Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria Aerospaziale
A.a. 2021/2022
Sessione di Laurea Luglio 2022

Validation of a mathematical model for the Bs Prime

Relatori:
pof.ssa Manuela Battipede

Candidati:
Andrea Corino 269573

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

Symbol	Description	Units
$c.g.$	center of gravity	
CAS	calibrated airspeed	$\mathrm{m\,s^{-1}}$
EAS	equivalent airspeed	$\mathrm{m\,s^{-1}}$
IAS	indicated airspeed	$\mathrm{m\,s^{-1}}$
mac	mean aerodynamic chord	m
MAP	manifold pressure	"Hg
MTO	take-off weight	kg
$MTOW$	maximum take-off weight	kg
ROC	rate of climb	$\mathrm{m\,s^{-1}}$
TAS	true airspeed	$\mathrm{m\,s^{-1}}$
TPR	transient peak ratio	–

Greek Symbols

Symbol	Description	Units
$\Delta\Pi_{excess}$	excess power	W
α	angle of attack	rad
β	sideslip angle	rad
χ	azimuth angle	rad
δ_a	deflection of ailerons	rad
δ_e	deflection of elevator	rad
δ_f	deflection of flaps	rad
δ_r	deflection of rudder	rad
Γ	angle between X_{body}-axis and the ground	rad

Γ	dihedral angle	deg
γ	flight-path angle	rad
λ_a	longitudinal plane eigenvector	–
λ_b	lateral direction plane eigenvector	–
λ_i	eigenvalue	–
μ	dynamic viscosity	$\mathrm{kg\,m^{-1}\,s^{-1}}$
ω_d	damped frequency	$\mathrm{rad\,s^{-1}}$
ω_n	natural frequency	$\mathrm{rad\,s^{-1}}$
Φ	bank angle	rad
ϕ	roll angle	rad
Π	engine power	W
Π^*	flight manual engine power	W
Ψ	it takes into account the change in altitude	–
ψ	yaw angle	rad
ρ	density	$\mathrm{kg\,m^{-3}}$
θ	pitch angle	rad
ξ	throttle percentage	%
ζ	damping	–

Roman Symbols

Symbol	**Description**	**Units**
$\bar{c}$	mean aerodynamic chord	m
C_L	lift coefficient	–
E	potential energy	J
J	cost function	–
K	kinetic energy	J
m	mass	kg
$N_{1/2}$	halving cycles	–
T	period	s
t	time	s
$t_{1/2}$	halving time	–

V_x	best angle-of-climb speed	$\mathrm{m\,s^{-1}}$
V_y	best rate-of-climb speed	$\mathrm{m\,s^{-1}}$
W	weight	N
$(XYZ)_{datum}$	cartesian axes centered in the datum	
$(XYZ)_{body}$	cartesian axes centered in the aricraft center of gravity	
A	state matrix of the linearized system	
a	acceleration	$\mathrm{m\,s^{-2}}$
A_a	longitudinal plan state matrix of the linearized system	
A_b	lateral-directional plan state matrix of the linearized system	
AR	aspect ratio	–
b	wing span	m
C_{D_i}	induced drag coefficient	–
C_L	lift coefficient	–
D	drag	N
D_i	induced drag	N
e	Oswald efficiency	–
H	altitude	m
$I_{xx_{tot}}$	aircraft moment of inertia along X_{body}-axis	$\mathrm{kg\,m^2}$
$I_{yy_{tot}}$	aircraft moment of inertia along Y_{body}-axis	$\mathrm{kg\,m^2}$
$I_{zz_{tot}}$	aircraft moment of inertia along Z_{body}-axis	$\mathrm{kg\,m^2}$
$J_{xy_{tot}}$	aircraft product of inertia in X_{body} Y_{body}-plane	$\mathrm{kg\,m^2}$
$J_{xz_{tot}}$	aircraft product of inertia in X_{body} Z_{body}-plane	$\mathrm{kg\,m^2}$
$J_{yz_{tot}}$	aircraft product of inertia in Y_{body} Z_{body}-plane	$\mathrm{kg\,m^2}$
n	engine speed	RPM
p, q, r	respectively angular rate of: roll, pitch, yaw	$\mathrm{rad\,s^{-1}}$
pz	manifold pressure	”Hg
q	dynamic pressure	Pa
S	wing surface area	$\mathrm{m^2}$
T	temperature	K

u, v, w	speed components respectively along: X_{body}, Y_{body} and Z_{body} axes	m
V	true airspeed	$\mathrm{m\,s^{-1}}$
X	state vector	
x_e, y_e	respectively: X-coordinate and Y-coordinate in Earth-fixed reference frame	m
$x_{c.g\%}$	position of the center of gravity along the X_{body}-axis as a percentage of the mean aerodynamic chord	%
$x_{c.g}, y_{c.g}, z_{c.g}$	aircraft center of gravity position with respect to the datum, respectively along the X_{datum}, Y_{datum} and Z_{datum} axes	m

Abstract

This master's thesis work was carried out abroad at the EURO FLIGHT TEST company, based in Winningen, Germany. The main purpose is to create a mathematical model in MATLAB / Simulink that allows to simulate the behavior of the BS PRIME aircraft. The company, which has recently chartered a BS PRIME aircraft, has provided its willingness to support the project, both during the development phase and during the validation phase. It was in this second phase that the possibility of performing flight tests with a specialized pilot was of fundamental importance. This thesis focuses on model validation, comparing the manual and flight test data. It was necessary to separate the modeling discussion from the validation one for bureaucratic reasons. It is noted that in order to have a global understanding it is necessary to consider the two parts as a single work. **The work and all the necessary activities, net of the contribution provided by EURO FLIGHT TEST, were carried out by both candidates: Andrea Corino and Diego Orio**.

Chapter 1

Validation

This second thesis is a continuation of the first thesis "Development of a mathematical model for the Bs Prime", which focused on modeling, while this one focuses on model validation. According to the Department of Defense (DOD), modeling means [4]:

A model is defined as "a physical, mathematical, or otherwise logical representation of a system, entity, phenomenon, or process." Modeling is the "application of a standard, rigorous, structured methodology to create and validate" this model

While validation is defined as:

"The process of determining the degree to which a model or simulation is an accurate representation of the real world from the perspective of the intended uses of the model or simulation"

In general, in addition to validation, importance is also given to verification. Again according to the DOD, the latter is defined as:

"The process of determining that a model or simulation implementation accurately represents the developer's conceptual description and specification. Verification also evaluates the extent to which the model or simulation has been developed using sound and established software engineering techniques"

So to better describe the difference between validation and verification, consider the following example: a model that closely reflects reality, but has many bugs in the simulation phase, satisfies validation but not verification; on the other hand, if a model being simulated is free-bug, but does not faithfully represent reality, it satisfies the verification but not the validation, [2]. In this discussion we will not refer to the concept of verification, as the project linked to the thesis did not have specific technical requirements. For future developments there is the possibility of inserting this model in a simulation environment, in which case verification will be necessary. It should be noted that for time and economic reasons, the concept of validation is not to be intended as a rigorous process that is defined on the basis of particular techniques, but more as a first comparison. So this means that we will still talk about validation as we refer to the definition, so we analyze how close the model is to reality, but it is not a complete process. In general, the validation process is iterative: once the first version of the mathematical model is completed, simulations are carried out, the data obtained are compared with those of the real aircraft, how much the model deviates from the latter is analyzed, and the mathematical model is changed

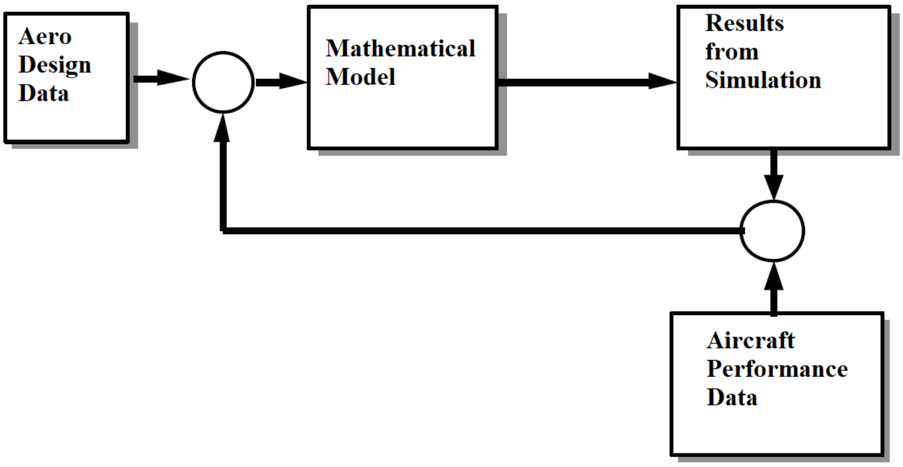


Figure 1.1: Block scheme for validation

again, and so on [5, see]. In the following discussion this cycle was covered once, i.e. defined the first mathematical model, a first validation phase was carried out which led to the definition of a second mathematical model, more close to the data of the real aircraft. In this discussion, a first validation of the model was carried out using the performance data present in the flight manual, then a subsequent validation through flight tests, exploiting the climb performance of the aircraft.

Chapter 2

FDC: functions for trim, stability and ROC

2.1 ACTRIM

The ACTRIM routine allows you to calculate trim conditions. First you will be asked to load the necessary data from the file, then you can choose the type of stationary flight to be analyzed (in this discussion, option 1 "Steady wings-level flight" has always been used). Finally, the flight conditions will be requested: H, V, heading, deltaf, MAP or Γ. The user has the possibility to choose whether to keep a certain manifold pressure value fixed or whether to impose the value of the flight path angle Γ. In the second case an initial estimate of the MAP value will be requested, but the real MAP value at sea level will be output anyway. ACTRIM actually converts the calculation of the trim conditions into a maximum and minimum problem, in which the function of cost to minimize is: $J = 10(\dot{u}^2 + \dot{v}^2 + \dot{w}^2) + 100(\dot{p}^2 + \dot{q}^2 + \dot{r}^2)$. For this process it is possible to modify appropriately the tolerance, and the maximum number of iterations and function evaluations. The cost function has high relevance because, in order to correctly calculate the trim conditions, it is necessary to ensure that the minimum found is global and not local. Once the process converges, and the conditions are correctly calculated, it is possible to save them to a file with the ".TRI" extension. During this project, additional routines were used which, starting from the ACTRIM code lines, made it possible to automate the calculation of trim conditions for different flight conditions, or for different initial MAP values. These routines will be better explained in the section about calculating trim conditions.

2.1.1 Cost function

As mentioned, the cost function for minimization is

$$J = 10(\dot{u}^2 + \dot{v}^2 + \dot{w}^2) + 100(\dot{p}^2 + \dot{q}^2 + \dot{r}^2) \tag{2.1}$$

It is different from that used for the Beaver model. To use this specific function it was necessary to modify ACTRIM by inserting the lines of code to calculate the new terms used. The speed components in body axes are:

$$\begin{aligned} u &= V\cos\alpha\cos\beta \\ v &= V\sin\beta \\ w &= V\sin\alpha\cos\beta \end{aligned} \tag{2.2}$$

The respective derivatives, introduced in ACTRIM for the evaluation of J are:

$$\begin{aligned}\dot{u} &= \dot{V}\cos\beta\cos\alpha + V(-\sin\beta\cos\alpha\cdot\dot{\beta} - \cos\beta\sin\alpha\cdot\dot{\alpha})\\ \dot{v} &= \dot{V}\sin\beta + V\cos\beta\cdot\dot{\beta}\\ \dot{w} &= \dot{V}\cos\beta\sin\alpha + V(-\sin\beta\sin\alpha\cdot\dot{\beta} - \cos\beta\cos\alpha\cdot\dot{\alpha})\end{aligned} \tag{2.3}$$

The tollernace used are: tollerance 10^{-30}, MaxFunEvals $5\cdot10^5$,'MaxIter' $5\cdot10^5$

2.2 Actrimforcond

This new function introduced in FDC is based on the ACTRIM function, already present. It uses the same methodologies and the same cost function. Actrimforcond allows to calculate the trim conditions automatically. It was designed to quickly compare the trim conditions of the model with those reported in the flight manual, in particular comparisons on engine power. Speeds, altitudes, flight path angle values, and an initial value of MAP are not requested from the user, but they must be entered within the code of the function. First it will be asked to load the aircraft model from the file, then it must be chosen the type of stationary flight to be analyzed (in this discussion, option 1 "Steady wings-level flight" has always been used). After these requests the process becomes automatic. The power values calculated by the model, the MAP values, and the difference between the model power and the actual power are provided as an output. During this process, it is necessary to ensure that every minimum found is global and not local, as the importance of finding the correct minimum is crucial.

2.3 Actrim_forV_forH

This new function introduced in FDC is based on the ACTRIM function, already present. So it uses the same methodologies and the same cost function of Actrim. Actrim_forV_forH allows you to calculate the trim conditions without interruption by giving as output vectors that contain the desired quantities as the altitude and speed vary. The user is only asked to choose the model and type of stationary flight to be analyzed (in this discussion, option 1 "Steady wings-level flight" has always been used). The speed and altitude values must be inserted inside the code of the function itself. The output quantities (α, δ_e, δ_a, δ_r *and throttle*) are recalled by a separate script with which the graphs can be generated. This function is very important as it can be seen from the graphs whether trim conditions corresponding to local or global minimums have been defined.

2.4 CtrimforROC

This new function introduced in FDC is based on the ACTRIM function, already present. So it uses the same methodologies and the same cost function. CtrimforROC allows you to calculate the rate of climb in trim conditions without interruption. The speeds are included in the code of the function. The user is required to load the aircraft model from the file, then it must be chosen the type of stationary flight to be analyzed (in this discussion, option 1 "Steady wings-level flight" has always been used). The output is a vector containing the ROC.

2.5 ACLIN

The ACLIN routine allows you to linearize the system of equations around a certain flight condition. As input, the trim conditions around which the linearization will be carried out will therefore be required, along a ".DAT" file generated by MODBUILD. Once the linearization has been carried out, the same routine will give the user the option to generate a reduced matrix of the system, selecting the variables to be taken into account. This option is what is needed to separate the longitudinal and lateral-directional dynamics. Also in this case it will be possible to save the results on a file, with the extension “.LIN”.

2.6 AclinforCG

In order to evaluate the dynamic analysis as the position of the center of gravity varies, and similarly to what was done with ACTRIM, also in this case a similar routine have been developed to automate the process: AclinforCG. It works exactly like ACLIN, only that it allows to have as input an aerodynamic model referred to the datum. In fact, within it the aerodynamic derivatives are recalculated as the position of the center of gravity varies, thus being able to linearize the system in an appropriate way. Then the eigenvalues of the linearized system are obtained as the position of the center of gravity varies. Within the same function, the positions of the center of gravity are defined. To obtain the eigenvalues referred to the longitudinal or lateral directional dynamics, the correct state vector must be selected within the code of the function.

Chapter 3

Trim

3.1 Introduction

Once the mathematical modeling part of the aircraft was finished, it was possible to begin a first validation phase, starting from the analysis of the trim conditions in straight horizontal flight and climb. A trim condition is a flight condition in which the controls remain fixed and the sum of forces and moments on the aircraft is zero. The research of the trim condition is a very important part of the validation of a mathematical model, because it allows to evaluate if what has been done before is correct.

3.2 Trim condition

The first step to validate the model is to check if it's actually possible to identify trim conditions for different values of speed and altitude. Using ACTRIM, several tests were made varying the flight conditions, and in particular speed, altitude and RPM. Below is an example. First, the load condition is defined. This is an important step because it influences the TOW, the position of the center of gravity, and inertias. This is all done in the "primo_run_fdc.m" routine, where it is possible to define the mass of pilot, passenger, fuel and luggage. For the purpose of this example, the following load condition have been used. The same load condition will be used as a sort of “standard” load configuration.

	Mass $[kg]$	Momentum $[kg\ m]$
pilot	85	-
passenger	85	-
baggage	0	-
fuel	40	-
empty weight	390	-
TOW	600	648.17

Table 3.1: Load diagram

Afterwards, the ACTRIM routine allows to define the flight conditions, which in this case are:

- Speed 60 m/s
- Altitude 1220 m

- Engine RPM $5000 RPM$

- Pz (first estimation) 21 $''Hg$

The routine outputs are the state vector X, the derivative of the state vector $\dot{X}$ and the control vector U.

$$X = \begin{bmatrix} V \\ \alpha \\ \beta \\ p \\ q \\ r \\ \psi \\ \theta \\ \phi \\ x_e \\ y_e \\ H \end{bmatrix} = \begin{bmatrix} 60 & [m/s] \\ 4.3898 \cdot 10^{-2} & [rad] \\ -5.7228 \cdot 10^{-5} & [rad] \\ 0 & [rad/s] \\ 0 & [rad/s] \\ 0 & [rad/s] \\ 0 & [rad] \\ 4.3898 \cdot 10^{-2} & [rad] \\ 0 & [rad] \\ 0 & [m] \\ 0 & [m] \\ 1220 & [m] \end{bmatrix} \quad ; \quad U = \begin{bmatrix} \delta_e \\ \delta_a \\ \delta_r \\ \delta_f \\ n \\ pz \\ uw \\ vw \\ ww \\ \dot{u}w \\ \dot{v}w \\ \dot{w}w \end{bmatrix} = \begin{bmatrix} -2.0889 \cdot 10^{-1} & [rad] \\ -5.2442 \cdot 10^{-3} & [rad] \\ -4.5772 \cdot 10^{-4} & [rad] \\ 0 & [rad] \\ 5000 & [RPM] \\ 23.41 & [''Hg] \\ 0 & [m/s] \\ 0 & [m/s] \\ 0 & [m/s] \\ 0 & [m/s^2] \\ 0 & [m/s^2] \\ 0 & [m/s^2] \end{bmatrix}$$

$$\dot{X} = \begin{bmatrix} \dot{V} \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{p} \\ \dot{q} \\ \dot{r} \\ \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \\ \dot{x_e} \\ \dot{y_e} \\ \dot{H} \end{bmatrix} = \begin{bmatrix} -3.3407 \cdot 10^{-15} & [m/s^2] \\ 2.7734 \cdot 10^{-17} & [rad/s] \\ -1.7386 \cdot 10^{-17} & [rad/s] \\ -4.8762 \cdot 10^{-16} & [rad/s^2] \\ -2.3598 \cdot 10^{-16} & [rad/s^2] \\ -2.4686 \cdot 10^{-16} & [rad/s^2] \\ 0 & [rad/s] \\ 0 & [rad/s] \\ 0 & [rad/s] \\ 60 & [m/s] \\ -3.4337 \cdot 10^{-3} & [m/s] \\ 0 & [m/s] \end{bmatrix}$$

$$\Pi = 38.04 \ kW \qquad \xi = 50 \ \%$$

As it can be seen, the results are actually trim conditions: the incidence has a reasonable value of about 2.5°, the elevator reaches about 12°, it is correctly deflected upwards and fully falls within the upper excursion limit of 28°. Similarly, the aileron deflection is very low: this is correct since in trim conditions the only purpose of ailerons is to compensate the torque generated by the propeller. The rudder is about two orders of magnitude smaller than the elevator, since it has to compensate for the very small yaw moment generated by the ailerons. In addition it can be noted that this condition can be defined as a trim condition since the quantities in the vector $\dot{X}$ are in the order of 10^{-18} e 10^{-15}, therefore negligible variations.

3.3 Trim validation

Of fundamental importance were the comparisons between the trim conditions calculated with ACTRIM by using the BS Prime DATCOM model and those reported in the flight manual. The comparisons were made both in uniform straight flight conditions and in climb conditions. It is specified that all the tests have been carried out considering a load condition corresponding to the MTOW, therefore in the most critical condition. For straight horizontal flight conditions, the flight manual [1, p. 100] provides, through appropriate graphs, the relationship between altitude and cruising speed for fixed values of RPM and throttle. Wherase in the figure 3.1 the power level is shown.

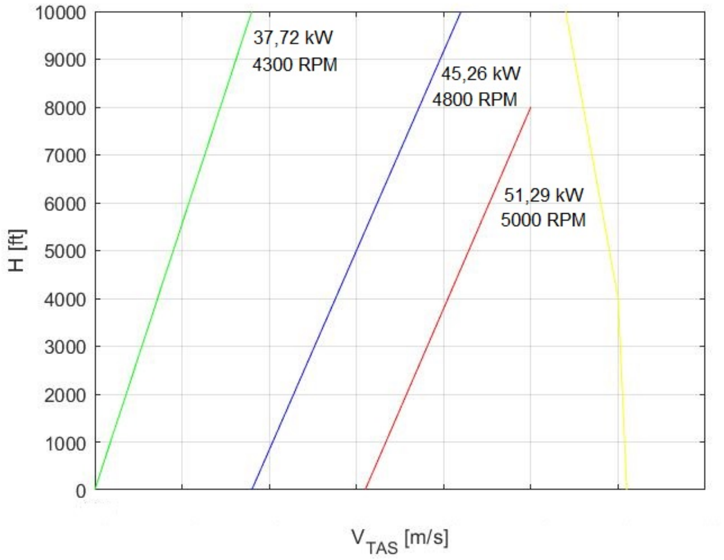


Figure 3.1: Cruise Speed at MTOW 600 Kg

Also from the manual of Bs Prime [1, pp. 96, 97], only one example with the associated data (Best Rate-of-Climb Speed (V_y), RPM and Power) is provided for the climb conditions. So it will be used as the only comparison example for the climb. Since the flight regime is subsonic, the effects of compressibility are neglected as previously done, thus assuming EAS $\simeq$ CAS. Then the true speed is calculated as

$$TAS = EAS\sqrt{\rho_{sl}/\rho} \tag{3.1}$$

As for the climb condition, the ROC is defined, converted to m/s, and used for the calculation of the flight-path-angle Γ as:

$$\Gamma = \arcsin(ROC/V) \tag{3.2}$$

Remember that all engine data refer to sea level conditions, therefore the following data also refer to this condition. For more details see thesis "Development of a mathematical model for the Bs Prime".

3.4 Comparison on requested power between the model 1 and the flight manual

As already said, the starting point is the Bs prime model obtained with the interpolation of the derivatives of datcom and those of the Napolitano. The aerodynamic model as seen in the thesis "Development of a mathematical model for the Bs Prime" is shown in Table 3.2 It is possible to begin a first validation phase, starting from the analysis of the trim conditions in straight horizontal flight and climb. The comparison was made only on the power required to maintain trim conditions. The flight manual does not provide information on other values such as, for

C_{x_0}	-0.0164	C_{z_0}	-0.1198	C_{m_0}	-0.2036
C_{x_α}	0.08598	C_{z_α}	-5.731	C_{m_α}	-5.831
$C_{x_{\alpha^2}}$	3.932	$C_{z_{\alpha^3}}$	9.719	$C_{m_{\alpha^2}}$	1.288
$C_{x_{\alpha^3}}$	-2.068	C_{z_q}	16.94	C_{m_q}	-31.35
$C_{x_{\delta_f}}$	-0.01334	$C_{z_{\delta_e}}$	-0.2634	$C_{m_{\delta_e}}$	-0.9516
$C_{x_{\alpha\delta_f}}$	0.4584	$C_{z_{\delta_f}}$	-0.7792	$C_{m_{\delta_f}}$	-0.8073
		$C_{z_{\alpha\delta_f}}$	-0.4539		
C_{y_β}	-0.4271	C_{l_β}	-0.03905	C_{n_β}	0.1316
C_{y_p}	-0.0739	C_{l_p}	-0.41547	C_{n_p}	-0.055921
C_{y_r}	0.3076	C_{l_r}	0.09279	C_{n_r}	-0.37982
$C_{y_{\delta_a}}$	0	$C_{l_{\delta_a}}$	-0.1467	$C_{n_{\delta_a}}$	0.00161
$C_{y_{\delta_r}}$	0.0534	$C_{l_{\delta_r}}$	0.0033	$C_{n_{\delta_r}}$	-0.0349
$C_{y_{\dot\beta}}$	0				

Table 3.2: Aerodynamic derivatives calculated, model 1

example, the deflection of mobile surfaces. As mentioned, for each condition the manual provides a combination of RPM and throttle, from which the corresponding power value can be easily obtained [1, p. 100]. At the same time, the power values required for the mathematical model were calculated through the ACTRIMforcond routine which, as mentioned, provides the necessary MAP value as output (while the RPMs must be supplied as input). For each combination of altitude and speed frome Figure 3.1, the obtain power value (Π^*) from the engin Operators Manual [7] , the calculated power value (Π), the absolute error ($\Delta\Pi$) and the relative one ($error$) are shown. Where relative error is $error = \Delta\Pi/\Pi^*$.

H $[ft]$	0	2000	4000	6000	8000
V $[m/s]$	59,16	60,19	61,21	62,24	63,27
Π^* $[kW]$	37,72	37,72	37,72	37,72	37,72
Π $[kW]$	35,24	37,12	39,14	41,27	43,47
$\Delta\Pi$ $[kW]$	-2,46	-0.60	1,42	3,55	5,76
error [%]	-7	-2	-4	9	15

Table 3.3: Horizontal flight 4300 RPM

H $[ft]$	0	2000	4000	6000	8000
V $[m/s]$	63,79	65,02	66,26	67,49	68,73
Π^* $[kW]$	45,258	45,258	45,258	45,258	45,258
Π $[kW]$	42,53	44,85	47,13	49,88	52,67
$\Delta\Pi$ $[kW]$	-2,724	-0,41	2,04	4,61	7,41
error [%]	-6	-9	5	1	16

Table 3.4: Horizontal flight 4800 RPM

H $[ft]$	0	2000	4000	6000	8000
V $[m/s]$	66,88	67,91	68,93	69,96	70,99
Π^* $[kW]$	51,29	51,29	51,29	51,29	51,29
Π $[kW]$	48,03	50,13	52,3	54,6	57,02
$\Delta\Pi$ $[kW]$	-3,26	-1,16	1,01	3,30	5,73
error [%]	-6	-2	2	6	11

Table 3.5: Horizontal flight 5000 RPM

Γ [°]	8,38	7,62	6,75	5,78	5,01
V $[m/s]$	41,15	42,15	43,2	45,39	46,56
Π^* $[kW]$	67,89	67,89	67,89	67,89	67,89
Π $[kW]$	84,72	84,71	83,26	81,66	80,02
$\Delta\Pi$ $[kW]$	16,83	16,82	15,37	13,77	12,12
error [%]	20	20	18	16	14

Table 3.6: Climb 5500 RPM

As it can be easily seen, the model used does not satisfy the climb conditions. In fact, in this condition the relative error remains around twenty percent. Being positive means that the power required to climb, according to the model, is bigger than the one evaluated using the flight manual and even bigger than the maximum power that the engine is able to produce; in particular, there's an average excess of about 15 kW with respect to the model. On the other hand, for level flight conditions there is a maximum relative error of 15 percent positive. In general, it can be observed within the same table that: for lower speeds the relative error is negative, i.e., the model requires less power than the real aircraft; on the other hand, for higher speeds the relative error becomes positive, i.e., the model requires more power than the real aircraft. For these reasons, the model thus defined is inevitably affected by errors and it does not allow to faithfully reproduce the trim conditions. Of course this is not surprising, as many assumptions were made during the modeling process. Specifically, there may be errors in the derivatives that derive from the calculations made by DATCOM, an aspect that is also reported in [9, p. 27] and in [17]; during the same modeling in DATCOM approximations were made on the shapes; measurement errors may have been introduced by using the caliper on the drawing and then making the conversions. The idea is to work on the aerodynamic model and change the aerodynamic derivatives. Also, bear in mind that the aerodynamic derivatives calculated by datcom do not consider the effects of the propeller.

3.5 Correction on aerodynamic derivatives

From the analysis of the trim conditions, it is evident that the climb conditions are the most critical: the mathematical model calculates a required power value significantly higher than that expected by the flight manual, and higher than what the engine can supply. Level flight conditions, on the other hand, are characterized by more acceptable results. In general, the model obviously requires corrections, if only to ensure that all the flight conditions reported in the manual can be effectively maintained. Among all the aerodynamic derivatives, we have chosen to focus on four of them in particular: C_{x_0} C_{x_α} $C_{x_{\alpha^2}}$ and $C_{x_{\alpha^3}}$. This choice is due to the fact that these are the main coefficients that determine the force along the X axis, and consequently strongly influence the required power. The idea was to proceed with a trial-and-error approach, trying to gradually modify the chosen values, and checking each time the difference between the results of the model and the data in the manual. The priority was to make sure that the climb conditions could also be maintained, while at the same time trying to reduce the error for the other flight conditions as well. The best results were obtained with the following changes: C_{x_0} was increased by 10%, C_{x_α} by 30%, $C_{x_{\alpha^2}}$ by 28%, and $C_{x_{\alpha^3}}$ was set equal to 0. Of course, this trial-and-error approach, while making it possible to significantly improve the model, makes it very difficult to arrive at an optimal correspondence with the experimental data.

C_{x_0}	**-0.01804**	C_{z_0}	-0.1198	C_{m_0}	-0.2036
C_{x_α}	**0.1118**	C_{z_α}	-5.731	C_{m_α}	-5.831
$C_{x_{\alpha^2}}$	**5.03296**	$C_{z_{\alpha^3}}$	9.719	$C_{m_{\alpha^2}}$	1.288
$C_{x_{\alpha^3}}$	**0**	C_{z_q}	16.94	C_{m_q}	-31.35
$C_{x_{\delta_f}}$	-0.01334	$C_{z_{\delta_e}}$	-0.2634	$C_{m_{\delta_e}}$	-0.9516
$C_{x_{\alpha\delta_f}}$	0.4584	$C_{z_{\delta_f}}$	-0.7792	$C_{m_{\delta_f}}$	-0.8073
		$C_{z_{\alpha\delta_f}}$	-0.4539		
C_{y_β}	-0.4271	C_{l_β}	-0.03905	C_{n_β}	0.1316
C_{y_p}	-0.0739	C_{l_p}	-0.41547	C_{n_p}	-0.055921
C_{y_r}	0.3076	C_{l_r}	0.09279	C_{n_r}	-0.37982
$C_{y_{\delta_a}}$	0	$C_{l_{\delta_a}}$	-0.1467	$C_{n_{\delta_a}}$	0.00161
$C_{y_{\delta_r}}$	0.0534	$C_{l_{\delta_r}}$	0.0033	$C_{n_{\delta_r}}$	-0.0349
$C_{y_{\dot\beta}}$	0				

Table 3.7: Aerodynamic derivatives modified after trim analysis, model 2

3.6 Comparison on requested power between the model 2 and the flight manual

The modified model produced more acceptable results, which are reported in the following tables. For each combination of altitude and speed, the manual power value (Π^*), the calculated power value (Π), the absolute error ($\Delta\Pi$) and the relative one (*error*) are shown. As can be seen in trim conditions in level flight, the maximum error with the new model is 12 percent, compared to 15 percent for the previous model. The most load-bearing aspect, however, occurs in climb conditions, where the error with the new model is below nine percent, against the 20 percent with the old model.

H $[ft]$	0	2000	4000	6000	8000
V $[m/s]$	59,16	60,19	61,21	62,24	63,27
Π^* $[kW]$	37,72	37,72	37,72	37,72	37,72
Π $[kW]$	33,069	34,522	35,998	37,549	39,086
$\Delta\Pi$ $[kW]$	-4,651	-3,198	-1,722	-0,171	1,366
error [%]	-12	-8	-5	0	4

Table 3.8: Horizontal flight 4300 RPM

H $[ft]$	0	2000	4000	6000	8000
V $[m/s]$	63,79	65,02	66,26	67,49	68,73
Π^* $[kW]$	45,258	45,258	45,258	45,258	45,258
Π $[kW]$	41,779	43,816	45,937	48,103	50,372
$\Delta\Pi$ $[kW]$	-3,479	-1,442	0,679	2,845	5,114
error [%]	-8	-3	2	6	11

Table 3.9: Horizontal flight 4800 RPM

H $[ft]$	0	2000	4000	6000	8000
V $[m/s]$	66,88	67,91	68,93	69,96	70,99
Π^* $[kW]$	51,29	51,29	51,29	51,29	51,29
Π $[kW]$	48,259	50,046	51,848	53,711	55,613
$\Delta\Pi$ $[kW]$	-3,031	-1,244	0,558	2,421	4,323
error [%]	-6	-2	1	5	8

Table 3.10: Horizontal flight 5000 RPM

Γ [°]	8,38	7,62	6,75	5,78	5,01
V $[m/s]$	41,15	42,15	43,2	45,39	46,56
Π^* $[kW]$	67,89	67,89	67,89	67,89	67,89
Π $[kW]$	74,22	73,29	70,939	69,178	66,491
$\Delta\Pi$ $[kW]$	6,33	5,4	3,049	1,288	-1,399
error [%]	9	8	4	2	-2

Table 3.11: Climb 5500 RPM

3.7 Comparison of a trim condition between model 1 and 2

Now go back to the example shown in section 3.2, a comparison can now be made by using model 2. The load conditions and initial conditions (altitude, speed, rpm) are the same as in the example in the section 3.2. The procedure for obtaining these results is the same as already described.

$$X = \begin{bmatrix} V \\ \alpha \\ \beta \\ p \\ q \\ r \\ \psi \\ \theta \\ \phi \\ x_e \\ y_e \\ H \end{bmatrix} = \begin{bmatrix} 60 & [m/s] \\ 4.3898 \cdot 10^{-2} & [rad] \\ -5.1974 \cdot 10^{-5} & [rad] \\ 0 & [rad/s] \\ 0 & [rad/s] \\ 0 & [rad/s] \\ 0 & [rad] \\ 4.3898 \cdot 10^{-2} & [rad] \\ 0 & [rad] \\ 0 & [m] \\ 0 & [m] \\ 1220 & [m] \end{bmatrix} ; \quad U = \begin{bmatrix} \delta_e \\ \delta_a \\ \delta_r \\ \delta_f \\ n \\ pz \\ uw \\ vw \\ ww \\ \dot{uw} \\ \dot{vw} \\ \dot{ww} \end{bmatrix} = \begin{bmatrix} -2.0889 \cdot 10^{-1} & [rad] \\ -4.7627 \cdot 10^{-3} & [rad] \\ -4.1570 \cdot 10^{-4} & [rad] \\ 0 & [rad] \\ 5000 & [RPM] \\ 22.74 & ["Hg] \\ 0 & [m/s] \\ 0 & [m/s] \\ 0 & [m/s] \\ 0 & [m/s^2] \\ 0 & [m/s^2] \\ 0 & [m/s^2] \end{bmatrix}$$

$$\dot{X} = \begin{bmatrix} \dot{V} \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{p} \\ \dot{q} \\ \dot{r} \\ \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \\ \dot{x_e} \\ \dot{y_e} \\ \dot{H} \end{bmatrix} = \begin{bmatrix} 2.1487 \cdot 10^{-15} & [m/s^2] \\ 2.3715 \cdot 10^{-17} & [rad/s] \\ 9.1909 \cdot 10^{-18} & [rad/s] \\ 9.9431 \cdot 10^{-18} & [rad/s^2] \\ -3.7454 \cdot 10^{-16} & [rad/s^2] \\ 1.9298 \cdot 10^{-16} & [rad/s^2] \\ 0 & [rad/s] \\ 0 & [rad/s] \\ 0 & [rad/s] \\ 60 & [m/s] \\ -3.1185 \cdot 10^{-3} & [m/s] \\ 0 & [m/s] \end{bmatrix}$$

$$\Pi = 34.55 \; kW \quad \xi = 45 \; \%$$

In general, if you compare the results just obtained with those in the 3.2 section, it is possible to see minimal changes in the results in the state vector X as well as in the vector U, while some terms in the vector $\dot{(X)}$ are decreased by an order of magnitude. What varies instead is the power and therefore the throttle, in fact they have decreased. By using model 2 the power is $\Pi = 34.55 \; kW$, whereas by model 1 the power is $38.04 \; kW$. These results are in line with expectations, since in model 2 the force coefficient C_x was modified, because it affects the calculation of the power more markedly.

3.7.1 Trim conditions as speed and altitude vary

In the paragraph about the ACTRIM routine we talked about the possibility that this function makes an error in the calculation of the trim conditions. This error is due to the process of minimizing the cost function J, which can lead to the identification of local and not absolute minimums. A good way to ensure that this does not happen is to analyze the graphs shown in this section, which show the main quantities as speed and altitude vary. The graphs shown here refer to the modified mathematical model.

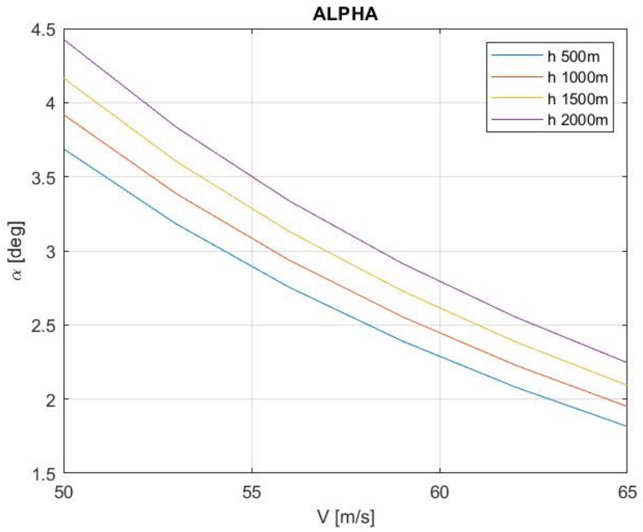


Figure 3.2: α trim

All the graphs show a gradual trend, without abrupt variations of the quantities involved. This is sufficient to verify that no local minimum was found under any of the calculated trim conditions. In Figure 3.2, the angle of attack is undoubtedly one of the fundamental variables for studying the flight mechanics of an aircraft, because it measures the inclination of the longitudinal axis of the aircraft in relation to the direction of the flow. The first effect that can be seen is that alpha decreases as the speed grows; this was expected, as the lift required to maintain the flight condition remains the same, so when the speed increases, a lower value for alpha is necessary. Also, α increases for higher altitudes, as the lift is proportional to density too.

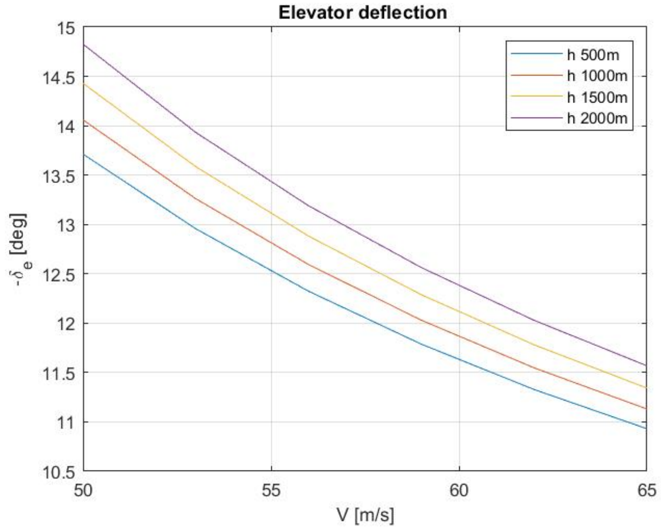


Figure 3.3: δ_e trim deflections

In figure 3.3, the elevator is the control surface that ensures the longitudinal control of the aircraft, so the angular positions it assumes at different speeds affect the controllability and stability of the aircraft. As it can be seen, as the speed increases there is a decrease in the deflection of elevator. This was expected because, while the speed increases, the force generated by the elevator need to remain constant, so a smaller deflection is required. Also, the value decreases for higher altitudes, for the same reason explained for α.

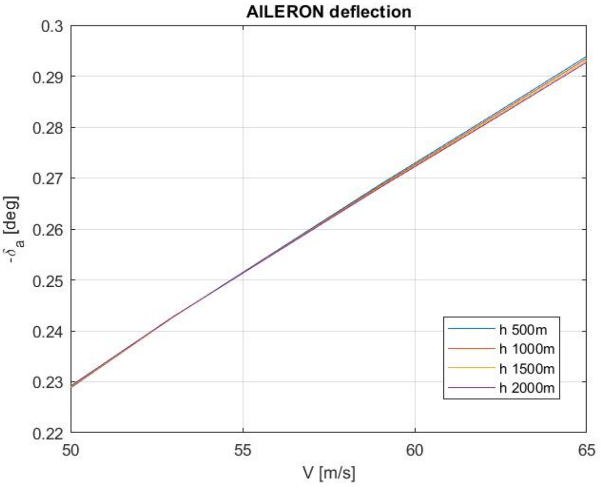


Figure 3.4: δ_a trim deflections

In Figure 3.4, control over the roll moment is provided by ailerons. In trim conditions these surfaces should not have particularly large deflections, but they are however slightly deflected as they must balance the propeller's torque. In fact, it is well shown that as the speed increases, the ailerons tend to deflect more as the torque increases. It should be noted that the deflections are however small as the ailerons of the Bs Prime are very extended in terms of surface, so small deflections are enough to produce important control moment.

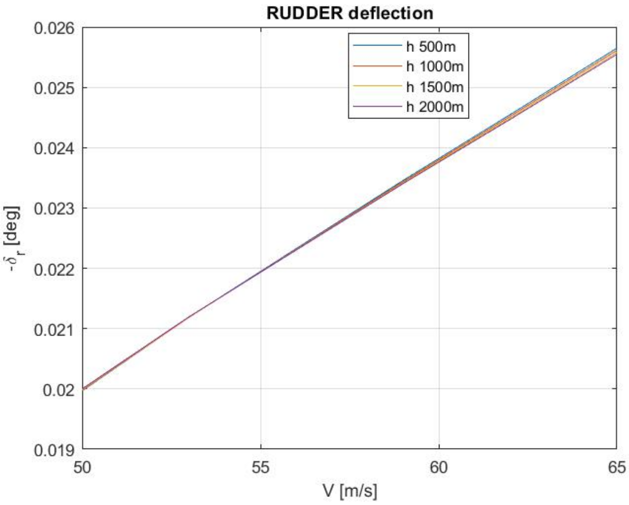


Figure 3.5: δ_r trim deflections

In Figure 3.5, the rudder provides the control over the aircraft along the Z_{body} axis, which is used to generate and compensate for the yawing moment. As already observed for the ailerons, also in this case the rudder must be as close as possible to the neutral position. From the graph it is possible to see this: its deflection is an order of magnitude lower than the one of the ailerons. In any case, a minimum deflection is present, as in theory it serves to compensate the effect of the coupling of the dynamics due to the deflection of the other mobile surfaces. In particular the deflection increases for higher speed values, exactly as it happens for the ailerons. In fact, this is aimed at compensating for the yaw effect due to the deflection of the ailerons. In Figure

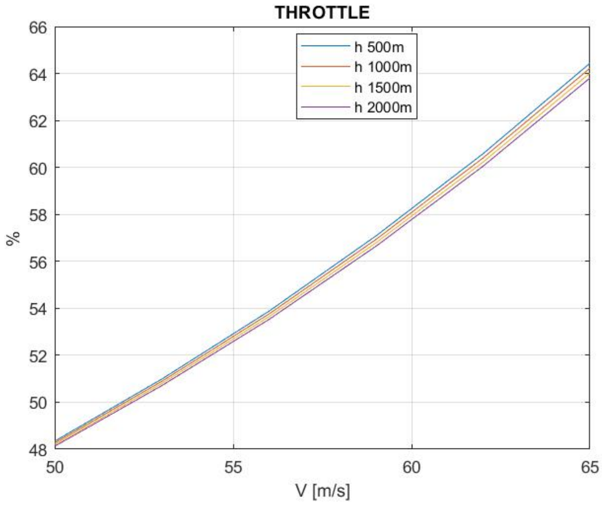


Figure 3.6: Throttle trim %

3.6, as already explained in chapter 5, the throttle is obtained as an interpolation function using only the data available from engine Operators Manual [7]. Therefore this interpolation function is affected by a certain error, both for the interpolation process and for the fact that the idle power values have been assumed, as they are absent in the flight manual and in the available engine documents. So the throttle percentage (ξ) is given by the following equation:

$$\xi = 0.00136 \cdot \Pi_{d_{sl}} - 2.033 \tag{3.3}$$

From the Figure 3.6 it is seen that as the speed increases, the percentage of throttle required increases.

3.8 Maintenance of trim conditions

As already observed, the trim represents a condition for which the sum of moments and forces on the aircraft are zero, therefore a condition of equilibrium. This means that if this condition were maintained, the aircraft would have to fly in a uniform straight motion. Therefore, there's the need to verify the maintenance of the trim over time. To do this, the complete model of the Bs Prime in open loop is used, as shown in Figure 5.30. Note that to observe the effective maintenance of the trim all the external controls that are present in the figure have been excluded. To proceed with the verification of the maintenance of the trim conditions over time, first it is necessary to find the trim conditions using ACRIM. The latter generates the trim vectors: X, $\dot{X}$ and U. These vectors are the quantities that the open looop receives as input. Also in this case reference is made to the trim conditions identified in the previous example. The following figures show the trajectory of the aircraft and the trend of the parameters: incidence α, sideslip angle β, TAS and altitude. The time of simulation is 200 seconds.

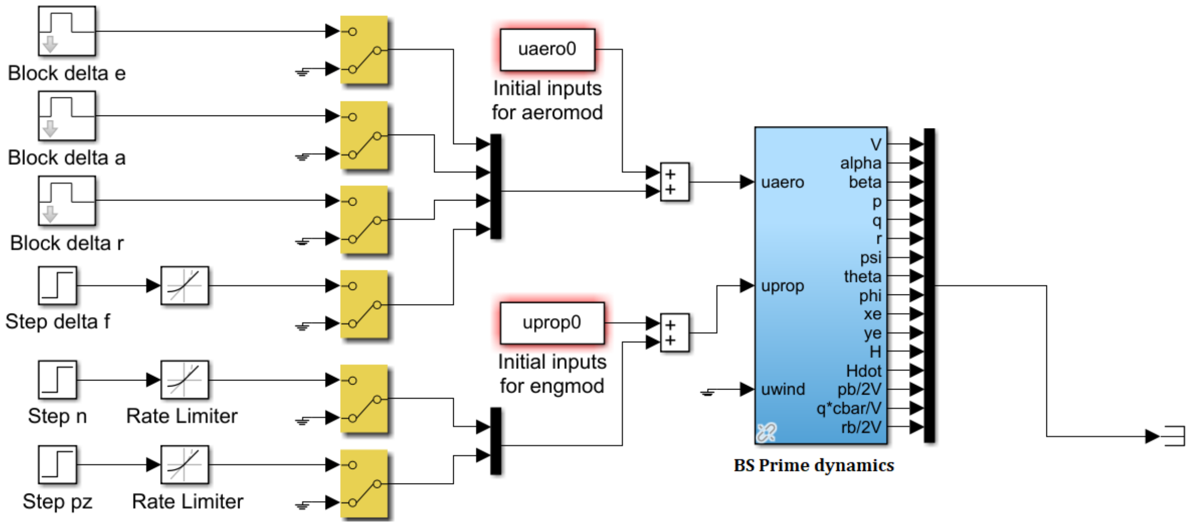


Figure 3.7: BS Prime in openloop

It can be easily seen how the trim conditions are effectively maintained over time. In fact, the trajectory is straight and the incidence remains practically constant. At the same time the speed is maintained as well as the altitude. The variation of β is negligible as can be seen from the graph. All this suggests that the conditions found globally guarantee the aircraft a condition of equilibrium.

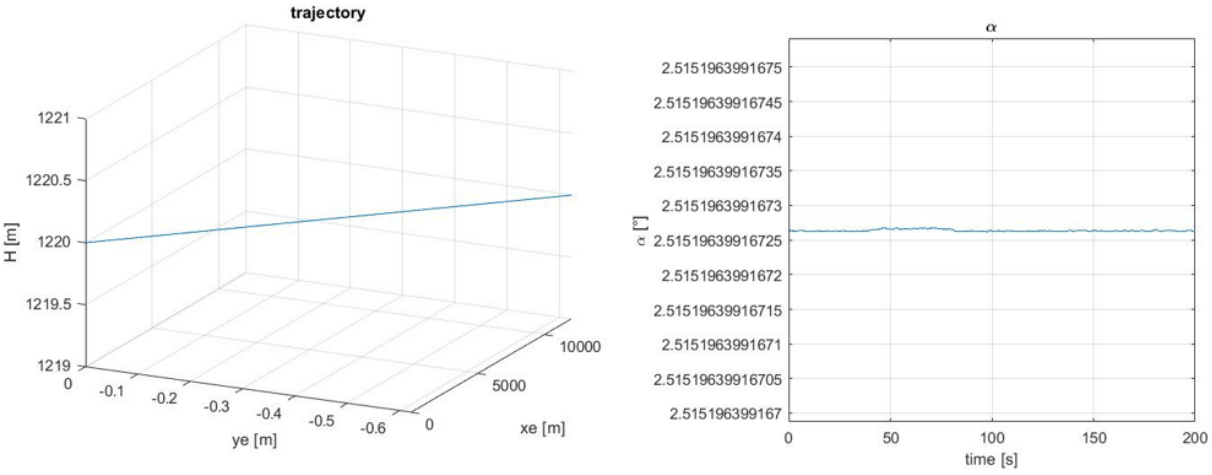


Figure 3.8: Trajectory and Incidenze

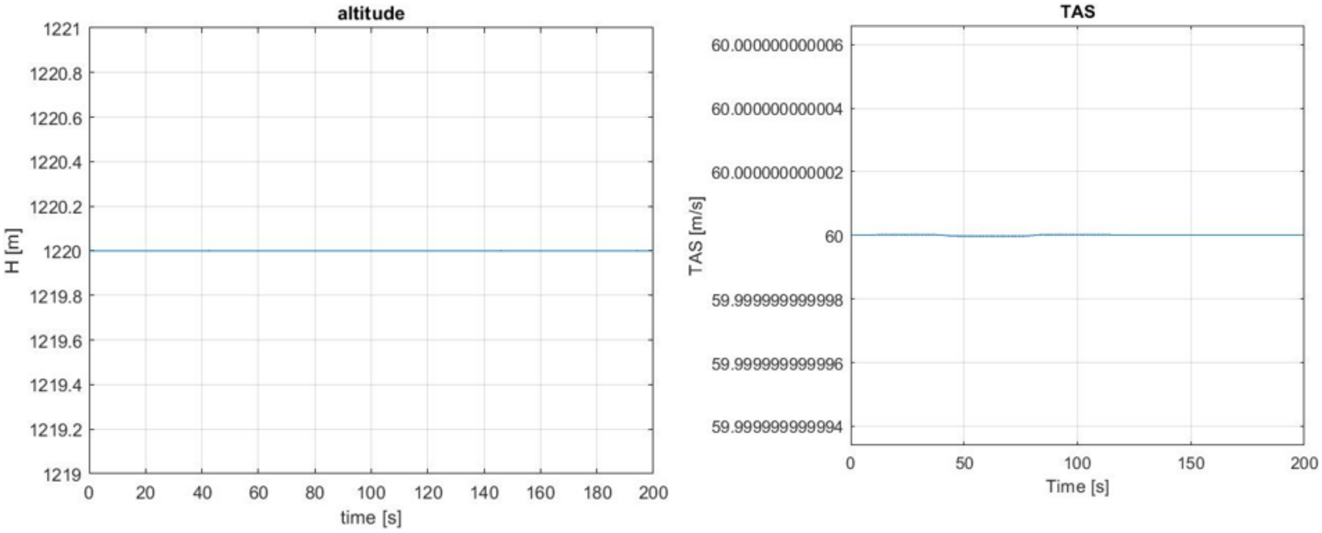


Figure 3.9: Altitude and TAS

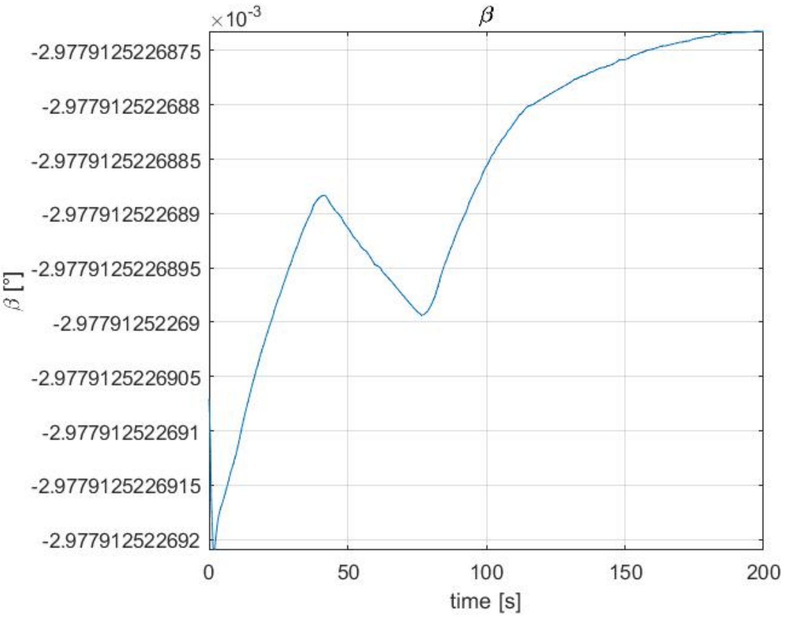


Figure 3.10: Sideslip angle

Chapter 4

Stability

4.1 Introduction

After the tests for the various trim conditions, the longitudinal and lateral-directional dynamic response was analyzed, through the calculation of the eigenvalues of the corresponding state matrices, and through the linearization of the equations of motion. As regards the longitudinal plane, the solutions expected for the characteristic polynomial of the fourth order are: a pair of complex and conjugated eigenvalues that corresponds to the phugoid, a periodic mode characterized by the variables V and α, with a much greater period than the other mode; a pair of complex and conjugated eigenvalues that corresponds to the short-period mode, a periodic mode characterized by the variables q and θ, with a much shorter period than the phugoid. As regards the lateral-directional plane, the solutions expected for the fourth-order characteristic polynomial are: a real eigenvalue, much lower than zero, which corresponds to the roll mode (aperiodic); a second real eigenvalue, with an absolute value closer to zero, which corresponds to the spiral mode (aperiodic); a complex and conjugated couple of eigenvalues that corresponds to the Duch roll (periodic).

4.2 Linearization of the equations of motion

The linearization of the equations of motion occurs through the appropriate ACLIN.m routine, already included in FDC [11]. The process requires a trim condition as input, which can be called from file, entered manually, or calculated on the spot by calling ACTRIM.m. This trim condition is used as a starting point around which to linearize the equations. This last aspect allows to separate the study of the longitudinal dynamics from the lateral-directional one. ACLIN also allows you to reduce the system state matrix, selecting the variables to be analyzed. The linearization process is done inside ACLIN by a maltlab function called LINMOD. For the purposes of this discussion the ACLIN function has been modified: once the aerodynamic model referred to the datum is received as input, the aerodynamic derivatives are recalculated based on the position of the center of gravity.

4.2.1 State matrix

Once the state matrix of the linearized system has been obtained, ACLIN provides the possibility of reducing the system by selecting a limited number of variables; thus a reduced state matrix is obtained. This allows the variables of longitudinal and lateral-directional dynamics to be isolated separately, [11, p. 150]. In this discussion, the variables $[V\ \alpha\ q\ \theta]$ were used for the longitudinal dynamics, while for the lateral-directional dynamics $[\beta\ p\ r\ \phi]$ were used, [14, p. 205]. The variables related to navigation have been completely ignored, since they do not affect the dynamics modes that we intend to analyze in this chapter. In this discussion, the two reduced matrices will be denoted as A_a and A_b respectively.

4.3 Example of dynamic response analysis

Once the A_a and A_b matrices have been defined, it is possible to calculate their eigenvalues, so as to perform a dynamic stability analysis. The load conditions are those reported in the 3.2 section, instead the trim conditions are those reported in the 3.7 section, i.e. using model 2. For this case, a center of gravity position equal to 28 percent of the mean aerodynamic chord was considered.

$$A_a = \begin{bmatrix} -2.7413 \cdot 10^{-2} & 9.4387 & 2.2083 \cdot 10^{-1} & -9.8029 \\ -5.4260 \cdot 10^{-3} & -2.9452 & 1.0838 & -8.9574 \cdot 10^{-13} \\ 3.3442 \cdot 10^{-3} & -17.588 & -4.6724 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$A_b = \begin{bmatrix} -2.2093 \cdot 10^{-1} & 4.1078 \cdot 10^{-2} & -9.9121 \cdot 10^{-1} & 1.6322 \cdot 10^{-1} \\ -2.4197 \cdot 10^{1} & -1.4446 \cdot 10^{1} & 2.7422 & 0 \\ 6.1533 & 3.0652 \cdot 10^{-1} & -1.1743 & 0 \\ 0 & 1.0000 & 4.3927 \cdot 10^{-2} & 0 \end{bmatrix}$$

In addition, period, natural frequancy, damping, halving time, halving cycles are also calculated for periodic modes. In the case of aperiodic modes, however, only the halving time can be calculated.

Longitudinal dynamic

The eigenvalues for the A_a matrix are:

$$\lambda_a = \begin{bmatrix} -3.8092 + 4.2823i \\ -3.8092 - 4.2823i \\ -1.3319 \cdot 10^{-2} + 1.7676 \cdot 10^{-1} i \\ -1.3319 \cdot 10^{-2} - 1.7676 \cdot 10^{-1} i \end{bmatrix}$$

- **Short period**: the first pair of complex and conjugated eigenvalues $\lambda_{1,2}$, as confirmed by the calculations, defines the short period; the fact that the real part is negative guarantees that the mode is stable. The following are calculated, being a periodic mode, respectively: period, halving time, halving cycles.

  $$T = \frac{2\Pi}{Im(\lambda_{1,2})} = 1.47\ s \qquad t_{1/2} = \frac{ln2}{|Re(\lambda_{1,2}|)} = 0.18 \qquad N_{1/2} = \frac{t_{1/2}}{T} = 0.12 \tag{4.1}$$

  The values are in line with those typically expected for small aircraft Natural frequency and damping can also be calculated, being a periodic mode, as:

  $$\omega_n = \sqrt{Im(\lambda_{1,2})^2 + Re(\lambda_{1,2})^2} = 5.73\ rad/s \qquad \zeta = \frac{-Re(\lambda_{1,2})}{\omega_n} = 0.66 \tag{4.2}$$

- **Phugoid**: the second pair of complex and conjugated eigenvalues $\lambda_{3,4}$, as confirmed by the following calculations, defines the phugoid; the fact that the real part is negative guarantees that the mode is stable. The following are calculated, being a periodic mode, respectively:

period, halving time, halving cycles.

$$T = \frac{2\Pi}{Im(\lambda_{3,4})} = 35.56\ s \qquad t_{1/2} = \frac{ln2}{|Re(\lambda_{3,4}|)} = 52 \qquad N_{1/2} = \frac{t_{1/2}}{T} = 1.46 \tag{4.3}$$

The values are in line with those typically expected for small aircraft. Natural frequency and damping can also be calculated, being a periodic mode, as:

$$\omega_n = \sqrt{Im(\lambda_{3,4})^2 + Re(\lambda_{3,4})^2} = 0.177\ rad/s \qquad \zeta = \frac{-Re(\lambda_{3,4})}{\omega_n} = 0.075 \tag{4.4}$$

Lateral-directional dynamic

The eigenvalues for the A_b matrix are:

$$\lambda_b = \begin{bmatrix} -1.4418 \cdot 10^{1} + 0.0000i \\ -7.0431 \cdot 10^{-1} + 2.3725i \\ -7.0431 \cdot 10^{-1} - 2.3725i \\ -1.4716 \cdot 10^{-2} + 0.0000i \end{bmatrix}$$

- **Roll mode**: the first eigenvalue λ_1 is real, in particular with very negative real part, therefore stable.

- **Dutch roll**: the complex conjugated pair $\lambda_{2,3}$, defines the Dutch roll motion, characterized by the real negative part, therefore it is stable. The following are calculated, being a periodic mode, respectively: period, halving time, halving cycles.

$$T = \frac{2\Pi}{Im(\lambda_{2,3})} = 2.65\ s \qquad t_{1/2} = \frac{ln2}{|Re(\lambda_{2,3}|)} = 0.98 \qquad N_{1/2} = \frac{t_{1/2}}{T} = 0.37 \tag{4.5}$$

The values are in line with those typically expected for small aircraft. Natural frequency and damping can also be calculated, being a periodic mode, as:

$$\omega_n = \sqrt{Im(\lambda_{2,3})^2 + Re(\lambda_{2,3})^2} = 2.47\ rad/s \qquad \zeta = \frac{-Re(\lambda_{2,3})}{\omega_n} = 0.29 \tag{4.6}$$

- **Spiral mode**: the second real eigenvalue λ_4, is characterized only by the real part. It is weakly negative, therefore always stable, thus denoting Spiral mode.

The values are in line with those typically expected for small aircraft.

4.4 Influence of C.G. position, weight, speed and altitude on eigenvlues

The dynamic response of the aircraft depends on several factors: position of center of gravity, overall weight, flight speed, and altitude. In this section it will be analyzed how these parameters affect the eigenvalues of the reduced state matrices.

4.4.1 Influence of C.G. position

To analyze the dependence of the dynamic response on the position of the center of gravity, the AclinforCG function was used. Therefore it was necessary to start from an initial trim condition that refers to the initial load conditions reported in the section 4.3. The overall weight also remains unchanged, but the position of the center of gravity has been made to vary. The flight manual provides a range of expected values within which the position of the center of gravity must fall, Table 4.1. The values are provided as a percentage of the mean aerodynamic chord. In order to test the model, two further positions outside this range were then added to analyze the eigenvalue trend in a more complete way. The positions used are shown below, both in %MAC and in m with respect to the datum.

$x_{c.g}$ [%]	21.5	24	26	28	30	32	34	36	40	45
$x_{c.g}$ [m]	0.9492	0.9805	1.0055	1.0306	1.0556	1.0806	1.1057	1.1307	1.1808	1.2434

Table 4.1: $x_{c.g}$ positions

The analysis of the eigenvalue trend, on the other hand, is carried out through the use of two types of graphs: the stability diagrams and the Root Locus. The first type shows the real or imaginary part of the eigenvalues as the derivative C_{m_α} varies, respectively Figures 4.1 and 4.2 .

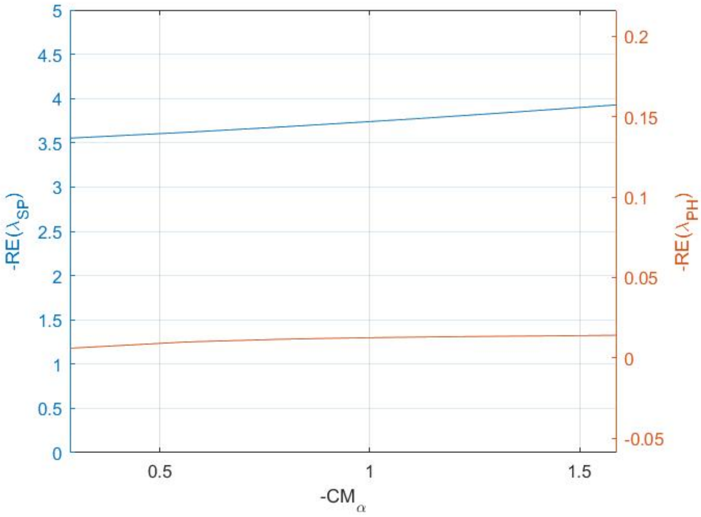


Figure 4.1: Stability diagram ($C_{m_\alpha}, \Re$)

Remember that this derivative is directly proportional to the difference between the position of the center of gravity and that of the neutral point; therefore, it is in fact a measure of the position of the center of gravity itself. By convention, the axes of this chart are reversed, in this way the eigenvalues corresponding to stable modes fall into the first quadrant. In the second quandrant, dynamic stability is guaranteed, but not static stability; in the fourth the viceversa occurs; finally, the eigenvalues falling into the third quadrant imply both static and dynamic instability. The Root Locus, on the other hand, is a simple Cartesian plane on which the complex eigenvalues are represented, and allows you to immediately view the trend of both the real and the complex part.

The stability diagrams refer only to the longitudinal dynamics, while the root locus also refer to the lateral-directional one, as it can be seen in Figurs 4.3 and 4.4.

From Figures 4.3 and 4.4, it can be seen very easily how the retreat of the position of the center of gravity leads the various modes towards a condition of greater instability: the real part becomes less negative, yet always remains stable. It is to be noticed that the trend of imaginary part of the phugoid eigenvalues is opposite to that expected from literature [6], characterized by an increase in the imaginary part and a decrease in damping.

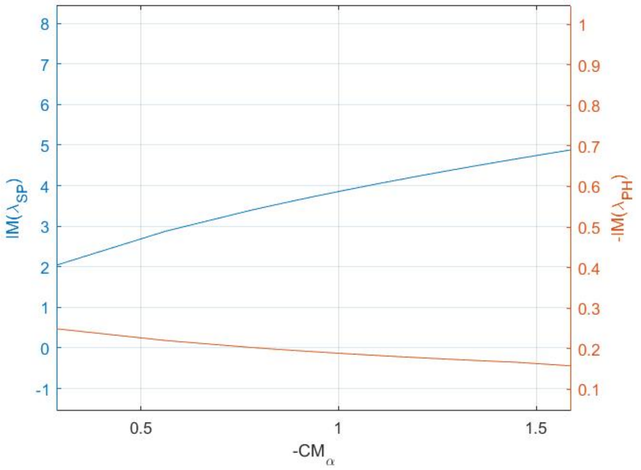


Figure 4.2: Stability diagram, $(C_{m_\alpha}, \Im)$

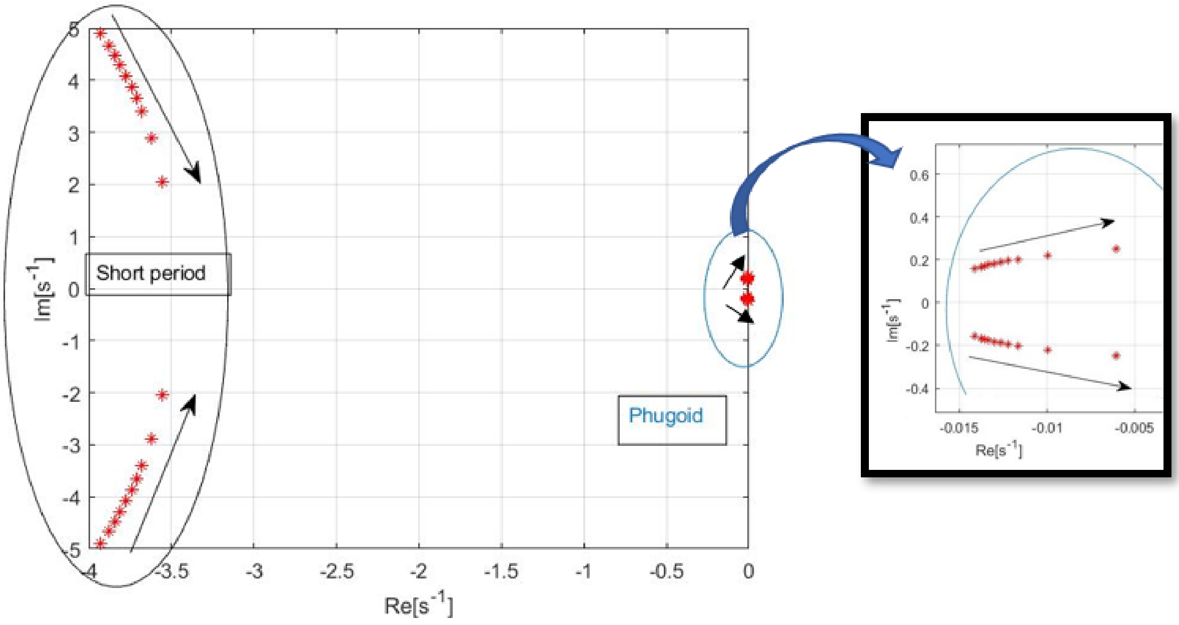


Figure 4.3: Root Locus of longitudinal dynamics

This last figure 4.5 represents the expected trend from literature of the eigenvalues of the longitudinal dynamics. As the position of the center of gravity recedes, and consequently moves from right to left in the graph, the two pairs of complex and conjugated eigenvalues are replaced by 4 real eigenvalues, corresponding to 4 non-periodic modes. As we continue to retract the position of the center of gravity, two of these eigenvalues will tend to couple again, giving rise to a third periodic mode.

4.4.2 Influence of weight, speed and altitude

The AclinforCG routine linearizes the equations of motion starting from a specific trim condition, which depends mainly on altitude and speed and, to a lesser extent, on TOW. It is therefore necessary to analyze how much these quantities influence the model behavior. Even if this function

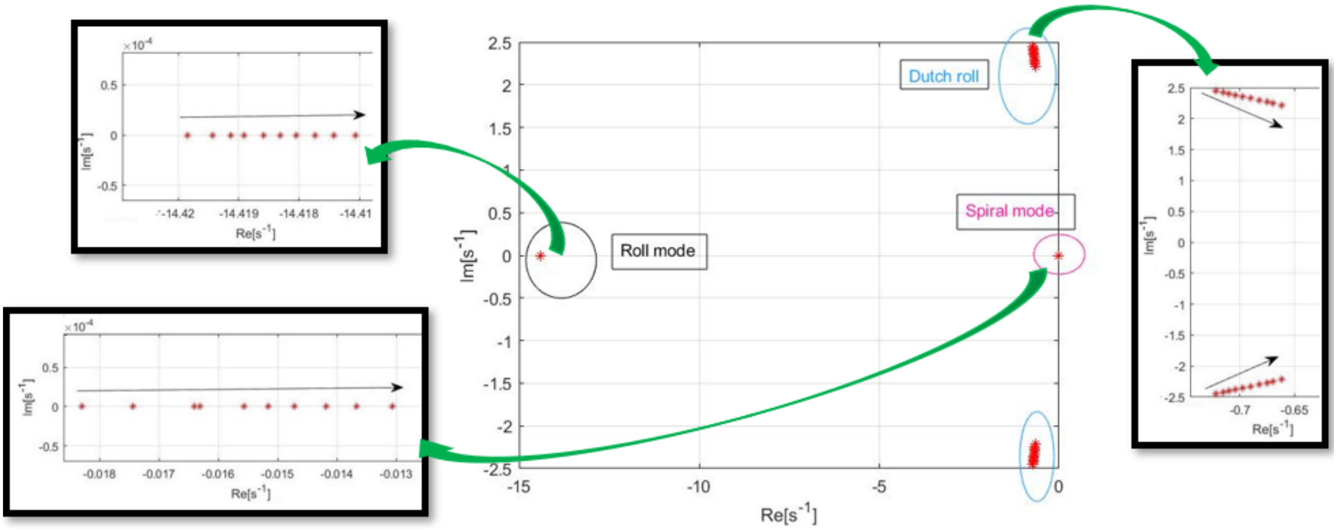


Figure 4.4: Root Locus of latero directional dynamics

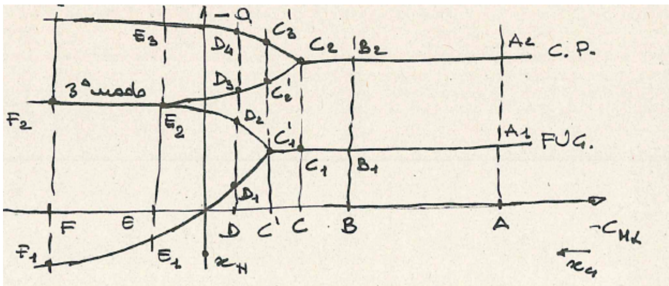


Figure 4.5: Stability diagram from literature

makes the center of gravity vary, it is possible to study the dynamic behavior when the conditions listed above vary, defined a certain position of the center of gravity. In this section, 28 percent of the mean aerodynamic chord has been chosen as the position, as well as in the 4.3 section. The initial trim conditions, necessary for the functions, differ here from those used in the other examples. Two tables are reported below, one for the longitudinal dynamics and one for the lateral directional dynamics, within which the eigenvalues for the different conditions are inserted.

	Short Period	Phugoid
600 kg	-4.3874 ± 4.7856	$-1.4911 \cdot 10^{-2} \pm 1.6425 \cdot 10^{-1}$
500 kg	-4.3880 ± 4.8050	$-1.3586 \cdot 10^{-2} \pm 1.8954 \cdot 10^{-1}$
0 m	-4.6548 ± 4.9347	$-1.5933 \cdot 10^{-2} \pm 1.6292 \cdot 10^{-1}$
3048 m	-3.4332 ± 4.2179	$-1.1241 \cdot 10^{-2} \pm 1.6947 \cdot 10^{-1}$
30 m/s	-2.3055 ± 2.5399	$-1.1241 \cdot 10^{-2} \pm 1.6947 \cdot 10^{-1}$
76 m/s	-5.4736 ± 5.7958	$-1.9672 \cdot 10^{-2} \pm 1.4404 \cdot 10^{-1}$

Table 4.2: Influence on longitudinal dynamics

The tables have been prefered to the Root Locus because the variations are very small in some cases, therefore they can be better appreciated as values in the Tabel 4.2 and Table 4.3

In the following, the characteristics of the modes in terms of natural frequency, damping and

	Roll mode	Dutch roll	Spiral Mode
600 kg	$-1.6611 \cdot 10^1$	$-7.9892 \cdot 10^{-1} \pm 2.6324$	$-1.5339 \cdot 10^{-2}$
500 kg	$-1.6617 \cdot 10^1$	$-8.0108 \cdot 10^{-1} \pm 2.6356$	$-1.6229 \cdot 10^{-2}$
0 m	$-1.7629 \cdot 10^1$	$-8.4304 \cdot 10^{-1} \pm 2.7038$	$-1.5728 \cdot 10^{-2}$
3048 m	$-1.2977 \cdot 10^1$	$-6.4436 \cdot 10^{-1} \pm 2.3565$	$-1.3431 \cdot 10^{-2}$
30 m/s	-8.9512	$-4.9181 \cdot 10^{-1} \pm 1.4548$	$4.1716 \cdot 10^{-3}$
76 m/s	$-2.0736 \cdot 10^1$	$-9.8124 \cdot 10^{-1} \pm 3.1610$	$-1.5012 \cdot 10^{-2}$

Table 4.3: Influence on latero directional stability

halving time are reported, in isolated tables so as to be able to better notice the differences.

- Weight influence:

Mass $[kg]$	Short Period		Phugoid	
	ζ	ω_n $[rad/s]$	ζ	ω_n $[rad/s]$
600	0.65	6.72	0.09	0.165
500	0.67	6.51	0.072	0.19

Table 4.4: Influence of weight on longitudinal dynamics

Mass $[kg]$	Roll mode	Dutch roll		Spiral Mode
	$t_{1/2}$	ζ	ω_n $[rad/s]$	$t_{1/2}$
600	0.042	0.29	2.75	45.19
500	0.042	0.29	2.75	42.71

Table 4.5: Influence of weight on lateral-directional stability

- Altitude influence:

Altitude $[m]$	Short Period		Phugoid	
	ζ	ω_n $[rad/s]$	ζ	ω_n $[rad/s]$
0	0.69	6.78	0.097	0.164
3048	0.63	5.44	0.066	0.17

Table 4.6: Influence of altitude on longitudinal dynamics

Altitude $[m]$	Roll mode	Dutch roll		Spiral Mode
	$t_{1/2}$	ζ	ω_n $[rad/s]$	$t_{1/2}$
0	0.039	0.30	2.83	44.07
3048	0.053	0.26	2.44	51.6

Table 4.7: Influence of weight on lateral-directional stability

- Speed influence:

As it can be seen, a variation in TOW minimally affects the results. In fact, there are small differences in the longitudial dynamics, the values on the other hand for the lateral directional

Speed (TAS) $[m/s]$	Short Period		Phugoid	
	ζ	ω_n $[rad/s]$	ζ	ω_n $[rad/s]$
33	0.69	3.34	0.039	0.304
76	0.687	7.97	0.136	0.145

Table 4.8: Influence of speed on longitudinal dynamics

Speed (TAS) $[m/s]$	Roll mode	Dutch roll		Spiral Mode
	$t_{1/2}$	ζ	ω_n $[rad/s]$	$t_{1/2}$
33	0.077	0.32	1.54	not stable
76	0.033	0.30	3.31	46.17

Table 4.9: Influence of speed on lateral-directional stability

remain almost unchanged except for the spiral mode. On the other hand, the differences following a change in altitude or speed are more evident. In particular it is interesting to note that when the speed value is very low, 30 m/s, the spiral mode becomes unstable.

4.5 Comparison of dynamics characteristics with similar aircrafts

As mentioned, the calculated eigenvalues and the related dynamics characteristics are in line with the expected values for ultralight aircrafts. In order to have a further comparison on the results, it was decided to search for data relating to similar aircrafts. In particular, it was possible to find some characteristic values for the following aircrafts: BS115, CESSNA 172, DA-20, [9, p. 100]. The first two have an MTOW of around 750 kg, while the third only reaches up to around 530 kg. However, these are aircraft similar enough to BS prime to be used for comparison. The BS115, in particular, can be considered an "ancestor" of the BS Prime. The latter was in fact developed starting from BS115, with the aim of reducing the MTOW to 600 kg, in order to classify the aircraft as an ultralight. The following tables compare the values that could be found in the literature. Note that the position of the center of gravity to which the data of BS115, CESSNA 172, DA-20 correspond is not specified in the refrence [9]

- Spiral mode, comparison on halving time [s]

Aircraft	halving time [s]
Cessna 172	34.3
BS115 DATCOM	27.3
DA-20	140
BS Prime	47.1

Table 4.10: Spiral mode

- Dutch Roll, comparison on damping and natural frequency[rad/s].
  It must be noted that the regulation requires that the damping must be greater than 0.08, and that the natural frequency is greater than 1 rad/s
- Short period, comparison on damping

Aircraft	ζ	ω_n $[rad/s]$
Cessna 172	0.1749	2.66
BS115 DATCOM	0.29	1.74
DA-20	0.149	2.17
BS Prime	0.29	2.47

Table 4.11: Dutch Roll

Aircraft	ζ
Cessna 172	0.685
BS115 DATCOM	0.574
DA-20	0.559
BS Prime	0.66

Table 4.12: Short period

- Phugoid, comparison on damping

Aircraft	ζ
Cessna 172	0.104
BS115 DATCOM	0.115
BS115 Flight test	0.156
DA-20	0.056
BS Prime	0.075

Table 4.13: Phugoid

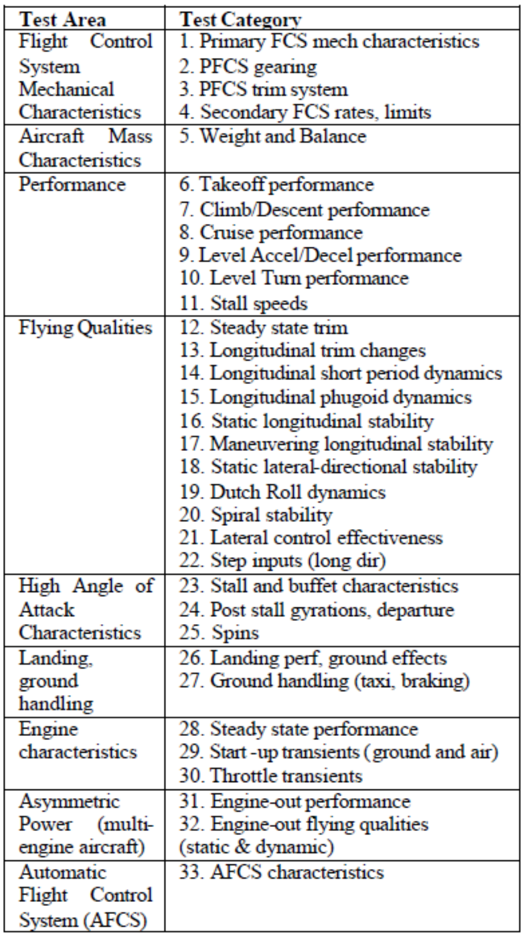
The comparison demonstrates the validity of the results obtained. Unfortunately it was impossible to make a comparison on Roll mode, due to the difficulty in finding data of this type.

Chapter 5

Flight test

5.1 Introduction

The possibility of carrying out in-flight tests, using a Blackshape Prime aircraft, was very useful in order to have a further term of comparison for the mathematical model, in addition to the aforementioned data obtained from the flight manual. The "EURO FLIGHT TEST" company has given its willingness to make two flights of about one hour each, in order to analyze the performance of the aircraft. The tests focused on performance in climb condition, in particular



Test Area	Test Category
Flight Control System Mechanical Characteristics	1. Primary FCS mech characteristics 2. PFCS gearing 3. PFCS trim system 4. Secondary FCS rates, limits
Aircraft Mass Characteristics	5. Weight and Balance
Performance	6. Takeoff performance 7. Climb/Descent performance 8. Cruise performance 9. Level Accel/Decel performance 10. Level Turn performance 11. Stall speeds
Flying Qualities	12. Steady state trim 13. Longitudinal trim changes 14. Longitudinal short period dynamics 15. Longitudinal phugoid dynamics 16. Static longitudinal stability 17. Maneuvering longitudinal stability 18. Static lateral-directional stability 19. Dutch Roll dynamics 20. Spiral stability 21. Lateral control effectiveness 22. Step inputs (long dir)
High Angle of Attack Characteristics	23. Stall and buffet characteristics 24. Post stall gyrations, departure 25. Spins
Landing, ground handling	26. Landing perf, ground effects 27. Ground handling (taxi, braking)
Engine characteristics	28. Steady state performance 29. Start -up transients (ground and air) 30. Throttle transients
Asymmetric Power (multi-engine aircraft)	31. Engine-out performance 32. Engine-out flying qualities (static & dynamic)
Automatic Flight Control System (AFCS)	33. AFCS characteristics

Figure 5.1: Classical Flight Tests for Fixed Wing Aircraft

the cruise acceleration method was used. A fundamental part of the validation process are the flight tests. They are unanimously recognized and accepted methods. They are divided into two categories: performance tests and flight quality tests. The former focus on characteristics such as lift, drag, fuel consumption, etc. Flight quality tests, on the other hand, analyze the characteristics of stability and controllability that affect the pilot's workload. A technical flight test list is shown in Figure 5.1. In this discussion for the validation of the model, only the performance tests were carried out as already observed, in particular for the climb performance.

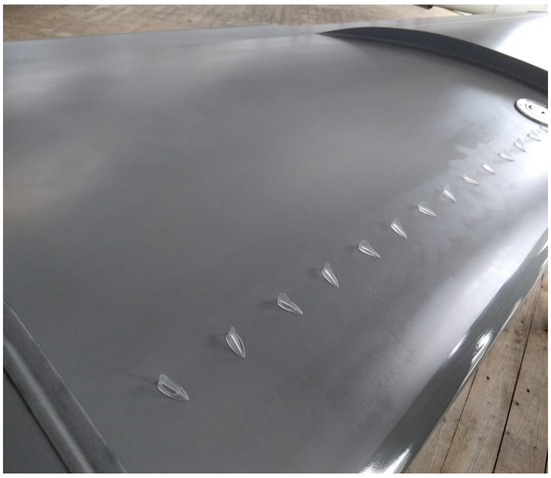


Figure 5.2: Vortex generators on BS Prime wing

It is important to note that at the time of the tests, vortex generators were installed on the aircraft. These components significantly vary the performance of the aircraft: drag increases, stall speed is slightly reduced, while maximum speed is significantly lower. It follows that the performance of the aircraft in climb conditions will also be affected. In particular, the aircraft will find it harder to climb in altitude. Considering the cruise acceleration tests, however, the greater aerodynamic drag reduces the acceleration of the aircraft.

5.1.1 Vortex generators

Vortex generators (VGs) are devices that can be installed on the surfaces of aircraft, motor vehicles, rotors or turbines, with the aim of influencing their aerodynamic characteristics. Typically VGs consist of thin sheets with a straight or curved shape. When the surface on which they are mounted is hit by a flow, the VGs divert part of the flow in order to create vortex. The purpose is to cause a remixing of the boundary layer, which allows the exchange of momentum within it, and which allows the lower layers to be energized. In the purely aeronautical field, the VGs are typically installed on the wings, and more precisely between 2% and 15% of the mean aerodynamic chord. In this way it is possible to delay the separation of the boundary layer, reducing the stall speed and increasing the effectiveness of the control surfaces. The negative effect is to increase drag, with a consequent increase in consumption and a decrease in the maximum cruising speed. According to some studies conducted by the Lufthansa flight company, in collaboration with the German Aerospace Center, the use of VG can help reduce the noise generated by Airbus A320 aircraft by about 2 dB, eliminating annoying frequencies. In this case, the VGs should be installed on the lower part of the wing, to reduce the noise generated, during landing, by the airflow on the fuel tank vents.

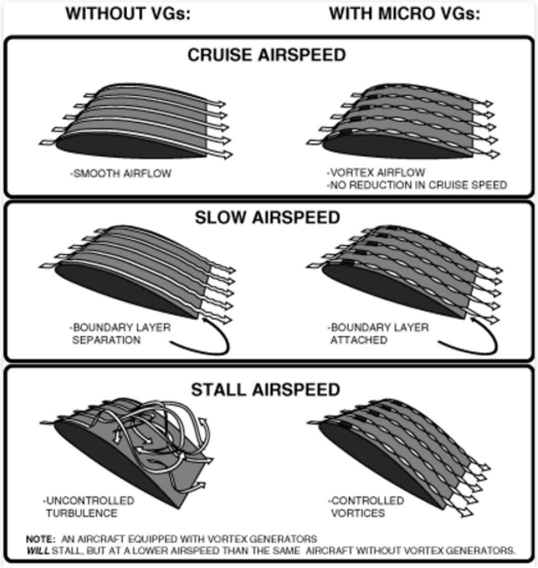

WITHOUT VGs:
WITH MICRO VGs:
CRUISE AIRSPEED
-SMOOTH AIRFLOW
-VORTEX AIRFLOW
-NO REDUCTION IN CRUISE SPEED
SLOW AIRSPEED
-BOUNDARY LAYER SEPARATION
-BOUNDARY LAYER ATTACHED
STALL AIRSPEED
-UNCONTROLLED TURBULENCE
-CONTROLLED VORTICES
NOTE: AN AIRCRAFT EQUIPPED WITH VORTEX GENERATORS WILL STALL, BUT AT A LOWER AIRSPEED THAN THE SAME AIRCRAFT WITHOUT VORTEX GENERATORS.

Figure 5.3: Vortex generators on BS Prime wing

5.2 Climb performance

In general terms climb performance is directly linked to excess thrust/power, i.e. the differenece between thrust/power available and thrust/power required. Whereas thrust/power available depends on the propulsion system and the environment (density, temperature), thrust/power required depends on weight (induced drag), configuration (parasitic drag) and airspeed (total drag). In order to determine flight performance data all aspects which have impact on the excess thrust/power must be taken into consideration. Two important climb performance parameters are: the Maximum Angle of Climb for obstacle clearance, the Maximim Rate of Climb for maximum altitide gain per time, and their associated air speeds V_x and V_y . Excess power is a fundamental aspect, as it is necessary for climbing and acceleration. Applying excess power increases the aircraft potential energy or its kinetic energy or both. Understanding the relationship between climb and acceleration allows mathematical analysis of data. As it can be seen from equation 5.1, the excess power ($\Delta\Pi_{excess}$) is the change of potential and kinetic energy with respect to time:

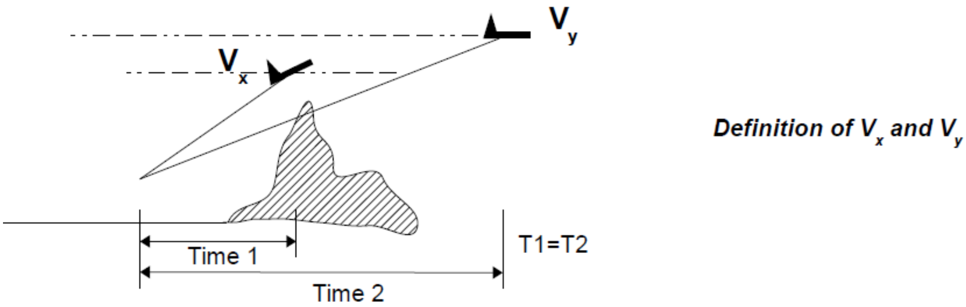


Figure 5.4: Vx and Vy definitions

$$\Delta\Pi_{excess} = \frac{\Delta E}{\Delta t} + \frac{\Delta K}{\Delta t} \tag{5.1}$$

It is possible to rewrite 5.1 in a differential expression:

$$d\Pi_{excess} = \frac{dE}{dt} + \frac{dK}{dt} \tag{5.2}$$

Where E is potential energy in J, and K is kinetic energy in J.
The potential energy can be defined as:

$$E = m \cdot g \cdot H = W \cdot H \tag{5.3}$$

Where W is the weight in N, m is the mass in kg, g is the gravitational acceleration in m/s^2 and H is the altitude in m. Whereas the kinetic energy is defined as:

$$K = \frac{1}{2} \cdot m \cdot V^2 \tag{5.4}$$

Where V is the speed in m/s. By using the definition of weight as $W = m \cdot g$, it is possible to rewrite the 5.5 using 5.3 and 5.4 as

$$d\Pi_{excess} = \frac{d}{dt}(W \cdot H + \frac{1}{2} \cdot W/g \cdot V^2) \tag{5.5}$$

Now, using the properties of differentials to 5.5, it is possible to obtain:

$$d\Pi_{excess} = H\frac{dW}{dt} + W\frac{dH}{dt} + \frac{V^2}{2g}\frac{dW}{dt} + \frac{W \cdot V}{g}\frac{dV}{dt} \tag{5.6}$$

If the relative variation of weight dW/dt is compared to the toltal weight of the aircraft, it is possible to notice that it is very small during climb, so it is possible to assume that $dW/dt \sim 0$. Therefore the equation 5.6 can be rewritten as:

$$d\Pi_{excess} = W\frac{dH}{dt} + \frac{W \cdot V}{g}\frac{dV}{dt} \tag{5.7}$$

Where it is possible to define the new dH/dt quantity as the rate of climb or ROC in m/s. The quantity dV/dt is the inertial acceleration of the aircraft. Therefore it is important to underline that V is the true air speed (TAS) of the aircraft in m/s. The excess power can either cause an acceleration or an altitude increment or both. During the aircraft climb at maximum permissable power it is possible to assume that the true air speed is constant: this means that $dV/dt = 0$, so the excess power becomes:

$$d\Pi_{excess} = W\frac{dH}{dt} \tag{5.8}$$

This means that all the excess power is only used to increase the altitude, so the $d\Pi_{excess}$ is converted into potential energy.

During the aircraft level acceleration at maximum permissable power, it is possible to assume that $dH/dt = 0$, so the excess power becomes:

$$d\Pi_{excess} = \frac{W \cdot V}{g}\frac{dV}{dt} \tag{5.9}$$

This means that all the excess power is only used for acceleration, so the $d\Pi_{excess}$ is converted into kinetic energy. Thus for small incremets of altitude or speed per small increment of time, where the excess power does not change due to engine power or propeller characteristics or drag or weight, it is possible to obtain:

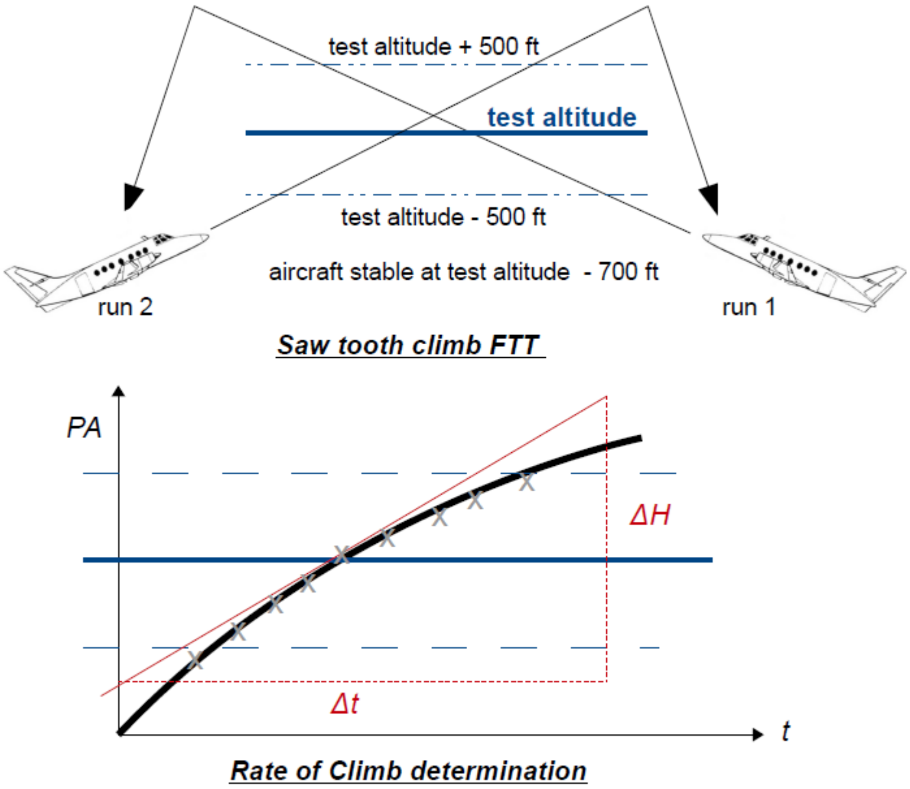
$$\frac{dH}{dt} = \frac{V}{g}\frac{dV}{dt} \tag{5.10}$$

5.3 Climb performance thest method: Saw Tooth Climb and Level Acceleration

As already noted, to perform tests on climb performance it was decided to use the **Level Acceleration method**, as it allows a lower fuel consumption, less weight variation and therefore more economy.

5.3.1 Saw Tooth Climb

For most general aviation piston engine aircraft the Saw Tooth Climb method establish a climb of about 1000 ft: once a target altitude is set, it starts from 500 ft below and ends at 500 ft higher. The aircaft should be in a stable climb with max power setting at the targeted test air speed about 200 ft below the test band. Test starts when reaching the lower limit of the test band. The time required to climb Δ100 ft is recorded. The data are plotted in an Altitude vs. Time graph. It is not expected to have a perfectly linear correlation between ΔH and Δt since the engine performance will usually decrease with altitude, thus the power available will decrease. In order to get the rate of climb for the target altitude, a tangent is drawn at the target altitude. The slope of the tangent $\Delta H/\Delta t$ provides the Rate of Climb at target altitude and target air speed.



Saw tooth climb FTT

Rate of Climb determination

Figure 5.5: Saw Tooth Climb

5.3.2 Level Acceleration

During a saw tooth climb a few things could change such as the weight, due to fuel consuption, and the outside temperature. It is possible to consider an alternative and economically more efficient test method. Recall the equation 5.10 about the correlation between the Rate of Climb and acceleration at constant altitude, which is a Level Acceleration. Thus, by measuring the air speed and the corresponding acceleration for an incremental period of time dV/dt at a constant altitude, it can be analytically obtained the Rate of Climb for various airspeeds at the tested altitude. To do so the aircraft is stabilized in a climb at max power and with sufficient margin near to stall speed. When reaching the target altitude the aircraft is levelled of by pushing the nose down and the excess power will now accelerate the aircraft. Data to be taken are the airspeed about every 5 kts of acceleration and the corresponding time. Since the analytics of climb and acceleration is based on the physics of energy, work and power, all data must be taken as true data (TAS, tapeline altitude i.e. height).

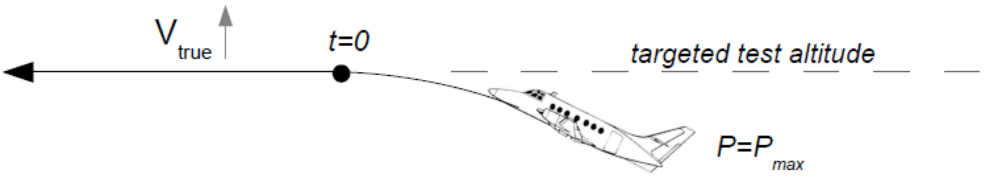


Figure 5.6: Level acceleration

5.4 Experimental evaluation of neutral point

In trim conditions, one of the fundamental quantities is the deflection of the elevator. This measurement is complex to obtain, as it is not simply measurable with the goniometer. One way it is to exploit the direct proportionality between the deflection of the elevator and the displacement of the stick:

$$\frac{d\delta_{trim}}{dC_{L,trim}} \propto \frac{ds_{trim}}{dC_{L,trim}} \tag{5.11}$$

Where s is the displacement of the stick. Typical testing methodology is to find a speed at which the aircraft can easily be trimmed and record the position of the stick as the starting point. After that, keeping the altitude constant, the speed is varied by ±5 knots and then by ± 10 knots, recording the position of the stick each time. It can be measured with respect to any point of the cabin, after having equipped oneself with a precise measuring instrument. Then a plot of the stick displacement s is made as the C_L varies. Where C_L is obtained from the TAS (V) and weight, assuming $L = W$ for constant altitue:

$$C_L = \frac{W}{1/2 \cdot \rho \cdot V^2 \cdot S} \tag{5.12}$$

Where V as always is the true air speed (TAS) and ρ is the density at the altitude where the test takes place.

This procedure is repeated for several positions of the center of gravity (at least three), as in Figure 5.7.

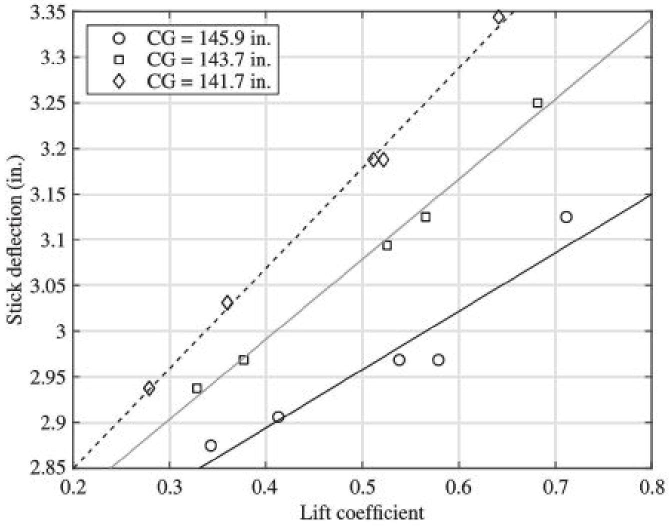


Figure 5.7: $ds_{trim}/dC_{L,trim}$

Once the curves are obtained, the slope for each line is calculated. These slopes are plotted as the position of the center of gravity varies. The resulting line intersects the abscissa axis at the position of the neutral point, as in Figure 5.8.

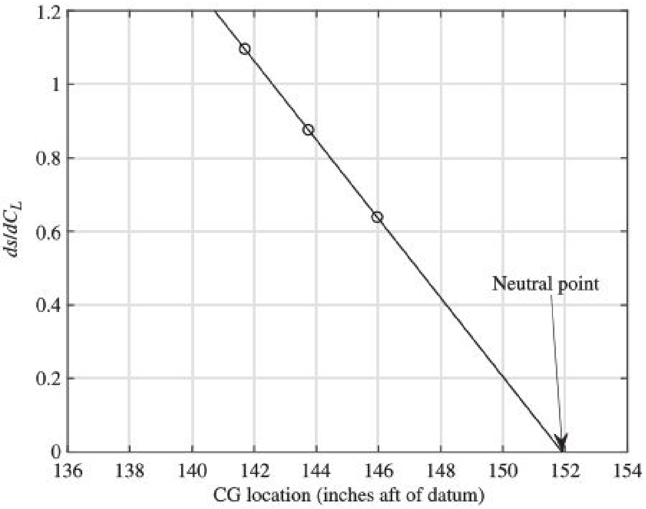


Figure 5.8: $dC_{L,trim}$

5.5 Stability test method

5.5.1 Test method for Phugoid

To test the stability of the phugoid mode [9, p. 42], it is necessary to start from a stable trim condition, and then a pitch command is given to pitch up or down. The speed must vary by about 10/15 knots. Once this speed perturbation is reached, the stick is returned to neutral position and the aircraft is allowed to move freely. The data can be collected manually or digitally, using for example the on-board computer. The data to be carefully analyzed are airspeed and pitch angle. By plotting these quantities over time it is possible to obtain the characteristic quantities. First of all, from the graphs it is necessary to identify the peaks corresponding to the time frame in which the test took place. From them, as Figure 5.9, it is possible first to determine the period as:

$$T_p = t_2 - t_1 \tag{5.13}$$

Once the period has been defined, the frequency of the dumped system is identified as:

$$\omega_d = \frac{2 \cdot \pi}{T_p} \tag{5.14}$$

Since the phugoid is a slow mode and develops over long periods, it is possible to detect the damping using the transient peak ratio (TPR) method [9, p. 42].

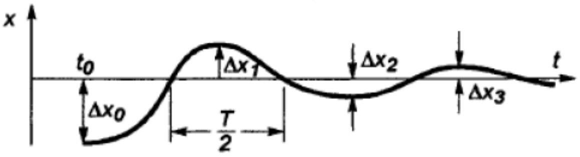


Figure 5.9: Period T_p and TPR

It is calculated using the following formula:

$$TPR = \frac{\Delta X_1}{\Delta X_0} = \frac{\Delta X_2}{\Delta X_1} = ... \tag{5.15}$$

This formula is applied for all peaks that can be identified, after which an average value is taken. The dumping is definde as:

$$\zeta = \frac{1}{\sqrt{1 + \left(\frac{\pi}{\log(TPR)}\right)^2}} \tag{5.16}$$

So from the dumping ratio it is possible to define the natural frequency, as

$$\omega_n = \frac{\omega_d}{1 - \zeta_p^2} \tag{5.17}$$

5.5.2 Test method for Short Period

Similarly to the phugoid mode, the test method for Short Period starts from a trim condition [9, p. 23], then a doublet type elevator command is given. Pulse width and pulse duration may vary depending on aircraft type. In this case, to identify the characteristic quantities, it is necessary to plot α over time.

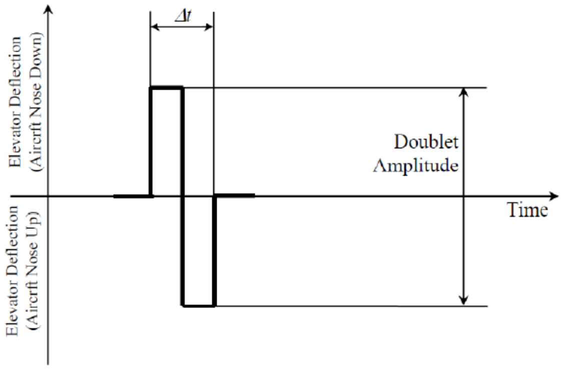

Δt
Elevator Deflection (Aircrft Nose Down)
Elevator Deflection (Aircrft Nose Up)
Doublet Amplitude
Time

Figure 5.10: Elevator doublet

5.5.3 Test method for Dutch Roll

Similarly to what was done for short period and phugoid, the test method for Dutch Roll starts from a stable trim condition [9, p. 24]. A rudder-frequency-sweep is then performed. It consists of moving the rudder to trigger the oscillation of the Dutch roll. Also in this case the characteristics are obtained starting from the flight data plots. In this case the damped system frequency is defined as:

$$\omega_d = \frac{2\pi \cdot n^\circ_{cycles}}{t_{total}} \tag{5.18}$$

Where n°_{cycles} is the number of cycles and t_{total} is the total time. Therefore using the TPR method and the equation 5.16 it is possible to calculate the damping, and then after with the 5.17 it is possible to obtain the natural frequency.

5.5.4 Test method for Roll Mode

The methodology in this case consists in stabilizing the aircraft with a bank angle of 30 or 60 degrees, subsequently an aileron input is applied and maintained until the roll rate stabilizes [9, p. 25]. When the roll-rate has stabilized, the controls are returned and held in their initial position.

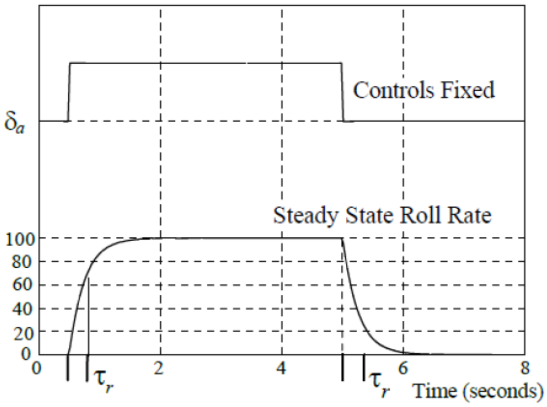


Figure 5.11: Roll Mode

From the flight data it is possible to obtain the characteristic time τ, defined as the time needed to reach 63 percent of the stationary value, as shown in the Figure 5.11.

5.6 Data standardization for Level Acceleration

In the following, the general steps that must be carried out to standardize the data obtained from the flight tests are set out and described, following the reference [15]. It was decided to report the description only for those procedures that are used for level acceleration.

5.6.1 Weight correction

The performance in climb necessarily depends on the weight of the aircraft. Performance worsens as weight increases. For this reason it is advisable to put oneself in a conservative condition, that is to measure the performance in the worst possible conditions. This means taking the MTOW as the standard weight. So if the actual weight during testing is different from the MTOW, it is needed to make a correction to the ROC. In general terms the lifting power is defined as

$$\Pi = \frac{\Delta E}{\Delta t} \tag{5.19}$$

In test condition, lifting power is

$$\Pi = \frac{m_{test} \cdot g \cdot \Delta H_{test}}{\Delta t} \tag{5.20}$$

In standard condition, lifting power is

$$\Pi = \frac{m_{stand} \cdot g \cdot \Delta H_{stand}}{\Delta t} \tag{5.21}$$

The power provided by the propulsion system remains constant, independent of weight, so $\Pi_{test} = \Pi_{standard}$, this leads to:

$$\frac{m_{test} \cdot g \cdot \Delta H_{test}}{\Delta t} = \frac{m_{stand} \cdot g \cdot \Delta H_{test}}{\Delta t} \tag{5.22}$$

By rearranging it is possible to obtain:

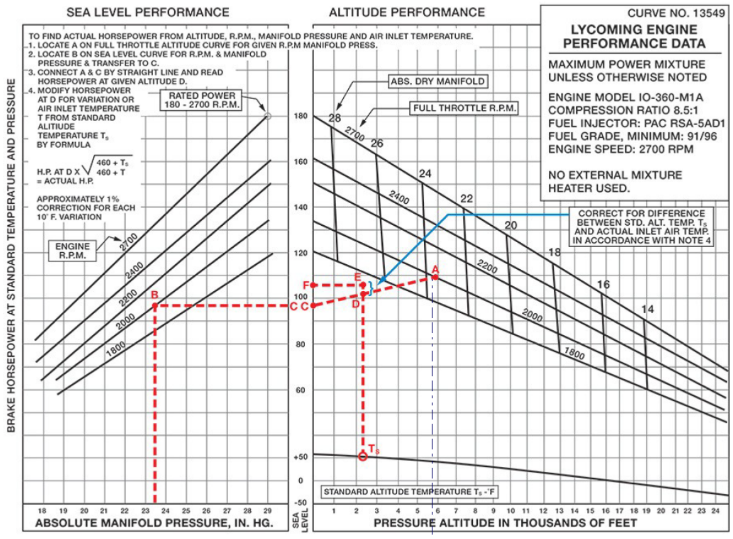
$$\left(\frac{\Delta H}{\Delta t}\right)_{stand} = \left(\frac{\Delta H}{\Delta t}\right)_{test} \cdot \frac{m_{test}}{m_{stand}} \tag{5.23}$$

5.6.2 Correction for Non-Standard-Engine-Power

As already observed in the thesis "Development of a mathematical model for Bs Prime" the power generated by the engine varies with the altitude, therefore with the variation of temperature and density. High temperatures and high altitudes inevitably reduce engine performance. Of course it is very likely that the temperature and pressure at the test altitude differ from those in the standard conditions, which determines the need to introduce corrections on the power. In order to determine the standard engine power, power charts provided by manufacturers are needed. The precise determination of standard engine power for a piston aircraft however has limitations which must be understood in order to properly assess the magnitude of possible error:

1. The variable mixture must be set for best power
2. Engine power charts are for dry air only. If the air is humid, power will decrease.
3. The engine configuration must match that of the engine model listed on the chart. Any changes in compression ratio, ignition system, cooling system or fuel delivery system may affect the power produced.

4. The engine must be in good condition. An engine with low compression, leaky valves, weak ignition system, etc. won't generate the power predicted by the chart.
5. The quality of fuel may affect engine power.



(L)IO-360-M1A Power Chart *(source: Lycoming Operation and Installation Manual)*

Figure 5.12: Example of engine power charts

Note that the Figure 5.12 is only an example, it is not the actual power charts of the Bs Prime engine, because the manufacturer has not provided it. It should be remembered that the Bs Prime is characterized by a 4-cylinder Rotax 912 ULS.

As previously shown there is a direct correlation between power and the altitude increment by 5.21, after solving for the climb rate it is possible to obtain that:

$$\Delta\left(\frac{dH}{dt}\right)_{correction} = \frac{\Delta\Pi_{excess}}{W_{test}} \tag{5.24}$$

- If a manufacturer provided precise engine power chart is not available, a "thumb" approximation for Non-Standard- Engine-Power can be made by the following relation:

$$\frac{\Pi_{stand}}{\Pi_{test}} = \frac{\sqrt{\sigma_{stand}}}{\sqrt{\sigma_{test}}} \tag{5.25}$$

  Where σ is the density ratio, calculated by pressure ratio devided by temperature ratio.

- If the pressure altitude remains constant from test day to test day and the temperature varies, the Non-Standard-Engine-Power may be approximated by:

$$\frac{\Pi_{stand}}{\Pi_{test}} = \sqrt{\frac{T_{stand}}{T_{test}}} \tag{5.26}$$

So the excess power for correction in equation 5.24 is given by:

$$\Delta\Pi_{excess} = \Pi_{stand} - \Pi_{test} \tag{5.27}$$

These correction methods require at least a very basic engine power table for sea level conditions in order to determine the standard power at sea level pressure at standard temperature.

5.6.3 Correction for Induced Drag

Depending on whether the aircraft is heavier or lighter, a smooth straight-line flight condition may require a higher or lower α condition, so that lift continues to balance weight. This change in incidence leads to an induced change in resistance and consequently a change in performance. In a nutshell, weight affects climbing performance not only in terms of inertia but also in terms of induced drag. Consequently, excess power is also affected. This difference in excess power $\Delta\Pi_{excess}$ will either increase or decrease the climb performance and must be analytically considered in flight test. Starting from the definition of total drag of a subsonic aircraft:

$$D = D_p + D_i \tag{5.28}$$

Where D_p is the parasitic drag, whose drag coefficient C_{D_0} does not cange with weight. The focus must be on induced drag, defined as:

$$D_i = q \cdot S \cdot C_{D_i} \tag{5.29}$$

Where q is the dynamic pressure $q = 1/2\rho V^2$, C_{D_i} is the induced drag coefficient and S is the wing surface in m^2. So the variation of induced drag can be written as:

$$D_i = (q \cdot S \cdot C_{D_i})_{test} - (q \cdot S \cdot C_{D_i})_{stand} \tag{5.30}$$

The induced drag coefficient C_{D_i} can be expressed as a function of the lift coefficient C_L:

$$C_{D_i} = \frac{C_L^2}{\pi \cdot AR \cdot e} \tag{5.31}$$

Where AR is the aspect ratio and e is Oswald efficiency. The equation 5.30 can be rewritten by using 5.31 as:

$$D_i = \left[\left(\frac{C_L^2}{\pi \cdot AR \cdot e}\right)_{test} - \left(\frac{C_L^2}{\pi \cdot AR \cdot e}\right)_{stand}\right] qS \tag{5.32}$$

The lift in flight depends on weight and air speed. To analyze the influence on the lift coefficient due to the weight variation, for the level flight condition, characteristic of the level acceleration, the relation could be considered $L = W$. So from the lift definition it is possible to obtain:

$$C_{L_{test}} = \frac{W_{test}}{qS} \quad and \quad C_{L_{stand}} = \frac{W_{stand}}{qS} \tag{5.33}$$

Inserting these elements in the 5.32 and by reworking the terms, it can be obtained:

$$\Delta D_i = (W_{test} - W_{stand})\frac{1}{qS\pi AR\ e} \tag{5.34}$$

The delta in induced drag (ΔD_i) is generating a difference in power. In general, power is defined as: $\Pi = F \cdot V$. Where the force is the delta drag, the velocity is the true air speed thus:

$$\Delta\Pi = \Delta D_i \cdot TAS = \Delta\Pi_{excess} \tag{5.35}$$

The $\Delta\Pi$ is the difference in excess power, which can or cannot be used for mechanical lifting of the standard weight aircraft. Using equation 5.21, 5.34 and 5.35, it is possible to obtain:

$$\Delta\left(\frac{dH}{dt}\right) = \frac{W_{test}^2 - W_{stand}^2}{W_{stand}} \cdot \frac{1}{1/2 \cdot \rho_{test} \cdot TAS \cdot S \cdot \pi \cdot AR \cdot e} \tag{5.36}$$

5.6.4 Correction for temperature and for climbing/descending

As for the temperature correction, it must be taken into account that the tests are not carried out under standard conditions. So it is necessary to make the following correction:

$$\frac{dH}{dt} = \left(\frac{dH}{dt}\right)_{test} \cdot \frac{T_{test}}{T_{stand}} \tag{5.37}$$

It must also be taken into account the correction for climbing/descending while accelerating in order to compensate the exchange of potential and kinetic energy.

$$\Delta\left(\frac{dH}{dt}\right) = \frac{(H_{final} - H_{initial})}{\Delta t_{acceleration}} \cdot \frac{T_{test}}{T_{stand}} \tag{5.38}$$

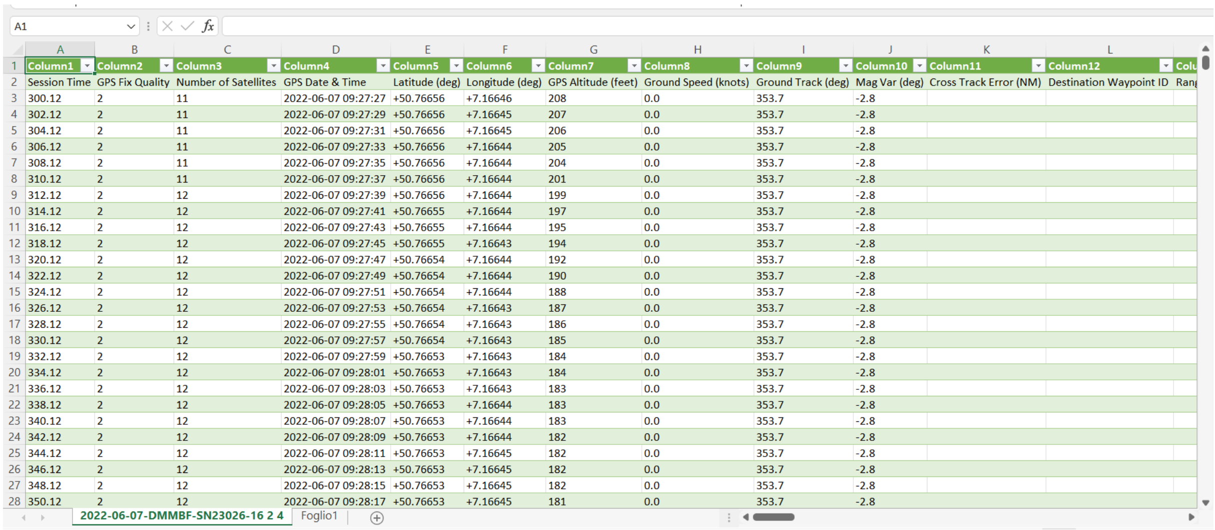
Where, therefore, the correction takes place only by taking into consideration the altidude at which the acceleration $H_{initial}$ is started and the altitude in correspondence of the time in which the acceleration ended H_{final} . $\Delta t_{acceleration}$ corresponds to the time interval in which the acceleration occurred.

Chapter 6

Flight test data analysis

6.1 Introduction

As already mentioned, the tests focused on performance in climb conditions, and in particular the cruise acceleration method was used. This method requires the pilot to trim the aircraft for a low speed value, while maintaining a margin with respect to the stall speed. Then an acceleration is carried out at a constant altitude, by setting the throttle to the maximum, for a period of time that can vary from 60 to 90 seconds. The on-board computer stores all the data with a frequency of 16 Hz, and makes them available in a .csv file.



A1 fx

	A	B	C	D	E	F	G	H	I	J	K	L	
1	Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9	Column10	Column11	Column12	Colu
2	Session Time	GPS Fix Quality	Number of Satellites	GPS Date & Time	Latitude (deg)	Longitude (deg)	GPS Altitude (feet)	Ground Speed (knots)	Ground Track (deg)	Mag Var (deg)	Cross Track Error (NM)	Destination Waypoint ID	Ran
3	300.12	2	11	2022-06-07 09:27:27	+50.76656	+7.16646	208	0.0	353.7	-2.8			
4	302.12	2	11	2022-06-07 09:27:29	+50.76656	+7.16645	207	0.0	353.7	-2.8			
5	304.12	2	11	2022-06-07 09:27:31	+50.76656	+7.16645	206	0.0	353.7	-2.8			
6	306.12	2	11	2022-06-07 09:27:33	+50.76656	+7.16644	205	0.0	353.7	-2.8			
7	308.12	2	11	2022-06-07 09:27:35	+50.76656	+7.16644	204	0.0	353.7	-2.8			
8	310.12	2	11	2022-06-07 09:27:37	+50.76656	+7.16644	201	0.0	353.7	-2.8			
9	312.12	2	12	2022-06-07 09:27:39	+50.76656	+7.16644	199	0.0	353.7	-2.8			
10	314.12	2	12	2022-06-07 09:27:41	+50.76655	+7.16644	197	0.0	353.7	-2.8			
11	316.12	2	12	2022-06-07 09:27:43	+50.76655	+7.16644	195	0.0	353.7	-2.8			
12	318.12	2	12	2022-06-07 09:27:45	+50.76655	+7.16643	194	0.0	353.7	-2.8			
13	320.12	2	12	2022-06-07 09:27:47	+50.76654	+7.16644	192	0.0	353.7	-2.8			
14	322.12	2	12	2022-06-07 09:27:49	+50.76654	+7.16644	190	0.0	353.7	-2.8			
15	324.12	2	12	2022-06-07 09:27:51	+50.76654	+7.16644	188	0.0	353.7	-2.8			
16	326.12	2	12	2022-06-07 09:27:53	+50.76654	+7.16643	187	0.0	353.7	-2.8			
17	328.12	2	12	2022-06-07 09:27:55	+50.76654	+7.16643	186	0.0	353.7	-2.8			
18	330.12	2	12	2022-06-07 09:27:57	+50.76654	+7.16643	185	0.0	353.7	-2.8			
19	332.12	2	12	2022-06-07 09:27:59	+50.76653	+7.16643	184	0.0	353.7	-2.8			
20	334.12	2	12	2022-06-07 09:28:01	+50.76653	+7.16643	184	0.0	353.7	-2.8			
21	336.12	2	12	2022-06-07 09:28:03	+50.76653	+7.16643	183	0.0	353.7	-2.8			
22	338.12	2	12	2022-06-07 09:28:05	+50.76653	+7.16644	183	0.0	353.7	-2.8			
23	340.12	2	12	2022-06-07 09:28:07	+50.76653	+7.16644	183	0.0	353.7	-2.8			
24	342.12	2	12	2022-06-07 09:28:09	+50.76653	+7.16644	182	0.0	353.7	-2.8			
25	344.12	2	12	2022-06-07 09:28:11	+50.76653	+7.16645	182	0.0	353.7	-2.8			
26	346.12	2	12	2022-06-07 09:28:13	+50.76653	+7.16645	182	0.0	353.7	-2.8			
27	348.12	2	12	2022-06-07 09:28:15	+50.76653	+7.16645	182	0.0	353.7	-2.8			
28	350.12	2	12	2022-06-07 09:28:17	+50.76653	+7.16645	181	0.0	353.7	-2.8			

Figure 6.1: Example of flight test data.csv file

After keeping only the essential data, it is possible to proceed to process the data by switching to matlab.

6.2 How to evaluate the ROC from the fligfht test data

To calculate the ROC using flight test data, first of all the TAS values can be plotted in a graph as time changes. Then it is necessary to identify the time span in which the acceleration is carried out. Once this has been identified, only the time interval of interest is isolated. With the MATLAB "curve fitting" toolbox it is possible to perform an interpolation to obtain a polynomial function (in this case a second degree polynomial was chosen), and then the derivative dV/dt is

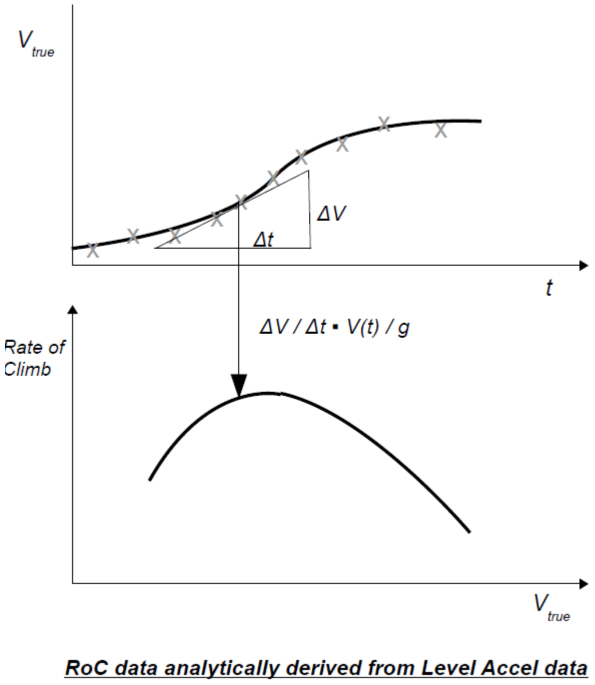


Figure 6.2: ROC from TAS graphic

calculated . Then numerically the equation already mentioned is implemented:

$$ROC = \frac{dH}{dt} = \frac{V}{g}\frac{dV}{dt}$$

6.3 Standardization and correction of data

In the previous chapter, the various techniques required for data standardization were discussed. Since the actual load condition can be defined in the model implemented in simulink, it is not necessary to correct the weight: $W_{stand} = W_{test}$. Consequently, the correction on the induced drag is also not carried out. Since the manufacturer has not provided the engine power chart it is not possible to make a correction using it. As for the power of the engine, the effects of the variation of the temperature and altitude are already contemplated in the model of the engine-propeller through the function ψ. Therefore the function thus obtained must be subjected to only two corrections: the first takes into account the fact that the temperature at which the tests were carried out is not the standard one; the second takes into account the fact that, during acceleration, the altitude does not remain perfectly constant. The final formula used to calculate the ROC is therefore the following:

$$ROC = \frac{dH}{dt} = \left(\frac{V}{g}\frac{dV}{dt} - \frac{\Delta H}{\Delta t}\right) \cdot \frac{T_{stand}}{T_{test}}$$

6.4 Data processing

During the two hours of flight it was possible to perform nine accelerations at different altitudes, in order to have an estimate of the performance in several flight conditions. For each of these accelerations it was possible to produce a comparison graph, plotting at the same time the ROC deriving from the flight data and the one calculated with the mathematical model. As mentioned, the aircraft stores all flight data in a .csv file, which can be opened and processed with excel. Since the data is stored with a frequency of 16 Hz, the file is large in size. The first step was therefore to isolate the columns and rows of our interest. For convenience, the columns were imported into MATLAB as vectors, making it easier for you to view and manipulate the graphs you need. By analyzing the speed and altitude trends over time, it was possible to precisely identify the time windows in which the accelerations occurred.

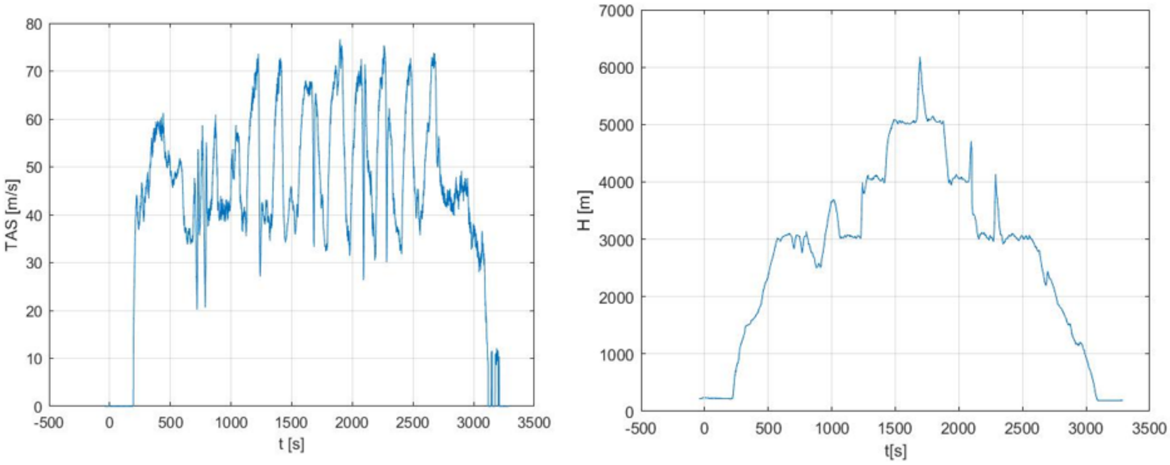


Figure 6.3: First test session

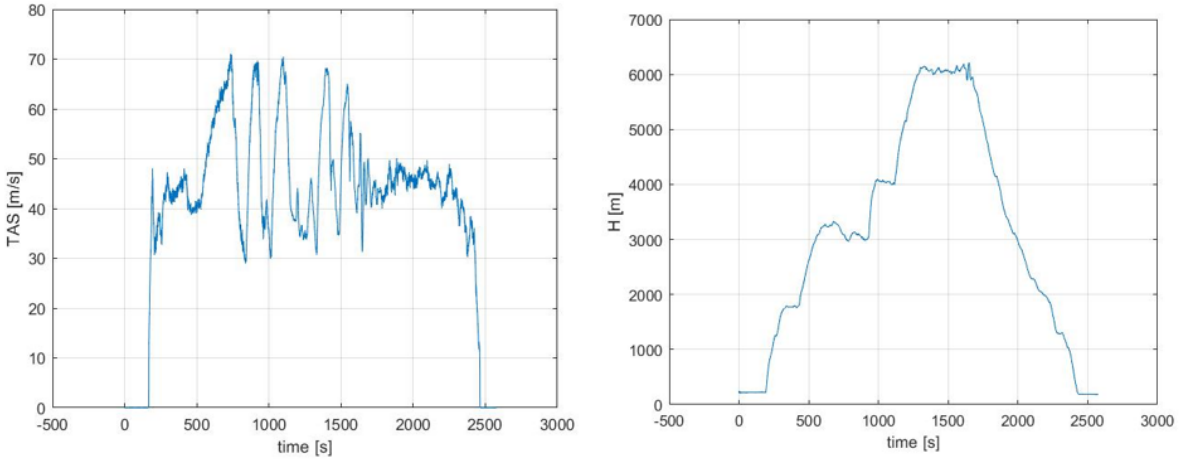


Figure 6.4: Second test session

To calculate the ROC with the mathematical model, however, a new routine based on ACTRIM was used. As input they are entered the altitude at which the test is carried out, the MAP value (which corresponds to the maximum available, in this case 28 "Hg), the speed range in the form of a vector, and the RPM. This last value was perhaps the most problematic. The values recorded in the flight data, in fact, were clearly incorrect: in some cases they exceeded the maximum value of the engine's maximum RPM, while in others they were too low to realistically think that the engine could work for those values. In order to enter meaningful values within the model it was therefore necessary to make an estimate. The value used was 5500 RPM, as in the flight manual [1]. In this case, and unlike what was done for the calculation of trim conditions in level flight, a flight path angle value equal to zero has not been set; doing so the calculated trim conditions will include a certain value of $\dot{H}$. This value corresponds to the ROC searched, and can therefore be compared with that deriving from the flight tests. Using the procedure described in the previous paragraphs, it was possible to generate nine comparison graphs.

6.4.1 Example of ROC calculation form flight test data

As mentioned, the first step is to identify the time interval in which the acceleration took place, using the graph referred to the TAS of the first or second flight test session, Figure 6.3 and Figure 6.4. Once the interval has been defined, it is possible to proceed to create the graph of the speed variation and of the altitude only for the defined interval.

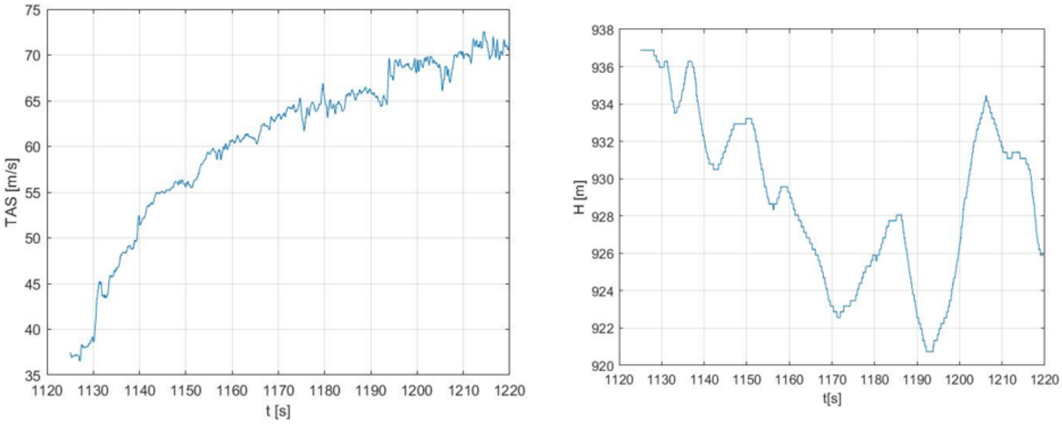
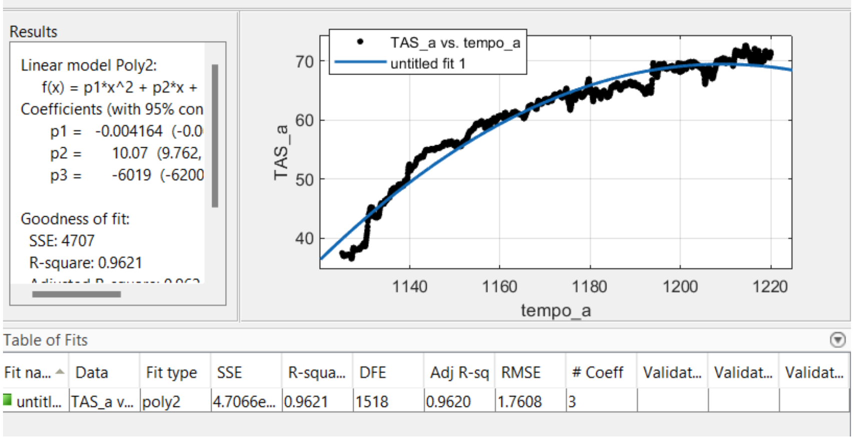


Figure 6.5: Acceleration

The second step is to interpolate the speed with a quadratic function, as it approximates the behavior well.



Results

Linear model Poly2:
f(x) = p1*x^2 + p2*x +
Coefficients (with 95% con
p1 = -0.004164 (-0.0
p2 = 10.07 (9.762,
p3 = -6019 (-6200

Goodness of fit:
SSE: 4707
R-square: 0.9621
Adjusted R-square: 0.962

Table of Fits

Fit na...	Data	Fit type	SSE	R-squa...	DFE	Adj R-sq	RMSE	# Coeff	Validat...	Validat...	Validat...
🟩 untitl...	TAS_a v...	poly2	4.7066e...	0.9621	1518	0.9620	1.7608	3			

Figure 6.6: TAS interpolation

So the speed function is:

$$V = -0.004164 \cdot t^2 + 10.07 \cdot t - 6019 \tag{6.1}$$

Acceleration a can now be identified as being derived from velocity:

$$a = \frac{d}{dt}V = -0.008328 \cdot t + 10.07 \tag{6.2}$$

Where t is the time. Now it is possible to proceed to the ROC calculations taking into account the two corrections to be made: one for the temperature and one for the variation of the altitude. The correction for the altitude variation as already mentioned is due to acceleration or deceleration, therefore a passage from potential energy to kinetic and vice versa. Since this is all about energy, the correction on the altitude concerns only the difference between the starting instant and the ending instant of acceleration. Where in the example, the change in altitude is 11 m in 100 s

$$ROC = \left(\frac{V}{9.81} \cdot a - \frac{11\ m}{100\ s}\right) \cdot \frac{T_{stand}}{T_{test}}$$

Where T_{stand} is a temperature vector at the altitude defined in ISA conditions, while T_{test} is a temperature vector actually recorded during the test.

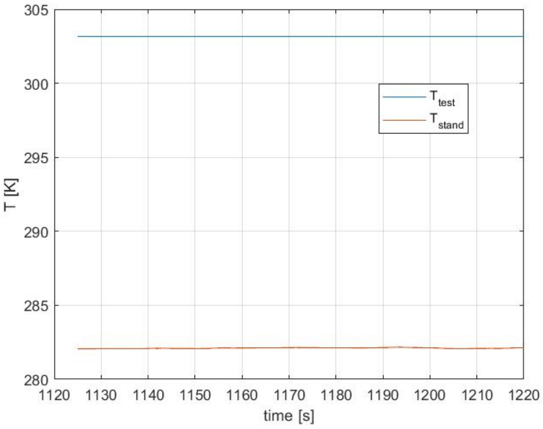


Figure 6.7: Test and standard temperature

So the ROC has the following pattern repeated in the Figure 6.8. Note that the ROC to which no corrections have been made and the standardized one have been plotted.

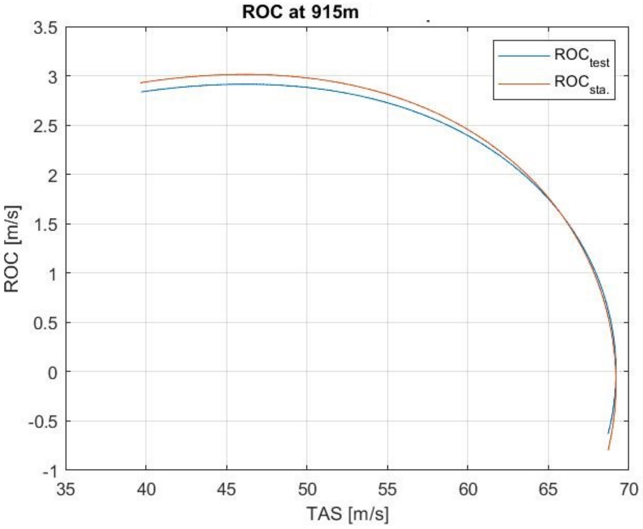


Figure 6.8: Test and standard temperature

6.4.2 Load condition during flight test

Unfortunately, the on-board computer did not record the fuel flow, so it was not possible to take in account the weight variation during the flight. Therefore, the load conditions at the take-off weight were used for all the calculations. It must be noted that therefore the standardization on weight could not have been applied in each test in the absence of the W_{test} in each phase. It should be noted that during these tests the pilot was seated in the rear seat and the passenger in the front seat.

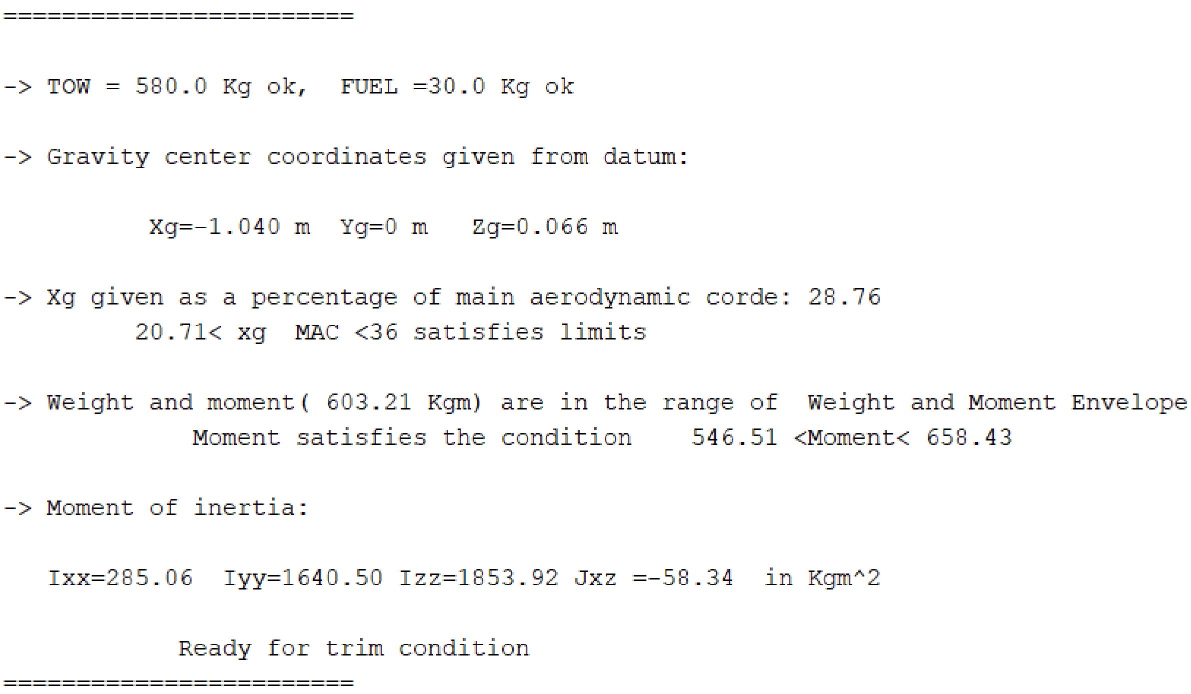
The load conditions for the two test sessions are different:

- Firt sension of flight test:

	Mass $[kg]$
passenger	70
pilot	90
baggage	0
fuel	30
empty weight	390
TOW	590

Table 6.1: Load condition: first session

Then using this conditions, in the appropriate function related to FDC, "primo_run _fdc.m", it is possible to obtain the following quantity in Figure 6.9.



```
========================


-> TOW = 580.0 Kg ok,   FUEL =30.0 Kg ok

-> Gravity center coordinates given from datum:

          Xg=-1.040 m  Yg=0 m   Zg=0.066 m

-> Xg given as a percentage of main aerodynamic corde: 28.76
         20.71< xg  MAC <36 satisfies limits

-> Weight and moment( 603.21 Kgm) are in the range of  Weight and Moment Envelope
             Moment satisfies the condition    546.51 <Moment< 658.43

-> Moment of inertia:

   Ixx=285.06  Iyy=1640.50 Izz=1853.92 Jxz =-58.34  in Kgm^2

            Ready for trim condition
========================
```

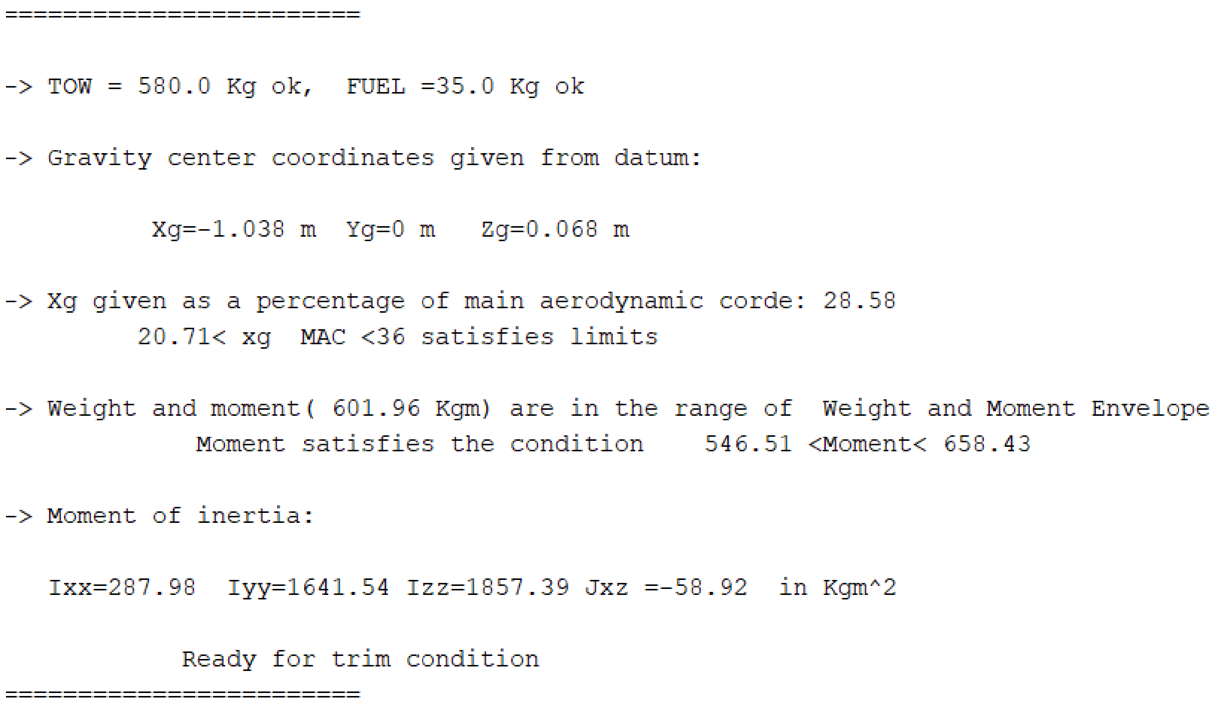
Figure 6.9: "primo_run _fdc.m" results for first session

- Second sension of flight test:

	Mass $[kg]$
passenger	65
pilot	90
baggage	0
fuel	35
empty weight	390
TOW	590

Table 6.2: Load condition: second session

Then using this conditions, in the appropriate function related to FDC, "first_run _fdc.m", it is possible to obtain the following quantity in Figure 6.10.



```
========================


-> TOW = 580.0 Kg ok,  FUEL =35.0 Kg ok

-> Gravity center coordinates given from datum:

          Xg=-1.038 m  Yg=0 m   Zg=0.068 m

-> Xg given as a percentage of main aerodynamic corde: 28.58
         20.71< xg  MAC <36 satisfies limits

-> Weight and moment( 601.96 Kgm) are in the range of  Weight and Moment Envelope
             Moment satisfies the condition    546.51 <Moment< 658.43

-> Moment of inertia:

   Ixx=287.98  Iyy=1641.54 Izz=1857.39 Jxz =-58.92  in Kgm^2

            Ready for trim condition
========================
```

Figure 6.10: "primo_run _fdc.m" results for second session

These values are used by FDC and in particular by the function CtrimforROC, to calculate the ROC of the mathematical model of the BS Prime. This quantity is shown in the following graphs in yellow with the name of "ROC_{fdc}". Remember that the position of the center of gravity x_g is the effective one obtained with the formula of the flight manual [1, p. 114]. The positions of y_g, z_g and the moment of inertia have been deducted by appropriate hypotheses and arguments set out in the thesis "Development of a mathematical model for Bs Prime"

6.4.3 Test point 3000 ft, first session, V_y

From the interpolation of the TAS graph it is obtained :

$$V = -0.004164t^2 + 10.07t - 6019$$

So the acceleration is:

$$a = dV/dt = -0.004164 \cdot 2 \cdot t + 10.07$$

The ROC is calculated using the 6.3, where the correction on the altitude corresponds to:

$$\frac{\Delta H}{\Delta t} = \frac{11\ m}{100\ s}$$

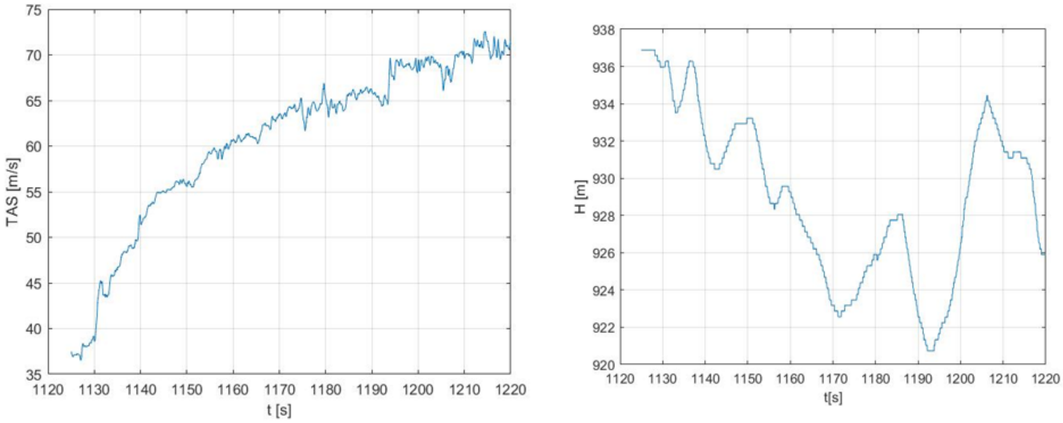


Figure 6.11: TAS and altitude at test point 3000 ft

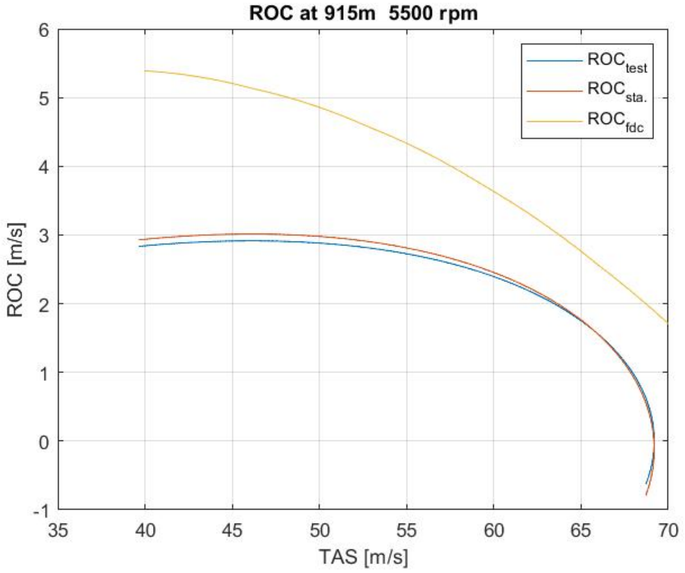


Figure 6.12: ROC at test point 3000 ft

6.4.4 Test point 4000 ft, first session, V_y

From the interpolation of the TAS graph it is obtained :

$$V = -0.007362t^2 + 20.66t - 1.442 \cdot 10^4$$

So the acceleration is:

$$a = dV/dt == -0.007362 \cdot 2 \cdot t + 20.66$$

The ROC is calculated using the 6.3, where the correction on the altitude corresponds to:

$$\frac{\Delta H}{\Delta t} = \frac{35\ m}{75\ s}$$

.

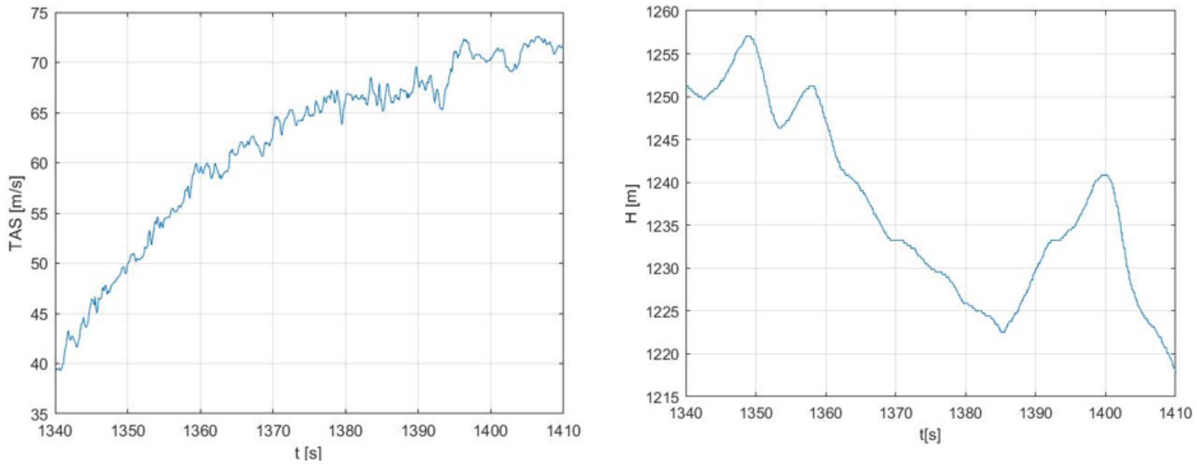


Figure 6.13: TAS and altitude at test point 4000 ft

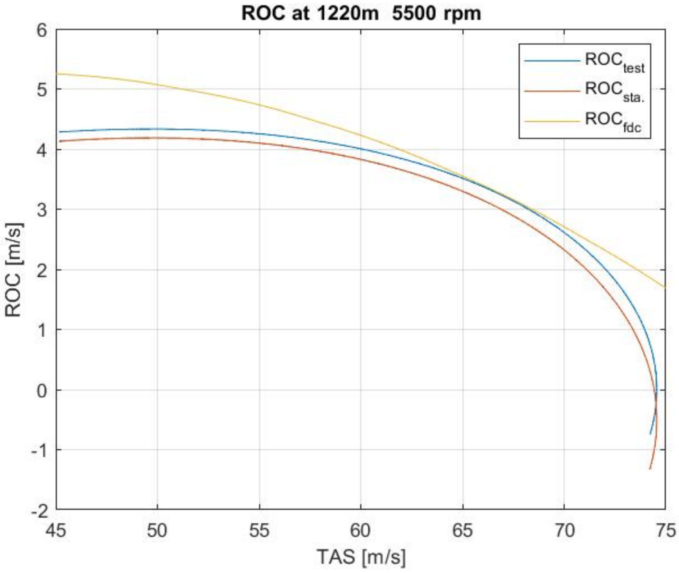


Figure 6.14: ROC at test point 4000 ft

6.4.5 Test point 5000 ft, first session, V_y

From the interpolation of the TAS graph it is obtained :

$$V = -0.006598t^2 + 21.29t - 1.711e \cdot 10^4$$

So the acceleration is:

$$a = dV/dt == -0.006598 \cdot 2 \cdot t + 21.29$$

The ROC is calculated using the 6.3, where the correction on the altitude corresponds to:

$$\frac{\Delta H}{\Delta t} = \frac{10\ m}{70\ s}$$

.

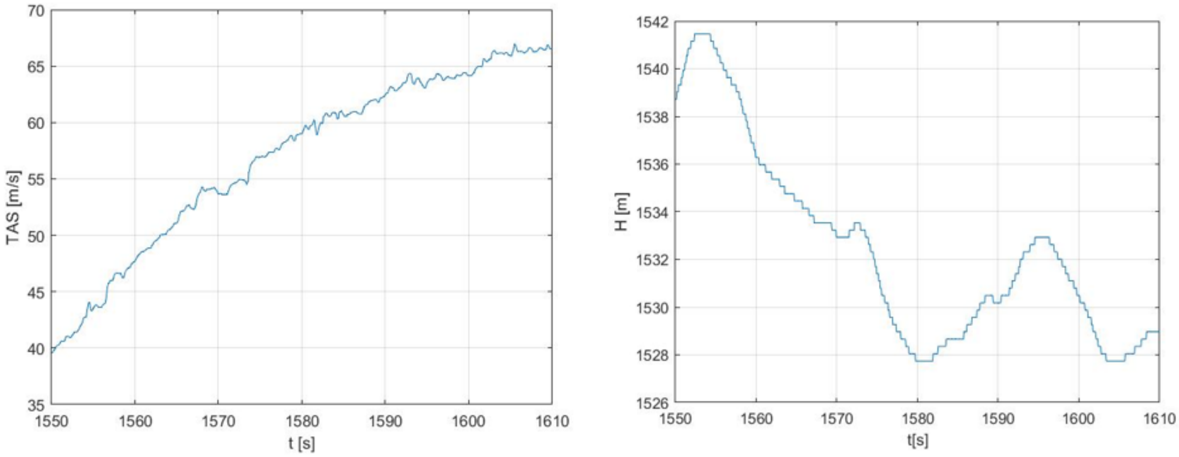


Figure 6.15: TAS and altitude at test point 5000 ft

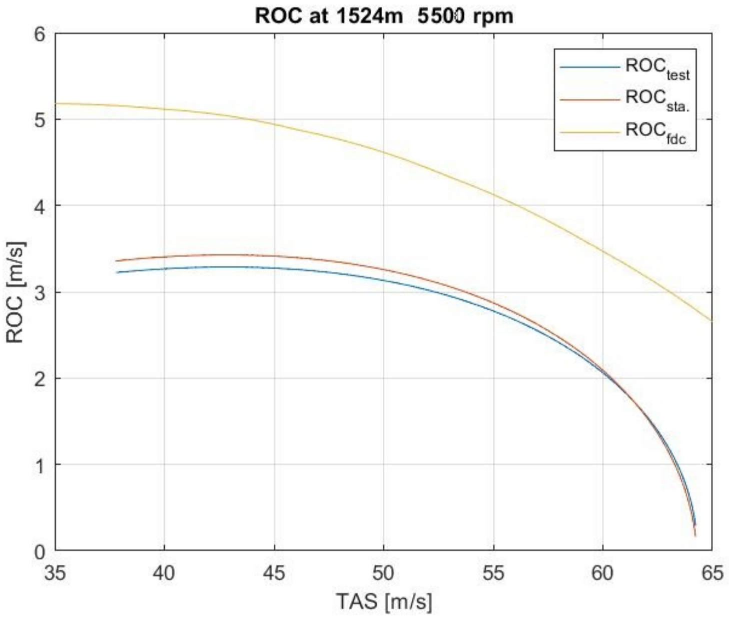


Figure 6.16: ROC at test point 5000 ft

6.4.6 Test point 5000 ft, first session, V_x

From the interpolation of the TAS graph it is obtained :

$$V = -0.009727t^2 + 35.99t - 3.323 \cdot 10^4$$

So the acceleration is:

$$a = dV/dt == -0.009727 \cdot 2 \cdot t + 35.99$$

The ROC is calculated using the 6.3, where the correction on the altitude corresponds to:

$$\frac{\Delta H}{\Delta t} = \frac{10\ m}{70\ s}$$

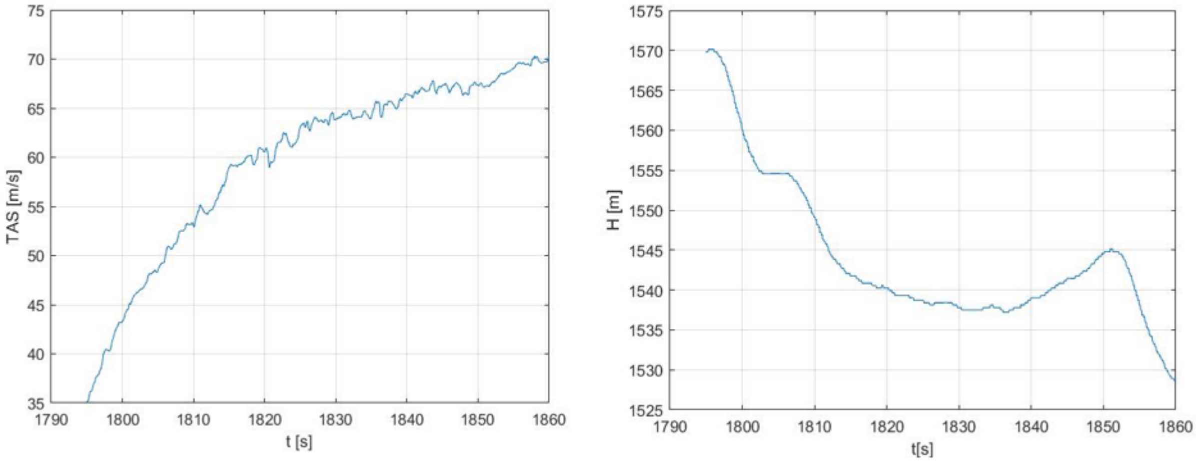


Figure 6.17: TAS and altitude at test point 35.995000 ft

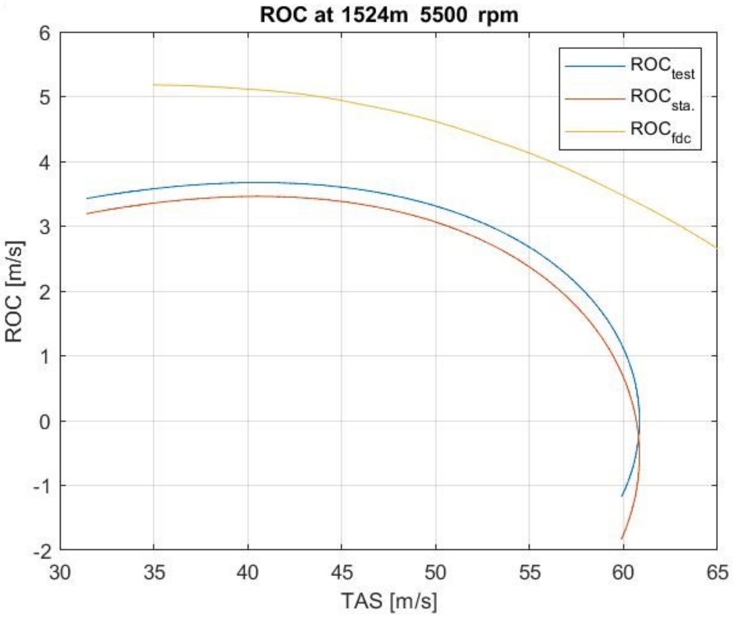


Figure 6.18: ROC at test point 5000 ft

6.4.7 Test point 4000 ft, first session, V_x

From the interpolation of the TAS graph it is obtained :

$$V = -0.006235t^2 + 25.64t - 2.628 \cdot 10^4$$

So the acceleration is:

$$a = dV/dt = -0.006235 \cdot 2 \cdot t + 25.64$$

The ROC is calculated using the 6.3, where the correction on the altitude corresponds to:

$$\frac{\Delta H}{\Delta t} = \frac{30\ m}{70\ s}$$

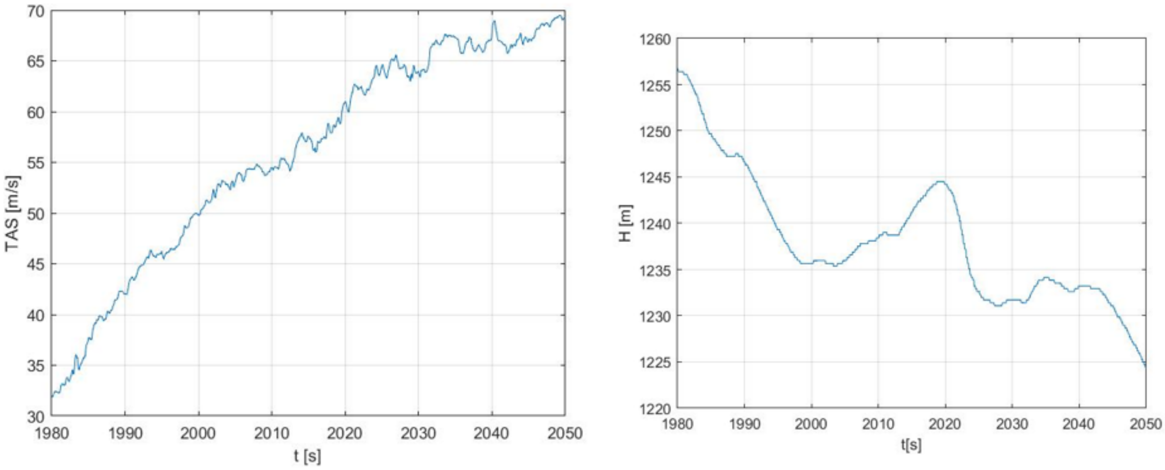


Figure 6.19: TAS and altitude at test point 5000 ft

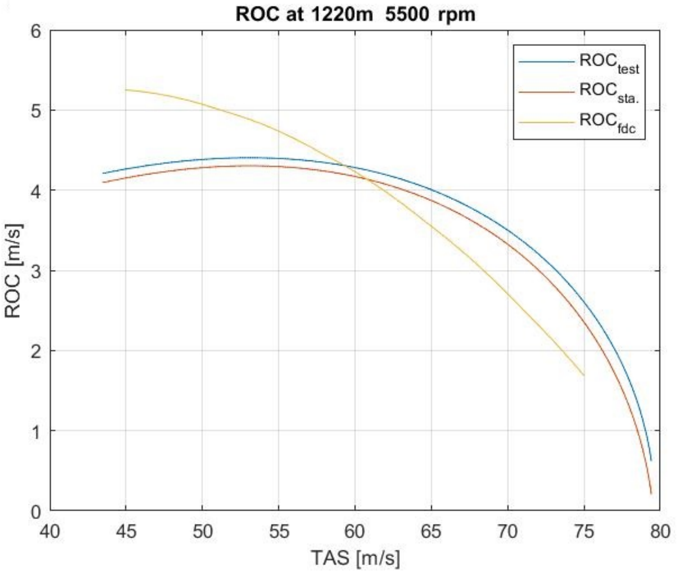


Figure 6.20: ROC at test point 4000 ft

6.4.8 Test point 3000 ft, first session, V_x

From the interpolation of the TAS graph it is obtained :

$$V = -0.007402t^2 + 36.63t - 4.525 \cdot 10^4$$

So the acceleration is:

$$a = dV/dt = -0.007402 \cdot 2 \cdot t + 36.63$$

The ROC is calculated using the 6.3, where the correction on the altitude corresponds to:

$$\frac{\Delta H}{\Delta t} = \frac{12\ m}{70\ s}$$

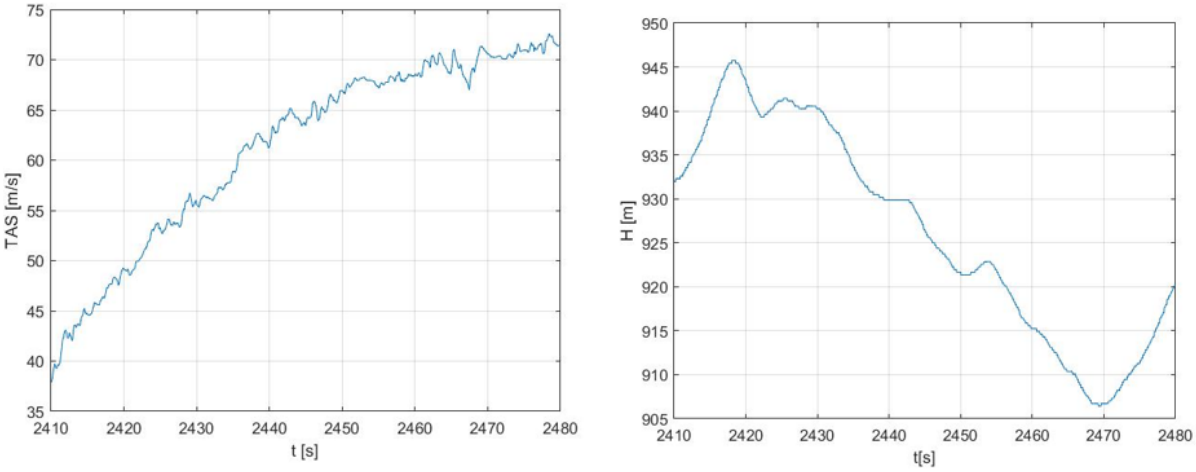


Figure 6.21: TAS and altitude at test point 3000 ft

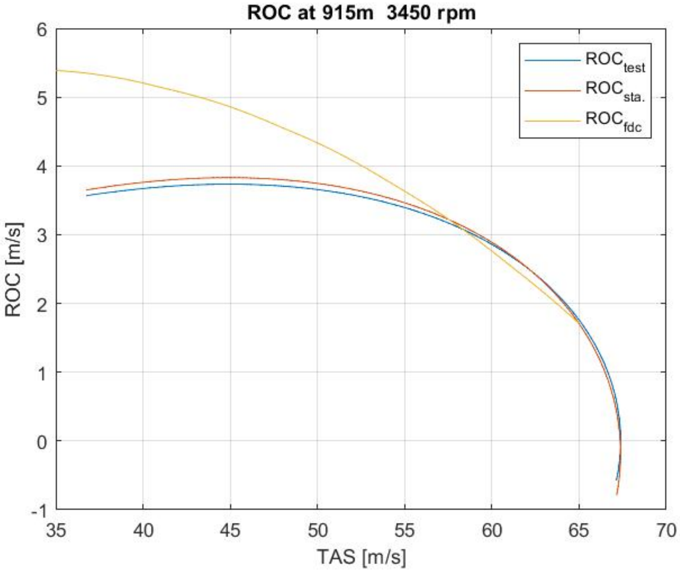


Figure 6.22: ROC at test point 3000 ft

6.4.9 Test point 3000 ft, second session

From the interpolation of the TAS graph it is obtained :

$$V = -0.007831t^2 + 14.31t - 6466$$

So the acceleration is:

$$a = dV/dt = -0.007831 \cdot 2 \cdot t + 14.31$$

The ROC is calculated using the 6.3, where the correction on the altitude corresponds to:

$$\frac{\Delta H}{\Delta t} = \frac{37\ m}{60\ s}$$

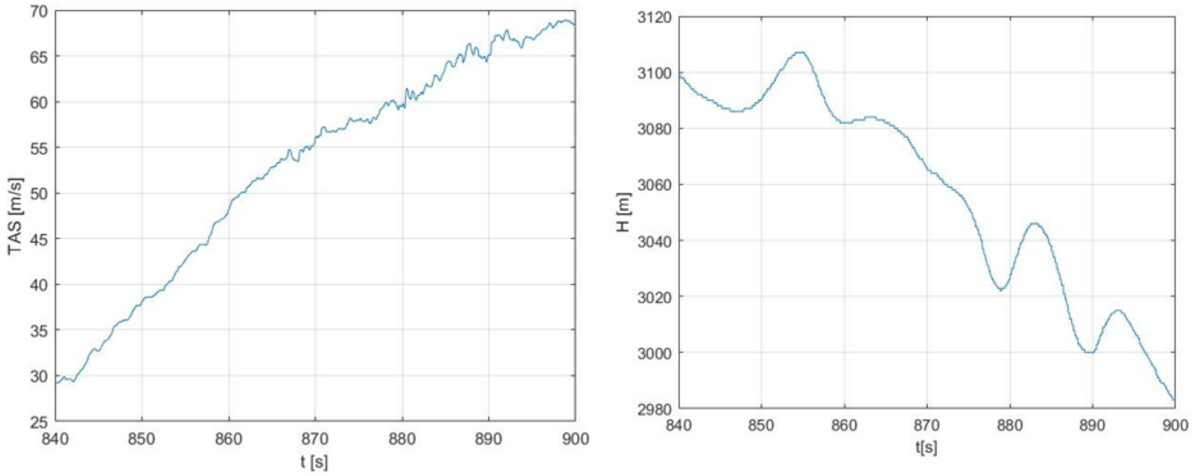


Figure 6.23: TAS and altitude at test point 3000 ft

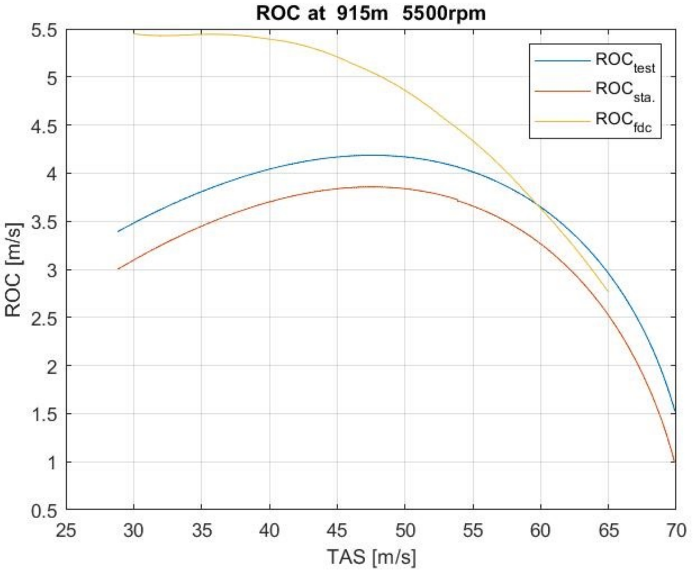


Figure 6.24: ROC at test point 3000 ft

6.4.10 Test point 4000 ft, second session

From the interpolation of the TAS graph it is obtained :

$$V = -0.004401t^2 + 9.738t - 5318$$

So the acceleration is:

$$a = dV/dt = -0.004401 \cdot 2 \cdot t + 9.738$$

The ROC is calculated using the 6.3, where the correction on the altitude corresponds to:

$$\frac{\Delta H}{\Delta t} = \frac{15\ m}{77\ s}$$

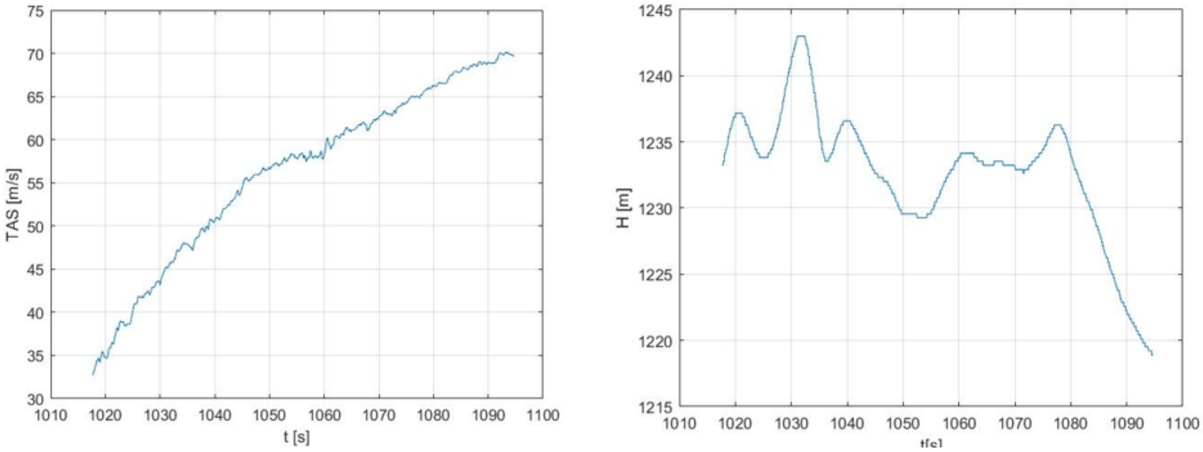


Figure 6.25: TAS and altitude at test point 4000 ft

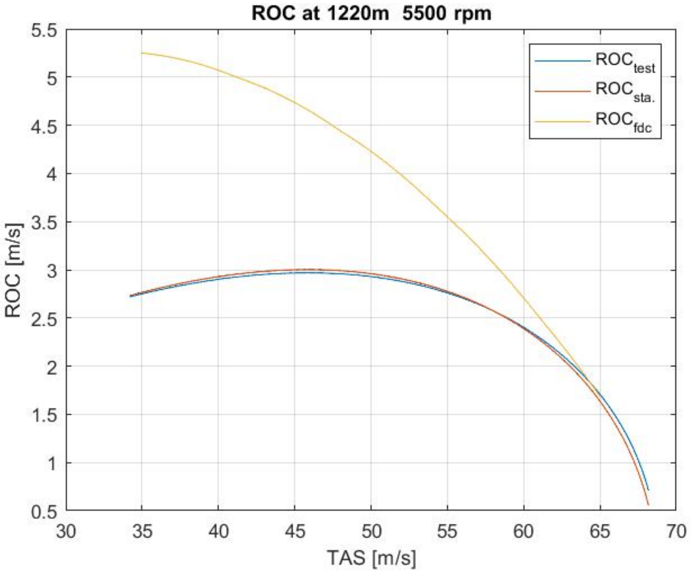


Figure 6.26: ROC at test point 4000 ft

6.4.11 Test point 6000 ft I , second session

From the interpolation of the TAS graph it is obtained :

$$V = -0.009448t^2 + 26.25t - 1.817 \cdot 10^4$$

So the acceleration is:

$$a = dV/dt = -0.009448 \cdot 2 \cdot t + 26.25$$

The ROC is calculated using the 6.3, where the correction on the altitude corresponds to:

$$\frac{\Delta H}{\Delta t} = \frac{43\ m}{60\ s}$$

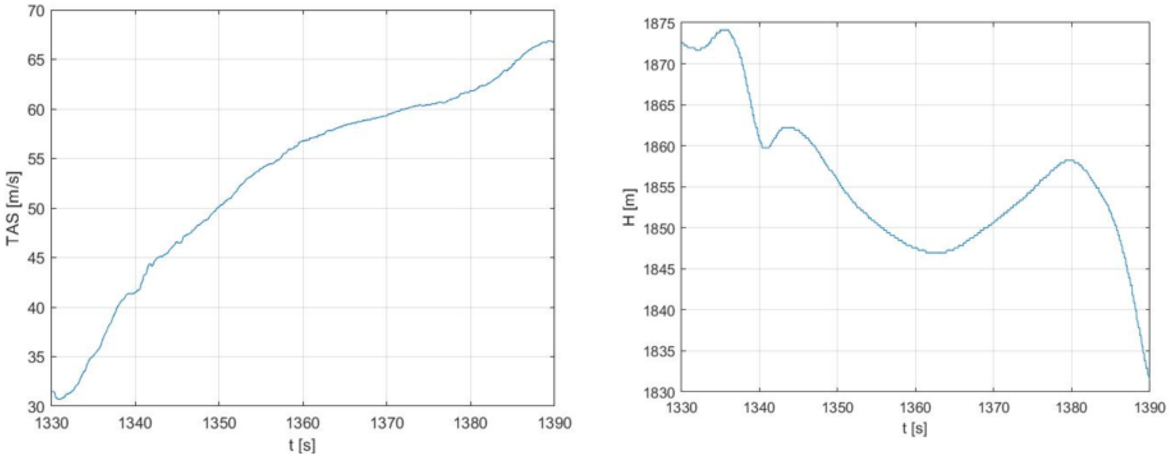


Figure 6.27: TAS and altitude at test point 6000 ft

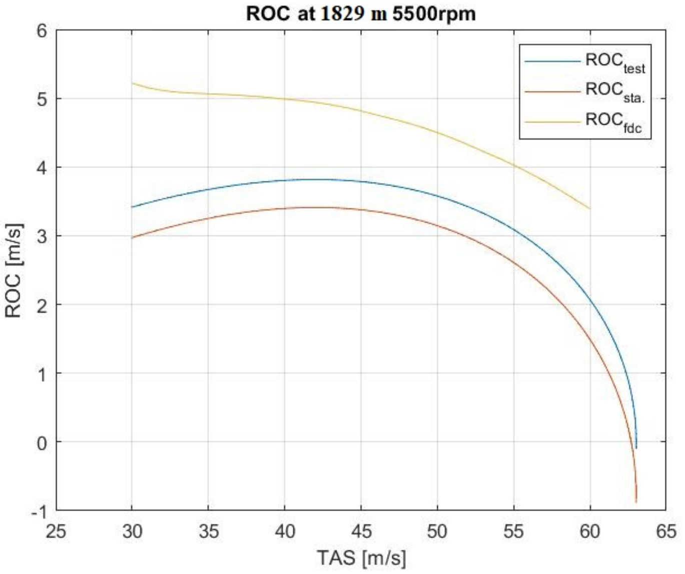


Figure 6.28: ROC at test point 6000 ft

6.4.12 Test point 6000 ft II , second session

From the interpolation of the TAS graph it is obtained :

$$V = -0.01015t^2 + 31.27t - 2.403 \cdot 10^4$$

So the acceleration is:

$$a = dV/dt = -0.01015 \cdot 2 \cdot t + 31.27$$

The ROC is calculated using the 6.3, where the correction on the altitude corresponds to:

$$\frac{\Delta H}{\Delta t} = \frac{9\ m}{55\ s}$$

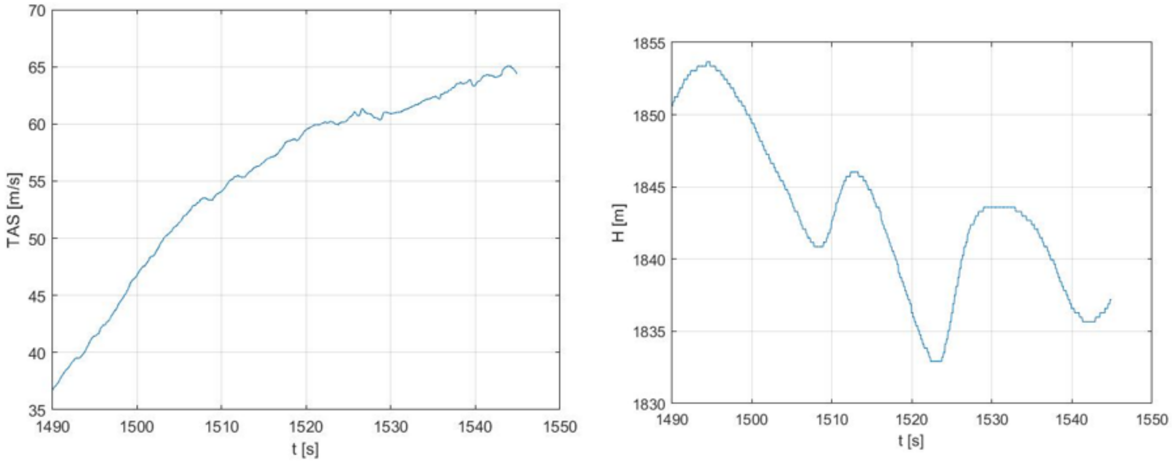


Figure 6.29: TAS and altitude at test point 6000 ft

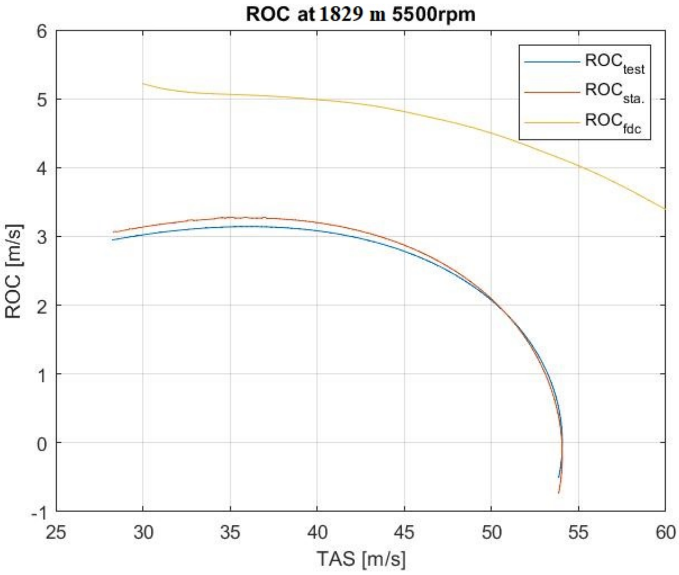


Figure 6.30: ROC at test point 6000 ft

6.5 Comments on the data analyses

It should be noted that the on-board computer did not directly record the power values of the engine, nor the fuel consumption, with which power values could have been obtained. The recorded rpm are not reliable values as in some cases for long periods of time they exceed the maximum value by 1000 RPM. The elevator deflection is not recorded continuously, but only discreetly and as a percentage. Often this value remains constant for long periods of time. Therefore, unfortunately not having reliable rpm values and more precise values on the elevator, it was not possible to make comparisons on simple trim conditions between the aircraft and the model. At the same time, therefore, it was not possible to verify the graphs made in trim condition with varying speed and altitude. Flight testing was not conducted with the purpose of official certification. Having only about two hours of flight, it was only possible to carry out tests on the performance in climb. Please note that the vortex generators were installed on the wings, so a decrease in performance is actually expected. In the flight manual, in the performance chapter [1, p. 97], there are no graphs or values of the ROC as the speed varies. This means that it is not possible to define a certain error between the values obtained from the flight tests and those of the flight manual. In it, instead, there is only a graph that allows to obtain a maximum rate of climb, starting from the altitude and the mass of the aircraft, therefore speed variations are not taken into account. It follows that the analysis carried out are more qualitative than quantitative. As already noted, the flight tests were not aimed at certification, so it was not possible to wait for the optimal day. Unfortunately the weather conditions were not optimal because of clouds, wind and rain. Therefore, particular attention must be paid when comparing the ROC obtained from flight tests for different altitudes or speeds. In general, the ROC curve is expected to decrease as the altitude increases. This is visible in some graphs above, such as the graphs at 4000 ft and 5000 ft in the 6.4.4 and 6.4.5 sections . In general, the graphs shown above show how the mathematical model of the Bs Prime differs from the real one. From a qualitative point of view, it is observed that the ROC calculated by FDC does not show that parabolic-like trend, which instead characterizes the results of the flight tests. In particular, it can be observed that the ROC_{fdc} at low speeds does not tend to curve downwards, ie towards lower rates of climb, but continues to grow or remains approximately constant. Especially in the second test session, in which the initial speeds were very low, about 30 m/s, the ROC calculated by FDC shows a sort of inversion: after being almost constant it tends to grow again, as if there was an inflection point. A possible explanation could be linked to the fact that 30 m/s already represents a very low speed close to stall, and that the aerodynamic model for the hypotheses made does not model the stall condition in a suitable way. The maximum values of ROC_{fdc} remain below or similar to the maximum expected in the graphs in the flight manual [1]

Chapter 7

Conclusion

As already mentioned, this thesis continued the work set out in the thesis "Development of a mathematical model for the BS Prime", with the aim of verifying the reliability of the developed mathematical model. The results of this discussion showed how the model gives results in line with expectations, and comparable with the values that characterize similar aircraft. The difference compared to the real aircraft, which emerged from the performance comparison, makes it clear that the mathematical model still has too many gaps to be used on a practical level. This is due to different reasons: for the geometry modeling in DATCOM, the measurements taken with the caliper are subjected to a certain precision error; DATCOM itself introduces errors on the derivatives as shown in the reference [9], in particular the effect of the propeller is not taken into account; in the absence of data, the airfoil was chosen on the basis of some assumptions, therefore, since it is not the real one, an error is introduced on the value of the aerodynamic derivatives; in the absence of data, the modeling of the propeller required many assumptions as well as the calculation of the center of gravity, inertia and mass distribution. Future developments of this project could lead to a complete validation process. The validation of a mathematical model is an extremely long and laborious process, which requires enormous experience and considerable economic resources. It would have been impossible to carry out such work as part of a university thesis, and without adequate financial coverage. The achieved results shown in this discussion should therefore be understood as a first step. A complete validation would inevitably lead to a modification of the mathematical model, trying to obtain more and more precise results. In addition to the possibility of completing the validation phase, the continuation envisaged for this project is the application in the context of flight simulation. The EURO FLIGHT TEST company has a flight simulator integrated with X-plane and Microsoft Simulator software, within which the mathematical model could be inserted. This would allow pilots to get used to the aircraft before making actual flights, or simply to practice with specific procedures avoiding the costs associated with the actual use of the aircraft. From a didactic point of view, the project was of great interest, and allowed to deepen the knowledge regarding many aspects: aerodynamics, flight mechanics, modeling, numerical methods, etc. Practical experiences, and in particular that of in-flight tests, were also invaluable.

Appendix A

Actrimforcond

```
% The FDC toolbox. Trim routine ACTRIM.
format short e;
options = [];
turntype = 'c';
% Display header; welcome to ACTRIM!
% -----------------------------------------
clc
disp('The FDC toolbox - ACTRIM');
disp('========================');
disp(' ');
disp('This program searches determines a steady-state trimmed-flight condition');
disp('for a non-linear aircraft model in Simulink.');
disp(' ');
disp(' ');
% Check if AM, EM, GM1, and GM2 have been defined in the Matlab workspace.
% If not, run DATLOAD to load them from file.
% ------------------------------------------------------------------------------
if exist('AM')==0 | exist('EM')==0 | exist('GM1')==0 | exist('GM2')==0
h=newMsgBox(['First, the model parameters need to be retrieved from file ', ...
'(e.g. AIRCRAFT.DAT). Click ''OK'' or press Enter to continue.']);
uiwait(h);
datload('aircraft');
end
% If model parameters are still not present in the workspace,
% e.g. due to an incorrect datafile, force an abort of ACTRIM.
% ----------------------------------------------------------------------
if exist('AM')==0 | exist('EM')==0 | exist('GM1')==0 | exist('GM2')==0
error(['ERROR: the model parameters are still not present in the workspace! ', ...
'Aborting ACTRIM.']);
end
% Set xinco (= initial value of the state vector).
% ------------------------------------------------------------------------------
xinco = [45 0 0 0 0 0 0 0 0 0 0 0]';
% Here, all state variables are allowed to vary, so xfix will be set to
% one. Replace this line by:
% ------------------------------------------------------------------------------
xfix = 1;
%%%% fixstate;
% The name of the aircraft model to be evaluated will be stored in the
% stringvariable sysname.
% -----------------------------------------------------------------------------
```

```
disp('Give name of system with aircraft model');
disp('default = beaver');
sysname = input('> ','s');
if isempty(sysname)
sysname = 'beaver';
end
% Display menu in which the user can choose between a number of trimmed—
% flight conditions and quitting.
% ——————————————————————————————————————————————————————————————————————
clc
opt = txtmenu('Select type of steady—state flight',...
'Steady wings—level flight','Steady turning flight',...
'Steady pull—up','Steady roll','Quit');
skip  = 0; % Do not skip iteration block, unless option 'quit' is used
% (then skip will be set to 1).
matrnostra = eye(0);

speed = [41.15 42.15 43.2 45.39 46.56]; n = 5500; pot_attesa = 67.89e3;
coeffgamma = 1;
%  speed = [66.88 67.91 68.93 69.96 70.99]; n = 5000; pot_attesa = 51.29e3;
%  coeffgamma = 0;
% speed = [63.79 65.02 66.26 67.49 68.73]; n = 4800; pot_attesa = 45.258e3;
% coeffgamma = 0;
% speed = [59.16 60.19 61.21 62.24 63.27]; n = 4300; pot_attesa = 37.72e3;
% coeffgamma = 0;

for cond = 1:1:5
% Define flight condition, depending upon trim option chosen
% ————————————————————————————————————————————————————————————
if opt == 1                                           % STEADY WINGS—LEVEL FLIGHT
if cond ==  1
V = speed(cond);
H = 0;
psi = 0;
gammatype = 'f';
gamma = (8.38*pi/180)*coeffgamma;
pz = 22;
elseif cond == 2
V = speed(cond);
H = 609.76;
psi = 0;
gammatype = 'f';
gamma = (7.62*pi/180)*coeffgamma;
pz = 21;
elseif cond == 3
V = speed(cond);
H = 1220;
psi = 0;
gammatype = 'f';
gamma = (6.75*pi/180)*coeffgamma;
pz = 24;
elseif cond == 4
V = speed(cond);
H = 1829.27;
psi = 0;
gammatype = 'f';
```

```
gamma = (5.78*pi/180)*coeffgamma;
pz = 21;
elseif cond == 5
V = speed(cond);
H = 2439;
psi = 0;
gammatype = 'f';
gamma = (5.01*pi/180)*coeffgamma;
pz = 24;
end

phidot   = 0;
psidot   = 0;
thetadot = 0;

if skip ~= 1          % DEFINE CONFIGURATION OF THE AIRPLANE, IF NOT QUITTING
%------------------------------------------------------------
% For the 'Beaver' model, the flap angle and engine speed define the
% configuration of the aircraft
%---------------------------------------------------------------------------
deltaf = 0;
% G is the centripetal acceleration, used for the coordinated turn
% constraint.
%----------------------------------------------------------------------
G = psidot*V/9.80665;
%--------------------------------------------------------------------------
phi = [];
if opt == 2 & turntype == 'u';  % Steady turn, uncoordinated.
phi = input('Give desired roll angle phi [deg], default = 0: ')*pi/180;
end
if isempty(phi)
phi = 0;
end
% Note: vtrim contains the first estimation of the non-constant states
% and inputs. The final values will be determined iteratively.
%-----------------------------------------------------------------------------
if gammatype == 'f' % gamma in ctrim, pz in vtrim
ctrim = [V H psi gamma G psidot thetadot phidot deltaf n phi]';
vtrim = [0 0 0 0 0 pz]';
else % gamma in vtrim, pz in ctrim
ctrim = [V H psi pz G psidot thetadot phidot deltaf n phi]';
vtrim = [0 0 0 0 0 0]';  %<-- initial guess for gamma = 0!
end

% The Simulink system BEAVER or an equivalent aircraft model is called by
% the function call xdot = feval('beaver',t,x,u,'outputs'),
%-----------------------------------------------------------------------------
modfun   = ['xdot = feval(''',sysname,''',0,x,u,''outputs'');  '];
modfun   = [modfun 'xdot = feval(''',sysname,''',0,x,u,''derivs'');'];
%--------------------------------------------------------------------------
warning off
feval(sysname,[],[],[],'compile');
clear ans % (the above feval statement somehow generates an 'ans' variable)
warning on

% Call minimization routine FMINSEARCH.
% ---------------------------------------------------------------------------
clc;
disp('Searching for stable solution. Wait a moment...');
disp(' ');
options = optimset('Display','off','TolX',1e-30,'MaxFunEvals',5e5,'MaxIter',5e5);
[vtrimmed,fval,exitflag,output] = fminsearch('accost',vtrim,options,...
ctrim,rolltype,turntype,gammatype,modfun)

% Display error message when maximum number of iterations is
% reached before finding converged solution
% ----------------------------------------------------------------
if exitflag == 0
warning('Maximum number of iterations was exceeded!');
disp(['- number of function evaluations: ' num2str(output.funcCount)]);
disp(['- number of iterations: ' num2str(output.iterations)]);
else
disp('Converged to a trimmed solution...');
end
% Call subroutine ACCONSTR, which contains the flight-path constraints
% once more to obtain the final values of the inputvector and states.
% -------------------------------------------------------------------------
[x,u] = acconstr(vtrimmed,ctrim,rolltype,turntype,gammatype);

% Call a/c model once more, to obtain the time-derivatives of the states
% in trimmed-flight condition.
% ---------------------------------------------------------------------------
eval(modfun);

% Release compiled aircraft model now that all results are known
% ------------------------------------------------------------------
feval(sysname,[],[],[],'term');
%------------------------------------------------------------------------
xinco  = x;           % Initial value of state vector
xdot0  = xdot;        % Initial value of time-derivative of state vector
uaero0 = u(1:4);      % Initial value of input vector for aerodynamic model
uprop0 = u(5:6);      % Initial value of input vector for engine
%                             forces and moments model
% --------------------------------------------------------------------------
end
Power = 1.7*(uprop0(2)-16.14)*(uprop0(1)-1922);
deltaP = Power-pot_attesa;
manetta = 0.00136*Power-2.033;
MAP = uprop0(2);
vettnostro = [Power manetta MAP xinco(1) xinco(2)*180/pi uaero0(1)*180/pi deltaP];
matrnostra = [matrnostra vettnostro'];
end                   % END OF TRIM-LOOP (SKIPPED IF OPTION 5 = QUIT IS SELECTED)

disp('MAP')
matrnostra(3,:)
disp('potenza calcolata')
matrnostra(1,:)
disp('equilibratore')
matrnostra(6,:)
disp(' ');
disp('Ready.');
disp(' ');
```

Appendix B

Actrim_froV_forH

```
format short e;
options = [];
turntype = 'c';
clc
disp('The FDC toolbox - ACTRIM for H and for V');
disp('========================');
disp(' ');
disp('This program searches determines a steady-state trimmed-flight condition');
disp('for a non-linear aircraft model in Simulink, changing V and H');
disp(' ');
disp(' ');
% Check if AM, EM, GM1, and GM2 have been defined in the Matlab workspace.
% If not, run DATLOAD to load them from file.
%------------------------------------------------------------------------
if exist('AM')==0 | exist('EM')==0 | exist('GM1')==0 | exist('GM2')==0
h=newMsgBox(['First, the model parameters need to be retrieved from file ', ...
'(e.g. AIRCRAFT.DAT). Click ''OK'' or press Enter to continue.']);
uiwait(h);
datload('aircraft');
end
% If model parameters are still not present in the workspace,
% e.g. due to an incorrect datafile, force an abort of ACTRIM.
%---------------------------------------------------------------
if exist('AM')==0 | exist('EM')==0 | exist('GM1')==0 | exist('GM2')==0
error(['ERROR: the model parameters are still not present in the workspace! ', ...
'Aborting ACTRIM.']);
end
% Set xinco (= initial value of the state vector).
%------------------------------------------------------------------------
xinco = [45 0 0 0 0 0 0 0 0 0 0 0]';
% Here, all state variables are allowed to vary, so xfix will be set to
% one.
xfix = 1;
%%%% fixstate;
% The name of the aircraft model to be evaluated will be stored in the
% stringvariable sysname.
%------------------------------------------------------------------------
disp('Give name of system with aircraft model');
disp('default = beaver');
sysname = input('> ','s');
```

```
if isempty(sysname)
sysname = 'beaver';
end
% Display menu in which the user can choose between a number of trimmed-
% flight conditions and quitting.
clc
opt = txtmenu('Select type of steady-state flight',...
'Steady wings-level flight','Steady turning flight',...
'Steady pull-up','Steady roll','Quit');

skip  = 0; % Do not skip iteration block, unless option 'quit' is used
% (then skip will be set to 1).

% Define flight condition, depending upon trim option chosen
%----------------------------------------------------------------
if opt == 1                                          % STEADY WINGS-LEVEL FLIGHT
%-------------------------------
clc
disp('Steady wings-level flight.');
disp('==========================');
V = 45
H = 0;
psi  = input('Give heading [deg], default = 0: ')*pi/180;
if isempty(psi)
psi = 0;
end

ok = 0;
while ok~= 1
gammatype = input('Use specified manifold pressure or flight-path angle ([m]/f)? ','s');
if isempty(gammatype)
gammatype = 'm';
end
if gammatype == 'f'
gamma = input('Give flightpath angle [deg], default = 0: ')*pi/180;
if isempty(gamma)
gamma = 0;
ok = 1;
else
ok=1;
end
elseif gammatype == 'm'
ok = 1;
else
disp('   Invalid entry. Please choose either ''m'', or ''f''');
end
end

phidot   = 0;
psidot   = 0;
thetadot = 0;

rolltype = 'b';   % No rolling, so for reasons of simplicity the default setting,
% i.e. a body-axes roll, will be used.

%%%%
```

```
if skip ~= 1          % DEFINE CONFIGURATION OF THE AIRPLANE, IF NOT QUITTING
%-----------------------------------------------------------------------------
deltaf = input('Give flap angle [deg], default = 0: ')*pi/180;
if isempty(deltaf)
deltaf = 0;
end

n = input('Give engine speed [RPM], default = 1800: ');
if isempty(n)
n = 1800;
end

pz = 20;

% constraint.
%-----------------------------------------------------------------------
G = psidot*V/9.80665;

%----------------------------------------------------------------------------
phi = [];
if opt == 2 & turntype == 'u';  % Steady turn, uncoordinated.
phi = input('Give desired roll angle phi [deg], default = 0: ')*pi/180;
end

if isempty(phi)
phi = 0;
end
% Note: vtrim contains the first estimation of the non-constant states
% and inputs. The final values will be determined iteratively.
%-------------------------------------------------------------------------------

MAP_m = eye(0);
deltae1_m = eye(0);
deltar1_m = eye(0);
deltaa1_m = eye(0);
alpha1_m = eye(0);
POTENZA_m=eye(0);
MANETTA_m=eye(0);

MAP = [];
deltae1 = [];
deltar1 = [];
deltaa1 = [];
alpha1 = [];
POTENZA=[];
MANETTA=[];

for H = 500:500:2000
if H == 500
pz = 24;
else
pz = 22;
end
for V = 50:3:65;
if gammatype == 'f' % gamma in ctrim, pz in vtrim
```

```
ctrim = [V H psi gamma G psidot thetadot phidot deltaf n phi]';
vtrim = [0 0 0 0 0 pz]';
else % gamma in vtrim, pz in ctrim
ctrim = [V H psi pz G psidot thetadot phidot deltaf n phi]';
vtrim = [0 0 0 0 0 0]';  %<-- initial guess for gamma = 0!
end
% The Simulink system BEAVER or an equivalent aircraft model is called
% -----------------------------------------------------------------
modfun   = ['xdot = feval(''',sysname,''',0,x,u,''outputs'');  '];
modfun   = [modfun 'xdot = feval(''',sysname,''',0,x,u,''derivs'');'];

% First pre-compile the aircraft model
%-----------------------------------------------------------------
warning off
feval(sysname,[],[],[],'compile');
clear ans % (the above feval statement somehow generates an 'ans' variable)
warning on

% Call minimization routine FMINSEARCH.

options = optimset('Display','off','TolX',1e-30,'MaxFunEvals',5e5,'MaxIter',5e5);
[vtrimmed,fval,exitflag,output] = fminsearch('accost',vtrim,options,...
ctrim,rolltype,turntype,gammatype,modfun)

% Display error message when maximum number of iterations is
% reached before finding converged solution
% -----------------------------------------------------
if exitflag == 0
warning('Maximum number of iterations was exceeded!');
disp(['- number of function evaluations: ' num2str(output.funcCount)]);
disp(['- number of iterations: ' num2str(output.iterations)]);
else
disp('Converged to a trimmed solution...');
end

% Call subroutine ACCONSTR, which contains the flight-path
[x,u] = acconstr(vtrimmed,ctrim,rolltype,turntype,gammatype);

Potenza=(u(6)-5.813)*(u(5)-2392);
Manetta=0.00136*Potenza-2.033;

MANETTA=[MANETTA Manetta];
POTENZA=[POTENZA Potenza];
MAP = [MAP u(6)];
deltaa1 = [deltaa1 u(2)];
deltar1 = [deltar1 u(3)];
deltae1 = [deltae1 u(1)];
alpha1 = [alpha1 x(2)];

% Call a/c model once more, to obtain the time-derivatives of the states
% in trimmed-flight condition.
% -----------------------------------------------------------------
eval(modfun);
% Release compiled aircraft model now that all results are known
% --------------------------------------------------------
feval(sysname,[],[],[],'term');
```

```
end

MAP_m = [MAP_m  MAP'];
deltae1_m = [deltae1_m deltae1'];
deltar1_m = [deltar1_m deltar1'];
deltaa1_m = [deltaa1_m  deltaa1'];
alpha1_m = [alpha1_m  alpha1'];
POTENZA_m=[POTENZA_m  POTENZA'];
MANETTA_m=[MANETTA_m  MANETTA'];

MAP = [];
deltae1 = [];
deltar1 = [];
deltaa1 = [];
alpha1 = [];
POTENZA=[];
MANETTA=[];
end
disp('==================================================');
disp('            End, now run Plot_grafici');
disp('==================================================');
disp(' ');
%
```

Appendix C

CtrimforROC

```
format short e;
options = [];
turntype = 'c';

% Display header; welcome to ACTRIM!
%------------------------------------------------------
clc
disp('The FDC toolbox - ACTRIM');
disp('========================');
disp(' ');
disp('This program searches determines a steady-state trimmed-flight condition');
disp('for a non-linear aircraft model in Simulink.');
disp(' ');
disp(' ');

if exist('AM')==0 | exist('EM')==0 | exist('GM1')==0 | exist('GM2')==0
h=newMsgBox(['First, the model parameters need to be retrieved from file ', ...
'(e.g. AIRCRAFT.DAT). Click ''OK'' or press Enter to continue.']);
uiwait(h);
datload('aircraft');
end
% If model parameters are still not present in the workspace,
% e.g. due to an incorrect datafile, force an abort of ACTRIM.
%--------------------------------------------------------------------------
if exist('AM')==0 | exist('EM')==0 | exist('GM1')==0 | exist('GM2')==0
error(['ERROR: the model parameters are still not present in the workspace! ', ...
'Aborting ACTRIM.']);
end
xinco = [45 0 0 0 0 0 0 0 0 0 0 0]';
xfix = 1;
%%%% fixstate;
disp('Give name of system with aircraft model');
disp('default = beaver');
sysname = input('> ','s');
if isempty(sysname)
sysname = 'beaver';
end

clc
opt = txtmenu('Select type of steady-state flight',...
'Steady wings-level flight','Steady turning flight',...
'Steady pull-up','Steady roll','Quit');

skip  = 0; % Do not skip iteration block, unless option 'quit' is used
% (then skip will be set to 1).

% Define flight condition, depending upon trim option chosen
% ---------------------------------------------------------------------
if opt == 1                                             % STEADY WINGS-LEVEL FLIGHT
% ------------------------------
clc
disp('Steady wings-level flight.');
disp('==========================');

V = input('Give desired airspeed [m/s], default = 45: ');
if isempty(V)
V = 45;
end

H = input('Give (initial) altitude [m], default = 0: ');
if isempty(H)
H = 0;
end

psi   = input('Give heading [deg], default = 0: ')*pi/180;
if isempty(psi)
psi = 0;
end
ok = 0;
gammatype = 'm';

phidot    = 0;
psidot    = 0;
thetadot = 0;

rolltype = 'b';    % No rolling, so for reasons of simplicity the default setting,
% i.e. a body-axes roll, will be used.
end
%%%%

% DEFINE CONFIGURATION OF THE AIRPLANE, IF NOT QUITTING
% ---------------------------------------------------------------

% For the 'Beaver' model
deltaf = input('Give flap angle [deg], default = 0: ')*pi/180;
if isempty(deltaf)
deltaf = 0;
end
n = input('Give engine speed [RPM], default = 1800: ');
if isempty(n)
n = 1800;
end

% ------------------------------------------------------------------------------------
if gammatype == 'f'
pz = input('Give estimate of manifold pressure pz ["Hg], default = 20: ');
else  % gammatype == 'm'
```

```
pz = input('Give manifold pressure pz ["Hg], default = 20: ');
end
if isempty(pz)
pz = 20;
end

% G is the centripetal acceleration, used for the coordinated turn
% constraint.
% ----------------------------------------------------------------
G = psidot*V/9.80665;
% If steady, uncoordinated, turning condition must be determined, the
% roll angle can be specified freely; equilibrium will be obtained for
% a trimmed-flight sideslip angle, which in this case usually will be
% quite large.
% --------------------------------------------------------------------
phi = [];

if isempty(phi)
phi = 0;
end
%*******************************************************
ROC = [];
deltae1 = [];
alpha1 = [];
TAS = [];
for V = 30:1:60

if gammatype == 'f' % gamma in ctrim, pz in vtrim
ctrim = [V H psi gamma G psidot thetadot phidot deltaf n phi]';
vtrim = [0 0 0 0 0 pz]';
else % gamma in vtrim, pz in ctrim
ctrim = [V H psi pz G psidot thetadot phidot deltaf n phi]';
vtrim = [0 0 0 0 0 0]';  %<-- initial guess for gamma = 0!
end

% The Simulink system BEAVER or an equivalent aircraft model is called by
modfun   = ['xdot = feval(''',sysname,''',0,x,u,''outputs''); '];
modfun   = [modfun 'xdot = feval(''',sysname,''',0,x,u,''derivs'');'];

% First pre-compile the aircraft model (temporarily suppress warnings
% to prevent harmless, but inconvenient messages when the model is
% already compiled).
%-------------------------------------------------------------------
warning off
feval(sysname,[],[],[],'compile');
clear ans % (the above feval statement somehow generates an 'ans' variable)
warning on

% Call minimization routine FMINSEARCH.
% ----------------------------------------------------------------------
clc;
options = optimset('Display','off','TolX',1e-30,'MaxFunEvals',5e5,'MaxIter',5e5);
[vtrimmed,fval,exitflag,output] = fminsearch('accost',vtrim,options,...
ctrim,rolltype,turntype,gammatype,modfun)

% Call subroutine ACCONSTR, which contains the flight-path constraints
% once more to obtain the final values of the inputvector and states.
%------------------------------------------------------------------------
[x,u] = acconstr(vtrimmed,ctrim,rolltype,turntype,gammatype);

% Call a/c model once more, to obtain the time-derivatives of the states
% in trimmed-flight condition.
%---------------------------------------------------------------------------
eval(modfun);
% Release compiled aircraft model now that all results are known
%------------------------------------------------------------------
feval(sysname,[],[],[],'term');
ROC = [ROC xdot(end)];
deltae1 = [deltae1 u(1)];
alpha1 = [alpha1 x(2)];
TAS = [TAS x(1)];
end
plot(TAS, ROC);
disp('end');
```

Appendix D

AclinforCG

```
format short e
skip = 0;          % Help variable which is set to 1 if user selects
% QUIT from main menu. If skip = 1, the last part of
% ACLIN, where the actual linearization takes place,
% is skipped.
clc
disp('FDC toolbox - ACLIN');
disp('====================');
disp('This program will linearize the nonlinear aircraft model in SIMULINK.');
disp(' ');
% Enter name of the aircraft model.
% ------------------------------------
disp('Enter name of the aircraft model in Simulink (default: BEAVER)');
sysname = input('> ','s');
if isempty(sysname)
sysname = 'beaver';
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Define or load operating point.
% ==================================

ok = 0;
while ok ~= 1
% -----------------------------------------------------------------
clc
opt = txtmenu('Choose one of the following options',...
'Load operating point from file',...
'Manually define operating point',...
'Use operating point defined in workspace',...
'Run ACTRIM to determine trimmed flight condition',...
'Quit');

if opt == 1                                        % LOAD OPERATING POINT
% ---------------------
triload;
ok = 1;
end
elseif opt == 3                       % USE OPERATING POINT FROM WORKSPACE
% ------------------------------------
clc
if exist('xinco') == 0 | exist('uaero0') == 0 | exist('uprop0') == 0
```

```
% Currently, no operating point has been defined in the Matlab
% workspace. Display error message and return to main menu.
% ------------------------------------------------------------
clc
disp('ACLIN expects the following variables to be present in the');
disp('Matlab workspace:');
disp(' ');
disp('   xinco = state vector in operating point');
disp('   uaero0= vector with aerodynamic control inputs');
disp('   uprop0= vector with engine control inputs');
disp(' ');
disp('At least one of these vectors is currently not present, so');
disp('linearization cannot proceed!');
disp(' ');
disp('<<<Press a key to return to main menu>>>');
pause
else
% Ask if current definition of the operating point is correct.
% If not, program will return to main menu.
% ------------------------------------------------------------
clc
disp('Current definition of operating point (xinco = states,');
disp('uaero0 = aerodynamic inputs, uprop0 = engine inputs):');
xinco
uaero0
uprop0
answ = input('Is this correct (y/[n])? ','s');
if isempty(answ) | answ ~= 'y'
disp(' ');
disp('<<<Press a key to return to main menu>>>');
pause
else
ok = 1;  % If current definition is ok, proceed with
% linearization
end
end

elseif opt == 4   % RUN ACTRIM TO DETERMINE STEADY-STATE TRIMMED-FLIGHT
% CONDITION. Type HELP ACTRIM for more details.
% ---------------------------------------------------

% ------------------------------------------------------------------
clear uaero0 uprop0 xinco xdot0 % delete operating point definition
%              from workspace (if present)

save aclin.tmp            % save remaining variables to temporary file
clear                     % ... and clear workspace
actrim                    % run ACTRIM
load aclin.tmp -mat       % retrieve variables from temporary file
delete('aclin.tmp');

% ----------------------------------------------------------------
if exist('xinco') == 0 | exist('uaero0') == 0 | exist('uprop0') == 0
disp(' ');
disp('No operating point defined!');
disp(' ');
disp(' ');
disp('<<< Press a key to return to main menu >>>');
pause
else
ok = 1;
end
else                                                                % QUIT
%-----
ok   = 1;
skip = 1;  % Return to end of program rightaway when quitting.
end
end % of operating-point definition.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% if skip ~= 1   % If not quitting from ACLIN, proceed with program.
% Else, go to end of program rightaway!
disp(' ');
disp('<<< Press a key to proceed with model definition >>>');
pause
%----------------------------------------------------------------------------
if exist('AM')==0 | exist('EM')==0 | exist('GM1')==0 | exist('GM2')==0
h=newMsgBox(['First, the model parameters need to be retrieved from file ', ...
'(e.g. AIRCRAFT.DAT). Click ''OK'' or press Enter to continue.']);
uiwait(h);
datload('aircraft');
end
% If model parameters are still not present in the workspace,
% e.g. due to an incorrect datafile, force an abort of ACLIN.
%----------------------------------------------------------------------------
if exist('AM')==0 | exist('EM')==0 | exist('GM1')==0 | exist('GM2')==0
error(['ERROR: the model parameters are still not present in the workspace! ', ...
'Aborting ACLIN.']);
end
xfix = 1;


xg_vett = [0.9492 0.9805 1.0055 1.0306 1.0556 1.0806 1.1057 1.1307 1.1808 1.2434];
CM0_vett = -[0.108 0.102 0.09969 0.09731 0.09496 0.09262 0.09024 0.08789 0.08316 0.0772
CMalpha_vett = -[1.589 1.450 1.339 1.228 1.117 1.007 0.8957 0.7848 0.5631 0.2861];
CMalpha2_vett = [0.2401 0.2386 0.2367 0.2354 0.2335 0.234 0.2313 0.2299 0.227 0.2238];
CMq_vett = -[16.57 16.25 16.00 15.76 15.52 15.30 15.08 14.86 14.45 13.98];
CMdeltae_vett = -[0.7678 0.7618 0.757 0.7522 0.7473 0.7424 0.7377 0.7328 0.7232 0.7111];
CLq_vett = [8.492 8.214 7.993 7.77 7.548 7.326 7.104 6.882 6.438 5.884];
CYr_vett = [0.2393 0.2372 0.2356 0.2340 0.2323 0.2307 0.2290 0.2274];
CYp_vett = -0.0839;
Clb_vett = -0.0424;
Clp_vett = -0.447;
Clr_vett = 0.0759;
Cnb_vett = [0.07426 0.07237 0.07087 0.06935 0.06785 0.06634 0.06433 0.06333];
Cnp_vett = -[0.03027 0.03006 0.02987 0.02968 0.02951 0.02933 0.02914 0.02897];
Cnr_vett = -[0.22315 0.22055 0.2185 0.21643 0.2139 0.2111 0.2082 0.2054];
Cndr_vett = -[0.0278 0.0276 0.0274 0.0273 0.0271 0.0269 0.0268 0.0266];
autovalRe = eye(0);
autovalIm = eye(0);
dampingCP = [];
dampingPH = [];
pulseCP = [];
```

```
pulsePH = [];
TCP = [];
TPH = [];
tmezziCP = [];
tmezziPH = [];
%xdef = [1 2 5 8];
xdef = [3 4 6 9];
for vett=1:1:(length(CM0_vett)-2)
AM(5,1) = CM0_vett(vett);
AM(5,2) = CMalpha_vett(vett);
AM(5,3) = CMalpha2_vett(vett);
AM(5,9) = CMq_vett(vett);
AM(5,11) = CMdeltae_vett(vett);
AM(3,9) = CLq_vett(vett);
AM(2,10) = CYr_vett(vett);
AM(2,8) = CYp_vett;
AM(4,5) = Clb_vett;
AM(4,8) = Clp_vett;
AM(4,10) = Clr_vett;
AM(6,5) = Cnb_vett(vett);
AM(6,8) = Cnp_vett(vett);
AM(6,10) = Cnr_vett(vett);
AM(6,14) = Cndr_vett(vett);
%-----------------------------------------------------------------------
uinco = [uaero0; uprop0; 0; 0; 0; 0; 0; 0];
%-----------------------------------------------------------------------
[Aac, Bac, Cac, Dac] = linmod(sysname,xinco,uinco);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-----------------------------------------------------------------------
allstates = 0;
ok = 0;
xdef = xdef;
allstates = 1;                    % Set all states flag.
%-----------------------------------------------------------------------
udef = [1 2 3 4 5 6];
allinputs = 0;
%---------------------------------------------------------------------
% Determine new state matrix Aac_s, depending upon element numbers
% selected above.
%--------------------------------------------------------------------
for ii = 1:length(xdef)               % ii = row number
for jj = 1:length(xdef)            % jj = column number
Aac_s(ii,jj) = Aac(xdef(ii),xdef(jj));
end
end
% Determine new state matrix Bac_s, depending upon element numbers
% selected above.
%--------------------------------------------------------------------
for ii = 1:length(xdef)               % ii = row number
for jj = 1:length(udef)            % jj = column number
Bac_s(ii,jj) = Bac(xdef(ii),udef(jj));
end
end
lambda = eig(Aac_s);
autovalRe = [autovalRe real(lambda)];
autovalIm = [autovalIm imag(lambda)];
```

```
pulseCP(vett) = sqrt((imag(lambda(1)))^2 + (real(lambda(1)))^2);
pulsePH(vett) = sqrt((imag(lambda(3)))^2 + (real(lambda(3)))^2);
dampingCP(vett) = -real(lambda(1))/pulseCP(vett);
dampingPH(vett) = -real(lambda(3))/pulsePH(vett);
TCP(vett) = 2*pi/imag(lambda(1));
TPH(vett) = 2*pi/imag(lambda(3));
tmezziCP(vett) = log(2)/real(lambda(1));
tmezziPH(vett) =log(2)/real(lambda(3)) ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Display results for user.
% =============================
end
autovalRe
disp(' ');
autovalIm
disp('Ready.');
disp(' ');
```

Appendix E

First flight test session data analysis

```
%% diego 3000ft
tempo=VarName1-VarName1(1);
TAS=VarName8*0.514444;
quota=VarName2;
deltae=VarName12;
T=VarName12+237.15;
inizio=18001;
fine= 19521;

% figure(1)
% plot(tempo,TAS)
% xlabel('t [s]')
% ylabel('TAS [m/s]')
%
% figure(2)
% plot(tempo,quota)
% xlabel('t[s]')
% ylabel('H [m]')
%
% figure(3)
% plot(tempo,deltae)

%find indice dal tempo
for i=inizio:1:fine
if tempo(i)==1220
casper=i;
end
end

rho_sl  = 1.225;          % a livello del mare [kg/m^3]
T_sl    = 288.15;         % Temperatura a livello del mare [K]
a       = -0.0065;         % Gradiente termico [K/m]
h = quota./3.28;
% Valutazione della temperatura e della rho al variare della quota
% (fino a tropopausa)
for  i = 1:length(h)
T_s(i)      = T_sl + a.*h(i);
rho_s(i)    = rho_sl.*(T_s(i)./T_sl).^4.25;
end

figure(1)
```

```
plot(tempo(inizio:fine),TAS(inizio:fine))
xlabel('t [s]')
ylabel('TAS [m/s]')

figure(2)
plot(tempo(inizio:fine),quota(inizio:fine)./3.28)
xlabel('t[s]')
ylabel('H [m]')

rpm=VarName9;
figure(3)
plot(tempo(inizio:fine),T(inizio:fine))
xlabel('time [s]')
ylabel('T [K]')
hold on
plot(tempo(inizio:fine),T_s(inizio:fine))
legend('T_{test}', 'T_{stand}')
grid on

TAS_a=TAS(inizio:fine);
tempo_a=tempo(inizio:fine);
T_out=T(inizio:fine);
T_s=T_s(inizio:fine);
T_s=T_s';
%funzione che interpola

x=tempo_a;
v_TAS= -0.004164.*x.^2+10.07.*x-6019;
a_TAS=-0.004164.*2.*x+10.07;

ROC= a_TAS.*v_TAS./9.81;
ROC_s=ROC.*T_out./T_s-(11)/100.*T_out./T_s;

V_fdc=40:1:70;

%Vx
m_vett=ROC_s./v_TAS;
i=1;
while v_TAS(i)<max(v_TAS)
m=ROC_s(i)/v_TAS(i);
if m==max(m_vett)
Vx=v_TAS(i)
end
if ROC_s(i)==max(ROC_s)
Vy=v_TAS(i)
end
i=i+1;
end

ROCfdc=[    5.3889     5.3690     5.3411     5.3047     5.2596     5.2055
  5.1422     5.0819     5.0164     4.9418     4.8580     4.7649     4.6622
   4.5528  4.4480     4.3340     4.2106     4.0779     3.9355     3.7835     3.6310
  3.4761     3.3117     3.1379     2.9544     2.7612     2.5581     2.3597
2.1523     1.9352     1.7083  ];
figure(5)
plot(v_TAS,ROC)
```

```
hold on
plot(v_TAS,ROC_s)
grid on
plot(V_fdc,ROCfdc)
legend('ROC_{test}', 'ROC_{sta.}', 'ROC_{fdc}')
xlabel('TAS [m/s]')
ylabel('ROC [m/s]')
title('ROC at 915m  5500 rpm')

%% 4000ft
tempo=VarName1-VarName1(1);
TAS=VarName8*0.514444;
quota=VarName2;
deltae=VarName12;
T=VarName12+237.15;
inizio=  21441;
fine=  22561;

% figure(1)
% plot(tempo,TAS)
%
% figure(2)
% plot(tempo,quota)
%
% figure(3)
% plot(tempo,deltae)

%find indice dal tempo
for i=inizio:1:fine
if tempo(i)==1410
casper=i;
end
end

rho_sl  = 1.225;            %  livello del mare [kg/m^3]
T_sl    = 288.15;           % Temperatura a livello del mare [K]
a       = -0.0065;          % Gradiente termico [K/m]
h = quota./3.28;

% Valutazione della temperatura e della rho al variare della quota
% (fino a tropopausa)
for  i = 1:length(h)
T_s(i)      = T_sl + a.*h(i);
rho_s(i)    = rho_sl.*(T_s(i)./T_sl).^4.25;
end
figure(1)
plot(tempo(inizio:fine),TAS(inizio:fine))
xlabel('t [s]')
ylabel('TAS [m/s]')

figure(2)
plot(tempo(inizio:fine),quota(inizio:fine)./3.28)
xlabel('t[s]')
ylabel('H [m]')

rpm=VarName9;
```

```
figure(3)
plot(tempo(inizio:fine),deltae(inizio:fine))

TAS_a=TAS(inizio:fine);
tempo_a=tempo(inizio:fine);
T_out=T(inizio:fine);
T_s=T_s(inizio:fine);
T_s=T_s';
%funzione che interpola

x=tempo_a;
v_TAS= -0.007362.*x.^2+ 20.66.*x-1.442e+04;
a_TAS=-0.007362.*2.*x+ 20.66;

ROC= a_TAS.*v_TAS./9.81;
ROC_s=ROC.*T_out./T_s-(35)/75.*T_out./T_s;

V_fdc=45:1:75;

%Vx
m_vett=ROC_s./v_TAS;
i=1;
while v_TAS(i)<max(v_TAS)
m=ROC_s(i)/v_TAS(i);
if m==max(m_vett)
Vx=v_TAS(i)
end
if ROC_s(i)==max(ROC_s)
Vy=v_TAS(i)
end
i=i+1;
end

ROCfdc=[5.2500    5.2302    5.2028    5.1674    5.1236    5.0712    5.0100
   4.9518    4.8885    4.8165    4.7355    4.6455    4.5463    4.4405
4.3393    4.2291    4.1099    3.9814    3.8438    3.6967    3.5491    3.3992
   3.2402    3.0719    2.8942    2.7071    2.5105    2.3183
2.1174    1.9071    1.687];

figure(5)
plot(v_TAS,ROC)
hold on
plot(v_TAS,ROC_s)
grid on
plot(V_fdc,ROCfdc)
legend('ROC_{test}', 'ROC_{sta.}', 'ROC_{fdc}')
xlabel('TAS [m/s]')
ylabel('ROC [m/s]')
title('ROC at 1220m  5500 rpm')

%% 5000
tempo=VarName1-VarName1(1);
TAS=VarName8*0.514444;
quota=VarName2;
deltae=VarName12;
T=VarName12+237.15;
```

```
inizio=  24801;
fine=  25761;

% figure(1)
% plot(tempo,T)
%
% figure(2)
% plot(tempo,quota)
%
% figure(3)
% plot(tempo,deltae)

%find indice dal tempo
for i=inizio:1:fine
if tempo(i)==1610
casper=i;
end
end

rho_sl  = 1.225;                % livello del mare [kg/m^3]
T_sl    = 288.15;                   % Temperatura a livello del mare [K]
a       = -0.0065;              % Gradiente termico [K/m]
h = quota./3.28;

% Valutazione della temperatura e della rho al variare della quota
% (fino a tropopausa)
for  i = 1:length(h)
T_s(i)      = T_sl + a.*h(i);
rho_s(i)    = rho_sl.*(T_s(i)./T_sl).^4.25;
end
figure(1)
plot(tempo(inizio:fine),TAS(inizio:fine))
xlabel('t [s]')
ylabel('TAS [m/s]')

figure(2)
plot(tempo(inizio:fine),quota(inizio:fine)./3.28)
xlabel('t[s]')
ylabel('H [m]')

rpm=VarName9;
figure(3)
plot(tempo(inizio:fine),rpm(inizio:fine))

tempo_a=tempo(inizio:fine);
T_out=T(inizio:fine);
T_s=T_s(inizio:fine);
T_s=T_s';
%funzione che interpola

x=tempo_a;
v_TAS= -0.006598.*x.^2+ 21.29.*x-1.711e+04;
a_TAS=-0.006598.*2.*x+ 21.29;

ROC= a_TAS.*v_TAS./9.81;
ROC_s=ROC.*T_out./T_s-(10)/70.*T_out./T_s;
```

```
V_fdc=35:1:65;

%Vx
m_vett=ROC_s./v_TAS;
i=1;
while v_TAS(i)<max(v_TAS)
m=ROC_s(i)/v_TAS(i);
if m==max(m_vett)
Vx=v_TAS(i)
end
if ROC_s(i)==max(ROC_s)
Vy=v_TAS(i)
end
i=i+1;
end

ROCfdc=[
5.1789    5.1741    5.1661    5.1534    5.1350    5.1145    5.0946
  5.0676    5.0329    4.9904    4.9396    4.8802    4.8240    4.7628
   4.6933    4.6151    4.5281    4.4323    4.3300    4.2322    4.1258
     4.0105    3.8863    3.7532    3.6109    3.4681    3.3232    3.1693
3.0065    2.8345    2.6534 ];

figure(5)
plot(v_TAS,ROC)
hold on
plot(v_TAS,ROC_s)
grid on
plot(V_fdc,ROCfdc)
legend('ROC_{test}', 'ROC_{sta.}', 'ROC_{fdc}')
xlabel('TAS [m/s]')
ylabel('ROC [m/s]')
title('ROC at 1524m  5500 rpm')

%% 5000 ritorno
tempo=VarName1-VarName1(1);
TAS=VarName8*0.514444;
quota=VarName2;
deltae=VarName12;
T=VarName12+237.15;
inizio=   28721;
fine= 29761;

figure(1)
plot(tempo,TAS)
figure(2)
plot(tempo,quota)
figure(3)
plot(tempo,deltae)

%find indice dal tempo
for i=inizio:1:fine
if tempo(i)==1795
casper=i;
end
```

```
end

rho_sl  = 1.225;            % livello del mare [kg/m^3]
T_sl    = 288.15;           % Temperatura a livello del mare [K]
a       = -0.0065;          % Gradiente termico [K/m]
h = quota./3.28;

% Valutazione della temperatura e della rho al variare della quota
%(fino a tropopausa)
for  i = 1:length(h)
T_s(i)     = T_sl + a.*h(i);
rho_s(i)   = rho_sl.*(T_s(i)./T_sl).^4.25;
end
figure(1)
plot(tempo(inizio:fine),TAS(inizio:fine))
xlabel('t [s]')
ylabel('TAS [m/s]')

figure(2)
plot(tempo(inizio:fine),quota(inizio:fine)./3.28)
xlabel('t[s]')
ylabel('H [m]')

rpm=VarName9;
figure(3)
plot(tempo(inizio:fine),rpm(inizio:fine))

TAS_a=TAS(inizio:fine);
tempo_a=tempo(inizio:fine);
T_out=T(inizio:fine);
T_s=T_s(inizio:fine);
T_s=T_s';
%funzione che interpola

x=tempo_a;
v_TAS=  -0.009727.*x.^2+ 35.99  .*x-3.323e+04 ;
a_TAS= -0.009727.*2.*x+ 35.99;

ROC= a_TAS.*v_TAS./9.81;
ROC_s=ROC.*T_out./T_s-(35)/70.*T_out./T_s;

V_fdc=35:1:65;

%Vx
m_vett=ROC_s./v_TAS;
i=1;
while v_TAS(i)<max(v_TAS)
m=ROC_s(i)/v_TAS(i);
if m==max(m_vett)
Vx=v_TAS(i)
end
if ROC_s(i)==max(ROC_s)
Vy=v_TAS(i)
end
i=i+1;
end
```

```
ROCfdc=[ 5.1789     5.1741     5.1661     5.1534     5.1350     5.1145
5.0946     5.0676     5.0329     4.9904     4.9396     4.8802     4.8240     4.7628
4.6933     4.6151     4.5281     4.4323     4.3300     4.2322     4.1258
 4.0105     3.8863     3.7532     3.6109     3.4681     3.3232     3.1693
3.0065     2.8345     2.6534];
figure(5)
plot(v_TAS,ROC)
hold on
plot(v_TAS,ROC_s)
grid on
plot(V_fdc,ROCfdc)
legend('ROC_{test}', 'ROC_{sta.}', 'ROC_{fdc}')
xlabel('TAS [m/s]')
ylabel('ROC [m/s]')
title('ROC at 1524m  5500 rpm')

%% 4000 ritorno

tempo=VarName1-VarName1(1);
TAS=VarName8*0.514444;
quota=VarName2;
deltae=VarName12;
T=VarName12+237.15;
inizio=    31681;
fine=  32801;

% figure(1)
% plot(tempo,TAS)
%
% figure(2)
% plot(tempo,quota)
%
% figure(3)
% plot(tempo,deltae)

%find indice dal tempo
for i=inizio:1:fine
if tempo(i)==2050
casper=i;
end
end

rho_sl  = 1.225;                %livello del mare [kg/m^3]
T_sl    = 288.15;                   % Temperatura a livello del mare [K]
a       = -0.0065;              % Gradiente termico [K/m]
h = quota./3.28;

% Valutazione della temperatura e della rho al variare della quota
for  i = 1:length(h)
T_s(i)      = T_sl + a.*h(i);
rho_s(i)    = rho_sl.*(T_s(i)./T_sl).^4.25;
end
figure(1)
plot(tempo(inizio:fine),TAS(inizio:fine))
xlabel('t [s]')
```

```
ylabel('TAS [m/s]')

figure(2)
plot(tempo(inizio:fine),quota(inizio:fine)./3.28)
xlabel('t[s]')
ylabel('H [m]')

rpm=VarName9;
figure(3)
plot(tempo(inizio:fine),rpm(inizio:fine))

TAS_a=TAS(inizio:fine);
tempo_a=tempo(inizio:fine);
T_out=T(inizio:fine);
T_s=T_s(inizio:fine);
T_s=T_s';
%funzione che interpola

x=tempo_a;
v_TAS=  -0.006235.*x.^2+ 25.64.*x-2.628e+04 ;
a_TAS= -0.006235.*2.*x+ 25.64;

ROC= a_TAS.*v_TAS./9.81;
ROC_s=ROC.*T_out./T_s-(30)/70.*T_out./T_s;

V_fdc=45:1:75;

%Vx
m_vett=ROC_s./v_TAS;
i=1;
while v_TAS(i)<max(v_TAS)
m=ROC_s(i)/v_TAS(i);
if m==max(m_vett)
Vx=v_TAS(i)
end
if ROC_s(i)==max(ROC_s)
Vy=v_TAS(i)
end
i=i+1;
end

ROCfdc=[ 5.2500    5.2302    5.2028    5.1674    5.1236    5.0712    5.0100
4.9518    4.8885    4.8165    4.7355    4.6455    4.5463    4.4405
4.3393    4.2291    4.1099    3.9814    3.8438    3.6967    3.5491    3.3992
3.2402    3.0719    2.8942    2.7071    2.5105    2.3183
2.1174    1.9071    1.687];
figure(5)
plot(v_TAS,ROC)
hold on
plot(v_TAS,ROC_s)
grid on
plot(V_fdc,ROCfdc)
legend('ROC_{test}', 'ROC_{sta.}', 'ROC_{fdc}')
xlabel('TAS [m/s]')
ylabel('ROC [m/s]')
title('ROC at 1220m  5500 rpm')
```

```
%% 3000 ritorno
tempo=VarName1-VarName1(1);
TAS=VarName8*0.514444;
quota=VarName2;
deltae=VarName12;
T=VarName12+237.15;
inizio=     38561;
fine=  39681;
% figure(1)
% plot(tempo,TAS)
%
% figure(2)
% plot(tempo,quota)
%
% figure(3)
% plot(tempo,deltae)

%find indice dal tempo
for i=inizio:1:fine
if tempo(i)==2480
casper=i;
end
end

rho_sl  = 1.225;           %  livello del mare [kg/m^3]
T_sl    = 288.15;          % Temperatura a livello del mare [K]
a       = -0.0065;         % Gradiente termico [K/m]
h = quota./3.28;

% Valutazione della temperatura e della rho al variare della quota
for  i = 1:length(h)
T_s(i)      = T_sl + a.*h(i);
rho_s(i)    = rho_sl.*(T_s(i)./T_sl).^4.25;
end
figure(1)
plot(tempo(inizio:fine),TAS(inizio:fine))
xlabel('t [s]')
ylabel('TAS [m/s]')

figure(2)
plot(tempo(inizio:fine),quota(inizio:fine)./3.28)
xlabel('t[s]')
ylabel('H [m]')

rpm=VarName9;
figure(3)
plot(tempo(inizio:fine),rpm(inizio:fine))

TAS_a=TAS(inizio:fine);
tempo_a=tempo(inizio:fine);
T_out=T(inizio:fine);
T_s=T_s(inizio:fine);
T_s=T_s';
%funzione che interpola
```

```
x=tempo_a;
v_TAS=    -0.007402.*x.^2+   36.63.*x-4.525e+04  ;
a_TAS=   -0.007402.*2.*x+   36.63;

ROC= a_TAS.*v_TAS./9.81;
ROC_s=ROC.*T_out./T_s-(12)/70.*T_out./T_s;

V_fdc=35:1:67;

%Vx
m_vett=ROC_s./v_TAS;
i=1;
 while v_TAS(i)<max(v_TAS)
m=ROC_s(i)/v_TAS(i);
 if m==max(m_vett)
Vx=v_TAS(i)
end
 if ROC_s(i)==max(ROC_s)
Vy=v_TAS(i)
end
i=i+1;
end

ROCfdc=[ 5.3889     5.3690     5.3411     5.3047     5.2596     5.2055
5.1422     5.0819     5.0164     4.9418     4.8580     4.7649     4.6622
4.5528 4.4480     4.3340     4.2106     4.0779     3.9355     3.7835     3.6310
3.4761     3.3117     3.1379     2.9544     2.7612     2.5581     2.3597
2.1523     1.9352     1.7083
];
figure(5)
plot(v_TAS,ROC)
hold on
plot(v_TAS,ROC_s)
grid on
plot(V_fdc,ROCfdc)
legend('ROC_{test}', 'ROC_{sta.}', 'ROC_{fdc}')
xlabel('TAS [m/s]')
ylabel('ROC [m/s]')
title('ROC at 915m  3450 rpm')
```

Appendix F

Second flight test session data analysis

```
%% 4000 ft test andrea
tempo=VarName1-VarName1(1);
TAS=VarName8*0.514444;
quota=VarName2;
deltae=VarName12;
T=VarName12+237.15;
inizio=16284;
fine=17516;

rho_sl  = 1.225;              %  livello del mare [kg/m^3]
T_sl    = 288.15;                 % Temperatura a livello del mare [K]
a       = -0.0065;             % Gradiente termico [K/m]
h = quota./3.28;

% Valutazione della temperatura e della rho al variare della quota
for  i = 1:length(h)
T_s(i)      = T_sl + a.*h(i);
rho_s(i)    = rho_sl.*(T_s(i)./T_sl).^4.25;
end
figure(1)
plot(tempo(inizio:fine),TAS(inizio:fine))
xlabel('t [s]')
ylabel('TAS [m/s]')

figure(2)
plot(tempo(inizio:fine),quota(inizio:fine)./3.28)
xlabel('t[s]')
ylabel('H [m]')

rpm=VarName9;
figure(3)
plot(tempo(inizio:fine),deltae(inizio:fine))

% figure(1)
% plot(tempo,TAS)
% xlabel('time [s]')
% ylabel('TAS [m/s]')
% grid on
% figure(2)
% plot(tempo,quota)
% xlabel('time [s]')
% ylabel('H [m]')
% grid on
% figure(3)
% plot(tempo,deltae)

TAS_a=TAS(inizio:fine);
tempo_a=tempo(inizio:fine);
T_out=T(inizio:fine);
T_s=T_s(inizio:fine);
T_s=T_s';
%funzione che interpola

x=tempo_a;
v_TAS= -0.004401.*x.^2+9.738.*x-5318;
a_TAS=-0.004401.*2.*x+9.738;

ROC= a_TAS.*v_TAS./9.81;
ROC_s=ROC.*T_out./T_s;

V_fdc=35:1:65;

%Vx
m_vett=ROC_s./v_TAS;
i=1;
while v_TAS(i)<max(v_TAS)
m=ROC_s(i)/v_TAS(i);
if m==max(m_vett)
Vx=v_TAS(i)
end
if ROC_s(i)==max(ROC_s)
Vy=v_TAS(i)
end
i=i+1;
end

ROCfdc=[   5.2500    5.2302    5.2028    5.1674    5.1236    5.0712    5.0100
   4.9518    4.8885    4.8165    4.7355    4.6455    4.5463    4.4405
4.3393    4.2291    4.1099    3.9814    3.8438    3.6967    3.5491    3.3992
  3.2402    3.0719    2.8942    2.7071    2.5105    2.3183
2.1174    1.9071    1.6873
];
figure(5)
plot(v_TAS,ROC)
hold on
plot(v_TAS,ROC_s)
grid on
plot(V_fdc,ROCfdc)
legend('ROC_{test}', 'ROC_{sta.}', 'ROC_{fdc}')
xlabel('TAS [m/s]')
ylabel('ROC [m/s]')
title('ROC at 1220m  5500 rpm')

%% 3000m test andrea
tempo=VarName1-VarName1(1);
TAS=VarName8*0.514444;
quota=VarName2;
```

```
deltae=VarName12;
T=VarName12+237.15;
inizio=13441;
fine= 14401;

rho_sl  = 1.225;               % aria a livello del mare [kg/m^3]
T_sl    = 288.15;                 % Temperatura a livello del mare [K]
a       = -0.0065;             % Gradiente termico [K/m]
h = quota./3.28;

% Valutazione della temperatura e della rho al variare della quota
for  i = 1:length(h)
T_s(i)      = T_sl + a.*h(i);
rho_s(i)    = rho_sl.*(T_s(i)./T_sl).^4.25;
end

figure(1)
plot(tempo(inizio:fine),TAS(inizio:fine))
xlabel('t [s]')
ylabel('TAS [m/s]')

figure(2)
plot(tempo(inizio:fine),quota(inizio:fine))
xlabel('t[s]')
ylabel('H [m]')

rpm=VarName9;
figure(3)
plot(tempo(inizio:fine),deltae(inizio:fine))

%find indice dal tempo
for i=inizio:1:fine
if tempo(i)==900
casper=i;
end
end

TAS_a=TAS(inizio:fine);
tempo_a=tempo(inizio:fine);
T_out=T(inizio:fine);
T_s=T_s(inizio:fine);
T_s=T_s';
%funzione che interpola

x=tempo_a;
v_TAS=  -0.007831.*x.^2+14.31.*x-6466;
a_TAS= -0.007831.*2.*x+14.31;

ROC= a_TAS.*v_TAS./9.81;
ROC_s=ROC.*T_out./T_s-(120/3.28)/60.*T_out./T_s;%correzione su temperatura e quota

V_fdc=30:1:65;
%Vx
m_vett=ROC_s./v_TAS;
i=1;
while v_TAS(i)<max(v_TAS)
```

```
m=ROC_s(i)/v_TAS(i);
if m==max(m_vett)
Vx=v_TAS(i)
end
if ROC_s(i)==max(ROC_s)
Vy=v_TAS(i)
end
i=i+1;
end

ROCfdc=[ 5.4466    5.4331    5.4294    5.4299    5.4370    5.4424    5.4439
   5.4402    5.4301    5.4130    5.3928    5.3728    5.3448    5.3083
5.2630    5.2088    5.1453    5.0850    5.0194    4.9447    4.8608    4.7676
   4.6648    4.5553    4.4504    4.3364    4.2130    4.0801
3.9377    3.7856    3.6330    3.4780    3.3136    3.1397    2.9562    2.7629];

figure(5)
plot(v_TAS,ROC)
hold on
plot(v_TAS,ROC_s)
grid on
plot(V_fdc,ROCfdc)
legend('ROC_{test}', 'ROC_{sta.}', 'ROC_{fdc}')
xlabel('TAS [m/s]')
ylabel('ROC [m/s]')
title('ROC at 3000 ft 5500rpm')

%% 6000 m 1
tempo=VarName1-VarName1(1);
TAS=VarName8*0.514444;
quota=VarName2;
deltae=VarName12;
T=VarName12+237.15;
inizio= 21281;
fine=22241;%26768;
rpm=VarName9;

rho_sl  = 1.225;              %  livello del mare [kg/m^3]
T_sl    = 288.15;                 % Temperatura a livello del mare [K]
a       = -0.0065;            % Gradiente termico [K/m]
h = quota./3.28;

% Valutazione della temperatura e della rho al variare della quota
for  i = 1:length(h)
T_s(i)      = T_sl + a.*h(i);
rho_s(i)    = rho_sl.*(T_s(i)./T_sl).^4.25;
end

figure(1)
plot(tempo(inizio:fine),TAS(inizio:fine))

figure(2)
plot(tempo(inizio:fine),quota(inizio:fine))
rpm=VarName9;
figure(3)
plot(tempo(inizio:fine),deltae(inizio:fine))
```

```
%find indice dal tempo
for i=inizio:1:fine
if tempo(i)==1390
casper=i;
end
end

TAS_a=TAS(inizio:fine);
tempo_a=tempo(inizio:fine);
T_out=T(inizio:fine);
T_s=T_s(inizio:fine);
T_s=T_s';
%funzione che interpola

x=tempo_a;
v_TAS=   -0.009448.*x.^2+26.25.*x-1.817e+04 ;
a_TAS= -0.009448.*2.*x+26.25;

ROC= a_TAS.*v_TAS./9.81;
ROC_s=ROC.*T_out./T_s-(140/3.28)/60.*T_out./T_s;

%Vx
m_vett=ROC_s./v_TAS;
i=1;
while v_TAS(i)<max(v_TAS)
m=ROC_s(i)/v_TAS(i);
if m==max(m_vett)
Vx=v_TAS(i)
end
if ROC_s(i)==max(ROC_s)
Vy=v_TAS(i)
end
i=i+1;
end
V_fdc=30:1:60;

ROCfdc=[   5.2171     5.1519     5.1115     5.0854     5.0723     5.0628
 5.0535     5.0422     5.0273     5.0074     4.9860     4.9656     4.9386     4.9045
4.8628     4.8133     4.7557     4.7011     4.6420     4.5746     4.4990     4.4149
4.3222     4.2233     4.1288     4.0259     3.9145     3.7944  3.6656     3.5280
3.3899];

figure(5)
plot(v_TAS,ROC)
hold on
plot(v_TAS,ROC_s)
grid on
plot(V_fdc,ROCfdc)
legend('ROC_{test}', 'ROC_{sta.}', 'ROC_{fdc}')
xlabel('TAS [m/s]')
ylabel('ROC [m/s]')
title('ROC at 6000 ft 5500rpm')

%% 6000 m 2
tempo=VarName1-VarName1(1);
```

```
TAS=VarName8*0.514444;
quota=VarName2;
deltae=VarName12;
T=VarName12+237.15;
inizio=23841;
fine=24721;

rho_sl  = 1.225;                  % livello del mare [kg/m^3]
T_sl    = 288.15;                     % Temperatura a livello del mare [K]
a       = -0.0065;                % Gradiente termico [K/m]
h = quota./3.28;

% Valutazione della temperatura e della rho al variare della quota (fino a tropopausa)
for  i = 1:length(h)
T_s(i)      = T_sl + a.*h(i);
rho_s(i)    = rho_sl.*(T_s(i)./T_sl).^4.25;
end
figure(1)
plot(tempo(inizio:fine),TAS(inizio:fine))
xlabel('t [s]')
ylabel('TAS [m/s]')

figure(2)
plot(tempo(inizio:fine),quota(inizio:fine))
xlabel('t [s]')
ylabel('H [m]')

rpm=VarName9;
figure(3)
plot(tempo(inizio:fine),rpm(inizio:fine))

%find indice dal tempo
for i=inizio:1:fine
if tempo(i)==1545
casper=i;
end
end

TAS_a=TAS(inizio:fine);
tempo_a=tempo(inizio:fine);
T_out=T(inizio:fine);
T_s=T_s(inizio:fine);
T_s=T_s';
%funzione che interpola

x=tempo_a;
v_TAS= -0.01015 .*x.^2+31.27.*x-2.403e+04;
a_TAS=-0.01015 .*2.*x+31.27;

ROC= a_TAS.*v_TAS./9.81;
ROC_s=ROC.*T_out./T_s-(30/3.28)/55.*T_out./T_s;

V_fdc=30:1:60;

ROCfdc=[   5.2171    5.1519    5.1115    5.0854    5.0723    5.0628
 5.0535    5.0422    5.0273    5.0074    4.9860    4.9656    4.9386    4.9045
4.8628      4.8133      4.7557      4.7011      4.6420      4.5746      4.4990      4.4149
4.3222      4.2233      4.1288      4.0259      3.9145      3.7944 3.6656      3.5280
3.3899];

figure(5)
plot(v_TAS,ROC)
hold on
plot(v_TAS,ROC_s)
grid on
plot(V_fdc,ROCfdc)
legend('ROC_{test}', 'ROC_{sta.}', 'ROC_{fdc}')
xlabel('TAS [m/s]')
ylabel('ROC [m/s]')
title('ROC at 6000 ft 5500rpm')
```

Bibliography

[1] *BLACKSHAPE PILOT'S OPERATING HANDBOOK AND AIRPLANE FLIGHT MANUAL.* 2015.

[2] David A. Cook. "How to Perform Credible Verification , Validation , and Accreditation for Modeling and Simulation". In: 2005.

[3] Agostino De Marco and Domenico P Coiro. "Dinamica e simulazione di volo". In: (2011).

[4] A t foaaamiGH-KXTatart. "DoD Modeling and Simulation (M&S) Glossary". In: ().

[5] R. Thomas Galloway. "MODEL VALIDATION TOPICS FOR REAL TIME SIMULATOR DESIGN COURSES". In.

[6] P. Gili. "Dispense per il corso di Meccanica del Volo". 2020.

[7] BRP-Powertrain GmbH&Co. "Operators Manual for ROTAX® Engine Type 912 Series". In: *OM-912* (2013).

[8] Type Certificate Holder. "TYPE-CERTIFICATE DATA SHEET". In: *RED* 3 (2016), p. 102.

[9] Bas van Lierop. "Handling qualities criteria for training effectiveness assessment of the BS115 aircraft". In: (2017).

[10] Robert C Nelson et al. *Flight stability and automatic control.* Vol. 2. WCB/McGraw Hill New York, 1998.

[11] MO Rauw. "FDC 1.2-A Simulink Toolbox for Flight Dynamics and Control Analysis". In: *Haarlem, The Netherlands* (2001).

[12] Daniel Raymer. *Aircraft design: a conceptual approach.* American Institute of Aeronautics and Astronautics, Inc., 2012.

[13] Jan Roskam. *Airplane design.* DARcorporation, 1985.

[14] Brian L Stevens, Frank L Lewis, and Eric N Johnson. *Aircraft control and simulation: dynamics, controls design, and autonomous systems.* John Wiley & Sons, 2015.

[15] EURO FLIGT TEST. "Climb performance".

[16] Zhongshi Wang and Axel Lehmann. "Verification and validation of simulation models and applications: a methodological approach". In: *Recent Advances in Modeling and Simulation Tools for Communication Networks and Services.* Springer, 2007, pp. 227–240.

[17] John E Williams and Steven R Vukelich. *The USAF stability and control digital dATCOM. Volume I. Users manual.* Tech. rep. MCDONNELL DOUGLAS ASTRONAUTICS CO ST LOUIS MO, 1979.

Acknowledgements

Diego and I we would like to thank our supervisor prof. Manuela Battipede for giving us this opportunity and for the help provided during the course of the project.

We would like to thank the company EURO FLIGHT TEST, in particular Peter Hemmert and Rolf Hellbutsch, for giving us this opportunity and for all the support they have given us.

We would like to thank the prof. Agostino De Marco, for the important contribution in the use of the Digital DATCOM software.

We would like to thank the company MT-Propeller Entwicklung for providing us with important data.

Finally we would like to thank all the friends and relatives that supported us during our studies.