Università degli Studi di Napoli Federico II

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

Tesi di Laurea Triennale
in
Ingegneria Aerospaziale

Assessment of minimum unstick speed of a complete Aircraft using JPAD (Java toolchain Program for Aircraft Design)

Relatore:
Prof. Ing. Agostino De Marco

Correlatore:
Ing. Manuela Ruocco

Candidato:
Bruno Spoti
Matricola N35/2018

Anno Accademico 2017/2018

Sono sempre i sogni a dare forma al mondo,
sono sempre i sogni a fare la realtà.

Abstract

The purpose of this Thesis work is to implement a specific Java routine in order to assess the minimum unstick speed of a complete aircraft. This has been done implementing new features in JPAD (Java toolchain Program for Aircraft Design), a Java-based framework conceived as a fast and efficient tool as a support in the preliminary design phases of an aircraft and during its optimization process.
The JPAD development originates in the Departement of Industrial Engineering of University of Naples "Federico II", where is still in progress. JPAD is a framework equiped with a graphic user interface (GUI) 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 structure of this thesis work is articulated in 3 chapter and 1 appendix.
First of all, the JPAD framework is presented.
The second chapter concerns a theoretical background and, starting from a brief introduction of take off speeds, it will move on the explanation of the ground effect and its implementation. This chapter ends with the presentation of Minimun Unstick Speed (VMU).
The third chapter regards a case study performed on a regional turboprop of 130 pax, concerning aerodynamic characteristics, high lift devices and ground effect both on wing lift curve and on downwash angle.
The development of Java functionalities, in JPAD, has made considering further developments thus to allow the possibility to introduce new methods.
At the end, in appendix A, it has been shown how to digitalizes a chart, create an HDF dataset and set up the database-reader class in JPAD.

Sommario

Lo scopo di questo lavoro di tesi è implementare una specifica routine Java per valutare la VMU di un aereo completo. Questo è stato fatto implementando nuove caratteristiche in JPAD (Java toolchain Program for Aircraft Design), un framework scritto in Java concepito come un veloce ed efficiente strumento a supporto nelle fasi preliminari di progettazione di un aereo e durante il suo processo di ottimizzazione.
Lo sviluppo di JPAD ha origine nel Dipartimento di Ingegneria Industriale dell'Università di Napoli "Federico II", dove è ancora in corso. JPAD è un software dotato di un'interfaccia grafica (GUI) e, al momento, è in grado di eseguire un'analisi multidisciplinare di un aeromobile i cui dati possono essere inseriti dall'utente, con un file XML, o caricati in memoria.

La struttura di questo lavoro di tesi è articolata in 3 capitoli e 1 appendice.
Prima di tutto, viene presentata la struttura di JPAD.
Il secondo capitolo riguarda un background teorico il quale, partendo da una breve introduzione della velocità di decollo, si sposterà sulla spiegazione dell'effetto suolo e sulla sua implementazione. Questo capitolo termina con la presentazione della VMU.
Il terzo capitolo riguarda un caso studio eseguito su un turboelica regionale di 130 passeggeri, riguardante le caratteristiche aerodinamiche, i dispositivi di ipersostentazione e l'effetto suolo sia sulla curva di portanza dell'ala che sull'angolo di downwash. Lo sviluppo delle nuove funzionalità Java, in JPAD, è stato fatto prendendo in considerazione la possibilità di ulteriori sviluppi per consentire l'introduzione di nuovi metodi.
Alla fine, nell'appendice A, viene mostrato come digitalizzare un grafico, creare un set di dati in formato .h5 e impostare la classe che legge di database in JPAD.

Contents

List of Figures

Chapter 1

Introduction to JPAD

Nowadays the preliminary design phase of an aircraft is becoming very challenging due to the need for more demanding requirements which deals with different fields of applications. In this perspective, there is a certain need for simple design tools both in aircraft industries and academic research groups which can perform fast and reliable multi-disciplinary analyses and optimizations.

This chapter provides a comprehensive overview of JPAD (Java toolchain of Programs for Aircraft Design), a Java-based open-source library conceived as a fast and efficient tool useful as support in the preliminary design phases of an aircraft, and during its optimization process. The library has been completely realized at the Department of Industrial Engineering of the University of Naples "Federico II" where is still in development.

One of the main features of JPAD lies in the smart management of both the aircraft parametric model, which is conceived as a set of interconnected and parameterized components, and the available analyses. The library has been developed with the purpose of simplify the composition of the input file for the user and doing fast analysis with a satisfying grade of accuracy. The following section will focus on the description of the software structure and its main advantages. Another key point is the possibility to easily interface JPAD with other external tools in order to achieve a higher level of accuracy.

The JPAD library is an alternative to a plethora of similar software tools, both freeware and commercial. Most of these tools have an important history, and many of them have been in use for decades. Some of them were conceived with poor software design criteria, have a rigid textual input and come with no visualization features. This is the main reason why JPAD has been developed paying a lot of attention to simplicity and flexibility. Moreover, it has been conceived as an open-source tool.

1.1 Software structure

To achieve a clear input file organization a considerable study has been done. The result is an input structure composed by different XML files with the purpose to allow users to easily manage all data needed to execute the desired analyses. In figure fig. 1.1 on the following page the entire structure of the software is schematized. It is possible to clearly note that there are two main blocks: input and core.

The input block is defined by two main parts: aircraft and analyses definitions. The

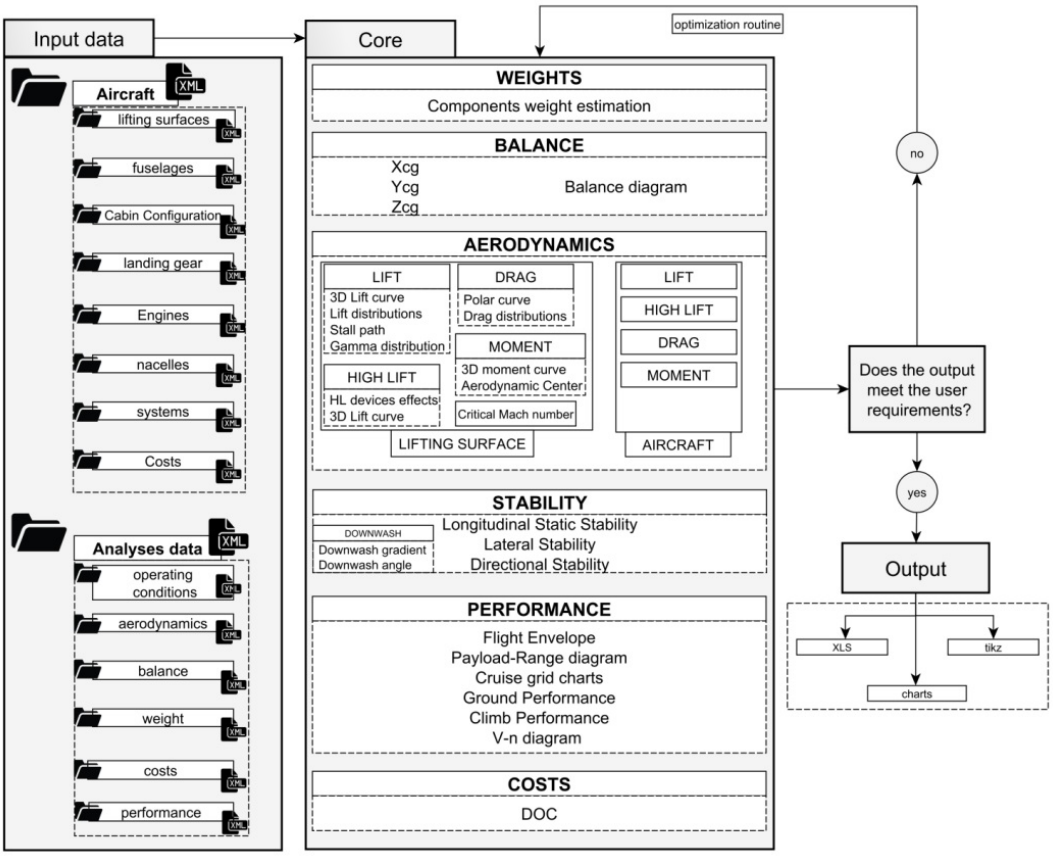

Figure 1.1: JPAD schematic flow-chart

```
<?xml version="1.0" encoding="utf-8"?>
<jpad_config>
                <!-- Input data, shared across analysis tasks -->
                <global_data>
                        <n_limit>2.5</n_limit>
                        <cruise_lift_coefficient>0.45</cruise_lift_coefficient>
                        <reference_range unit="nmi">825</reference_range>
                        <maximum_altitude_at_maximum_speed unit="ft">16000</
                            maximum_altitude_at_maximum_speed>
                        <maximum_cruise_mach_number>0.43</
                            maximum_cruise_mach_number>
                        <optimum_cruise_altitude unit="ft">16000</
                            optimum_cruise_altitude>
                        <optimum_cruise_mach_number>0.45</
                            optimum_cruise_mach_number>
                        <block_time unit="h">1.5</block_time>
                        <flight_time unit="h">1.35</flight_time>
                </global_data>
                <!-- Required analysis tasks -->
                <analyses id="JPAD Test analysis">
                        <weights
                                file="analysis_weights.xml"
                                method_fuselage="JENKINSON"
                                method_wing="ROSKAM"
                                method_htail="JENKINSON"
                                method_vtail="JENKINSON"
                                method_canard=""
                                method_nacelles="TORENBEEK_1976"
                                method_landing_gears="ROSKAM"
                                method_systems="TORENBEEK_2013"
                                /> <!-- method_XXX="AVERAGED" if not present -->
                        <balance
                                file="analysis_balance.xml"
                                method_fuselage=""
                                method_wing=""
                                method_htail=""
                                method_vtail=""
                                method_canard=""
                                /> <!-- method_XXX="AVERAGED" if not present -->
                        <aerodynamics file="analysis_aerodynamics.xml"/>
                        <performance file="analysis_performance.xml"/>
                        <costs file="analysis_costs.xml"/>
                </analyses>
</jpad_config>
```

Figure 1.2: An example of the analysis.xml file

Aircraft.xml

```
<?xml version="1.0" encoding="utf-8"?>
<jpad_config>
        <aircraft id="JPAD Test Aircraft DAF-2016" type="JET" regulations="FAR_25">
                <global_data>
                        <cabin_configuration file="cabin_configuration.xml">
                        </cabin_configuration>
                </global_data>
                <lifting_surfaces>
                        <wing file="wing.xml">
                                <position>
                                        <x unit="m">12.0</x>
                                        <y unit="m">0.0</y>
                                        <z unit="m">-0.5</z>
                                </position>
                                <rigging_angle unit="deg">2.0</rigging_angle>
                        </wing>
                        <horizontal_tail file="htail.xml">
                        <vertical_tail file="vtail.xml">
                </lifting_surfaces>
                <fuselages>
                        <fuselage file="fuselage.xml">
                </fuselages>
                <power_plant>
                        <engine file="engineTurbofan.xml">
                        <engine file="engineTurbofan.xml">
                </power_plant>
                <nacelles>
                        <nacelle file="nacelle.xml">
                        <nacelle file="nacelle.xml">
                </nacelles>
                <landing_gears file="landing_gear.xml">
                </landing_gears>
                <systems file="systems.xml">
                </systems>
</aircraft>
</jpad_config>
```

Wing.xml

```
<?xml version="1.0" encoding="utf-8"?>
<jpad_config>
        <aircraft id="JPAD Test Aircraft DAF-2016" type="JET" regulations="FAR_25">
                <global_data>
                        <main_spar_non_dimensional_position type="PERCENT_CHORD"
                            ref_to="LOCAL_CHORD">0.25</
                            main_spar_non_dimensional_position>
                        <secondary_spar_non_dimensional_position type="
                            PERCENT_CHORD" ref_to="LOCAL_CHORD">0.55</
                            secondary_spar_non_dimensional_position>
                        <composite_correction_factor>0.1</
                            composite_correction_factor>
                        <roughness unit="m">0.405e-5</roughness>
                </global_data>
                <equivalent_wing>
                        <surface unit="m^{2}">36.9</surface>
                        <aspect_ratio>13.09</aspect_ratio>
                        <non_dimensional_span_station_kink>0.363</
                            non_dimensional_span_station_kink>
                        <sweep_quarter_chord unit="deg">12.56</sweep_quarter_chord>
                        <twist_at_tip unit="deg">-2.5</twist_at_tip>
                        <dihedral unit="deg">2.5</dihedral>
                        <taper_ratio>0.556</taper_ratio>
                        <x_offset_root_chord_leading_edge unit="m">0.477</
                            x_offset_root_chord_leading_edge>
                        <x_offset_root_chord_trailing_edge unit="m">0.367</
                            x_offset_root_chord_trailing_edge>
                        <airfoils>
                                <airfoil_root file="naca63209.xml"/>
                                <airfoil_kink file="naca63209.xml"/>
                                <airfoil_tip file="naca63209.xml"/>
                        </airfoils>
                </equivalent_wing>
                <panels>
                        <panel id="Inner panel">
                        </panel>
                        <panel id="Outer panel" linked_to="Inner panel">
                        </panel>
                </panels>
                <symmetric_flaps>
                </symmetric_flaps>
                <slats>
                </slats>
                <asymmetric_flaps>
                </asymmetric_flaps>
                <spoilers>
                </spoilers>
</aircraft>
</jpad_config>
```

Figure 1.3: An extract from a general Aircraft.xml input file

first one defines the aircraft model in parametric way using a main file (Aircraft.xml, see figure 1.3) which collects all the components, linking them to their related XML file (i.e. fuselage.xml, vtail.xml, and so on) which contains all geometrical data.

The second one defines all necessary data for each analysis presents into core module (see figure 1.2). Since the aircraft model contains only geometrical data, it is necessary to define several further data referred to each analysis.

The structure described above allows to generate different aircrafts, or different configurations of the same model, combining different components. The possibility to generate a series of different aircrafts in a simple and fast way, allows to easily perform comparisons between these latter. For example, assuming different wings and engines, it is possible to estimate the effects that some design parameters have on a specific output. Figure fig. 1.4 shows how the FAR-25 take-off field length behaves with different values of the wing surface and the engine static thrust at fixed aircraft maximum take-off weight. This feature plays also a key role in the optimization process described in figure 1.1.

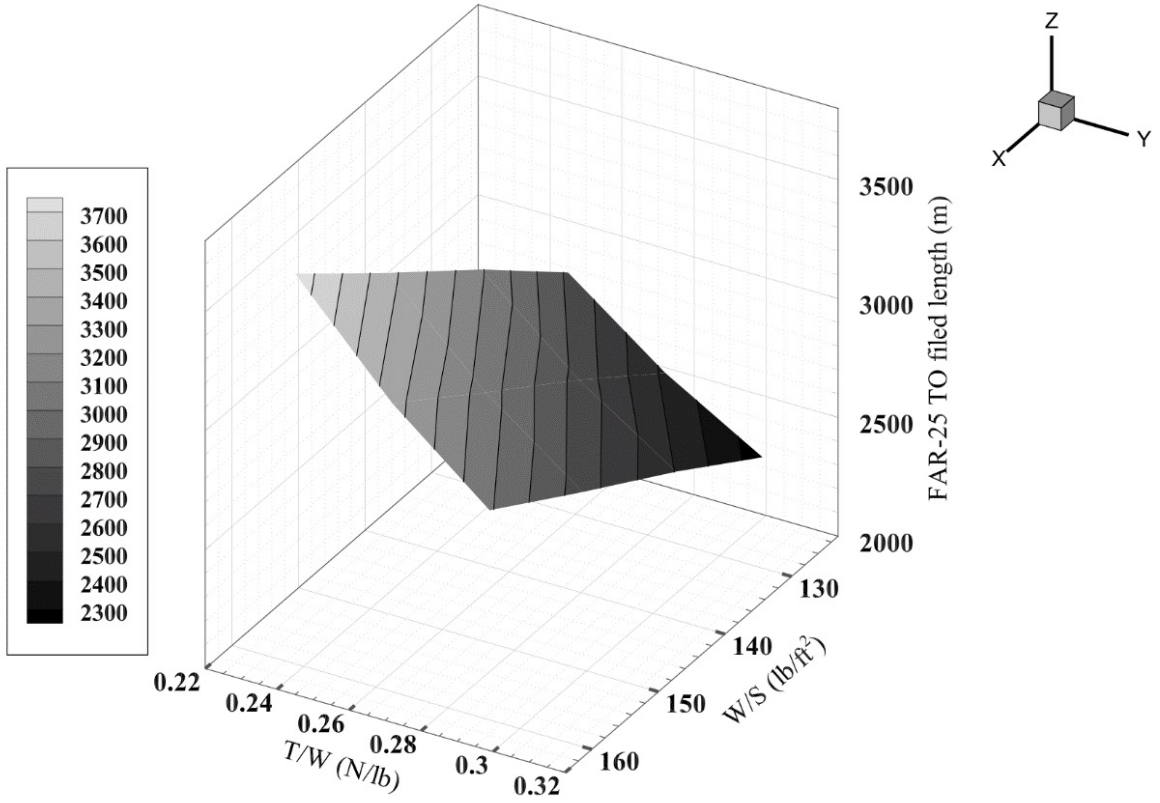

Figure 1.4: FAR-25 take-off field length at different wing loadings and thrust-weight ratios

In the same way, it is possible to perform a complete analysis (those present into core block in figure 1.1), or a specific one, combining different analyses files. This allows an easier evaluation of generic cost function during optimization tasks, resulting in reduced amount of computational costs required for this kind of operations.

Besides the input, the second main block is the core, which manages all the available analyses. This contains several independent modules, as shown in the figure 1.1, that deals with following application fields.

- Weights: estimates the aircraft weight breakdown starting from a first guess max-

imum take-off weight and some mission requirements. In particular, it evaluates each aircraft component mass using well-known semi-empirical equations.

- Balance: estimates the centre of gravity position related to each weight condition and draws the balance diagram.
- Aerodynamics and Stability: the aerodynamics module estimates all the aerodynamic characteristics concerning lift, drag and moments coefficients at different operating conditions for each aircraft component (wing, tails, fuselage and nacelles), whereas the stability module gives useful data about static stability of the whole aircraft.
- Performance: evaluates most important aircraft performance such as Payload-Range diagram, mission profile, cruise flight envelope, ground performance, climb performance and the cruise grid chart.
- Costs: estimates the DOC (Direct Operating Costs) breakdown.

Finally, JPAD allows to obtain different kind of output: charts and data in XLS format.

1.2 The Java Language

Java was developed by Sun Microsystems, a company that was incorporated in Oracle from a few years. This programming language is a general-purpose, concurrent, class-based, object-oriented language. It is designed to be simple enough that many programmers can achieve fluency in the language [4].

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 [2].

There were five primary goals in the creation of the Java language [6]:

- it must be “simple, object-oriented, and familiar”;
- it must be “robust and secure”;
- it must be “architecture-neutral and portable”;
- it must execute with “high performance”;
- it must be “interpreted, threaded, and dynamic”.

Actually Java is the most used programming language according to TIOBE (see figure fig. 1.5 on the following page). 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.

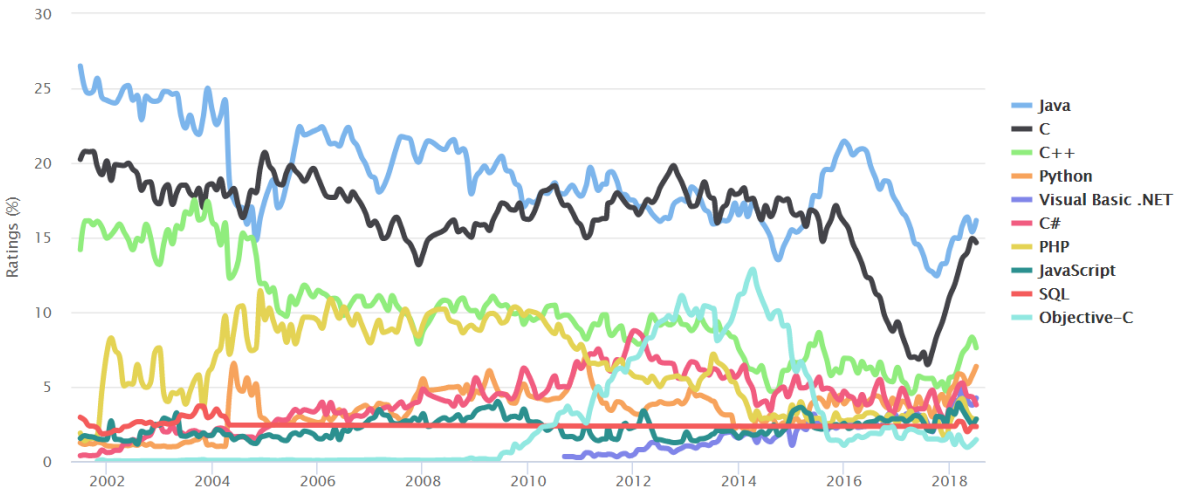

Figure 1.5: TIOBE Programming Community index (www.tiobe.com)

1.3 Java choice

The choice of Java as 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 is object oriented;
- 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; moreover 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 has been chosen to develop JPAD.

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 analyse the entire aircraft is very large (presently the whole project counts more than 10 millions 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.

Chapter 2

Theoretical background

Takeoff speeds are a safety key element for takeoff, and enable pilot situational awareness and decision-making in this very dynamic situation. The use of erroneous takeoff speeds can lead to tail strikes, high-speed rejected takeoff or initial climb with degraded performance.

2.1 Take off speeds

In this section a brief introduction to takeoff speeds will be provided [1].
In aviation, these speeds are standard terms used to define airspeeds important or useful to the operation of all aircraft. They are derived from data obtained by aircraft designers and manufacturers during flight testing and verified in most countries by government flight inspectors during aircraft type-certification testing. Using them is considered a best practice to maximize aviation safety, aircraft performance or both. Takeoff speeds are calculated prior to a take off in accordance with aircraft weight, environmental factors etc.

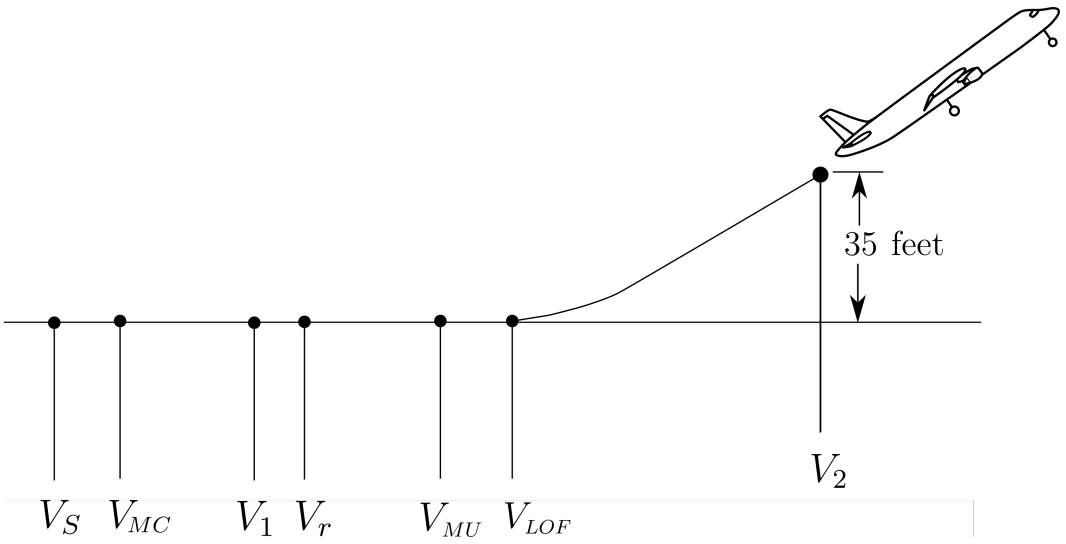

Figure 2.1: Takeoff speeds

Velocity of stall V_S

Aircraft stall speed during take off run.

Velocity of Minimum Control V_{MC}

During the takeoff roll, it is of utmost importance to know the minimum speed at which the aircraft will remain controllable, in the event of an engine failure on ground. This is because, in such a case, and if the takeoff is continued, only the rudder will be able to counteract the yaw moment that is generated by asymmetric engine(s) thrust. According to current regulations, the minimum speed at which an aircraft is defined to be "controllable" (lateral excursion lower than 30 feet) after an engine failure on ground, is referred to as V_{MC} (Velocity of Minimum Control on Ground).

Decision Speed V_1

V_1 is the maximum speed at which a rejected takeoff can be initiated, in the event of an emergency. V_1 is also the minimum speed at which a pilot can continue a takeoff after an engine failure. If an engine failure is detected after V_1, the takeoff must be continued. This implies that the aircraft must be controllable on ground. Therefore, V1 is always greater than V_{MC}.

Rotation speed V_r

Speed at which planes with a tricycle trolley lift the front wheel off the ground during take off.

Velocity of Minimum Unstick V_{MU}

V_{MU} is achieved by pitching the aircraft up to the maximum (tail on the runway, for aircraft that are are geometrically-limited) during the takeoff roll (Refer to Figure 2.2). The speed at which the aircraft first lifts off is V_{MU}. Therefore, lift-off is not possible prior to V_{MU}.

Figure 2.2: V_{MU} Flight Test on an Airbus A330

Lift-off speed V_{LOF}

Effective take off speed. It is assessed by adding a 10% to the Minimum unstick speed with all the operating engines or a 5% with an inoperative engine.

Takeoff safety speed V_2

V_2 is the speed at which the aircraft may safely be climbed with one engine inoperative. This speed is nicknamed a "take off safety speed"; it is the speed an aircraft with one engine inoperative must be able to attain in order to leave the runway and get 35 feet off the ground at the end of the runway, maintaining a 200 ft/min climb thereafter. This is the lowest speed at which the aircraft complies with the handling criteria associated with a climb after a take off, followed by the failure of an engine.

2.2 Ground effect

Due to the fact that during the takeoff the aircraft is on the ground, in order to asses V_{MU}, it is important to consider the ground effect. This generally become measurable at a height above the ground of one wing-span and increase in magnitude as the height above the ground decreases. Both theoretical and experimental investigations indicate that ground proximity produces an increase in the lift-curve slope, a decrease in drag, and a reduction of nose-up pitching moment for most aircraft planforms in the clean configuration. However, high-lift configurations deviate from this trend in that the ground effect tends to reduce the lift-curve slope [7].

The majority of the theoretical approaches analyzing ground effects employ an image-vortex theory to represent the ground plane. The salient aspects of this theory are discussed below.

Away from the ground plane, the downwash of the two trailing vortices contributes to the wing drag die to lift by rotating the force vector rearward.
However, near the ground plane, the trailing vortices of the image vortex system have an upwash component.

This upwash velocity component reduces the downward rotation of the flow direction caused by the wing trailing vortices, thus decreasing the wing drag due to lift.

The ground effects on lift are determined somewhat by the planform of the configuration. For low-aspect-ratio delta configurations, the general trend is a constant increase in C_L due to ground effect, as shown in figure 2.3. However, transport-type configurations show quite a different trend, as presented in figure 2.4.
This trend is dependent upon tie type of high-lift system employed.

2.3 Datcom methods

For most vehicles, calculating the change in lift due to ground effects consists of evaluating two components:

1. the change in wing-body lift;

2. the change in tail-body lift due to the effects of downwash.

The change in tail-body lift due to the presence of the ground is generally small in comparison to the downwash effects and is neglected in the Datcom methods.
Both of the Datcom methods presented require the user to construct wing-body and

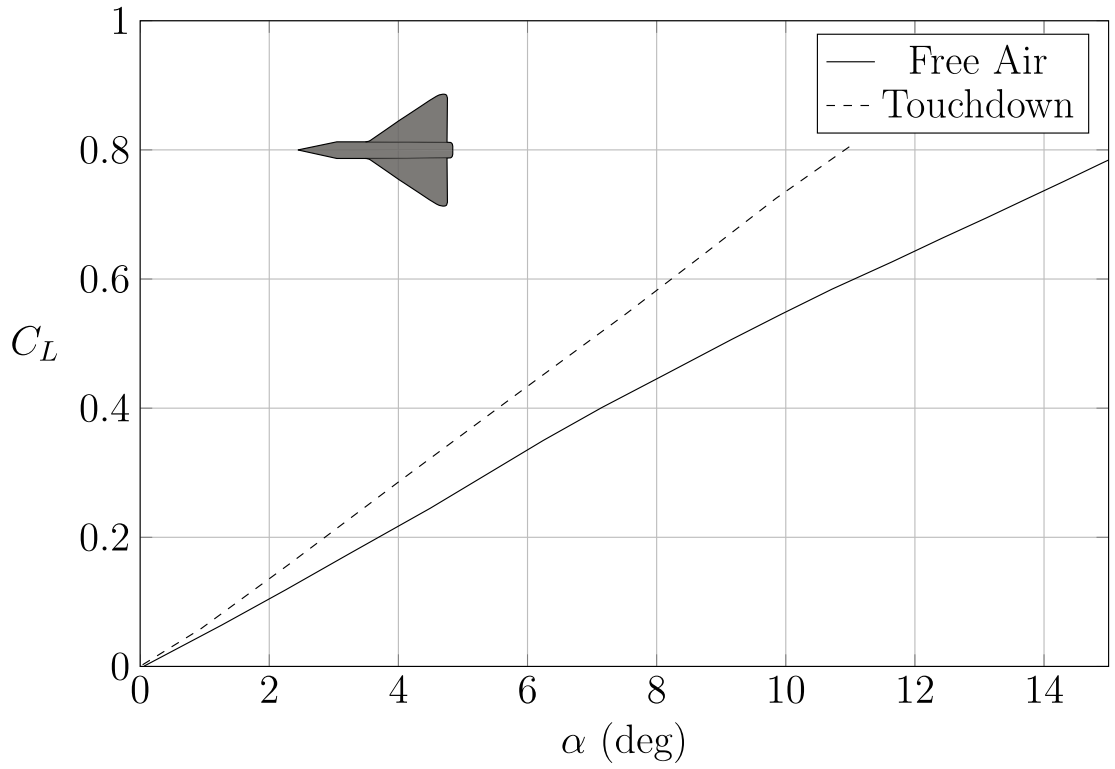

Figure 2.3: Ground effect on 55° delta configuration

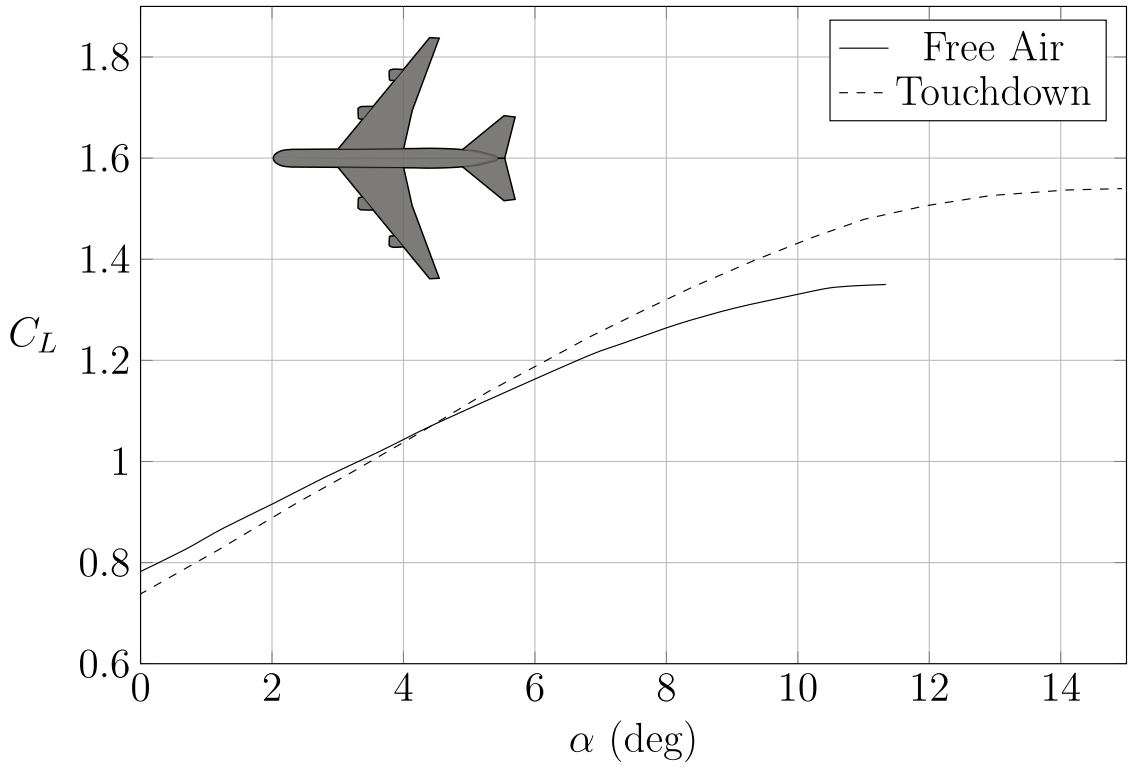

Figure 2.4: Ground effect on a jet transport configuration

tail-body lift curves in ground effect based on their corresponding free-air lift curves. Equations are given that calculate the change in angie of attack due to ground effect at a constant lift coefficient. The ground-effect lift curves are then constructed by shifting the free-air lift curves at every C_L by the corresponding increment in angle of attack due to ground effect at constant lift coefficients.

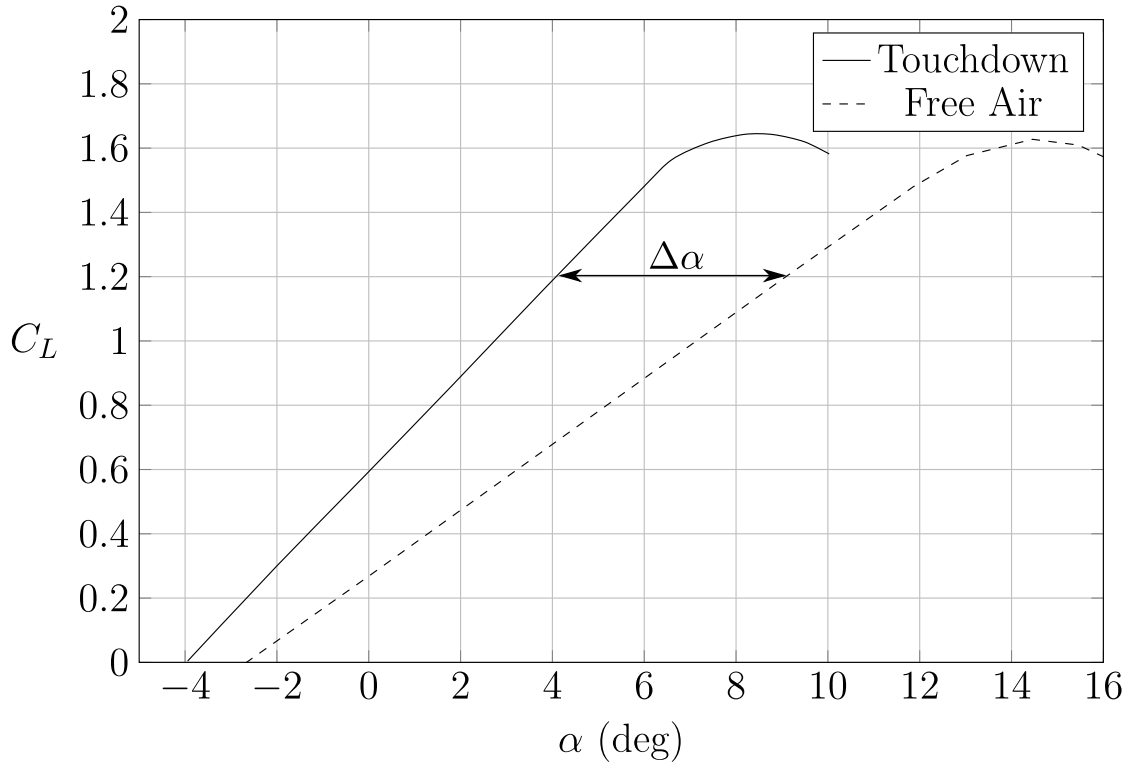

Figure 2.5: Ground effect on a jet transport configuration

Method I

This method estimates the ground effects on lift in the linear-lift range for a subsonic transport configuration. It includes the effects of taper ratio, sweep-back, dihedral, and flap deflection, while neglecting the effects of wing thickness since they are generally small. The wing-flap effects are valid only for split and slotted flaps as they are accounted for by empirical curves.
The change in wing-body angle of attack at a constant lift coefficient due to ground effect with respect to the out-of-ground-effect lift curve is given by:

$$\begin{aligned}(\Delta\alpha)_G = &-\left[\frac{9.12}{\mathcal{AR}} + 7.16\left(\frac{c_r}{b}\right)\right]\left(C_{L_f}\right)_{WB} x + \\ &-\frac{\mathcal{AR}}{2\left(C_{L_\alpha}\right)_{WB}}\left(\frac{c_r}{b}\right)\left(\frac{L}{L_0} - 1\right)\left(C_{L_f}\right)_{WB} r + \\ &-\frac{\left(\delta_f/50\right)^2}{\left(C_{L_\alpha}\right)_{WB}}\Delta\left(\Delta C_L\right)_{flap} \text{ (per deg)}\end{aligned} \tag{2.1}$$

where

$\Delta\left(\Delta C_L\right)_{flap}$	is an empirical factor to account for the effect of flaps and is obtained from fig. 2.6 as a function of the height of the quarter-chord point of the wing root chord above the ground,
$\mathbf{\mathcal{R}}$	is the wing aspect ratio,
$\frac{c_r}{b}$	is the ratio of wing root chord to wing span,
$\left(C_{L_f}\right)_{WB}$	is the wing-body lift coefficient including flap effects, out of ground effect,
x	accounts for the effects on lift due to the image trailing vortex and is obtained from fig. 2.8 on the following page as a function of wing geometry and the wing height above the ground,
$\frac{L}{L_0} - 1$	accounts for the effects on lift due to the image bound vortex and is obtained from fig. 2.9 on page 20 as a function of wing geometry, lift coefficient, and the height of the quarter-chord point of the wing root chord above the ground.
$(C_{L_\alpha})_{WB}$	is the wing body lift-curve slope, per degree, out of ground effect
r	accounts for the effect of finite span and is obtained from fig. 2.10 on page 20 as a function of wing height above the ground,

The first term in Equation 2.1 accounts for the effects of he trailing vortex, the second term for the effects of the trailing vortex, the second term for the effects of the bound vortex, and the third term for wing-flap effects. The method does not account for the effects of wing-leading-edge devices.

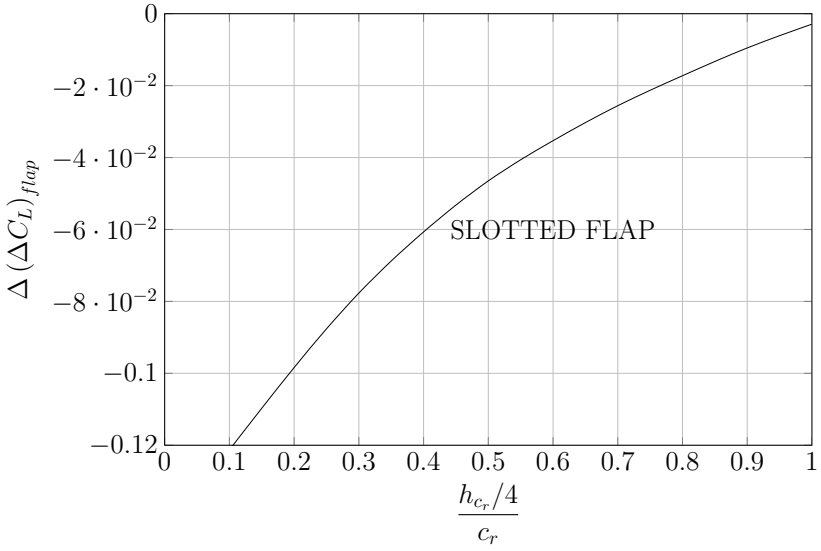

Figure 2.6: Effect of flap deflection on the ground influence on lift

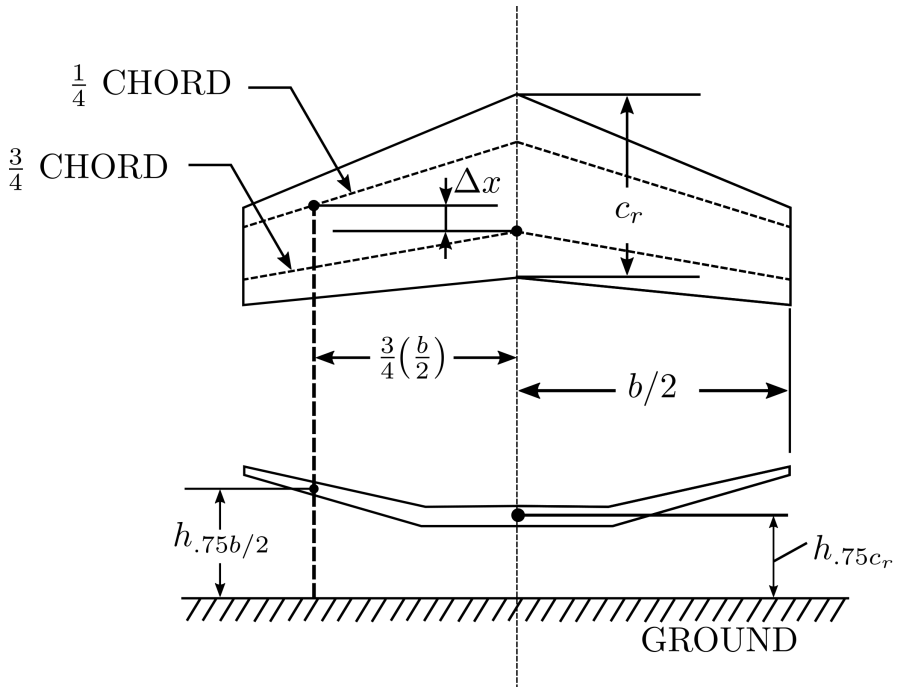

Figure 2.7: Description of geometric parameters

where

$$\frac{h}{b/2} = \frac{0.5\left(h_{.75b/2} + h_{.75c_r}\right)}{b/2}$$

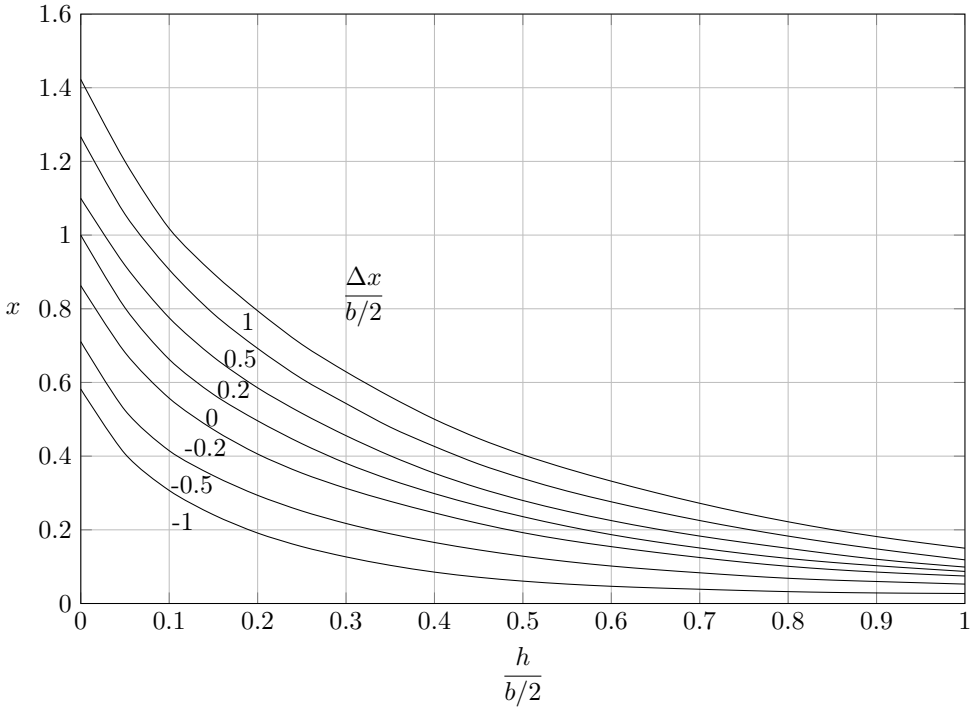

Figure 2.8: Parameter accounting for ground effect on lift due to trailing vortices

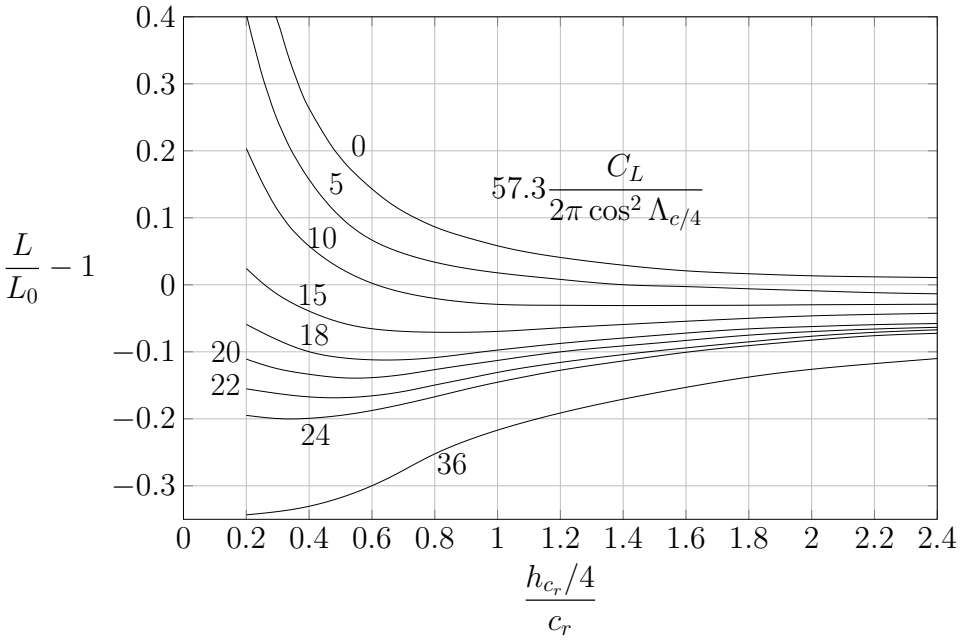

Figure 2.9: Parameter accounting for ground effect on lift due to bound vortices

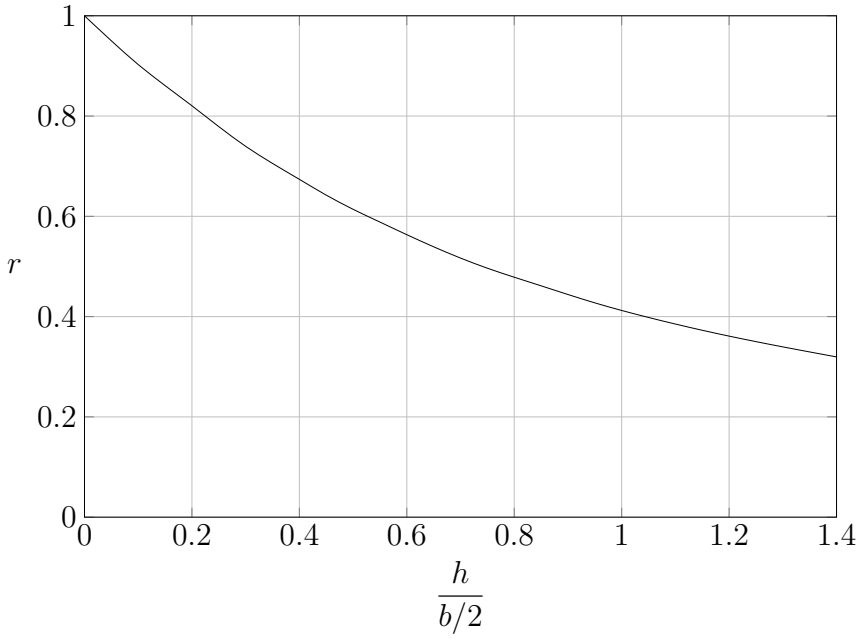

Figure 2.10: Factor accounting for finite span in ground effect

In the linear-lift region, the change in downwash (a decrease) on this tail-body due to ground effects is derived theoretically by representing the ground plane as an image-vortex system.
The change (a decrease) in tail-body downwash due to ground effects in the linear-lift range is given by:

$$(\Delta\varepsilon)_G = \varepsilon \left[\frac{b_{eff}^2 + 4\left(H_H + H\right)^2}{b_{eff}^2 + 4\left(H_H - H\right)^2} \right] \tag{2.2}$$

where

$(\Delta\varepsilon)_G$ is the difference between the downwash in free air and the downwash in ground effect,

ε is the downwash out of ground effect,

H is the height of $\frac{\bar{c}}{4}$ of the wing above the ground,

H_H is the height of $\frac{\bar{c}}{4}$ of the horizontal tail above the ground.

b_{eff} is the effective wing span defined as

$$b_{eff} = \frac{C_{L_{WB}} + \Delta C_{L_f}}{\frac{C_{L_{WB}}}{b'_W} + \frac{\Delta C_{L_f}}{b'_f}} \tag{2.3}$$

where

$C_{L_{WB}}$ is the wing-body lift coefficient, flaps retracted, out of ground effect,

ΔC_{L_f} is the change in lift coefficient due to flaps, out of ground effect.

$$b'_W = \left(\frac{b'_W}{b}\right) b \tag{2.4}$$

$$b'_W = \left(\frac{b'_f}{b'_W}\right)\left(\frac{b'_W}{b}\right) b \tag{2.5}$$

The ratio $\frac{b'_W}{b}$ is given in fig. 2.11 on the next page as a function of taper ratio and aspect ratio, and $\frac{b'_f}{b'_W}$ is given in fig. 2.12 on the following page as a function of the ratio of flap span to wing span.

The horizontal-tail lift curve in ground effect is constructed by shifting the free-air lift curve at every C_L by the corresponding $-(\Delta\varepsilon)_G$, i.e.,

$$(\Delta\alpha_H)_G = -(\Delta\varepsilon)_G \tag{2.6}$$

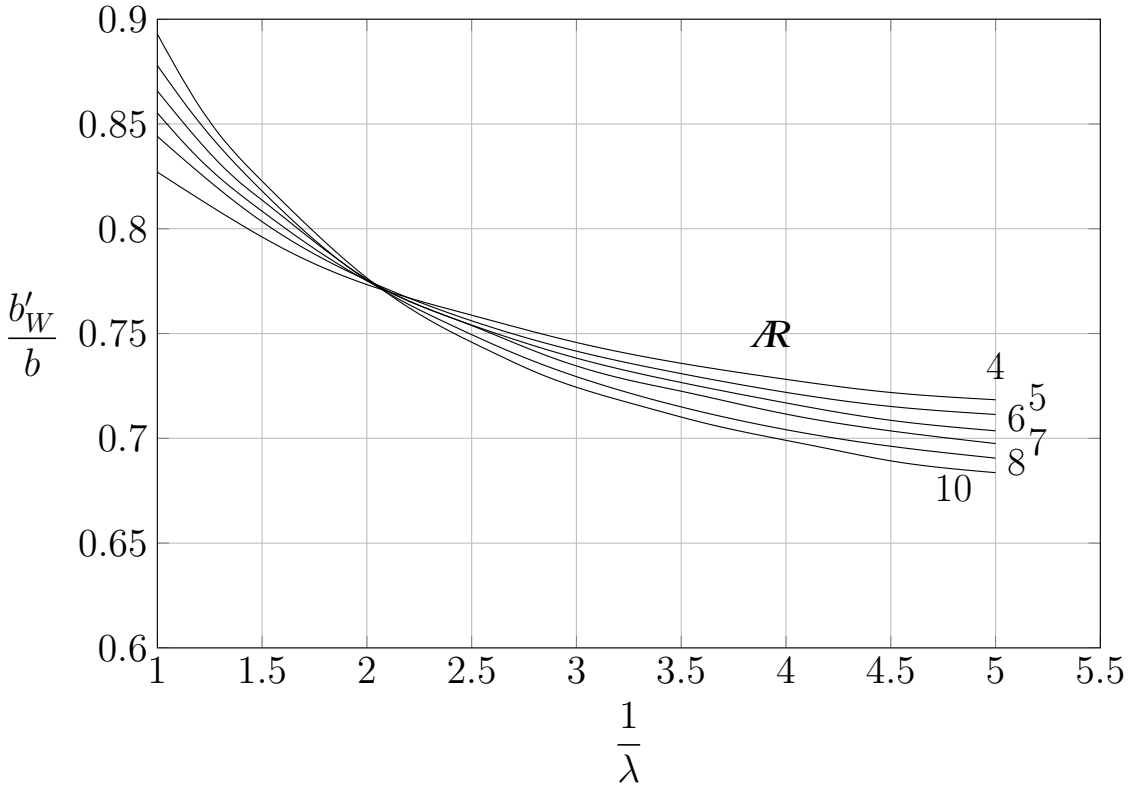

Figure 2.11: Effective wing span in the presence of the ground

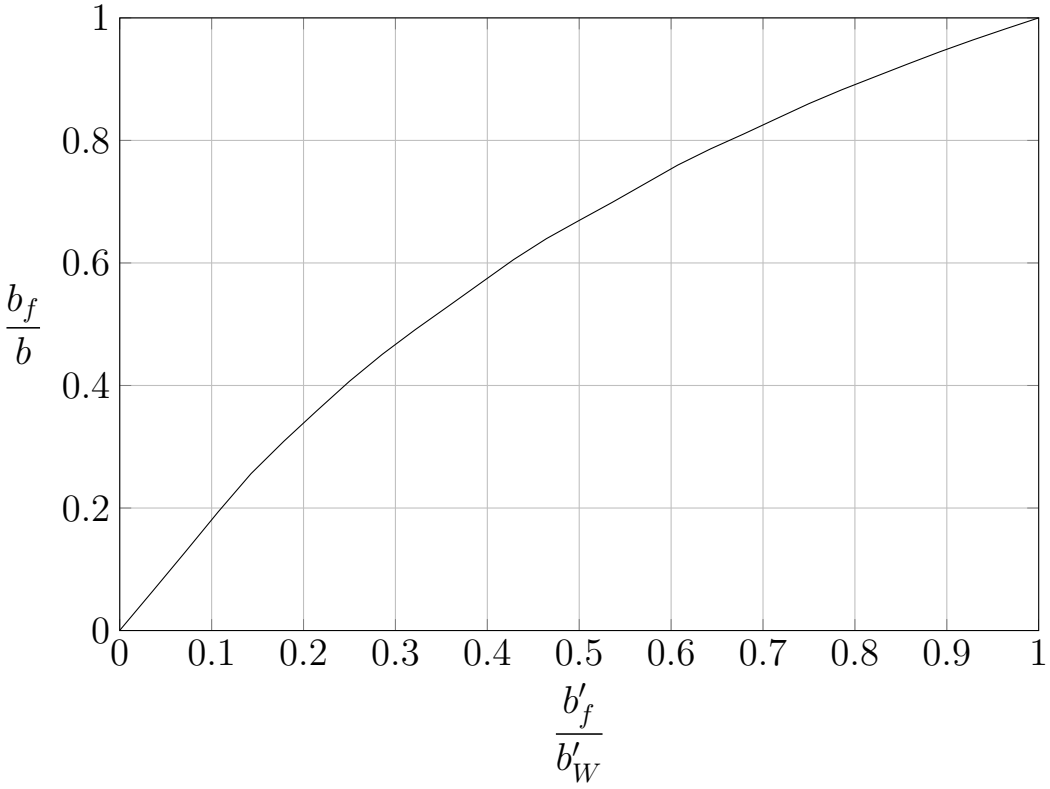

Figure 2.12: Effective flap span in the presence of the ground

Method II

This method estimates the ground effects on wing-body lift in the linear-lift range for all configurations not included in Method I. The change in wing-body angle of attack due to ground effects with respect to the out-of-ground-effect lift curve is given by:

$$(\Delta\alpha)_G = -18.4\frac{\left(C_{L_f}\right)_{WB}\sigma}{\mathcal{AR}} + rT\frac{\left(C_{L_f}\right)^2_{WB}}{57.3\,(C_{L_\alpha})_{WB}} - rB + K\left(\frac{t}{c}\right)_{max} \text{ (per deg)} \quad (2.7)$$

where

σ	is Prandtl's interference coefficient from multiplane theory and is obtained from fig. 2.13 on the following page as a function of wing height above the ground,
r	accounts for the effect of finite span and is obtained from fig. 2.10 on page 20 as a function of wing height above the ground,
T	accounts for the reduction of the longitudinal velocity and is obtained from fig. 2.14 on the following page as a function of wing height above the ground,
B	accounts for the change in circulation and is obtained from fig. 2.15 on page 25 as a function of wing height above the ground,
K	accounts for the effective wing thickness and is obtained from fig. 2.16 on page 25 as a function of wing height above the ground,
$(C_{L_\alpha})_{WB}$	is the wing-body lift-curve slope, per degree, out of ground effect,
$\left(\frac{t}{c}\right)_{max}$	is the ratio of maximum wing thickness to wing chord,
$\left(C_{L_f}\right)_{WB}$	is the wing-body lift coefficient including flap effects, out of ground effect.

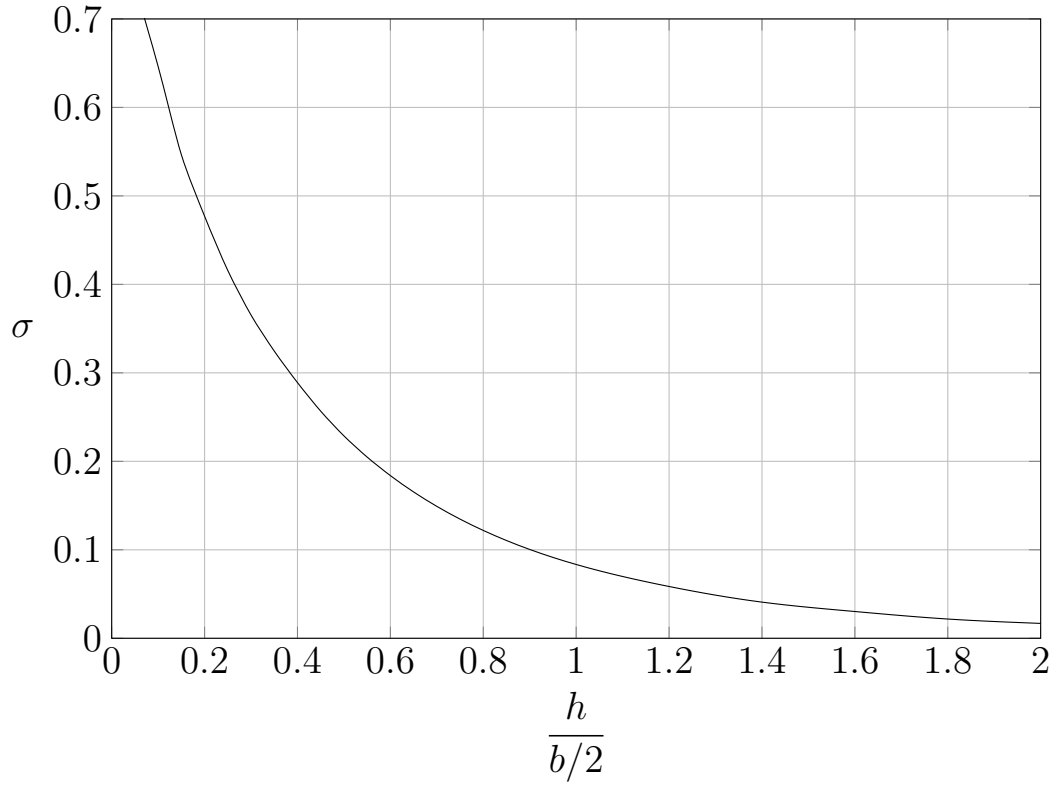

Figure 2.13: Prandtl's interference coefficient - indicative of variation in induced vertical velocity with ground effect

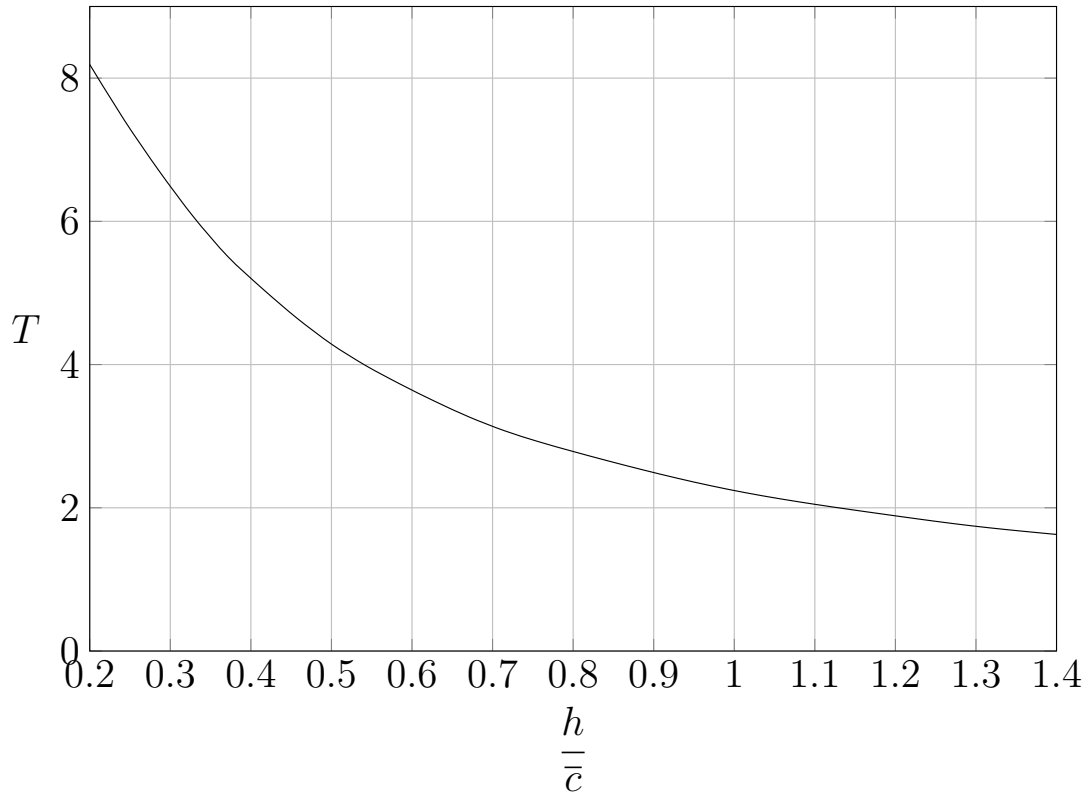

Figure 2.14: Parameter accounting for variation in longitudinal velocity with ground height

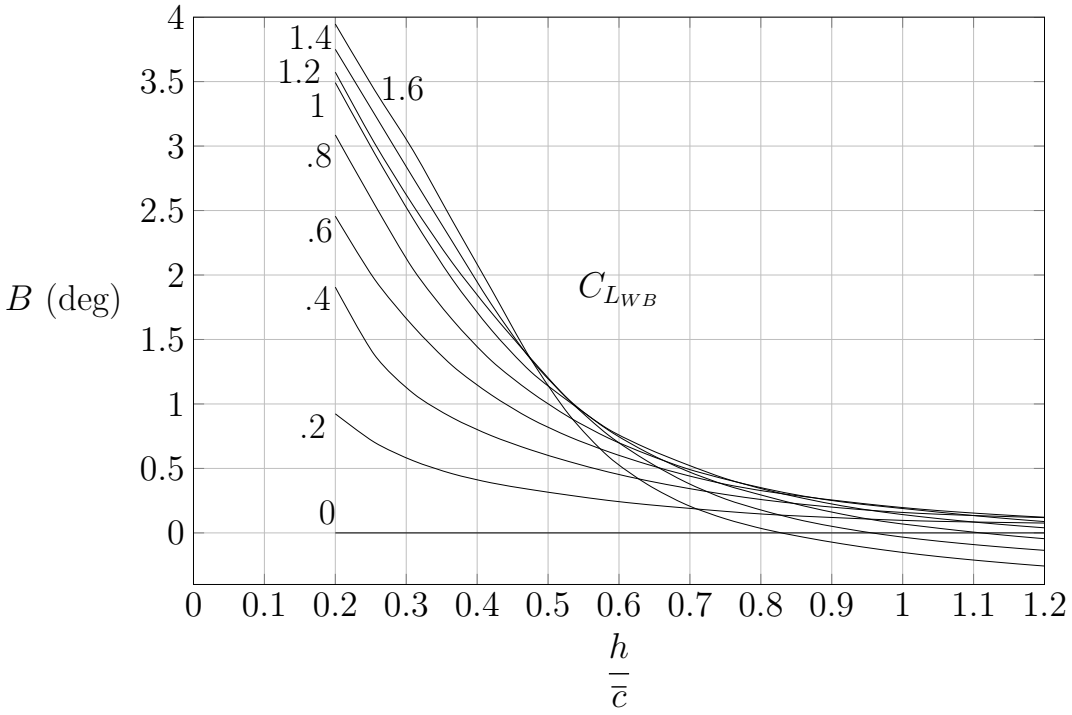

Figure 2.15: Parameter accounting for variation in circulation with lift and height above ground

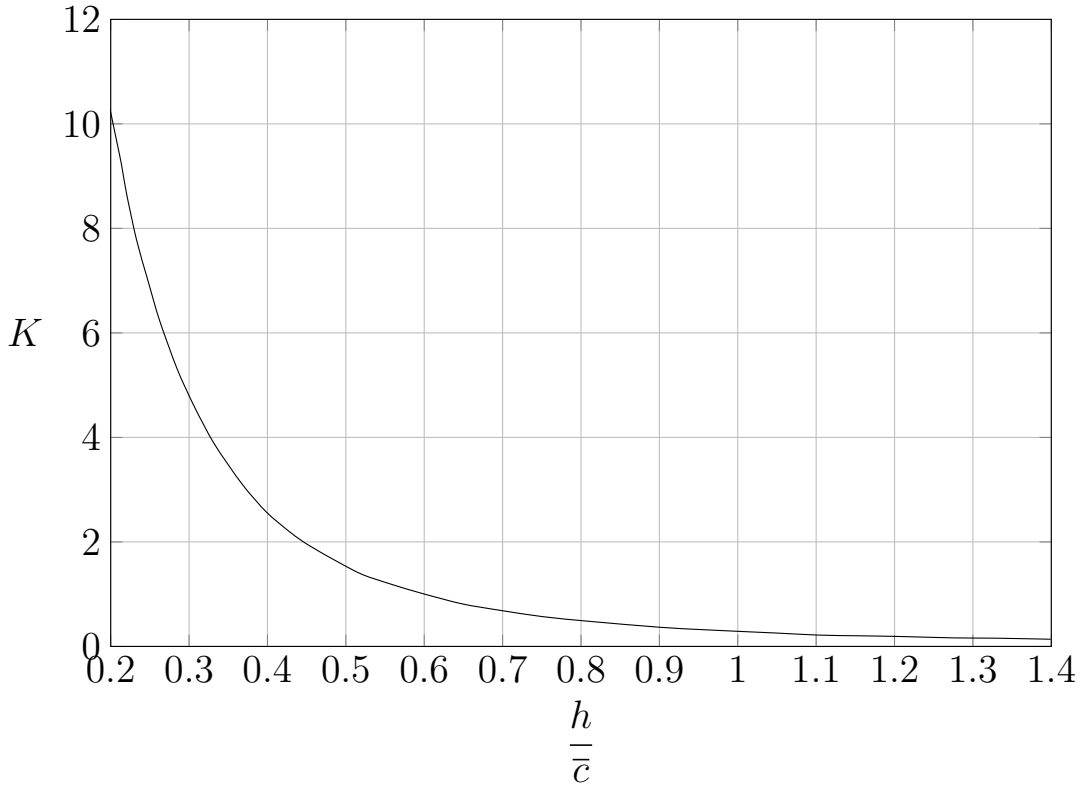

Figure 2.16: Parameter accounting for influence of wing thickness due to height above ground

2.4 Assessment of minimum unstick speed

The aircraft will be in the takeoff configuration at the maximum angle of attack allowed by the geometry, as shown in fig. 2.17

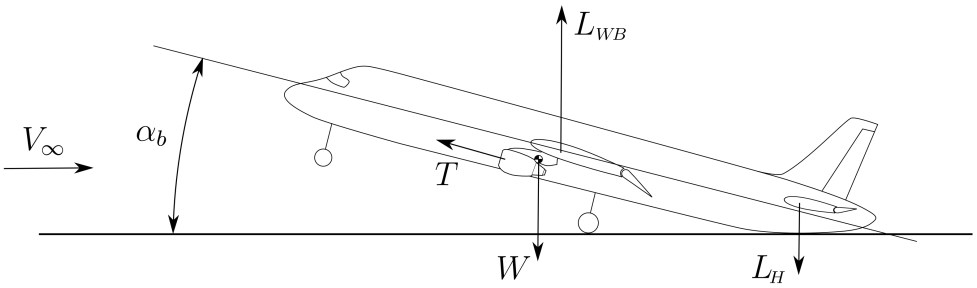

Figure 2.17: Takeoff configuration for the assessment of VMU

Known the geometry of the fuselage can be calculated α_W and α_H:

$$\alpha_W = \alpha_b + i_W$$
$$\alpha_H = \alpha_b - \varepsilon + i_H$$

The V_{MU} is the speed at which the aircraft takes off in this configuration, i.e. the speed that makes the algebraic sum of lift (wing-body lift and horizontal tail lift) and thrust upwards component equals to the weight of the aircraft.
In formulas:

$$L_{TOT} = L_{WB} + L_H + T\sin\alpha_b = W \tag{2.8}$$

$$L_{TOT} = C_{L_{WB}}\frac{1}{2}\rho V_\infty^2 S_W + C_{L_H}\frac{1}{2}\rho V_\infty^2 \eta S_H + T\sin\alpha_b = W \tag{2.9}$$

from which it is obtained:

$$V_{MU} = \sqrt{\frac{W - T\sin\alpha_b}{C_{L_{WB}}\frac{1}{2}\rho S_W + CL_H\frac{1}{2}\rho\eta S_H}} \tag{2.10}$$

where $C_{L_{WB}}$ and C_{L_H} are influenced by the ground effect. In particular there will be a new wing-body lift coefficient curve and new downwash angle as shown in section 2.3.

Chapter 3

Results and analysis

In this chapter is presented a case study performed on a regional turboprop of 130 pax, concerning aerodynamic characteristics, high lift devices and ground effect both on wing lift curve and on downwash angle.

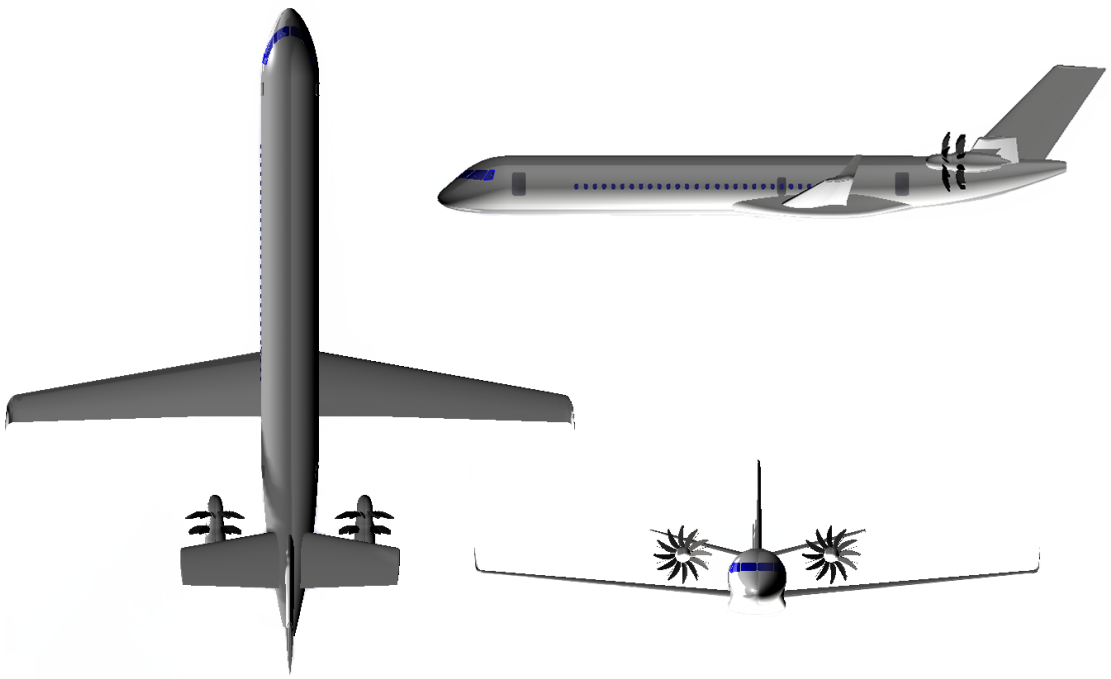

Figure 3.1: Layout of Regional Turboprop. Side, top and front section.

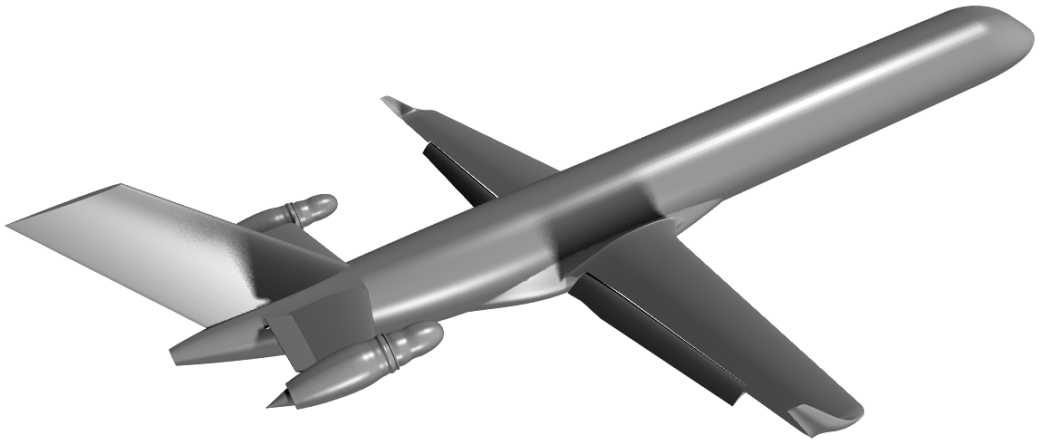

Figure 3.2: Regional Turboprop rendering in takeoff configuration

Data	**Value**	
Wing	Span	35.34 m
	AR	12
	C_r	5.325 m
	C_k	4.5 m
	C_t	3.4 m
	MAC	3.16 m
	Flap type	FOWLER
	$\delta_{f_{TO}}$	25°
Horizontal Tail	Span	12.5 m
	C_r	3.64 m
	C_t	2.272 m
	$\delta_{e_{min}}$	−25°
Other data	MTOW	51847 kg

Table 3.1: Aircraft main data.

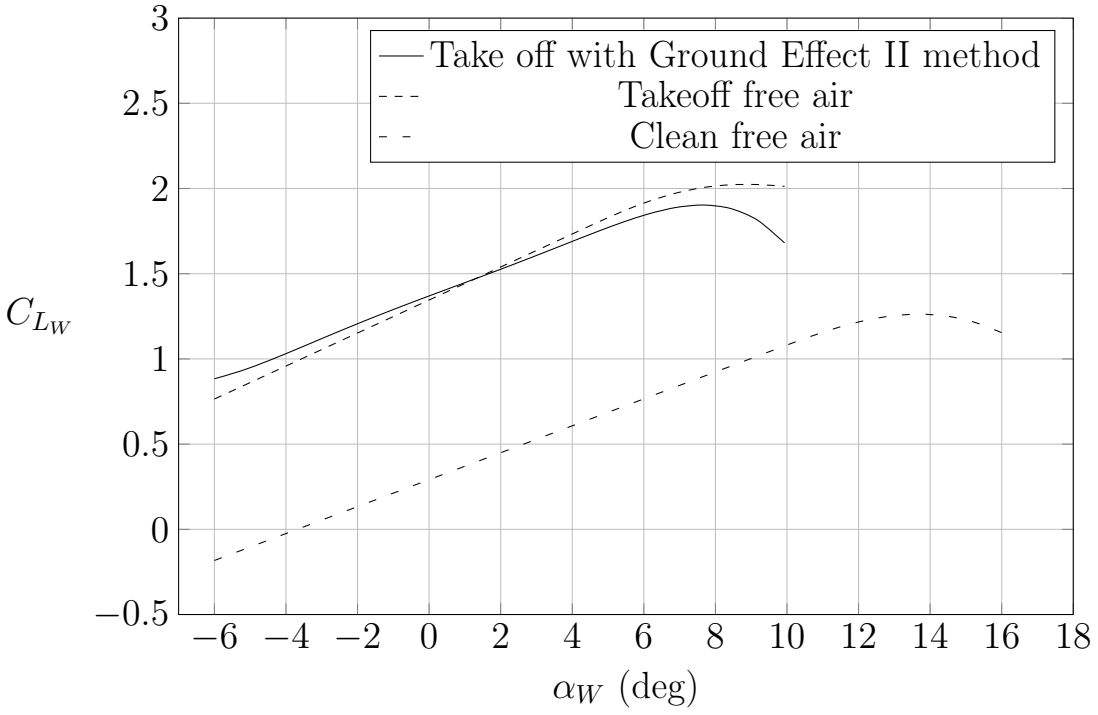

Figure 3.3: Ground effect on wing lift curve

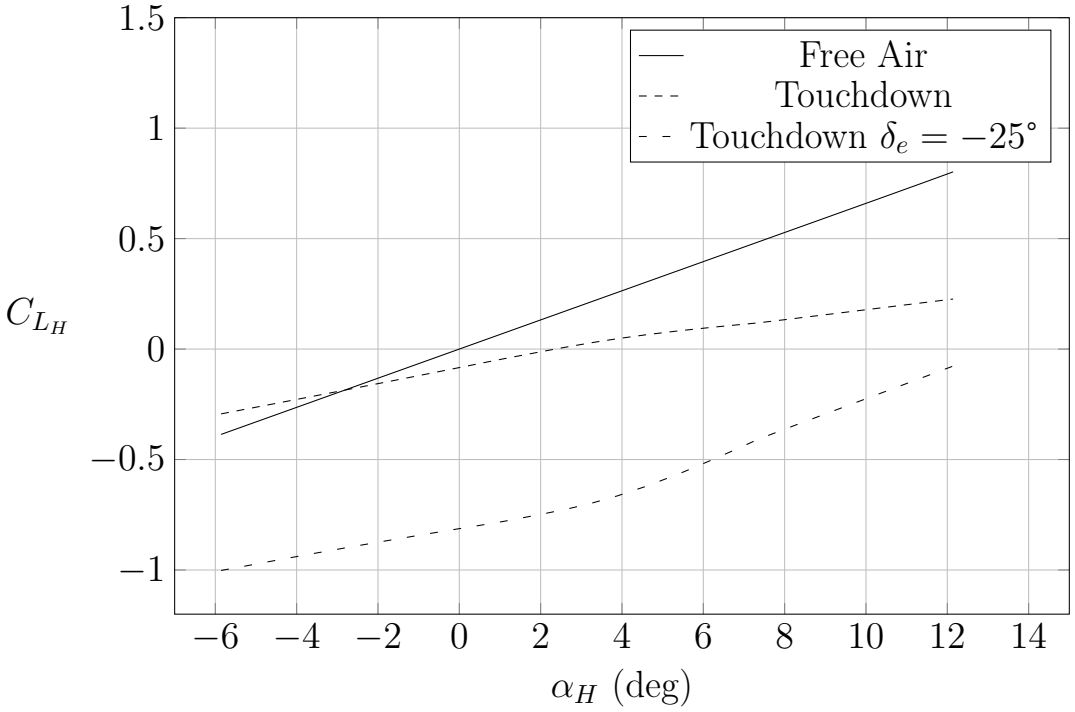

Figure 3.4: Ground effect on horizontal tail lift curve

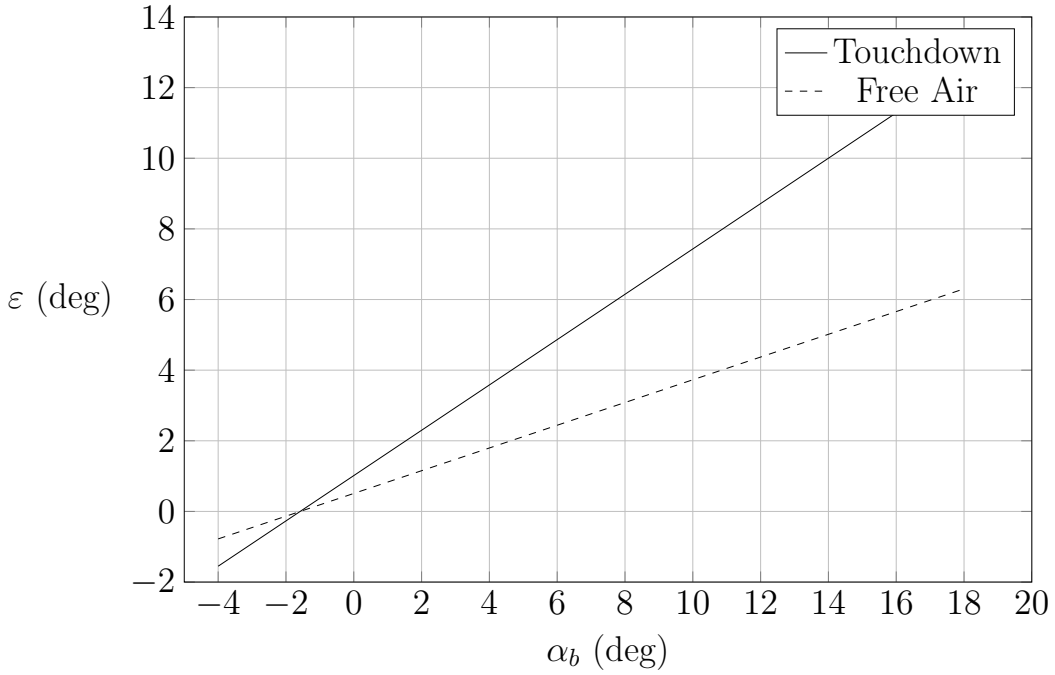

Figure 3.5: Ground effect on downwash angle

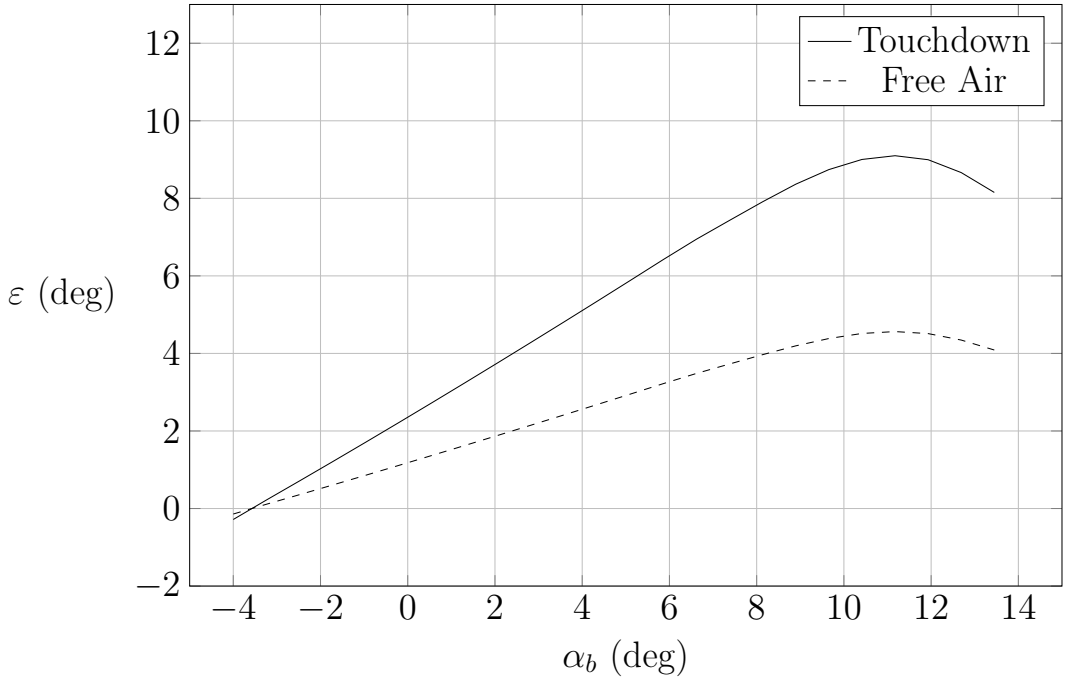

Figure 3.6: Ground effect on downwash angle considering non linear effects

Appendix A

HDF dataset and database reader creation

In a tool for preliminary design phase of an aircraft, it's very important to have available 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 are 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 a MATLAB tool, such as *Grabit*. Grabit 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 and quickly digitize values off the plot just by clicking the mouse on each data point [3].

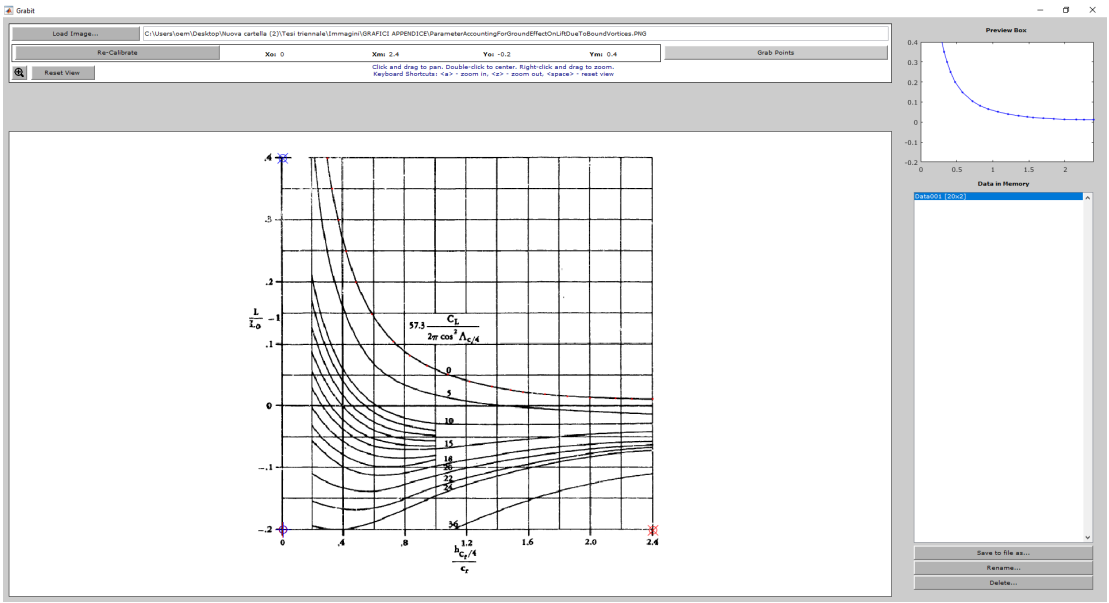

Figure A.1: Chart digitization using Grabit.

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

curve simply click on it. The values obtained can then be saved to. mat file (Matlab data).

A.2 Creation of an HDF file with Matlab

Obtained the .mat file from digitization is necessary to create the HDF file. 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

```
clear all, close all, clc
%% load data file generated by Grabit
% https://it.mathworks.com/matlabcentral/fileexchange/7173-grabit

fileBaseNames = {'L_L0_minus_1_vs_h_cr_4_cr_0' , 'L_L0_minus_1_vs_h_cr_4_cr_5' ,...
                 'L_L0_minus_1_vs_h_cr_4_cr_10' ,  'L_L0_minus_1_vs_h_cr_4_cr_15' ,...
                 'L_L0_minus_1_vs_h_cr_4_cr_18' , 'L_L0_minus_1_vs_h_cr_4_cr_20' ,...
                 'L_L0_minus_1_vs_h_cr_4_cr_22' , 'L_L0_minus_1_vs_h_cr_4_cr_24' ,...
                 'L_L0_minus_1_vs_h_cr_4_cr_36'};
nPoints = 49;

xx = linspace(0.2, 2.4, nPoints);

for kFile = 1:length(fileBaseNames)

%     fileName = sprintf('%s.mat',fileBaseNames{kFile});
    s = load(fileBaseNames{kFile}, '-mat');

    % Allocate imported array to column variable names

    x{kFile} = s.(fileBaseNames{kFile})(:, 1);
    x{kFile}(end) = 2.4;

    y{kFile} = s.(fileBaseNames{kFile})(:, 2);
    %% Smoothing
    pp(kFile) = csaps(x{kFile}, y{kFile}, 0.999999);
    c{kFile} = ppval(pp(kFile), xx');
    data(:,kFile) = c{kFile};

    plot (xx', data(:,kFile), '-*');
    hold on
end
xlabel('$\frac{h_{{c_r}/4}}{c_r}$','interpreter','latex');
ylabel('$\frac{L}{L_0}-1$','interpreter','latex');
set(get(gca,'ylabel'),'rotation',0)

title('Parameter␣accounting␣for␣ground␣effect␣on␣lift␣due␣to␣trailing␣vortices');
axis([0 2.4 -0.5 .6]);
legend('0','5','10', '15','20','22' ,'24', '36');


%% preparing output to HDF

CL_2_2pi_cos_2_Gamma_c4 = [0 5 10 15 20 22 24 36]';
h_cr_4_cr = xx';


%columns --> curves

hdfFileName = 'Delta_alpha_CL_Ground_Effect_L_L0_minus1_vs_h_cr_4_cr';

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
```

```
h5create(hdfFileName,...
 '/(Delta_alpha_CL_Ground_Effect)_L_L0_minus1_vs_h_cr_4_cr/data', size(data'));
h5write(hdfFileName,...
 '/(Delta_alpha_CL_Ground_Effect)_L_L0_minus1_vs_h_cr_4_cr/data', data');

h5create(hdfFileName,...
 '/(Delta_alpha_CL_Ground_Effect)_L_L0_minus1_vs_h_cr_4_cr/var_0',...
 size(CL_2_2pi_cos_2_Gamma_c4'));
h5write(hdfFileName,...
 '/(Delta_alpha_CL_Ground_Effect)_L_L0_minus1_vs_h_cr_4_cr/var_0',...
 CL_2_2pi_cos_2_Gamma_c4');

h5create(hdfFileName,...
 '/(Delta_alpha_CL_Ground_Effect)_L_L0_minus1_vs_h_cr_4_cr/var_1', size(h_cr_4_cr'));
h5write(hdfFileName,...
 '/(Delta_alpha_CL_Ground_Effect)_L_L0_minus1_vs_h_cr_4_cr/var_1', h_cr_4_cr');
```

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

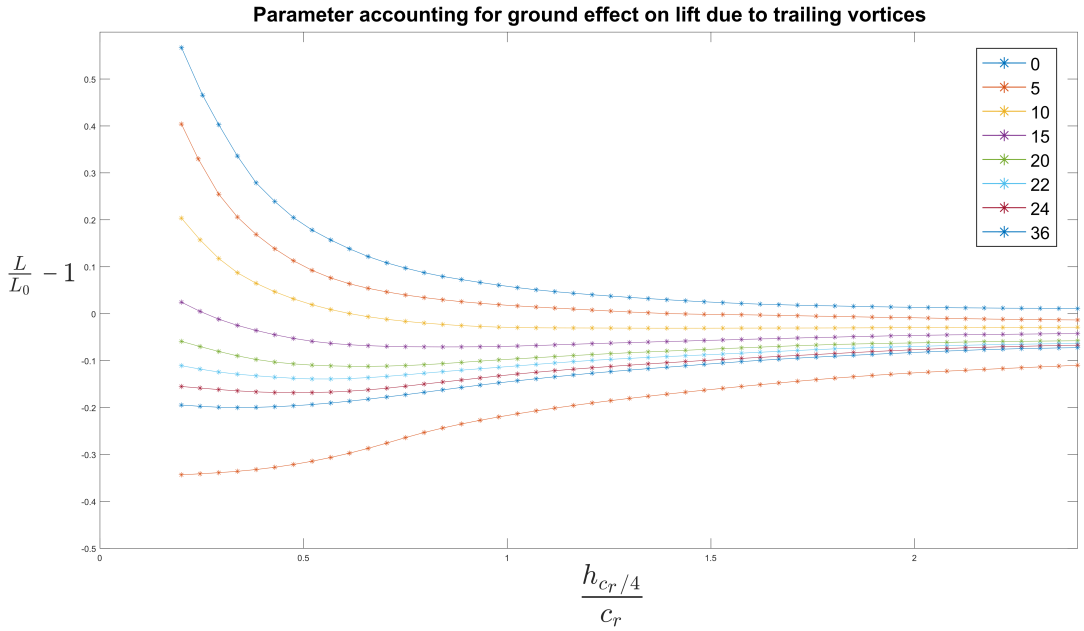

Figure A.2: Chart digitization plot.

Having created the .h5 file it is necessary to import it in database using *HDF View* [5] and to set up the database reader implementing in specific class the variable declaration of an interpolating function starting from the .h5 file. In conclusion the getter method has to be defined.

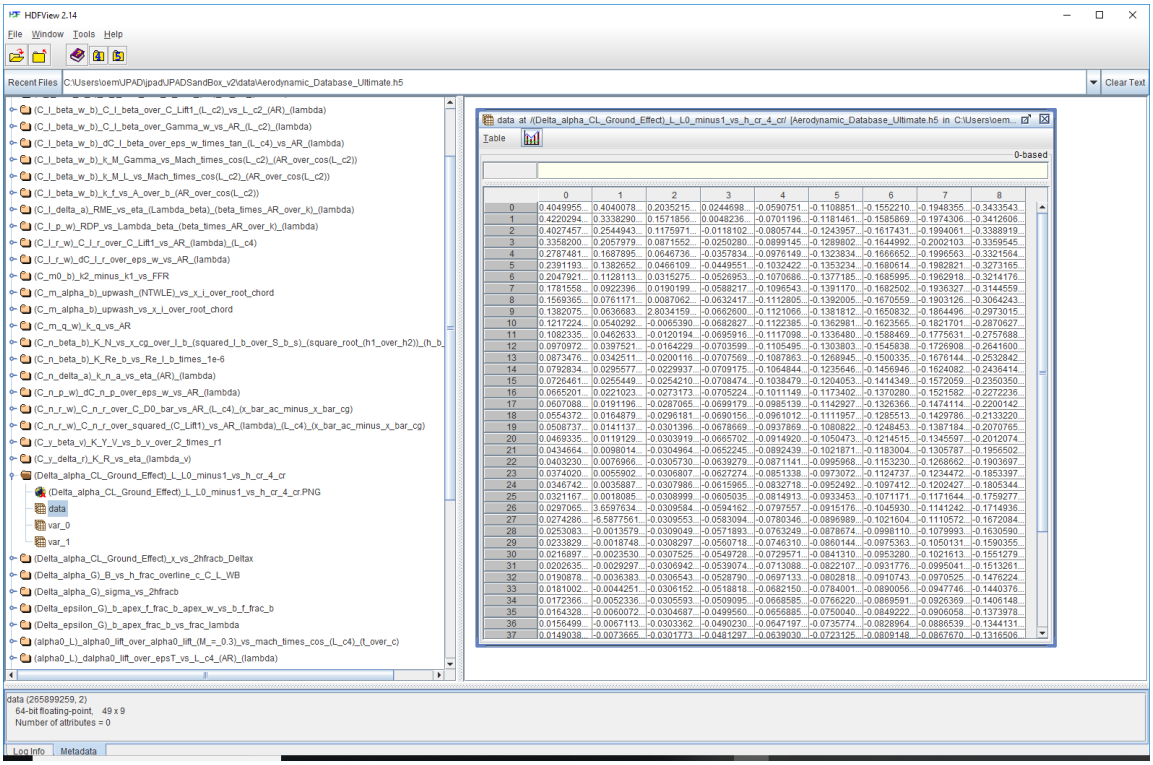

Figure A.3: Chart digitization using Grabit.

Listing A.2 Java extract from database reader class

```
public class AerodynamicDatabaseReader{

        private MyInterpolatingFunction
        Delta_alpha_CL_Ground_Effect_L_L0_minus1_vs_h_cr_4_cr;

        public AerodynamicDatabaseReader
        (String databaseFolderPath, String databaseFileName) {
                Delta_alpha_CL_Ground_Effect_L_L0_minus1_vs_h_cr_4_cr
                        = database.interpolate2DFromDatasetFunction
                        ("Delta_alpha_CL_Ground_Effect_L_L0_minus1_vs_h_cr_4_cr");
        }

        public double getDeltaAlphaCLGroundEffectLL0minus1vshcr4cr(
                double cLParameter,     // var0
                double hFracC           //var1
                      ) {
                        return
                        Delta_alpha_CL_Ground_Effect_L_L0_minus1_vs_h_cr_4_cr.valueBilinear(
                                hFracC,         // var1
                                cLParameter     // var0
                                );
                }
}
```

Bibliography

[1] Airbus. *Flight Operations Briefing Notes*. URL: https://www.skybrary.aero/bookshelf/books/493.pdf.

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

[3] Various authors. *Grabit description*. URL: https://www.mathworks.com/matlabcentral/fileexchange/7173-grabit.

[4] James Gosling et al. *The Java Language Specification*. 2015.

[5] Hdf Groups. *HDF View Description*. URL: https://support.hdfgroup.org/products/java/hdfview/.

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

[7] R. Fink. *USAF Stability and control DATCOM*. McDonnell Douglas Corporation, 1977.