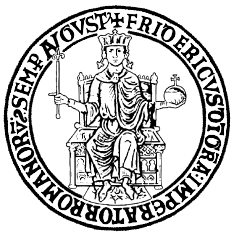


Università degli Studi di Napoli Federico II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea Magistrale in Ingegneria Aerospaziale

Tesi di Laurea Magistrale
in
Ingegneria Aerospaziale

Development of a Java-Based Framework for Aircraft Preliminary Design

Wing Aerodynamic Analysis Module,
Aircraft Longitudinal Static Stability Module

Relatori:
Prof. Ing. Fabrizio Nicolosi
Prof. Ing. Agostino De Marco

Candidato:
Manuela Ruocco
Matricola M53/408

Anno Accademico 2014/2015

Dedica

Abstract

The purpose of this Thesis work is to introduce the features and the potenciality of ADOpT (*Aircraft Design and Optimization Tool*), a java-based framework concieved as a fast and efficient tool useful as support in the preliminary design phases of an aircraft, and during its optimizaton process.
The Aircraft Design and Optimization Tool (ADOpT) development originates in the Departement of Industrial Engineering of University of Naples "Federico II", where is still in development.
ADOpT is the software with the GUI that uses the functionality of the library Java Program toolchain for Aircraft Design (JPAD) that, at present, is capable to perform a multidisciplinary analysis of an aircraft whose data can be entered by the user, with an XML, or loaded into memory. The ultimate goal of JPAD is to carry out an optimization process where the analysis are cyclically repeated in order to optimize some parameters while keeping others in fixed limits. Currently the software is able to esimate the aircraft weight breakdown, the center of gravity location, calculate some aerodynamic parameters and estimate the performance. All these types of estimates can be usually performed using several analysis methods, comparable and interchangeable. It is also provided a static longitudinal stability analysis, take-off and landing performances and the generation of Payload Range chart.
JPAD can be used from the command line or with a dedicated graphical user interface (ADOpT). The GUI allows the user to have an immediate feedback about the aircraft features when changing the input parameters, to manage multiple aircraft simultaneously and compare them side by side, and to view a 3D CAD model of the aircraft.
The ADOpT potentiality, in the world of research or in industry, are remarkable and the software strengths are a considerable computing speed and flexibility, with an user-friendly GUI.

The structure of this thesis work has as ultimate goal to provide a comprehensive overview about JPAD and, at the same time, it is intended to be a developer's manual. The first chapters provide a complete software overview paying particular attention at actual features and future goals. Following chapters introduce some case of study and the results achieved. At the beginning of each chapter is exposed the theoretical background, afterwards there is a description of the Java architecture and, at the end is reported the Test class used for the analysis and its results.

Sommario

Lo scopo che il presente lavoro di Tesi auspica raggiungere è quello di presentare le capacità possedute e le potenzialità future di ADOpT, un software scritto in Java che si configura come uno strumento veloce ed efficiente per il supporto nella fase di progetto preliminare di un velivolo e per la sua ottimizzazione.
Lo sviluppo di ADOpT (*Aircraft Design and Optimization Tool*) nasce all' interno del Dipartimento di Ingegneria Industriale dell' Università degli Studi di Napoli Federico II, ove è tutt'ora in fase di progresso.
ADOpT rappresenta il software con interfaccia grafica le cui funzionalità sono implementate nella libreria JPAD il quale è attualmente in grado di svolgere una parziale analisi multidisciplinare di un velivolo i cui dati sono immessi dall' utente tramite XML o caricati in memoria. La linea guida dello sviluppo del software porta verso l'implementazione di un processo di ottimizzazione nel quale le analisi sono ciclicamente ripetute al fine di ottimizzare alcuni parametri mantenendone altri all' interno di limiti imposti.
Attualmente JPAD è in grado di effettuare una completa stima dei pesi, valutare la posizione del baricentro, calcolare un notevole numero di parametri aerodinamici e caratteristiche di performance. La stima di ciascun parametro può essere effettuata tramite diversi metodi implementati, tra di loro confrontabili ed intercambiabili. Inoltre è prevista un' analisi di stabilità statica, prestazioni di decollo ed atterraggio e la generazione del diagramma *Payload Range*.
JPAD può essere utilizzato sia in modalità *batch*, ossia da riga di comando, che tramite interfaccia grafica (ADOpT). Tale duplice scelta consente di ottenere le migliori prestazioni sia nei processi di analisi, ove la GUI consente di avere un immediato riscontro grafico, sia nei processi di ottimizzazione.
Dunque le potenzialità di ADOpT nel mondo della ricerca od anche in quello industriale sono senza dubbio notevoli e il software gioca i suoi punti di forza in un' elevata flessibilità e una notevole rapidità di calcolo, senza dimenticare una *user-friendly* interfaccia grafica.

L' organizzazione di questo lavoro di Tesi è stata studiata per cercare di fornire una completezza di esposizione, ma allo stesso tempo risultare un utile manuale per lo sviluppatore. I primi capitoli forniscono una visione globale del software con particolare attenzione alle funzionalità presenti e alle scelte effettuate. I capitoli che seguono presentano una panoramica su alcune funzionalità di ADOpT con i relativi casi di studio e i risultati ottenuti dalle analisi. Per ogni capitolo viene preliminarmente fornita una visione globale circa la teoria alla base dei metodi implementati, seguita dalla descrizione delle classi e dei metodi relativi in Java ed è, infine, riportato il codice della *Test Class* implementata per lo svolgimento delle analisi con i relativi risultati.

Acknowledgements

Contents

Part I

Aircraft Design Overview

Chapter 1

Aircraft Design

Scientists discover that which exists,
engineers creates that which never was.
– T. Von Karman

Engineering design is a non-unique iterative process, the aim of which is to reach the best compromise of a number of conflicting requirements. Whether the need is for a totally new item or for a development of an existing one the design procedure commences with an interpretation of the requirements into a first concept.
The conceptual and preliminary design phases play a key role for the best development of future transport aircraft. A software framework that could help in finding a configuration which satisfies several basic requirements and eventually a constrained optimum is an essential tool for academic and industrial aircraft design activity. Since in the conceptual and preliminary design phases a lot of different configurations have to be analyzed, the software has been conceived to provide results in a relatively short computational time. For these reasons the the aircraft design is projected to the utilization of new analysis tools who hold a main role in the design and optimization process.
Some new software have recently followed innovative approaches considering concepts like KBE (knowledge Based Engineering) and Multi-disciplinary Design Optimization (MDO) (Multi-Disciplinary Optimization) and have highlighted the necessity of an efficient graphical user interface and the importance of making the application results easily exploitable with external software.
In this scenery ranks ADOpT and its reference library JPAD that is a Java-based desktop application developed at the University of Naples Federico II, conceived as a fast, reliable and user friendly computational aid for aircraft designers in the conceptual and preliminary design phases.

1.1 Historical Overview

During the last century, three design methods have been developed in the aerodynamic design process. Until the early 1960s, only two methods were available: empirical and mathematical methods. Since the 1960s, a third method has been developed: computational fluid dynamics, which can be considered as a combination of the two earlier methods. These three methods and their history will be discussed briefly in this chapter.
The empirical method is the oldest method of aerodynamic design; until about 1940, it was the only practical tool available. Based on either theory or experiments or a combination

of both, the empirical methods consist of handbooks which are essentially a collection of graphs and equations. They are meant to give a relation between elementary parameters of the geometry of the aircraft and the desired characteristics of the aircraft.
Through the mathematical methods the aerodynamic characteristics of the detailed geometry are obtained by physical insight, in other words, through pressure distributions. This approach uses mathematical modelling of flow physics.
With the development of computers, more and more powerful numerical methods became available to aircraft designers. These numerical methods are known as computational fluid dynamics. It allows them to obtain the intended characteristics of the aircraft (or one of its major components) by directly determining the required detailed aerodynamic shape through the use of fluid dynamics and the associated pressure distribution.[60]

Modern aerospace design begins with the desire of mankind to achieve powered, heavier-than-air flight. Early aerospace designs were characterized not only by a great deal of trial and error but also by a not inconsiderable amount of analysis.
At the beginning of the aviation history the entire design process occurred in six months. Some companies produced more than ten aircraft in only 20 years. Today in order to develope a new transport aircraft it takes more than 10 years. The current aircraft design strategy is linked to industrial growth, which in turn depends on national infrastructure, governmental policies, workforce capabilities, and natural resources; these are generally related to global economic–political circumstances. More than any other industry, the aerospace sector is linked to global trends. It is possible to view the first 75 years of aerospace design as a series of stages: first came the early pioneers, working at a time when many were skeptical about the benefits of flight or even if it was possible at all. Next came the phase where outstanding individuals dominated aeronautical science and design during the interwar and early postwar years. This was followed in the 1950s and 1960s by a greater emphasis on design teams and the massive expansion of the great aerospace companies that we recognize today, such as Lockheed Martin and Rolls-Royce. Following these stages, the advent of CAE, design and manufacture in the mid-1970s transformed activities in aerospace with an increasing focus on faster design cycles, more accurate predictive capabilities and the rise of computer science. At the same time, the great drawing offices of the 1950s declined and fewer people were involved in the design of each individual component.[33]
Recent developments in information technology have had a major impact upon the aeronautical design process. The emphasis has become one where the whole design procedure from concept to the derivation of manufacturing data is covered by an integrated information technology system. Computer aided design techniques are used to produce digital data. While these procedures are particularly relevant to the detail phases of the design process their application commences as soon as a requirement is defined and numerical data are derived from it.[40]

1.2 General Process of Aircraft Design

The design process is usually divided into the following phases:

1. **Requirements phase.**

2. **Conceptual design phase.** This phase determines the feasibility of meeting the requirements with a credible aircraft design. The concept definition phase can be characterized

as the highly creative and imaginative idea stage during which the component geometry, placement and connectivity of a future aircraft designed to fulfil the needs of a specific market are defined. Conceptual design also entails the development of a novel aircraft concept at an overall system level in competition with a more traditional layout. The objective is to explore a preferred configuration to determine a layout which is technically superior and economically aviable.

3. **Preliminary design phase.** This phase aims at defining subsystems and making component trade-offs. Specialists from different functional groups contribute to this process of refining the initial vehicle concept. This team will (re)design the delivered baseline vehicle in suffcient detail to carry out supporting analysis and specify peripheral testing programmes but not with enough detail to specify each sub-assembly.

4. **Detail design phase.** In the detail design phase, the confi guration is "frozen" and the decision has been made to build the aircraft. Detailed structural design is completed. All of the detail design and shop drawings of the mechanisms, joints, fittings, and attachments are accomplished. Interior layout is detailed with respect to location and mounting of equipment, hydraulic lines, ducting, control cables, and wiring bundles. This development phase is entered soon after the aircraft is committed to production and lasts between two and three years.

5. **Testing phase.**

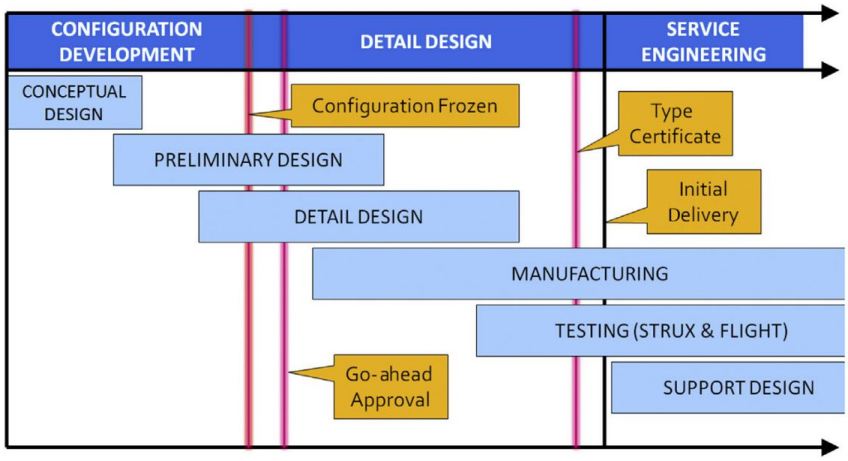


Figure 1.1: Aircraft design process for Torenbeek (1986).

1.3 Design Requirements and Objectives of an Aircraft

The general design objective of a transport aircraft is to transport a payload over a distance between airports against minimum costs (i.e. at an optimum speed). So, in the design of an aircraft there is never a right answer only a best answer at a point in time. The reason is that the design of an aircraft is a balance between the following competing requirements:

- **Technical.** Performance, survivability
- **Signature.** Survivability, appearance
- **Economic.** Cost, LCC

- **Political.** Policy, payback, risk, and so on
- **Schedule.** When needed? The need to be first to market
- **Environmental.** Limited energy source, noise, hydrocarbon emissions

Also, the priorities of these requirements change with time. An aircraft might be designed to certain technical and economic requirements, but if the government administration changes, then the priority requirement becomes political or environmental. The advice to the designer is to remain flexible and develop as robust a design as possible so that it will survive as the requirements change over time. The watchwords are compromise, balance, and flexibility.

Significant characteristics of the various classes of aircraft	
Private aircraft	"Poor people" / Flying clubs Costs are dominant
	"Rich people" / business men Costs / performance
Business aircraft	Balance between costs, comfort and performance
Agricultural aircraft	Very specialized, good balance between costs and revenue with probably accent on costs
Aerobatic aircraft	Very specialized, performance (manoeuvrability) is dominant, often very expensive
Civil transport aircraft	Costs may be high if, given a certain fare structure, revenues are higher
Military aircraft (non-transport types)	Costs may be high, military capabilities and operational readiness are dominant
	Most military aircraft have apart from load-carrying requirements high demands on both manoeuvrability ("dog-fighting") and stability ("aiming platform")

Figure 1.2: Main design requirements of various aircraft categories.[60]

An aircraft comprised of several major components. It mainly includes wing, horizontal tail, vertical tail, fuselage, propulsion system, landing gear and control surfaces. In order to make a decision about the configuration of each aircraft component, the designer must be fully aware of the function of each component. Each aircraft component has inter-relationships with other components and interferes with the functions of other components.

No	**Component**	**Primary function**	**Major areas of influence**
1	Fuselage	Payload accommodations	Aircraft performance, longitudinal stability, lateral stability, cost
2	Wing	Generation of lift	Aircraft performance, lateral stability
3	Horizontal tail	Longitudinal stability	Longitudinal trim and control
4	Vertical tail	Directional stability	Directional trim and control, stealth,
5	Engine	Generation of thrust	Aircraft performance, stealth, cost, control
6	Landing gear	Facilitate take-off and landing	Aircraft performance, stealth, cost
7	Control surfaces	Control	Maneuverability, cost

Figure 1.3: Aircraft major components and their functions.[52]

1.4 Aircraft Design Optimization

At the beginning of the aircraft design history, optimization was limited to relatively superficial investigations to study the effects of varying a few major design parameters. Since the 1970s, there has been a tremendous expansion of optimization strategies and algorithms supporting advanced designers. The introduction of automated optimization has enabled designers to go into much greater in depth and fidelity of analysis than before. Synthesis programs effectively connecting the inputs and outputs of the functional group disciplines by means of an automatic control logic have been developed at aircraft manufacturers, research establishments and academia. Sophisticated computer assisted design (Computer-Aided Design (CAD)) systems for defining three-dimensional body geometries and computer graphics tools for rapidly preparing parametric surveys are available at a modest cost.
There are several options available to an aerodynamicist to obtain the best possible solution for a design problem. In general the optimization is an iterative process control system which interprets numerical results and then iteratively assesses variables to seek the global optimum of an objective function. In the aircraft design context, the objective function is used to decide which configuration is considered the best combination of design variables.[79]
It is emphasized frequently that optimization of individual goals through separate design considerations may prove counterproductive and usually prevents the overall (i.e., global) optimization of ownership cost. MDO offers good potential but it is not easy to obtain global optimization; it is still evolving. In a way, global MDO involving many variables is still an academic pursuit. Industries are in a position to use sophisticated algorithms in some proven areas. An example is reducing manufacturing costs by reshaping component geometry as a compromise – such as minimizing complex component curvature. The compromises are evident in offering a family of variant aircraft because none of the individuals in the family is optimized, whereas together, they offer the best value.

According to [40], the optimisation of the configuration of the aircraft within the constraints imposed by the specification is an essential feature of the project definition process. Optimisation implies that the proposed design concept not only meets the specification, but does so when a target criterion has been imposed, usually the minimisation of mass or some aspect of costs. The basis of optimisation is a comparison of different design concepts and configuration variations within a given concept to determine the one which best meets the specification. Broadly the process is undertaken at two levels.

- **Concept/configuration studies** At this level alternative concepts and configurations are investigated to establish the one which would seem best suited to meet the requirements. For example a military combat requirement might possibly be met by aircraft of conventional tail, foreplane or tailless layout. Mostly the configuration of transport aircraft is well established. This phase of the study may be included at the feasibility level.

- **Parametric studies within a given configuration** At this level the dominant parameters are varied to ascertain the best combination of them. These parameters include such items as wing geometry determined by aspect ratio, sweep, taper and thickness as well as variations in fuselage layout, powerplant installation and so on. The benefits of selecting near optimum values of certain parameters are "traded-off" against the implied penalties imposed upon other parameters. The parametric studies may be commenced during feasibility investigations and certainly form a major part of the project definition phase. The two fundamental design characteristics which drive the parametric studies are:

- Wing loading, that is wing area as a function of take-off mass or weight
- Thrust or power loading, defined as the ratio of the basic powerplant to the aircraft mass or weight

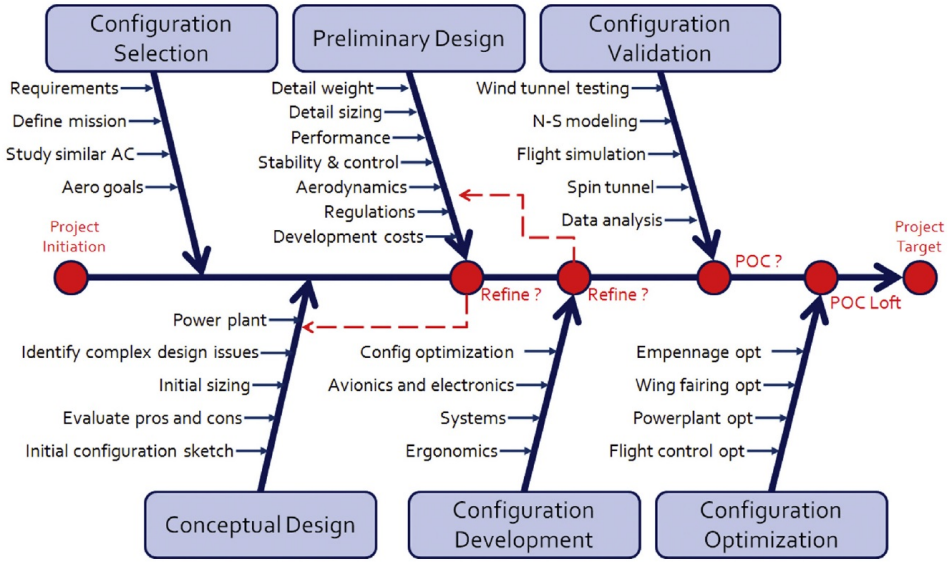


Figure 1.4: A typical fishbone diagram.

Chapter 2

Application overview

I do not fear computers.
I fear the lack of them.
– Isaac Asimov

In this Chapter an overview of the ADOpT software, and its reference library JPAD will be presented. ADOpT is a Java-based desktop application developed at the University of Naples Federico II, conceived as a fast, reliable and user friendly computational aid for aircraft designers in the conceptual and preliminary design phases. The ultimate goal of such a tool is to perform a parametric, multi-disciplinary analysis of an aircraft and then search for an optimized configuration. The search domain boundaries are usually defined by the user through a set of specifyed parameters. An important design requirement of ADOpT is related to its interoperability with other engineering analysis tools. In fact, the application can be easily integrated into a comprehensive aircraft optimization cycle. This is made possible because ADOpT can be launched both in Graphical User Interface (GUI) and command line mode. Much care has been given to input/output and configuration files to increase the possible uses of the software.

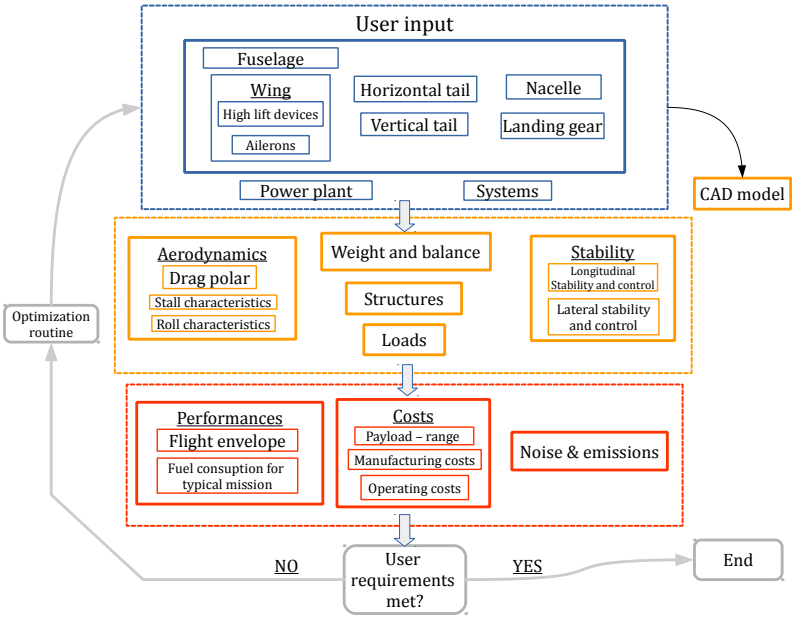


Figure 2.1: Software calculation modules.

2.1 Software architecture

JPAD library is actually divided into three package: JPADConfigs, JPADCore and JPADSandbox. Each package is organized in several classes or more package in order to have a clear and simple classification. The possibility to place similar classes in the same package has been extensively exploited for the same reasons, see section A.2. The source code has been extensively commented following Javadoc practices. This enabled us to automatically generate documentation using Doxygen [82]

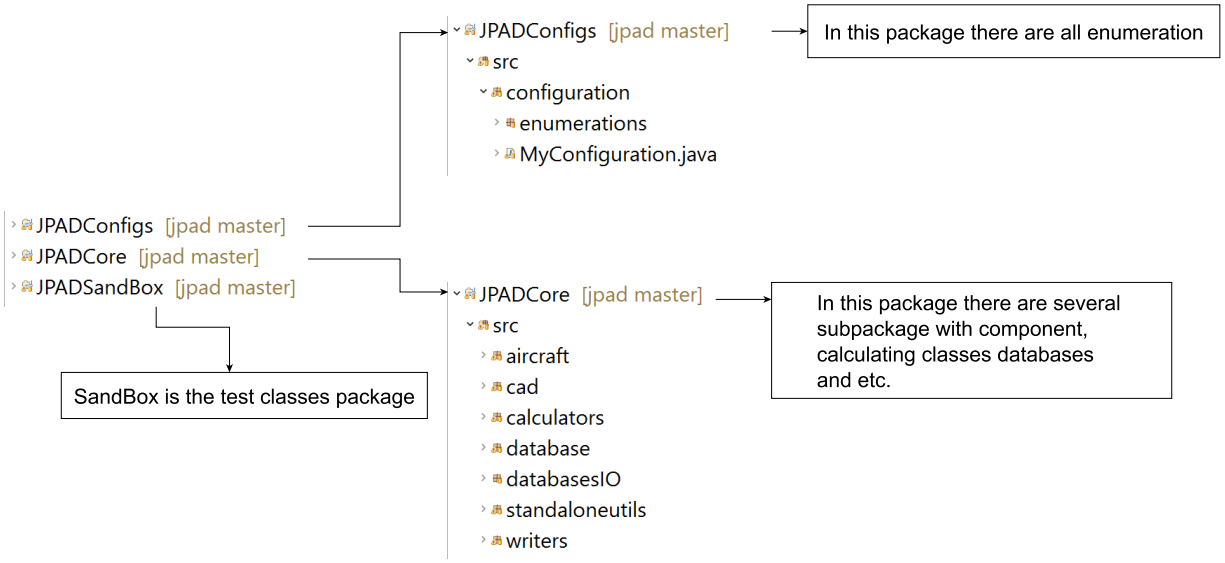


Figure 2.2: Software package division.

The main package of JPAD library is JPADCore where there are all aircraft component, the managers of calculator and its computers classes. In the following subsection will explained the main features of JPADCore.
JPADCore contains the following sub-packages:

1. `aircraft` → This package contains all aircraft's component classes, their calculators and the related managers and also the operating condition's class.
2. `cad` → This package contains everything concerning the creation of CAD model.
3. `calculators` → In this package there are the classes, divided in package by type of analysis, whose methods implement the calculation formulas.
4. `database` → This package contains the classes that manage the database .h5 whose data are useful for the analysis.
5. `standaloneutils` → In this package there are several calculation classes with mathematical tools to support the analysis.
6. `writers` → this class is in charge of writing files.

Following will be explained the main features of the fundamental packages.

2.1.1 `Aircraft` package

Each component (e.g., the fuselage, the wing, the engine) is therefore defined in its own class. In the aircraft package there are other sub-package for components. In each of these there are a class for the object definition and a manager for the analysis. These manager classes uses the utilities located in the calculator package.
The sub-package in the `aircraft` package are the following:

- `Auxiliary` that manages the airfoils
- `Component` that contains the `aircraft` class, the aircraft components packages (fuselage, lifting surfaces, nacelles, power plant, fuel tank, landing gear, systems) and, inside them, their manager analysis classes used to call related analysis methods.
- `OperatingConditions`

The main aircraft components are organised as follows:

Basic data for describing the **fuselage** are contained in the `fuselage` class. This holds all the fuselage overall properties, such as the length, maximum width and maximum height, length ratios between the nose and the tail parts length to the constant section part length, number of decks and so on. Other classes in the package manage the shape of fuselage sections and outlines (which define the fuselage shape in the xz and xy planes) and aerodynamics calculations (Aerodynamics class).

The data relating to the **lifting surfaces** are contained in the `liftingSurface` class. Since the lifting surfaces of an aircraft share several characteristics, a single class has been-created to manage the wing, the horizontal and vertical tail and, eventually, the canard. The lifting surface specific category (that is, wing, horizontal tail etc.) is acknowledged through a `MyComponentEnum` variable which has to be specified when creating the lifting surface object. By default, a lifting surface has three primary span stations: root ($\eta = 0$), middle and tip ($\eta = 1$). The middle station location is user-defined and is used to represent a change in chord.y/ law; such a change can be due to a lifting surface kink or to the beginning of a tapered part. If the wing is simple tapered the middle station can also be omitted.

The data relating to the **airfoils** are handled by `Airfoil` class, that creates for each lifting surface three airfoils, located respectively at $\eta = 0$, $\eta = 1$ and at middle station. An airfoil object holds:

1. the position along the semispan, η;
2. twist value relative to root, ϵ_g;
3. zero lift angle of attack, $\alpha_{0\ell}$;
4. angle of attack value at the end of the linear part of $C_l(\alpha)$ curve, α^*;
5. stall angle of attack, α_{stall};
6. lift gradient of the linear part of $C_l(\alpha)$ curve, C_{ℓ_α};
7. minimum drag coefficient, $C_{d,min}$;

8. lift coefficient at $C_{d,min}$, $C_l@C_{d,min}$;
9. lift cofficient at the end of linear part;
10. maximum lift coefficient, $C_{\ell_{\max}}$
11. drag polar K factor;
12. aerodynamic moment coefficient gradient, C_{m_α};
13. aerodynamic center x coordinate, x_{ac};
14. aerodynamic moment coefficient with respect to the aerodynamic center, $C_{m_{ac}}$ or C_{m_0};
15. aerodynamic moment coefficient with respect to the aerodynamic center at stall, $C_{m_{ac},stall}$;
16. maximum thickness to chord ratio, t/c_{max};
17. x,z non dimensional coordinates.

At the time of writing all these quantities have to be entered by the user. The `Airfoil` class provides the `populateCoordinateList()` method which transforms the non dimensional coordinates provided by the user in order to obtain their actual coordinates, which takes into account of actual chord length, ACRF position, sweep, twist and dihedral.

The **power plant** is defined in the package `PowerPlant` in `components`. In this package there is the `engine` class that initializes the data related to the single engine. An other class in the same package, `powerPlant` creates the parametric model of the power plant using a list of objects creates by `engine`.

2.1.2 `MyOperatingConditions` class

As the name suggests, this class contains all the data related to atmosphere conditions (which are currently derived from altitude value using the 1976 ISA model), current speed (Mach number), gravitational acceleration (9.806 65 m/s^2) and sea level pressure (101 325 Pa).

A `MyOperatingConditions` object can be created regardless of the aircraft configuration as the class is entirely self contained: the aircraft could also be undefined at the time of the object creation. If the user wants to run several analysis at different operating conditions, a `MyOperatingConditions` instance has to be created for each one.

2.1.3 `CAD` package

Throughout the development of the application, great care has been given to the making of the CAD model for several reasons:

- it enables the user and the user developer to have an immediate feedback about the data provided to the application: if some geometrical parameter is wrong, the CAD model makes it impossible not noticing it;
- it allows the user to run a CFD analysis with an external program. The CAD model has been in fact built so that it is ready to be meshed by an external mesher without any further adjustment;

- it provides an accurate estimate of the wetted surface of each component.

The creation of the CAD model was made possible by the occjava library, whose classes and methods have been used to build each component's model.

2.1.4 `Calculators` package

This package includes all the calculators classes. Calculator classes have been created to evaluate quantities related with more than one component. The calculator classes are the following:

- `aerodynamics` package
- `cost` package
- `geometry` package
- `performance` package
- `stability` package

Each of these package contains calculators classes and concerning methods that implement all mathematical equations. These methods are called by the analysis manager classes that are in the package of relatives components (such as fuselage, lifting surfaces etc.).

2.1.5 `Database` package

In order to manage a large quantity of data, it has been necessary to read data from graphs available in literature; for such a reason an extensive database has been built over the years by several of my colleagues to digitalize this data in order to exploit it in a computer program. The data has been stored in an hdf5 file.
In the `database` package there are all the classes that manage and reads the .h5 files. To obtain the useful data in JPAD, interpolating functions are used . These functions can be of one, two or three dimensions and read data from graphics that have been digitize previously.

2.1.6 `Writers` package

This class is based on the Apache POI library and provides methods which enable the user developer to easily add a sheet to an XLS file, to set the styles of the XLS file and to compare different XLS files.The last function permits to create an XLS file which contains a different aircraft data for each column; in such a way the comparison between two or more aircraft (or, simply, between slightly different configurations of the same aircraft) is easier and effective.

2.2 Graphical User Interface (GUI)

An extensive work has been done to set up an effective Graphical User interface (GUI) to reduce the time the user has to spend to obtain relevant results. The result presented in this section is the early step of graphical interface, which is currently in development.

The Java programming language greatly helped to build the GUI: several open source libraries (SWT, JFace) allowed us to build a functional yet pleasant GUI which allows the

user to easily change the aircraft's parameters, to view a 3D model of the defined geometry, to launch a new analysis and view the corresponding results. The current GUI appearance is shown in fig. 2.3. It is composed of several items:

1. a menu bar (on top), which holds all the available actions divided in sub-categories;
2. a toolbar (below the menu bar), which holds the most important actions needed to interact with the application. The toolbar, as the menu bar, is always visible to the user;
3. a project tree (on the left), a key component of the GUI since it provides access to all the components of the aircraft and the analysis results any time during the execution of the application. The project tree appears once an aircraft is created and can be eventually hidden;
4. a 3D view, which shows the CAD model of the aircraft selected by the user. The CAD model can be updated each time the user wants to check the changes made to the aircraft geometry;
5. a log message window, placed at the bottom, that tells the user the status of pending operations. This window can be hidden;
6. a tab folder, which contains all the windows opened using the project tree; these windows can be closed and reopened any time.

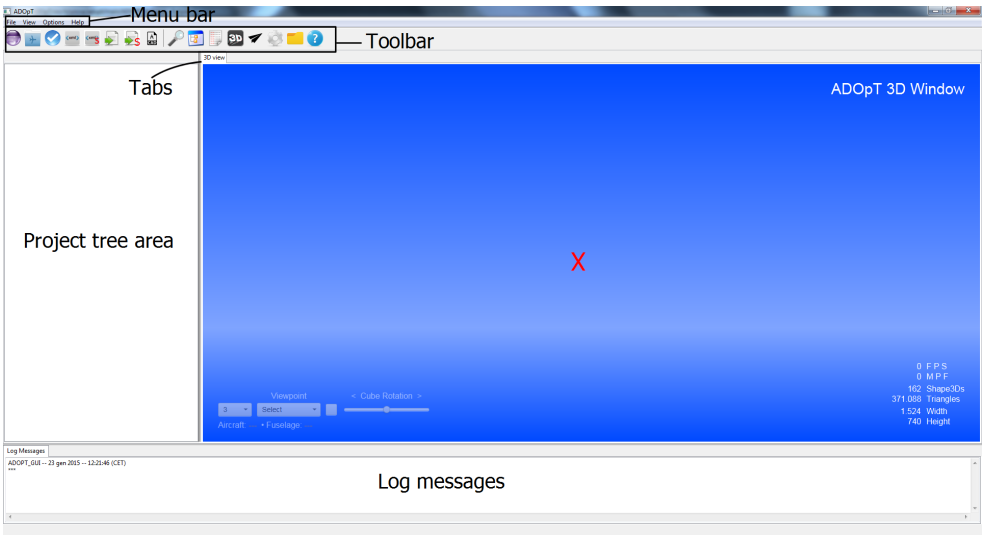


Figure 2.3: The GUI when the application is started.

2.2.1 Typical work session

In developing the application we focused on making the user's typical work session as simple as possible. Few basics steps are required for running an analysis:

1. create a new aircraft, which we will call A. This can be done using the corresponding button ( ) in the toolbar, which instantiates the default aircraft
2. set the parameters that define the aircraft model using the corresponding window opened using the tree

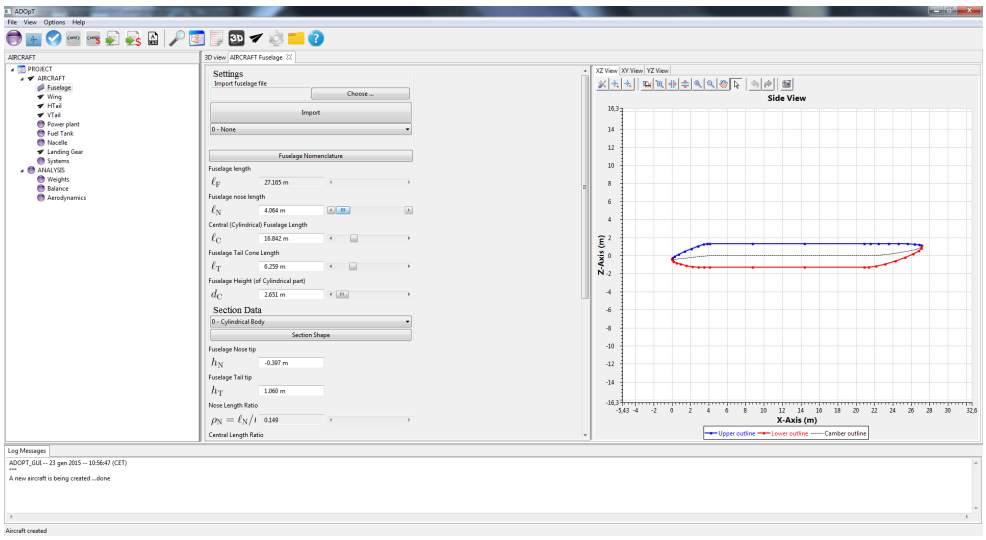


Figure 2.4: The window for changing the fuselage parameters.

3. explore the 3D model (3D) of the aircraft and eventually change some parameters if there is some error

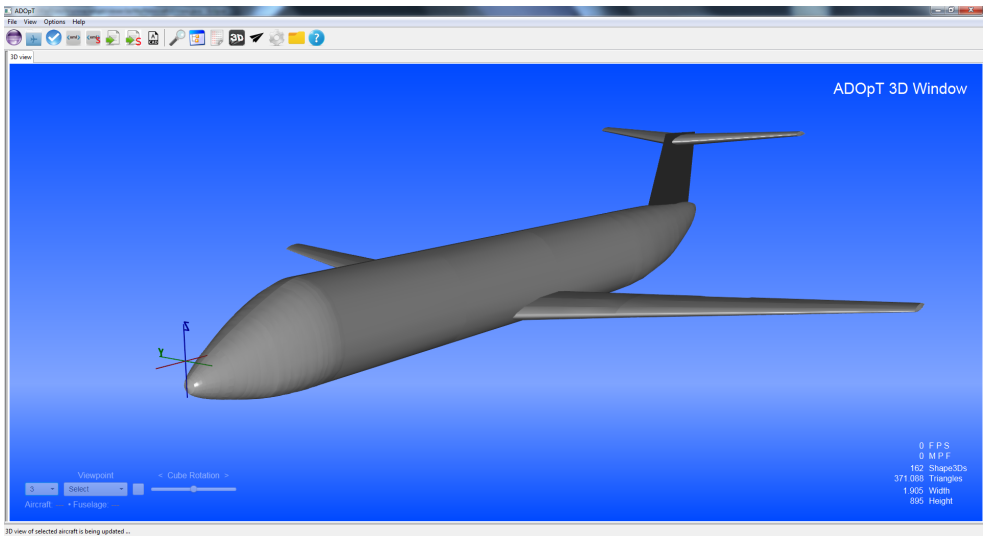


Figure 2.5: The aircraft 3D view (log window and project tree hidden).

4. execute a complete analysis (🔍) of the aircraft previously defined;
5. export the analysis results to an XML and/or an XLS file (📄);
6. eventually export the CAD model of the aircraft (CAD);
7. save the current aircraft to an XML file (xml).

At this point the user could simply change the current configuration until the analysis results are satisfactory. The application can however help the user in finding such a configuration, since it can hold multiple configurations simultaneously, analyse all of them and compare them side by side. To accomplish this task, once the first aircraft has been created, the user should:

1. import the aicraft previously saved or create an entirely new aircraft, which we will call B;
2. change B parameters and run a new analysis on it;
3. study the results and eventually change some of the parameters;
4. save every configuration and the corresponding results to file;
5. export both aircraft and the corresponding results to an XLS file.

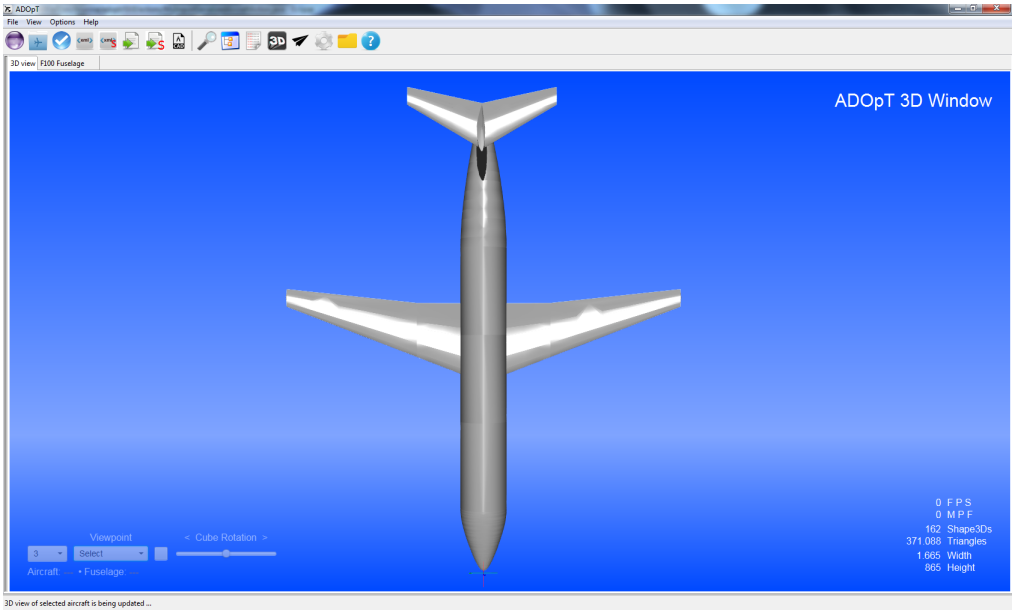


Figure 2.6: The aircraft 3D view.

There is no limit to the number of aircraft the application can handle; each aircraft is added to the project tree as shown in fig. 2.7 providing access to the corresponding components and analysis.

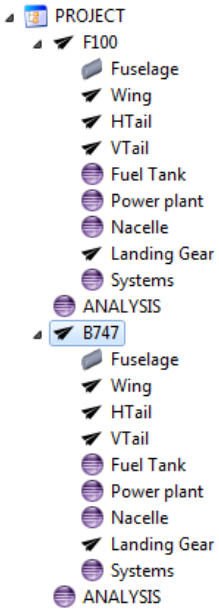


Figure 2.7: The project tree holding two different aircraft

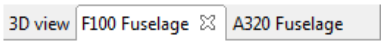


Figure 2.8: Same component tab belonging to different aircraft

2.2.2 CAD modelling

The application can also be used as a basic parametric CAD modeler. The capability to change the aircraft parameters using the corresponding controls in the GUI, coupled with the 3D view, allows the user to change each component shape and dimension, view the updated CADmodel and eventually export it to file once some satisfactory results have been obtained. The creation of the CAD model was made possible by the occjava library, whose classes and methods have been used to build each component's model. The CAD model can be saved in two different file formats: STEP and BREP; they have proven to take both little memory and to give the best results in terms of geometry representation. The IGES format gave instead mixed results so we preferred to use only the first two.

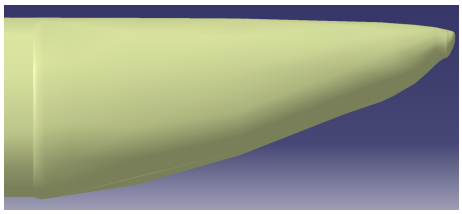


Figure 2.9: A detail of the tail of the fuselage obtained as a unique loft.

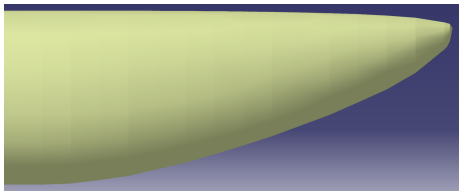


Figure 2.10: A detail of the tail of the fuselage obtained sewing together its parts.

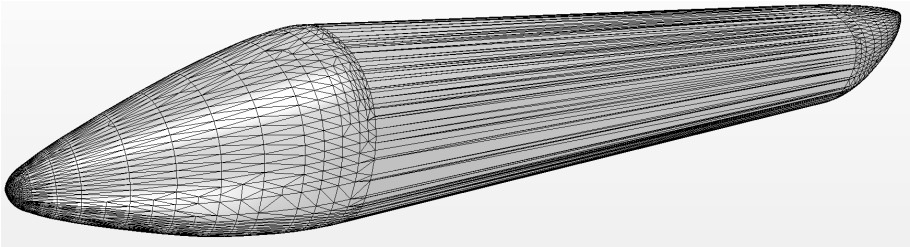


Figure 2.11: External fuselage shape exported as STEP file.

2.2.3 Interface with external software

With ADOpT it's also possible to execute some high-fidelity analyses using external tools (i.e. CFD Navier-Stokes solver, panel method, FEM for structural static and dynamic analysis, etc.) by importing the dedicated output (such as a CAD model) produced by the software. The higher-fidelity tools could improve the results obtained and give therefore some useful clues in order to start a new design loop with different requirements and/or different constraints. The possibility to interface the software with drawings and graphical representation (i.e. CAD or Catia) of the designed aircraft is also very important.

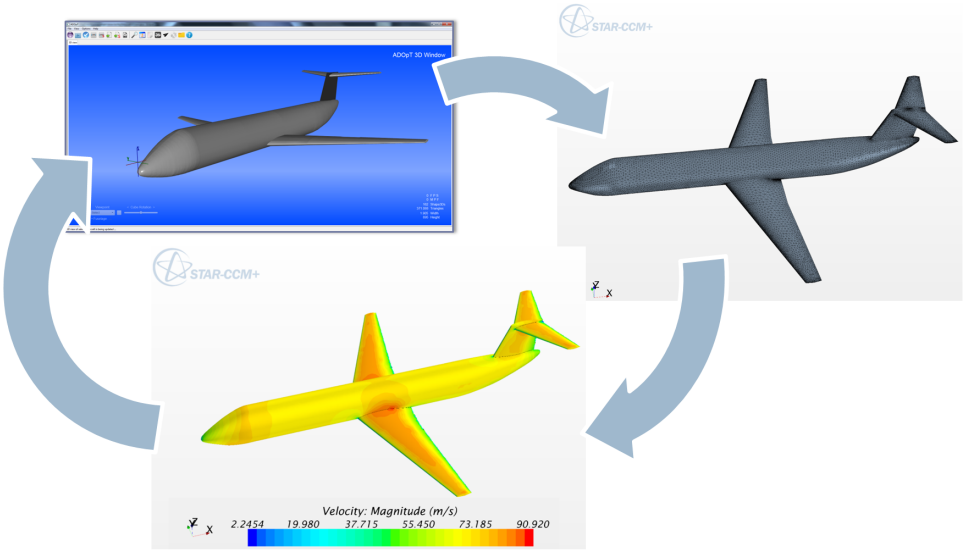


Figure 2.12: Interface with external CFD tools.

2.3 In Development

JPAD library is currently being developed in order to achieve a complete analysis of an aircraft and its optimization. The development is actually moving on a complete reorganization of the input file, and the implementation of other methods of analysis. Anyway, to date, it was reached an advanced level of implemented methods and a full functionality of some modules.

2.3.1 Input File

In order to obtain an useful and intuitive input structure a new data arrangement is in development. The JPAD input files are actually in XML data format. This extension will not change, but the purpose of the new structure is to allow the user to easily manage the input data in order to execute the desired analyzes.
The structure is inspired by that proposed by SUAVE. SUAVE is an open source suite constructed as a modular set of analysis tools written in Python and it is currently being developed in the Aerospace Design Lab at Stanford University. [17]

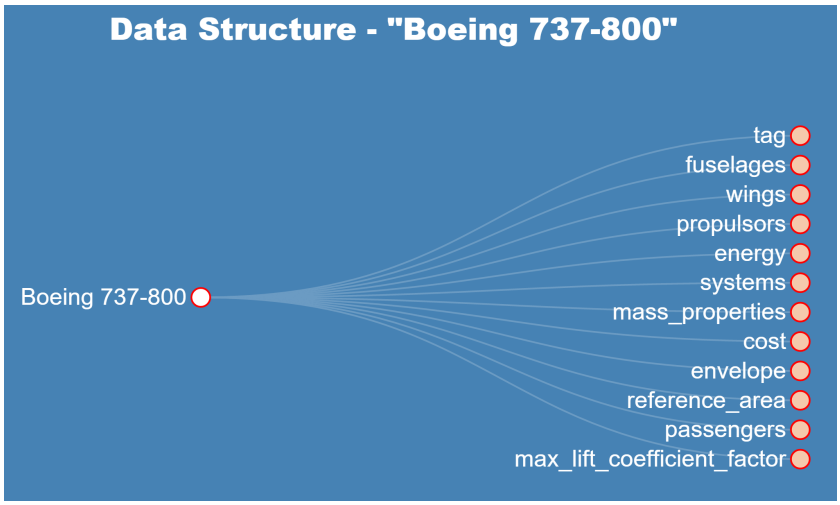


Figure 2.13: Data Structure used by SUAVE.

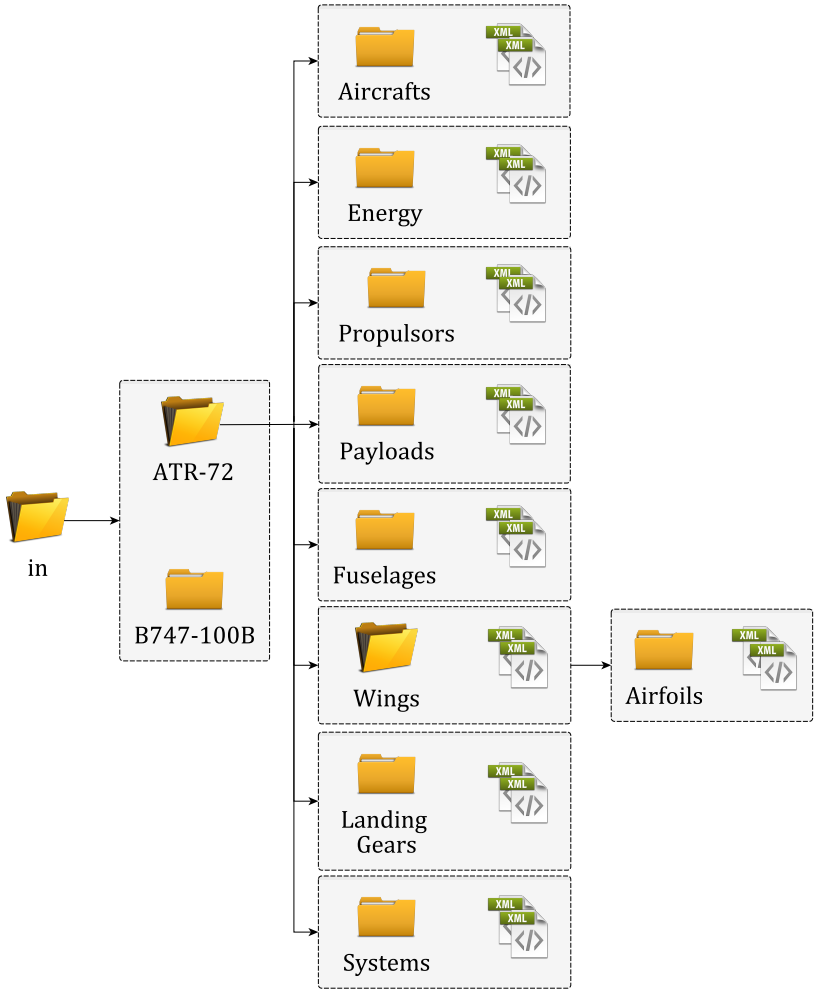


Figure 2.14: Folder Structure in development.

The fundamental criteria on which the construction is based are expressed following. In the input file folder, there are many sub-folder for different aircraft. In each of these there are several folders for component in which may be more XML file that corresponding to a different configuration. In fact, there is an XML file for each configuration of each aircraft component (fuselage, lifting surfaces etc.).In this way it's possible to choose different configuration for the same aircraft simply changing one or more XML input file.
For the analysis the useful data will be explained in a user guide. User could make a complete analysis on a new aircraft, simply defining the required data and organize them in separated xml file.
The main XML file is "aircraft.xml" which calls the other XML of the aircraft components. A typical file structure is in the following listings.

Listing 2.1 XML input file for an aircraft.

```
<?xml version="1.0" encoding="utf-8"?>
<jpad_config>
    <aircraft>
        <wings>
            <wing type="WING" file="wing.xml">
                <position>
                    <x unit="m">12.0</x>
                    <y unit="m"> 0.0</y>
                    <z unit="m"> 2.0</z>
                </position>
                <rigging_angle unit="deg">2.0</rigging_angle>
            </wing>
            <wing type="HTAIL" file="htail.xml">
```

```
                <position>
                    <x unit="m">30.0</x>
                    <y unit="m"> 0.0</y>
                    <z unit="m"> 4.0</z>
                </position>
                <rigging_angle unit="deg">0.0</rigging_angle>
            </wing>
            <wing type="VTAIL" file="vtail.xml">
                <position>
                    <x unit="m">30.0</x>
                    <y unit="m"> 0.0</y>
                    <z unit="m"> 5.0</z>
                </position>
                <rigging_angle unit="deg">0.0</rigging_angle>
            </wing>
        </wings>
        <fuselages>
            <fuselage file="fuselage.xml">
                <position>
                    <x unit="m">0.0</x>
                    <y unit="m">0.0</y>
                    <z unit="m">0.0</z>
                </position>
            </fuselage>
        </fuselages>
        <propulsors>
            <propulsor file="propulsor.xml">
                <position>
                    <x unit="m">12.0</x>
                    <y unit="m"> 5.0</y>
                    <z unit="m"> 0.0</z>
                </position>
                <tilting_angle unit="deg">2.0</tilting_angle>
                <yawing_angle unit="deg">0.0</yawing_angle>
            </propulsor>
            <propulsor file="propulsor.xml">
                <position>
                    <x unit="m">12.0</x>
                    <y unit="m">-5.0</y>
                    <z unit="m"> 0.0</z>
                </position>
                <tilting_angle unit="deg">2.0</tilting_angle>
                <yawing_angle unit="deg">0.0</yawing_angle>
            </propulsor>
        </propulsors>
    </aircraft>
</jpad_config>
```

Listing 2.2 XML input file for an aircraft.

```
<?xml version="1.0" encoding="utf-8"?>
<jpad_config>
    <wing mirrored="TRUE">
        <panels>
            <panel id="Inner panel">
                <semispan unit="m">4.0</semispan>
                <dihedral unit="deg">1.5</dihedral>
                <inner_section>
                    <airfoil file="naca63209.xml"/>
                    <geometric_twist unit="deg">0.0</geometric_twist>
                </inner_section>
                <outer_section>
                    <airfoil file="naca63209.xml"/>
                    <geometric_twist unit="deg">0.0</geometric_twist>
                </outer_section>
            </panel>
            <panel id="Outer panel" linked_to="Inner panel">
                <semispan unit="m">7.0</semispan>
                <dihedral unit="deg">3.5</dihedral>
                <outer_section>
                    <airfoil file="naca63209.xml"/>
                    <geometric_twist unit="deg">-2.5</geometric_twist>
                </outer_section>
            </panel>
        </panels>
```

```
        <symmetric_flaps>
            <symmetric_flap id="Inner flap" type="SINGLE_SLOTTED">
                <inner_station_spanwise_position type="PERCENT_SEMISPAN" ref_to="FULL_SEMISPAN">
                    <value>0.15<value/>
                </inner_station_spanwise_position>
                <outer_station_spanwise_position type="PERCENT_SEMISPAN" ref_to="FULL_SEMISPAN">
                    <value>0.45<value/>
                </outer_station_spanwise_position>
                <percent_chord value="40"/>
                <angle_range unit="deg">[0.0,40.0]</angle_range>
            </symmetric_flap>
        </symmetric_flaps>
        <slats>
            <slat linked_to="Inner flap" type="SINGLE">
                <percent_chord value="10"/>
                <angle_range unit="deg">[0.0,10.0]</angle_range>
            </slat>
        </slats>
        <asymmetric_flaps>
            <asymmetric_flap id="Aileron" type="PLAIN">
                <inner_station_spanwise_position type="PERCENT_SEMISPAN" ref_to="FULL_SEMISPAN">
                    <value>0.75<value/>
                </inner_station_spanwise_position>
                <outer_station_spanwise_position type="PERCENT_SEMISPAN" ref_to="FULL_SEMISPAN">
                    <value>0.97<value/>
                </outer_station_spanwise_position>
                <percent_chord value="20"/>
                <angle_range unit="deg">[-5.0,10.0]</angle_range>
                <differential_deflection_factor value="1.0"/>
            </asymmetric_flap>
        </asymmetric_flaps>
    </wing>
</jpad_config>
```

2.3.2 Calculation modules

Currently the software is able to esimate the aircraft weight breakdown, the center of gravity location, calculate some aerodynamic parameters and estimate the performance. All these types of estimates can be usually performed using several analysis methods, comparable and interchangeable. It is also provided a static longitudinal stability analysis, take-off performances and the generation of Payload Range chart.
Future targets for the software are the implementation of landing performances, ultimate the directional static stability

2.3.3 Optimization Process

It has always been the engineer's dream to have all aspects of analysis done in a relatively short time period so that many different configurations can be examined and the best suitable product can be delivered on time. Although this may still be a dream, actual design turn-around time has become shorter due to the use of mathematical optimization techniques which have been introduced into the design process.[79] As all analysis modules inside the JPADCore package will be completed and tested, the final purpose of the code will be to allow users to define a certain numbers of macroscopical geometrical parameters, along with a given objective function, and to receive as output the best set of the previous parameters which suits the wanted objective.

Part II

Development of Application

Chapter 3

Introdution to Java

Language is only the instrument of science,
and words are but the signs of ideas.
– Samuel Johnson

3.1 The java Language

The term Java refers both to a programming language with the higher number of developers in the worldwide and to a technology that has several sub-technologies that have emerged in different fields of use of the software. Javawas developed by Sun Microsystems, a company that was incorporated in Oracle from a few years. In 1996, James Gosling, Bill Joy, and Guy Steele wrote for the First Edition of The Java Language Specification: "*We believe that the Java programming language is a mature language, ready for widespread use. Nevertheless, we expect some evolution of the language in the years to come. We intend to manage this evolution in a way that is completely compatible with existing applications.*"
This programming language is a general-purpose, concurrent, classbased, object-oriented language. It is designed to be simple enough that many programmers can achieve fluency in the language.[77] One design goal of Java is portability, which means that programs written for the Java platform must run similarly on any combination of hardware and operating system with adequate runtime support. This is achieved by compiling the Java language code to an intermediate representation called Java bytecode, instead of directly to architecture-specific machine code. Java bytecode instructions are analogous to machine code, but they are intended to be executed by a virtual machine (VM) written specifically for the host hardware. End users commonly use a Java Runtime Environment (JRE) installed on their own machine for standalone Java applications, or in a web browser for Java applets.[13] Standard libraries provide a generic way to access host-specific features such as graphics, threading, and networking.
There were five primary goals in the creation of the Java language:[61]

1. It must be "simple, object-oriented, and familiar".
2. It must be "robust and secure".
3. It must be "architecture-neutral and portable".
4. It must execute with "high performance".

5. It must be “interpreted, threaded, and dynamic”.

Java SE 8 represents the single largest evolution of the Java language in its history. A relatively small number of features - lambda expressions, method references, and functional interfaces - combine to offer a programming model that fuses the objectoriented and functional styles. Under the leadership of Brian Goetz, this fusion has been accomplished in a way that encourages best practices - immutability, statelessness, compositionality - while preserving “the feel of Java” - readability, simplicity, universality.
Actually Java is the most used programming language according to TIOBE. The TIOBE Programming Community index is an indicator of the popularity of programming languages. The ratings are based on the number of skilled engineers world-wide, courses and third party vendors.

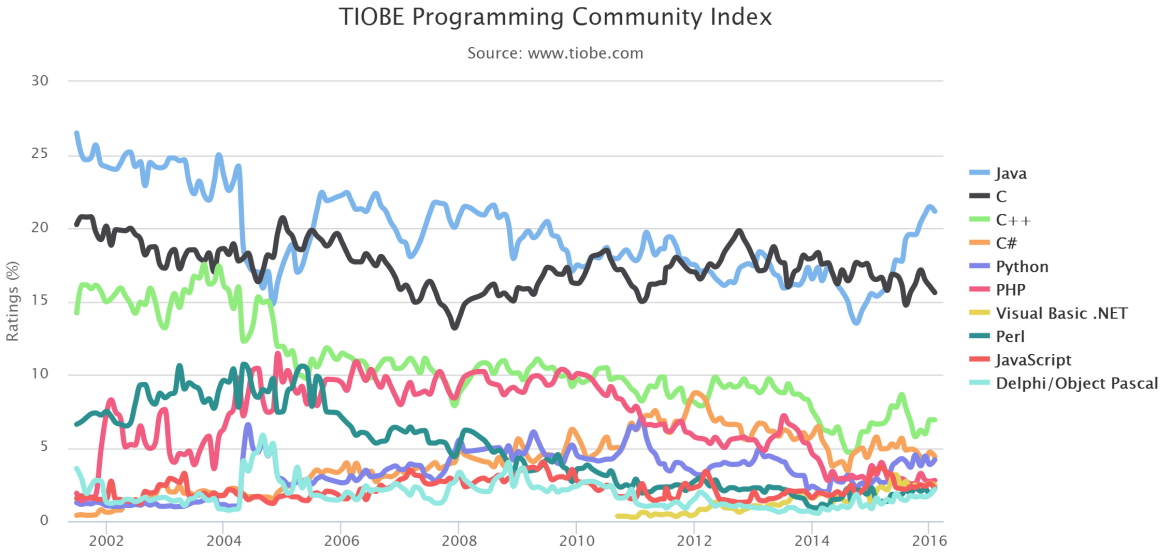


Figure 3.1: Programming Language popularity trade.

3.2 Java choice

The software presented in this thesis work is completely written in Java. The choice of the programming language was driven by several considerations. These include the following:

- the language should be widely supported; this to avoid the case of many valid aircraft design applications and libraries that became obsolete due to the aging of the programming language used to build them;
- the language should promote the use of open source libraries, especially for I/O tasks and for complex mathematical operations;
- the language and the companion Integrated Development Environment (IDE) should provide a widely supported Graphical User Interface (GUI) framework and a GUI visual builder;
- the language should support and promote modularity.

The Java programming language meets all these requirements: it is backed by Oracle and by a huge community of developers so it is continuously updated. Also, advanced and

free IDEs (such as Eclipse) allow programmers to streamline and simplify the development process. In particular, the Eclipse IDE and the SWT/JFace libraries have been chosen to develop JPAD and its GUI. Being Java a pure object oriented programming language, it greatly encourages and simplifies modularization. Each module (package) can be programmed quite independently so that it is relatively easy to divide the work among several programmers. This is essential since the amount of classes and calculations needed to abstract, manage and analyze the entire aircraft is very large (presently the whole project counts more than 56000 lines of code). For such a reason the establishment of common practices and the adherence to fundamental principle of software development (Don't Repeat Yourself, Separation of Concerns, Agile software development) are equally important. An important design requirement of ADOpT is related to its interoperability with other engineering analysis tools. In fact, the application can be easily integrated into a comprehensive aircraft optimization cycle. This is made possible because ADOpT can be launched both in GUI and command line mode. Much care has been given to input/output and configuration files to increase the possible uses of the software.[1]

Chapter 4

Work Object

If the facts don't fit the theory,
change the facts.
– Albert Einstein

4.1 Introduction

In JPAD it is possible to read an .XML file as input or generate an object whose data are written in the code. Both in the first and in the second case all needed variables are initialized with data relating the choosen aircraft. The difference between these two methods is that using an .XML file, user can define its own aircraft having a clear view about the needed data useful for the analysis.
Contrariwise in order to perform test of program functionality, to use a default aircraft is the most simple way to generate a work object.

4.2 Input data from .XML file

XML is a file extension for an *Extensible Markup Language (XML)* file format which is a markup language that defines a set of rules for encoding documents in a format which is both human-readable and machine-readable. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures such as those used in web services. It is defined "Markup Language" due to the use of tags that describes the content. XML is considered extensible because the markup symbols are unlimited and self-defining. So it is possible to use a personal tag for each data. In this way to read an .XML file results relatively simple.[10]
The key concepts of an .XML File Format are the followings:

- markup symbol (tag)
- attribute
- tree structure

As mentioned, each part of the test is contained between an opening **markup symbol** and an end markup symbol that expressed the meaning of the text.

```
<name>Test XML</name>
```

Figure 4.1: Use of markup symbols in XML language.

In addition to tag name, the markup symbols may contain also some **attributes** that introduce more informations such as the unit of measure.

```
<tag attribute1='value' attribute2='value'> text </tag>
```

Figure 4.2: Use of attributes in XML language.

An .XML file has a tree structure where there are external knots that branch into internal knots.

In order to read an XML file it is necessary, first of all, to give the file path. The class `JPADXmlReader` opens the file and the methods of the class `MyXMLReaderUtils` reads the useful data from the XML having the tag path as input. It is possible to read data as `Amount`, namely with units of measurement or as `double`. The unit of measurement is written in the attributes of data in XML file.

Likewise it is possible to write output data on XML file using `JPADDataWriter` class. First of all it is necessary to define and build the xml tree structure. After each variable is associated with a name that is the markup symbol of the XML file.

4.3 Default Aircraft

Actually it is possible to define two different aircraft in order to test the functionality of the application: **ATR-72** and **B747-100B**.

The **ATR 72** is a twin-engine turboprop made by the French-Italian aircraft manufacturer ATR entered service in 1989. It was developed as a variant of the ATR 42 with a 4.5 m stretched fuselage. The ATR 72 was developed from the ATR 42 in order to increase the seating capacity (48 to 66 in standard configuration) by stretching the fuselage, increasing the wingspan, adding more powerful engines, and increasing fuel capacity by approximately 10 percent.[**atr:atr**] It has been typically employed as a regional airliner, although other roles have been performed by the type such as corporate transport, cargo aircraft and maritime patrol aircraft. [7]

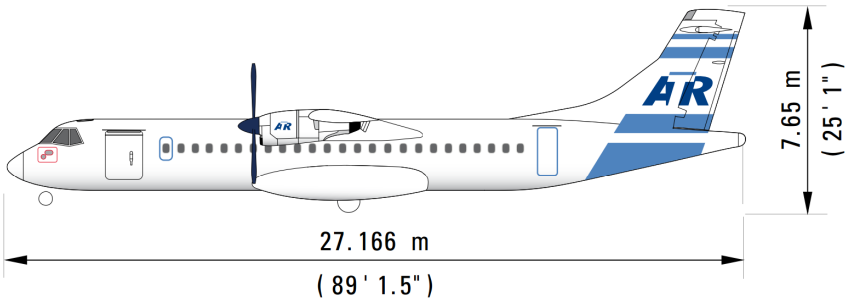


Figure 4.3: ATR 72. Side view.

The **Boeing 747-100B** is a four-engined long-range widebody commercial jet airliner and cargo aircraft produced by the American manufacturer Boeing Commercial Airplanes. It has

a capacity of maximum 480 passengers in a partial double deck configuration. The Boeing 747 It is also known as Jumbo Jet. The basic B747-100 entered service with Pan American On January 15, 1970.
One of the reasons to create the 747 was reductions in airfares with a consequent increase of passenger traffic[8]. The original version of the 747 had two and a half times greater capacity than the Boeing 707, one of the common large commercial aircraft of the 1960s and it was the largest passenger carrier from 1970 until the introduction of Airbus A380.[**boeing:boeing**] The Boeing 747 had two aisles and four wing-mounted engines. The upper deck is its distinctive "hump" along the forward part of the aircraft. It provides space for a lounge or extra seating. The raised cockpit allows front loading of cargo on freight variants.
The 747-100B model was developed from the 747-100SR. This configuration had a typical 452 passengers and unlike the original 747-100, the 747-100B was offered with Pratt & Whitney JT9D-7A, General Electric CF6-50, or Rolls-Royce RB211-524 turbofan engines.

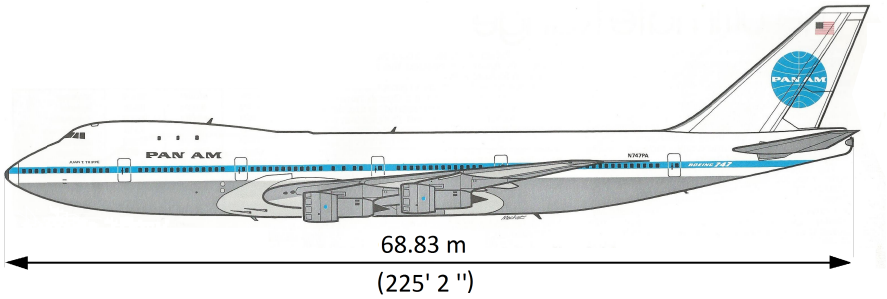


Figure 4.4: Boeing 747-100B. Side view.

4.3.1 How is made a default Aircraft

In order to define a Default Aircraft in a test class, and use it to check the functionalities of the application, it is necessary to follow some step. First of all it is necessary to initialize the working directory tree using the method `initWorkingDirectoryTree` of `MyConfiguration` class located in `JPADConfigs` package that initializes the working directory tree and fill the map of folders. This step is required in order to create the following default folders that are necessary for the right behavior of the code:

- Database directory
- Input directory
- Output directory

Using `MyConfiguration` class it's possible to point at a specific folder, like the input or output directory, with the static method `getDir`. This is a crucial step that must be executed at the beginning of every test. To set the working directory with the useful folders, it's necessary to call the function `initWorkingDirectoryTree()` at the beginning of each test. The function creates all necessary folders. Morover the function has been overloaded and it can be even called with a variable number of arguments (`initWorkingDirectoryTree(String...str)`). These strings are the directory strings in `MyConfiguration` class. The second step is the creation of an`Aircraft` object choosing between the ones grouped into a dedicated `Enum`, named `AircraftEnum`. After it is possible to create an `Aircraft` using the method `createDefaultAircraft` from `Aircraft` class. This method defines a new Aircraft object and invokes another Aircraft's method that creates the components using default data. In the method `createDefaultAircraft` there is a calling to the

builder of `Aircraft` class that initializes the objects of the classes that perform calculations. At this step all the components of the aircraft are created. It is possible also to define new airfoils for the aircraft or change some data from the existing.
Afterwards it is necessary to set the operating conditions such as the number of Mach of analysis or altitude. Each default aircraft has a set of default conditions but the user could change them.
In order to manage all the aircraft related analysis it is necessary to define an object of the class `ACAnalysisManager`. Similarly to the aircraft, exist an analysis manager also for the wing that is an object of the `LSAerodynamicManager` class.
The next step is to define and assign the needed databases. This will be explained in detail in the next section. Finally it is possible to do analysis.

Listing 4.1 Generation of default aircraft

```
public static void main(String[] args) {

        // ------------------------------------------------------------
        // Define directory
        // ------------------------------------------------------------
        MyConfiguration.initWorkingDirectoryTree();


        // ------------------------------------------------------------
        // Generate default Aircraft
        // ------------------------------------------------------------
        Aircraft aircraft = Aircraft.createDefaultAircraft(AircraftEnum..B747_100B);
        LiftingSurface theWing = aircraft.get_wing();

        // Default operating conditions
        OperatingConditions theConditions = new OperatingConditions();


        // ------------------------------------------------------------
        // Define an ACAnalysisManager Object
        // ------------------------------------------------------------
        ACAnalysisManager theAnalysis = new ACAnalysisManager(theConditions);
        theAnalysis.updateGeometry(aircraft);


        // ------------------------------------------------------------
        // Define an LSAerodynamicsManager Object
        // ------------------------------------------------------------
        LSAerodynamicsManager theLSAnalysis = new LSAerodynamicsManager (
                        theConditions,
                        theWing,
                        aircraft
                        );


        // ------------------------------------------------------------
        // Setup database(s)
        // ------------------------------------------------------------

        theLSAnalysis.setDatabaseReaders(
                        new Pair(DatabaseReaderEnum.AERODYNAMIC,
                          "Aerodynamic_Database_Ultimate.h5"),
                        new Pair(DatabaseReaderEnum.HIGHLIFT,
                          "HighLiftDatabase.h5")
                        );


        // ------------------------------------------------------------
        // Do analysis
        // ------------------------------------------------------------
        theAnalysis.doAnalysis(aircraft,
                        AnalysisTypeEnum.AERODYNAMIC);
}
```

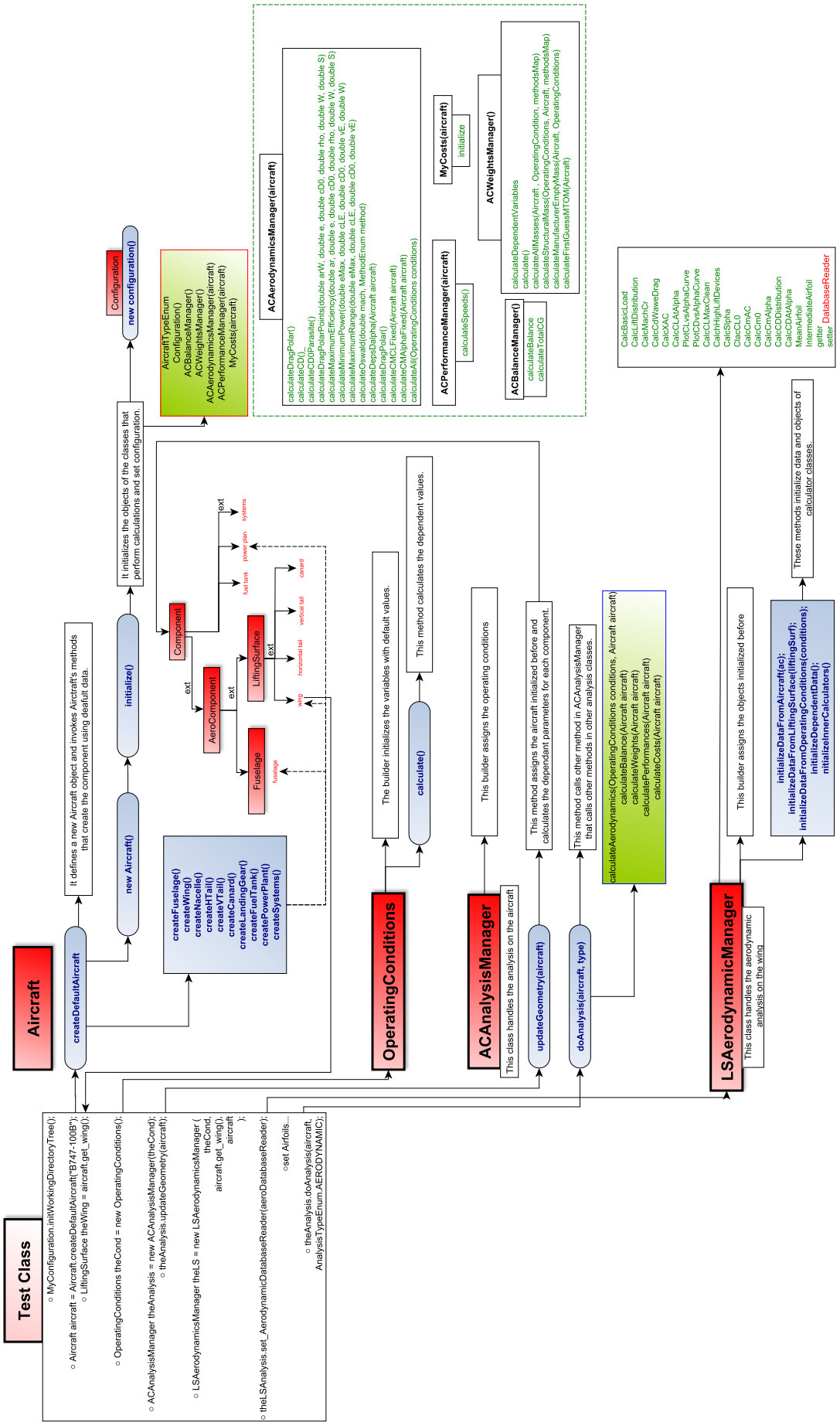


Figure 4.5: Flow chart of the creation of default Aircraft.

4.3.2 How is made a default Wing

Similary to the default aircraft it is possible to define a default wing. This is very useful if the user wants to make an analysis only on a wing. In this case it is necessary to define the origin of the Local Reference Frame (LRF) in Body Reference Frame (BRF) and the coordinates of the Gravity Center (GC).
Contrary to the case of the aircraft, for an isolated wing there isn't necessary to define a fuselage in order to create a `Lifting Surface` object, but there is an overload of the builder that doesn't need a fuselage as input. In this case the exposed surface is calculated as the surface of the wing.

Listing 4.2 Generation of an isolated Wing

```
public static void main(String[] args) {

        // Assign all default folders
        MyConfiguration.initWorkingDirectoryTree();

        // ------------------------------------------------------------------------
        // Coordinates of LRF
        // ------------------------------------------------------------------------

        double xAw = 11.0; //meter
        double yAw = 0.0;
        double zAw = 1.6;
        double iw = 0.0;

        // ------------------------------------------------------------------------
        // Generate default Wing
        // ------------------------------------------------------------------------

        LiftingSurface theWing = new LiftingSurface(
                        "Wing", // name
                        "Data_from_AC_ATR_72_REV05.pdf",
                        xAw, yAw, zAw, iw,
                        ComponentEnum.WING
                        );

        theWing.calculateGeometry();
        theWing.getGeometry().calculateAll();

        // ------------------------------------------------------------------------
        // Center of Gravity
        // ------------------------------------------------------------------------

        double xCgLocal= 1.5; // meter
        double yCgLocal= 0;
        double zCgLocal= 0;

        CenterOfGravity cg = new CenterOfGravity(
                        Amount.valueOf(xCgLocal, SI.METER), // coordinates in LRF
                        Amount.valueOf(yCgLocal, SI.METER),
                        Amount.valueOf(zCgLocal, SI.METER),
                        Amount.valueOf(xAw, SI.METER), // origin of LRF in BRF
                        Amount.valueOf(yAw, SI.METER),
                        Amount.valueOf(zAw, SI.METER),
                        Amount.valueOf(0.0, SI.METER),// origin of BRF
                        Amount.valueOf(0.0, SI.METER),
                        Amount.valueOf(0.0, SI.METER)
                        );

        cg.calculateCGinBRF();
        theWing.set_cg(cg);
        theWing.set_aspectRatio(6.0);

        // Default operating conditions
        OperatingConditions theOperatingConditions = new OperatingConditions();
        theOperatingConditions.set_alphaCurrent(Amount.valueOf(2.0, NonSI.DEGREE_ANGLE)
```

```
        // ------------------------------------------------------------
        // Define an LSAerodynamicsManager Object
        // ------------------------------------------------------------

        LSAerodynamicsManager theLSAnalysis = new LSAerodynamicsManager (
                        theOperatingConditions,
                        theWing
                        );

        // ------------------------------------------------------------
        // Setup database(s)
        // ------------------------------------------------------------

        theLSAnalysis.setDatabaseReaders(
                        new Pair(DatabaseReaderEnum.AERODYNAMIC,
                                        "Aerodynamic_Database_Ultimate.h5"),
                        new Pair(DatabaseReaderEnum.HIGHLIFT, "HighLiftDatabase.h5")
                        );

        // ------------------------------------------------------------
        // Assign Airfoil(s) ...
        // ------------------------------------------------------------

          // Define airfoilRoot...

        // ------------------------------------------------------------
        // Set Airofoil(s)
        // ------------------------------------------------------------
        List<MyAirfoil> myAirfoilList = new ArrayList<MyAirfoil>();
        myAirfoilList.add(0, airfoilRoot);
        myAirfoilList.add(1, airfoilKink);
        myAirfoilList.add(2, airfoilTip);
        theWing.set_theAirfoilsList(myAirfoilList);
        theWing.updateAirfoilsGeometry();
        theLSAnalysis.initializeDependentData();

}
```

4.4 Database in JPAD

In JPAD it is possible to consult external databases in .h5 format. **HDF 5** (Hierarchical Data Format Release 5) is a data file format designed by the *National Center for Supercomputing Applications* (NCSA) to assist users in the storage and manipulation of scientific data across different operating systems and machines.
To obtain the useful data in JPAD interpolating functions are used . These functions can be of one, two or three dimensions and read data from graphics that have been digitize previously. Starting from these digitalizations, databases in .h5 format are built. Reading data from databases is entrusted to methods of classes in the `database` package.
In order to read these databases, and obtain the useful data, it is necessary to define an object of the database reading class and associate it with the object of analysis.
This is a crucial step to read correctly the external data. In fact JPAD allows to work with an aircraft object or only with an isolated lifting surface object. Aircraft is usually composed of a fuselage, lifting surfaces, nacelle and power plant. Furthermore, `Aircraft` and `Wing` are associated with classes of calculation like `LSAerodynamicManager` or `ACAnalysisManager`. So it is necessary that these databases are also visible from these classes.
So because both in aircraft and in wing there is a lifting surface object, databases relative to wing are associated to `LSAerodynamicManager`.

4.4.1 Setup database

Here the database path it's created and associated to object that interpolates the required data from the .h5 file using a `MyInterpolatingFunction` object. After this it's possible to access the double value of the interpolating function using the `standaloneutils` method called `value`.

Now the procedure to assign the database is different if is used an Aircraft object or a Wing object.

4.4.2 Assign database using an Aircraft object

In order to assign correctly the database and associate it to all analysis management is necessary to practise the following order.

1. Define an Aircraft Object.
   This command associates to Aircraft an object that defines the aerodynamic. From the wing it is possible to obtain the Wing, that is a `LiftingSurface` object.
2. Define an `ACAnalysisManager` object.
   All the aircraft computations are managed by this class.
3. Define an `LSAerodynamicManager` object.
   All the lifting surfaces computations are managed by this class.
4. Associate database to `LSAerodynamicManager`.
5. Eventually do analysis.

4.4.3 Assign database using a Wing object

Using a Wing object it isn't necessary to define a manager for Aircraft aerodynamic analysis. So the step to follow are the same of aircraft starting from the third.

1. Define a Wing Object.
2. Define an `LSAerodynamicManager` object.
3. Associate database to `LSAerodynamicManager`.

The definition of an isolated Wing is explained in the relative section.

Listing 4.3 Assign database using an Aircraft object

```
// ------------------------------------------------------------
// Setup database(s)
// ------------------------------------------------------------

theLSAnalysis.setDatabaseReaders(
                new Pair(DatabaseReaderEnum.AERODYNAMIC,
                  "Aerodynamic_Database_Ultimate.h5"),
                new Pair(DatabaseReaderEnum.HIGHLIFT,
                  "HighLiftDatabase.h5")
                );
```

The databases are assigned to `LSAerodynamic` using a method of this class. This method accept as input a variable number of `Pair` objects.Using `Pair` objects it is possible to assign, for each database, both name and type.

Listing 4.4 `setDatabaseReaders` method

```
public void setDatabaseReaders(Pair... args) {
        String databaseFolderPath = MyConfiguration.getDir(FoldersEnum.DATABASE_DIR);

        for (Pair a : args) {
                DatabaseReaderEnum key = (DatabaseReaderEnum)a.getKey();
                String databaseFileName = (String)a.getValue();

                switch (key) {
                        case AERODYNAMIC:
                                _aerodynamicDatabaseReader =
                                new AerodynamicDatabaseReader(
                                                databaseFolderPath,
                                                databaseFileName);
                                listDatabaseReaders.add(_aerodynamicDatabaseReader);
                                break;

                        case HIGHLIFT:
                                _highLiftDatabaseReader =
                                new HighLiftDatabaseReader(
                                                databaseFolderPath,
                                                databaseFileName);
                                listDatabaseReaders.add(_highLiftDatabaseReader);
                        break;

                }
```

4.4.4 User's guide

In order to execute some analysis in JPAD it is necessary, first of all, to define an analysis object in the Test class. The method `createDefaultAircraft` creates a new aircraft and the object that composes it. This method also populates the data of aircraft with default value corresponding to ATR-72 or Boieng 747_100B. Moreover the method `createDefaultAircraft` calls another method in `Aircraft` class: `initialize` that initializes the objects of the classes that perform calculations.
The purpose of this structure is to have only a way to assign the databases at an aircraft. Inasmuch as the wing is always present, the chosen strategy is to assign the database to the aerodynamic manager of the wing.
In order to bring to use the database also for the aircraft calculation, it is assigned at the aerodynamic manager of the aircraft in the method called `doAnalysis`.
At the same time `LSAerodynamicManager` sets itself as aerodynamic in the wing object. So it is possible to call the database using equally the following codes:

- `theWingObject.getAerodynamics.get_Database;`
- `theAircraftObject.get_theAerodynamic.get_Database;`
- `theLSManagerObject.get_Database;`
- `theACManagerObject.get_Database;`

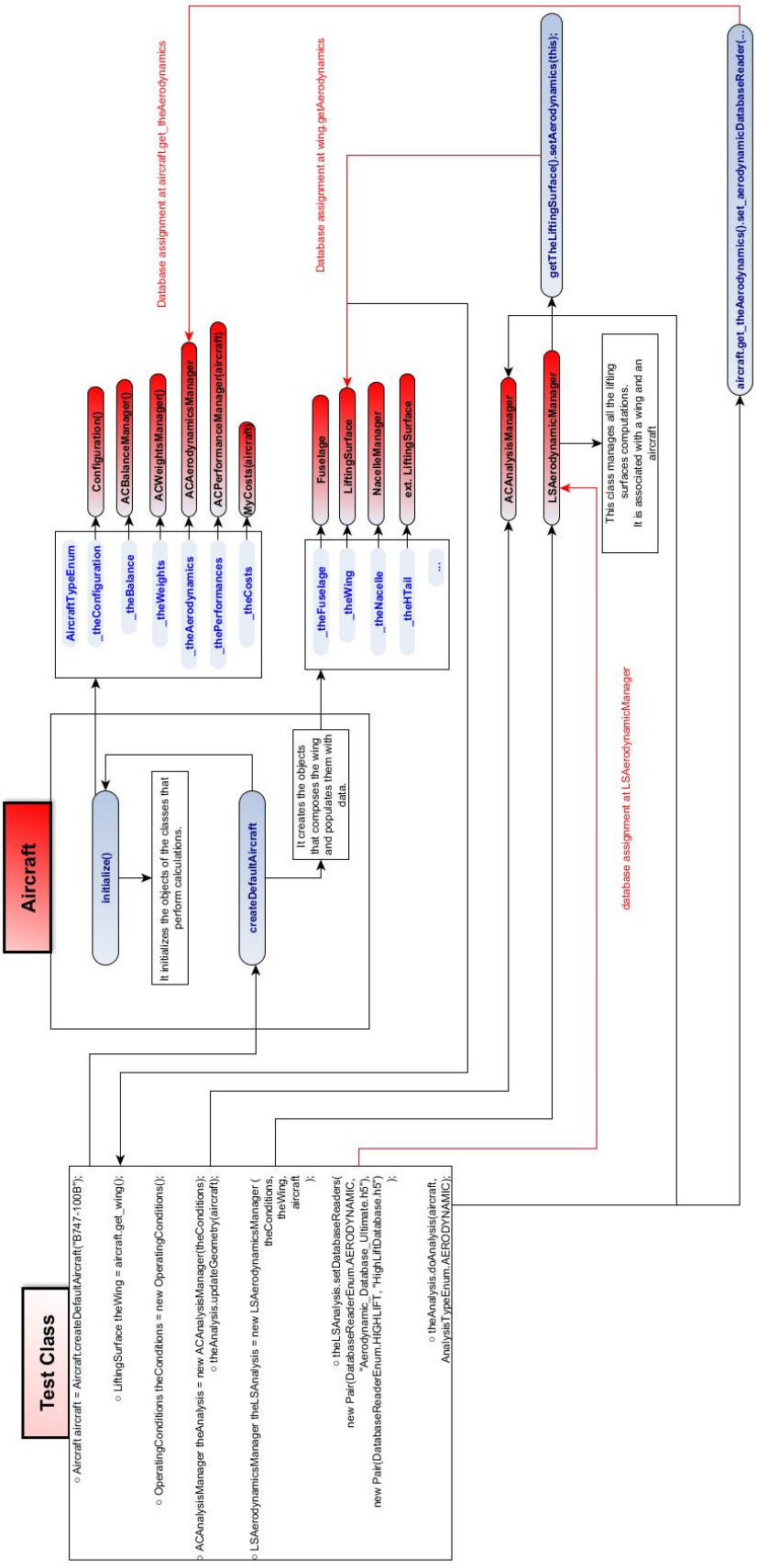


Figure 4.6: Flow chart of database assignment.

Part III

Functionality Overview

Chapter 5

Work Object Data

It is not certain
that everything is uncertain
– Blaise Pascal

All the following analysis will be carried out using a default aircraft. This choice is made in order to avoid the collection and validation data phase for each analysis and make focus on the results. As mentioned, there are two default aircraft in the code: ATR-72 and Boieng 747_100B whose data are shown in the table below.
All the analysis that follow, refer to the data presented in this chapter and those derived from them .

	ATR-72	**B747-100B**
Operating Conditions		
Altitude	6000.0000 m	10 000.0000 m
Cruising Mach number	0.4300	0.8300
Weights		
Maximum Take-Off Mass	23 063.5790 kg	354 991.5060 kg
Operating Empty Mass	12 935.5790 kg	153 131.9860 kg
Maximum Fuel Mass	5000.0000 kg	147 409.5200 kg
Maximum number of passengers	72	550
Fuselage		
Length	27.1660 m	68.6300 m
Diameter	2.7561 m	6.7934 m
Fineness Ratio	9.8566	10.1025
Wing		
Surface	61.0000 m^2	511.0000 m^2
$\mathcal{R}$	12.0000	6.9000
Span	27.055 49 m	59.3792 m
Taper Ratio	0.5450	0.2840
Root Chord	2.9186 m	14.6152 m
Mean Aerodynamic Chord	2.3198 m	9.6913 m
Sweep_{LE}	2.8390°	38.4290°

continued on next page

(*continued from previous page*)

	ATR-72	**B747-100B**
$\text{Sweep}_{C/4}$	1.3997°	35.4999°
t/c_{max}	0.1675	0.1292
C_{D0}	0.03170	0.01820
Oswald Factor	0.7585	0.6277
Airfoil type	Conventional	Modern Supercritical
Horizontal Tail		
Surface	11.7300 m^2	136.6000 m^2
$\mathcal{AR}$	4.5550	3.5700
Span	7.3095 m	22.083 07 m
Taper Ratio	0.5700	0.2650
Root Chord	2.044 30 m	9.7798 m
Mean Aerodynamic Chord	1.6450 m	6.8835 m
Sweep_{LE}	3.4410°	38.2250°
$\text{Sweep}_{C/4}$	0.0000°	32.0003°
t/c_{max}	0.1500	0.1500
Vertical Tail		
Surface	12.4800 m^2	77.1000 m^2
$\mathcal{AR}$	1.6600	1.3400
Span	4.5515 m	10.1643 m
Taper Ratio	0.3200	0.3300
Root Chord	4.1544 m	11.4065 m
Mean Aerodynamic Chord	2.9323 m	8.2285 m
Sweep_{LE}	40.5485°	53.9913°
$\text{Sweep}_{C/4}$	28.5998°	45.0001°
t/c_{max}	0.2227	0.2245
Power Plant		
Static Thrust (per engine)	7700.0000 N	204 000.0000 N
Engines Number	2	4
Engines type	Turboprop	Turbofan
BRR	0.0000	5.0000
η_p	0.85	0.00

Table 5.1: ATR-72 and B747-100B input data

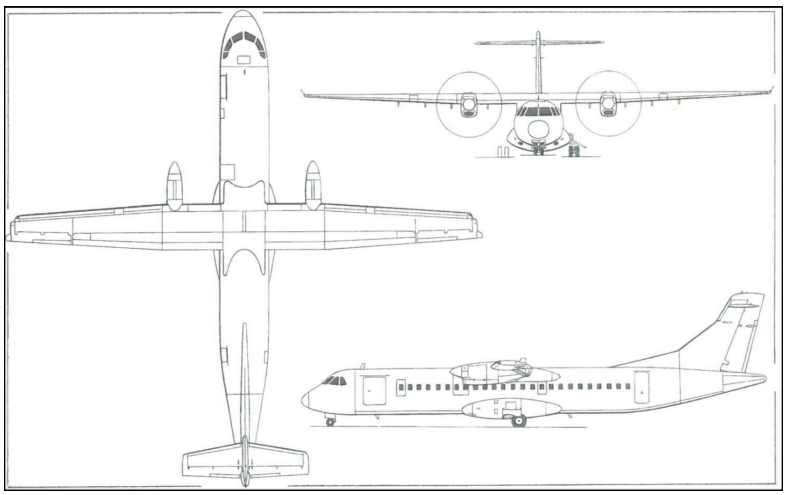


Figure 5.1: ATR-72 views – Jane's All the World's Aircraft 2004-2005.

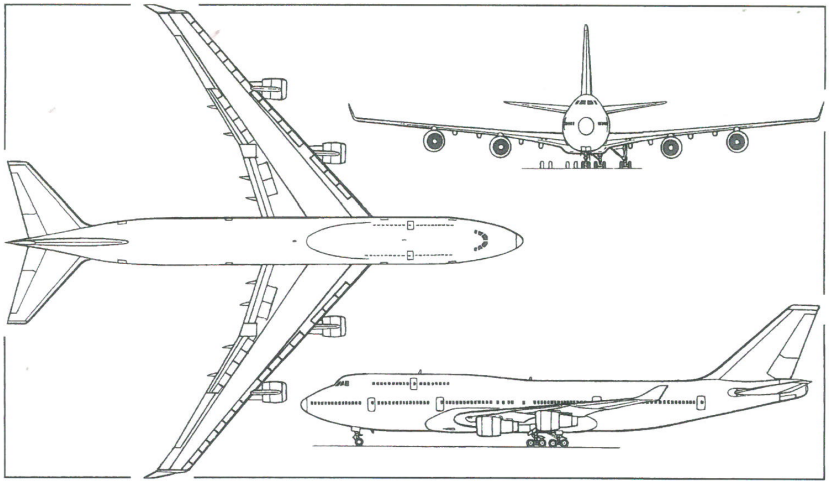


Figure 5.2: B747-100B views – Jane's All the World's Aircraft 2004-2005.

Chapter 6

Wing Lift Characteristics

C'è qualcuno che può rompere il muro del suono,
mentre tutto il mondo si commenta da solo.
– L. Ligabue

Any body in motion in a fluid presents a result of force acting on it, which can be decomposed in two components:

- A **Lift** acting normal to the Velocity direction and is positive upward.
- A **Drag** acting in the opposite direction to the airspeed vector.

The lifting surfaces of an airplane are designed to generate lift exceeding their drag, in order to obtain a positive efficiency.
In this chapter it will be explained how it's possible to evaluate the complete lift coefficient curve of a wing, both in its linear trait and non linear.
The lift coefficient is a dimensionless number used to model all of the complex dependencies of shape, inclination, and some flow conditions on lift. The lift coefficient also contains the effects of air viscosity and compressibility that are molded respectively with the Raynolds and Mach number.

6.1 Theoretical background

In order to achieve the complete curve of coefficient list it's necessary to obtain the following parameters:

1. Zero-Lift Angle
2. Lift Coefficient slope
3. End of Linearity Angle
4. Maximum Lift Coefficient
5. Stall Angle of attack

Following it will be explain the theory behind how these contributions have been made.

6.1.1 Zero-Lift Angle and Lift Coefficient Slope

In order to evaluate the linear trait of the wing lift curve, it's been used the Nasa-Blackwell method.
The **Nasa-Blackwell** is a numerical method for calculating the subsonic load distribution for arbitrary lifting surface arrangements at a fixed angle of attack. The method is suitable for swept wings, non-planar wings, wing with pylons and/or end-plates; it can be also used to study the aerodynamic interaction of the wing and the horizontal tail. The method has been implemented because its results, as shown in the report, were in good agreement with the experimental results. The lifting surfaces are divided in several rectangular horse-shoe vortices along the span; one horseshoe vortex along the chord is used, that is, the midpoints of the vortices are placed only at points along the quarter-chord lines. An equal number of control points are located along the three-quarter-chord lines. The velocity from the total vortex system is equated to the component of free-stream velocity normal to the lifting surface chord at each control point. Application of this tangent-flow boundary condition for a symmetrical loading provides a set of N simultaneous equations in the N unknown circulation strengths. Solution of this set of equations provides the loading distributions over the lifting surfaces. Mach number effect is introduced through a Prandtl-Glauert correction. Further details can be found in [23].

In this way it's calculated the Zero Lift angle using the integral of the load distribution and the linear trait slope starting from the value of CL at $\alpha = 0°$ and $\alpha = 2°$.

$$CL_\alpha = \frac{CL|_2 - CL|_0}{\Delta\alpha} \tag{6.1}$$

In JPAD it is possible also evaluating these two contributes using different methods.
For the evaluation of the α_{0L} it's possible to use a method of the class `CalcAlpha0L`, a nested class in `LSAerodynamicManager`. Using these method, the zero-lift angle is calculated with the following formula, where ϵ_T is the twist angle:

$$\alpha_{0L} = \frac{1}{S}\int_{-\frac{b}{2}}^{\frac{b}{2}} c(y)[\alpha_0(y) - \epsilon_T(y)]\,dy \tag{6.2}$$

This formula is applied to the exposed wing. When a default aricraft is defined, during the creation of components, is calculated the exposed wing. This is the wing outside of the fuselage. It starts at semi-diameter of fuselage and its root airfoil is the airfoil of wing at this station calculated by the method `calculateIntermediateAirfoil`.

In order to evaluate the lift coefficient linear slope it's possible to use different method belonging to `CalcCLAlpha` class. The first one use the Polhamus formula which is valid for arbitrary aspect ratios and sweep angles in subsonic flow:

$$CL_\alpha = \frac{2\pi\, AR}{\left\{2 + \sqrt{\frac{AR^2\beta^2}{k^2}\left(1 + \frac{\tan^2(\Lambda_{\frac{c}{2}})}{\beta^2}\right) + 4}\right\}} \tag{6.3}$$

Thus CL_α is a function of wing aspect ratio, mid-chord sweep angle $\Lambda_{\frac{c}{2}}$, Mach number, and airfoil section (defined parallel to the free stream) lift curve slope. The factor K in the equation is the ratio of the experimental two-dimensional lift curve slope.
Alternately it's possible to use the Anderson formula for swept wing, compressible and

subsonic flow:

$$CL_\alpha = \frac{Cl_\alpha \cos \Lambda_{\frac{c}{2}}}{\sqrt{1 - M^2 cos^2 \Lambda_{\frac{c}{2}} + \left[\frac{Cl_\alpha \cos \Lambda_{\frac{c}{2}}}{\pi \mathcal{R}}\right]^2} + \frac{Cl_\alpha \cos \Lambda_{\frac{c}{2}}}{\pi \mathcal{R}}} \tag{6.4}$$

6.1.2 End of Linearity angle of attack

The angle of end linearity is calculated using the characteristics of the mean airfoil of the wing. This is obtained through the influence area of the airfoils as shown in fig. 6.1.

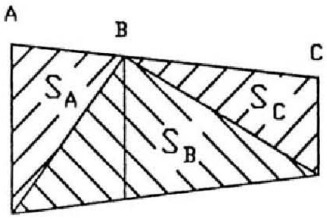


Figure 6.1: Influence area of the sections for finite wing.

Then is possible to calculate the influence coefficients.

$$K_i = \frac{2S_i}{S} \tag{6.5}$$

The mean parameters can be obtained from:

$$\overline{x} = x_1 K_1 + x_2 k_2 + x_3 k_3 ... \tag{6.6}$$

In particular, for a wing defined by three airfoils, the end of linearity angle of attack is obtained from the following equation:

$$\alpha^* = \alpha_1^* K_1 + \alpha_2^* k_2 + \alpha_3^* k_3 \tag{6.7}$$

6.1.3 Maximum Lift Coefficient

In order to estimate the maximum lift coefficient for a clean wing it's been used the Nasa-Blackwell method.

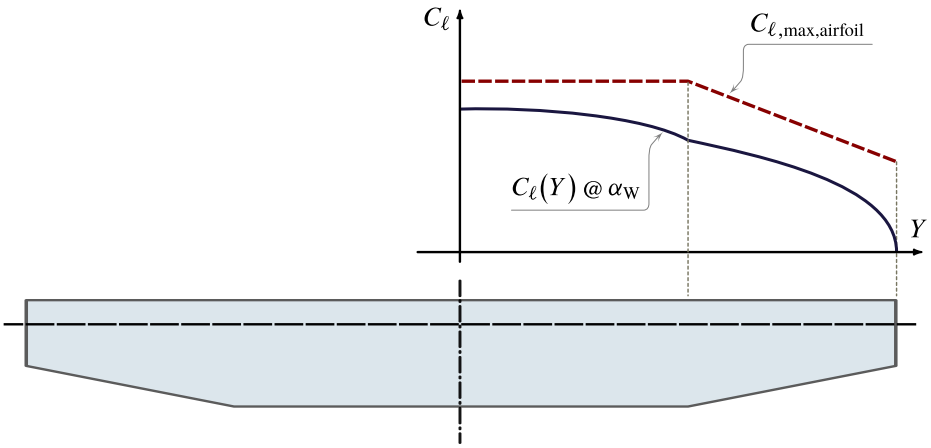


Figure 6.2: Determination of wing lift distribution and Cl distribution of airfoils.

The practiced procedure is the following:

1. For each value of an Alpha array the load distribution is calculated using the NasaBlack method. The distribution of Cl_{max} is known. fig. 6.3.
2. At $\alpha = \alpha_j$ the load distribution curve intersects for the first time the Cl_{max} curve of the airfoils. fig. 6.3.
3. For each $y > y_1$, along y axis (where y_1 is the station of first intersection) is evaluated the difference between CL_{wing} and Cl_{max}. fig. 6.4.
4. $\Delta\alpha$ is evaluated until the maximum difference between CL_{wing} and Cl_{max} is smaller than the required accuracy. Actually the accuracy is 0.0001.

The evauation of $\Delta\alpha$ is not simple.
After finding the first value of intersection between CL_{wing} and Cl_{max} a new α is evaluated. The first value used for the optimization is

$$\Delta\alpha = \alpha_j - \alpha_{j-1}$$
$$\alpha_{new} = \alpha_j - \frac{\Delta\alpha}{2}$$
$$\alpha_{old} = \alpha_j$$

For the following step if there isn't a point of intersectionat α_{new} the new value of α is, for the step $j + 1$:

$$\Delta\alpha = |\alpha_{j-1} - \alpha_j|$$
$$\alpha_{new} = \alpha_j + \frac{\Delta\alpha}{2}$$
$$\alpha_{old} = \alpha_j$$

Instead, if there is a point of intersectionat α_{new} the new value of α is, for the step $j + 1$:

$$\Delta\alpha = |\alpha_{j-1} - \alpha_j|$$
$$\alpha_{new} = \alpha_j - \frac{\Delta\alpha}{2}$$
$$\alpha_{old} = \alpha_j$$

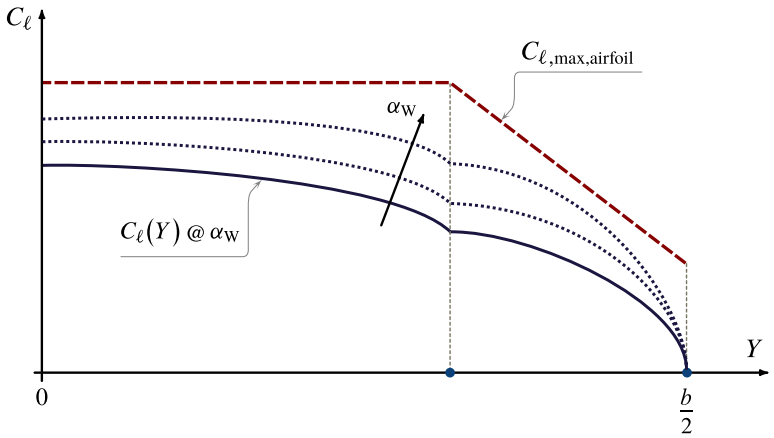


Figure 6.3: Determination of wing lift distribution and Cl distribution of airfoils.

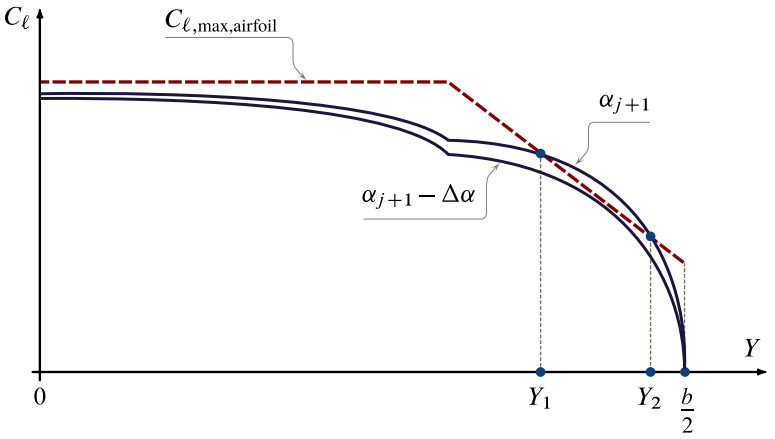


Figure 6.4: Intersection between CL_{wing} and Cl_{max} curves.

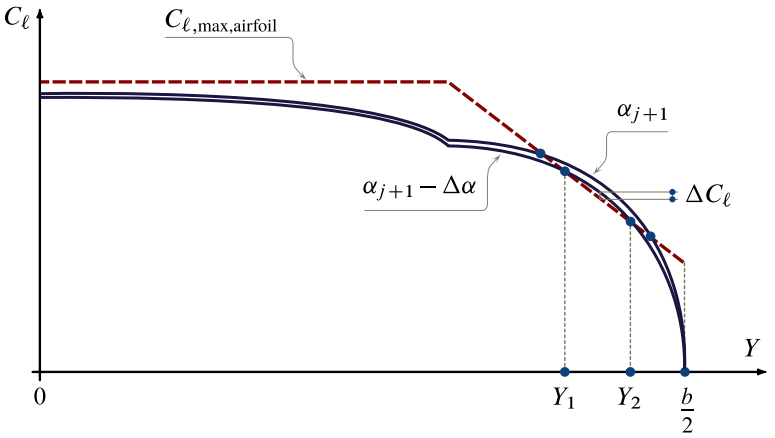


Figure 6.5: Determination of $\Delta\alpha$.

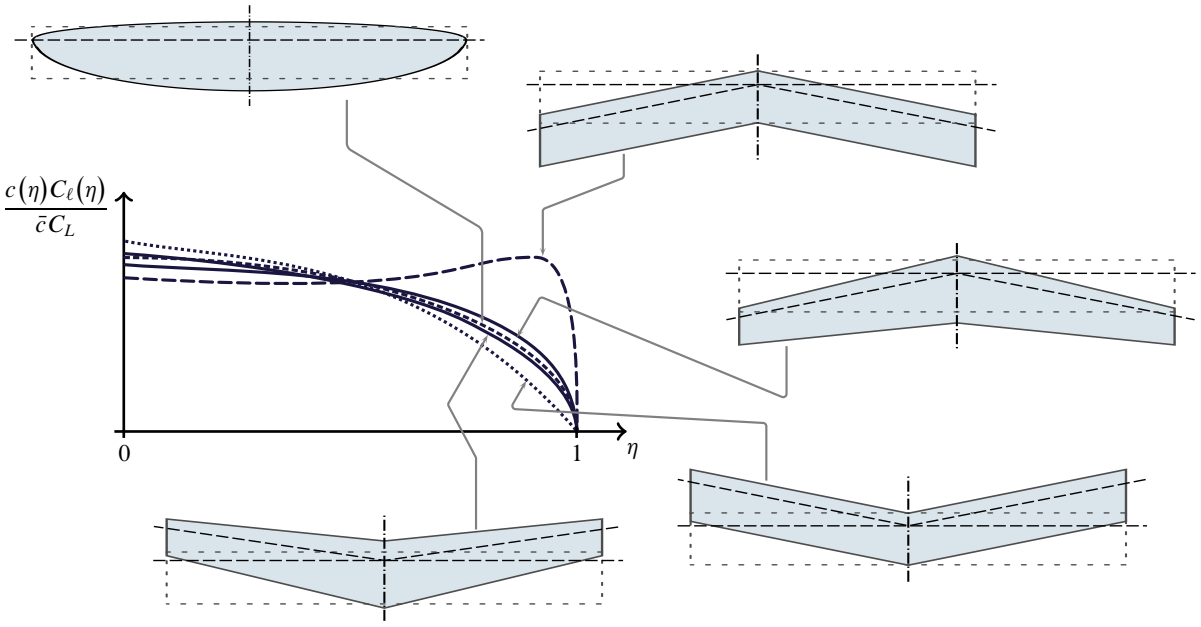


Figure 6.6: Wing Loading distribution.

6.1.4 Stall Angle of Attack

Nasa-Blackwell is an inviscid method for calculating the subsonic aerodynamic load distributions for arbitrary lifting-surface arrangements. This means that it doesn't see the non linear trait of the lift curve. So in correspondence of the value of $C_{L_{MAX}}$ calculated, the angle of attack obtained from the Nasa-Blackwell method is not correct because it is the maximum angle of attack that you would get if you reach the max C_L linearly.
Obtained the maximum cl is therefore necessary to calculate the $\Delta\alpha$ from the following graph. So first it's evaluated the angle of attack at maximum lift coefficient through the linear trend of the lift line, then it's added to this angle the increment evaluated by the diagram in fig. 6.7. , this is valid strictly for wing with high taper ratio without twist, with unique airfoil type and Mach number included between 0.2 and 0.6.

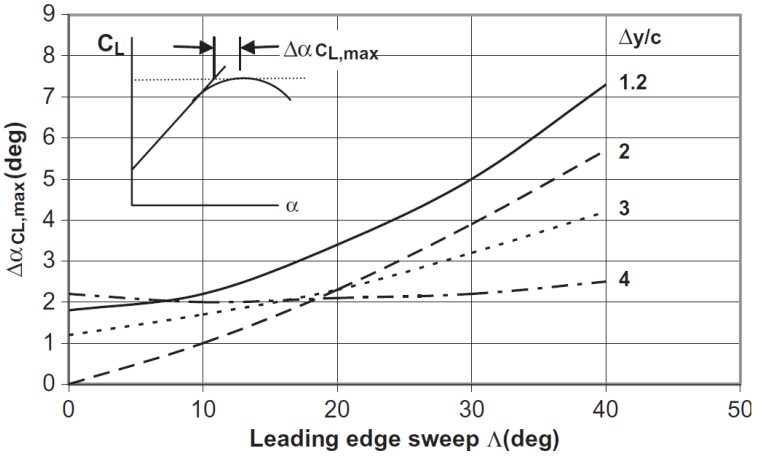


Figure 6.7: Diagram useful for evaluation of $\Delta\alpha$.

6.1.5 Construction of the curve

At this point all elements are available in order to draft the CL vs α curve.
The linear trait is evaluated using the equation of straight line.

$$C_L = C_{L_\alpha}\alpha + C_{L_0} \tag{6.8}$$

In order to plot the non linear trait is used a cubic function. In fact in this zone we have four conditions:

- Pass to $\alpha^\star$, $C_L^\star$
- The derivatine in $\alpha^\star$, $C_L^\star$ is C_{L_α} for continuity
- Pass to α_{stall} , $C_{L_{MAX}}$
- The derivatine in α_{stall} , $C_{L_{MAX}}$ is 0.Here there is the maximum of the curve.

So the system of equations is :

$$\begin{cases} C_L^\star = a{\alpha^\star}^3 + b{\alpha^\star}^2 + c\alpha^\star + d \\ C_{L_\alpha} = 3a{\alpha^\star}^2 + 2b\alpha^\star + c \\ C_{L_{MAX}} = a{\alpha_{stall}}^3 + b{\alpha_{stall}}^2 + c\alpha_{stall} + d \\ 0 = 3a{\alpha_{stall}}^2 + 2b\alpha_{stall} + c \end{cases} \tag{6.9}$$

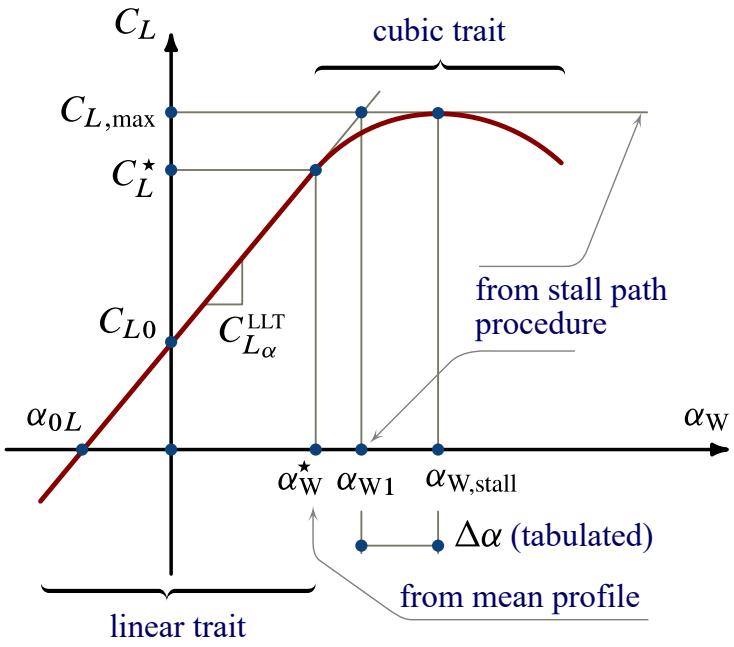


Figure 6.8: Wing C_L vs α curve.

6.1.6 Mach number effects on Lift curve

The characteristics of airflow depend heavily from the Mach number. These changes in airflow have a significant effect on the airplane lift. For airplanes that operate entirely within the subsonic speed range, there are no significant effects of compressibility of the air on the airplane lift. For airplane that operate at high subsonic speeds in the transonic speed region, the airplane lift and drag curves will vary as the flight Mach number is increased due to the compressible nature of the air, so there is a family of lift curves one for each flight Mach number of interest.
The family of lift curves is characterized by an increase in the slope of the curve and decrease in maximum lift coefficient as the Mach number increased in the high subsonic region as shown in the fig. 6.9. The explanation of these Mach number effects on the lift curve has been derived from the theory of compressible flow, and confirmed by experimental data obtained in wind tunnels and from flight tests. It can be shown that, for an airplane at given angle of attack, the lift coefficient will increase with Mach number because the suction on the wing upper surfac, and the pressures on the wing lower surface tend to grow with Mach number. [45]

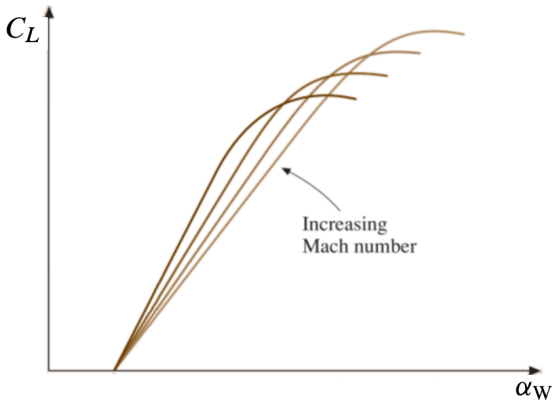


Figure 6.9: Mach number effects on Lift curve.

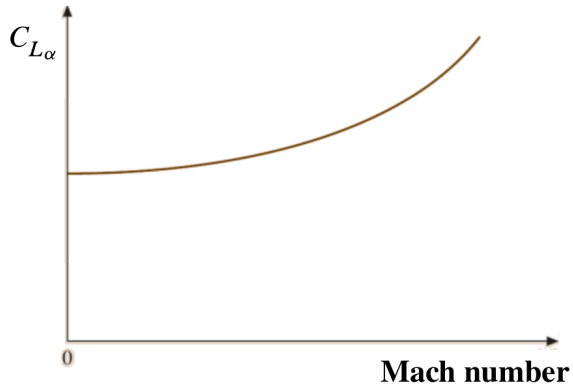


Figure 6.10: Mach number effects on Lift linear slope.

6.1.7 High Lift devices

The aircraft's performances at low speed are very important for their mission, infact good high lift devices allow to land or take off on many airports, with their various runways.[28] To assure high lift in take-off and landing without varying weight and wing surface, it's varied the airfoil shape, for this reasons high-lift systems are used. High-lift systems are commonly defined as the devices that allow to increase the maximum lift coefficient of the aircraft and consequently to decrease the stalling speeds. A great variety of high lift devices exists, many of which are mechanically complex. The simplest systems change only the camber of the aerofoil. More complex concepts not only change camber but also extend the chord, opening up slots as they do so. [40]
High-lift devices can be classified in two main categories: trailing edge devices, or leading edge devices. TE devices are small aerofoil-like elements that are fitted at the trailing edge of the wing that are called flap, while at the LE as a slat.

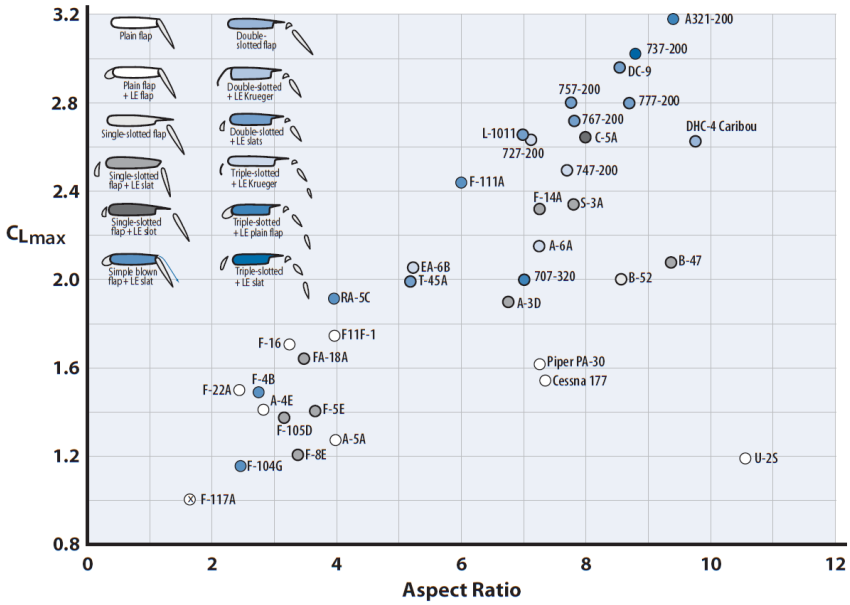


Figure 6.11: Different kind of High Lift devices and their effects.

In order to evaluate the new wing lift curve with flap and slat it's necessaty to evaluate the contribution due to these devices starting from the clean C_L vs α curve. In particular these contribution on the lift curve are the following:

- ΔC_{L_0} flap
- $\Delta C_{L_{MAX}}$ flap
- CL_α flap
- $\Delta\alpha_{MAX}$ flap
- $\Delta C_{L_{MAX}}$ slat
- $\Delta\alpha_{MAX}$ slat

All of these parameters are calculated using formulas or graphs. For further details see [58]

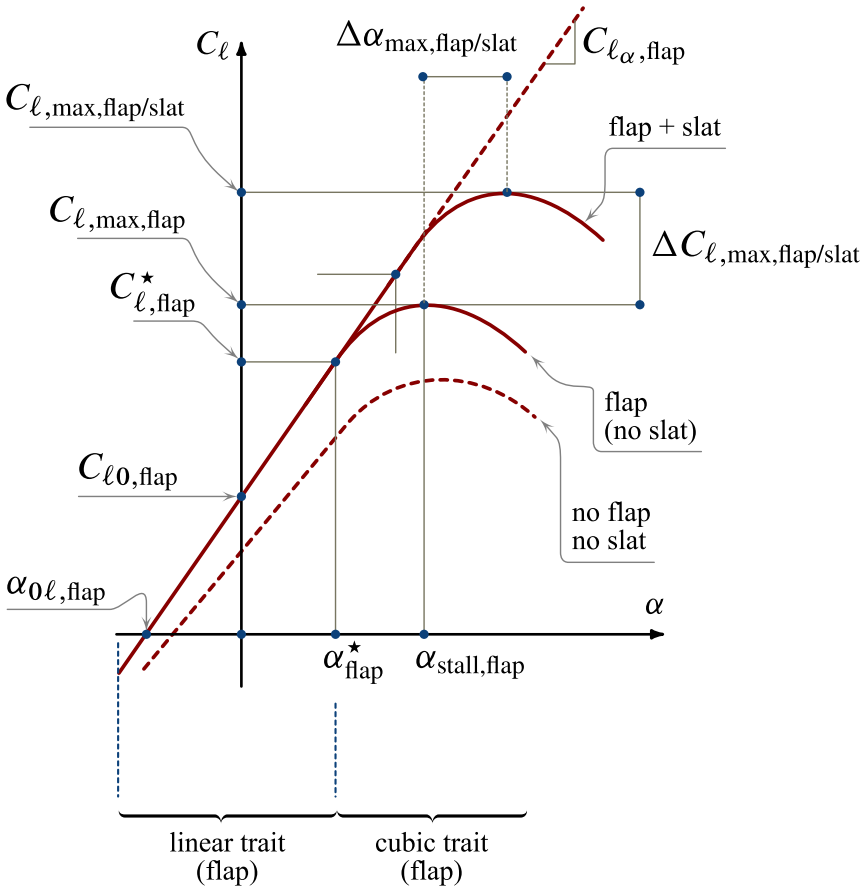


Figure 6.12: Wing C_L vs α curve for clean wing and with high lift devices.

6.2 Java Class Architecture

6.3 Case Study

Chapter 7

Wing Drag Characteristics

Le avversità
possono essere delle formidabili occasioni
– Thomas Mann

As mentioned in the previous chapter, the drag is the force component acting in the opposite direction to the airspeed vector.
There isn't a single classification of the drag but, dependent on the purpose of the work, the drag may be broken down in different way. Following will be explained the two main classifications.

- The drag is subdivided using a causal breakdown. In this way the drag contributes are in accordance with the physical mechanism such as the viscosity of the flow.
- The drag is subdivided using a component breakdown. Every component of aircraft added an own drag contribute.

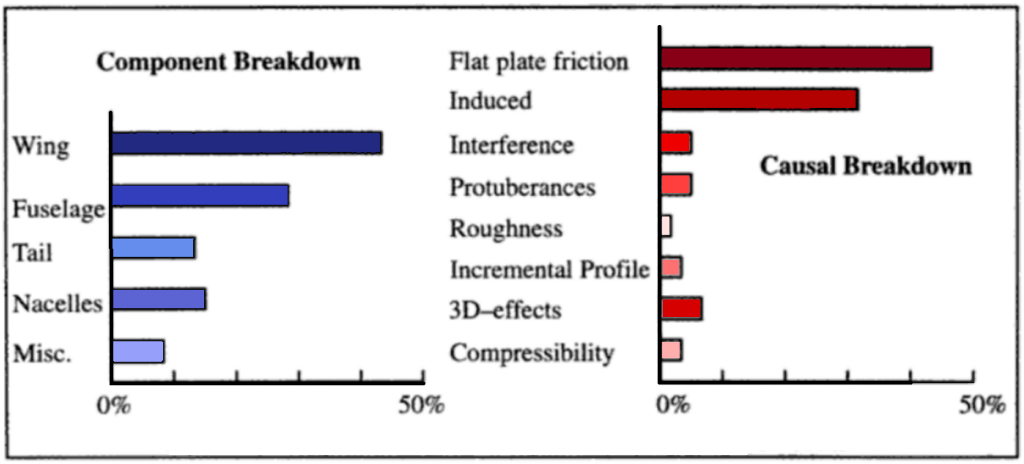


Figure 7.1: Drag breakdown for a business jet in cruise.

According to the casual breakdown it's possible to make a preliminary division considering normal and tangential stress. The tangential forces produce the **friction drag**. While it's possible to divide the drag due of the normal component in viscous, that generates **form drag**,

and inviscid. A further division can be made for the last one, in **induced drag** and **wave drag**.

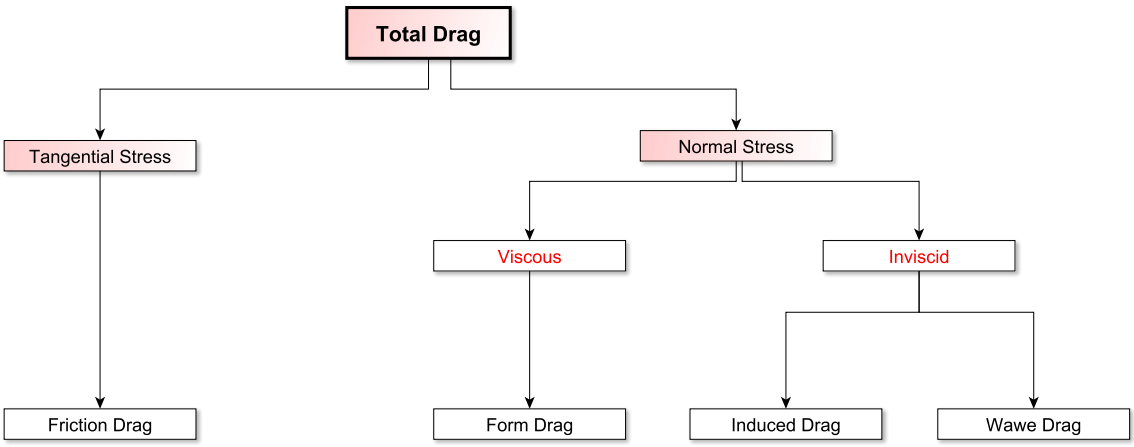


Figure 7.2: Causal breakdown of airplane drags.

Friction drag is caused by the air closest to the body's surface that is dragged along with it. Due to this interaction shearing stresses are born within the thin layer of air (boundary layer) adiacent to the skin. The magnitude of this drag depends on the kind of boundary layer what can be laminar or turbulent in dependence on the Raynolds number. Usually it's accustomed to assume, at the flight speed and altitudes at which aircraft fly, a fully turbulent flow over the entire airplane. In this way a conservative result is obtained.
Form drag is caused by the air that is flowing over the body. The separation of air creates turbulence and results in pockets of low and high pressure that leave a wake behind the airplane. The departure of the boundary layer alters the pressure field from its inviscid distribution resulting in an additional drag component. The general size and shape of the body are the most important factors in form drag; bodies with a larger presented cross-section will have a higher drag than thinner bodies.
Induced drag is the drag due to lift. It is the drag created by the vortices at the tip of an aircraft's wing. The high pressure underneath the wing causes the airflow at the tips of the wings to curl around from bottom to top in a circular motion. So it's depend on the spanwise distribution of lift.
Wave drag is a component of the drag due to the presence of shock waves. Wave drag is independent of viscous effects, and tends to present itself as a sudden and dramatic increase in drag as the vehicle increases speed.

In this work the drag will be classified using a component breakdown. In this way it's possible to evaluate separately the wing drag and tail drag, considering the aerodynamic centers of these lifting surfaces as application point.

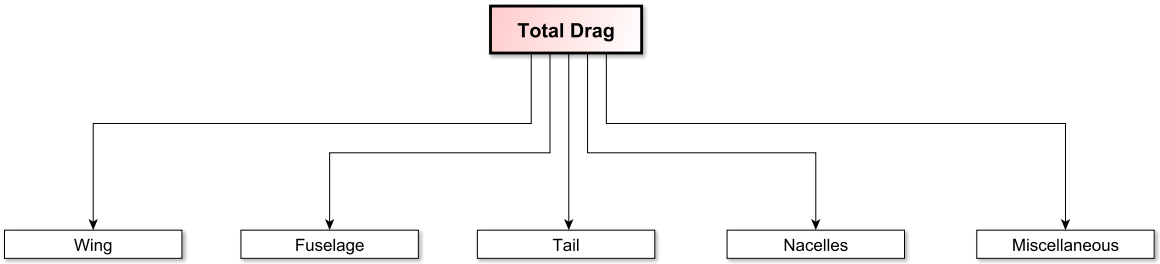


Figure 7.3: Components breakdown of airplane drags.

7.1 Theoretical background

In this thesis the drag coefficient of an isolated lifting surface is calculated starting from the load distribution and considering the parabolic approximation for the drag polar. The implemented process is the following:

1. First of all the load distribution at a given angle of attack has been calculated
2. Fifty points along the semispan are defined
3. For each point, an intermediate profile is calculated
4. To each profile correspond the lift coefficient at that station
5. Starting from the CL, using a parabolic approximation for the drag polar, the CD is calculated with the following equation

$$CD = CD_{min} + (CL - CL_{CD_{min}})^2 + k \tag{7.1}$$

6. Known the drag distribution it is possible to calculate the drag coefficient of the lifting surface integrating

In the tool it is possible to choose the method used to calculate the load distribution. In particular it's possible to use Shrenk or Nasa-Blackwell method.

Schrenk Method is based on an elliptical lifting coefficient distribution span wise hypothesis on the wing. This method also assumes that the pressure distribution is proportional to the wing area.

The **Nasa-Blackwell**, as mentioned, is a numerical method for calculating the subsonic load distribution for arbitrary lifting surface arrangements at a fixed angle of attack. 6

7.2 Java Class Architecture

In order to give more flexibility to the code, the calculation of drag coefficient is made from three different class summarized in the folloving table.

`integralFromCdAirfoil`	This method calculates the drag coefficient of the lifting surface using an integral and calling other method that fills the needed field. This is in the nested class `CalcCDAtAlpha`
`nasaBlackwell`	This method calculates drag distribution starting from the load distribution calculated with Nasa-Blackwell method and using a parabolic approximation for the drag polar. This is in the nested class `CalcCdDistribution`
`nasaBlackwell`	This method calculates the drag coefficient of an airfoil having the lift coefficient and using a parabolic approximation.

Table 7.1: Methods for drag coefficient calculatorof a lifting surface.

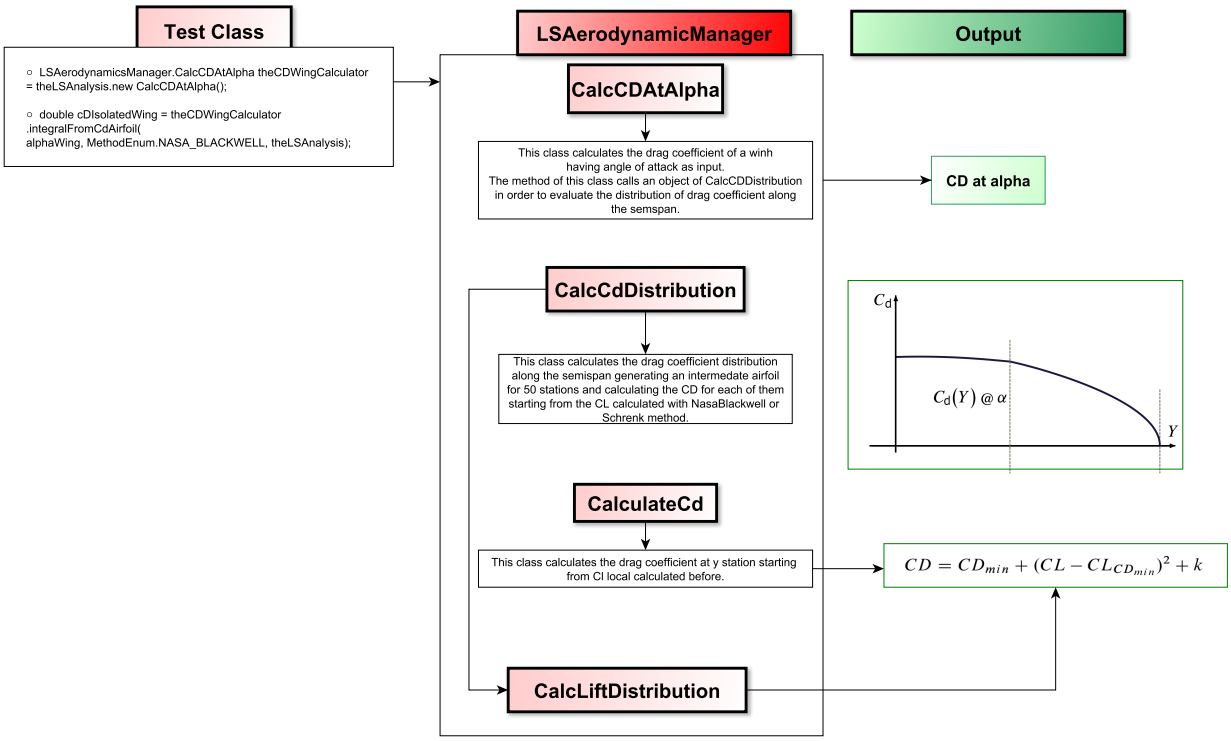


Figure 7.4: Flow chart of drag estimation classes.

7.3 Case Study

After initializing the Test class and defining the work object, it is possible to define an object of the class `CalcCDAtAlpha` that calculates the drag coefficient of a wing at a given angle of attack. This method accepts as input three parameters: the angle of attack of the wing, the method used in order to calculate of lift distribution and an object of the lifting surface aerodynamic manager.
The it is possible to plot the curves of drag coefficient versus α_w or lift coefficient that is the drag polar of the wing.

Listing 7.1 Use of Drag Calculator class

```
// ------------------------------------------------------------------------
// DRAG CHARACTERISTICS
// ------------------------------------------------------------------------

System.out.println("\n\n-----------------------------------");
System.out.println("\n DRAG CHARACTERISTICS  ");
System.out.println("\n-----------------------------------");

LSAerodynamicsManager.CalcCDAtAlpha theCDWingCalculator = theLSAnalysis
                .new CalcCDAtAlpha();
double cDIsolatedWing = theCDWingCalculator.integralFromCdAirfoil(
                alphaWing, MethodEnum.NASA_BLACKWELL, theLSAnalysis);
System.out.println(" CD of Wing at alpha body = (deg) "
                + alphaBody.to(NonSI.DEGREE_ANGLE).getEstimatedValue()
                + " is " + cDIsolatedWing);

System.out.println(" ...waiting for plotting");
theLSAnalysis.PlotCDvsAlphaCurve(subfolderPath);
System.out.println("\n\n\t\t\tDONE PLOTTING CD vs ALPHA WING");

}
```

Following there are the summary diagrams of the results obtained for the ATR 72:

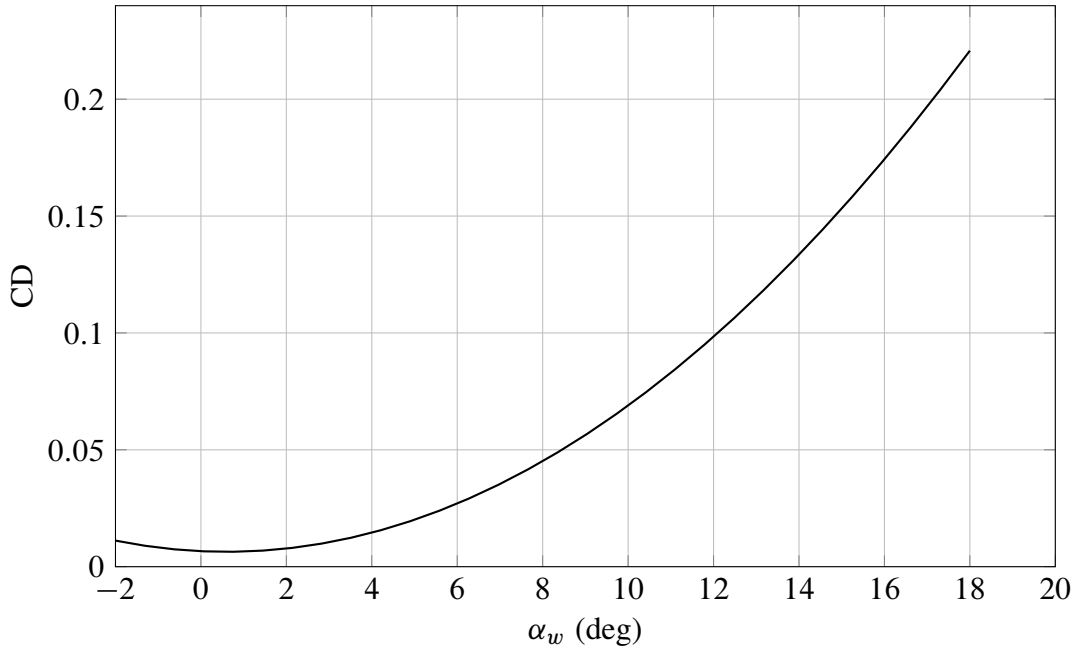


Figure 7.5: ATR 72 CD vs Alpha_w at Mach 0.43 and Altitude: 6000 m.

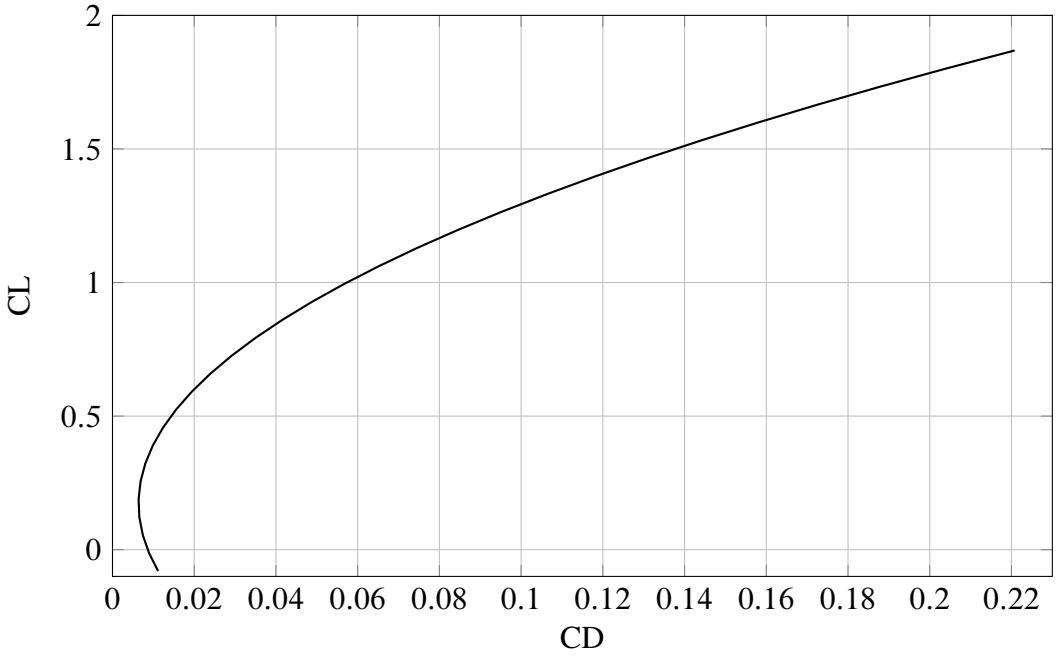


Figure 7.6: ATR 72 Parabolic Polar Drag at Mach 0.43 and Altitude: 6000 m.

Chapter 8

Downwash Estimation

The important thing is not to stop questioning.
Curiosity has its own reason for existing.
– Albert Einstein

In order to evaluate the characteristics of longitudinal stability of an aircraft it's necessary to assess the flow direction aft of the wing. The contribution of horizontal tail surface to the airplane equilibrium and stability, in fact, depends seriously on the flow direction. The pourpose of this chapter is to introduce and evaluate the downwash gradient due from the wing's vortex system, considering a dependence of the downwash angle from the absolute angle of attack.

8.1 Theoretical background

Due to the finite extension of the wing the lift distribution in span is not uniform. For this reason the difference of pressure between upper and lower surfaces generates a movement of air aroud the wingtips. The tendency is for particles of air to move from the region of high pressure around the wing tip to the region of low pressure (for positive lift from the lower wing surface to the upper surface). This made the wing's vortex system that consisting of the bound vortex, located at the wing quarter chord and a vortex sheet which rolling up, at the wing tip, in two trailing vortex.[62] [76]

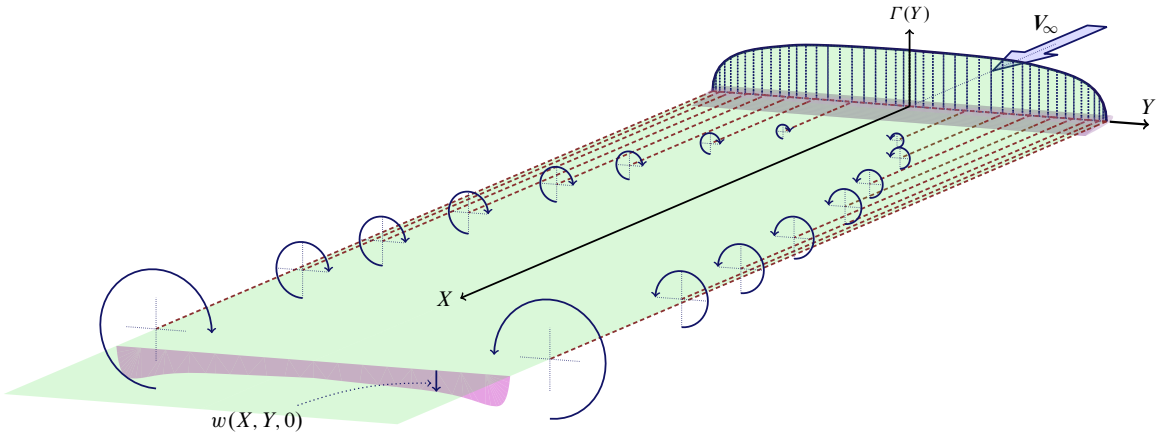


Figure 8.1: The wing vortex sheet.

The main effect of this vortex system is to deflect the airflow behind the wing downward relative to the direction of freestream flow. This angle of deviation is known as *Downwash Angle* ϵ. This phenomenon occurs for every lifting surface, but in subsonic flow a lifting surface also affects the flow forward of itself. In this region the vortex creates an itshape upwash, that is an upward flow deflection.

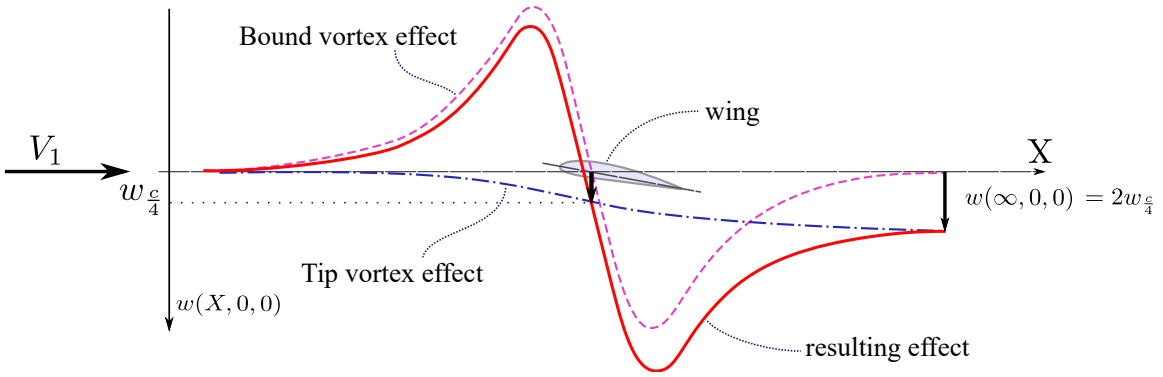


Figure 8.2: Upwash and Downwash in a finite wing.

As consequence of the downwash behind the wing, the local angle of attack on the horizontal tail is reduced by ϵ. In order to evaluate the flow direction behind the wing, an other important parameter is the change in downwash angle with angle of attack, that is the *Downwash Gradient* $\frac{d\epsilon}{d\alpha}$.
This parameter depends principally on the location of the horizontal tail with respect to the wing and the vortex plane. As first approximation this value could be considered constant in alpha, but more accurately it's possible to evaluate this dependence considering the reference variable for the calculation of the distances. Both if the vertical distance is considered as constant and it is considered variable with alpha, starting from the downwash gradient, the downwash angle is :

$$\epsilon = \frac{d\epsilon}{d\alpha_w}(\alpha_w - \alpha_{0_w}) \tag{8.1}$$

In order to evaluate the downwash gradient it refers to fig. 8.3, where "$r\frac{b}{2}$" is the distance between the aerodynamic center of wing and the aerodynamic center of the horizontal tail. This is a geometric an fixed distance. Conversely, in order to have a greater accuracy it's possible to consider the distance "$m\frac{b}{2}$" variable with the angle of attack. Properly this is the distance between the horizontal tail and the vortex shed plane, but it's possible to approximate it with the distance between the horizontal tail and the wing root chord.[74]

Referring to the equation used in order to evaluate the downwash gradient is the following:

$$\frac{d\epsilon}{d\alpha} = \frac{K_{\epsilon\Lambda}}{K_{\epsilon_{\Lambda=0}}}\left(\frac{r}{r^2+m_{tv}^2}\frac{0.4876}{\sqrt{r^2+0.6319+m_{tv}^2}} + \left[1+\left(\frac{r^2}{r^2+0.7915+5.0734m_{tv}^2}\right)^{0.3113}\right]\left\{1-\sqrt{\frac{m_{tv}^2}{1+m_{tv}^2}}\right\}\right)\frac{C_{L_{\alpha w}}}{\pi \mathcal{AR}} \tag{8.2}$$

Considering a variable downwash gradient the changing parameter is $m\frac{b}{2}$.

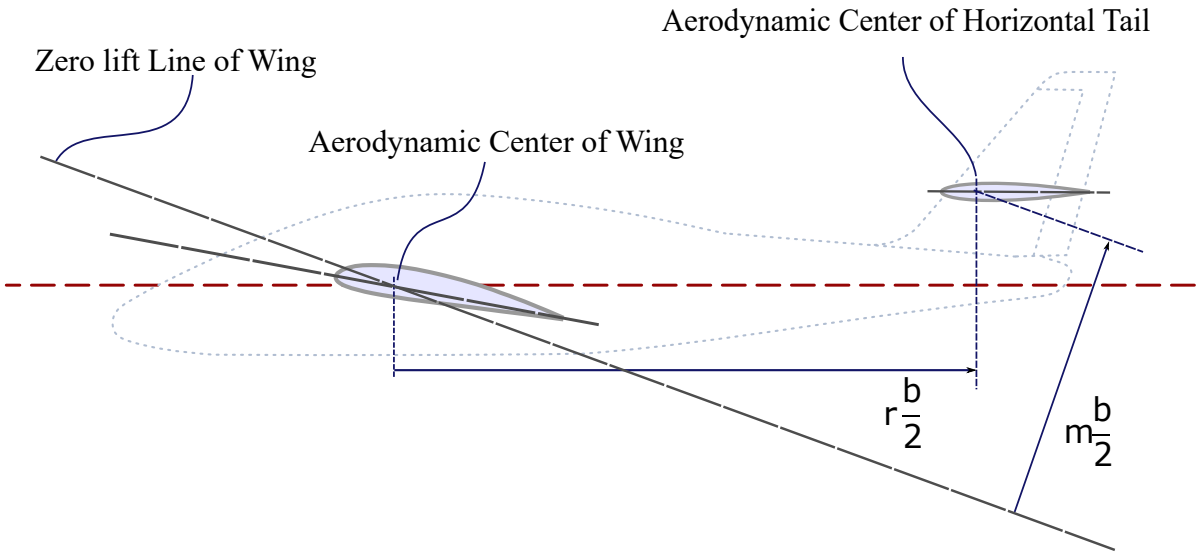


Figure 8.3: Dimensions for determination of Downwash Gradient, considering constant distances.

The two K_ϵ terms in the eq 8.3 accounting for the wing sweep angle effect are defined as follow (where Λ expressed in radians):

$$K_{\epsilon\Lambda} = \frac{0.1124 + 0.1265\Lambda + 0.1766\Lambda^2}{r^2} + \frac{0.1024}{r} + 2 \tag{8.3}$$

$$K_{\epsilon\Lambda=0} = \frac{0.1124}{r^2} + \frac{0.1024}{r} + 2 \tag{8.4}$$

These two terms are constant with α because it does not appear the variable parameter in them.

8.2 Java Class Architecture

In order to simplify the calculation of downwash, as mentioned, it's possible to assume the downwash gradient constant with the angle of attack. In this case the reference line to calculate the distance along z axis is the plane from the wing root chord or else the zero-lift line of the wing.
To obtain a more accurate analysis it's possible to consider the variation in alpha of the downwash gradient. So the reference line of wing it's not costant, but is the vortex sheed plane. The location of this plane depends from the value of downwash, but this location is itself necessary to evaluate the downwash. So it's necessary an iterative process in which the position of the vortex reference line at alpha is calculated from the value of downwash gradient at previous step.

In this process the reference angle of attack is the absolute angle α_a, that is the angle between the flow direction and the zero lift line of the wing. This choice is necessary because for $\alpha_a = 0$ it's possible to assume the downwash zero, but the downwash gradient is not null. In this way it's possible to assume the downwash value in the first step of the iteration and to continue for each step with the previous value as first attempt. In view of stability, however, the reference angle of attack is the α_B. There two angles they are related by the relation: $\alpha_B = \alpha_{0L} - i_w + \alpha_a$.
The relation between the angles is in the fig **??**.

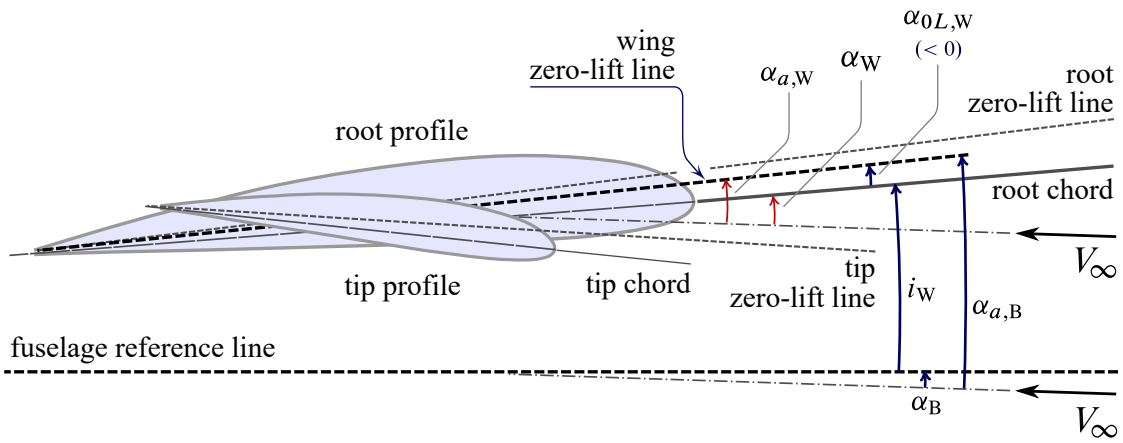


Figure 8.4: Definition of wing angles.

The downwash angle is calculated by a class named `DownwashCalculator`. The builder of this class defines and assings all the geometrical variables, necessary to implement the calculation. The `DownwashCalculator` class has seven methods and some overloads. The methods are explained in the table 8.1. The methods will be explained more in detail below.

`calculateDownwashGradientConstantDelft`	This method calculates the downwash gradient considering the vertical distance geometrical and constant
`calculateDownwashNonLinearDelft`	This method calculates the downwash considering a non constant downwash gradient.
`getDownwashAtAlphaBody`	This method returns the value of downwash angle, interpolating data filled before.
`calculateZDistanceZeroLift`	This method calculates the distance between the aerodynamic centre of horizontal tail and the zero lift line of the wing.
`Plot Methods ...`	Using these methods it's possible to plot the downwash angle, the downwash gradient and the distance in function of α_B

Table 8.1: Methods of `DownwashCalculator` class.

8.2.1 Constant Downwash Gradient

In order to evaluate the downwash angle case of constant downwash gradient it's necessary only to call the method `calculateDownwashGradientConstantDelft` using the distance from aerodynamic center of horizontal tail and the alpha zero lift line as input. It's possible to calculate this distance geometrically using the method `calculateZ DistanceZeroLift` of the same class. The choice to calculate separately the distance and

the downwash gradient is made to reuse the method to calculate downwash gradient simply varying the input distance.
This method has the downwash gradient as output. To obtain the angle of downwash it simply need to moltiplicate the output value and the absolute angle of attack.

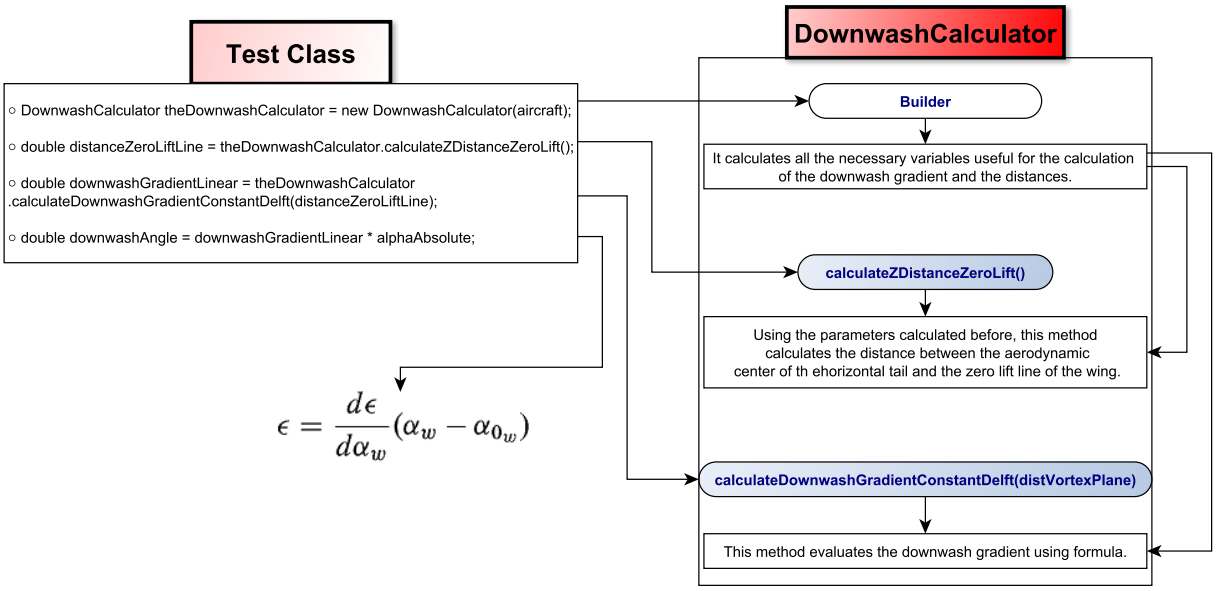


Figure 8.5: Flow chart of the calculation of linear downwash angle.

8.2.2 Variable Downwash Gradient

In order to evaluate the non-constant downwash gradient must use the method textttcalculateDownwashNonLinearDelft. This method calculates the downwash gradient using Delft formula. The downwash gradient is considered variable in alpha absolute. The distance along x considered in the formula is geometric and fixed. Conversely the other distance is variable and it is considered as the distance between the horizontal tail the vortex shed plane. This method works through the following steps:

1. First of all this method creates an array of absolute angle of attack starting from $\alpha = 0°$ to $\alpha = 20°$ with a step of $0.25°$.
2. The results array are initialized (α_a , α_B , $\frac{d\epsilon}{d\alpha}$, ϵ , $m\frac{b}{2}$).
3. For the first step the state is the following:
   - $\alpha_a = 0°$
   - $\alpha_B = \alpha_{0L} - i_w$
   - $\frac{d\epsilon}{d\alpha}$ is the constant value
   - $m\frac{b}{2}$ distance is the same of the previous case and it is calculated using the method `calculateZDistanceZeroLift`.
   - $\epsilon = 0$
4. Starting from the second step the process is iterative. Starting from $\alpha_a = 0°$ the absolute angle of attack increase of $\Delta\alpha$. For the step i:
   - $\alpha_a|_i = i\,\Delta\alpha$

- $\epsilon_{temp} = \frac{d\epsilon}{d\alpha}|_{i-1} * \alpha_a|_i$
- $m\frac{b}{2}|_{temp}$ is calculated considering the temporary value of downwash angle.
- $\frac{d\epsilon}{d\alpha}|_{temp}$ is calculated using the formula and the temporary value of distance.
- $\epsilon_i = \frac{d\epsilon}{d\alpha}_{temp} * \alpha_a|_i$
- $m\frac{b}{2}|_i$ is calculated considering the new value of downwash angle.
- $\frac{d\epsilon}{d\alpha}|_i$ is updated.
- $\alpha_B = \alpha_{0L} - i_w + \alpha_a$

In order to relieve the calculations, the evaluation of downwash angle and downwash gradient should be done only one time. To obtain the value of epsilon at alpha body it's possible to call the method `getDownwashAtAlphaBody` that interpolates the value of downwash angle and angle of attack which field must be filled before.

Step by step the value of $m\frac{b}{2}$ is calculated with the following geometrical construction:

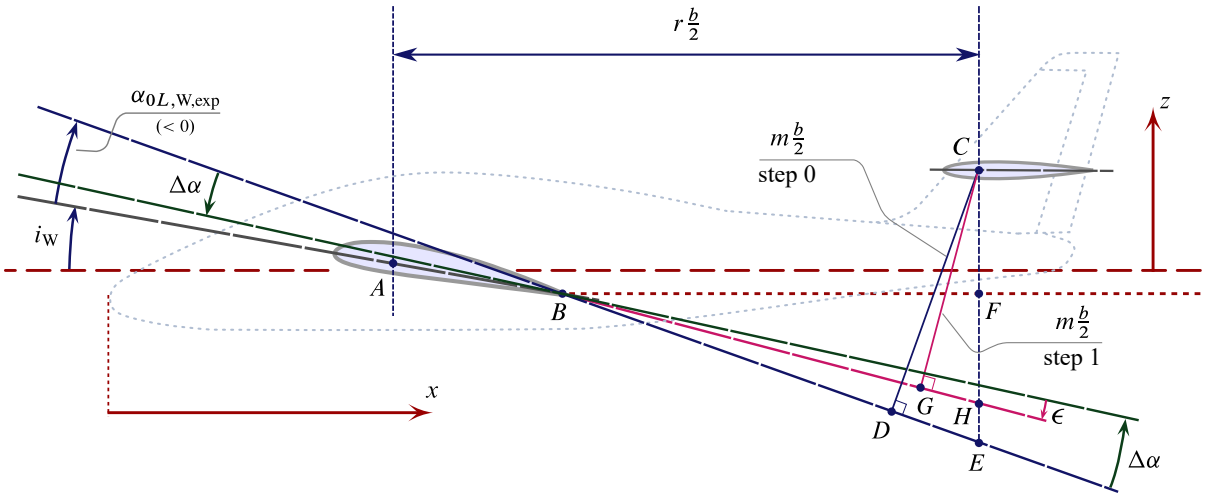


Figure 8.6: Arm definitions for downwash gradient evaluation.

Referring to the step 0, the distance $m\frac{b}{2}$ is the segment $\overline{CD}$. It's possible to calculate this distance geometrically as follows:

$$\overline{CD} = (\overline{CF} + \overline{FE}) * \cos(i_w - \alpha_{0L}) \tag{8.5}$$

$\overline{CF}$ is the distance along Z between the aerodynamic center of horizontal tail and the trailing edge of root airfoil of the wing.

$$\overline{CF} = Z_{ac_H} - Z_{ac_W} \tag{8.6}$$

Is possible to calculate the distance $\overline{FE}$ considering the triangle BFE. The side $\overline{BF}$ is the distance along X axis between the aerodynamic center of horizontal tail and the trailing edge of root airfoil of the wing, while the angle between this side and the hypotenuse is $i_w - \alpha_{0L}$.

For each step the method is the same, but the difference is that the angle $i_w - \alpha_{0L}$ becomes $i_w - \alpha_{0L} - i\,\Delta\alpha + \epsilon$. In this way the distance that changes is $\overline{FE}$

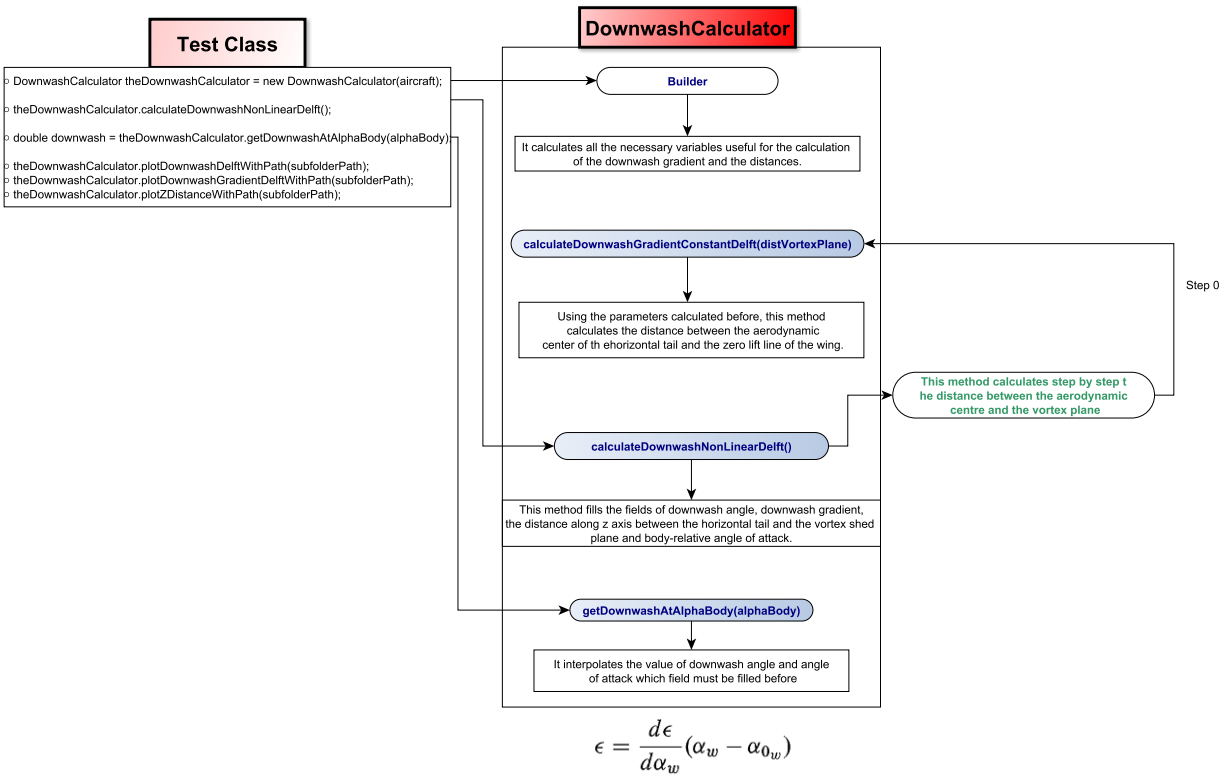


Figure 8.7: Flow chart of the calculation of non-linear downwash angle.

8.3 Case Study

In the following listing is reported the Test Class used in order to evaluate the variability of the downwash gradient with α_B. First of all the test class is initialized, after an Aircraft object is defined with the related analysis classes.

Listing 8.1 Downwash Test Class

```
        // ------------------------------------------------------------------------
        // INITIALIZE TEST CLASS ...
        // ------------------------------------------------------------------------

        // ------------------Downwash---------------

        System.out.println("\n-----Start␣of␣downwash␣calculation-----\n" );

        DownwashCalculator theDownwashCalculator = new DownwashCalculator(aircraft);

        theDownwashCalculator.calculateDownwashNonLinearDelft();

        theDownwashCalculator.plotDownwashDelftWithPath(subfolderPath);
        theDownwashCalculator.plotDownwashGradientDelftWithPath(subfolderPath);
        theDownwashCalculator.plotZDistanceWithPath(subfolderPath);
        System.out.println("␣DONE␣PLOTTING␣DOWNWASH␣ANGLE␣vs␣ALPHA␣BODY");

        double downwash = theDownwashCalculator.getDownwashAtAlphaBody(alphaBody);
        Amount<Angle> downwashAmountRadiant = Amount
                                        .valueOf(Math.toRadians(downwash), SI.RADIAN);
                        System.out.println( "At␣alpha␣" + alphaBody
                                        .to(NonSI.DEGREE_ANGLE)
                .getEstimatedValue() + "␣(deg)␣the␣downwash␣angle␣is␣(deg)␣=␣" + downwash );


        }

}
```

Below are the charts representing the results obtained by applying the method described in ATR 72.

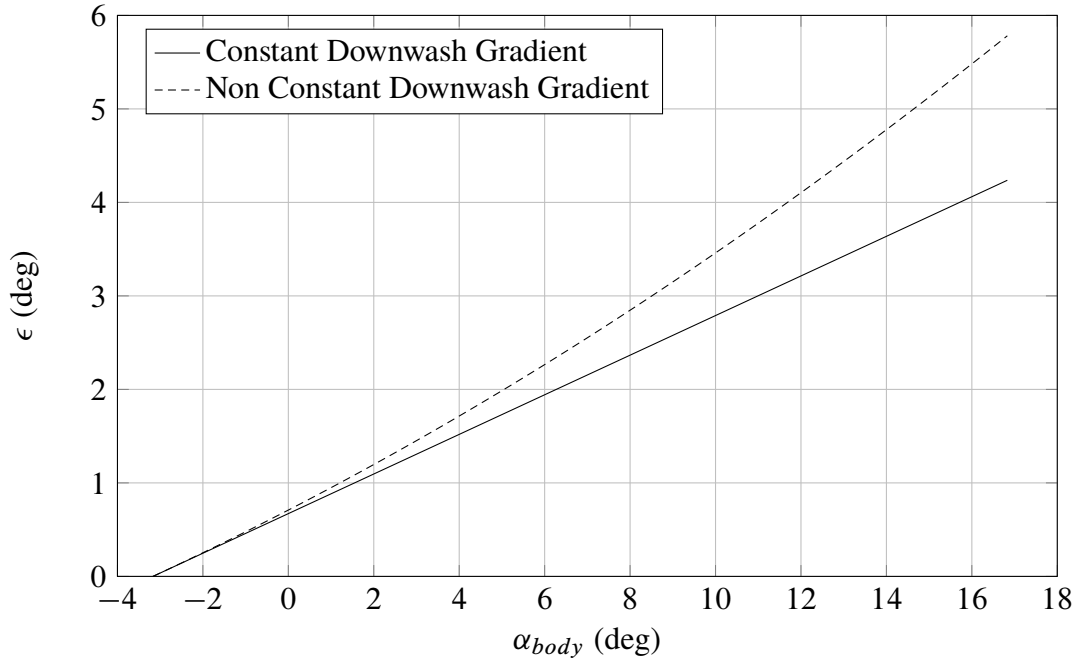


Figure 8.8: ATR 72 Downwash angle vs α_B.

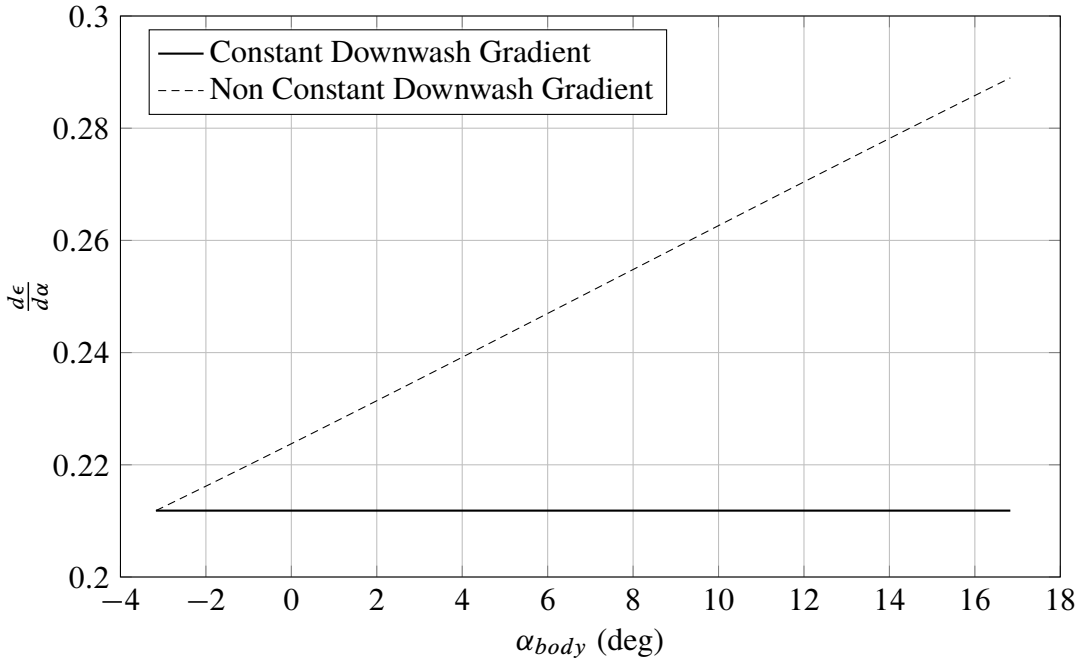


Figure 8.9: ATR 72 Downwash gradient vs α_B.

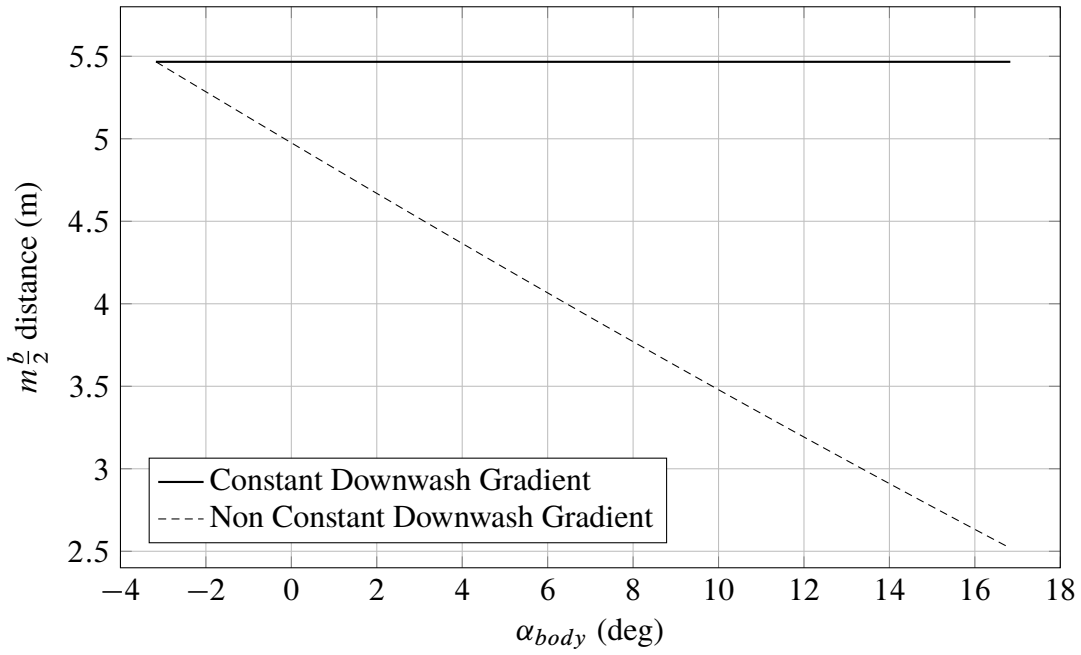


Figure 8.10: ATR 72 Distance between AC horizontal tail and vortex plane vs α_B.

Chapter 9

Aircraft Longitudinal Static Stability

For every bad thing in life,
there are more good things to tip the balance.
– Richelle Mead

Static stability is the reaction of a body to a disturbance from equilibrium. To determine the static stability of a body, the body must be initially disturbed from its equilibrium state. If, when disturbed from equilibrium, the initial tendency of the body is to return to its original equilibrium position, the body displays positive static stability or is stable. If the initial tendency of the body is to remain in the disturbed position, the body is said to be neutrally stable. However, should the body, when disturbed, initially tend to continue to displace from equilibrium, the body has negative static stability or is unstable. [24]
In addition to the static stability it's possible to analize the dynamical stability that is the time history of an aircraft response after it has been disturbed. A statically stable aircraft may not be dynamically stable, as explained in subsequent discussions. However, it is clear that a statically unstable aircraft also is dynamically unstable.

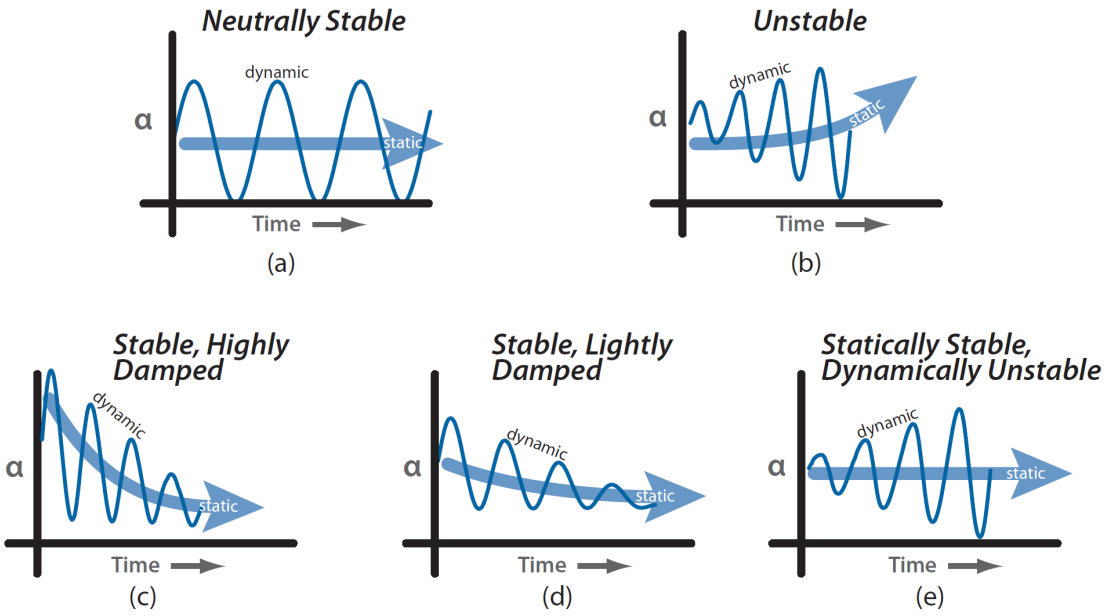


Figure 9.1: Static and dynamic stability about the pitch axis.

It is important to note that there is a substantial difference between the concepts of "Stability" and "Equilibrium". In fact, stability is a characteristic of an aircraft, while equilibrium is a state in which airplane can be, that means the equilibrium of the forces and the moments acting on it.
"Longitudinal static stability" is the stability of an aircraft in the longitudinal, or pitching, plane under steady-flight conditions. This characteristic is important because an airplane must have the tendency to return to the equilibrium position if perturbed. The pitch plane is the XZ plane of aircraft symmetry. The linear velocities are along the X-axis and w along the Z-axis. Angular velocity is about the Y-axis , known as pitching (positive if nose up). Pilot-induced activation of the elevator changes the aircraft pitch. In the plane of symmetry, the aircraft motion is uncoupled; that is, motion is limited only to the pitch plane.[48]

In order to evaluate the characteristics of longitudinal stability of an aircraft it's ncessary to express all the forces and the moments acting on it and evaluate the resultant pitching moment about the center of gravity.
The lift and drag are by definition always perpendicular and parallel to V_∞, ively. It is, therefore, inconvenient to use these forces to obtain moments because their moment arms relative to the center of gravity vary with angle-of-attack a. For this reason, all forces are resolved into normal, N, and chordwise, C, forces whose axes remain fixed with the aircraft and whose arms are, therefore, constant. [58]

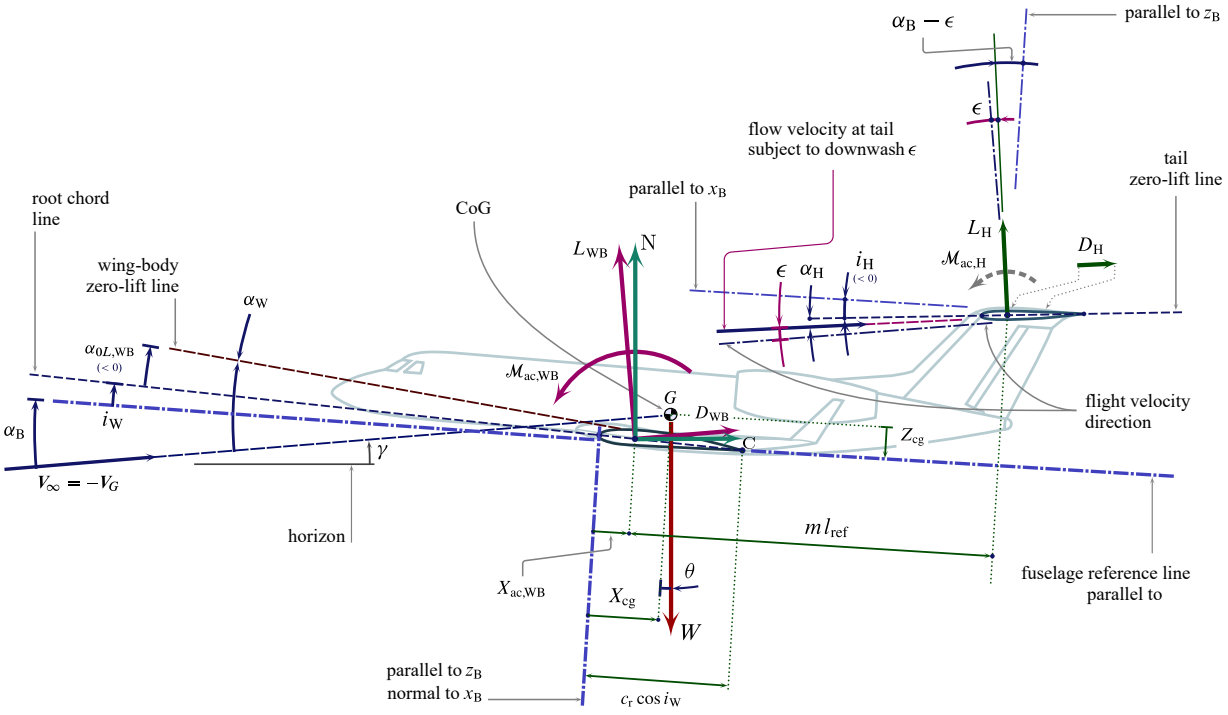


Figure 9.2: Forces and moments acting on an aircraft..

The relation between the lift, the drag and the normal and chordwise forces are the following:

$$N = Lcos(\alpha) + Dsin(\alpha) \tag{9.1}$$

$$C = Dcos(\alpha) - Lsin(\alpha) \tag{9.2}$$

According to what said, the aircraft can, in general, be trimmed, that is, put into an

equilibrium state where the combined lift of the wing and the tail balances the weight while the moment about the center of gravity is zero. It's necessary that this equilibrium point corresponds to positive lift. As regards the stability it is necessary that the response of the aircraft to a disturbance in angle of attack is to tend to return to the original equilibrium position. Thus, if flying in equilibrium at one angle of attack and a disturbance increases the angle of attack, the moment produced at this angle of attack must act to reduce it, that is, tend back toward the original equilibrium state. Conversely, if a disturbance decreases the angle of attack, the moment at the new, lower, angle of attack must serve to increase that angle. This means that the rate of change of the moment about the center of gravity must be negative for static stability: an increase in α should reduce $C_{M_{cg}}$ and a decrease in α should increase $C_{M_{cg}}$.[75] As higher is the slope of the curve, more is greater the the restoring moment. Thus the slope of the curve is called *Static Margin*. Static margin is a concept used to characterize the static longitudinal stability and controllability of aircraft. In aircraft analysis, static margin is defined as the distance between the center of gravity and the neutral point of the aircraft, expressed as a percentage of the mean aerodynamic chord of the wing. In order to have static stability, thus, is necessary that the neutral point is behind the cenetr of gravity.

In conclusion the requirement for static stability are the following:

- The pitching-moment curve exhibit a negative slope
- At a zero angle of attack there is positive nose-up moment

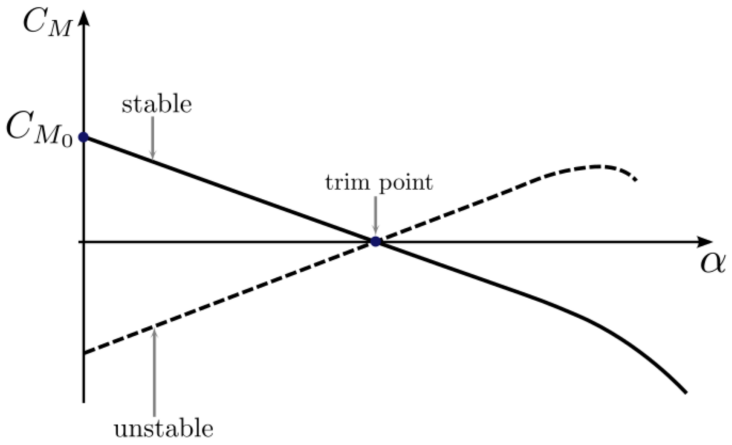


Figure 9.3: Qualitative representation of the moment coefficient vs α.

Below will be calculated all the contributions in order to evaluate the pitching moment coefficient about the center of gravity.

9.1 Aerodynamic Lift

Considering a reference angle of attack α_B, the aircraft components generates lift. So it is necessary to evaluate these contributes. In particular, first of all, is wanted to evaluate the C_L of isolated wing. This value will be correct with fuselage influence. Afterwards it is necessary to evaluate the contribute of horizontal tail. It is important to note that each component works at a different angle of attack in dipendence of angles of incidence and downwash , meanwhile for longitudinal stability the reference angle of attack is α_B.

9.1.1 Wing

The wing may be considered as the most important component of an aircraft. Its main purpose is to generate sufficient lift force. However, the wing has two other productions, namely drag force or drag and pitching moment. While a wing designer is looking to maximize the lift, the other two (drag and pitching moment) must be minimized.
The theoretical background and the followed process are shown in the CHAPT 6.

9.1.2 Fuselage

The fuselage is that portion of the aircraft wherein the payload is carried. In jet transports the payload consists of the passengers and their baggage and/or cargo. It is important that the fuselage has some features as a low aerodynamic drag; a minimum aerodynamic instability; comfort and attractiveness in terms of seat design, placement, and storage space; safety during emergencies; minimization of noise and ease of cargo handling in loading and unloading.[75]

The lift distribution on a wing is affected by the presence of the fuselage as a result of the following effects:

- The presence of the fuselage disturbs the longitudinal velocity field near the wing.
- At an angle of attack relative to the free stream, the fuselage also perturbs the flow about the wing in planes normal to the free stream.
- The fuselage has a blocking effect on the flow.

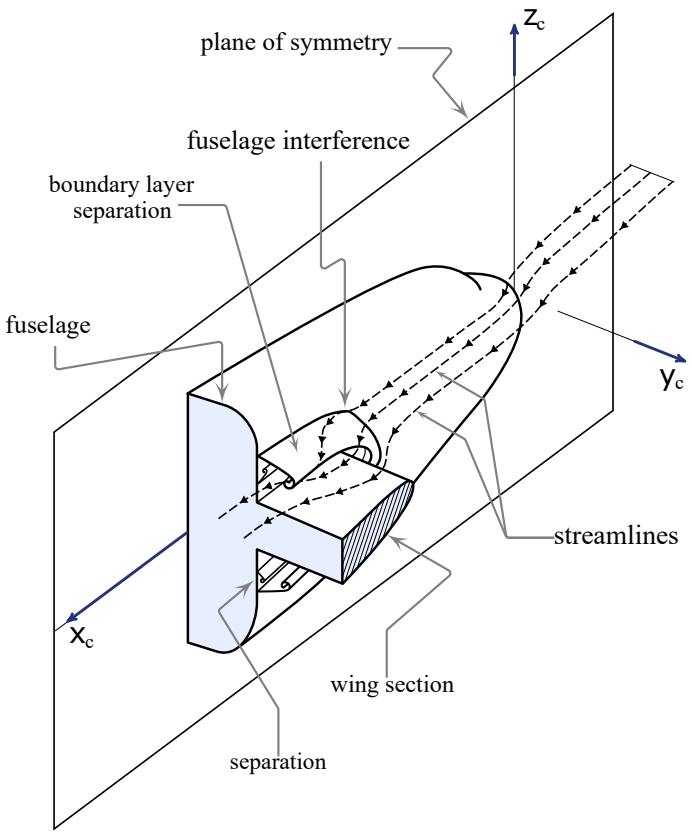
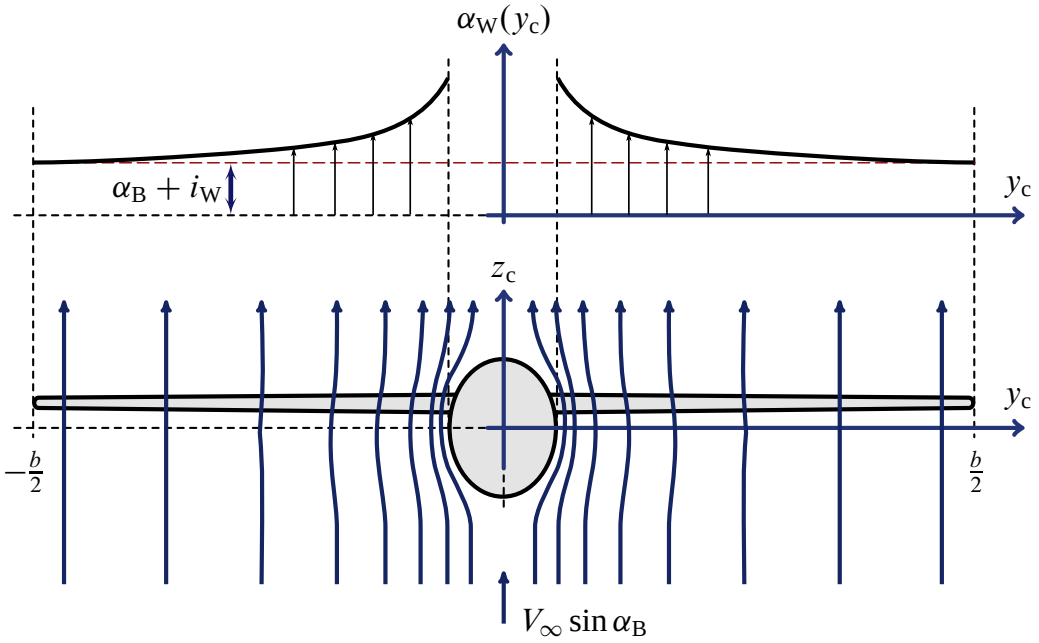


Figure 9.4: Wing fuselage junction flow.

These effects are not large for a slender fuselage but may be important when the fuselage is relatively bulky, so that a substantial alteration to the local longitudinal velocity may result. The cross-flow caused by the fuselage at angle of attack changes the component of free stream velocity normal to the fuselage axis and affects the downwash flow produced by the wing. The blocking effect of the fuselage is always present even if the fuselage is an infinite cylinder aligned with the free stream, in which case the other two effects are not present. [75]



i

Figure 9.5: Wing fuselage front view interference.

However, theoretical analysis showed that the presence of a slender fuselage does not have an important effect on the lift distribution on an unswept wing of moderate aspect ratio, but a larger change in the lift distribution on a wing in the presence of a fuselage may be anticipated if the wing is swept.[78] For greater accuracy of the calculation, the value of lift linear slope has been corrected using the following equation from [71]:

$$\left(\frac{dC_L}{d\alpha}\right)_{wb} = \left[1 + \frac{1}{4}\left(\frac{d}{b}\right) - \frac{1}{40}\left(\frac{d}{b}\right)^2\right]\left(\frac{dC_L}{d\alpha}\right)_w \tag{9.3}$$

Note that for typical airliners $\frac{d}{b} \approx 0.1$ and therefore the lift curve slope of the wing-body is approximately equal to that of the wing alone.
In addition to the calculation of the linear slope of the wing body group, it is necessary to calculate the intersection point between the lift curve of wing and the lift curve of wing-body. This point is the α_{0_L} of the exposed wing. The exposed wing is the part of wing outside of the fuselage that generates lift in a complete aircraft. This value has been calculated with the following integral formula.

$$\alpha_{0L}|_{EW} = \frac{2}{S_E}\int_{y_{0_E}}^{\frac{b}{2}} c(y)[\alpha_0(y_E) - \epsilon_T(y_E)]\,dy \tag{9.4}$$

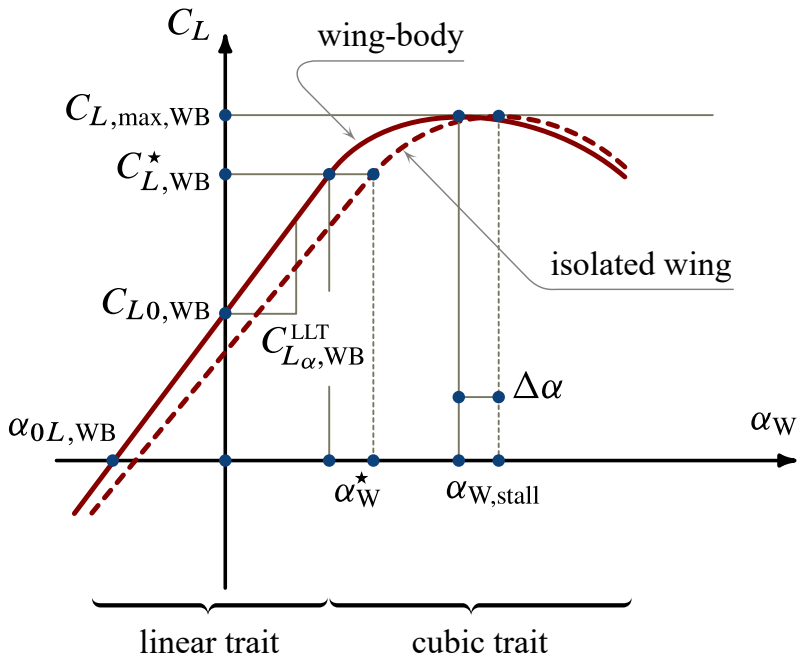


Figure 9.6: Lift curve of isolated wing and of wing-body group.

9.1.3 Horizontal Tail

The horizontal tail is an aerodynamic surface, typically including one or more movable control surfaces,that provides longitudinal stability and control. A stabilizer can feature a fixed or adjustable structure on which any movable control surfaces are hinged, or it can itself be a fully movable surface. In the first case the horizontal tail consists of the *stabilizer* and the *elevator* (moving) for handling the pitch degree of freedom. In case of a fully movable surface it is called *stabilator*.
The H-tail can be positioned low through the fuselage, in the middle cutting through the V-tail, or at the top of the V-tail to form a T-tail. [48] The horizontal tail provides longitudinal stability and control. It may be sized by one or more of the following conditions[60]:

- Provide static and dynamic stability. Provide a state of equilibrium in each flight condition.
- Enable aircraft control.
- Provide a state of equilibrium in each flight condition.

The stabilizer is a fixed wing section whose job is to provide stability for the aircraft, to keep it flying straight. The horizontal stabilizer prevents up-and-down, or pitching, motion of the aircraft nose. The elevator is the small moving section at the rear of the stabilizer that is attached to the fixed sections by hinges. Because the elevator moves, it varies the amount of force generated by the tail surface and is used to generate and control the pitching motion of the aircraft. There is an elevator attached to each side of the fuselage. The elevators work in pairs; when the right elevator goes up, the left elevator also goes up.

Considering an horizontal tail formed by stabilizer and elevator, in order to evaluate the lift coefficient of this lifting surface it's necessary to consider the deflection of control surface. The elevator, in fact, can be considered as a *plain flap*.

Elevator index of effectiveness

In order to evaluate the contribution to the longitudinal stability of horizontal tail it's necessary to consider the deflection of the elevator.

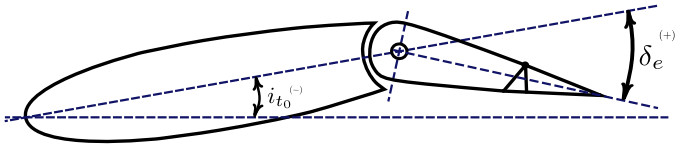


Figure 9.7: Characteristic angles of the horizontal tail.

The variation of zero lift angle is not constant with the angle of deflection. So it's necessary to evaluate the tau factor which is defined as follows:

$$\tau_e = \frac{d\alpha_{0l}}{d\delta_e} \tag{9.5}$$

Introducing this parameter the lift coefficient of the horizontal tail can be rated as follows:

$$C_{L_H} = C_{L_0} + C_{L_{\alpha_H}} \alpha_H + C_{L_{\alpha_H}} \tau_e \delta_e \tag{9.6}$$

Considering a symmetrical horizontal tail, the term C_{L_0} is zero, so it's possible to express the lift coefficient in the following form:

$$C_{L_H} = C_{L_{\alpha_H}} (\alpha_H + \tau_e \delta_e) \tag{9.7}$$

In general the value of τ is constant until about 15 deg; after this value, due to the flow separation, the effectiveness of elevator decrease and consequently the product $\tau_e \delta_e$ that appears in the equation of lift coefficient.

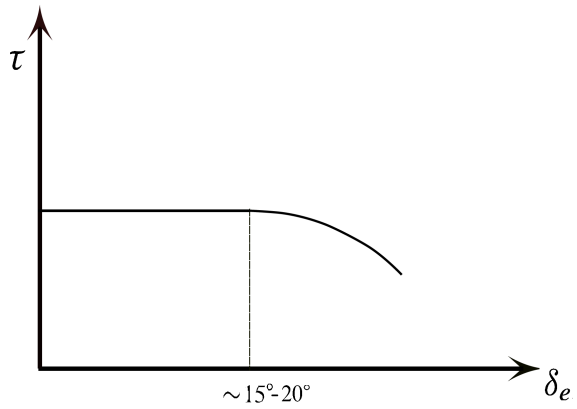


Figure 9.8: Qualitative trend of τ with the deflection of elevator.

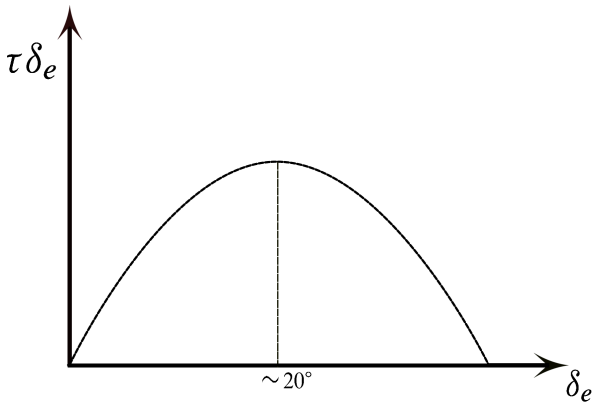


Figure 9.9: Qualitative trend of the term $\tau \cdot \delta_e$ with the deflection of elevator.

The evaluation of tau is made by reading of external database, considering the following graphs.

$$\tau = \alpha_\delta \eta_\delta = \frac{\alpha_{\delta_{c_L}}}{\alpha_{\delta_{c_l}}} \alpha_{\delta_{c_l}} \eta_\delta \tag{9.8}$$

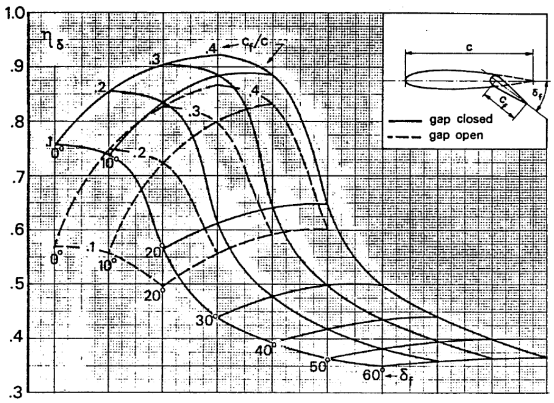


Figure 9.10: 2D efficiency correction for elevator.

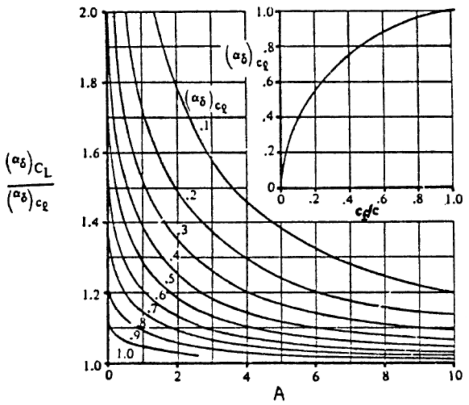


Figure 9.11: $\frac{d\alpha_{0l}}{d\delta_e}$ 2D and 3D correction.

9.1.4 Complete Aircraft

In order to evaluate the lift coefficient of the entire airplane it's possible to consider it as consisting of the following parts[68]:

- Wing and Fuselage
- Horizontal Tail
- Canard

It's important to consider the effectiveness angles of attack in which the surfaces work. This is made considering the angles of incidence of the lifting surfaces and the downwash angle aft of the wing. An horizontal tail and a canard may be equipped with a trailing edge control surface. So in order to evaluate these contributes it's important to know the angle of deflection δ of these control surfaces.
The calculation of the individual contributions it's reported in the relevant sections. In this section will be shown the method to evaluate the aircraft lift coefficient, known the single contributes.
For an aircraft with no canard, the formula is the following:

$$C_L = C_{L_{wb}} + \frac{S_t}{S_w}\eta_t C_{L_t} \tag{9.9}$$

Where η_t is the ratio of dynamic pressure, called *tail efficiency factor*. In fact the dynamic pressure seen by horizontal tail differ from the free stream dynamic pressure due to two main reasons: the combination wing-fuselage and the presence of the propeller. The dynamic pressure of the tail depends on the location of the tail. If the tail is in the wake of the wing-body, the local dynamic pressure will be less than the freestream because the flow gradually loses its kinetic energy. While if the tail is in the slipstream of propeller, the local dynamic pressure may increase due to the power absorbed by the propeller.

9.2 Aerodynamic Drag

The drag of an aircraft depends on its shape and speed, which are design-dependent, as well as on the properties of air, which are nature-dependent. Drag is a complex phenomenon arising from several sources, such as the viscous effects that result in skin friction and pressure

differences as well as the induced flow field of the lifting surfaces and compressibility effects. The aircraft drag estimate starts with the isolated aircraft components. Each component of the aircraft generates drag largely dictated by its shape. Total aircraft drag is obtained by summing the drag of all components plus their interference effects when the components are combined. It is important to note that the drag of two isolated bodies increases when they are brought together due to the interference of their flow fields.[48]

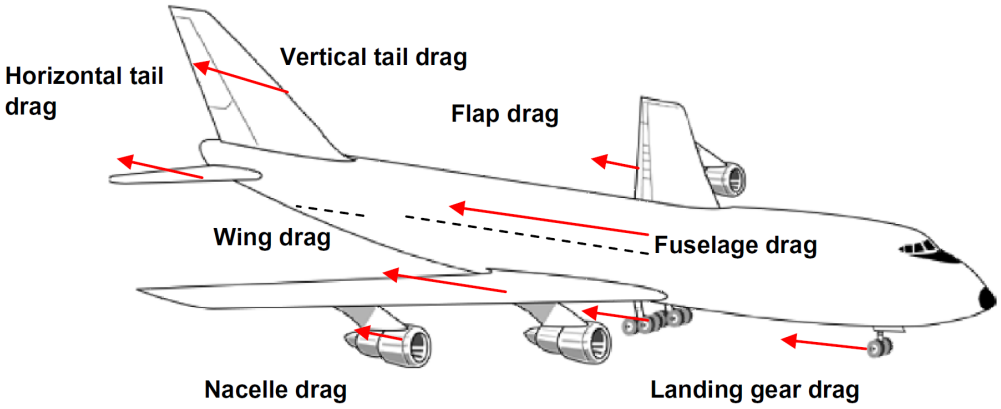


Figure 9.12: Drag components making up the total commercial airplane drag.[75]

In this thesis work, in order to evakuate the pitching moment coefficient of an aircraft has been considered the drag forces due to wing and horizontal tail.

9.2.1 Drag of a lifting surface

As widely expressed in the CHAPT 7, the drag coefficient of a lifting surface is calculated starting from the drag coefficient of the airfoils, expressed by the following equation:

$$CD = CD_{min} + (CL - CL_{CD_{min}})^2 + k \tag{9.10}$$

For each station, known the semi-spanwise lift distribution, it's possible to calculate the local lift coefficient of the intermediate airfoil and, consequently, the drag coefficient. After the drag coefficient of the entire lifting surface is calculated with an integral.

9.2.2 Additional drag due to elevator

For the horizontal tail it's necessary to consider the additional drag due to the elevator deflection. It's possible to consider the elevator as a *plain flap* and use the concerning methods.
A plain flap is an high lift devices where a portion of the rear of a wing is simply hinged. The effectiveness of the flap derives from the fact that on rotation it changes the camber of the section and so permits of a change of circulation and therefore, of lift at a given incidence.
The increment in profile drag coefficient due to a flap at a given incidence is rather more influenced by test conditions than is the lift coefficient increment. Nevertheless, the influence of test conditions and wing incidence is still sufficiently small over a wide range of incidence for us to accept the increment at a standard incidence as a reliable measure of the profile drag characteristics of a flap. [84] The increment of drag is a ΔCD_0 Young and Hufton assumed that $\Delta C_D 0$ can be calculated as follow for a full-span flap:

$$\Delta C_{D0} = \delta_1 \left(c_f/c\right) \cdot \delta_2 \left(\delta_f\right) \tag{9.11}$$

where δ_1 and δ_2 are functions that were determined from experimental data. Their related curves are shown in figure 9.13 and 9.14 for plain flap.

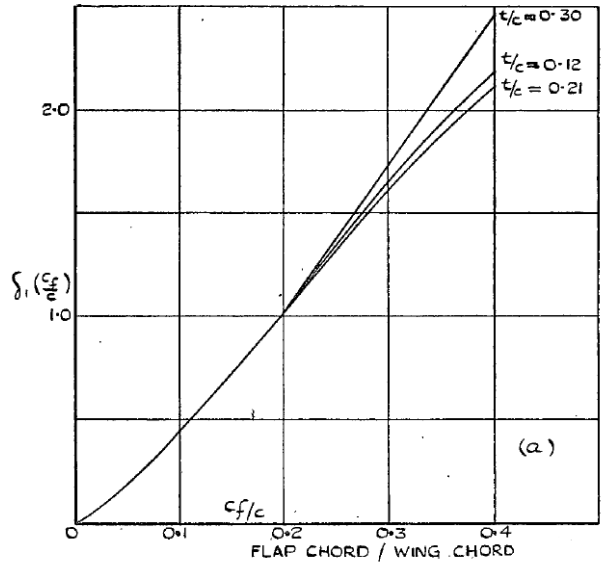


Figure 9.13: The functions $\delta_1\left(c_f/c\right)$ for split and plain flaps

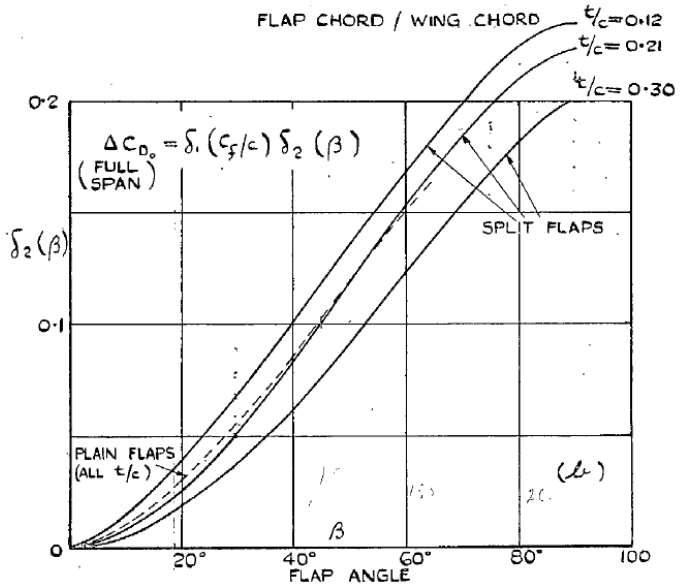


Figure 9.14: The functions $\delta_2\left(\delta_f\right)$ for plain flaps

9.3 Pitching Moments

The purpose of this section is to explain the followed procedure used to evaluate the pitching coefficient moments of the aircraft components.
Due to the interaction between the body and the flow a pressure distribution is generated whose result is the aerodynamic force acting at the center of pressure. In consequence of what there is a pitching moment that is a moment acting on the pitch axis of a moving body. As angle of attack changes on a cambered airfoil, there is movement of the center of pressure forward and aft. One of the remarkable properties of a cambered airfoil is that, even though

the center of pressure moves forward and aft, there is a point, called Aerodynamic Center, with respect to which the moment coefficient is constant. The aerodynamic center is the most convenient place to locate the lift, drag, and moment of an aircraft wing or airfoil section. So, for each component of the aircraft, it will be calculated the pitching moment coefficient respect to its aerodynamic center.

9.3.1 Lifting surface

In order to evaluate the pitching moment coefficient of a lifting surface, at a given angle of attack, the implemented process is the following.
First it's calculated the pitching momet coefficient respect to the point at a quarter of the mean aerodynamic chord. Starting from these values, to vary the angle of attack, it's possible to evaluate the position of aerodynamic center.

1. Fifty points along the semispan are defined
2. For each point, an intermediate airfoil is calculated
3. For each airfoil, at given station, the lift coefficient is calculated with the local CL vs α curve. In this way it's possible to consider both the linear trait and non linear
4. For each airfoil the center of pressure is calculated using the following equation:

$$\frac{x_{CP}}{c} = \frac{x_{AC}}{c} - \frac{C_{M_{ac}}}{C_l} \tag{9.12}$$

5. For each airfoil the arm between the local center of pressure and the point at a quarter of lifting surface's MAC is calculated. The equations that define the MAC are the following:

$$MAC = \frac{2}{S}\int_0^{\frac{b}{2}} c^2\, dy \tag{9.13}$$

$$x_{MAC} = \frac{2}{S}\int_0^{\frac{b}{2}} x_{le}(y)c(y)\, dy \tag{9.14}$$

$$y_{MAC} = \frac{2}{S}\int_0^{\frac{b}{2}} y_{le}(y)c(y)\, dy \tag{9.15}$$

6. Finally it's possible to evaluate the pitching moment respect to the point at a quarter of MAC with the product between the lift force and the arm

Repeating the process to vary the angle of attack it's possible to estimate the slope of the pitching moment curve. In fact, the aerodynamic center, is that point on an aircraft, wing, or airfoil section about which the pitching moment is independent of angle of attack. If the slope is positive, the aerodynamic center is behind the quarter of the MAC, conversely if the slope is negative, the aerodynamic center is the quarter of the MAC is behind the aerodynamic center.

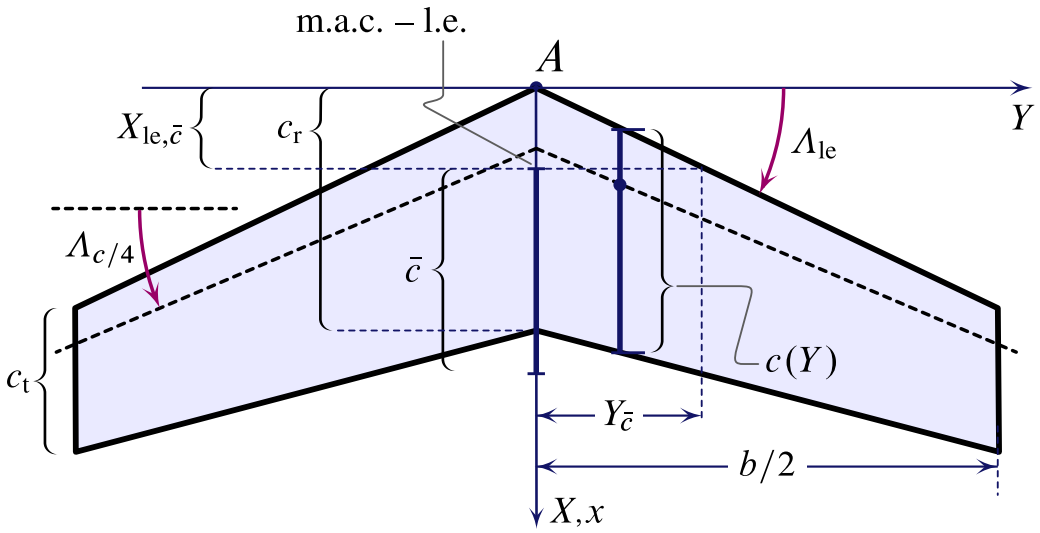


Figure 9.15: Wing topview definitions.

9.3.2 Fuselage

The moment coefficient of the fuselage is a pure couple, so is not necessary specify the pole.

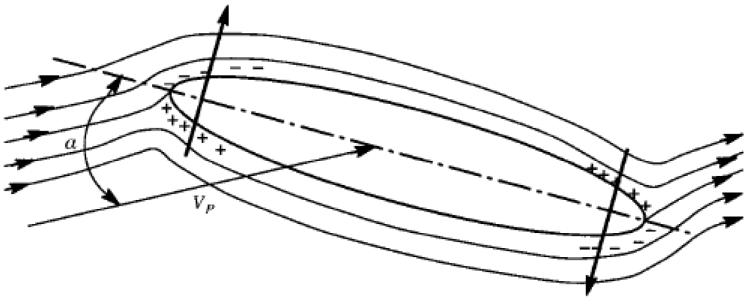


Figure 9.16: Fuselage in potential flow at angle of attack.

Assuming the linearity, is possible to represented the pitching moment coefficient by the following equation in function of the angle of attack.

$$C_{M_f} = C_{M_{0_f}} + C_{M\alpha_f}\alpha_{body} \tag{9.16}$$

In JPAD it's possible to evaluate these two contributes using different methods.

Multhopp method

Each term in the equation 9.16 can be calculated by the Multhopp method. The fuselage is divided into strips, each of which gives a contribution to the pitching moment according to its distance from the wing.

The two coefficients $C_{M_{0_f}}$ and $C_{M\alpha_f}$ in eq. 9.16 can be obtained from the following equations.

$$C_{M_0 f} = \frac{K_2 - K_1}{36.5 \cdot S \cdot MAC} \cdot \int_0^{l_f} {W_f}^2 \cdot \left(\alpha_{0L_w} + i_w + i_{cl_f}\right) \mathrm{d}x \tag{9.17}$$

$$C_{M\alpha_f} = \frac{1}{36.5 \cdot S \cdot MAC} \cdot \left\{ \int_0^{l_{f1}} {W_f}^2 \cdot \left[\left(\frac{\partial \epsilon_u}{\partial \alpha} \right)_1 + 1 \right] \mathrm{d}x_1 + \int_0^{l_{f2}} {W_f}^2 \cdot \left[\left(\frac{\partial \epsilon_d}{\partial \alpha} \right)_2 + 1 \right] \mathrm{d}x_2 \right\} \tag{9.18}$$

Instead of the integrals it's possible to substitute them with summations.

$$C_{M_0 f} = \frac{K_2 - K_1}{36.5 \cdot S \cdot MAC} \cdot \sum_{j=1}^{n} {W_f}_j{}^2 \cdot \left(\alpha_{0L_w} + i_w + i_{cl_f} \right) \cdot \Delta x; \tag{9.19}$$

$$C_{M\alpha_f} = \frac{1}{36.5 \cdot S \cdot MAC} \cdot \left\{ \sum_{j=1}^{n_1} {W_f}_j{}^2 \cdot \left[\left(\frac{\partial \epsilon_u}{\partial \alpha} \right)_1 + 1 \right] \cdot \Delta x_1 + \sum_{j=1}^{n_2} {W_f}_j{}^2 \cdot \left[\left(\frac{\partial \epsilon_d}{\partial \alpha} \right)_2 + 1 \right] \cdot \Delta x_2 \right\} \tag{9.20}$$

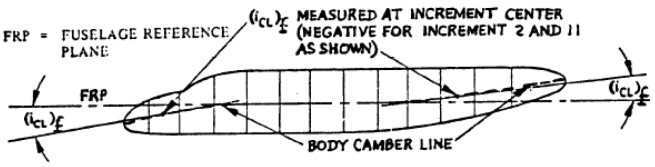


Figure 9.17: Strips definition for the determination of $C_{M_{0_f}}$.

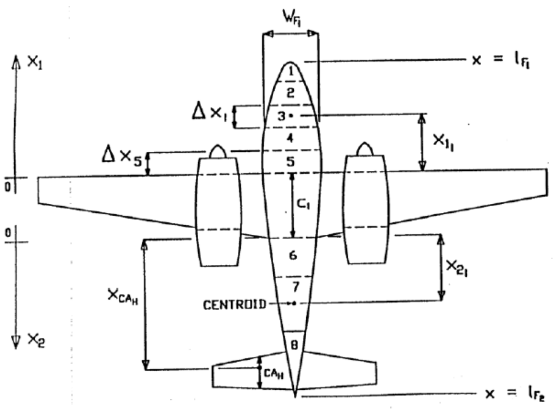


Figure 9.18: Strips definition for the determination of $C_{M\alpha_f}$.

The parameter $K_2 - K_1$ is a correction factor that is dependent on the value of the fineness ratio and is evaluated by the following figure.[28]

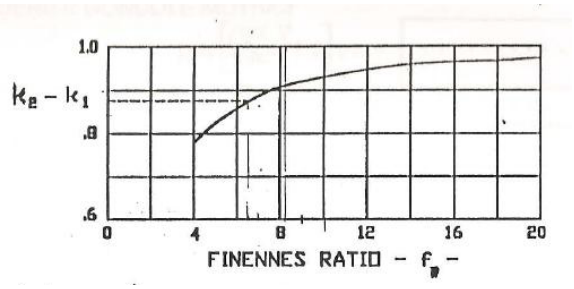


Figure 9.19: $K_2 - K_1$ correction factor in function of fineness ratio.

Dor stability it's necessary to evaluate the new aerodynamic center (wing-body) and the pitching moment coefficient respect to this point. In fact the fuselage introduces a shift of the aerodynamic center to the leading edge. It's possible to obtain the new aerodynamic center using the Eq. 9.21

$$\Delta X_{ac} = x_{ac_{wb}} - x_{ac_w} = -\frac{C_{M_{\alpha_f}}}{C_{L_{\alpha_f}}} \tag{9.21}$$

The wing-body pitching moment respect to the new aerodynamic center is :

$$C_{M_{ac_{wb}}} = C_{M_{ac_w}} + C_{M_{0l_f}} \tag{9.22}$$

Fuselage Pitching moment prediction method

This method has been developed in the Dept. of Industrial Engineering, University of Study of Naples Federico II, by numerical aerodynamic analyses performed with STAR-CCM+. From a reference fuselage layout the nose, cabin (constant fuselage diameter), and tailcone geometry have been parametrically changed. Then, the aerodynamic drag, the pitching moment at zero incidence, the longitudinal static stability derivative, and the yawing moment coefficients C_D, C_{M_0}, C_{M_α}, and C_{N_β} respectively, have been evaluated by numerical analysis. Aerodynamic effects of each component (nose, cabin, tailcone) have been directly obtained from the CFD aerodynamic solver, separating the forces and moments contributions of each fuselage part. [6]

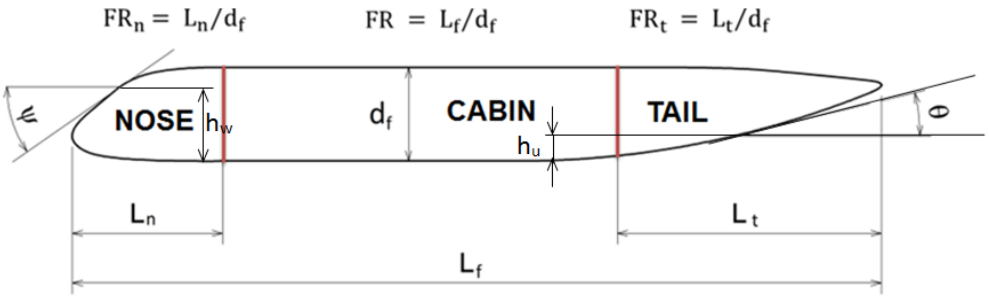


Figure 9.20: Main fuselage geometrical parameters.

The fuselage pitching moment coefficient can be estimated according to the equation 9.16 where :

$$C_{M_{0_f}} = C_{M_{0_{FR}}} + \Delta C_{M_{0_{NOSE}}} + \Delta C_{M_{0_{TAIL}}} \tag{9.23}$$

$$C_{M_{\alpha_f}} = C_{M_{\alpha_{FR}}} + \Delta C_{M_{\alpha_{NOSE}}} + \Delta C_{M_{\alpha_{TAIL}}} \tag{9.24}$$

where the subscript FR refers to the pitching moment coefficient as function of fuselage fineness ratio. The subscript $NOSE$ the pitching moment nose correction factor. It depends on windshield angle Φ and on the nose fineness ratio FR_n. Finally the subscript $TAIL$ refers to the pitching moment tail correction factor. It depends on upsweep angle θ and on the tail fineness ratio FR_t.
The following figures show the variability of introduced parameters.

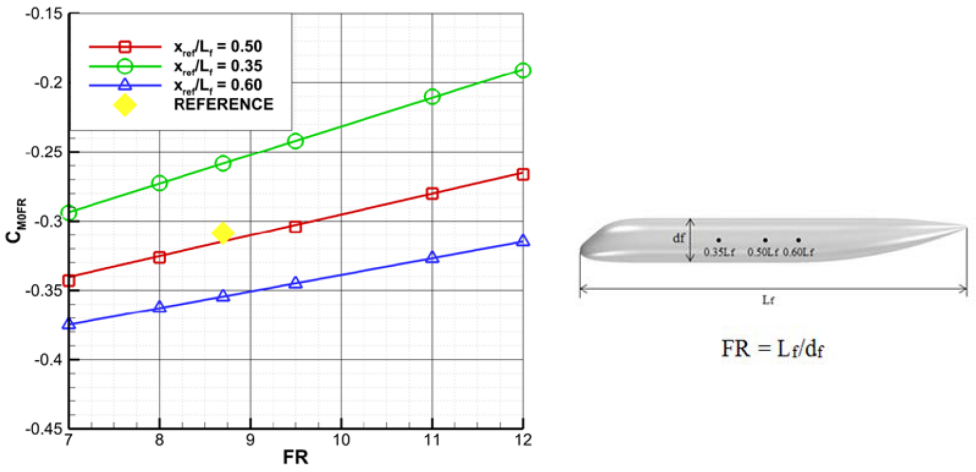


Figure 9.21: Zero angle of attack pitching moment coefficient $C_{M_{0_{FR}}}$.

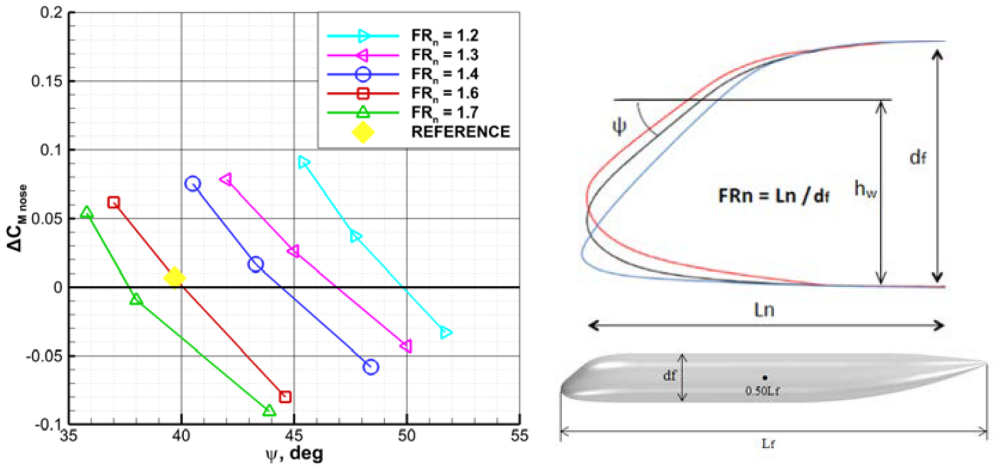


Figure 9.22: Zero angle of attack pitching moment coefficient nose correction factor.

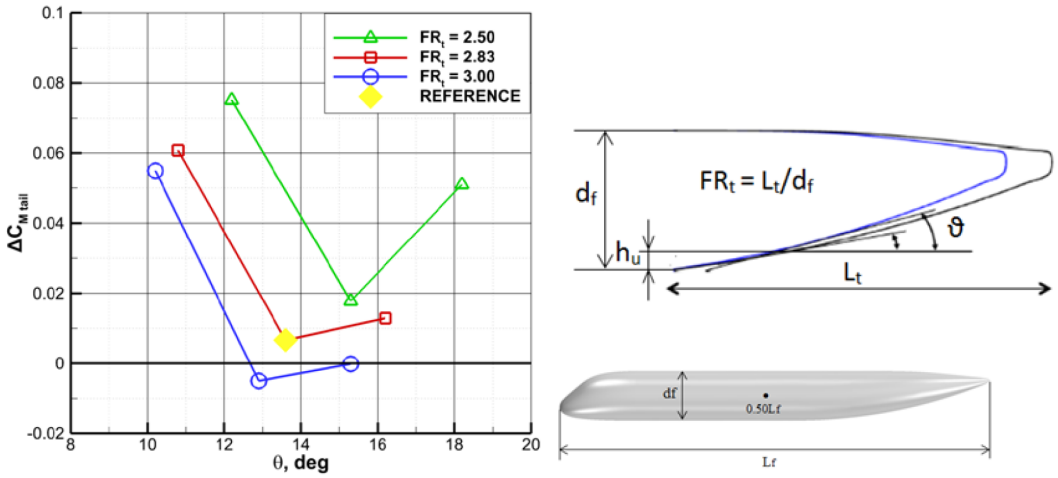


Figure 9.23: Zero angle of attack pitching moment coefficient tail correction factor.

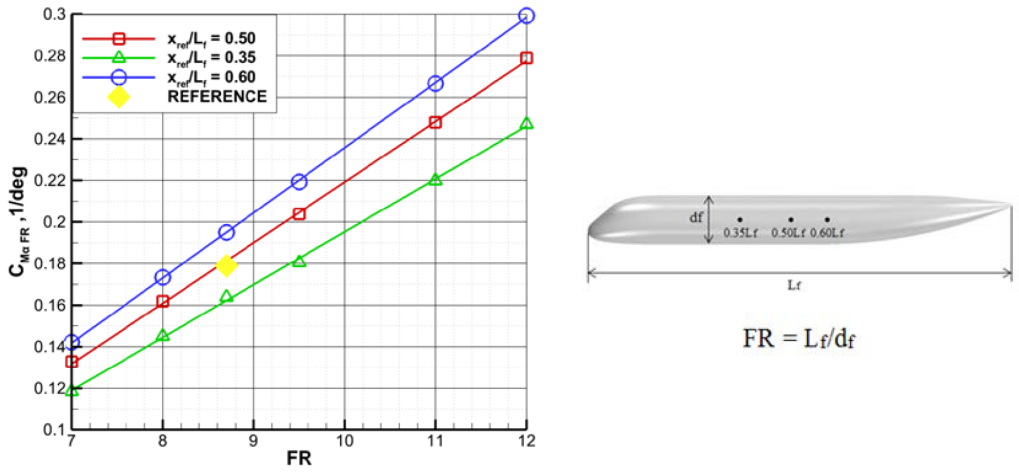


Figure 9.24: Pitching moment derivative with respect to FR.

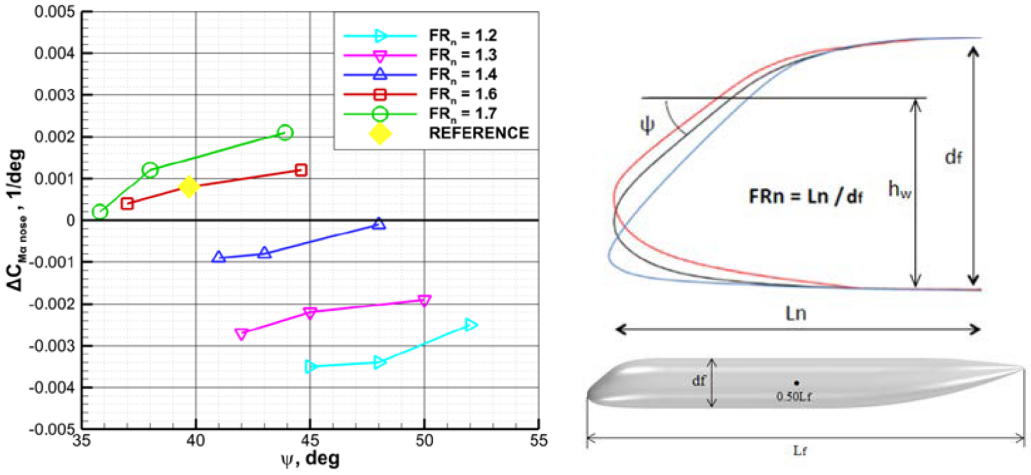


Figure 9.25: Pitching moment derivative nose correction factor.

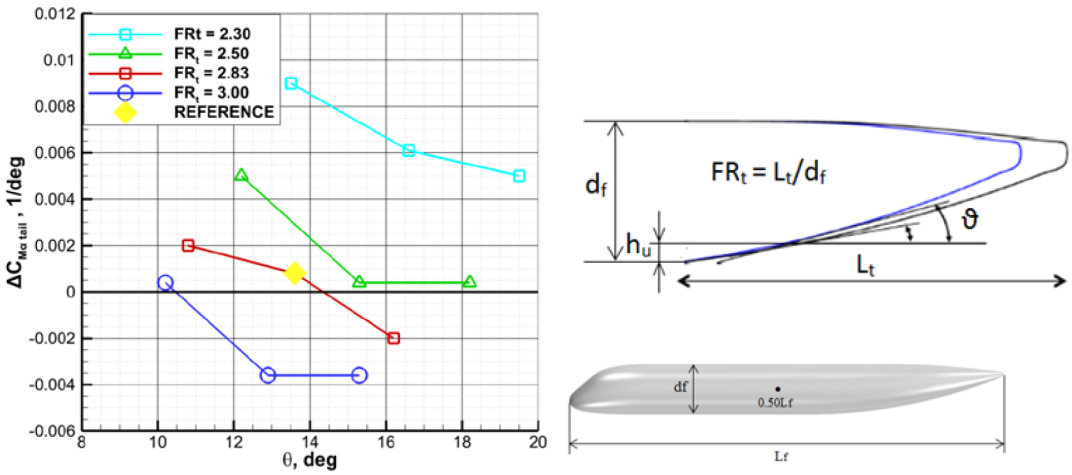


Figure 9.26: Pitching moment derivative tail correction factor.

9.4 Propulsors

The effects of propeller and slipstream on airplane static longitudinal stability may be significant. It's possible to divide the power effects on longitudinal static stability in two categories: direct effects and Indirect effects.

9.4.1 Direct Effects

These effects are principally the following:

- Effect due to the thrust and moments result from the line of action of force with respect to the CG.
- Pitching moment due to non axial flow.
- Contrast couple on the that must be compensated by the aileron.

Due to these effects, the airplane pitching curve must be accounted for a change in slope $\frac{dC_m}{dC_L}$. The equations differs between turboprop engine aircraft and turbojet. In case of turboprop it's possible to express the power direct effects with the following equations: Pitching moment due to thrust effect:

$$C_{M_P} = 2T_C \frac{D^2}{S}\frac{h_p}{\bar{c}} = 2(KC_L^{\frac{3}{2}})\frac{D^2}{S}\frac{h_p}{\bar{c}} \tag{9.25}$$

This equation means that if the thrust acting under the cg ($h_p > 0$) with increasing alpha (and CL) the moment coefficient undergoes a positive change, ie the power effect has an unstable contributions.

Pitching moment due to non axial effect:

$$\frac{dC_{M_P}}{dC_L} = \frac{dC_{N_P}}{d\alpha_p}\frac{S_p}{S}\frac{l_p}{\bar{c}}\frac{\left(1+\frac{d\epsilon}{d\alpha}\right)}{C_{L_{\alpha w}}}N_{prop} \tag{9.26}$$

Whre $\frac{d\epsilon}{d\alpha}$ is the upwash effect forward wing, represented in fig. 9.27

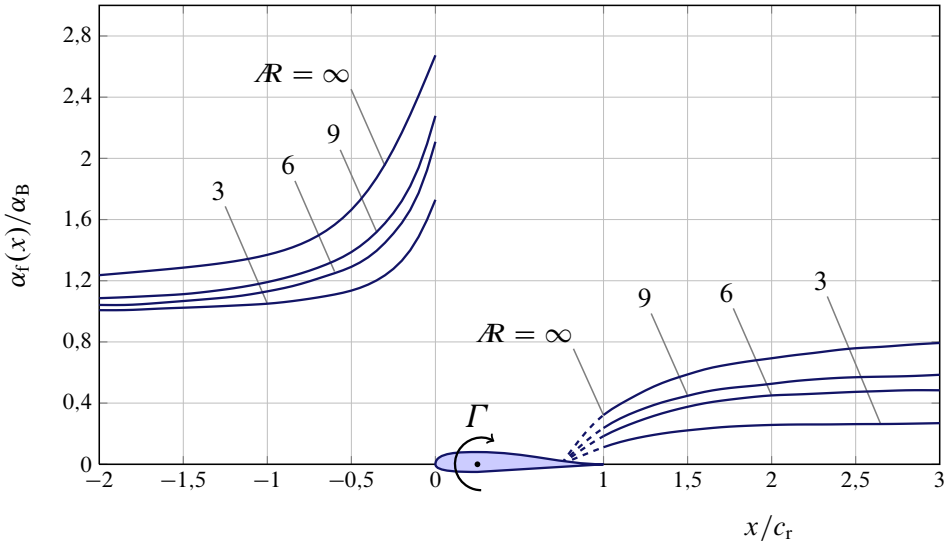


Figure 9.27: Wing upwash and downwash.

9.5 Stability Calculation

9.5.1 Neutral Point and Static Margin

9.6 Java Class Architecture

9.7 Case Study

Chapter 10

Minor Works

Once we accept our limits,
we go beyond them.
– Albert Einstein

10.1 Induced Angle of Attack

When a wing passes through the air, it creates a perturbation, which leads to a vertical component on the relative motion on the air over the wing. Consequently the air flow past the wing is inclined in regard of the initial free stream. Then, the wing behaves as thought the flow were coming from a different direction: the local flow direction.

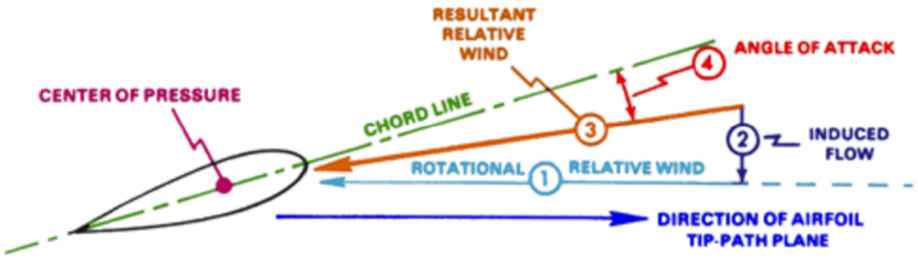


Figure 10.1: Definitions of induced end effective angles of attack.

In practice, these perturbations are generated by the wingtip vortices, which create downwash and deflects the local airflow in the vicinity of the wing downward. This angle is the induced angle of attack α_i. The airfoil section itself is then responding to an effective angle of attack equal to the geometric angle of attack minus the induced angle of attack: $\alpha_e = \alpha - \alpha_i$. In most cases, the absolute value of the angle of attack is decreased, and so as for the lift force. Then, in order to produce a given lift force, the AOA of the wing must be greater than the AOA that would be given by a theoretical study of the wing section (2D wing). Moreover, the resulting force exerted by the air on the wing, now perpendicular to the local flow direction, is tilted backwards by an angle equal to the induced angle of attack and then is no longer perpendicular to the free stream velocity. This important phenomenon is at the origin of the induced drag (i.e. the component of the resulting force parallel to the free stream velocity), and therefore for a given lift, a greater induced AOA implies a greater drag. One must bear in mind that because this is a 3-dimentional effect.[12]

In order to evaluate the effective angle of attack for each spanwise station, it's possible to implement an iterative process in witch at a load distribution it's evaluated the related induced angles that generates a new load distribution.
The induced angle of attack is in eq. 10.1, where w is the vertical downwash component:

$$\alpha = \arctan\left(\frac{w}{V}\right) \tag{10.1}$$

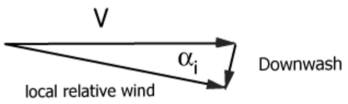


Figure 10.2: Geometrical representation of induced angle of attack.

The lifting surfaces is considered as divided in several rectangular horse-shoe vortices along the span; one horseshoe vortex along the chord is used, that is, the midpoints of the vortices are placed only at points along the quarter-chord lines. An equal number of control points are located along the three-quarter-chord lines. The velocity from the total vortex system is equated to the component of free-stream velocity normal to the lifting surface chord at each control point.
The downwash velocity at any of the control points P, which results from the 2N horseshoe vortices is:

$$w = \frac{1}{4\pi}\sum_{n=1}^{N} \Gamma_N F'_{w,\nu_N} \tag{10.2}$$

where Γ is circulation strength, F is the downwash influence function, N are the vortex point and ν are the vortex points.
In JPAD this effect is calculated in the class `AlphaEffective` by the method `calculateAlphaEffe`
This method has as output an array of value that indicates the effective angls of attack semi-spanwise using the equations introduced before. In order to evaluate these angles you must have as input the operating conditions.

Listing 10.1 Effective angle of attack Test Class

```
// ------------------------------------------------------------------
// Evaluate effective angle of attack
// ------------------------------------------------------------------

System.out.println("\n\n------------------------------------------------------");
System.out.println("STARTING EVALUATE EFFECTIVE ANGLE OF ATTACK ");
System.out.println("------------------------------------------------------");

double[] alphaEffective;

AlphaEffective theAlphaCalculator = new AlphaEffective(
                theLSAnalysis, theWing, theOperatingConditions);
Amount<javax.measure.quantity.Angle> inputAngle =
                Amount.valueOf(toRadians(8.), SI.RADIAN);

alphaEffective = theAlphaCalculator.calculateAlphaEffective(inputAngle);

System.out.println("\n\n------------------------------------------------------");
System.out.println(" alpha --> " + inputAngle);
System.out.println(" alpha effective --> " + Arrays.toString(alphaEffective));

System.out.println("\n\n------------------------------------------------------");
System.out.println("DONE");
System.out.println("------------------------------------------------------");
```

Conclusions

An aircraft design and optimization desktop application written in Java, and its functionality, has been introduced. The adoption of established software engineering practices, the use of advanced development tools, and concurrent development enabled the developer team to build a feature-rich application in a relatively short period of time. As of its design, the application is easily maintainable and extendable. The software is still growing and the choice of Java language was of considerable help. In fact being Java a pure object oriented programming language, it greatly encourages and simplifies modularization. Each module (package) can be programmed quite independently so that it is relatively easy to divide the work among several programmers working simultaneously or one after the other.
The application, moreover, can be easily integrated into a comprehensive aircraft optimization cycle. As all analysis modules inside the JPADCore package will be completed and tested, the final purpose of the code will be to allow users to define a certain numbers of macroscopical geometrical parameters, along with a given objective function, and to receive as output the best set of the previous parameters which suits the wanted objective. These future targets will make the software able to carry out an analysis of an aircraft during its preliminary design phase in a fast and flexible way.

Appendix A

HDF dataset and database reader creation

In a tool for preliminary design phase of an aircraft, it's very important to have aviable database. It's possible to create database starting from graphics using external software. In this appendix will be explained the step required in order to digitalize the graphics, create an HDF dataset and set up the database-reader class in Jpad.

A.1 Chart Digitization

The first step required for create a dataset is to digitalize a chart. Often data is found presented in reports and references as functional X-Y type scatter or line plots. In order to use this data, it must somehow be digitized. This is made with an external software, such as *Plot Digitizer*. Plot Digitizer is a Java program used to digitize scanned plots of functional data. This program will allow you to take a scanned image of a plot (in GIF, JPEG, or PNG format) and quickly digitize values off the plot just by clicking the mouse on each data point.[16]

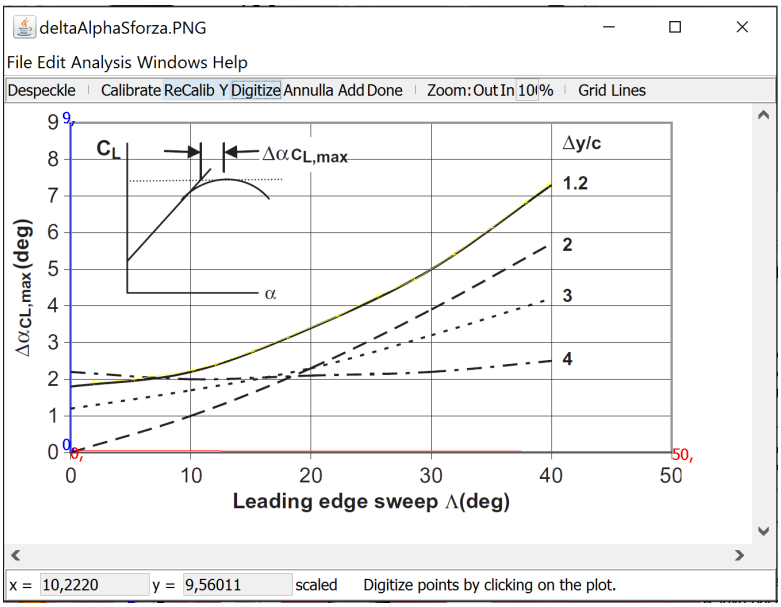


Figure A.1: Chart digitization using Plot Digitizer.

In order to digitize a chart, first of all it's necessary to calibrate the axis. Plot Digitizer works with both linear and logarithmic axis scales. After it's possible to digitize a curve

simply click o it. The values obtained can then be saved to a tex file or .csv file.

A.2 Creation of an HDF file with Matlab

Obtained the .csv file from digitization is necessary to create the HDF file. First of all it's necessary to import the file with couple of coordinates as matrix. After saving the imported files as .mat file, Matlab code comes in play to manage these data and to generate the digitalized curves and the HDF dataset. The code interpolates curves points with cubic splines in order to have more points to plot for each curve.

Listing A.1 MATLAB script for creating the HDF Database

```
clc; close all; clear all;

%% Import data
DeltaAlphaCLmax_vs_LambdaLE_dy1p2 = importdata('DeltaAlphaCLmax_vs_LambdaLE_dy1p2.mat');
DeltaAlphaCLmax_vs_LambdaLE_dy2p0 = importdata('DeltaAlphaCLmax_vs_LambdaLE_dy2p0.mat');
DeltaAlphaCLmax_vs_LambdaLE_dy3p0 = importdata('DeltaAlphaCLmax_vs_LambdaLE_dy3p0.mat');
DeltaAlphaCLmax_vs_LambdaLE_dy4p0 = importdata('DeltaAlphaCLmax_vs_LambdaLE_dy4p0.mat');

nPoints = 30;
lambdaLEVector_deg = transpose(linspace(0, 40, nPoints));

%% dy/c = 1.2
smoothingParameter = 0.999999;
DAlphaVsLambdaLESplineStatic_Dy1p2 = csaps( ...
    DeltaAlphaCLmax_vs_LambdaLE_dy1p2(:,1), ...
    DeltaAlphaCLmax_vs_LambdaLE_dy1p2(:,2), ...
    smoothingParameter ...
    );

DAlphaVsLambdaLEStatic_Dy1p2 = ppval( ...
    DAlphaVsLambdaLESplineStatic_Dy1p2, ...
    lambdaLEVector_deg ...
    );

%% dy/c = 2.0

smoothingParameter = 0.999999;
DAlphaVsLambdaLESplineStatic_Dy2p0 = csaps( ...
    DeltaAlphaCLmax_vs_LambdaLE_dy2p0(:,1), ...
    DeltaAlphaCLmax_vs_LambdaLE_dy2p0(:,2), ...
    smoothingParameter ...
    );

DAlphaVsLambdaLEStatic_Dy2p0 = ppval( ...
    DAlphaVsLambdaLESplineStatic_Dy2p0, ...
    lambdaLEVector_deg ...
    );

%% dy/c = 3.0

smoothingParameter =0.999999;
DAlphaVsLambdaLESplineStatic_Dy3p0 = csaps( ...
    DeltaAlphaCLmax_vs_LambdaLE_dy3p0(:,1), ...
    DeltaAlphaCLmax_vs_LambdaLE_dy3p0(:,2), ...
    smoothingParameter ...
    );

DAlphaVsLambdaLEStatic_Dy3p0 = ppval( ...
    DAlphaVsLambdaLESplineStatic_Dy3p0, ...
    lambdaLEVector_deg ...
    );

%% dy/c = 4.0

smoothingParameter = 0.999999;
DAlphaVsLambdaLESplineStatic_Dy4p0 = csaps( ...
    DeltaAlphaCLmax_vs_LambdaLE_dy4p0(:,1), ...
    DeltaAlphaCLmax_vs_LambdaLE_dy4p0(:,2), ...
```

```
    smoothingParameter ...
    );

DAlphaVsLambdaLEStatic_Dy4p0 = ppval( ...
    DAlphaVsLambdaLESplineStatic_Dy4p0, ...
    lambdaLEVector_deg ...
    );


%% Plots
figure(1)
plot ( ...
    lambdaLEVector_deg, DAlphaVsLambdaLEStatic_Dy1p2, '-*b' ... , ...
 );
 hold on

 plot ( ...
    lambdaLEVector_deg, DAlphaVsLambdaLEStatic_Dy2p0, '-b' ... , ...
 );

hold on

plot ( ...
    lambdaLEVector_deg, DAlphaVsLambdaLEStatic_Dy3p0, '*b' ... , ...
 );

hold on

plot ( ...
    lambdaLEVector_deg, DAlphaVsLambdaLEStatic_Dy4p0, 'b' ... , ...
 );

 xlabel('\Lambda_{le} (deg)'); ylabel('\Delta\alpha_{C_{L,max}}');
 title('Angle of attack increment for wing maximum lift in subsonic flight');
  legend('\Delta y/c = 1.2', '\Delta y/c = 2.0', '\Delta y/c = 3.0','\Delta y/c = 4.0');
 axis([0 50 0 9]);
 grid on;



%% preparing output to HDF

% dy/c
dyVector = [ ...
    1.2;2.0;3.0;4.0 ...
    ];

%columns --> curves
myData = [ ...
    DAlphaVsLambdaLEStatic_Dy1p2,...
        DAlphaVsLambdaLEStatic_Dy2p0, ... % -> 2
        DAlphaVsLambdaLEStatic_Dy3p0, ... % -> 3
        DAlphaVsLambdaLEStatic_Dy4p0];    % -> 4

hdfFileName = 'DAlphaVsLambdaLEVsDy.h5';

if ( exist(hdfFileName, 'file') )
    fprintf('file %s exists, deleting and creating a new one\n', hdfFileName);
    delete(hdfFileName)
else
    fprintf('Creating new file %s\n', hdfFileName);
end

% Dataset: data
h5create(hdfFileName, '/DAlphaVsLambdaLEVsDy/data', size(myData'));
h5write(hdfFileName, '/DAlphaVsLambdaLEVsDy/data', myData');

% Dataset: var_0
h5create(hdfFileName, '/DAlphaVsLambdaLEVsDy/var_0', size(dyVector'));
h5write(hdfFileName, '/DAlphaVsLambdaLEVsDy/var_0', dyVector');

% Dataset: var_1
h5create(hdfFileName, '/DAlphaVsLambdaLEVsDy/var_1', size(lambdaLEVector_deg'));
h5write(hdfFileName, '/DAlphaVsLambdaLEVsDy/var_1', lambdaLEVector_deg');
```

This script plot the graph after digitization. In this way it's possible to compare the initial graph and the digitized one.

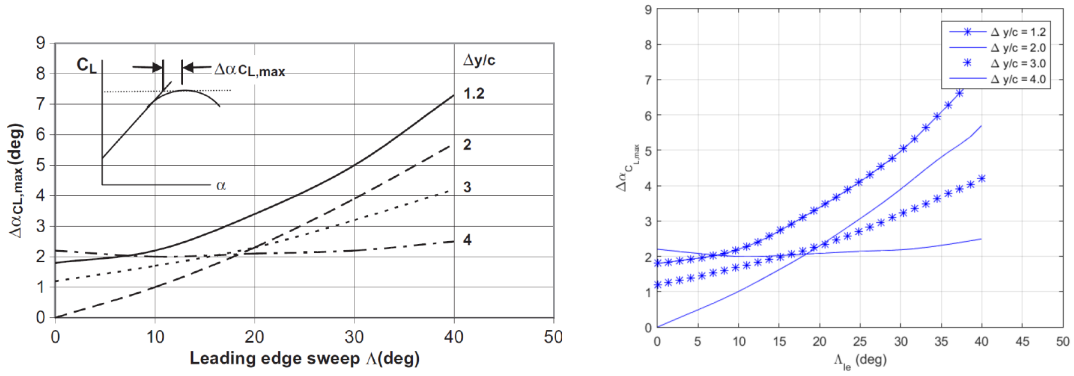


Figure A.2: Chart and its digitization.

Appendix B

Java Reference

B.1 Packages

A Java package is a mechanism for organizing Java classes into namespaces, providing modular programming. Java packages can be stored in compressed files called JAR files, allowing classes to be downloaded faster as groups rather than individually. Programmers also typically use packages to organize classes belonging to the same category or providing similar functionality.

A package provides a unique namespace for the types it contains. In general, a namespace is a container for a set of identifiers (also known as symbols, names). Namespaces provide a level of direction to specific identifiers, thus making it possible to distinguish between identifiers with the same exact name. For example, a surname could be thought of as a namespace that makes it possible to distinguish people who have the same given name. In computer programming, namespaces are typically employed for the purpose of grouping symbols and identifiers around a particular functionality.

B.2 Documentation

The doc comments, which are a particular type of comments provided by Java, are automatically recognized by the most popular IDE programs; this helps the developer to quickly get an idea of the actions performed by each method, the parameter which has to be passed to it and the what it returns to the user.

Glossary

Aicraft Construction Reference Frame The reference frame which has its origin in the fuselage forwardmost point, the x-axis pointing from the nose to the tail, the y-axis from fuselage plane of symmetry to the right wing (from the pilot's point of view) and the z-axis from pilot's feet to pilot's head.

client code the code where the code in question will be effectively exploited.

DATCOM Digital Datcom is a computer program which calculates static stability, high lift and control, and dynamic derivative characteristics using the methods contained in the USAF Stability and Control Datcom (Data Compendium). Configuration geometry, attitude, and Mach range capabilities are consistent with those accommodated by the Datcom. The program contains a trim option that computes control deflections and aerodynamic increments for vehicle trim at subsonic Mach numbers.

Enumeration The java.util.Enumeration interface represents a special data type that enables for a variable to be a set of predefined constants. The variable must be equal to one of the values that have been predefined for it. Because they are constants, the names of an enum type's fields are in uppercase letters..

Graphical User Interface In computing, a Graphical User Interface is a type of interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, as opposed to text-based interfaces, typed command labels or text navigation.

Integrated Development Environment An integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools and a debugger. Most modern IDEs have an intelligent code completion. Some IDEs contain a compiler, interpreter, or both, such as NetBeans and Eclipse; others do not, such as SharpDevelop and Lazarus. The boundary between an integrated development environment and other parts of the broader software development environment is not well-defined. Sometimes a version control system, or various tools to simplify the construction of a Graphical User Interface (GUI), are integrated. Many modern IDEs also have a class browser, an object browser, and a class hierarchy diagram, for use in object-oriented software development.

Interface In the Java programming language, an interface is a reference type, similar to a class, that can contain only constants, method signatures, default methods, static methods, and nested types. Method bodies exist only for default methods and static methods. Interfaces cannot be instantiated—they can only be implemented by classes or extended by other interfaces..

Knowledge-Based Engineering KBE is essentially engineering on the basis of knowledge models. A knowledge model uses knowledge representation to represent the artifacts of the design process (as well as the process itself) rather than, or in addition to, conventional programming and database techniques. KBE can have a wide scope that covers the full range of activities related to Product Lifecycle Management and Multidisciplinary Design Optimization. KBE's scope includes design, analysis, manufacturing, and support.

List The java.util.List interface is a subtype of the java.util.Collection interface. It represents an ordered list of objects, meaning you can access the elements of a List in a specific order, and by an index too. You can also add the same element more than once to a List.

Map The java.util.Map interface represents a mapping between a key and a value. The Map interface is not a subtype of the Collection interface. Therefore it behaves a bit different from the rest of the collection types.

Multi-disciplinary Design Optimization Multi-disciplinary design optimization (MDO) is a field of engineering that uses optimization methods to solve design problems incorporating a number of disciplines.MDO allows designers to incorporate all relevant disciplines simultaneously. The optimum of the simultaneous problem is superior to the design found by optimizing each discipline sequentially, since it can exploit the interactions between the disciplines. However, including all disciplines simultaneously significantly increases the complexity of the problem.

parsing Parsing or syntactic analysis is the process of analysing a string of symbols, either in natural language or in computer languages, conforming to the rules of a formal grammar.

static method The term static means that the method is available at the Class level, and so does not require that an object is instantiated before it's called.

user developer The term refers to the developer which will use a method without being interested in how the method performs the required action. This is the case of a utility method: the developer is the one who writes the method, while the user developer is who uses that method to accomplish some action which requires the functionality provided by the utility method. It has to be noticed that the user developer and the developer can be the same person.

Acronyms

ACRF Aicraft Construction Reference Frame.

ADOpT Aircraft Design and Optimization Tool.

AIAA American Institute of Aeronautics and Astronautics.

BRF Body Reference Frame.

CAD Computer-Aided Design.

FAA Federal Aviation Administration.

GC Gravity Center.

GNC Guidance Navigation and Control.

GUI Graphical User Interface.

GUI Graphical User Interface.

IDE Integrated Development Environment.

IDE Integrated Development Environment.

JPAD Java Program toolchain for Aircraft Design.

KBE Knowledge-Based Engineering.

LRF Local Reference Frame.

MAC Mean Aerodynamic Chord.

MAPE Mean Absolute Percentage Error.

MDO Multi-disciplinary Design Optimization.

MLW Maximum Landing Weight.

MSL Mean Sea Level.

MTOW Maximum Take Off Weight.

MZFW Maximum Zero Fuel Weight.

List of symbols

$()_{\mathbf{H}}$ quantity related to the horizontal tail.

$()_{\mathbf{LG}}$ quantity related to the landing gear.

$()_{\mathbf{N}}$ quantity related to the nacelle.

$()_{\mathbf{S}}$ quantity related to systems.

$()_{\mathbf{V}}$ quantity related to the vertical tail.

$()_{\mathbf{F}}$ quantity related to the fuselage.

$()_{\mathbf{WF}}$ quantity related to the wing-fuselage configuration.

D aerodynamic drag.

CG Center of Gravity.

$\vec{g}$ gravitational acceleration.

$i_{\mathbf{H}}$ the angle between the horizontal tail root chord and the ACRF x-axis.

$i_{\mathbf{W}}$ the angle between the wing root chord and the ACRF x-axis.

L aerodynamic lift.

T Thrust.

M Mach number.

$\mathbf{AR}$ aspect ratio.

c chord.

d diameter.

l length.

m mass, in kg or lb.

$n_{\mathbf{lim}}$ limit load factor.

$n_{\mathbf{ult}}$ ultimate load factor.

b span.

S surface.

Λ sweep.

λ taper ratio.

t thickness.

V scalar velocity.

W weight, in Newtons.

m_{W} wing mass.

q dynamic pressure.

Re Reynolds number (evaluated with respect to $\bar{c}$).

$\alpha_{\mathbf{W}}$ angolo d'attacco riferito alla corda di radice dell'ala.

β sideslip angle.

ρ air density.

Bibliography

[1] *A Java desktop application for Aircraft Preliminary Design*. Naples, 2015.

[2] I. H. Abbott and A. E. von Doenhoff. *Theory of Airfoil Sections*. Dover, 1959.

[3] Joe Walnes et al. *XStream*. URL: http://xstream.codehaus.org/.

[4] Guillaume Alleon. *occjava*. URL: http://jcae.sourceforge.net/occjava.html.

[5] L. Attanasio. «Development of a Java Application for Parametric Aircraft Design». Master thesis. University of Naples "Federico II", 2014.

[6] L. Attanasio. *Fuselage aerodynamic prediction methods*. Naples, 2015.

[7] Various Authors. *ATR 72. Aircraft description*. URL: https://en.wikipedia.org/wiki/ATR_72.

[8] Various Authors. *Boeing 747. Aircraft description*. URL: https://en.wikipedia.org/wiki/Boeing_747.

[9] Various Authors. *ESDU-76003. Geometrical properties of cranked and straight tapered wing planforms*. London, 2001.

[10] Various Authors. *eXtensible Markup Language (XML)*. URL: https://en.wikipedia.org/wiki/XML.

[11] Various Authors. *Guava*. URL: https://github.com/google/guava.

[12] Various Authors. *Induced and Effective AOA for a Wing*. URL: http://gtae6343.wikia.com/wiki/Angle_of_Attack#cite_ref-1.

[13] Various Authors. *Java (programming language)*. URL: https://en.wikipedia.org/wiki/Java_(programming_language).

[14] Various Authors. *JFace*. URL: https://wiki.eclipse.org/JFace.

[15] Various Authors. *Model View Controller*. URL: https://en.wikipedia.org/wiki/Model%C3%A2%C2%80%C2%93view%C3%A2%C2%80%C2%93controller.

[16] Various Authors. *Plot Digitizer, description*. URL: http://plotdigitizer.sourceforge.net/.

[17] Various Authors. *SUAVE: AN AEROSPACE VEHICLE ENVIRONMENT FOR DESIGNING FUTURE AIRCRAFT*. URL: http://suave.stanford.edu/.

[18] Various Authors. *The Lift Coefficient*. 2006. URL: https://www.grc.nasa.gov/www/K-12/airplane/liftco.html.

[19] Various Authors. *The Standard Widget Toolkit*. URL: https://www.eclipse.org/swt/.

[20] Various Authors. *Window Builder*. URL: https://eclipse.org/windowbuilder/.

[21] Azeckoski. *ReflectUtils*. URL: https://code.google.com/p/reflectutils/.

[22] A. et al. Bilgin. *Graphviz*. URL: http://www.graphviz.org/.

[23] J. A. Jr Blackwell. *A Finite-Step Method for Calculation of Theoretical Load Distributions for Arbitrary Lifting-Surface Arrangements at Subsonic Speeds*. Washington, D.C., 1969.

[24] AIR FORCE TEST PILOT SCHOOL EDWARDS AFB CA. *Volume II. Flying Qualities Phase. Chapter 5: Longitudinal Static Stability.* published by the author, OCT 1990.

[25] M. Calcara. *Elementi di dinamica del velivolo*. Napoli: Edizioni CUEN, 1988. ISBN: 9788871460062. URL: http://books.google.it/books?id=HqBoAQAACAAJ.

[26] Jean-Marie Dautelle. *Jscience*. URL: http://jscience.org/.

[27] M. Drela and H. Youngren. *XFoil, Subsonic Airfoil Development System*. 2008. URL: http://web.mit.edu/drela/Public/web/xfoil/.

[28] Giuseppe Paduano Fabrizio Nicolosi. *BEHIND ADAS 1.0*. Napoli, 2011.

[29] The Apache Software Foundation. *Apache Commons Math*. URL: http://commons.apache.org/proper/commons-math/.

[30] The Apache Software Foundation. *Apache POI*. URL: http://poi.apache.org/.

[31] The Apache Software Foundation. *Apache projects*. URL: http://www.apache.org/.

[32] «Full-configuration drag estimation». In: Journal of Aircraft 47 (4) (2010).

[33] D. Gambardella. «Development of a Java-Based Framework for Aircraft Preliminary Design». Master thesis. University of Naples "Federico II", 2014.

[34] D. et al Gilbert. *JFreeChart*. URL: http://www.jfree.org/jfreechart/.

[35] R.R Gilruth and M.D. White. *NACA TR 711 - Analysis And Prediction Of Longitudinal Stability of Airplanes*.

[36] The HDF Group. *HDF*. URL: http://www.hdfgroup.org/HDF5/.

[37] M. Hepperle. *Applet* JavaFoil. 2006. URL: http://www.mh-aerotools.de/airfoils/javafoil.htm.

[38] W.F. Hilton. *High-speed Aerodynamics*. Longmans, Green, 1951. URL: https://books.google.it/books?id=%5C_RRFAAAAMAAJ.

[39] D. E. Hoak et al. *The USAF Stability and Control DATCOM*. Tech. rep. (Revised 1978). 1960.

[40] D. Howe. *Aircraft conceptual design synthesis*. Aerospace Series. Professional Engineering Publishing, 2000. ISBN: 9781860583018. URL: https://books.google.it/books?id=QJZTAAAAMAAJ.

[41] D. Howe. *Aircraft Loading and Structural Layout*. AIAA Education Series, 2004.

[42] J. Huwaldt. *1976 Standard Atmosphere Program*. URL: http://thehuwaldtfamily.org/java/Applications/StdAtmosphere/StdAtmosphere.html.

[43] E. N. Jacobs and K. E. Ward. *Interference of wing and fuselage from tests of 209 combinations in the NACA variable-density tunnel*. Tech. rep. 1936.

[44] L.R. Jenkinson, D. Rhodes, and P. Simpkin. *Civil jet aircraft design*. AIAA education series. Arnold, 1999. ISBN: 9780340741528. URL: https://books.google.it/books?id=6tMeAQAAIAAJ.

[45] Erik K. Antonsson Karl-Heinrich Grote. *Springer Handbook of Mechanical Engineering*. Grote Antonsson.

[46] K. Kawaguchi. *Args4j*. URL: http://args4j.kohsuke.org/.

[47] I. Kroo and R. Shevell. *Aircraft Design: Synthesis and Analysis*. 1999.

[48] Ajoy Kumar Kundu. *Aircraft Design*. Cambridge University Press, 2010.

[49] S. Ledru and C. Denizet. *JLatexMath*. URL: http://forge.scilab.org/index.php/p/jlatexmath/.

[50] W.H. Mason. «Analytic Models for Technology Integration in Aircraft Design». In: AIAA/AHS/ASEE (1990).

[51] B. W. McCormick. *Aerodynamics, Aeronautics, and Flight Mechanics*. John Wiley & Sons, 1979.

[52] Daniel Webster College Mohammad Sadraey. *Aircraft conceptual design*. 2012.

[53] H. Multhopp. *Aerodynamics of Fuselage*. 1942.

[54] M. M. Munk. *Aerodynamic Forces of Airship Hulls*. Tech. rep. 1924.

[55] Marcello R. Napolitano. *Aircraft Dynamics: From Modeling to Simulation*. John Wiley, 2012. ISBN: 9780470626672. URL: http://www.amazon.com/Aircraft-Dynamics-From-Modeling-Simulation/dp/0470626674.

[56] R. Nelson. *Flight Stability and Automatic Control*. McGraw-Hill, 1989.

[57] R.C. Nelson. *Flight Stability and Automatic Control*. Aerospace Science & Technology. McGraw-Hill International Editions, 1998. ISBN: 9780071158381. URL: https://books.google.it/books?id=Uzs8PgAACAAJ.

[58] L.M. Nicolai and G. Carichner. *Fundamentals of Aircraft and Airship Design*. AIAA education series v. 1. American Institute of Aeronautics and Astronautics, 2010. ISBN: 9781600867514. URL: https://books.google.it/books?id=qA0oAQAAMAAJ.

[59] M. Nita and D. Scholz. «Estimating Oswald Factor From Basic Aircraft Geometrical Parameters». In: *none* (2012).

[60] E. Obert. *Aerodynamic Design of Transport Aircraft*. IOS Press, 2009. ISBN: 9781607504078. URL: https://books.google.it/books?id=axPvAgAAQBAJ.

[61] Oracle. *1.2 Design Goals of the Java Programming Language*. 2013.

[62] C. D. Perkins and R. E. Hage. *Airplane Performance Stability and Control*. New York: John Wiley & Sons, 1949. ISBN: 9780471680468. URL: http://books.google.it/books?id=DHxTAAAAMAAJ.

[63] W. F. Phillips. «Lifting-Line Analysis for Twisted Wings and Washout-Optimized Wings». In: *Journal of Aircraft* Vol. 41.No. 1 (Jan.–Feb. 2004).

[64] Warren F. Phillips. *Mechanics of Flight*. John Wiley and Sons, 2004. URL: https://books.google.it/books/about/Mechanics_of_Flight.html?id=6-_iGbJHM-8C&redir_esc=y.

[65] D.P. Raymer. *Aircraft Design: A Conceptual Approach*. 1992.

[66] D.P. Raymer. *Aircraft Design / RDS-Student: A Conceptual Approach*. AIAA Education Series. Amer Inst of Aeronautics &, 2013. ISBN: 9781600869211. URL: https://books.google.it/books?id=bvxllgEACAAJ.

[67] J. Roskam. *Airplane Design*. Airplane Design pt. 5. DARcorporation, 1999. ISBN: 9781884885501. URL: https://books.google.it/books?id=mMU47Ld7yQkC.

[68] J. Roskam. *Airplane Design*. Airplane Design pt. 8. DARcorporation, 2002. ISBN: 9781884885556. URL: https://books.google.it/books?id=GIHHFkd829cC.

[69] J. Roskam. *Airplane Flight Dynamics and Automatic Flight Controls*. DARcorporation, 2001.

[70] J. Roskam. *Airplane Flight Dynamics and Automatic Flight Controls*. Airplane Flight Dynamics and Automatic Flight Controls. Darcorporation, 2003. ISBN: 9781884885181. URL: http://books.google.it/books?id=pWUAzy87zLYC.

[71] J. Roskam. *Methods for Estimating Stability and Control Derivatives of Conventional Subsonic Airplanes*. published by the author, 1971.

[72] M.H. Sadraey. *Aircraft Design: A Systems Engineering Approach*. Aerospace Series. Wiley, 2012. ISBN: 9781118352809. URL: https://books.google.it/books?id=VT-Tc3Tx5aEC.

[73] OPEN CASCADE S.A.S. *OpenCASCADE*. URL: http://www.opencascade.org/.

[74] David Schmidt. *Modern Flight Dynamics*. McGraw-Hill, 2012. ISBN: 97800711339811.

[75] P.M. Sforza. *Commercial Airplane Design Principles*. Elsevier Science, 2014. ISBN: 9780124199774. URL: https://books.google.it/books?id=knhHAgAAQBAJ.

[76] Abe Silverstein and S. Katzoff. *Design Charts for Predicting Downwash Angles and Wake Characteristics behind Plain and Flapped Wings*. Tech. rep. 1939.

[77] *The Java Language Specification*. 2015.

[78] *Theoretical Calculation of the Effect of The Fuselage on the Spanwise Lift Distribution on a Wing*. Washington, D.C., 1952.

[79] E. Torenbeek. *Advanced Aircraft Design: Conceptual Design, Technology and Optimization of Subsonic Civil Airplanes*. Aerospace Series. Wiley, 2013. ISBN: 9781118568095. URL: https://books.google.it/books?id=C0gUyoYewhoC.

[80] E. Torenbeek. *Synthesis of Subsonic Aircraft Design*. 1976.

[81] E. Torenbeek. *Synthesis of Subsonic Airplane Design: An Introduction to the Preliminary Design of Subsonic General Aviation and Transport Aircraft, with Emphasis on Layout, Aerodynamic Design, Propulsion and Performance*. Springer, 1982. ISBN: 9789024727247. URL: https://books.google.it/books?id=NG2%5C_qiSjmMEC.

[82] Dimitri Van Heesch. *Doxygen*. 1997-2014. URL: www.doxygen.org.

[83] S. R. Vukelich and J. E. Williams. *The USAF Stability and Control Digital Datcom*. AFFDL-TR-79-3032. Updated by Public Domain Aeronautical Software 1999. Apr. 1979. Volume I.

[84] A.D. Young. *The Aerodynamic Characteristics of Flaps*. Technical Report. 1947.