Momentum | Thermal | Radiation | Species | DPM | Multiphase | UDS

Velocity Specification Method: Components
Reference Frame: Absolute
Supersonic/Initial Gauge Pressure (pascal): 0 constant
Coordinate System: Cartesian (X, Y, Z)
X-Velocity (m/s): -8 constant
Y-Velocity (m/s): 0 constant
Z-Velocity (m/s): 0 constant

Turbulence
Specification Method: K and Epsilon
Turbulent Kinetic Energy (m2/s2): 1 constant
Turbulent Dissipation Rate (m2/s3): 1 constant

Figure 3.16: The setup of the inlet.

Plot under **Options** and leave the remaining as default values.

To record the drag and lift coefficients during the calculation, we can select the wall zones of our interest and the force vector.

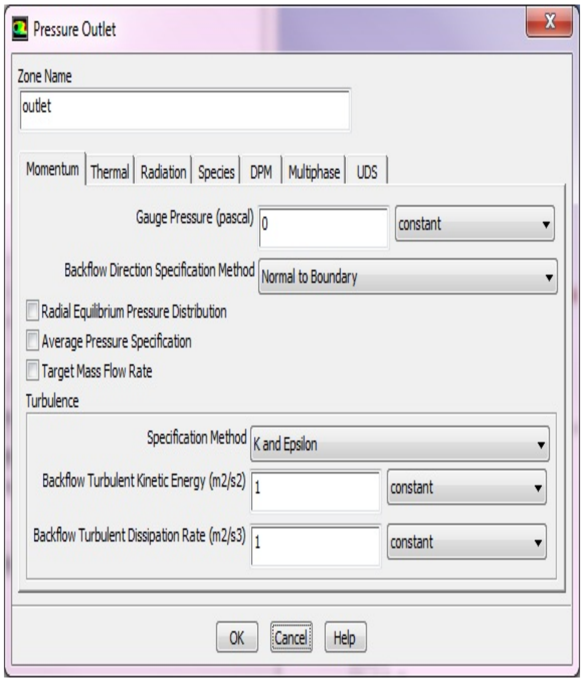
Monitors → Drag/Lift → Edit...

In the panel of the drag/lift settings (Fig. 3.23 and Fig. 3.24), we select **Write** and **Per Zone** for both coefficients, and set the value $(\mathbf{1}, \mathbf{0}, \mathbf{0})$ and $(\mathbf{0}, \mathbf{1}, \mathbf{0})$ for **(X,Y,Z)** under **Force Vector**, respectively. We also enable both of **outer cylinder** and **turbine blades** under **Wall Zones**.

For the report of moment during the calculation, we could select the wall zones of our interest, the moment axis, and the moment center.

Monitors → Moment → Edit...

In the panel of the moment settings (Fig. 3.25), we enable **Write** and **Per Zone** and set the value $(\mathbf{1}, \mathbf{0}, \mathbf{0})$ for **(X,Y,Z)** under both **Moment Center** and **Moment**


Pressure Outlet

Zone Name
outlet

Momentum | Thermal | Radiation | Species | DPM | Multiphase | UDS

Gauge Pressure (pascal) 0 constant

Backflow Direction Specification Method Normal to Boundary

Radial Equilibrium Pressure Distribution

Average Pressure Specification

Target Mass Flow Rate

Turbulence

Specification Method K and Epsilon

Backflow Turbulent Kinetic Energy (m2/s2) 1 constant

Backflow Turbulent Dissipation Rate (m2/s3) 1 constant

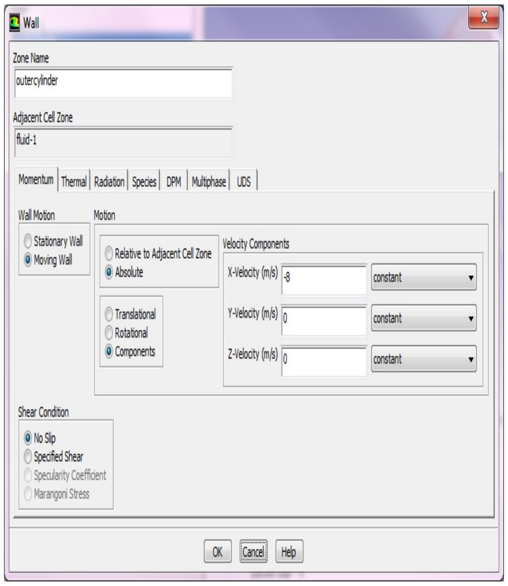
OK Cancel Help

Figure 3.17: The setup of the outlet.

Axis. We also enable both of **outer cylinder** and **turbine blades** under **Wall Zones**.

3.2.11 Solution Initialization

Before FLUENT runs the calculation, the flow field in the entire domain must be initialized. In the panel of the solution initialization (Fig. 3.26), we select **Standard Initialization** under **Initialization Methods** for our case. We choose **inlet** under **Compute from** to start the initialization. The relative velocity formulation is preferred to be used for the problem that most of fluid flow is rotating in the domain in contrast to the absolute velocity formulation. For our case, if we select **Relative to Cell Zone**, the calculation may have problems about convergence in first few iteration steps by containing discontinuities. We select **Absolute** from **Ref-**


Wall

Zone Name: outercylinder

Adjacent Cell Zone: fluid-1

Wall Motion: Stationary Wall [✓ Moving Wall]

Motion: Relative to Adjacent Cell Zone [✓ Absolute]

Translational Rotational [✓ Components]

Velocity Components

X-Velocity (m/s)	-8	constant
Y-Velocity (m/s)	0	constant
Z-Velocity (m/s)	0	constant

Shear Condition: [✓ No Slip] Specified Shear Specularity Coefficient Marangoni Stress

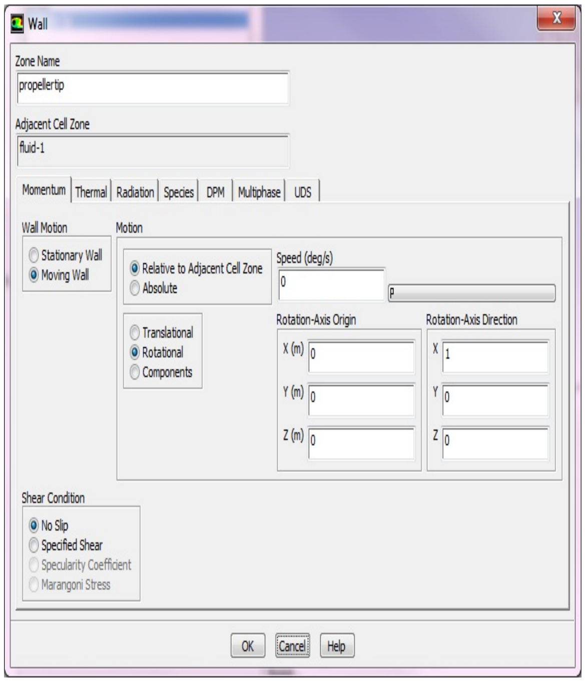
Figure 3.18: The setup of the outercylinder.

erence Frame to avoid the convergence problems. Meanwhile, we set the values **0**, ($-\mathbf{8}, \mathbf{0}, \mathbf{0}$), **0.06**, and **0.0495** for **Gauge Pressure (pascal)**, **(X Velocity (m/s),Y Velocity (m/s),Z Velocity (m/s))**, **Turbulent Kinetic Energy ($\mathbf{m^2/s^2}$)**, and **Turbulent Dissipation Rate ($\mathbf{m^2/s^3}$)**.

3.2.12 Calculation Activities

FLUENT can save the numerical results for the calculation. In order not to occupy more space, we select to save the results for every 10 time steps. We set the number **10** under **Autosave Every (Time Steps)** and start setting up the options for the saved results.

Calculation Activities → Edit... (Fig. 3.27) In the panel of autosave (Fig. 3.28), we enable **Each Time** in the **When the Data File is Saved, Save the Case** in order to do the post-processing work in other softwares. We can use the file names


Wall

Zone Name: propellertip

Adjacent Cell Zone: fluid-1

Momentum | Thermal | Radiation | Species | DPM | Multiphase | UDS

Wall Motion: Stationary Wall [✓ Moving Wall]

Motion: [✓ Relative to Adjacent Cell Zone] Absolute

Speed (deg/s): 0

Translational [✓ Rotational] Components

Rotation-Axis Origin: X (m) 0, Y (m) 0, Z (m) 0

Rotation-Axis Direction: X 1, Y 0, Z 0

Shear Condition: [✓ No Slip] Specified Shear, Specularity Coefficient, Marangoni Stress

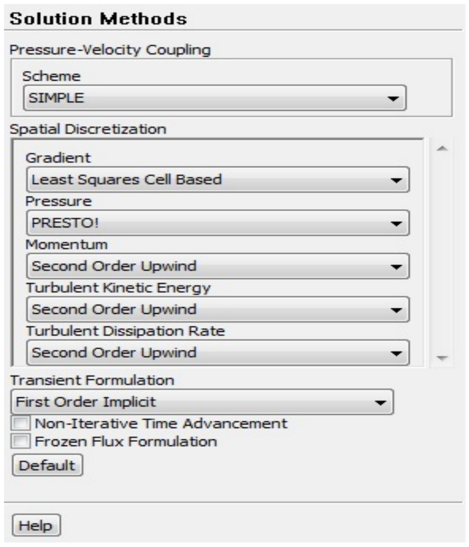
OK Cancel Help

Figure 3.19: The setup of the propellertip.

by typing or browsing in the folder under **File Name**. In order to make the results organized, we prefer the file names ordered by time steps so we select **time-step** under **Append File Name with**.

3.2.13 Run Calculation

After finishing all settings for our model, we are ready prepare to run the calculation. (Fig. 3.29) **Check Case** (Fig. 3.30) is recommended to do before the calculation starts. After checking the quality of mesh, FLUENT shows the details of mesh and offers recommendations for the case. We select **Fixed** under **Time Stepping Method** for our case and set the numbers **0.0001** and **1000** under **Time Step Size (s)** and **Number of Time Steps**, respectively, to start our calculation. We keep the remaining values as the default values.


Solution Methods

Pressure-Velocity Coupling

Scheme: SIMPLE

Spatial Discretization

Gradient: Least Squares Cell Based

Pressure: PRESTO!

Momentum: Second Order Upwind

Turbulent Kinetic Energy: Second Order Upwind

Turbulent Dissipation Rate: Second Order Upwind

Transient Formulation: First Order Implicit

- [ ] Non-Iterative Time Advancement
- [ ] Frozen Flux Formulation

Figure 3.20: The setup of the solution methods.

3.2.14 Graphics and Animations

FLUENT can display results by graphs or animations. (Fig. 3.31)

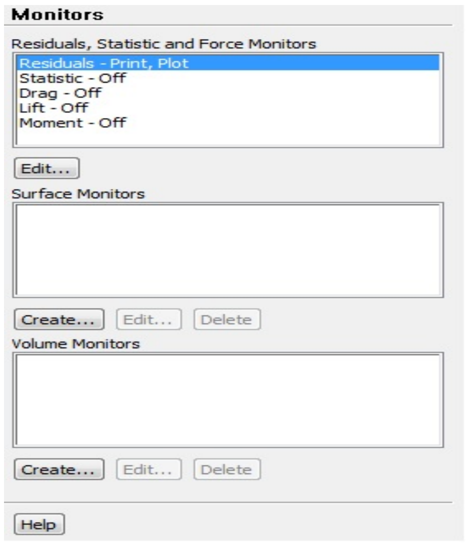
We can display contours of velocity magnitude on the wall zones. (Fig. 3.32)

Graphics and Animations → **Contours** → **Set Up...**

We enable **Filled** under **Options** and select **Velocity...** and **Velocity Magnitude** from **Contours of**. To display the contours of velocity on all wall zones, we select **wall** from **Surface Types**.

3.2.15 Reports

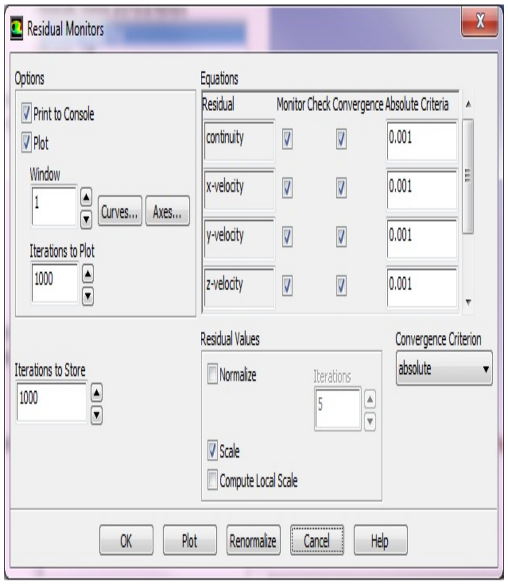
FLUENT can calculate and display the reports for many statistics. (Fig. 3.33) Since we have already set up the moment, drag coefficients, and lift coefficients in **Monitor**, we can get reports in the prescribed folder.


Monitors

Residuals, Statistic and Force Monitors

Residuals - Print, Plot
Statistic - Off
Drag - Off
Lift - Off
Moment - Off

Figure 3.21: The panel of the monitors.


Residual Monitors

Options

- ✓ Print to Console
- ✓ Plot

Window: 1

Iterations to Plot: 1000

Iterations to Store: 1000

Equations

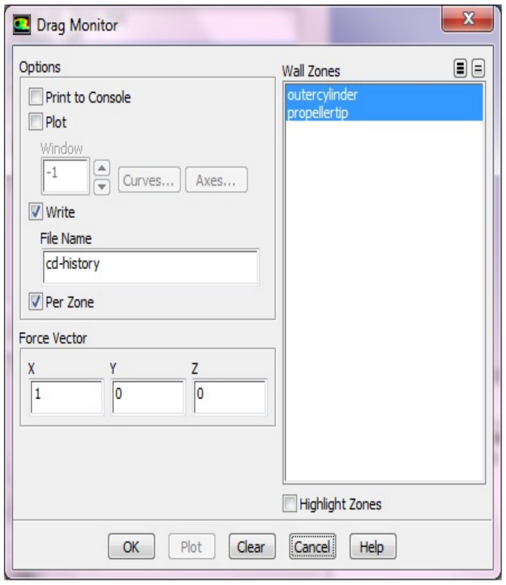
Residual	Monitor	Check Convergence	Absolute Criteria
continuity	✓	✓	0.001
x-velocity	✓	✓	0.001
y-velocity	✓	✓	0.001
z-velocity	✓	✓	0.001

Residual Values

- Normalize
- Iterations: 5
- ✓ Scale
- Compute Local Scale

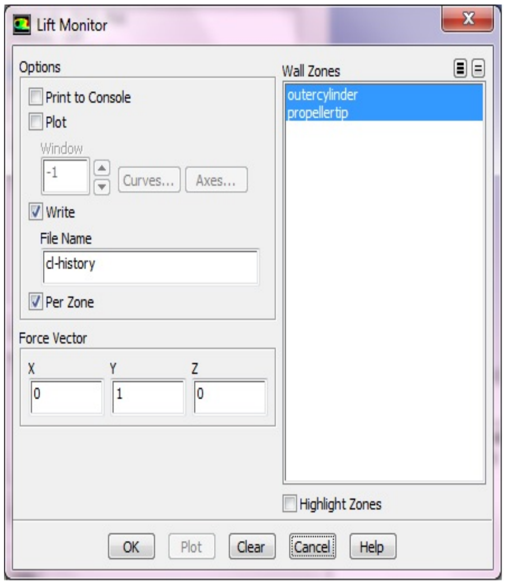
Convergence Criterion: absolute

Figure 3.22: The setup of the residual monitors.



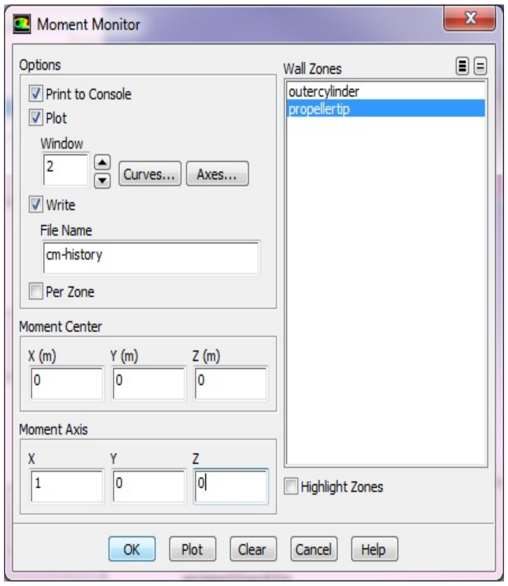
Wall Zones: outercylinder, propellertip
Window: -1
Write: ✓
File Name: cd-history
Per Zone: ✓
Force Vector: X 1, Y 0, Z 0

Figure 3.23: The setup of the drag monitors.



Wall Zones: outercylinder, propellertip
Window: -1
Write: ✓
File Name: cl-history
Per Zone: ✓
Force Vector: X 0, Y 1, Z 0

Figure 3.24: The setup of the lift monitors.



Moment Monitor

Options

[✓] Print to Console

[✓] Plot

Window: 2

[✓] Write

File Name: cm-history

[ ] Per Zone

Moment Center

X (m): 0 Y (m): 0 Z (m): 0

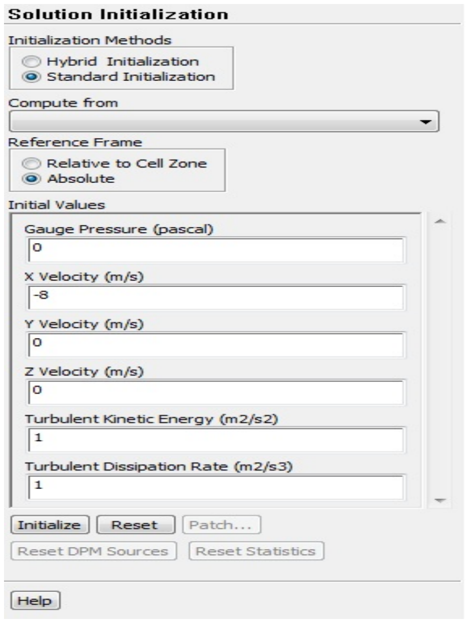
Moment Axis

X: 1 Y: 0 Z: 0

Wall Zones: outercylinder, [✓ propellertip]

[ ] Highlight Zones

Figure 3.25: The setup of the moment monitors.



Solution Initialization

Initialization Methods: Hybrid Initialization [✓ Standard Initialization]

Compute from

Reference Frame: Relative to Cell Zone [✓ Absolute]

Initial Values

Gauge Pressure (pascal): 0

X Velocity (m/s): -8

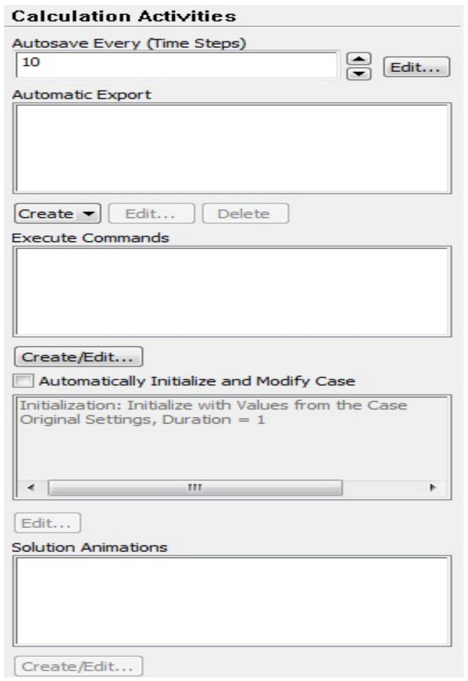
Y Velocity (m/s): 0

Z Velocity (m/s): 0

Turbulent Kinetic Energy (m2/s2): 1

Turbulent Dissipation Rate (m2/s3): 1

Figure 3.26: The setup of the solution initialization.


Calculation Activities

Autosave Every (Time Steps): 10

Automatic Export

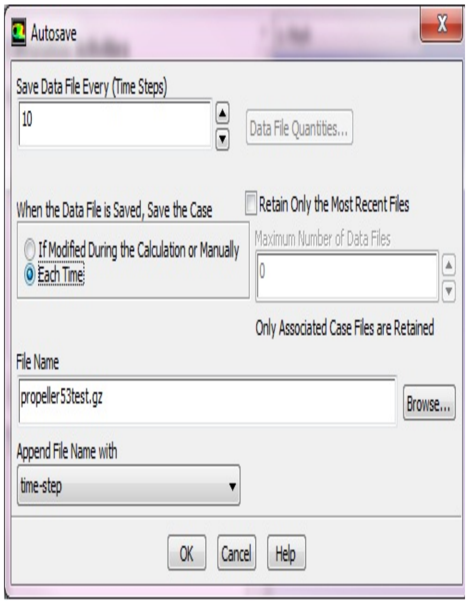
Execute Commands

Automatically Initialize and Modify Case

Initialization: Initialize with Values from the Case
Original Settings, Duration = 1

Solution Animations

Figure 3.27: The panel of the calculation activities.


Autosave

Save Data File Every (Time Steps): 10

When the Data File is Saved, Save the Case: If Modified During the Calculation or Manually [✓ Each Time]

Retain Only the Most Recent Files

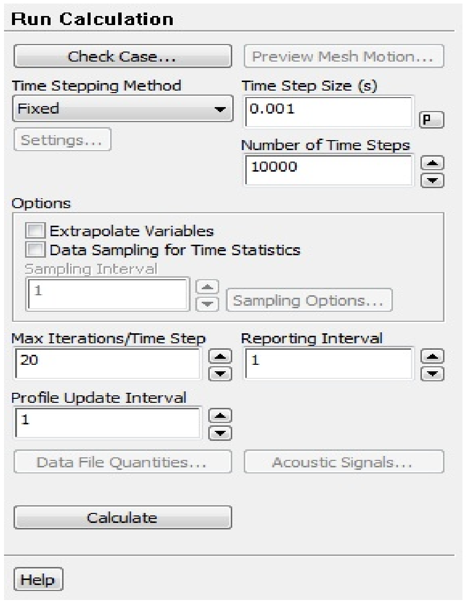
Maximum Number of Data Files: 0

Only Associated Case Files are Retained

File Name: propeller53test.gz

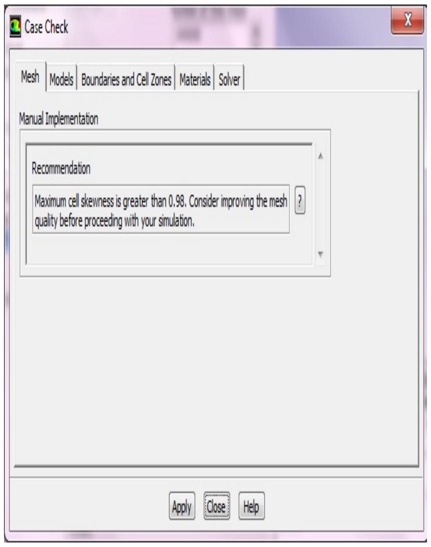
Append File Name with: time-step

Figure 3.28: The setup of the autosave.



Time Stepping Method: Fixed
Time Step Size (s): 0.001
Number of Time Steps: 10000
Extrapolate Variables: unchecked
Data Sampling for Time Statistics: unchecked
Sampling Interval: 1
Max Iterations/Time Step: 20
Reporting Interval: 1
Profile Update Interval: 1

Figure 3.29: The setup of the run calculation.



Maximum cell skewness is greater than 0.98. Consider improving the mesh quality before proceeding with your simulation.

Figure 3.30: The panel of the check case.

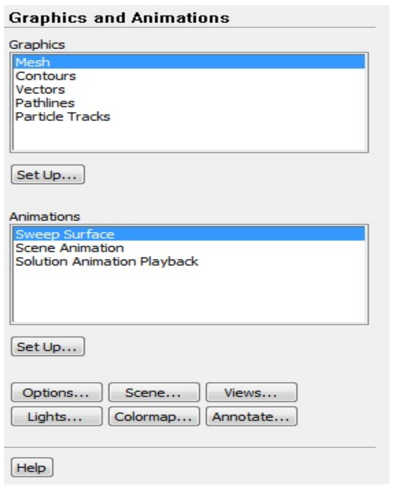


Figure 3.31: The panel of the graphic and animations.

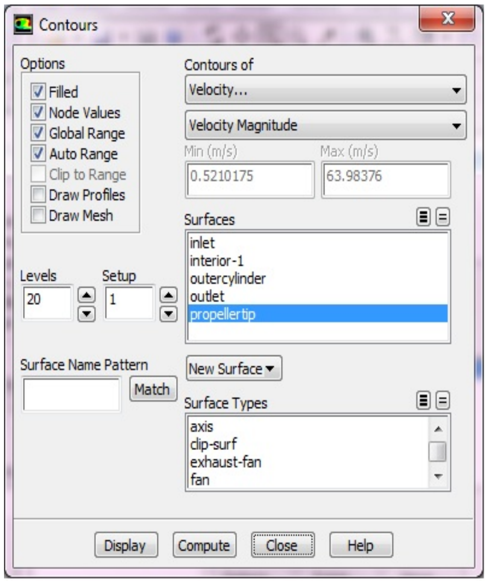


Figure 3.32: The setup of the contours.

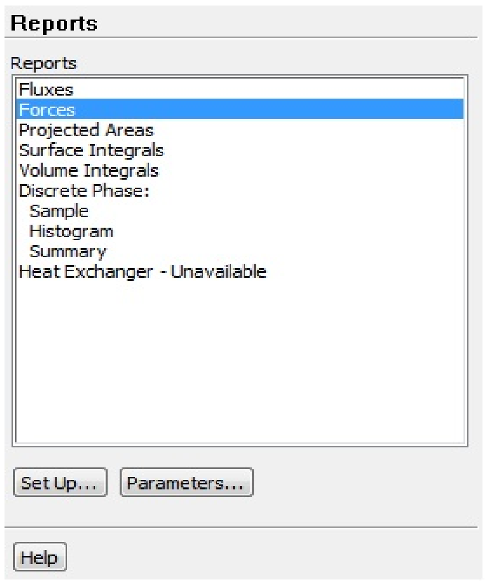


Figure 3.33: The panel of the reports.

4. TURBULENCE MODELING

The fluid motion can be expressed by a mathematical model. The model observes properties of the fluid motion by including three conservation laws: conservation of mass, momentum and energy. The flow is assumed to be incompressible, a continuum, and not moving at relativistic velocities so the motion of fluid substances can be described by the continuity equation and the N-S equations. Since exact solutions to the N-S equations are not straightforward and mostly unavailable, we need to rely on numerical methods to find approximate solutions. There are a lot number of numerical methods. The three classical and important ones are described here:

- Direct numerical simulation (DNS)
- Large eddy simulation (LES)
- Reynolds averaged Navier-Stokes simulation (RANS)
  - one-equation model: Spalart-Allmaras model
  - two-equation models: $k - \epsilon$ models, $k - \omega$ models

4.1 Direct Numerical Simulation (DNS)

The computation of numerical solutions, without the introduction of any additional approximations, except those associated with the numerical algorithms, is called *Direct Numerical Simulation (DNS)*. From a mathematical viewpoint this would seem to be most logically sound, as mathematicians and many other theoreticians normally desire great purity by being truly faithful to the model of problems under treatment. In DNS, all of the motion which are contained in the flow are analyzed. The great majority of mathematical, numerical analysis papers published are of the DNS type.

However, numerical solutions obtained using DNS are of quite limited usefulness. The reason is that fluids exhibit *turbulent behavior.* Turbulence is characterized by *rapid and irregular fluctuations in the fluid properties with a wide range of length and time scales.* DNS computes a turbulent flow by discretizing and solving the N-S equations on a sufficiently fine spatial mesh (spatial scale) along with sufficiently small time steps (temporal scale) to resolve and capture the tiniest eddies and the fastest fluctuations. There is very detailed information in the numerical results from DNS. On one hand, engineers or researchers do not need that much information; on the other hand, it is too expensive to employ DNS often.

The transition of flow from laminar or smooth to turbulent or irregular is determined by the *Reynolds number.* For example, in a pipe, transition occurs at a Reynolds number of approximately 2300, where the Reynolds number in this case is defined as,

$$Re = \frac{\rho U d}{\mu}, \tag{4.1}$$

where, ρ, μ, d and U are the fluid density, molecular viscosity, pipe diameter and average velocity, respectively. Beyond $Re = 4000$ the flow is fully turbulent. The range of length and time scales in a turbulent flow depends on the Reynolds number. Kolmogorov [34] argued that the smallest scales of turbulence should be independent of the largest scales. Dimensional analysis then gives the smallest spatial and time scale, respectively, as

$$\eta = (\frac{\nu^3}{\varepsilon})^{\frac{1}{4}},\ \tau_\eta = (\frac{\nu}{\varepsilon})^{\frac{1}{2}}, \tag{4.2}$$

where ν and ϵ are the fluid kinematic viscosity and average viscous dissipation rate of turbulent energy per unit mass. For turbulence in equilibrium the rate of viscous dissipation at the smallest scales must equal the rate of supply of energy from the large scales. That is, $\varepsilon \sim U^3/L$, where U and L are the largest velocity and length scales of the turbulence. This gives

$$\frac{L}{\eta} \sim \left(\frac{UL}{\nu}\right)^{3/4} = Re^{3/4}. \tag{4.3}$$

Thus, to simulate all scales of motion in a turbulent flow the grid size increases as $Re^{9/4}$. The number of grids is restricted by the storage space and CPU performance of the machine so that DNS can be used possibly only in relatively low Reynolds number problems and in simple geometric domain. Since the Reynolds number in flows of engineering interest are of the order of 10^5 (or much higher in geophysical flows), DNS is of little use in such problems.

To overcome this limitation researchers have resorted to different levels of approximation. This is referred to as *turbulence modeling*. A comprehensive coverage of turbulent flows and turbulence modeling is given by Pope [24]. The two most widely-used approaches are *large eddy simulation (LES)* and *Reynolds-averaged N-S (RANS)* simulations ; cf. e.g., [24], [28].

4.2 Large Eddy Simulation (LES)

Instead of resolving all scales of the eddies as DNS does, the large eddy simulation (LES) approach is to separate the larger and smaller eddies. LES computes and captures the larger eddies with a time-dependent simulation, and discards the smaller eddies in the domain. Although LES reduces computational cost, it still costs a lot of space and memory of the machine .

LES is based on a *spatial average* of the N-S equations by using a *box, Gaussian, or spectral cutoff filter*. This method chooses a filtered function and a cutoff scale to retain the eddies whose length scales are larger than the cutoff scale. The action of turbulent scales smaller than the cutoff scale is modeled by using a *sub-grid scale (SGS)* model. The interaction effects between larger-scale resolved eddies and smaller-scale unresolved eddies cause the SGS stresses. Then, LES performs the spatial filtering operation on the time-dependent flow equations.

In LES, a spatial filtered function is defined by,

$$\bar{\phi}(\mathbf{x},t) \equiv \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G(\mathbf{x},\mathbf{x}',\Delta)\phi(\mathbf{x}',t)\mathrm{d}x_1'\mathrm{d}x_2'\mathrm{d}x_3' \tag{4.4}$$

where $\phi(\mathbf{x},t)$ is the original (unfiltered) function, $G(\mathbf{x},\mathbf{x}',\Delta)$ is a filter function which determines the length scale of all resolved eddies, and Δ is a filter cutoff scale.

The most common three filter functions $G(\mathbf{x},\mathbf{x}',\Delta)$ used in LES are as follows:

(1) box filter:

$$G(\mathbf{x},\mathbf{x}',\Delta) = \begin{cases} \dfrac{1}{\Delta^3} & |\mathbf{x}-\mathbf{x}'| \le \dfrac{\Delta}{2} \\ 0 & |\mathbf{x}-\mathbf{x}'| > \dfrac{\Delta}{2} \end{cases} \tag{4.5}$$

(2) Gaussian filter:

$$G(\mathbf{x},\mathbf{x}',\Delta) = (\frac{\gamma}{\pi\Delta^2})^{3/2}\exp(-\gamma\frac{|\mathbf{x}-\mathbf{x}'|^2}{\Delta^2}) \tag{4.6}$$

(3) spectral cutoff filter:

$$G(\mathbf{x},\mathbf{x}',\Delta) = \prod_{i=1}^{3}\frac{\sin[(x_i-x_i')/\Delta]}{(x_i-x_i')} \tag{4.7}$$

The cutoff scale Δ is used as an indicative measure for the size of the eddies which are kept in the computations and the eddies which are discarded. Δ can be chosen to be any size, however, it is meaningless to choose the size of Δ to be smaller than the size of a grid in the finite volume method of CFD computations. The most common selection of the size of Δ is the size of the grid. In three-dimensional computations, Δ is represented by the cubic root of the volume of the grid cell.

$$\Delta = \sqrt[3]{\triangle x \triangle y \triangle z} \tag{4.8}$$

where $\triangle x$, $\triangle y$, and $\triangle z$ are the length, width, and height of the grid cell, respectively.

For incompressible flow, the filtered N-S equations are:

$$\begin{cases} \dfrac{\partial p}{\partial t} + \mathrm{div}(\rho \bar{\mathbf{u}}) = 0 \quad \text{(LES continuity equation)} \\ \dfrac{\partial(\rho \bar{u_1})}{\partial t} + \mathrm{div}(\rho \overline{u_1 \mathbf{u}}) = -\dfrac{\partial \bar{p}}{\partial x_1} + \ \mu \ \mathrm{div}(\mathrm{grad}(\bar{u_1})) \\ \dfrac{\partial(\rho \bar{u_2})}{\partial t} + \mathrm{div}(\rho \overline{u_2 \mathbf{u}}) = -\dfrac{\partial \bar{p}}{\partial x_2} + \ \mu \ \mathrm{div}(\mathrm{grad}(\bar{u_2})) \\ \dfrac{\partial(\rho \bar{u_3})}{\partial t} + \mathrm{div}(\rho \overline{u_3 \mathbf{u}}) = -\dfrac{\partial \bar{p}}{\partial x_3} + \ \mu \ \mathrm{div}(\mathrm{grad}(\bar{u_3})) \end{cases} \tag{4.9}$$

where ρ is the fluid density, $\mathbf{u} = (u_1, u_2, u_3)$ is the velocity vector, p is the pressure field, and μ is the constant viscosity.

The convective term can be expressed in the following way:

$$\mathrm{div}(\rho \overline{u_i \mathbf{u}}) = \mathrm{div}(\rho \bar{u_i} \bar{\mathbf{u}}) + (\mathrm{div}(\rho \overline{u_i \mathbf{u}}) - \mathrm{div}(\rho \bar{u_i} \bar{\mathbf{u}})) \tag{4.10}$$

After substituting 4.10 in 4.9 and rearranging 4.9, the N-S equations can be rewritten:

$$\frac{\partial(\rho \bar{u_i})}{\partial t} + \mathrm{div}(\rho \bar{u_i} \bar{\mathbf{u}}) = -\frac{\partial \bar{p}}{\partial x_i} + \ \mu \ \mathrm{div}(\mathrm{grad}(\bar{u_i})) - \frac{\partial \tau_{ij}}{\partial x_j} \qquad i, j = 1, 2, 3 \tag{4.11}$$

where τ_{ij} are the SGS stresses which is defined by:

$$\tau_{ij} \equiv \rho\overline{u_i\mathbf{u}} - \rho\bar{u}_i\bar{\mathbf{u}} = \rho\overline{u_iu_j} - \rho\bar{u}_i\bar{u}_j \tag{4.12}$$

The most usual SGS turbulence model which is used to model and approximate the SGS stresses τ_{ij} is *Smagorinsky-Lilly SGS model*. It is an eddy viscosity model. τ_{ij} is modeled:

$$\tau_{ij} = -2\mu_{\mathrm{SGS}}\bar{S}_{ij} + \frac{1}{3}\tau_{kk}\delta_{ij} \tag{4.13}$$

where μ_{SGS} is the SGS eddy viscosity and $\bar{S}_{ij}$ is the rate-of-strain tensor for the resolved flow which is defined by

$$\bar{S}_{ij} \equiv \frac{1}{2}(\frac{\partial\bar{u}_i}{\partial x_j} + \frac{\partial\bar{u}_j}{\partial x_i}) \tag{4.14}$$

The SGS eddy viscosity μ_{SGS} is derived:

$$\mu_{\mathrm{SGS}} = \rho(C_{\mathrm{SGS}}\Delta)^2|\bar{S}| \tag{4.15}$$

where C_{SGS} is Smagorinsky constant and $\bar{S} \equiv \sqrt{2\bar{S}_{ij}\bar{S}_{ij}}$.

Lilly presented a value between 0.17 and 0.21 for C_{SGS} for homogeneous isotropic turbulent eddies in the inertial subrange. These values caused excessive damping of large-scale turbulence, especially in the near-wall regions. Generally speaking, C_{SGS} is not constant. It is a parameter to be determined and may use different values in different flow configurations. However, the value $C_{\mathrm{SGS}} = 0.1$ has been found to be most appropriate and attain the best results for this type of internal flow computation. C_{SGS} can also be computed dynamically. The value of C_{SGS} would be based on the information of the resolved turbulent scales of eddies' motion. This means that C_{SGS} is related to the size of the grid.

The SGS model is usually in the form of an eddy viscosity model that can involve a constant or dynamic coefficient. In the latter case the eddy viscosity or Smagorinsky constant is allowed to vary in space and time and is calculated based on two filterings of the flow variables. Some averaging is generally required for stability. This could be averaging in a homogeneous flow direction or a local spatial average. LES methods are still computationally expensive, though not as much as DNS. This is especially true for wall-bounded turbulent flows since the “large” scales close to the wall can be very small.

4.3 Reynolds Averaged Navier-Stokes Simulation (RANS)

RANS methods are based on the *time or ensemble averaged* N-S equations. The main concept to derive the RANS equations is *Reynolds decomposition* which means that a flow variable is decomposed into the mean (time-averaged) component and fluctuating component. For an incompressible Newtonian fluid, the flow velocity can be represented by:

$$\mathbf{u}(\mathbf{x},t) = \mathbf{U}(\mathbf{x}) + \mathbf{u}'(\mathbf{x},t), \quad \mathbf{x} = (x,y,z) \text{ is the position vector} \tag{4.16}$$

where $\mathbf{U}(\mathbf{x}) = \lim_{T\to\infty} \dfrac{1}{T}\displaystyle\int_0^T \mathbf{u}(\mathbf{x},t)\mathrm{d}t$ and $\mathbf{u}'(\mathbf{x},t)$ are the mean and fluctuating values for the velocity $\mathbf{u}(\mathbf{x},t)$, respectively. The RANS equations can be derived after substituting 4.16, taking the time average, and using properties of the mean operator in the N-S equations:

$$\begin{cases} \mathrm{div}\mathbf{U} = 0 \\ \dfrac{\partial U_1}{\partial t} + \mathrm{div}(U_1\mathbf{U}) = -\dfrac{1}{\rho}\dfrac{\partial P}{\partial x_1} + \nu\mathrm{div}(\mathrm{grad}(U_1)) + \dfrac{1}{\rho}\Big[\dfrac{\partial(-\rho\overline{u_1'^2})}{\partial x_1} + \dfrac{\partial(-\rho\overline{u_1'u_2'})}{\partial x_2} \\ \qquad + \dfrac{\partial(-\rho\overline{u_1'u_3'})}{\partial x_3}\Big] \\ \dfrac{\partial U_2}{\partial t} + \mathrm{div}(U_2\mathbf{U}) = -\dfrac{1}{\rho}\dfrac{\partial P}{\partial x_2} + \nu\mathrm{div}(\mathrm{grad}(U_2)) + \dfrac{1}{\rho}\Big[\dfrac{\partial(-\rho\overline{u_1'u_2'})}{\partial x_1} + \dfrac{\partial(-\rho\overline{u_2'^2})}{\partial x_2} \\ \qquad + \dfrac{\partial(-\rho\overline{u_2'u_3'})}{\partial x_3}\Big] \\ \dfrac{\partial U_3}{\partial t} + \mathrm{div}(U_3\mathbf{U}) = -\dfrac{1}{\rho}\dfrac{\partial P}{\partial x_3} + \nu\mathrm{div}(\mathrm{grad}(U_3)) + \dfrac{1}{\rho}\Big[\dfrac{\partial(-\rho\overline{u_1'u_3'})}{\partial x_1} + \dfrac{\partial(-\rho\overline{u_2'u_3'})}{\partial x_2} \\ \qquad + \dfrac{\partial(-\rho\overline{u_3'^3})}{x_3}\Big] \end{cases} \tag{4.17}$$

where the first equation in 4.17 is the continuity equation for the mean flow and the remaining equations are the time-averaged x_1, x_2, and x_3-momentum equation respectively. Compared with the N-S equations, the process of time averaging method generates some extra stress terms which describe the turbulent motion, which are three normal stresses:

$$\tau_{x_ix_i} = -\rho\overline{u_i'^2},\ i = 1, 2, 3 \tag{4.18}$$

and three shear stresses:

$$\tau_{x_ix_j} = \tau_{x_jx_i} = -\rho\overline{u_i'u_j'},\ i, j = 1, 2, 3,\ i \neq j \tag{4.19}$$

These six extra stress terms are the Reynolds stresses. Since this process results in the appearance of additional terms, the Reynolds stresses which are involving the average of products of the fluctuating velocity (additional terms arise in *compressible* turbulent flows where the density fluctuates), equations must be developed to describe the Reynolds stresses of which there are six independent components.

These are generally differential equations. Hence, turbulence models are here used to predict the Reynolds stresses in order to close the RANS equations.

The majority of RANS models are based on the concept of a *turbulent/eddy viscosity* μ_t. This is a diffusion coefficient, equivalent to the kinematic viscosity of the fluid, that describes the turbulent mixing or diffusion of momentum. Boussinesq suggested in 1877 that the Reynolds stresses are proportional to mean velocity gradients. For incompressible flows, the equation can be written as:

$$\tau_{ij} = -\rho\overline{u_i'u_j'} = \mu_t\Big(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\Big) - \frac{2}{3}\rho k\delta_{ij} \tag{4.20}$$

where $k = \frac{1}{2}\Big(\overline{u_1'u_1'} + \overline{u_2'u_2'} + \overline{u_3'u_3'}\Big)$ is the turbulent kinetic energy per unit mass and $\delta_{ij} = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$ is the Kronecker delta. The relation between the Reynolds stresses and the turbulent viscosity can be represented by **??**. The Reynolds stresses can be solved if we can obtain the turbulent viscosity from other variables in turbulence models. Since the effect of turbulence can be described by a velocity scale ϑ and a length scale ℓ, Dimensional Analysis shows that the turbulent viscosity involves the product of a characteristic turbulent velocity and length scale:

$$\mu_t = C\rho\vartheta\ell, \tag{4.21}$$

where C is a dimensionless constant. *Two-equation turbulence models*, such as the $k-\varepsilon$ and $k-\omega$ models, [28] provide these scales. Here ω is the specific dissipation rate, k and ε were defined earlier. It should be noted that though exact equations can be developed from the equations of motion for these quantities, additional unknown terms arise that must be modeled. One RANS approach should be mentioned. It is the one equation *Spalart-Allmaras model* [28]. This involves a differential equation for the eddy viscosity. It was developed specifically for external aerodynamics problems and is not based on modeling terms in the exact equations but on a more general phenomenological approach. These involve equations (including modeled terms) for the individual Reynolds stress components.

RANS methods involve empirical models with numerous coefficients that must be specified. In general, these coefficients are valid within a particular class of turbulent flow: for example wall-bounded or free shear flows. This is because the turbulent mixing is controlled by the large scale turbulent motions that differ from one class of flow to another.

4.3.1 Standard $k - \varepsilon$ Model

The $k - \varepsilon$ model is a two-equation model and the most widely used turbulence model. The fist $k - \varepsilon$ model was developed by Jones and Launder in 1972. The velocity scale ϑ and the length scale ℓ can be defined by using k and ε:

$$\vartheta = k^{1/2} \quad \text{and} \quad \ell = \frac{k^{3/2}}{\varepsilon}. \tag{4.22}$$

By Dimensional Analysis (4.21), the turbulent viscosity is represented as follows:

$$\mu_t = \rho C_\mu \frac{k^2}{\varepsilon} \tag{4.23}$$

where C_μ is a dimensionless constant. In order to solve for the turbulent viscosity μ_t, two additional equations for k and ε have to be made. By using Boussinesq approximation, the transport equations for k and ε in the standard $k-\varepsilon$ model are:

$$\frac{\partial(\rho k)}{\partial t} + \text{div}(\rho k \mathbf{U}) = \text{div}\Big[\frac{\mu_t}{\sigma_k}\text{grad } k\Big] + 2\mu_t S_{ij}\cdot S_{ij} - \rho\varepsilon \tag{4.24}$$

$$\frac{\partial(\rho \varepsilon)}{\partial t} + \text{div}(\rho \varepsilon \mathbf{U}) = \text{div}\Big[\frac{\mu_t}{\sigma_\varepsilon}\text{grad } \varepsilon\Big] + C_{1\varepsilon}\frac{\varepsilon}{k}2\mu_t S_{ij}\cdot S_{ij} - C_{2\varepsilon}\rho\frac{\varepsilon^2}{k} \tag{4.25}$$

where the constants σ_k and σ_ε are the turbulent Prandtl numbers for k and ε, respectively, $S_{ij} = \frac{1}{2}\Big(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\Big)$ is the mean component of the rate of deformation of the fluid, $C_{1\varepsilon}$, and $C_{2\varepsilon}$ are constants. In these three equations (4.23, 4.24, 4.25), there are five adjustable constants: C_μ, σ_k, σ_ε, $C_{1\varepsilon}$, and $C_{2\varepsilon}$. The values for these five constants for the standard $k-\varepsilon$ model are:

$$C_\mu = 0.09,\ \sigma_k = 1.0,\ \sigma_\varepsilon = 1.3,\ C_{1\varepsilon} = 1.44,\ \text{and } C_{2\varepsilon} = 1.92.$$

The Reynolds stresses can be computed by Boussinesq approximation:

$$-\rho\overline{u_i' u_j'} = 2\mu_t S_{ij} - \frac{2}{3}\rho k \delta_{ij} \tag{4.26}$$

This model is very useful in free-shear layer flows away from boundaries and wake regions. For flows in these situations such as flow with large adverse pressure gradients, unconfined flows, curved boundary layers,and rotating flows, this model performs poorly.

4.3.2 Standard $k-\omega$ Model

The standard $k-\omega$ model has two model equations, one for k and one for ω. It was originally proposed by Wilcox in 1988 and he introduced the concept that the turbulence frequency $\omega = \frac{\varepsilon}{k}$ which has dimension of (second)$^{-1}$. The length scale l

can be written by k and ω:

$$\ell = \frac{k^{1/2}}{\omega}. \tag{4.27}$$

Hence, the turbulent viscosity is modeled:

$$\mu_t = \rho \frac{k}{\omega}. \tag{4.28}$$

The transport equations for k and ω in the standard $k - \omega$ model are:

$$\frac{\partial(\rho k)}{\partial t} + \operatorname{div}(\rho k \mathbf{U}) = \operatorname{div}\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\text{grad }(k)\right] + P_k - \beta^* \rho k \omega \tag{4.29}$$

$$\begin{aligned}\frac{\partial(\rho \omega)}{\partial t} + \operatorname{div}(\rho \varepsilon \mathbf{U}) = \operatorname{div}\left[\left(\mu + \frac{\mu_t}{\sigma_\omega}\right)\text{grad }(\omega)\right] + \gamma_1\left(2\rho S_{ij} \cdot S_{ij} - \frac{2}{3}\rho\omega\frac{\partial U_i}{\partial x_j}\delta_{ij}\right) \\ - \beta_1 \rho \omega^2\end{aligned} \tag{4.30}$$

where $P_k = 2\mu_t S_{ij} \cdot S_{ij} - \frac{2}{3}\rho k \frac{\partial U_i}{\partial x_j}\delta_{ij}$. For the standard $k - \omega$ model, the values for five adjustable constants are:

$$\sigma_k = 2.0,\ \sigma_\omega = 2.0,\ \gamma_1 = 0.553,\ \beta_1 = 0.075, \text{and } \beta^* = 0.09.$$

The Reynolds stresses are computed as usual in (4.26).

This model does not need to use damping function near the wall region so it is known to perform well in the boundary sublayer. It also allows simple Dirichlet boundary conditions to be applied on some specified walls. Hence, we could use this model throughout the boundary layer. However, outside the boundary layer, this model has a quite high sensitivity to the freestream values of ω.

4.3.3 Spalart-Allmaras Model

The Spalart-Allmaras model involves one model equation for kinematic eddy viscosity parameter $\tilde{\nu}$. This model was introduced by Spalart and Allmaras in 1992 and it was designed specifically for aerospace application. The turbulent viscosity is computed by:

$$\mu_t = \rho \tilde{\nu} f_{\nu 1} \tag{4.31}$$

where $\nu = \dfrac{\mu_t}{\rho}$ is the kinematic eddy viscosity with dimension m^2/s, $f_{\nu 1} = \dfrac{\chi^3}{\chi^3 + C_{\nu 1}^3}$ is the wall damping function, and $\chi = \dfrac{\tilde{\nu}}{\nu}$. $f_{\nu 1}$ approaches to unity for high Reynolds numbers and zero at the wall. The transport equation for $\tilde{\nu}$ in the standard Spalart-Allmaras model is:

$$\begin{aligned}\frac{\partial \tilde{\nu}}{\partial t} + u_j \frac{\partial \tilde{\nu}}{\partial x_j} = C_{b1}(1 - f_{t2})\tilde{S}\tilde{\nu} - \Big[C_{w1} f_{w1} \frac{C_{b1}}{\kappa^2} f_{t2}\Big]\Big(\frac{\tilde{\nu}}{d}\Big)^2 + \frac{1}{\sigma}\Big[\frac{\partial}{\partial x_j}\Big((\nu + \tilde{\nu})\frac{\partial \tilde{\nu}}{\partial x_j}\Big) \\ + C_{b2} \frac{\partial \tilde{\nu}}{\partial x_i} \frac{\partial \tilde{\nu}}{\partial x_i}\Big]\end{aligned} \tag{4.32}$$

where

$\tilde{S} = \Omega + \dfrac{\tilde{\nu}}{\kappa^2 d^2} f_{\nu 2}$

$\Omega = \sqrt{2\Omega_{ij}\Omega_{ij}}$ is the magnitude of the vorticity

d is the distance to the closest wall

$\Omega_{ij} = \dfrac{1}{2}\Big(\dfrac{\partial U_i}{\partial x_j} - \dfrac{\partial U_j}{\partial x_i}\Big)$

$f_{\nu 2} = 1 - \dfrac{\chi}{1 + \chi f_{\nu 1}}$

$f_w = g\Big[\dfrac{1 + C_{w3}^6}{g^6 + C_{w3}^6}\Big]^{1/6}$

$g = r + C_{w2}(r^6 - r)$

$$r = \min\Big[\frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2}, 10\Big]$$

$$f_{t2} = C_{t3}\exp(-C_{t4}\chi^2).$$

The value for these ten adjustable constants in the standard Spalart-Allmaras model are:

$$\sigma = \frac{2}{3},\ C_{b1} = 0.1355,\ C_{b2} = 0.622,\ \kappa = 0.41,\ C_{w2} = 0.3,$$
$$C_{w3} = 2,\ C_{\nu 1} = 7.1,\ C_{t3} = 1.2,\ C_{t4} = 0.5,\ C_{w1} = \frac{C_{b1}}{\kappa^2} + \frac{1 + C_{b2}}{\sigma}.$$

This model has a good performance in boundary layers with adverse pressure gradients. It is unsuitable for general internal flows and is insensitive for the processes of transport in rapidly changing flows. However, this model has convincing results for turbomachinery application.

4.4 Conclusion

For these three basic turbulence models in this Section, DNS, LES, and RANS, there are pros and cons. Table 1 is the comparison among these models.

Table 4.1: Comparisons among DNS, LES, and RANS.

Turbulence models	Advantages	Disadvantages
DNS	Most accurate. Doesn't need empirical correlations. Capable of characterizing all the flow details.	Highly computationally expensive. Difficult to include accurate initial and boundary conditions for engineering applications.
LES	Capable of capturing the dynamics of the dominant eddies in the system. Relatively more economical than DNS. More accurate than RANS.	Still computationally intensive. Some difficulties in representing flow in complex geometries.
RANS	Suitable for engineering problems. Computational cost is modest.	Incapable of capturing flow details. High dependence on empirical correlations.

5. NUMERICAL SIMULATION *

5.1 Comparison among Three Turbulence Models DNS, LES, and RANS by 2D and 3D Lid-Driven Flow

In this section, to compare the simulation capability of the three basic turbulence modeling strategies, i.e., DNS, LES, and RANS, a simple *lid-driven* flow is simulated in both two- (2D) and three-dimension (3D) in Cases 1 and 2 ([5]). The lid-driven cavity flow [18] is a classical test problem for N-S codes and benchmarks. Its geometry and boundary conditions are indicated in Fig. 5.1.

The parameter values of this problem are summarized in Table **??**. The turbulent viscosity sub-models chosen for RANS and LES are standard $k - \varepsilon$ model and k-equation eddy-viscosity model, respectively in OpenFOAM [37], [36]. Wall functions [28, pp. 76-78] are applied to turbulent viscosity at all wall types. Computations for all three Cases 1-3 were run on Texas A&M Supercomputing Facility's Eos, an IBM iDataPlex Cluster 64-bit Linux, Intel Nehalem processors.

Case 1. 2D lid-driven flow Graphical results are displayed in Fig. 5.2. The numerical data agree favorably with those in the literature (but we omit the details of comparisons here for lack of space). □

Case 2. 3D lid-driven flow The 3D DNS requires huge resources. Here we used 1024 cores for parallel computing at TAMU Supercomputing Facility to run this case. It took 64 hours to run for the numerical simulation for just 0.15 second.

*Reprinted with permission from "OpenFOAM for Computational Fluid Dynamics" by Goong Chen and Qingang Xiong and Philip J Morris and Eric G Paterson and Alexey Sergeev and Yi-Ching Wang, 2014. Notices of the AMS, 61(4): 354-363, 2014, Copyright [2014] by American Mathematical Society, and Reprinted with permission from "Malaysia Airlines Flight MH370: Water Entry of an Airline" by Goong Chen, Cong Gu, Philip J Morris, Eric G Paterson, Alexey Sergeev, Yi- Ching Wang, and Tomasz Wierzbicki, 2015. Notices of the AMS, 62(4):330-344, 2015. Copyright [2015] by American Mathematical Society.

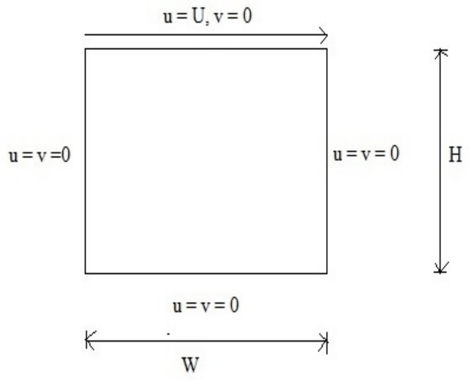


Figure 5.1: Geometry and boundary conditions for a 2D lid-driven cavity, where (u, v) are the components of flow velocity. The case for 3D is similar. Note that the upper "lid" has a constant horizontal velocity U. Note, however, for the 3D DNS computations, because a huge memory space and CPU time are required, the domain has been reduced to the size of [0, 0.1]×[0, 0.1]×[0, 0.01].

Table 5.1: Physical and geometrical parameters for the lid-driven flow simulation.

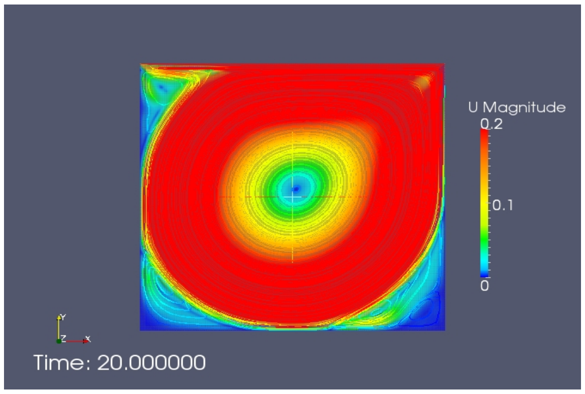
Length L	0.1 m
Height H	0.1 m
Width W	0.1 m
Kinematic viscosity ν	0.00001 m^2s^{-1}
Lid velocity U	1 m/s
Grid length in RANS, LES and DNS Δx	0.005 m, 0.001 m, 0.0002 m
Unit time step in RANS, LES and DNS Δt	0.005 s, 0.001 s, 0.0002 s

The streamline flow pattern computed by DNS at t=0.15 can be seen in Fig. 5.3 part (a).

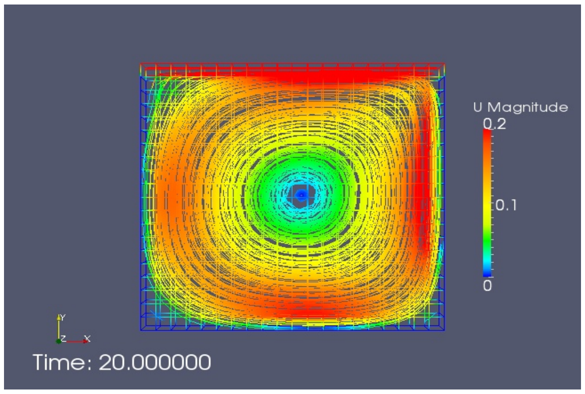
For 3D RANS and LES computations, we are able to compute flow fields up to t=20 sec; see their flow patterns in parts (b) and (c) of Fig. 3.

To *visualize* the *dynamics* of fluid motion, we have made three short animation videos. The dynamic motion of the fluid computed by DNS can be seen by clicking (or pasting)

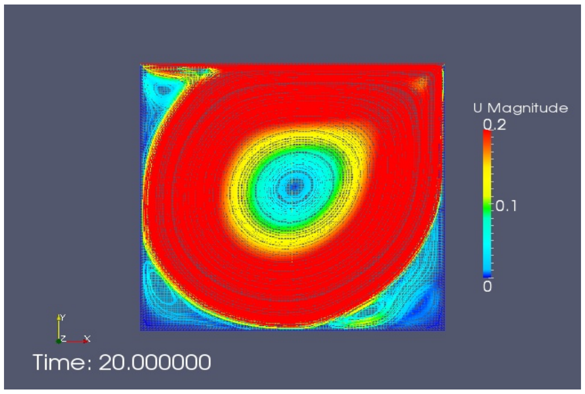
Figure 5.2: 2D lid-driven flow calculations by OpenFOAM.



(a) Flow streamlines at t=20 sec obtained by DNS [35].

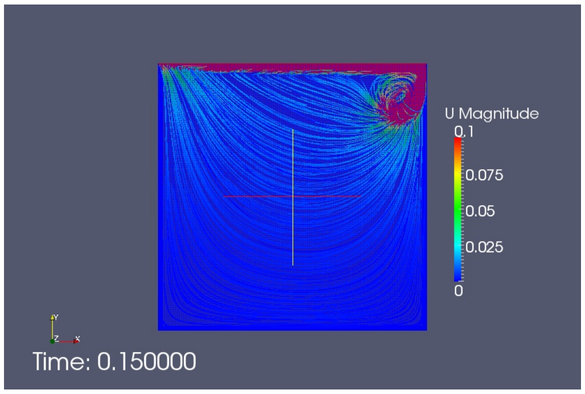


(b) Flow streamlines at t=20 sec obtained by RANS [37].

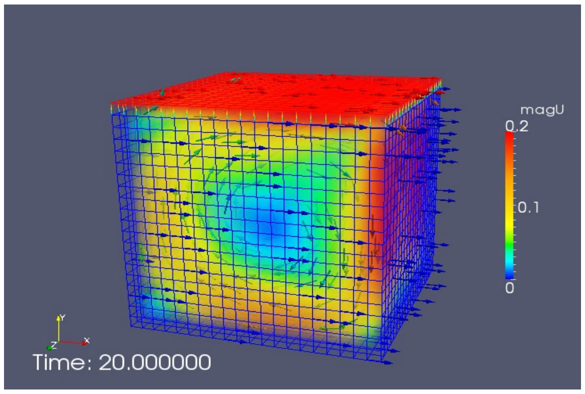


(c) Flow streamlines at t=20 sec obtained by LES [36].

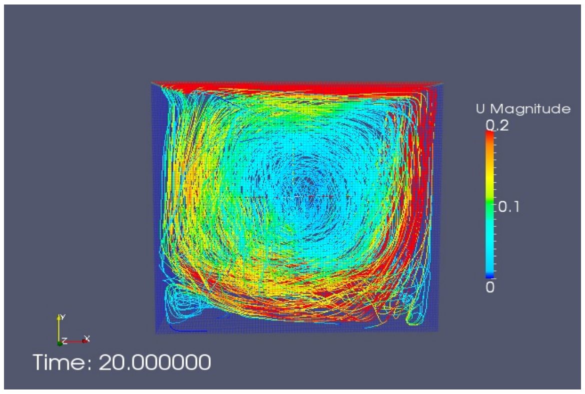
Figure 5.3: 3D lid-driven flow calculations by OpenFOAM. Note that the snapshot of the DNS case in part (a) is at t=0.15, while those in parts (b) and (c) are at t=20. One can rank the richness of fine features of flow in the order of (a), (c) and (b).



(a) Flow field at t=0.15 sec obtained by DNS [35]. (Note that the domain here is [0, 0.1]×[0, 0.1]×[0, 0.01], not a cube as in subcases (b) and (c))



(b) Flow field at t=20 sec obtained by RANS [37].



(c) Flow field at t=20 sec obtained by LES [36].

`https://www.dropbox.com/s/htoms253d3ckt0n/DNS3Dstreamline2.avi/`,
while that by OpenFOAM RANS, containing two different graphical representations: field and streamlines, can be viewed by clicking
`https://www.dropbox.com/s/6cwjsdxrmcnud3o/RANS3Dfiledstreamline.wmv/`.
The counterpart, computed by OpenFOAM LES, can be seen by clicking `https://www.dropbox.com/s/6hzz3ct1wljur9n/LES3Dfiledstreamline.wmv/`. □

5.2 Computational Examples for Wind Turbines

5.2.1 One Wind Turbine with Fixed Angular Velocity

In this section, we consider one wind turbine case with fixed angular velocity. We set up the wind speed to be 8 m/s and angular velocity of the wind turbine to be 75 deg/s. All details of CFD settings in FLUENT and OpenFOAM have been described in Section 2 and 3. A rotating wind turbine is simulated with RANS k-ε turbulence modeling.

Case 1. **Simulation by OpenFOAM** The snapshot of velocity plot at $t = 10\ s$ is shown in Fig. 5.4.

Case 2. **Simulation by FLUENT** The snapshot of velocity plot is shown in Fig. 5.5. The animation can be viewed at the following link:

`https://www.dropbox.com/s/nj566uyvrkxjuax/ansys1.avi?dl=0`

We compare the results of velocity between **Cases 1 and 2** along the line $y = 30,\ z = 0$. The snapshot of velocity plot at $t = 70.59\ s$ can be seen in Fig 5.6. The animation can be viewed at the following link:

`https://www.dropbox.com/s/i8dn60wu1m4ykvz/comparison.avi?dl=0`

We can see velocity distribution for a rotating wind turbine calculated by Open-FOAM and ANSYS in the video. When we use the same physical and modeling parameters in OpenFOAM and FLUENT, both software perform close results.

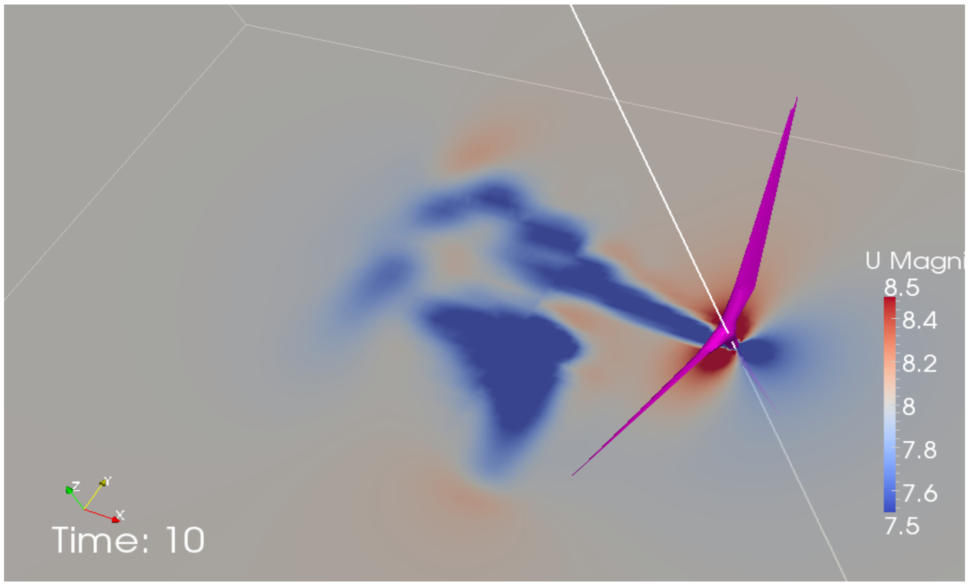


Figure 5.4: A snapshot of velocity plot of a rotating wind turbine with fixed angular velocity at $t = 10$ s calculated by OpenFOAM.

5.2.2 One Wind-Driven Wind Turbine

In this section, we consider one wind-driven wind turbine case. The formula

$$\omega_k = \omega_{k-1} + (\frac{180}{\pi}) \times (\frac{\tau_{k-1}}{I}) \times \triangle t \tag{5.1}$$

is used to make this wind-driven case, where ω_i, τ_i, and I are the angular velocity, torque, and moment of inertia of the wind turbine, respectively. The unit of the angular velocity in FLUENT is deg/s so the value $\frac{180}{\pi}$ is added to the formula to

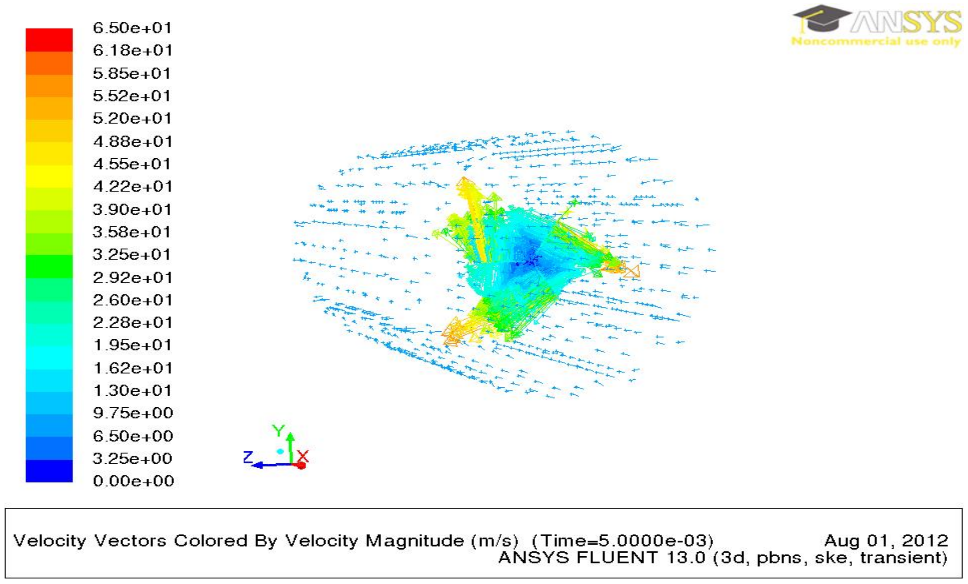


Figure 5.5: A snapshot of velocity plot of a rotating wind turbine with fixed angular velocity calculated by FLUENT.

convert this unit to rad/s. In our case, we use moment of inertia 10^7 and wind speed 8 m/s and 16 m/s. The size of time step is 0.005. The CFD settings of Fluent and OpenFOAM are the same.

Case 1. wind speed 8 m/s For this case, the angular velocity of the wind turbine converges to 14.5 m/s in 17 seconds in FLUENT and 13.9 m/s in 26.6 seconds in OpenFOAM. The snapshots of velocity and pressure plots calculated by Fluent at $t = 20$ s are shown in Fig 5.7 and Fig 5.8, respectively.

The snapshots of velocity and pressure plots calculated by OpenFOAM at $t = 10$ s and $t = 14.06$ s are shown in Fig 5.9 and Fig 5.10, respectively.

The comparison of the angular velocity of the wind turbine between FLUENT and OpenFOAM is shown in Fig 5.11.

Case 2. wind speed 16 m/s For this case, the angular velocity of the wind

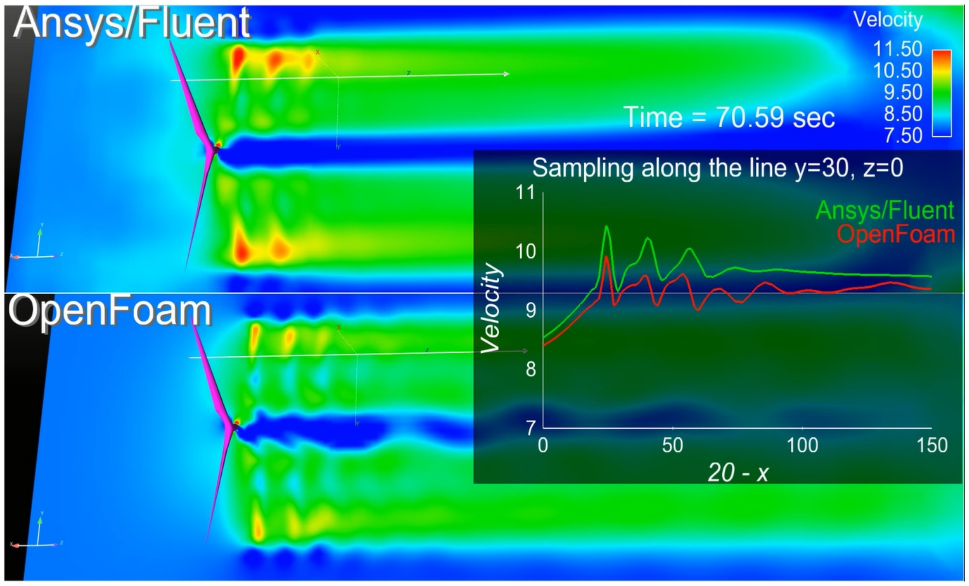


Figure 5.6: A snapshot of the comparison of velocity of a rotating wind turbine with fixed angular velocity calculated by FLUENT and OpenFOAM at $t = 70.59\ s$.

turbine converges to 29.3 m/s in 9 seconds in Fluent and 27.9 m/s in 19.6 seconds in OpenFOAM. The snapshots of velocity and pressure plots calculated by Fluent at $t = 20.5\ s$ are shown in Fig 5.12 and Fig 5.13, respectively.

The snapshots of velocity plot calculated by OpenFOAM at $t = 20\ s$ is shown in Fig 5.14.

The comparison of the angular velocity of the wind turbine between Fluent and OpenFOAM is shown in Fig 5.15.

5.2.3 Two Serial Wind Turbines with Fixed Angular Velocity

In this section, we consider two wind turbines in serial alignment with the distance 60 m. The wind speed is 8 m/s and angular velocity of both two wind turbines are 75 deg/s. The CFD settings are the same as **Case 1** in Section 5.2.1.

A snapshot at $t = 30\ s$ is shown in Fig 5.16. The animation can be viewed at the

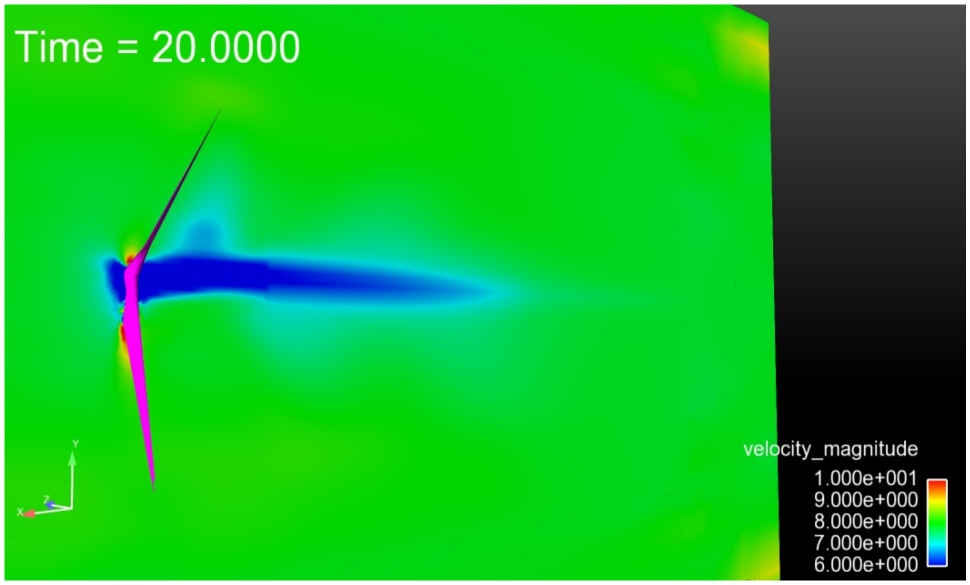


Figure 5.7: A snapshot of velocity plot calculated by FLUENT for wind speed 8 m/s at $t = 20\ s$.

following link:

`https://www.dropbox.com/s/9jqmuzexqzpypqo/ansys2.avi?dl=0`

In the video, we can see that the wake of the front wind turbine reaches the rear one at $t = 7\ s$. The wake interaction of these two in-line rotating wind turbines is diffusing.

The residuals at various iterations can be seen in Fig 5.17. The residual of continuity equation drops to 10^{-3}, while the residuals of equations of x-, y- and z-components of the moment equations reach around 10^{-6}. This level of convergence criterion is considered to be sufficient for our flow simulation.

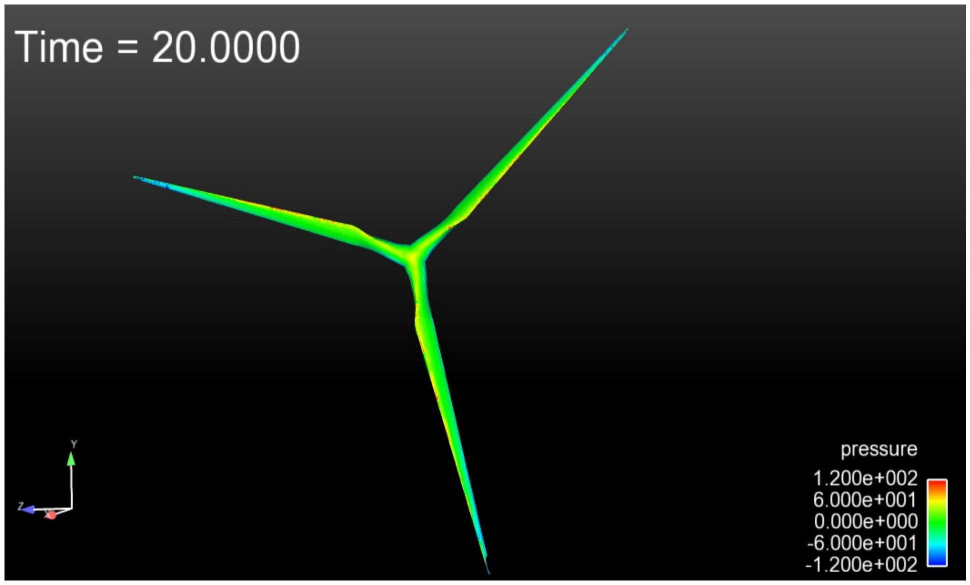


Figure 5.8: A snapshot of pressure plot calculated by FLUENT for wind speed 8 m/s at $t = 20\ s$.

5.3 Simulation of Ditching/Crashing of an Airliner into Water and Mountain

5.3.1 The Water Entry Problem

The problem was motivated by the air incident, Malaysia Airlines Flight MH370 disappeared in the Southern Indian Ocean in March 8, 2014. The prerequisite of this problem is the *water entry problem*.

The *water entry problem* is a classical problem in applied mathematics and fluid dynamics. It considers the dynamic motion of an object upon its entry into the water. The problem was motivated by several applications: the landing of a hydroplane, the entry into water of a rocket or the Apollo module returning from space, and the ditching or crashing of aircraft.

A major contribution to this field was made by the celebrated applied mathe-

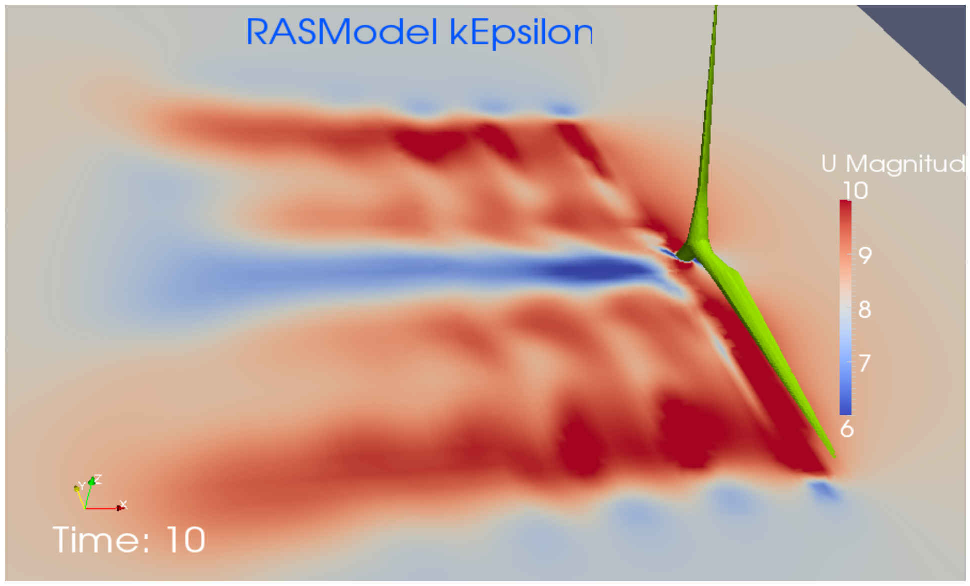


Figure 5.9: A snapshot of velocity plot calculated by OpenFOAM for wind speed 8 m/s at $t = 10$ s.

matician and fluid dynamicist Theodore von Karman (1881-1963). He developed the idea of *"added mass"* (a mass of the fluid that is co-moving with the body) to study the problem [29]; see Figure 5.18. Von Karman inferred that the impact force on the body is related to the instantaneous change of total momentum of the body with its own mass but with an extra mass augmented by the "added mass" of the fluid around the submerged portion of the body. That is,

$$\frac{d}{dt}\Big[(M + m(t))\dot{\zeta}(t)\Big] = Mg - F_B - F_C - F_D \qquad \text{(cf. [2, eq.(2.3)])} \tag{5.2}$$

where M =mass of the projectile, $m(t)$ = "added mass", F_B = buoyancy force, F_C = capillary force, F_D = steady-state drag force, and $\zeta(t)$ = depth of penetration into fluid.

We note that the precise value of added mass $m(t)$ is not known. For small time or

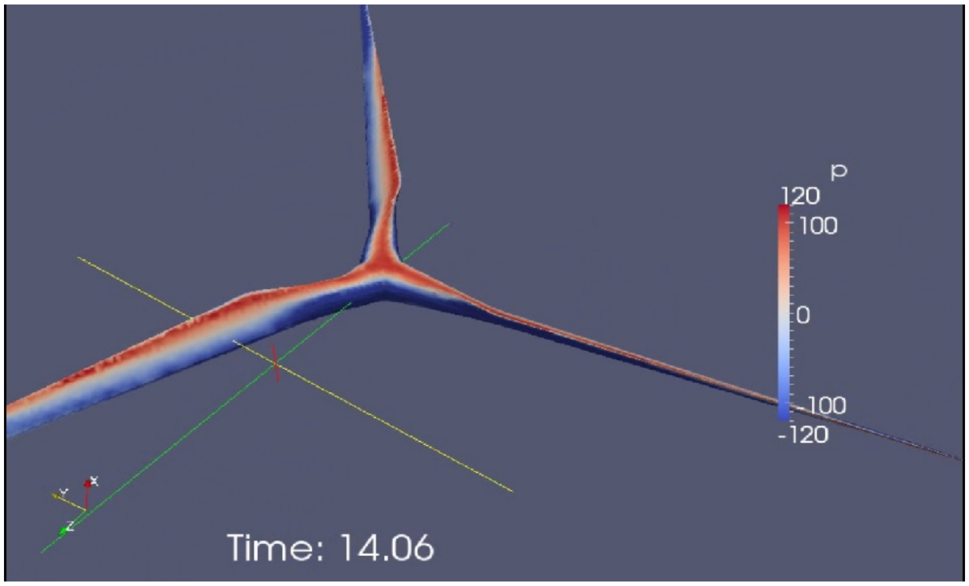


Figure 5.10: A snapshot of pressure plot calculated by OpenFOAM for wind speed 8 m/s at $t = 14.06$ s.

submerged depth upon entry of the body into the water, von Karman estimated the added mass to be half that of a flat plate with the same area as the instantaneous *still* water-plane of the body. Wagner [30] further improved von Karman's work by including the effect of the *pile up* of the water and by associating the added mass with the *wetted* water-plane. Further work such as [7] took account of the submerged geometry for the estimation of the added mass. The analysis and results from these simple approaches are found to compare favorably with experiments for simple geometries such as a wedge or a cone. They also helped the designs of air-to-subsea anti-submarine missiles, for example.

On the mathematical side, papers studying the water entry problem for a two-dimensional (2D) wedge were written by Shiffman and Spencer [27] for a normal incidence problem, and by Garabedian [9] for oblique incidence, for example. These

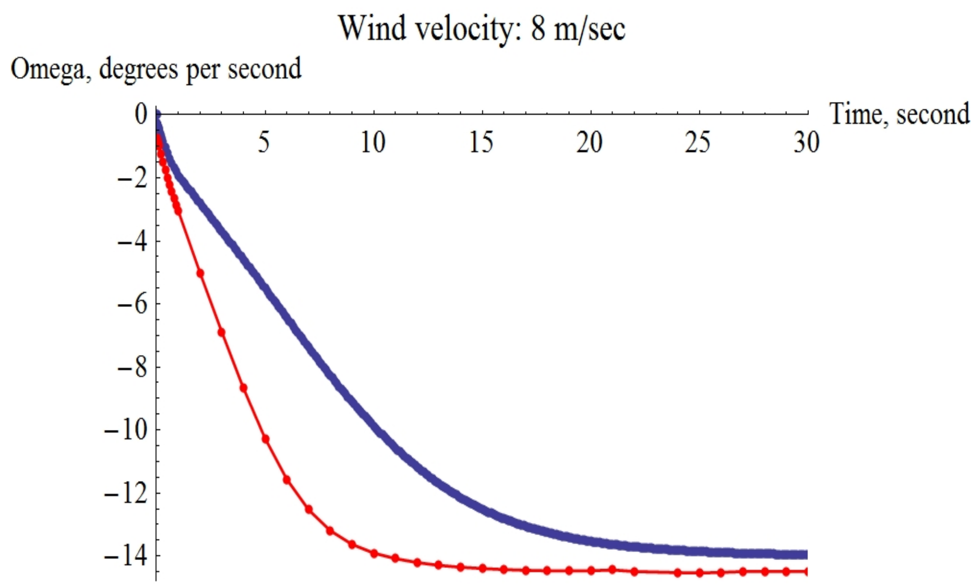


Figure 5.11: The comparison of angular velocity between FLUENT and OpenFOAM for wind speed 8 m/s.

papers treated the case of 2D incompressible, irrotational, inviscid flow by complex variables and potential theory and offered rigorous analysis.

A comprehensive survey of water entry problems (up to the year 2011) can be found in [2], where 476 references are listed, and where a dozen more mathematical (-oriented) papers than [27] and [9] can also be found.

The contributions made by von Karman, Wagner, and others were truly remarkable, and they continue to be used today. However, the physics of water-entry is far more complex to model than the idea of "added mass" alone. In reality, there are several phases of water entry that have been observed in experiments [15]: (1) cavity-opening and jet splashing; (2) cavity-closing and formation of an air pocket; and (3) cavity-detachment and cavitation; see Figure 5.19. A good way to capture the rich physics is through state-of-the-art *computational fluid dynamics (CFD)*. The CFD

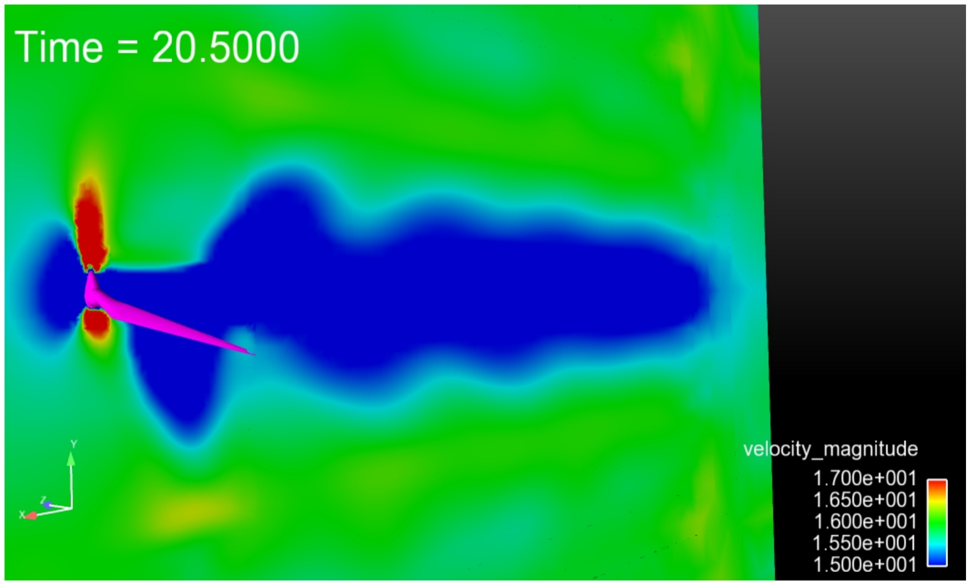


Figure 5.12: A snapshot of velocity plot calculated by FLUENT for wind speed 16 m/s at $t = 20.5$ s.

approach will enable us to simulate water entry for *complex, general geometries* than the simplified ones such as cones, cylinders and wedges treated in the early era by encompassing (5.2) naturally into the two-phase fluid-structure interaction models.

5.3.2 Damage and Breakup

Aircraft crashworthiness and human survivability are of utmost concerns in any emergency landing situation. The earth is covered 71% by water and many major airports are situated oceanside. Therefore, the Federal Aviation Administration (FAA) requires all aircraft be furnished with life vests and the pilots be given water-landing guidlines and manuals.

Assume that an aircraft such as MH370 did not have a mid-air explosion. Then all available signs indicate that it crashed somewhere in the Indian Ocean. This is an aircraft water-entry problem.

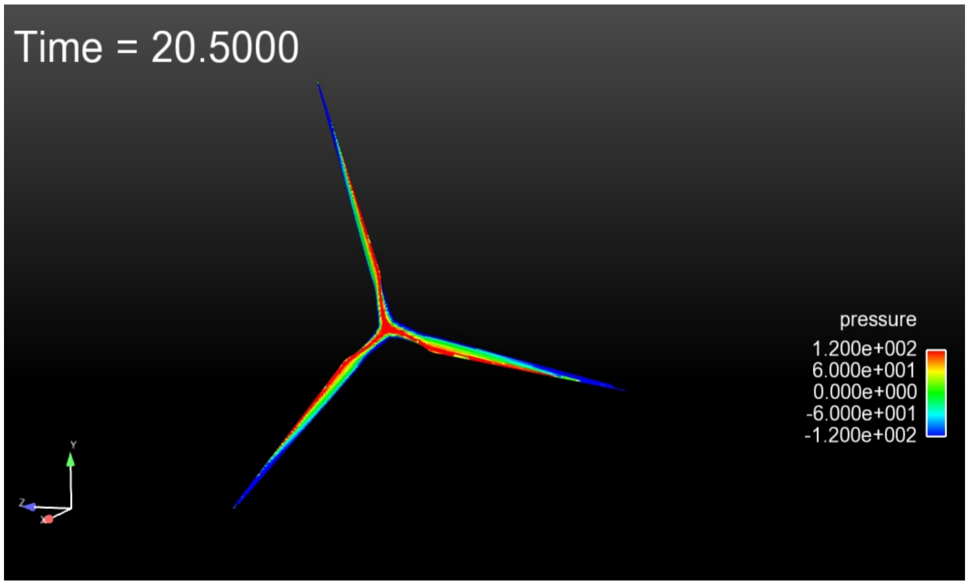


Figure 5.13: A snapshot of pressure plot calculated by FLUENT for wind speed 16 m/s $t = 20.5$ s.

The underpinning subject of this study is *continuum mechanics*, including the water-entry problem first as *fluid-structure interaction with a free fluid-gas interface* and the subsequent impact and structural failure analysis.

As described in the Introduction, not all emergency water landings end in disaster. The dramatic successful landing in the "Miracle on the Hudson" is such a case. The fact that no lives were lost is a testament to the experience and fast thinking under pressure of the captain and crew. The aircraft had a hole ripped open but was otherwise structurally virtually intact. The speed of the aircraft at ditching was estimated to be 150 mph (240 km/hr or 67 m/sec). It was deemed by NTSB as "the most successful ditching in aviation history." [42]

In addition to the Comoros Island air disaster, there is another ditching effort, whose outcome was not so fortunate as the US Airways flight 1549. On August 6,

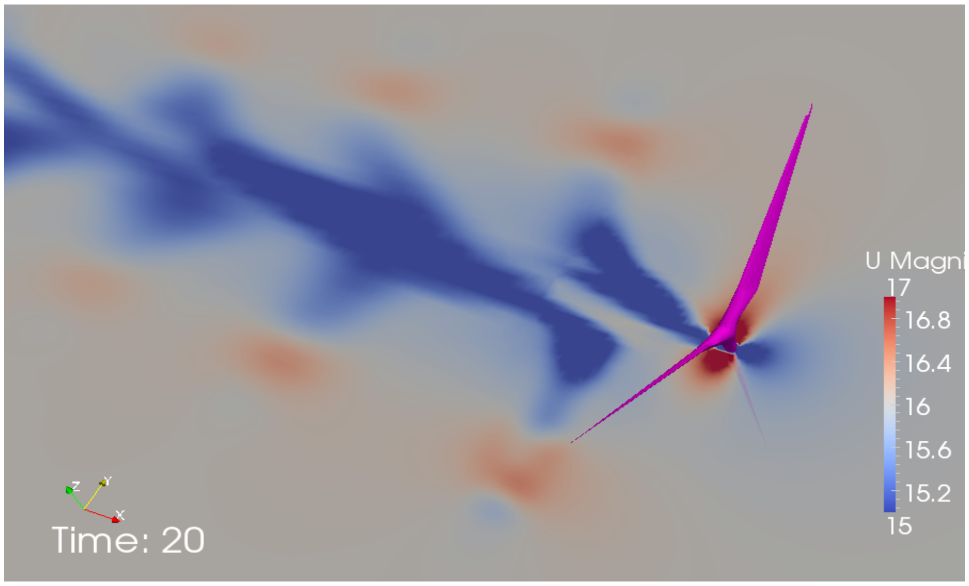


Figure 5.14: A snapshot of velocity plot calculated by OpenFOAM for wind speed 16 m/s at $t = 20$ s.

2005, a Tuninter Airlines Flight 1153 ATR-72 aircraft, flying from Bari International Airprt, Bari, Italy, to Djerba-Zarzis Airport, in Djerba, Tunisia, ran out of fuel and ditched into the Mediterranean 43 km northeast of Palermo, Italy. Upon impact, the aircraft broke up into three pieces. Sixteen persons out of the thirty nine passengers and crew died. Eight of the deaths were actually attributed to drowning after the bodily injuries from impact.

The effect of *Rupture and structural disintegration* are almost certain to happen upon the entry of the aircraft into water when the speed is sufficiently high. This happened even in the miracle on the Hudson case with smooth gliding. The study of impact damage and breakup belongs to a field called *impact engineering*, which is based on the *plasticity* and *fracture* properties of solids that are totally different from fluid dynamics.

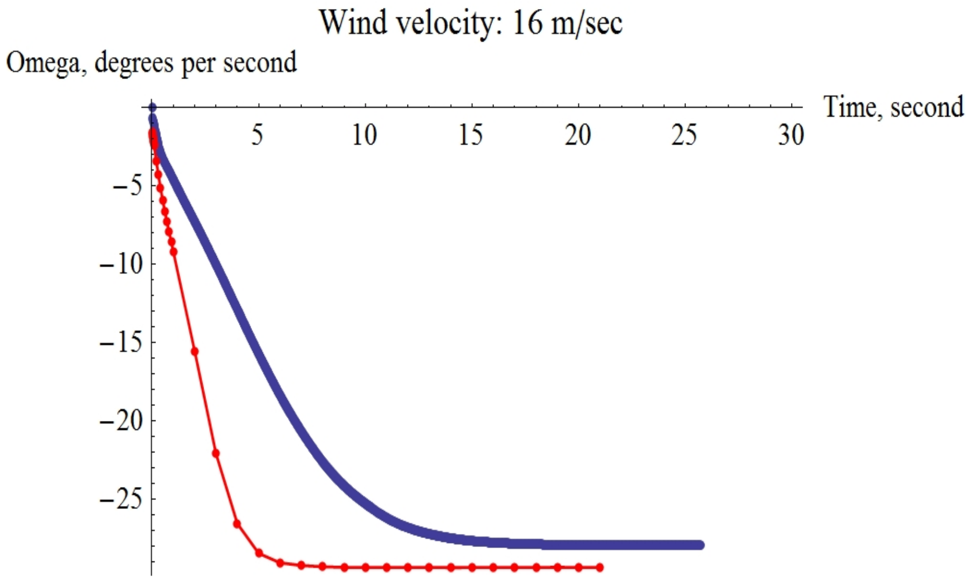


Figure 5.15: The comparison of angular velocity between FLUENT and OpenFOAM for wind speed 16 m/s.

For impact effects, one famous example, the disaster of the Space Shuttle Challenger, can be used to understand what may happen, based on the analysis of one of the coauthors (Wierzbicki) in [32, 31].

The airframe of the Space Shuttle Challenger, an assemblage of ring and stringer-stiffened panels, was constructed essentially like a wide-body Boeing 747 airliner. This in turn is similar to a wide-body aircraft such as the example Boeing 777 under discussion here. Thus, we expect that much of the material and structural failure analysis performed in [32, 31] for Challenger continues to hold.

There is a distinction between the following:

(i) *global failure* mode of fuselage, caused by *large contact forces* between water and structure;

(ii) *local failure* mode due to *excessive pressure.*

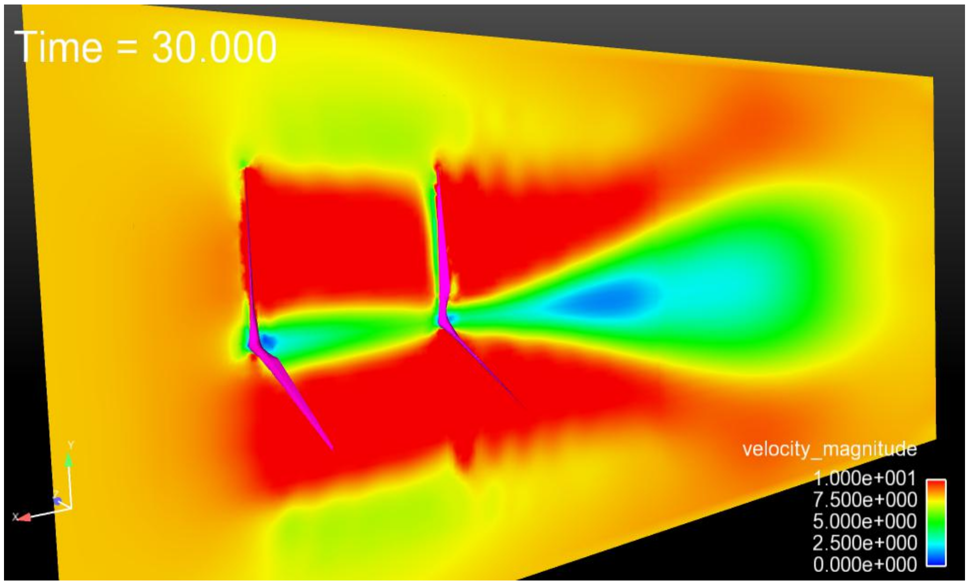


Figure 5.16: A snapshot of the flow of two in-line rotating wind turbines at $t = 30\ s$.

Both such contact forces and pressure vary spatially and temporally. They are obtained from the CFD part of the solution in the preceding Section and used to assess the damage. In the analysis of global failure, simple structural models of *beams and rods* are used for the fuselage. In what follows, we give a quick review of how to study structural breakup upon impact, but defer the more technical study to a sequel.

A flying aircraft was modeled in [32] as a *free-free beam* and with known spatial and temporal variation of external loading, where the distribution of bending moments can be uniquely found from the equations of dynamic equilibrium. Thence, the *maximum cross-sectional bending moment* can be compared with the fully plastic bending capacity of the fuselage. This will indicate the *onset* of structural collapse and break up.

The local failure mode is composed of tearing of fuselage skin, tensile and shear

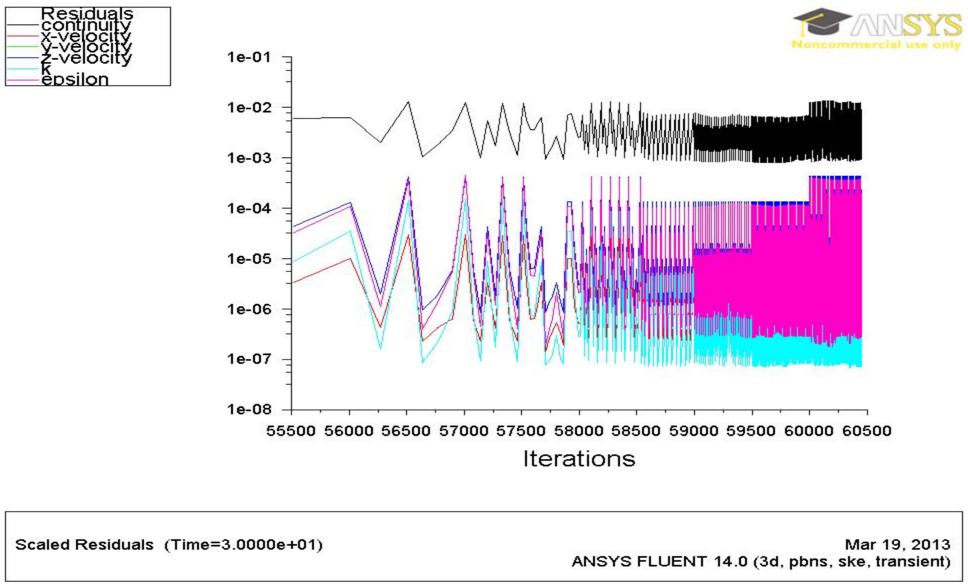


Figure 5.17: Residues at various iterations.

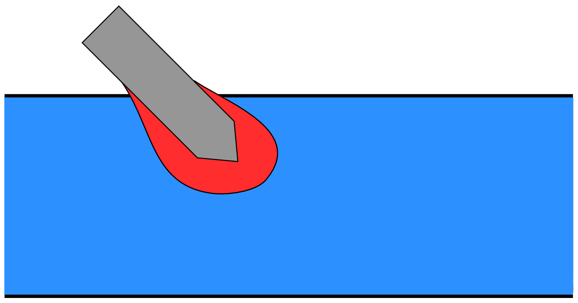


Figure 5.18: Von Karman's idea of "added mass" for the water entry problem, which is an idealization and simplification. Here the red region represent "added mass". This is the mass moving together with the mass of the wedge projectile. The portion of the (red) added mass lying above the still water surface is called the "pile up".

rupture of the system of stringers and ring frames; cf. Figure 5.20. Depending on the impact velocity, the local failure can involve progressive buckling and folding of the fuselage or fragmentation. Such failure modes occur at *low impact velocities*, as has been demonstrated with a real model of a retired aircraft in DYCAST (Dynamic Crash Analysis of Structures) by NASA [8]. These findings were published nearly

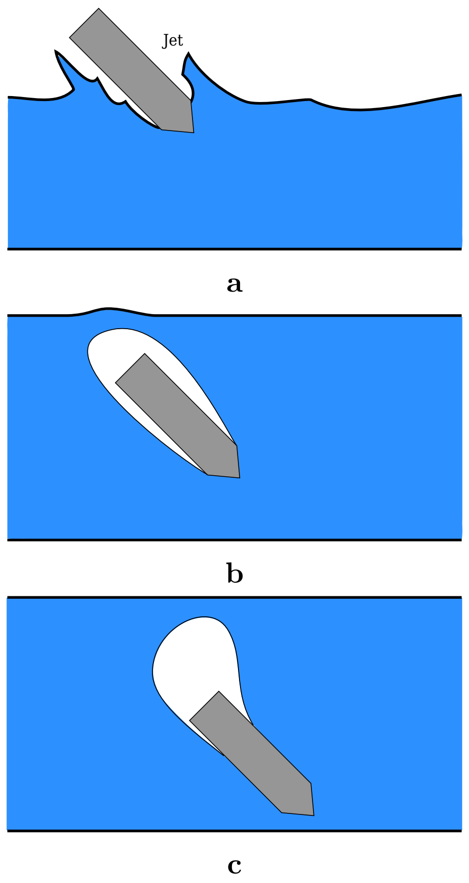


Figure 5.19: The several phases of a projectile entering water according to Mackey [15]: a a cavity of air opens; b a cavity of air pocket encloses the projectile when it is totally submerged; and c the cavity begins to be detached from the projectile, leaving it totally surrounded by water. Some water vapor may exist in the cavity, and cavitation usually happens. (Adapted from [2, p. 060803-2])

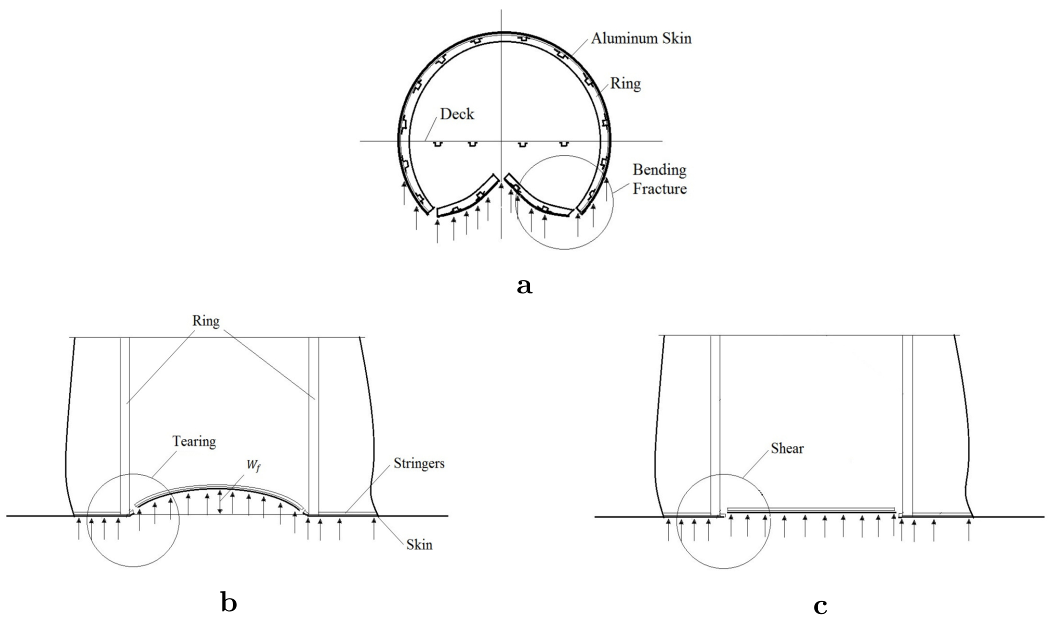


Figure 5.20: Three modes of structural failure for a wide-body airliner: a flexural failure of rings; b tearing fracture; and c shear of the longitudinally stiffened shell. (Adapted from [32, p. 651])

three decades ago but remain valid today.

Fracture failure mode is estimated to happen when the vertical component of velocity exceeds certain critical value V_{cr}. Rupture of fuselage and wings as shear and tensile cracks will be initiated and then propagate through the stiffened shell, leading to global structural failure. This is a dynamic process whose analysis is very challenging. Nevertheless, a simple estimate on the onset of local failure can be given using the condition of dynamic continuity in uniaxial wave propagation along a rod based on the equation

$$[\sigma] = \rho c[u], \tag{5.3}$$

where $[\sigma]$ and $[u]$ denote jump discontinuities across the water-structure interface,

$\rho = 2.8$ g/cm^3 is the mass density of the aluminum fuselage and $c = \sqrt{E/\rho}$ is the speed of the uniaxial wave propagation in an elastic rod with elastic modulus $E = 85$ GPa (i.e., 10^9 Pascal). The *critical impact velocity* V_{cr} (vertical component only) is reached where *the stress equals to the yield stress* of the material σ_y. Thus, from (5.3) one gets the following estimate on V_{cr}:

$$V_{cr} = \frac{\sigma_y}{E} c. \tag{5.4}$$

Depending on the material, the critical descending speed of aircraft is normally in the range of $V_{cr} = 15$–20 m/sec. A common fuselage material is 2024 T351 aluminum alloy with the yield stress of $\sigma_y = 324$ MPa (10^6 Pascal). The critical impact velocity is thus $V_{cr} = 22$ m/sec, which is close to the value 18.8 m/sec predicted for the water ditching of the Space Shuttle Challenger, but using a different approach in [32].

The vertical component V_{cr} of V_0, the aircraft speed at ditching, is related through the angle of approach β by

$$\sin\beta = \frac{V_{cr}}{V_0}. \tag{5.5}$$

Therefore, it is essential to keep the angle of approach small, especially when ditching with a high speed.

In addition to structural rupture and disintegration, the *acceleration due to free fall* and the *deceleration due to the impact of the structure* are important for human survival in a crash. In [32], it was analyzed that if the vertical component of the terminal impact velocity lies in the range of *62.5 m/sec and 80.5 m/sec*, maximum decelerations could reach in the order of *100g to 150g* (g is the gravitational acceleration constant) over a short period of time, within a regime labeled "severe injuries" [32, 16] by NASA.

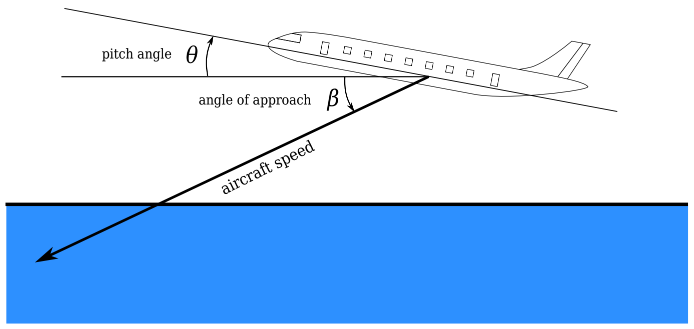


Figure 5.21: Angle θ here is the pitch angle signified and β is the angle of approach. The speed of the aircraft denotes the speed of its center of mass.

Aviation experts generally agree that *how the airliner enters the water determines its breakup*, which then gives major clues and directions of the search operations [25]. In [4], the assessment is the *nose-dive water-entry, or a water-entry with a steep pitch angle* , is the most likely scenario for the final moments of Flight MH370.

On March 24, 2015, one deliberate air incident, Germanwings Flight 9525, an Airbus A320-200, crashed into the French Alps during the fight from Barcelona to D$\ddot{u}$sseldorf. The co-pilot locked the captain out of the cockpit during the flight and began a rapid descent intensionally. The aircraft descended from 38000 ft to 6000 ft in 8 minutes and crashed into the mountain peak with 430 mph. Due to this high speed, the impact was very hard so there were no big pieces like wings or cockpit, only a lot of little pieces in the crash site. All 150 passengers and crew members were killed in this crash. The objective is to conduct numerical simulations of aircraft crashing into a wall by using ANSYS Explicit Dynamics and LS-DYNA.

5.3.3 Results and Simulation

The representative aircraft model we use here is Airbus A320. We use the aircraft speed 200 m/s. The physical parameter values of the aircraft are listed in Table 5.2

([33]). A common fuselage material is 2024 T351 aluminum alloy. For our cases, we add three material models, Density, Isotropic Elasticity, and Johnson-Cook Strain Model ([3] and [26]). The values of these material parameters are listed in Table 5.3

5.3.3.1 ANSYS Explicit Dynamics

ANSYS Explicit Dynamics is a computational structural dynamics (CSD) software package developed by vendor ANSYS to simulate the impacts or short-duration high-pressure loadings problems. It uses the Finite Element Method to solve the governing equations.

The values of Aluminum 2024 T351's properties need to be entered in Engineering Data. After adding the name of new material, we need to define this new material model. Three physical properties, Density, Isotropic Elasticity, and Johnson-Cook Strength Model ([3] and [26]), are included to complete the material definition (Fig. 5.24). The material concrete can be added from General Materials in Engineering Data Sources.

We build geometry of the aircraft and a mountain in *Gometry* panel under Explicit Dynamics. The geometry of the aircraft can be seen in Fig. 5.22. After completing the aircraft geometry file in Geometry panel, all values of parameter for the model can be set up in *Model* panel. In Mechanical panel, the materials for the aircraft and the mountain can be chosen from Assignment under Material in Details of those geometry files (Fig 5.25). Under Mesh branch, several methods can be chosen to generate mesh for the geometry files (Fig 5.26). We use *Face Sizing method* with element size 0.1 m to make mesh for the aircraft and *Body Sizing method* with element size 0.4 m for the wall. The snapshot of the mesh layout for this model is shown in Fig 5.23. The total nodes and elements are 803993 and 776687, respectively.

We insert Fixed Support and Velocity under Explicit Dynamics setting. The body of the mountain is chosen in Fixed Support. In Velocity, we choose the geometry to be the aircraft and set X, Y, and Z Components to be 200 m/s, Free, and Free, respectively.

Step controls, solver control, erosion control, and output control can de defined under the Analysis Settings (Fig 5.27). We only set up the end time and keep all other values of parameters as default values.

The plot requests of deformation, strain, and stress can be inserted under the Solution branch for the postprocessing of calculation (Fig 5.28). After adjusting the settings in each branch, the simulation can be solved by hitting the Solve icon. The calculation process can be observed and monitored by selecting Solver Output and Energy Summary under Solution Information.

All computations and simulations were performed on Texas A&M Supercomputing Facility's Eos which is an IBM iDataPlex Cluster 64-bit Linux with Intel Nehalem processors. It takes 30 CPU hours to calculate 0.03 second with 8 CPUs for these cases.

We consider three cases here, the angle of approach $\beta = 0°$, $\beta = 15°$, and $\beta = 30°$.

A snapshot of comparison of the Total Deformation, and Equivalent Elastic Strain presentations of postprocessing for these three cases are shown in Fig 5.29, Fig 5.30, and Fig 5.31, respectively. A snapshot of comparison of mesh layout of postprocessing for these three cases can be seen in Fig 5.32. The comparison of total deformation, equivalent elastic strain, and equivalent stress curves of postprocessing are shown in Fig 5.33, Fig 5.34, and Fig 5.35, respectively. The comparison of energy and momentum among these three cases are shown in Fig. 5.36 and Fig. 5.37. The dynamic motions of the animation videos for these three cases can be viewed in the following link:

`https://www.dropbox.com/s/yli3sv3pihdiu4o/case1g.wmv?dl=0`

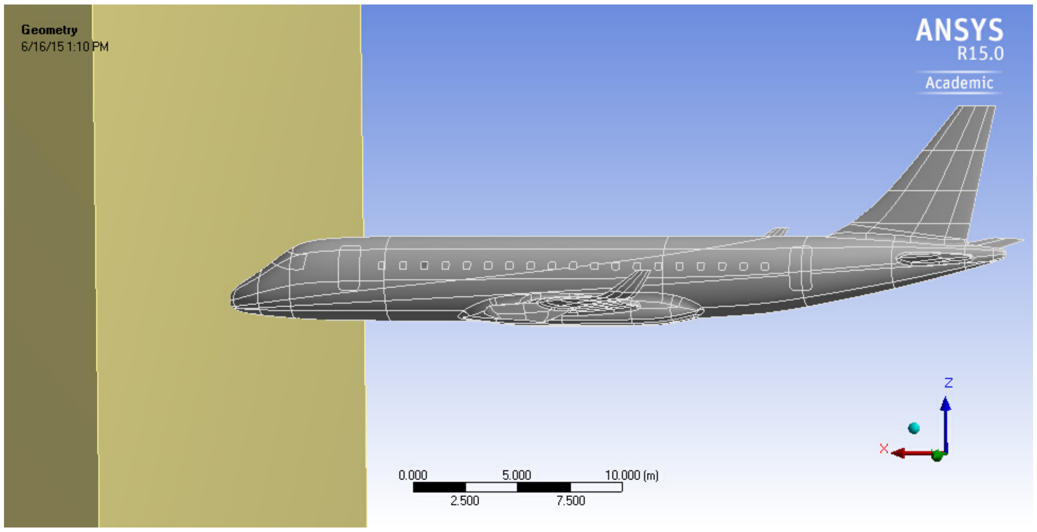


Figure 5.22: The geometry of Airbus A320 and a wall.

Total weight	6.45×10^4 kg
Wing span	35.8 m
Height	11.76m
Overall length	37.57 m

Table 5.2: Physical parameters for Airbus A320 used in CSD calculations.

`https://www.dropbox.com/s/0f8t8mnqj7b2yi4/case2.wmv?dl=0`

`https://www.dropbox.com/s/lcm84217e7weeap/case3.wmv?dl=0`

In the videos, we didn't see too much breakup of aircraft. We still need to adjust parameters of Aluminum Alloy 2024 T351 material models to make our cases more close to the real case.

5.3.3.2 LS-DYNA

LS-DYNA is a finite element analysis software package developed by Livermore Software Technology Corporation (LSTC) to solve complex real world problems. It combines both implicit and explicit solvers. It provides the capabilities to simulate

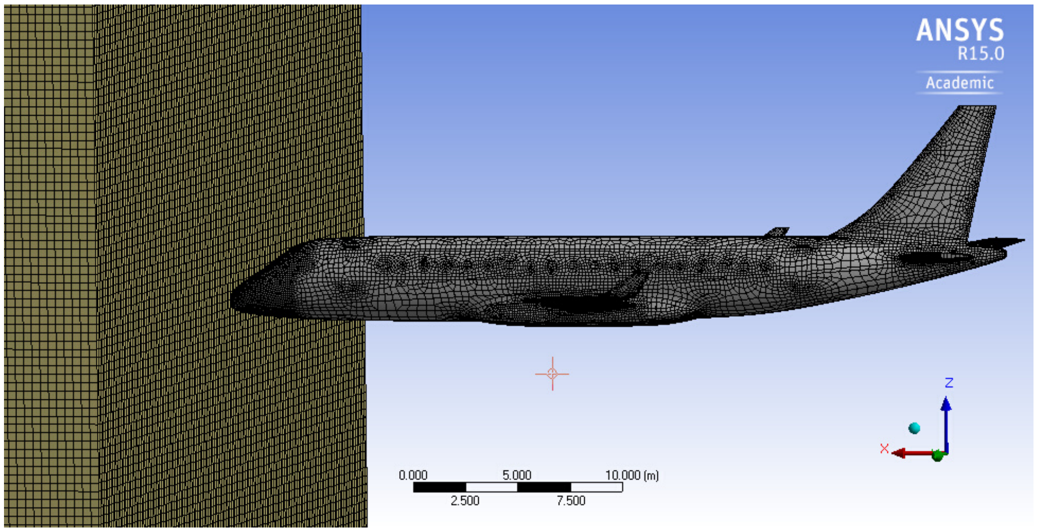
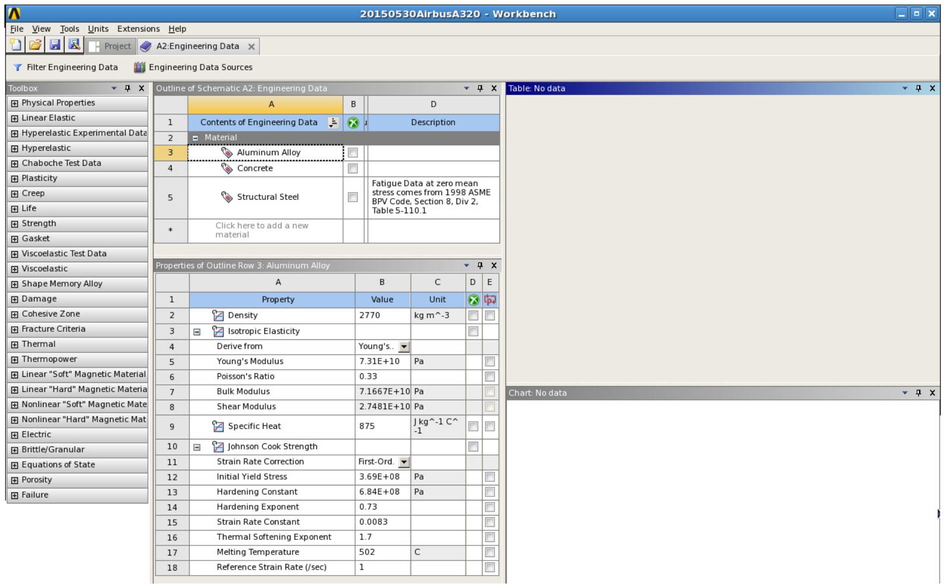


Figure 5.23: The mesh layout of Airbus A320 and a wall.

Density	2770 kg m^{-3}
Isotropic Elasticity	
Young's Modulus	7.31E+10 Pa
Poisson's Ratio	0.33
Specific Heat	875 J kg^{-1} C^{-1}
Johnson-Cook Strength	
Strain Rate Correction	First-Order
Initial Yield Stress	3.69E+08 Pa
Hardening Constant	6.84E+08 Pa
Hardening Exponent	0.73
Strain Rate Constant	0.0083
Thermal Softening Exponent	1.7
Melting Temperature	502 C
Reference Strain Rate (/sec)	1

Table 5.3: Material parameters chosen in CSD calculations .



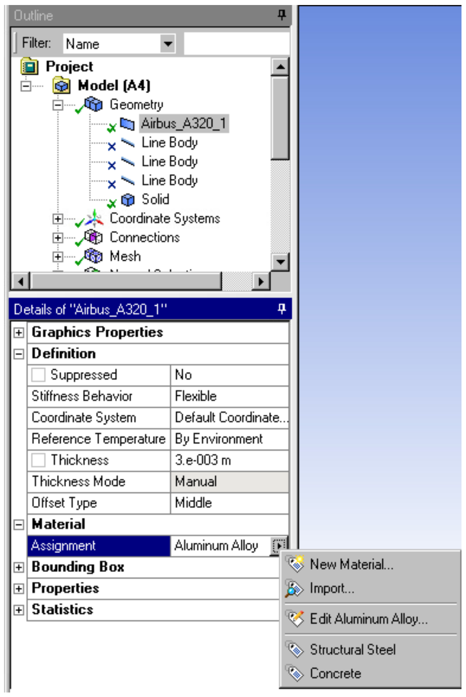
Outline of Schematic A2: Engineering Data

	A	B	D
1	Contents of Engineering Data		Description
2	Material		
3	Aluminum Alloy	☐	
4	Concrete	☐	
5	Structural Steel	☐	Fatigue Data at zero mean stress comes from 1998 ASME BPV Code, Section 8, Div 2, Table 5-110.1
*	Click here to add a new material		

Properties of Outline Row 3: Aluminum Alloy

	A	B	C	D	E
1	Property	Value	Unit		
2	Density	2770	kg m^-3	☐	☐
3	Isotropic Elasticity			☐	
4	Derive from	Young's..			
5	Young's Modulus	7.31E+10	Pa		☐
6	Poisson's Ratio	0.33			☐
7	Bulk Modulus	7.1667E+10	Pa		☐
8	Shear Modulus	2.7481E+10	Pa		☐
9	Specific Heat	875	J kg^-1 C^-1	☐	☐
10	Johnson Cook Strength			☐	
11	Strain Rate Correction	First-Ord.			
12	Initial Yield Stress	3.69E+08	Pa		☐
13	Hardening Constant	6.84E+08	Pa		☐
14	Hardening Exponent	0.73			☐
15	Strain Rate Constant	0.0083			☐
16	Thermal Softening Exponent	1.7			☐
17	Melting Temperature	502	C		☐
18	Reference Strain Rate (/sec)	1			☐

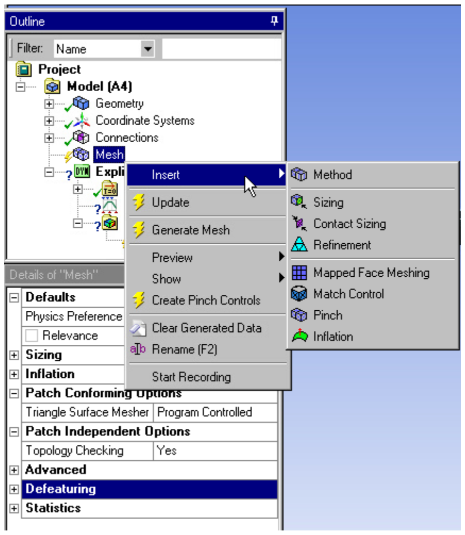
Figure 5.24: The physical properties of aluminum alloy in engineering data.



Details of "Airbus_A320_1"

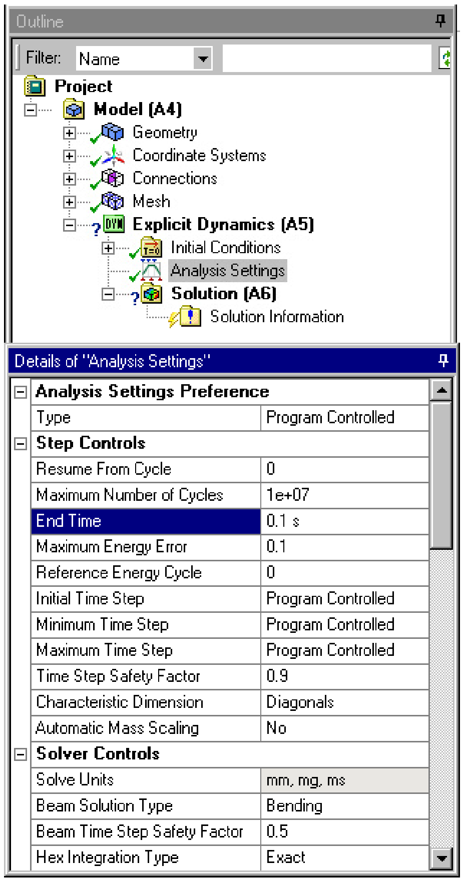
Graphics Properties	
Definition	
Suppressed	No
Stiffness Behavior	Flexible
Coordinate System	Default Coordinate...
Reference Temperature	By Environment
Thickness	3.e-003 m
Thickness Mode	Manual
Offset Type	Middle
Material	
Assignment	Aluminum Alloy
Bounding Box	
Properties	
Statistics	

Figure 5.25: Material settings of the geometry files.



Defaults	
Physics Preference	
Relevance	
Sizing	
Inflation	
Patch Conforming Options	
Triangle Surface Mesher	Program Controlled
Patch Independent Options	
Topology Checking	Yes
Advanced	
Defeaturing	
Statistics	

Figure 5.26: Methods for mesh generation.



Details of "Analysis Settings"

Analysis Settings Preference	
Type	Program Controlled
Step Controls	
Resume From Cycle	0
Maximum Number of Cycles	1e+07
End Time	0.1 s
Maximum Energy Error	0.1
Reference Energy Cycle	0
Initial Time Step	Program Controlled
Minimum Time Step	Program Controlled
Maximum Time Step	Program Controlled
Time Step Safety Factor	0.9
Characteristic Dimension	Diagonals
Automatic Mass Scaling	No
Solver Controls	
Solve Units	mm, mg, ms
Beam Solution Type	Bending
Beam Time Step Safety Factor	0.5
Hex Integration Type	Exact

Figure 5.27: Analysis settings.

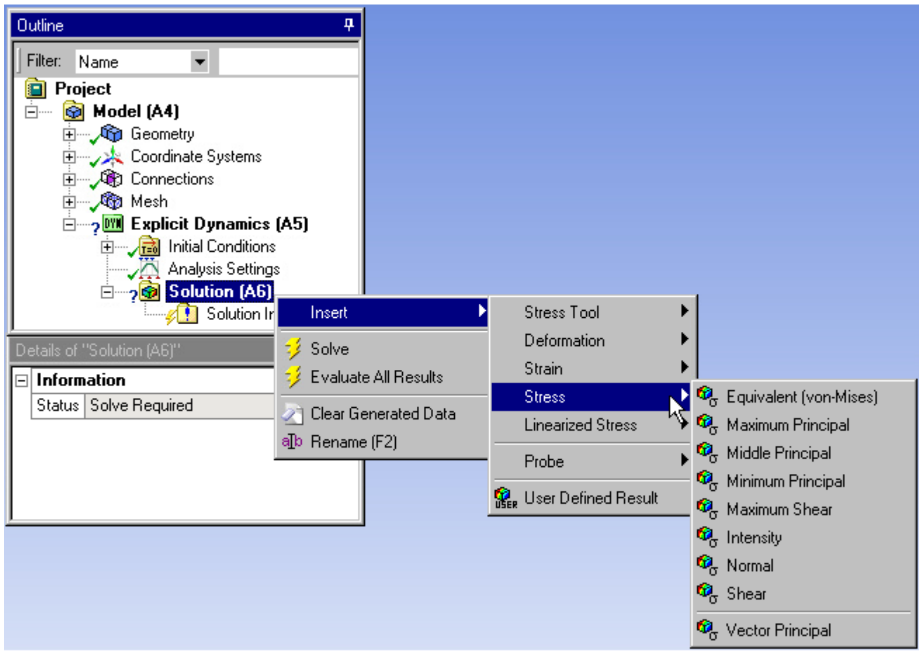


Figure 5.28: Postprocessing.

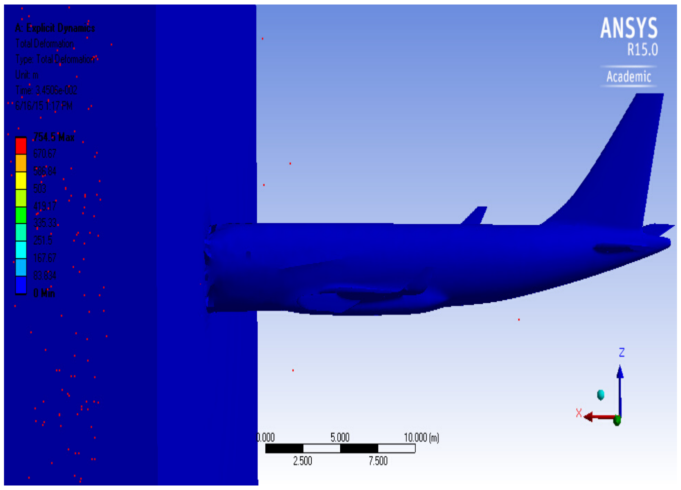
different engineering problems in the automobile, aerospace, construction, military, manufacturing, and bioengineering industries.

We use the same geometry, material models, and aircraft speed as previous section. All models need to be set up in an input file (.k). The formats of settings of time step, initial speed, and parameters of material models in .k file are shown in Fig. 5.38, Fig. 5.39, and Fig. 5.40, respectively.

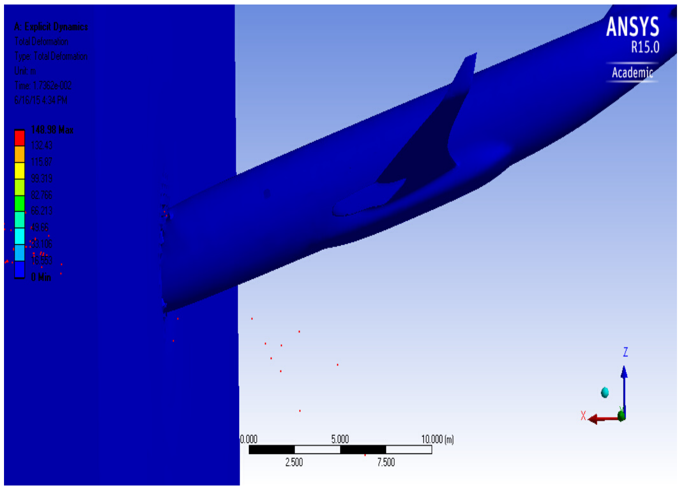
The computation and simulation was performed on Texas A&M Supercomputing Facility's Ada which is an an IBM NeXtScale Cluster 64-bit 10-core Linux with IvyBridge processors. It takes 24 CPU hours to calculate 0.19 second with 20 CPUs for this case.

We consider one case which is the angle of approach $\beta = 0°$. The animation

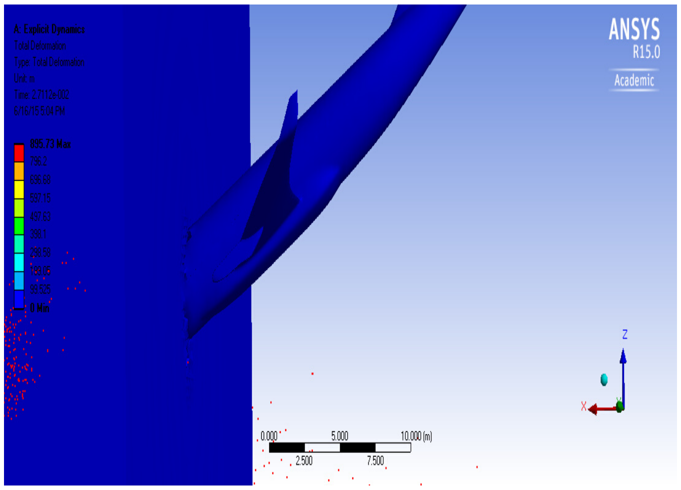
Figure 5.29: A snapshot of comparison of the total deformation presentation for the aircraft crashing into a mountain with the angle of approach 0°, 15°, and 30°.



(a) The angle of approach 0°.

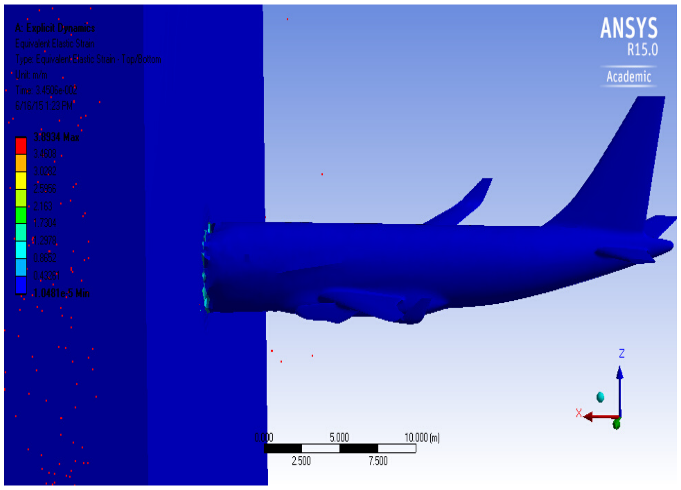


(b) The angle of approach 15°.

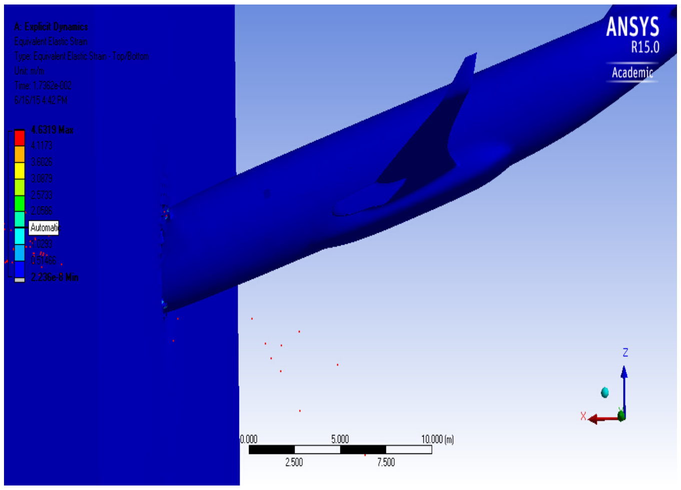


(c) The angle of approach 30°.

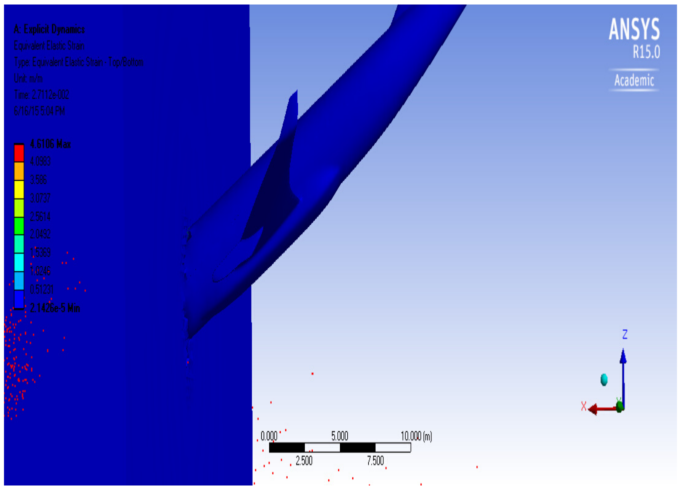
Figure 5.30: A snapshot of comparison of the equivalent strain presentation for the aircraft crashing into a mountain with the angle of approach 0°, 15°, and 30°.



(a) The angle of approach 0°.

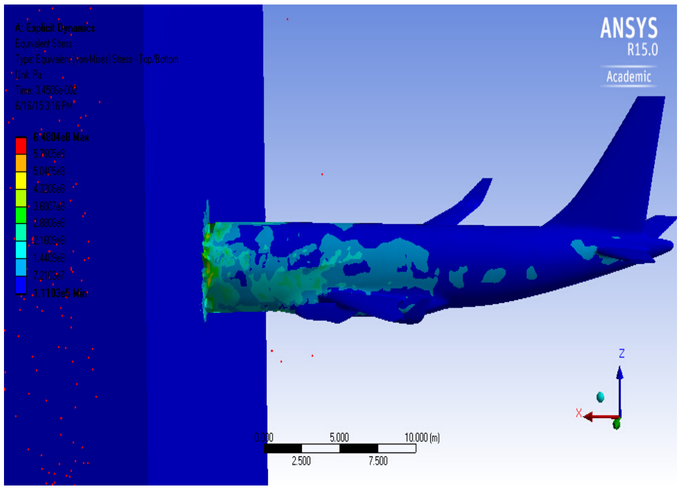


(b) The angle of approach 15°.

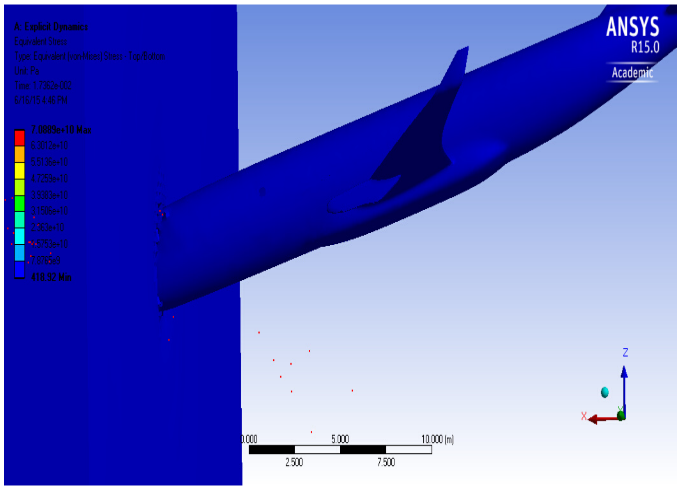


(c) The angle of approach 30°.

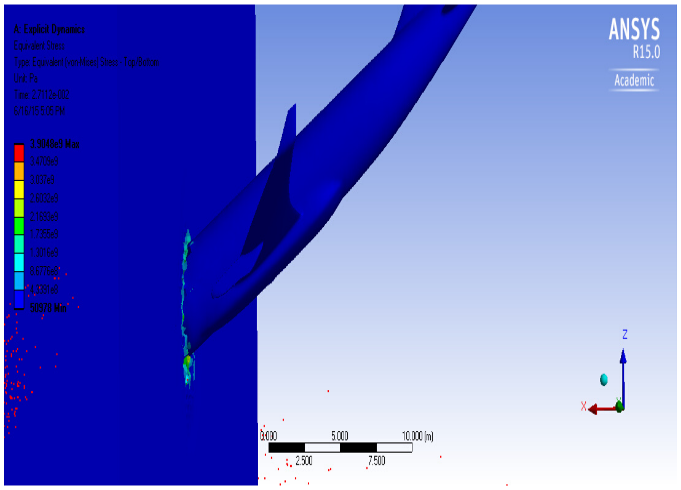
Figure 5.31: A snapshot of comparison of the equivalent stress presentation for the aircraft crashing into a mountain with the angle of approach 0°, 15°, and 30°.



(a) The angle of approach 0°.

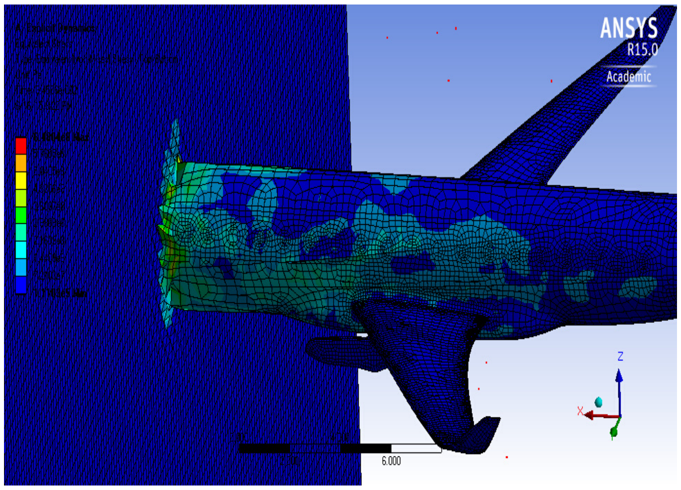


(b) The angle of approach 15°.

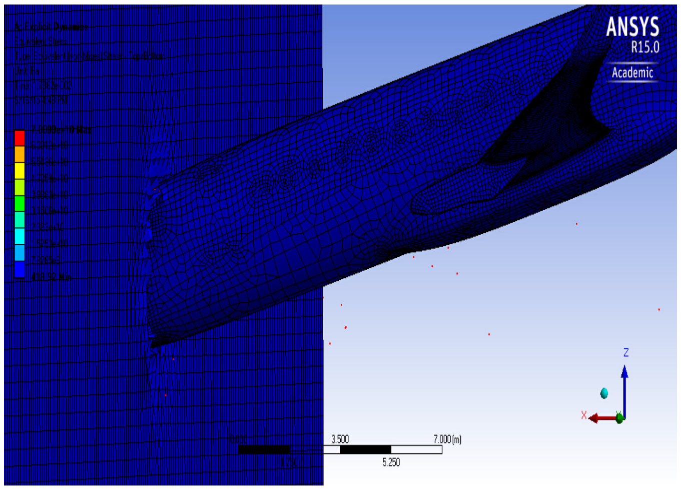


(c) The angle of approach 30°.

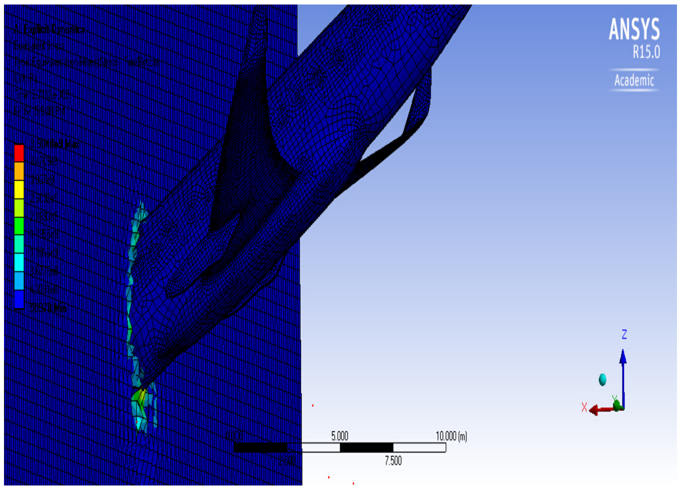
Figure 5.32: A snapshot of comparison of the mesh layout for the aircraft crashing into a mountain with the angle of approach 0°, 15°, and 30°.



(a) The angle of approach 0°.

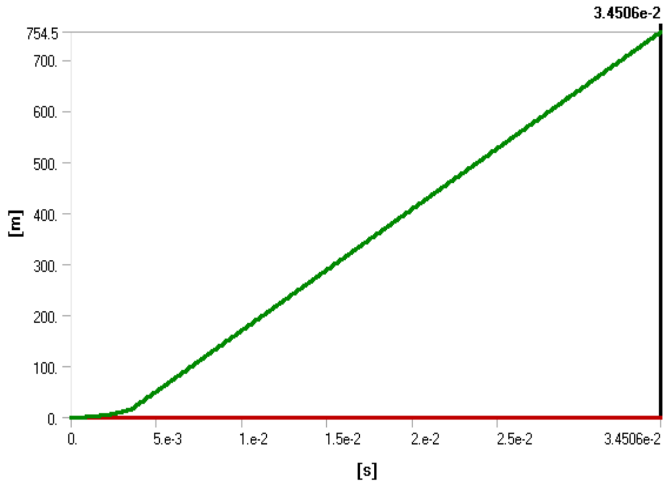


(b) The angle of approach 15°.

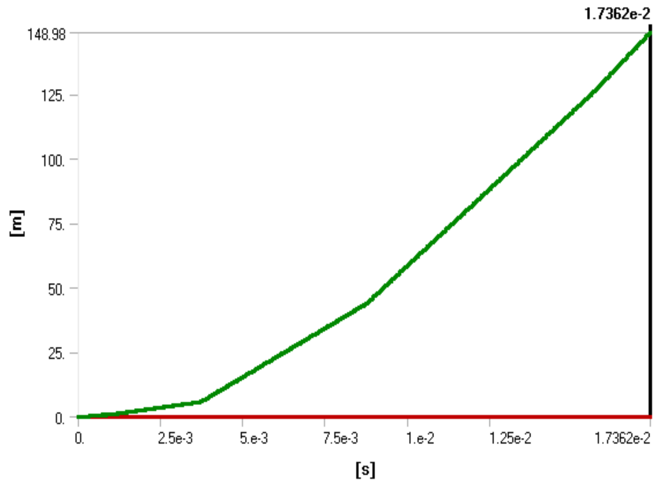


(c) The angle of approach 30°.

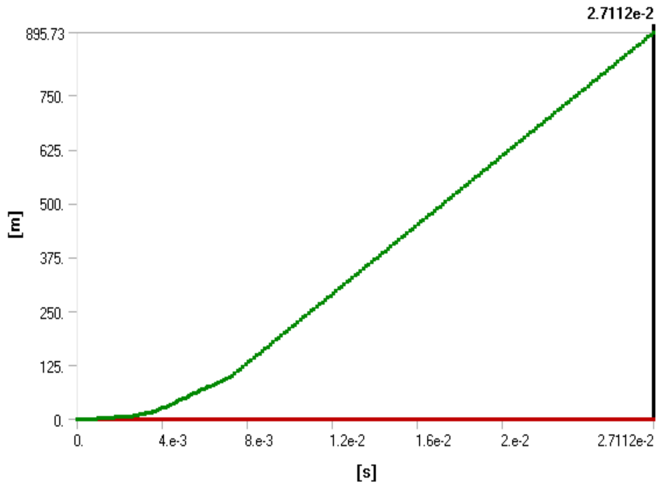
Figure 5.33: The comparison of the total deformation curves for these three cases.



(a) The angle of approach 0°.

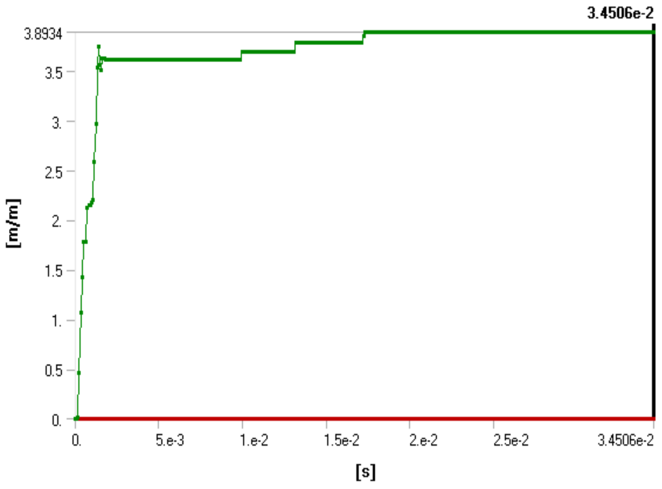


(b) The angle of approach 15°.

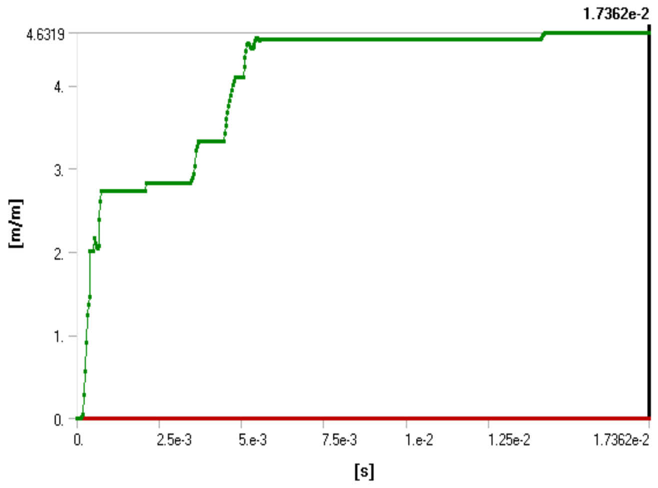


(c) The angle of approach 30°.

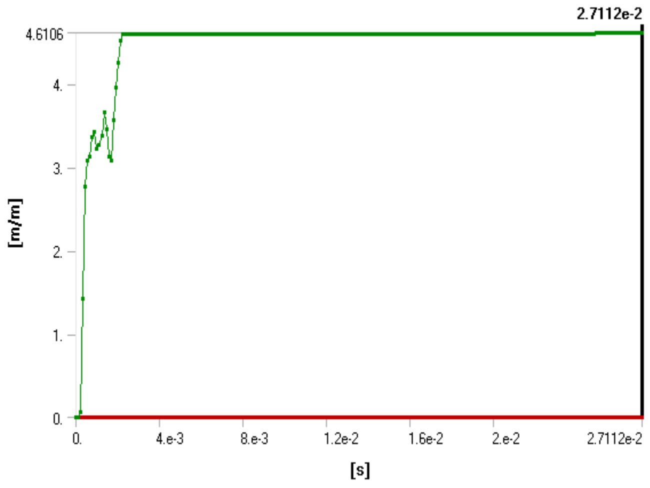
Figure 5.34: The comparison of the equivalent elastic strain curves for these three cases.



(a) The angle of approach 0°.

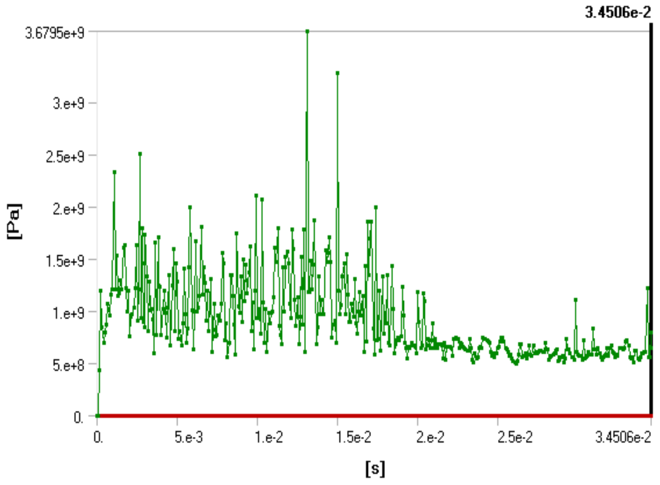


(b) The angle of approach 15°.

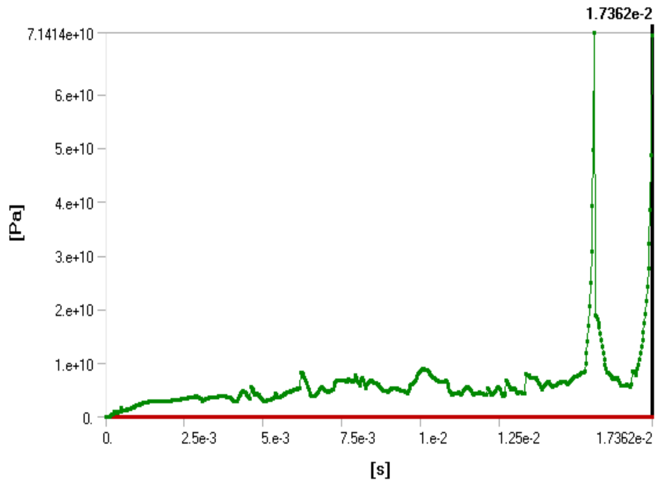


(c) The angle of approach 30°.

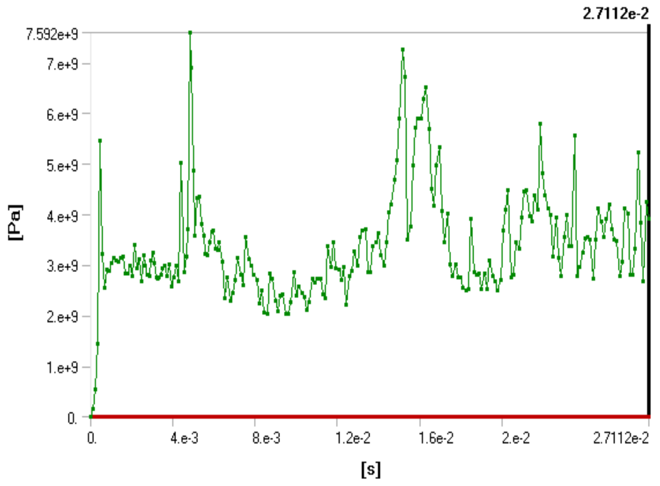
Figure 5.35: The comparison of the equivalent stress curves for these three cases.



(a) The angle of approach 0°.

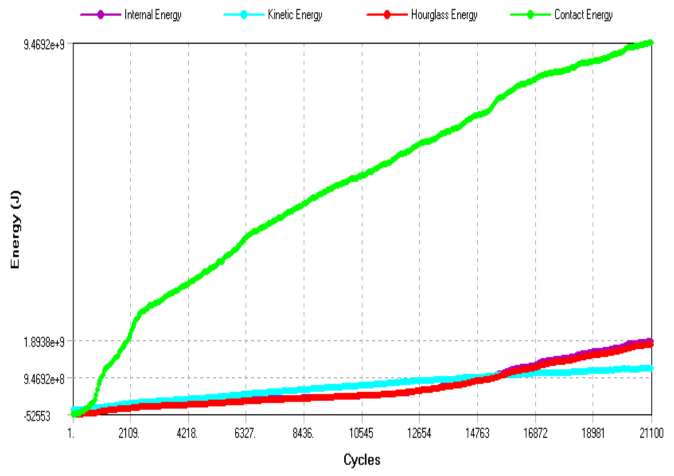


(b) The angle of approach 15°.

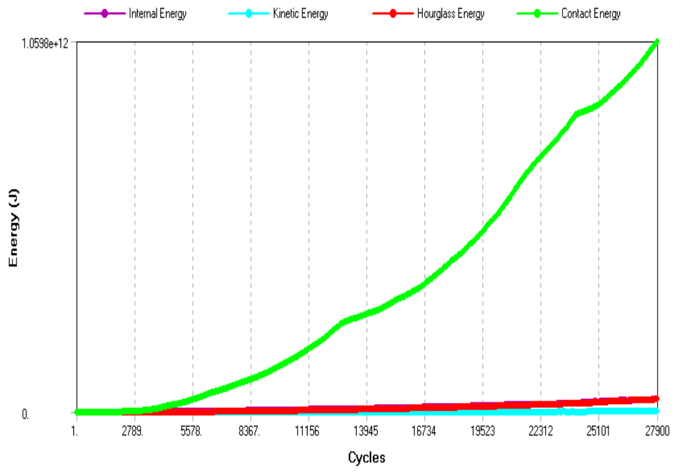


(c) The angle of approach 30°.

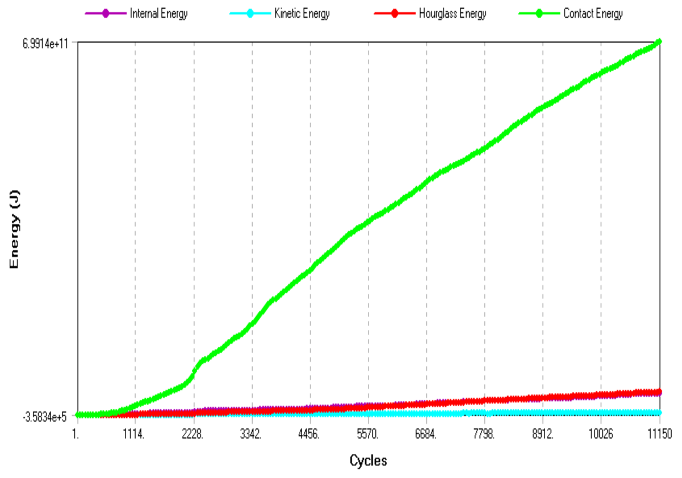
Figure 5.36: The comparison of energy for these three cases.



(a) The angle of approach 0°.

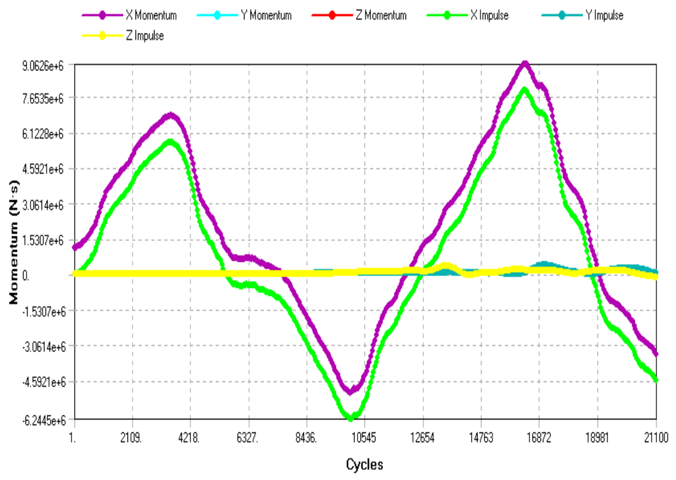


(b) The angle of approach 15°.

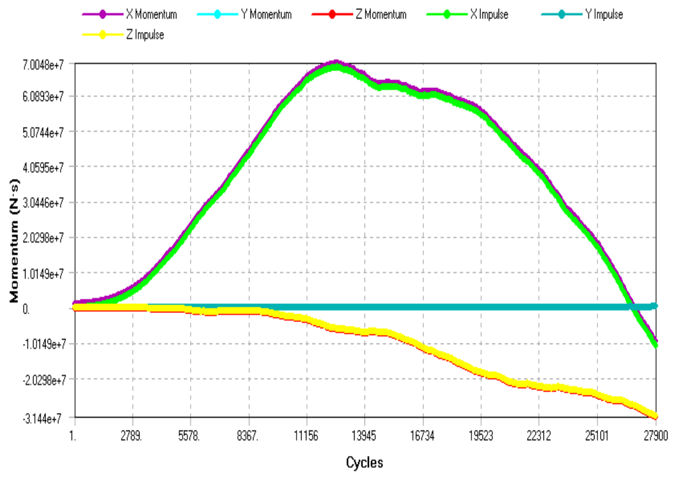


(c) The angle of approach 30°.

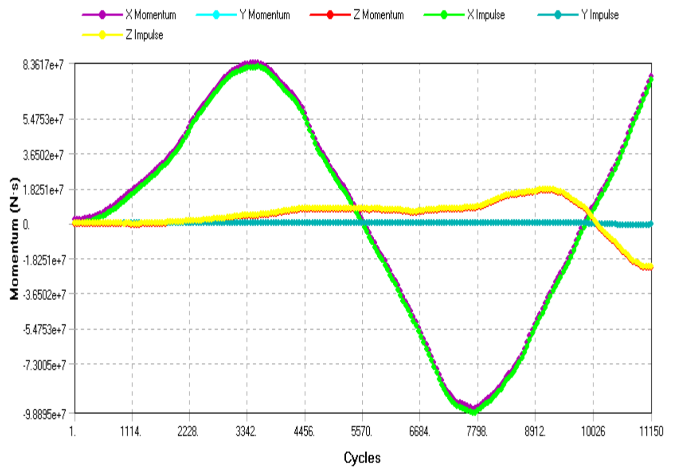
Figure 5.37: The comparison of momentum for these three cases.



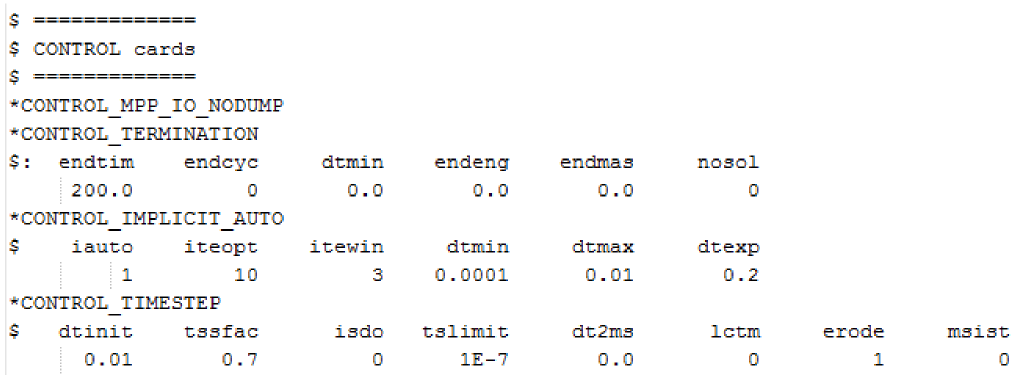
(a) The angle of approach 0°.



(b) The angle of approach 15°.

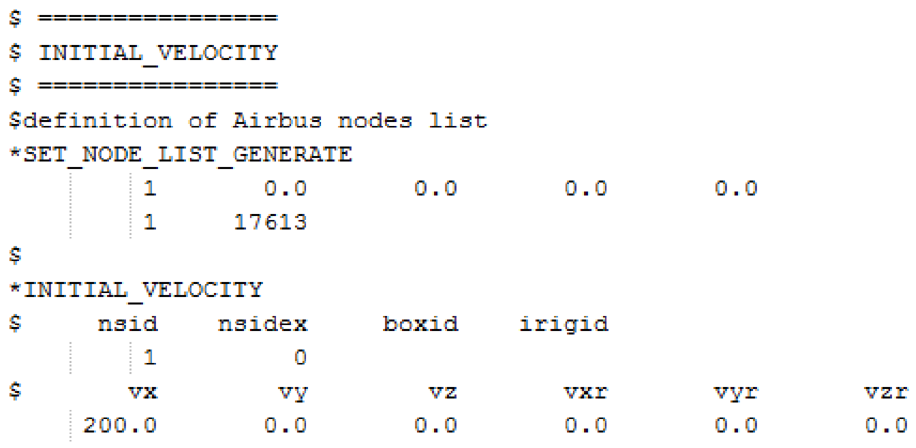


(c) The angle of approach 30°.



```
$ =============
$ CONTROL cards
$ =============
*CONTROL_MPP_IO_NODUMP
*CONTROL_TERMINATION
$:  endtim    endcyc     dtmin    endeng    endmas     nosol
     200.0         0       0.0       0.0       0.0         0
*CONTROL_IMPLICIT_AUTO
$    iauto    iteopt    itewin     dtmin     dtmax     dtexp
         1        10         3    0.0001      0.01       0.2
*CONTROL_TIMESTEP
$   dtinit    tssfac      isdo   tslimit     dt2ms      lctm     erode     msist
      0.01       0.7         0      1E-7       0.0         0         1         0
```

Figure 5.38: Settings of time step in .k file of aircraft crash problem.



```
$ ================
$ INITIAL_VELOCITY
$ ================
$definition of Airbus nodes list
*SET_NODE_LIST_GENERATE
         1       0.0       0.0       0.0       0.0
         1     17613
$
*INITIAL_VELOCITY
$     nsid    nsidex     boxid    irigid
         1         0
$       vx        vy        vz       vxr       vyr       vzr
     200.0       0.0       0.0       0.0       0.0       0.0
```

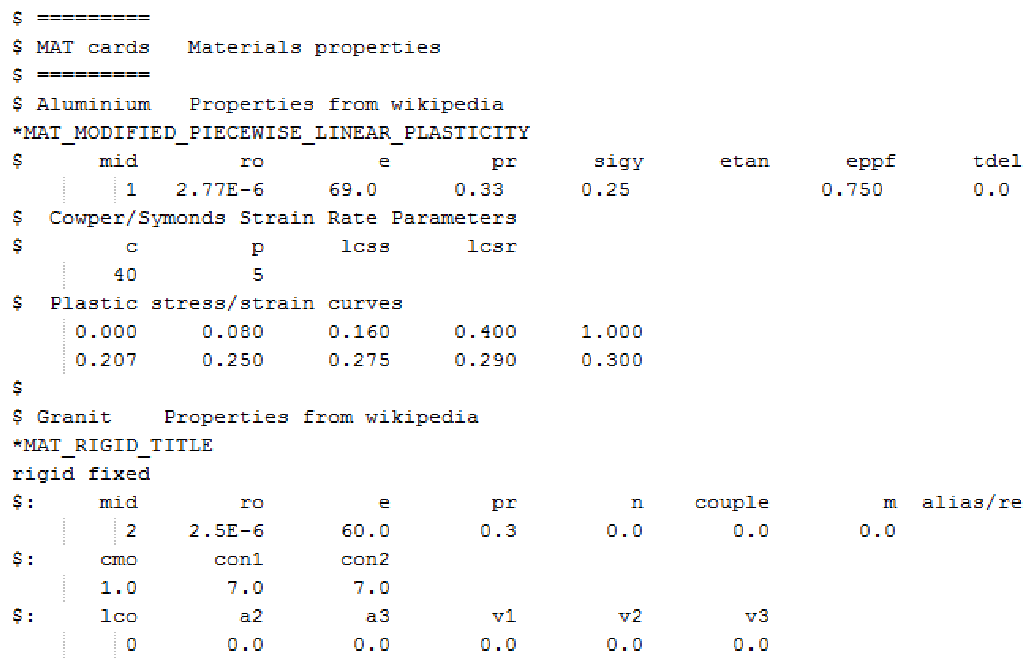
Figure 5.39: Settings of initial speed of aircraft in .k file of aircraft crash problem.

videos from different angles can be viewed in the following links:

https://www.dropbox.com/s/oh11l097e21ocft/movieAB00_000.avi?dl=0

https://www.dropbox.com/s/gq6rzyie9xjwrci/movieAB00_003.avi?dl=0

In the video, we didn't see wall damages. We need to add and adjust parameters of concrete material models to make our case close to the realistic incident.



```
$ =========
$ MAT cards   Materials properties
$ =========
$ Aluminium   Properties from wikipedia
*MAT_MODIFIED_PIECEWISE_LINEAR_PLASTICITY
$      mid        ro         e        pr      sigy      etan      eppf      tdel
         1   2.77E-6      69.0      0.33      0.25               0.750       0.0
$  Cowper/Symonds Strain Rate Parameters
$        c         p      lcss      lcsr
        40         5
$  Plastic stress/strain curves
     0.000     0.080     0.160     0.400     1.000
     0.207     0.250     0.275     0.290     0.300
$
$ Granit    Properties from wikipedia
*MAT_RIGID_TITLE
rigid fixed
$:     mid        ro         e        pr         n    couple         m  alias/re
         2    2.5E-6      60.0       0.3       0.0       0.0       0.0
$:     cmo      con1      con2
       1.0       7.0       7.0
$:     lco        a2        a3        v1        v2        v3
         0       0.0       0.0       0.0       0.0       0.0
```

Figure 5.40: Settings of parameters of material models for both aircraft and wall in .k file of aircraft crash problem.

6. CONCLUSIONS AND FUTURE RESEARCH *

In this dissertation, we use ANSYS FLUENT and OpenFOAM to simulate two problems:

- Compare the simulation capability of three turbulence modelings, DNS, LES, and RANS by using a lid-driven flow in both two- (2D) and three-dimension (3D)
- Compare the numerical results of the airflow interaction of one wind turbine with both fixed angular velocity and wind-driven angular velocity, and two wind turbines in serial alignment

We plan to simulate wind-driven turbine cases and interaction of multiple wind-driven turbines such as wind farm, and also include power generation system in wind turbines.

We use ANSYS Explicit Dynamics and LS-DYNA to simulate Airbus A320 crash problem. For our cases, we still need to adjust material models to get more reasonable damage of aircraft and wall. Our computer simulation has not included wind speed and ocean surface conditions. We also need to add more structures of aircraft such as rings, seats, engine, and fuel tank. The explosion caused by fuel tank during the crash needs to be considered as well. We would also like to include cases with different pitch angles and velocities. Our simulations still need to be improved to be close to the real case.

*Reprinted with permission from "OpenFOAM for Computational Fluid Dynamics" by Goong Chen and Qingang Xiong and Philip J Morris and Eric G Paterson and Alexey Sergeev and Yi-Ching Wang, 2014. Notices of the AMS, 61(4): 354-363, 2014, Copyright [2014] by American Mathematical Society, and Reprinted with permission from "Malaysia Airlines Flight MH370: Water Entry of an Airline" by Goong Chen, Cong Gu, Philip J Morris, Eric G Paterson, Alexey Sergeev, Yi- Ching Wang, and Tomasz Wierzbicki, 2015. Notices of the AMS, 62(4):330-344, 2015. Copyright [2015] by American Mathematical Society.

The CFD approach is advantageous in saving long and expensive processes of laboratory setup and measurements. Now, with the availability of more abundant free and open-source computational tools and user-friendly software, it has become much easier for mathematicians to conduct interdisciplinary collaboration with engineers and physicists for the modeling and computation of complex, “real world” problems, just as this dissertation has hoped to demonstrate.

REFERENCES

[1] OpenFOAM Wiki. `http://openfoamwiki.net/index.php/Main_Page`. Accessed: 2015-07-09.

[2] Serge Abrate. Hull Slamming. *Applied Mechanics Reviews*, 64(6):060803, 2011.

[3] Inc. ASM Aerospace Specification Metals. Properies of Aluminum 2024 T351. `http://asm.matweb.com/search/SpecificMaterial.asp?bassnum=MA2024T4`. Accessed: 2015-07-09.

[4] Goong Chen, Cong Gu, Phillip J Morris, Eric G Paterson, Alexey Sergeev, Yi-Ching Wang, and Tomasz Wierzbicki. Malaysia Airlines Flight MH370: Water Entry of an Airliner. *Notices of the American Mathematical Society*, 62(4):330–344, 2015.

[5] Goong Chen, Qingang Xiong, Philip J Morris, Eric G Paterson, Alexey Sergeev, and Yi-Ching Wang. OpenFOAM for Computational Fluid Dynamics. *Notices of the American Mathematical Society*, 61(4):354–363, 2014.

[6] CFD Direct. Lid-driven cavity flow (in OpenFOAM). `http://openfoam.org/docs/user/cavity.php`. Accessed: 2015-07-09.

[7] Andrew G Fabula. Ellipse-Fitting Approximation of Two-Dimensional, Normal Symmetric Impact of Rigid Bodies on Water. *Proceedings of Fifth Midwestern Conference on Fluid Mechanics*, pages 299–315, 1957.

[8] Edwin L Fasanella, E Widmayer, and Martha P Robinson. Structural Analysis of the Controlled Impact Demonstration of a Jet Transport Airplane. *Journal of Aircraft*, 24(4):274–280, 1987.

[9] PR Garabedian. Oblique Water Entry of a Wedge. *Communications on Pure and Applied Mathematics*, 6(2):157–165, 1953.

[10] Belgium Global Wind Energy Council, Brussels and The Netherlands. Greenpeace, Amsterdam. The Global Wind Energy Council. `http://www.gwec.net/`. Accessed: 2015-07-09.

[11] ANSYS Inc. FLUENT. `http://www.ansys.com/Products/Simulation+Technology/Fluid+Dynamics/Fluid+Dynamics+Products/ANSYS+Fluent`. Accessed: 2015-07-09.

[12] CEI Inc. Ensight. `http://ensight.com/`. Accessed: 2015-07-09.

[13] Raad I Issa. Solution of the Implicitly Discretised Fluid Flow Equations by Operator-Splitting. *Journal of Computational Physics*, 62(1):40–65, 1986.

[14] Steve Lindenberg. *20% Wind Energy By 2030: Increasing Wind Energy's Contribution to US Electricity Supply.* DIANE Publishing, 2009.

[15] AM Mackey. A Mathematical Model of Water Entry. *Admiralty Underwater Weapons Establishment TN*, (636/79), 1979.

[16] John R McGehee, Melvin E Hathaway, and Victor L Vaughan. *Water-Landing Characteristics of a Reentry Capsule.* National Aeronautics and Space Administration, 1959.

[17] CFD Online. Gambit meshing software. `http://www.cfd-misc.com/Forums/ansys-meshing/114276-gambit-ansys-meshing-software.html`. Accessed: 2015-07-09.

[18] CFD Online. Lid-driven cavity problem. `http://www.cfd-online.com/Wiki/Lid-driven_cavity_problem`. Accessed: 2015-07-09.

[19] CFD Online. OpenFOAM discussion forum. `http://www.cfd-misc.com/Forums/openfoam/`. Accessed: 2015-07-09.

[20] OpenCFD Ltd. OpenFOAM. `http://openfoam.org/`. Accessed: 2015-07-09.

[21] OpenCFD Ltd. OpenFOAM 2.1.0 user guide. `https://www.dropbox.com/s/v6buyn8t9e7kixr/UserGuide.pdf`. Accessed: 2015-07-09.

[22] OpenCFD Ltd. OpenFOAM Workshop. `http://www.openfoamworkshop.org/`. Accessed: 2015-07-09.

[23] OpenFOAMWiki. Graphical user interfaces for OpenFOAM. `http://openfoamwiki.net/index.php/GUI`. Accessed: 2015-07-09.

[24] Stephen B Pope. *Turbulence Flows*. Cambridge University Press Cambridge, UK, 2000.

[25] Associated Press. 4 possible ways Malaysia Flight 370 hit the water and how each would affect the search. `http://www.syracuse.com/news/index.ssf/2014/04/4_possible_ways_malaysia_fligh.html`. April 8, 2014. Accessed: 2015-07-09.

[26] NK Sanjeev, Vinayak Malik, and H Suresh Hebbar. Verification of Johnson-Cook Material Model Constants of AA2024-T3 for Use in Finite Element Simulation of Friction Stir Welding and its Utilization in Sever Plastic Deformation Process Modeling. *International Journal of Research in Engineering and Technology*, 3(6):98–102, 2014.

[27] M Shiffman and DC Spencer. The Force of Impact on a Cone Striking a Water Surface (Vertical Entry). *Communications on Pure and Applied Mathematics*, 4(4):379–417, 1951.

[28] Henk Kaarle Versteeg and Weeratunge Malalasekera. *An Introduction to Computational Fluid Dynamics: the Finite Volume Method*. Pearson Education, 2007.

[29] Theodore von Kármán. The Impact on Seaplane Floats during Landing. *NACA TN*, (321), 1929.

[30] Herbert Wagner. Landing of Seaplanes. *NACA TM*, (622), 1931.

[31] T. Wierzbicki and D. K. Yue. Spacecraft Crashworthiness - Towards Reconstruction of the Challenger Accident. *American Society of Mechanical Engineers, Applied Mechanics Division, AMD*, 79:31–46, 1986.

[32] Tomasz Wierzbicki and Dick K Yue. Impact Damage of the Challenger Crew Compartment. *Journal of Spacecraft and Rockets*, 23(6):646–654, 1986.

[33] Wikimedia Foundation Inc. Airbus A320 dimensions and key data. `https://en.wikipedia.org/wiki/Airbus_A320_family`. Accessed: 2015-07-09.

[34] Wikimedia Foundation Inc. Kolmogorov microscales, see. `http://en.wikipedia.org/wiki/Kolmogorov_microscales`. Accessed: 2015-07-09.

[35] Wikimedia Foundation Inc. OpenFOAM DNS. `http://en.wikipedia.org/wiki/Direct_numerical_simulation`. Accessed: 2015-07-09.

[36] Wikimedia Foundation Inc. OpenFOAM LES. `http://en.wikipedia.org/wiki/Large_eddy_simulation`. Accessed: 2015-07-09.

[37] Wikimedia Foundation Inc. OpenFOAM RANS. `http://en.wikipedia.org/wiki/Turbulence_modelling`. Accessed: 2015-07-09.

[38] Wikimedia Foundation Inc. Paraview. http://en.wikipedia.org/wiki/ParaView. Accessed: 2015-07-09.

[39] Wikimedia Foundation Inc. Pressure-Correction method. http://en.wikipedia.org/wiki/Pressure-correction_method. Accessed: 2015-07-09.

[40] Wikimedia Foundation Inc. Redhat. http://en.wikipedia.org/wiki/RedHat. Accessed: 2015-07-09.

[41] Wikimedia Foundation Inc. The Cathedral and the Bazaar. http://en.wikipedia.org/wiki/The_Cathedral_and_the_Bazaar. Accessed: 2015-07-09.

[42] Wikimedia Foundation Inc. U.S. Airways flight 1549. http://en.wikipedia.org/wiki/US_Airways_Flight_1549. Accessed: 2015-07-09.