COMPUTATIONAL MECHANICS FOR AIRCRAFT WATER ENTRY AND WIND ENERGY

A Dissertation

by

CONG GU

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

Chair of Committee,	Goong Chen
Committee Members,	Peter Stiller
	Ronald DeVore
	Siu Chin
Head of Department,	Emil Straube

December 2015

Major Subject: Mathematics

ABSTRACT

In this thesis, two problems in computational mechanics, namely aircraft water entry and wind energy, have been studied together with description of related theory and methodology. Fluid calculations are carried out with proper schemes and computational techniques, including the use of dynamic mesh with OpenFOAM as the platform. Subsequent analysis of the data provides valuable information for these real world problems.

First, algorithms and numerical methods to solve the equations related to the problems are proposed. Model problems are solved to test these methods. Then, in the aircraft water entry problem, the complex and dynamic process of aircraft water entry problem is simulated under several cases. External loading data has been analyzed to estimate the severity of structural damage. The main finding is that the vertical diving case is actually a reasonable theory regarding the final moments of flight MH370 given the currently available information. In the wind energy problem, blade resolved simulations of wind turbines are carried out. The proper orthogonal decomposition analysis is shown to be capable of extracting dominant features of the turbulent flow. Interaction between wind generators are studied to find out that contra-rotating turbines can better capture energy in the wind.

It has been demonstrated that the computational approach is advantageous in saving long and expensive processes of laboratory setup and measurements, while providing valuable information to the subject problem.

DEDICATION

I would like dedicate this thesis to my wife, Yu Yang, who has offered unwavering support and encouragement during my Ph.D study, and my parents, who have provided me with the best education they could from my childhood and continues to support me throughout my life.

ACKNOWLEDGEMENTS

First I wish to thank my adviser, Professor Goong Chen. It has been an honor to be his Ph.D. student. His contributions of time, guidance and support allow me to continue my Ph.D. I also appreciate the financial support of research assistantships, and for travel to conferences and colloquia. I also enjoyed the time doing research in Texas A&M University at Qatar. Also, I want to thank my committee members, Professors Ronald DeVore, Peter Stiller and Siu Chin, for sharing their valuable time.

I want to thank the high performance computing department at Texas A&M University and Texas A&M University at Qatar for allowing me to carry out computations in their facilities, and for their technical support. An acknowlegement is also given to the free software community, particularly OpenFOAM, ParaView, GNU/Linux, octave and LaTeXprojects.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Modeling and computation of fluid motion is a major field in mathematics, science and engineering. There are numerous examples: large scale weather events like hurricanes, fluid interactions with bridges and aircrafts, blood flows, micro scale medical devices. As a first step, mathematical models described by partial differential equations, such as Navier-Stokes equation, are developed. Despite huge efforts being devoted to the mathematical analysis of these equations, fundamental theory for these equations are still not complete. On the other hand, numerical approximations to these systems are needed and must be developed for practical purposes in order to help prediction and decision making. The goal of this thesis is to do modeling and computation for concrete problems using the OpenFOAM[1] framework.

The plan of the dissertation is laid out as follows. In order to provide the ability to carry out such practical computational studies, the basic mathematical model, Navier-Stokes equations, along with its numerical approximations will be given in Section 2. A general projection method framework to numerically solve pressure coupled systems in the Navier-Stokes system will be given. Volume-of-fluid method is added to enable us to simulate two phase flows. And dynamic mesh techniques with moving boundaries will be used to simulate interaction between fluids and immersed solid objects.

The first problem considered is the water entry of an aircraft. Water entry considers the dynamic motion of an object upon its entry into the water. It is a classical problem in applied mathematics and fluid dynamics and is motivated by several applications such as water landing of aircrafts, ship slamming and return of space

[1]OPENFOAM® is a registered trade mark of OpenCFD Limited, the producer of the OpenFOAM software.

capsules. With the help of large scale computation, three dimensional Navier-Stokes equations will be solved numerically with the complexity of two fluid phases, and turbulence modeling. In addition to the scientific interest of understanding the process, it will be shown that practical damage assessment can be carried out using these computation data. This will be discussed in Section 3. Part of this section was first published in [8] by American Mathematical Society.

The second application is wind energy generation. Wind energy is one of the major forms of renewable energy. High fidelity, blade resolving simulation of the wind turbine flows hopefully will provide a more accurate approach for estimating the performance of wind generators. The fully transient numerical simulations also give us the opportunity to study the dynamic interaction between turbine blades and tower, interaction between between generator within a wind farm, and dynamic control strategies of the generators. This will be discussed in Section 4.

A summary will be given in Section 5.

2. NUMERICAL SIMULATION OF FLUID FLOWS

In this section, we offer a review of computational methods for the Navier-Stokes equations and finite volume method. We then introduce the physical and numerical method to be computed.

2.1 Solution Algorithms for Pressure-Coupled Systems

The time dependent Navier-Stokes system is the common model to describe fluids. Let Ω be an open, bounded spatial domain, and $T_0 > 0$ be the fixed terminal time. Consider the incompressible Navier-Stokes system for the velocity $\mathbf{u} : \Omega \times [0, T_0] \to \mathbb{R}^d$ and pressure $p : \Omega \times [0, T_0] \to \mathbb{R}$ by

$$\frac{\partial(\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) - \nabla \cdot \boldsymbol{\tau} + \nabla p = \mathbf{f} \qquad \text{in } \Omega \times [0, T_0], \tag{2.1}$$

$$\nabla \cdot \mathbf{u} = 0 \qquad \text{in } \Omega \times [0, T_0], \tag{2.2}$$

where $\boldsymbol{\tau}$ is the deviatoric stress tensor, ρ is the density and $\mathbf{f}$ is the external body force. For incompressible Newtonian fluid, $\boldsymbol{\tau} = \mu(\nabla \mathbf{u} + \nabla \mathbf{u}^\top)$, where μ is the viscosity.

The system is supplemented by initial condition

$$\mathbf{u}(\cdot, t) = \mathbf{u}_0 \qquad \text{in } \Omega, \tag{2.3}$$

and appropriate boundary conditions. For instance, constant inflow and wall can be modeled by the Dirichlet boundary condition for velocity as

$$\mathbf{u} = \mathbf{u}_b \qquad \text{on } \partial\Omega \times [0, T_0]. \tag{2.4}$$

Note that the divergence-free condition (2.2) (also called continuity condition)

needs to be coupled with the convection-diffusion problem. Various algorithms have been proposed such as the projection method [6], SIMPLE algorithm [34] and PISO algorithm [25]. These methods are basically composed of a prediction step and a subsequent correction step to ensure continuity. The process is then iterated if necessary. To simplify the discussion in the current section, we focus on the linear system

$$\frac{\partial(\rho\mathbf{u})}{\partial t} - \mu\nabla^2\mathbf{u} + \nabla p = \mathbf{f} \qquad \text{in } \Omega \times [0, T_0], \tag{2.5}$$

$$\nabla \cdot \mathbf{u} = 0 \qquad \text{in } \Omega \times [0, T_0], \tag{2.6}$$

which is called the Stokes problem. When the full Navier-Stokes system (2.1)-(2.2) is considered, the nonlinearity can be treated with iteration over the linearized system. Given a temporal discretization, for example, the backward Euler

$$\rho\frac{\mathbf{u}^n - \mathbf{u}^{n-1}}{\delta t} - \mu\nabla^2\mathbf{u}^n + \nabla p^n = \mathbf{f}^n, \tag{2.7}$$

where $\delta t = t^n - t^{n-1}$, the system can be expressed formally as

$$\begin{pmatrix} \mathbf{A} & \frac{\delta t}{\rho}\mathbf{G} \\ \mathbf{D} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u}^n \\ p^n \end{pmatrix} = \begin{pmatrix} \mathbf{r} \\ 0 \end{pmatrix}, \tag{2.8}$$

where

$$\mathbf{A} = \mathbf{I} - \frac{\delta t\mu}{\rho}\mathbf{L}, \tag{2.9}$$

$$\mathbf{r} = \mathbf{u}^{n-1} + \frac{\delta t}{\rho}\mathbf{f}^n. \tag{2.10}$$

Here $\mathbf{L}$, $\mathbf{G}$ and $\mathbf{D}$ are the Laplacian, gradient and divergence operators, respectively. They can be either regarded as differential operators with appropriate boundary con-

ditions or algebraic matrices after spatial discretization of the equations. The above equation can be split and factorized into various forms. This idea is explored as the inexact factorization formulation [35]. In general, consider the following decomposition

$$\begin{pmatrix} \mathbf{A} & \frac{\delta t}{\rho}\mathbf{G} \\ \mathbf{D} & 0 \end{pmatrix} = \mathcal{A}\mathcal{C} + \mathcal{R}, \tag{2.11}$$

with the definition of the block operators

$$\begin{aligned} \mathcal{A} &= \begin{pmatrix} \mathbf{A}_0 & 0 \\ \mathbf{D} & -\frac{\delta t}{\rho}\mathbf{D}\mathbf{P}^{-1}\mathbf{G} \end{pmatrix}, \\ \mathcal{C} &= \begin{pmatrix} \mathbf{I} & \frac{\delta t}{\rho}\mathbf{P}^{-1}\mathbf{G} \\ 0 & \mathbf{I} \end{pmatrix}, \\ \mathcal{R} &= \begin{pmatrix} \mathbf{A} - \mathbf{A}_0 & \frac{\delta t}{\rho}(\mathbf{I} - \mathbf{A}_0\mathbf{P}^{-1})\mathbf{G} \\ 0 & 0 \end{pmatrix}. \end{aligned} \tag{2.12}$$

Here $\mathbf{A}_0$ and $\mathbf{P}$ are introduced to approximate $\mathbf{A}$, in order to reduce or eliminate the number of direct inversions of operator $\mathbf{A}$, at the cost of a residual term $\mathcal{R}$ to the whole system. An iterative procedure for the system (2.8), rewritten as

$$\mathcal{A}\mathcal{C}\boldsymbol{\psi} + \mathcal{R}\boldsymbol{\psi} = \mathbf{b}, \tag{2.13}$$

with

$$\boldsymbol{\psi} = \begin{pmatrix} \mathbf{u}^n \\ p^n \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} \mathbf{r} \\ 0 \end{pmatrix},$$

can then be constructed by

$$\mathcal{A}\mathcal{C}\boldsymbol{\psi}_l = \mathbf{b} - \mathcal{R}\boldsymbol{\psi}_{l-1}, \quad l = 1, 2, \ldots, m \tag{2.14}$$

for m sub-step iterations. After the final step, the approximate solution is obtained as $\boldsymbol{\psi}_m$.

Equations (2.12) and (2.14) are a general representation of a large category of solution algorithms for the pressure coupled systems (2.8). They can differ by the choice of $\mathbf{A}_0$, $\mathbf{P}$, m, $\boldsymbol{\psi}_0$. Also note that temporal discretizations can be easily adjusted, namely by $\mathbf{A}$ and $\mathbf{r}$. Several specific cases of the proposed general method will be discussed below.

Example 2.1 (Original projection method [10]). Let

$$m = 1, \quad \mathbf{A}_0 = \mathbf{A}, \quad \mathbf{P} = \mathbf{I}, \quad \boldsymbol{\psi}_0 = \mathbf{0}.$$

Equation (2.14) become

$$\begin{pmatrix} \mathbf{A} & 0 \\ \mathbf{D} & -\frac{\delta t}{\rho}\mathbf{DG} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \frac{\delta t}{\rho}\mathbf{G} \\ 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{u}^n \\ p^n \end{pmatrix} = \begin{pmatrix} \mathbf{r} \\ 0 \end{pmatrix}, \tag{2.15}$$

which is essentially the following series of operations

$$\mathbf{A}\tilde{\mathbf{u}} = \mathbf{r}, \tag{2.16}$$

$$\frac{\delta t}{\rho}\mathbf{DG}p^n = \mathbf{D}\tilde{\mathbf{u}}, \tag{2.17}$$

$$\mathbf{u}^n = \tilde{\mathbf{u}} - \frac{\delta t}{\rho}\mathbf{G}p^n. \tag{2.18}$$

Equation (2.16) is called the *prediction* step, it deals everything except the pressure gradient. Then the pressure Poisson equation (2.17) and a *correction* step (2.18) make sure that the continuity condition (2.2) is satisfied. By subtracting the original

equation (2.13) it can be seen that the splitting error involved in this method is

$$\begin{aligned} E =& \mathcal{R}[\boldsymbol{\psi}_0 - \boldsymbol{\psi}] \\ =& -\frac{\delta t}{\rho}(\mathbf{I} - \mathbf{A})\mathbf{G}p^n = -\frac{(\delta t)^2 \mu}{\rho^2}\mathbf{G}p^n = \mathcal{O}(\delta t^2). \end{aligned}$$

This means the method has order one. Therefore, any temporal discretization higher than order one will likely have no benefit over the backward Euler method. Rigorous error analysis for the Navier-Stokes system solved with this scheme can be found in [38].

Example 2.2 (Incremental correction method [19]). Let

$$m = 1, \quad \mathbf{A}_0 = \mathbf{A}, \quad \mathbf{P} = \mathbf{I}, \quad \boldsymbol{\psi}_0 = \begin{pmatrix} 0 \\ p^{n-1} \end{pmatrix}.$$

The difference with the original projection method is that (2.16) is replaced by

$$\mathbf{A}\tilde{\mathbf{u}} = \mathbf{r} - \frac{\delta t}{\rho}(\mathbf{I} - \mathbf{A})\mathbf{G}p^{n-1}. \tag{2.19}$$

By making an substitution

$$\tilde{\tilde{\mathbf{u}}} = \tilde{\mathbf{u}} - \frac{\delta t}{\rho}\mathbf{G}p^{n-1}, \tag{2.20}$$

we arrive at the following series of operations:

$$\mathbf{A}\tilde{\tilde{\mathbf{u}}} = \mathbf{r} - \frac{\delta t}{\rho}\mathbf{G}p^{n-1}, \tag{2.21}$$

$$\frac{\delta t}{\rho}\mathbf{D}\mathbf{G}(p^n - p^{n-1}) = \mathbf{D}\tilde{\tilde{\mathbf{u}}}, \tag{2.22}$$

$$\mathbf{u}^n = \tilde{\tilde{\mathbf{u}}} - \frac{\delta t}{\rho}\mathbf{G}(p^n - p^{n-1}). \tag{2.23}$$

This method uses incremental corrections, which means that the Poission equation is for the pressure increment rather than pressure itself. The error involved is

$$\begin{aligned} E =& \mathcal{R}[\boldsymbol{\psi}_0 - \boldsymbol{\psi}] \\ =& -\frac{\delta t}{\rho}(\mathbf{I} - \mathbf{A})\mathbf{G}(p^n - p^{n-1}) = -\frac{(\delta t)^2\mu}{\rho^2}\mathbf{G}(p^n - p^{n-1}) = -\frac{(\delta t)^3\mu}{\rho^2}\mathbf{G}\frac{p^n - p^{n-1}}{\delta t} \\ =& \mathcal{O}(\delta t^3). \end{aligned}$$

Therefore, this method is formally second order.

Example 2.3 (SIMPLE method [34]). *Semi-Implicit Method for Pressure-Linked Equations* (SIMPLE) type methods deals with the algebraic formulation of the operator $\mathbf{A}$. The diagonal elements of $\mathbf{A}$ therefore can be selected by $\mathbf{P} = \mathrm{diag}(\mathbf{A})$ for quick inversion. More specifically, at a given control volume $K \in \mathcal{T}$, suppose that the discretized Laplacian is

$$(-\mathbf{L}\mathbf{u})_K = a_{KK}\mathbf{u}_K + \sum_{L \in \mathcal{T}\backslash K} a_{KL}\mathbf{u}_L,$$

then by (2.9),

$$(\mathbf{A}\mathbf{u})_K = \mathbf{u}_K + \frac{\delta t\mu}{\rho}\left(a_{KK}\mathbf{u}_K + \sum_{L \in \mathcal{T}\backslash K} a_{KL}\mathbf{u}_L\right),$$

and hence the diagonal of $\mathbf{A}$ can be explicitly written as

$$(\mathrm{diag}(\mathbf{A})\mathbf{u})_K = \left(1 + \frac{\delta t\mu}{\rho}a_{KK}\right)\mathbf{u}_k.$$

SIMPLE-type methods can be used with incremental or non-incremental correction similar to the above two examples. For simplicity, let us consider the non-

incremental version

$$m = 1, \quad \mathbf{A}_0 = \mathbf{A}, \quad \mathbf{P} = \mathrm{diag}(\mathbf{A}), \quad \boldsymbol{\psi}_0 = \mathbf{0}.$$

The following series of operations are obtained

$$\mathbf{A}\tilde{\mathbf{u}} = \mathbf{r}, \tag{2.24}$$

$$\frac{\delta t}{\rho}\mathbf{D}\mathbf{P}^{-1}\mathbf{G}p^n = \mathbf{D}\tilde{\mathbf{u}}, \tag{2.25}$$

$$\mathbf{u}^n = \tilde{\mathbf{u}} - \frac{\delta t}{\rho}\mathbf{P}^{-1}\mathbf{G}p^n. \tag{2.26}$$

It may be viewed as an original *projection method* with Jacobi preconditioning for the pressure correction. It have the same formal order, namely, first order for non-incremental correction and second order for incremental correction when we take

$$\boldsymbol{\psi}_0 = \begin{pmatrix} 0 \\ p^{n-1} \end{pmatrix}$$

Example 2.4 (PISO method [25]). *Pressure implicit with splitting of operator (PISO)* algorithm is similar to SIMPLE in that it also deals with algebraic formulation. However, it has at least two corrections, namely $m \geq 2$. Let

$$m = 2, \quad \mathbf{A}_0 = \mathbf{P} = \mathrm{diag}(\mathbf{A}), \quad \boldsymbol{\psi}_0 = \begin{pmatrix} \mathbf{u}^{n-1} \\ 0 \end{pmatrix}.$$

The following series of operations are obtained for the first iteration:

$$\mathbf{A}_0\tilde{\mathbf{u}}^{(1)} = \mathbf{r} - (\mathbf{A} - \mathbf{A_0})\mathbf{u}^{(1)}, \tag{2.27}$$

$$\frac{\delta t}{\rho}\mathbf{D}\mathbf{P}^{-1}\mathbf{G}p^{n,1} = \mathbf{D}\tilde{\mathbf{u}}^{(1)}, \tag{2.28}$$

$$\mathbf{u}^{(1)} = \tilde{\mathbf{u}}^{(1)} - \frac{\delta t}{\rho}\mathbf{P}^{-1}\mathbf{G}p^{(1)}, \tag{2.29}$$

and for the second iteration

$$\mathbf{A}_0\tilde{\mathbf{u}}^{(2)} = \mathbf{r} - (\mathbf{A} - \mathbf{A_0})\mathbf{u}^{(1)}, \tag{2.30}$$

$$\frac{\delta t}{\rho}\mathbf{D}\mathbf{P}^{-1}\mathbf{G}p^{n} = \mathbf{D}\tilde{\mathbf{u}}^{(2)}, \tag{2.31}$$

$$\mathbf{u}^{n} = \tilde{\mathbf{u}}^{(2)} - \frac{\delta t}{\rho}\mathbf{P}^{-1}\mathbf{G}p^{n}. \tag{2.32}$$

By comparing to the exact equation (2.13), the splitting error involved in the algorithm can be expressed by

$$\begin{aligned}
E =& \mathcal{R}[\boldsymbol{\psi}_1 - \boldsymbol{\psi}] \\
=& \mathcal{R}[(\mathcal{A}\mathcal{C})^{-1}(\mathbf{b} - \mathcal{R}\boldsymbol{\psi}_0) - \boldsymbol{\psi}] \\
=& \mathcal{R}[(\mathcal{A}\mathcal{C})^{-1}(\mathbf{b} - \mathcal{A}\mathcal{C}\boldsymbol{\psi} - \mathcal{R}\boldsymbol{\psi}_0)] \\
=& \mathcal{R}[(\mathcal{A}\mathcal{C})^{-1}\mathcal{R}(\boldsymbol{\psi} - \boldsymbol{\psi}_0)] \\
=& (\mathbf{A} - \mathbf{A}_0)\mathbf{A}_0^{-1}(\mathbf{A} - \mathbf{A}_0)\delta t\frac{\mathbf{u}^n - \mathbf{u}^{n-1}}{\delta t} \\
=& \mathcal{O}(\delta t)\mathcal{O}(1)\mathcal{O}(\delta t)\delta t\mathcal{O}(1) \\
=& \mathcal{O}(\delta t^3),
\end{aligned}$$

realizing the fact that

$$\mathcal{R}\begin{pmatrix}\mathbf{u}\\ p\end{pmatrix}=\begin{pmatrix}(\mathbf{A}-\mathbf{A}_0)\mathbf{u}\\ 0\end{pmatrix},$$

$$(\mathcal{A}\mathcal{C})^{-1}\begin{pmatrix}\mathbf{u}\\ 0\end{pmatrix}=\begin{pmatrix}\mathbf{A}_0^{-1}\mathbf{u}\\ 0\end{pmatrix}.$$

From the above analysis, we observe that by increasing the number of iteration to 2, we are able to make *explicit prediction* steps such as (2.27) while staying second order. By explicit, we mean that the equation only involves the inversion of a diagonal matrix. Furthermore, by induction, the order of accuracy grows the same way as the number of iteration m, at least formally. However, as m increases, the above analysis requires bounds for higher order spatial derivatives of $\partial\mathbf{u}/\partial t$. Therefore, in practice, larger m won't have a significant benefit on the result, which is tested in [4] for example.

In conclusion, a general framework (2.12) and (2.14) for solving pressure coupled systems (2.8), such as the Stokes system, is proposed. Popular methods such as projection method and SIMPLE-type methods can be easily represented within this framework. Order of accuracy can be formally derived. However, more rigorous mathematical analysis is required in future research to gain insights into the nature of this general method, and provide key information to optimal choices to the adjustable parts. For an overview of analysis on the projection method, see [20].

2.2 Finite Volume Discretization

When comparing the spatial discretization methods, the main advantage of *finite volume method* (FVM) is its conservativity, which is an desired properties in many applications. Mathematical analysis of the cell centered finite volume method can be found in [12, 13].

Consider the spatial domain which satisfy the following assumption

$$\begin{aligned} &\Omega \subset \mathbb{R}^d (d = 2, 3) \text{ is open, polygonal, bounded, connected,} \\ &\partial\Omega = \overline{\Omega} \backslash \Omega \text{ is Lipschitz-continuous.} \end{aligned} \tag{2.1}$$

In the above definition, polygonal means $\partial\Omega$ is a finite union of polygons in hyperplanes of $\mathbb{R}^d$. Admissible polygonal finite volume space discretization of Ω can be defined as the following.

Definition 2.5 (Admissible finite volume space discretization of Ω)**.** Let Ω be a domain satisfying (2.1). An *admissible finite volume space discretization of* Ω is defined by $(\mathcal{T}, \mathcal{E}, \mathcal{P})$ satisfying the following properties:

- $\mathcal{T}$ is a family of *control volumes*, which are open polygonal convex subsets of Ω such that

  $$\overline{\Omega} = \bigcup_{K \in \mathcal{T}} \overline{K}.$$

  For any $K \in \mathcal{T}$, $|K| > 0$ is the measure of K.

- $\mathcal{E}$ is a family of *edges*, which are subsets of $\overline{\Omega}$ contained in hyperplanes of $\mathbb{R}^d$. For any $\sigma \in \mathcal{E}$, $|\sigma| > 0$ is the $(d-1)$-dimensional (surface) measure. For any $K \in \mathcal{T}$, there exists a subset $\mathcal{E}_K$ of $\mathcal{E}$ such that

  $$\partial K = \overline{K} \backslash K = \bigcup_{\sigma \in \mathcal{E}_K} \overline{\sigma}.$$

  Let $\mathcal{T}_\sigma = \{K \in \mathcal{T} : \sigma \in \mathcal{E}_K\}$. For any $\sigma \in \mathcal{E}$, if $\sigma \in \partial\Omega$, then $\mathcal{T}_\sigma$ has exactly one element, and $\sigma \in \mathcal{E}_{\text{ext}}$, otherwise, $\mathcal{T}_\sigma$ have exactly two elements, and $\sigma \in \mathcal{E}_{\text{int}}$.

- $\mathcal{P}$ is a family of *points* $\{\mathbf{x}_K\}_{K \in \mathcal{T}}$, such that $\mathbf{x}_K \in K$. For all $\sigma \in \mathcal{E}_{\text{int}}$ with $\mathcal{T}_\sigma = \{K, L\}$, the line that is through $\mathbf{x}_K$ and $\mathbf{x}_L$ is orthogonal to σ and intersects

σ at $\mathbf{x}_\sigma$. For all $\sigma \in \mathcal{E}_{\text{ext}}$ with $\mathcal{T}_\sigma = K$, the line that is through $\mathbf{x}_K$ and orthogonal to σ intersects σ at $\mathbf{x}_\sigma$.

Note that some requirement in Definition 2.5 can be relaxed, such as the convexity of control volumes and orthogonality. In such cases, additional regularity conditions on the mesh and modification on the numerical schemes are usually needed to prove convergence.

We further list some notations to be used in the following. Let $d_{K,\sigma}$ be the distance from $\mathbf{x}_K$ to edge σ, and

$$d_{K,\sigma} = \begin{cases} d_{L,\sigma} + d_{K,\sigma} & \mathcal{T}_\sigma = \{K, L\}, \\ d_{K,\sigma} & \mathcal{T}_\sigma = \{K\}, \end{cases}$$

$$D_{K,\sigma} u = \begin{cases} u_L - u_K & \mathcal{T}_\sigma = \{K, L\}, \\ -u_K & \mathcal{T}_\sigma = \{K\}. \end{cases}$$

Let $\mathbf{n}_{K,\sigma}$ be the normal vector to σ outward from K. Let $H_\mathcal{T}$ be the family of piecewise constant function

$$\left\{ u \in L^2(\Omega) : u = \sum_{K \in \mathcal{T}} u_K \chi_K \right\}$$

where χ_K are the characteristic function of K.

Finite volume schemes are usually defined through local integration over control volume K. For $u \in H_\mathcal{T}$,

$$\int_K \nabla \cdot \mathbf{F}(u)\, dV = \sum_{\sigma \in \mathcal{E}_K} \int_\sigma \mathbf{F}(u) \cdot \mathbf{n}_{K,\sigma}\, dS = \sum_{\sigma \in \mathcal{E}_K} F_{K,\sigma}(u),$$

where $F_{K,\sigma}(u)$ is the approximation of the integral $\int_\sigma \mathbf{F}(u) \cdot \mathbf{n}_{K,\sigma}\, dS$. Conservativity

is guaranteed if

$$F_{K,\sigma}(u) + F_{L,\sigma}(u) = 0 \qquad \text{for } \mathcal{T}_\sigma = \{K, L\}.$$

For convection-diffusion equation with Dirichlet boundary condition

$$\begin{aligned} -\nabla^2 u + \nabla \cdot (\mathbf{v}u) &= f && \text{in } \Omega, \\ u &= 0 && \text{on } \partial\Omega, \end{aligned} \tag{2.2}$$

namely $\mathbf{F}(u) = \nabla u + \mathbf{v}u$, the numerical approximation $F_{K,\sigma}(u)$ can be proposed as

$$F_{K,\sigma}(u) = |\sigma| \frac{D_{K,\sigma}u}{d_\sigma} + v_{K,\sigma}\Pi_\sigma u,$$

where $v_{K,\sigma} = \int_\sigma \mathbf{v} \cdot \mathbf{n}_{K,\sigma}\, dS$, and $\Pi_\sigma u$ is the interpolation of $u \in H_\mathcal{T}$ to σ. The upwind scheme, namely

$$\Pi_\sigma u = \begin{cases} u_K & \text{if } v_{K,\sigma} \geq 0, \\ u_L & \text{if } v_{K,\sigma} < 0, \mathcal{T}_\sigma = \{K, L\}, \\ 0 & \text{if } v_{K,\sigma} < 0, \mathcal{T}_\sigma = \{K\}, \end{cases}$$

provides first order accuracy. Proof of existence, stability, and error analysis for equation (2.2) with upwind scheme on general admissible finite volume mesh defined by Definition 2.5 can be found in [17]. For higher order schemes in one dimension, we have the so called *total variation diminishing (TVD)* criteria, proposed in [22], to obtain stable second order approximation to hyperbolic conservation laws. For general mesh in higher dimensions, the extension is not straightforward, though one can apply the one dimensional theory anyway on the line $(\mathbf{x}_K, \mathbf{x}_L)$ in many computational codes. Mathematical analysis on this topic can be found in [11, 13].

2.3 Simulation of Compressible Two Phase Flows

In flows with two immiscible fluids, the phases are separated by a deformable interface, which could go through topological changes such as breakup or coalescence.

Volume-of-fluid (VOF) method [23] is used here. A volume fraction field is coupled into the system to differentiate the phases in a single fluid domain. The advantage of VOF is that it can easily deal with topological changes of the interface. Mass conservation of each phase is in line with the advantage of the finite volume method. Description and comparison with various other interface tracking methods can be found in a review [47]. The flavor of VOF method used do not a have an interface reconstruction step. Instead, the volume fraction field is advected by a discretization scheme of the differential equation.

To simulate compressible flow, a pressure based solver is used. A pressure based solver works similar to the methods described in Section 2.1. The pressure-based solver traditionally has been used for incompressible and mildly compressible flows, for example, water. The method used here is similar to the one described in [32].

Suppose there are two phases in the domain Ω satisfying (2.1). The density of them are, respectively, $\rho_i : \Omega \times [0, T_0] \to \mathbb{R}^+$, $i = 1, 2$. Let $\alpha_i : \Omega \times [0, T_0] \to [0, 1]$, $i = 1, 2$ be the volume fraction of the phases. We use α and α_1 interchangeably. Note that $\alpha_1 + \alpha_2 = 1$. Ideally, α would be a step function across the interface for immiscible fluids. However, it is assumed that it smears a little so that we can actually consider its gradient, etc. We will consider temperature $T : \Omega \times [0, T_0] \to \mathbb{R}^+$. Also, recall that the velocity is $\mathbf{u} : \Omega \times [0, T_0] \to \mathbb{R}^d$ and pressure is $p : \Omega \times [0, T_0] \to \mathbb{R}$.

Mass balance for each phase is given by

$$\frac{\partial(\alpha_i \rho_i)}{\partial t} + \nabla \cdot (\alpha_i \rho_i \mathbf{u}) = 0, \quad i = 1, 2, \tag{2.1}$$

Conservation of momentum is

$$\frac{\partial(\rho\mathbf{u})}{\partial t} + \nabla\cdot(\rho\mathbf{u}\mathbf{u}) - \nabla\cdot\boldsymbol{\tau} + \nabla p = \rho\mathbf{g} + \mathbf{f}_{\text{surf}}, \tag{2.2}$$

where $\mathbf{g}$ is the gravitational acceleration. The deviatoric stress tensor is

$$\boldsymbol{\tau} = \mu\left(\nabla\mathbf{u} + \nabla\mathbf{u}^{\top} - \frac{2}{3}(\nabla\cdot\mathbf{u})\boldsymbol{I}\right),$$

where μ is the effective mixture viscosity, $\rho = \alpha_1\rho_1 + \alpha_2\rho_2$ is the mixture density. We apply the continuum surface force model to describe the force caused by surface tension as

$$\mathbf{f}_{\text{surf}} = \gamma\kappa\nabla\alpha,$$

where γ is the given surface tension, κ is the curvature approximated by

$$\kappa = -\nabla\cdot\left(\frac{\nabla\alpha}{|\nabla\alpha| + \varepsilon}\right),$$

with a small number $\varepsilon > 0$ to avoid division by zero.

For general compressible fluid, pressure, density and temperature are linked through an *equation of state*. Since we are using a pressure based solution algorithm. Density is expressed in terms of pressure and temperature

$$\rho_i = \rho_i(p, T), \tag{2.3}$$

which is locally linearized at given p and T,

$$\frac{d\rho_i}{dt} = \psi_i(p, T)\frac{dp}{dt},$$

in order to facilitate implicit pressure treatment caused by compressible effects. The time derivative d/dt here is understood as *material derivative*. In particular, for water, a linear equation of state used by us here is

$$\rho_1 = \rho_0 + \psi_1 p,$$

where ρ_0 and ψ_1 are constants, while for air, ideal gas law is used as

$$\rho_2 = \frac{M}{RT} p, \quad \psi_2 = \frac{M}{RT},$$

where M is the molar mass, and R is the gas constant.

The energy balance for each phase is

$$\begin{aligned} &\frac{\partial(\alpha_i \rho_i (e_i + k))}{\partial t} + \nabla \cdot (\alpha_i \rho_i \mathbf{u}(e_i + k)) \\ &-\nabla \cdot (\kappa_i \nabla T) \frac{\alpha_i \rho_i}{\rho} + \nabla \cdot (p\mathbf{u} - \boldsymbol{\tau} \cdot \mathbf{u}) \frac{\alpha_i \rho_i}{\rho} - \alpha_i \rho_i \mathbf{g} \cdot \mathbf{u} = 0, \end{aligned} \tag{2.4}$$

where $e_i = C_{Vi} T$ is the internal energy per unit mass for phase i, $k = \frac{1}{2} |\mathbf{u}|^2$ is the kinetic energy per unit mass, and κ_i is the thermal conductivity for phase i.

The above governing equations are reformulated to suit the needs of a pressure-based solver. First, equation (2.1) is expanded by product rule to get

$$\begin{aligned} \frac{\partial \alpha_i}{\partial t} + \nabla \cdot (\alpha_i \mathbf{u}) &= -\frac{\alpha_i}{\rho_i} \left(\frac{\partial \rho_i}{\partial t} + \mathbf{u} \cdot \nabla \rho_i \right) \\ &= -\frac{\alpha_i}{\rho_i} \frac{d\rho_i}{dt} = -\frac{\alpha_i \psi_i}{\rho_i} \frac{dp}{dt}. \end{aligned} \tag{2.5}$$

Summing the above equations for $i = 1, 2$, we get

$$\nabla \cdot \mathbf{u} = -\left(\frac{\alpha_1 \psi_1}{\rho_1} + \frac{\alpha_2 \psi_2}{\rho_2} \right) \frac{dp}{dt}. \tag{2.6}$$

By reformulation right hand side of equation (2.5) with $i = 1$, we get

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha \mathbf{u}) = \alpha \nabla \cdot \mathbf{u} + \alpha(1-\alpha)G, \tag{2.7}$$

where

$$G = \left(\frac{\psi_2}{\rho_2} - \frac{\psi_1}{\rho_1}\right)\frac{dp}{dt} = \left(\frac{\psi_2}{\rho_2} - \frac{\psi_1}{\rho_1}\right)\left(\frac{\partial p}{\partial t} + \mathbf{u} \cdot \nabla p\right).$$

Then, equation (2.4) is expanded by product rule, then use (2.1) to obtain

$$\begin{aligned}
&\alpha_i \rho_i C_{Vi}\frac{\partial T}{\partial t} + \alpha_i \rho_i C_{Vi}\mathbf{u} \cdot \nabla T + \alpha_i \rho_i \frac{\partial k}{\partial t} + \alpha_i \rho_i \mathbf{u} \cdot \nabla k \\
&-\nabla \cdot (\kappa_i \nabla T)\frac{\alpha_i \rho_i}{\rho} + \nabla \cdot (p\mathbf{u} - \boldsymbol{\tau} \cdot \mathbf{u})\frac{\alpha_i \rho_i}{\rho} - \alpha_i \rho_i \mathbf{g} \cdot \mathbf{u} = 0,
\end{aligned}$$

Divide the above equation by $\rho_i C_{Vi}$, sum for $i = 1, 2$, then multiply by ρ, we get

$$\begin{aligned}
&\rho\frac{\partial T}{\partial t} + \rho \mathbf{u} \cdot \nabla T - a\nabla^2 T \\
&+\left(\frac{\alpha_1}{C_{V1}} + \frac{\alpha_2}{C_{V2}}\right)\left(\rho\frac{\partial k}{\partial t} + \rho \mathbf{u} \cdot \nabla k + \nabla \cdot (p\mathbf{u} - \boldsymbol{\tau} \cdot \mathbf{u}) - \rho \mathbf{g} \cdot \mathbf{u}\right) = 0,
\end{aligned}$$

where

$$a = \frac{\alpha_1 \kappa_1}{C_{V1}} + \frac{\alpha_2 \kappa_2}{C_{V2}}.$$

Finally, use $\partial \rho / \partial t + \nabla \cdot (\rho \mathbf{u}) = 0$, we get

$$\begin{aligned}
&\frac{\partial (\rho T)}{\partial t} + \nabla \cdot (\rho \mathbf{u} T) - a\nabla^2 T \\
&+\left(\frac{\alpha_1}{C_{V1}} + \frac{\alpha_2}{C_{V2}}\right)\left(\frac{\partial (\rho k)}{\partial t} + \nabla \cdot (\rho \mathbf{u} k) + \nabla \cdot (p\mathbf{u} - \boldsymbol{\tau} \cdot \mathbf{u}) - \rho \mathbf{g} \cdot \mathbf{u}\right) = 0.
\end{aligned} \tag{2.8}$$

In summary, we have unknowns α, $\mathbf{u}$, p, ρ_1 ρ_2, and T. On the other hand, we have equation (2.7) to solve α, equations (2.2) and (2.6) as one pressure-coupled system to solve $\mathbf{u}$ and p, equation (2.8) to solve T, and equations of state (2.3) to solve ρ_i.

The equation for pressure is a little different from the incompressible case described in Section 2.1. But since $\nabla \cdot \mathbf{u}$ is implicitly related to p in (2.6), we can modify the equation as

$$\frac{\delta t}{\rho}\mathbf{D}\mathbf{P}^{-1}\mathbf{G}p^n = \mathbf{D}\tilde{\mathbf{u}} + \left(\frac{\alpha_1\psi_1}{\rho_1} + \frac{\alpha_2\psi_2}{\rho_2}\right)\left(\frac{p^n - p^{n-1}}{\delta t} + \mathbf{D}(\tilde{\mathbf{u}}p^n) - p^n\mathbf{D}\tilde{\mathbf{u}}\right)$$

The equations mentioned above are solved in sequence in each time step , and iterated if necessary.

2.4 Numerical Test Problems

The first test case is the Taylor-Green vortex solution to the time-dependent incompressible Navier-Stokes equations. The domain is $(x, y) \in [0, 2\pi]^2$, and pick $\rho = 1$, $\mu = 1$ the exact solution is

$$\mathbf{u} = e^{-\frac{2\mu t}{\rho}}\begin{pmatrix}\sin(x)\cos(y)\\ -\cos(x)\sin(y)\end{pmatrix},$$

$$p = e^{-\frac{4\mu t}{\rho}}(\cos 2x + \cos 2y)\frac{\rho}{4}.$$

The spatial discretization is 200×200. Exact solution is imposed at the boundary of the domain. PISO algorithm is used for pressure coupling, with two outer iterations because of the nonlinear convective term. The convective term uses the van Leer limiter. Crank-Nicolson scheme is used for temporal discretization. Numerical convergence test is shown in Figure 2.1. The exact solution and error distribution can be seen in Figure 2.2.

The second test is the one dimensional shock tube solved by a pressure-based solver for the time-dependent compressible system, or the Euler equations, on the domain $x \in [-0.5, 0.5]$. The solver also solves an energy equation. The initial condi-

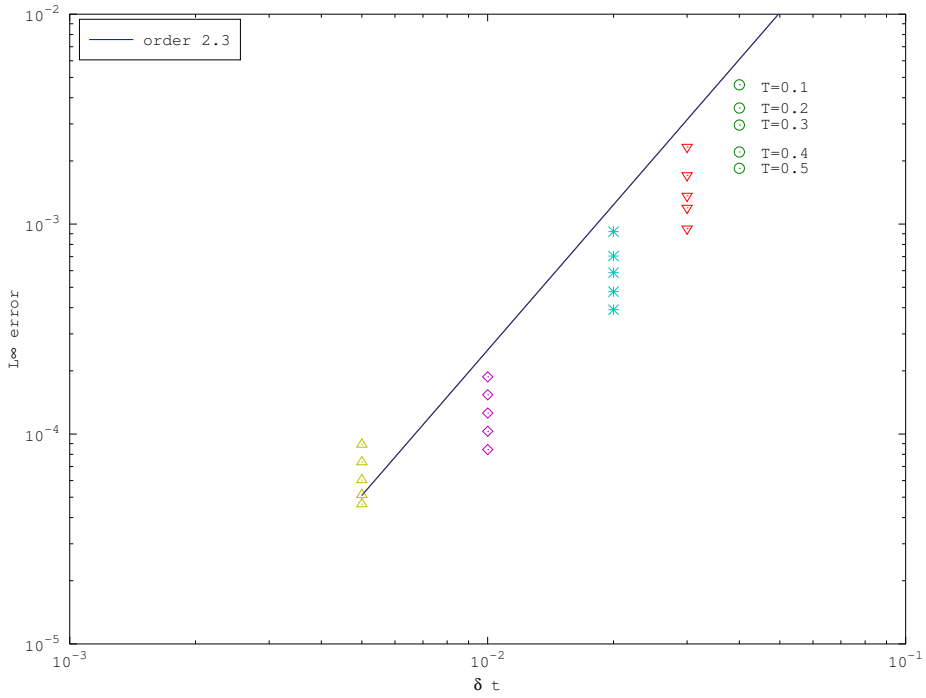


Figure 2.1: Error convergence test for Taylor-Green vortex solution of the time-dependent incompressible Navier-Stokes system. The convergence order with respective to time step size (δt) is approximately 2.3 before it slows down.

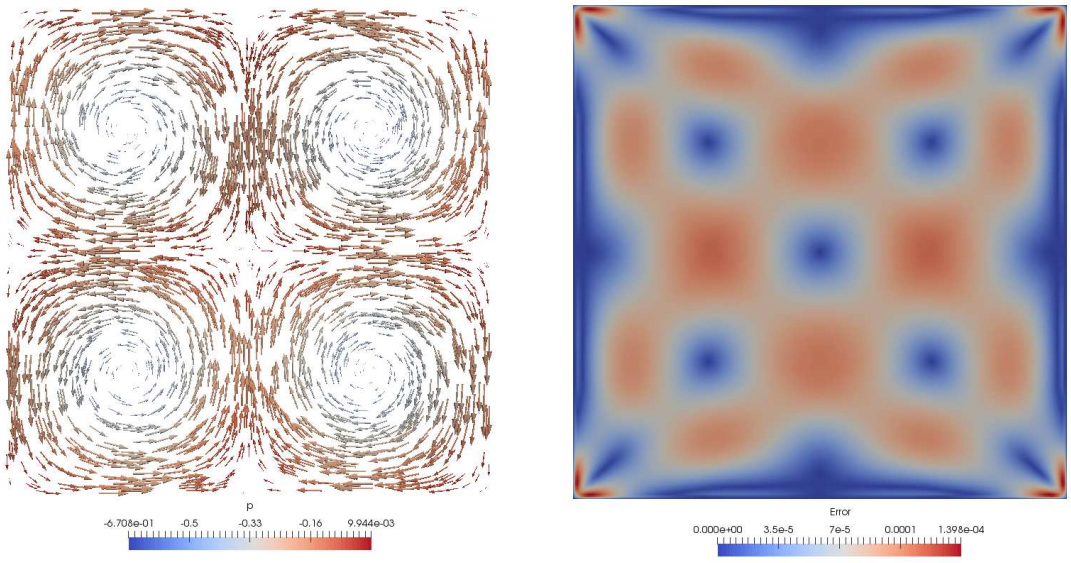


Figure 2.2: The figure on the left is the exact Taylor-Green solution at t = 0.2 sec. The figure on the right is the error distribution of the solution with δt = 0.01 sec and t = 0.2 sec.

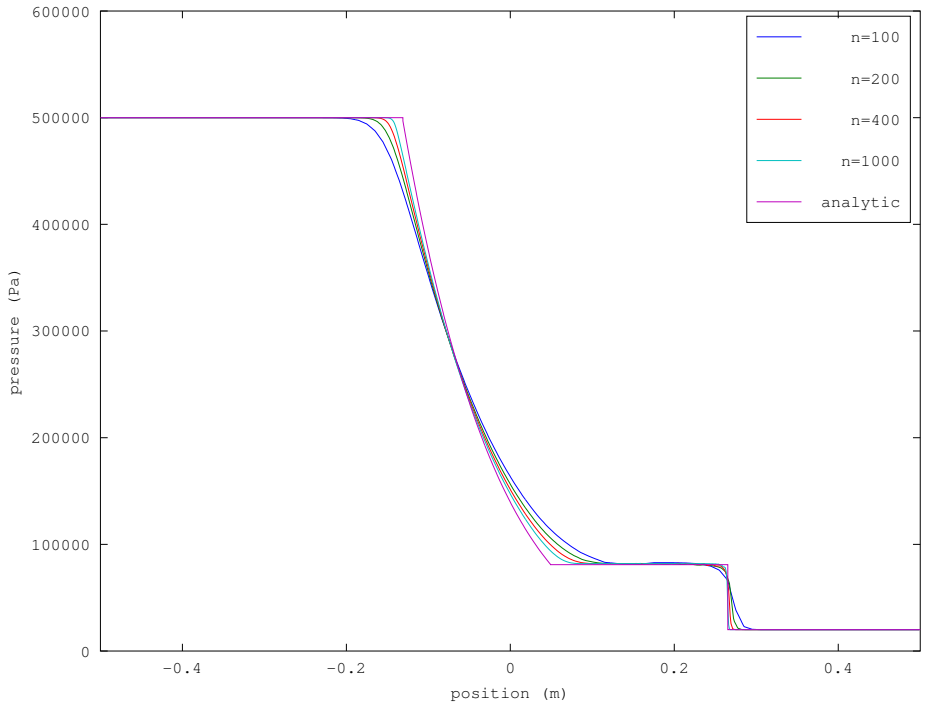


Figure 2.3: Numerical Solution for the pressure of a shock tube at $t = 0.4$ ms, with different spatial resolution n, in comparison with the analytical solution.

tion for the high pressure side ($x \in [-0.5, 0]$) is $p = 5 \times 10^5$ Pa, and for the low pressure side ($x \in (0, 0.5]$) is $p = 2 \times 10^4$ Pa. Both sides have an initial temperature $T = 303$ K, and zero initial velocity. The gas is modeled as ideal air, with 28.9 g/mol mass. Upwind scheme is used for the convective term, and Crank-Nicolson scheme is used for temporal discretization. The Numerical solutions for the pressure at $t = 0.4$ ms are shown in Figure 2.3 with different spatial resolutions n in comparison with the analytical solution.

3. WATER ENTRY OF AN AIRLINER*

On March 8, 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 a bird strike on take-off from La Guardia Airport. There was no loss of life.

Figure 3.1 show our "representation" of a commercial airliner, a Boeing 777 model, plunging into the ocean. (See our commentary in Box 1 of Section 3.1). Such simulations can help to understand the physical mechanisms at work and also to improve passenger safety. But these are highly challenging simulations that require the cooperation of engineers, mathematicians and computational scientists. Any scientific investigation of the mishap, apart from human factors of foul play and conspiracy, appears mostly of an *engineering nature*, such as machine and instrumentation breakdown, midair explosion, weather, navigation, etc. But this should not prevent mathematicians' curiosity—and our fascination with airplanes since childhood—from entering the fray to also add and contribute something valuable, regarding this in-

*Part of this section is first published in Notices of the American Mathematical Society 62 (April 2015): 330-344; © by the 2015 American Mathematical Society (www.ams.org).

vestigation and recovery effort. The fact is, mathematics is closely intertwined with engineering, and is not detached from the "real world" as some people may think. The statement made by the Malaysian Prime Minister that Flight MH370 "ended" in the Southern Indian Ocean was based on the assessment by the British company *Inmarset*. An article articulating how the radar signal backtracking made by Inmarsat works was published in SIAM News[1], where John Zweck of the Math Dept of the University of Texas - Dallas argued in support of Inmarsat by using the Doppler frequency shift, time and locations of ping, trigonometry and other mathematical methods and MATLAB ® software. Nevertheless, Inmarsat's radar tracking methodology and data analysis have not yet convinced everybody that they are ironclad; see some counter arguments by David Finkleman in [16], for example. (Dr. Finkleman is Director of Studies and Analysis, and Senior Scientist, North American Aerospace Defense Command and U.S. Space Command, at Peterson Air Force Base, Colorado.)

We discuss this air incident from a mathematical as well as interdisciplinary perspective. We show how computational mathematics and mechanics can help us understand the physical nature of an aircraft emergency water landing, how to model and compute it, and how this knowledge is helping safe civil aviation and other aerospace related undertakings. This kind of problems has become more and more typical for the work of a mathematician as part of an interdisciplinary team in industry or government labs.

3.1 The Water Entry Problem Revisited

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.

[1] `https://sinews.siam.org/DetailsPage/tabid/607/ArticleID/146/How-Did-Inmarsat-Deduce-Possible-Flight-Paths-for-MH370.aspx`

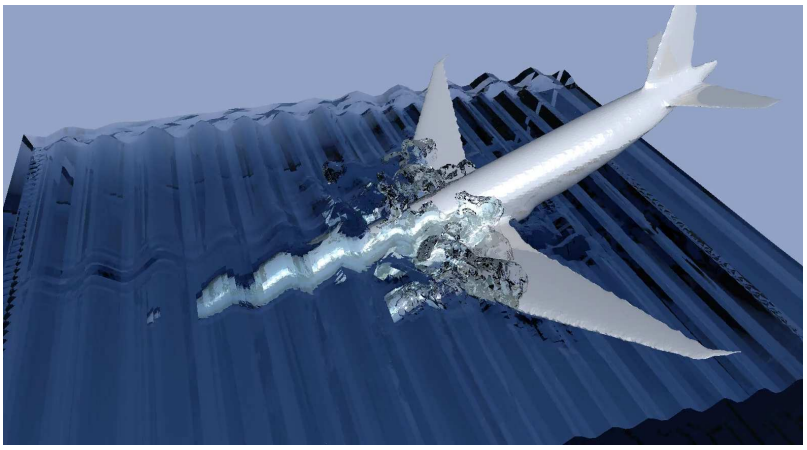


Figure 3.1: The aircraft is a Boeing 777 model flying into ocean at the speed of 70m/sec, with pitch angle = -20°, at time t=0.36 sec. A volume-of-fluid (VOF) scheme in OpenFOAM is used to simulate the two-phase flow for the fluid-aircraft body interaction.

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 mathematician 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 [43]; see Figure 3.2. 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,

$$\frac{d}{dt}\left[(M + m(t))\dot{\zeta}(t)\right] = Mg - F_B - F_C - F_D \qquad \text{(cf. [1, eq. (2.3)])} \qquad (3.1)$$

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.

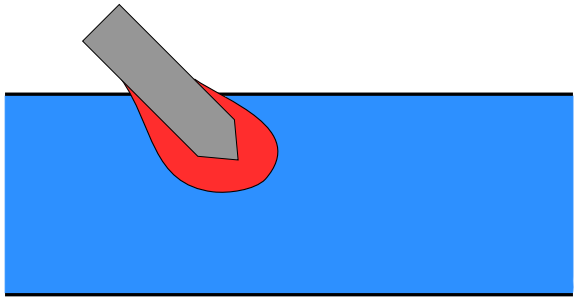


Figure 3.2: Von Karman's idea of "added mass" for the water entry problem, which is an idealization and simplification. Here the red region represent "added mass". This is the mass moving together with the mass of the wedge projectile. The portion of the (red) added mass lying above the still water surface is called the "pile up".

We note that the precise value of added mass $m(t)$ is not known. For small time or 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 [44] 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 [14] 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 [40] for a normal incidence problem, and by Garabedian [18] for oblique incidence, for example. These 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 [1], where 476 references are listed, and where a dozen more mathematical

(-oriented) papers than [40] and [18] 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 [30]: (1) cavity-opening and jet splashing; (2) cavity-closing and formation of an air pocket; and (3) cavity-detachment and cavitation; see Figure 3.3. A good way to capture the rich physics is through state-of-the-art *computational fluid dynamics (CFD)*. The CFD 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 (3.1) naturally into the two-phase fluid-structure interaction models.

3.2 Simulation of the Ditching/Crashing of an Aircraft into Water as a Two-Phase Fluid-Structure Interaction Problem

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. Our objective in this section is to conduct numerical simulations for several hypothetical scenarios using CFD.

For a representative Boeing 777 aircraft, we use the values of parameters as given in Table 3.1.

The underpinning subject of this study is *continuum mechanics*, including the

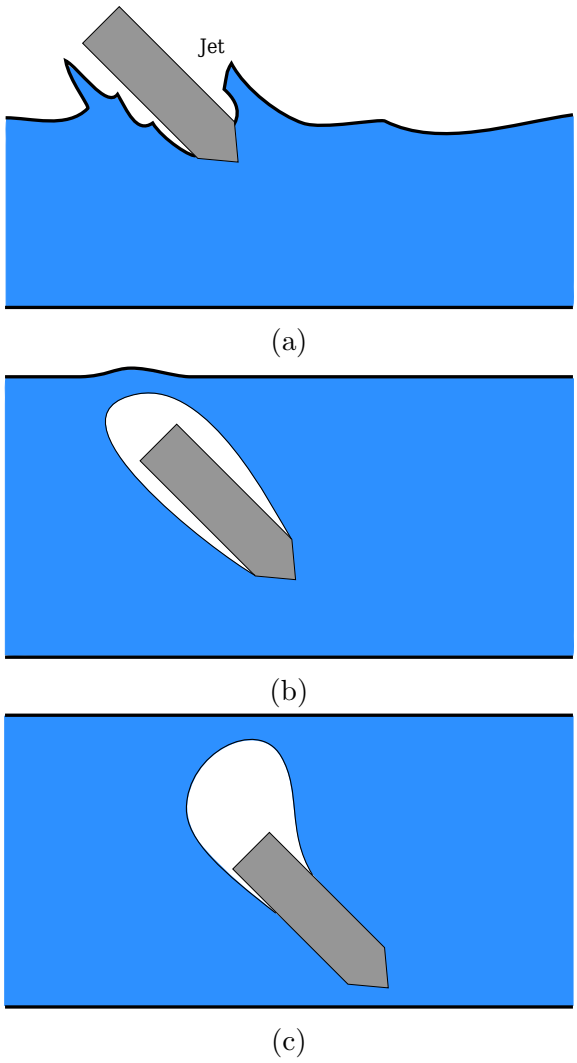


Figure 3.3: The several phases of a projectile entering water according to Mackey [30]: a a cavity of air opens; b a cavity of air pocket encloses the projectile when it is totally submerged; and c the cavity begins to be detached from the projectile, leaving it totally surrounded by water. Some water vapor may exist in the cavity, and cavitation usually happens. (Adapted from [1, p. 060803-2])

Total weight	1.8×10^5 kg
Wing span	60.9 m
Fuselage cross section	29.6 m^2
Length	63.7 m
Roll Moment of Inertia	1.06×10^7 kg m^2
Pitch Moment of Inertia	2.37×10^7 kg m^2
Yaw Moment of Inertia	3.34×10^7 kg m^2

Table 3.1: Parameter values for Boeing 777 used in CFD calculations.

Atmospheric pressure	1×10^5 Pa
Lower bound for pressure	1×10^4 Pa
Kinematic viscosity of water	1×10^{-6} m^2/sec
Kinematic viscosity of air	1.589×10^{-5} m^2/sec
Water-air surface tension (γ)	0.07 N/m
Gravitational acceleration (g)	9.80665 m/sec^2
Reference density of water (ρ_0)	1000 kg/m^3
Compressibility of water (ψ_1)	1×10^{-5} sec^2/m^2
Constants in $k-\epsilon$ turbulence model	$C_\mu = 0.09, C_1 = 1.44, C_2 = 1.92, \sigma_\epsilon = 1.3$
Initial values for $k-\epsilon$ turbulence model	$k = 0.1$ $m^2/sec^2, \epsilon = 0.1$ m^2/sec^3
Initial aircraft speed relative to stationary water (V_0)	58 m/sec ($\approx$ 130 mph)

Table 3.2: Parameter values for fluid flow used in CFD calculations.

The splashing and piling up of water waves surrounding the submerged part of the aircraft are close to realism, as the motion of the free (water) surface is modeled and computed by the *volume-of-fluid method*. We have also used the *level-set method* and obtained similar graphical results. However, several other physical factors and phenomena have not been taken into account:

- The *deceleration* of the aircraft motion, as its speed is maintained at 70m/sec. In addition, in general, the presence of water will cause *deflection* of the flight path.
- At the speed of 70 m/sec, *structural fracture and disintegration* of aircraft are likely to occur.
- Hydrodynamic force, fluid buoyancy, and drag force have not been incorporated into the model.

Box 1: Commentary on the water-entering motion of aircraft as shown in Figure 3.1

We are dealing with two fluids: air and water. Depending on the operating conditions (speed and altitude), we can regard air either as compressible or incompressible. For water, as a liquid, it is generally considered as *incompressible*. However, if we choose incompressibility as model for water here, the CFD calculations have *severe difficulty of convergence*. A likely cause is that in water landing situations, local contact interface pressure can get *very high*, in the order of 10^6 Pascal, causing a compressed state of water. Therefore, we choose compressibility for both air and water as in [21].

Box 2: Modeling selections: compressible or incompressible?

water-entry problem first as *fluid-structure interaction with a free fluid-gas interface* and the subsequent impact and structural failure analysis. Here, water and air are modeled as compressible flows using the Navier-Stokes equations; cf. Box 2. Our mathematical model is similar to that in Guo et al. [21].

The CFD software we have adopted here is *OpenFOAM*[2], which is open-source and now widely used by industry and research communities. See an introductory article by several of us in [9]. In particular, we will be using `compressibleInterDyMFoam` for two-phase flow, and RANS $k-\epsilon$ for *turbulence modeling*. (See some mathematical study on the $k-\epsilon$ turbulence modeling in [33, 36], for example.) Computations were performed on the EOS supercomputer at Texas A&M University and RAAD supercomputer at Texas A&M University at Qatar. For the computational work shown in the examples of this section, each run took one to several days on the campus supercomputers.

We assume that the aircraft is a *rigid body*. Except for the sample case shown in Figure 3.1, we did not include the under-wing engines in the Boeing 777 aircraft, with the understanding that the strut-mounted engine nacelles would likely be the first things to be torn off in a water-entry situation. (But, computationally, it is

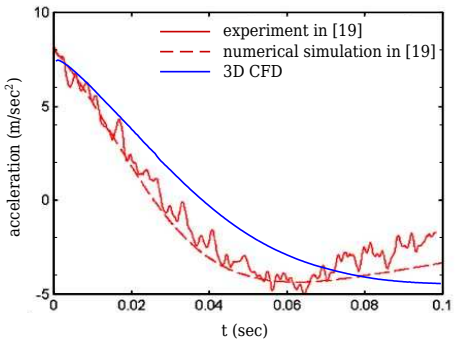
[2]OPENFOAM® is a registered trade mark of OpenCFD Limited, the producer of the OpenFOAM software.

straightforward to include the engines in our CFD work, such as shown in Figure 3.1.)

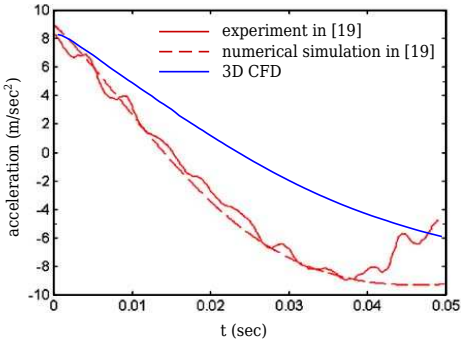
Method and algorithm of simulation is described in Section 2. Note that at time $t = 0$, the velocity of the center of mass of the aircraft is V_0 along various angles of approach; cf. Table 3.2. Various physical and computational parameter values are listed in Table 3.2.

Remark 3.1. Every CFD treatment needs to be validated. Why? CFD approaches have their roots in *theory, experiments and computation*. Validation determines if the computational results agree with physical reality – the experimental data. CFD codes must produce numerical results of desired accuracy so that they can be used with confidence. Here we use the experimental data available in [48] for a simplified scenario, that is, a constrained free-falling "wedge" entering water. The wedge has only the vertical translational degree of freedom. The *acceleration(/deceleration)* of the wedge is measured throughout its impact with the water. The study in [48] also employed a 2D potential flow model to study the problem numerically. In order to validate our CFD method, the setup for the experiment is replicated as a 3D mesh. Figure 3.4 shows the comparison of the acceleration time curves with a variety of parameters. (One of the varying parameters, the "deadrise angle", is defined as the angle formed between the angled-side of the wedge with the horizon.) Although some differences of values are observed, our CFD simulation shows a strong qualitative match of the acceleration/deceleration curves. We also note that the numerical model in [48] is a very simplified one without the incorporation of several aero-hydrodynamic effects.

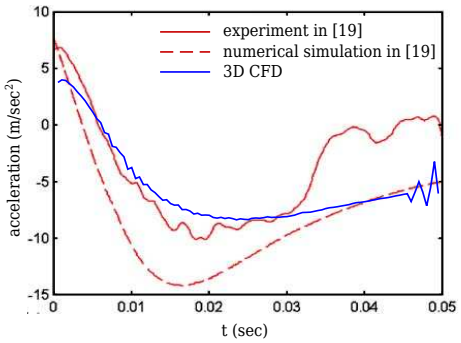
Aviation experts generally agree that *how the airliner enters the water determines its breakup*, which then gives major clues and directions of the search operations.



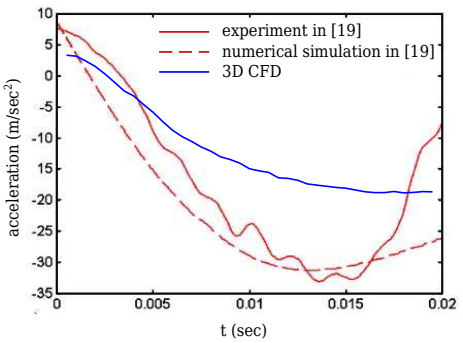
(a) deadrise angle is $\pi/4$, effective gravity is 8.0062 m/sec^2, mass of wedge is 13.522 kg, speed at water entry is 0.95623 m/sec.



(b) deadrise angle is $\pi/4$, effective gravity is 8.9716 m/sec^2, mass of wedge is 30.188 kg, speed at water entry is 1.69673 m/sec.



(c) deadrise angle is $\pi/9$, effective gravity is 7.8144 m/sec^2, mass of wedge is 12.952 kg, speed at water entry is 0.86165 m/sec.



(d) deadrise angle is $\pi/9$, effective gravity is 8.6103 m/sec^2, mass of wedge is 29.618 kg, speed at water entry is 1.54405 m/sec.

Figure 3.4: Curves of acceleration versus time as benchmarks in comparisons with Wu et al. [48, p. 28]. The curves obtained from experiment and numerical simulations are compared under different settings. The blue curves represent the data obtained by our computational methods in this paper.

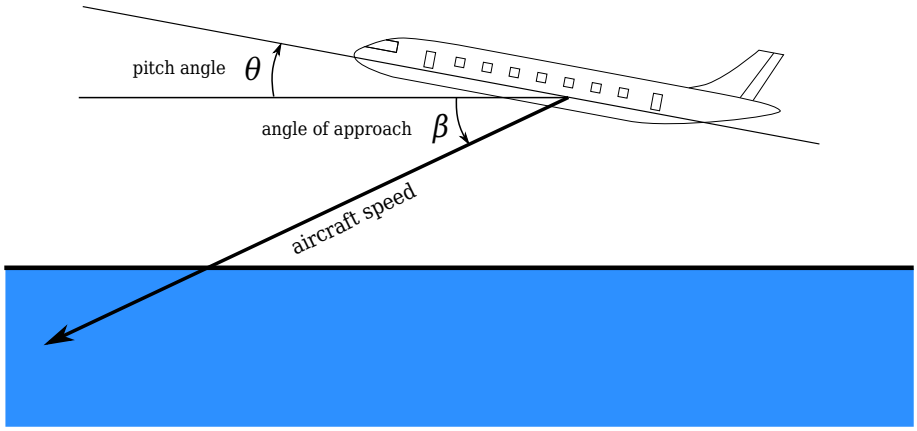


Figure 3.5: Angle θ here is the pitch angle signified in the computations of case 1-5 and β is the angle of approach. The speed of the aircraft denotes the speed of its center of mass.

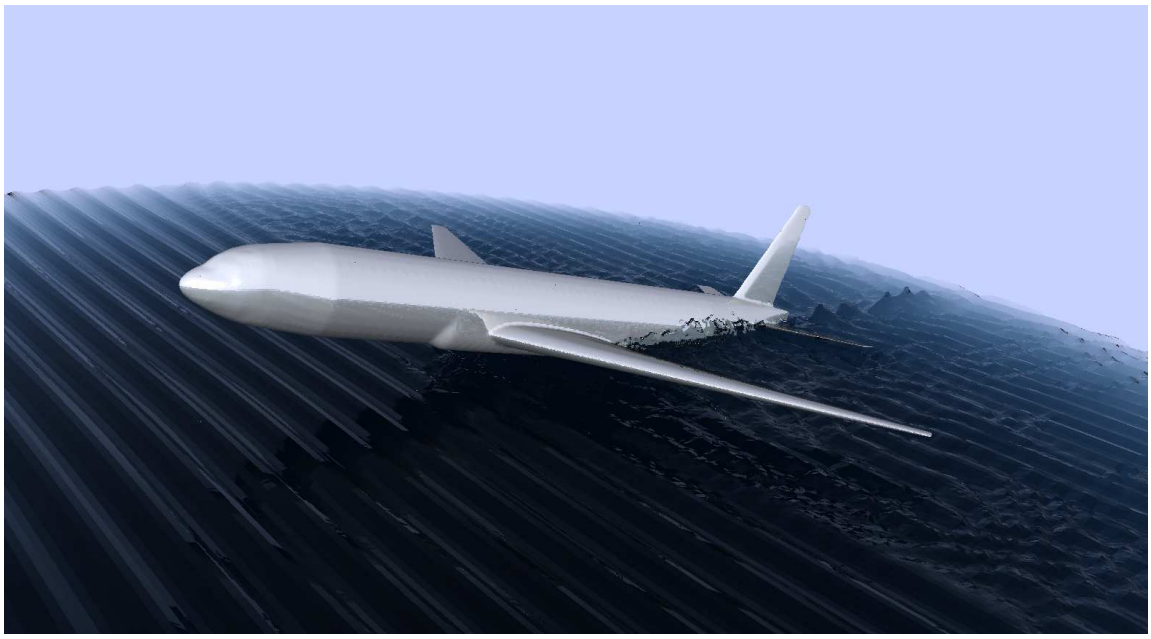
Therefore, in the following, we provide *five scenarios* of water entry. In each case, we provide comments, schematics, snapshots and a CFD animation. Each animation consists of two parts, with the first part showing visual effects and with the second part showing pressure loading.

Case 1: pitch angle = 8°, angle of approach = 1°

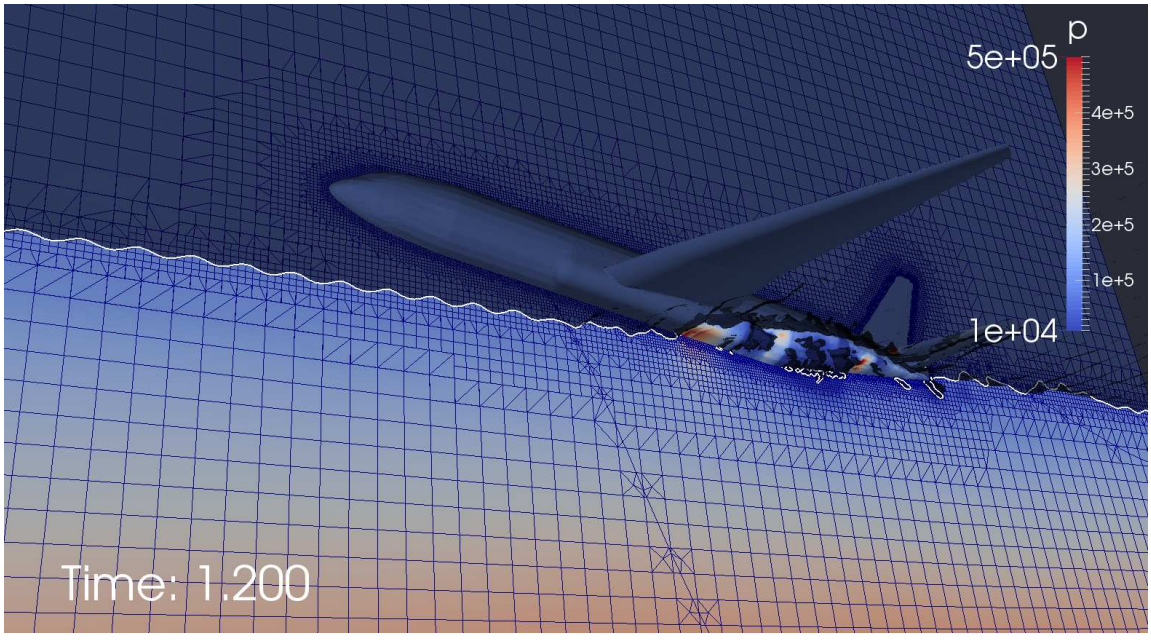
This is what one might call *glided ditching* similar to the US Airways Flight 1549 mentioned in Section 3.4.2; see Figure 3.6 and the accompanying animation. The vertical component of velocity of the airliner is found from (3.3) in the next section to be 1–2 m/sec. This is much smaller than the critical speed V_{cr} = 15–20 m/sec for structural failure in the next section and, thus, is good. See also Figure 3.7 for the interpretations of motion.

Case 2: pitch angle = −3°, angle of approach = 3°

See Figure 3.8 and its animation. Here we see an interesting phenomenon, namely, even though the original pitch angle is *negative*, the aircraft will



(a) gliding water entry



(b) pressure distribution and mesh

Figure 3.6: Pitch angle = 8°, angle of approach = 1°. This corresponds to Case 1.

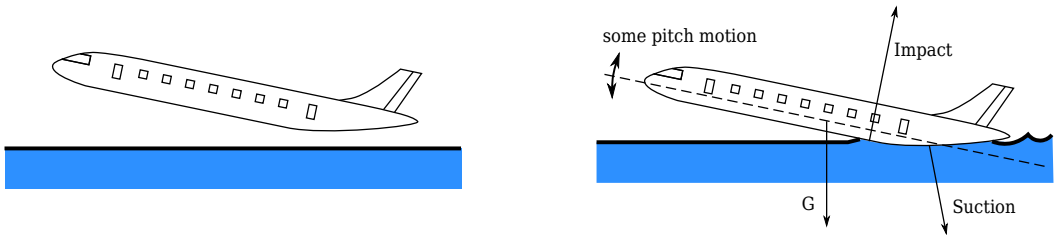


Figure 3.7: Schematics for the process of glided ditching. Major forces are illustrated. This corresponds to Case 1.

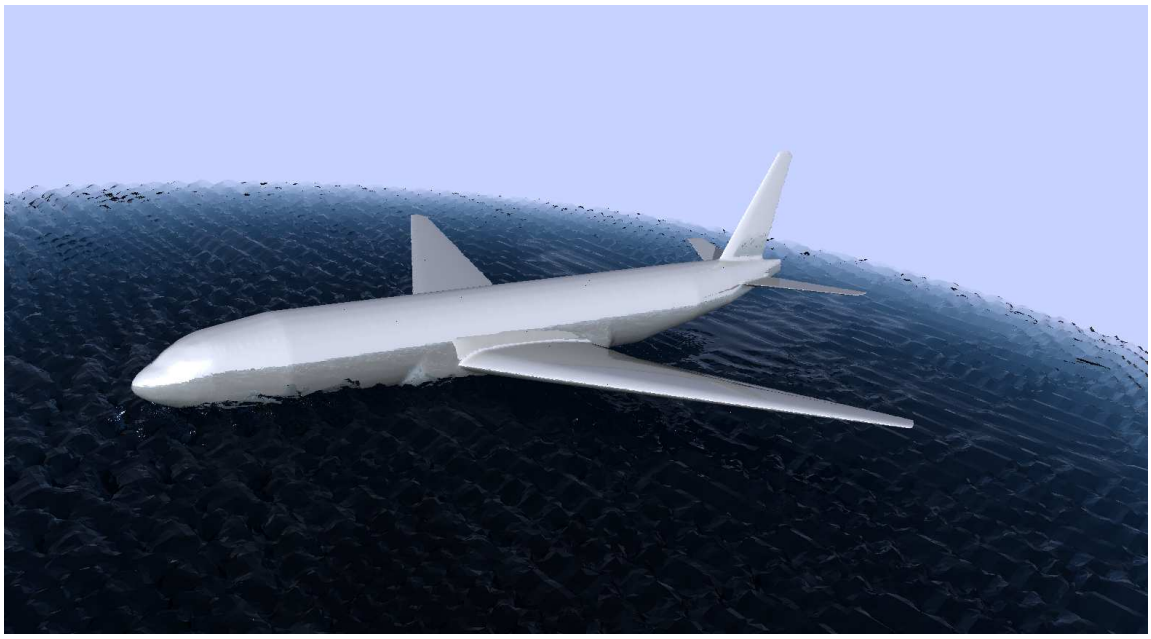
"bounce" on the water and make the pitch angle *positive*. See Figure 3.9. At the moment this happens, the bottom of the midsection (fuselage-water contact surface) of the aircraft undergoes high bending moment and surface pressure. This may cause the aircraft to break up in the middle section, a global failure to be described in the following section.

Case 3: pitch angle = $-30°$, angle of approach = $30°$

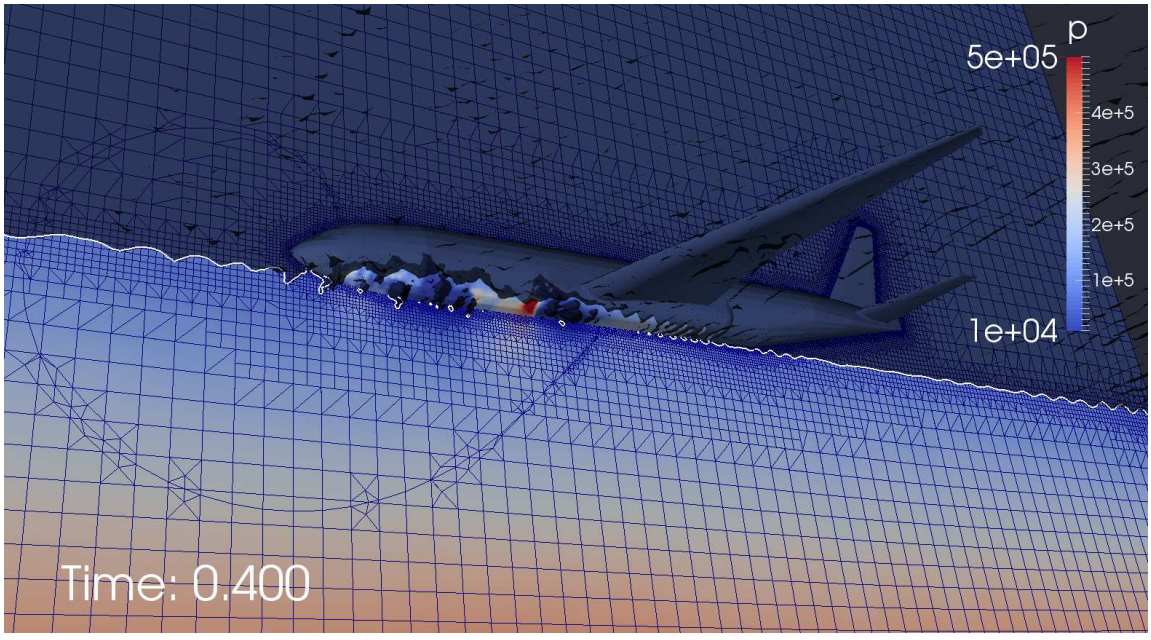
See Figure 3.10 and its animation. Here we see that the aircraft nose is subject to high pressure throughout the time sequence. See also the schematics in Figure 3.11 in contrast to Figure 3.9. Once the wings enter the water, the leading edge of the wing also is subject to high pressure loading up to 10^6 Pa.

Case 4: pitch angle = $-90°$, angle of approach = $93°$

See Figure 3.12 and its animation. This is a *nose-dive* situation. Here we further assume that the ocean current flows from left to right at a velocity of 3 m/sec. Then once the aircraft enters the water, the current gradually drives the aircraft toward the 5 o'clock direction. Eventually this could cause it to fall on the ocean floor *belly-up*. See Figure 3.13. Cf. more discussions in Box 3.



(a) gliding water entry (with a negative initial pitch)



(b) pressure distribution and mesh

Figure 3.8: Pitch angle = -3°, angle of approach = 3°. This corresponds to Case 2.

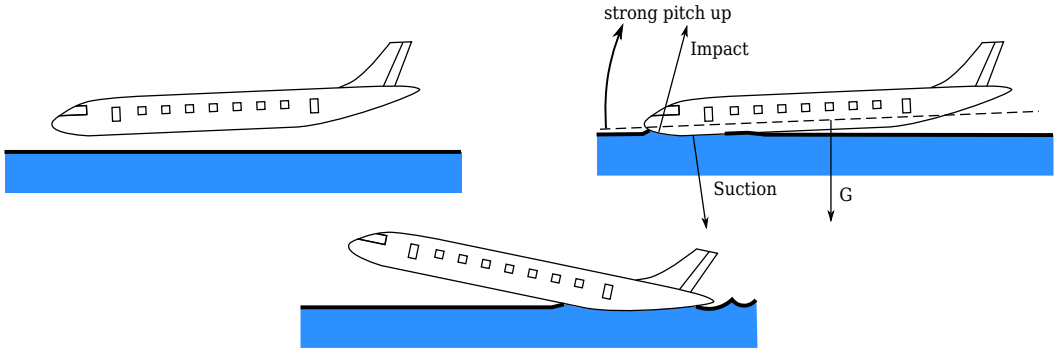


Figure 3.9: Schematics for the process of ditching with negative initial pitch. The plane is able to recover to the glided ditching attitude similar to Figure 3.7. This corresponds to Case 2.

If an aircraft stalls in a climb, or if any control surfaces – ailerons, rudder and stabilizers – malfunction, or if it runs out of fuel and the autopilot stops working (while the pilots are incapcitated or are deliberate), it can fall into a steep nose-dive or even vertical drop (our Case 4 here).

What happens upon water-entry? Here, we directly quote the article "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`)

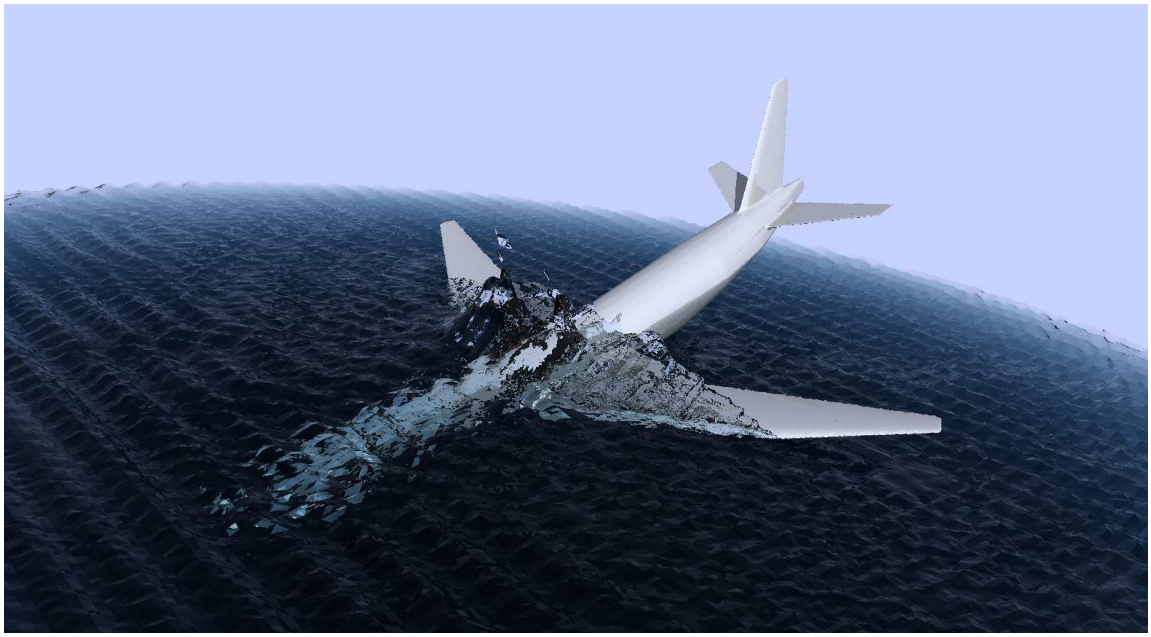
" ... The wings and tail would be torn away and the fuselage could reach a depth of 30 meters or 40 meters within seconds, then sink without resurfacing. Wing pieces and other heavy debris would descend soon afterward.

Whether buoyant debris from the passenger cabin – things like foam seat cushions, seatback tables and plastic drinking water bottles – would bob up to the surface would depend on whether the fuselage ruptured on impact, and how bad the damage was.

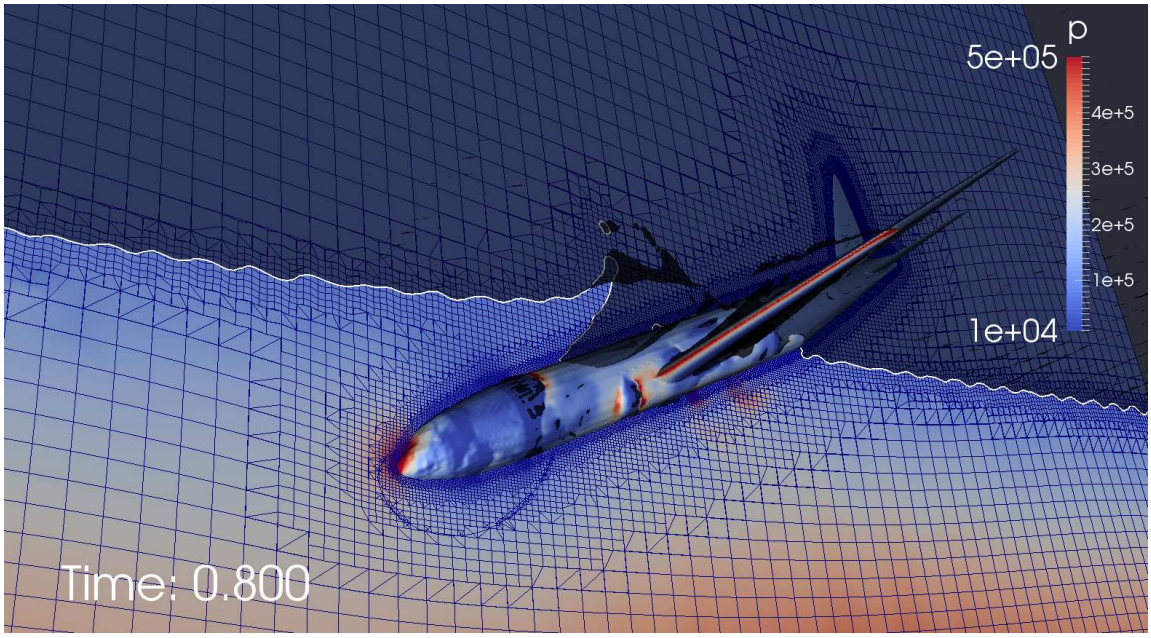
"It may have gone in almost complete somehow, and not left much on the surface," said Jason Middleton, an aviation professor at Australia's University of New South Wales. ..."

This may well offer a powerful clue as to why so frustratingly none of the debris of MH370 has been found so far.

Box 3: Does nose-dive have anything to do with the lack of debris?



(a) diving water entry



(b) pressure distribution and mesh

Figure 3.10: Pitch angle = −30°, angle of approach = 30°. This corresponds to Case 3.

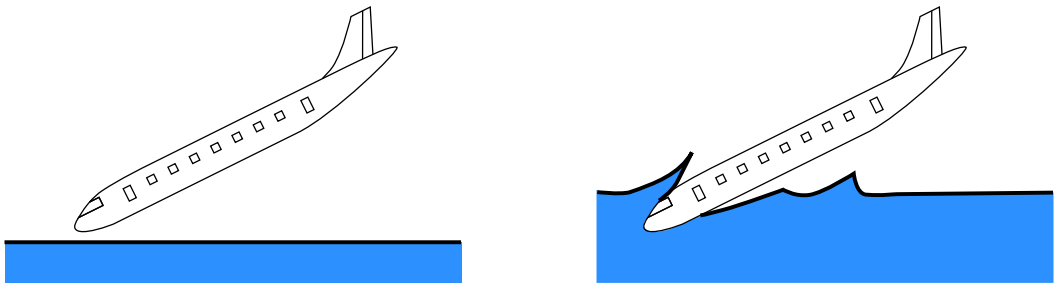


Figure 3.11: The pitch angle is too negative to recover to the glided ditching attitude. The plane's nose dives into the water with little bouncing motion. This corresponds to Case 3.

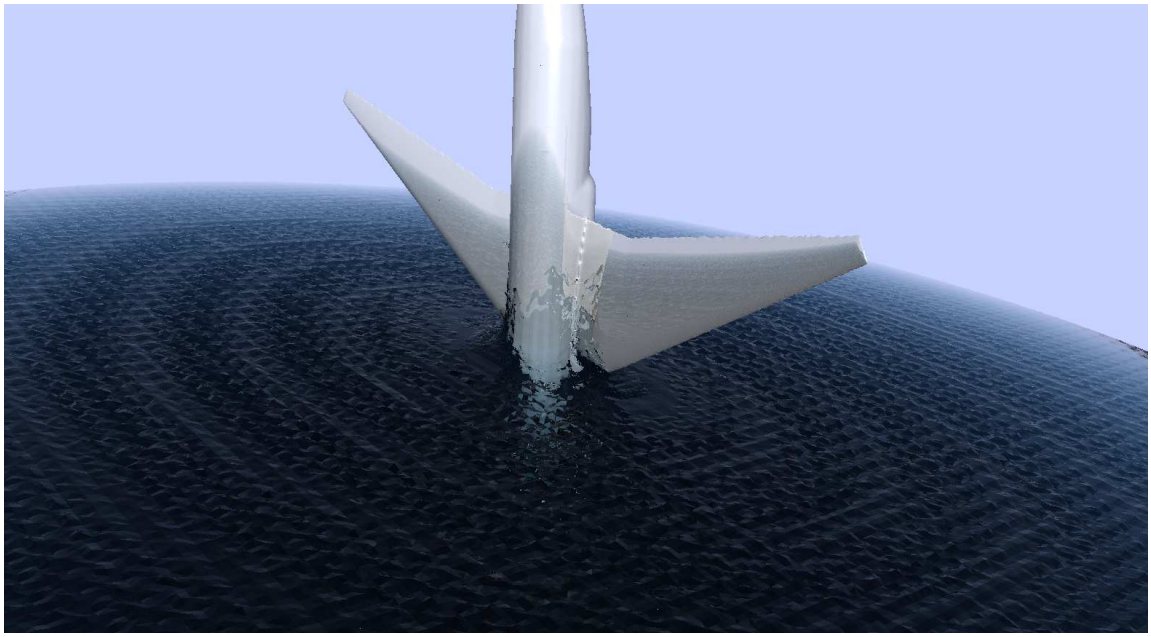
There is an incredible complete video recording of this air disaster. The hijacked wide-body Boeing 767-260ER jetliner flew and rolled into the ocean with the left wing clipping water and getting torn off first. Immediately afterwards, the same happened to the right wing. The fuselage went into cartwheeling and broke up. Only 50 of the 175 crew and passengers survived.

Debris were spreading over a wide area and the light ones could have floated for a long time.

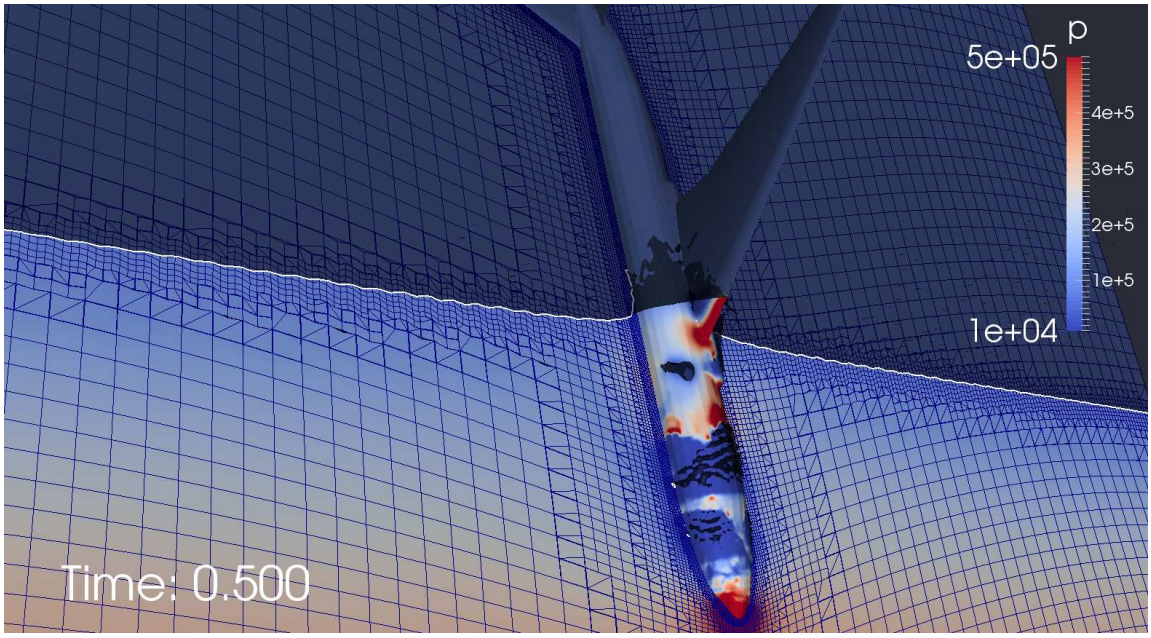
Box 4: A rolling water-entry case: hijacked Ethiopia Airlines flight 961 ditching by Comoros Island, Africa in 1996

Case 5: pitch angle = $-3°$ with *roll angle* = $20°$, angle of approach = $3°$

See Figure 3.14 and its animation. Here, with a 20 degree roll, the left wing of the plane enters the water first. Almost inevitably, this would cause structural failure of the left wing. Read more in Box 4 about an air disaster on the seaside of Comoros Island, Africa.



(a) nose-dive water entry



(b) pressure distribution and mesh

Figure 3.12: Pitch angle = −90°, angle of approach = 93°. This corresponds to Case 4.

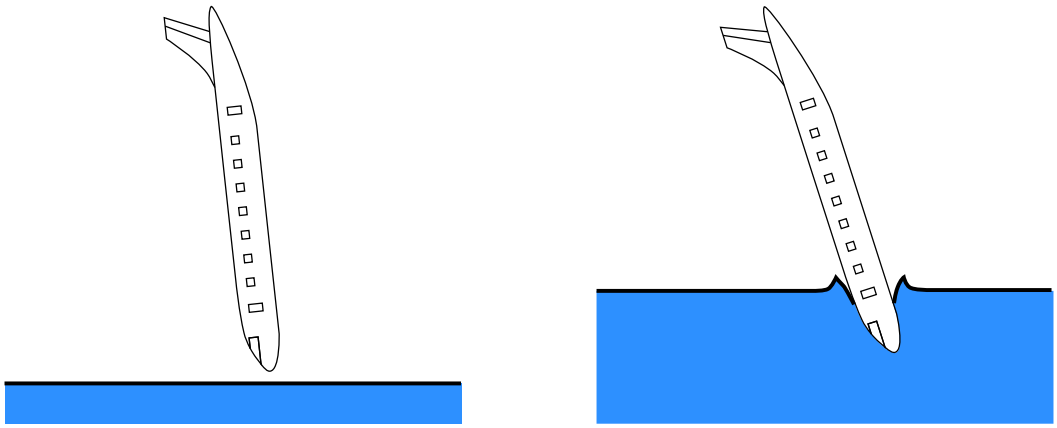
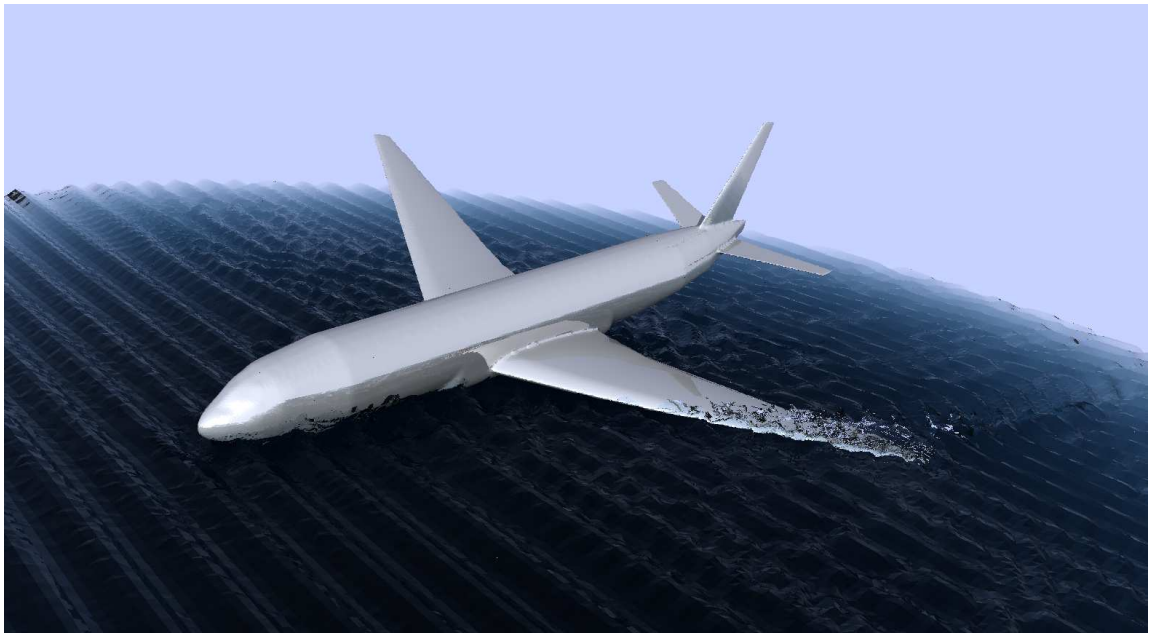
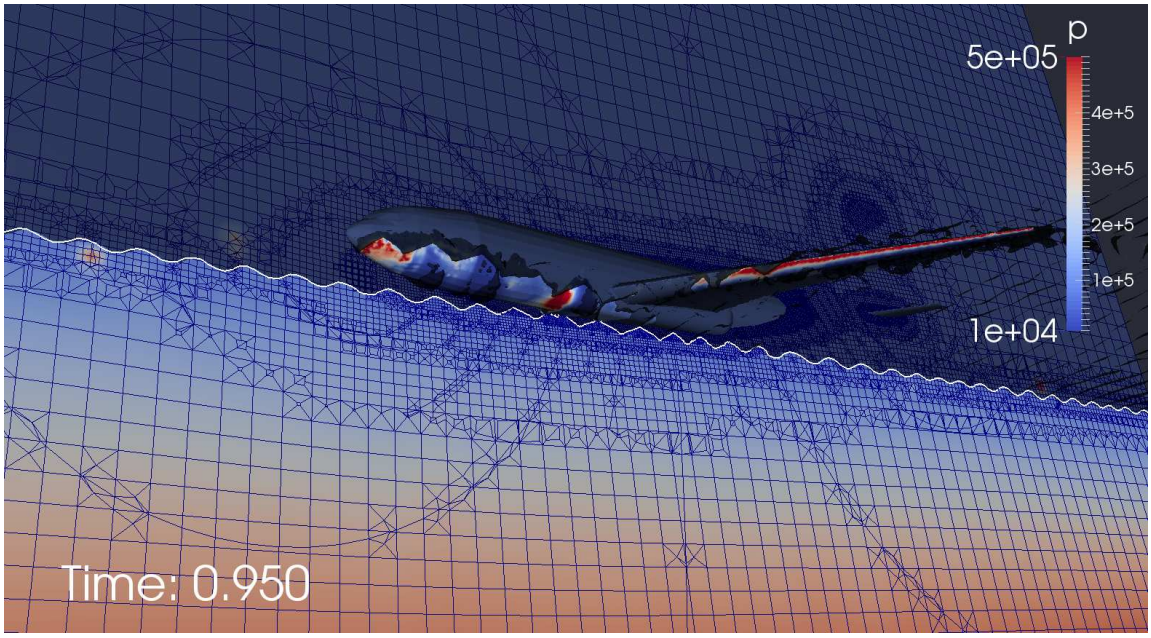


Figure 3.13: Schematics for nose-diving. The ocean current pushes the aircraft to the right, causing it possibly to finish belly up on the ocean floor. This corresponds to Case 4.



(a) rolling water entry



(b) pressure distribution and mesh

Figure 3.14: Pitch angle = −3°, angle of approach = 3°, but with a left-roll angle of 20°. This corresponds to Case 5.

3.3 General Discussion on Damage and Breakup

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."

In addition to the Comoros Island air disaster in Box 4, we further mention another ditching effort, whose outcome was not so fortunate as the US Airways flight 1549. On August 6, 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.

In the numerical simulations provided in the preceding section, we have not included the effects of *rupture and structural disintegration*. But they 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 we have been talking about up to this point.

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 [45, 46] 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*.

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 [46] 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 rupture of the system of stringers and ring frames; cf. Figure 3.15. 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

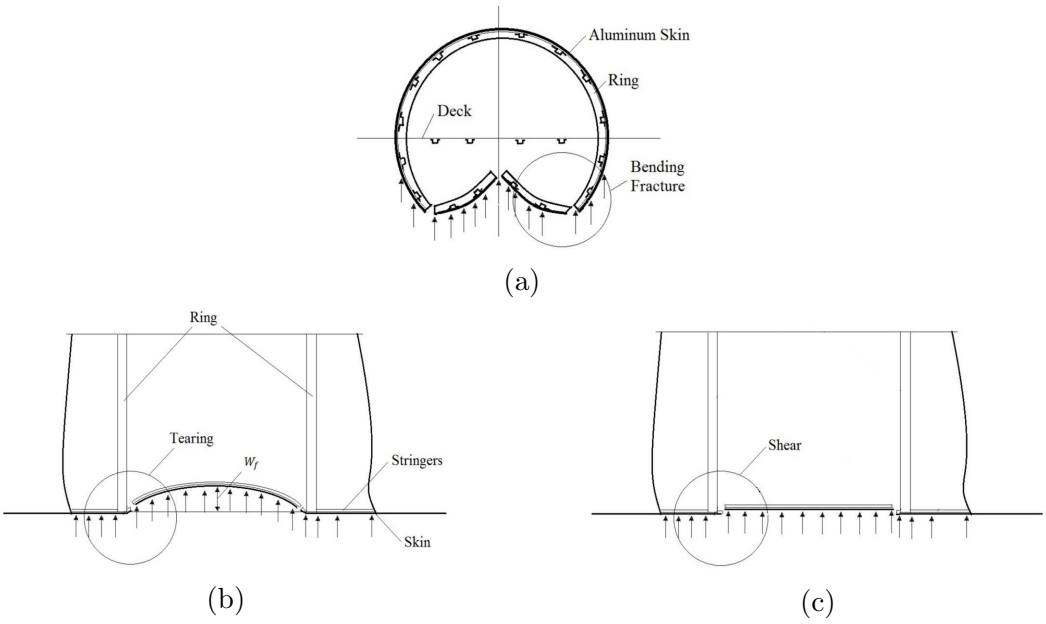


Figure 3.15: Three modes of structural failure for a wide-body airliner: a flexural failure of rings; b tearing fracture; and c shear of the longitudinally stiffened shell. (Adapted from [46, p. 651])

Crash Analysis of Structures) by NASA [15]. These findings were published nearly 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

$$[\sigma] = \rho c[u], \tag{3.1}$$

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 (3.1) one gets the following estimate on V_{cr}:

$$V_{cr} = \frac{\sigma_y}{E} c. \tag{3.2}$$

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 [46].

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

$$\sin \beta = \frac{V_{cr}}{V_0}. \tag{3.3}$$

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 [46], 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" [31, 46] by NASA.

As a consequence of this, it now becomes clear that *the vertical component of the*

terminal water-entry velocity should be reduced as much as possible, such as the glided water-landing approach taken by Captain Sullenberger for US Airways Flight 1549 on the Hudson River. That is, some "pitching attitudes" of the aircraft will have a much higher probability of survival by averting structural damage and decelerations of the occupants [45, p. 34]. Indeed, according to Guo, et al. [21], it is recommended that for a transport aircraft with a low horizontal tail, the pitch angle be chosen between 10° and 12° for safer ditching, which is consistant with the prediction of (3.3). Such knowledge enhances air travel safety, and, as shown here, can be obtained by CFD simulations.

3.4 Beam Analysis of Fuselage

In the previous sections, the water entry process of an aircraft is simulated using CFD techniques. The aircraft is assumed to be a rigid body with free motion in a mixture of air and water. Some general discussions regarding damage and breakup are also given. The current section aims at answering the question whether the aircraft can structurally survive the water entry process described in the simulation.

Ideally, a coupled fluid-structural interaction simulation is required to fully understand the process. However, such a simulation, especially involving rapid fracture and disintegration can be quite challenging. The strategy employed here is an *uncoupled structural analysis*. Data are obtained from CFD simulations to serve as external load in the structural analysis.

Some tradeoffs are made here. A full-blown 3D analysis is avoided in this study, since such an analysis would require a more detailed description of the aircraft structure to be useful. Analysis based on rigid beam theory is used here instead as a simplified model. To this end, beam theory will be described and applied to the fuselage of the aircraft in the following.

3.4.1 Free-Free Rigid Beam

Dynamic failure of free-free beam was studied in [28, 49]. Plastic failure is predicted when the bending moment developed in the beam exceeds a critical threshold. Here, as we just mentioned, the fuselage of the aircraft is modeled as a *rigid beam*. It takes into account the bending moment, lateral displacement and rotary inertia, but no deformation of any kind. The governing equations are, for $x \in [0, L]$,

$$\begin{aligned} \frac{\partial V}{\partial x} + q_z &= \lambda a_z, \\ \frac{\partial N}{\partial x} + q_x &= \lambda a_x, \\ \frac{\partial M}{\partial x} - V + \tau &= \eta\alpha, \end{aligned}$$

where (x, y, z) are body-local axes for roll, pitch and yaw respectively (see Figure 3.16)

- q_z and q_x (N/m) are external force in z and x direction per unit length,
- τ (N·m/m) is external pure torque in y direction per unit length measured at the center of cross section,
- V (N) is internal shear force,
- M (N·m) is internal bending moment,
- N (N) is internal axial (x) force,
- η (kg·m) is sectional moment of inertia in y direction measured at the neutral position,
- λ (kg/m) is linear mass density,

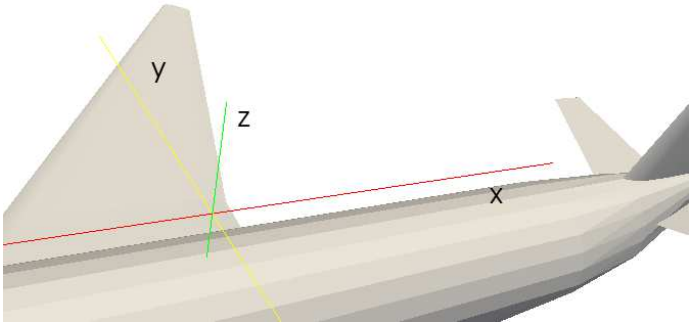


Figure 3.16: Direction of axes.

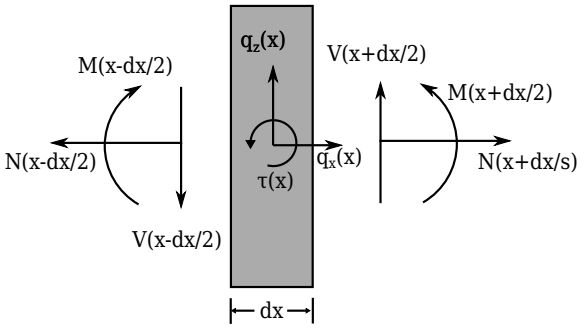


Figure 3.17: Beam element subject to forces and moments.

- a_z and a_x (m/sec^2) are the acceleration in z and x direction respectively,
- α (1/sec^2) is the angular acceleration in y direction.

See Figure 3.17 for an illustration of a beam element. All other motion are ignored.

Free-free boundary conditions, namely zero forcing at both ends, are used as

$$M(0) = M(L) = 0, \qquad N(0) = N(L) = 0, \qquad V(0) = V(L) = 0.$$

Since CFD data is available only at certain snapshots in time, acceleration terms

in the equation need to be evaluated using information from the current time instance only. This is also essential for the satisfaction of boundary conditions at both ends simultaneously, which will be shown in Section 3.4.2.

In general, neutral positions of cross sections should be given. But for estimation purposes, they are aligned to a line parallel to x-axis. The instantaneous global motion is described by the *following rigid body dynamics*,

$$m = \int_0^L \lambda(x_1)\, dx_1, \qquad F_x = \int_0^L q_x(x_1)\, dx_1, \qquad F_z = \int_0^L q_z(x_1)\, dx_1,$$

$$T = \int_0^L \Big[\tau(x_1) - q_z(x_1)(x_1 - x_0)\Big] dx_1, \qquad J = \int_0^L \Big[\lambda(x_1)(x_1 - x_0)^2 + \eta(x_1)\Big] dx_1,$$

$$\alpha = T/J, \qquad a_x(x) = -\omega^2(x - x_0) + F_x/m, \qquad a_z(x) = -\alpha(x - x_0) + F_z/m,$$

where ω is the angular velocity (1/sec) in the y direction.

3.4.2 Data Accumulation and Processing

Values needed for the above calculation are q_z, q_x, τ, λ, η and ω. Angular velocity ω is directly read from the simulation, since it won't interfere with the boundary conditions. Other data input from CFD are the aircraft geometry and the external stress $\boldsymbol{\sigma}$ on the aircraft surface at each snapshot in time. Figure 3.18 shows an example of instant pressure distribution on the geometry.

To perform a computational beam analysis along the x-axis, the aircraft surface is partitioned, equally in x-axis, into n segments S_j, $j = 1, 2, ..., n$. Figure 3.19 shows an example of partitioning into 100 segments. Piecewise constant values are assumed for numerical calculations. In the current implementation, the partition is performed for the whole aircraft surface, including the wings and empennages. We can see from the figure that such a treatment is not ideal, but assumed to be acceptable whenever

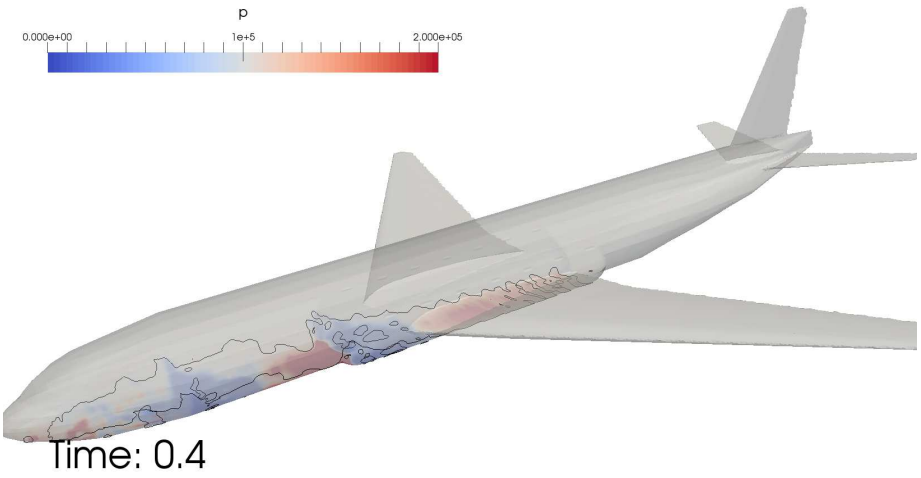


Figure 3.18: Pressure distribution on aircraft surface. Black line is the three-phase contact line.

there isn't a strong load on the wings and empennages.

First, the center in the z direction is calculated by

$$z_j = \frac{1}{|S_j|} \int_{S_j} z \, dS.$$

This z_j serves as the neutral position. Values of z_j are shown in Figure 3.20.

The distribution of mass and rotary inertia are calculated also using the available geometry. Since detailed interior model of the aircraft is not available, for simplicity, we assume mass is distributed to each segment proportional to the surface area $|S_j|$.

$$\lambda_j = \frac{c}{l} |S_j| ,$$

where $l = L/n$, and coefficient c (kg/m^2) is determined by matching a given total mass of the aircraft. As for the sectional moment of inertia, we assume half of the mass is uniformly distributed on the surface, and the other half is located near the

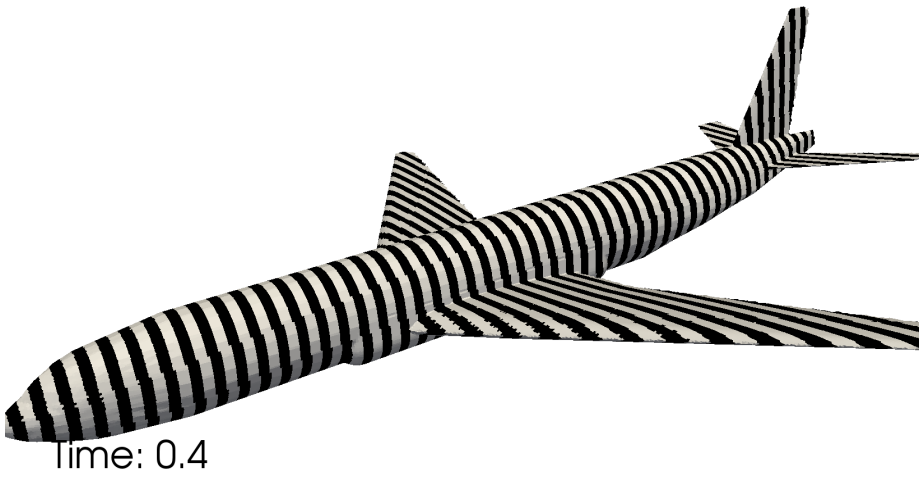


Figure 3.19: Partition of the aircraft surface along the x-axis. Number of segments here is $n = 100$.

neutral position, thus does not have much contribution, yielding

$$\eta_j = \frac{c}{2l} \int_{S_j} (z - z_j)^2 \, dS.$$

Values of λ and η are shown in Figure 3.21 and 3.22

It is assumed that there is a pressure of p_0 in the cabin, therefore, the external load on the beam is calculated as

$$\mathbf{q}_j = \frac{1}{l} \int_{S_j} (\boldsymbol{\sigma} - p_0 \boldsymbol{I}) \hat{\mathbf{n}} \, dS, \qquad \tau_j = \frac{1}{l} \int_{S_j} (z - z_j) \hat{\mathbf{k}} \times (\boldsymbol{\sigma} - p_0 \boldsymbol{I}) \hat{\mathbf{n}} \, dS.$$

$\mathbf{q}_j$ is then projected to x and z directions as $q_{x,j}$ and $q_{z,j}$ respectively. Figures 3.23–3.25 show the distributions of external force load q_x, q_z and torque τ for the example given in Figure 3.18. Data is smoothed out a little and peak values are reduced if there is a smaller number of segments n. Relative magnitude and direction of the load along aircraft body is illustrated in Figure 3.26, to be compared with Figure

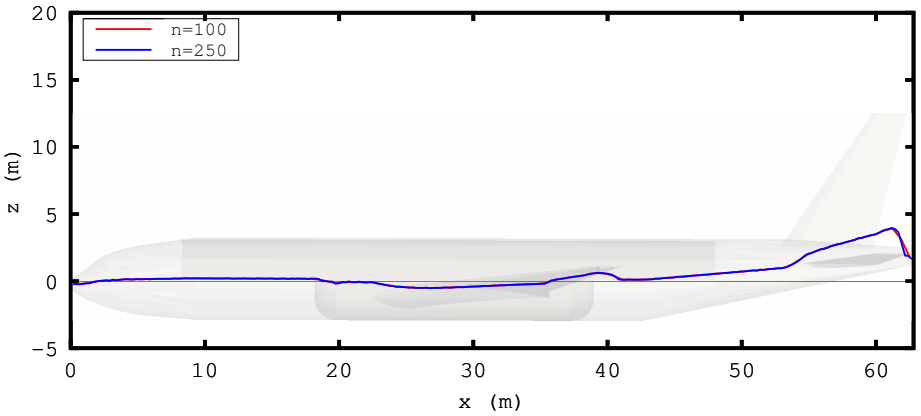


Figure 3.20: Center in z direction along the aircraft body.

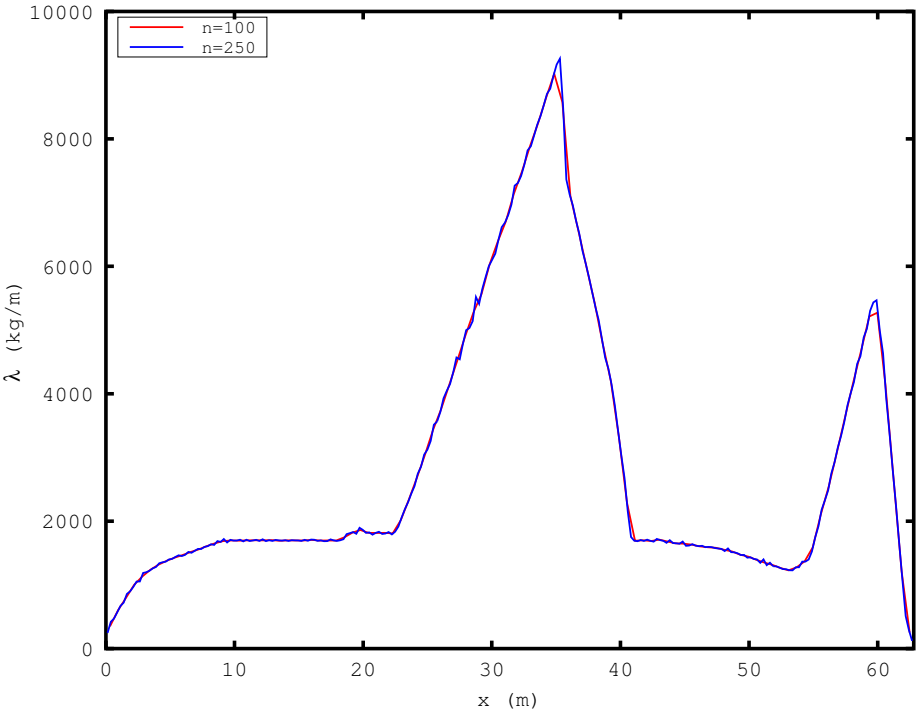


Figure 3.21: Distribution of mass λ along the aircraft body.

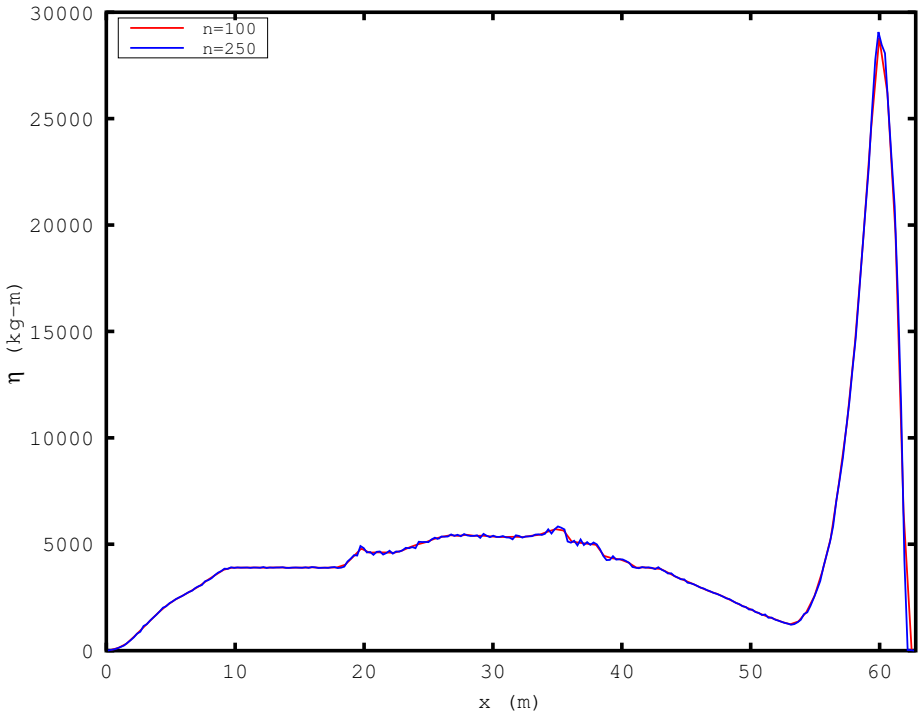


Figure 3.22: Distribution of rotary inertia η along the aircraft body.

3.18.

Note that the vertical center z_j is only considered in this subsection. As mentioned in Section 3.4.1, neutral positions are artificially aligned to a line parallel to x-axis in order to simplify calculation.

Integration of the beam equations with boundary condition at $x = 0$ gives

$$V(x) = \int_0^x \Big[-q_z(x_1) + \lambda(x_1)a_z(x_1) \Big] dx_1,$$

$$N(x) = \int_0^x \Big[-q_x(x_1) + \lambda(x_1)a_x(x_1) \Big] dx_1,$$

$$M(x) = \int_0^x \Big[V(x_1) - \tau(x_1) + \alpha\eta(x_1) \Big] dx_1.$$

We need to make sure the boundary condition is also satisfied at the other end

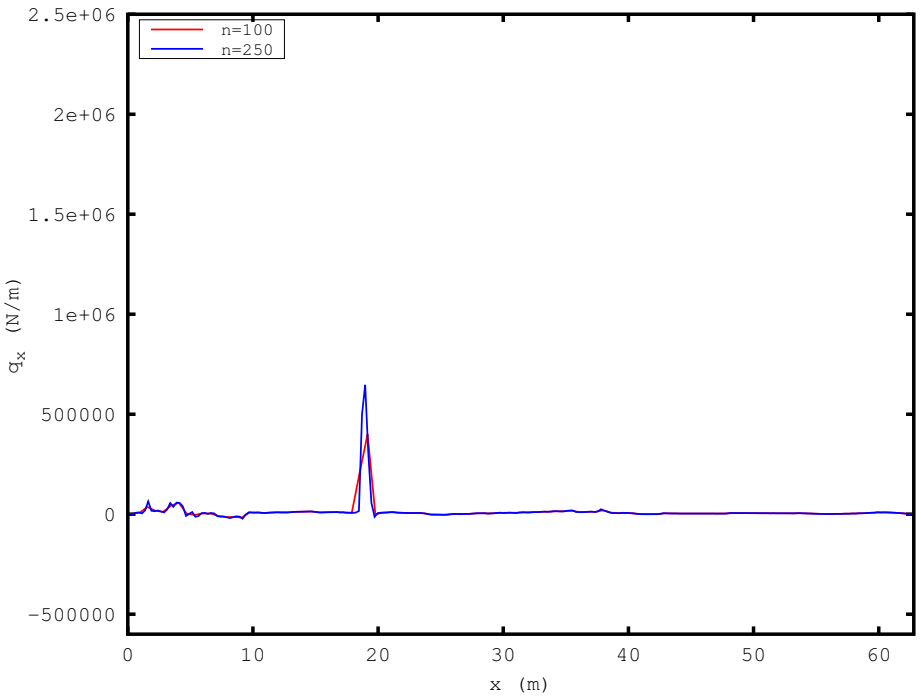


Figure 3.23: Distribution of external load q_x along the aircraft body for the example given in Figure 3.18.

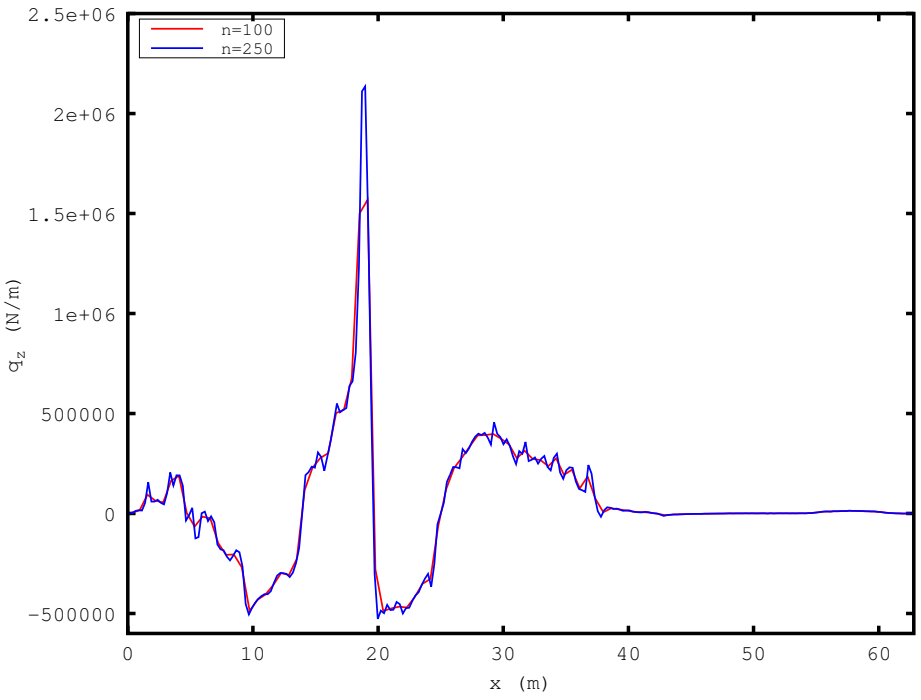


Figure 3.24: Distribution of external load q_z along the aircraft body for the example given in Figure 3.18.

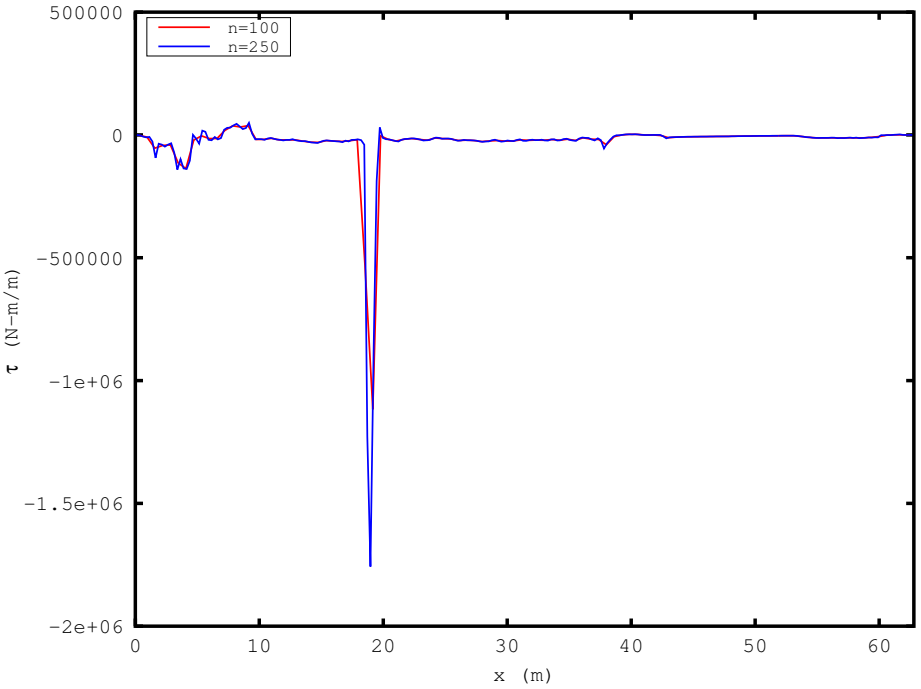


Figure 3.25: Distribution of external pure torque τ along the aircraft body for the example given in Figure 3.18.

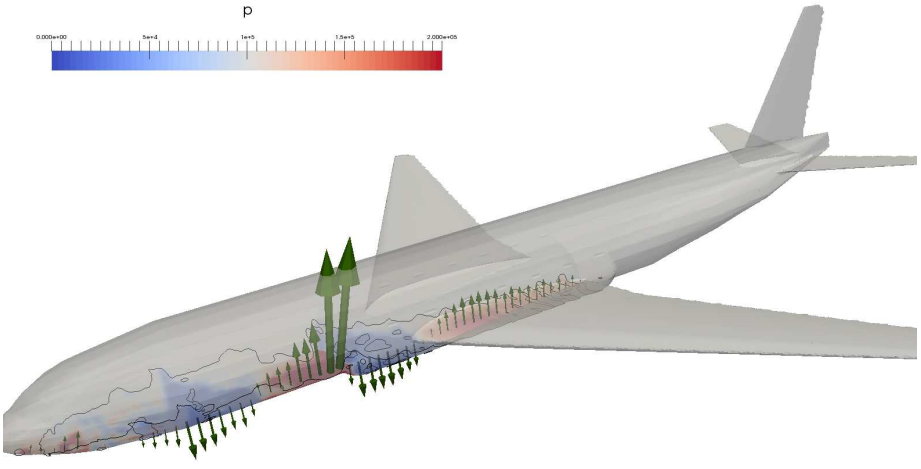


Figure 3.26: Relative magnitude and direction of the external load obtained in data processing is added as vector arrows to Figure 3.18.

$x = L$. In fact, given all the assumptions made above, it is straight-forward to check $V(L) = 0$ and $N(L) = 0$. And for $M(L)$,

$$
\begin{aligned}
&M(L)\\
=&\int_0^L V(x_2)\,dx_2 - \int_0^L \tau(x_1)\,dx_1 + \alpha\int_0^L \eta(x_1)\,dx_1\\
=&\int_0^L \int_0^{x_2} \Big[-q_z(x_1) + \lambda(x_1)a_z(x_1)\Big]\,dx_1\,dx_2 - \int_0^L \tau(x_1)\,dx_1 + \alpha\int_0^L \eta(x_1)\,dx_1\\
=&\int_0^L \Big[(L-x_0)-(x_1-x_0)\Big]\Big[-q_z(x_1) + \lambda(x_1)a_z(x_1)\Big]\,dx_1 - \int_0^L \tau(x_1)\,dx_1 + \alpha\int_0^L \eta(x_1)\,dx_1\\
=&(L-x_0)\int_0^L \Big[-q_z(x_1) + \lambda(x_1)\Big(-\alpha(x_1-x_0) + F_z/m\Big)\Big]\,dx_1\\
&-\int_0^L (x_1-x_0)\Big[-q_z(x_1) + \lambda(x_1)\Big(-\alpha(x_1-x_0) + F_z/m\Big)\Big]\,dx_1\\
&-\int_0^L \tau(x_1)\,dx_1 + \alpha\int_0^L \eta(x_1)\,dx_1\\
=&(L-x_0)\left[-\int_0^L q_z(x_1)\,dx_1 - \alpha\int_0^L \lambda(x_1)(x_1-x_0)\,dx_1 + (F_z/m)\int_0^L \lambda(x_1)\,dx_1\right]\\
&-\int_0^L \Big[\tau(x_1) - q_z(x_1)(x_1-x_0)\Big]\,dx_1 + \alpha\int_0^L \Big[\lambda(x_1)(x_1-x_0)^2 + \eta(x_1)\Big]\,dx_1\\
&-(F_z/m)\cdot\int_0^L \lambda(x_1)(x_1-x_0)\,dx_1\\
=&(L-x_0)\Big[-F_z - 0 + (F_z/m)\cdot m\Big] - T + \alpha J - 0\\
=&0.
\end{aligned}
$$

Figures 3.27-3.29 show the strength of internal forces and bending moment for the example given in Figure 3.18. The comparison between $n = 100$ and $n = 250$ reveals that results do not depend much on the choice of n if n is as large as 100.

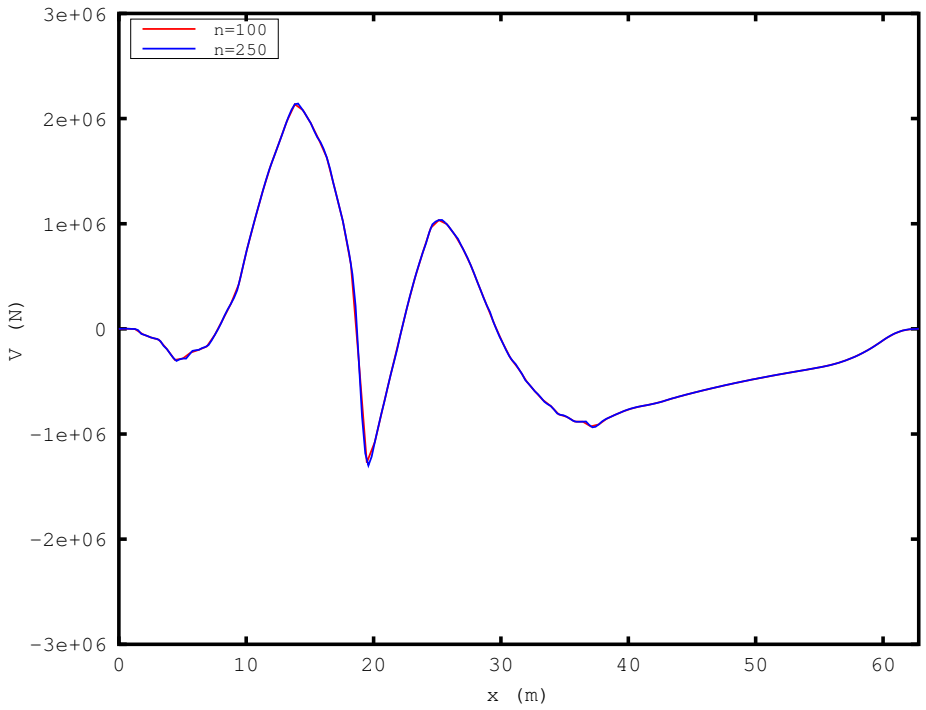


Figure 3.27: Internal shear force V along the aircraft body.

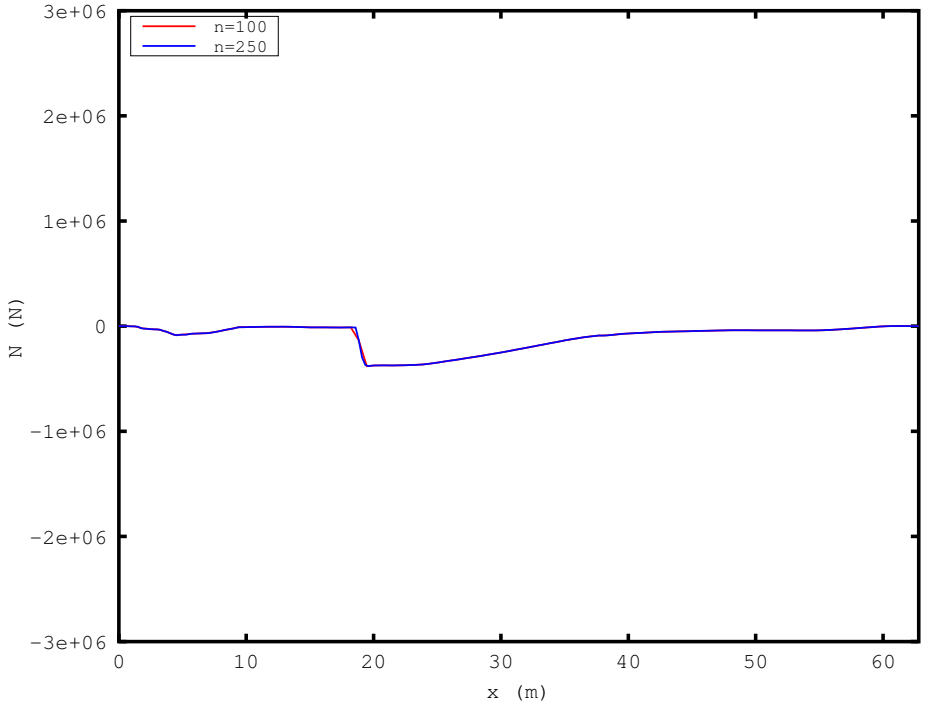


Figure 3.28: Internal axial force N along the aircraft body.

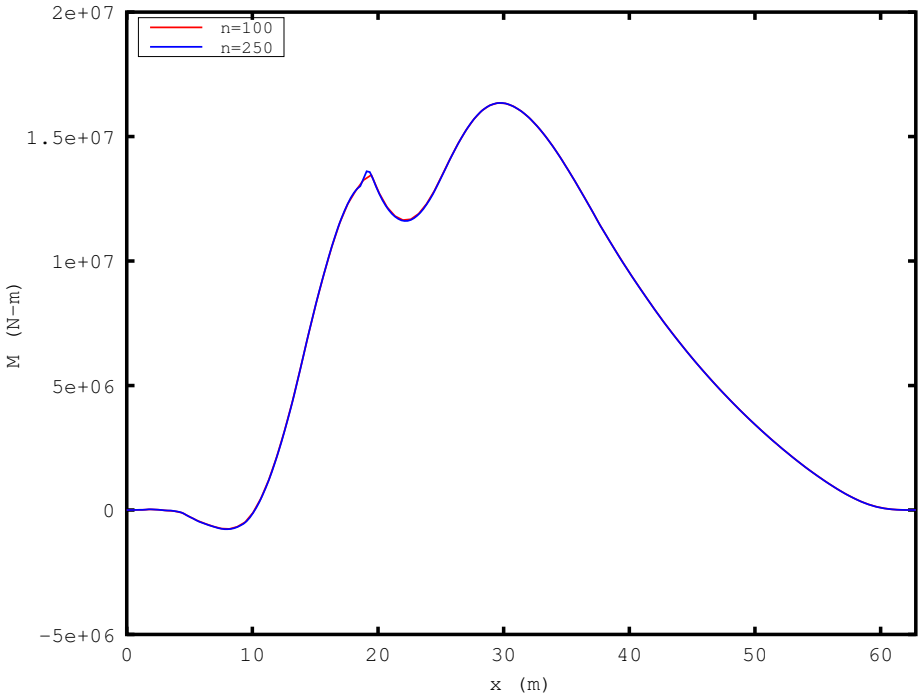


Figure 3.29: Internal bending moment M along the aircraft body

3.4.3 Prediction of Failure: Bending of Cylindrical Shell

Prediction of failure can be done by comparing values calculated with M, V and N from Section 3.4.2 with critical values.

Buckling caused by excessive compressive stress is considered here. As proposed in [42], critical compressive stress for a cylindrical shell before buckling failure is given by

$$\sigma_{\text{cr}} = \frac{Et_{\text{eq}}}{R\sqrt{3(1-\nu^2)}},$$

where E is the Young's modulus, t_{eq} is shell wall thickness, R is radius of the cylinder and ν is Poisson's ratio. The maximum compressive stress induced by M and N is

given by

$$\sigma_{\max} = \frac{M}{\pi R^2 t_{\text{eq}}} - \frac{N}{2\pi R t_{\text{eq}}}.$$

Therefore, we compare the effective bending moment

$$M' = M - NR/2,$$

with the critical value

$$M_{\text{cr}} = \frac{\pi E R t_{\text{eq}}^2}{\sqrt{3(1-\nu^2)}}.$$

This is regarded as a good estimation for short cylinders. Using values from Table 3.3, we get $M_{\text{cr}} = 4.35 \times 10^7$ N·m.

Stringers and rings are not explicitly considered. But it is possible to choose an appropriate effective thickness t_{eq} that provides comparable strength.

Now we find the maximum of effective bending moment M' over the whole fuselage for each time snapshot t, getting $M'_{\max}(t)$. They are then plotted versus t in Figures 3.30–3.33 in comparison with the critical bending moment M_{cr}. Here are some interpretations for those four scenarios.

- **Case 1** with 8° pitch angle (Figure 3.30). Plane is generally safe from structural failure. The process is also known as ditching. Large temporary bending moment can be observed if ditching on a wavy sea and when the speed of the aircraft is still high, for example, at around $t = 0.7$ s. See Figure 3.34.

- **Case 2** with −3° pitch angle (Figure 3.31). The plane might recover to the ditching posture. However, it have to overcome a period of large bending

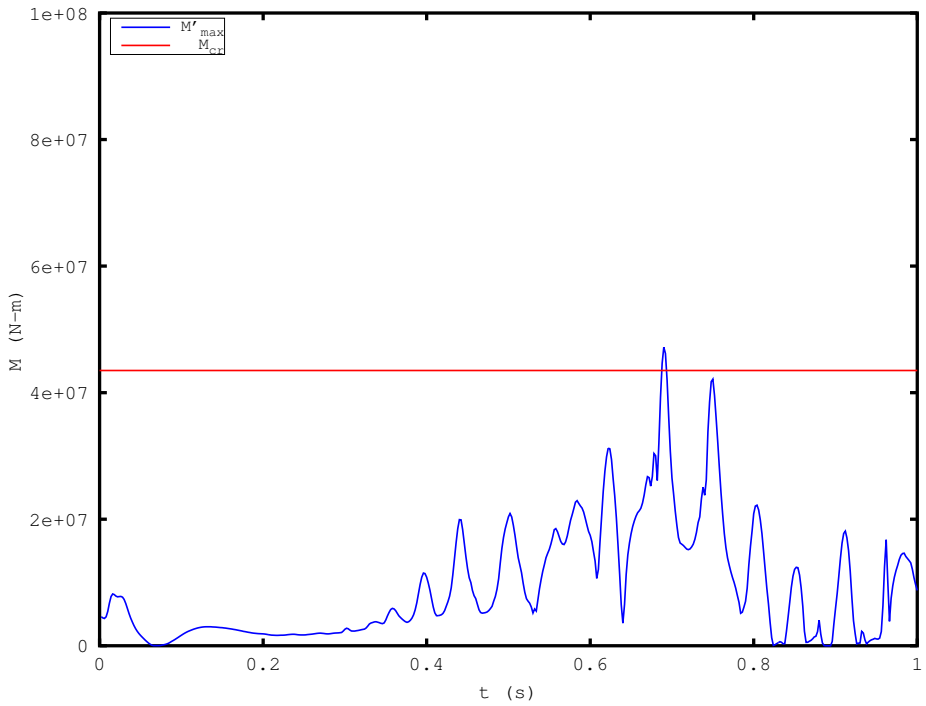


Figure 3.30: Maximum effective bending moment $M'_{\max}$ for Case 1 with 8° pitch angle.

moment when the middle or tail parts of the fuselage hit water, for example, at around $t = 0.78$ s. See Figure 3.35.

- **Case 3** with −30° pitch angle (Figure 3.32). The plane is subject to large axial compression and asymmetric external load, for example, starting from $t = 0.4$ s. Therefore the aircraft is most likely to suffer global failure. See Figure 3.36.

- **Case 4** with −90° pitch angle (Figure 3.33). The plane is subject to axial compression, but not much bending due to the symmetric external load. This lasts until wings reach the water, which is not simulated. See Figure 3.37.

The color of Figures 3.34–3.37 is the estimation of axial compression reconstructed

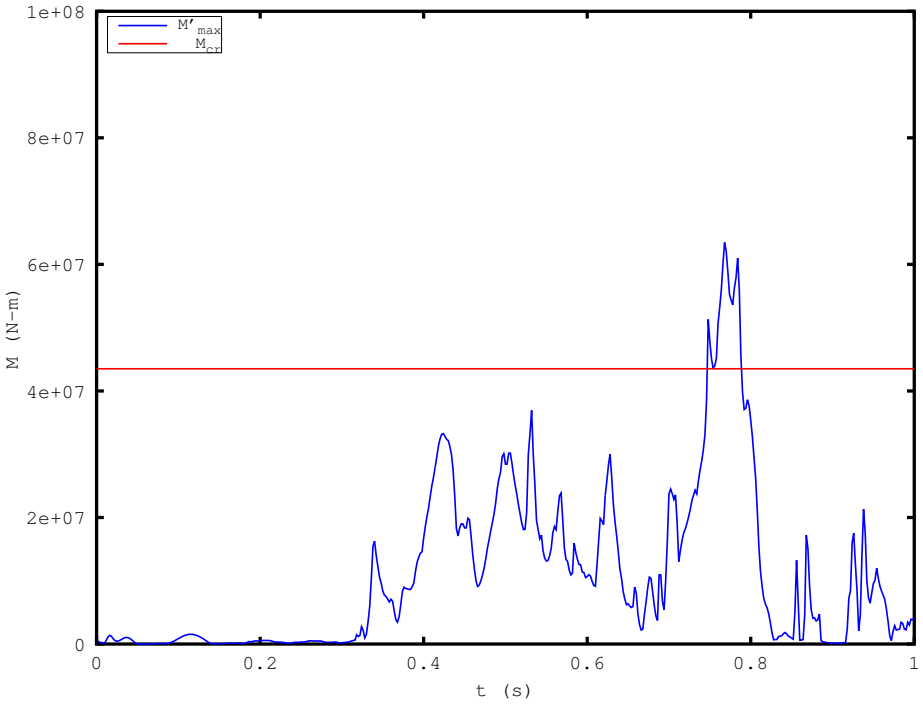


Figure 3.31: Maximum effective bending moment $M'_{\max}$ for Case 2 with $-3°$ pitch angle.

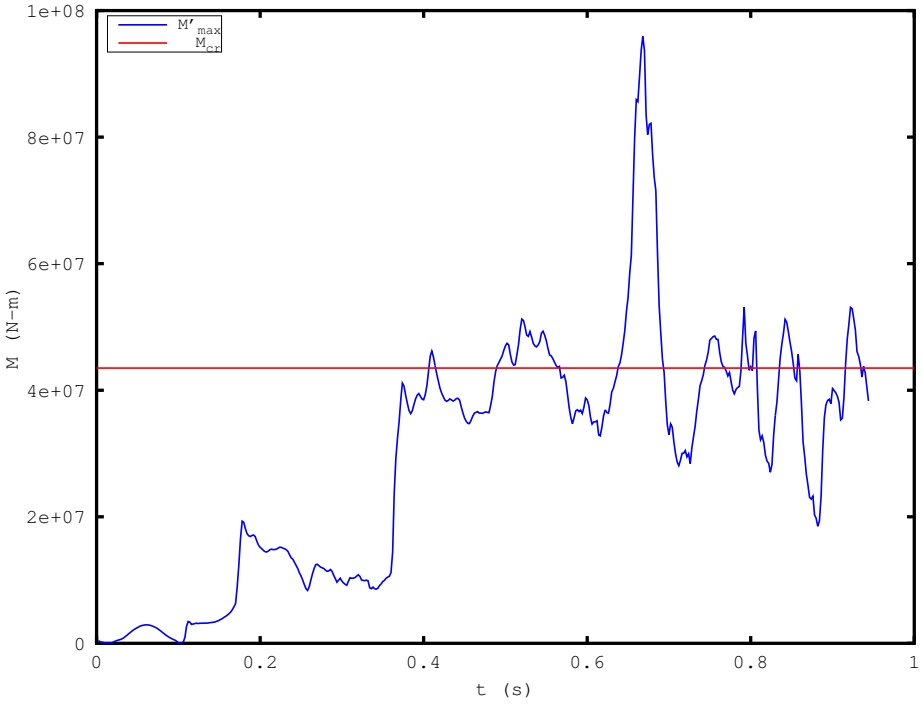


Figure 3.32: Maximum effective bending moment $M'_{\max}$ for Case 3 with $-30°$ pitch angle.

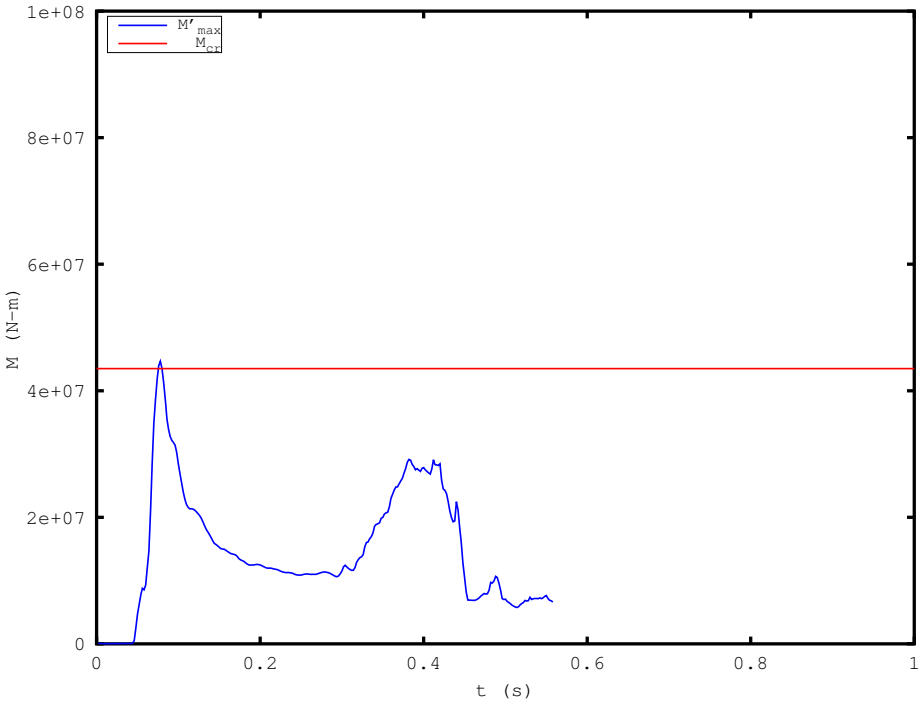


Figure 3.33: Maximum effective bending moment $M'_{\max}$ for Case 4 with $-90°$ pitch angle.

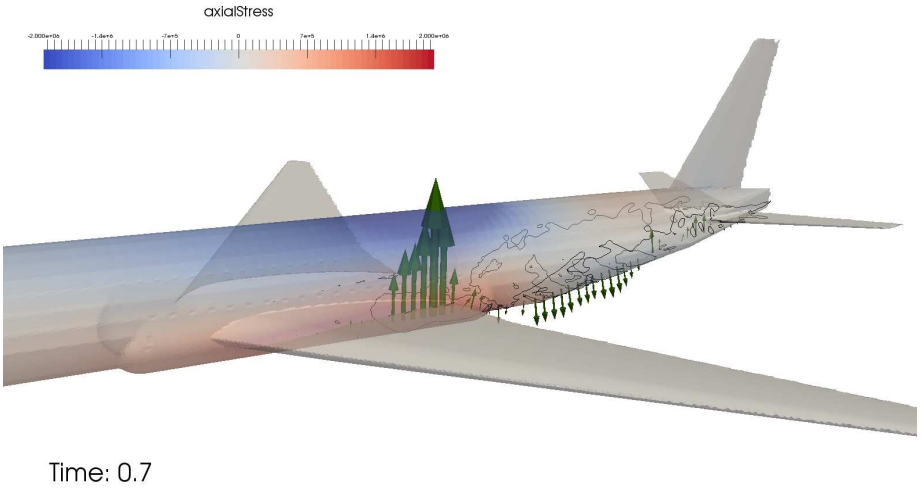


Figure 3.34: External load and axial stress for Case 1 with 8° pitch angle at $t = 0.7$ s.

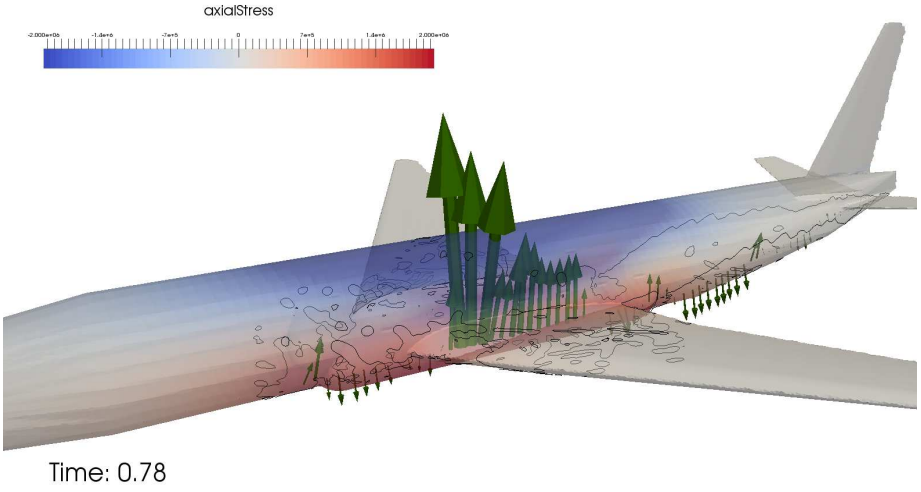


Figure 3.35: External load and axial stress for Case 2 with −3° pitch angle at $t = 0.78$ s.

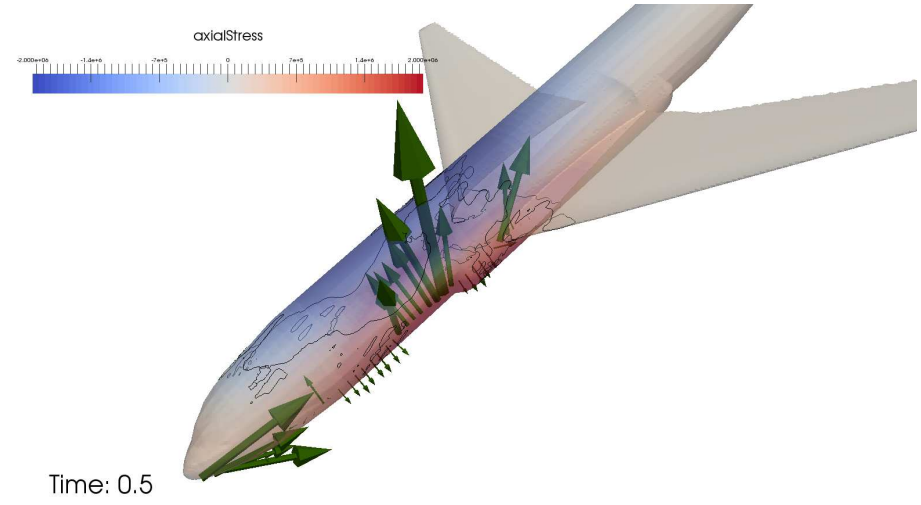


Figure 3.36: External load and axial stress for Case 3 with −30° pitch angle at $t = 0.5$ s.

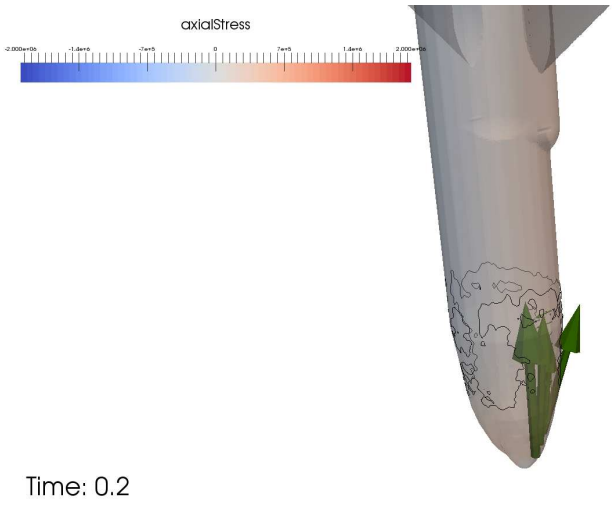


Figure 3.37: External load and axial stress for Case 4 with −90° pitch angle at $t = 0.2$ s.

Geometry	
Radius of cylinder	$R = 3.1$ m
Equivalent thickness	$t_{eq} = 1$ cm
Material (Aluminum 2024-T351)	
Poisson's ratio	$\nu = 0.33$
Young's modulus	$E = 73.1$ GPa

Table 3.3: Parameters for critical bending moment calculation.

with M, N and the aircraft geometry as

$$-\frac{N}{2\pi r} - \frac{M(z - z_j)}{\pi r^2}$$

for every point on aircraft skin, where r is its distance to the center of cross section.

3.5 Conclusion

After all the computation and analysis, it's time matching the results to reality and trying to answer the question what happened in the final moments of the flight MH370. The current situation is that, despite search efforts going on for over a year, people only found one flaperon. The fact that no floating debris field have ever been found strongly suggest that the fuselage didn't break up. This is because global break-up of the fuselage will inevitably expose light materials inside, such as cushions and luggages, which could float around for a extended period to be eventually found. This fact really limits the possibilities to **Case 1** and **Case 4**, namely, the following two scenarios:

- The airliner successfully ditched as in **Case 1**. Although no global break-up happened, there was inevitable local damages. With water flowing into the cabins, the aircraft shortly sunk almost as a whole, while the passengers and crew members were either unconscious or unable to get out in time.

- The pilot (or "auto pilot") still tried to pull up when there wasn't enough fuel and speed. This lead to a nose-dive. In such low speed diving process, some hanging metal parts, such as wings and engines, were broken off while the structural integrity of the fuselage was maintained.

The fact that a flaperon of the aircraft has been found further suggests that the second possibility is actually more likely, because a sccessful ditching might not break the flaperon off the aircraft. Of course, we haven't "proved" these are what actually happened in the final moments of flight MH370. However, these possible scenarios surely provide us with more insights into this matter.

The crash of an airliner into ocean is a profoundly tragic event. But on the

mathematical and engineering side, there should be significant interest in its modeling and computation so that one can understand the physical mechanisms better in the hope of improving aircraft crashworthiness and survivability. The CFD approach is advantageous in saving long and expensive processes of laboratory setup and measurements. However, many challenges remain. Regarding CFD for the study of aircraft ditching in water, see an excellent review and outlook paper in Liu et al. [29]. For an analysis-minded mathematician, it would be nice to formulate a list of problems dealing with the *rigor* of generality of approach, robustness and stability issues, which are being considered.

On any given day, there are now hundreds of thousands of people traveling by air worldwide. Air travel has never been safer and continues to become even safer. According to Barnett [3] in the 2000-2007 time period the death risk per flight on a First-World airliner was 1 in every 2 million: and 2 million days is nearly 5,500 years! There are always bound to be unfortunate and tragic incidents. However, it is to be expected that data generated by numerical simulations will further improve passenger survival in emergency water landings.

4. INTERACTING WIND TURBINE FLOWS

In this section, we treat a different computational mechanics problem, that of wind energy and wind turbine flows.

Wind energy is a major form of renewable energy. It is being actively developed by many nations in the world. It reduces carbon footprints and the potential harm of anthropogenic warming of the atmosphere. According to the data reported by The Global Wind Energy Council[1] in 2013, 44.7 Gigawatts of new wind power was added to worldwide capacity in 2012, representing a 19% increase over the preceding year. In comparison with solar energy, the cost for wind energy per kilowatt is still lower. The primary disadvantage of wind energy is due to its intermittency, which is now being mitigated by the development of electric-power restoring devices and other smart grid designs.

Wind power research and development (R&D) involves wide-ranging interdisciplinary topics and pragmatic tasks. In this section, our main interest will focus on the computational fluid dynamics (CFD) intensive portions of the work involved. These portions constitute key technological issues in regards to the flow patterns and interactions, effects of airfoil (shape) design of turbine rotor blades, electro-mechanical power-generating units, generalization and extension to other similar applications such as ocean currents power generation, etc. These are challenging, fundamental issues of high technological as well as of intellectual interest.

Even though there is already a large amount of literature on wind energy's R&D, one of the fundamental problems appears to be under-studied: *high-fidelity blade resolved CFD* for wind turbine flow calculations. For example, highly valuable in-

[1]The Global Wind Energy Concil. `http://www.gwec.net`.

sights regarding wind turbine flows have been derived by using an "actuator line" method (cf., e.g., Shen et al. [39]). Such a method is widely regarded now as being "standard", however it is not what we regard as high-fidelity blade resolved CFD calculation. There is a fundamental reason commonly understood for why such high-fidelity blade resolved CFD calculations are *avoided*: the lack of computational power and resources for such on even the most advanced supercomputers in the world as the computations are very large scale. Nevertheless, we see this approach (of high-fidelity blade resolved CFD) eventually as most natural and inevitable, and possibly even advantageous especially in view of the rapid increase of computational power in succeeding generations of supercomputers.

There are now numerous software applications available for CFD calculations. The major toolkit we have adopted here is *OpenFOAM*[2]. OpenFOAM is *free and open-source* software currently used by thousands of engineers and researchers worldwide. It uses the *finite volume method* with object oriented C++ programming and vector-tensor field operations with several key advantageous features; see a recent introductory article on OpenFOAM by Chen et al. in [9].

Our section is organized as follows: Subsection 4.1 studies the *preprocessing* of CFD by OpenFOAM, mainly mesh generation and choice of parameters; Section 4.2 describes *problem solving and running of the codes*, especially regarding the turbulence modeling aspects and interacting turbine flows; Subsection 4.3 deals with *postprocessing* of computed CFD data output, in the form of snapshots; Subsection 4.5 shows OpenFOAM calculations of turbines operating in a two-phase flow. Subsection 4.4 uses the POD (proper orthogonal decomposition) to show the most prominent modes; Subsection 4.6 provides brief concluding remarks.

[2]OPENFOAM® is a registered trade mark of OpenCFD Limited, the producer of the OpenFOAM software.

4.1 Problem Setup and Mesh Generation

4.1.1 Use of a Dynamic Mesh

The rotational, variable-speed motion of the turbine blades' motion is a major issue, requiring a *dynamic mesh*. During the past several years, free and open-source computer programs, which can incorporate relative motions of solid boundaries present in fluid dynamics, were developed by individuals and groups. Libraries with the capability of dynamic meshes [26] are available in OpenFOAM:

(i) `dynamicRefineFvMesh`: This can refine or coarsen the mesh on demand.

(ii) `dynamicMotionSolverFvMesh`: This solves for the motion of the mesh through diffusivity equations.

(iii) `(multi)SolidBodyMotionFvMesh`: This can move the mesh or parts of the mesh as a solid body.

(iv) `topoChangerFvMesh`: This contains mesh modifying models involving topological changes such as attach, detach, automatic layer addition and removal.

Here, we summarize OpenFOAM's capability in handling *prescribed* rigid body boundary motion: in particular, the rotating motion inside the flow. Fixed motion parameters, such as the center, axis and angular velocity, are defined in the file named `dynamicMeshDict`. See Listing 4.1 for the codes.

The technique of *sliding mesh interface* is used to deal with rotating structures inside the flow. In this way, part of the mesh is rotated, but not deformed in any way. Computationally, the matching or interpolation between the rotating mesh and the static mesh is done essentially *seamlessly.* In OpenFOAM, the current way of implementation of a sliding interface is called *arbitrary mesh interface (AMI).* It was

```
dynamicFvMesh    multiSolidBodyMotionFvMesh;
motionSolverLibs ("libfvMotionSolvers.so");
multiSolidBodyMotionFvMeshCoeffs
{
    cylinder1
    {
        solidBodyMotionFunction  rotatingMotion;
        rotatingMotionCoeffs
        {
            origin      (0 0 0);
            axis        (0 1 0);
            omega       1.58; // rad/s = 15 RPM
        }
    }
}
```

Listing 4.1: `<case>/constant/dynamicMeshDict`. This file shows the codes for specifying fixed-speed rotation in OpenFOAM computation. The rotating mesh region is called `cylinder1` and the type of motion is `rotatingMotion`. Required parameters are origin, axis and angular velocity (omega). (Here, omega is chosen as 15 RPM, for example.)

newly introduced in version 2.1, and works well in parallel computing. A comparable implementation in *foam-extend*[3] is called *general grid interpolation (GGI)* [5]. Figure 4.1 shows an example of a mesh with sliding interfaces (used by us in OpenFOAM execution). However, the rotating motion (by direct application of OpenFOAM) is *prescribed with a fixed angular velocity* instead of being driven by wind force in run time, and is thus, inappropriate for wind-driven motion.

OpenFOAM also contains a library for coupling force and torque with motion of structures inside the flow. A field named `pointDisplacement` in OpenFOAM is created and reserved for solving and recording movement of each mesh point. The solid boundary structure is regarded as a *patch* (a piece of boundary in the mesh). This type of patch is designated to be a force-driven object. Displacement on the patch is updated first as a result of the specified motion. Internal field of `pointDisplacement` is subsequently solved, usually by a potential (Laplace) equation. The equation

[3] `http://www.extend-project.de/`.

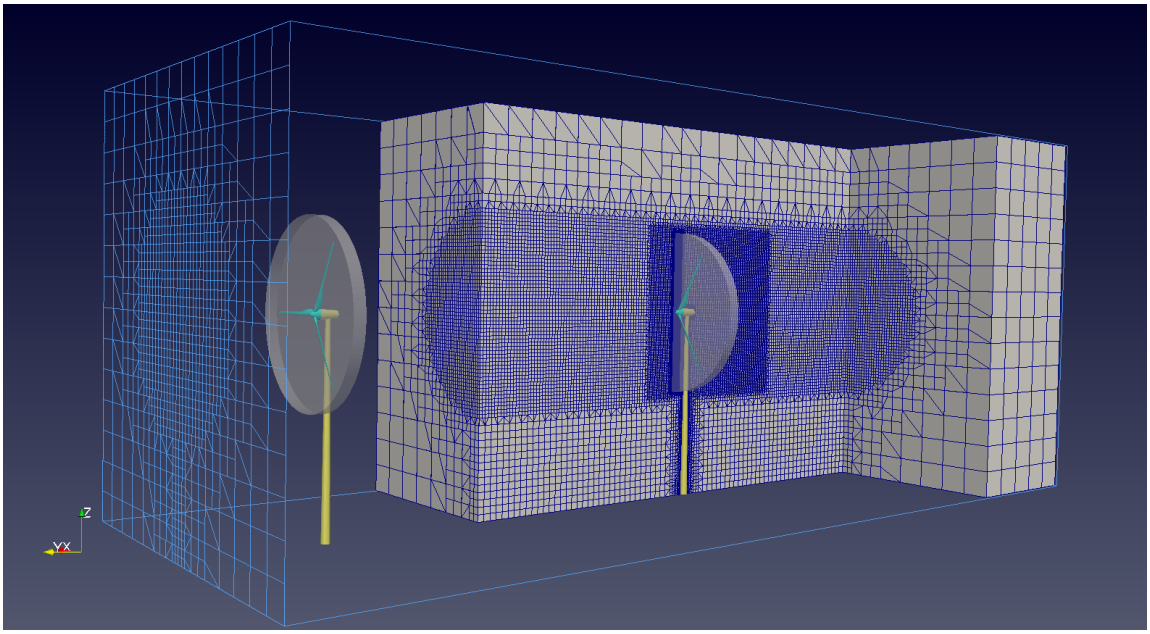


Figure 4.1: Mesh layout for a case of two wind generators. The thin disk surrounding the turbine blades contain the rotating part of the mesh. The surfaces of the disks form a sliding interface to the stationary part of the mesh.

takes the displacement of the structure as its boundary condition. The mesh is modified/updated according to the solved `pointDisplacement` field. (This serves as a concrete example of how `dynamicMotionSolverFvMesh` works.) There are also provisions for applied restraints, such as a spring attached to the objects, and/or other constraints such as a fixed axis of rotation. However, the potential equation-based `pointDisplacement` solver can only handle small mesh displacements. It would not be long before the mesh deteriorates if we use it for continuing rotation, and numerical errors begin to grow.

As the above existing OpenFOAM provisions cannot readily serve our purposes, we need to write and develop our own codes. The coding of OpenFOAM is based on the principle of object oriented programming, and has numerous modules selectable as building blocks. Our adaptation and refinement can be relatively easily integrated

into the C++ class hierarchy of OpenFOAM, general enough to meet our current objectives and possible future extension, while not overly complex.

Since our main concern is rotational motion, we choose sliding interface for dynamic mesh. However, we replace the mesh type `solidBodyMotionFvMesh`, which can only handle *prescribed* solid body motion, by a force-coupled one, named by us as `forceDrivenMotionFvMesh`. Motion functions (types of motion) that can access force and torque data at each time step are also implemented under the C++ based class `forceDrivenMotionFunction`. (See the flow chart in Figure 4.2, where the shaded blocks are new codes for rotating-motion sliding surface dynamic mesh.) For example, `FDRotatingMotion` is the force driven version of the original `rotatingMotion` in OpenFOAM. The codes in Figure 4.2 can now be used to model freely, variable angular speed rotating wind turbines. Please compare the codes in Listing 4.2 with its predecessor Listing 4.1. Also note that `forceDrivenMotionFvMesh` has the flexibility to combine the case of *force driven motion* with that of *prescribed motion* into a *single*, *unified* case.

A remark is in order here: the motion of a body not only depends on the current state of force and torque, but also its current velocity and angular velocity. Their data need to be saved for future reference when computation is continued. To this end, we set up a file to keep track of such data, called `motionState`, that describes whatever state of motion we need. See Listing 4.3 as an example.

In principle, the mesh of any force driven motion of a solid body can be implemented as a C++ subclass of `forceDrivenMotionFunction`. However, the "code stream" feature provides a way to quickly specify and test a motion response. For example, `codedFDRotatingMotion` (within Listing 4.4) enables the user to directly write C++ code to specify how states of motion should be updated. Listing 4.4 shows a typical case. States of motion are updated in exactly the same way as

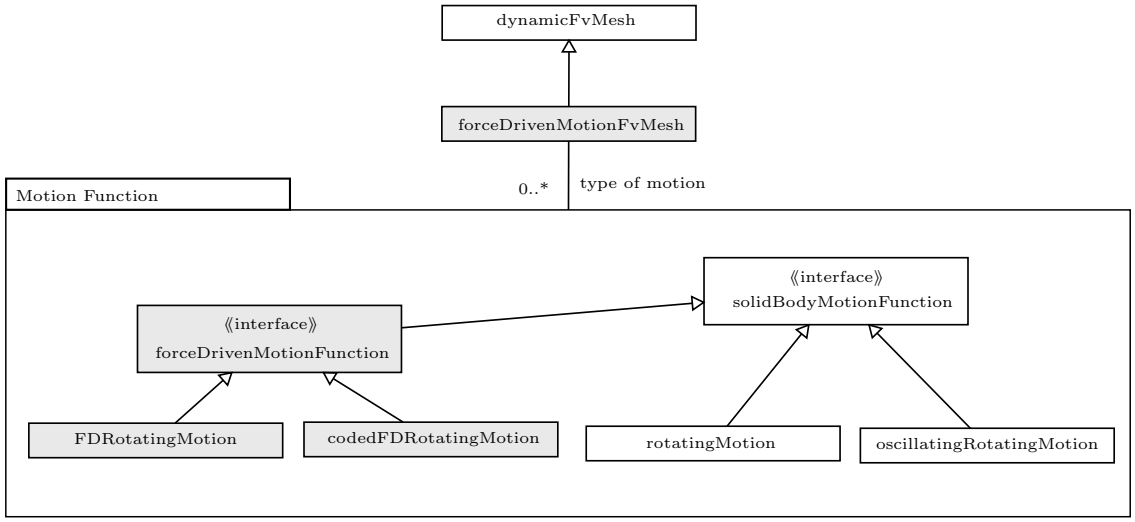


Figure 4.2: C++ class diagram for the implementation of force driven rotation. The shaded blocks represent newly added features. Here *forceDrivenMotionFvMesh* is a force/torque-couplable type of mesh. The mesh can move according to various "motion functions", either force driven or prescribed.

```
dynamicFvMesh    forceDrivenMotionFvMesh;
motionSolverLibs ("libfvMotionSolvers.so"
                  "libforceDrivenMotion.so");
forceDrivenMotionFvMeshCoeffs
{
   cylinder1
   {
       solidBodyMotionFunction  FDRotatingMotion;
       FDRotatingMotionCoeffs
       {
           origin      (0 0 0);
           axis        (0 1 0);
           momentOfInertia 1e6;
       }
       forces
       {
           type forces;
           patches (cylinder1Turbine);
           rhoName rhoInf;
           rhoInf 1.205;
       }
   }
}
```

Listing 4.2: `<case>/constant/dynamicMeshDict`. This file shows the codes for specifying wind driven free rotation for OpenFOAM computation, using our newly adapted codes. Instead of the angular velocity, we specify the moment of inertia and the name of the driven surface, in this case `cylinder1Turbine`.

```
cylinder1
{
    angle                   0;
    angularVelocity         0;
    angularAcceleration     0;
}
```

Listing 4.3: `<case>/0/uniform/motionState`. This file saves data of dynamical motion, such as the angular displacement, velocity and acceleration, in order to keep proper track of the state of motion of a wind driven body.

`FDRotatingMotion`, which models a freely rotating structure driven by the flow, except that the code also increments a *counter* defined therein along the way. *Any control mechanism actions* can be incorporated, too. The numerical integration of the rotational equation of motion $I_R\alpha = \tau_{wind}$, $\alpha = \dot{\omega}$ can be carried out, e.g., in the following way known as the *leap frog scheme*

$$\omega^n = \omega^{n-\frac{1}{2}} + \frac{1}{2}\Delta t^{n-1}\alpha^{n-1}, \tag{4.1}$$

$$\theta^n = \theta^n + \Delta t^n \omega^n, \tag{4.2}$$

$$\alpha^n = \frac{\tau_{wind}}{I_R}, \tag{4.3}$$

$$\omega^{n+\frac{1}{2}} = \omega^n + \frac{1}{2}\Delta t^n \alpha^n \ , \tag{4.4}$$

where in addition to α and ω specified above, θ is the angular position, and Δt is the step size of time marching. The superscript is the index of time steps (including half steps for ω).

4.1.2 Mesh Generation and Case Setup

In the previous subsections, considerable space has been devoted to address the critical needs and coding of dynamic mesh generation in order to develop a capability for handling the variable angular speed of rotational motion to incorporate the wind

```
1  dynamicFvMesh    forceDrivenMotionFvMesh;
2  motionSolverLibs ("libfvMotionSolvers.so"
3                    "libforceDrivenMotion.so");
4  forceDrivenMotionFvMeshCoeffs
5  {
6      cylinder1
7      {
8          solidBodyMotionFunction  codedFDRotatingMotion;
9          codedFDRotatingMotionCoeffs
10         {
11             redirectType  withCounter;
12             origin      (0 0 0);
13             axis        (0 1 0);
14             momentOfInertia 1e6;
15             code
16             #{
17                 // get time step length
18                 scalar deltaT = time_.deltaTValue();
19                 scalar deltaT0 = time_.deltaT0Value();
20
21                 // update states of motion (leap-frog)
22                 omega_ += 0.5 * deltaT0 * alpha_;
23                 theta_ += deltaT * omega_;
24                 alpha_ = (moment & axis_) / momentOfInertia_;
25                 omega_ += 0.5 * deltaT * alpha_;
26
27                 // increment the counter
28                 scalar counter;
29                 motionState_->lookup("counter") >> counter;
30                 counter += 1;
31                 motionState_->set("counter", counter);
32             #}
33         }
34         forces
35         {
36             type forces;
37             patches (cylinder1Turbine);
38             rhoName rhoInf;
39             rhoInf 1.205;
40         }
41     }
42 }
```

Listing 4.4: `<case>/constant/dynamicMeshDict`. This file shows the ability to directly code the equations of motion (4.1)-(4.4) in C++ into an OpenFOAM case. Inserted C++ code (lines 17 to 31) updates states of motion as free rotation. In addition, a counter is incremented every time step to show the ability to couple custom variables as new states of motion.

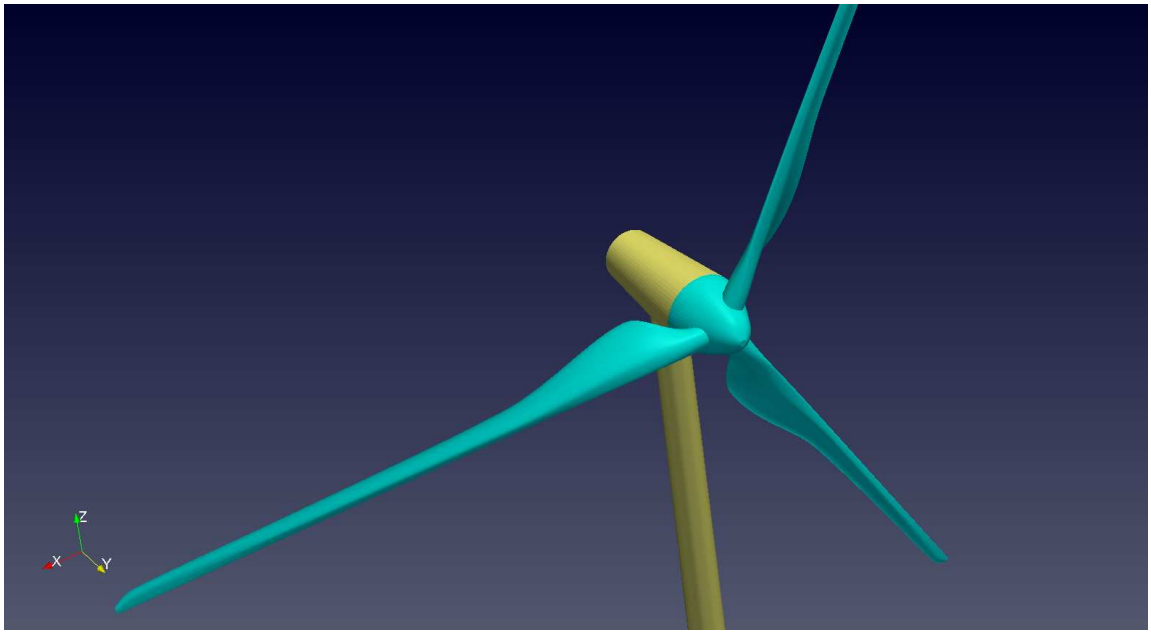
driving force. Now we are ready to carry out the *preprocessing* of the OpenFOAM computation.

Here, we use (standard) OpenFOAM tools, `blockMesh`, `snappyHexMesh` and `mergeMeshes`, to perform mesh generation. These tools are fairly user-friendly. For an experienced user or PhD student, it takes only several days to generate meshes for a domain with certain sophisticated geometry.

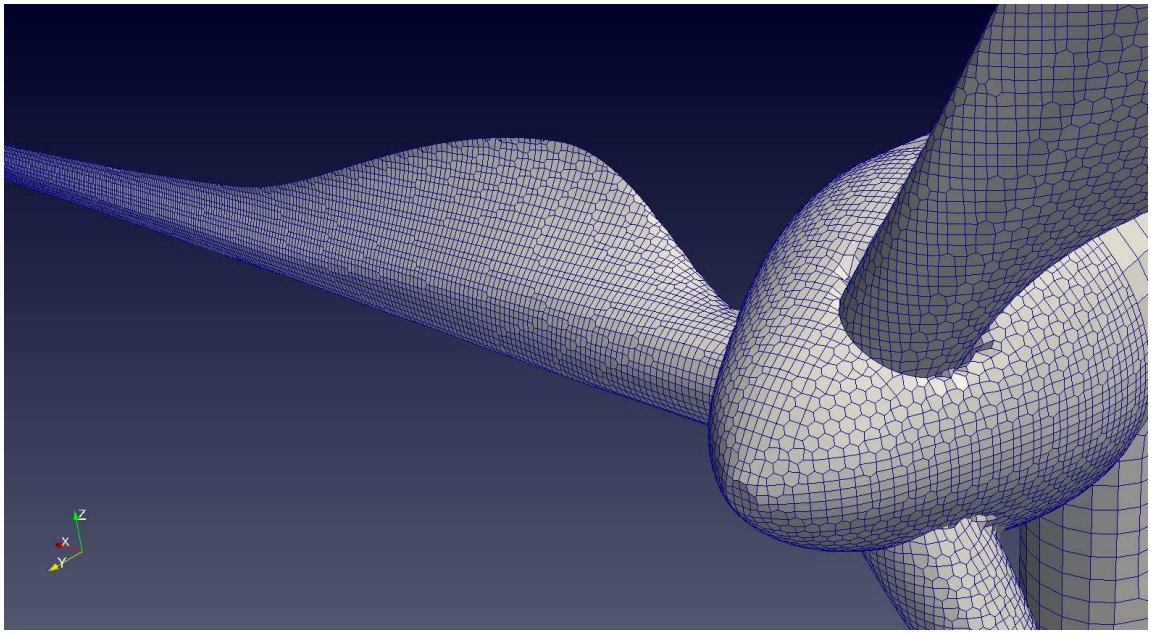
The turbine blades are composed of NREL airfoils numbered S816, S817 and S818 [41]. The radius of the turbine (R) is 50 m. See Figure 4.3. A mesh layout is shown in Figure 4.1. Also see Figure 4.4 for a cross section of mesh. There are approximately 0.65 million cells in the mesh to be used in the computational examples mentioned below. This is not yet a satisfactorily fine resolution, as we are constrained by our supercomputing resources, but we will be able to refine mesh to any degree of desired resolution in the future when more supercomputing resources become available.

We consider two linearly aligned, coaxial wind turbines. They can rotate in the same direction, called by us the *iso-rotating case*, or in the opposite direction, e.g., one is counterclockwise while the other is clockwise. We call the latter the *contra-rotating case*. The distance (D) between the turbines is either 300 m ($D/R = 6$) or 200 m ($D/R = 4$). The wind speed at the inlet is chosen as 10.5 m/s, 13 m/s or 15.5 m/s. See Figure 4.5.

Differences in mesh may affect the computational aerodynamic properties of turbines. In order to conduct meaningful comparisons, mesh around the blades, or the rotating cylinder in particular, is not generated independently but *exactly identical* for the iso-rotating case. The rotating cylinder is generated once and merged into the stationary part of the mesh at both the front and rear turbines positions. Corresponding patches at cylinder surfaces are then matched to form a pair of *sliding interfaces*. It is also beneficial in terms of time, since refinement near the blades



(a)



(b)

Figure 4.3: (a) Shape of wind turbine and tower. The blades are composed of NREL airfoils numbered S816, S817 and S818 [41]. (b) Close-up look of the actual computational mesh near the hub of the wind turbine.

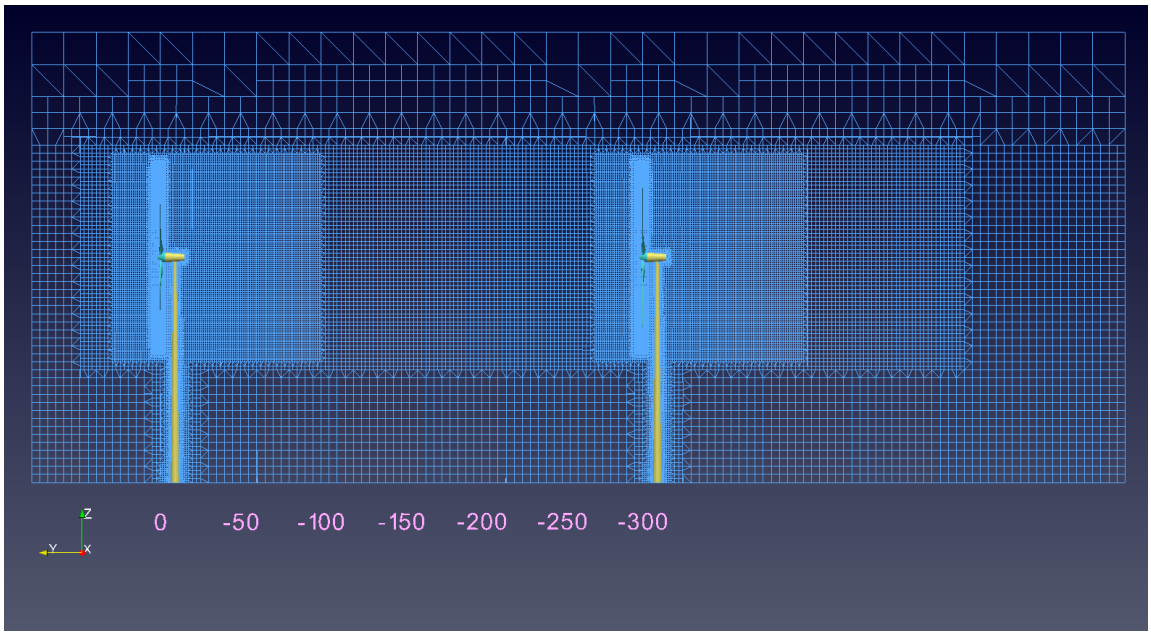


Figure 4.4: Cross section of mesh. Regions of different resolution levels can be seen.

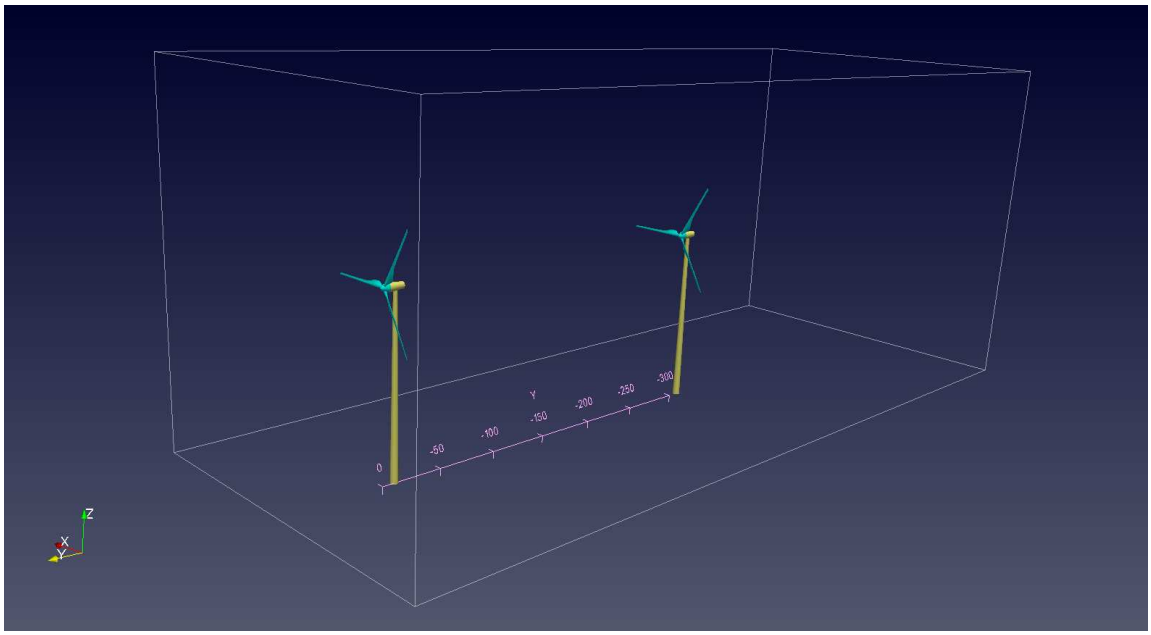


Figure 4.5: Configuration of flow domain. The dimensions are $280 \times 280 \times 680$ m. The direction of wind is aligned with the 680 m dimension. The front turbine is situated 80 m away from the domain inlet.

is very time-consuming. For the contra-rotating case, the mesh on the rear turbine blades are exactly the *mirror image* of that of the front turbine.

4.2 OpenFOAM CFD Coding and Turbulence Modeling

Computer codes for the Navier-Stokes equations are already available in OpenFOAM. Here, we describe the implementation of *PISO* (pressure-implicit splitting of operators) algorithm [25] in Listing 4.5 for illustrative purposes.

```
// define field vector fluid velocity u and f, face flux phi,
// pressure p, and dynamical viscosity nu

volVectorField u, f;
volScalarField p;
surfaceScalarField phi;
scalarField nu;

// construct the fluid velocity equation

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

// 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
```

```
30 // continuity equation
31
32 fvScalarMatrix pEqn
33 (
34    fvm::laplacian(rUA,p) == fvc::div(phi)
35 );
36 pEqn.solve();
37
38 // correct the fluid velocity by the post-solve pressure and update
39 // face flux
40
41 u = u - rUA * fvc::grad(p);
42 phi = phi - pEqn.flux();
```

Listing 4.5: The OpenFOAM code to solve the Navier-Stokes equation of incompressible fluid [9]. It implements the PISO algorithm [25].

Turbulence modeling can be further added to the solver above. OpenFOAM readily provides solvers, such as `pimpleFoam`, with run-time selectable turbulence modeling, which makes testing different models easier than ever. In addition to run-time selection of turbulence models, users also need to provide initial conditions for whatever fields are required by the chosen turbulence model, such as k, ϵ and ν_t.

We have run as OpenFOAM CFD case with our sliding interface dynamic mesh `forceDrivenMotionFvMesh` described above together with RANS and LES turbulence modeling. The turbines are assumed to be *freely rotating* subject to the wind force only, without any actions from the power generating unit. The case is computed with the standard solver `pimpleDyMFoam` in OpenFOAM, which implements a *PISO* pressure correction algorithm as well as a *SIMPLE* iteration.

Regarding the choice of constants in governing equations, please see Table 4.1

kinematic viscosity ν	1.4833×10^{-5} m^2/s
density ρ	1.205 kg/m^3
The Reynolds number (based on the 8 meter chord length)	$\approx 5 \times 10^6$
Initial and inlet values for $k-\epsilon$ turbulence model	k = 0.06 m^2/s^2, ϵ = 0.0495 m^2/s^3

Table 4.1: Physical and modeling parameters chosen in the CFD computation

4.3 Numerical Results and Visualization of Wind Turbine Flows and Interactions

We now illustrate numerical results. This is aimed at demonstrating our capability to compute *interacting* wind turbine flows. All the calculations were performed on the Texas A&M University Supercomputing Facility's EOS, an IBM iDataPlex Cluster 64-bit Linux with Intel Nehalem processors. Each computation took 30 hours of wall clock time with the use of 8 parallel processors.

For *postprocessing* our OpenFOAM computation, we produce graphical output using a free and open-source, multi-platform data analysis and visualization application named *ParaView* [2].

Snapshots of iso- and contra-rotating wind turbine flows are shown in Figures 4.6 and 4.7, respectively.

Example 4.1 (**Iso-rotating case)**. Here we consider two wind turbines co-axially placed with a separation distance of 300 m, with wind impinging from the left with velocity at 10.5 m/s. See a snapshot at t = 72 s in Figure 4.6. □

Example 4.2 (**Contra-rotating case)**. The two turbines are aligned in the same way as in Example 4.1, but the rear turbine blades are mirror images of the front turbine with the rear one contra-rotating. Under the same conditions as in Example 4.1, a snapshot at t = 72 s is shown in Figure 4.7. □

The flow streamlines before and after the wake of the front turbine reaches the

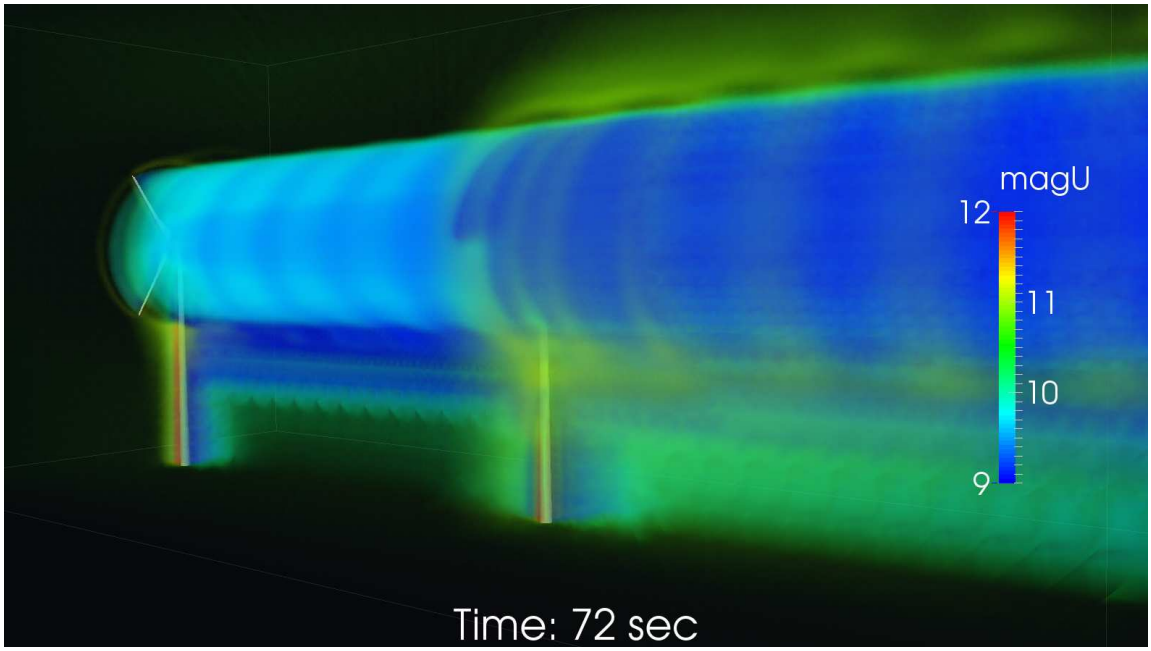


Figure 4.6: A snapshot of the flow of an iso-rotating case of two co-axially placed wind turbines at t = 72 s, where the separating distance is 300 m, and the wind is impinging from the left with velocity 10.5 m/s.

rear one is shown in Figure 4.8 for a RANS turbulence model ($k-\epsilon$) calculation and Figure 4.9 for calculations with an LES turbulence model (k-equation eddy-viscosity). When the two turbulence models are compared, an apparent difference is that the LES model generates smoother streamlines for the vortices, while streamlines obtained from RANS may show zigzags.

Remark 4.3. By comparing the flow patterns of iso-rotating and contra-rotating cases in Figures 4.8(c) and 4.8(d) (respectively Figures 4.9(c) and 4.9(d)), one sees that the flow goes *straight-through* the rear turbine in the contra-rotating case, in contrast to the iso-rotating case where a vortex is generated after the rear turbine.

Typical transient curves for angular velocity are shown in Figure 4.10. Comparisons of thrust (axial force), omega (angular velocity), and kinetic energy ($\frac{1}{2}I_R\omega^2$)

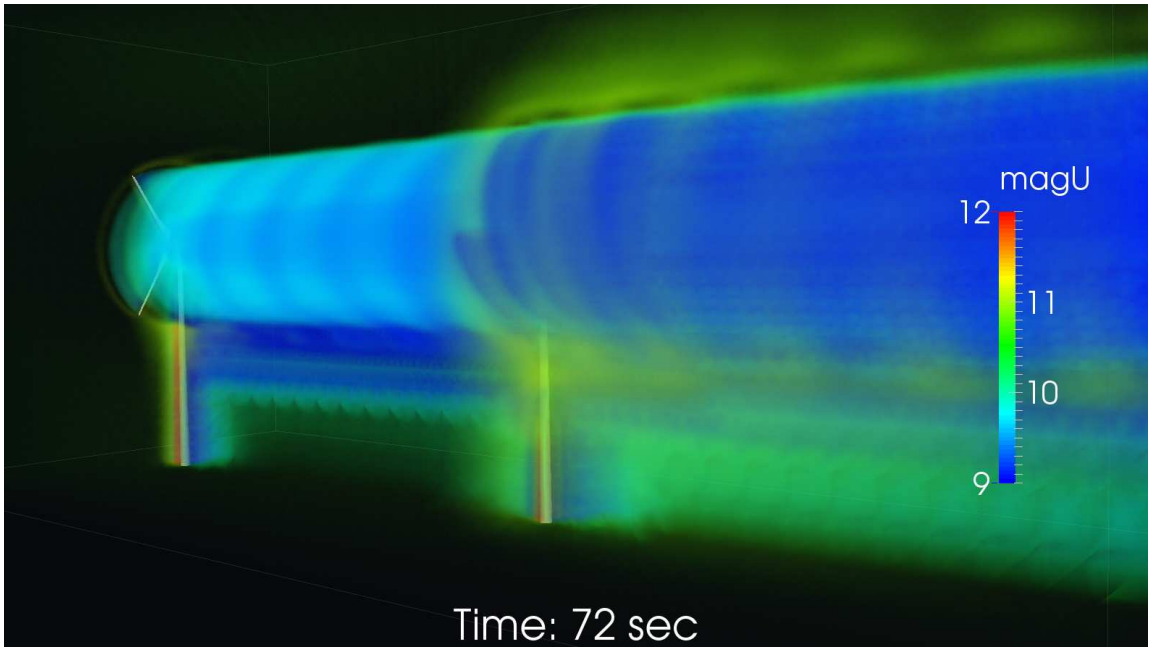


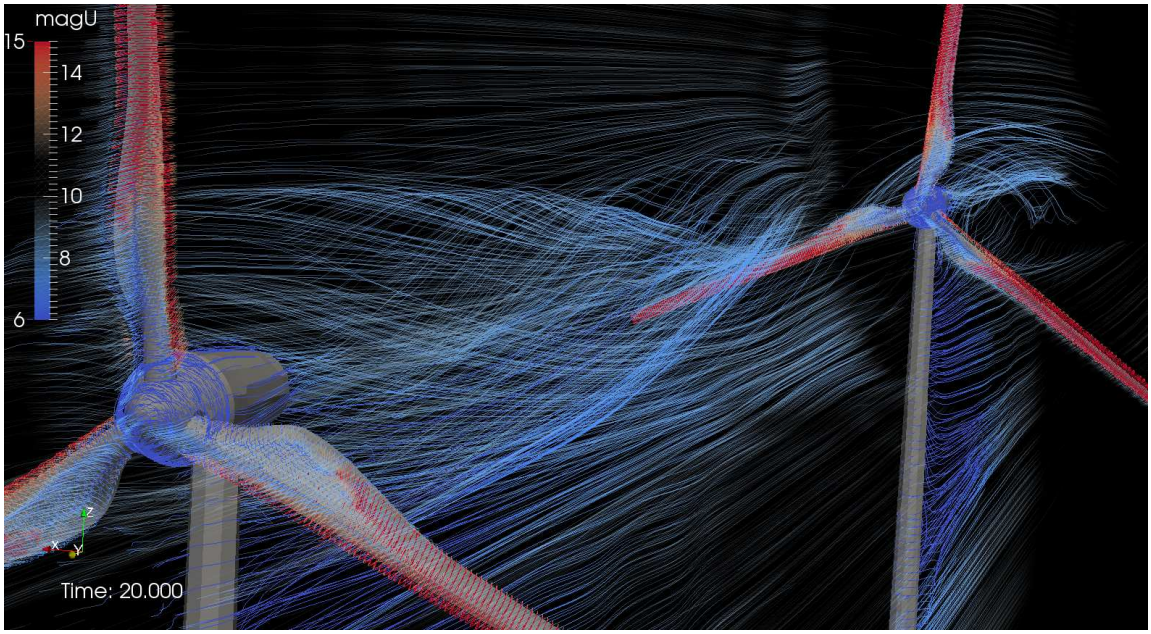
Figure 4.7: A snapshot of the flow of a contra-rotating case under the same conditions as those in Figure 4.6.

after they have almost reached a periodic state is shown in Table 4.2. *The Contra-rotating rear turbine has slightly faster rotation than that of the iso-rotating rear one.*

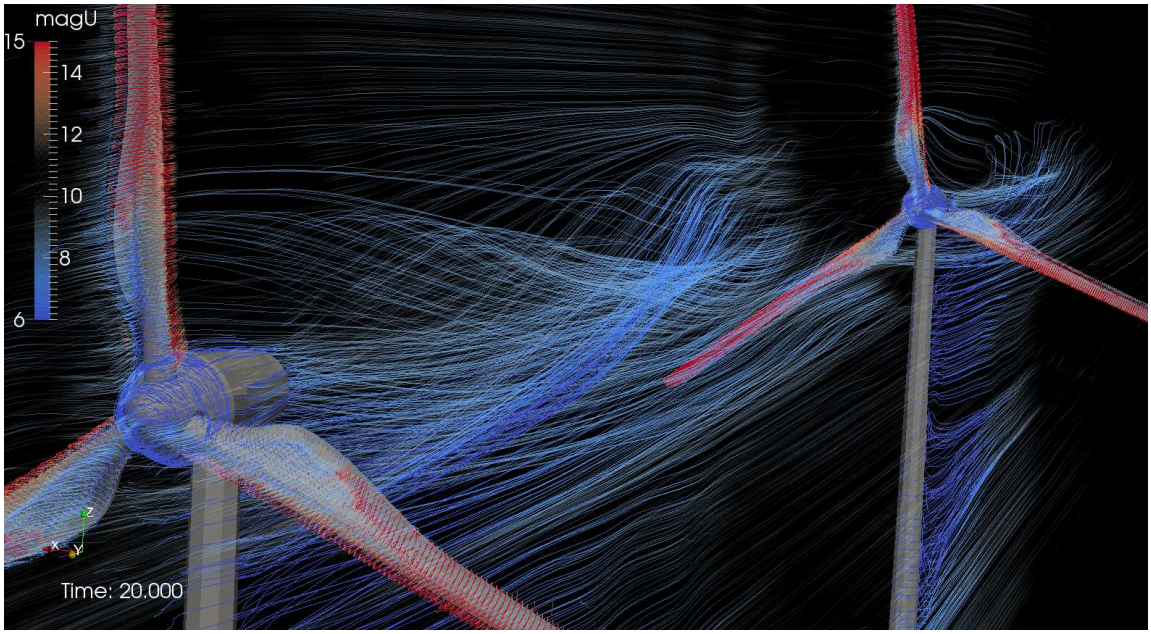
Example 4.4. Our calculations can be immediately adapted to compute a multi-turbine (wind farm) configuration. Here we consider a four-turbine layout, see Figure 4.11. □

In a turbulent flows, unsteady vortices appear on many scales. Therefore visualization of vortices is essential in understanding turbulent flows. Vortices are usually visualized as isosurfaces of values like Q or λ_2, which are called Q-criterion [24] and λ_2-criterion [27]. Here, Q and λ_2 are defined as

$$Q = \frac{1}{2}\left(|\boldsymbol{\Omega}|^2 - |\boldsymbol{D}|^2\right), \tag{4.1}$$

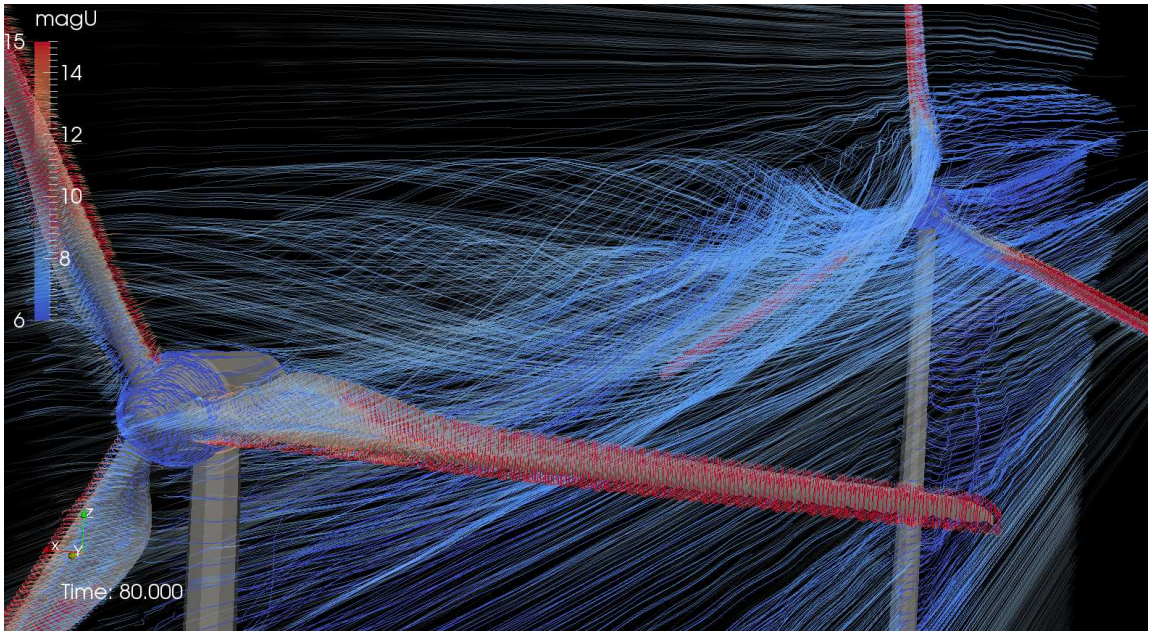


(a) RANS contra-rotating case, t = 20 s

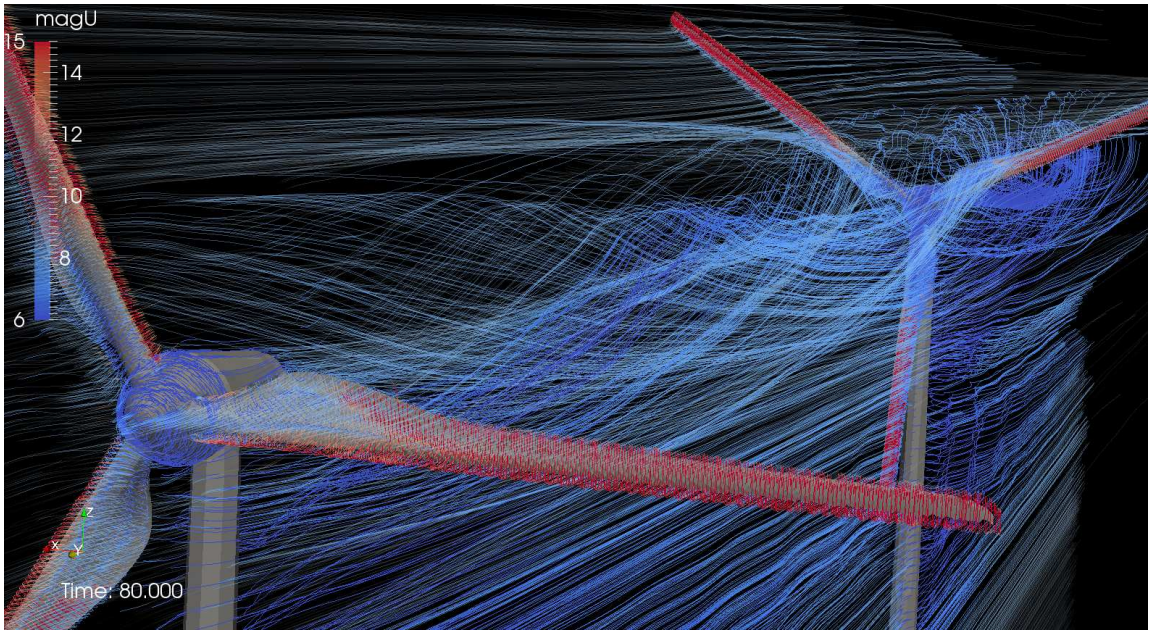


(b) RANS iso-rotating case, t = 20 s

Figure 4.8: OpenFOAM computational results by RANS turbulence model ($k-\epsilon$). (200 m spacing and 10.5 m/s wind speed). (a) (b) wake of the front turbine has not reached the rear one. (c) (d) wake of the front turbine has already reached the rear one.

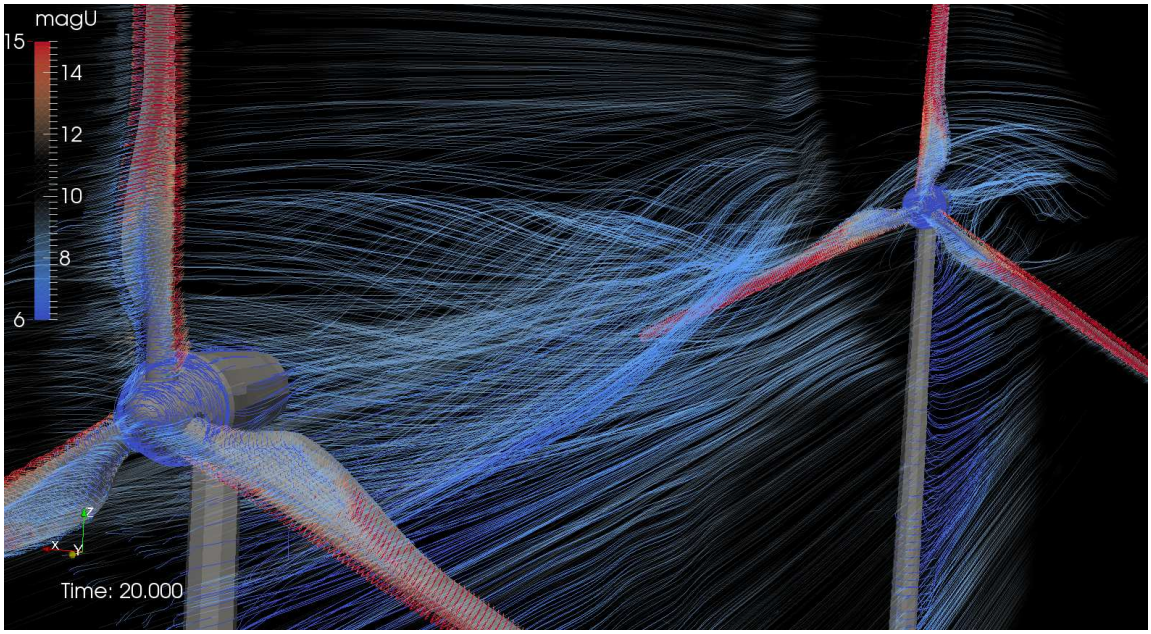


(c) RANS contra-rotating case, $t = 80$ s

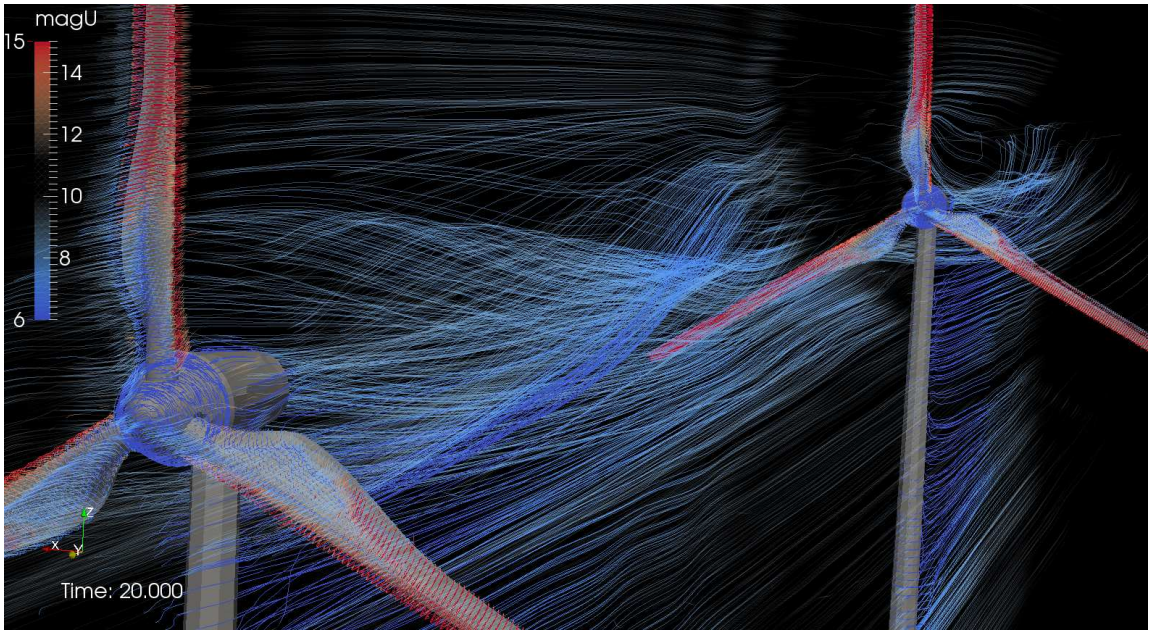


(d) RANS iso-rotating case, $t = 80$ s

Figure 4.8: Continued.

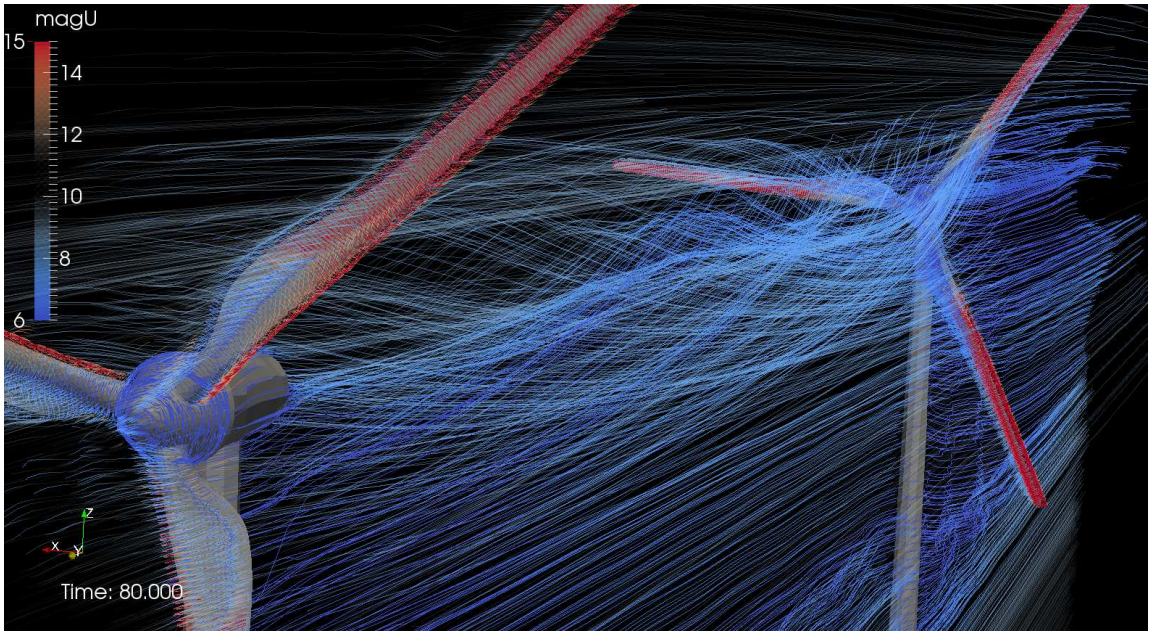


(a) LES contra-rotating case, $t = 20$ s

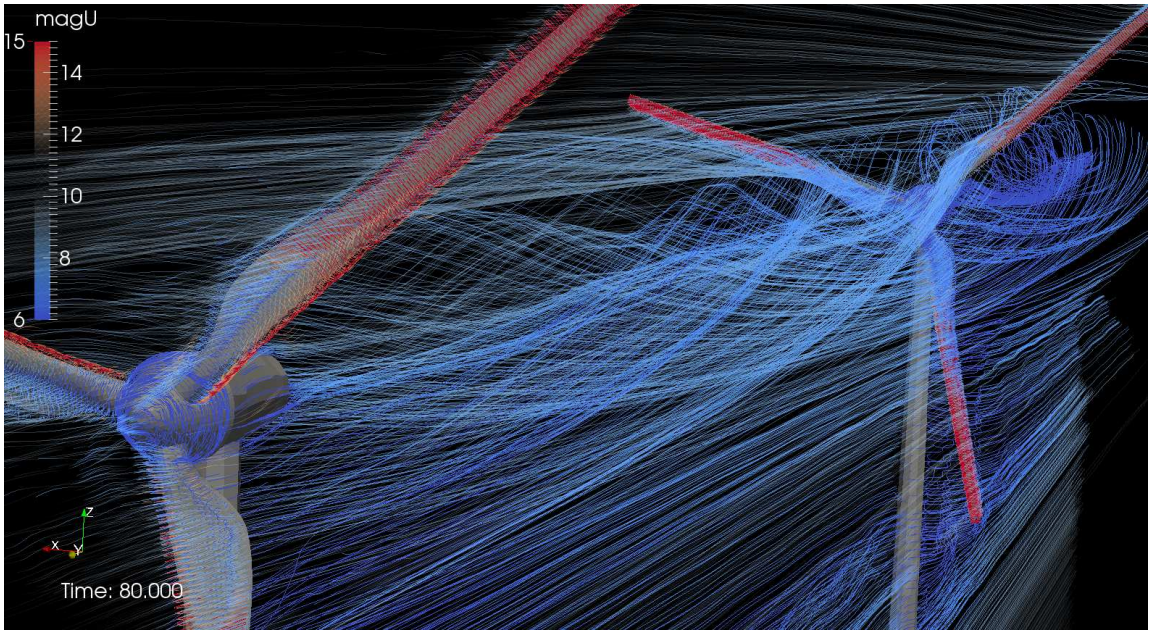


(b) LES iso-rotating case, $t = 20$ s

Figure 4.9: OpenFOAM computational results by LES turbulence model (oneEqEddy). (200 m spacing and 10.5 m/s wind speed). (a) (b) wake of the front turbine has not reached the rear one. (c) (d) wake of the front turbine has already reached the rear one.



(c) LES contra-rotating case, $t = 80$ s



(d) LES iso-rotating case, $t = 80$ s

Figure 4.9: Continued.

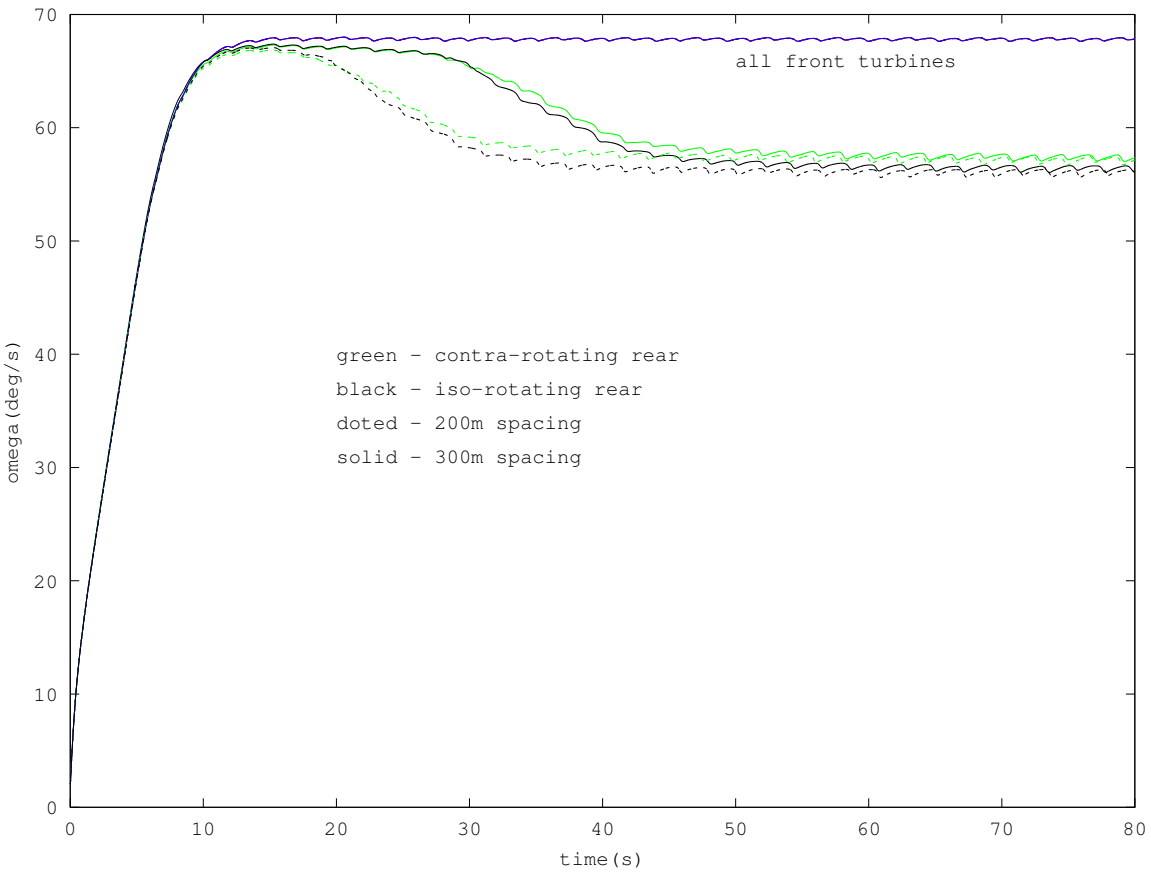


Figure 4.10: Transient curves for omega (ω, angular velocity). (200/300 m spacing and 10.5 m/s wind speed.) The speed of the front and rear turbines ramps up in the same way before the wake reaches the rear one, forming a drop in its speed. Contra-rotating rear turbine has slightly faster rotation than the iso-rotating rear one.

(a) 10.5 m/s wind speed

turbine	iso-rotating rear	contra-rotating rear	front
thrust (kN)	90.6	93.3	127.3
normalized thrust	1.000	1.030	1.405
omega (deg/s)	56.46	57.44	67.82
normalized omega	1.000	1.017	1.201
normalized kinetic energy	1.000	1.035	1.443

(b) 13 m/s wind speed

turbine	iso-rotating rear	contra-rotating rear	front
thrust (kN)	141.4	143.1	197.7
normalized thrust	1.000	1.012	1.398
omega (deg/s)	70.72	71.98	84.86
normalized omega	1.000	1.018	1.200
normalized kinetic energy	1.000	1.036	1.440

(c) 15.5 m/s wind speed

turbine	iso-rotating rear	contra-rotating rear	front
thrust (kN)	195.9	197.1	274.1
normalized thrust	1.000	1.006	1.399
omega (deg/s)	81.59	82.82	98.33
normalized omega	1.000	1.015	1.205
normalized kinetic energy	1.000	1.030	1.452

Table 4.2: Comparison of thrust (axial force), omega (angular velocity), and kinetic energy ($\frac{1}{2} I_R \omega^2$) after they are almost periodic with small fluctuation. (300 m spacing)

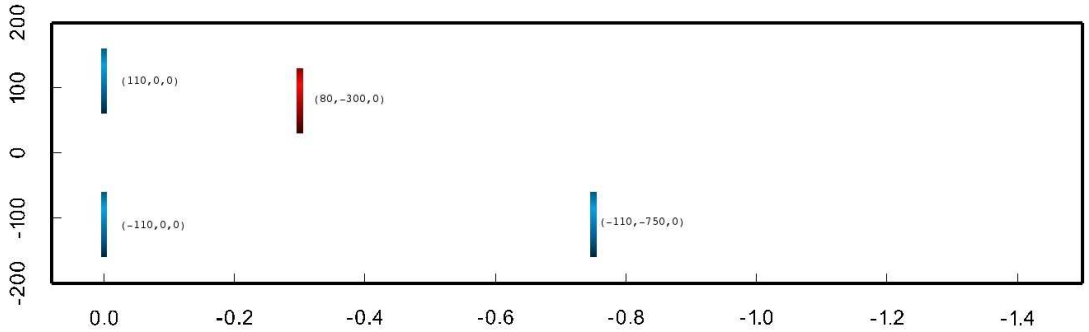


Figure 4.11: A four-turbine layout for Example 4.4. The center coordinates are displayed next to each turbine. The rotational direction of the upper-right turbine is different from the other three.

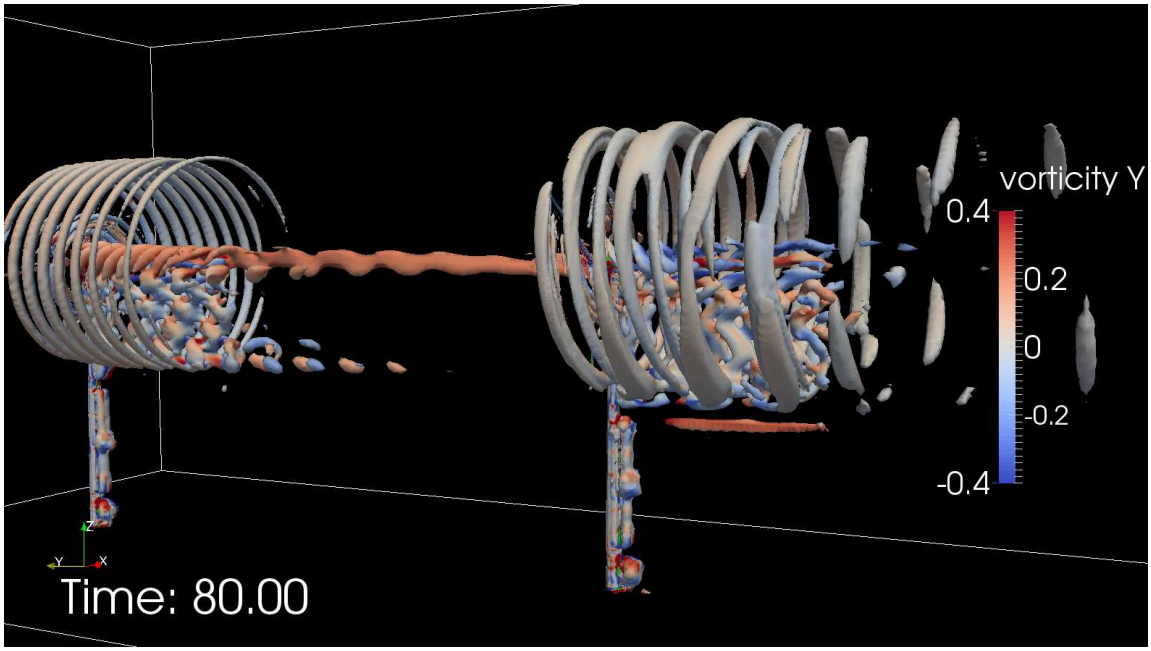
$$\lambda_2 = \text{second largest eigenvalue of } \boldsymbol{\Omega}^2 + \boldsymbol{D}^2, \tag{4.2}$$

where the strain rate tensor $\boldsymbol{D}$ and rotation rate tensor $\boldsymbol{\Omega}$ are defined as

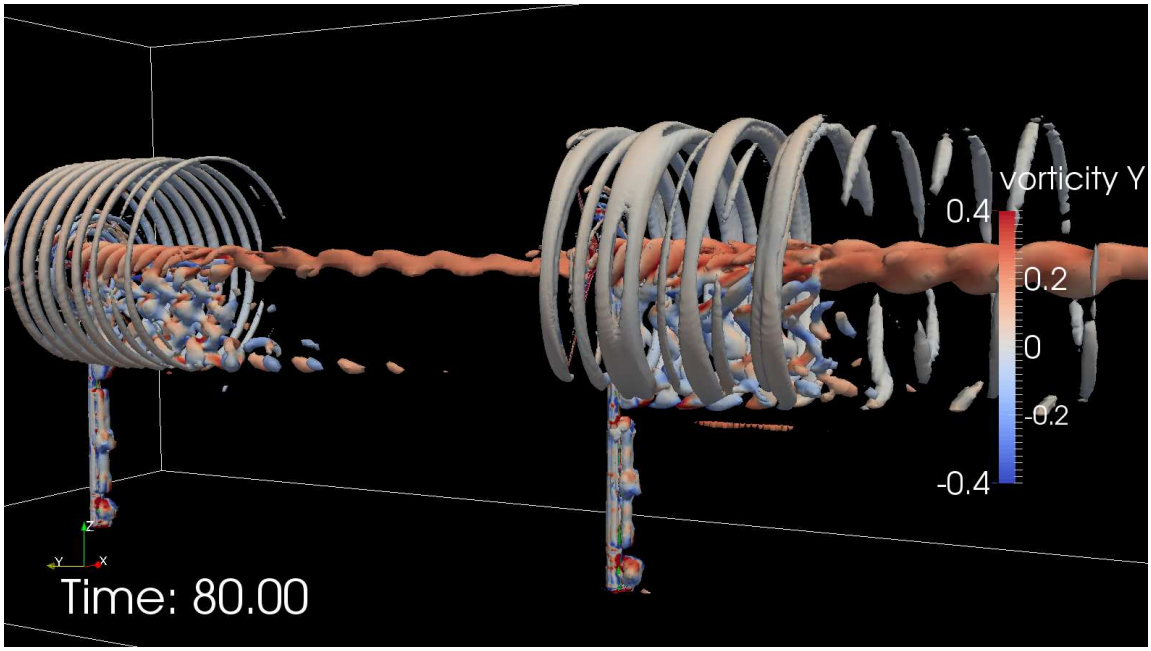
$$\boldsymbol{D} = \frac{1}{2}(\nabla \mathbf{u} + \nabla \mathbf{u}^\top), \qquad \boldsymbol{\Omega} = \frac{1}{2}(\nabla \mathbf{u} - \nabla \mathbf{u}^\top). \tag{4.3}$$

The criterion "$Q > 0$" or "$-\lambda_2 > 0$" can be generally interpreted as "local rotation rate being stronger than the strain rate".

Snapshots of Q and λ_2 isosurfaces are shown respectively in Figure 4.12 and Figure 4.13, where we can clearly identify the relatively strong vortices generated by blade tip, root and tower-blade interference. Note that these cases have uniform inlet flow, while in reality, the background turbulence will considerably affect the result.

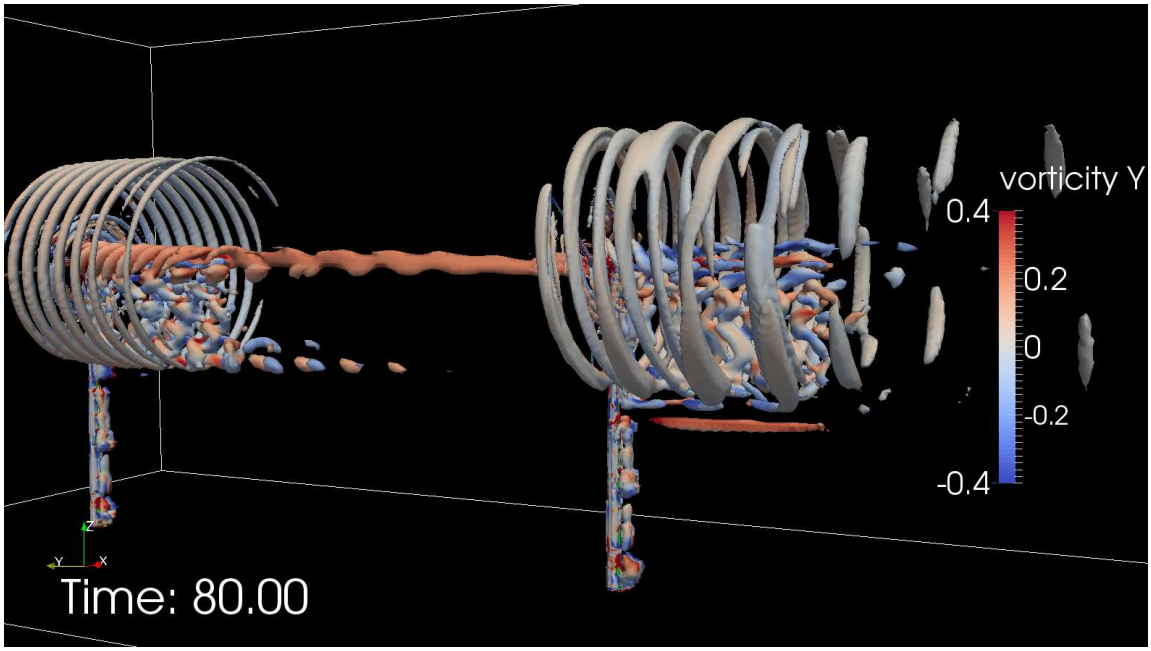


(a) LES contra-rotating case, $t = 80$ s

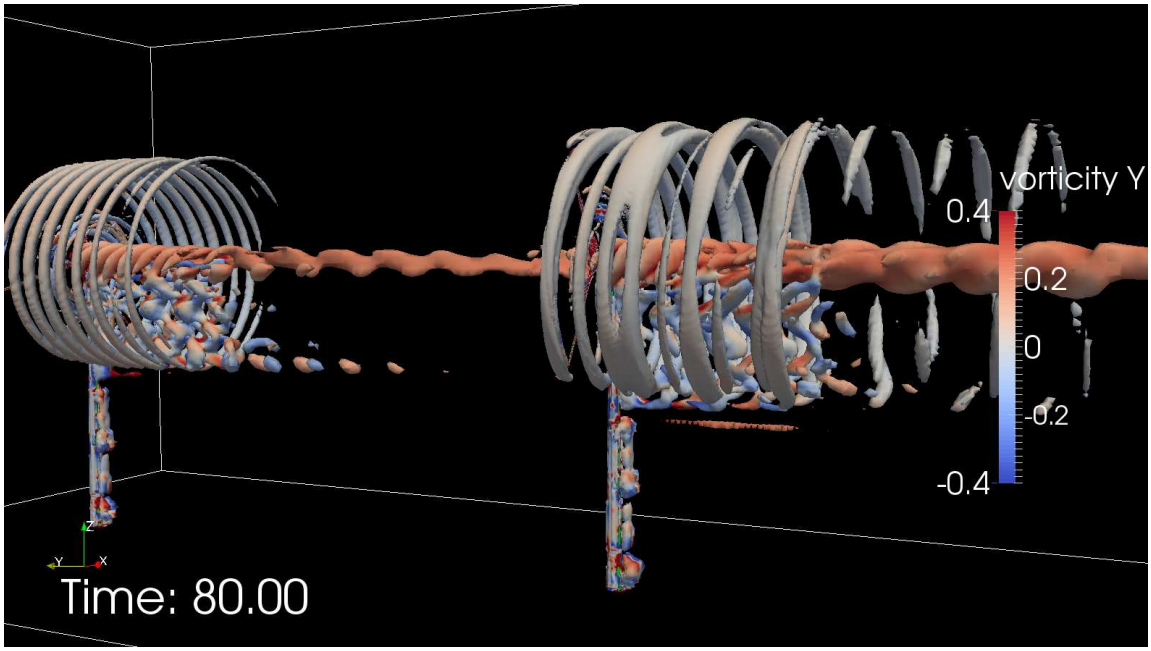


(b) LES iso-rotating case, $t = 80$ s

Figure 4.12: OpenFOAM computational results with Q-isosurface ($Q > 0.01$) is visualized for vortices. Blue and red color on the isosurface indicate different direction of rotation in the flow. Turbulent flow appears to be mainly generated by tip and root of the blades and by tower-blade interference.



(a) LES contra-rotating case, $t = 80$ s



(b) LES iso-rotating case, $t = 80$ s

Figure 4.13: OpenFOAM computational results with λ_2-isosurface ($-\lambda_2 > 0.01$) is visualized for vortices. Blue and red color on the isosurface indicate different direction of rotation in the flow. Turbulent flow appears to be mainly generated by tip and root of the blades and by tower-blade interference.

4.4 Proper Orthogonal Decomposition

Proper orthogonal decomposition (POD) is a procedure to extract information on coherent structures within a turbulent flow. It decomposes flow data into a basis that maximize the turbulent kinetic energy in a finite sum approximation. The procedure has been applied to turbulent data analysis [7], and more specifically, to wake analysis for wind turbines in [37].

The flow is sampled as $\mathbf{u}(\mathbf{x}_i, t_j)$, where $\{\mathbf{x}_i\}_{i=1}^n$ are cell centers of the mesh, and $\{t_j\}_{j=1}^m$ are equally spaced sampling times. The time averaging operation is denoted as $\langle\cdot\rangle = \frac{1}{m}\sum_{j=1}^m$. The inner product of vector fields $\{\mathbf{v}(\mathbf{x}_i)\}_{i=1}^n$ and $\{\mathbf{w}(\mathbf{x}_i)\}_{i=1}^n$ is defined as

$$(\mathbf{v}, \mathbf{w}) = \sum_{i=1}^n V_i \mathbf{v}(\mathbf{x}_i) \cdot \mathbf{w}(\mathbf{x}_i),$$

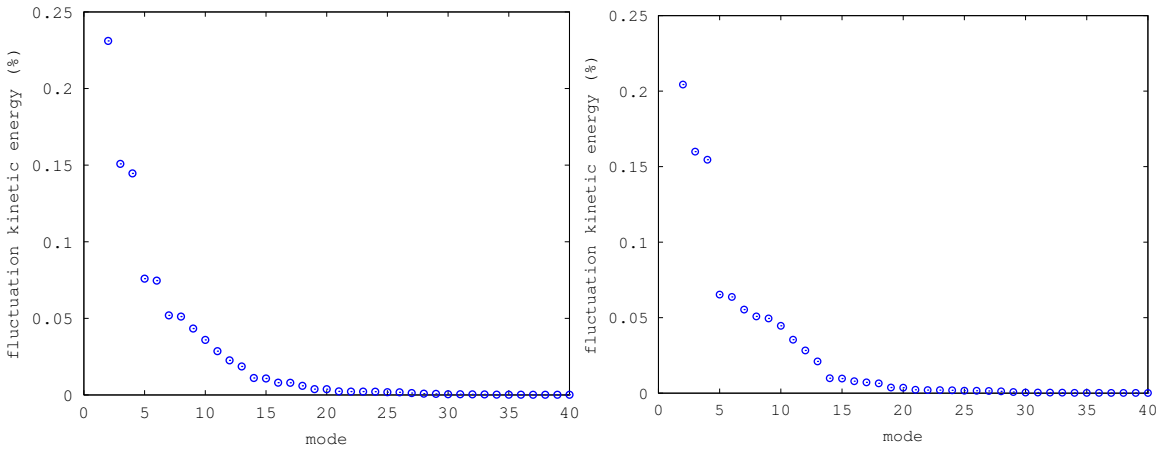
where V_i is the volume of the cell at $\mathbf{x}_i$. The weighted summation resembles the integral over the domain. The norm is subsequently defined as $\|\mathbf{v}\| = (\mathbf{v}, \mathbf{v})^{1/2}$.

The data is decomposed into

$$\mathbf{u}(\mathbf{x}_i, t_j) = \sum_{k=1}^m s_k c_k(t_j) \boldsymbol{\phi}_k(\mathbf{x_i}), \tag{4.1}$$

The set of basis, or modes, $\boldsymbol{\phi}_k$ is normalized such that $\|\boldsymbol{\phi}_k\| = 1$, and c_k are normalized such that $\langle c_k^2 \rangle = 1$ for all k. Then, the fluctuation energy on average is $\sum_{k=1}^m s_k^2$, and the portion carried by mode $\boldsymbol{\phi}_k$ is just s_k^2. We further order the modes $\boldsymbol{\phi}_k$ such that $s_1 > s_2 > \cdots > s_m$. Then the leading modes are the most energy-carrying modes in the turbulent flow.

In this study, we took 100 samples from the flow during $t = 60$ s to $t = 80$ s. The rotating part of the mesh (cf. Figure 4.1) is excluded from the analysis. Figure 4.14 shows the energy contribution of the leading modes to the total energy in the

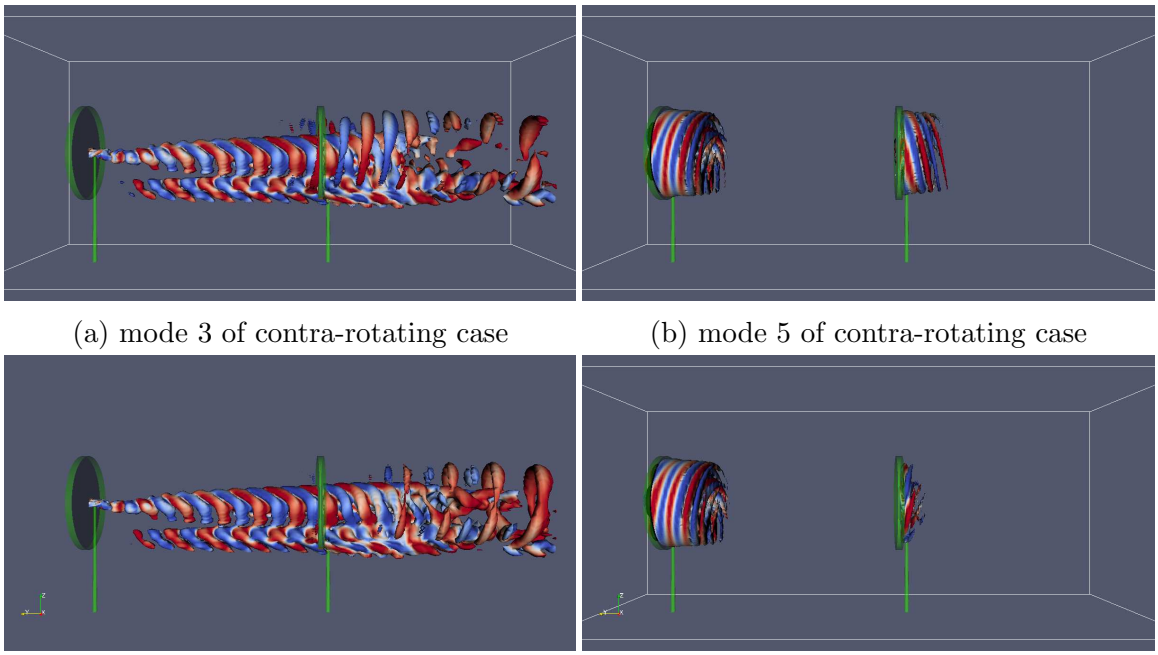


(a) contra-rotating case (b) iso-rotating case

Figure 4.14: Energy contribution of mode 2–40 to the total kinetic energy in the fluctuation.

fluctuation. The first mode is omitted since it mainly represent the mean flow. Modes often appear in pairs. For example, mode pair $(3, 4)$ share the same structure except a phase shift, and they contain almost the same fluctuation energy. The spatial structures of some of the leading modes, namely mode 3 and mode 5, can be seen in Figure 4.15.

Frequency analysis of the coefficients c_k is carried out and the *power spectral density* (PSD) is shown in log scale in Figure 4.16. The frequency is normalized as the Strouhal number $St = fR/U_\infty$. We can clearly identify the two dominant frequencies at $St \approx 1$ and $St \approx 3$. For instance, the dominant frequency is at $St \approx 1$ for mode pair $(3, 4)$, while it is $St \approx 3$ for mode pair $(5, 6)$. From Figure 4.15, we can see that mode 3 is closely related to blade-tower interaction, while mode 5 is mainly related to the helical blade wake. This perfectly explains the frequency ratio of 3.



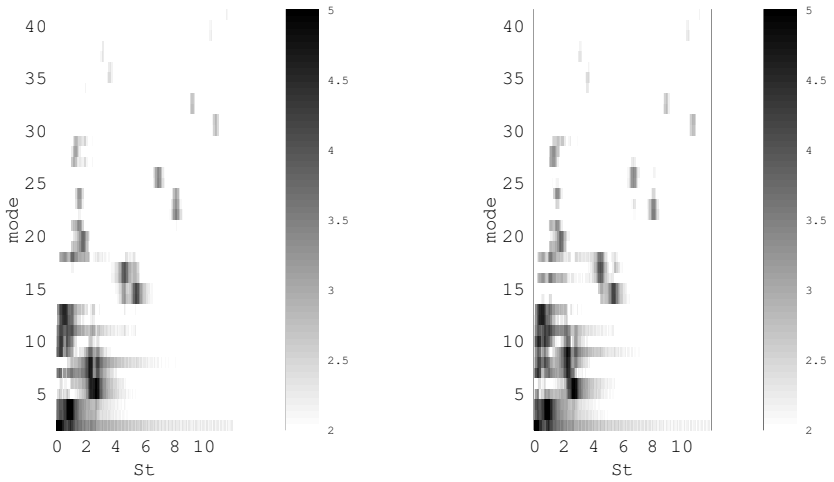
(a) mode 3 of contra-rotating case

(b) mode 5 of contra-rotating case

(c) mode 3 of iso-rotating case

(d) mode 5 of iso-rotating case

Figure 4.15: Structure of some selected modes



(a) contra-rotating case

(b) iso-rotating case

Figure 4.16: Power spectral density (PSD) of the POD time coefficients c_k in log scale for mode 2–40.

turbine radius	50 m
turbine center elevation	30 m under water
angular velocity	constant at 1 rad/s (~ 9.5 rpm)
inlet velocity (aligned with turbine axis)	4 m/s
water density ρ_{water}	998.2 kg/m^3
water kinematic viscosity ν_{water}	1×10^{-6} m^2/s
air density ρ_{air}	1 kg/m^3
air kinematic viscosity ν_{air}	1.48×10^{-5} m^2/s
Initial and inlet values for $k - \epsilon$ turbulence model	k = 0.06 m^2/s^2, ϵ = 0.0495 m^2/s^3

Table 4.3: Setting, physical and modeling parameters chosen in the two-phase CFD computation

4.5 A Turbine Operating in a Two-Phase Flow

Our preceding code development and calculations can be immediately extended and generalized to other similar applications such as ocean tidal or current power generation. In the following, we include a hypothetical situation where a turbine operates in a two-phase flow. The case setting is as shown in Table 4.3. The case is computed with the standard solver `interDyMFoam` in OpenFOAM, which implements the PISO/SIMPLE algorithm to solve the momentum equation, and *volume of fluid (VOF)* method with interface compression to track the fluid-fluid interface. A snapshot of the water-air interface is shown in Figure 4.17. Although the mesh is not in high resolution, one can clearly see the waves, ripples, and bubbles.

4.6 Concluding Remarks

Snapshots offered in this section provide useful insights for flow patterns near the blade boundary layer, farfield turbulence, and dominant modes (through POD), which will eventually help engineering of the best designs for wind turbines. The calculations are quite time consuming – each run takes several days on the Texas

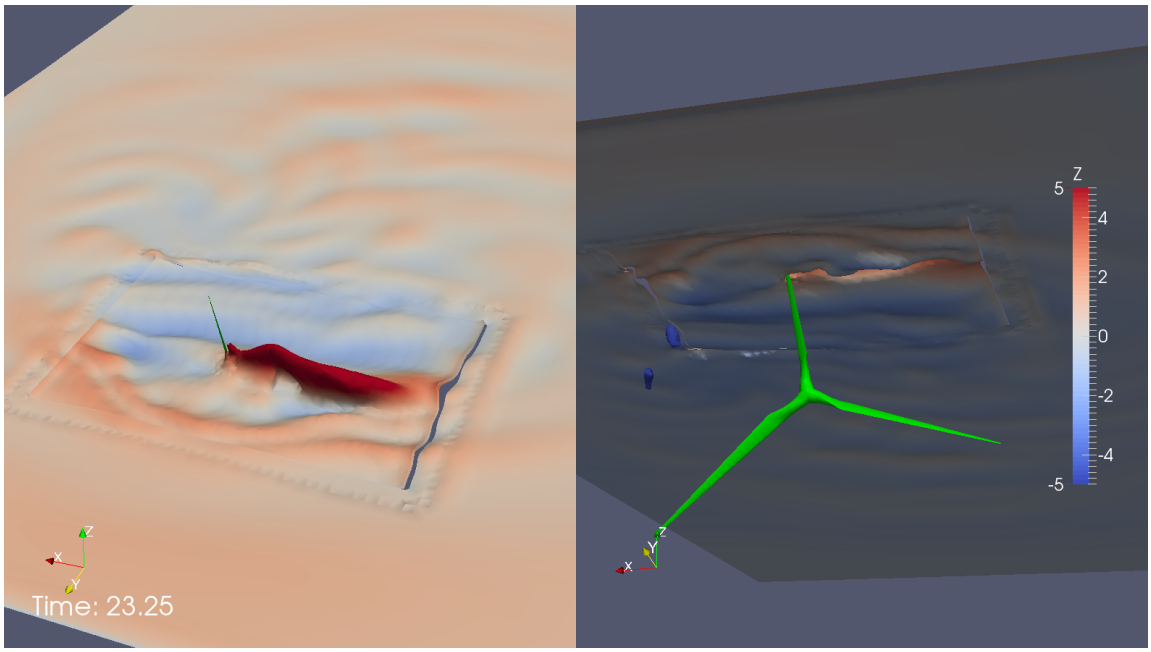


Figure 4.17: Snapshot of a wind turbine operating partially under water. On the left, the surface shown is the 0.4-isosurface of water volume fraction, viewed from above water level. On the right, the surface shown is the 0.6-isosurface of water volume fraction, viewed from below water level. The color indicates the elevation of the surface relative to original water level.

A&M supercomputer, even for a rather coarse mesh consisting of 0.6 million cells. Our hope is to eventually run our codes with about 30 - 50 million cells.

We have only considered the case where each wind turbine is freely-rotating. However, power-generating mechanisms and operating power-curves need to be carefully taken into account in the future. Nevertheless, this section have clearly demonstrated our ability to study dynamic behaviour of the wind power generators.

5. SUMMARY

In this thesis, two problems in computational mechanics, namely aircraft water entry and wind energy, have been studied together with the description of related theory and methodology. CFD calculations are carried out with proper schemes and computational techniques.

In the aircraft water entry problem, CFD techniques are used to simulate this complex and dynamic process under several cases. External loading data has been analyzed to estimate the severity of structural damage under the given scenarios. In the wind energy problem, blade resolved CFD calculations of wind turbines with towers are carried out. Interaction between wind turbines are studied.

It has been demonstrated that the CFD approach is advantageous in saving long and expensive processes of laboratory setup and measurements, while providing valuable information to the subject problem. Now, with the availability of more abundant free and open-source computational tools and user-friendly software such as OpenFOAM, 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*. However, challenges still remain in such interdisciplinary research. Sometimes mathematically advantageous methods fail when situation is not ideal, for example, on mesh with sub-optimal quality. On the other side, computational research sometimes can become very empirical, which comes with a lot of best practice guidelines and trial-and-error. In such cases, mathematical analysis are needed to provide more insights.

REFERENCES

[1] Serge Abrate, *Hull slamming*, Applied Mechanics Reviews **64** (2011), no. 6, 060803.

[2] James Ahrens, Berk Geveci, and Charles Law, *ParaView: An end-user tool for large data visualization*, The Visualization Handbook **717** (2005), 731.

[3] Arnold Barnett, *Cross-national differences in aviation safety records*, Transportation science **44** (2010), no. 3, 322–332.

[4] IE Barton, *Comparison of simple-and piso-type algorithms for transient flows*, International Journal for numerical methods in fluids **26** (1998), no. 4, 459–483.

[5] Martin Beaudoin and Hrvoje Jasak, *Development of a generalized grid interface for turbomachinery simulations with OpenFOAM*, Open source CFD international conference, 2008.

[6] John B Bell, Phillip Colella, and Harland M Glaz, *A second-order projection method for the incompressible navier-stokes equations*, Journal of Computational Physics **85** (1989), no. 2, 257–283.

[7] Gal Berkooz, Philip Holmes, and John L Lumley, *The proper orthogonal decomposition in the analysis of turbulent flows*, Annual review of fluid mechanics **25** (1993), no. 1, 539–575.

[8] Goong Chen, Cong Gu, Philip 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** (2015), no. 4, 330–344.

[9] Goong Chen, Qingang Xiong, Philip J Morris, Eric G Paterson, Alexey Sergeev, and Y Wang, *Openfoam for computational fluid dynamics*, Notices of the American Mathematical Society **61** (2014), no. 4, 354–363.

[10] Alexandre Joel Chorin, *Numerical solution of the navier-stokes equations*, Mathematics of computation **22** (1968), no. 104, 745–762.

[11] MS Darwish and F Moukalled, *Tvd schemes for unstructured grids*, International Journal of heat and mass transfer **46** (2003), no. 4, 599–611.

[12] R Eymard, Th Gallouët, R Herbin, and JC Latche, *Analysis tools for finite volume schemes*, Acta Math. Univ. Comenianae **76** (2007), no. 1, 111–136.

[13] Robert Eymard, Thierry Gallouët, and Raphaèle Herbin, *Finite volume methods*, Handbook of numerical analysis **7** (2000), 713–1018.

[14] Andrew G Fabula, *Ellipse-fitting approximation of two-dimensional, normal symmetric impact of rigid bodies on water*, Proceedings of the 5th midwestern conference on fluid mechanics, 1957, pp. 299–315.

[15] 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** (1987), no. 4, 274–280.

[16] David Finkleman, *A mathematical and engineering approach to the search of MH370*, 2014. Powerpoint Presentation (unpublished).

[17] Thierry Gallouët, Raphaele Herbin, and Marie Hélene Vignal, *Error estimates on the approximate finite volume solution of convection diffusion equations with general boundary conditions*, SIAM Journal on Numerical Analysis **37** (2000), no. 6, 1935–1972.

[18] PR Garabedian, *Oblique water entry of a wedge*, Communications on Pure and Applied Mathematics **6** (1953), no. 2, 157–165.

[19] Katuhiko Goda, *A multistep technique with implicit difference schemes for calculating two-or three-dimensional cavity flows*, Journal of Computational Physics **30** (1979), no. 1, 76–95.

[20] JL Guermond, Peter Minev, and Jie Shen, *An overview of projection methods for incompressible flows*, Computer methods in applied mechanics and engineering **195** (2006), no. 44, 6011–6045.

[21] Baodong Guo, Peiqing Liu, Qiulin Qu, and Jiawen Wang, *Effect of pitch angle on initial stage of a transport airplane ditching*, Chinese Journal of Aeronautics **26** (2013), no. 1, 17–26.

[22] Ami Harten, *High resolution schemes for hyperbolic conservation laws*, Journal of computational physics **49** (1983), no. 3, 357–393.

[23] Cyril W Hirt and Billy D Nichols, *Volume of fluid (vof) method for the dynamics of free boundaries*, Journal of computational physics **39** (1981), no. 1, 201–225.

[24] Julian CR Hunt, AA Wray, and Parviz Moin, *Eddies, streams, and convergence zones in turbulent flows*, Studying turbulence using numerical simulation databases, 2, 1988, pp. 193–208.

[25] Raad I Issa, *Solution of the implicitly discretised fluid flow equations by operator-splitting*, Journal of computational physics **62** (1986), no. 1, 40–65.

[26] Hrvoje Jasak, Aleksandar Jemcov, and Zeljko Tukovic, *OpenFOAM: A C++ library for complex physics simulations*, International workshop on coupled methods in numerical dynamics, 2007, pp. 1–20.

[27] Jinhee Jeong and Fazle Hussain, *On the identification of a vortex*, Journal of fluid mechanics **285** (1995), 69–94.

[28] Norman Jones and Tomasz Wierzbicki, *Dynamic plastic failure of a free-free beam*, International Journal of Impact Engineering **6** (1987), no. 3, 225–240.

[29] Feiqing Liu, Qiulin Qu, Baodong Guo, Xing Jin, Jiali Wu, and Kai Zhang, *Application of computational fluid dynamics in the planned ditching of a transport airplane*, Mechanics and Practices **36** (2014), 278–284. (in Chinese with English abstract).

[30] AM Mackey, *A mathematical model of water entry*, Admiralty Underwater Weapons Establishment TN **636/79** (1979).

[31] John R McGehee, Melvin E Hathaway, and Victor L Vaughan, *Water-landing characteristics of a reentry capsule*, National Aeronautics and Space Administration, 1959.

[32] ST Miller, Hrvoje Jasak, DA Boger, EG Paterson, and A Nedungadi, *A pressure-based, compressible, two-phase flow finite volume method for underwater explosions*, Computers & Fluids **87** (2013), 132–143.

[33] B Mohammadi and Olivier Pironneau, *Analysis of the k-epsilon turbulence model*, Wiley Hoboken, NJ, 1994.

[34] Suhas V Patankar and D Brian Spalding, *A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows*, International Journal of Heat and Mass Transfer **15** (1972), no. 10, 1787–1806.

[35] Alfio Quarteroni, Fausto Saleri, and Alessandro Veneziani, *Factorization methods for the numerical approximation of navier–stokes equations*, Computer methods in applied mechanics and engineering **188** (2000), no. 1, 505–526.

[36] T Chacón Rebollo and Roger Lewandowski, *Mathematical and numerical foundations of turbulence models and applications*, Springer Birkhäuser Basel, 2014.

[37] Sasan Sarmast, Reza Dadfar, Robert F Mikkelsen, Philipp Schlatter, Stefan Ivanell, Jens N Sørensen, and Dan S Henningson, *Mutual inductance instability of the tip vortices behind a wind turbine*, Journal of Fluid Mechanics **755** (2014), 705–731.

[38] Jie Shen, *On error estimates of projection methods for navier-stokes equations: first-order schemes*, SIAM Journal on Numerical Analysis **29** (1992), no. 1, 57–77.

[39] Wen Zhong Shen, Vinod Arun Kumar Zakkam, Jens Nørkær Sørensen, and Kari Appa, *Analysis of counter-rotating wind turbines*, Journal of physics: Conference series, 2007, pp. 012003.

[40] 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** (1951), no. 4, 379–417.

[41] James L Tangler and Dan M Somers, *NREL airfoil families for HAWTs*, National Renewable Energy Laboratory Golden, CO, 1995.

[42] Stephen Timoshenko and James M Gere, *Theory of elasticity stability*, McGraw-Hill New York, NY, 1961.

[43] Theodore von Kármán, *The impact on seaplane floats during landing*, Technical Notes National Advisory Committee for Aeronautics **321** (1929).

[44] Herbert Wagner, *Landing of seaplanes*, Technical Memorandum National Advisory Committee for Aeronautics **622** (1931).

[45] T. Wierzbicki and D. K. Yue, *Spacecraft crashworthiness - towards reconstruction of the Challenger accident*, American Society of Mechanical Engineers, Applied Mechanics Division, AMD **79** (1986), 31–46.

[46] Tomasz Wierzbicki and Dick K Yue, *Impact damage of the challenger crew compartment*, Journal of Spacecraft and rockets **23** (1986), no. 6, 646–654.

[47] Martin Wörner, *Numerical modeling of multiphase flows in microfluidics and micro process engineering: a review of methods and applications*, Microfluidics and nanofluidics **12** (2012), no. 6, 841–886.

[48] GX Wu, H Sun, and YS He, *Numerical simulation and experimental study of water entry of a wedge in free fall motion*, Journal of Fluids and Structures **19** (2004), no. 3, 277–289.

[49] JL Yang, TX Yu, and SR Reid, *Dynamic behaviour of a rigid, perfectly plastic free–free beam subjected to step-loading at any cross-section along its span*, International journal of impact engineering **21** (1998), no. 3, 165–175.