Development of a high-fidelity fluid-structure interaction framework

Master Thesis Aerospace Engineering

Lennarth Cohen (5002591)

Delft University of Technology

TU Delft

Development of a high-fidelity fluid-structure interaction framework

by

Lennarth Cohen (5002591)

For obtaining the degree of Master of Science in Aerospace Engineering at Delft University of Technology

Project Duration:	May 5, 2025 – May 22, 2026	
Thesis Supervisors:	Dr. J. Sodja	Responsible supervisor
	Dr. ir. A.H. van Zuijlen	Supervisor
External Committee Members:	Dr. B. Giovanardi	Chair
	Dr. W. Yu	Examiner
Github repository:	click here	

Cover: `https://wallspic.com/image/174606-airplane-airplane_portrait-flight-aircraft-airline`

TU Delft

(This page has intentionally been left blank.)

Preface

This thesis marks the culmination of a challenging yet deeply rewarding academic journey within the fields of engineering and aerodynamics. My interest in Fluid-Structure Interaction (FSI) stemmed from a desire to move beyond the traditional, theoretical constraints of rigid-body aerodynamics. While classical models often assume undeformable wing sections, the reality of modern aviation demands an understanding of structural deformation. Developing an FSI solver for aeroelastic modeling offered a compelling opportunity to bridge this gap, providing a tool to predict and design for the flexible behaviors inherent in contemporary aircraft design. It is my hope that this solver serves as a valuable contribution to the University and provides a foundation for future research in this field.

I am profoundly grateful to the individuals who supported me throughout this process. I extend my sincerest thanks to my supervisors, Jurij Sodja and Alexander van Zuijlen, for their steadfast guidance and the insightful discussions that defined out biweekly meetings. Their expertise was instrumental in shaping the direction of this work.

I would also like to express my gratitude to Stefan de Boer for this invaluable assistance with MSC Nastran. Stefan was always available to navigate the complexities of structural modeling, consistently providing thorough explanations that deepend my understanding of the software. Finally, I am incredibly thankful to my family and my girlfriend; their constant encouragement, patience, and support have been the bedrock of my education and this achievement.

Lennarth Cohen (5002591)
Delft, May 8, 2026

Abstract

The analysis of flight loads and their structural impact is fundamental to aircraft design and certification. Analysis generally relies on low-fidelity aeroelastic solvers because of their low computational cost. However, these linear models fail to capture critical non-linear flow phenomena such as transonic shocks, flow separation, thickness- and viscous effects. Yet these effects are necessary to accurately asses extreme limiting load cases, potentially leading to non-conservative design estimates.

This research addresses this physical modeling gap by developing and validating a high-fidelity partitioned static Fluid-Structure Interaction (FSI) framework. The proposed framework establishes a two-way coupling between Ansys Fluent and MSC Nastran, leveraging Computational Fluid Dynamics (CFD) to resolve complex aerodynamic phenomena and the Finite Element Method (FEM) for high-fidelity structural modeling. The framework is applied to the Onera M6 wing, serving as the primary validation case.

A parametric study involving multiple high-fidelity static FSI simulations across varied angles of attack and Mach numbers is then conducted. These results are contrasted against low-fidelity predictions, quantifying the significant errors introduced by neglecting non-linear flow characteristics. This comparison proves the high-fidelity framework's definitive capability to capture the complex physics required for accurate static aeroelastic modeling.

Contents

Nomenclature

List of Abbreviations

Abbreviation	Definition
AR	Aspect Ratio
BDF	Bulk Data File
BPR	By-Pass Ratio
CFD	Computational Fluid Dynamics
CoP	Center of Pressure
CV	Control Volume
DLM	Doublet Lattice Method
DNS	Direct Numerical Simulation
DOF	Degree of Freedom
FEA	Finite Element Analysis
FEM	Finite Element Method
FSI	Fluid-Structure Interaction
FVM	Finite Volume Method
HDF	Hierarchical Data Format
HPC	High-Performance Computing
LE	Leading Edge
NN	Nearest-Neighbor
PBS	Portable Bash File
RANS	Reynolds-Averaged Navier-Stokes
RBE	Rigid Body Element(s)
SAF	Sustainable Aviation Fuel
TE	Trailing Edge
UDF	User-Defined Function

List of Symbols

Symbol	Definition	Unit
A	Area magnitude	[m^2]
$\vec{A}$	CFD surface mesh face area vector	[m^2]
A_x	CFD surface mesh face x-direction area	[m^2]
A_y	CFD surface mesh face y-direction area	[m^2]
A_z	CFD surface mesh face z-direction area	[m^2]
AR	Wing Aspect Ratio	[-]
b	Wing span	[m]
$[C]$	Global damping matrix	[kg/s]
c_{mean}	Mean aerodynamic chord	[m]
C_D	Drag-coefficient	[-]
$C_{D,fine}$	Drag-coefficient of finest CFD mesh	[-]
C_L	Lift-coefficient	[-]
$C_{L,fine}$	Lift-coefficient of finest CFD mesh	[-]
C_M	Moment coefficient	[-]
C_p	Pressure coefficient	[-]
d	Distance from dynamic mesh surface boundary	[m]
$\vec{d}$	Moment arm vector	[m]
d_x	Distance between CFD mesh node and loading line node in x-direction	[m]
d_y	Distance between CFD mesh node and loading line node in y-direction	[m]
d_z	Distance between CFD mesh node and loading line node in z-direction	[m]
E	Total energy	[J]
$\vec{F}$	External body forces	[N]
$\{F\}$	Static global load vector	[N]
$\{F(t)\}$	Dynamic global load vector	[N]
$\vec{F}_{LL,i}$	Force vector at iteration i	[N]
$\vec{F}_{LL,i-1}$	Force vector at iteration i-1	[N]
$F_{LL,x}$	Force in x-direction on loading line node	[N]
$F_{LL,y}$	Force in x-direction on loading line node	[N]
$F_{LL,z}$	Force in x-direction on loading line node	[N]
$\vec{F}_p$	Pressure force vector	[N]
$\vec{F}_{tot,face}$	Total force vector on CFD surface mesh face	[N]
$\vec{F}_v$	Viscous force vector	[N]
F_x	Force in x-direction	[N]
F_y	Force in y-direction	[N]
F_z	Force in z-direction	[N]
$F_{centroid,x}$	Force in x-direction in CFD surface mesh centroid	[N]
$F_{centroid,y}$	Force in x-direction in CFD surface mesh centroid	[N]

Symbol	Definition	Unit
$F_{centroid,z}$	Force in x-direction in CFD surface mesh centroid	[N]
$\vec{g}$	Gravity acceleration vector	[m/s^2]
h	Specific enthalpy	[J/kg]
$\vec{J}$	Heat flux vector	[W/m^2]
$[K]$	Static global stiffness matrix	[kg/s^2]
$[K(u)]$	Dynamic global stiffness matrix	[kg/s^2]
l_{max}	Maximum CFD volume cell length	[mm]
M	Mach number	[-]
$[M]$	Global mass matrix	[kg]
$\vec{M}_{LL,i}$	Moment vector at iteration i	[N $\cdot$ m]
$\vec{M}_{LL,i-1}$	Moment vector at iteration i-1	[N $\cdot$ m]
$M_{LL,x}$	Moment around x-axis on loading line node	[N $\cdot$ m]
$M_{LL,y}$	Moment around y-axis on loading line node	[N $\cdot$ m]
$M_{LL,z}$	Moment around z-axis on loading line node	[N $\cdot$ m]
$\vec{n}$	Normal vector	[-]
$\vec{n}_f$	Normal vector fluid mesh face	[-]
$\vec{n}_s$	Normal vector structure mesh face	[-]
N_{cells}	Number of CFD volume mesh cells	[-]
p	Pressure	[Pa]
p_f	Pressure on fluid mesh boundary	[Pa]
p_s	Pressure on structure mesh boundary	[Pa]
p_∞	Reference air presure	[Pa]
r_x	Rotation around x-axis	[rad]
$r_{x,LL}$	Rotation of loading line node around x-axis	[rad]
r_y	Rotation around y-axis	[rad]
$r_{y,LL}$	Rotation of loading line node around y-axis	[rad]
r_z	Rotation around z-axis	[rad]
$r_{z,LL}$	Rotation of loading line node around z-axis	[rad]
S	Wing surface area	[m^2]
S_{max}	Maximum CFD surface mesh area	[mm^2]
S_{min}	Minimum CFD surface mesh area	[mm^2]
$\mathbf{S}_h$	Mass conservation source term	[kg/(m^3$\cdot$s)]
$\mathbf{S}_m$	Momentum conservation source term	[kg/(m^2$\cdot$s^2)]
t	Time	[s]
t_{final}	End time simulation	[s]
$\vec{u}$	Pseudo velocity of dynamic mesh	[m/s]
$\{u\}$	Displacement vector of structure node(s)	[m] and/or [rad]
$\vec{u}_f$	Displacement vector fluid Ω_{FS}	[m]
$\vec{u}_s$	Displacement vector structure Ω_{FS}	[m]
$\{\dot{u}\}$	Time derivative of $\{u\}$	[m/s] and/or [rad/s]
$\{\ddot{u}\}$	Second time derivative of $\{u\}$	[m/s^2] and/or [rad/s^2]
v	Velocity magnitude	[m/s]
V	Volume	[m^3]

Symbol	Definition	Unit
V_∞	Reference air velocity	[m/s]
x	Chordwise location	[m]
x_{CFD}	CFD surface mesh face vertex x-location	[m]
x_{LL}	FEM loading line node x-location	[m]
y	Spanwise location	[m]
y_{CFD}	CFD surface mesh face vertex y-location	[m]
y_{LL}	FEM loading line node y-location	[m]
z	Vertical location	[m]
z_{CFD}	CFD surface mesh face vertex z-location	[m]
z_{LL}	FEM loading line node z-location	[m]
α	Angle of attack	[deg]
α_{damp}	Mass proportional damping coefficient	[1/s]
α_{diff}	Dynamic mesh diffusion parameter	[-]
α_{eff}	Effective angle of attack	[deg]
β_{damp}	Stiffness proportional damping coefficient	[s]
γ	Dynamic mesh diffusion coefficient	[-]
ΔC_p	Pressure coefficient difference	[-]
$\Delta\vec{F}_{LL,i}$	Force increment vector at iteration i	[N]
$\Delta\vec{M}_{LL,i}$	Moment increment vector at iteration i	[N·m]
Δt	Time step	[s]
Δx_{CFD}	Displacement of surface mesh vertex x-direction	[m]
Δy_{CFD}	Displacement of surface mesh vertex y-direction	[m]
Δz_{CFD}	Displacement of surface mesh vertex z-direction	[m]
Δx_{LL}	Displacement of loading line in x-direction	[m]
Δy_{LL}	Displacement of loading line in y-direction	[m]
Δz_{LL}	Displacement of loading line in z-direction	[m]
Δx_{tip}	Wing tip displacement of structure in x-direction	[m]
Δy_{tip}	Wing tip displacement of structure in y-direction	[m]
Δz_{tip}	Wing tip displacement of structure in z-direction	[m]
λ	Taper ratio	[-]
Λ_{LE}	Leading edge sweep angle	[deg]
Λ_{TE}	Trailing edge sweep angle	[deg]
ρ	Fluid density	[kg/m^3]
ρ_∞	Reference air density	[kg/m^3]
$\bar{\bar{\tau}}$	Viscous stress tensor	[Pa]
τ_x	Viscous force on mesh surface x-direction	[N]
τ_y	Viscous force on mesh surface y-direction	[N]
τ_z	Viscous force on mesh surface z-direction	[N]
Ω_F	Physical fluid domain	[-]
$\Omega_{F,num}$	Numerical fluid domain	[-]
Ω_S	Physical structure domain	[-]
$\Omega_{S,num}$	Numerical structure domain	[-]
Ω_{FS}	Physical interaction domain	[-]
$\Omega_{FS,num}$	Numerical interaction domain	[-]

List of Figures

List of Tables

(This page has intentionally been left blank.)

1 Introduction

Aviation has undergone significant changes since its dawn in the early 1900s [1]. The challenges have shifted from staying airborne to adhering to certification and societal needs. These challenges include the need for sustainable and noise reducing technology in the aviation sector, and this has led to a significant change in aircraft design. Apart from the changes in the aircraft design, the effect of the aviation sector on the global economy has changed significantly [2, 3].

In 2023, the global economic impact of the commercial aviation sector was estimated at $4.1 trillion. This estimate includes revenues of the civil aviation sector (passenger- and cargo transport), as well as revenues that were enabled by the civil aviation sector such as tourism revenues [4]. It was estimated that 58% of international tourism was enabled by the civil aviation sector. This underscores the significance the aviation sector induces globally, which is a huge contrast to its early days. Additionally, it is expected to grow annually by 3.9% in the coming years [4]. However, the significant contributions of the civil aviation sector do come at a cost. The emissions of this sector contributed to approximately 2.0% of the global CO_2 emissions among other emissions. The number might seem insignificant, however, continuing like this would mean the aviation emission goals for 2030 and 2050 would not be reached[1] [4].

The majority of the emissions of the aviation sector is caused by engine emissions of aircraft. This means a reduction of the fuel usage or improving the fuel efficiency would directly cause a reduction of emissions. A study by EUROCONTROL [5] on the average cost breakdown of several airlines in 2019 has shown that fuel is a significant contributor to the expenditure to keep an airline running. They highlighted that the cost of fuel and oil contributes to approximately 24.7% of total airline costs [5]. Thus reducing fuel usage is not only beneficial for the environment, but also for the airlines to reduce operational costs.

There are different strategies to reduce the fuel consumption. The consumption is dictated by the efficiency of the aircraft, and it can therefore be modified by increasing the overall efficiency of the aircraft during operations. Increasing the overall efficiency of the aircraft means either, reducing the structural weight, improving engine efficiency, or improving the aerodynamic efficiency. This thesis will focus on improving the aero-

[1] The goal for 2050 is for the aviation sector to have net-zero carbon emissions. The goal for 2030 is scaled up usage of SAF, improved operation efficiency and transition to new aircraft technologies. This 2030 goal is to set up for the changes in the aviation sector that are needed to achieve the 2050 goal [4].

dynamic efficiency, however, it is noteworthy to mention that the theory discussed can also be employed to reduce the structural weight. The attempts of aircraft manufacturers to employ the mentioned fuel reduction strategies will be shortly discussed below by highlighting the trends in aircraft design.

Figure 1.1 depicts the attempts of aircraft manufacturers through the years to save weight by making use of composites. Composites are materials that are constructed by combining two or more constituent materials with different physical or chemical properties. This combination creates a new material with characteristics superior to the individual materials, and these characteristics can be tailored to the application requirements. They can be properties like reduced weight, increased strength, and greater durability among others. This tailoring of materials resulted in a more prominent role of composites in the contemporary aircraft designs [6, 7].

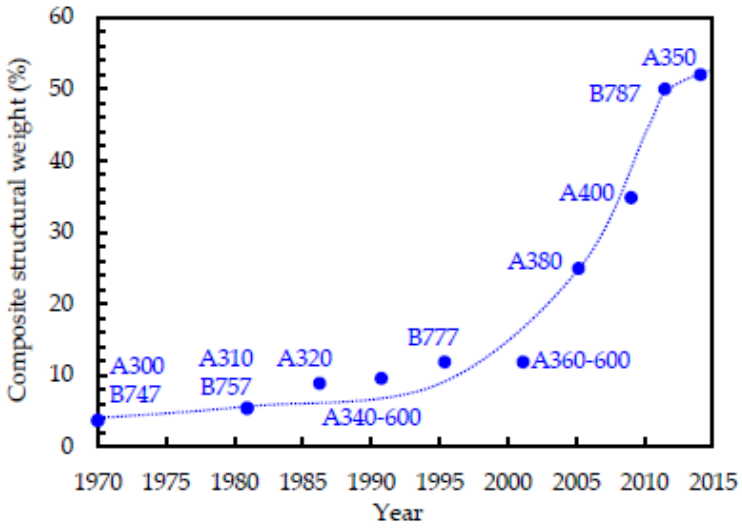

Figure 1.1: Composite percentage of aircraft through the years [8].

The second trend is shown in Figure 1.2, this figure shows the trend in increasing engine efficiency. The early propulsion systems used in aircraft were mainly piston- and turbojets engines, this shifted to mainly turbofan engines in more recent aircraft. In turbofan engines the flow through the engine consists of part cold air, and part hot air. The area of cold to hot air is defined as the By-Pass Ratio (BPR). This shift from turbojet to first low BPR is what started the first significant step in engine fuel efficiency improvement. The trend from these low BPR turbojets to high BPR turbojet engines is then what further improved the fuel efficiency of engines. Increasing the BPR of aircraft engines would thus further improve the fuel efficiency. However, this also means a significant increase in engine size. The size of these theoretical very high BPR engines would significantly negatively impact interference between the engine and airframe [9]. This suggests that further increasing the BPR of turbofan engines is not an option to facilitate further improvement of fuel efficiency, so better fuel efficiency requires novel engine designs [1, 10, 11].

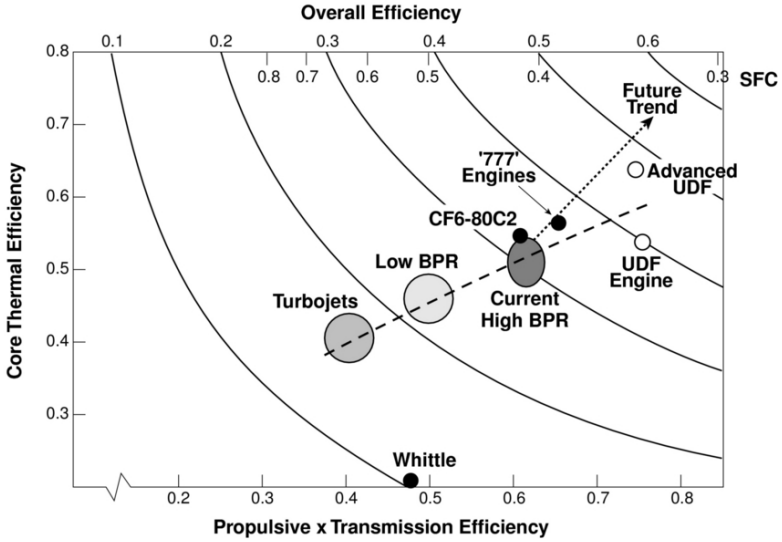

Figure 1.2: Engine propulsive efficiency [11].

The third strategy for reducing fuel consumption is to improve the aerodynamic efficiency of the aircraft. Aerodynamic efficiency can be improved by reducing the drag that is generated by the lift of the aircraft. Lift-induced drag constitutes up to 80% of total drag in the climb phase and up to 40% during cruise [12]. To address this, designers focus on the wing's Aspect Ratio (AR), as induced drag is inversely proportional to this geometric value [13]. The trend in Figure 1.3 reveals the attempts to increase the AR to reduce the significance of the lift-induced drag.

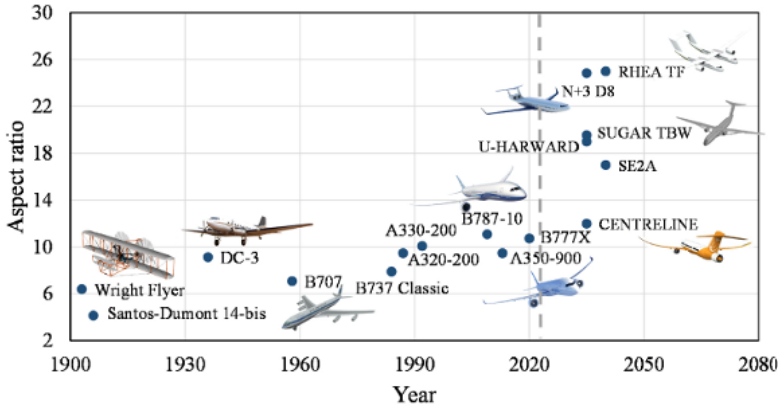

Figure 1.3: Aspect ratio through the years [13].

The increasing AR also causes the wings of the aircraft to be more susceptible to interactions between airflow and the structure. These phenomena in which airflow and structure interact are called aeroelastic effects, and they need to be modeled accurately to ensure compliance with regulations and optimal aerodynamics [11]. The theory used to model aeroelastic effects is part of the field Fluid-Structure Interaction (FSI) theory [14, 12]. However, a significant challenge remains in the implementation of this theory. Specifically, while aerodynamic and structural modeling have each advanced significantly, these disciplines have historically evolved in isolation. This

traditional separation persists despite the fact that, in high-AR designs, the two fields are fundamentally and inextricably interconnected[1, 11].

There are two options to assess the aeroelastic properties of aircraft wings: experimentally or numerically. Testing aeroelastic behavior experimentally is often expensive as creating the models for wind tunnel testing and wind tunnel time itself are valuable. For this reason numerical simulations are often preferred in early design phases. Software for preliminary FSI simulations is generally available in numerical structural solvers like MSC Nastran [15]. The lack of fidelity in the flow physics, however, limits usage of this software to simple flow conditions which are not representative of the flow which is encountered in-flight. In addition to this, the detail design phase requires significantly more fidelity in flow physics. This means that a high-fidelity FSI solver is required in this phase. However, such tools are currently less accessible and more computationally demanding than their low-fidelity counterparts.

To bridge this gap between necessity and availability, this research will focus on creating a high-fidelity FSI framework using Computational Fluid Dynamics (CFD) and Finite Element Method (FEM) software available to the Delft University of Technology. The software will enable the simulation of complex flow physics interaction with clamped structural designs of aircraft wings. The result of these simulations can be used to optimize aerodynamic behavior due to the structure's movement to improve aerodynamic efficiency. Building upon previous work developing high-fidelity frameworks that still had limitations in simulating different problems. This research will also focus on making the framework applicable to a wide variety of simulation cases [12].

2

State of the Art

This chapter outlines the theoretical foundations required to construct a Fluid-Structure Interaction (FSI) framework. The review is structured into three sections FSI-, aerodynamic- and structural modeling. Each of these sections discusses the necessary literature to understand and develop the high-fidelity framework. This chapter also evaluates existing tools which can help streamline framework development.

Aeroelasticity is the science in which the interaction between aerodynamic-, elastic- and inertial forces is analyzed (see Figure 2.1). It describes how a structure (i.e., an aircraft wing) deforms when subject to aerodynamic (and inertial) forces [16]. In flight, aircraft encounter different types of loading which introduce complex load patterns on it's structure. Aircraft are to be structurally optimized to handle these complex loading cases that might occur in-flight. If the structure is not designed to handle these loads, the system could fail during flight. Which could lead to catastrophic disasters, like the Northwest Airlines Flight 710 disaster on March 17, 1960[1].

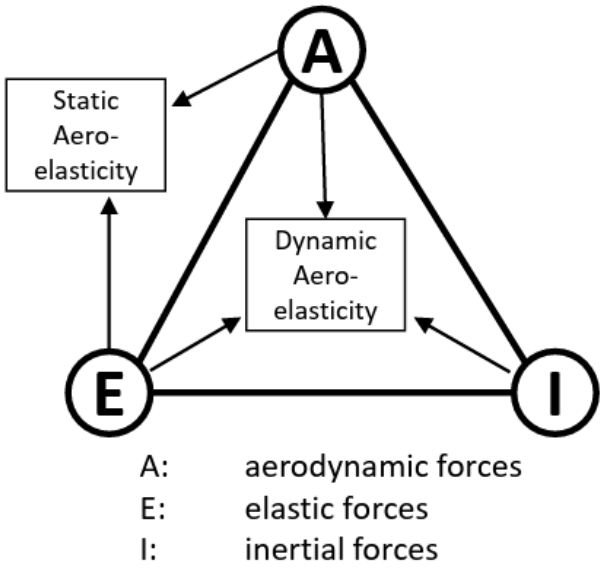

Figure 2.1: Collar's triangle [16].

The main effects of aeroelasticity govern the redistribution of lift over the wings. This

[1]The aircraft lost its left wing in-flight due to destructive wing flutter [17].

redistribution changes the external load distribution over the aircraft, which leads to changes in the aerodynamics, and subsequently the structural behavior of the aircraft [18]. This fight against interactions between the structure and the flow has been present for quite some time. In the early days of aeroelastic research, the main method to avoid aeroelastic failures was to over-design the structure. So reinforcing the structure to the point that engineers are sure the structure can withstand the forces. This often leads to structures that are far too strong, and subsequently heavier than necessary, which is detrimental to the fuel efficiency of the aircraft [19].

Initially, modeling was based on experimental testing of models and preliminary calculations. However, numerical modeling techniques have developed significantly over the years. This development has also led to modeling techniques that combine the structural and aerodynamic design in the science called Fluid-Structure Interaction (FSI). FSI modeling is used to model aeroelastic effects and has reduced the over-designing problem. This is done by tailoring the structure to handle the aerodynamic and inertial forces it is subjected to. The introduction of these modeling methods has made aeroelastic analysis an integral part of aircraft design. Modeling these interactions can be done using different fidelities, depending on the computational cost and time limits [20].

As mentioned, the introduction of the FSI modeling has made a significant positive influence on aircraft design. Over-designing of structures is limited, since more accurate approximations of in-flight loads can be made. However currently, the fidelity of the available FSI solvers is commonly low. This means that the accuracy of the results is limited. These low-fidelity models include assumptions for the aerodynamic and structural conditions which results in less accurate modeling of specific aerodynamic conditions like stall- and compressible conditions, as well as structural modeling effects like non-linear effects [21]. The conditions where these low-fidelity models are valid are therefore shallow. This property of low-fidelity modeling is what underlines the limitations of current FSI modeling software. In order to enable valid modeling of aeroelastic effects in complex conditions, a high-fidelity FSI framework is necessary. The accurate modeling of complex flow phenomena in such high-fidelity FSI framework increases the need for computational power. The general rule of thumb reads, the higher the fidelity of a framework, the more computationally expensive the simulations [22].

2.1. Fluid-Structure Interaction Modeling

To solve the physical problem numerically, the fluid domain Ω_F, structural domain Ω_S, and their shared interface Ω_{FS} must be discretized into computational counterparts. These are defined as $\Omega_{F,num}$, $\Omega_{S,num}$ and $\Omega_{FS,num}$, respectively. This transition from physical domains to their discretized representations (illustrated in Figure 2.2) is a fundamental step that enables the application of numerical solvers to otherwise continuous physical phenomena.

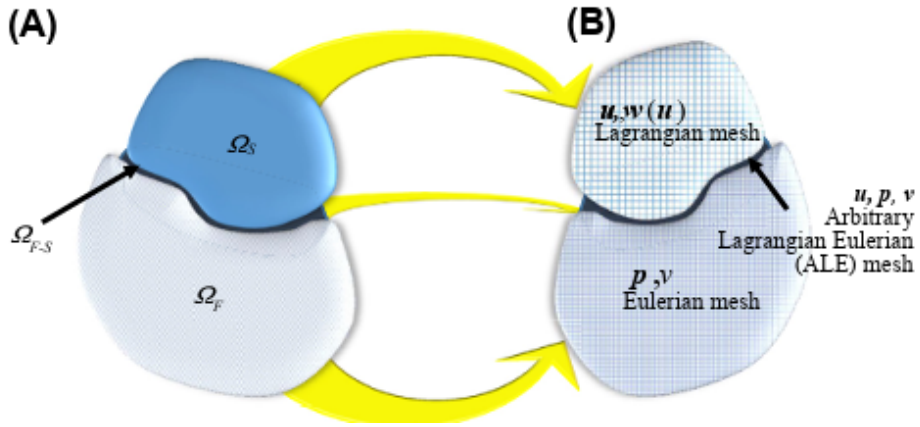

Figure 2.2: Physical and numerical domains of FSI problem [23].

The interaction domain $\Omega_{FS,num}$ is the core to the FSI simulations. This domain is used to model the interaction effects $\Omega_{F,num}$ and $\Omega_{S,num}$ have on each other. The first of these two interaction effects is the force effect of $\Omega_{F,num}$ on the structure domain $\Omega_{S,num}$. In this process the force is obtained by the fluid modeling software, and it calculates the forces on $\Omega_{FS,num}$. These fluid forces are then projected along this interaction domain $\Omega_{FS,num}$ to the structure domain $\Omega_{S,num}$. The forces that are applied on the structure, displace the structure, and with it the structure at the interaction domain $\Omega_{FS,num}$. This displacement of the structure domain $\Omega_{S,num}$ has to be followed by the fluid domain $\Omega_{F,num}$, as the fluid solver has to model the change of the aerodynamic shape. The projection of the displacement of the structure domain to the fluid domain is what enables this aerodynamic shape change. This projection of displacement is thus the second crucial step required for FSI modeling [12, 14].

The meshes of the fluid- $\Omega_{F,num}$ and structural domains $\Omega_{S,num}$ generally do not match one-to-one. This means that the locations at which the results are obtained on both meshes is slightly (or significantly) different. It thus requires interpolation from fluid to structure mesh, and vice versa to project forces and displacements. Figure 2.3 shows an example of fluid- $\Omega_{F,num}$ and structure domains $\Omega_{S,num}$ of which the meshes do not match.

In order to project the displacements and forces, the kinematic- and dynamic interface conditions in Equation 2.1 and Equation 2.2 must be met. The conditions must be satisfied by the fluid- and structure domains to couple the two. Equation 2.1 relates the compatibility between the displacement of both the fluid- and structure domains at the interface, while Equation 2.2 equates the tractions on the wet structure surface to the tractions on the fluid walls.

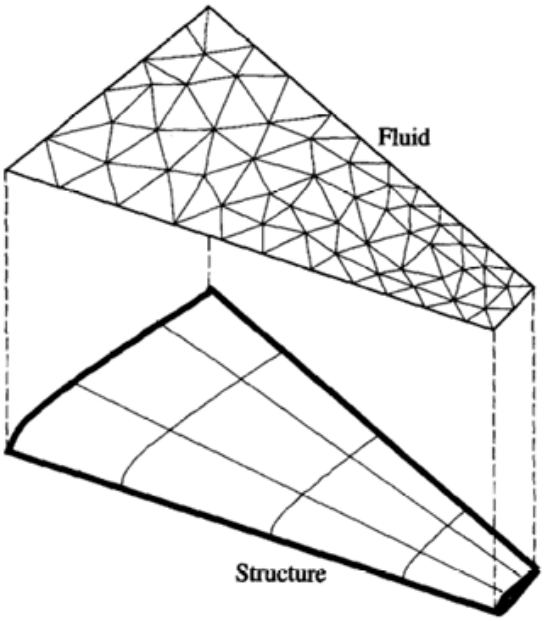

Figure 2.3: Non-matching computational domains [24].

$$\vec{u}_f = \vec{u}_s \tag{2.1}$$

$$p_s \vec{n}_s = p_f \vec{n}_f \tag{2.2}$$

In the development of a FSI framework, the distinction can be made between two types of simulations: static- and dynamic FSI simulations. In static simulations, the aim is to find the steady-state solution of the interaction problem. In contrast, the dynamic simulations focus on finding the transient response of the structure as a result of the loads applied. Each of these framework types has a specific process to solving the numerical problem, which is shown in Figure 2.4 and Figure 2.5 respectively. Static FSI simulations includes only one convergence check. On the contrary, dynamic simulations include a convergence check for every timestep executed in the transient simulations [12]. Moving on, the next sections (section 2.2 and section 2.3) discuss the fluid- and structural modeling theory and software that has to be utilized to generate the static- and dynamic FSI frameworks.

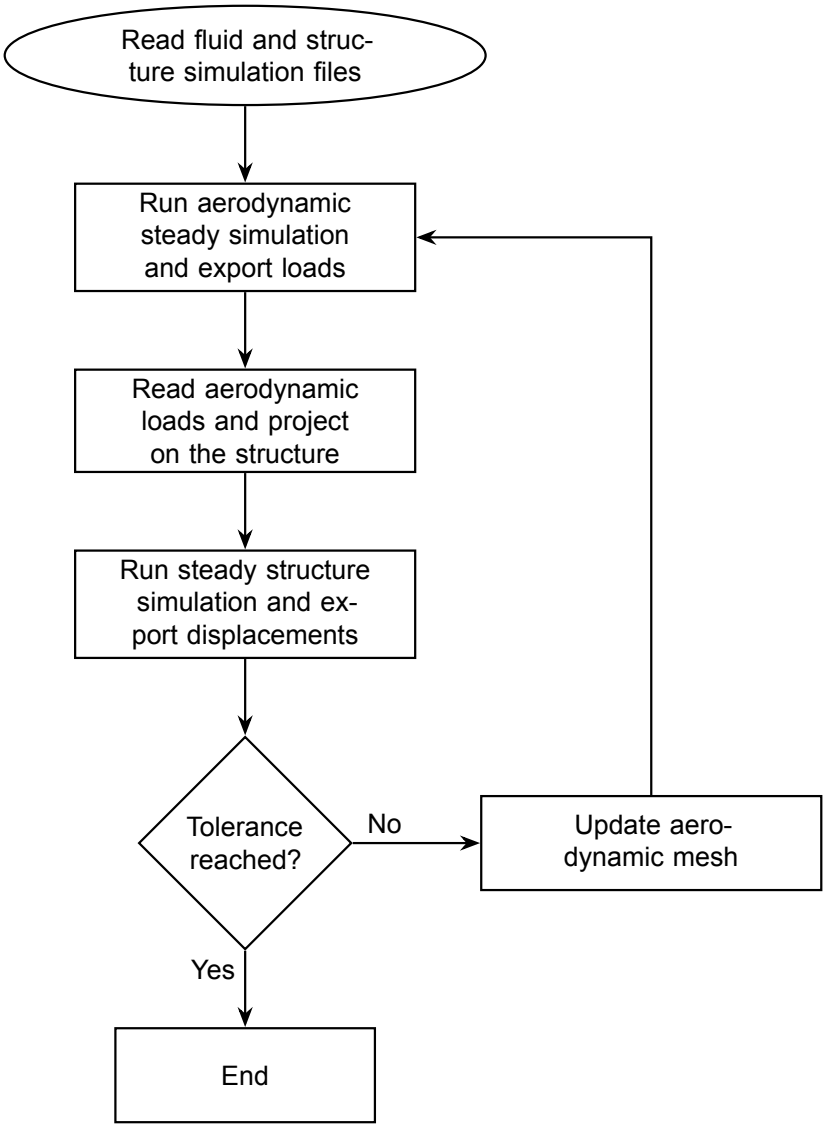

Figure 2.4: Static partitioned FSI framework workflow diagram [12, 21].

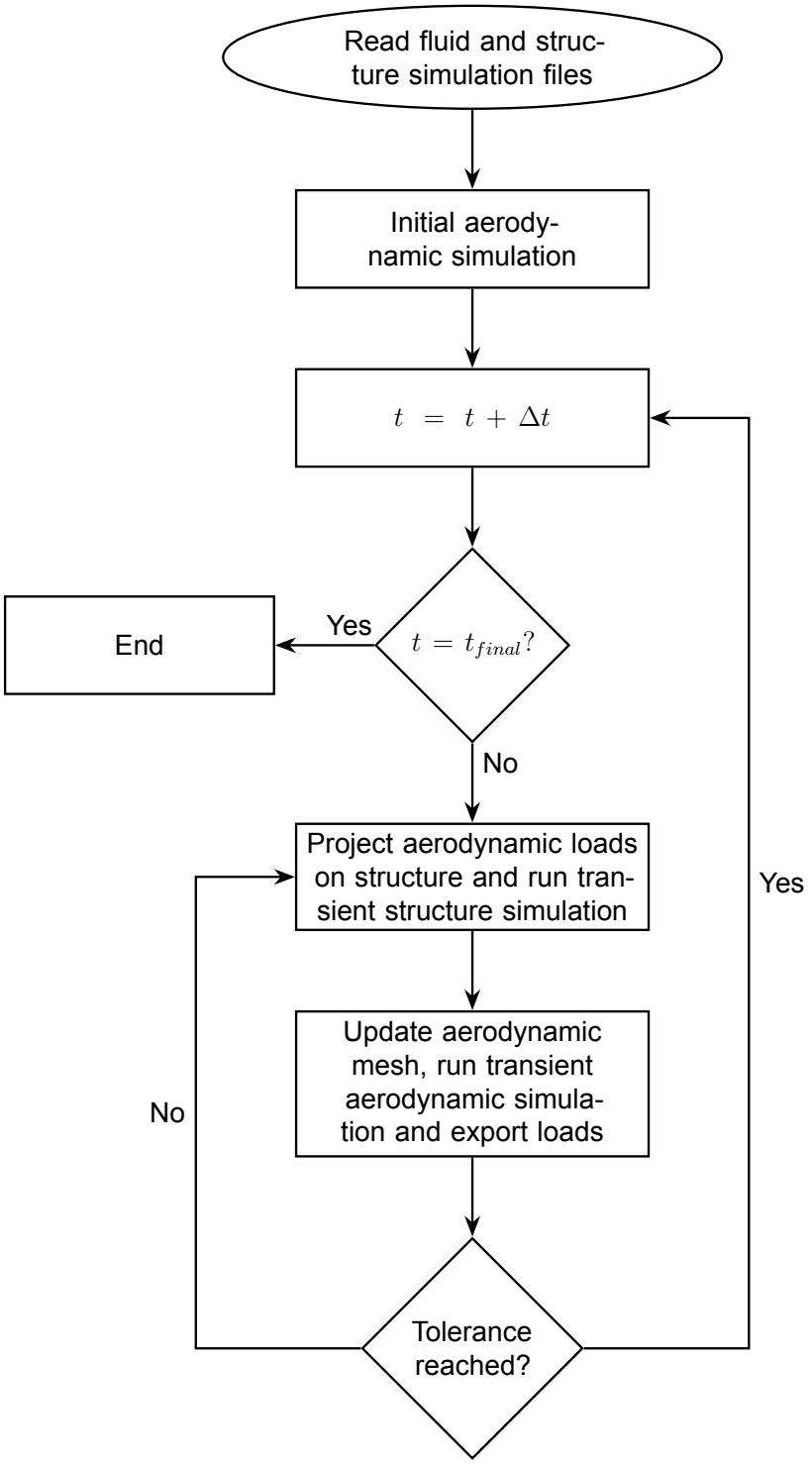

Figure 2.5: Dynamic partitioned FSI framework workflow diagram [12, 21].

2.2. Aerodynamic Modeling

As discussed, the aerodynamics can be modeled using low- or high fidelity solvers. Low-fidelity models are a result of assumptions made in the governing equations. They result in the simplification of the equations, which makes solving them easier. These simpler equations yield a decrease in the computational cost, but also a lack

in physics modeling for certain flow phenomena. This means one might opt for low-fidelity models when the dominant flow phenomena can be modeled by the low-fidelity method or when the computational cost is undesired. Therefore, the choice for low- or high fidelity should be made with care to ensure the most accurate modeling of the flow phenomena whilst staying within computational cost and time limits [22].

An example of low-fidelity aerodynamic modeling in FSI frameworks is the static aeroelastic solver in MSC Flighloads. This framework uses Doublet Lattice Method (DLM) to model the aerodynamics of the flow [21]. The assumptions of DLM modeling mean the aerodynamics is modeled as inviscid, irrotational flow, neglecting shock effects, and the thickness effects of the wing. This low-fidelity method will be used for the comparison between low- and high fidelity models to assess the validity of the different models [25, 26].

Each low-fidelity model is based on the simplification of the Navier-Stokes equations. These Navier-Stokes equations are the governing equations for aerodynamics that do not include any assumptions. Modeling these equations therefore enables a high-fidelity aerodynamic modeling method, and is commonly known as Computational Fluid Dynamics (CFD) modeling. In order to obtain the flow results of this type of modeling, the continuous domain has to be discretized into smaller sections by generating a mesh[2]. This combined with numerical schemes that are used to solve the governing equations, influence the accuracy and computational cost of the simulation. Additionally, the accuracy of the simulation is also influenced by the modeling of turbulence, which is done by so-called turbulence models. These models simulate the turbulence in places where the mesh is not fine enough to resolve the turbulent scales [12, 25].

As mentioned before, CFD solves the Navier-Stokes equations. These equations consist of the conservation of mass, momentum, and energy for a given numerical problem [1, 25]. These numerical problems consists of the mesh, flow properties, numerical methods and boundary conditions. The method commonly used in steady aerodynamic problems to solve the Navier-Stokes equations is called Reynolds Averaged Navier-Stokes (RANS). In this method any instantaneous fluid property is split into two parts, a mean value and a fluctuating value. This means that if the average fluctuations are taken over a long time, the fluctuations are equal to zero. This method results in the average solution rather than instantaneous solutions. In the derivation of the Navier-Stokes equations for the RANS applications a new term, the Reynolds Stress, appears. The introduction of this extra term requires a closure model to enable solving the Navier-Stokes equations. This closure model consists of equations resolving the turbulence of the flow. Transient CFD simulations require a different approach in solving the governing equations such as Unsteady Reynolds Averaged Navier-Stokes (URANS) or Large Eddy Simulations (LES). More about solving transient simulations can be found in relevant literature [25, 27].

Solving the governing equations in CFD is generally done using the Finite Volume Method (FVM). In FVM the mesh cells describe the control volumes of the numerical domain. In each of these control volumes the governing equations are solved to find the flow properties. The resulting flow properties describe the average values of the

[2]The mesh is the numerical representation of the physical problem.

flow variables over each control volume. This method is 2nd order accurate, which is considered enough for engineering applications as it provides a balance between accuracy, computational cost and stability [1, 12].

In steady simulations the solver iterates over the results of the control volumes multiple times using numerical schemes until a residual threshold or maximum number of solver iterations is achieved. In transient or unsteady simulations, an additional time integration is necessary. These simulations yield the results of the flow as a function of time [25, 27].

The Navier-Stokes equations, also referred to as the governing equations, ensure that physical conservation principles are maintained across the entire domain. This forms the mathematical foundation upon which the accuracy of the FSI framework depends. The first of the governing equations to be solved is the conservation of mass, given by Equation 2.3. The first term of this equation describes the time rate of change of mass, this describes how much mass is building up or decreasing inside the control volume over time. The second term of this equation describes the sum of mass flowing in through the boundaries minus the mass flowing out through the boundaries of the control volume. Finally, the last term represents the mass added or removed from the control volume by a source or sink [1, 25].

$$\frac{\partial}{\partial t}\int_V \rho \, dV + \oint_A \rho v \cdot n \, dA = \int_V S_m \, dV \tag{2.3}$$

Equation 2.4 represents the conservation of momentum. The first term is the time rate of change of momentum. The second term describes the convection of momentum, this is the momentum carried across the cell faces by the flow of fluid though the mesh. The third term describes how the pressure acts on the faces of the control volumes. Then, the fourth term describes the viscous/shear force exerted by the fluid of the surrounding control volumes on the control volume surface. The last two terms show the contributions of the body forces (gravitational and other). These forces act on the entire mass within the control volume [25].

$$\frac{\partial}{\partial t}\int_V \rho v \, dV + \oint_A \rho v (v \cdot n) \, dA = -\oint_A pn \, dA + \oint_A \bar{\bar{\tau}} \cdot n \, dA + \int_V \rho g \, dV + \int_V F \, dV \tag{2.4}$$

If the flows include heat transfer and compressibility effects, CFD software additionally solves the energy conservation equation. This equation is given by Equation 2.5 where the total energy per unit mass E is given by Equation 2.6. The total energy consists of the internal, kinetic and potential energy present in a control volume. The first term of this equations describes the change in total energy per unit mass in a control volume over time. The second term describes the energy moved out of the control volume by the fluid flow. The third term is the pressure work, which describes the work done on the fluid moving past the cell faces by the pressure. Then, the fourth term defines the energy transport through the control volumes. This, for example, could be

due to combustion or another chemical process. Finally, the last term shows the contribution of volumetric heat sources, such as contributions of chemical energy release, radiation, et cetera [1, 25].

$$\frac{\partial}{\partial t}\int_V \rho E\,dV + \oint_A \rho E(v\cdot n)\,dA + \oint_A p(v\cdot n)\,dA = -\oint_A \left(\sum_i h_i J_i\right)\cdot n\,dA + \int_V S_h\,dV \quad (2.5)$$

$$E = h - \frac{p}{\rho} + \frac{v^2}{2} \quad (2.6)$$

As mentioned, turbulence equations are necessary to resolve the Reynolds stress as required for the closure model. This is in addition to the mass, momentum and energy conservation equations. Different turbulence models can be used to resolve the Reynolds stress, and the selection of a turbulence model is often a result of a comparison between numerical simulations of different turbulence models and experimental data. When experimental data is not available, however, turbulence models can be selected based on the properties of each model. Table 2.1 shows the properties of commonly used turbulence models in RANS simulations. The category indicates how many transport equations need to be solved to obtain the Reynolds stresses. The more equations, the more complex the turbulence model. This means for every extra equation, the equation can get more accurate, but also gets more computationally costly and carries bigger risk of decreased stability [1, 25, 27].

Table 2.1: Common RANS Turbulence Models and Applications [25, 12].

Model	Category	Primary Application	Key Limitation
Spalart-Allmaras	1-Eq	External Aero, Wings, Fuselages	Poor internal flow accuracy
Standard k-ϵ	2-Eq	HVAC, Large Industrial Flows	Fails at predicting separation
k-ω SST	2-Eq	General Aero, Flow with Separation	Requires fine mesh

2.2.1. Ansys Fluent Software

In this research the software that will be used to solve the Navier-Stokes equations is Ansys Fluent. This software is available through Delft University of Technology and is commonly used by the university in aerodynamic research. Ansys Fluent is a commercial software package ane has a wide variety of built-in capabilities that can be used to streamline the development of the high-fidelity FSI framework [27].

Running aerodynamic simulations in Fluent requires information about settings for mesh, flow properties, numerical methods, and boundary conditions. The settings, numerical methods, and governing equations differ slightly for steady and transient simulations, more about this can be found in relevant literature by ANSYS Inc. [27].

The results of the steady and transient simulations in Fluent generally consist of the flow variable results of the variables present in the governing equations in each control volume. This results in either the steady state solution or a solution in time. When flow results are desired that are not included in this list, one can opt to use User-Defined Functions (UDFs). This method can be used to obtain the forces on each wing surface element. In addition to using UDFs for requesting specific data, they can also used to supply Fluent with information required for methods within Fluent. One of the methods within Fluent that uses UDFs to supply this information is the dynamic mesh method. As discussed in section 2.1, the aerodynamic mesh should change as a result of the structural deformation. In order to change it, Fluent's dynamic mesh method can be used. The UDF required to execute this dynamic mesh method has to give Fluent the information about the updated shape of the aerodynamic mesh [28].

Each of the discussed UDFs has their own type [29]. The types that will be most relevant for this research are the Define on Demand and Define Grid Motion UDFs. The Define on Demand UDF can be used to execute the script at a moment that is desired by the user, e.g., after a certain amount of solver iterations have passed. This type of UDF is relevant for the force extraction as this should be done after the CFD simulation is considered converged. The Define Grid Motion UDF is relevant for assigning the deformation of the aerodynamic mesh for each FSI iteration. The implementation as well as the process of each of these UDFs is discussed further in the reports written by ANSYS Inc. [27, 29].

Running simulations using Fluent is generally executed through the Fluents Graphical User Interface (GUI). When running a FSI simulation however, the manual usage of the GUI is not desired as this is not efficient as it requires significant human intervention. In order to keep the high-fidelity FSI framework efficient, the process has to be scripted. In order to do this, one can use the Python library called PyFluent. This library has capabilities to access Fluent functionalities through Python. It can be used to automate, customize and streamline Fluent with other libraries to form a strong tool such as a high-fidelity FSI framework. PyFluent connects to Ansys Fluent through a Python API. This enables functionalities in PyFluent through text-user interface (TUI) commands or the settings API. This connection allows for the editing of the simulation setup, execution, monitoring, and result extraction [30].

2.3. Structural Modeling

Finite Element Method (FEM) is a method commonly used when simulating the steady or transient response of the structure to loading. The method divides the physical problem into nodes which are interconnected by elements of different types and materials. The type of element and material determines the deformation and load behavior of the structural model. Each element type has its specific applications and assumptions. This means the appropriate elements should be used, as incorrect elements will lead to inaccurate modeling of the structure. The element types include different truss, beam, and shell elements, and a complete list of available element types can be found in the FEM software manual by MSC Software [15]. Some elements relevant for this research and their application can be found in Table 2.2.

Table 2.2: FEM elements [15].

Element	Name	Nodes	DOFs/Node	Total DOFs	Stiffness Size	Primary Use
3D Truss	CROD	2	3	6	6×6	Axial members
3D Beam (Euler)	CBAR	2	6	12	12×12	Slender frames
3D Beam (Timoshenko)	CBEAM	2	6	12	12×12	Thick beams
4-Node Shell	CQUAD4	4	6	24	24×24	Plates/panels

The element type does not only yield the stiffness matrix, but also the shape of displacement vector u. Some elements such as the truss elements only allow for translation (in 3D), this means that per node these truss elements only have 3 Degrees of Freedom (DOFs). As truss elements (CROD) consist of 2 nodes, this means the displacement vector of such element has 6 total DOF's. However, elements such as 4-Node Shell (CQUAD4) consist of 4 nodes with each node translation (in 3D) and rotations (in 3D), meaning 6 DOFs per node (24 DOFs total). So elements such as CQUAD4 lead to larger matrices and vectors compared to CROD elements, and thus result in more computationally expensive simulations [15, 31].

The nodes that are connected by each of these elements also encounter forces and moments. These are given by the load vector F. This is generally defined by the user to obtain the displacement vector. The forces are often a result of prior analysis or measurements. The force vector can also be obtained by giving the displacement vector. In this research these forces and moments on the structure will be a result of the aerodynamic forces and moments of the CFD software [29, 15, 31]. The displacement of the structure will then be used to change the CFD mesh to match the wing deformation subsequently as was discussed in section 2.1.

The established global stiffness matrix, consisting of the combination of the individual stiffness matrices of the elements, has a null space corresponding to rigid body motions. This means that the matrix is singular, so the stiffness matrix cannot be inverted. However, the governing equation for linear static simulation Equation 2.7 requires the invertible property to be solved for the displacement vector u. Boundary conditions are thus applied to make the matrix invertible. Subsequently Equation 2.8 can be used to solve for the structural displacements. This results in a set of translations and rotations of the structural grid nodes [15, 31].

$$[K]\{u\} = \{F\} \tag{2.7}$$

$$\{u\} = [K]^{-1}\{F\} \tag{2.8}$$

The linear static method given by Equation 2.7 can be used to obtain the steady-state behavior of a structure. However, in certain situations the behavior of the structure is desired to be known as a function of time, for instance in dynamic FSI simulations. In these cases dynamic FEM simulations are often chosen, and the governing equation is given by Equation 2.9. In this equation, an additional damping matrix [C] and mass matrix [M] are present. The mass matrix is used to simulate the inertial effects of the

structure, whilst the damping matrix is used to model the energy dissipation of the structure. The mass and stiffness matrix are constructed through material and geometry properties, however, there is no universal way to construct the damping matrix. A method commonly used to construct the damping matrix is the Rayleigh damping. Here, the damping matrix is a function of the mass and stiffness matrix as given in Equation 2.10. In this equation α damps the low-frequency modes and β damps the high-frequency modes. The $\{\dot{u}\}$ and $\{\ddot{u}\}$ represent the single and double time derivative of the displacement vector, thus velocity and acceleration of the structure [31].

$$[M]\{\ddot{u}\} + [C]\{\dot{u}\} + [K]\{u\} = \{F(t)\} \tag{2.9}$$

$$[C] = \alpha_{damp}[M] + \beta_{damp}[K] \tag{2.10}$$

The structure solvers are equipped with numerical schemes to solve these transient governing equations. A method commonly used as default by FEM solvers to solve an equation like Equation 2.9 is the Newmark-β method. This method is an implicit time integration method to obtain the displacement vector at the next timestep. The advantages of this method is the unconditional stability when given proper parameters. Additionally, the method is $2^n d$ order accurate, its robust for stiff systems, and it is well-established. In contrast, the disadvantage of this method is the need for matrix factorization. This makes the method computationally expensive and not efficient for non-linear problems [12, 15, 31].

The governing equations discussed above model the structure as a linearly deforming structure. This means that an assumption is made that the loads are purely imposed on the initial structure and the materials behaves linearly. However, non-linear modeling of the structure is more realistic. This models the load application on the changing structure and non-linear behavior of the material. Non-linear effects are most significant when structures deform significantly. In situations where the deformation is minimal, it is more useful to use linear analysis as non-linear analysis is significantly more computationally expensive [12, 15].

The governing equations for these static and dynamic non-linear simulation models are given by Equation 2.11 and Equation 2.12 respectively. In these equations the stiffness of the structure is dependent on the displacement vector of the structural nodes. These non-linear equations are more complex to solve due to this change in stiffness. The static equation requires an iterative approach to find the equilibrium solution of the displacement vector. The dynamic equation needs an additional time integration besides the iterative approach. Methods used to solve these non-linear governing equations can be found in structural modeling software documentation written by MSC Software [15] and Megson [31].

$$[K(u)]\{u\} = \{F\} \tag{2.11}$$

$$[M]\{\ddot{u}\} + [C]\{\dot{u}\} + [K(u)]\{u\} = \{F(t)\} \tag{2.12}$$

In FEM simulations, there are two prominent methods to model the structure. These methods are a beam-like and box-like representation of the structure [12]. In beam-like representation a wingbox is represented by an one-dimensional line as shown in Figure 2.6. This line is located along a specified reference axis. Beam-like models can carry bending, shear, torsional, and axial deformation. This modeling type, however, only gives information about the general loading of the structure. Information about the local loads in the wingbox are not obtained. In contrast, a box-like representation (see Figure 2.6) models the complete wingbox structure. It can include a wide variety of elements, each with its own load carrying properties, and does give information about local loading in the structure [12].

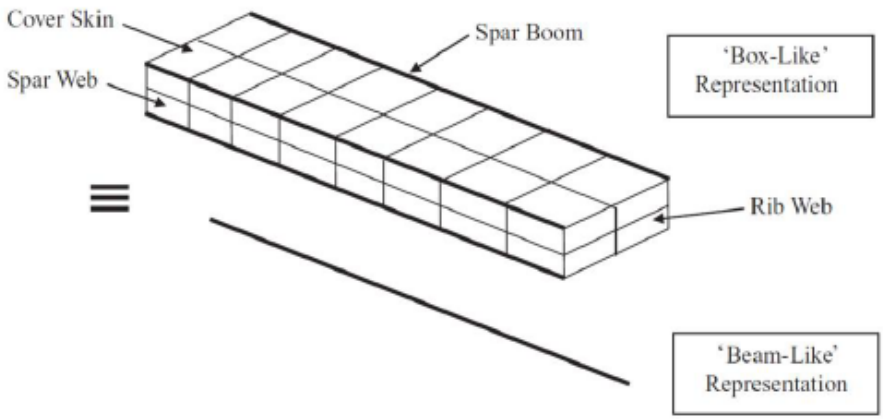

Figure 2.6: Beam and box-like structural representation of a wingbox [19].

Whilst beam-like modeling is generally less computationally costly than box-like modeling, it is suboptimal. As mentioned before, box-like modeling allows for local structural optimization, where beam-like does not. This local optimization can be seen as reinforcing the structure only where it is needed. It thus allows for a minimization of the structural weight by selecting specific materials and structure geometries locally [19].

2.3.1. MSC Nastran Software

The structural component of the FSI framework will be developed using MSC Nastran, a high-fidelity Finite Element Method (FEM) solver widely utilized within the Aerospace Engineering faculty at Delft University of Technology. In this workflow, structural models are pre-processed in Patran to generate a Bulk Data File (BDF). This human-readable text file serves as the primary interface between the modeler and the solver. It encapsulates all critical simulation parameters, including nodal coordinates, element definitions, material properties, and the specific boundary conditions and loads required for the analysis [15]. This BDF is then loaded into the MSC Nastran solver executable. This executable translates the information in the BDF into the matrices and vectors required as input to the governing equations. The solving of the governing equations is then started with the selected numerical methods. When the simulation is finished, the results are in the form of the translation and rotation of the nodes, stresses and strains in the elements, or as requested by the user [15].

The FSI framework should edit the structural model automatically as a result of the forces applied to the structure. Human interaction at each iteration is not desired as this is time-consuming, impractical and defeats the purpose of fast iteration of aeroelastic design tailoring to flight loads. This means the FSI framework should automatically edit the BDF through a pre-programmed algorithm. Therefore, the modification of the BDF files can be executed through the pyNastran python library. This Python library is made to edit BDF similar to what the user would do through Patran [15, 32].

2.4. HPC Cluster TU Delft

The significant computational demands of high-fidelity FSI simulations often exceed the processing capabilities of standard workstations. This necessitates the use of distributed computing resources. Consequently, a key requirement for the framework is its compatibility with high-performance environments. This research leverages the Delft University of Technology High-Performance Computing (HPC) Cluster. It is a shared infrastructure that provides the parallel processing power essential for handling the large-scale numerical grids and complex physics inherent in aeroelastic modeling.

The operating system on the cluster is CentOS 7. CentOS 7 is a server class Linux distribution which requires Portable Bash System (PBS) files to run simulations. In these files different parameters such as the name of the queue, number of nodes, number of processors per node and walltime are required. Each of these queues has different computational capabilities. The user sets the desired queue name and submits the PBS file to the cluster. The cluster then runs the job submission once the previous other jobs in the queue are finished [33]. Besides the required parameters, the PBS file must contain the actual script and/or programs that must be run on the cluster. In the case of this FSI framework this will be a combination of Ansys Fluent, MSC Nastran and a Python script to interact between the fluid and the structure solver. This Python script is what will form the actual FSI framework. Setting up the PBS file to run the actual simulations on the cluster is what will be further discussed in the Github repository [3].

[3]The Github repository for this research can be accessed through the following link: `https://github.com/Lennarth1998/master_thesis_fsi`

3
Objectives and Structure

As established in the existing literature, significant effort has been dedicated to fuel efficiency. Specifically through innovations in airframe weight, propulsion systems, and aerodynamics. These trends, especially the one for increasing AR, have led to increasingly more flexible aircraft wings [11]. This structural flexibility of the wing has thus made way for a more prominent role of aeroelastic analysis in aircraft design.

In order to accurately analyze the aerodynamic behavior of these flexible aircraft, low-fidelity models are not accurate enough as they neglect important flow characteristics. The lack of fidelity has proven that there is a need for high-fidelity FSI frameworks to model these aeroelastic effects in flight. This lack of fidelity in modeling complex flow phenomena and their impact on structural behavior is what drives this research.

3.1. Objectives and Scope

After research into the literature regarding Fluid-Structure Interaction modeling, the main objective of this master research can be formulated as follows:

- Develop a high-fidelity Fluid-Structure Interaction framework to model aeroelastic behavior of a (clamped) wing model.

This main objective can be split into two research questions which will be answered in chapter 5 and chapter 6 respectively.

1. How can a high-fidelity partitioned FSI solver be developed through Python scripts?
2. For which conditions does the high-fidelity FSI solver provide more reliable results compared to low-fidelity?

The simulations in this research will be run using the Onera M6 wing model. This model is a classic CFD validation case for external flows due to its simple geometry combined with complex transonic flow phenomena. This combination makes the model an ideal test case for the FSI framework [34]. Furthermore, the scope of this project is the testing of aerodynamic simulations' influence on the structure. The framework is made to simulate clamped wing models as tested in wind tunnels. This means free-flying aircraft structures are not accurately simulated with this framework. Moreover, the structural model that is simulated should not include the wing skin, solely the wingbox structure.

3.2. Report Structure

This report consists of 7 chapters, these chapters resemble the order of steps taken during the research and development. The content of each chapter is shortly discussed below.

- Chapter 1 introduces the significant trends in the aviation sector. The most significant trend being the increased flexibility of aircraft wings as a result of AR increase. This flexibility of the wings and the need for aerodynamic efficiency is what drives the needs for high-fidelity Fluid-Structure Interaction solvers.
- Chapter 2 discusses the relevant literature which is required to develop the high-fidelity FSI framework. The literature governs the theory of fluid- and structure modeling, as well as mesh metrics used to assess the accuracy of simulations. Finally, the chapter describes the methods that will be used to model the force and displacement interaction.
- This chapter discusses the justification and motivation for this research as a result of the literature studies. Additionally, it discusses the objectives, scope, and the structure of the project. This chapter concludes the introduction of the report.
- Chapter 4 covers a description of the wing model that is used for FSI simulations in this report. The description includes an in-depth explanation of the aerodynamic and structural simulation models as well as expected behavior in the FSI simulations.
- Chapter 5 discusses the algorithm of the static FSI framework. This chapter discusses the methodology for the static FSI framework answering the first research question discussed above. The methodolgy consists of building blocks which combined form the complete methodology of the static framework. It is paramount to ensure every building block works before all blocks are integrated. Checking the building blocks will be done through intermediate sanity checks and diagrams will be used to explain the steps to completing each of the building blocks.
- Chapter 6 will present and discuss the static simulation results of the simulation models described in chapter 4. This includes the comparison of the high-fidelity simulations to the low-fidelity models for different simulation cases. The simulation results which will be compared are the Onera M6 static simulations. The presentation and discussion of results will be used to assess the second research question posted above.
- Chapter 7 finalizes this report with the conclusions of the research based on the research questions. Secondly, this chapter will discuss recommendations for additional research. The recommendations are based on methods that were too grand to implement due to time restrictions or were not in the scope of this research.

4
Research Models

Developing a partitioned Fluid-Structure Interaction (FSI) framework requires an integration of both aerodynamic and structural modeling. This chapter addresses the experimental and numerical models for the Onera M6 wing used in proceeding FSI simulations. This description includes the discussion of the computational architecture, and the expected behavior of these models in the FSI framework.

4.1. Experimental Model

The Onera M6 model, shown in Figure 4.1 and Figure 4.2, was developed by the *Office National d'Études et de Recherches Aérospatiales* (ONERA) in the 1970s and remains a cornerstone for validating aerodynamic solvers against experimental data. As detailed in Table 4.1, its geometry is designed to induce a transonic flow regime characterized by complex phenomena, including local supersonic regions, shock waves, and turbulent boundary layer separation [34, 35]. While these features provide a rigorous test for solver accuracy, they often exceed the capabilities of low-fidelity FSI frameworks. These frameworks typically fail to capture such effects due to simplifying assumptions in their governing equations [21, 22]. Consequently, the Onera M6 serves as a critical benchmark for quantifying the performance gap between low- and high-fidelity numerical approaches.

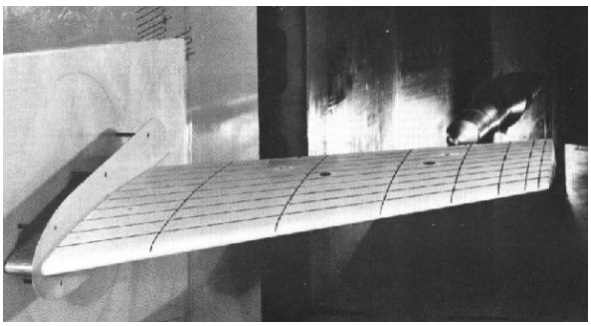

Figure 4.1: Onera M6 windtunnel model [34].

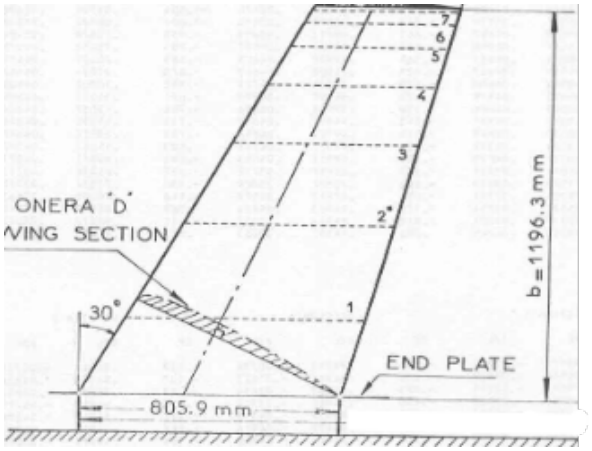

Figure 4.2: Onera M6 wing planform [34].

Table 4.1: Onera M6 wing geometry specifications [34].

Span, b	1.1963 m
Wing Area, S	0.723 m^2
MAC, c_{mean}	0.64607 m
AR	3.8
Taper Ratio, λ	0.56
LE Sweep, Λ_{LE}	30.0 deg
TE Sweep, Λ_{TE}	15.8 deg

The experimental model includes chordwise and spanwise lines that indicate the locations of pressure taps used to measure static surface pressure. These experimental measurements serve as the baseline for validating the numerical aerodynamic model. The experimental case used to gather data is provided by Schmitt and Charpin, featuring a Mach number M of 0.8395 and a Reynolds number Re of $11.72 \cdot 10^6$ at an angle of attack α of 3.06. The ambient conditions are defined by an inflow pressure of $80,600\ Pa$ and a temperature of $255.56\ K$ [34].

4.2. Aerodynamic Model

The aerodynamic model consists of a numerical model which simulates the experiment discussed in section 4.1. Modeling the experiment numerically starts by translating the experimental domain into a numerical domain. This process begins with a Computer Aided Design (CAD) model of the wing. The half-span CAD model used to generate the aerodynamic model for this research is shown in Figure 4.3.

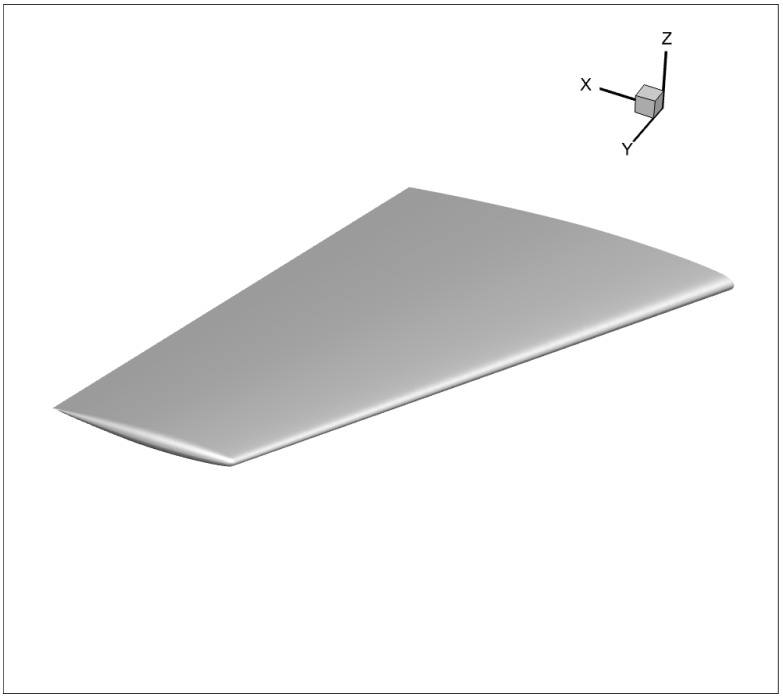

Figure 4.3: Onera M6 wing half-span CAD model [34].

To accurately simulate the airflow around the Onera M6, a fluid domain is established with dimensions of approximately $80 \cdot c$ in the x and z-directions and $40 \cdot c$ in the y-direction. This substantial volume (illustrated in Figure 4.4) is sized specifically to prevent far-field boundary conditions from artificially influencing the flow physics near the wing. While the fluid zone represents the internal volume of this domain, its boundaries are defined by a series of surfaces: the far-field and symmetry zones (shown in orange and green, respectively) and the wing zone itself. As shown in Figure 4.5, the wing surface is further discretized into three distinct regions: the top (blue), bottom (red), and side (yellow) zones. The side zone, located at the trailing edge, is particularly critical as it closes the gap between the upper and lower surfaces. Ensuring this geometric closure is a vital prerequisite for the meshing process, the details of which are explored in the following subsection.

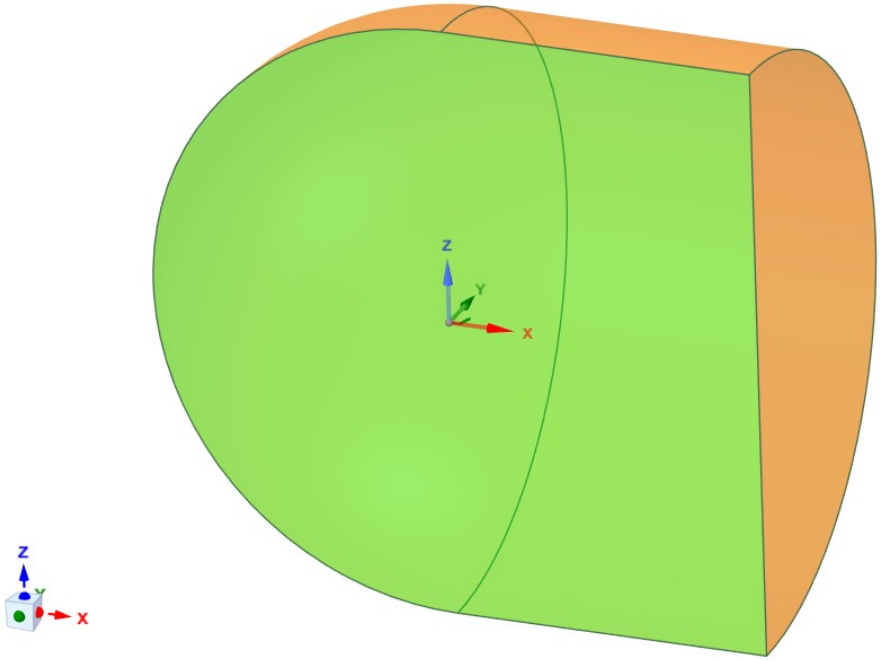

Figure 4.4: Computational domain symmetry- and far-field boundaries.

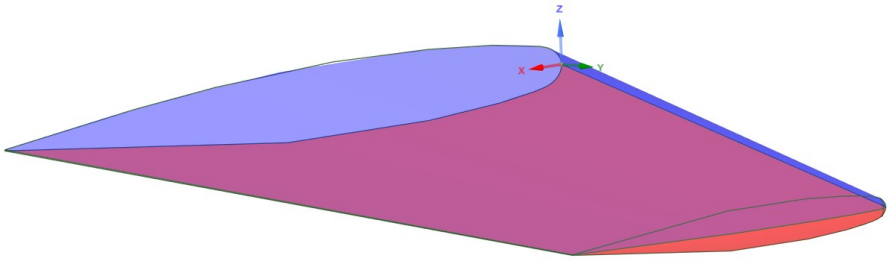

Figure 4.5: Computational domain wing.

4.2.1. Ansys Fluent Meshing

Following the generation of the computational domain, the surfaces and volumes have to be converted into a mesh. The mesh enables the CFD software to solve the gov-

erning equations discussed in section 2.2. The generation of a mesh or meshing is a step which differs for each software. The computational domain that was created using the Onera M6 wing model provides a continuous, manifold geometry without gaps. The Watertight Geometry workflow is the most efficient meshing approach for these types of computational domains according to Ansys Fluent documentation [27]. Upon importing the computational domain into Fluent, the discretization process is divided into two primary configurations: surface and volume mesh generation. The surface settings define the resolution for the wing zones, the symmetry- and the far-field zone. Whereas the volume settings encompass the boundary layer inflation layers, which are critical for resolving near wall gradients, as well as the global mesh parameters for the remaining fluid domain [27].

Even though volume mesh cell types such as poly-hexcore offer high computational efficiency, their topological complexity restricts the allowable limits of mesh deformation. Since during FSI simulations the deformation of the structure, and subsequently the fluid domain can be significant, it is desired the allowable limits of mesh deformation are maximized. In contrast to complex mesh cells, simpler geometric volumes such as tetrahedral cells provide the robustness necessary to accommodate the large deformations inherent in high-fidelity FSI simulations. The higher allowable deformation limit of tetrahedral cells is a result of their simple geometry and Fluent's ability to keep track of the shape of this simple cell. The complexity of poly-hexcore cause the mesh cells to invert easier than tetrahedral cells. This inversion of the mesh cells leads to negative volumes in the cell mesh which give fatal errors when changing the CFD mesh locations. To maximize the allowable limits of CFD mesh deformations, a tetrahedral volume mesh is used in the generation of the CFD model. Achieving equivalent numerical accuracy with tetrahedral elements, however, typically requires a significantly higher cell count [12, 27]. The impact of the mesh cell type and additional settings on the accuracy of the results is analyzed in subsection 4.2.3.

4.2.2. Ansys Fluent Flow Setup

Following the generation of the mesh, the flow properties are assigned to the mesh zones. The first step in this process involves selecting an appropriate turbulence model. Given the presence of shock-induced flow separation of the Onera M6 wing, the k - ω Shear Stress Transport (SST) model is selected for its superior performance in simulations involving flow separation [34]. As discussed in section 2.2 the usage of the k - ω SST turbulence model does require a sufficiently refined mesh. In subsection 4.2.3 a convergence studies will be performed to investigate how refined the mesh should be to achieve sufficiently accurate results.

Boundary conditions are assigned to the mesh zones to define the physical environment of the simulation. The far-field boundary conditions establish the inflow state, which is configured by specifying the Mach number, temperature, and pressure values, alongside the turbulence intensity and viscosity ratios (see Table 4.2). For the XZ-plane, a symmetry boundary condition is applied to effectively model the semi-span of the wing. Using the symmetry boundary condition enables modeling of half of the wing, as the wing is symmetrical in this plane. This reduces the computational time significantly whilst achieving accurate results. Lastly, the wing surfaces are assigned

no-slip wall conditions, the parameters of which are listed in Table 4.3[1].

Table 4.2: Fluent inflow boundary condition settings [34, 27].

Parameter	Value	Unit/Method
Momentum		
Gauge Pressure	0	Pa
Mach Number	0.8395	–
X-Component of Flow Direction	0.998574	–
Y-Component of Flow Direction	0	–
Z-Component of Flow Direction	0.053382	–
Turbulence		
Specification Method	Intensity and Viscosity Ratio	
Turbulent Intensity	5	%
Turbulent Viscosity Ratio	10	–
Thermal		
Temperature	255.56	K
Operating conditions		
Pressure	80600	Pa

Table 4.3: Fluent wall boundary condition settings [34, 27].

Parameter Group	Setting / Value
Motion Type	Relative to Adjacent Cell Zone
Shear Condition	No Slip
Wall Roughness	Standard Roughness Model
Roughness Height	0 m
Roughness Constant	0.5

4.2.3. Convergence Studies

To ensure the validity of the numerical results, a grid convergence study was conducted by comparing three mesh densities; coarse, medium and fine. They are compared against experimental pressure tap data. The discretization accuracy is primarily governed by the surface mesh size (S_{min} and S_{max}) and the maximum volume cell length (l_{max}). The specific parameters for these three levels are detailed in Table 4.4.

[1]The CFD simulation file, including all settings can be found on the Github repository accessible through the following link `https://github.com/Lennarth1998/master_thesis_fsi`

Table 4.4: Mesh settings convergence studies.

Mesh	$S_{min}\ [mm^2]$	$S_{max}\ [mm^2]$	$l_{max}\ [mm]$	$N_{cells}\ [-]$
Coarse	4	2000	1024	2567153
Medium	3	1500	768	3084512
Fine	2	1000	512	5062420

The simulated pressure coefficients (C_p) were evaluated at seven spanwise locations. The pressure coefficients were calculated using Equation 4.1. Where p is the local pressure, p_∞ is the gauge pressure, ρ_∞ the free stream air density, and V_∞ the free-stream fluid velocity.

$$C_p = \frac{p - p_\infty}{\frac{1}{2}\rho_\infty V_\infty^2} \tag{4.1}$$

The convergence studies results illustrated in Appendix A show that all three mesh densities yield similar chordwise C_p distributions. These pressure results are used together with the force-coefficients to select a CFD mesh which will be used for subsequent FSI analyses. When comparing these results to the reference data, however, it can be seen that the modeling error is significantly larger than the discretization error. This difference is a result of the simulation solving the equations that do not exactly represent the physical problem. It can however, be concluded that key flow features such as suction peaks, and separation points align closely with the experimental reference data. This indicates that the CFD model effectively captures the underlying physics.

While the pressure distributions show good agreement, the final mesh selection is based on the convergence of the integrated force coefficients. A common heuristic for mesh independence is a variation of $\leq$ 1.0% in force coefficients relative to the finest mesh force coefficients ($C_{L,fine}$ & $C_{D,fine}$). This threshold balances the numerical accuracy of the simulation with computational efficiency [12]. The relative lift and drag coefficients of the three mesh densities are presented in Table 4.5. The medium mesh results fall within the $\leq$ 1.0% tolerance for both the lift- and drag coefficient. Consequently, the medium mesh density provides sufficient accuracy for the objectives of this study and will be utilized for all subsequent FSI simulations.

Table 4.5: Relative convergence force coefficients.

Mesh	$C_L/C_{L,fine}$	$C_D/C_{D,fine}$
Coarse	1.013	1.011
Medium	1.010	1.010
Fine	1.0	1.0

4.3. Structural Model

The structural model was derived by extracting surface nodes from the Onera M6 CAD model to create a reference shell. This shell served as the geometric boundary for

constructing the internal wingbox assembly. As illustrated in Figure 4.6, the resulting model represents the primary load-carrying structure.

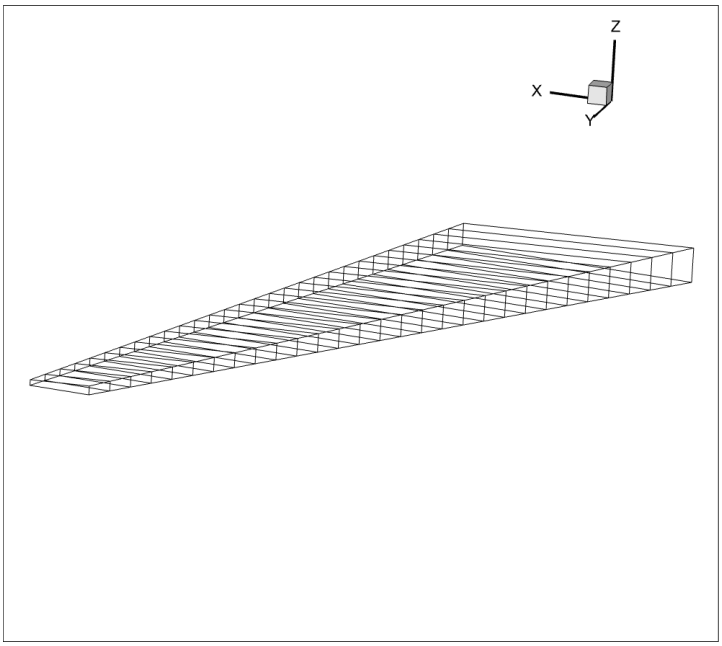

Figure 4.6: Onera M6 wingbox model FEM.

The structural topology is defined by a framework of 120 discrete nodes, positioned at the four vertices of the wingbox cross-sections from the root to the tip. To ensure a continuous and consistent geometry, a spanwise interpolation was performed to establish the x- and z-coordinates of these vertices as a function of their spanwise location. By discretizing the span into 30 equidistant intervals, a total of 30 spanwise stations were generated for each corner of the wingbox. The element types used to model the wingbox are summarized in Table 4.6. The table shows the wingbox is comprised of three primary structural constituents to model the different wingbox components: panels, stringers, and ribs. The validity of each of these elements is further discussed in relevant documentation by MSC Software [15]. The material used for the wingbox was decided to be aluminium for its simple isotropic behavior and simplicity in implementation compared to non-isotropic materials. The properties of the aluminium type used are documented in Table 4.7 [36, 12, 31].

Table 4.6: Structural elements wingbox model.

Part	Element IDs	Element Type
Panels	1 – 116	CQUAD4
Stringers	117 – 232	CBAR
Ribs	233 – 262	CQUAD4

A critical requirement for the physical validity of the models for the FSI simulations is the strict containment of the internal wingbox within the aerodynamic mold line. Any

Table 4.7: Material properties wingbox model [12].

Material	Aluminium
Young's Modulus (E)	71000 Mpa
Poisson's Ratio (ν)	0.33 [-]
Density (ρ)	2700 kg/m^2

protrusion of the structural elements beyond the aerodynamic surface would compromise the geometric fidelity of the coupled system. This could lead to non-physical results during mapping of aerodynamic loads and mesh deformations. Verifying this spatial containment serves as an essential validation of the coordinate system alignment and scaling consistency between the aerodynamic and structural sub-models. As shown in Figure 4.7, the aerodynamic outer loft completely encapsulates the structural assembly. This successful enclosure confirms that both models are synchronized in scale and spatial orientation, ensuring their readiness for integrated multidisciplinary analysis.

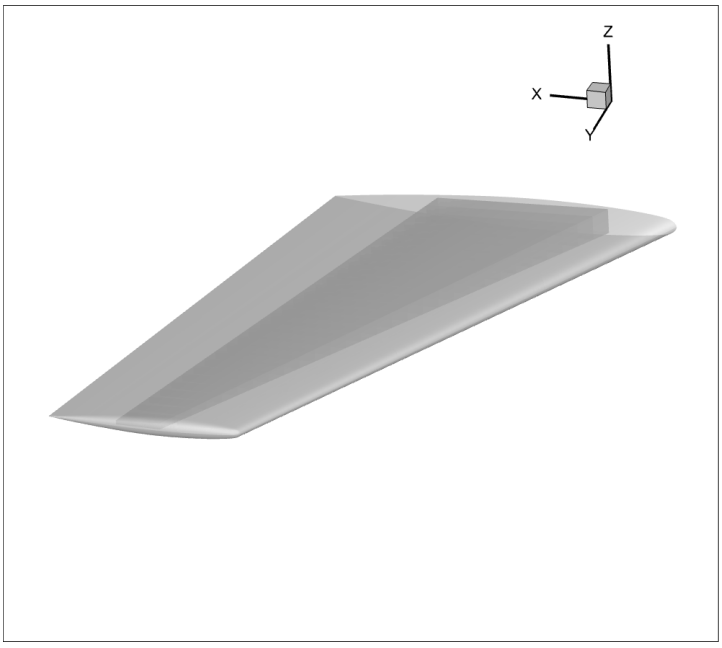

Figure 4.7: Onera M6 aerodynamic surface- and wingbox models.

4.4. Fluid-Structure Interaction Model Behavior

The integration of the aerodynamic and structural models concludes the preparation phase of the FSI simulations. However, prior to executing coupled FSI simulations, it is recommended to characterize the fundamental aeroelastic behavior of the system. Determining this behavior acts as a sanity check for the FSI simulations that will be executed. The key behavior to monitor in aeroelastic simulations is the chordwise pitching motion. It influences the increase or decrease in effective angle of attack. The behavior is governed by the spatial relationship between the Center of Pressure (CoP) of the aerodynamic model and the flexural axis of the structural model [18].

The center of pressure (CoP) represents the point on the wing cross-sections where the distributed pressure field can be replaced by a single equivalent force. The chordwise location of this point is determined by the ratio of the first moment of pressure to the total integrated pressure, as defined in Equation 4.2. This equation is valid when the cross section of the wing is simplified to a line from Leading Edge (LE) to Trailing Edge (TE). In this formulation, x represents the distance from the reference point to the point of known difference in pressure coefficient between the lower and upper wing surfaces $\Delta C_p(x)$. This equation allows for direct extraction from the CFD results obtained in subsection 4.2.3 [1].

$$CoP = \frac{\int_{LE}^{TE} x \cdot \Delta C_p(x)\, dx}{\int_{LE}^{TE} C_p(x)\, dx} \tag{4.2}$$

Parallel to this, the flexural axis or shear center, is necessary to identify the pitching behavior of the FSI simulations. This axis is defined as the locus of points where an applied vertical force produces pure bending without inducing torsional rotation about the spanwise axis (y-axis). Within the FEM wingbox, a loading line is introduced by a connecting the chordwise rib centroids to the rib vertices. To locate the flexural axis at each spanwise station, vertical forces (in positive z direction) are applied at varying chordwise positions, and the resulting loading line nodal rotations are monitored. This transfer of force to the central loading line is modeled using infinitely Rigid Body Elements (RBE) [15, 37].

The relative positioning of these two axes dictates the stability of the wing. If the flexural axis is located upstream of the center of pressure, the aerodynamic lift results in a nose-down pitching for positive angles of attack and nose-up behavior for negative angles of attack. This behavior which reduces the magnitude of the effective angle of attack. Conversely, if the flexural axis lies downstream of the center of pressure, the resulting behavior induces a nose-up pitch for positive angles of attack and nose-down pitch for negative angles of attack [37]. For the Onera M6 configuration developed in this chapter, the center of pressure was found to be situated upstream of the flexural axis, see Figure 4.8. This alignment suggests that the coupled system will exhibit the response described, leading to an increase in magnitude of the effective angle of attack. This means the FSI simulation will increase the magnitude of the angle of attack until the internal loads in the structure prohibit the wing from pitching back more [15, 12].

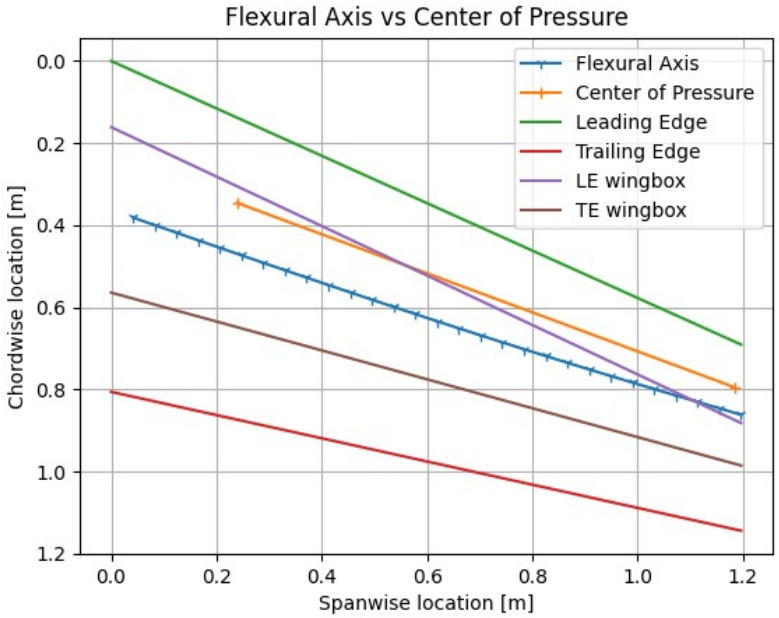

Figure 4.8: Flexural axis vs center of pressure location M=0.8395 and α=3.06 case.

5
Static FSI Framework

This chapter details the development and implementation of the static Fluid-Structure Interaction (FSI) solver designed to capture steady-state aeroelastic response of the wing geometry. To provide a comprehensive overview of the simulation architecture, the solver is decomposed into its fundamental algorithmic building blocks shown in Figure 2.4. Each component is discussed in isolation before demonstrating their integration into a unified static FSI framework. Beyond the primary algorithm, this chapter details a series of incremental verification tests conducted to verify whether the expected outcome is achieved every step of the process.

5.1. Automated Loading and Running of Simulation Files

The foundational phase in the development of the static FSI framework involves the programmatic automation of the loading and execution protocols for both the Computational Fluid Dynamics (CFD) and Finite Element Method (FEM) solvers. Given that the framework employs a partitioned approach, the aerodynamic and structural domains are treated as separate entities. Each requires specialized initialization and control logic. A primary challenge in this automation phase is the synchronization of two diverse software environments. Because the interaction protocols for Ansys Fluent (for CFD) and MSC Nastran (for FEM) differ significantly (spanning from session management to input-deck processing) their implementation strategies are addressed independently [27, 15].

5.1.1. CFD File Loading and Running

The integration of the aerodynamic solver into the static FSI framework is facilitated through PyFluent, a high-level Python API that enables programmatic control over Ansys Fluent [38]. The execution workflow begins with the initialization of a Fluent session, which serves as a computational backbone for the solver. This session acts as a virtual instance of the Fluent environment. However, before simulation data can be read, specific configuration parameters must be defined to ensure the solver is optimized for the iterative nature of FSI coupling.

To maintain the rigor required for high-fidelity wing analysis, several critical session settings are pre-defined when initializing the Fluent session:

- **Execution mode:** The session is initialized in *solver* mode rather than *meshing*. As the framework assumes the availability of a pre-generated, high-fidelity simulation file containing accurate mesh and boundary conditions. The pre-

generated file is a result of the mesh convergence studies in subsection 4.2.3.

- **User Interface (UI) Configuration:** The *ui_mode* is set to *no_ui*. This suppresses the Graphical User Interface, thereby reducing the memory overhead and significantly accelerating the execution of iterative commands. Furthermore, the selection of a headless environment is critical for deployment on the TU Delft High-Performance Computing (HPC) cluster. Here graphical rendering is often resource-prohibitive or unsupported.
- **Numerical Precision:** To mitigate rounding errors and ensure the fidelity of the FSI coupling, *double precision* (64-bit) is selected over *single precision* (32-bit). In FSI simulations, small structural displacements often lead to subtle changes in the pressure field. Double precision is essential to capture these minute gradients and to prevent the accumulation of numerical noise.
- **Parallel Processing:** The *processor_count* is configured to leverage multi-core architectures. While increasing the core count reduces the wall-clock time for the flow solution, the scalability of the solver is subject to *Amdahl's Law*. Here the total speedup is limited by the serial portions of the code and the latency of the Message Passing Interface (MPI) [39]. In the context of this FSI framework, a balance must be struck between the speed of the CFD solve and the increased complexity of extracting integrated surface forces from partitioned domains via *User-Defined Functions (UDFs)*.

Once the session is successfully initialized, the simulation file is loaded via its absolute directory path. To streamline the repetitive data exchange between Ansys Fluent and MSC Nastran, the working directory is programmatically synchronized with the simulation project folder. This synchronization allows for the consistent use of relative file paths, effectively minimizing the risk of "path-not-found" exceptions within the automated data-handling pipeline.

Notably, the simulation state remains persistent within the PyFluent session throughout the FSI loop. By maintaining this persistence, the overhead associated with re-initializing the solver and re-allocating memory for the mesh is eliminated. This state-retention strategy represents a significant efficiency gain, as it allows for near-instantaneous updates to the boundary conditions and mesh deformations between coupling iterations without restarting the computational process. Following the initialization of the PyFluent session and the reading of the simulation data, the execution of the aerodynamic solver is governed by defined convergence criteria. The convergence criteria within the framework are in twofold:

- **Residual Thresholds:** The solver monitors the L^2 norms of the residuals for continuity, momentum (u, v, and w velocities), and the energy equation. The simulation is considered converged when these residuals fall below a predefined tolerance, signifying that the iterative changes in the flow field have become negligible.
- **Iteration Limits:** To prevent the framework from entering infinite loops in cases of marginal numerical instability, a secondary criterion based on a maximum number of iterations is implemented.

The simulation terminates upon satisfying either of these conditions. This stoppage is vital for the efficiency of the overall FSI loop. Once the flow field has stabilized, further iterations provide diminishing returns in accuracy while unnecessarily increasing the total computational wall-clock time. In the context of the static FSI framework, achieving a converged aerodynamic solution in each coupling step is paramount to ensuring that the subsequent structural deformation is based on a flow field which satisfies the Navier-Stokes equations with minimal imbalance between the left- and right hand side of the equations. The residuals are what describes the total imbalance of the flow field, these are obtained by adding the imbalances of all control volume in the computational domain.

The aerodynamic simulation files are iteratively updated throughout the static FSI coupling process to maintain the physical consistency between the fluid and structural domains. This updating of the aerodynamic mesh is executed by satisfying the kinematic condition discussed in section 2.1. As the structural solver computes new equilibrium positions of the wing, the CFD mesh must be re-synchronized to match the deformed geometry. This procedural adaptation ensures that the flow field is solved around a representative wing shape, capturing the FSI effects inherently in the steady-state response.

The specific execution logic is illustrated in the computational workflow in Figure 5.1. The upper horizontal path represents the initialization phase, where the initial aerodynamic model is loaded and executed for the first time. After the initial CFD iteration, each time the CFD mesh is updated as a result of the structural deformation, the CFD simulation is executed for this updated CFD mesh to start the aerodynamic simulation for the next FSI iteration. This process is illustrated in the bottom section of the diagram. The cycle of updating the CFD mesh and running the CFD simulations repeats until the maximum number of FSI iterations or the FSI convergence criterion is reached.

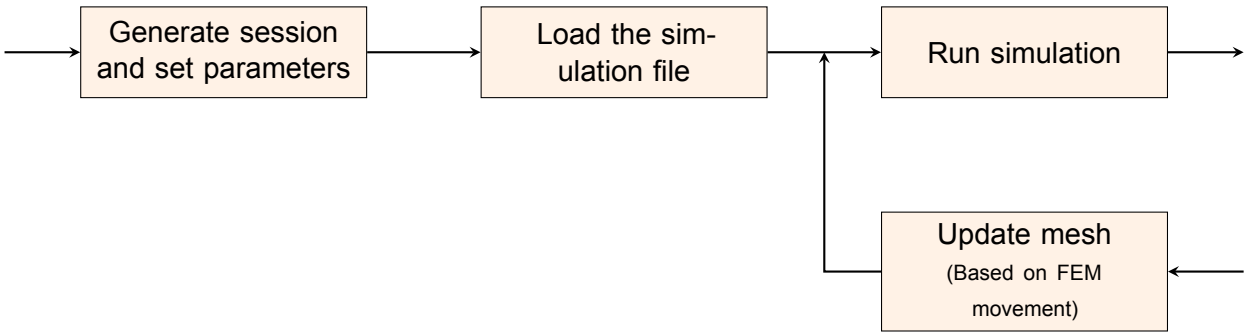

Figure 5.1: Automated CFD solver file loading, running and mesh updating workflow.

5.1.2. FEM File Loading and Running

The structural component of the FSI framework is managed through MSC Nastran, which utilizes a batch-processing paradigm rather than the persistent session-based approach employed by the aerodynamic solver. Consequently, the structural input deck, the Bulk Data File (BDF), must be re-initialized and re-loaded during each coupling iteration to incorporate updated aerodynamic load distributions. This pre-processing

phase is facilitated by PyNastran, a specialized Python library designed for the programmatic manipulation of Nastran input files [32].

While PyNastran provides a robust interface for defining the structural model and its spatial constraints, it lacks an internal solver engine [32]. Therefore, the execution of the BDF is handled via the Python sub-process library[1]. The implementation of the sub-process library is designed for both automation and reliability, managing the system-level invocation of the MSC Nastran executable by targeting the BDF path within a synchronized working directory. Crucially, this execution includes return-code verification to ensure the solver terminates successfully. By monitoring these system exit codes, the FSI framework can detect issues such as solver divergence or license availability in real time. This monitoring prevents the coupling loop from proceeding with invalid or corrupted structural data.

A critical configuration within the BDF file is the specification of HDF5 (Hierarchical Data Format) output format. The transition from traditional ASCII-based output files, such as the *.f06* file, to the HDF5 binary format significantly reduces the computational latency associated with post-processing. HDF5 allows for direct, indexed access to nodal displacement arrays. This eliminates the need for complex and error-prone string-parsing algorithms, and this structured binary approach is essential for the efficient and automated extraction of nodal translation and rotation results. The extraction of nodal results will be further discussed in section 5.4.

The sequential execution and iterative logic of the structural solver are summarized in Figure 5.2. In alignment with the batch-processing requirements described above, the workflow is initiated through a new FSI iteration that triggers the generation of a simulation session. As illustrated by the feedback loop, the process must return to this initial stage for every iteration. This ensures that the updated aerodynamic forces, resulting from the satisfied dynamic conditions discussed in section 2.1, are in the BDF before the solver is re-invoked via the subprocess library. This process repeats itself by receiving the aerodynamic loads from the most recent CFD simulation. The exit arrow following the simulation represents the transfer of the HDF5-encoded displacement results. These results are then used to inform the subsequent aerodynamic mesh deformation. This cycle ensures that the structural response remains synchronized with the evolving aerodynamic loads throughout the FSI iterations.

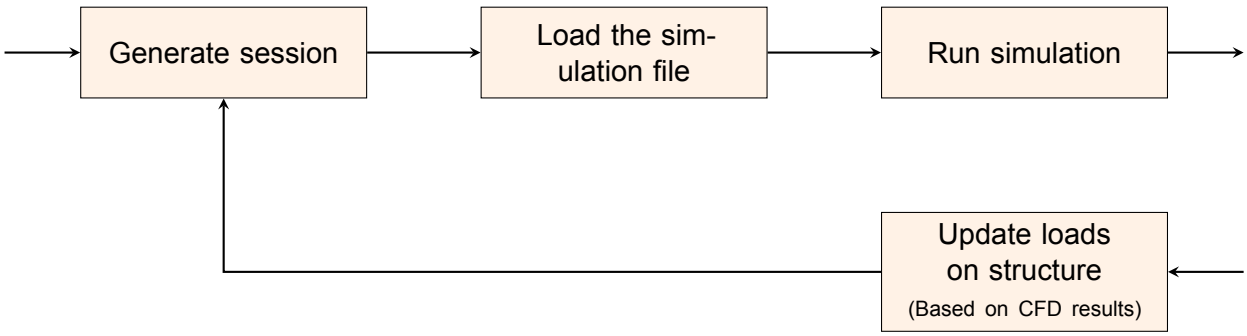

Figure 5.2: Automated FEM solver file loading, running and force updating workflow.

[1] `https://docs.python.org/3/library/asyncio-subprocess.html`

5.2. Force Result Extraction

As Ansys Fluent does not provide the specific aerodynamic load distributions required for structural coupling as a native output format, a customized User-Defined Function (UDF) was developed to facilitate the extraction of these forces. This process constitutes the primary data transduction step, where the pressure and shear stress distributions on the wing surface are integrated and mapped into discrete force vectors. The implementation is divided into two primary phases. These phases are the algorithmic formation of the extraction logic and the programmatic execution within the solver environment. Consequently, subsection 5.2.1 delineates the mathematical and C-based architecture of the UDF, subsequently subsection 5.2.2 details the procedures for compiling and invoking the function through the PyFluent interface.

5.2.1. Force Extraction UDF

The transfer of aerodynamic loads to the structural domain is managed by a customized Define on Demand UDF. It is specifically designed to automate the extraction of surface forces from the converged CFD simulation. This function performs a discrete integration of the pressure and viscous stress distributions across the specified wing boundary faces of the wing zones. By capturing the state of the flow field at the fluid-structure interface, the UDF provides the high-fidelity load inputs necessary for the subsequent structural analysis. The extraction process is not merely a data-retrieval task but a critical transduction phase that converts the continuum mechanics of the fluid flow into a discrete force distribution suitable for Finite Element Analysis (FEA) [12].

The total aerodynamic force vector $\vec{F}_{tot}$ exerted on a single boundary face of the wing surface is established through superposition of the normal and tangential effects on the face. The pressure force component, F_p, represents the load acting normal to the face. It is calculated by multiplying the scalar static pressure p, retrieved from the solver's finite volume center, by the face area vector $\vec{A}$, see Equation 5.1. Simultaneously, the viscous force component, F_v, is given by the skin friction acting upon the wing. This is derived from the wall shear stress vector $\vec{\tau}$, which represents the viscous drag exerted by the fluid on the solid boundary, see Equation 5.2. The total force is subsequently represented by Equation 5.3 [28].

$$\vec{F}_p = p\vec{A} = p \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} \tag{5.1}$$

$$\vec{F}_v = -\vec{\tau} = - \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} \tag{5.2}$$

$$\vec{F}_{tot,face} = \vec{F}_p + \vec{F}_v = p \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} - \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} \tag{5.3}$$

To ensure the integrity of the data exchange across the FSI interface, the implementation adheres to a strict requirement for coordinate system consistency as mentioned in chapter 4. All load vectors are extracted relative to the solver's Global Coordinate System (GCS). This synchronization is vital for preventing numerical misalignment that could lead to erroneous moment calculations. The UDF logic iterates through the nodal connectivity of each face thread to determine the spatial coordinates of the centroid location where these forces are located [12, 27].

The framework utilizes a distributed I/O strategy to maintain high throughput on High-Performance Computing architectures. In a parallel environment, a global synchronization of all surface data would create a substantial communication bottleneck. To mitigate this, each computational node independently serialized its local boundary data into unique, indexed CSV files. This parallel serialization includes the processor identifier, unique face identifiers, and the three-dimensional components of the resultant forces. By allowing each processor to write to a dedicated file simultaneously, the framework maximizes disk-write speeds, minimizes the idle time of the compute nodes and reduces writing errors of different nodes to a file simultaneously. These distributed datasets are subsequently aggregated by a secondary Python-based processing layer that performs a global spatial sort, preparing the data for the interpolation to the structure [28, 29].

To ensure the modularity and flexibility of the FSI framework, the UDF does not rely on hard-coded zone identifiers. Instead, the function implements a zone-identification protocol that interfaces with an external configuration file, named zone_names.txt. At the beginning of the extraction sequence the UDF invokes a standard file-reading routine to ingest a string containing the the user-defined names. This user-defined string contains FSI interaction face zone names such as “wing_top”, “wing_bottom”, “wing_edge”.

The implementation then employs a recursive string-tokenization process to isolate individual zone names from the input string. Once these names are isolated, a search algorithm iterates through the internal data structure of the Fluent domain, utilizing the THREAD_NAME and Lookup_Thread macros to find matching pairs [29]. This allows the UDF to map the human-readable labels provided by the user to their corresponding internal thread pointers and integer IDs. If a match is successful, the thread is added to an active extraction list. If a zone is not found within the current domain, the UDF outputs a warning to the console but continues the execution for the remaining valid zones. This decoupled architecture allows for rapid iteration across different wing geometries and mesh configurations, as the user can simply update the external text file to point the extraction logic to new surface boundaries without modifying the underlying C code of the UDF.

The procedural logic of the UDF is visually summarized in Figure 5.3, which illustrates the transition from initial zone identification to the parallelized serialization of the surface loads. For a comprehensive review of the implementation, the complete source code of this UDF is given in the Github repository[2]. The C-based UDF includes extensive internal documentation that further details the specific macros and data-handling

[2] https://github.com/Lennarth1998/master_thesis_fsi/tree/main

routines employed during the interfacial force extraction process.

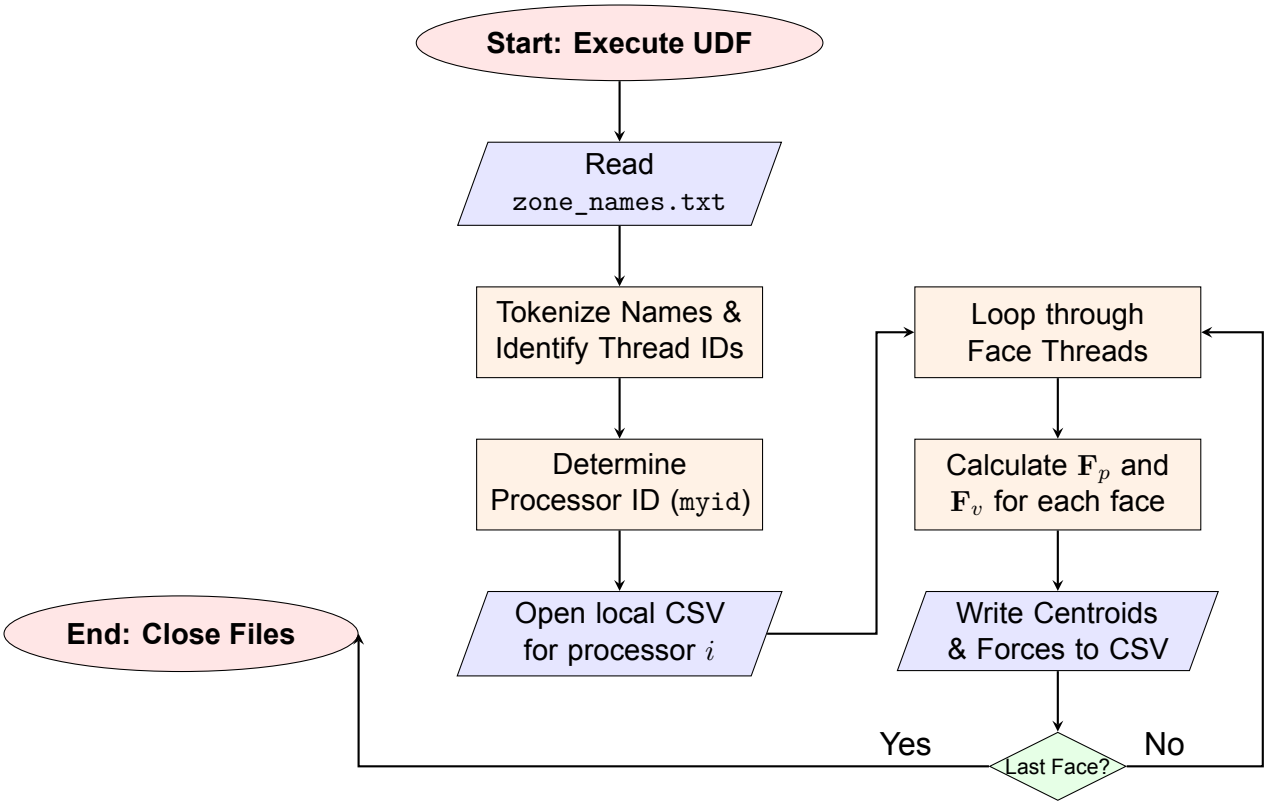

Figure 5.3: UDF force extraction workflow.

5.2.2. Running the Force Extraction UDF

Following the algorithmic definition of the force extraction logic, the UDF must be integrated into the solver's memory space and triggered at the appropriate stage of the FSI process. This integration is achieved through an automated compilation process within the PyFluent environment. During this phase, the C-based source code is translated into machine-executable binary format compatible with the solver's architecture. By utilizing Fluent's build-in compiler, the framework maintains a self-contained execution environment that eliminates the need for external development tools or third-party compiler dependencies, thereby increasing the portability of the automation script [27, 38, 29].

Following the automated compilation process, the lifecycle of the UDF is structured into two distinct operational phases: the initial setup, as discussed in subsection 5.2.1, and the iterative execution. The stages of this automated process are illustrated in the hierarchical workflows provided in Figure 5.4 and Figure 5.5. The initialization phase (see Figure 5.4) represents a non-recurring configuration step performed after loading the Fluent simulation file. As indicated, this step is only activated during the first iteration of the FSI loop. During this stage, the C-source code is compiled into a machine-executable binary and loaded into the solver's memory as a shared library. The extraction logic given by the UDF is stored in the system's RAM for computation on demand. This architectural overhead associated with compilation is incurred only

once, regardless of the number of subsequent coupling iterations.

In contrast, the execution phase shown in Figure 5.5 represents the recurring on-demand trigger that facilitates data transfer within every FSI coupling iteration. This recurring trigger starts when the CFD simulation has obtained convergence or has reached the maximum number of iterations. The moment this happens, the force extraction UDF is executed to obtain the force components of the forces in the given face zones. Once all the forces of the face zones are obtained in the form of CSV files, the UDF execution is completed and the forces are ready for interpolation to the structure. The interpolation of the forces to the structure will be discussed in detail in section 5.3.

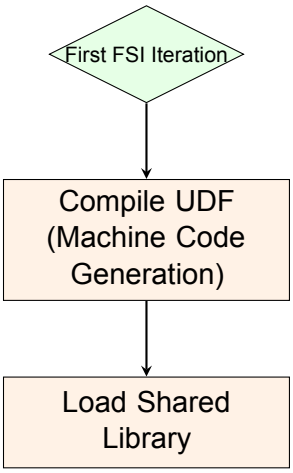

Figure 5.4: UDF initialization workflow

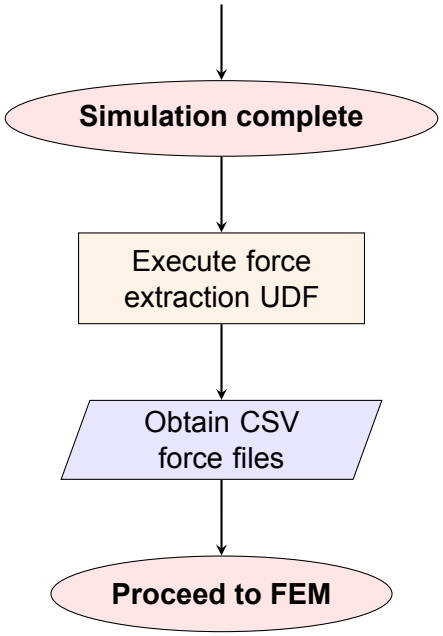

Figure 5.5: UDF execution workflow.

To establish the mathematical validity of the customized force extraction routine, it is essential to perform a comparative analysis between the UDF output and the reference data provided by the native Ansys Fluent monitors. While the solver does not provide a direct list of forces for individual faces, it offers global integrated force magnitudes in the x-, y- and z directions for any specified boundary zone. Consequently, verification is achieved by aggregating the discrete nodal force vectors extracted by the UDF across the targeted surface threads and comparing them to the resultant sums of Fluent's internal solver monitors.

The results of this check are summarized in Table 5.1. The data demonstrates an exceptional level of correlation between the two methods, with relative errors ranging from approximately 10^{-5}% to 10^{-6}%. The marginal discrepancies observed are attributed to cumulative rounding errors and variations in floating-point summation sequence between the UDF's serialized data processing and the solver's internal parallel reduction algorithms. Given that these differences are several orders of magnitude smaller than the primary force components, they are considered numerically insignificant and do not impact the fidelity of the subsequent structural analysis.

Table 5.1: Comparison of total aerodynamic force components between UDF extraction and the internal Fluent force monitors.

Source	F_x [N]	F_y [N]	F_z [N]
UDF Extraction	$160.448\,165$	$373.110\,84$	$7532.033\,11$
Fluent Monitor	$160.448\,220$	$373.110\,72$	$7532.033\,20$
Absolute Difference	$0.000\,055$	$0.000\,12$	$0.000\,09$
Relative Error (%)	3.4×10^{-5}	3.2×10^{-5}	1.2×10^{-6}

5.3. Force Projection on Structure

As previously established, the initial phase of the FSI process involves the projection of aerodynamic loads from the CFD domain onto the structural model to elicit a mechanical response. Given that the wing structure is modeled as a primary load-carrying skeleton without the aerodynamic skin, the force transduction process requires an indirect mapping strategy. This mapping strategy utilizes a load-line approach similar to the method outlined by Dillinger [36], it employs a loading line upon which the forces and moments of the aerodynamic forces of the CFD simulation is projected.

The implementation of this mapping procedure begins with the generation of a load-line geometry which is integrated into the existing structural model using the PyNastran library to manipulate the BDF. A critical prerequisite for this process is the synchronization of the spatial domains. Both the aerodynamic mesh and the structural model share a common global origin and orientation. This spatial alignment ensures that the spatial coordinates extracted from the CFD solver are directly compatible with the FEM nodal positions without the need for transformation of the models.

Subsequently, a Nearest-Neighbor (NN) interpolation algorithm is utilized to associate each CFD face centroid with the geometrically closest node on the structural load line. Because the aerodynamic forces act on the physical wing surface and are thus offset from the internal load line, the algorithm must account for both the direct force vectors and the equivalent moments invoked by the force vector offset from the loading line [12, 36]. These coupled loads are systematically appended to the BDF by iterating through the serialized CSV data generated during the UDF-based force extraction phase as discussed in section 5.2. Due to the numerical sensitivity of these coordinate-based mapping, several intermediary validation steps are required to ensure the consistency of the load transfer. These steps of the force projection process are detailed in the following subsections.

5.3.1. Loading Line Generation

The generation of the structural loading line is a critical step, serving as the interface through which the aerodynamic forces are introduced to the structure and subsequently the dynamic conditions are satisfied. The loading line consists of a discrete set of grid nodes that act as the primary application points for the extracted forces and equivalent moments. To ensure these loads are physically transferred into the internal wingbox structure, Rigid Body Elements of type 3 (RBE3) are utilized [15, 36].

In this implementation, the RBE3 elements are configured such that the loading line nodes serve as the reference nodes, while the vertices of the corresponding spanwise wingbox cross-section act as the independent nodes. The RBE3 element is specifically chosen for its ability to distribute applied loads without any artificial stiffening to the structure. This is a crucial requirement in high-fidelity modeling to maintain the natural response of the wing [12]. The load distribution within the element is managed through weighting factors derived from the relative distance between the central loading line node and each vertex location, ensuring a physically consistent transduction of the aerodynamic load into the structural skeleton.

The framework automates the generation and manipulation of these structural entities through the PyNastran library [32]. The process begins by parsing a specialized input configuration file where each row contains comma-separated grid identifiers representing the vertices of a specific spanwise cross-section. To determine the spatial coordinates of the loading line nodes, the framework utilizes PyNastran to reference the initial BDF and retrieve the three-dimensional positions of the specified vertex IDs. For every defined spanwise cross-section (row of the file), a new GRID card is instantiated at the geometric centroid of these vertices. Simultaneously, an RBE3 card is programmatically defined to link this centroid (loading line) node to the structural vertices. The updated model is then serialized back into the BDF format to facilitate load transfer during the Nastran solver execution [15].

As the loading line geometry is intrinsically linked to the structural mesh, these nodes and elements deform and translate in unison with the wingbox throughout the FSI process. Consequently, this initialization sequence of the loading line is executed solely during the first iteration of the coupling process. Once established, these loading line nodes serve as the persistent spatial targets for all subsequent aerodynamic load projections. The process is graphically represented by the diagram in Figure 5.6.

To verify the structural loading line generation and the programmatic manipulation of the Nastran BDF, a consistency check was performed on the generated RBE3 entities. The validation is based on a controlled input sequence where the initial structural model contains 120 grid nodes and 262 elements. A snippet of the configuration file used to define the RBE3 vertex associations is presented in Listing 5.1.

```
1,2,3,4
5,6,7,8
```

Listing 5.1: Example of the RBE3 vertex input file structure.

Following the execution of the script that generates the RBE3 elements, the updated BDF should reflect the instantiation of two additional grid nodes (identified as 121 and 122) alongside their corresponding two additional RBE3 cards (elements 263 and 264). Specifically, the framework is expected to generate an RBE3 element linking the dependent node 121 to the independent grid nodes 1, 2, 3 and 4. It then should link a second element, node 122, to grid nodes 5, 6, 7 and 8. A snippet of the resulting BDF output, processed via PyNastran, is shown in Listing 5.2.

```
GRID    121     .362655     0.          .0000001
GRID    122     .3823414    .0412517    .0000001
```

```
...


RBE3     263      121      123456.2037395   123456   1    2    3    4
RBE3     264      122      123456.1985186   123456   5    6    7    8
...
```

Listing 5.2: Example BDF output.

As seen in the output snippet, element 263 (the first of the newly defined RBE3 cards), correctly assigns node 121 as the reference node for the independent nodes 1, 2, 3 and 4. The inclusion of the “123456” component notation confirms that all six degrees of freedom (three translational forces and three rotational moments) are successfully coupled between the dependent loading line node and the structural vertices. Furthermore, the weighting factors applied to the independent nodes are derived from their relative distance to the central loading line node. Because the framework instantiates the loading line node at the exact geometric centroid of the vertices, the distances between dependent and independent nodes is equal for each cross-section. Additionally, the connectivity of the loading line nodes and the vertices can be shown graphically for the complete wingbox in Figure 5.7. This and the consistency in the BDF validates the geometric logic and the distance based weighting algorithm are operating as intended.

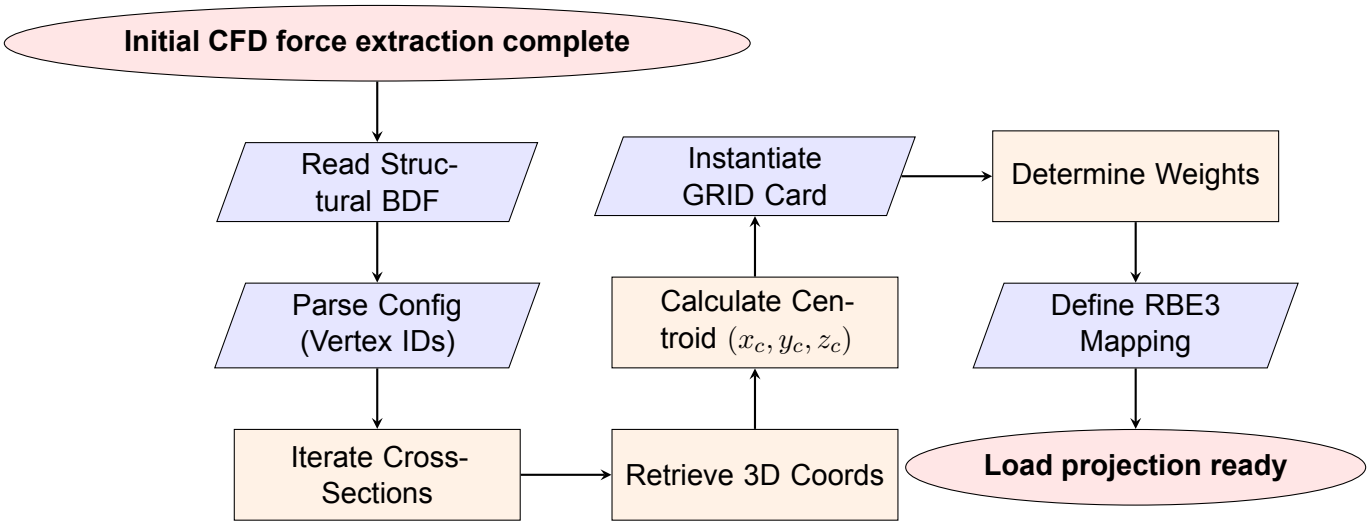

Figure 5.6: Loading line generation workflow.

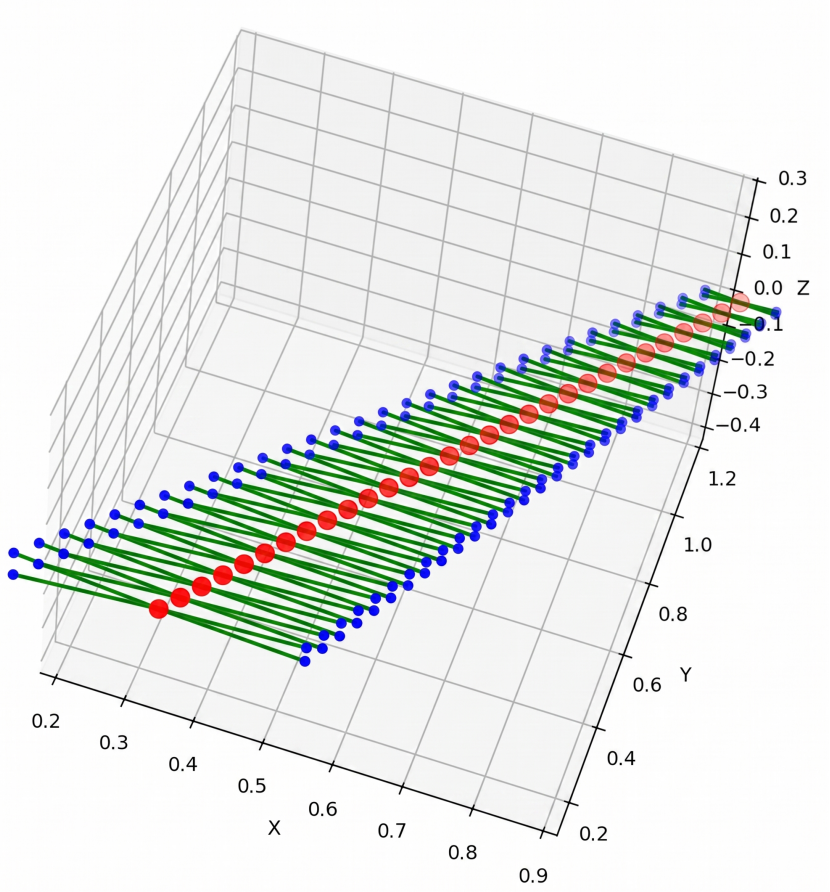

Figure 5.7: Connectivity loading line nodes and wingbox vertices.

5.3.2. Force and Moment Projection on Loading Line

Following the generation of the loading line, the aerodynamic loads obtained using the process discussed in section 5.2 are mapped onto the structural domain. It is important to note that while the loading line initialization is a non-recurring process performed only after the primary force extraction, the loading line nodes themselves are non-stationary. Meaning they translate and rotate in unison with the deforming wingbox structure. Consequently, the projection of forces and moments must be re-calculated at each FSI iteration to account for the current spatial configuration of the loading axis.

The projection of the discrete forces from the CFD face centroids to the structural loading line is governed by the NN search algorithm. For each centroid in the CFD surface mesh upon which the forces are extracted, the framework identifies the geometrically closest loading line node in the spanwise direction. The force vector is projected directly onto this target loading line node, while the associated moments are derived from the force and the three-dimensional moment arm. This moment arm is the vector extending from the loading line node to the CFD face centroid. The forces and moments in the NN loading line node due to a face centroid force vector are given in Equation 5.4 and Equation 5.5 respectively. Once these spatial associations are established, the framework aggregates the cumulative forces and moments for each

loading line node. These integrated loads are then serialized into Nastran FORCE and MOMENT cards to be applied in the subsequent structural step.

$$\begin{bmatrix} F_{LL,x} \\ F_{LL,y} \\ F_{LL,z} \end{bmatrix} = \begin{bmatrix} F_{centroid,x} \\ F_{centroid,y} \\ F_{centroid,z} \end{bmatrix} \tag{5.4}$$

$$\begin{bmatrix} M_{LL,x} \\ M_{LL,y} \\ M_{LL,z} \end{bmatrix} = \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} \times \begin{bmatrix} F_{centroid,x} \\ F_{centroid,y} \\ F_{centroid,z} \end{bmatrix} \tag{5.5}$$

To maintain a physically consistent FSI response, the framework employs an incremental loading strategy. In an iterative FSI loop, the structure already exists in a loaded and deformed state from the previous iteration. For the first FSI iteration this previous loading and deformation is both zero as the structure has not experienced any load or deformation yet. The framework calculates the differential increment between the current aerodynamic loads on the structure and the loads applied in the preceding step using Equation 5.6 and Equation 5.7. By isolating this increment, the framework accounts for the internal stresses and geometric stiffening already present in the deformed structure. At the start of each new iteration, the existing FORCE and MOMENT cards are purged from the BDF. This allows the newly calculated increments of forces and moments to be applied to the current deformed configuration. This ensures that the structural solver correctly converges toward a steady-state equilibrium where the internal elastic forces balance the external aerodynamic loads. The process of reading forces and applying them on the loading line is graphically represented by Figure 5.8.

$$\Delta\vec{F}_{LL,i} = \vec{F}_{LL,i} - \vec{F}_{LL,i-1} \tag{5.6}$$

$$\Delta\vec{M}_{LL,i} = \vec{M}_{LL,i} - \vec{M}_{LL,i-1} \tag{5.7}$$

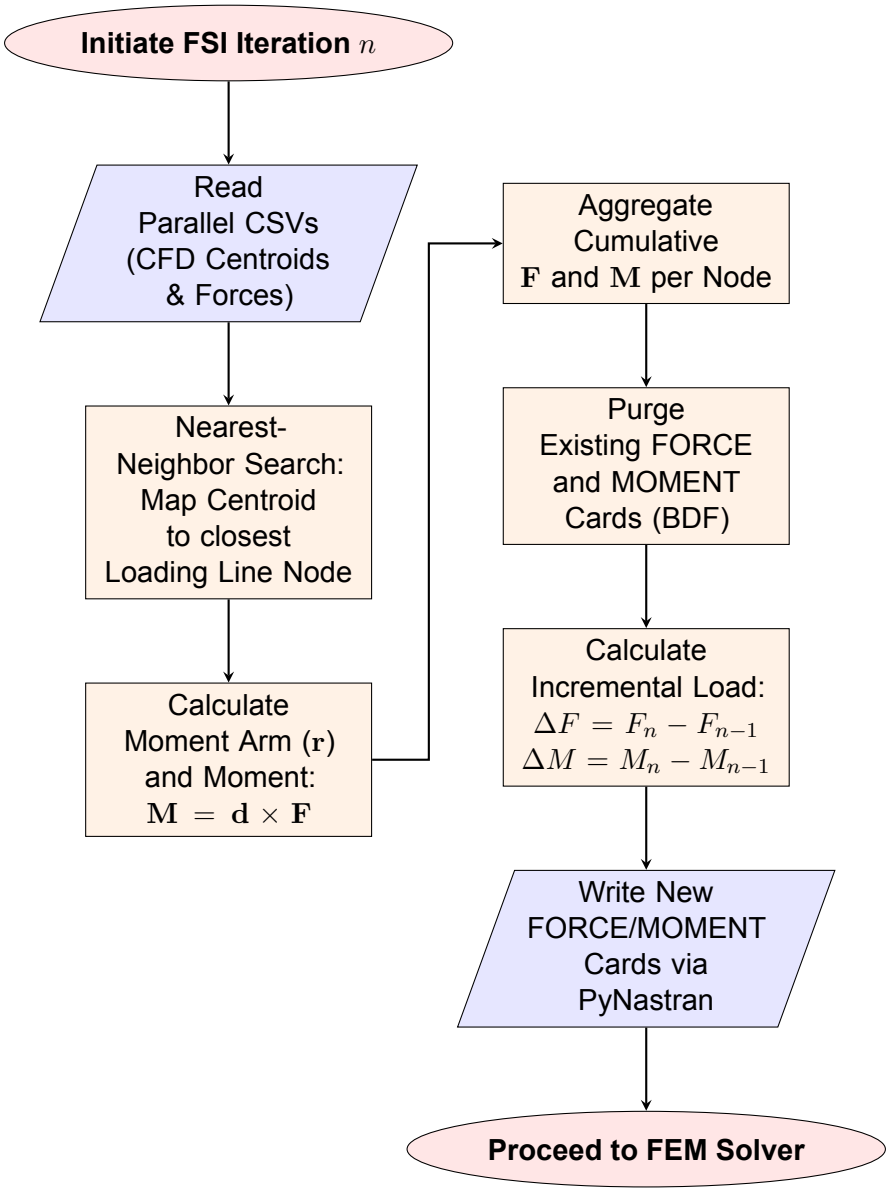

Figure 5.8: Load transduction and application workflow.

To verify the load-transduction process, the spatial coupling between the CFD face centroids and the structural loading line nodes were examined. Figure 5.9 illustrates these associations, where CFD centroids sharing a common color represent data points assigned to the same discrete loading line node. Given that the NN interpolation is governed by the spanwise (y-axis) proximity, it was expected that chordwise strips of surface centroids would be mapped to each corresponding station on the loading line. The visualization confirms this chordwise grouping, demonstrating that the algorithm correctly clusters CFD loads on the loading line nodes based on spanwise discretization. This alignment verifies the mapping methodology and ensures that the aerodynamic loads are distributed into the structural skeleton as intended.

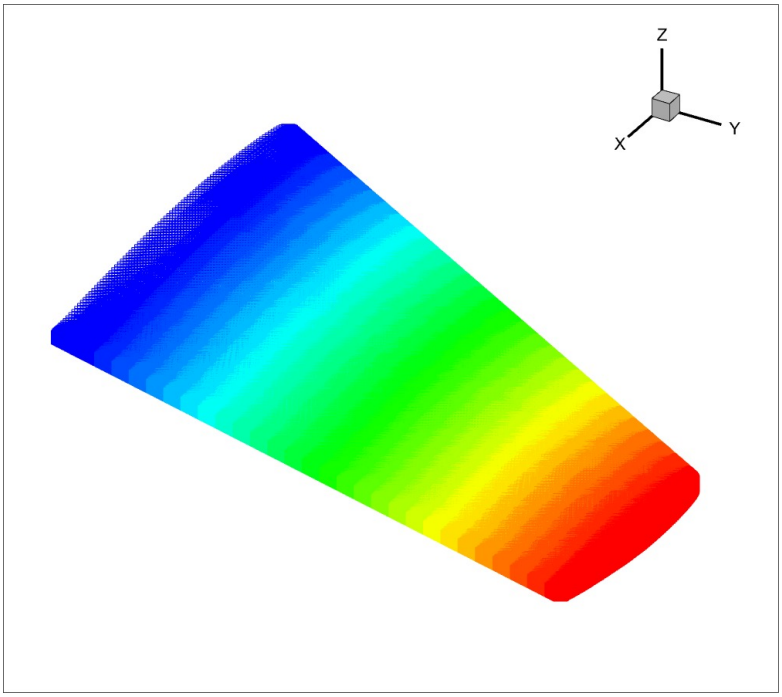

Figure 5.9: Connectivity CFD face centroids and loading line nodes in load mapping process.

5.4. Displacement Result Extraction

Upon the successful completion of the FEM simulation, the framework must quantify the structural response by extracting the displacement and rotation vectors from the structural solver output. While the global wingbox deformation results provide the necessary results to update the wingbox structure, the high-fidelity FSI coupling relies specifically on the discrete translational and rotational results of the loading line nodes. These parameters represent the updated spatial configuration of the wing's aerodynamic skeleton. By relating the loading line nodes' behavior to the wing's behavior, the system can prescribe the CFD mesh deformation for each subsequent FSI iteration [12].

The extraction framework utilizes the h5py[3] Python library to programmatically navigate the internal HDF5 file (explained insubsection 5.1.2). Specifically, the routine targets the nodal displacement arrays located within the /NASTRAN/RESULT/NODAL/DISPLACEMENT table. In this table, the framework filters the results by the specific GRID identifiers corresponding to the loading line nodes, retrieving both the translational vector components (T_x, T_y, T_z) and the rotational vectors (R_x, R_y, R_z). These vectors are subsequently passed to the mesh deformation utility to update the CFD mesh boundary, effectively closing the FSI iteration loop and ensuring the aerodynamic solver accounts for the most recent structural shape. The process of transducing the displacement results and updating the structural model is graphically represented by the diagram in Figure 5.10.

[3] `https://docs.h5py.org/en/stable/`

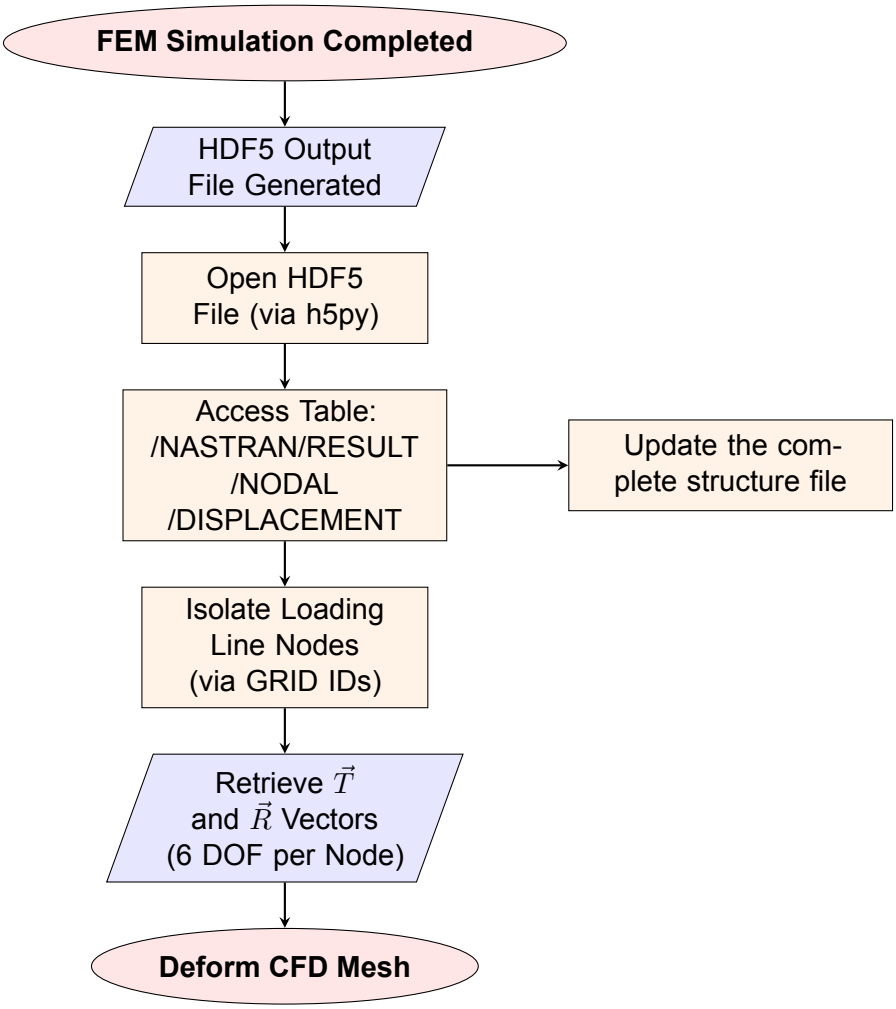

Figure 5.10: Displacement result transduction and structure model update workflow.

5.5. Aerodynamic Mesh Deformation

With the extraction of the translational and rotational vectors from the loading line nodes complete, the framework possesses the necessary kinematic data to update the CFD mesh domain. The transformation of the wing surface is achieved through a linear interpolation of these discrete nodal results as a function of spanwise location (y-coordinate). This approach generates a spanwise distribution of both translation and rotation results, effectively defining the new geometric state of the aerodynamic surface. An advantage of this implementation is the spatial synchronization between the two solvers. Since the CFD and FEM models share an identical global origin, axes, and rotational orientations, the structural results can be mapped directly to the aerodynamic mesh without the requirement for coordinate transformations.

The physical deformation of the CFD mesh is managed by the face vertices on the wing boundary. These vertices are translated and rotated according to the interpolation functions derived from the loading line kinematics at their respective spanwise positions. To execute this deformation within the solver, the native Ansys Fluent Dynamic Mesh utility is employed. This utility provides a robust method for boundary-driven mesh motion, where the specific displacement of the face vertices is prescribed via

a UDF [27, 29]. The method of the mesh deformation process depends on two critial components: the mathematical definition of the motion within the C-based UDF, and the configuration of the dynamic mesh parameters within Ansys Fluent. Given the technical requirements for the dynamic mesh process, the following sections provide a detailed breakdown of the UDF architecture and configuration of the smoothing and remeshing parameters.

5.5.1. Dynamic Mesh UDF

The prescribed movement of the wing surface is implemented via a specialized Fluent UDF utilizing the *DEFINE_GRID_MOTION* macro. This macro provides the computational framework necessary to iterate through the boundary face vertices and update their spatial coordinates each instance the dynamic mesh sequence is invoked. By operating at the nodal level, the UDF allows for a high degree of granularity in the deformation field. This ensures that the structural response of the wingbox is mapped onto the aerodynamic skin in the CFD mesh to high fidelity [29].

To ensure the wing surface deformation is both accurate and computationally efficient, the UDF incorporates several low-level optimizations designed to handle the high volume of nodal operations. A key refinement in the data-handling logic is the use of conditional I/O. The external CSV file containing the loading line node positions, translations and rotations is parsed only during the initial call of the UDF at each iteration. This is achieved by utilizing static variables and a logical flag within the C-code. The framework stores the spanwise interpolation data in the solver's persistent memory. This architectural choice bypasses the substantial I/O overhead that would otherwise occur by re-reading and re-parsing the text file for every individual vertex in the mesh.

$$\begin{bmatrix} \Delta x_{CFD} \\ \Delta y_{CFD} \\ \Delta z_{CFD} \end{bmatrix} = \begin{bmatrix} \Delta x_{LL} \\ \Delta y_{LL} \\ \Delta z_{LL} \end{bmatrix} + [R](r_{x,LL}, r_{y,LL}, r_{z,LL}) \cdot \vec{d} \tag{5.8}$$

$$[R](r_x, r_y, r_z) = \begin{bmatrix} \cos r_y \cos r_z & \sin r_x \sin r_y \cos r_z - \cos r_x \sin r_z & \cos r_x \sin r_y \cos r_z + \sin r_x \sin r_z \\ \cos r_y \sin r_z & \sin r_x \sin r_y \sin r_z + \cos r_x \cos r_z & \cos r_x \sin r_y \sin r_z - \sin r_x \cos r_z \\ -\sin r_y & \sin r_x \cos r_y & \cos r_x \cos r_y \end{bmatrix} \tag{5.9}$$

$$\vec{d} = \begin{bmatrix} x_{CFD} - x_{LL} \\ y_{CFD} - y_{LL} \\ z_{CFD} - z_{LL} \end{bmatrix} = \begin{bmatrix} x_{CFD} - x_{LL} \\ 0 \\ z_{CFD} - z_{CFD} \end{bmatrix} \tag{5.10}$$

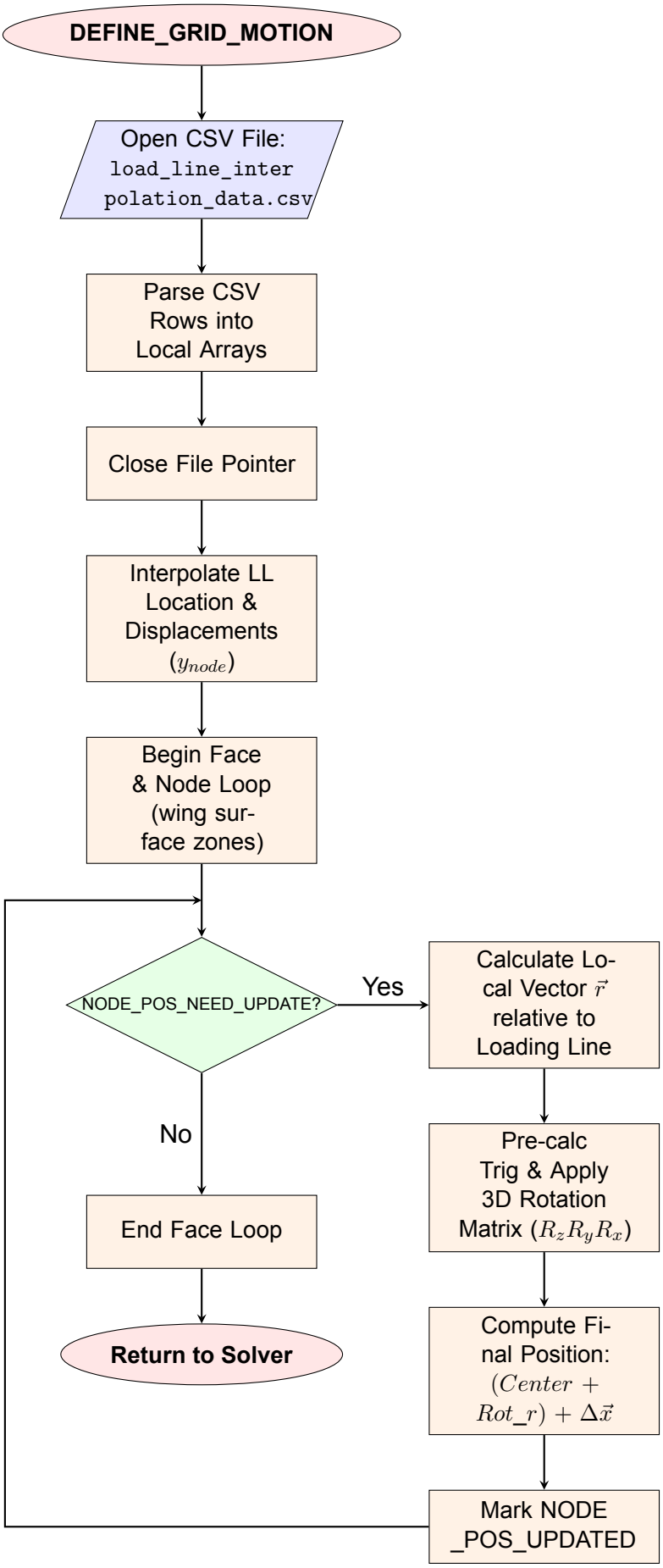

Figure 5.11: UDF boundary mesh deformation workflow.

Furthermore, the UDF adresses a critical standard unit discrepancy between the structural solver and the CFD C-code. While structural output from Nastran typically provides rotations (r_x, r_y, r_z) in degrees, the trigonometric functions in the standard C-language library *(sin, cos)* require input in radians. To prevent catastrophic errors in the wing's rotations around axes during FSI iterations, the UDF performs an internal conversion before generating the rotation matrix. Once the rotations are normalized, the framework implements the rotational transformation given by Equation 5.8 and Equation 5.9 [40]. This equation is formulated to sum the pure spanwise translation vector with the relative displacement generated by the rotation of the chordwise arm between the CFD vertex and loading line. Equation 5.10 shows how the moment arm between a CFD vertex and the loading line is calculated. The translation and rotation results obtained for each CFD face vertex are substituted in Equation 5.8 to obtain the updated location of the CFD face vertex that is processed. These translations and rotations per face vertex are obtained by substituting the spanwise location into the linear interpolation functions of the loading line displacements and rotations as mentioned before. Once all the vertices are processed, the UDF is finished and Fluent's dynamic mesh method will adjust the internal volume locations to obtain the updated mesh. The diagram of the steps of the UDF file are also visually represented in Figure 5.11.

5.5.2. Dynamic Mesh Smoothing Configuration

While the UDF detailed in subsection 5.5.1 describes the motion of the boundary, this represents only a part of the dynamic mesh process. To maintain numerical stability and spatial accuracy, the surrounding fluid volume mesh and adjacent surface meshes, such as the symmetry and far-field boundaries, must deform accordingly. The primary objective is to preserve mesh quality and prevent cell degradation during mesh deformation. This preservation of mesh quality is governed by the dynamic mesh parameters assigned to the deformation method [12, 27].

Two primary methods that can be employed to manage boundary-induced deformations are: mesh smoothing and remeshing. Smoothing techniques propagate boundary displacements through the volume and surface meshes by updating the locations of the vertices which are not assigned any motion by the UDF. Conversely, remeshing serves as a robust alternative when cell skewness exceeds acceptable limits. Given its high computational cost, remeshing is treated as a secondary measure, triggered generally only when smoothing fails to maintain sufficient mesh integrity [27].

In this FSI framework, diffusion smoothing is utilized to absorb the wing's motion. This method is particularly effective for problems involving a combination of moderate translation and rotation [27]. This smoothing method distributes the deformations of the mesh smoothly throughout the domain. To prioritize the preservation of the boundary layer and near-wing mesh quality, a boundary distance-based diffusivity approach is adopted. The governing equation for the mesh velocity $\vec{u}$ is given by Equation 5.11.

$$\nabla \cdot (\gamma \nabla \vec{u}) = 0 \tag{5.11}$$

In this context, $\vec{u}$ is expressed in terms of pseudo-time as the CFD simulations are

steady [27]. The diffusion coefficient, γ, determines how the motion is attenuated as a function of the distance d from the moving boundary. Moving on, Equation 5.12 shows the diffusion coefficient dependency on the boundary distance.

$$\gamma = \frac{1}{d^{\alpha_{diff}}} \tag{5.12}$$

In this equation, α represents the diffusion parameter, which dictates the local rigidity of the mesh. A higher value of α increases the rigidity of the cells in the immediate vicinity of the wing, forcing the near-field mesh to move more uniformly with the boundary. Tuning this parameter is critical. If the mesh cannot adequately absorb the displacement, cells may collapse or "fold", resulting in negative cell volumes and subsequent termination of the FSI simulation [27].

When smoothing alone is insufficient to prevent high-skewness or cell inversion, local remeshing is invoked. This method improves the mesh by locally regenerating cells and re-triangulating faces that violate predefined quality thresholds. These thresholds are primarily based on the maximum cell skewness and length scales. Once a cell's skewness exceeds the set limit, the solver remeshes the region until the local quality satisfied the convergence criteria. Extensive details on these algorithms can be found in the Ansys Fluent documentation [27].

At the onset of the first FSI coupling iteration, the user specifies the parameters and mesh methods for the dynamic mesh zones. These settings are subsequently cached within the Fluent solver's global environment. The solver retains these preferences in its internal session state for duration of the static FSI simulation [27]. The complete dynamic meshing process discussed in this subsection is illustrated in the workflow diagram in Figure 5.12.

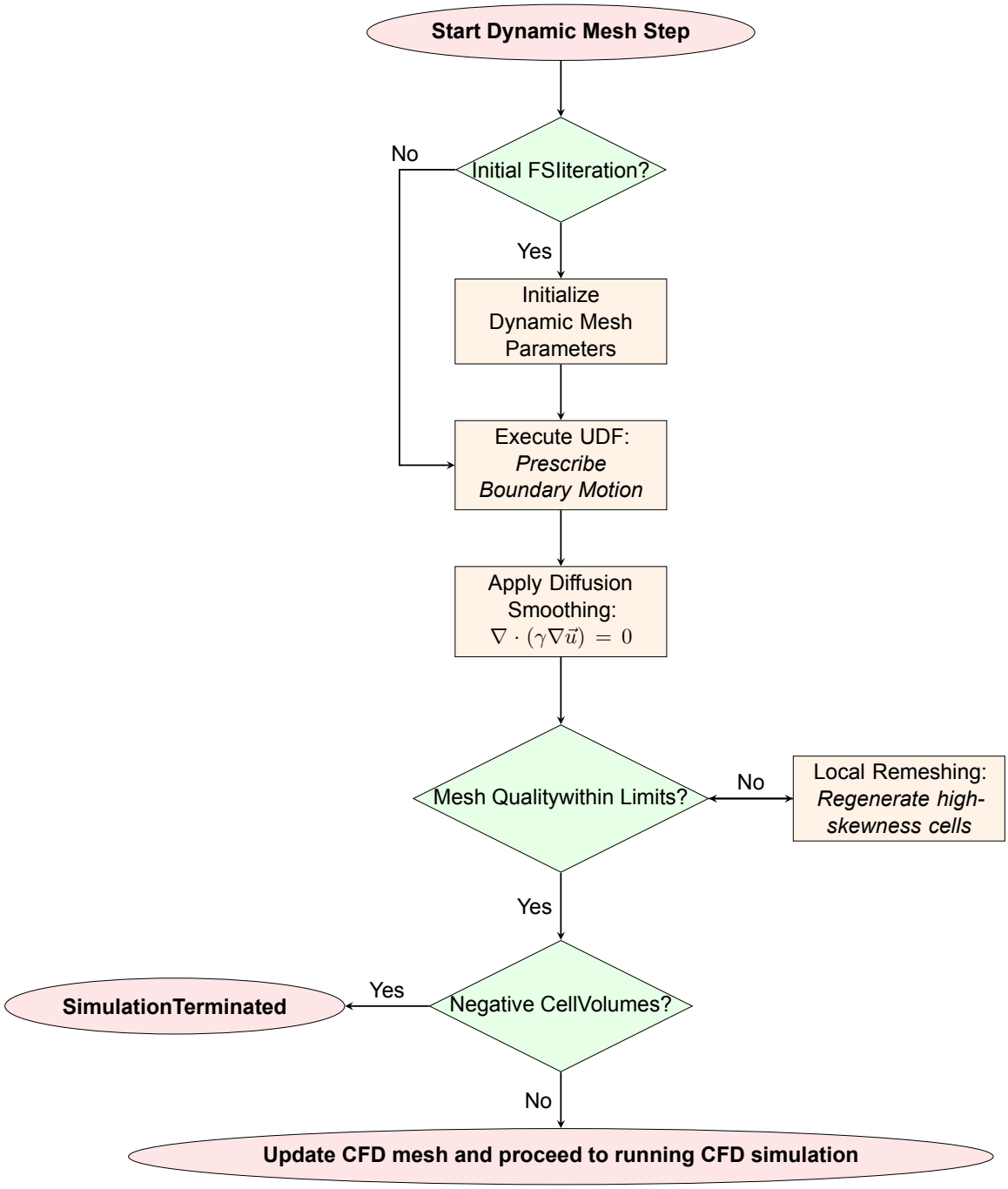

Figure 5.12: Dynamic mesh smoothing and remeshing workflow.

The efficiacy of the dynamic mesh strategy is verified by examining the aerodynamic and structural models before and after a single FSI dynamic mesh iteration. The objective is to confirm that the FSI remains conformal and that the structural model resides correctly within the deformed wing zones. Figure 5.13 illustrates the deformation of both the structural and fluid meshes after one dynamic mesh iteration. The alignment between the updated structural position and the fluid boundary confirms that the fluid mesh deformation logic reflects the structural displacements.

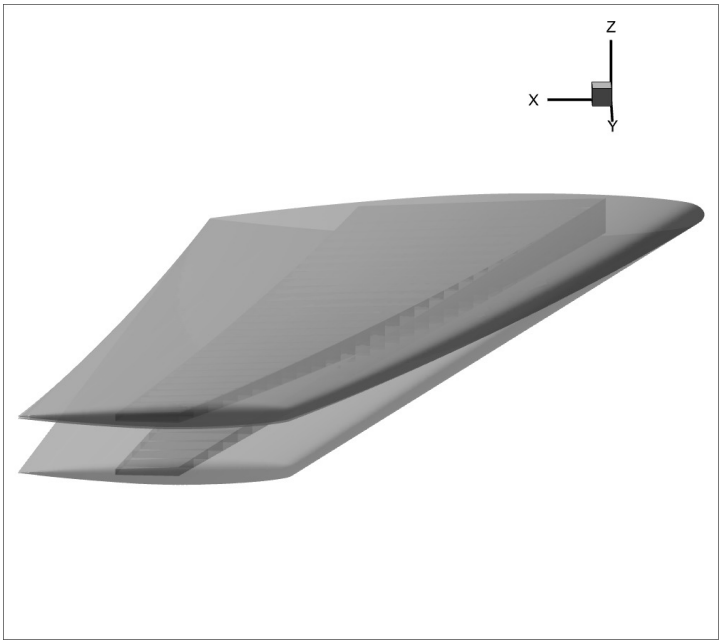

Figure 5.13: Deformed fluid- and structural meshes before and after one FSI dynamic mesh iteration.

5.6. Closing the Static FSI Framework Loop

With the individual algorithmic components established, they are integrated to form a unified static FSI framework. The high-level workflow diagram of this algorithm is illustrated in Figure 5.14, it delineates the FSI data exchange loop while omitting low-level routines. The process including the low-level routines is discussed in detail in the code available in the public repository of this research[4].

The iterative progression of the solver is regulated by a convergence check, positioned at the termination of each coupling cycle. This module is critical for preventing infinite execution loops. The termination of the FSI process is governed by two criteria: a predefined tolerance criterion, and a maximum iteration limit. Similar to the convergence logic employed within the aerodynamic solver, the FSI loop is finalized once the aeroelastic response reaches a steady state or the computational budget is exhausted. For this framework, the convergence metric is defined by the vertical tip movement of the loading line, see Equation 5.13. As the aerodynamic loads elicit a mechanical response, the displacement of the wing tip serves as a sensitive indicator of equilibrium [12].

$$tolerance = \sqrt{\Delta x_{tip}^2 + \Delta y_{tip}^2 + \Delta z_{tip}^2} \tag{5.13}$$

To initiate the static FSI framework, a specific set of input requirements must be satisfied. Beyond the baseline CFD and FEM simulation files, the solver requires a centralized parameter configuration file. This configuration file acts as the primary control deck for the simulation, containing the directory structure, dynamic mesh settings, and FSI convergence thresholds.

[4] `https://github.com/Lennarth1998/master_thesis_fsi`

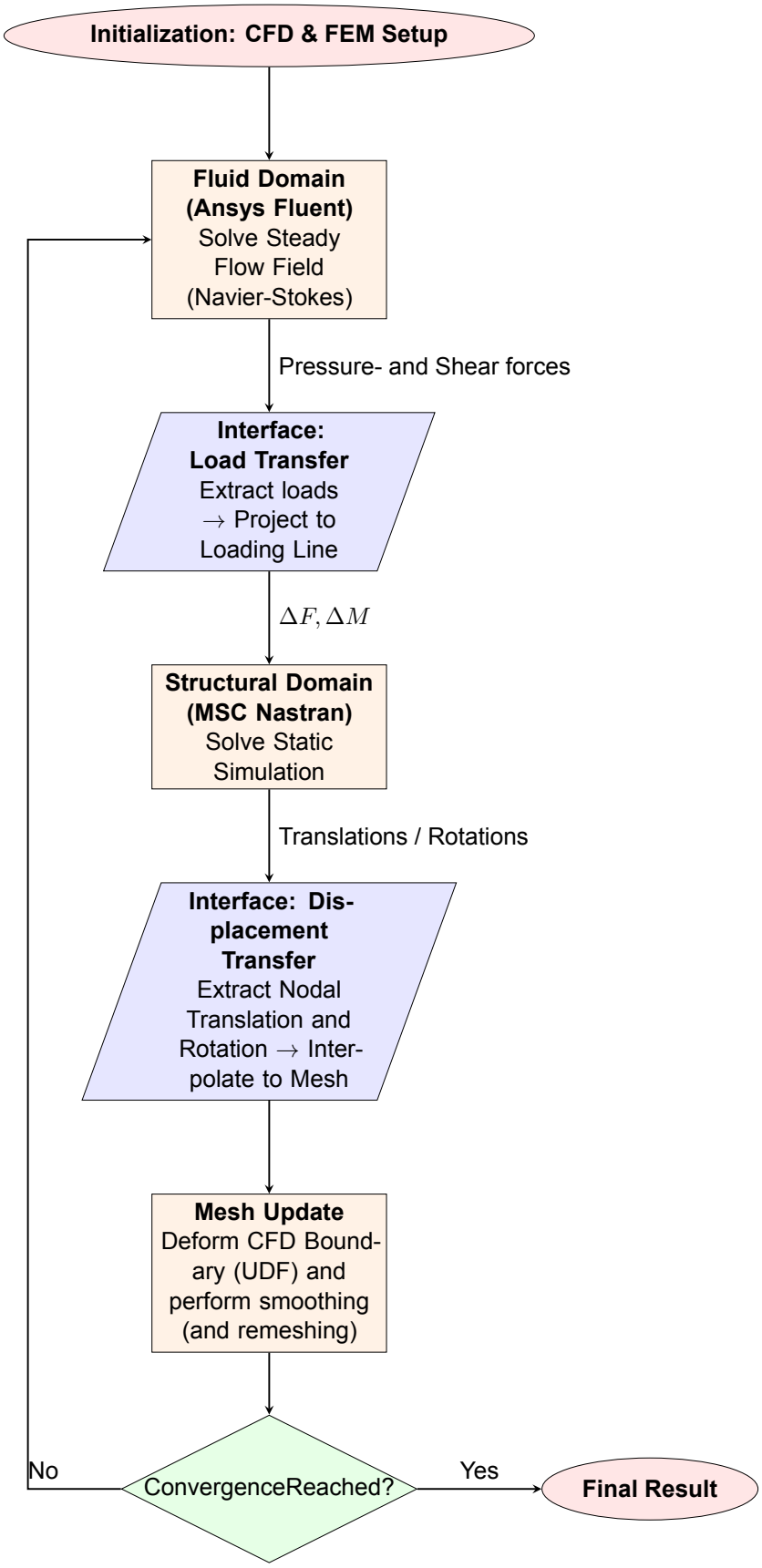

Figure 5.14: High-Level Partitioned Static FSI Framework workflow.

6

Simulation Results and Discussion

This chapter presents the simulation results obtained through the static FSI framework detailed in the preceding sections and the subsequent discussion of these results. The primary objective of this chapter is to conduct comparative analysis between low- and high-fidelity numerical results. By comparing the low-fidelity model against high-fidelity results, this study highlights the specific regions where the low-fidelity framework diverges from high-fidelity results. Ultimately, these comparisons serve to underline the operational limits of the framework and identify the critical phenomena that require more sophisticated modeling to ensure physical accuracy.

To show the operational boundaries, the simulation cases in this section focus on the inherent theoretical limitations of low-fidelity modeling. The baseline for this comparative analysis is the MSC Flighloads internal static aeroelastic solver (SOL144), which utilizes the DLM to model aerodynamics [21]. As established in chapter 2, the DLM formulation relies on the assumptions of inviscid and irrotational flow. It neglects transonic phenomena such as shock development, and utilizes flat-plate geometries that cannot account for thickness effects [26]. In contrast, the high-fidelity framework solves the full Navier-Stokes equations, allowing it to resolve the complex physical effects that are simplified or omitted by DLM [25, 28].

The following test cases are designed to isolate and quantify the impact of these assumptions by contrasting the low-fidelity approach with the high-fidelity framework's ability to resolve non-linear flow physics. To investigate the roles of viscosity, flow separation, and rotationalility, simulations were performed at constant Mach number across a range of angles of attack. Complementary to this, Mach number sweeps were conducted at a constant angle of attack to target the transonic regime. This is where the aerodynamics of finite-thickness bodies deviate significantly from linear theory due to compressibility and shock-induced effects. Together, these scenarios provide a comprehensive assessment of the conditions under which high-fidelity predictive capabilities become necessary for physical accuracy.

Beyond aerodynamic modeling, different structural models are employed to assess the impact of structural stiffness. This is executed using two distinct structural models. This includes a standard "flexible" model with a Young's modulus of $7.1 \cdot 10^{10}$ Pa and a "highly flexible" variant with a modulus reduced to 75% of the original values, specifically $5.325 \cdot 10^{10}$ Pa.

To maintain clarity in the following figures and discussions, the results are categorized using a specific labeling convention. The low-fidelity static aeroelastic (SOL144) re-

sults of the flexible wing are denoted as DLM Flexible, while high-fidelity aerodynamic results on a non-deforming mesh are labeled as CFD Rigid. Finally, results simulated in the static high-fidelity framework with two structural flexibility configurations are identified as FSI Flexible and FSI Highly Flexible, respectively.

6.1. Angle of Attack Studies

The first set of simulations are carried out to show the influence of viscosity-, flow separation-, thickness- and rotationality effects. The flight conditions selected for this analysis are a Mach number of 0.8395. The angle of attack, however, is varied from -5°to 10°. This range of angles of attack are used to highlight the difference in results between low-fidelity (based on DLM) and high-fidelity (based on CFD). The low-fidelity simulations are executed using the static aeroelastic analysis (SOL144), rigid CFD mesh analysis, and FSI simulations for two flexible structure cases.

6.1.1. Lift-coefficient Results

The lift-coefficient results of these simulations are shown in Figure 6.1. The figure shows an increase in the lift for positive angles of attack and a negative lift for negative angles of attack as expected from the pitching behavior. Moreover, since the wing is symmetric, zero lift is expected for α = 0.0°, this is visible for all simulation cases. It can be seen for increasing angle of attack, in the linear lift-curve regime the simulations based on DLM for aerodynamic modeling are different from the simulations based on CFD. From $\alpha \approx 8.0°$, however, the methods differ significantly more in lift-coefficient results. This difference is a result of the low-fidelity DLM method not modeling the viscosity, thickness effects, and rotationality of the flow present around stall conditions [26]. However, the governing equations modeled in CFD do allow these effects to be modeled, this causes stall and thus the drop in lift-coefficient to be visible [12, 25].

So the separation of flow around stall conditions is not modeled by DLM, but it is modeled by CFD. Figure 6.2 and Figure 6.3 show this ability of CFD to model separation effects. The figures show the behavior of the flow when stall kicks in for the top and bottom surfaces at $\alpha = 8°$ using the chordwise shear stress. In cases where the flow is attached to the wing, the direction of the shear stress points from leading to trailing edge. This is indicated by a positive shear stress. However, when the flow is detached, rotation of the flow leads to negative chordwise shear stress. In the figures it can be seen that the flow on the top wing surface is largely separated, which influences the aerodynamic performance of the wing. The flow on the bottom of the wing is experiencing the incoming air head on due to the large angle of attack. This results in the flow staying attached to the bottom wing surface as seen in Figure 6.3.

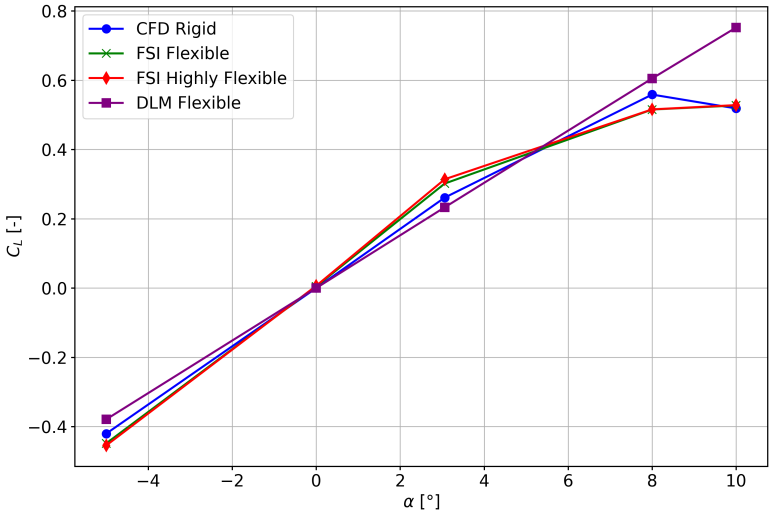

Figure 6.1: Comparison of C_L vs α for low- and high-fidelity frameworks.

Figure 6.4 and Figure 6.5 show the top and bottom wing surface shear stress distribution in chordwise direction for $\alpha = 10°$. The top wing surface distribution shows a significantly larger region of separation compared to Figure 6.2. Comparing the region of separation in Figure 6.2 and Figure 6.4 showcases that the separation region moves forward with increasing angle of attack from $8°$ to $10°$. Similarly, the chordwise shear force on the bottom of the wing in Figure 6.5 shows that the flow stays attached to the bottom wing surface. The increased region of separation on the top wing surface is what leads to the decrease in lift-coefficient when the stall angle of attack is reached and beyond.

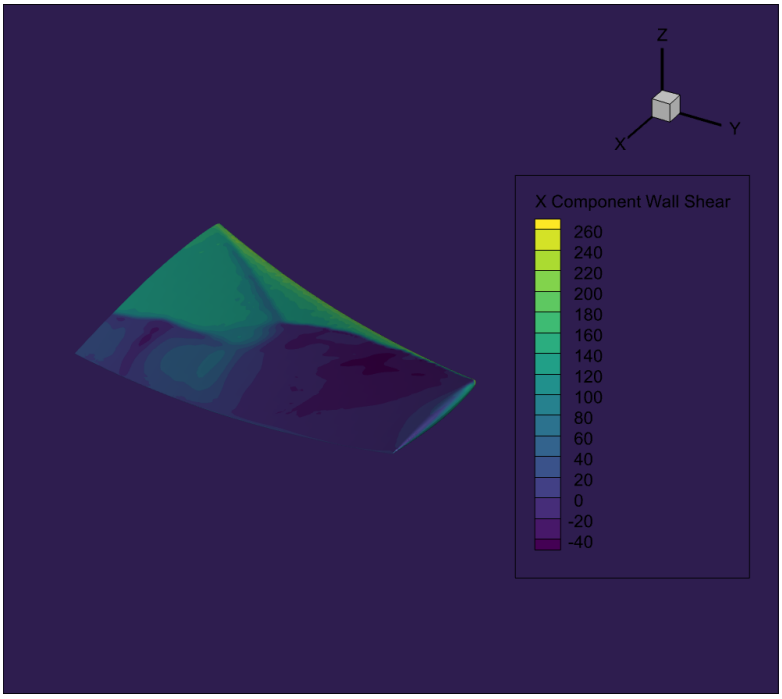

Figure 6.2: Shear stress chordwise-direction top wing $\alpha = 8°$.

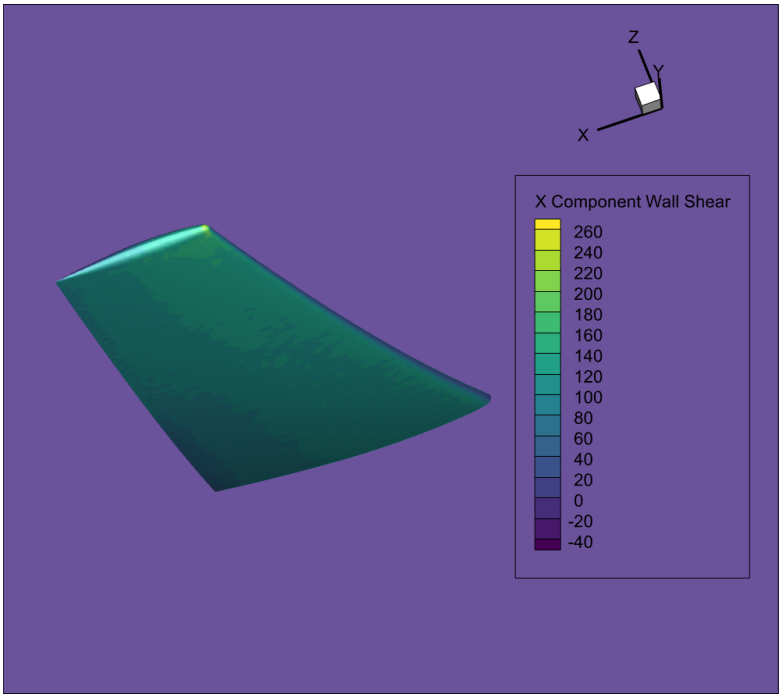

Figure 6.3: Shear stress chordwise-direction bottom wing $\alpha = 8°$.

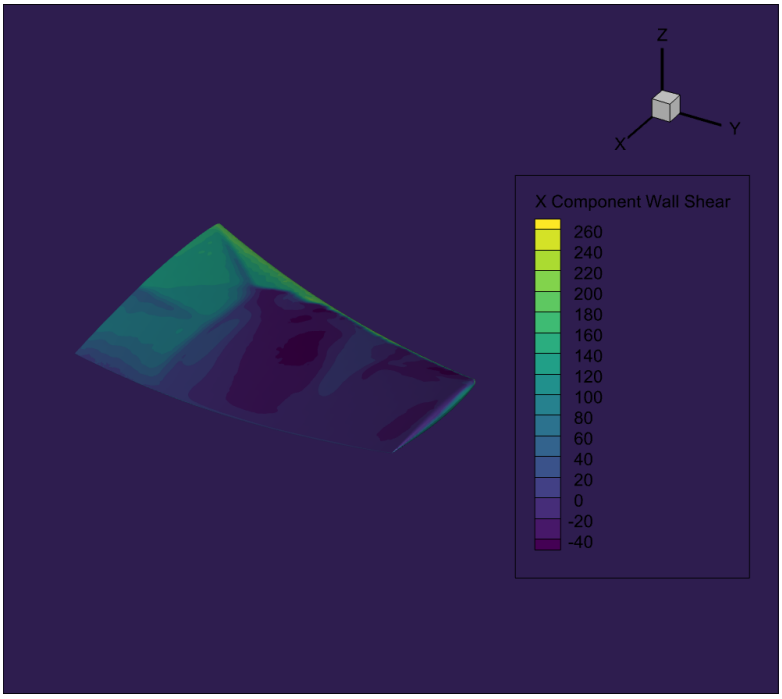

Figure 6.4: Shear stress chordwise-direction top wing $\alpha = 10°$.

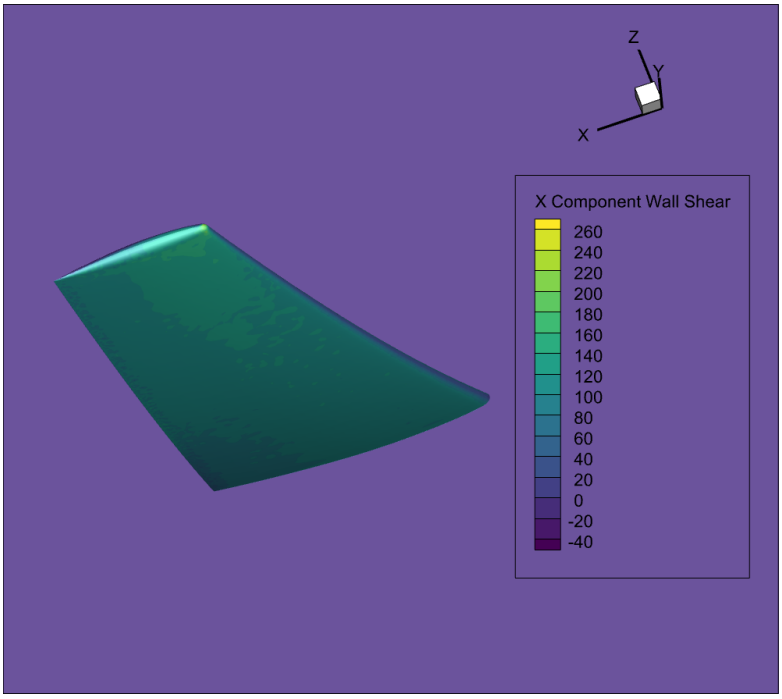

Figure 6.5: Shear stress chordwise-direction bottom wing $\alpha = 10°$.

The flow conditions around stall are highly unsteady. Modeling the unsteadiness of the flow with a steady CFD solver may result in fluctuating simulation results [27, 25]. This means the simulations give an indication of the flow behavior such as the separation of flow, but the solver might not find a steady state solution however. This effect of flow

unsteadiness on the convergence results can also be seen in Figure 6.6. This figure shows the residual criterion as discussed in section 5.6 for different simulation cases. The closer the angle of attack to stall conditions, the more prominent the unsteady effects. In Figure 6.6 this relationship can be observed in the small angles of attack α = 0° and α = 3.06°. In these simulations the residual decreases rapidly. However, the simulations with angles of attack α = 8° and α = 10° seem to converge at first, but deform again after a few FSI iterations. This shows the unsteadiness of the flow results, which are used to deform the structure. This confirms that these static FSI simulations will either not converge or converge slowly whilst oscillating when modeling unsteady flow phenomena. So in aeroelastic analysis around stall conditions, the simulations should be about analyzing the interaction of flow and structure in a dynamic sense. As steady solutions might not be found for these conditions.

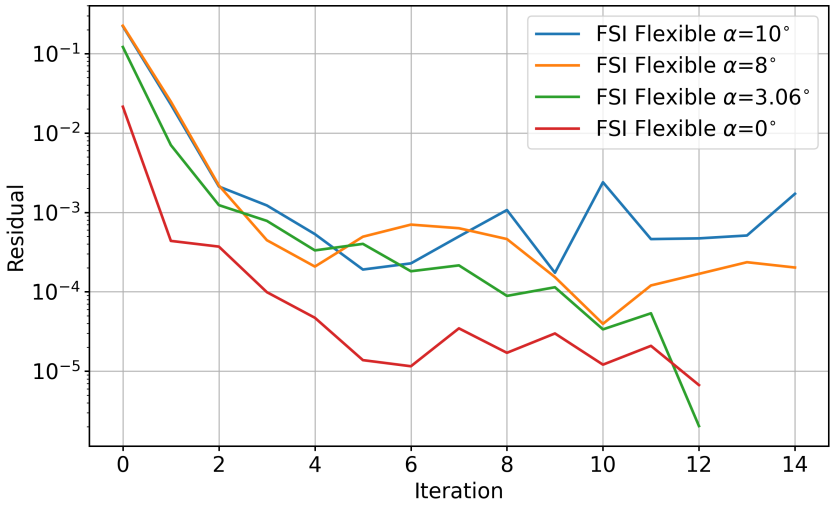

Figure 6.6: Residuals for simulation cases $M = 0.8395$.

Compared to the high-fidelity flexible simulations, the rigid CFD case exhibits a shallower lift-curve slope, resulting in lower lift-coefficients across both positive and negative angles of attack. In the flexible cases, the lift-curve slope is driven by aerodynamic mesh deformations resulting from FSI. Specifically, the wing's pitching behavior, detailed in section 4.4, changing the effective angle of attack (α_{eff}). This change in angle increases the lift coefficient. This enhancement continues until the internal structural loads reach an equilibrium that limits further deformation of the wing structure [12].

Figure 6.7 illustrates this spanwise increase in α_{eff} for the flexible and highly flexible cases at $M = 0.8395$ and $\alpha = 3.06°$. This highlights how structural stiffness influences the lift results. Furthermore, Figure 6.1 demonstrates that the highly flexible model produces a higher lift coefficient than the standard flexible case. This discrepancy arises because lower structural rigidity allows for greater wing deformation, further increasing the effective angle of attack. Detailed α_{eff} distributions for all remaining simulation cases are provided in Appendix B.

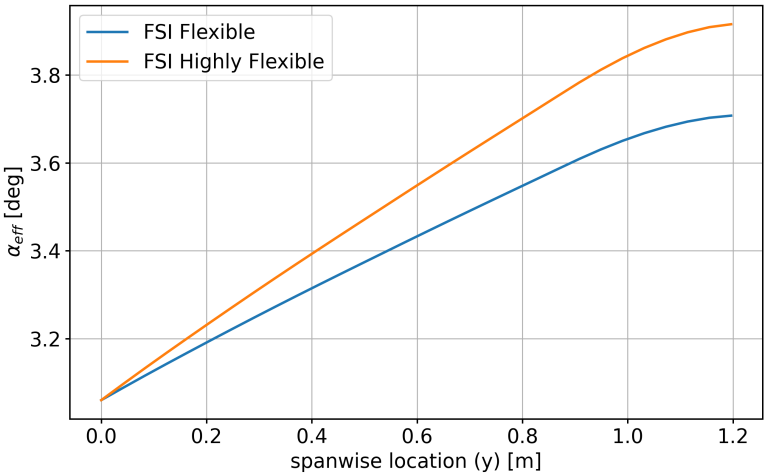

Figure 6.7: Effective angle of attack spanwise distribution $M = 0.8395$, $\alpha = 3.06°$ cases.

Figure 6.8 and Figure 6.9 illustrate the final aerodynamic and structural models after FSI simulations, respectively. These figures demonstrate that the highly flexible wing undergoes greater vertical displacement, driven by the increased lift associated with this case. This higher lift production is a direct consequence of a larger effective angle of attack compared to the standard flexible case. Both the flexible and highly flexible configurations are plotted against the initial, undeformed models to visualize the total deformation. The aerodynamic and structural states for all remaining simulation cases are documented in Appendix C.

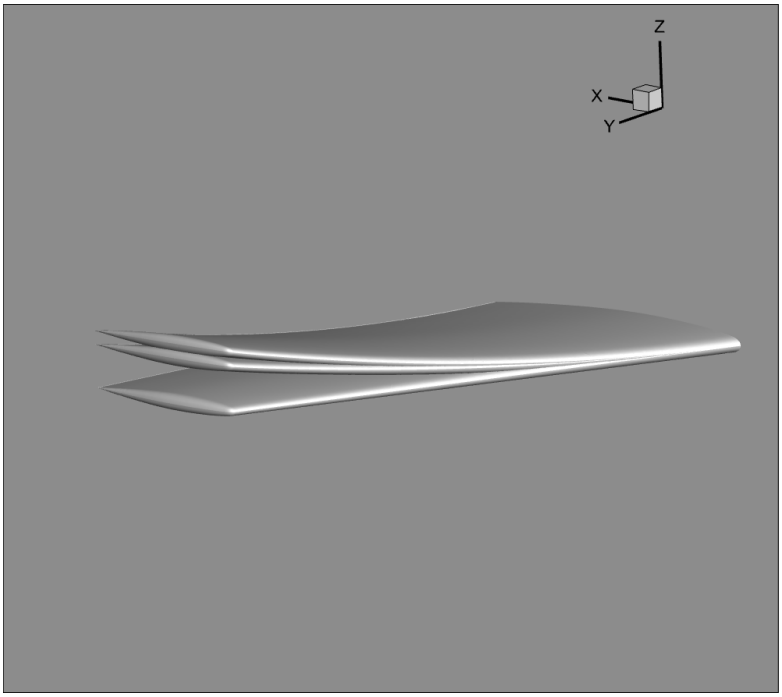

Figure 6.8: Comparison initial, Flexible and Highly Flexible (bottom to top) aerodynamic models for $\alpha = 3.06$ with $M = 0.8395$ case.

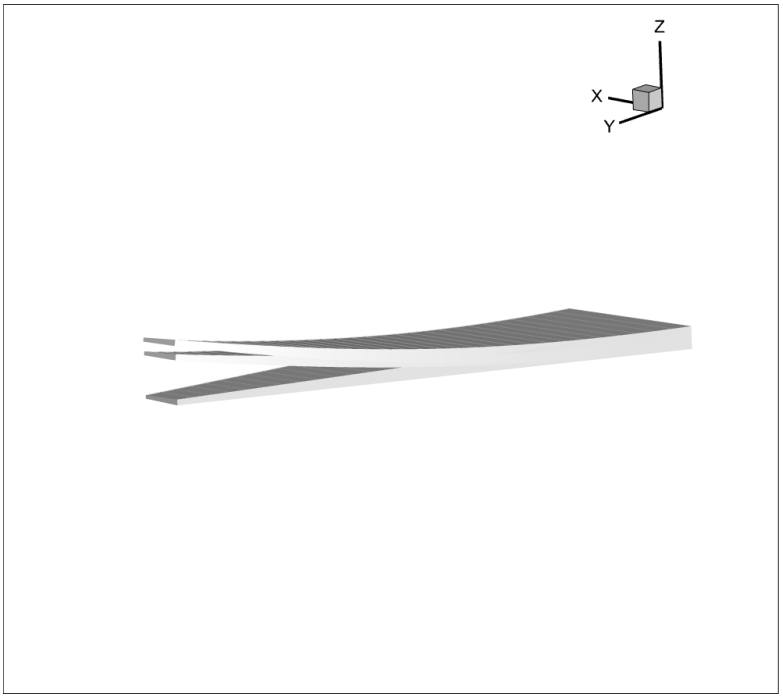

Figure 6.9: Comparison initial, Flexible and Highly Flexible (bottom to top) structural models for $\alpha = 3.06°$ with $M = 0.8395$ case.

6.1.2. Drag-coefficient Results

Figure 6.10 displays the drag-coefficient across a range of angles of attack. While the CFD-based results show close agreement, the DLM-based case deviates significantly. This discrepancy arises because the DLM only accounts for the induced drag of the wing [26]. This limitation is most apparent at $\alpha = 0.0°$, where the DLM predicts a drag coefficient of zero. In contrast, the high-fidelity simulations (rigid CFD, flexible FSI, and highly flexible FSI) capture both pressure and viscous drag components). This results in a realistic, non-zero drag-coefficient as a result of thickness effects of the wing for $\alpha = 0.0°$.

Furthermore, the increased angle of attack previously noted in Figure 6.7 leads to a corresponding rise in drag compared to the rigid CFD baseline. Consequently, as structural flexibility increases, the higher α_{eff} drives the elevated drag coefficients observed in the highly flexible case relative to the standard flexible FSI simulation.

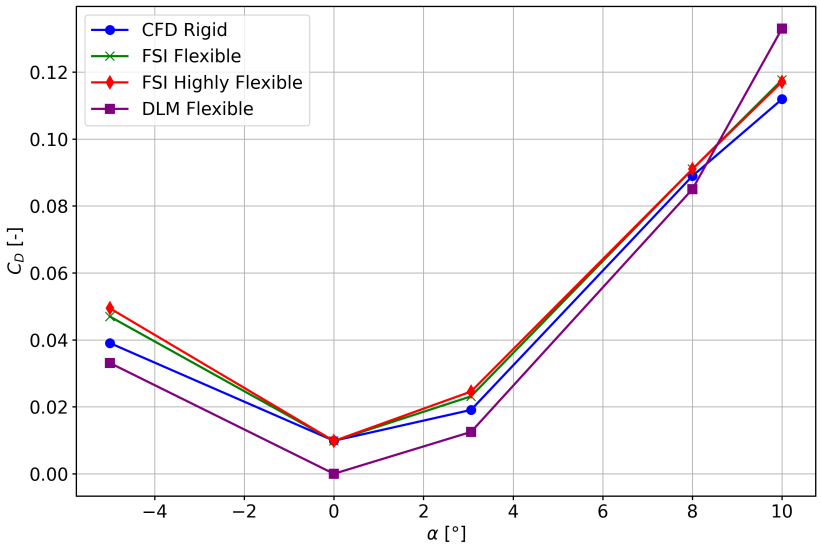

Figure 6.10: Comparison of C_D vs α for low- and high-fidelity frameworks.

6.1.3. Moment-coefficient Results

The relationship between the moment coefficient (C_M) and the angle of attack is illustrated in Figure 6.11. While the DLM simulations exhibit a characteristic linear trend, the high-fidelity methods capture a distinct departure from linearity as stall conditions are approached. This divergence is driven by the redistribution of aerodynamic forces. So as stall occurs, the shift in the suction peak and the growth of the separation region result in a simultaneous reduction in lift and an increase in drag. Consequently, this force distribution leads to an increase in the moment coefficient for flexible FSI simulations relative to the rigid CFD case.

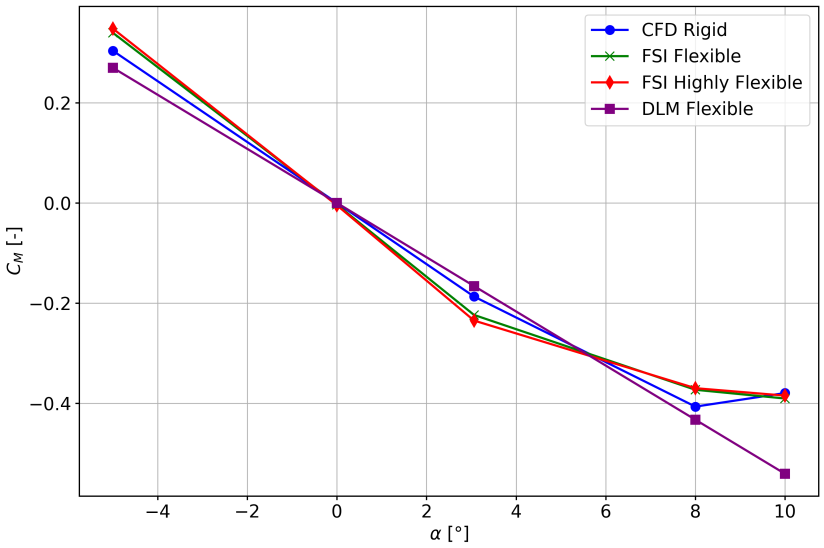

Figure 6.11: Comparison of C_M vs α for low- and high-fidelity frameworks.

6.1.4. Discussion of Angle of Attack Studies Results

To quantify the discrepancy between the low-fidelity (DLM) and high-fidelity (FSI) flexible results, the percentage difference for each aerodynamic coefficient is calculated using Equation 6.1 and summarized in Table 6.1. This comparison underscores the significance of the modeling approach on the resulting aerodynamic coefficients.

$$Difference(\%) = \frac{C_{FSI} - C_{DLM}}{C_{DLM}} \times 100\% \tag{6.1}$$

Table 6.1: Percentage difference between DLM and FSI flexible results per α case ($M = 0.8395$).

α [°]	C_L Diff (%)	C_D Diff (%)	C_M Diff (%)
-5.00	+18.28	+41.92	+25.88
0.00	N/A	N/A	N/A
3.06	+29.46	+85.12	+34.78
8.00	-14.82	+7.06	-13.81
10.00	-29.92	-11.53	-27.79

As indicated in the table, the deviations exceed 10% in nearly all cases, with particularly large discrepancies observed in the drag coefficient. These results reinforce the conclusion that when high predictive accuracy is required, particularly in regimes where non-linear flow phenomena are present, high-fidelity FSI simulations are necessary. The inherent theoretical assumptions of the DLM, such as the neglect of viscous effects and thickness-induced pressure drag, prevent it from capturing essential flow physics as detailed in the preceding sections.

Notably, at $\alpha = 0.0°$, the differences are listed as N/A because the DLM predicts zero drag and lift, rendering a percentage-based comparison mathematically undefined. Across the remaining range, the substantial variance in C_L and C_M confirms that the structural coupling in the FSI framework captures a significantly different aerodynamic state than the linearized DLM approach.

6.2. Mach Number Studies

The second set of simulations are carried out to primarily show influences of thickness and shock effects. The flight conditions selected for these analyses are α = 3.06°. The Mach number is varied from 0.5 to 0.95. This range of Mach numbers is used to show the increasing prominence of compressibility effects and how this is modeled by both low- and high-fidelity frameworks.

6.2.1. Lift-coefficient Results

Figure 6.12 shows the relationship between the lift-coefficient and the Mach number. It can be seen that for the CFD based methods the lift-coefficient increases until $M \approx 0.8395$, after this point the transonic shock gains strength causing a loss in lift [12, 41]. The lift-coefficient results for the DLM based simulations follow the same behavior until $M \approx 0.8395$ after which the results differ significantly. This is the result of the DLM based method not being able to model the said transonic shock effects. The low-fidelity framework uses a Prandtl-Glauert correction to account for compressibility effects. However, this method is not able to model the strength gain in the transonic shock. The correction is merely used to correct the lift-coefficient. It does not model the effect the transonic shock has on the pressure distribution over the wing [12, 36, 26].

The impact of structural flexibility on the lift-coefficient across varying Mach numbers is characterized by the coupling between aeroelastic deformation and aerodynamic loading. Because flexible structures are more susceptible to deformation, they exhibit a more pronounced pitch-up behavior. This behavior increases the effective angle of attack. In high-fidelity FSI simulations, this mechanism results in a higher lift-coefficient for more flexible structural configurations compared to their more rigid counterparts [1, 12].

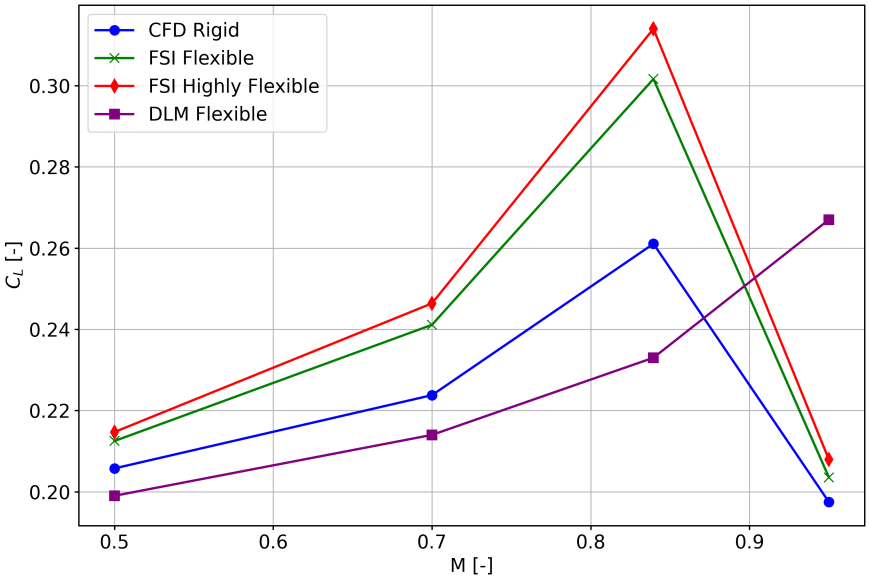

Figure 6.12: Comparison of C_L vs M for low- and high-fidelity frameworks.

The gain in strength of the transonic as mentioned above is shown in Figure 6.13 and Figure 6.16 by plotting the locations where the Mach number over the wing becomes sonic (M=1.0). The leading edge location where the Mach number becomes sonic indicates the location where the local flow velocity first reaches M=1.0 as a result of the thickness of the wing. This shows the presence of modeling of the thickness effects, which are neglected by DLM. On the contrary, the aft location of the sonic iso-surface indicates the location where the shock on the wing leads to a decrease in Mach number, resulting in a local flow velocity smaller than M=1.0 [1].

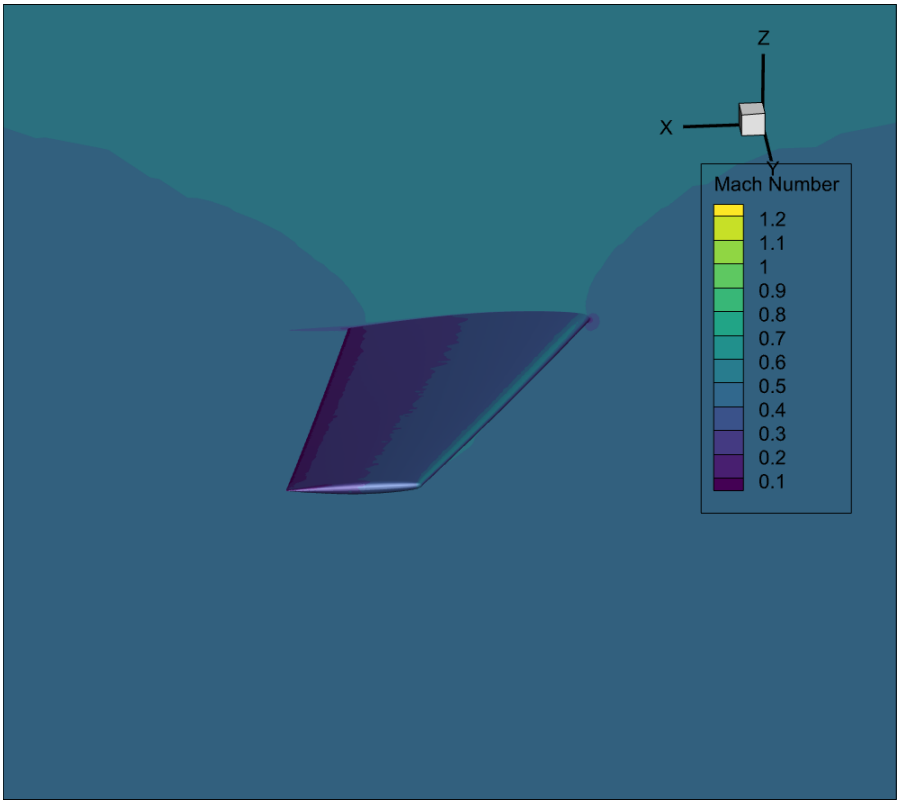

Figure 6.13: M=1.0 iso-surface M=0.5, α = 3.06° Flexible case.

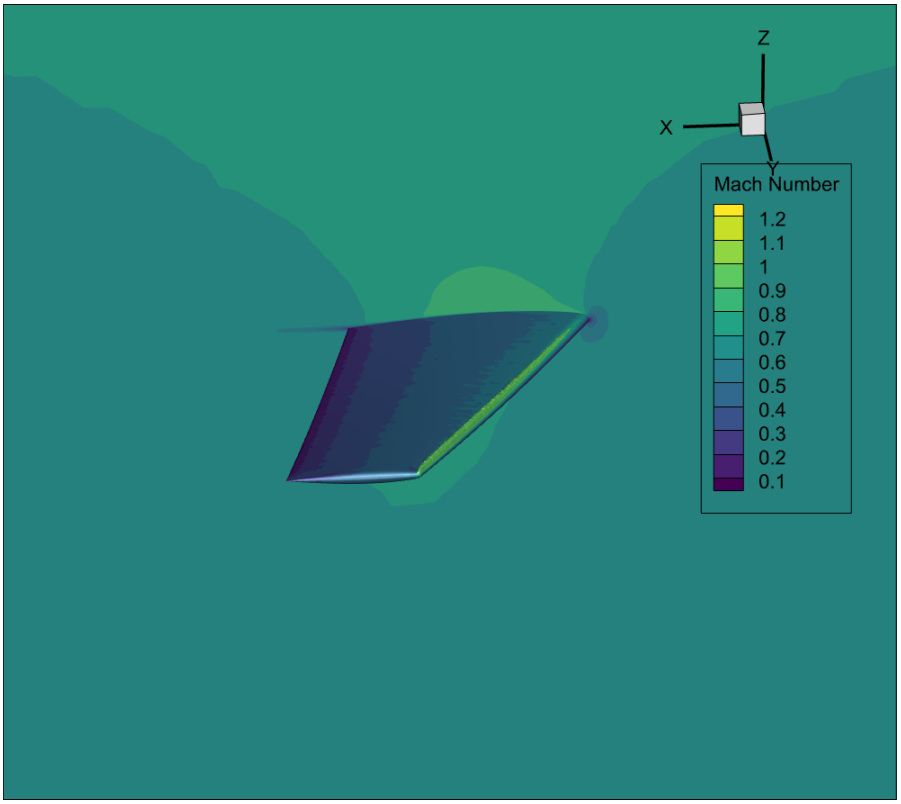

Figure 6.14: M=1.0 iso-surface M=0.7, α = 3.06° Flexible case.

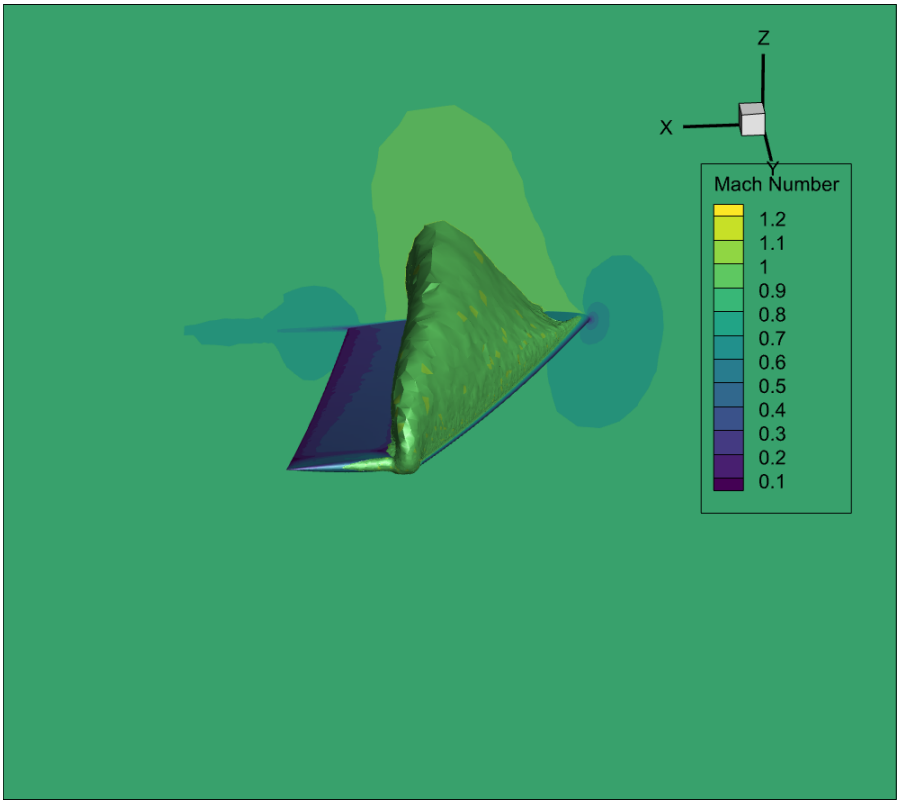

Figure 6.15: M=1.0 iso-surface M=0.8395, α = 3.06° Flexible case.

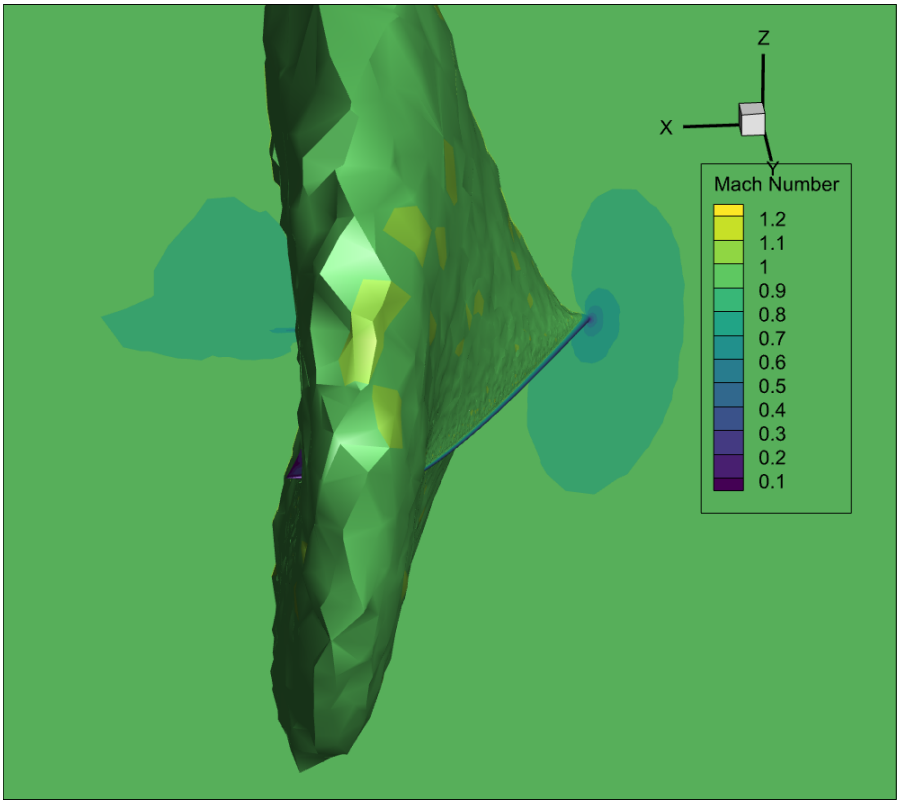

Figure 6.16: M=1.0 iso-surface M=0.95, α = 3.06° Flexible case.

The pitching behavior of the wing causes the wing to pitch up more with increasing Mach number as can be seen in Appendix B. The more flexible the structure, the easier it twists under the same loads. The increase in spanwise effective angle of attack for $M = 0.5, 0.7$ and 0.8395 causes the lift-coefficient to increase, until the shock is located on top of the entire wing as shown in Figure 6.16. At this point the shock causes a significant disruption of the flow and circulation needed for lift generation. This results in a significant loss of lift compared to the previous flow conditions [41].

6.2.2. Drag-coefficient Results

The variation of the drag coefficient (C_D) with respect to the Mach number is presented in Figure 6.17. A notable divergence between the DLM and CFD-based frameworks is observed, with the discrepancy in drag magnitude increasing significantly as the Mach number rises. This trend is a direct consequence of the differing aerodynamic fidelity within each method. While the DLM is restricted to modeling lift-induced drag, the CFD solver accounts for the full suite of transonic flow physics. Specifically, in the transonic regime, the intensification of shock waves introduces a substantial wave-drag component [1]. Since this wave-drag is captured exclusively by the high-fidelity CFD approach, the performance gap between the two methods becomes most pronounced at higher Mach numbers, where compressibility effects dominate [25, 26].

The incremental drag observed in the flexible and highly flexible cases is a direct consequence of the aeroelastic coupling inherent in these models. The structural pitch-up behavior, intensified by lower torsional stiffness, forces a spanwise increase in the effective angle of attack. This deformation triggers a multifaceted rise in total drag comprised of viscous-, pressure-, induced- and wave-drag [1, 12].

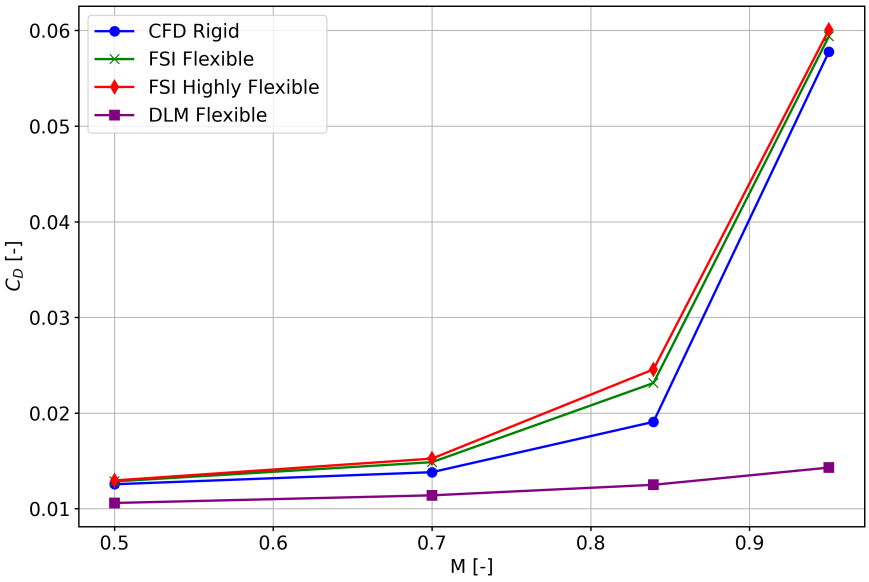

Figure 6.17: Comparison of C_D vs M for low- and high-fidelity frameworks.

6.2.3. Moment-coefficient Results

The variation of the moment coefficient (C_M) with Mach number is presented in Figure 6.18. Consistent with the trends observed in the lift coefficient (Figure 6.12), the discrepancy between the DLM and CFD-based results intensifies as the Mach number increases. Beyond $M \approx 0.8395$, a significant divergence occurs. The DLM prediction continues to decrease, whereas the high-fidelity CFD results exhibit a reversal in the trend. This behavior is attributed to the inherent inability of the DLM to account for the shock-induced pressure redistribution that occurs as the transonic shock intensifies, a phenomenon previously detailed in subsection 6.2.1 [12, 26]. Furthermore, the impact of structural deformation is evident. Increasing the flexibility of the wing results in a higher effective angle of attack. This higher α_{eff} increases the magnitude of the moment around the origin subsequently making the moment-coefficient more negative (Appendix B).

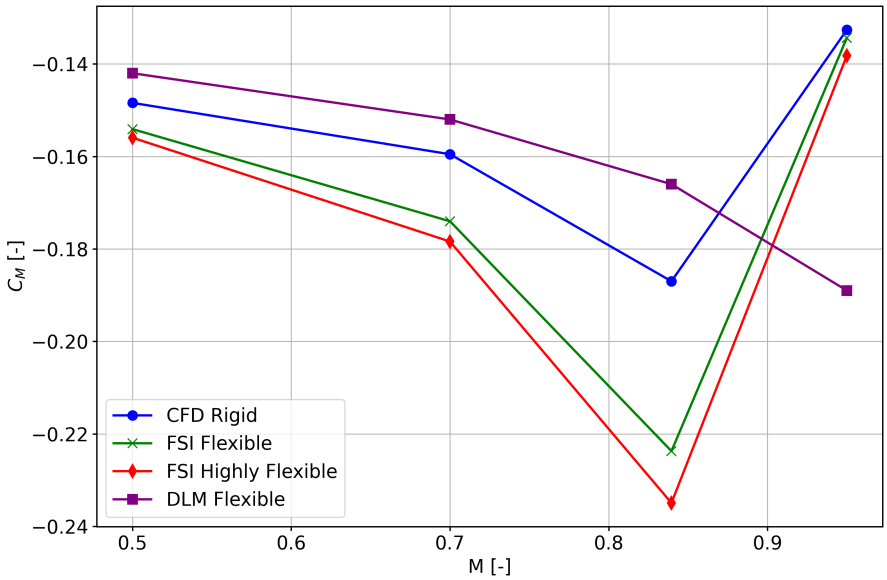

Figure 6.18: Comparison of C_M vs M for low- and high-fidelity frameworks.

The impact of low-fidelity modeling assumptions on static aeroelastic results for the flexible wingbox (M=0.8395, $\alpha = 3.06°$) is illustrated in Figure 6.19. A clear discrepancy exists between the low- and high-fidelity frameworks regarding vertical deformation. This difference is driven by the spanwise vertical force distribution as seen in Figure 6.20. The force distribution reveals that the low-fidelity model underestimates lift compared to CFD-based high-fidelity simulations. Consequently, the high-fidelity FSI wing experiences significantly higher vertical loading, particularly near the tip, leading to a more pronounced structural deformation in the vertical direction.

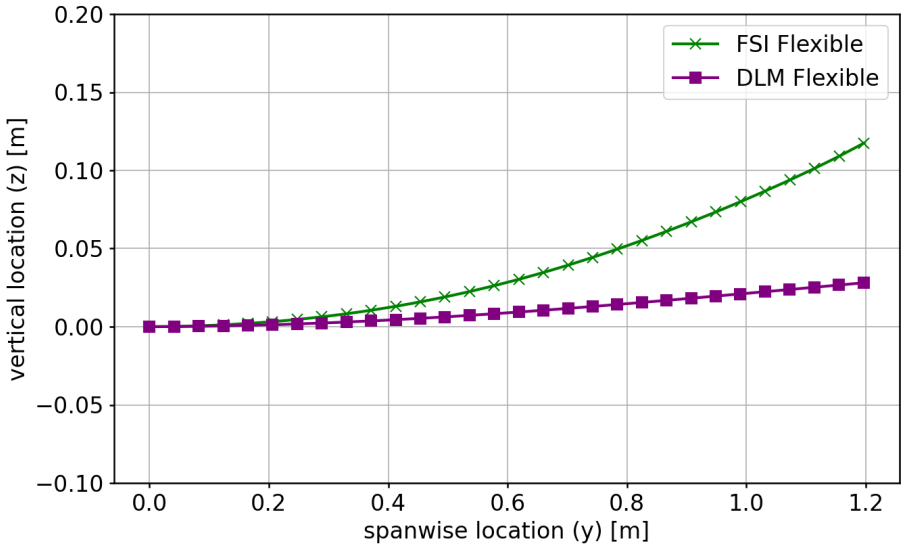

Figure 6.19: Comparison centerline deformation of wing DLM Flexible and FSI Flexible case.

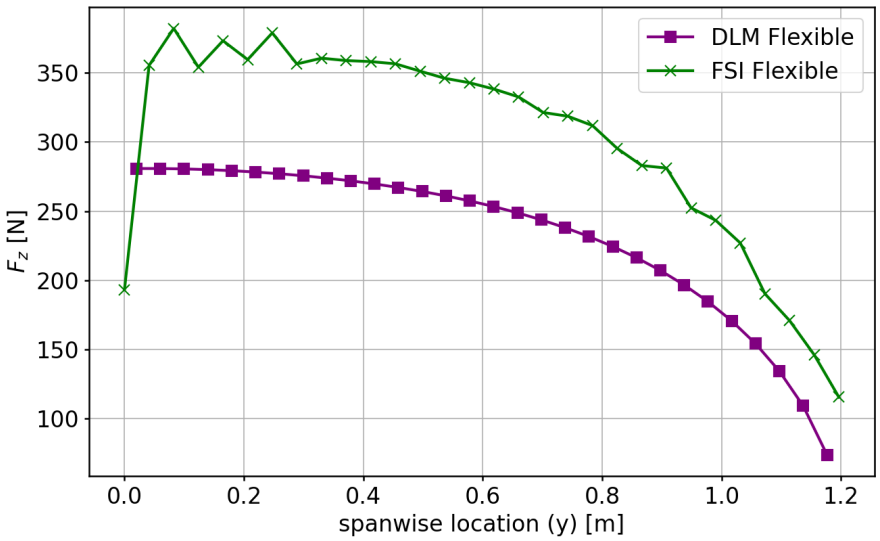

Figure 6.20: Comparison spanwise vertical force distribution DLM Flexible and FSI Flexible case.

Figure 6.21 shows the impact of the different solver fidelities on the chordwise ΔC_p distribution. The distribution is shown at the wing semi-span for the rigid CFD and flexible DLM and FSI cases at $M = 0.8395$, $\alpha = 3.06°$. A significant discrepancy in pressure-distribution is observed between the low- and high-fidelity results. This discrepancy is the result of the linearized assumptions of DLM, the CFD-based approach captures these complex aerodynamic effects such as localized jumps in the ΔC_p distribution which cannot be resolved using DLM.

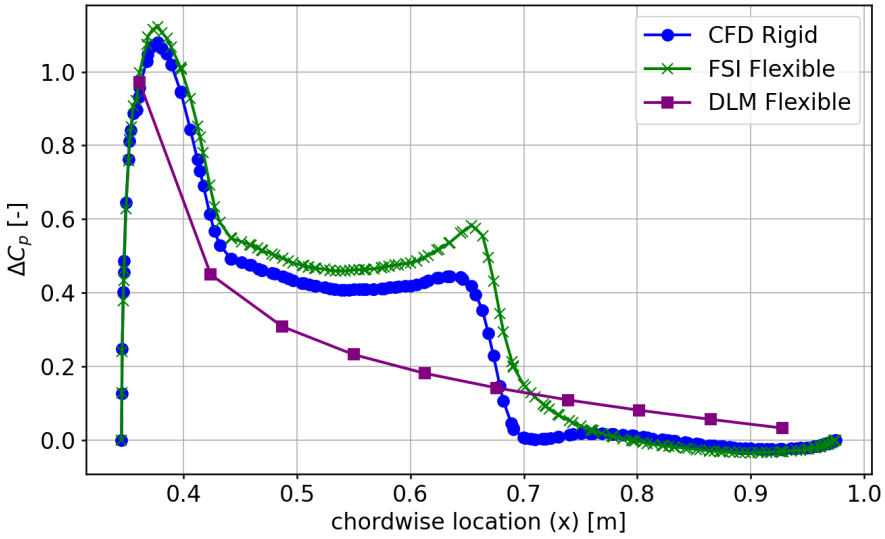

Figure 6.21: Comparison of ΔC_p vs chordwise location at semi-span.

6.2.4. Discussion of Mach Number Studies Results

The discrepancies between the low-fidelity (DLM) and high-fidelity (FSI) flexible results are quantified in Table 6.2, with values calculated via the metric defined in Equation 6.1. This comparison underscores how the choice of aerodynamic modeling approach fundamentally alters the predicted aerodynamic coefficients across different flow regimes.

As demonstrated in the table, deviations exceed 10% in nearly all cases, with the most extreme discrepancies occurring in the drag coefficient at high Mach numbers. At $M = 0.95$, the difference reaches a staggering +315.22%, a direct consequence of the DLM's inability to account for transonic wave drag. Becuase the DLM is based on linearized potential flow theory, it inherently neglects the strength and positioning of the transonic shock [26]. These effects are the primary drivers for the aerodynamic loads in these cases.

The variance remains substantial for lift- and moment-coefficients as well. These findings confirm that the aerodynamic solver within the FSI framework captures the essential non-linear physics of high-speed flow.

Furthermore, the results indicate that the structural deformation itself is strongly dictated by the fidelity of the aerodynamic model. Becuase the pressure distribution varies signficantly between the two methods, the resulting loads on the wingbox structure differ, leading to distinct aeroelastic equilibirium states. This highlights that using a high-fidelity aerodynamic modeling method is not only necessary for accurate force prediction, but also for correctly modeling the structural response of the wing.

Table 6.2: Percentage difference between DLM and FSI flexible results per M case ($\alpha = 3.06°$).

Mach	C_L **Diff (%)**	C_D **Diff (%)**	C_M **Diff (%)**
0.50	+6.78	+21.26	+8.51
0.70	+12.67	+30.39	+14.49
0.8395	+29.46	+85.12	+34.78
0.95	-23.77	+315.22	-28.92

7

Conclusions and Recommendations

This chapter presents the conclusions and recommendations derived from the the development process and research findings. These conclusions evaluate the accuracy of the Fluid-Structure Interaction (FSI) framework and serve as the foundation for future work. Additional recommendations address potential implementations that were not completed due to time constraints within the research period.

7.1. Conclusions

The objective of coupling the aerodynamic and structural solvers was successfully achieved through the development of a partitioned FSI framework. This specific approach ensures the framework remains applicable to a wide variety of engineering problems. The system operates by receiving simple inputs, such as names of wing zones required for force extraction, and then outputting force locations and vectors to the framework. This process is reinforced by a detailed User-Defined Function (UDF) to ensure reliability. To maintain generality across various simulation files, the framework utilizes functions that take minimal input, which minimalizes the points of failure. The loads obtained through aerodynamic simulations are projected on the loading line of the wingbox structure through Nearest-Neighbor (NN) interpolation. The loads on this load line are introduced to the structure through Rigid Body Elements (RBE's) which function as a skeleton of the wingbox structure. The result of the structure simulation is a translation and rotation of the load line which is followed by the wing surface (CFD mesh). This deformation of the CFD mesh is executed through a UDF, which again, takes the deformation of the loading line and the aerodynamic names of the wing zones as input. These force extraction and mesh deformation UDF's together with the structural line load application is what generalizes the process of the static FSI iteration independent of simulated problem. The script for this process also contains detailed error messages to warn the user of where the process is going wrong. This generalization is what makes the difference in the static FSI framework compared to previous work.

The simulation results reveal a distinct difference between low-fidelity (DLM) results and the high-fidelity (CFD-based methods) results generated by the framework developed in this research project. Low-fidelity methods typically neglect non-linear effects such as viscosity, rotationality, separation and shock development. As shown in chapter 6, ignoring these factors leads to discrepancies in flow predictions, and it impacts the simulated behavior of the wing. Discrepancies are particularly noticeable for drag predictions for small angles and high Mach numbers. In general, the difference be-

tween low- and high-fidelity results discussed in this report are in the order of 10's of %, except for outliers. Overall, it can be concluded that low-fidelity static aeroelastic modeling is valid for small Mach numbers for $\alpha \neq 0.0°$. For cases around $\alpha = 0.0°$, high Mach numbers and stall conditions, high-fidelity static FSI is superior.

7.2. Recommendations

The current static FSI framework utilizes relative mesh deformations to update the aerodynamic model. This method is less effective for periodic motions typically found in dynamic simulations. To prepare for future dynamic analysis, the framework should be extended to include absolute mesh deformations. This change will better preserve mesh quality during oscillating motions of a wing surface in transient analysis of for example flutter. Additionally, the static framework should be tested for more flexible wing models. In these models the importance of non-linear structural effects become increasingly important and the effect of linear and non-linear modeling methods can be assessed.

As the current framework is limited to steady-state problems, future work should focus on modeling transient behavior. Moving from static to dynamic capabilities requires achieving FSI convergence at every timestep. Although the core building blocks for load application and mesh displacement remain the same and can be re-used, the framework will require more robust data handling. Specifically, a strongly coupled dynamic framework must compute sub-iterations that rely on information from previous FSI cycles. Instead of relying solely on CSV files for force and deformation data, the system should use internal computer memory to store and retrieve solver information more efficiently. Subsequently, the structural response that occurs under load should be investigated through linear and non-linear transient solution methods.

Once the static and dynamic frameworks are refined, the system should be tested using a more flexible wing model like the uCRM9. This model features a significantly higher aspect ratio than the model used in this study. These dynamic results should be compared to the static results to inspect whether the wing converges to the steady-state position obtained in static FSI simulations. Finally, because the accuracy of the flow is dictated by the quality of the aerodynamic mesh, an algorithm should be developed to automatically assess mesh metrics in critical regions. This type of analysis would proved essential context regarding the reliability of the simulation results.

References

[1] John D.. Anderson. *Introduction to flight*. English. 8th ed. New York: McGraw-Hill Education, 2016, p. 910. ISBN: 9780078027673.

[2] Michael V. Cook. “Flying and Handling Qualities”. In: *Flight Dynamics Principles* (Jan. 2013), pp. 259–291. DOI: 10.1016/B978-0-08-098242-7.00010-9. URL: https://www.sciencedirect.com/science/article/pii/B9780080982427000109.

[3] Xueli Xiong et al. “Aviation and carbon emissions: Evidence from airport operations”. In: *Journal of Air Transport Management* 109 (June 2023), p. 102383. ISSN: 0969-6997. DOI: 10.1016/J.JAIRTRAMAN.2023.102383. URL: https://www.sciencedirect.com/science/article/pii/S0969699723000261.

[4] David Hyde et al. *Aviation Benefits Beyond Borders December 2024*. English. Tech. rep. Chicago: Air Transport Action Group, Dec. 2024. URL: https://aviationbenefits.org/downloads/aviation-benefits-beyond-borders-2024/.

[5] EUROCONTROL. *GAES-Future Engine Technology Environmental Impact Study*. Tech. rep. EUROCONTROL, 2005. URL: https://www.eurocontrol.int/publication/gaes-future-engine-technology-environmental-impact-study.

[6] R.P.L. Nijssen. *Composite Materials an introduction*. English. Tech. rep. Den Haag: Hogeschool Inholland, Jan. 2015.

[7] Edson Cocchieri Botelho, Evandro Luís Nohara, and Mirabel Cerqueira Rezende. “Lightweight structural composites with electromagnetic applications”. In: *Multifunctionality of Polymer Composites: Challenges and New Solutions* (May 2015), pp. 419–434. DOI: 10.1016/B978-0-323-26434-1.00012-X. URL: https://www.sciencedirect.com/science/article/pii/B978032326434100012X.

[8] Shahrzad Daghighi. *Structurally Efficient Composite Super Ellipsoidal Shells*. Tech. rep. URL: https://www.researchgate.net/publication/353890059.

[9] Andrea Magrini et al. “A review of installation effects of ultra-high bypass ratio engines”. In: *Progress in Aerospace Sciences* 119 (Nov. 2020), p. 100680. ISSN: 0376-0421. DOI: 10.1016/J.PAEROSCI.2020.100680. URL: https://www.sciencedirect.com/science/article/pii/S0376042120300920.

[10] Serhan Ahmet Cihangir, Hakan Aygun, and Onder Turan. “Energy and performance analysis of a turbofan engine with the aid of dynamic component efficiencies”. In: *Energy* 260 (Dec. 2022), p. 125085. ISSN: 0360-5442. DOI: 10.1016/J.ENERGY.2022.125085. URL: https://www.sciencedirect.com/science/article/pii/S0360544222019806.

[11] Intergovernmental Panel on Climate Change. *Special Report on Aviation and the Global Atmosphere*. Accessed: 2025-05-11. 1999. URL: https://archive.ipcc.ch/ipccreports/sres/aviation/index.php?idp=92.

[12] Daniel Piñeiro Rielo et al. *Escola de Enxenería Industrial Development of a Fluid-Structure Interaction simulation algorithm for the analysis of the static and dynamic response of an airplane wing*. Tech. rep. 2019.

[13] Yiyuan Ma and Ali Elham. "Designing high aspect ratio wings: A review of concepts and approaches". In: *Progress in Aerospace Sciences* 145 (Feb. 2024), p. 100983. ISSN: 0376-0421. DOI: 10.1016/J.PAEROSCI.2024.100983. URL: https://www.sciencedirect.com/science/article/pii/S0376042124000095.

[14] Hans-Joachim Bungartz et al. *Fluid-Structure Interaction Modelling, Simulation,Optimization*. English. Tech. rep. München and Darmstadt: Technische Universität München and Technische Universität Darmstadt, Mar. 2006.

[15] MSC Software. *MSC Nastran 2021 Reference Guide*. English. Tech. rep. Newport Beach: MSC Software, July 2021. URL: http://msc-documentation.questionpro.com..

[16] A. R. Collar. "The first 50 years of aeroelasticity". English. In: (Dec. 1977).

[17] Federal Aviation Administration. *Lessons Learned from Civil Aviation Accidents: Braniff Airways Flight 542 (N9705C)*. https://www.faa.gov/lessons_learned/transport_airplane/accidents/N9705C. Accessed: Friday 8th May, 2026. 2024.

[18] Terrence A. Weisshaar. "Static and Dynamic Aeroelasticity". English. In: *Encyclopedia of Aerospace Engineering* 1 (Jan. 2010). DOI: 10.1002/9780470686652.eae149.

[19] J. R. Wright and J. E. Cooper. *Introduction to Aircraft Aeroelasticity and Loads*. English. Ed. by P. Belobaba, J. Cooper, and A. Seabridge. 2nd ed. Chichester: Wiley, Dec. 2014.

[20] C Farhat, M Lesoinnea, and P Letallecb. *Computer methods in applied mechanics and engineerlng Load and motion transfer algorithms for fluid/ structure interaction problems with non-matching discrete interfaces: Momentum and energy conservation, optimal discretization and application to aeroelasticity*. English. Tech. rep. Domaine de Voluceau: Elsevier, Oct. 1996, pp. 95–109.

[21] Hexagon Nexus. *MSC FlightLoads 2021.4 User's Guide*. English. Tech. rep. Irvine: Hexagon Nexus, July 2021. URL: https://simcompanion.hexagon.com/.

[22] Frederico Afonso et al. "Comparison of low, medium and high fidelity numerical methods for unsteady aerodynamics and nonlinear aeroelasticity". In: *Journal of Fluids and Structures* 91 (Nov. 2019), p. 102744. ISSN: 03760421. DOI: 10.1016/j.paerosci.2016.12.004. URL: https://doi.org/10.1016/j.paerosci.2016.12.004.

[23] Zhuming Bi. “Applications—Multiphysics Systems”. In: *Finite Element Analysis Applications* (Jan. 2018), pp. 407–453. DOI: 10.1016/B978-0-12-809952-0.00011-X. URL: https://www.sciencedirect.com/science/article/pii/B978012809952000011X.

[24] Martin Lacroix et al. “A comparative study of interpolation algorithms on non-matching meshes for PFEM-FEM fluid-structure interactions”. In: *Computers & Mathematics with Applications* 155 (Feb. 2024), pp. 51–65. ISSN: 0898-1221. DOI: 10.1016/J.CAMWA.2023.11.045. URL: https://www.sciencedirect.com/science/article/pii/S0898122123005485.

[25] J. D. Anderson. *Computational Fluid Dynamics The Basics with Applications*. English. 1st ed. Maryland: McGraw-Hill, Jan. 1995.

[26] E. Albano and W. P. Rodden. “A doublet-lattice method for calculating lift distributions on oscillating surfaces in subsonic flows.” In: *AIAA Journal* 7.2 (Feb. 1969), pp. 279–285. ISSN: 0001-1452. DOI: 10.2514/3.5086.

[27] ANSYS Inc. *ANSYS Fluent Users Guide*. English. Tech. rep. Canonsburg: Ansys, July 2025. URL: http://www.ansys.com.

[28] ANSYS Inc. *Ansys Fluent User’s Guide*. Tech. rep. 2025. URL: http://www.ansys.com.

[29] ANSYS Inc. *ANSYS FLUENT UDF Manual*. English. Tech. rep. 14.0. Canonsburg: ANSYS, Nov. 2011.

[30] ANSYS Inc. *PyFluent documentation 0.38.1*. https://fluent.docs.pyansys.com/. Accessed: 04-05-2026. 2026. URL: https://fluent.docs.pyansys.com/.

[31] T.H.G. Megson. “Chapter 6 - Matrix methods”. In: *Aircraft Structures for Engineering Students (Sixth Edition)*. Sixth Edition. Butterworth-Heinemann, 2017, pp. 183–230. ISBN: 978-0-08-100914-7. DOI: https://doi.org/10.1016/B978-0-08-100914-7.00006-2. URL: https://www.sciencedirect.com/science/article/pii/B9780081009147000062.

[32] Steven Doyle et al. *SteveDoyle2/pyNastran*. https://github.com/SteveDoyle2/pyNastran. Apr. 2026. URL: https://github.com/SteveDoyle2/pyNastran.

[33] Delft University of Technology. *HPC Wiki: Main Page*. https://hpcwiki.tudelft.nl/index.php?title=Main_Page. 2025.

[34] Advisory Group for Aerospace Research and Development (AGARD). *AGARD Advisory Report No. 138*. English. Tech. rep. London: North Alantic Treaty Organization, May 1979.

[35] D. Lakhshmanan et al. “Computational fluid dynamics simulation on aerodynamic characteristics of ONERA M6-wing”. In: *Materials Today: Proceedings* 47 (Jan. 2021), pp. 2194–2199. ISSN: 22147853. DOI: 10.1016/j.matpr.2021.06.042. URL: https://www.sciencedirect.com/science/article/pii/S221478532104387X.

[36] J.K.S. Dillinger. *Static Aeroelastic Optimization of Composite Wings with Variable Stiffness Laminates*. English. Tech. rep. Delft: Delft University of Technology, June 2014.

[37] T.H.G. Megson. “Shear of Beams”. In: *Introduction to Aircraft Structural Analysis* (2010), pp. 479–501. DOI: 10.1016/b978-1-85617-932-4.00016-6. URL: https://www.sciencedirect.com/science/chapter/monograph/abs/pii/B9781856179324000166.

[38] Mainak Kundu et al. *ansys/pyfluent*. https://github.com/ansys/pyfluent. Apr. 2026. URL: https://github.com/ansys/pyfluent.

[39] M. D. Hill and M. R. Marty. “Amdahl’s Law in the Multicore Era”. English. In: *IEEE Computer Society* 1 (July 2008), pp. 33–38.

[40] David C.. Lay, Steven R.. Lay, and Judith. McDonald. *Linear algebra and its applications*. Pearson, 2016. ISBN: 9780321982384.

[41] Yun TIAN et al. “Transonic buffet control research with two types of shock control bump based on RAE2822 airfoil”. In: *Chinese Journal of Aeronautics* 30.5 (Oct. 2017), pp. 1681–1696. ISSN: 1000-9361. DOI: 10.1016/J.CJA.2017.07.011. URL: https://www.sciencedirect.com/science/article/pii/S1000936117301863.

A

CFD Convergence Studies Results

The results in this appendix provide a comparison of pressure coefficient (C_p) distributions for the $M = 0.8395$, $\alpha = 3.06°$ test case, evaluated across three distinct mesh densities. By benchmarking these numerical results against experimental data, the figures demonstrate the sensitivity of the flow solution, particularly regaring shock capture and pressure gradients, to spatial discretization. These comparisons form the empirical foundations for the grid convergence studies detailed in subsection 4.2.3. Each subsequent plot illustrates the chordwise C_p distribution at a specific spanwise station, allowing for a localized assessment of solver performance. Ultimately, this comparative analysis was one of the decisive factors in selecting the CFD mesh used for high-fidelity FSI simulations.

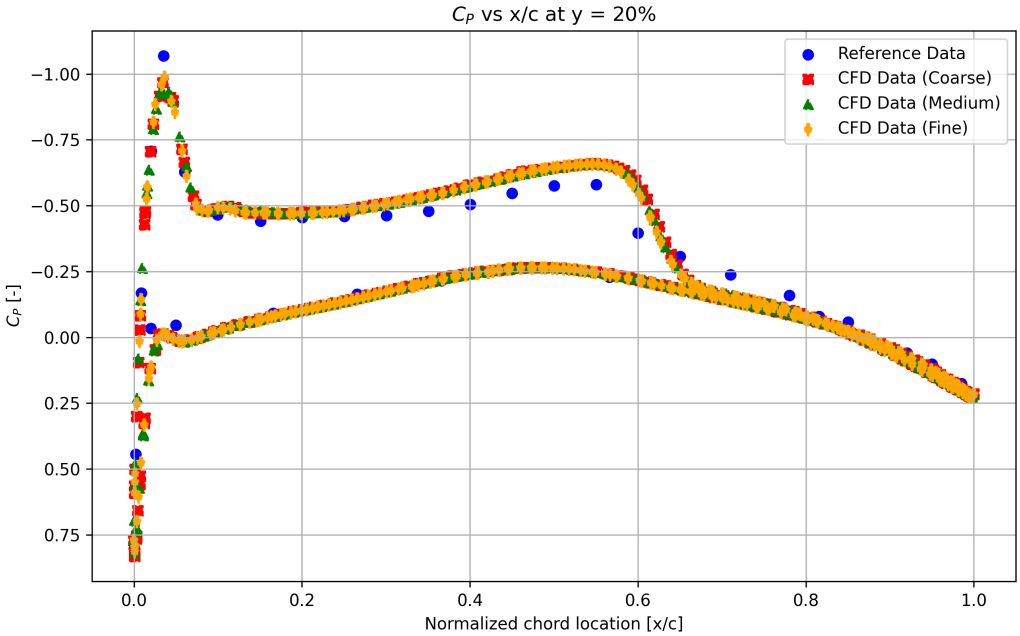

Figure A.1: Pressure coefficient results 20% span.

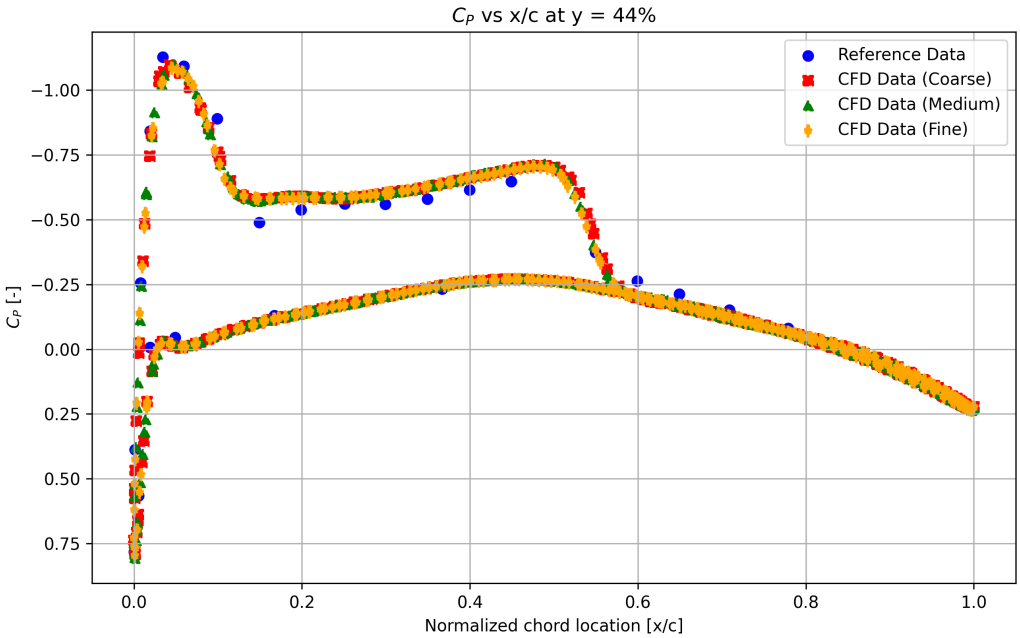

Figure A.2: Pressure coefficient results 44% span.

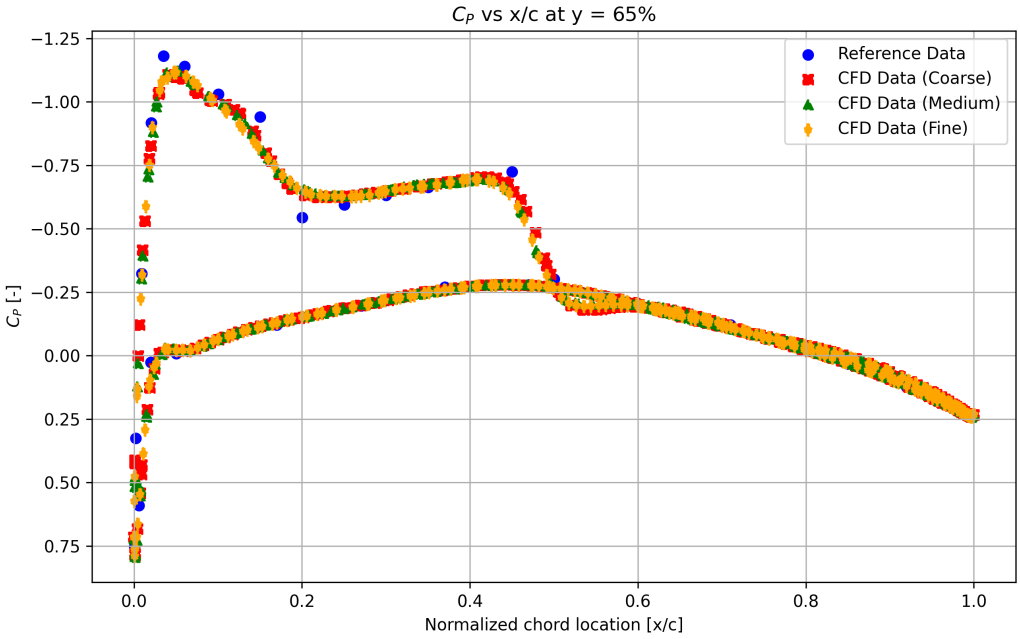

Figure A.3: Pressure coefficient results 65% span.

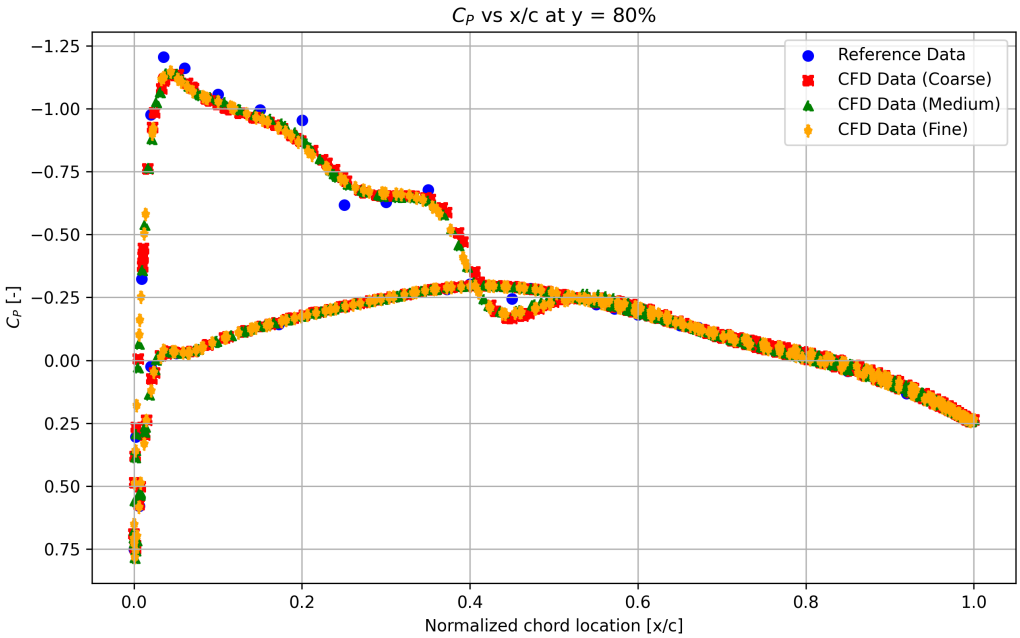

Figure A.4: Pressure coefficient results 80% span.

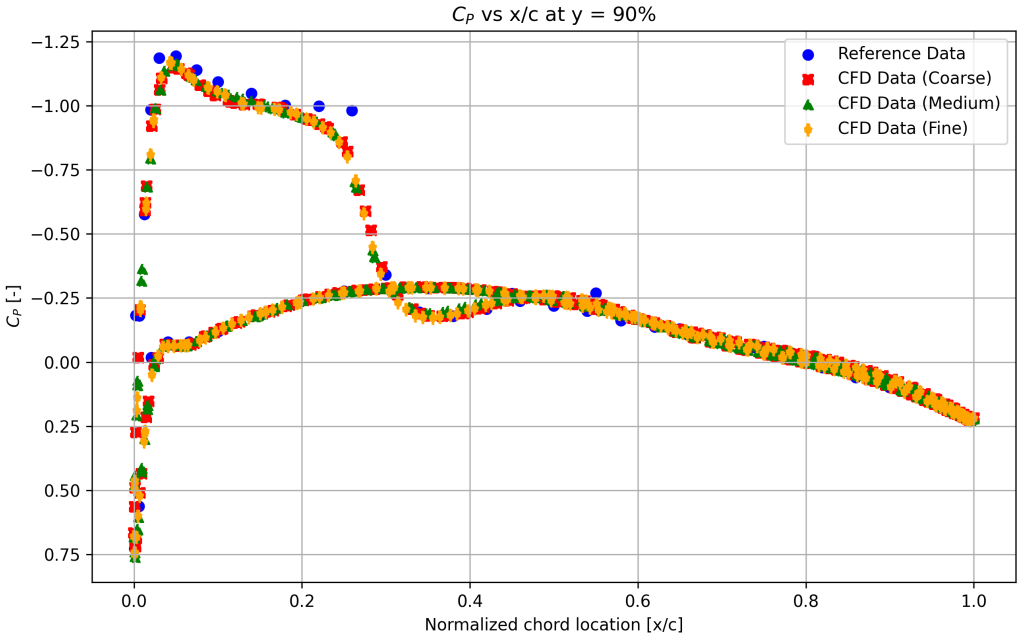

Figure A.5: Pressure coefficient results 90% span.

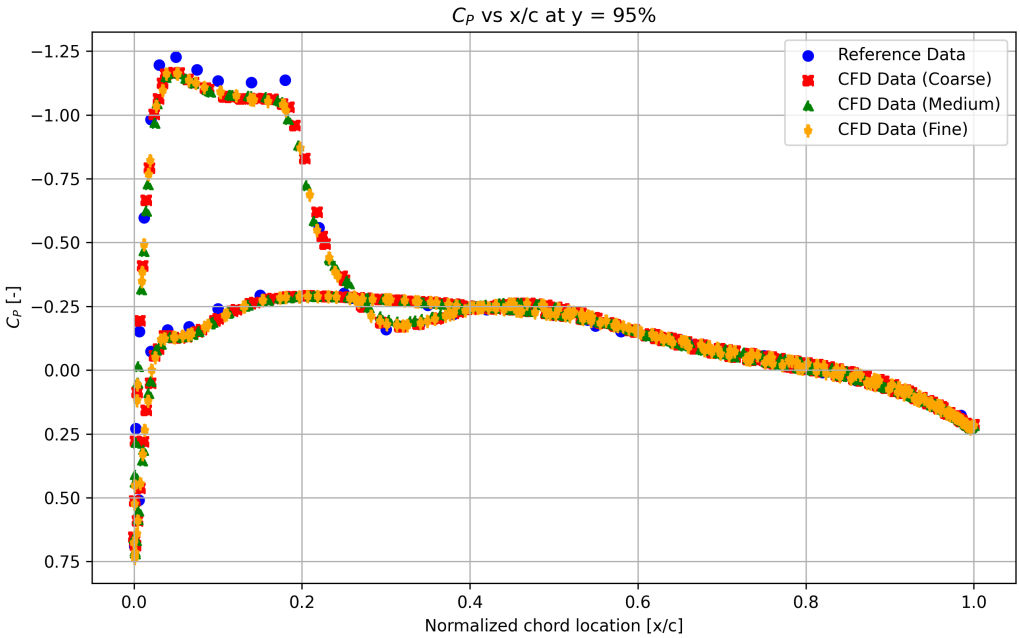

Figure A.6: Pressure coefficient results 95% span.

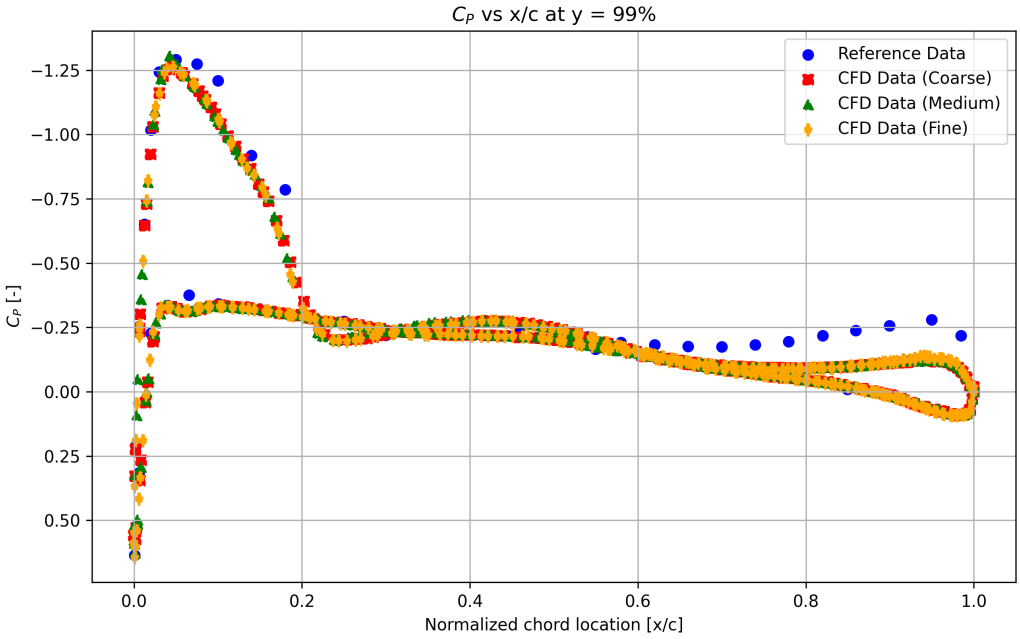

Figure A.7: Pressure coefficient results 99% span.

B

Spanwise Distributions of Effective Angle of Attack

This appendix provides a compilation of the spanwise distributions for the effective angle of attack across the various high-fidelity FSI models evaluated in this study. The included figures represent the full suite of simulation cases conducted to generate the primary data set discussed in chapter 6. By visualizing these distributions, one can further examine the aerodynamic loading and structural responses that characterize each test case.

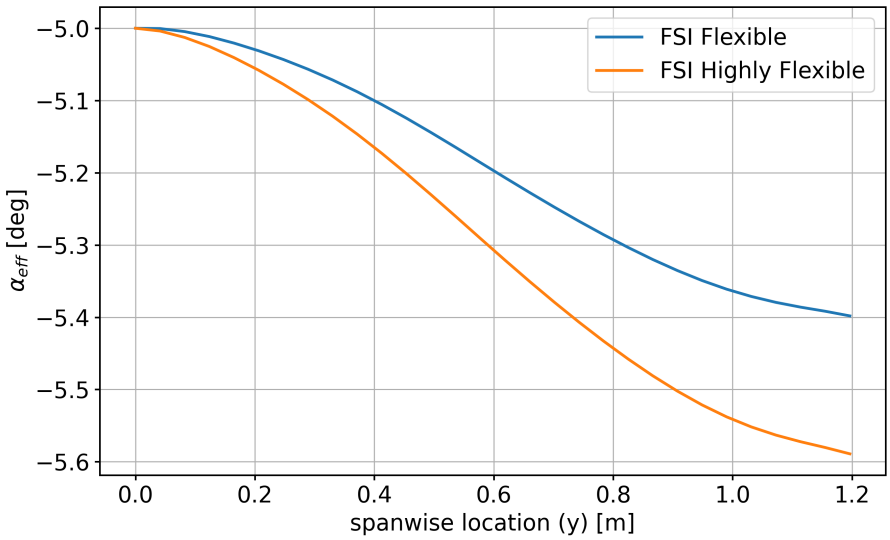

Figure B.1: Effective angle of attack spanwise distribution $M = 0.8395$, $\alpha = -5.0°$.

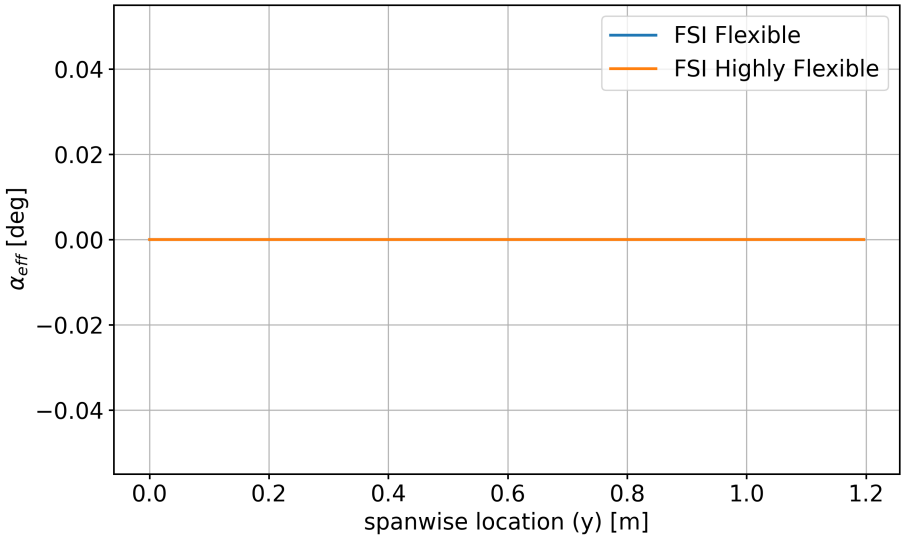

Figure B.2: Effective angle of attack spanwise distribution $M = 0.8395$, $\alpha = 0.0°$.

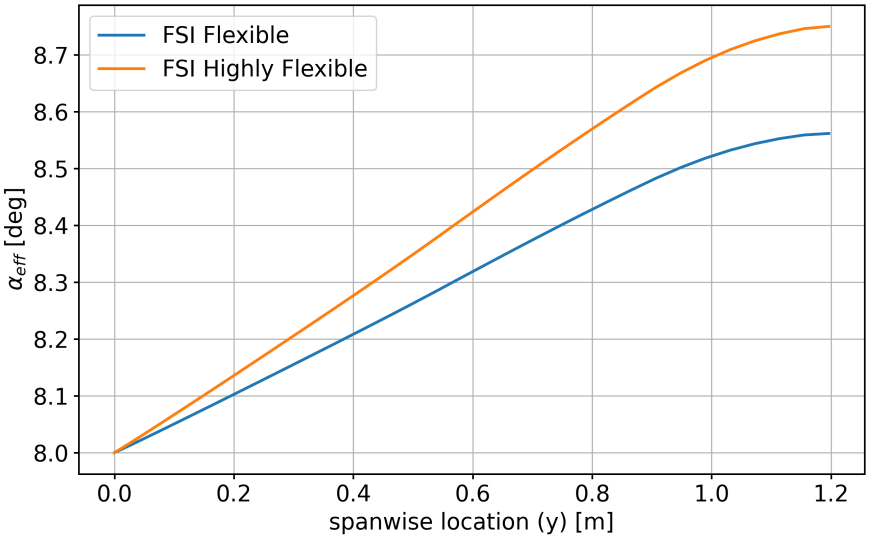

Figure B.3: Effective angle of attack spanwise distribution $M = 0.8395$, $\alpha = 8.0°$.

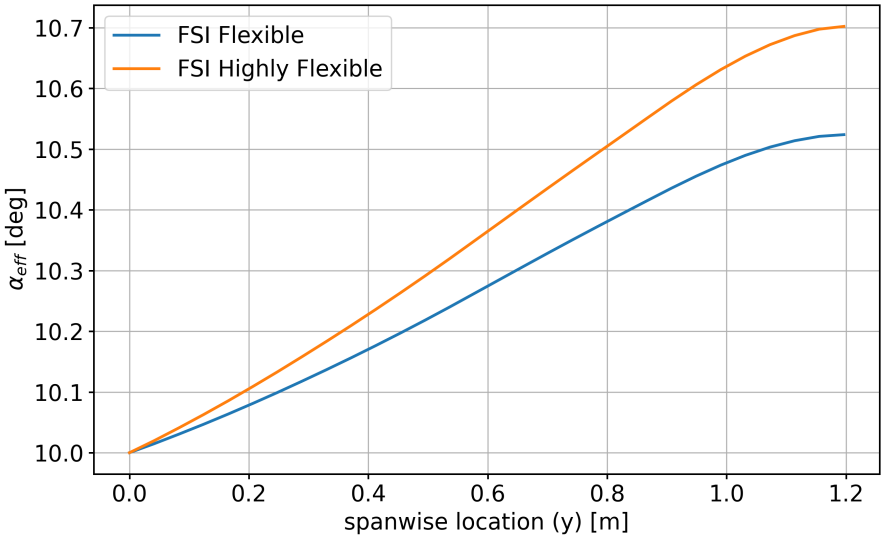

Figure B.4: Effective angle of attack spanwise distribution $M = 0.8395$, $\alpha = 10.0°$.

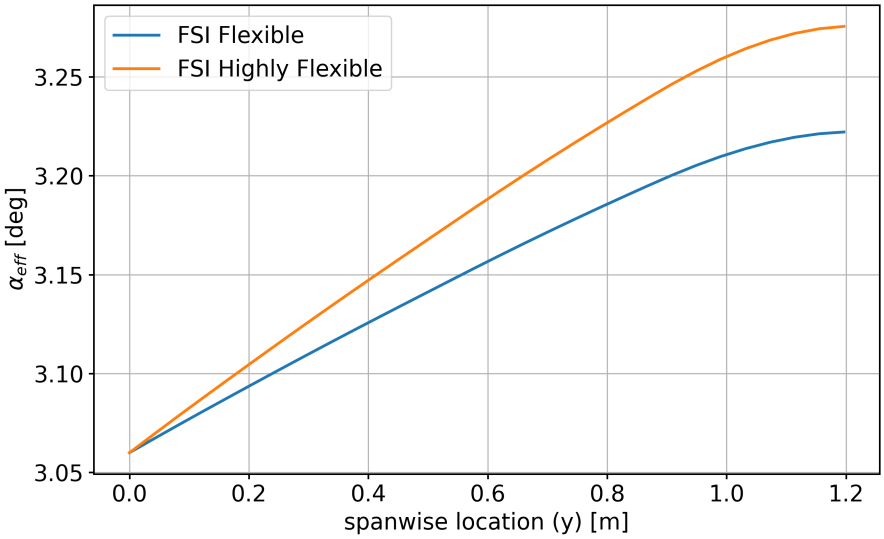

Figure B.5: Effective angle of attack spanwise distribution $M = 0.5$, $\alpha = 3.06°$.

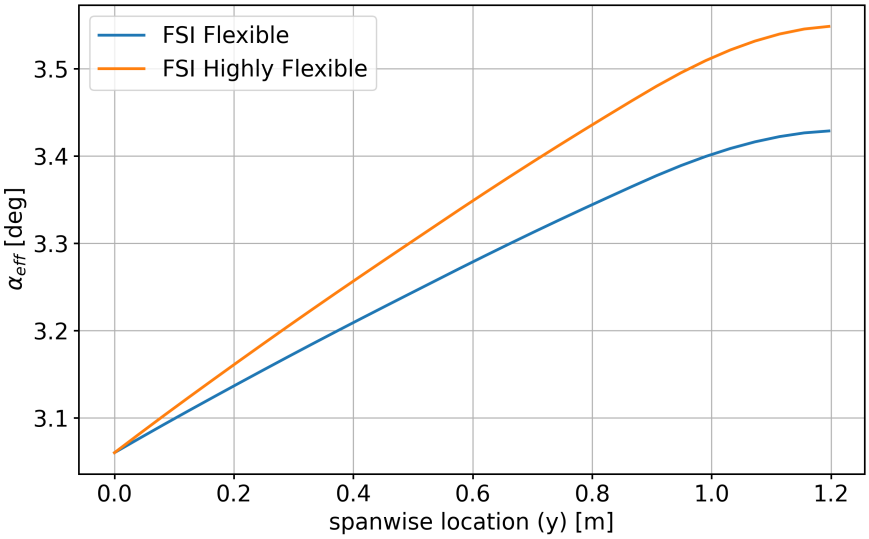

Figure B.6: Effective angle of attack spanwise distribution $M = 0.7$, $\alpha = 3.06°$.

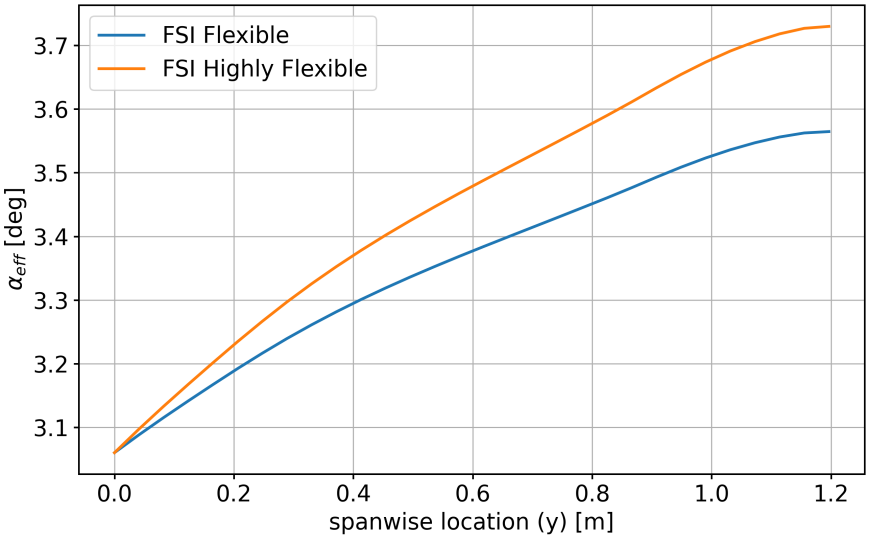

Figure B.7: Effective angle of attack spanwise distribution $M = 0.95$, $\alpha = 3.06°$.

C

Final Aerodynamic- and Structural Models

To supplement the primary findings discussed in chapter 6, this appendix provides the resulting aerodynamic and structural models for the remaining FSI simulation cases. These models were generated using the different simulation cases discussed in chapter 6 and offer a broader view of the model's behavior across various flow conditions and structural flexibilities.

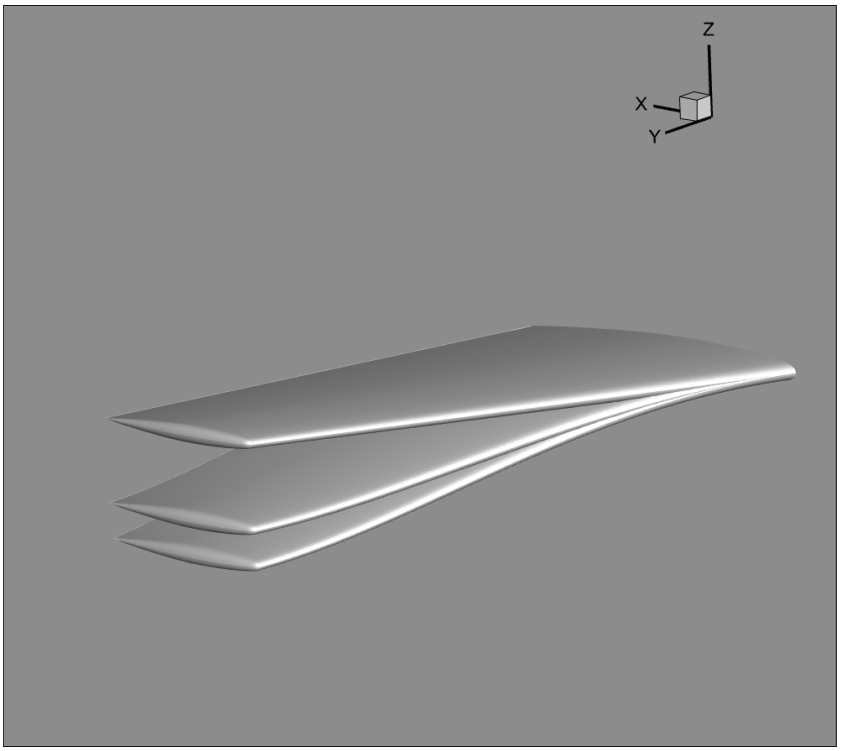

Figure C.1: Rigid CFD, FSI Flexible and FSI Highly (top to bottom) Flexible aerodynamic models for $M = 0.8395$, $\alpha = -5^\circ$ case.

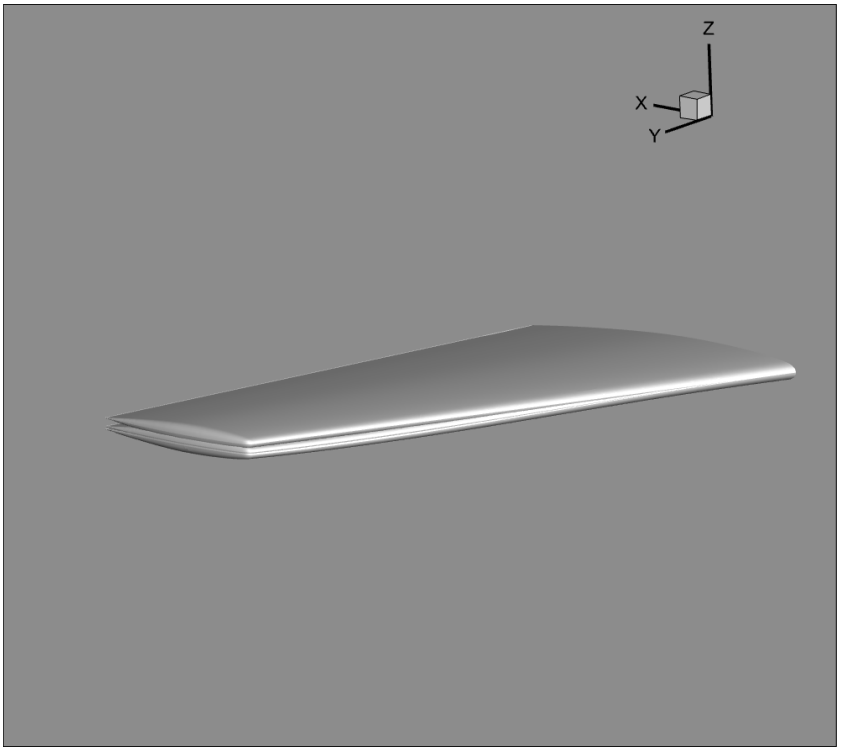

Figure C.2: Rigid CFD, FSI Flexible and FSI Highly (top to bottom) Flexible models for $M = 0.8395$, $\alpha = 0^\circ$ case.

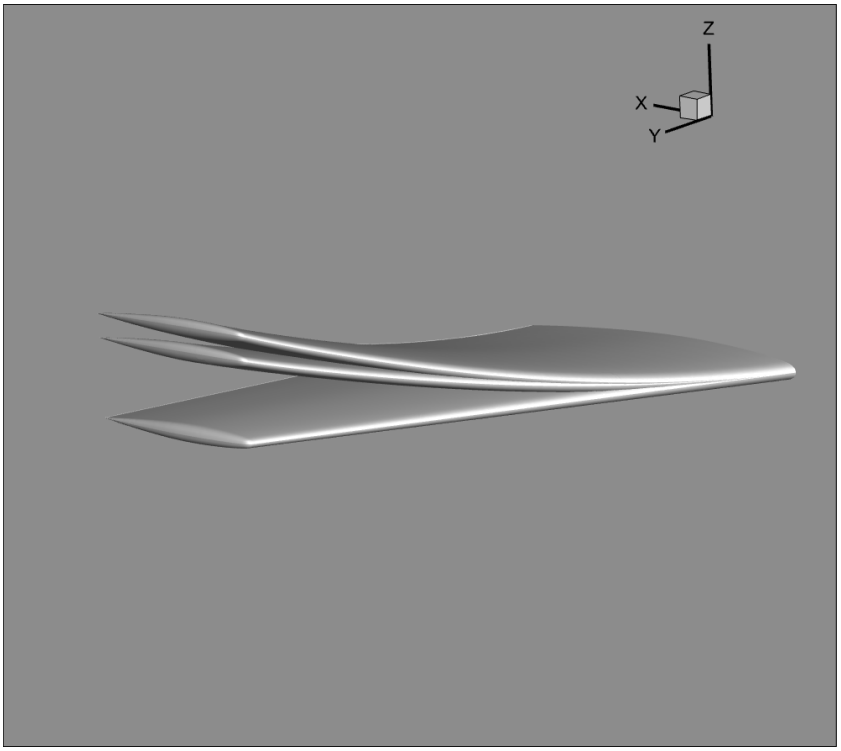

Figure C.3: Rigid CFD, FSI Flexible and FSI Highly Flexible (bottom to top) aerodynamic models for $M = 0.8395$, $\alpha = 8^\circ$ case.

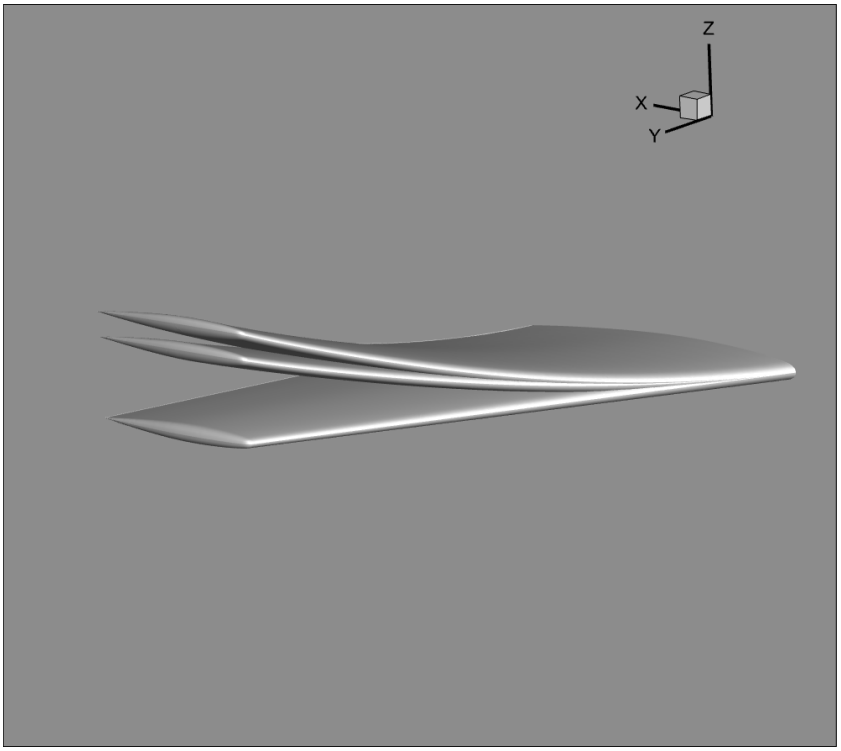

Figure C.4: Rigid CFD, FSI Flexible and FSI Highly Flexible (bottom to top) aerodynamic models for $M = 0.8395$, $\alpha = 10°$ case.

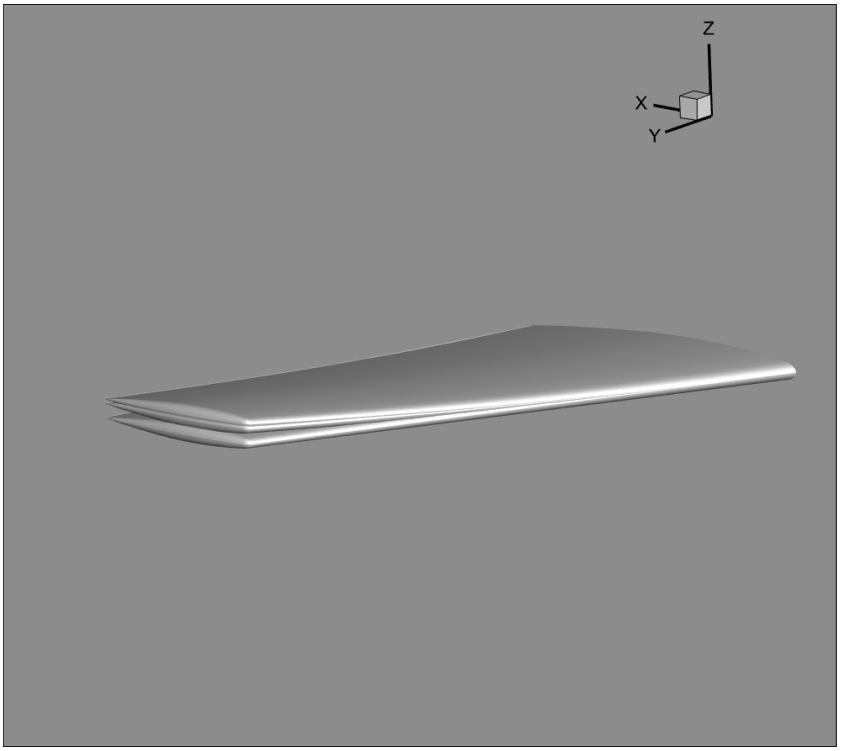

Figure C.5: Rigid CFD, FSI Flexible and FSI Highly Flexible (bottom to top) aerodynamic models for $M = 0.5$, $\alpha = 3.06°$ case.

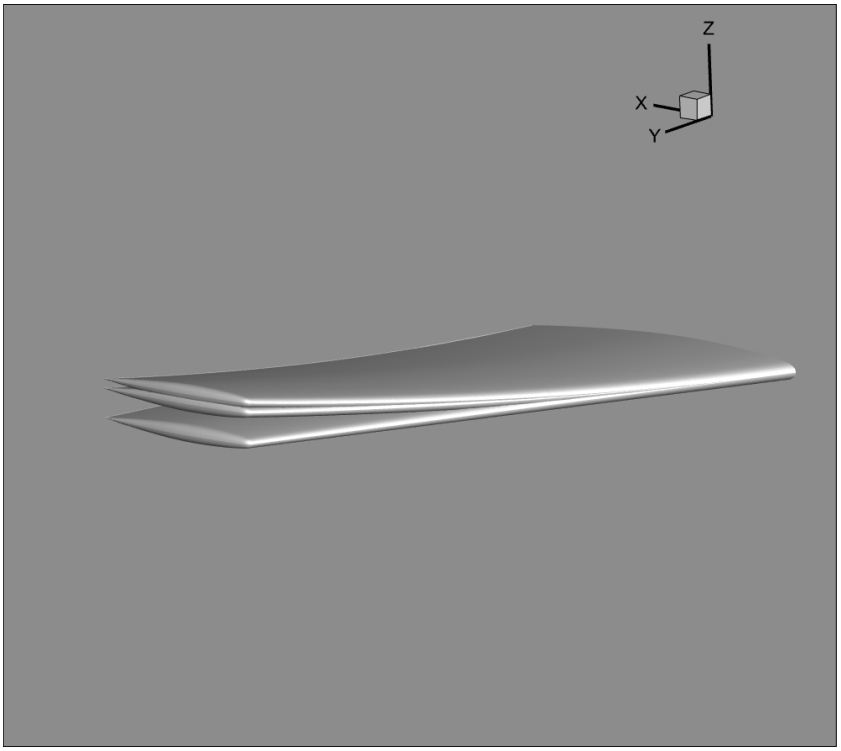

Figure C.6: Rigid CFD, FSI Flexible and FSI Highly Flexible (bottom to top) aerodynamic models for $M = 0.7$, $\alpha = 3.06°$ case.

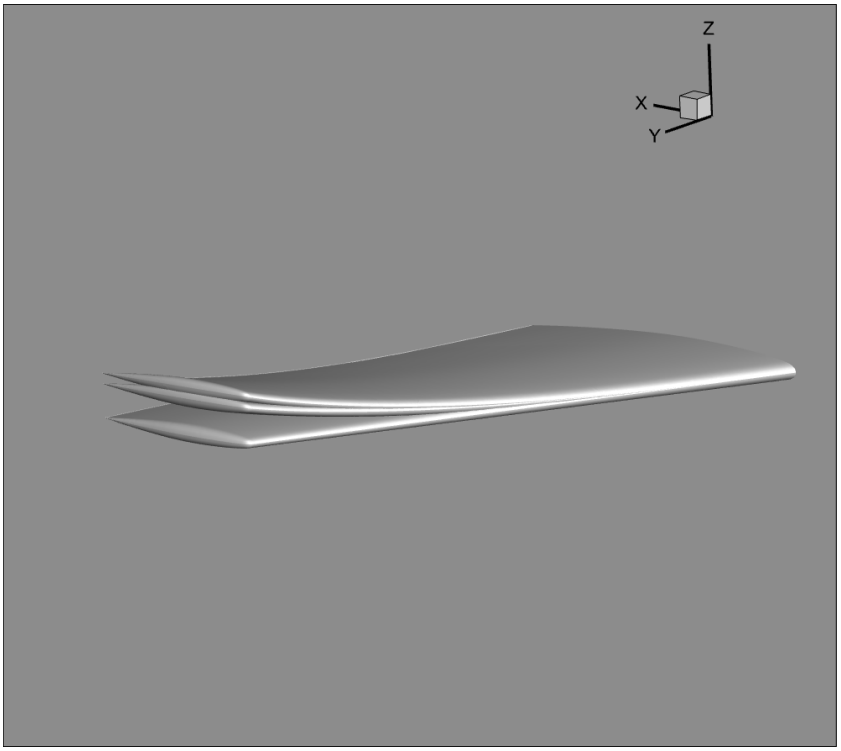

Figure C.7: Rigid CFD, FSI Flexible and FSI Highly Flexible (bottom to top) aerodynamic models for $M = 0.95$, $\alpha = 3.06°$ case.

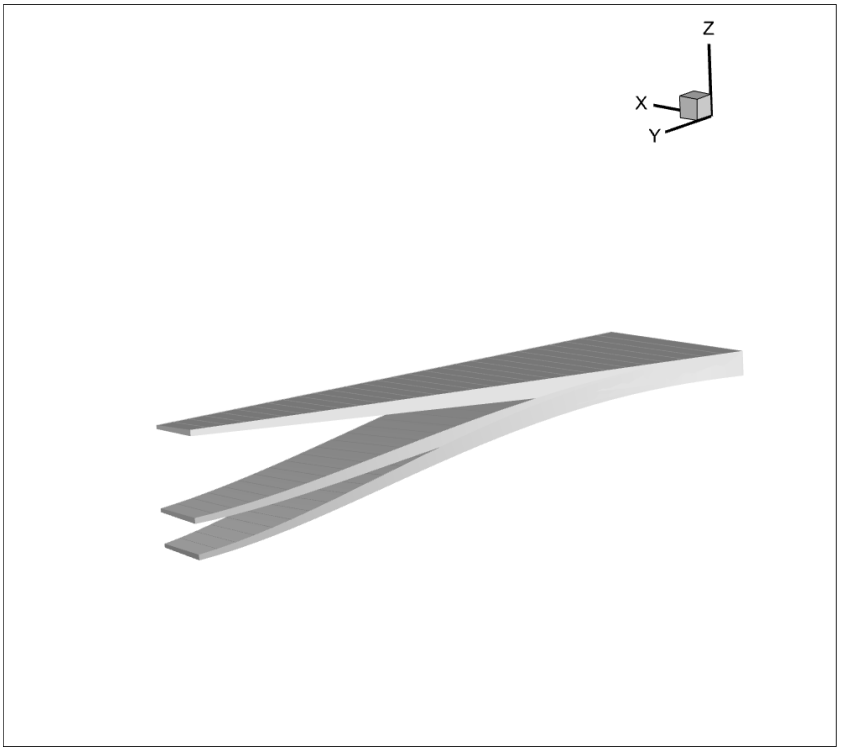

Figure C.8: Rigid CFD, FSI Flexible and FSI Highly Flexible (top to bottom) structural models for $M = 0.8395$, $\alpha = -5°$ case.

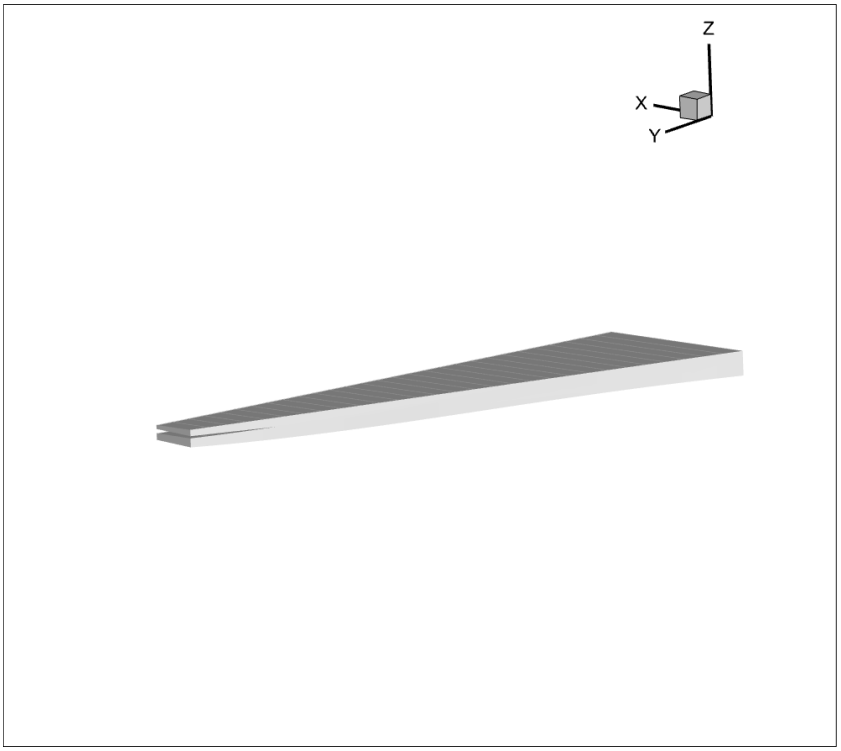

Figure C.9: Rigid CFD, FSI Flexible and FSI Highly Flexible (top to bottom) structural models for $M = 0.8395$, $\alpha = 0°$ case.

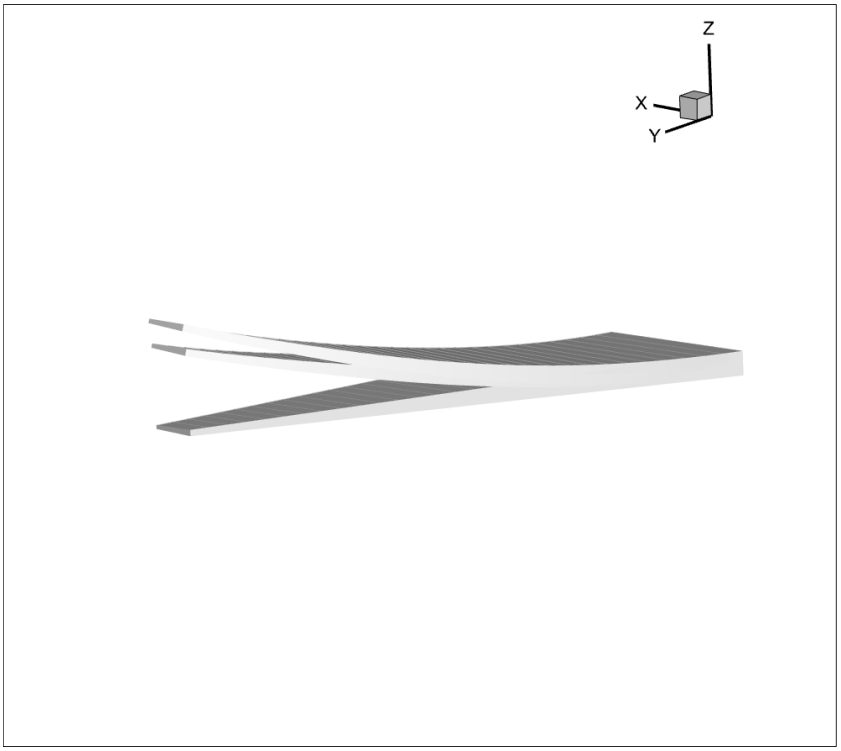

Figure C.10: Rigid CFD, FSI Flexible and FSI Highly Flexible (bottom to top) structural models for $M = 0.8395$, $\alpha = 8°$ case.

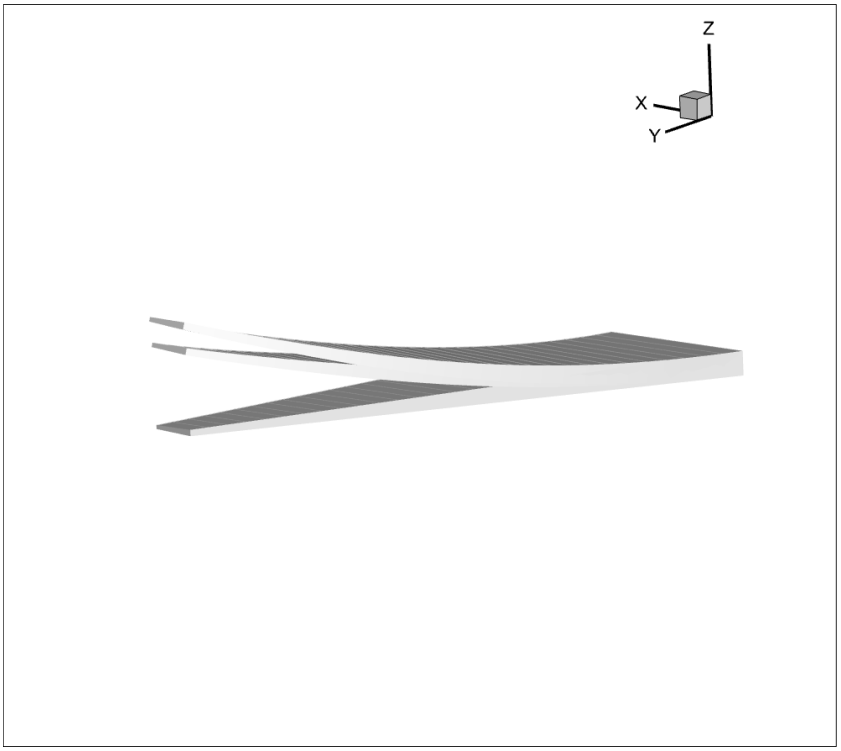

Figure C.11: Rigid CFD, FSI Flexible and FSI Highly Flexible (bottom to top) structural models for $M = 0.8395$, $\alpha = 10°$ case.

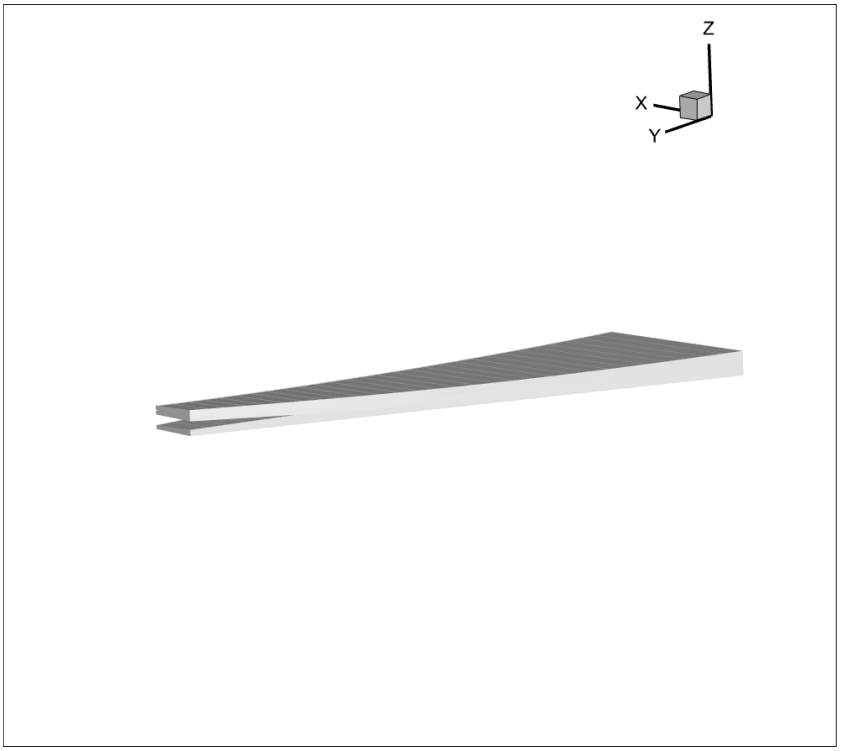

Figure C.12: Rigid CFD, FSI Flexible and FSI Highly Flexible (bottom to top) structural models for $M = 0.5$, $\alpha = 3.06°$ case.

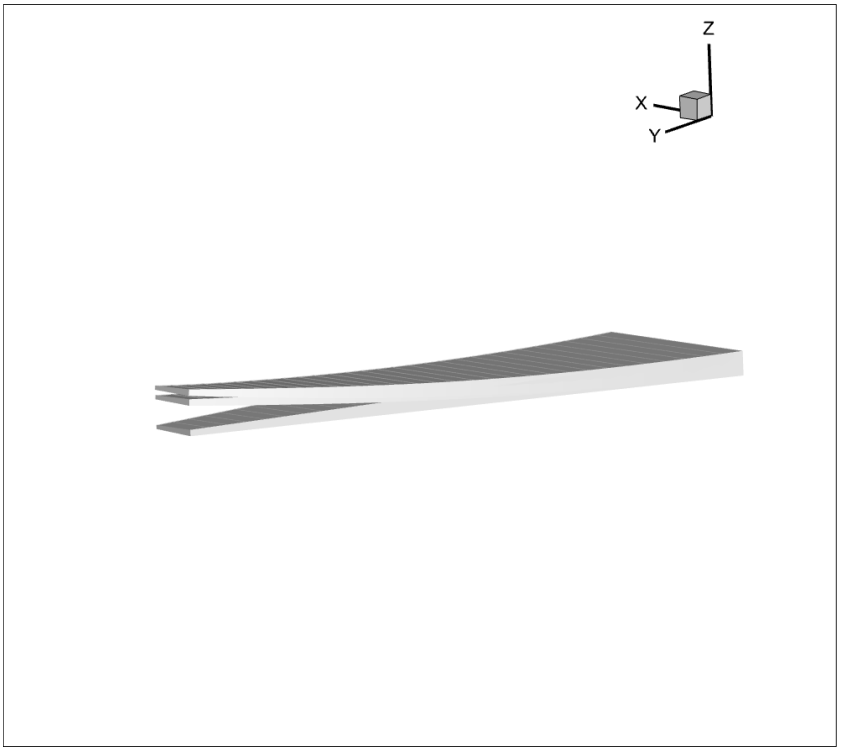

Figure C.13: Rigid CFD, FSI Flexible and FSI Highly Flexible (bottom to top) structural models for $M = 0.7$, $\alpha = 3.06°$ case.

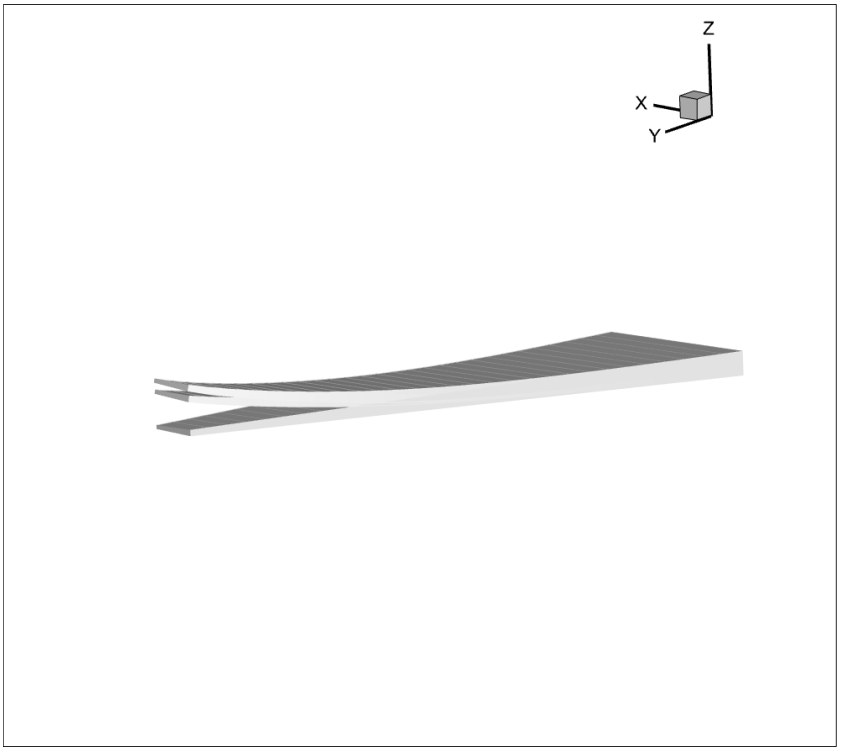

Figure C.14: Rigid CFD, FSI Flexible and FSI Highly Flexible (bottom to top) structural models for $M = 0.95$, $\alpha = 3.06°$ case.