POLITECNICO DI TORINO

Department of
Aerospace and Mechanical Engineering

Master's Thesis

Machine Learning Techniques for Aeroelastic Analyses of Simplified Aircraft Configurations

Supervisors:
Prof. Marco Petrolo
Prof. Matteo Filippi

Student:
Mahmut Burak Cinar

In the first age,
In the first battle,
When the shadows first lengthened,
One stood...
He chose the path of perpetual torment.

Abstract

In this thesis, our main goal is to create an effective machine-learning model to predict flutter speed. At first, we made preliminary investigations about aeroelasticity, starting from a single wing modeled as a beam to a whole aircraft configuration. We analyzed and tried to find the optimum parameters to use in the flutter analysis. After that, we selected a base aircraft model to create different aircraft configurations to use as input to create training, validation, and test datasets. We designated intervals for the aircraft such as dimensions, engine locations, and materials. Then we created an automation tool in Python to create aeroelastic outputs (frequency and damping values for different modes of the aircraft, in a speed interval designated beforehand) using MUL2 and NASTRAN. With this tool, we created data for 7000 input instances and stored it for the machine learning model creation. Then we preprocessed the raw data for the deep neural network that we will be utilizing, we created a deep neural network architecture and tuned the hyperparameters accordingly. And then we created a final model. We tried different loss functions, but the mean absolute error with Adam optimizer was the most accurate one. We managed to predict flutter speeds in the test data accurately. We showed that especially for design optimization, machine learning is a valuable tool since it can capture the relation for flutter, and it has the potential to do much more such as to create large deep learning models to accelerate every part of the design and analyzing processes.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Aeroelasticity

Aeroelasticity is a field of study that examines the interplay between the deformation of an elastic structure in airflow and the aerodynamic forces that result from this interaction. This interdisciplinary field's complexity is best illustrated by Collar's triangle (refer to Figure 1.1), a concept introduced by Professor A. R. Collar in the 1940s. This triangle represents the interconnections among the three primary disciplines of aerodynamics, dynamics, and elasticity. Aerodynamics is concerned with predicting the forces exerted on a body of a specific shape. Elasticity deals with forecasting the shape of an elastic body under stress. Dynamics focuses on the impact of inertial forces. The integration of these disciplines leads to various interaction areas, including structural dynamics (a combination of elasticity and dynamics), flight mechanics (a combination of dynamics and aerodynamics), static aeroelasticity (a combination of aerodynamics and elasticity), and dynamic aeroelasticity (a combination of elasticity, dynamics, and aerodynamics). [14].

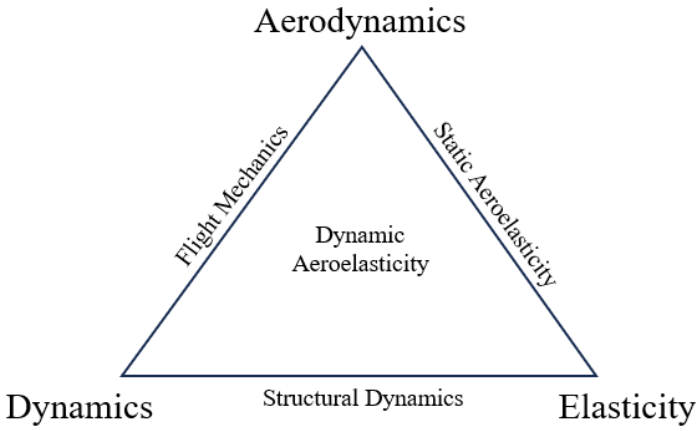


Figure 1.1: Collar's Triangle

It is usual to classify aeroelastic phenomena as being either static or dynamic. Static aeroelasticity considers the non-oscillatory effects of aerodynamic forces acting on the flexible aircraft structure. There is also the potentially disastrous phenomenon of divergence to consider, where the wing twist can increase without limit when the aerodynamic pitching moment on the wing due to twist exceeds the structural restoring moment. Dynamic aeroelasticity is concerned

with the oscillatory effects of the aeroelastic interactions, and the main area of interest is the potentially catastrophic phenomenon of flutter. This instability involves two or more modes of vibration and arises from the coupling of aerodynamic, inertial, and elastic forces; it means that the structure can effectively extract energy from the air stream. The presence of flexible modes influences the dynamic stability modes of the rigid aircraft and so affects the flight dynamics [29].

1.1.1 Historical Background

Aeroelastic phenomena have significantly influenced the history of powered flight. In 1903, the Wright brothers achieved lateral control on their Wright Flyer by employing controlled warping of the wings. In the same year, Samuel Langley's attempts at powered flight resulted in catastrophic wing failure due to excessive flexibility and overloading. These aeroelastic phenomena, including torsional divergence, contributed to the dominance of biplane designs until the 1930s. At that time, "stressed-skin" metallic structures were introduced to provide torsional rigidity for monoplanes. The first documented instance of aircraft flutter occurred in 1916 when the Handley Page O/400 bomber experienced severe tail oscillations due to the absence of a torsion-rod connection between the port and starboard elevators. Another significant aeroelastic concern emerged in 1927 with the Bristol Bagshot, a twin-engine, high-aspect-ratio aircraft. As the aircraft's speed increased, the effectiveness of the ailerons decreased to zero and then became negative, a phenomenon is now known as "aileron reversal". Catastrophic failures due to aircraft flutter became a major design concern during World War I and continue to be so today. R.A. Frazer and W.J. Duncan of the National Physical Laboratory in England compiled a seminal document on this topic, "The Flutter of Aeroplane Wings," as R&M 1155 in August 1928. Following the successful analysis of the 1927 Bristol Bagshot aileron reversal and the development of design criteria to prevent it by Roxbee Cox and Pugsley at the Royal Aircraft Establishment in the early 1930s, the term "aeroelasticity" was proposed to describe these phenomena. [14].

1.2 Machine Learning

Machine Learning (ML), a branch of artificial intelligence, utilizes algorithms and statistical models to empower computers to carry out tasks without the need for explicit programming. These ML algorithms construct a mathematical model from sample data, often referred to as "training data", enabling them to make predictions or decisions without being specifically programmed for the task [19, 6].
According to UC Berkeley, the machine learning algorithm can be divided into three main parts:

1. Decision Process: Machine learning algorithms are used to make categorizations or predictions. Based on labeled or unlabeled input data, the algorithm produces an estimate about a pattern in the data.

2. Evaluation Metric: An error function evaluates the model's predictions. If there are now examples available, an error function can make comparisons to assess the accuracy of the model.

3. Optimization: If the model can better align with the data points in the training set, then the weights tweaked to reduce the minimize the difference between the known examples

and the model's approximation. The algorithm will repeat this iterative "evaluate and optimize" process until it reaches a certain accuracy threshold.

1.2.1 Machine Learning, Deep Learning and Neural Networks

While the terms of machine learning and deep learning are often used interchangeably, there are nuances between them. Both machine learning and deep learning, along with neural networks, are subsets of artificial intelligence. However, neural networks fall under machine learning, and deep learning is a sub-field of neural networks.

The distinction between deep learning and machine learning lies in their learning methods. "Deep" machine learning can utilize labeled datasets, also known as supervised learning, to guide its algorithm, but it does not strictly need a labeled dataset. Deep learning can process unstructured data in its raw form (like text or images), and it can autonomously identify the features that differentiate various data categories. This reduces the need for human intervention and facilitates the use of large data volumes. Deep learning can be thought of as "scalable machine learning", as Lex Fridman mentions in his MIT lecture.

In contrast, classical or "non-deep" machine learning relies more on human intervention for learning. Human experts identify the features to distinguish between data inputs, typically requiring more structured data for learning.

Neural networks, or artificial neural networks (ANNs), consist of node layers, including an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron connects to another and has an associated weight and threshold. If the output of any individual node exceeds the specified threshold value, that node is activated, transmitting data to the network's next layer. If not, no data is passed to the network's next layer by that node. The "deep" in deep learning refers to a number of layers in a neural network. A neural network with more than three layers, including the input and output, can be considered as a deep learning algorithm or deep neural network. A neural network with only three layers is simply a basic neural network. Deep learning and neural networks have been instrumental in advancing fields such as computer vision, natural language processing, and speech recognition.

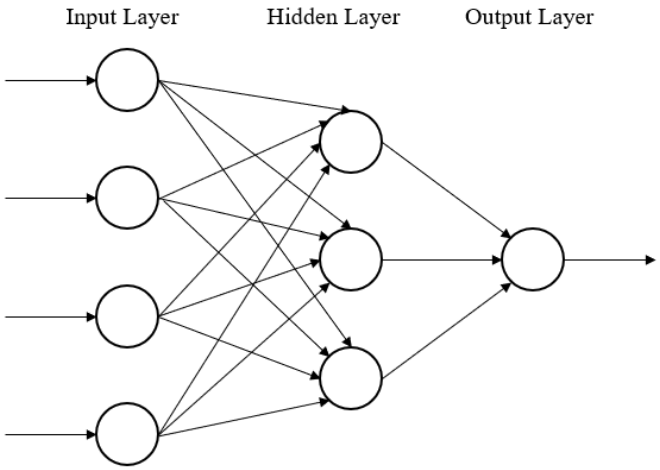


Figure 1.2: A Simple Neural Network

1.2.2 Machine Learning Methods

- Supervised Machine Learning: This type of machine learning uses labeled datasets to train algorithms to accurately classify data or predict outcomes. As the model receives input data, it adjusts its weights until it fits the data appropriately. This adjustment happens during the cross-validation process to prevent the model from overfitting or underfitting. Supervised learning is used to solve various real-world problems at scale, such as segregating spam e-mails into a separate folder. Techniques used in supervised learning include neural networks, naïve bayes, linear regression, logistic regression, random forest, and support vector machine (SVM).

- Unsupervised machine learning employs algorithms to analyze and cluster unlabeled datasets (known as clusters). These algorithms can identify hidden patterns or data groupings without human intervention. Its ability to find similarities and differences in data makes it suitable for exploratory data analysis, cross-selling strategies, customer segmentation, and image and pattern recognition. It is also used to decrease the number of features in a model through a process called dimensionality reduction. Common methods for this include Principal Component Analysis (PCA) and Singular Value Decomposition (SVD). Other algorithms used in unsupervised learning include neural networks, k-means clustering, and probabilistic clustering methods.

- Semi-supervised Machine Learning: Semi-supervised learning strikes a balance between supervised and unsupervised learning. During training, it uses a smaller labeled dataset to guide the classification and feature extraction from a larger, unlabeled dataset. Semi-supervised learning can address the issue of insufficient labeled data for a supervised learning algorithm. It is also beneficial when labeling data is too expensive.

- Reinforcement Learning: Reinforcement machine learning is a type of machine learning that, while bearing similarities to supervised learning, does not rely on sample data for training. Instead, this model learns on the fly through a process of trial and error. A series of successful results are reinforced to formulate the optimal recommendation or policy for a specific problem.

1.2.3 Common Machine Learning Algorithms

- Neural Networks: These algorithms mimic the functioning of the human brain by using a vast network of interconnected processing nodes. They excel at pattern recognition and are crucial in applications such as natural language processing, image and speech recognition, and image generation.

- Linear Regression: This algorithm predicts numerical values based on a linear relationship between various values. For instance, it could be used to forecast house prices using historical data from a specific area.

- Logistic Regression: This supervised learning algorithm predicts categorical response variables, like ‘yes/no’ responses to questions. It finds use in applications like spam classification and quality control in a production line.

- Clustering: Clustering algorithms, which employ unsupervised learning, can detect patterns in data for grouping purposes. These algorithms assist data scientists by identifying differences in data items that might have been missed by humans.

- Decision Trees: Decision trees can predict numerical values (regression) and classify data into categories. They use a series of interconnected decisions that can be represented with a tree diagram. One advantage of decision trees is their ease of validation and auditing, unlike the opaque nature of neural networks.

- Random Forests: In a random forest, the machine learning algorithm predicts a value or category by amalgamating the results from multiple decision trees.

1.2.4 Machine Learning in Aeroelastic Research

Traditional approaches to aeroelastic analysis have leaned heavily on physical modeling and computational fluid dynamics (CFD), which are effective but can be computationally demanding and slow [3]. The rise of machine learning technologies expedites a new era for aeroelastic research, offering innovative ways to model complex interactions, predict aeroelastic phenomena, and optimize structures with unprecedented efficiency [13].

Particularly, deep learning, a subset of machine learning, has shown promise in addressing aeroelastic problems. This is exemplified in the work by Yi-Ren Wang et al., where deep learning techniques were applied to predict flutter speed with a level of accuracy and computational efficiency that surpasses traditional methods [28]. This study not only underscores the potential of ML in enhancing predictive capabilities in aeroelasticity but also paves the way for its broader application in aerospace engineering challenges.

Additionally, neural networks have been utilized for flutter prediction, demonstrating significant reductions in computational costs and improvements in prediction accuracy over conventional methods [17]. Reinforcement learning, a branch of ML that focuses on teaching computers to learn from interaction with the environment, has been explored for developing adaptive control strategies in aeroelastic systems, showing potential in managing wing flutter across different flight conditions [26].

The synergy between machine learning and computational fluid dynamics (CFD) has also been fruitful, leading to the development of surrogate models based on CFD-generated data. These models can predict aerodynamic loads and structural responses with reduced computational requirements, facilitating faster aerospace structure design and testing [32]. Such advancements illustrate ML's role in expanding the scope for design exploration and optimization in aeroelastic research.

In optimizing aerospace structures, machine learning, particularly genetic algorithms combined with ML models, has been instrumental in refining wing geometries for improved aeroelastic performance. This optimization balances structural weight, strength, and aerodynamic efficiency, showcasing ML's capability in enhancing aerospace design [5].

Despite these advancements, integrating machine learning into aeroelastic research presents challenges, including data quality, model interpretability, and the need for extensive training datasets. The opaque nature of some ML models also raises questions about their reliability and safety in critical aerospace applications [15].

Machine learning offers transformative potential for aeroelastic research, enabling more sophisticated modeling, accurate predictions, and efficient optimization. As machine learning technologies continue to evolve, they are set to play a pivotal role in the future of aeroelasticity, with ongoing research needed to address the current challenges fully.

Chapter 2

Aeroelastic Phenomena

2.1 Divergence

Static aeroelasticity studies the interplay between the constant aerodynamic forces exerted on an aircraft and the subsequent elastic bending of its wings. This field primarily deals with two types of design challenges. The first, which is applicable to all aircraft, pertains to the effects of elastic deformation on aerodynamic forces and vice versa during regular flight conditions. These effects can have a significant impact on various factors such as performance, handling properties, flight stability, distribution of structural loads, and control effectiveness. The second challenge is associated with the possibility of static instability in the wing structure, which could result in a severe failure, often termed as "divergence". Divergence occurs when a wing bends under aerodynamic forces in a manner that amplifies the applied load, leading to further distortion of the structure, and potentially causing a failure. This failure is not simply due to an overload for the designed structure; instead, it's the interaction of the aerodynamic forces with the structure that results in a reduction of effective stiffness [14].

2.1.1 Wind Tunnel Wall-Mounted Model

Imagine a rigid, uniformly spanned wing model that is attached to the sidewalls of a wind tunnel. This setup allows the wing to pitch around the support axis, as depicted in Figure 2.1. The support is torsionally flexible, limiting the wing's pitch rotation in the same way a rotational spring would.

The rotational stiffness of the support is denoted by k, as shown in Figure 2.2. If we consider the body to be pivoted about its support O, which is located at a distance x_O from the leading edge, the principle of moment equilibrium necessitates that the sum of all moments about O must be zero. Thus the moment equilibrium equation around the elastic axis O is:

$$M_{\text{ac}} + L\left(x_O - x_{\text{ac}}\right) - W\left(x_O - x_{\text{cg}}\right) - k\theta = 0 \tag{2.1}$$

If support were rigid, the angle of attack of the wing would be α_r, positive nose-up. However, it is not rigid and the elastic part of the pitch angle is denoted by θ, which is also a positive nose-up. The angle of attack of the wing can be written as $\alpha = \alpha_r + \theta$. Considering, that linear aerodynamics is in use, the angle of attack can be assumed to be a small angle, such that $sin(\alpha) \approx \alpha$ and $cos(\alpha) \approx 1$. In this case, the airfoil is assumed to be thin (small thickness to chord and small camber) and the flow is incompressible.

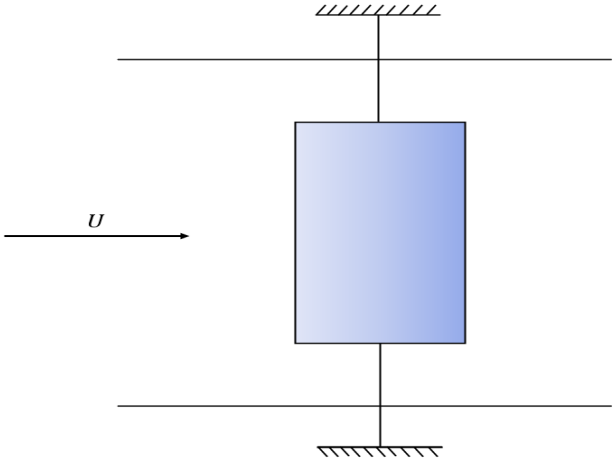


Figure 2.1: Platform view of a wind-tunnel model on torsionally elastic support

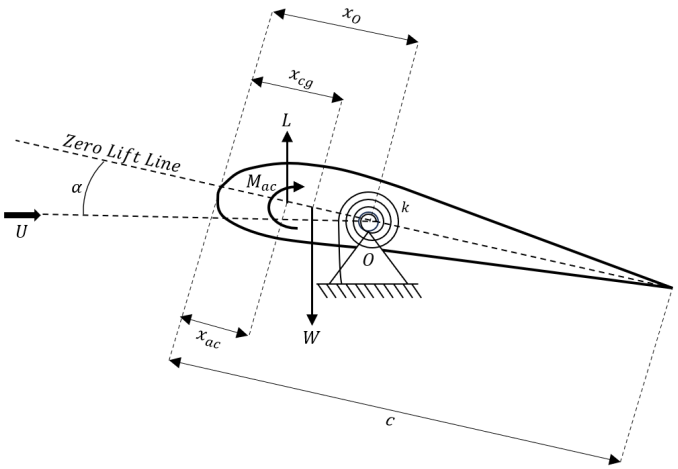


Figure 2.2: Airfoil for wind-tunnel model

For linear aerodynamics, the lift for an elastic support can be written as:

$$L = qSC_{L\alpha}\left(\alpha_R + \theta\right) \tag{2.2}$$

where $q = \frac{1}{2}\rho_\infty U^2$ is the freestream dynamic pressure, U is the freestream dynamic pressure, ρ_∞ is the freestream air density, S is surface area of the wing and $C_{L\alpha}$ is the wing lift-curve slope. It is also possible to express the moment of aerodynamic forces about the aerodynamic center as:

$$M_{ac} = qScC_{Mac} \tag{2.3}$$

If the angle of attack is small, C_{Mac} can be considered constant. Using Equation 2.2 and Equation 2.3, Equation 2.1 can be rewritten as:

$$qScC_{Mac} + qSC_{L\alpha}\left(\alpha_R + \theta\right)\left(x_O - x_{ac}\right) - W\left(x_O - x_{cg}\right) - k\theta = 0 \tag{2.4}$$

where it is possible to obtain elastic deflection:

$$\theta = \frac{qScC_{Mac} + qSC_{L\alpha}\alpha_R\left(x_O - x_{ac}\right) - W\left(x_O - x_{cg}\right)}{k - qSC_{L\alpha}\left(x_O - x_{ac}\right)} \tag{2.5}$$

When there is an upstream lift with respect to point O, α will get increased due to lift, and the latter creates more lift. So, lift is destabilizing, counteracting the action of the spring when $x_O > x_{ac}$. When the moment of the lift about point O exceeds the restoring moment from the spring, results in static aeroelastic instability called 'divergence'. From Equation 2.5, when $x_{ac} < x_O$ it is possible for the denominator to vanish or for θ to blow up with sufficient q values. The divergence dynamic pressure, or in other words, dynamic pressure at which the divergence occurs can be written as:

$$q = q_D = \frac{k}{SC_{L\alpha}\left(x_O - x_{ac}\right)} \tag{2.6}$$

Using this equation, the divergence speed can be found:

$$U_D = \sqrt{\frac{2k}{\rho SC_{L\alpha}\left(x_O - x_{ac}\right)}} \tag{2.7}$$

If the aerodynamic center is coincident with the pivot, so $x_{ac} = x_O$, the divergence dynamic pressure becomes infinite. Or when the aerodynamic center is coming later than the pivot $x_{ac} > x_O$, the divergence dynamic pressure becomes negative. Since the dynamic pressure must be positive and finite for physical reasons, it is clear that in either case divergence is impossible.

To understand the character of this instability, consider a symmetric airfoil ($C_{Mac} = 0$), and $x_{cg} = x_O$ so that the weight term drops out of the equation for θ. Using Equation 2.6,let $k = q_D SC_L\alpha(x_O - x_a c)$, so θ can be written as:

$$\theta = \frac{\alpha_r}{\frac{q_D}{q} - 1} \tag{2.8}$$

Previously, it was discovered that the lift is proportional to $\alpha_r + \theta$. So, the lift change divided by the rigid lift can be written as:

$$\frac{\Delta L}{L_{\text{rigid}}} = \frac{\theta}{\alpha_r} = \frac{\frac{q}{q_D}}{1 - \frac{q}{q_D}} \tag{2.9}$$

Both θ and $\frac{\Delta L}{L_{\text{rigid}}}$ approaches to infinity as q approaches to q_D.

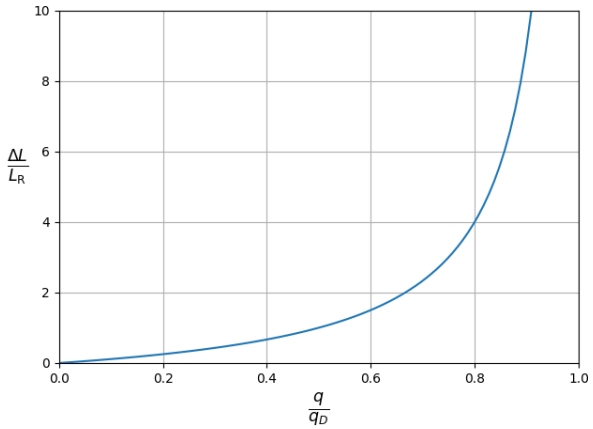


Figure 2.3: Relative change in lift due to aeroelastic effect

2.2 Flutter Analysis of a Typical Section

In this section, a demonstration of the flutter analysis of a linear aeroelastic system will be conducted. To do this, a simple model is needed, therefore simple, spring-restrained, rigid-wing models, just like Figure 2.4 are used. These models are called 'typical-section models'. This configuration can represent a typical airfoil section along a finite wing. The discrete springs can reflect the wing structural bending and torsional stiffness, and the reference point, therefore the elastic axis.

The points take place on the zero-lift line, P, C, Q, T which refer respectively to the reference point (where the displacement h is measured), the center of mass, the aerodynamic center, and the three-quarter-chord. The dimensionless parameters e and a $(-1 \leq e \leq 1)$ and $(-1 \leq a \leq 1)$ determine the locations of the points C and P. The chordwise offset of the center of mass is usually made dimensionless by airfoil semi-chord b and denoted as $x_\theta = e - a$. The rigid plunging and pitching of the model are restrained by light, linear springs with spring constraints k_h and k_θ. The equations of motion are formulated using Lagrange's equations. To achieve this, kinetic and potential energies and generalized forces resulting from aerodynamic loading are needed. The potential energy can be written as:

$$P = \frac{1}{2}k_h h^2 + \frac{1}{2}k_\theta \theta^2 \tag{2.10}$$

The kinetic energy can be expressed as:

$$K = \frac{1}{2}m\mathbf{V}_C \cdot \mathbf{V}_C + \frac{1}{2}I_C\dot{\theta}^2 \tag{2.11}$$

Where $\boldsymbol{V}_C$ can be found with the following equations:

$$\boldsymbol{V}_C = -\dot{h}\hat{\boldsymbol{i}}_2 + b\dot{\theta}(a-e)\hat{\boldsymbol{b}}_2 \tag{2.12}$$

Therefore the kinetic energy equation can be written as:

$$\begin{aligned} K &= \frac{1}{2}m\left(\dot{h}^2 + b^2x_\theta^2\dot{\theta}^2 + 2bx_\theta\dot{h}\dot{\theta}\right) + \frac{1}{2}I_c\dot{\theta}^2 \\ K &= \frac{1}{2}m\left(\dot{h}^2 + 2bx_\theta\dot{h}\dot{\theta}\right) + \frac{1}{2}I_p\dot{\theta}^2 \end{aligned} \tag{2.13}$$

I_C is the moment of inertia about C. And I_P is the moment of inertia around the P. I_P can be calculated using the parallel axis theorem: $I_P = I_C + mb^2x_\theta^2$

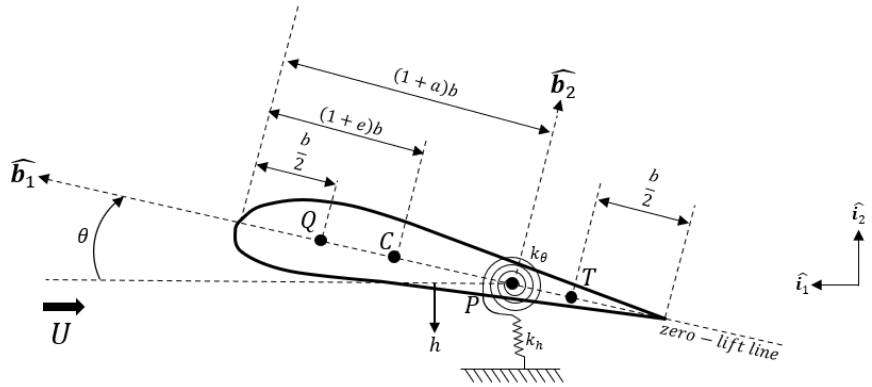


Figure 2.4: Schematic showing geometry of the wing section with pitch and plunge spring restraints

The virtual displacement of the point P can be obtained with the following equation:

$$\delta\boldsymbol{p}_Q = -\delta h\hat{\boldsymbol{i}}_2 + b\delta\theta\left(\frac{1}{2}+a\right)\hat{\boldsymbol{b}}_2 \tag{2.14}$$

The virtual work of the aerodynamic forces can be expressed as:

$$\overline{\delta W} = L\left[-\delta h + b\left(\frac{1}{2}+a\right)\delta\theta\right] + M_{\frac{1}{4}}\delta\theta \tag{2.15}$$

and the generalized forces become:

$$\begin{aligned} Q_h &= -L \\ Q_\theta &= M_{\frac{1}{4}} + b\left(\tfrac{1}{2}+a\right)L \end{aligned} \tag{2.16}$$

Lagrange's equations are specialized here, and the kinetic energy K depends on only $\dot{q}_1, \dot{q}_2, ...$; so,

$$\frac{d}{dt}\left(\frac{\partial K}{\partial \dot{q}_i}\right) + \frac{\partial P}{\partial q_i} = Q_i \quad (i = 1, 2, ..., n) \tag{2.17}$$

Here $n = 2$ and $q_2 = \theta$ and the equations of motion become:

$$\begin{cases} m\ddot{h} + mx_\theta b\ddot{\theta} + k_h h = -L \\ mx_\theta b\ddot{h} + I_P\ddot{\theta} + k_\theta \theta = b\left(\frac{1}{2} + a\right)L + M_{\frac{1}{4}} \end{cases} \tag{2.18}$$

For aerodynamics, when the steady-flow theory considered:

$$\begin{aligned} L &= 2\pi\rho_\infty bU^2\theta \\ M_{\frac{1}{4}} &= 0 \end{aligned} \tag{2.19}$$

where, per the thin-airfoil theory coming from the 'typical section' model used, the lift-curve slope is taken as 2π.

To simplify the notation, the uncoupled natural frequencies at zero airspeed can be introduced:

$$\omega_n = \sqrt{\frac{k_h}{m}}, \quad \omega_b = \sqrt{\frac{k_\theta}{I_p}} \tag{2.20}$$

Rearranging the equations of motions in the matrix form:

$$\begin{bmatrix} mb^2 & mb^2x_\theta \\ mb^2x_\theta & I_p \end{bmatrix} \begin{Bmatrix} \frac{\ddot{h}}{b} \\ \ddot{\theta} \end{Bmatrix} + \begin{bmatrix} mb^2\omega_h^2 & 2\pi\rho_\infty b^2U^2 \\ 0 & I_p\omega_\theta^2 - 2(\frac{1}{2} + a)\pi\rho_\infty b^2U^2 \end{bmatrix} \begin{Bmatrix} \frac{h}{b} \\ \theta \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}. \tag{2.21}$$

Several methods are developed to solve this system of equations, but the most efficient one for numerical solving is the PKNL method, which is a 'no-looping' variant of the PK method.

Chapter 3

Aeroelastic Model

In the execution of an aeroelastic analysis, the initial task is to determine the appropriate models to utilize. Given the interdisciplinary characteristics of aeroelasticity, it's crucial to strategically select both the structural and aerodynamic models. Moreover, the methodology employed to model the interaction between these two components is a significant aspect of the process. This approach ensures a robust and precise depiction of the aeroelastic phenomena under study.

3.1 Carrera Unified Formulation

Classical 1D models like (Timoshenko, Euler-Bernoulli, etc.) can deal properly with the bending of a compact cross-section beam, however, refined models are needed to describe the mechanical response of more complex boundaries (e.g. torsion) or geometrical (e.g. thin-walls) conditions. With CUF, displacement fields are obtained through a single form of expression in a unified manner regardless of the order of the theory (N), which is considered as an input of the analysis. The unified formulation of the cross-section displacement field is described by an expansion of a generic function (F_τ):

$$\boldsymbol{u} = F_\tau \boldsymbol{u}_\tau, \quad \tau = 1, 2, ..., M \tag{3.1}$$

where F_τ are functions of the cross section coordinates x and z, $\boldsymbol{u}_\tau$ is the displacement vector and M stands for the number of terms in the expansion [4].

In this thesis, aeroelastic analysis, therefore structural analyses of wings modeled as elongated and thin cross-sections conducted. So, considering those as 1D beam models is justified. The two types of polynomial expansions considered are the Taylor Expansion (TE) and Lagrange Expansion (LE).

3.1.1 Taylor Expansion for 1D Model

A possible choice for the type of expansion could be the Taylor polynomial expansion on the variables x and z. Considering a generic order N of expansion and indicating with M the number of the term of the expansion considered, the displacement field has the following form:

$$\begin{cases} u_x = \sum_{N_i=0}^{N} \left(\sum_{M=0}^{N_i} x^{N_i-M} z^M u_{x_{\frac{N_i(N_i+1)}{2}+M+1}} \right) \\ u_y = \sum_{N_i=0}^{N} \left(\sum_{M=0}^{N_i} x^{N_i-M} z^M u_{\frac{N_i(N_i+1)}{2}+M+1} \right) \\ u_x = \sum_{N_i=0}^{N} \left(\sum_{M=0}^{N_i} x^{N_i-M} z^M u_{\frac{N_i(N_i+1)}{2}+M+1} \right) \end{cases} \tag{3.2}$$

3.1.2 Lagrange Expansion for a 1D Model

Lagrange polynomials are usually given in terms of normalized coordinates. This is not a compulsory choice, since LE polynomials can also be implemented in terms of actual coordinates. However, the normalized formulation was preferred to take advantage of the isoparametric formulation [4]. In Figure 3.1, the 4-node cross-section element can be seen with normalized geometry. The Lagrange polynomials of the same element are:

$$F_\tau = \frac{1}{4}\left(1+\alpha\alpha_\tau\right)\left(1+\beta\beta_\tau\right) \quad \tau = 1,2,3,4 \tag{3.3}$$

where α and β are the normalized coordinates and α_τ and β_τ are the coordinates of the four nodes given in Table 3.1.

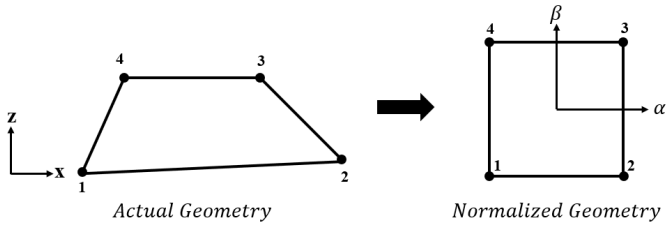


Figure 3.1: 4-node Lagrange element in actual coordinates and normalized coordinates

Node	α_τ	β_τ
1	-1	-1
2	1	-1
3	1	1
4	-1	1

Table 3.1: Normalized coordinates of the 4 nodes of the Lagrange element LE4

The displacement field for a model incorporating a 4-node element is defined as follows:

$$\begin{cases} u_x = F_1 u_{x_1} + F_2 u_{x_2} + F_3 u_{x_3} + F_4 u_{x_4} \\ u_y = F_1 u_{y_1} + F_2 u_{y_2} + F_3 u_{y_3} + F_4 u_{y_4} \\ u_z = F_1 u_{z_1} + F_2 u_{z_2} + F_3 u_{z_3} + F_4 u_{z_4} \end{cases} \tag{3.4}$$

In a LE4 element, the unknown variables are the displacement components of each node. This means that the problem unknowns are only physical translational displacements. Also, the problem unknowns can be placed on the physical surfaces of the body. These two fundamental characteristics are valid for each Lagrange element, regardless of the number of nodes [4].

3.1.3 Finite Element Formulation and CUF

The nodal displacement vector can be written as:

$$u_{\tau i} = \left\{ u_{x\tau i}, u_{y\tau i}, u_{z\tau i} \right\}^T, \quad \tau = 1, 2, \ldots, M, \quad i = 1, 2, \ldots, N_{EN} \tag{3.5}$$

where the index i indicates the element node and N_{EN} stands for the number of nodes per element. If a linear model is considered ($N = 1, M = 3$), and a two-node element is adopted, the element unknowns will be:

$$u_{\text{tri}} = \begin{Bmatrix} u_{x11} & u_{y11} & u_{z11} & u_{x21} & u_{y21} & u_{z21} & u_{x31} & u_{y31} & u_{z31} \\ u_{x12} & u_{y12} & u_{z12} & u_{x22} & u_{y22} & u_{z22} & u_{x32} & u_{y32} & u_{z32} \end{Bmatrix}^T \tag{3.6}$$

The displacement variables are interpolated along the axis of the beam utilizing the shape functions (N_i),

$$\boldsymbol{u} = N_i F_\tau \boldsymbol{u}_{\tau i} \tag{3.7}$$

N_i form functions depend on the element under consideration. When beam elements with 2 (B2), 3 (B3), and 4 nodes (B4) are considered whose shape functions are:

$$\begin{aligned} N_1 = \tfrac{1}{2}(1-r), \quad N_2 = \tfrac{1}{2}(1+r) \quad & \left(r_1 = -1, r_2 = 1\right) \\ N_1 = \tfrac{1}{2}r(r-1), \quad N_2 = \tfrac{1}{2}r(r+1), \quad N_3 = -(1+r)(1-r) \quad & \left(r_1 = -1, r_2 = 1, r_3 = 0\right) \\ N_1 = -\tfrac{9}{16}\left(r+\tfrac{1}{3}\right)\left(r-\tfrac{1}{3}\right)(r-1), \quad N_2 = \tfrac{9}{16}\left(r+\tfrac{1}{3}\right)\left(r-\tfrac{1}{3}\right)(r+1), & \\ N_3 = \tfrac{27}{16}(r+1)\left(r-\tfrac{1}{3}\right)(r-1), \quad N_4 = -\tfrac{27}{16}(r+1)\left(r+\tfrac{1}{3}\right)(r-1) & \\ \left(r_1 = -1, r_2 = 1, r_3 = -\tfrac{1}{3}, r_4 = \tfrac{1}{3}\right) & \end{aligned} \tag{3.8}$$

The natural coordinate (r) ranges from -1 to 1. r_i indicates the position of the node within the natural beam boundaries [4].

Stiffness Matrix, Mass Matrix, and Loading Vector

In the CUF, Finite Element matrices are formulated in terms of FN (Fundamental Nucleus). The FN is one of the main features of the CUF. FN is a compact formulation of the stiffness matrix and its mathematical statements are independent from the theory of structures used.

A compact form of the stiffness matrix can be obtained using the internal virtual work equation:

$$\delta L_{int} = \delta \boldsymbol{u}_{sj}^T \boldsymbol{k}^{\tau sij} \boldsymbol{u}_{\tau i} \tag{3.9}$$

where $\boldsymbol{k}^{\tau sij}$ is the stiffness matrix in the form of the FN.

Using the FN assembly technique, every Finite Element matrix or vector can be obtained. From the following inertial virtual work equation derived, the mass matrix in the form of FN can be found:

$$\delta L_{ine} = \delta \boldsymbol{u}_{sj}^T \boldsymbol{m}^{\tau sij} \boldsymbol{u}_{\tau i} \tag{3.10}$$

where $\boldsymbol{m}^{\tau sij}$ is the mass matrix in the form of FN.

The undamped dynamic problem can be written as follows:

$$\boldsymbol{M}\ddot{\boldsymbol{A}} + \boldsymbol{K}\boldsymbol{A} = \boldsymbol{P} \tag{3.11}$$

where $\boldsymbol{A}$ is the vector of the nodal unknowns and $\boldsymbol{P}$ is the loading vector. Introducing harmonic solutions, it is possible to compute the natural frequencies (ω_i) by solving an eigenvalue problem:

$$(-\omega_i^2 \boldsymbol{M} + \boldsymbol{K})\boldsymbol{A}_i = 0 \tag{3.12}$$

where A_i is the i^{th} eigenvector.

Using the following external virtual work equation derived, the loading vector $\boldsymbol{P}$ can be obtained:

$$\delta L_{ext} = F_s\left(x_P, z_P\right) N_j\left(y_P\right) \boldsymbol{P} \delta \boldsymbol{u}_{sj}^T \tag{3.13}$$

3.2 Aerodynamic Model

In this section, the implementation of the Doublet-Lattice Method (DLM) will be presented. Both VLM and DLM are widely known and used methods since they provide an effective low-order numerical solver for studying aerodynamics and aeroelasticity. They have considerable potential to be used in many areas of aerospace design [16].

The DLM is an aerodynamic finite element method for modeling oscillating interfering lifting surfaces in subsonic flows. It reduces to the Vortex-Lattice Method (VLM) at zero-reduced frequency. The number of finite elements required for accurate results depends on the aspect ratio and reduced frequency among other parameters [24].

3.2.1 Doublet Lattice Method (DLM)

In this thesis, for analyses MSC Nastran solver is used. MSC Nastran involves the Doublet-Lattice Method for the subsonic aerodynamic calculation. The theory is presented by Albano and Rodden (1969) [1], Giesing, Kalman, Rodden (1971) [7] and Giesing, Kalman, Rodden (1971) [8] and is not reproduced here, since it is not in the scope of this thesis. The following information is taken from MSC Nastran 2022.3 Aeroelastic Analysis User Guide [12].

The theoretical basis of the DLM is linearized aerodynamic potential theory. The undisturbed flow is uniform and is either steady or varying (gusting) harmonically. All lifting surfaces are assumed to lie nearly parallel to the flow.

Each of the interfering surfaces (panels) is divided into small trapezoidal lifting elements ("boxes") such that the boxes are arranged in strips parallel to the free stream with surface edges. The unknown lifting pressures are assumed to be concentrated uniformly across the one-quarter chord line of the box, and the surface normalwash boundary condition is satisfied at each of these points.

The representation of aerodynamic elements can be seen in the figure below.

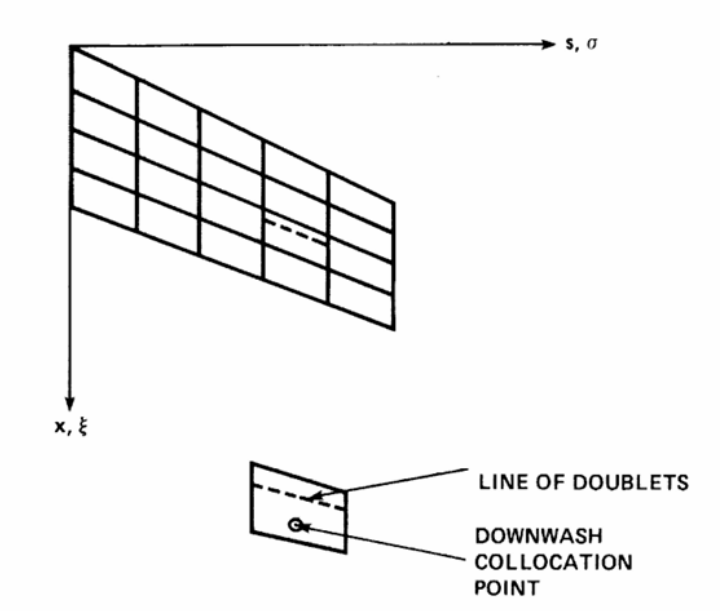


Figure 3.2: Aerodynamic Elements, location of doublets and collocation points for DLM (x and s are orthogonal coordinates on the surface S) [18]

3.3 Interconnection of Structure with Aerodynamics

The interconnection between structural and aerodynamic grids is made by interpolation. This allows independent selection of grid points of the structural and aerodynamic elements. The

interpolation method utilized is called 'splining'. The interpolation method used for the interconnection of the structural model with the aerodynamic model is the Infinite Plate Spline (IPS).

3.3.1 Infinite Plane Spline (IPS)

The following theory of Infinite Plane Splines is obtained from Hexagon's MSC Nastran 2022.3 Aeroelastic User Guide [12].

A surface spline is a mathematical tool used to find a surface function, $w(x, y)$, for all points (x, y) when w is known for a discrete set of points, $w_i = w(x_i, y_i)$. The Infinite Plate Spline is a special surface spline that is tailored to legacy flat plate aerodynamics. The theory introduces an infinite plate and solves for its deflections, given its deflections at a discrete set of points; that is, it is the problem of a plate with multiple deflecting supports. The surface spline is a smooth continuous function that will become nearly linear in x and y at large distances from the points (x_i, y_i). This problem can be solved in closed form.

The deflection of the plate is synthesized as the sum of deflections due to a set of point loads on the infinite plate. The deflection due to a single concentrated load is the fundamental solution and has polar symmetry. If the load is taken at $x_i = y_i = 0$, and polar coordinates are used ($x = rcos(\theta)$, $y = rsin(\theta)$, the governing differential equation is

$$D\nabla^4 w \;=\; D\frac{1}{r}\frac{d}{dr}\left\{ r\frac{d}{dr}\left[\frac{1}{r}\frac{d}{dr}\left(r\frac{dw}{dr}\right)\right]\right\} \;=\; q \tag{3.14}$$

The distributed load vanishes except near = 0. The general solution to the homogeneous form of equation 3.14 is

$$w = C_0 + C_1 r^2 + C_2 lnr + C_3 r^2 lnr \tag{3.15}$$

Set $C_2 = 0$ to keep the solution finite as $r \to 0$. Then multiply equation 3.15 by $2\pi r$ and integrate from $r = 0$ to (a small number) to obtain the concentrated force P,

$$P = \lim_{r\to 0}\int_0^{\varepsilon} 2\pi r q dr = \lim_{r\to 0} 2\pi r D\frac{d}{dr}\left[\frac{1}{r}\frac{d}{dr}\left(r\frac{dw}{dr}\right)\right] \tag{3.16}$$

Combining equation 3.15 and equation 3.16 leads to $C_3 = \frac{P}{8\pi D}$. The fundamental solution may therefore be written

$$w = A + Br^2 + (P/16\pi D)r^2 \ln r^2 \tag{3.17}$$

since $lnr = \frac{1}{2}lnr^2$ the fundamental solutions are superimposed to solve the entire plate problem with a solution of the form

$$w(x, y) = \sum [A_i + B_i r_i^2 + (p/16\pi D) r_i^2 lnr_i^2] \tag{3.18}$$

where $r_i^2 = (x - x_i)^2 + (y - y_i)^2$.

The remaining requirement is the satisfaction of the boundary condition at infinity: Radial lines emanating from loaded points (which all may be regarded as at the origin relative to infinity) appear to be straight lines. To do this, equation 3.18 is expanded in a series, assuming a large argument (x^2+y^2), and delete all terms of order $(x^2+y^2)\ln(x^2+y^2)$,(x^2+y^2),$x\ln(x^2+y^2)$, and $yln(x^2+y^2)$, leaving terms of order x, y, $ln(x^2+y^2)$, and 1. The details of the series expansion are given by Harder, MacNeal, and Rodden (1971) [11]. The deletion of the higher-order terms is accomplished by requiring

$$\sum B_i = 0 \tag{3.19}$$

$$\sum P_i = 0 \tag{3.20}$$

$$\sum x_i P_i = 0 \tag{3.21}$$

$$\sum y_i P_i = 0 \tag{3.22}$$

From equation 3.19 through equation 3.22 result in linear deflections at infinity; from equation 3.20 through equation 3.22 are also recognized as the equations of equilibrium. From 3.19 it is seen that

$$\sum (A_i + B_i r_i^2) \;=\; a_0 + a_1 x + a_2 y \tag{3.23}$$

A solution to the general spline problem, formed by superimposing solutions of equation 3.14, is given by

$$w(x, y) = a_0 + a_1 x + a_2 y + \sum_{i=1}^{N} K_i(x, y) P_i \tag{3.24}$$

where $K_i(x, y) = \frac{1}{16\pi D} r_i^2 \ln r_i^2$, $r_i^2 = (x - x_i)^2 + (y - y_i)^2$, P_i = concentrated load at (x_i, y_i). The $N + 3$ unknowns $(a_0, a_1, a_2, P_i; i = 1, N)$ are determined from the $N + 3$ equations

$$\sum P_i = \sum x_i P_i = \sum y_i P_i = 0 \tag{3.25}$$

and

$$w_j = a_0 + a_1 x_j + a_2 y_2 + \sum_{i=1}^{N} K_{ij} P_i \quad (j = 1, N) \tag{3.26}$$

where $K_{ij} = K_i(x_j, y_j)$. Note that $K_{ij} = K_{ji}$, and $K_{ij} = 0$ when $i = j$. The above derivation is also summarized by Harder and Desmarais (1972a) [10] and an application is shown. It is discussed further by Rodden, McGrew, and Kalman (1972) [23] and by Harder and Desmarais (1972b) [9]. Equation 3.24 can be written in matrix form:

$$w(x,y) = 1, x, y, K_1(x,y), K_2(x,y), \ldots, K_N(x,y) \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \\ P_1 \\ P_1 \\ P_2 \\ \vdots \\ P_N \end{Bmatrix} \tag{3.27}$$

Combining equation 3.24 and equation 3.25 into the matrix form:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ w_1 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & ... & 1 \\ 0 & 0 & 0 & x_1 & ... & x_N \\ 0 & 0 & 0 & y_1 & ... & y_N \\ 1 & x_1 & y_1 & 0 & ... & K_{1N} \\ 1 & x_2 & y_2 & K_{21} & ... & K_{2N} \\ \vdots & \vdots & \vdots & \vdots & ... & \vdots \\ 1 & x_N & y_N & K_{N1} & \cdots & 0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_2 \\ P_1 \\ P_2 \\ \vdots \\ P_N \end{bmatrix} = [C][P] \tag{3.28}$$

permits solution for the vector of a_i and P_i. The interpolation to any point in the (x, y) plane is then achieved by evaluating $w(x, y)$ from equation 3.27 at the desired points. This gives an overall equation of the form

$$\{w\}_a = \begin{bmatrix} 1 & x_{1a} & y_{1a} & K_{1a,1} & K_{1a,2} & \cdots & K_{1a,n} \\ 1 & x_{2a} & y_{2a} & K_{2a,1} & K_{2a,2} & \cdots & K_{2a,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_{na} & x_{na} & K_{na,1} & K_{na,2} & \cdots & K_{na,n} \end{bmatrix} [C]^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix} \tag{3.29}$$

Slopes of the aerodynamic panels, which are the negative of the slopes of the displacements, are also required. These are found by analytically differentiating equation 3.29 with respect to x:

$$\{\alpha\}_a = -\left[\frac{\partial w}{\partial x}\right]_a = -\begin{bmatrix} 0 & 1 & 0 & DK_{1a,1} & \cdots & DK_{1a,n} \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 1 & 0 & DK_{na,1} & \cdots & DK_{na,n} \end{bmatrix} [C]^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ w_1 \\ \vdots \\ w_N \end{bmatrix} \tag{3.30}$$

where:

$$DK_{ij} = \frac{\partial K_i(x_j, y_j)}{\partial x} = \left(\frac{x - x_i}{8\pi D}\right)(1 + lnr_i^2) \tag{3.31}$$

3.3.2 PK Method for Flutter Analysis

In this section, the following theory and equation are obtained from the NASTRAN User Guide's Theory section [12] and supplemented with Yuan and Zhang's Numerical Stabilization for Flutter Analysis Procedure [30].

There are numerous strategies for conducting flutter analysis to address the issue of frequency matching. The K-method and the PK-method are two of the most frequently utilized techniques. Both of these methods rely on the assumptions of simple harmonic motion, which are applicable at the flutter boundary but not above or below it. As a result, while these methods might predict identical flutter speeds and frequencies, their predictions for subcritical behavior might be inaccurate. It's been observed that the K-method's damping predictions are not dependable, as noted in Section 5.4.2 of Hodges and Pierce's book [14]. Hence, due to its proven reliability, the PK-method is often the preferred choice [30].

Following Rodden and Johnson [25], the equation for aeroelastic modal analysis using the PK-method can be written as:

$$[M_{hh}p^2 + (B_{hh} - \frac{1}{4}\rho\bar{c}VQ^I_{hh}/k)p + (K_{hh} - \frac{1}{2}\rho V^2 Q^R_{hh})]\{u_h\} = 0 \tag{3.32}$$

where M_{hh} is the generalized modal mass matrix, B_{hh} is the damping matrix and K_{hh} is the stiffness matrix. All matrix terms in the equation 3.32 are real. Q^R_{hh} and Q^I_{hh} are the real and imaginary parts of $Q_{hh}(k, Mach)$, where the inputs are reduced frequency k and Mach number. It should be noted that the circular frequency (ω) and the reduced frequency k are not independent of each other, $k = \frac{\omega\bar{c}}{2V}$.

The solution process for the PK Method was described thoroughly by Bellinger [2]. When the solution process of Bellinger is followed, equation 3.32 is rewritten in canonical form after multiplying it with 2:

$$[A - pI]\{\widetilde{u}\} = 0 \tag{3.33}$$

where $[A]$ is the doubled-size real matrix

$$[A] = \begin{bmatrix} 0 & I \\ -M^{-1}_{hh}(K_{hh} - \frac{1}{2}\rho V^2 Q^R_{hh}) & -M^{-1}_{hh}(B_{hh} - \frac{1}{4}\rho\bar{c}VQ^I_{hh}/k) \end{bmatrix} \tag{3.34}$$

$\{\widetilde{u}_h\}$ now incorporates both modal displacements, denoted as $\{u_h\}$, and their corresponding velocities, denoted as $\{\dot{u}_h\}$. During the solution procedure, the matrix from Equation (2) in its canonical form is initially transformed to the upper Hessenberg form via an elimination technique. Subsequently, complex conjugate eigenvalues are derived using the Double QR-Transformation method. It's noteworthy that most of the eigenvalues from Equation 3.33 exist as complex conjugate pairs

$$p = \mathrm{Re}(p) \pm i\mathrm{Im}(p) = \gamma\omega \pm i\omega\sqrt{1-\gamma^2} \approx \gamma\omega \pm i\omega \tag{3.35}$$

Where ω stands for the modal frequency and γ is the corresponding amplification rate coefficient. Since γ is quite small when near equilibrium or close to flutter points, the imaginary part of the eigenvalue p is simplified to the circular frequency in engineering applications. The reduced frequency k is calculated from the eigenvalue p as

$$k = \frac{\omega\overline{c}}{2V} = \frac{|\mathrm{Im}(p)|\overline{c}}{2V}. \tag{3.36}$$

To solve Equation 3.33 and ensure that Equation 3.36 is fulfilled, iterations are necessary. The process starts with a minimal non-zero value of the reduced frequency, denoted as k, which allows for the computation of $\frac{Q^I_{hh}}{k}$. The complex eigenvalue pairs can then be expressed as

$$p^{(j)}_{rs} = \omega^{(j)}_{rs}(\gamma^{(j)}_{rs} \pm i) \tag{3.37}$$

where r is a subscript representing the rank of the oscillatory mode, arranged in ascending order of frequency (for instance, 1s is less than 2s, and so on). The term s refers to the specific oscillatory mode being examined. The symbol j is used to indicate the iteration count for the solutions of the eigenvalues, which will be used to calculate the subsequent approximation of the nonzero reduced frequency.

$$k^{(j)}_s = \omega^{(j)}_{s,s}\overline{c}/2V \tag{3.38}$$

The iteration is converged when

$$\left|k^{(j)}_s - k^{(j-1)}_s\right| < \varepsilon \tag{3.39}$$

where ε stands for the convergence criterion. The converged complex eigenvalues are

$$p^{(c)}_{rs} = \omega^{(c)}_{rs}(\gamma^{(c)}_{rs} \pm i) \tag{3.40}$$

where only $p^{(c)}_{ss}$ complies with both Equations 3.33 and 3.36 as referenced in Bellinger [2], the procedure to identify the subsequent oscillatory mode started by incrementing s by one unit. As per the methodology of Rodden and Johnson [25], the initial prediction of the upcoming reduced frequency can be

$$k^{(0)}_s = \omega^{(c)}_{s,s-1}\overline{c}/2V \tag{3.41}$$

and the iteration process is continued until Equation 3.39 is satisfied.

Chapter 4

Deep Neural Networks

In this study, in order to predict flutter speeds for different configurations of an aircraft, deep neural networks are utilized. Before covering the details of deep neural networks, it is important to cover the basics of neural networks. The following sections are created with the Dive into Deep Learning, Release 0.17.2 [31].

4.1 Linear Regression

Regression refers to a set of methods for modeling the relationship between one or more independent variables and a dependent variable. In the natural sciences and social sciences, the purpose of regression is most often to characterize the relationship between the inputs and outputs. Machine learning, however, is often concerned with prediction.

Regression problems usually become visible when there is a need to predict a numerical value. Common examples can be listed as prediction prices, predicting the length of stay for patients at the hospitals, and many more. It is important to note that, not every prediction problem is a classical regression problem. There is a problem category called categorization, which aims to predict membership among a set of categories.

4.1.1 Basic Elements of Linear Regression

Linear regression utilizes a few simple assumptions. First, assuming that the relationship between the independent variable $\boldsymbol{x}$ and the dependent variable y is linear. which means y can be expressed as a weighted sum of the elements in $\boldsymbol{x}$, given some noise on the observations. The second one is to assume that any noise is well-behaved (following a Gaussian distribution)

In the terminology of machine learning, the dataset that contains past information is called a training dataset or training set. The outcome the learning algorithm trying to predict is called a label or a target. The independent variables upon which the predictions are based are called features or covariates.

Linear Model

Typically, n is used to denote the number of examples in a dataset. The indexing of the data examples are made with i, denoting each input as $\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}]^\top$ and the corresponding label as $y^(i)$.

Given a dataset, the goal is to choose weights $\mathbf{w}$ and bias b such that on average, the predictions made according to the model at hand best fit the true targets observed in the data. Models whose output prediction is determined by the affine transformation of input features are linear models, where the affine transformation is specified by the chosen weights and bias.

Collecting all features into vector $\mathbf{x}$ and all weights into a vector $\mathbf{w}$, it is possible to express the model compactly using a dot product:

$$\hat{y} = \mathbf{w}^\top \mathbf{x} + b \tag{4.1}$$

In the equation above, the vector $\mathbf{x}$ corresponds to features of a single dataset row. It is often convenient to refer to features of the entire dataset of n rows, via the design matrix $\mathbf{X}$. Here, $\mathbf{X}$ contains one row for every example and one column for every feature.

For a collection of features $\mathbf{X}$, the predictions $\hat{\mathbf{y}}$ can be expressed via the matrix-vector product:

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w} + b \tag{4.2}$$

where broadcasting is applied during the summation. Given features of a training set $\mathbf{X}$ and corresponding (known) labels $\mathbf{y}$, the goal of linear regression is to find the weight vector $\mathbf{w}$ and the bias term b that given features of a new data example sampled from the same distribution as $\mathbf{X}$, the new example's label will (in expectation) be predicted with the lowest error.

Loss Function

Before thinking about how to fit data with the model at hand, it is necessary to determine a measure of fitness. The loss function quantifies the distance between the real and predicted value of the target. The loss will usually be a non-negative number where the smaller values are better and perfect predictions incur a loss of 0. The most popular loss function in regression problems is the squared error. When the prediction for an example i is $\hat{y}^{(i)}$ and the corresponding true label is $y^{(i)}$, the squared error can be found as following:

$$l^{(i)}(\mathbf{w}, b) = \frac{1}{2}\left(\hat{y}^{(i)} - y^{(i)}\right)^2 \tag{4.3}$$

The constant $\frac{1}{2}$ makes no real difference but will prove notationally convenient, canceling out when the derivative of the loss is taken. Since the training dataset is given to us, and obviously out of our control, the empirical error is only a function of model parameters. To make things more concrete, the example below where a regression problem is plotted for a one-dimensional case can be observed.

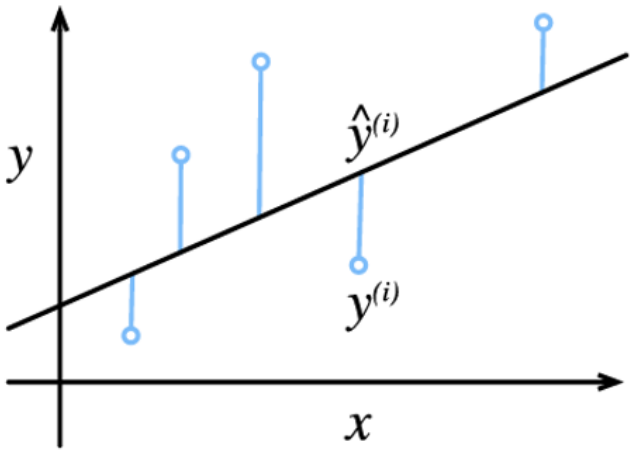


Figure 4.1: Fitted data with a linear model

To measure the quality of a model on the entire dataset of n examples, the losses on the training set are averaged. The equation used can be found below:

$$L(\mathbf{w}, b) = \frac{1}{n}\sum_{i=1}^{n} l^{(i)}(\mathbf{w}, b) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{2}\left(\mathbf{w}^\top \mathbf{x}^{(i)} + b - y^{(i)}\right)^2 \tag{4.4}$$

When training the model, the desire is to find the parameters $(\mathbf{w}^*, b^*)$ that minimize the total loss across all training examples:

$$\mathbf{w}^*, b^* = \underset{\mathbf{w}, b}{\operatorname{argmin}}\, L(\mathbf{w}, b). \tag{4.5}$$

Analytical Solution

Linear regression is a simple optimization problem. Therefore can be solved analytically by applying a simple formula. To start, the bias b can be taken into the parameter $\mathbf{w}$ by appending a column to the design matrix consisting of all ones. Then the prediction problem is to minimize $\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2$. There is just one critical point on the loss surface and it corresponds to the minimum of the loss over the entire domain. Taking the derivative of the loss with respect to $\mathbf{w}$ and setting it equal to zero yields the analytic (closed form) solution:

$$\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} \tag{4.6}$$

While simple problems like linear regression may admit analytic solutions, this is not usually the case.

Minibatch Stochastic Gradient Descent

Even in cases where the models cannot be solved analytically, the models can still be trained effectively in practice. The key technique for optimizing nearly any deep learning model consists of iteratively reducing the error by updating the parameters in a direction that incrementally lowers the loss function. This algorithm is called gradient descent.

The most naive application of gradient descent consists of taking the derivative of the loss function, which is the average of the losses computed on every single example in the dataset.

In practice, this can be extremely slow: the requirement is to pass over the entire dataset before making a single update. Thus, often, sampling a random minibatch of examples every time we need to compute the update, a variant called minibatch stochastic gradient descent.

In each iteration, first, a minibatch $\mathcal{B}$ is randomly sampled, which consists of a fixed number of training examples. Then the derivative (gradient) of the average loss on the minibatch with regard to model parameters gets computed. Finally, the gradient gets multiplied by a predetermined positive value η and subtracts the resulting term from the current parameter values.

We can express the update mathematically as follows:

$$(\mathbf{w}, b) \leftarrow (\mathbf{w}, b) - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \partial_{(\mathbf{w},b)} l^{(i)}(\mathbf{w}, b). \tag{4.7}$$

To summarize, the steps of the algorithm are the following: (i) initialize the values of the model parameters, typically at random; (ii) iteratively sample random minibatches from the data, updating the parameters in the direction of the negative gradient. For quadratic losses and affine transformations, this can be written as follows:

$$\begin{aligned} \mathbf{w} &\leftarrow \mathbf{w} - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \partial_{\mathbf{w}} l^{(i)}(\mathbf{w}, b) = \mathbf{w} - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \mathbf{x}^{(i)} \left(\mathbf{w}^\top \mathbf{x}^{(i)} + b - y^{(i)} \right), \\ b &\leftarrow b - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \partial_b l^{(i)}(\mathbf{w}, b) = b - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \left(\mathbf{w}^\top \mathbf{x}^{(i)} + b - y^{(i)} \right). \end{aligned} \tag{4.8}$$

Note that $\mathbf{w}$ and $\mathbf{x}$ are vectors in the equation above. η denotes the learning rate. It should be emphasized that the values of the batch size and learning rate are manually pre-specified and not typically learned through model training. These parameters that are tunable but not updated in the training loop are called hyperparameters. Hyperparameter tunning is the process by which hyperparameters are chosen, and typically requires adjustment based on the results of the training loop as assessed on a separate validation dataset.

After training for some predetermined number of iterations (or until some other stopping criteria are met), estimated model parameters are recorded, denoted $\hat{\mathbf{w}}, \hat{b}$. Note that even if our function is truly linear and noiseless, these parameters will not be the exact minimizers of the loss because, although the algorithm converges slowly towards the minimizers it cannot achieve it exactly in a finite number of steps.

Linear regression happens to be a learning problem where there is only one minimum over the entire domain. However, for more complicated models, like deep networks, the loss surfaces contain many minima.

Making Predictions with the Learned Model

Given the learned linear regression model $\hat{\mathbf{w}}^\top \mathbf{x} + \hat{b}$, now the target can be estimated (not contained in the training data). Estimating the targets given features is commonly called prediction.

The Normal Distribution and Squared Loss

Linear regression was invented by Gauss in 1795, who also discovered the normal distribution (also called Gaussian). It turns out that the connection between normal distribution and linear regression runs deeper than common parentage. The probability density of a normal distribution with a mean μ and variance σ^2 (standard deviation σ) is given as:

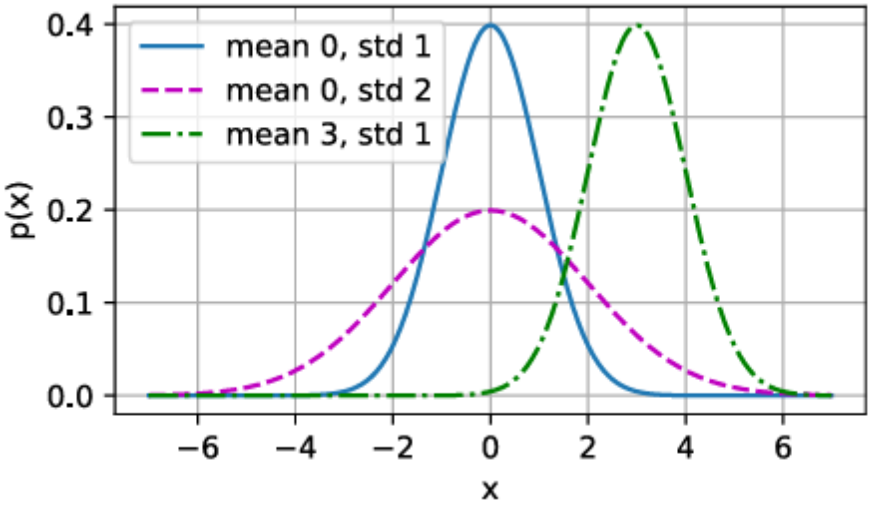


Figure 4.2: Change in normal distribution with mean and standard deviation

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right). \tag{4.9}$$

As can be seen in the figure above, changing the mean corresponds to a shift along the $x - axis$, and increasing the variance spreads the distributions outwards, lowering its peak.

One way to motivate linear regression, with the mean squared error loss function, is to formally assure that the observations arise from noisy observations, where the noise is normally distributed as follows:

$$y = \mathbf{w}^\top \mathbf{x} + b + \epsilon \text{where} \epsilon \sim \mathcal{N}(0, \sigma^2). \tag{4.10}$$

Thus, the likelihood of seeing a particular y for a given $\mathbf{x}$ via

$$P(y \mid \mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y - \mathbf{w}^\top \mathbf{x} - b)^2\right) \tag{4.11}$$

According to the principle of maximum likelihood, the best values of parameters $\mathbf{w}$ and b are those that maximize the likelihood of the entire dataset:

$$P(\mathbf{y} \mid \mathbf{X}) = \prod_{i=1}^{n} p(y^{(i)} | \mathbf{x}^{(i)}). \tag{4.12}$$

Estimators chosen according to the principle of maximum likelihood are called maximum likelihood estimators. While maximizing the product of many exponential functions might look difficult, significant simplifications can be made, without changing the objective by maximizing the log of the likelihood instead:

$$-\log P(\mathbf{y} \mid \mathbf{X}) = \sum_{i=1}^{n} \frac{1}{2} \log(2\pi\sigma^2) + \frac{1}{2\sigma^2}\left(y^{(i)} - \mathbf{w}^\top \mathbf{x}^{(i)} - b\right)^2 \tag{4.13}$$

Now, one more assumption needs to be made that σ is some fixed constant. Thus it is possible to ignore the first term because it does not depend on $\mathbf{w}$ or b. As a result, the second

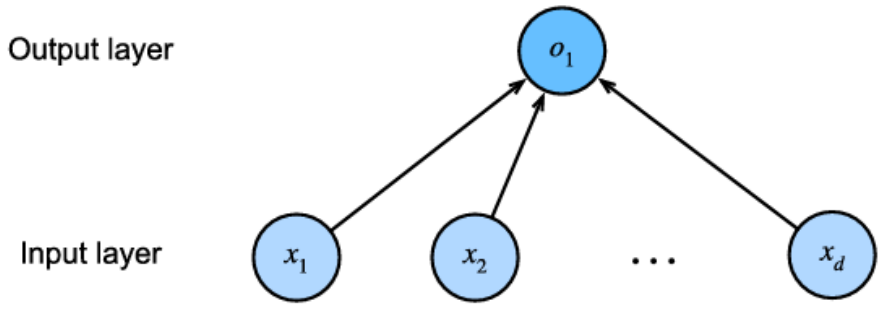


Figure 4.3: Linear regression is a single-layer neural network

term is now identical to the squared error loss introduced, except for the multiplicative constant $\frac{1}{\sigma^2}$. Fortunately, the solution does not depend on the σ. It follows that minimizing the mean squared error is equivalent to the maximum likelihood estimation of a linear model under the assumption of additive Gaussian noise.

4.1.2 From Linear Regression to Deep Networks

In the figure below, the linear regression model can be seen as a neural network. It should be noted that these diagrams highlight the connectivity pattern such as how each input is connected to the output, but not the values taken by weights or biases.

For the neural network shown above, the inputs are $x_1, ..., x_d$ so the number of inputs (or feature dimensionality) in the input layer is d. The output of the network in the figure above is o_1, so the number of outputs in the output layer is 1. Note that the input values are all given and there is just a single computed neuron. Focusing on where the computation takes place, conventionally the input layer is not considered as a layer when counting them. Linear regression models can be thought of as neural networks consisting of just a single artificial neuron, or as single-layer neural networks.

Since for linear regression, every input is connected to every output (in this case there is only one output), we can regard this transformation as a fully connected layer or dense layer.

4.2 Multilayer Perceptrons

The simplest deep networks are called multilayer perceptions, and they consist of multiple layers of neurons each connected to those in the layer below (from which they receive input and those above (which they, in turn, influence).

4.2.1 Hidden Layers

Linearity implies the weaker assumption of monotonicity: that any increase in a feature at hand must either always cause a decrease in the model's output (if the corresponding weight is positive), or always cause a decrease in our model's output (if the corresponding weight is negative). However linear models are not always the right tool for the job. It is easy to create examples that violate monotonicity. So it is safe to say it has limitations, but it is possible to

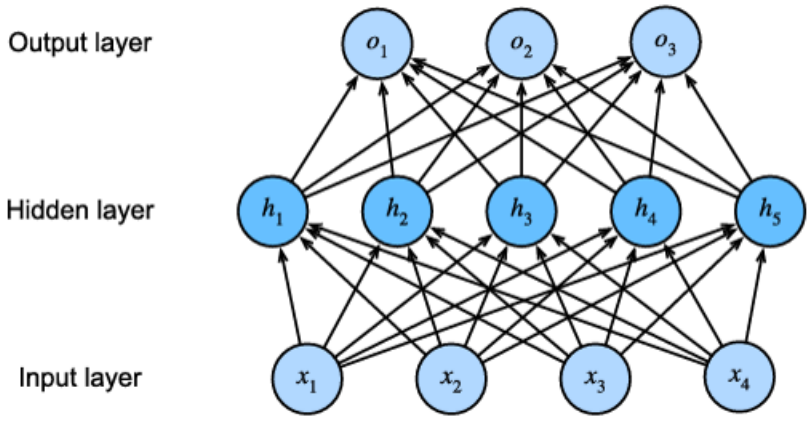


Figure 4.4: An MLP with a hidden layer of 5 hidden units

overcome these limitations of linear models and handle a more general class of functions by incorporating one or more hidden layers. The easiest way to do this is to stack many fully connected layers on top of each other. Each layer feeds into the layer above it until outputs are generated. The first $L-1$ layers can be thought of as a representation and the final layer as our linear predictor. This architecture is commonly called a multilayer perception, MLP.

Above, an MLP has 4 inputs, 3 outputs, and its hidden layer contains 5 hidden units. Since the input layer does not involve any calculations, producing outputs with this network requires implementing the computations for both the hidden and output layers; thus, the number of layers in this MLP is 2. Note that these layers are both fully connected. Every input influences every neuron in the hidden layer, and each of these in turn influences every neuron in the output layer. However, the parameterization of MLPs with fully connected layers can be prohibitively high, which may motivate trade-off between parameter saving and model effectiveness even without changing the input or output size.

From Linear to Nonlinear

For a one-hidden-layer MLP whose hidden layer has h hidden units, denoted $\mathbf{H}$, the outputs of the hidden layer, which are hidden representations. Since the hidden and output layers are both fully connected, we have a hidden-layer weights $\mathbf{W}^{(1)}$ and biases $\mathbf{b}^{(1)}$ and output-layer weights $\mathbf{W}^{(2)}$ and biases $\mathbf{b}^{(2)}$. Formally, outputs $\mathbf{O}$ of the one-hidden-layer MLP are calculated as follows:

$$\begin{aligned} \mathbf{H} &= \mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)}, \\ \mathbf{0} &= \mathbf{H}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}. \end{aligned} \tag{4.14}$$

In order to realize the potential of multilayer architectures, on more key ingredient is needed: a nonlinear activation function σ to be applied to each hidden unit following the affine transformation. The outputs of activation functions are called activations. In general, with activation functions in place, it is no longer possible to collapse an MLP into a linear model.

$$\begin{aligned} \mathbf{H} &= \sigma(\mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)}), \\ \mathbf{0} &= \mathbf{H}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}. \end{aligned} \tag{4.15}$$

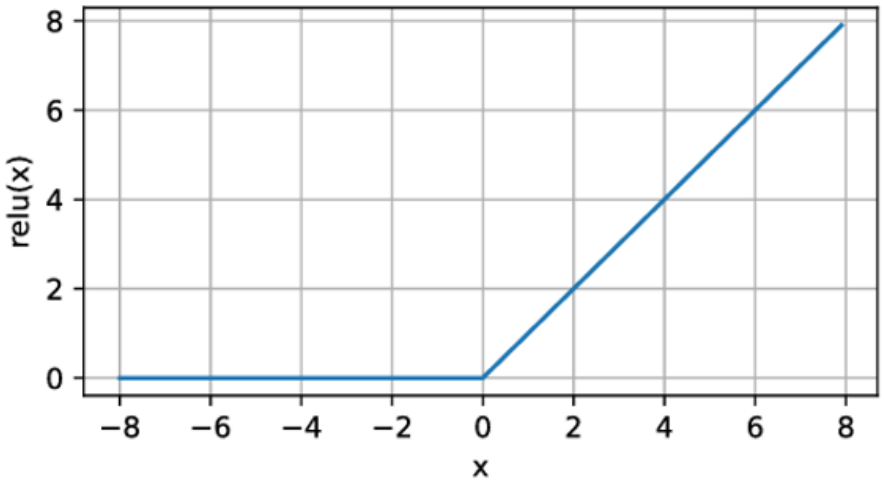


Figure 4.5: Representation of the ReLU funciton

Universal Approximators

MLPs can capture complex interactions among inputs via their hidden neurons, which depend on the values of each of the inputs. It is quite possible to design hidden nodes to perform arbitrary computation, for instance, basic logic operations on a pair of inputs. Moreover, for certain choices of the activation function, it is widely known that MLPs are universal approximators. Even with a single-hidden-layer network, given enough nodes and the right set of weights, it is possible to model any function, though learning that function is the hard part.

Just because a single-hidden-layer network can learn any function, it does not mean that one should try to solve all problems with single-hidden-layer networks. As a matter of fact, it is much more sensible to use deeper (wider) networks for approximation of the many functions in a more compact way.

4.2.2 Activation Functions

Activation functions decide whether a neuron should be activated or not by calculating the weighted sum and further adding bias to it. They are differentiable operators to transform input signals into outputs, while most of them add non-linearity.

ReLU Function

The most popular choice, due to both simplicity of implementation and its good performance on a variety of predictive tasks, is the rectified linear unit, (ReLU). ReLU provides a quite simple nonlinear transformation. Given an element x, the function is defined as the maximum of that element and 0.

$$\text{ReLU}(x) = \max(x, 0). \tag{4.16}$$

Informally, the ReLU function retains only positive elements and discards all negative elements by setting the corresponding activations to 0. As you can see in the figure below, the activation function is piecewise linear.

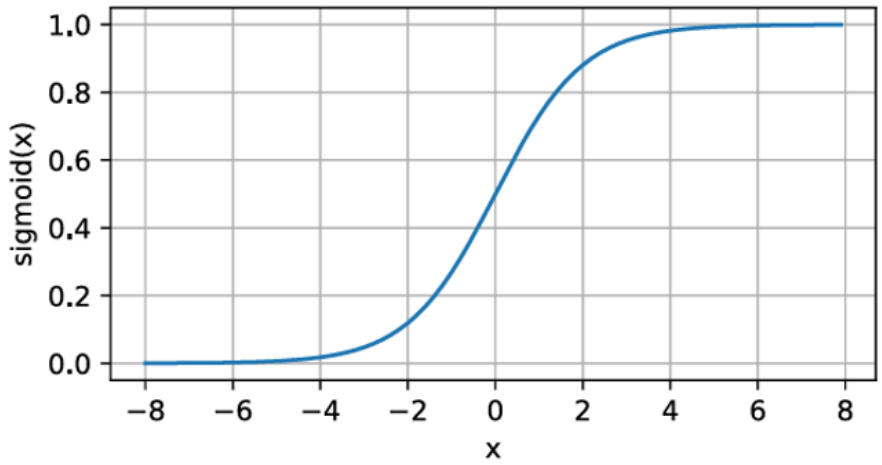


Figure 4.6: Sigmoid function on a 2D plot

When the input is negative, the derivative of the ReLU function is 0, and when the input is positive, the derivative of the ReLU function is 1.

Sigmoid Function

The sigmoid function transforms its inputs, for which values lie in the domain $\mathbb{R}$, to outputs that line on the interval (0, 1). For that reason, the sigmoid is often called a squashing function.

$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x)}. \tag{4.17}$$

In the earliest neural networks, scientists were interested in modeling biological neurons that either fire or not fire. Thus the pioneers of this field, going all the way back to McCulloch and Pitts, the inventors of the artificial neuron, focused on thresholding units. A thresholding activation takes value 0 when its input is below some threshold and value 1 when the input exceeds the threshold. Later on, when attention shifted to gradient based learning, the sigmoid function was a natural choice because it is a smooth, differentiable approximation to a thresholding unit. Sigmoids are still widely used especially for binary classification problems, when the need to interpret the outputs as probabilities for binary classification problems. However, sigmoid has mostly been replaced by the simpler and more easily trainable ReLU for most use in hidden layers.

In the figure below, one can see the representation of a sigmoid function on a 2D plot.
The derivative of the sigmoid function is given by the following equation:

$$\frac{d}{dx}\text{sigmoid}(x) = \frac{\exp(-x)}{(1 + \exp(-x))^2} = \text{sigmoid}(x)\,(1 - \text{sigmoid}(x)) \tag{4.18}$$

The derivative of the sigmoid function is plotted below. Note that when the input is zero, the derivative of the sigmoid function reaches a maximum of 0.25. As the input diverges from 0 in either direction, the derivative approaches 0.

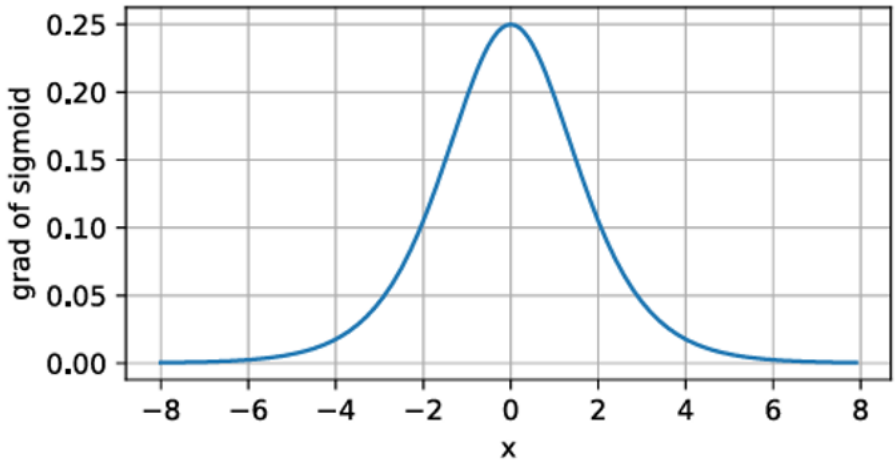


Figure 4.7: The derivative of sigmoid function plotted

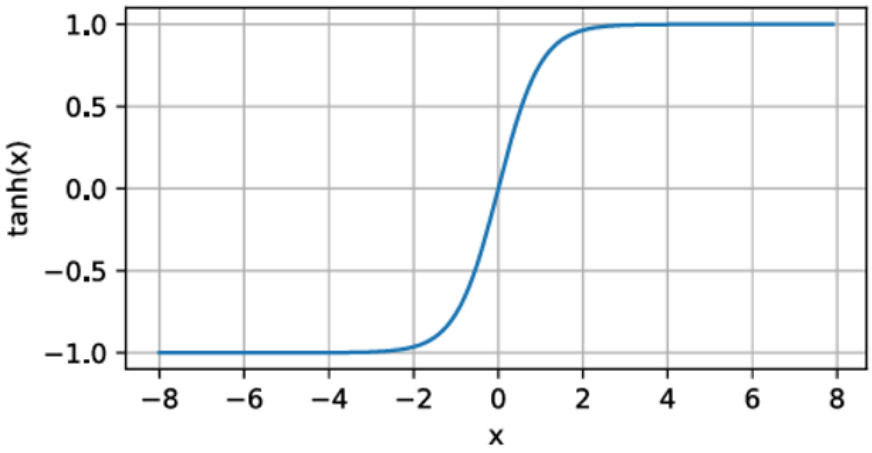


Figure 4.8: Plot of the tanh function

Tanh Function

Like the sigmoid function, the tanh (hyperbolic tangent) function also squashes its inputs, transforming them into elements on the interval between -1 and 1:

$$\tanh(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)}. \tag{4.19}$$

The plot of the tanh function can be found below. Note that as the input nears 0, the tanh function approaches a linear transformation. Although the shape of the function is similar to that of the sigmoid function, the tanh function exhibits point symmetry about the origin of the coordinate system.
The derivative of the tanh function is:

$$\frac{d}{dx}\tanh(x) = 1 - \tanh^2(x) \tag{4.20}$$

The derivative of the tanh function is plotted below. As the input nears 0, the derivative of the tanh function approaches a maximum of 1. As the input moves away from 0 in either direction,

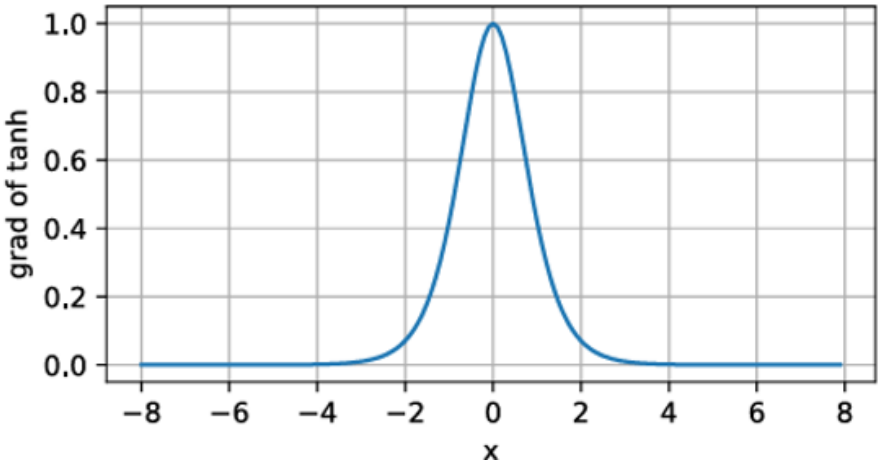


Figure 4.9: Plot of the derivative of the tanh function

the derivative of the tanh function approaches 0.

4.3 Backpropagation

Backpropagation refers to the method of calculating the gradient of neural network parameters. In short, the method traverses the network in reverse order, from the output to the input layer according to the chain rule of the calculus. The algorithm stores any intermediate variables (partial derivatives) required while calculating the gradient with respect to some parameters. Assume that functions $Y = f(X)$ and $Z = g(Y)$, in which the input and the output of X,Y,Z are tensors of arbitrary shapes. By using the chain rule, it is possible to compute the derivative of Z with respect to X via

$$\frac{\partial Z}{\partial X} = \text{prod}\left(\frac{\partial Z}{\partial Y}, \frac{\partial Y}{\partial X}\right) \tag{4.21}$$

Here, prod operator has been used to multiply its arguments after the necessary operations, such as transposition and swapping input positions, have been carried out. For vectors, this is straightforward: it is simply matrix-matrix multiplication. For higher dimensional tensors, the appropriate counterpart is used. The operator prod hides all the notation overhead.

Assume in a simple network with hidden one layer are $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$. The objective of backpropagation is to calculate the gradients $\partial J/\partial \mathbf{W}^{(1)}$ and $\partial J/\partial \mathbf{W}^{(2)}$. To accomplish this, the chain rule should be applied and in turn, the gradient of each intermediate variable and parameter should be calculated. The order of calculations are reversed as the name suggests, relative to those performed in forward propagation, since now the movement is from outputs to inputs. The first step is to calculate the gradients of the objective function $J = L + s$ with respect to the loss term L and the regularization term s.

$$\frac{\partial J}{\partial L} = 1 \text{ and } \frac{\partial J}{\partial s} = 1 \tag{4.22}$$

Next, the computation of the gradient of the objective function with respect to the variable of the output layer $\mathbf{o}$ according to the chain rule.

$$\frac{\partial J}{\partial \mathbf{o}} = \text{prod}\left(\frac{\partial J}{\partial L}, \frac{\partial L}{\partial \mathbf{o}}\right) = \frac{\partial L}{\partial \mathbf{o}} \tag{4.23}$$

Next, the calculation of the gradients of the regularization term with respect to both parameters:

$$\frac{\partial s}{\partial \mathbf{W}^{(1)}} = \lambda \mathbf{W}^{(1)} \text{ and } \frac{\partial s}{\partial \mathbf{W}^{(2)}} = \lambda \mathbf{W}^{(2)} \tag{4.24}$$

Now, it is possible to calculate the gradient $\partial J / \partial \mathbf{W}^{(2)}$ of the model parameters closest to the output layer. Using the chain rule yields:

$$\frac{\partial J}{\partial \mathbf{W}^{(2)}} = \text{prod}\left(\frac{\partial J}{\partial \mathbf{o}}, \frac{\partial \mathbf{o}}{\partial \mathbf{W}^{(2)}}\right) + \text{prod}\left(\frac{\partial J}{\partial s}, \frac{\partial s}{\partial \mathbf{W}^{(2)}}\right) = \frac{\partial J}{\partial \mathbf{o}} \mathbf{h}^\top + \lambda \mathbf{W}^{(2)} \tag{4.25}$$

To obtain the gradient with respect to $\mathbf{W}^{(1)}$, it is necessary to continue with the backpropagation along the output layer to the hidden layer. The gradient with respect to the hidden layer's outputs $\partial J / \partial \mathbf{h}$ is given by

$$\frac{\partial J}{\partial \mathbf{h}} = \left(\frac{\partial J}{\partial \mathbf{o}}, \frac{\partial \mathbf{o}}{\partial \mathbf{h}}\right) = \mathbf{W}^{(2)\top} \frac{\partial J}{\partial \mathbf{o}} \tag{4.26}$$

Since the activation function ϕ applies elementwise, calculating gradient $\partial J / \partial \mathbf{z}$ of the intermediate variable $\mathbf{z}$ requires to use the elementwise multiplication operator denoted by $\odot$:

$$\frac{\partial J}{\partial \mathbf{z}} = \text{prod}\left(\frac{\partial J}{\partial \mathbf{h}}, \frac{\partial \mathbf{h}}{\partial \mathbf{z}}\right) = \frac{\partial J}{\partial \mathbf{h}} \odot \phi'(\mathbf{z}) \tag{4.27}$$

Finally, it is possible to obtain the gradient $\partial J / \partial \mathbf{W}^{(1)}$ of the model parameters closest to the input layer. According to the chain rule:

$$\frac{\partial J}{\partial \mathbf{W}^{(1)}} = \text{prod}\left(\frac{\partial J}{\partial \mathbf{z}}, \frac{\partial \mathbf{z}}{\partial \mathbf{W}^{(1)}}\right) + \text{prod}\left(\frac{\partial J}{\partial s}, \frac{\partial s}{\partial \mathbf{W}^{(1)}}\right) = \frac{\partial J}{\partial \mathbf{z}} \mathbf{x}^\top + \lambda \mathbf{W}^{(1)}. \tag{4.28}$$

4.4 Optimizer for the Optimization Problem: Adam

Adam, Adaptive Moment Estimation, is usually the choice of optimizer for the problems, thanks to its nature that combines the advantages of the prior optimizers like Stochastic Gradient Descent with its inherent resilience to redundant data, Minibatch Stochastic Gradient Descent with its efficiency arising from vectorization, Momentum with its mechanism for aggregating a history of past gradients to accelerate convergence, Adagrad with its per-coordinate scaling to allow for a computationally efficient precondition, and RMSprop with its decoupling per-coordinate scaling from a learning rate adjustment. Adam combines all these techniques into one efficient learning algorithm. As expected, this is an algorithm that has become rather popular as one of the more robust and effective optimization algorithms to use in deep learning. However, it is not without issues. In particular, there are situations where Adam can diverge due to poor variance control.

4.4.1 Adam Algorithm

One of the key components of Adam is that it uses exponential weighted moving averages (also known as leaky averaging) to obtain an estimate of both the momentum and the second moment of the gradient. That is, it uses state variables:

$$\begin{aligned} \mathbf{v}_t &\leftarrow \beta_1 \mathbf{v}_{t-1} + (1 - \beta_1)\mathbf{g}_t, \\ \mathbf{s}_t &\leftarrow \beta_2 \mathbf{s}_{t-1} + (1 - \beta_2)\mathbf{g}_t^2. \end{aligned} \tag{4.29}$$

Here β_1 and β_2 are nonnegative weighting parameters. Common choices for them are $\beta_1 = 0.9$ and $\beta_2 = 0.999$. That is, the variance estimate moves much more slowly than the momentum term. Note that with the initialization of $\mathbf{v}_0 = \mathbf{s}_0 = 0$, there is a significant amount of bias initially towards smaller values. This can be addressed by using the fact that $\sum_{i=0}^{t} \beta^i = \frac{1-\beta^t}{1-\beta}$ to re-normalize terms. Correspondingly the normalized state variables are given by

$$\hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1 - \beta_1^t} \text{ and } \hat{\mathbf{s}}_t = \frac{\mathbf{s}_t}{1 - \beta_2^t} \tag{4.30}$$

Armed with the proper estimates, it is now possible to write out the update equations. First, rescaling of the gradient in a manner akin to that of RMSprop to obtain:

$$\mathbf{g}_t' = \frac{\eta \hat{\mathbf{v}}_t}{\sqrt{\hat{\mathbf{s}}_t} + \epsilon}. \tag{4.31}$$

Unlike RMSprop, the update for this case uses the momentum $\hat{\mathbf{v}}_t$ rather than the gradient itself.

Now that there are all the pieces in place to compute updates. There is a simple update of the form:

$$\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} - \mathbf{g}_t' \tag{4.32}$$

Chapter 5

Preliminary Investigations

Machine learning models need large training, validation, and test data to create efficient algorithms. Since there are no available data sources, the need for the creation of reliable data for the algorithm to train on is crucial. To achieve this, with cost and time also in mind, optimum parameters for aeroelastic analysis are needed to be found. In this section, utilizing MUL2 Software and MSC Nastran; static, dynamic, static aeroelastic, and flutter analyses will be conducted and compared with Femap Software to validate.

5.1 Static and Dynamic Analysis of An Isotropic Beam

In the following segment, a case involving an isotropic beam is taken into consideration. The first aspect to be studied is the static response of the structure when it is under load. This leads to an exploration of its dynamic characteristics, with a focus on the beam's frequencies and modes. The investigations are primarily conducted using the MUL2 software, while the Femap software is utilized later to confirm the findings.

5.1.1 Static Analysis

A beam composed of an isotropic material is examined. This beam, which is fixed at one end and subjected to a load at the free end, was previously studied in [20]. The properties of the material used are detailed in Table 5.1.

E [MPa]	ν	G [MPa]	ρ [kg/m^3]
73000	0.3	28076.92	2700

Table 5.1: Material characteristics

The beam, as depicted in Figure 5.1, is characterized by a length (L) of 1 m, a width (b) of 0.2 m, and a thickness (h) of 0.01 m. The reference system, originating from the center of the interlocking section, is illustrated in the same figure. The load, with an intensity (P) of $-100N$, is applied downwards at the coordinate point ($x = 0$ m, $y = 1$ m, $z = 0.005$ m). This point corresponds to the middle of the free-end section.

The MUL2 software is utilized for the static analysis. The FEM model is also defined (as shown in Figure 7.2), where the beam is discretized into 10 B4 elements. Each of these beam elements

comprises two internal nodes (denoted by the symbol ■) and two external nodes (denoted by the symbol •).

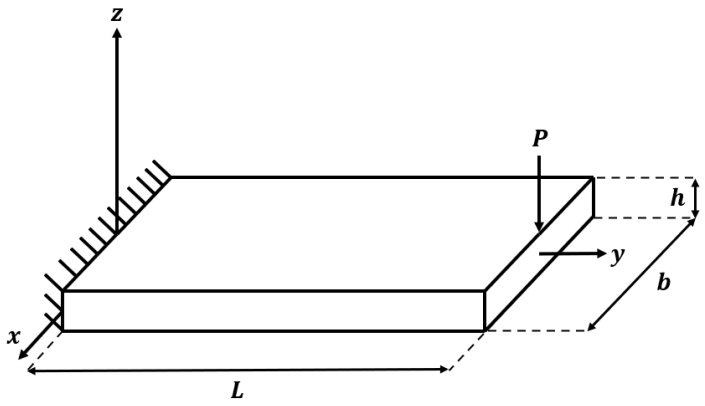


Figure 5.1: Beam Diagram

Considering the common nodes shared by two adjacent elements, a total of 31 nodes are utilized. The coordinates of these 31 nodes within the xyz reference system are defined in one of the input files. Notably, these nodes are equally distributed along the y-axis, with x and z values set to 0.

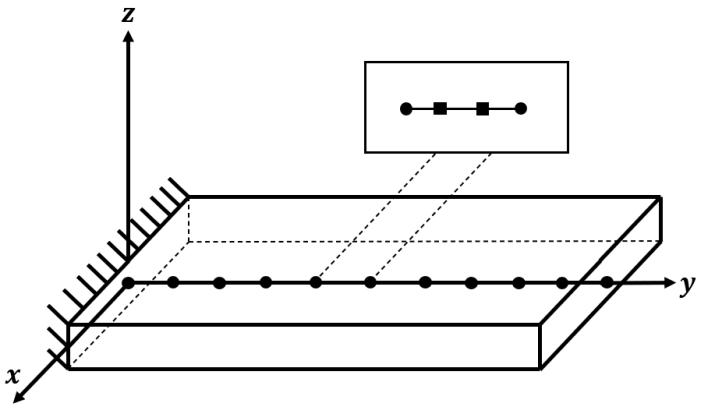


Figure 5.2: Nodes in the beam element

A rectangular cross-section with height of $\boldsymbol{h}$ and width of $\boldsymbol{b}$ is defined for each node. Figure 5.3 shows the cross-section with 4-node and 9-node $QUAD$ elements.

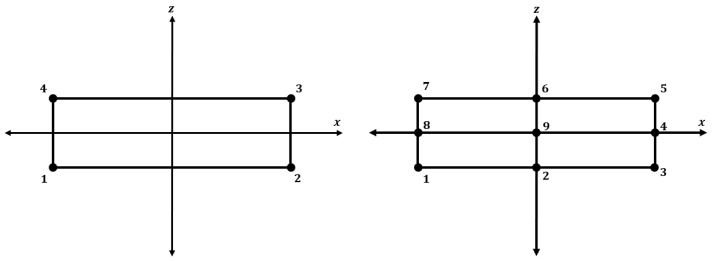


Figure 5.3: Nodes in beam cross-section

Note that, the type of structural models adopted are *Taylor Expansion* (TE) and *Lagrange Expansion* (LE).

The maximum value of z displacement u_z and maximum principal stress in y direction σ_{yy} values are obtained using different TE and LE models. The number of DOF (degree of freedom) is also given with the results obtained, which indicates the computational costs.

Model	DOF	u_z [m]	σ_{yy} [MPa]
EBBT A.	155	-2.739E-02	29.999
TBT A.	155	-2.740E-02	29.999
EBBT	155	-2.740E-02	29.994
TBT	155	-2.740E-02	29.994
TE1	279	-2.739E-02	29.997
TE2	558	-2.582E-02	34.213
TE3	930	-2.668E-02	40.381
TE4	1395	-2.675E-02	40.026
2LE4	558	-2.047E-02	31.286
1LE9	837	-2.669E-02	40.450
2LE9	1395	-2.670E-02	40.078

Table 5.2: Static analysis with MUL2

Stress and displacement values can also be obtained analytically with most used beam models, assuming that the beam is homogeneous and isotropic. The beam is clamped from one end and loaded from another with a concentrated force of -100 N.

Euler-Bernoulli

Since the cross-section is symmetrical (moment of inertia $I_{xy} = 0$ and the load is applied to xz plane, the bending moment can be written as:

$$M_x = -EI_x \frac{d^2 u_z}{dy^2} \tag{5.1}$$

The bending moment is related to the curvature of the beam by the flexural stiffness EI_x and the moment of inertia can be calculated with the following: $I_x = \frac{bh^3}{12}$. The derivation of the moment is equal to the shear:

$$\frac{dM_x}{dy} = T_z \tag{5.2}$$

which can be used to obtain the following equation:

$$-EI_x \frac{d^3 u_z}{dy^3} = T_z \tag{5.3}$$

where the shear is equal to the P load imposed on the beam. Applying the necessary boundary conditions (clamped end, zero moment at the free end), the expression for the elastic line can be obtained:

$$u_z(y) = \frac{Py^2}{6EI_x}(3L - y) \tag{5.4}$$

Thus the maximum displacement can be found:

$$u_z(L) = 2.739 \times 10^{-2} m \tag{5.5}$$

The σ_{yy} can also be obtained to compare with the MUL2 results:

$$\sigma_{yy} = E\epsilon_{yy} = Ez\frac{d^2 u_z(0)}{dy^2} = 29.99 MPa \tag{5.6}$$

Timoshenko

In this case, angular deflections are not zero, as a result, the beam cross-section is no longer perpendicular to the principal axis. Therefore the equation for Timoshenko can be written as follows:

$$\begin{cases} T_z = GA^*_{yz}\left(\frac{du_z}{dy} + \theta\right) \\ M_x = EI_x\frac{d\theta}{dy} \end{cases} \tag{5.7}$$

θ is the rotation about x-axis while GA^*_{yz} is the shear stiffness of the beam section. Since the shear is equal to the P and the moment is equal to the $P(y-L)$ and after the boundary conditions applied(no rotation and displacement in the clamped end) the following elastic line equation can be written:

$$u_z = \frac{P}{GA^*}y - \frac{Py^2}{6EI_x}(y - 3L) \tag{5.8}$$

Using this equation, maximum displacement can be found:

$$u_z(L) = 2.740 \cdot 10^{-2} m \tag{5.9}$$

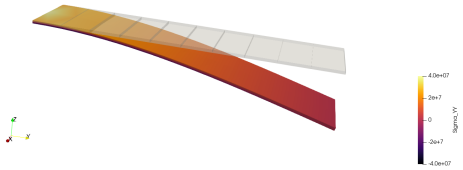
The principal stress can be found as follows:

$$\sigma_{yy} = E\epsilon_{yy} = -Ez\frac{d\theta(0)}{dy} = 29.99 MPa \tag{5.10}$$

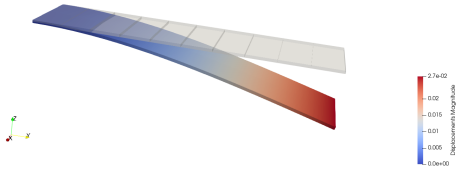
The analytically obtained values verify the results obtained with the same models using the software. EBBT is short for Euler-Bernoulli Beam Theory and TBT stands for Timeshenko Beam Theory. EBBT A. and TBT A. indicate the analytical solution versions for these theories.

When results obtained in Table 5.2 are analyzed, with the increasing degrees of freedom, the accuracy of the model is also increased. Especially TE4 and 2LE9 configurations provide the closest results to the real one, however, they are computationally more intensive. It can also be seen that changes in the model configurations significantly affect the stress results, however, displacement results seem to be affected much less. Also, the largest errors are observed for the Lagrange Expansion with 4 nodes in cross-section.

MUL2's output provides us with a ParaView file that visualizes the stress and displacement distributions. In the following Figures 5.4a and 5.4b said visualizations can be seen.



(a) TE4 model stress distribution for σ_{yy}



(b) TE4 model displacement distribution

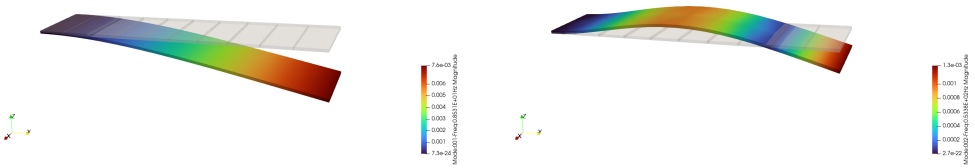
Figure 5.4: TE4 model stress and displacement distributions

To verify the results obtained with the MUL2, an additional static analysis on Femap was also conducted. A structural mesh with 8-node brick elements was selected for the solid model. QUAD type 2D mesh selected for the plate model. In Table 5.3 below, the comparison of the results can be seen.

Model	u_z [m]	σ_{yy} [MPa]
Femap (Solid)	-2.688E-02	34.657
Femap (Plate)	-2.689E-02	33.584
TE4	-2.675E-02	40.026
2LE9	-2.670E-02	40.078

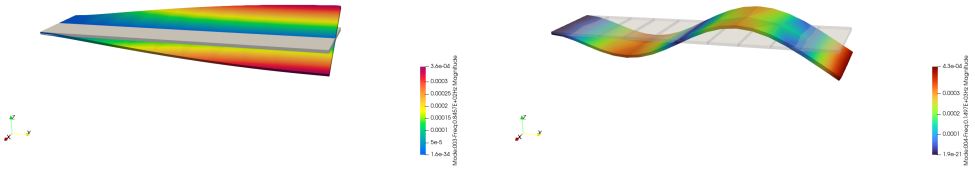
Table 5.3: Static analysis with Femap

5.1.2 Dynamic Analysis



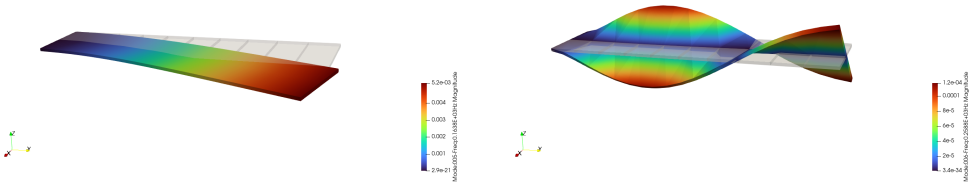
(a) Mode 1

(b) Mode 2



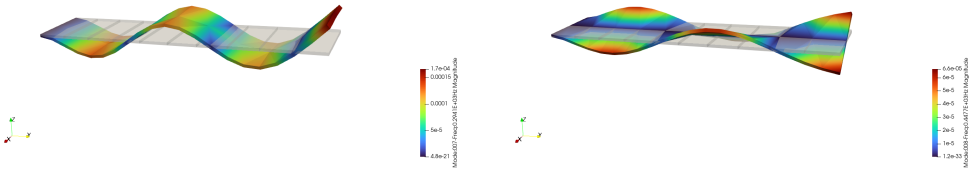
(c) Mode 3

(d) Mode 4



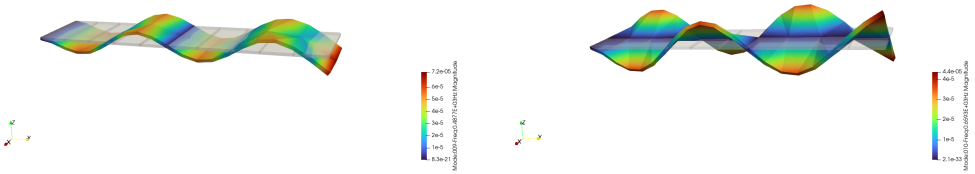
(e) Mode 5

(f) Mode 6



(g) Mode 7

(h) Mode 8



(i) Mode 9

(j) Mode 10

Figure 5.5: Vibration Modes for TE4 Model

The beam used in the static analysis is also considered here for the dynamic analysis. A free-vibration analysis was conducted to capture the proper modes and frequencies of the structure. As it was done for the static analysis, different model configurations are considered for free-vibration analysis and the obtained results can be found below Table 5.4

Mode	TBT	TE1	TE2	TE3	TE4	1LE4	1LE9	2LE9
1	8.399	8.399	8.697	8.544	8.531	9.737	8.543	8.541
2	52.615	52.615	54.266	53.448	53.378	60.992	53.448	53.435
3	147.250	147.250	85.245	84.819	84.574	85.245	84.830	84.660
4	163.532	163.532	153.067	149.891	149.674	170.662	149.906	149.865
5	288.425	288.425	164.373	163.825	163.784	171.548	164.371	163.850
6	476.759	476.761	261.973	259.534	258.799	261.973	259.649	259.040
7	712.676	712.679	301.094	294.691	294.119	334.188	294.743	294.642
8	894.913	806.182	456.544	448.944	447.656	456.544	449.339	448.010
9	997.108	894.913	498.703	489.002	487.731	552.185	489.137	488.918
10	1299.929	997.114	678.617	733.348	659.328	678.617	662.205	659.773
DOF	155	279	558	930	1395	372	837	1395

Table 5.4: Dynamic analysis with MUL2 frequency (Hz) results

Model configurations with a higher degree of freedom resulted in greater accuracy. In the figure below (figure 5.5), the mode shapes for the TE4 model can be observed.

As was done before for the static analysis, to further verify the results obtained from MUL2, a free-vibration analysis was performed with Femap for the same geometry. TE1 model configuration, TE4 model, and Femap model compared in the table below (Table 5.5)

Mode	TE1	Mode Type	TE4	Mode Type	Femap	Mode Type
1	8.399	I flexural x	8.531	I flexural x	8.505	I flexural x
2	52.615	II flexural x	53.378	II flexural x	53.217	II flexural x
3	147.250	III flexural x	84.574	I torsional	83.626	I torsional
4	163.532	I flexural z	149.674	III flexural x	149.189	III flexural x
5	288.425	IV flexural x	163.784	I flexural z	163.425	I flexural z
6	476.761	V flexural x	258.799	II torsional	255.778	II torsional
7	712.679	VI flexural x	294.119	IV flexural x	292.975	IV flexural x
8	806.182	II torsional	447.656	III torsional	442.049	III torsional
9	894.913	II flexural z	487.731	V flexural	485.178	V flexural
10	997.114	VII flexural x	659.328	IV torsional	650.327	IV torsional

Table 5.5: Comparison of frequencies (Hz) and modes between MUL2 models and Femap

The results from Femap verify that models with multiple degrees of freedom are far more accurate in both eigenfrequencies and vibration modes of the structure.

5.2 Aeroelastic Analysis of an Isotropic Plate

In this section, aeroelastic analyses on a simple isotropic plate will be conducted. The geometry used comes from an article by Petrolo [22]. A free-vibration analysis was conducted first to properly capture the first torsional and flexural modes of the structure. This analysis also helps to figure out whether the model at hand is suitable for aeroelastic analyses.

5.2.1 Dynamic Analysis

An isotropic plate with a thickness of $t = 0.001m$, length of $L = 0.305m$, and a chord length of $c = 0.076m$ is depicted in Figure 5.6 below. The structural model was discretized with 20 B4 elements (beams with 4 nodes). The total number of nodes turns out to be 61.

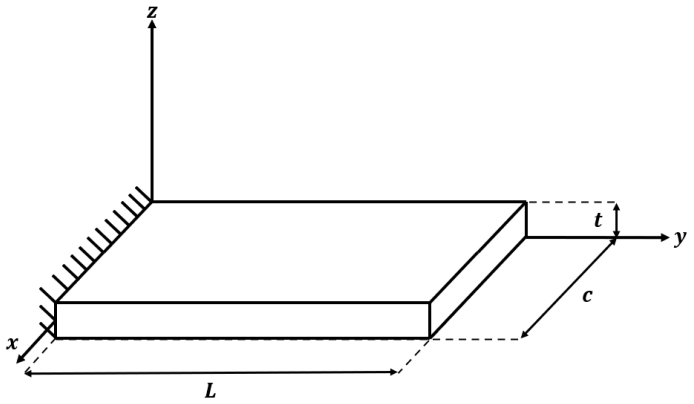


Figure 5.6: Geometry of the isotropic plate

The specifications of the material used in the isotropic plate are given in Table 5.6 below.

E [GPa]	G [GPa]	ρ [kg/m^3]
73.8	27.6	2768

Table 5.6: Material specifications

In Table 5.7, eigenfrequencies of the different model configurations and Femap can be observed. From the obtained results, it is clear that for the Taylor Expansion case, at least a third-order model configuration is needed to get accurate results for frequencies and proper modes.

In Table 5.8 below, a comparison of the different mode types for each model configuration can be seen. Also, Femap mode types are also included. For the Femap model, 4-node QUAD elements are used with a 40x10 structure. Clearly, the more refined meshes, the more convergent to accurate models obtained.

Modes	TE1	TE2	TE3	TE4	1LE9	2LE9	4LE9	Femap
1	8.966	9.395	9.137	9.131	9.135	9.135	9.123	9.138
2	56.190	58.792	57.136	57.110	57.131	57.129	57.068	57.104
3	157.324	74.389	73.865	73.847	73.869	73.850	73.823	71.896
4	308.272	164.928	160.430	160.350	160.416	160.411	160.225	160.311
5	509.575	231.312	228.360	228.189	228.395	228.167	228.029	222.048
6	654.085	323.363	315.794	315.578	315.746	315.734	315.239	315.421
7	761.230	411.150	402.065	401.399	402.173	401.281	400.892	390.265
8	1063.325	534.655	524.140	523.682	524.012	523.987	522.879	523.029
9	1416.029	656.957	605.017	603.434	605.244	603.181	602.353	586.135
10	1819.623	625.507	653.782	653.439	656.954	653.873	653.388	652.550
DOF	549	1098	1830	2745	1647	2745	4941	2706

Table 5.7: Frequency (Hz) values for first 10 modes of different models and Femap

Modes	TE4	Mode Type	Femap	Mode Type
1	9.131	I flexural x	9.138	I flexural x
2	57.110	II flexural x	57.104	II flexural x
3	73.847	I torsional	71.896	I torsional
4	160.350	III flexural x	160.311	III flexural x
5	228.189	II torsional	222.048	II torsional
6	315.578	IV flexural x	315.421	IV flexural x
7	401.399	III torsional	390.265	III torsional
8	523.682	V flexural	523.029	V flexural
9	603.434	IV torsional	586.135	IV torsional
10	653.439	I flexural z	652.550	I flexural z

Table 5.8: Frequencies and mode types for the TE4 model and Femap model

5.2.2 Flutter Analysis

After the dynamic analysis is conducted, flutter analysis can done. It is now necessary to highlight the reference air density $\rho_{air} = 1.225 kg/m^3$ and the chord of the aero-surface is $c = 0.076m$, same as the width of the plate. P_1 and P_2 points are positioned at the extremities of the leading edge. Aerodynamic paneling can be seen in Figure 5.7 below.

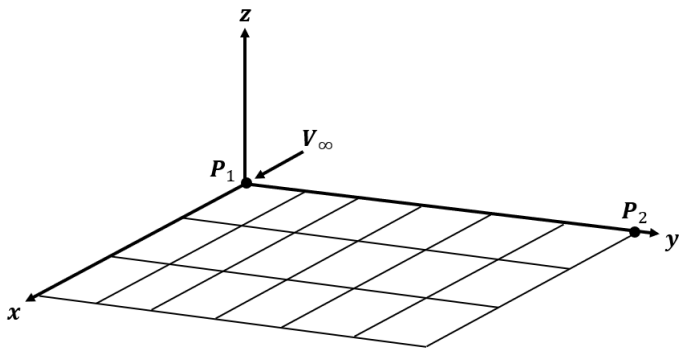


Figure 5.7: Aerodynamic Panel

A symmetry of the plane with respect to the xz plane is also considered. The number of aerodynamic panels to be used in the chord side and span side is also defined.

Figure 5.7 shows the representation of a 6x3 aerodynamic mesh. For this case; no effects of compressibility are considered, so the Mach number is kept equal to 0, and appropriately density is also kept constant and equal to the reference value. However, velocity varied in an interval of 2 m/s to 120 m/s with 40 velocity steps. The frequencies and damping values of the first 10 modes in this interval are evaluated.

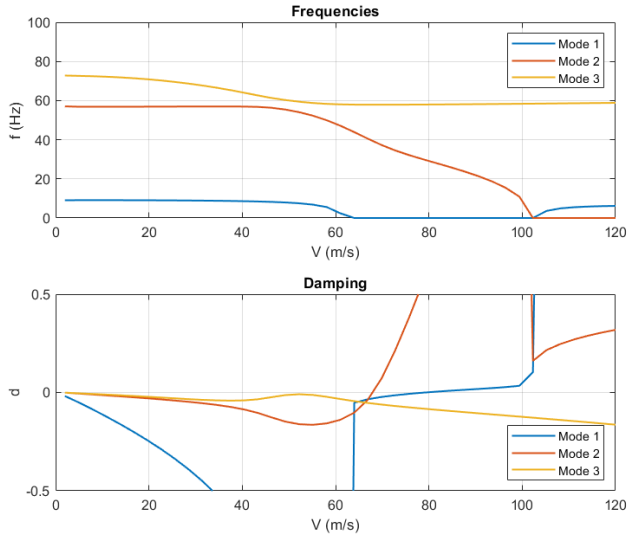


Figure 5.8: A TE4 model that uses a 30x8 aerodynamic mesh, and damping frequencies of the first three modes at different speeds

The frequencies and damping shown in the graphs above (5.8) correspond to the first 3 proper modes. The flutter condition is obtained when the negative damping values become positive, also with corresponding positive frequency values. It is evident from the graph above that the second mode is to first to go unstable. Once the range of velocities within the zero damping is identified, the flutter velocities can be obtained by interpolation. Table 5.9 below shows the flutter conditions for different models considering a 30x8 aerodynamic paneling. Lower-order models are not considered because they are not precise enough to detect the flutter phenomenon.

Model	Speed [m/s]	Flutter Frequency [Hz]	DOF
TE3	67.917	39.447	1830
TE4	67.908	39.434	2745
2LE9	67.914	39.443	2745
4LE9	67.888	39.415	4941

Table 5.9: Flutter conditions for different models with 30x8 aerodynamic mesh

It is also possible to evaluate the influence of different aerodynamic meshes. Table 5.10 below shows the velocity and flutter frequency for different aerodynamic mesh configurations for the 4LE9 structural model configuration. With finer meshes, more accurate results are obtained, converging values.

Aerodynamic Mesh	Speed [m/s]	Flutter Frequency [Hz]
6x3	63.012	43.979
10x4	65.427	41.673
15x4	66.436	40.930
30x8	67.888	39.415

Table 5.10: Flutter conditions for different aerodynamic meshes related to the 4LE9 model

To verify the results, flutter analysis with Femap is also needed. The same conditions apply for the Femap model as explained before and also for the solution method PKNL method is selected. This method is a more efficient version of the PK method. The structural mesh used is 40x10 once again. The frequency and damping plots extracted from Femap can be seen in Figure 5.9 below and the comparison of the flutter conditions found can be seen in Table 5.11 below.

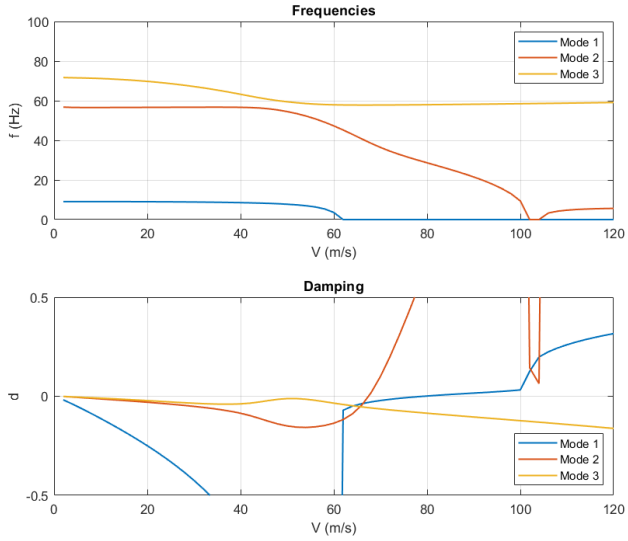


Figure 5.9: The frequencies and damping for the initial three modes on Femap with a 30x8 aerodynamic mesh at different speeds

Model	Speed [m/s]	Flutter Frequency [Hz]
TE4	67.908	39.434
Femap	67.419	39.142

Table 5.11: Comparison between the flutter conditions obtained with Femap and with the TE4 model considering a 30x8 aerodynamic mesh

Also, while keeping the structural mesh constant (40x10), different aerodynamic meshes were tested with Femap and the results can be seen in Table 5.12 below. The results obtained with the TE4 model and this case match satisfyingly.

Aerodynamic Mesh	Speed [m/s]	Flutter Frequency [Hz]
6x3	62.850	43.562
10x4	65.086	41.321
15x4	66.047	40.589
30x8	67.419	39.142

Table 5.12: Flutter conditions for different aerodynamic meshes relative to the Femap model considering a 40x10 structural mesh

Finally, to evaluate the effect the structural mesh has on flutter conditions, the aerodynamic mesh (30x8) was kept constant. With the meshes getting finer, results converged to the Taylor and Lagrange models shown previously.

Aerodynamic Mesh	Speed [m/s]	Flutter Frequency [Hz]
20x5	66.623	38.757
40x10	67.419	39.142
60x15	67.596	39.199
80x20	67.694	39.194

Table 5.13: Flutter conditions for different structural meshes related to the Femap model considering a 30x8 aerodynamic paneling

From the flutter analysis results, it is possible to obtain information about divergence. The divergence phenomenon occurs when the damping becomes positive and the corresponding frequency is zero. Table 5.14 below shows the divergence condition obtained from different model configurations and the model obtained from Femap. The divergence condition was found to be on the first mode, with larger values of velocity found for the flutter condition. ın this case, since the flutter speed is less than the divergence speed, the flutter speed dictates the limit.

Table 5.15 below shows the effect of different aerodynamic meshes on the divergence velocity values for different models: 4LE9 and Femap.

5.2.3 Static Aeroelastic Analysis

In this section, the goal is, by fixing the velocity, to find the vertical displacements of two points; one is on the leading edge, other is on the trailing edge. With these findings wing rotations at the tip can be found. The points are shown in Figure 5.10 below. The geometry is

Model	Speed [m/s]
TE3	79.223
TE4	79.195
2LE9	79.213
4LE9	79.188
Femap	79.219

Table 5.14: Divergence for different models and with Femap with 30x8 aerodynamic mesh

Aerodynamic Mesh	(4LE9) Speed [m/s]	Speed (Femap) [m/s]
6x3	78.211	77.641
10x4	78.943	78.529
15x4	79.524	79.220
30x8	79.188	79.219

Table 5.15: Divergence velocity for different aerodynamic meshes obtained from the 4LE9 model and the Femap model

the same as the previously used one in the flutter analysis and the coordinates for point A are $x = 0, y = 0.305, z = 0$ and for point B is $x = 0.076, y = 0.076, z = 0$.

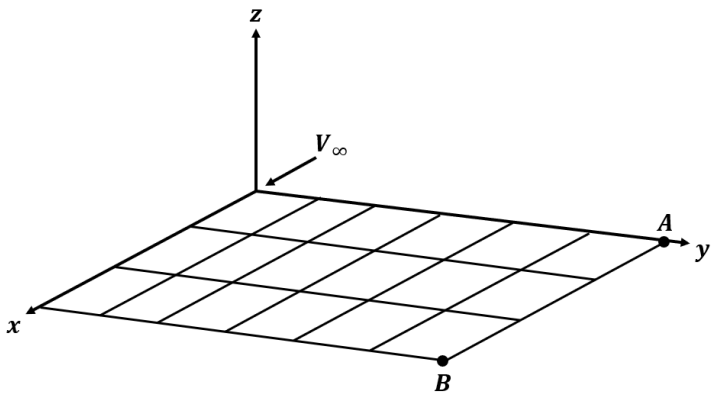


Figure 5.10: Points A and B where z-direction displacements are evaluated

Tables 5.16 and 5.17 show the displacements and rotations at these two points for different velocities for the TE4 and 1LE9 model configurations. It can be observed that with increasing speeds, the displacements and rotations are also increasing. When the velocity that causes the divergence is reached, beyond that velocity value displacement results, and therefore rotation results become negative and stop making sense in physical manner. From the results obtained from the two models, the divergence velocity can be found between 78 m/s and 80 m/s.

In the case of static aeroelasticity for the Femap model, the same model which used in the flutter analysis is used. 40x10 structural mesh applied with 30x8 aerodynamic paneling. 1 degree of angle of attack was applied as it was done with the other models. Table 5.18 below shows the displacements and calculated rotation values obtained from the Femap model.

In Figure 5.11 below, the speed vs. rotation of the wing tip is plotted. After the divergence point is reached, with increasing speeds rotation values become negative, as mentioned, physically not logical.

Model	u_{zA} [m]	u_{zB} [m]	Rotation [°]
10	7.349E-04	7.046E-04	0.023
30	7.808E-03	7.492E-03	0.239
50	3.246E-02	3.119E-02	0.957
70	1.896E-01	1.826E-01	5.263
78	1.818E+00	1.753E+00	40.591
80	-2.706E+00	-2.610E+00	-51.674
90	-2.631E-01	-2.542E-01	-6.698

Table 5.16: TE4 model displacements and rotations at points A and B at different speeds for $AoA = 1°$

Model	u_{zA} [m]	u_{zB} [m]	Rotation [°]
10	7.340E-04	7.037E-04	0.023
30	7.799E-03	7.483E-03	0.239
50	3.242E-02	3.115E-02	0.956
70	1.889E-01	1.820E-01	5.245
78	1.765E+00	1.702E+00	39.762
80	-2.831E+00	-2.730E+00	-52.915
90	-2.643E-01	-2.553E-01	-6.725

Table 5.17: 1LE9 model displacements and rotations at points A and B at different speeds for $AoA = 1°$

Model	u_{zA} [m]	u_{zB} [m]	Rotation [°]
10	7.441E-04	7.134E-04	0.023
30	7.910E-03	7.590E-03	0.242
50	3.318E-02	3.189E-02	0.971
70	1.978E-01	1.907E-01	5.385
78	2.067E+00	1.994E+00	43.512
80	-2.529E+00	-2.442E+00	-48.996
90	-2.749E-01	-2.659E-01	-6.744

Table 5.18: Femap model displacements and rotations at points A and B at different speeds for $AoA = 1°$

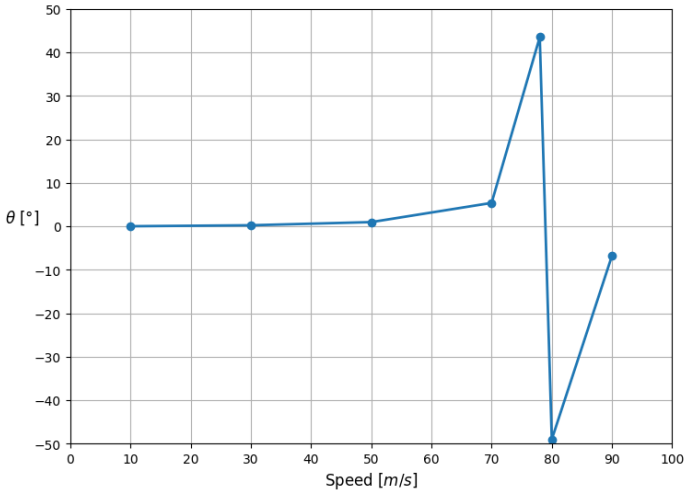


Figure 5.11: Speed vs tip rotations in Femap model

5.3 Aeroelastic Analysis of An Aircraft

In this section, aeroelastic analyses are performed on more complex configurations. A simple whole-aircraft model is considered which was also considered in a paper by Patil [21]. The dimensions of the aircraft are given in Table 5.19 below.

Parameter	Value [m]
Wingspan	32
Wing cord length	1
Tail-boom length	10
Tail span	5
Tail cord length	0.5

Table 5.19: Dimensions of the aircraft

Figure 5.12 shows the diagram of the aircraft.

The properties of the material used in the aircraft is given in the table below.

E [GPa]	ν	ρ [kg/m^3]
180	0.3	1800

Table 5.20: Material specifications for the aircraft

Firstly, the wing of the aircraft alone was studied. After dynamic and flutter analyses on the wing are done, the whole aircraft configuration will be studied.

5.3.1 Dynamic Analysis of the Wing

A dynamic analysis was conducted for the wing of the aircraft. The Wing and the tail of the aircraft are modeled as plates. And the cross-section of the wing with its structural discretization is given in the figure below.

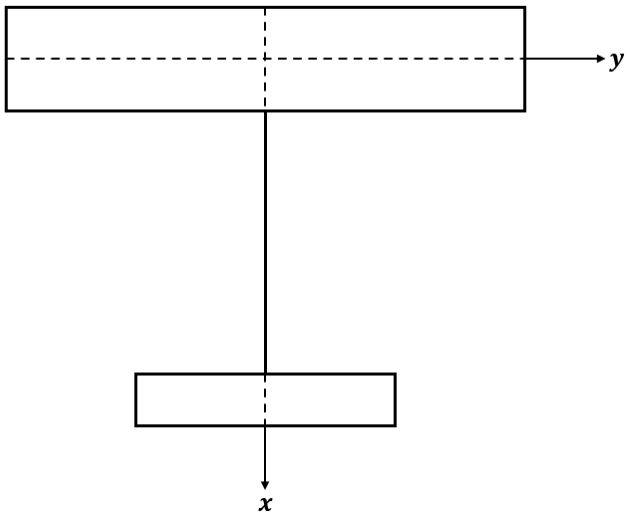


Figure 5.12: Representation of the aircraft

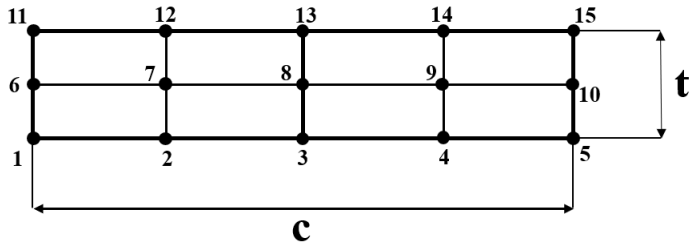


Figure 5.13: Cross-section of the wing

The dimensions of the wing cross-section are given in the table below.

Parameter	Value [m]
chord (c)	1
thickness (t)	0.025

Table 5.21: Dimensions of the wing cross-section

The first dynamic analysis was conducted with a clamped half-wing. The structural model consists of 7 B3 elements (3-node beam). Lagrange polynomial expansion is used in the beam cross-section 5.13. 4 Q9 elements are used to define the cross-section. The dynamic analysis resulted in the first 5 eigenfrequencies given in the table below.

Mode	Frequency [Hz]	Mode Type
1	0.159	I flexural around x
2	1.045	II flexural around x
3	3.116	III flexural around x
4	4.948	I torsional
5	6.314	I flexural around z
6	6.654	IV flexural around x

Table 5.22: First 6 modes of the half wing dynamic analysis and their mode types.

Another dynamic analysis was made with the whole unrestrained wing. The two half wings are symmetrical and are modeled as in the previous case. The first 6 eigenfrequencies are too small and correspond to the rigid body modes.

Mode	Frequency [Hz]	Mode Type
7	0.274	I flexural around x
8	0.708	II flexural around x
9	1.451	III flexural around x
10	2.412	IV flexural around x
11	3.580	V flexural around x
12	4.842	I torsional
13	5.385	VI flexural around x
14	6.789	VII flexural around x
15	9.705	II torsional
16	9.942	VIII flexural around x
17	10.197	I flexural around z

Table 5.23: Frequencies related to first 7-17 modes and their mode types of the whole wing, unconstrained

5.3.2 Flutter Analysis of the Wing

In the case of flutter analysis of the clamped wing, the same structural model from the dynamic analysis is used. The reference density, $\rho = 1.225 kg/m^3$. The symmetry in the xz plane

was activated, and 30x8 aerodynamic mesh panels were applied. To enable coupling between structural and aerodynamic models, the 50-node grid coordinates on the surface for which displacement is evaluated are also introduced.

Figure 5.14 shows the frequency and damping value changes for the first five modes in a velocity interval of [1 m/s, 300 m/s].

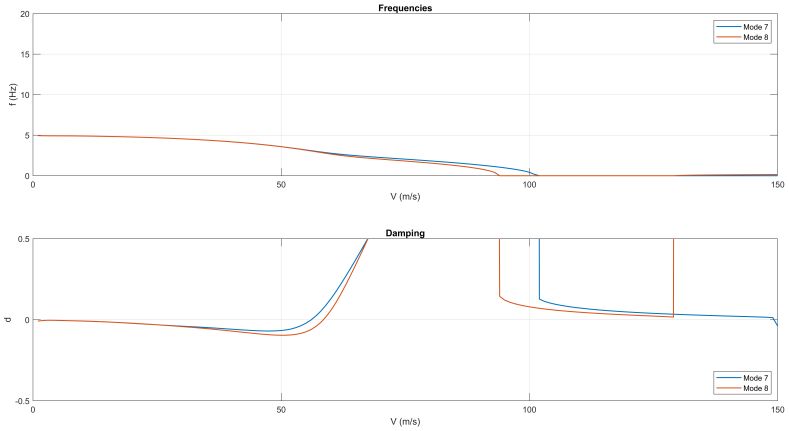


Figure 5.14: Model created on MUL2

The 7th mode is the first to go unstable, and the frequency and speed values are given in the table below.

Method	Flutter Frequency [Hz]	Flutter Speed [m/s]
MUL2	3.051	56.063

Table 5.24: Flutter frequencies and speeds found with MUL2 and Femap

5.3.3 Dynamic Analysis of the Whole Aircraft

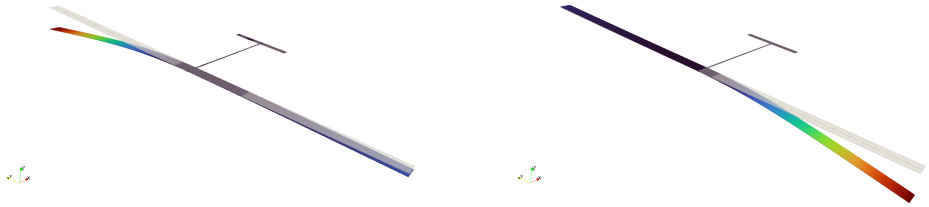
In this section, the whole aircraft is considered. The tail section which is also a plate, has a thickness of 0.01 m and a chord of 0.5 m. The fuselage is a 10 m beam connecting the wing and tail, has a thickness of 0.025 m and a width of 0.1 m. The structure of the tail consists of 7 B3 elements while the fuselage consists of 11 B3 elements. A Lagrange polynomial expansion is used for all three aircraft components, and the elements on the cross-section of each of them are Q9. To provide an effective connection between the different parts, it is a necessity to match all the common nodes on the sections coincident between wing and fuselage, and, between tail and fuselage.

A first dynamic analysis of the aircraft is carried out, introducing the constraint conditions. The $y = 0$ plane passing along the center of the wing, tail, and fuselage is constrained by the aircraft from all directions. The rest of the model is kept the same as before. The resulting first 18 frequencies are shown in Table 5.25 below.

Mode Number	Frequency [Hz]
1	0.176
2	0.176
3	1.040
4	1.040
5	2.702
6	2.744
7	2.892
8	2.892
9	4.943
10	4.943
11	5.956
12	5.956
13	6.439
14	6.439
15	10.743
16	10.743
17	14.895
18	14.895

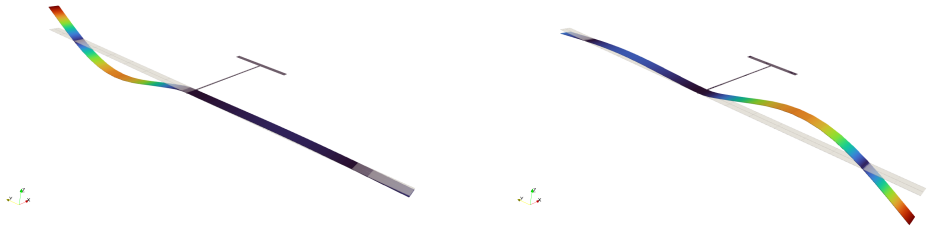
Table 5.25: The frequency values of the first 18 modes of the whole aircraft, $y = 0$ plane is constrained

The figures below show the mode shapes of the wing.



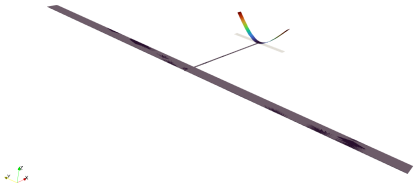
(a) Mode 1 (b) Mode 2

Figure 5.15: Vibration modes 1 and 2 for the whole aircraft, $y = 0$ plane constrained

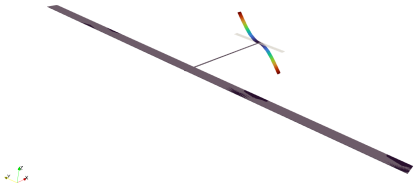


(a) Mode 3 (b) Mode 4

Vibration modes 3 and 4 for the whole aircraft, $y = 0$ plane constrained

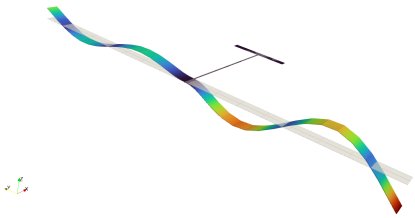


(a) Mode 5

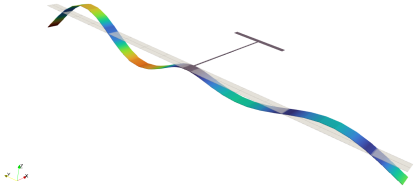


(b) Mode 6

Vibration modes 5 and 6 for the whole aircraft, $y = 0$ plane constrained

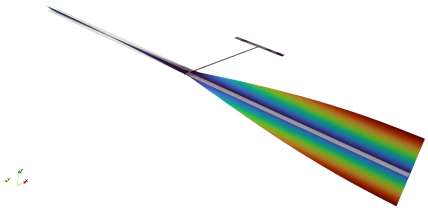


(a) Mode 7

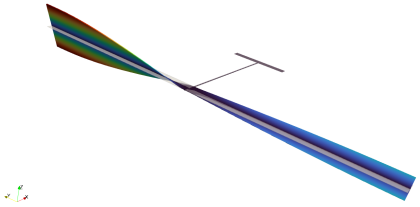


(b) Mode 8

Vibration modes 7 and 8 for the whole aircraft, $y = 0$ plane constrained

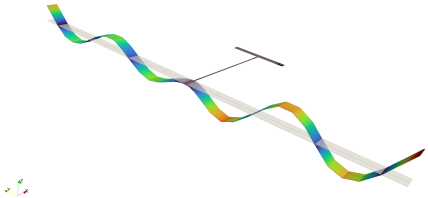


(a) Mode 9

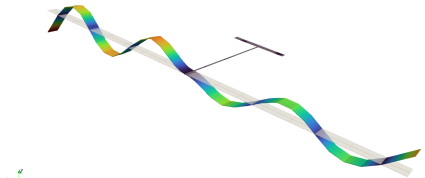


(b) Mode 10

Vibration modes 9 and 10 for the whole aircraft, $y = 0$ plane constrained

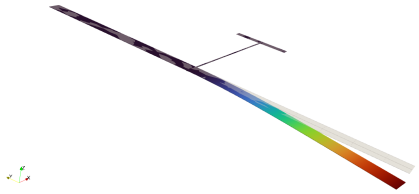


(a) Mode 11

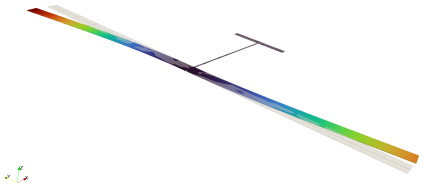


(b) Mode 12

Vibration modes 11 and 12 for the whole aircraft, $y = 0$ plane constrained

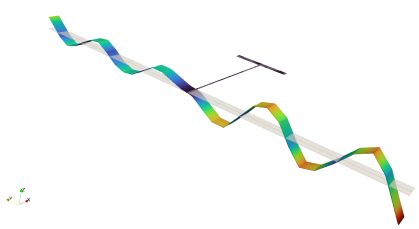


(a) Mode 13

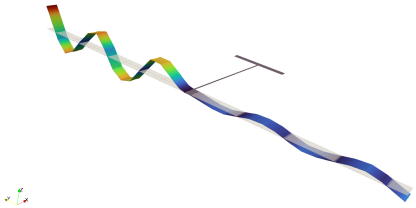


(b) Mode 14

Vibration modes 13 and 14 for the whole aircraft, $y = 0$ plane constrained

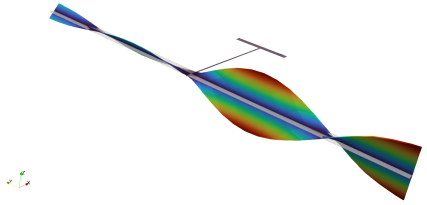


(a) Mode 15

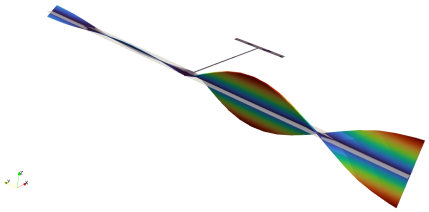


(b) Mode 16

Vibration modes 15 and 16 for the whole aircraft, $y = 0$ plane constrained



(a) Mode 17



(b) Mode 18

Vibration modes 17 and 18 for the whole aircraft, $y = 0$ plane constrained

Table 5.26 compares the frequencies obtained from the dynamic analysis of the $y = 0$ plane constrained wing alone and those for the whole aircraft, $y = 0$ plane is constrained.

Mode Type	Wing Frequency [Hz]	Aircraft Frequency [Hz]
I flexural around x	0.159	0.176
II flexural around x	1.045	1.040
III flexural around x	3.116	2.702
I torsional	4.948	4.943
I flexural around z	6.314	6.439

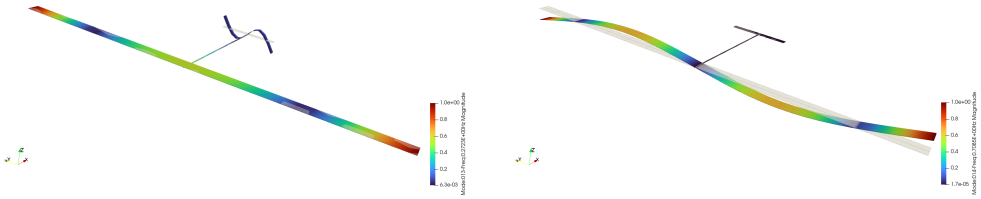
Table 5.26: Frequency comparison of the same modes belonging to $y = 0$ constrained wing configuration and $y = 0$ constrained aircraft

A second dynamic analysis on the aircraft is carried out by imposing no constraints on the structure. The resulting eigenfrequencies can be found in the table below.

Mode Number	Frequency [Hz]
13	0.272
14	0.708
15	1.358
16	1.503
17	2.412
18	3.555
19	3.884
20	4.842
21	5.179
22	5.220
23	5.385
24	6.782
25	9.592
26	9.780
27	9.942
28	11.070
29	11.626
30	11.903

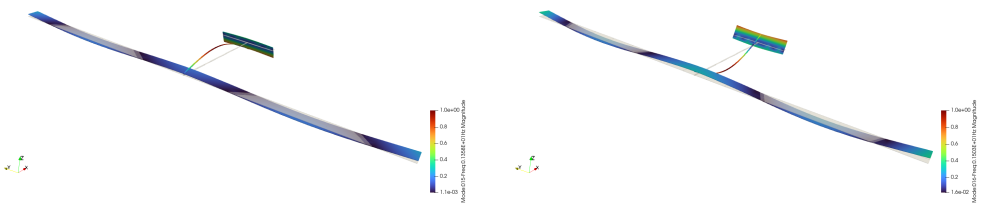
Table 5.27: The frequency values of the modes from 13 to 30 of the whole aircraft, unconstrained

The first 12 frequencies were too small and they are related to rigid body modes. While the other modes are shown in the figure below.



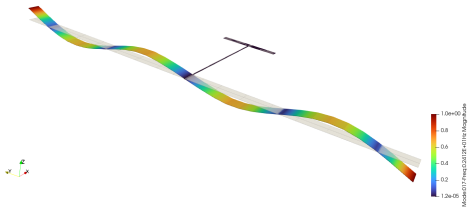
(a) Mode 13 (b) Mode 14

Figure 5.24: Vibration modes 13 and 14 for the whole aircraft, unconstrained

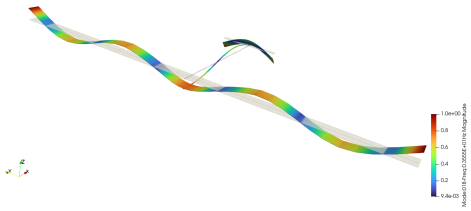


(a) Mode 15 (b) Mode 16

Vibration modes 15 and 16 for the whole aircraft, unconstrained

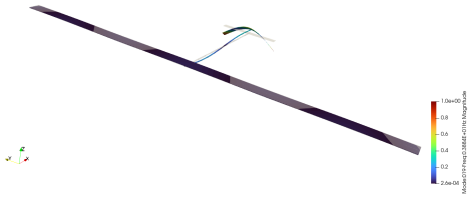


(a) Mode 17

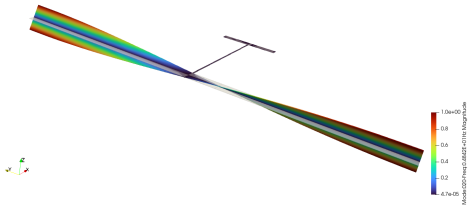


(b) Mode 18

Vibration modes 17 and 18 for the whole aircraft, unconstrained

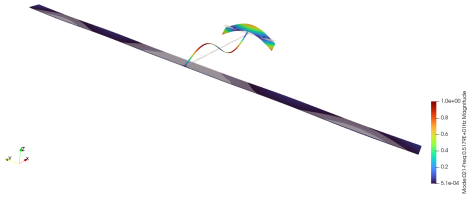


(a) Mode 19

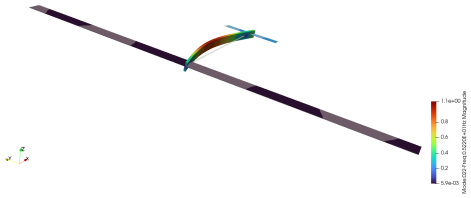


(b) Mode 20

Vibration modes 19 and 20 for the whole aircraft, unconstrained

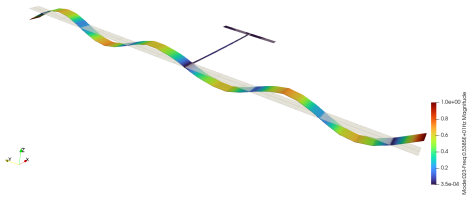


(a) Mode 21

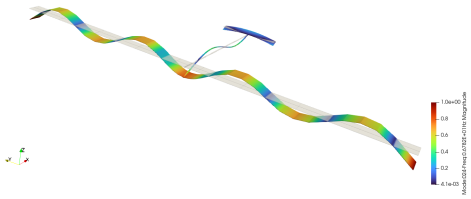


(b) Mode 22

Vibration modes 21 and 22 for the whole aircraft, unconstrained

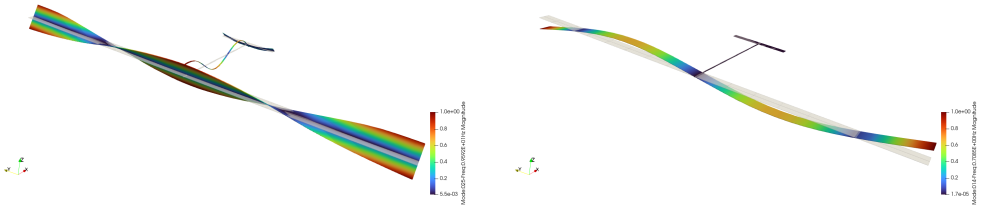


(a) Mode 23



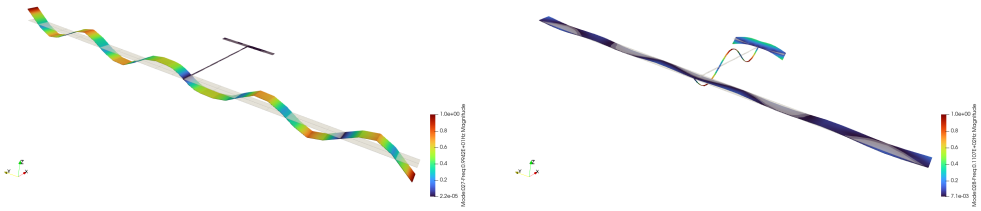
(b) Mode 24

Vibration modes 23 and 24 for the whole aircraft, unconstrained



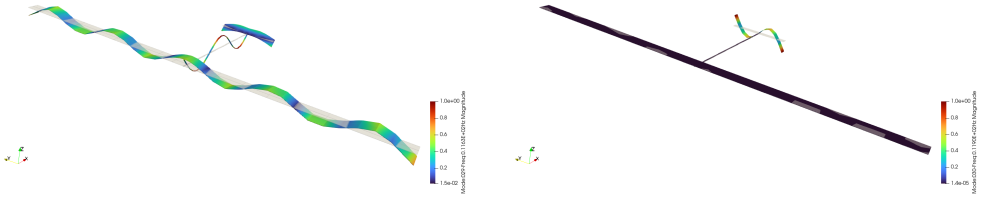
(a) Mode 25 (b) Mode 26

Vibration modes 25 and 26 for the whole aircraft, unconstrained



(a) Mode 27 (b) Mode 28

Vibration modes 27 and 28 for the whole aircraft, unconstrained



(a) Mode 29 (b) Mode 30

Vibration modes 29 and 30 for the whole aircraft, unconstrained

Table 5.28 compares the frequencies obtained from the dynamic analysis of the unrestrained wing alone and whole unrestrained aircraft below.

Mode Type	Wing Frequency [Hz]	Aircraft Frequency [Hz]
I flexural around x	0.274	0.272
II flexural around x	0.708	0.708
III flexural around x	1.451	1.358
IV flexural around x	2.412	2.412
V flexural around x	3.580	3.555
I torsional	4.842	4.842

Table 5.28: Frequency comparison of the same modes belonging to unconstrained wing configuration and unconstrained aircraft configuration

Finally, a dynamic analysis is performed on the same aircraft considered previously but changing the boundary conditions: the z-direction is free in $y = 0$ plane. In Table 5.29 below eigenfrequencies of the modes are given.

Mode Number	Frequency [Hz]
2	0.176
3	0.265
4	1.040
5	1.371
6	2.727
7	2.744
8	2.892
9	3.526
10	4.933
11	4.943
12	5.956
13	6.567
14	6.439
15	6.439
16	10.743
17	11.291
18	14.896

Table 5.29: The frequency values of the modes from 2 to 18 of the whole aircraft, $y = 0$ constrained, expect z-direction is free.

The mode shapes of the $y = 0$ constrained expect z-direction free condition are given in the figure below.

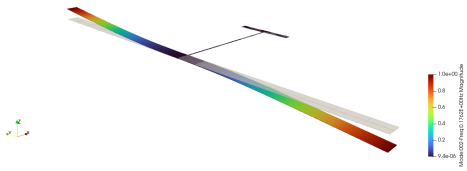
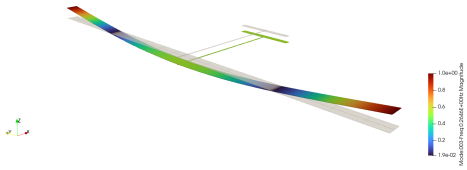
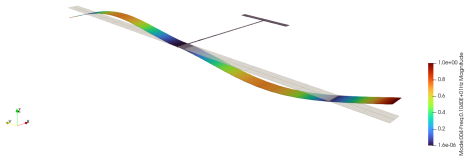


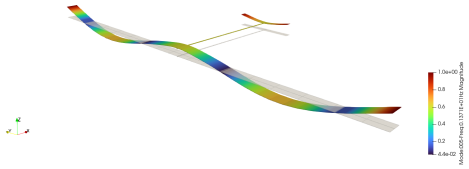
Figure 5.33: Vibration mode 2 for whole aircraft, $y = 0$ plane constrained, except z-direction is free



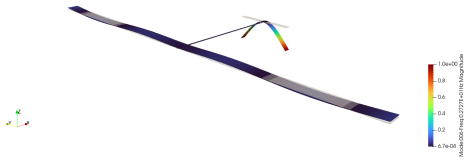
Vibration mode 3 for whole aircraft, $y = 0$ plane constrained, except z-direction is free



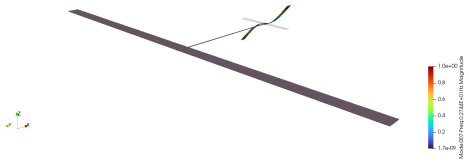
Vibration mode 4 for whole aircraft, $y = 0$ plane constrained, except z-direction is free



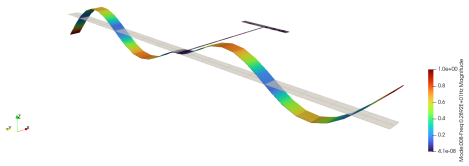
Vibration mode 5 for whole aircraft, $y = 0$ plane constrained, except z-direction is free



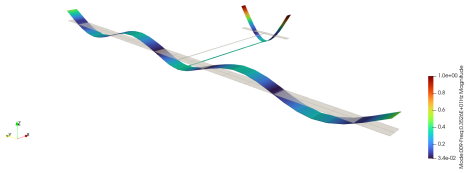
Vibration mode 6 for whole aircraft, $y = 0$ plane constrained, except z-direction is free



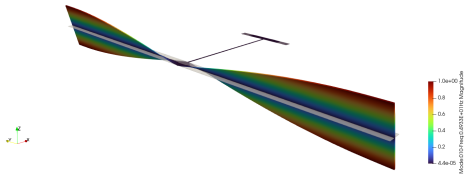
Vibration mode 7 for whole aircraft, $y = 0$ plane constrained, except z-direction is free



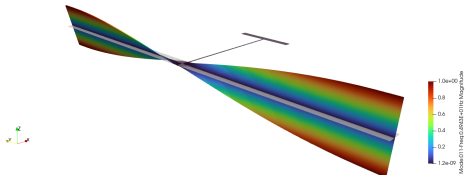
Vibration mode 8 for whole aircraft, $y = 0$ plane constrained, except z-direction is free



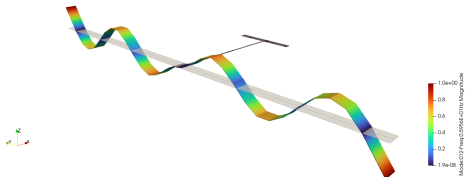
Vibration mode 9 for whole aircraft, $y = 0$ plane constrained, except z-direction is free



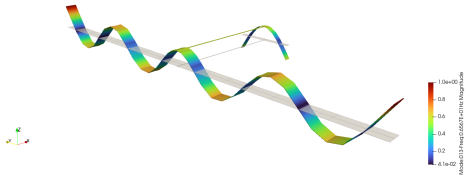
Vibration mode 10 for whole aircraft, $y = 0$ plane constrained, except z-direction is free



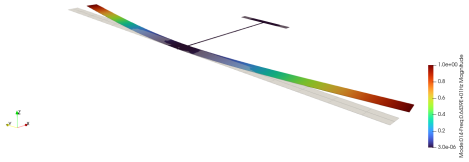
Vibration mode 11 for whole aircraft, $y = 0$ plane constrained, except z-direction is free



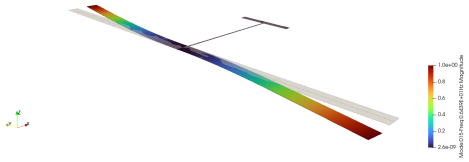
Vibration mode 12 for whole aircraft, $y = 0$ plane constrained, except z-direction is free



Vibration mode 13 for whole aircraft, $y = 0$ plane constrained, except z-direction is free



Vibration mode 14 for whole aircraft, $y = 0$ plane constrained, except z-direction is free



Vibration mode 15 for whole aircraft, $y = 0$ plane constrained, except z-direction is free

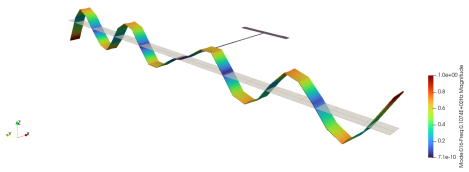
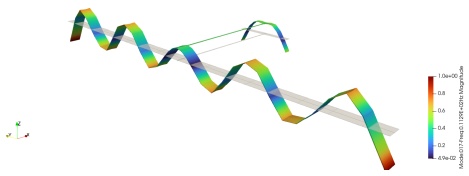
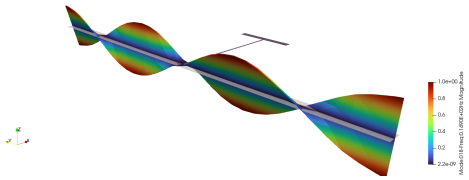


Figure 5.34: Vibration mode 16 for whole aircraft, $y = 0$ plane constrained, except z-direction is free



Vibration mode 17 for whole aircraft, $y = 0$ plane constrained, except z-direction is free



Vibration mode 18 for whole aircraft, $y = 0$ plane constrained, except z-direction is free

5.3.4 Aircraft Flutter Analysis

The first flutter analysis is carried out for the whole aircraft with $y = 0$ plane constrained. For this two aerodynamic surfaces are created by entering the coordinates of the two extreme leading edge points of the wing and tail with their chord. The panels that discretize the two surfaces are 60x8 aerodynamic mesh for the wing and 30x4 aerodynamic mesh for the tail. Two sets of points are used for the spline, for the wing 100 grid points are designated for the wing spline, and for the tail 50 grid points are designated for the tail spline.

In the figure below, the whole aircraft flutter result with $y = 0$ plane is constrained is given.

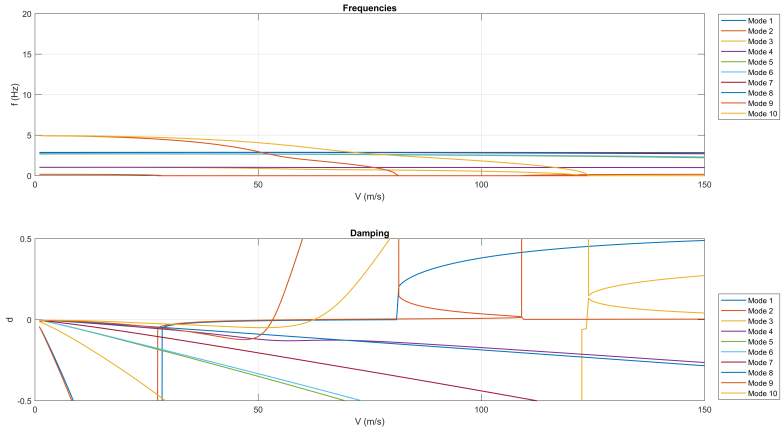


Figure 5.35: Frequency and damping change with respect to the velocity of the aircraft model created on MUL2, $y = 0$ constrained

In the table below, flutter conditions are given both for the model created in MUL2 and the model created on Femap.

Platform	Flutter Frequency [Hz]	Flutter Velocity [m/s]
MUL2	2.578	53.250
Femap	3.084	55.185

Table 5.30: Flutter frequency and velocity obtained from models created on MUL2 and Femap

A further flutter analysis is carried out on the same aircraft with no restrictions on any plane. In the figure below, frequency and damping plots are given.

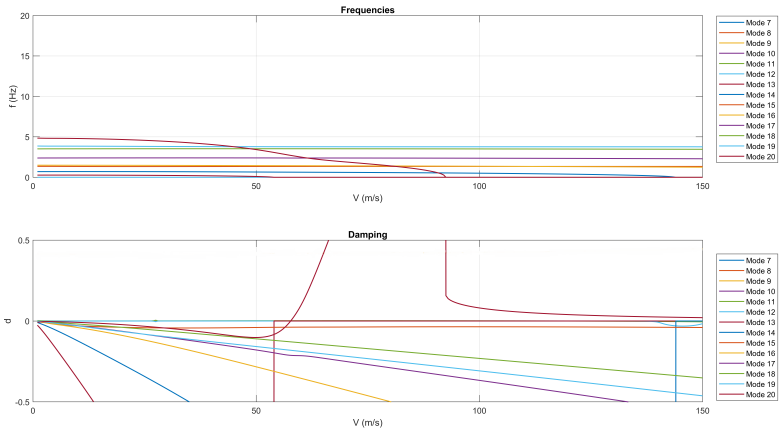


Figure 5.36: Frequency and damping change with respect to the velocity of the aircraft model created on MUL2, unconstrained

In the table below flutter conditions found for the unconstrained aircraft flutter analysis are given.

Platform	Flutter Frequency [Hz]	Flutter Velocity [m/s]
MUL2	2.693	57.723

Table 5.31: Flutter frequency and velocity obtained from the unconstrained model created on MUL2

In the figure below the mode shape for the unconstrained aircraft is shown.

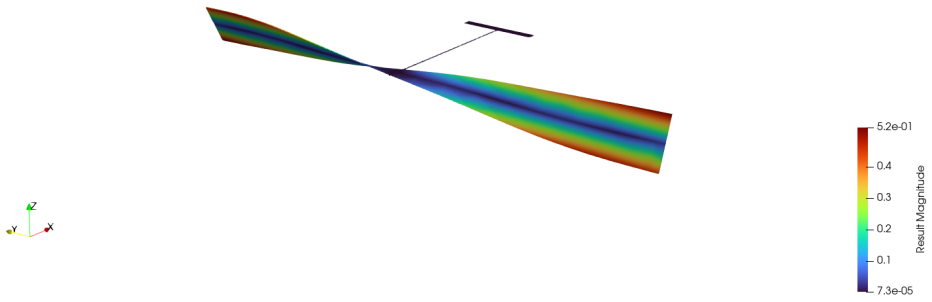


Figure 5.37: Flutter mode shape for the unconstrained aircraft

Finally, in the figure below, the result of the boundary condition of $y = 0$ plane is constrained in all directions except the z-direction configuration for the whole aircraft is given with frequency and damping plots.

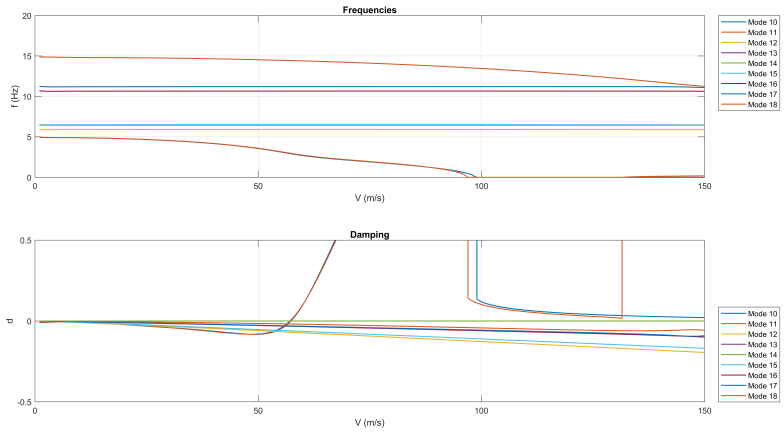


Figure 5.38: Frequency and damping change with respect to the velocity of the aircraft model created on MUL2, $y = 0$ constrained except z is free

In the table below, detected flutter conditions for the final case are given.

Platform	Flutter Frequency [Hz]	Flutter Velocity [m/s]
MUL2	2.925	57.308

Table 5.32: Flutter frequency and velocity obtained from the $y = 0$ constrained except z model created on MUL2

Chapter 6

Data Generation

In the previous sections, we established a base to study flutter analysis with whole aircraft before generating data for the machine learning process. In this section, we are going to proceed with the data generation. To achieve that, automation of MUL2 with NASTRAN was needed because we are planning to generate 7000 different aircraft configurations with varying wing spans, wing wall thicknesses, materials, and engine locations.

6.1 Base Aircraft

Since we are working with slender beams, the base aircraft model was selected Britten Norman BN-2 Islander. This aircraft's wings are rectangular and have straight leading and trailing edges, and its aspect ratio makes the wing slender, similar to the aircraft model we used in the previous section. The image of the base aircraft can be seen in the figure below.

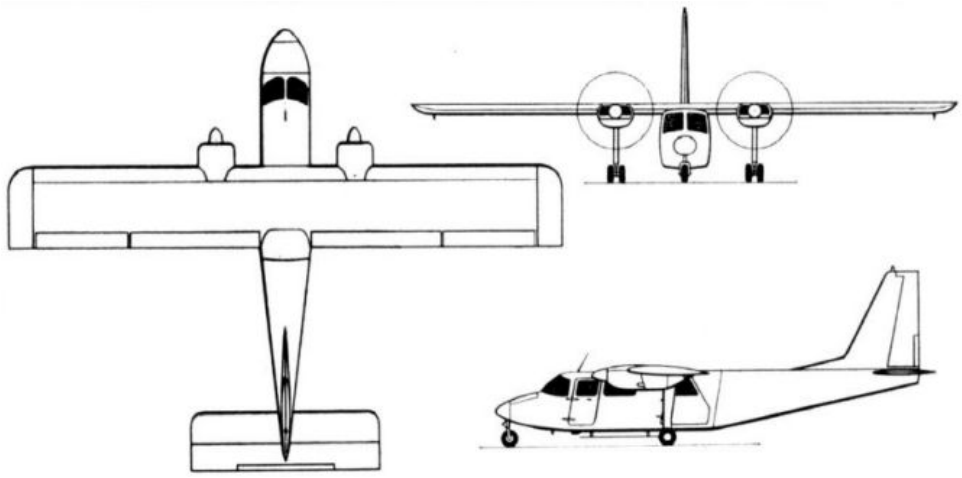


Figure 6.1: Britten-Norman BN2, Jane's All the World Aircraft 1966/02

BN-2's dimensions can be found in the table below.

Wing		Tail	
Span	15 m	Span	4.70 m
Chord	2 m	Chord	1.4 m
Area	30 m^2	Area	6.61 m^2

Table 6.1: BN-2A Islander's dimensions

BN-2 Islander's fuselage length is 8 m. It has two Lycoming O-540-A3D5 piston engines with a diameter of 0.84 m.

6.2 Modelling the Base Aircraft

The base model we created for the base aircraft, has 40 B3 elements throughout the wing span, 11 B3 elements for the fuselage, and 7 B3 elements for the tail. All of these elements Lagrange Expansion have been used with 2LE9 configuration with Q9 elements. All the dimensions in the table 6.1 are used. However, the fuselage width is set to 0.5 m and the fuselage height is set to 0.1 m.

In the figure below, the cross-section shape of the wing and tail can be seen.

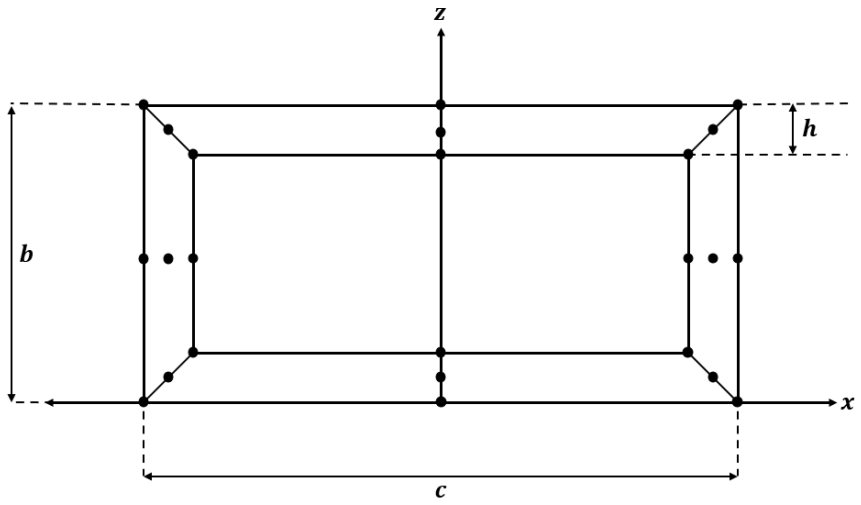


Figure 6.2: Hollow cross-section used for wing and tail

The base model of the wing has a wall thickness of 0.001 m and the tail has a wall thickness of 0.003 m. Also, starting from the wing roots, for every second element, a filled element is used instead of a hollow structure to increase the stiffness of the structure.

Engines are placed on the wings via engine mounts to achieve compatibility. Each engine's mass is equal to 0.1 mass of the half-wing of the aircraft as Libo Wang et al. did in their 2012 study [27].

In the figure below, the 3-D model of the aircraft is illustrated on ParaView.

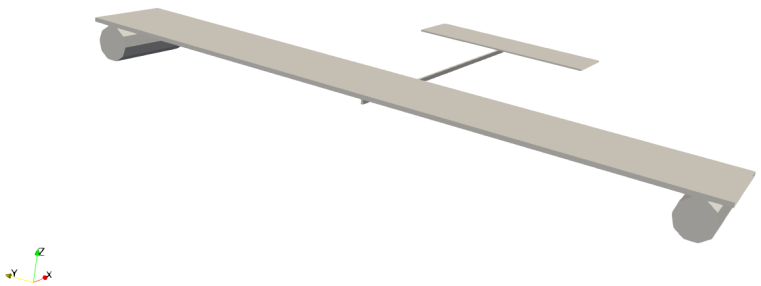


Figure 6.3: 3D model created on MUL2

6.3 Creation of Different Configurations

We can continue with the creation of different configurations. We designated 7 different materials, the interval for the wing spans, the interval for the wall thicknesses, and interval for the engine locations.

The selected materials for the different configurations with their specifications can be found below.

Material	E $[GPa]$	ν	$\rho[kg/m^3]$
2024-T3	73.1	0.33	2780
7075-T6	71.7	0.33	2810
Ti-6Al-4V	113.8	0.34	4430
T-300	140	0.3	1800
6061-T6	68.9	0.33	2700
5052-H32	70.3	0.33	2680
3003-H14	68.9	0.33	2730

Table 6.2: Material used for different aircraft configurations

The minimum wing span selected was 15 m, while the maximum span is 24 m, with 1 m of steps. For the wing wall thicknesses, the minimum thickness was selected as 0.001 m while the maximum thickness can be used in the wing is 0.003 m. For this interval, 0.0005 m of steps were used.

For the case of engine locations, since 40 B3 elements were used for the modeling of the wing, therefore each half-wing has 20 elements. The engine can be placed in the middle node of each of these elements. Therefore the number of engine placement options is 20.

In the figure below a different engine placement can be seen as opposed to Figure 6.3.

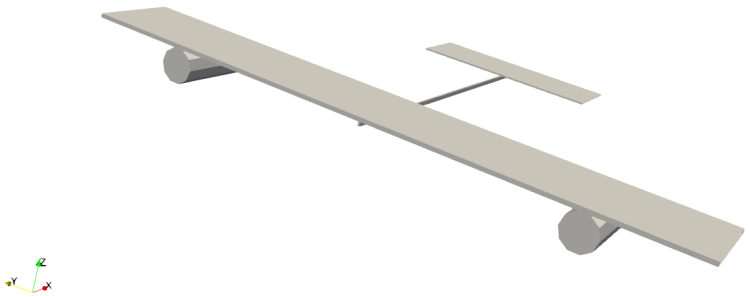


Figure 6.4: The base aircraft model with different engine location

Note that changing wing spans with constant element numbers creates an incompatibility if node locations are not adjusted for the new configuration.

6.4 Generation Step

With all the parameters decided, the automation of the generation process is prepared in a Python environment using various different libraries, such as Numpy for a healthy and error-free automation process.

With 7 different materials, 10 different wing spans, 5 different wall thicknesses, and 20 different engine locations, in total, we generated 7000 different aircraft configurations with MUL2 and made 7000 NASTRAN flutter analyses (one just after another). The flutter results consist of 151 rows for the 151 different different speed values and 20 columns for 20 different modes. Both frequency and damping cases are recorded in this way for each configuration of the aircraft.

Chapter 7

Machine Learning Model

In this section, we are going to create a machine-learning model that predicts an aircraft's flutter velocity. In the previous section we generated raw data for 7000 different aircraft configurations and we used them to make 7000 different flutter analyses. In this section, we are going to preprocess the data and will try to find a suitable model for our purposes using Tensorflow and Keras. For the data preprocessing we are going to utilize Pandas library.

7.1 Data Preprocessing

Since we generated raw data in the previous section, we need to do some preprocessing on it. The raw data consists of 4x7000 input values: material, wing span, wall thickness, and engine location. The output data consists of 1057000x21 for frequency and damping outputs.

As a first thing, we extracted the different speed values from the frequency values that give us the speed range for the flutter. After this step, we dropped the speed columns from both frequency and damping dataframes.

After examining the different mode frequencies and mode shapes, we found out that the first 9 modes are the rigid body modes. As a result, we dropped the first 9 modes from the frequency and damping dataframes.

We divided the main large frequency and damping dataframes to into smaller ones and put them into a list. Each element in these lists corresponds to an output of a single input.

A random configuration's damping-speed and frequency-speed plots can be seen in the figures below.

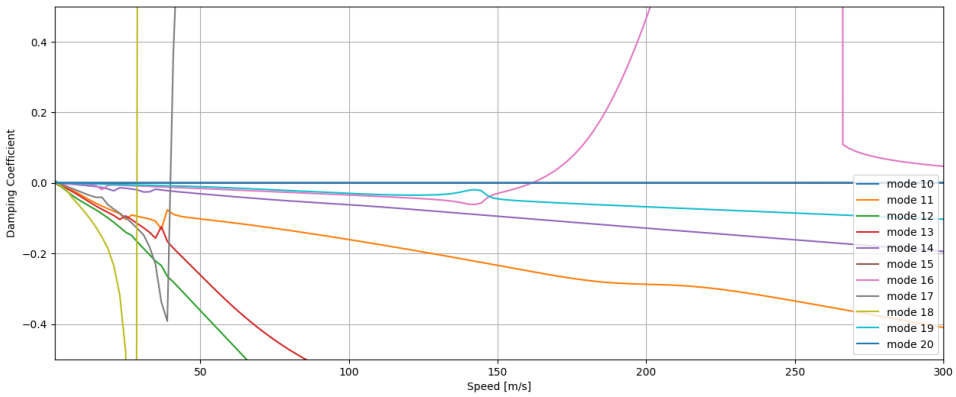


Figure 7.1: Damping-Speed plot for a random aircraft configuration

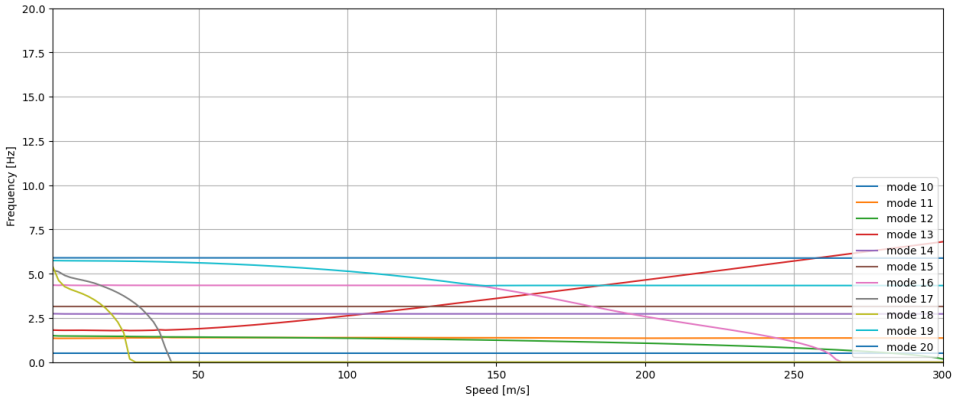


Figure 7.2: Frequency-Speed plot for a random aircraft configuration

To detect the flutter in each case and find the flutter speed we need more preprocessing. So in each input's output damping results, we searched for a negative to positive transition, and when this transition happens the frequency value must be different than zero. Otherwise, we would be detecting the divergence. Then we extract the smallest velocity value that satisfies our condition to label it as flutter speed. So now, for each input, we found a flutter speed, so we can train our model to detect the flutter speed according to the inputs we provide.

In the figure below, we can see a scatter plot that shows the flutter speed distribution for our 7000 different configurations.

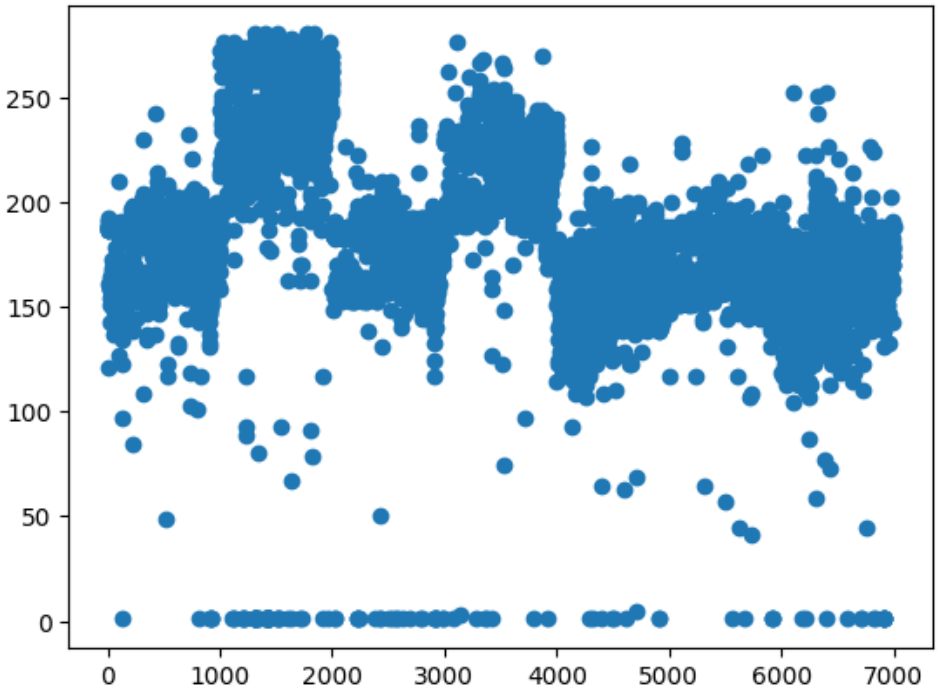


Figure 7.3: Scatter plot for flutter speed distribution for the non-mixed order aircraft configurations

The maximum flutter speed found is 280.1 m/s and the minimum one is 1 m/s which indicates no flutter detected. The total number of no-flutter cases is 102, and we are dropping them for both inputs and outputs, to help increase the model's accuracy. A classification model can be generated with a much more broad input space to detect flutter, and non-flutter conditions first. But for our case, we have limited data, so we are proceeding without it.

Before proceeding with the construction of the neural network, there are a few other preprocessing steps we need to follow. First of all, we need to encode our materials, so we can feed them into the network. However, if we use integer encoding, the model may think some of the materials are dominant than others. Therefore we are going to use one-hot encoding.

One-hot encoding is a process used to convert categorical data into a numerical format that can be fed into machine learning algorithms. Since many algorithms cannot directly handle categorical data, one-hot encoding transforms each categorical value into a new binary column and assigns a 1 or 0 (one-hot) to indicate the presence or absence of the feature.

In the table below we can see the first five rows of the material inputs encoded with one-hot encoding. The rows shows 2024-T3, 3003-H14, 5052-H32, 6061-T6, 7075-T6 in this order.

2024-T3	**3003-H14**	**5052-H32**	**6061-T6**	**7075-T6**	**T-300**	**Ti-6Al-4V**
1	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	1	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	1	0	0

Table 7.1: One-hot encoded material types

Engine location data shows the distance from the middle axis of the airplane. To feed them into the neural network, we divided each location into its corresponding wing span.

As last, we applied maximum-minimum normalization to our input data, and output data (flutter speeds). The formula used for the maximum-minimum normalization can be found below.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{7.1}$$

Also, we randomize the order of the inputs and then the outputs according to the inputs, since we create the data and it has a structured order.

7.2 Model Development and Training Process

We divided our data into a training set, a test set, and a validation set. We used a sequential model with Dense layers. We did not need to adopt any kind of regularization.

In the figure below, the model architecture data can be seen.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 64)                704

 dense_1 (Dense)             (None, 32)                2080

 dense_2 (Dense)             (None, 16)                528

 dense_3 (Dense)             (None, 8)                 136

 dense_4 (Dense)             (None, 1)                 9

=================================================================
Total params: 3457 (13.50 KB)
Trainable params: 3457 (13.50 KB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 7.4: Model summary

In the model, we utilized an Adam optimizer with a learning rate of 0.005. As a loss function, since we are doing a regression task, at first we tried a mean squared error and mean absolute error. However, with mean absolute error as a loss function, we get similar but more accurate results. We used 150 epochs with a batch size of 8.

7.3 Performance Metrics Explained

Before delving into the results, first, it is important to explain the metrics that are used.

Mean Squared Error

Mean Squared Error (MSE) is a widely used metric for evaluating the performance of a regression model. It measures the average of the squares of the errors, which are the differences between predicted values and actual values. MSE provides a way to quantify the difference between the values a model predicts and the values observed in the real world.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{7.2}$$

Where:

- n is the number of data points.
- y_i is the actual value of the data point.
- $\hat{y}_i$ is the model's predicted value.
- The term $(y_i - \hat{y}_i)^2$ is the squared difference between the actual and predicted values, also known as the squared error.

The MSE is always non-negative, and a value of 0 indicates that the model predicts the data perfectly. In practice, a lower MSE is preferable, indicating that the model's predictions are closer to the actual values.

Mean Absolute Error

Mean Absolute Error (MAE) is another common metric for assessing the performance of a regression model. Unlike the Mean Squared Error, MAE measures the average of the absolute errors, which are the absolute differences between the predicted and the actual values. It gives a linear score, which means all individual differences are weighted equally in the average.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{7.3}$$

Where:

- n is the total number of observations in the dataset.
- y_i is the actual value for the i-th observation.
- $\hat{y}_i$ is the predicted value for the i-th observation.
- The term $|y_i - \hat{y}_i|$ represents the absolute error between the actual and the predicted values for each observation.

The MAE is always non-negative, and an MAE of 0 indicates perfect predictions with no error. Because the MAE uses the absolute value of the residuals, it does not penalize large errors as heavily as the MSE does.

R2 Score

The R-squared score, or R^2, is a statistical measure used in the context of statistical models whose main purpose is either the prediction of future outcomes or the testing of hypotheses, on the basis of other related information. It provides a measure of how well-observed outcomes are replicated by the model, based on the proportion of total variation of outcomes explained by the model.

The formula to calculate R^2 is given by:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{7.4}$$

Where:

- n is the number of observations.
- y_i is the actual observed outcome.
- $\hat{y}_i$ is the predicted value, obtained from the model.
- $\bar{y}$ is the mean of the actual observed outcomes.
- The numerator, $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$, is the sum of the squared differences between the predicted values and the actual values, which is known as the "sum of squared residuals".
- The denominator, $\sum_{i=1}^{n}(y_i - \bar{y})^2$, is the "total sum of squares", which is proportional to the variance of the data.

A higher R^2 value indicates a higher proportion of variance accounted for by the model, with a value of 1 representing a perfect fit. It is important to note that a high R^2 does not necessarily indicate that the model has a good fit. It might just mean that the model is complex enough to fit the training data. Therefore, R^2 should not be used as the sole measure to evaluate a model's performance.

7.4 Results

In Figure 7.5 below, we can see the scatter plot with actual values in the x-axis, and predicted values in the y-axis for the model with mean absolute error has used as loss function. An ideal line is also drawn.

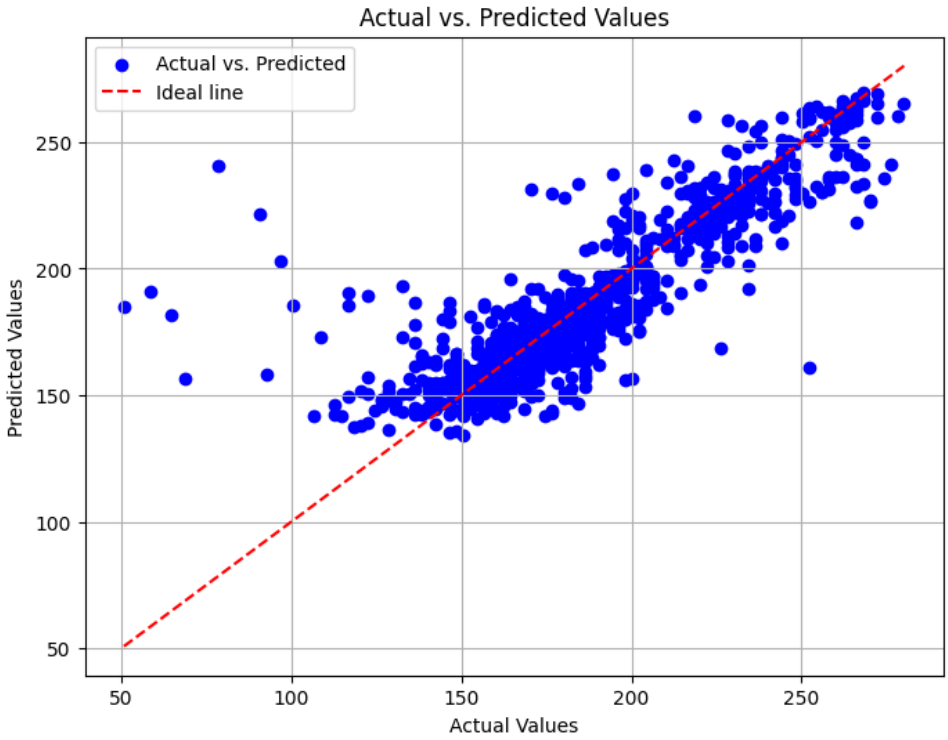


Figure 7.5: Actual vs predicted plot for MAE loss used model

In the table below we can see the performance metrics for the MAE loss model.

MSE	MAE	R^2 Score
248.109	8.718	0.782

Table 7.2: Performance metrics for the model with MAE as a loss function

In the figure below, we can see the actual versus predicted values scatter plot for the model trained with mean square error loss.

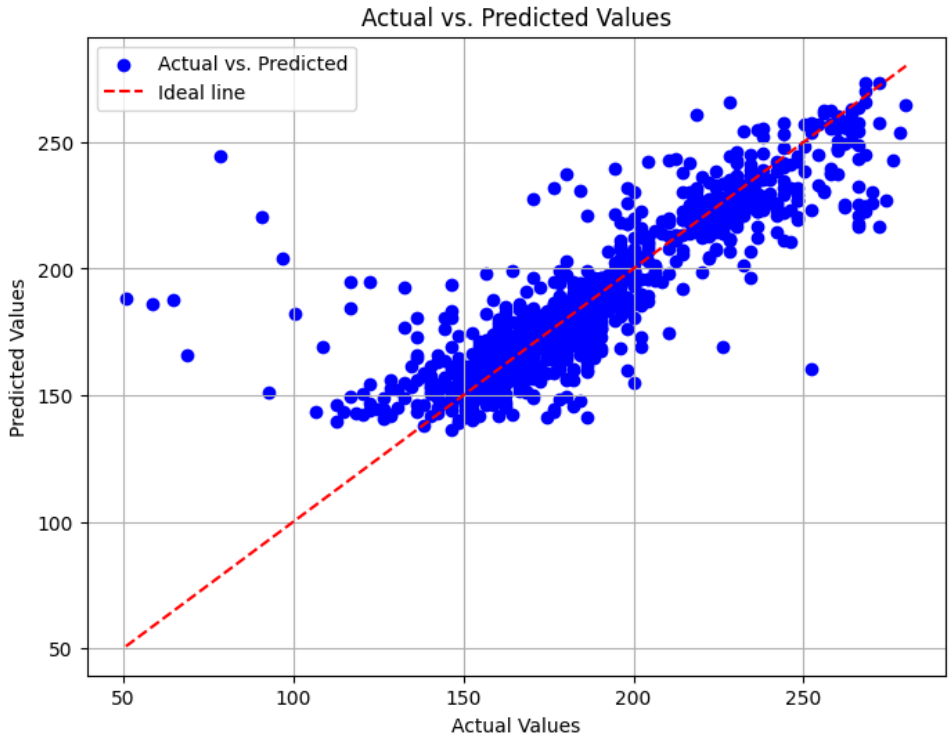


Figure 7.6: Actual vs predicted plot for MSE loss used model

In the table below we can see the performance metrics for the MSE loss model.

MSE	MAE	R^2 Score
267.592	9.414	0.764

Table 7.3: Performance metrics for the model with MSE as a loss function

From the performance metrics and actual vs. predicted values plots, the model trained with mean absolute error as the loss function performs better in terms of all the performance metrics used. It has lower absolute errors, with less mean squared error obtained, we can say that while the model is optimized to minimize absolute errors directly, it also minimizes the squared errors effectively, although it has a lesser degree. The lower MSE indicates fewer large errors. The higher R^2 score indicates a better fit of the model to the data, within the context of variance of data. According to this, the model trained with MAE seems to generalize better.

Chapter 8

Conclusion

The purpose of this thesis is to demonstrate the ability of machine learning with flutter analysis: prediction of the flutter speeds. The aeroelastic analyses are costly and computationally expensive, especially when there is a design or optimization problem at hand. Because a large amount of data has to be produced but not to be used again. The implementation of machine learning to these problems can help to reduce the number of runs that will be made and also for the data generation, now we have more efficient tools like MUL2, data-wise, creation of large databases and creating machine learning models with them for different purposes is getting more convenient.

Machine learning models, especially deep neural networks can learn and predict aeroelastic conditions, in our case, flutter speed. Since flutter prediction with simulations is time-consuming, costly, and computationally intense. With the created machine learning model, the prediction of the flutter speed is accurate and easy, and since training is already done, it is ready to use.

The model we created is limited to 7000 instances, while also 0.2 of them are used for testing and validation. Therefore model's accuracy and generalization capabilities are rather small. The paper by Wang [28], utilized nearly 350 thousand different inputs, therefore their model will be naturally more accurate and better at generalizing. However, since they used real-world data, their model may be too general, since we selected a base aircraft first, and then created different configurations for that aircraft, our model is more purpose-built and might be more useful for design optimization cases.

We created a machine-learning model for flutter speed prediction, this can also be extended into also detection of the divergence since divergence can be detected from the same data. Also, since the design configurations for aircraft are usually commercial secrets, these companies can create large models, similar to large language models, that sub-models are combined together to create a large deep learning model. I highly believe that we are close to this milestone and the aerospace industry will be revolutionized.

In this thesis, we utilized a lot of different disciplines for instance from computer science to mechanical engineering to statistics. With a small budget and limited resources, we created a rather large database that contains thousands of raw data about aeroelastic analyses. This created database can also be utilized to do some data mining since while the data was generated there was no specific output in mind. So it would be interesting to see a comprehensive data analysis on the database we created.

During this thesis journey, I learned some topics about statistics, machine learning, and aeroelasticity. I stumbled upon a lot of obstacles, but I learned a lot to make me a better engineer than I was before.

Bibliography

[1] Edward Albano and William P Rodden. A doublet-lattice method for calculating lift distributions on oscillating surfaces in subsonic flows. *AIAA journal*, 7(2):279–285, 1969.

[2] D. Bellinger. *MSC/NASTRAN Aeroelastic Supplement*. The MacNeal-Schwendler Corporation, Los Angeles, CA, USA, 1980.

[3] R. L. Bisplinghoff, H. Ashley, and R. L. Halfman. Aeroelasticity. 1955.

[4] Erasmo Carrera, Maria Cinefra, Marco Petrolo, and Enrico Zappino. *Finite element analysis of structures through unified formulation*. John Wiley & Sons, 2014.

[5] L. Chen et al. Optimization of wing geometry for aeroelastic performance using genetic algorithms and machine learning. *Aerospace Science and Technology*, 112:106604, 2022.

[6] Peter Flach. *Machine learning: the art and science of algorithms that make sense of data*. Cambridge university press, 2012.

[7] J. P. Giesing, T. P. Kalman, and W. P. Rodden. Subsonic unsteady aerodynamics for general configurations; part i, vol. i - direct application of the nonplanar doublet-lattice method. Technical Report AFFDL-TR-71-5, Part I, Vol. I, Air Force Flight Dynamics Laboratory, 1971.

[8] JP Giesing, TP Kalman, and WP Rodden. Subsonic steady and oscillatory aerodynamics for multiple interfering wings and bodies. *Journal of Aircraft*, 9(10):693–702, 1972.

[9] R. L. Harder and R. N. Desmarais. Reply by authors to w. p. rodden, j. a. mcgrew, and t. p. kalman. *J. Aircraft*, 9, 1972.

[10] Robert L Harder and Robert N Desmarais. Interpolation using surface splines. *Journal of aircraft*, 9(2):189–191, 1972.

[11] Robert L Harder, Richard H Mac Neal, and William P Rodden. A design study for the incorporation of aeroelastic capability into nastran. Technical report, NASA, 1971.

[12] Hexagon. *MSC Nastran 2022.3 Aeroelastic Analysis User's Guide*. Hexagon Manufacturing Intelligence, 2022. Available at nexus.hexagon.com/documentationcenter/.

[13] G. Hinton. Deep learning: A technology with the potential to transform health care. *JAMA*, 320(11):1101–1102, 2018.

[14] Dewey H Hodges and G Alvin Pierce. *Introduction to structural dynamics and aeroelasticity*, volume 15. cambridge university press, 2011.

[15] A. Johnson. The challenges of integrating machine learning into aeroelastic systems. *International Journal*, 2023.

[16] Hemant Joshi and Peter Thomas. Review of vortex lattice method for supersonic aircraft design. *The Aeronautical Journal*, pages 1–35, 2023.

[17] A. Lee. Neural network-based flutter prediction. *Journal of Aerospace Engineering*, 33(4):04020035, 2020.

[18] In Lee, Hirokazu Miura, and Mladen K Chargin. Static aeroelastic analysis for generic configuration wing. *Journal of aircraft*, 28(12):801–802, 1991.

[19] Nils J Nilsson. Introduction to machine learning. an early draft of a proposed textbook (1998). *Software available at http://robotics. stanford. edu/people/nilsson/mlbook. html*, 2020.

[20] Massimiliano Orlandi. *Tailoring aeroelastico di strutture composite con angolo di laminazione variabile= Aeroelastic tailoring of variable angle tow composite structures*. PhD thesis, Politecnico di Torino, 2022.

[21] Mayuresh J Patil, Dewey H Hodges, and Carlos ES Cesnik. Nonlinear aeroelasticity and flight dynamics of high-altitude long-endurance aircraft. *Journal of Aircraft*, 38(1):88–94, 2001.

[22] Marco Petrolo. Advanced aeroelastic models for the analysis of lifting surfaces made of composite materials. *Erasmo Carrera*, 2011.

[23] W. P. Rodden, J. A. McGrew, and T. P. Kalman. Comment on "interpolation using surface splines". *J. Aircraft*, 9:869–871, 1972.

[24] William P Rodden, Paul F Taylor, and Samuel C McIntosh Jr. Further refinement of the subsonic doublet-lattice method. *Journal of aircraft*, 35(5):720–727, 1998.

[25] W.P. Rodden and E.H. Johnson. *MSC NASTRAN Version 68 Aeroelastic Analysis User's Guide*. MSC Software Corporation, Santa Ana, CA, USA, 2004.

[26] B. Smith and C. Jones. Reinforcement learning for adaptive aeroelastic control. *AIAA Journal*, 59(5):1624–1635, 2021.

[27] Libo Wang, Zhiqiang Wan, Qiang Wu, and Chao Yang. Aeroelastic modeling and analysis of the wing/engine system of a large aircraft. *Procedia Engineering*, 31:879–885, 2012.

[28] Yi-Ren Wang and Yi-Jyun Wang. Flutter speed prediction by using deep learning. *Advances in Mechanical Engineering*, 13(11):16878140211062275, 2021.

[29] Jan Robert Wright and Jonathan Edward Cooper. *Introduction to aircraft aeroelasticity and loads*, volume 20. John Wiley & Sons, 2008.

[30] Weixing Yuan and Xiaoyang Zhang. Numerical stabilization for flutter analysis procedure. *Aerospace*, 10(3):302, 2023.

[31] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. Dive into Deep Learning, 0.17.2 edition, 2021. Accessed: 2024-03-31.

[32] Y. Zhang and N. H. Kim. Surrogate modeling for fluid-structural interaction problems using deep learning. *Computers & Fluids*, 185:21–31, 2019.