

DELFT UNIVERSITY OF TECHNOLOGY

SIMULATION, VERIFICATION & VALIDATION
GROUP B43

26.03.2020

Flight Test Report

Authors:

Luc Sträter 4672224
Jonas Appels 4682637
Yuri Laar 4659031
Malte Wegener 4672194
Alberto Van Meenen 4674030
Matteo Manieri 4565207

	Time (h)					
Work package	4672224	4682637	4659031	4672194	4674030	4565207
Report overview	4	0	0	0	4	4
Stationary measurements	26	0	24	0	16	10
Numerical model	0	24	0	24	0	18
Verification	16	20	16	4	24	18
Validation	4	8	8	22	0	0
Total time (h)	50	52	48	50	42	50

Contents

1	Introduction	2
2	Problem Analysis	2
2.1	Assumptions	2
2.2	Reference Frame	2
2.3	Equations of Motion	3
2.4	State Space Model	4
2.5	Stability and Control Coefficients	5
3	Stationary Measurements	6
3.1	First Measurement	7
3.1.1	Equivalent airspeed	7
3.1.2	Lift coefficient	8
3.1.3	Thrust	9
3.1.4	Drag coefficient	9
3.1.5	Output	10
3.2	Data Reduction	10
3.3	Second Measurements	11
3.3.1	Assumptions	12
3.3.2	Center of Gravity Changes	12
3.3.3	Elevator Effectiveness	12
3.3.4	Longitudinal Stability	13
4	Verification	15
4.1	Unit Testing	15
4.1.1	First Stationary Measurement	15
4.1.2	Second Stationary Measurement	16
4.2	Analysing Eigenmodes and Natural Frequencies	17
4.3	Analytic derivation of Eigenvalues	20
4.4	Model Response to Inputs and Disturbances	23
4.4.1	Response to Control Inputs	23
4.4.2	Response to Disturbances	25
5	Validation	27
5.1	Simulated vs Measured Responses	27
5.2	Eigenmotions and Eigenvalues	30
5.3	Model Matching	31
6	Conclusion	31
	References	32
A	Appendix B	33
B	Matlab Code	34

1 Introduction

This report outlines the steps and processes performed in building a simulation model that simulates the flight dynamics and stability of the Cessna Citation. Being the flight dynamics and stability team it is essential that the simulation model created properly simulates the necessary aircraft parameters. To ensure that the model created works properly the simulation model is verified and validated. If the model has been properly verified and validated it will be used to predict the static and dynamic stability properties of the aircraft.

The report starts with Chapter 2 where the general information on the simulation model is presented. In the chapter the assumptions are laid down, the reference frame is identified and the equations of motion are derived. Additionally, the state space model is presented and so are the characteristic motion parameters.

Once the general information is clear, Chapter 3 explains how the simulation model is created using the stationary measurements. It shows how the first set and second set of available measurements was used. Additionally, the chapter shows what steps were taken to reduce the data so it is comparable to other data.

Once the simulation model has been created it needs to be verified. This process is shown in done with the available reference data and is shown in Chapter 4. It identifies the methods of unit testing performed. The chapter discusses the results of the analysis on the eigenmodes and natural frequencies and gives an analytic derivation of the eigenvalues. On top of that it also shows the model's response to inputs and disturbances.

The validation process is explained and shown in Chapter 5. This chapter shows how the flight test data has been used to validate the simulation model, by means of comparing eigenmotions and eigenvalues and model matching.

2 Problem Analysis

In this chapter the governing equations on which the numerical simulation is based are explained. First some major assumptions on which the model leans are given in Section 2.1. Than in the Section 2.2 the reference frame is stated so that the equations of motion can be given in Section 2.3. In Section 2.4 the method to convert this system to a state space system is given and lastly in Section 2.5 an explanation on how the characteristic motion parameters are obtained is given.

2.1 Assumptions

1. The constructed model is a linearised model, so it does not take into account effects from higher order derivatives.
2. The international standard atmosphere is used to perform data reduction ($\rho_0 = 1.225kg/m^3$, $p_0 = 101325Pa$ and $T_0 = 288.15K$).
3. The air is assumed to have constant gas properties ($R = 287J/kg$ and $\gamma = 1.4$).

2.2 Reference Frame

For all the calculations in this report a body-fixed reference frame is used and specifically the stability reference frame. This is done since this is the easiest for deriving the equations of motions and it is analogous to Mulder et al. which will be cited a number of times in the report [3]. A graphical representation of this reference is given in Figure 1. The stability reference frame is a right-handed orthogonal axis system with the origin at the center of mass of the aircraft.

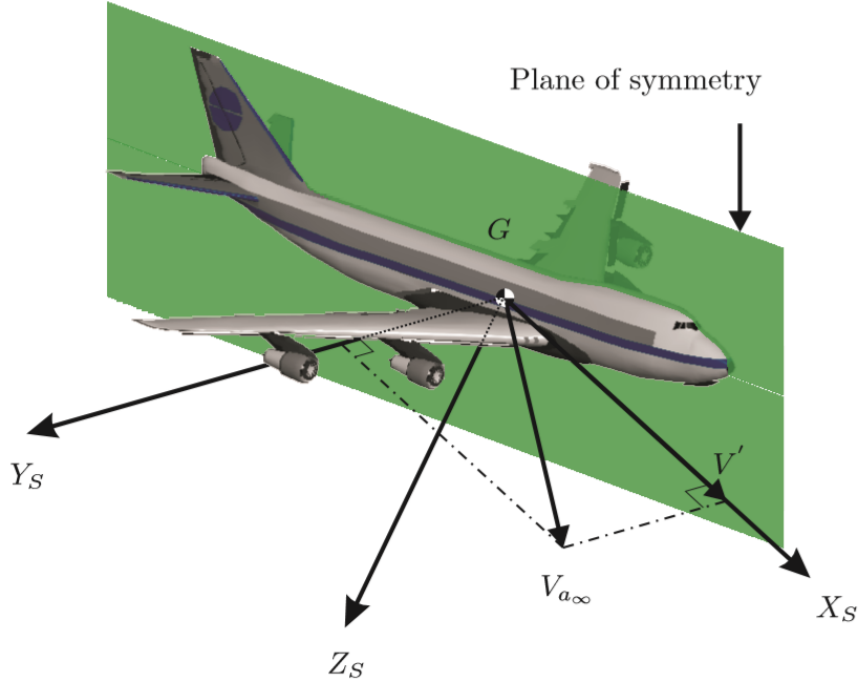


Figure 1: Stability reference frame [3]

2.3 Equations of Motion

This section discusses the equilibrium equations for the horizontal, stationary and symmetric flight. In a non-dimensional form the equilibrium equations are given by Equations 1 to 3 (taken from [3]). Where C_T and C_N are the tangential and normal force coefficients and C_m is the moment coefficient around the Y-axis.

$$C_T = C_{T_w} = -\frac{W}{\frac{1}{2}\rho V^2 S} \sin \theta \quad (1)$$

$$C_N = C_{N_w} + C_{N_h} \left(\frac{V_h}{V} \right)^2 \frac{S_h}{S} = \frac{W}{\frac{1}{2}\rho V^2 S} \cos \theta \quad (2)$$

$$C_m = C_{m_{ac}} + C_{N_w} \frac{x_{c.g.} - x_w}{\bar{c}} + C_{N_h} \left(\frac{V_h}{V} \right)^2 \frac{S_h}{S} \frac{x_{c.g.} - x_h}{\bar{c}} = 0 \quad (3)$$

The assumptions made to arrive to these equations are the following [3].

1. The propulsive force is aligned with the X-Axis ($i_p = 0$)
2. Neglecting terms with T_c and C_{N_p} , which are basically the propulsion effects.
3. $C_{T_h} \approx 0$, which means that the drag of the horizontal stabiliser is neglected since for small angles of attack $C_{T_h} \approx C_{D_h}$.
4. C_{T_w} does not create a moment around the c.g. since its moment arm is usually very small.
5. $C_{m_{ac_h}} \approx 0$ since compared to $C_{m_{ac_w}}$ it is very small or even 0 if the airfoil is symmetric.

This leads to the symmetric equations of motion given as a matrix system in Equation 4 and the asymmetric equations of motion as given by Equation 5. The derivations are analogues to those given by Mulder et al. [3].

$$\begin{bmatrix} C_{x_u} - 2\mu_c D_c & C_{x_c} & C_{z_0} & C_{x_i} \\ C_{z_x} & C_{z_s} + (C_{z_s} - 2\mu_c) D_c & -C_{x_0} & C_{z_t} + 2\mu_c \\ 0 & 0 & -D_c & 1 \\ C_{m_s} & C_{m_s} + C_{m_s} D_c & 0 & C_{m_t} - 2\mu_c K_s^2 D_c \end{bmatrix} \cdot \begin{bmatrix} \hat{u} \\ \alpha \\ \theta \\ \frac{q\bar{c}}{V} \end{bmatrix} = \begin{bmatrix} -C_{x_t} \\ -C_{z_t} \\ 0 \\ -C_{m_t} \end{bmatrix} \cdot \delta_e \quad (4)$$

$$\begin{bmatrix} C_{Y_p} + (C_{Y_p} - 2\mu_b) D_b & C_L & C_{Y_p} & C_{Y_\tau} - 4\mu_b \\ 0 & -\frac{1}{2}D_b & 1 & 0 \\ C_{\ell_p} & 0 & C_{\ell_p} - 4\mu_b K_{x\alpha}^2 D_b & C_q + 4\mu_b K_{xz} D_b \\ C_{n_\beta} + C_{n_\beta} D_b & 0 & C_{n_p} + 4\mu_b K_{xz} D_b & C_{n_r} - 4\mu_b K_{zz}^2 D_b \end{bmatrix} \begin{bmatrix} \beta \\ \varphi \\ \frac{pb}{2V} \\ \frac{rb}{2V} \end{bmatrix} = \begin{bmatrix} -C_{y_{k_z}} \\ 0 \\ -C_{t_{k_k}} \\ -C_{n_{k_k}} \end{bmatrix} \cdot \delta_a + \begin{bmatrix} -C_{y_s} \\ 0 \\ -C_{k_k} \\ -C_{m_k} \end{bmatrix} \cdot \delta_r \quad (5)$$

2.4 State Space Model

The state space model can be set up in the same way for the symmetric and the asymmetric equations of motion. Starting with the equations of motion in non-dimensional form, where C is the 4x4 matrix of Equation 4 or 5 respectively and \mathbf{x} is the vector of non-dimensional state variables. G is the 4x1 matrix of Equation 4 or the two 4x1 matrices of Equation 5 combined to a 4x2 matrix and \mathbf{u} is the vector of input variables.

$$C\mathbf{x} = G\mathbf{u} \quad (6)$$

Let $\bar{\mathbf{x}}$ be the vector of dimensional state variables, for which holds

$$\bar{\mathbf{x}} = M\mathbf{x} \quad (7)$$

For the symmetric case M is given by (8) and for the asymmetric case by (9).

$$M_s = \begin{bmatrix} V & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{V}{c} \end{bmatrix} \quad (8)$$

$$M_a = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{2V}{b} & 0 \\ 0 & 0 & 0 & \frac{2V}{b} \end{bmatrix} \quad (9)$$

Where M is diagonal and non-singular. From this the dimensional form of the EOM can be deduced to be

$$CM^{-1}M\mathbf{x} = G\mathbf{u} \quad (10)$$

$$CM^{-1}\bar{\mathbf{x}} = G\mathbf{u} \quad (11)$$

As C contains differential Operators, the Matrix has to be split, in order to form a first order system of differential equations.

$$C_1 + C_2 = CM^{-1} \quad (12)$$

Where C_1 collects all products that include a differential Operator. As all differential operators are of the same order and same (What is it called?)

$$C_1\mathbf{x} = C_1 \frac{d}{dt} \mathbf{x} = C_1 \dot{\mathbf{x}} \quad (13)$$

From this the system of ODE's can be formed

$$C_1 \dot{\mathbf{x}} + C_2 \mathbf{x} = G\mathbf{u} \quad (14)$$

As a state space is defined as

$$\dot{\mathbf{x}} = A\mathbf{x} + G\mathbf{u} \quad (15)$$

Equation 14 has to be rearranged

$$\dot{\mathbf{x}} = -C_1^{-1}C_2\mathbf{x} + C_1^{-1}G\mathbf{u} \quad (16)$$

Thus $A = -C_1^{-1}C_2$ and $B = C_1^{-1}G$

2.5 Stability and Control Coefficients

In order to make a good flight model, accurate stability derivatives are required. As aircraft manufacturers do not provide you these values and stability derivatives change as flight conditions change, they have to be modelled for steady flight and small deviations. The equations of motions will be linearized and the stability coefficients can be calculated for small oscillations. This chapter will briefly explain the determination of all stability derivatives that are recalculated in the code. Other stability derivatives are determined or taken from the provided model (appendix B).

Stability Coefficient Assumptions:

1. All stability coefficients, apart from C_L, C_{X_0} and C_{Z_0} are assumed to be independent of airspeed and altitude.
2. The variation of Reynoldsnumber with airspeed is neglected
3. The variation of the aircraft's aeroelastic deformation with airspeed is neglected.
4. Compressibility effects are neglected as the aircraft does not reach Mach 0.6 which requires to take it into account[3].
5. Thrust is independent of Airspeed.
6. A parabolic polar Drag curve
7. For C_m tangential forces, thrust and slipstream effects are neglected.

Derivatives airspeed

Stability derivative C_{X_u} can be described as equation 17[3]. Combining this with the definition of X ($X = C_X \frac{1}{2} \rho V^2 S$), the fact that C_X can be described as equation 18 ("The partial derivative with respect to airspeed is determined for deviations from the steady, initial flight condition in airspeed only"[3], thus α is 0) and $C_X = C_{X_0}$ for the same reason, gives equation 19.

$$C_{X_u} = \frac{1}{\frac{1}{2} \rho V S} \frac{\partial X}{\partial V} \quad (17)$$

$$C_X = -C_D + T'_c \quad (18)$$

$$C_{X_u} = 2C_{X_0} + \left(\frac{\partial T'_c}{\partial V} - \frac{\partial C_D}{\partial V} \right) V \quad (19)$$

In a similar way it can be shown that C_{Z_u} is equal to equation 20.

$$C_{Z_u} = 2C_{Z_0} + \left(-\frac{\partial C_L}{\partial V} - \frac{\partial T'_c}{\partial V} (\alpha_0 + i_p) \right) V \quad (20)$$

With the underlying condition of initial, steady flight (equation 21).

$$C_Z = -C_L - T'_c (\alpha_0 + i_p) \quad (21)$$

C_{m_u} again follows the derivation of C_{X_u} .

$$C_{m_u} = 2C_{m_0} + \frac{\partial C_m}{\partial V} V \quad (22)$$

As in steady flight $C_m = 0$, the first part of the equation falls away.

$$C_{m_u} = \frac{\partial C_m}{\partial V} V \quad (23)$$

The next step is to come up with all the partial derivatives still present in the formulas. As compressibility is not taken into account, it is not derived here. For a jet engine thrust can assumed to be independent of airspeed ($T_p = T'_c 0.5 \rho V^2 S = \text{constant}$) leading to equation 24.

$$\frac{dT'_c}{dV} = -\frac{2T'_c}{V} \quad (24)$$

For level flight $T=D$, so $T'_c = C_D$ and $C_{X_0} \approx C_L \sin \theta_0$. This simplifies equation 17 to equation 25. In the same way equations 26 and are established.

$$C_{x_u} = 2C_L \tan \theta_0 \quad (25)$$

$$C_{z_u} = 2C_L \cos \theta_0 \quad (26)$$

$$C_{m_u} = -2C_D \frac{\partial C_m}{\partial T'_e} \quad (27)$$

Derivatives angle of attack

Angle of attack derivatives are directly calculated in the code. C_{X_α} assumes a parabolic polar curve ($C_D = C_{D0} + \frac{C_L^2}{\pi A e}$) together with force equilibrium (equation 28). With the initial condition of 0 Angle of Attack and and T'_c as a constant taking the derivative leads to equation 29.

$$C_X = C_L \sin \alpha - C_D \cos \alpha + T'_c \quad (28)$$

$$C_{X_\alpha} = C_L \left(1 - \frac{2C_{L_\alpha}}{\pi A e} \right) \quad (29)$$

C_{Z_α} is constructed in a similar way with Z-force equilibrium (equation 30). It is assumed $C_D \ll C_{L_\alpha}$ and with the conditions described, differentiating the force equilibrium leads to equation 31 .

$$C_Z = -C_L \cos \alpha - C_D \sin \alpha - T'_c (\alpha_0 + i_p) \quad (30)$$

$$C_{Z_\alpha} = -C_{L_\alpha} - C_D \approx -C_{L_\alpha} \approx -C_{N_\alpha} \quad (31)$$

For C_m the tangential forces, thrust and slipstream effects are neglected (equation 32). Differentiating leads to equation 33.

$$C_m = C_{m_{ac}} + C_{N_{w_\alpha}} (\alpha - \alpha_0) \frac{x_{c,g} - x_w}{c} - \left[C_{N_{h_\alpha}} \left\{ (\alpha - \alpha_0) \left(1 - \frac{d\epsilon}{d\alpha} \right) + (\alpha_0 + i_h) \right\} + C_{N_{h_\delta}} \delta_e \right] \left(\frac{V_h}{V} \right)^2 \frac{S_h l_h}{S c} \quad (32)$$

$$C_{m_\alpha} = C_{N_{w_\alpha}} \frac{x_{c,g} - x_w}{c} - C_{N_{h_\alpha}} \left(1 - \frac{d\epsilon}{d\alpha} \right) \left(\frac{V_h}{V} \right)^2 \frac{S_h l_h}{S c} \quad (33)$$

All other stability and control derivatives are initially taken from the provided numerical simulation (Appendix B) and later possibly optimized in Validation. For further derivations of these coefficients please refer to the Flight Dynamics reader [3].

3 Stationary Measurements

The stationary measurements will yield basic aircraft properties of interest for the numerical model such as the lift curve slope, Oswald efficiency factor, zero lift drag coefficient, elevator effectiveness and the longitudinal stability coefficient. In the different sections of this chapter these coefficients will be calculated. For all the coefficients both data from the flight test and from the reference data were used to confirm that all sensors worked properly. If everything works well these should result in approximately the same coefficient as the same test are done with the same plane and configuration. In Table 1 the weight of the passengers is given. For the flight 4100 pounds of block fuel was taken and for the reference data flight 4050 pounds of block fuel. Note that all numbers are obtained for the aircraft in cruise configuration (gear down and flaps retracted).

Table 1: Passenger Weights

Position:	Flight data: mass [kg]	Reference data: mass [kg]
Pilot 1	80	95
Pilot 2	102	92
Co-ordinator	60	74
Observer 1L	71	66
Observer 1R	77	61
Observer 2L	76	75
Observer 2R	64	78
Observer 3L	74	86
Observer 3R	99	68

3.1 First Measurement

In the first stationary measurement the lift and drag properties are inspected by alternating the angle of attack, but importantly keeping the altitude similar. Keeping the altitude similar ensures that the plane can be modelled in horizontal flight which is use full for computations. By doing this the $C_{L\alpha}$ and the $C_L - C_D$ curves can be drawn from which the lift curve slope, Oswald efficiency factor and zero lift drag can be found for the numerical model. In Table 2 and 3 the results for respectively the test flight data and the reference flight data are given. Note that some processing of this data has to be done to get the necessary coefficients which will be done in the following subsections.

Table 2: Stationary measurements CL-CD flight data

Nr:	Time [hrs:min:sec]	hp [ft]	IAS [kts]	a [deg]	FFI [lbs/hr]	FFr [lbs/hr]	F. Used [lbs]	TAT [C]
1	0:18:50	5030	251	1.6	770	806	36 7	10.2
2	0:20:58	5030	221	2.4	638	670	422	7.8
3	0:23:18	5020	190	3.7	533	579	456	5
4	0:25:30	5040	161	5.6	444	488	492	4
5	0:27:44	5040	134	8.5	420	450	521	2.2
6	0:29:23	5030	121	10.4	420	450	545	1.5

Table 3: Stationary measurements CL-CD reference data

Nr:	Time [hrs:min:sec]	hp [ft]	IAS [kts]	a [deg]	FFI [lbs/hr]	FFr [lbs/hr]	F. Used [lbs]	TAT [C]
1	00:19:17	5010	249	1.7	798	813	360	12.5
2	00:21:37	5020	221	2.4	673	682	412	10.5
3	00:23:46	5020	192	3.6	561	579	447	8.8
4	00:26:04	5030	163	5.4	463	484	478	7.2
5	00:29:47	5020	130	8.7	443	467	532	6
6	00:32:00	5110	118	10.6	474	499	570	5.2

3.1.1 Equivalent airspeed

Before any of the coefficients can be calculated first the equivalent airspeed has to be obtained from the data. The measurement data gives the speed as an indicated airspeed, but for further calculations the equivalent airspeed is needed. The derivation begins with finding the the pressure and density at the altitude the aircraft is flying by Equation 34 and Equation 35. In these equations only the h_p (in meters) is filled in from the measurement while the rest are assumed to be those of the international standard atmosphere.

$$p = p_0 \left[1 + \frac{\lambda h_p}{T_0} \right]^{-\frac{g_0}{\lambda R}} \quad (34)$$

$$\rho = \frac{p}{R \cdot T} \quad (35)$$

With this data the Mach number can be calculated using Equation 36. In this formula the calibrated airspeed has to be put in, but the indicated airspeed is what was measured during the flight. To convert indicated airspeed to calibrated airspeed a conversion table was used [1].

$$M = \sqrt{\frac{2}{\gamma - 1} \left[\left(1 + \frac{p_0}{p} \left\{ \left(1 + \frac{\gamma - 1}{2\gamma} \frac{\rho_0}{p_0} V_c^2 \right)^{\frac{\gamma}{\gamma - 1}} - 1 \right\} \right)^{\frac{\gamma - 1}{\gamma}} - 1 \right]} \quad (36)$$

When the Mach number is found this can be multiplied with the speed of sound to get the true airspeed. To find the speed of sound at every measured point Equation 38 is used. In this formula the temperature has to be filled. This value was measured, but it still had to be adjusted for the ram rise using Equation 37. Than finally the equivalent airspeed can be calculated with equation 39. Here the density is as given before in Equation 35 and true airspeed is calculated by multiplying the mach number by speed of sound.

$$T = \frac{T_m}{1 + \frac{\gamma - 1}{2} M^2} \quad (37)$$

$$a = \sqrt{\gamma \cdot R \cdot T} \quad (38)$$

$$V_{EAS} = M \cdot a \cdot \sqrt{\frac{\rho}{\rho_0}} \quad (39)$$

3.1.2 Lift coefficient

Because the plane is flying in horizontal, symmetric, steady and straight flight the lift coefficient can be calculated using Equation 40. In this equation weight and equivalent airspeed are different for every moment in the flight and the rest are constant. The weight can be calculated using Equation 41 in which the fuel weight is again variable. The flight measurement gives the fuel used for every measurement point so the mass of the fuel can be computed easily by subtracting the fuel used from the block fuel. The equivalent airspeed is computed as given in Section 3.1.1.

$$C_L = \frac{L}{0.5\rho V^2 S} \approx \frac{W}{0.5\rho_0 V_{EAS}^2 S} \quad (40)$$

$$W = (BEM + MP + MF) \cdot g_0 \quad (41)$$

With the lift coefficients calculated for every measurement point and the angle of attack from the measurement itself the lift curve can be plotted. Since the plane hasn't stalled the lift coefficient can be also represented by the linear expression given in Equation 42. However since the measurements have some noise not all the points exactly lay on this line. The over determinate system is given in Equation 43.

$$C_L = C_{L0} + C_{L\alpha} \cdot \alpha \quad (42)$$

$$\begin{pmatrix} C_{L,1} \\ C_{L,2} \\ C_{L,3} \\ C_{L,4} \\ C_{L,5} \\ C_{L,6} \end{pmatrix} = \begin{pmatrix} 1 & \alpha_1 \\ 1 & \alpha_2 \\ 1 & \alpha_3 \\ 1 & \alpha_4 \\ 1 & \alpha_5 \\ 1 & \alpha_6 \end{pmatrix} \cdot \begin{pmatrix} C_{L0} \\ C_{L\alpha} \end{pmatrix} \quad (43)$$

To make the best estimation of the lift curve slope and the zero angle of attack lift a linear regression is done for this system. This leads to a C_{L0} of 0.0807 for the test flight data and 0.0670 for the reference flight data. Simultaneously this leads to a $C_{L\alpha}$ of $4.5827 \frac{1}{rad}$ for the test flight data and $4.8109 \frac{1}{rad}$ for the reference flight data. The lift curve is also depicted graphically in Figure 2 where the actual measurements are also shown for clarity. In the figure an angle of attack range of -2 till 12 degrees was chosen, but in reality the before and after the last measurement there might occur stall.

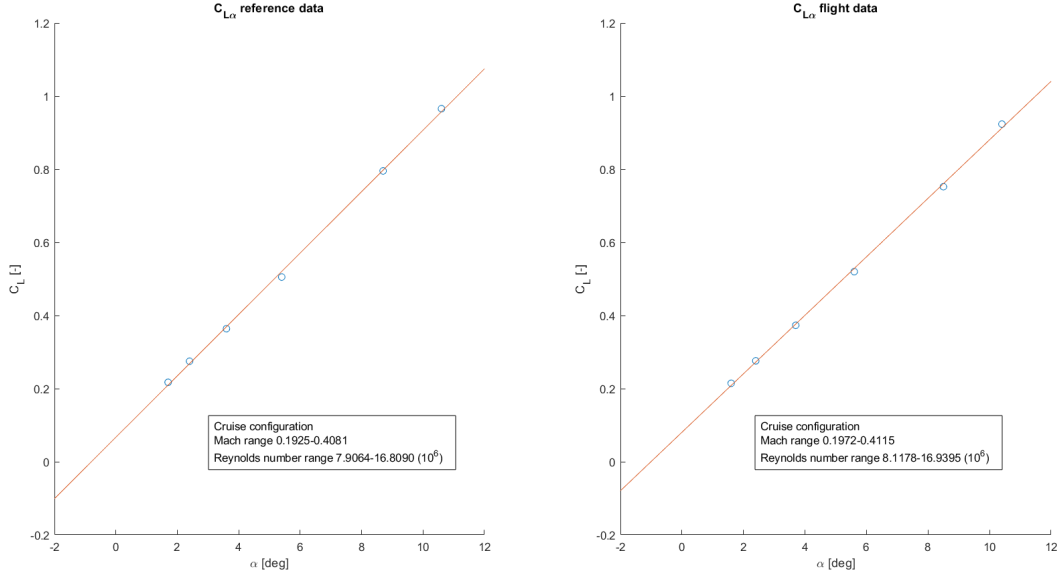


Figure 2: Lift rate curve

3.1.3 Thrust

Before the drag can be calculated first the thrust (T_p) has to be known. To do this an executable file was used which calculates the thrust as a function of the pressure altitude, mach number, fuel flow of both engines and the temperature difference between the corrected total air temperature and the temperature in the international standard atmosphere [1]. Most of these values are given in the measurement and only have to be converted to the right SI units. The mach number is calculated as in Section 3.1.1 and the temperature difference with equation 44.

$$\Delta T = \frac{T_m}{1 + \frac{\gamma-1}{2} M^2} - (T_0 + (\lambda \cdot h_p)) \quad (44)$$

3.1.4 Drag coefficient

Because the plane is in horizontal, symmetric, steady and straight flight the drag coefficient can be calculated using Equation 45. In this equation the thrust is as calculated in Section 3.1.3 and the mach number and the density as calculated in Section 3.1.1.

$$C_D = \frac{D}{0.5\rho V^2 S} \approx \frac{T_p}{0.5\rho V_{TAS}^2 S} \quad (45)$$

To obtain the zero lift drag coefficient and the oswald efficiency factor Equation 46 has to be solved for by using the measurements. Again this is an over determinate system as given in Equation 47 and a linear regression is used. This leads to a C_{D0} of 0.0220 for the test flight data and 0.0212 for the reference flight data. Simultaneously this leads to an oswald efficiency factor of 0.9670 for the test flight data and 0.7448 for the reference flight data.

$$C_D = C_{D0} + \frac{C_L^2}{\pi A e} \quad (46)$$

$$\begin{pmatrix} C_{D,1} \\ C_{D,2} \\ C_{D,3} \\ C_{D,4} \\ C_{D,5} \\ C_{D,6} \end{pmatrix} = \begin{pmatrix} 1 & C_{L,1}^2 \\ 1 & C_{L,2}^2 \\ 1 & C_{L,3}^2 \\ 1 & C_{L,4}^2 \\ 1 & C_{L,5}^2 \\ 1 & C_{L,6}^2 \end{pmatrix} \cdot \begin{pmatrix} C_{D0} \\ \frac{1}{\pi A e} \end{pmatrix} \quad (47)$$

Lastly the lift over drag curve is also depicted graphically in Figure 3. Here the lift coefficient is as calculated in Section 3.1.2.

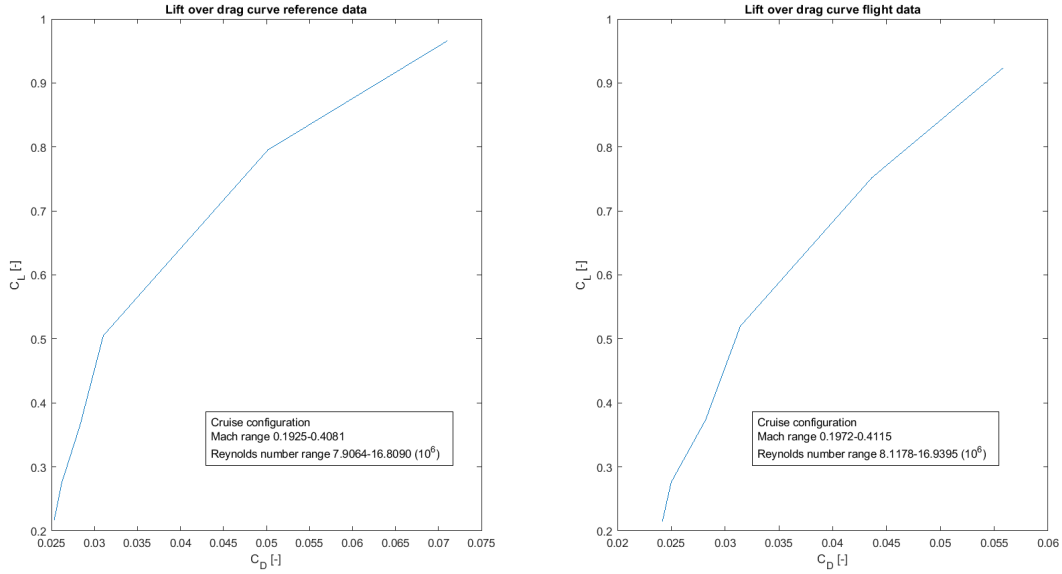


Figure 3: Lift over drag curve

3.1.5 Output

In Table 4 results of the first measurement series is stated. As can be seen the results of the test flight and the reference flight are quite different. The reason for this is that the plane is not truly in horizontal flight. For most measurements the altitude does not vary much ($<10\text{ft}$), but for the last measurement of the reference flight it does (90ft). When this point is excluded the numbers already seem much closer to each other. Since the change in altitude has such a big effect the remaining deviation between the numbers is most likely due to this inherent altitude difference and some measurement errors.

Table 4: Output First Measurements

	Reference Data	Reference Data 1:5	Test Flight Data
$C_{L\alpha}$	4.811	4.713	4.583
C_{L0}	0.0670	0.0727	0.0807
e	0.7448	0.8860	0.9670
$C_{D,0}$	0.0212	0.0225	0.0220

3.2 Data Reduction

The obtained data needs to be reduced in order to be comparable between different test flights. It is important that results (such as the elevator effectiveness $C_{m_{\delta_e}}$) are compared for equal conditions. However, test flights are carried out at different altitudes and weights. To obtain the results for those equal conditions, the effects of altitude and weight need to be removed. All results will therefore be reduced to the standard values in Table 5.

Table 5: Standard Values

	Standard Value
Aircraft Weight	60500 N
Engine Fuel Flow	0.048 kg/s
Air Density	1.225 kg/m^3

Reduction of Equivalent Airspeed The reduced equivalent airspeed is computed using equation 48. This scales the Velocity with the current weight of the aircraft to be able to compare values.

$$\tilde{V}_e = V_e \sqrt{\frac{W_s}{W}} \quad (48)$$

Stick Force Reduction The stick force is reduced by simply scaling it with weight and assuming $F_e = F_{eaer}$, thus ignoring friction.

$$F_{eaer}^* = F_{eaer} \cdot \frac{W_s}{W} \quad (49)$$

Reduction of Thrust Also the elevator deflection has to be reduced to be able to compare it. C_{m_δ} is computed from the data as shown in section 3.3.4 and $C_{m_{T_c}}$ is given.

$$\delta_{eq}^* = \delta_{e_{meas}} - \frac{1}{C_{m_\delta}} C_{m_{T_c}} (T_{c_s} - T_c) \quad (50)$$

$$T_c = \frac{T_p}{\frac{1}{2} \rho V_{TAS}^2 d^2} \quad (51)$$

$$T_{cs} = \frac{T_{p_{standard}}}{\frac{1}{2} \rho V_{TAS}^2 d^2} \quad (52)$$

Where T_p is provided by the thrust.exe file. Thrust.exe uses pressure altitude, Mach number (provided by Airspeed reduction calculations), Fuel flows and Temperature differences from ISA.

The fan diameter of an Pratt & Whitney Canada JT15D-4B engine is 686mm[2]. The difference between T_c and T_{c_s} is that T_{c_s} uses a standardized fuel flow of 0.048 kg/s.

3.3 Second Measurements

In the second stationary measurements, some elements of the stability and control properties of the aircraft are determined. The elevator trim curve and control force curve are plotted and the stick-fixed and stick-free static stability of the aircraft is explained. The goal is to determine the elevator effectiveness $C_{m_{de}}$ and the longitudinal stability C_{m_α} to be used in the numerical model. In Table 6 and 7 the measurements are given.

Table 6: Stationary Measurements Flight Data

Nr:	Time [hrs:min]	hp [ft]	IAS [kts]	a [deg]	de [deg]	detr [deg]	F N	FFI [lbs/hr]	FFr [lbs/hr]	F. Used [lbs]	TAT [C]
1	0:34	6510	161	5.4	0.1	3.3	0	454	500	636	1.8
2	0:36	6780	150	6.5	-0.4	3.3	-20	450	496	662	0.2
3	0:38	7200	140	7.4	-1	3.3	-34	440	488	697	-0.5
4	0:39	7540	130	8.6	-1.3	3.3	-45	437	484	721	-1.8
5	0:41	6900	170	4.8	0.4	3.3	23	455	500	745	1.5
6	0:42	6540	180	4	0.7	3.3	33	460	505	764	2.5
7	0:44	6050	190	3.5	0.9	3.3	64	469	515	790	4

Table 7: Stationary Measurements Reference Data

Nr:	Time [hrs:min]	hp [ft]	IAS [kts]	a [deg]	de [deg]	detr [deg]	F N	FFI [lbs/hr]	FFr [lbs/hr]	F. Used [lbs]	TAT [C]
1	0:37	6060	161	5.3	0.0	2.8	0	462	486	664	5.5
2	0:39	6350	150	6.3	-0.4	2.8	-23	458	482	694	4.5
3	0:41	6550	140	7.4	-0.9	2.8	-29	454	477	730	3.5
4	0:42	6880	130	8.5	-1.5	2.8	-46	449	473	755	2.5
5	0:45	6160	173	4.5	0.4	2.8	26	465	489	798	5.0
6	0:47	5810	179	4.1	0.6	2.8	40	472	496	825	6.2
7	0:48	5310	192	3.4	1.0	2.8	83	482	505	846	8.2

3.3.1 Assumptions

- The change in normal force coefficient ΔC_N due to a change in elevator position is neglected because the elevator only accounts for a small contribution to the total normal force.
- The change in the elevator pitch control derivative $\Delta C_{m_{de}}$ due to a change in centre of gravity location is neglected because the effect on the control derivative is small.
- The change in pitch moment due to the elevator trim tab angle is neglected, so its effect will not appear in the trim curve.

3.3.2 Center of Gravity Changes

Before the elevator effectiveness can be calculated first the center of gravity for each measurement point has to be known. For that Equation 53 can be used in which the small m denotes the mass in kilos, big M the moment in pound-inch and i the elements. For the plane this reduces to the zero fuel mass and its moment arm which are constant and the fuel mass and its arm which will change per measurement. As a datum the nose of the plane was chosen.

$$x_{cg} = \frac{\sum m_i \cdot x_{cg,i}}{\sum m_i} = \frac{\sum M_i}{\sum m_i} \quad (53)$$

The zero fuel mass consists of the basic empty mass and the payload, which in this case is just the passengers as no baggage was taken. The basic empty weight was measured by the pilots and is 9165.0 pounds with a center of gravity location of 291.65 inch from the nose. All the passenger weights for both the test flight and the reference flight are given in Table 1 and their distance to the front of the plane is given by the seating configuration [1]. The fuel weight at every measurement point can be found by subtracting the fuel used from the block fuel. To find the moment induced by that fuel weight a conversion table was used [1].

During the flight observer 3 moved from his original position to the cockpit. This induced a shift in the center of gravity. The measurement that was taken is presented in Table 9 for the test flight and Table ?? for the reference flight. Here the first measurement is when observer 3 is at his own seat and the second measurement is when he is standing in the cockpit. Note that some measurements were not done during the test flight, but those values are not necessary to obtain the center of gravity location. This leads to a shift in center of gravity which is given in Table 10.

Table 8: Shift in center of gravity measurement test flight

Nr:	Time	hp	IAS	a	de	detr	Fe	FFL	FFr	F. Used	TAT
	[hrs:min]	[ft]	[kts]	[deg]	[deg]	[deg]	[N]	[lbs/hr]	[lbs/hr]	[lbs]	[C]
1	0:47	6590	160	5.3	0.1	448	491	846	1.35		
2	0:49	6630	160	5.4	-0.5	446	490	871	1.5		

Table 9: Shift in center of gravity measurement test flight

Nr:	Time	hp	IAS	a	de	detr	Fe	FFL	FFr	F. Used	TAT
	[hrs:min]	[ft]	[kts]	[deg]	[deg]	[deg]	[N]	[lbs/hr]	[lbs/hr]	[lbs]	[C]
1	51:02	5730	161	5.3	0	2.8	0	471	493	881	5.0
2	52:46	5790	161	5,3	-0,5	2.8	-30	468	490	910	5.0

Table 10: Shift in center of gravity stationary data

	Test flight	Reference flight
Δcg	2.409 inch	1.666 inch
$\frac{\Delta cg}{MAC}$	0.0297	0.0206

3.3.3 Elevator Effectiveness

The elevator moment derivative describes how much the pitching moment changes for a unit change in elevator deflection. It can be determined by moving the centre of gravity in flight by a known amount, thus changing

the necessary elevator deflection to maintain moment equilibrium. The new elevator deflection can be measured and so the control derivative can be determined by Equation 54. The results are given in Table 11. Since the pitching moment is defined positive upwards and the elevator deflection is defined positive downwards (stick pushed forwards) this control derivative is always smaller than zero.

$$C_{m_{\delta_e}} = -\frac{1}{\Delta\delta_e} C_N \frac{\Delta x_{cg}}{\bar{c}} \quad (54)$$

Table 11: Computed elevator control derivatives for two data sets.

	Reference Data	Test Flight Data
$C_{m_{\delta_e}}$	-1.1726	-1.4227

The difference between the Reference and Test flight data is quite significant as $C_{m_{\delta}}$ should not change between flights.

3.3.4 Longitudinal Stability

In order for the aircraft to be longitudinally stable, it is required that the aircraft pitches down for any upward disturbance and vice versa. This means that for a positive pitch disturbance the aircraft needs to create a negative change in pitching moment. Therefore the derivative of the pitching moment with respect to the angle of attack needs to be negative. It can be determined by multiplying the elevator control derivative with the slope of the elevator trim curve as per equation Equation 55 [3]. The results are given in Table 12.

$$C_{m_{\alpha}} = -C_{m_{\delta}} \cdot \frac{d\delta_e}{d\alpha} \quad (55)$$

Table 12: Computed pitch moment stability derivative for two data sets.

	Reference Data	Test Flight Data
$C_{m_{\alpha}}$	-0.5621	-0.6453

Figure 4 plots the elevator deflection against the angle-of-attack using Equation 56. Note the inverted vertical axis. This is common practice because a positive elevator deflection causes a negative pitch moment. The control derivative $C_{m_{\delta_e}}$ should always be negative for that reason. A positive stick deflection (forwards) causes a positive elevator deflection (downwards) which makes the airplane pitch down (negative moment). Table 12 indeed shows negative values. Further analysing the curve, it indeed makes sense that the higher the angle-of-attack becomes, the further¹ the pilot needs to pull back on the stick to keep that certain attitude.

$$\delta_e = -\frac{1}{C_{m_{\delta_e}}} [c_{m_0} + C_{m_{\alpha}} (\alpha - \alpha_0)] \quad (56)$$

¹Further deflections don't necessarily mean pulling harder on the stick as will be discussed later.

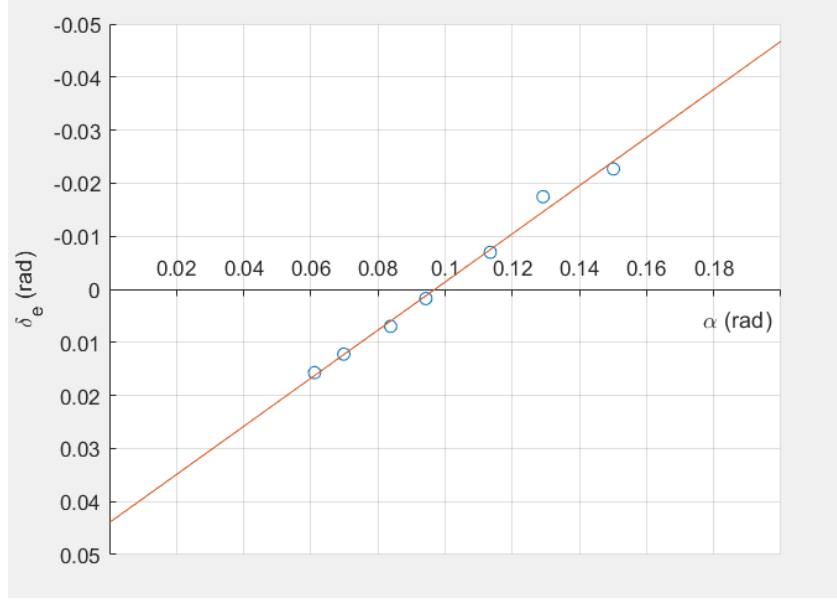


Figure 4: Elevator Trim curve (Flight data).

Equation 56 is reformulated to Equation 57 to investigate how the elevator deflection behaves for increasing velocity. This is used to plot the elevator deflection against reduced equivalent airspeed in Figure 5. The higher the velocity, the smaller the angle of attack has to be. This relation is quadratic. If the aircraft travels at twice the equivalent airspeed, the angle of attack is four times smaller assuming a linear lift curve. The figure proves that as the airspeed is increased, the deflection becomes positive. This means the aircraft moment coefficient increases with increasing airspeed and the pilot has to trim down the nose to maintain a horizontal attitude. Basically the aircraft wants to pitch up when the aircraft accelerates².

$$\delta_e = -\frac{1}{C_{m_{\delta_e}}} \left(C_{m_0} + \frac{C_{m_a}}{C_{N_a}} \frac{W}{\frac{1}{2}\rho V^2 S} \right) \quad (57)$$

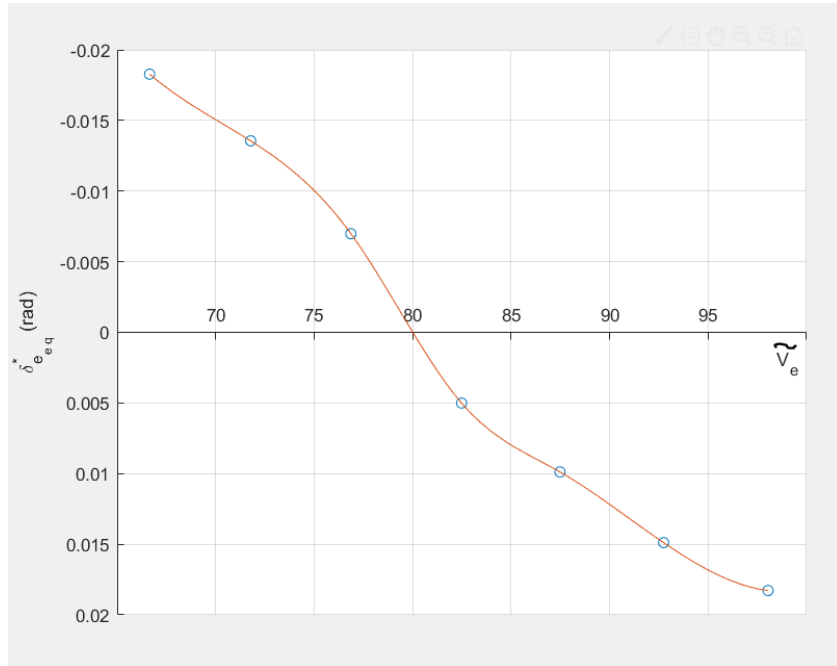


Figure 5: Elevator deflection against reduced equivalent airspeed. (Flight Data)

²Figure 5 only contains measurement points up to 100 m/s. One would expect the curve to flatten out for high airspeeds.

In order for the aircraft to be pleasant to fly, meaning the aircraft responds intuitively to stick inputs, the stick force should increase for increasing deflections. This is indeed the case as seen in Figure 6. For increasing velocity the stick force increases. This is always the case for a stable aircraft, as each longitudinally stable aircraft automatically has stable control position stability [3]. To fly at a lower speed, the pitch attitude is increased by making the elevator go up (negative direction). Then the deflection is returned a little to hold the pitch attitude.

Summarizing, the aircraft is longitudinally stable because $C_{m_\alpha} < 0$, $\frac{d\delta_{e_a}}{d\alpha} < 0$ and $\frac{d\delta_{e_a}}{dV} > 0$.

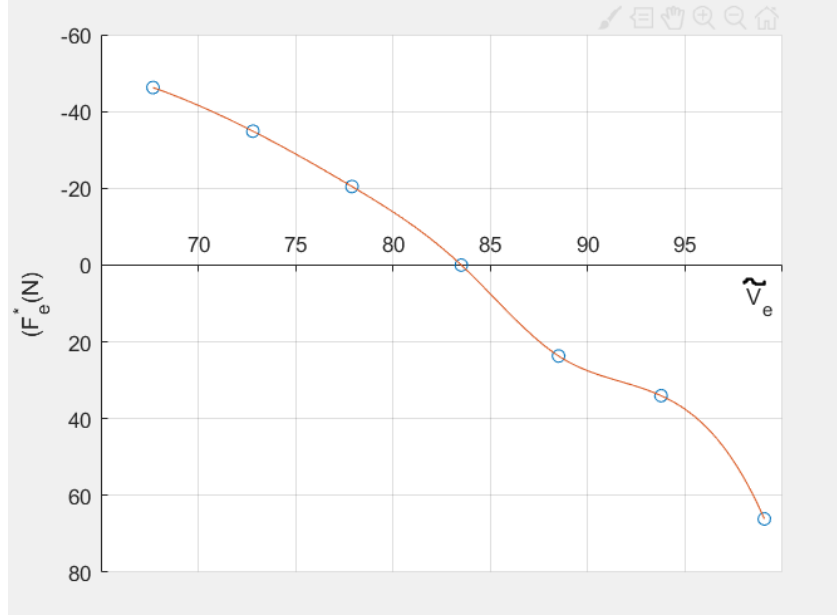


Figure 6: Stick Control Force curve (Flight Data).

4 Verification

Since the numerical simulation and the data processing of the stationary measurements are done using software (e.g. Matlab) it is key to perform a verification process. This process has to ensure that the model simulates what it has to simulate. Verification has to be done on different levels to make sure all mistakes are ironed out. The first step is making sure all the functions in the program return logical values, which can be done by unit testing. By setting up unit tests constantly while writing the code bugs are caught early on while also making the process much faster. Then when all functions are written system level tests can be made to check if the model gives logical responses. By writing the code in such a way that its functionalities are easily testable the verification process was again sped up. In Section 4.1 the unit testing is explained and the results that it gives. Then in Section 4.2 the numerical simulation is compared with an analytical solution for a specific case. Lastly in Section 4.4 it is checked if the model responds correctly to some disturbances.

4.1 Unit Testing

As stated before for the unit tests to be effectively implemented all functions should be written in such a way that they are easily verifiable. The values with which the code is compared are calculated by hand.

4.1.1 First Stationary Measurement

To unit tests the functions from the CLCD.m file, which processes the first stationary measurement, all calculations are done by hand in an excel file for the first measurement point of the test flight data. The data for the reference flight data is processed by exactly the same code so just testing for one of the two suffices. All the calculations are done according to the method described in Section 3.1. For closer look on how the values are computed the reader is referred to the excel file. In Table 13 the results of the unit test are given. In this table the W_{loc} function computes the weight; M computes the mach number; Thrustcalc computes the thrust for both engines using the executable file provided as explained in Section 3.1.3; VE calculates the equivalent airspeed; VT computes and array with the true airspeed and the density; cd calculates the drag coefficient; cl

calculates the lift coefficient. As can be seen all the errors are way smaller than the errors of the measurements and thus the functions used for the data processing of the first stationary measurement are verified. Note that a zero in the table means that the code outputs exactly the same value as the calculation in excel.

Table 13: Unit test first measurement of test flight data

	Expected result	Tolerance	Result
$\mathbf{W_loc}(i_1, i_2, i_3, i_4)$	6553.429616 <i>kg</i>	5 scientific decimal spaces	pass
$\mathbf{M}(i_1)$	0.411507997	4 scientific decimal spaces	pass
$\mathbf{Thrustcalc}(i_1, i_2, i_3, i_4, i_5)$	[3494.28 <i>N</i> 3745.81 <i>N</i>]	0	pass
$\mathbf{VE}(i_1)$	127.6582268 $\frac{m}{s}$	6 scientific decimal spaces	pass
$\mathbf{VT}(i_1)$	[136.5683881 $\frac{m}{s}$ 1.070368542 $\frac{kg}{m^3}$]	7 scientific decimal spaces	pass
$\mathbf{cd}(i_1, i_2, i_3, i_4, i_5, i_6)$	0.024177924	3 scientific decimal spaces	pass
$\mathbf{cl}(i_1, i_2, i_3)$	0.214690429	4 scientific decimal spaces	pass

4.1.2 Second Stationary Measurement

To verify the Second Stationary Measurement code an arbitrary data point of the flight data is selected and all reduced values are manually calculated. This is compared to the values generated by the code and equal values should arise. The first data point will be chosen for comparison, with a stick force of 1N instead of 0 N to make comparison possible(table 14).

Table 14: Measurement point

	Standard Value
Hp	6510 <i>ft</i>
IAS	161 <i>knots</i>
α	5.4 <i>deg</i>
δ	0.1 <i>deg</i>
Fe	1 <i>N</i>
TAT	1.8 <i>°Celsius</i>

The method of computing the Equivalent Airspeed (EAS) is explained in section 3.1.1. Using equations 34 and 35 the pressure and density as shown in table 16 are found.

Table 15: Inputs for finding atmospheric properties in Equation 34 and Equation 35.

	Inputs
λ	-0.0065 <i>K/m</i>
h_p	1984 <i>m</i>
T	270.9 <i>K</i>

Table 16: Outputs of Equation 34 and Equation 35.

	Outputs
Static Pressure	79641 <i>Pa</i>
Air Density	1.024 <i>kg/m³</i>

Next the Mach number, static temperature and Speed of sound are calculated using equations 36, 37 and 38. An IAS of 161 equals (161-2)*0.51444=81.8 m/s.

Table 17: Inputs for Equation 36, Equation 37 and Equation 38.

	Inputs
p	79641 <i>Pa</i>
V_c	81.80 <i>m/s</i>
T_m	270.9 <i>K</i>

Table 18: Outputs of Equation 36, Equation 37 and Equation 38.

	Outputs
Mach number	0.274 —
Static Temperature	266.9 <i>K</i>
Speed of Sound	327.5 <i>m/s</i>

The Equivalent Airspeed is computed using equation 39.

Table 19: Inputs for Equation 39.

Inputs	
ρ	1.024 kg/m^3
V_{TAS}	89.3 m/s

Table 20: Outputs of Equation 39.

Outputs	
Equivalent Airspeed	81.6 m/s

When plugging in the equivalent airspeed of 81.6 m/s, the standardised weight of 60500 N and the momentary weight of 63088 N (mass from Table 13 multiplied with the gravitational acceleration) the reduced equivalent airspeed equals 79.9 m/s.

Next the stick force will be reduced using equation 49. A stick force of 1 N leads to a reduced stick force of 0.959 N.

$$F_{e aer}^* = 1 \cdot \frac{60500}{63088} = 0.959 \quad (58)$$

Filling in the values from table 14 and the reduced airspeed computed above, with right and left fuel flows of respective 454 and 500 lbs/hr gives a total Thrust as explained in section 3.1.3 of 4157.1 N. The standard thrust in this case is 2763.6 N. This leads to $T_c = 2.164$, $T_{cs} = 1.438$ and a corresponding δ_{eq}^* of 0.0050 rad or 0.29 degrees. Using formulas 59, 51 and 52.

$$\delta_{eq}^* = 0.1 * \frac{\pi}{180} - \frac{1}{-1.4227} - 0.0064 (1.438 - 2.164) = 0.0050 \text{ rad} \quad (59)$$

$$T_c = \frac{4157.1}{\frac{1}{2} 1.024 * 89.3^2 * 0.686^2} = 2.164 \quad (60)$$

$$T_{cs} = \frac{2763.6}{\frac{1}{2} 1.024 89.3^2 0.686^2} = 1.438 \quad (61)$$

Comparing the final manual and program values, it can be seen that the program does what it is supposed to do. There are minor differences between some of the values but that can be explained by the truncation error of manually computing values. Setting the required number of significant digits to four, all parameters pass the test.

Table 21: Manually vs computer generated values

	Manual values	Program Values
δ_{eq}^*	0.0050 rad	0.0050 rad
F_e^*	0.959 N	0.959 N
V_{EAS}	81.64 m/s	81.64 m/s

4.2 Analysing Eigenmodes and Natural Frequencies

The state space model can be described by Equation 62, if no control inputs are present. This behaviour characterises the response of the model to initial states.

$$\frac{\partial}{\partial t} \mathbf{x} = A \mathbf{x} \quad (62)$$

When A is diagonalizable, it can be expressed as $A = PDP^{-1}$, where D is a diagonal matrix containing the eigenvalues of A and P is a block matrix whose columns are the eigenvectors of A . With this in mind the set of ODEs can be decoupled as Equation 63, where every every row n can be expressed as $\frac{d}{dt} y_n = \lambda_n y_n$, with $\mathbf{y} = P^{-1} \mathbf{x}$.

$$\frac{\partial}{\partial t} P^{-1} \mathbf{x} = DP^{-1} \mathbf{x} \quad (63)$$

These ODEs can be solved analytically to be $y_n = y_{n0} e^{\lambda_n t}$ and can be expressed in matrix form to be $\mathbf{y} = e^{Dt} \mathbf{y}_0$ or in terms of \mathbf{x} as $\mathbf{x} = P e^{Dt} P^{-1} \mathbf{x}_0$. Using the eigenvectors of A as initial conditions the eigenmotions of the system can be found. These eigenvectors are however complex, thus the real part of them is plotted as the imaginary component describes the phase shift of the outputs to each other.

Symmetric Motion In the symmetric motion, the eigenvalues can be found in Table 22, with their corresponding eigenmotions plotted in Figure 7 and Figure 8. It can be seen that λ_1 and λ_2 , as well as λ_3 and λ_4 are complex conjugates to each other, which results in only 2 characteristic motions for the linear model. The first 2 eigenvalues can be attributed to the short period of the aircraft, as the angular velocity of the motion as well as the damping is very high. The last 2 eigenvalues can be attributed to the phugoid motion of the aircraft, as the frequency is quite low, as well as a lower damping.

Table 22: Numerical eigenvalues of the symmetric statespace model

	Reference Data	Flight Test
λ_1	$-2.3439 + 3.3114i$	$-2.0865 + 3.5742i$
λ_2	$-2.3439 - 3.3114i$	$-2.0865 - 3.5742i$
λ_3	$-0.0109 + 0.1332i$	$-0.0042 + 0.1664i$
λ_4	$-0.0109 - 0.1332i$	$-0.0042 - 0.1664i$

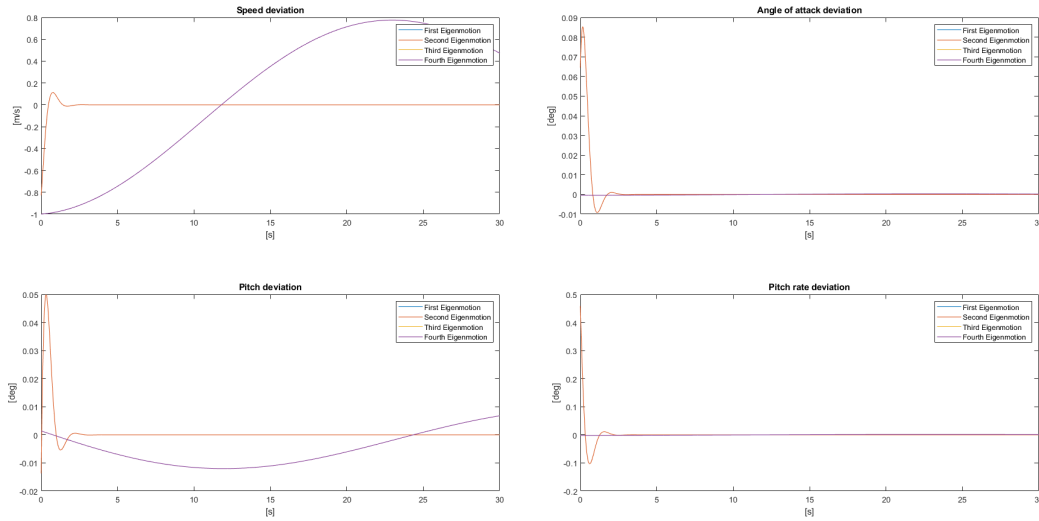


Figure 7: Symmetric eigenmotions of the aircraft with the reference data

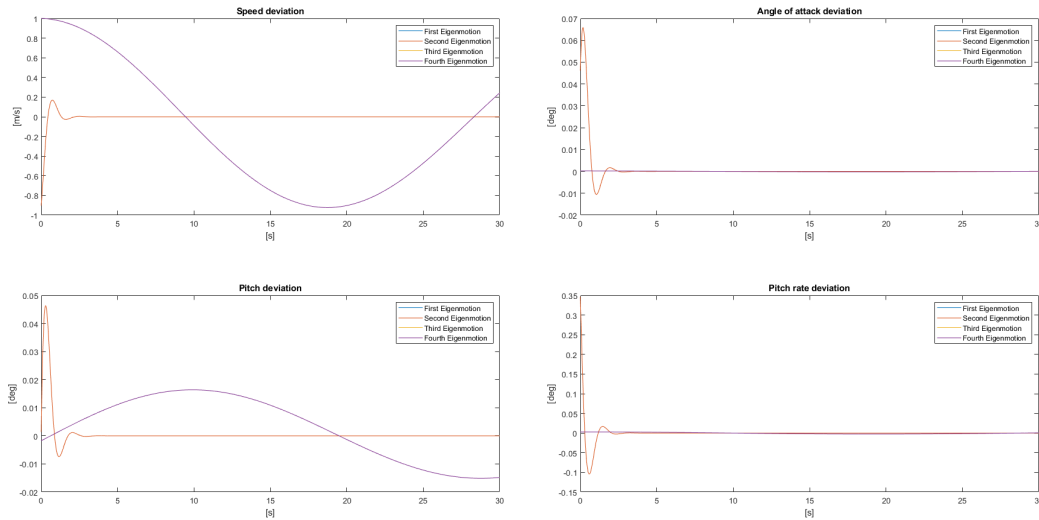


Figure 8: Symmetric eigenmotions of the aircraft for the flight test

Assymmetric Motion In the assymmetric motion, the eigenvalues can be found in Table 23, with their corresponding eigenmotions plotted in Figure 9 and Figure 10. It can be seen that λ_2 and λ_3 are complex conjugates to each other and thus describe the same eigenmotion of the aircraft. It furthermore can be seen that in contrast to the symmetric case not all eigenvalues have a negative real part. This positive eigenvalue indicates an instability in the aircraft, which can also be seen in the graphs. The first eigenmotion can be interpreted as the inherent stability of the aircraft, while the second and third eigenmotion describe the dutch roll indicated by the periodic nature of the motion. The last eigenmotion is a spiral motion of the aircraft, which is unstable for the Cessna Citation II.

Table 23: numerical eigenvalues of the assymmetric statespace model

	Reference Data	Flight Test
λ_1	-7.2529	-7.3017
λ_2	$-1.1058 + 3.3941i$	$-1.0881 + 3.4380i$
λ_3	$-1.1058 - 3.3941i$	$-1.0881 - 3.4380i$
λ_4	0.0064	0.0199

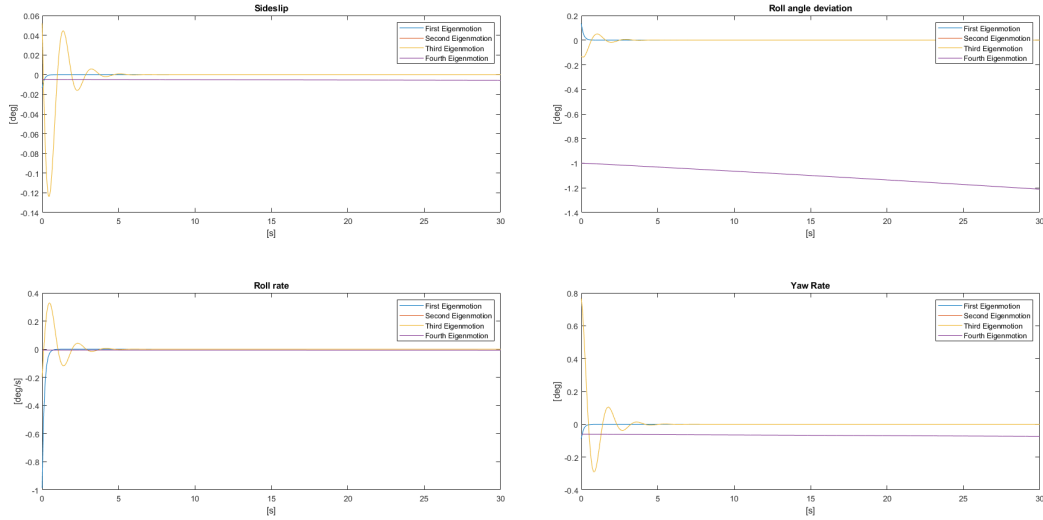


Figure 9: Assymmetric eigenmotions of the aircraft with the reference data

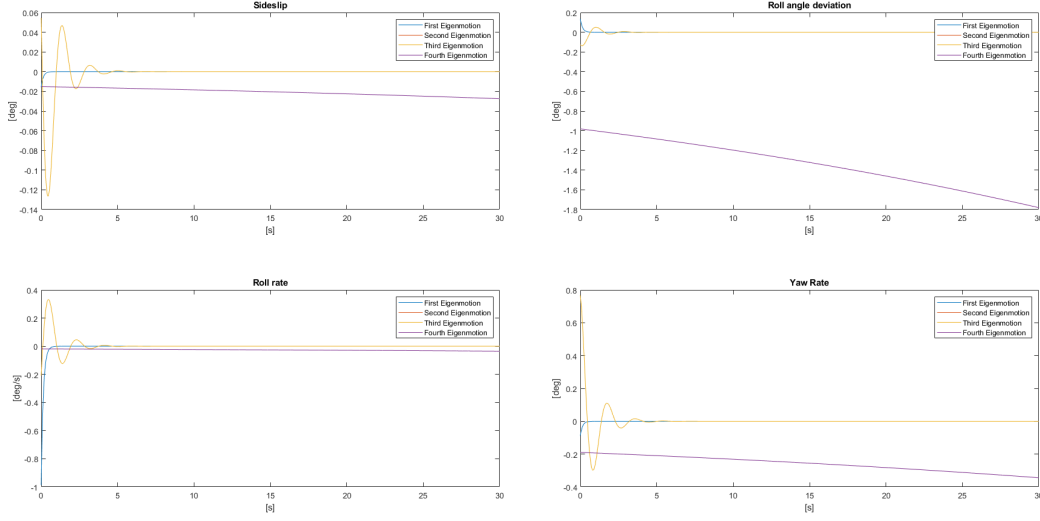


Figure 10: Asymmetric eigenmotions of the aircraft for the flight test

4.3 Analytic derivation of Eigenvalues

This section discusses the analytic derivation of the eigenvalues of the state matrix A for the symmetric and the asymmetric case. The eigenvalues correspond to the eigenmodes of the aircraft. For each eigenmode certain simplifications to the equations of motion can be made [3]. This simplifies the analytic derivation of the eigenvalues of A_s and A_a . All eigenvalues in this section are calculated with the values from the flight test data.

Starting with the two symmetric eigenmodes Short Period and Phugoid the general matrix of the EOM given in (64) will be simplified. After that with the asymmetric eigenmodes Dutch Roll and Spiral the matrix of the EOM given in (65) will be simplified.

$$\begin{bmatrix} C_{x_u} - 2\mu_c D_c & C_{x_c} & C_{z_0} & C_{x_i} \\ C_{z_x} & C_{z_x} + (C_{z_s} - 2\mu_c) D_c & -C_{x_0} & C_{z_t} + 2\mu_c \\ 0 & 0 & -D_c & 1 \\ C_{m_s} & C_{m_s} + C_{m_s} D_c & 0 & C_{m_t} - 2\mu_c K_s^2 D_c \end{bmatrix} \cdot \begin{bmatrix} \hat{u} \\ \alpha \\ \theta \\ \frac{q\bar{c}}{V} \end{bmatrix} \quad (64)$$

$$\begin{bmatrix} C_{Y_\beta} + (C_{Y_\beta} - 2\mu_b) D_b & C_L & C_{Y_p} & C_{Y_r} - 4\mu_b \\ 0 & -\frac{1}{2} D_b & 1 & 0 \\ C_{\ell_\beta} & 0 & C_{\ell_p} - 4\mu_b K_X^2 D_b & C_{\ell_r} + 4\mu_b K_{XZ} D_b \\ C_{n_\beta} + C_{n_\beta} D_b & 0 & C_{n_p} + 4\mu_b K_{XZ} D_b & C_{n_r} - 4\mu_b K_Z^2 D_b \end{bmatrix} \begin{bmatrix} \beta \\ \varphi \\ \frac{pb}{2V} \\ \frac{rb}{2V} \end{bmatrix} \quad (65)$$

Short period In the short period oscillation the velocity can be assumed to be constant ($V = \text{constant}$). Hence by definition $\hat{u} = 0$ which eliminates the first column of (64). Since there are no velocity changes there are also no accelerations along the X-Body axis. Thus the equilibrium equation along X in row 1 becomes obsolete and can be removed as well. Since the maneuver is performed in steady level flight $\gamma_0 = 0$ and by that $C_{X_0} = 0$. Also row 3 of the matrix can be omitted since $-D_c \theta + \frac{q\bar{c}}{V} = 0$. These assumptions reduce the matrix to matrix (66).

$$\begin{bmatrix} C_{Z_\alpha} + (C_{Z_\alpha} - 2\mu_c) D_c & C_{Z_q} + 2\mu_c \\ C_{m_\alpha} + C_{m_\alpha} D_c & C_{m_q} - 2\mu_c K_Y^2 D_c \end{bmatrix} \begin{bmatrix} \alpha \\ \frac{q\bar{c}}{V} \end{bmatrix} \quad (66)$$

As it was done in subsection 2.4 this matrix can be split in C_1 and C_2 .

$$C_1 = \begin{bmatrix} (C_{Z_\alpha} - 2\mu_c) \frac{\bar{c}}{V} & 0 \\ C_{m_\alpha} \frac{\bar{c}}{V} & -2\mu_c K_Y^2 \frac{\bar{c}}{V} \end{bmatrix}$$

$$C_2 = \begin{bmatrix} C_{Z_\alpha} & C_{Z_q} + 2\mu_c \\ C_{m_\alpha} & C_{m_q} \end{bmatrix}$$

Since $A = -C_1^{-1} C_2$ the inverse of C_1 must be determined.

$$C_1^{-1} = \begin{bmatrix} -\frac{1}{(2\mu_c - C_{Z\dot{\alpha}})\frac{\bar{c}}{V}} & 0 \\ -\frac{C_{m\dot{\alpha}}}{2\mu_c K_Y^2 (2\mu_c - C_{Z\dot{\alpha}})\frac{\bar{c}}{V}} & -\frac{1}{2\mu_c K_Y^2 \frac{\bar{c}}{V}} \end{bmatrix}$$

This results in A with A_{11}, A_{12}, A_{21} and A_{22} being known numerical values since all variables are known.

$$A = -C_1^{-1}C_2 = \begin{bmatrix} \frac{C_{Z\dot{\alpha}}}{2\mu_c \frac{\bar{c}}{V}} & \frac{V}{\bar{c}} \\ \frac{C_{Z\dot{\alpha}}C_{m\dot{\alpha}}}{4\mu_c^2 K_Y^2 \frac{\bar{c}}{V}} + \frac{C_{m\dot{\alpha}}}{2\mu_c K_Y^2 \frac{\bar{c}}{V}} & \frac{C_{m\dot{\alpha}} + C_{m\dot{\alpha}}}{2\mu_c K_Y^2 \frac{\bar{c}}{V}} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

To get the eigenvalues $\det(A - \lambda I) = 0$ must be solved for λ .

$$\det(A - \lambda I) = \begin{vmatrix} A_{11} - \lambda & A_{12} \\ A_{21} & A_{22} - \lambda \end{vmatrix} = 0$$

This results in the Characteristic Polynomial.

$$\lambda^2 - (A_{11} + A_{22})\lambda - A_{12}A_{21} = 0$$

Using the pq-formula this results in two solutions.

$$\lambda_{1/2} = \frac{A_{11} + A_{22}}{2} \pm \sqrt{\left(\frac{A_{11} + A_{22}}{2}\right)^2 + A_{12}A_{21}}$$

Substituting all variables results in a conjugate pair of complex eigenvalues.

$$\lambda_1 = -1.9869 + 3.7028i$$

$$\lambda_2 = -1.9869 - 3.7028i$$

The eigenvalues calculated by Matlab without the assumptions are $\lambda_{1/2} = -2.0865 \pm 3.5742i$. This is quite close to the analytic eigenvalues.

Phugoid In the phugoid the angle of attack can assumed to be 0. By that the whole second column in matrix (64) is eliminated. Since $C_{Z_q} \ll \mu_c$ it can be assumed that $C_{Z_q} + \mu_c \approx \mu_c$. Also the pitch rate can assumed to be constant which leads to $-2\mu_c K_Y^2 D_c = 0$. Since the system now is over-determined and the moment equation becomes redundant it can be crossed out. Also $C_{X_0} = 0$, for the same reasons as in the short period. These assumptions reduce matrix (64) to matrix (67).

$$\begin{bmatrix} C_{X_u} - 2\mu_c D_c & C_{Z_0} & 0 \\ C_{Z_u} & 0 & 2\mu_c \\ 0 & -D_c & 1 \end{bmatrix} \begin{bmatrix} \hat{u} \\ \theta \\ \frac{q\bar{c}}{V} \end{bmatrix} \quad (67)$$

Again this matrix can be split in C_1 and C_2 .

$$C_1 = \begin{bmatrix} -2\mu_c \frac{\bar{c}}{V} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -\frac{\bar{c}}{V} & 0 \end{bmatrix}$$

$$C_2 = \begin{bmatrix} C_{X_u} & C_{Z_0} & 0 \\ C_{Z_u} & 0 & 2\mu_c \\ 0 & 0 & 1 \end{bmatrix}$$

Since C_1 is a singular matrix an inverse can not be computed. A way to avoid C_1^{-1} is to compute $A^{-1} = -C_1 C_2^{-1}$ instead. Performing the calculations will result in the following A^{-1} .

$$A^{-1} = \begin{bmatrix} 0 & \frac{2\mu_c \frac{\bar{c}}{V}}{C_{Z_u}} & -\frac{4\mu_c^2 \frac{\bar{c}}{V}}{C_{Z_u}} & 0 \\ 0 & 0 & 0 & 0 \\ \frac{\bar{c}}{C_{Z_0}} & -\frac{C_{X_u} \frac{\bar{c}}{V}}{C_{Z_u} C_{Z_0}} & \frac{2\mu_c C_{X_u} \frac{\bar{c}}{V}}{C_{Z_u} C_{Z_0}} & 0 \end{bmatrix}$$

Solving for the eigenvalues of A^{-1} results in

$$\lambda_a = -0.6176 + 6.4498i$$

$$\lambda_b = -0.6176 - 6.4498i$$

$$\lambda_c = 0.0000 + 0.0000i$$

To get the relevant eigenvalues of A the inverse of λ_a and λ_b must be taken which results in λ_1 and λ_2 .

$$\begin{aligned}\lambda_1 &= -0.0147 - 0.1536i \\ \lambda_2 &= -0.0147 + 0.1536i\end{aligned}$$

The eigenvalues calculated by Matlab without the assumptions are $\lambda_{1/2} = -0.0042 \pm 0.1664i$. It can be seen that the complex part is quite close. The real part however is quite off. This can be explained by the very strong assumption that $\alpha = 0$. An alternative assumption introducing a smaller error would be that only $\dot{\alpha} = 0$. This however would lead to a more complicated 4x4 matrix which is less easy to solve by hand.

Dutch Roll For the Dutch Roll it can be assumed that the yawing is more dominant than the rolling and thus $\phi = 0$ and $p = 0$. By that column 2 and 3 as well as row 2 and 3 of matrix (65) drop out. Also $C_{Y_{\dot{\beta}}} \approx 0$, $C_{n_{\dot{\beta}}} \approx 0$ and $C_{Y_r} \ll \mu_b$. By that the matrix is reduced to matrix (68).

$$\begin{bmatrix} C_{Y_{\beta}} - 2\mu_b D_b & -4\mu_b \\ C_{n_{\beta}} & C_{n_r} - 4\mu_b K_Z^2 D_b \end{bmatrix} \begin{bmatrix} \beta \\ \frac{rb}{2V} \end{bmatrix} \quad (68)$$

Using the same procedure as for the Short Period results in matrix A.

$$A = \begin{bmatrix} \frac{C_{Y_{\beta}}}{2\mu_b \frac{\bar{c}}{V}} & -\frac{2V}{\bar{c}} \\ \frac{C_{n_{\beta}}}{4\mu_b K_Z^2 \frac{\bar{c}}{V}} & \frac{C_{n_r}}{4\mu_b K_Z^2 \frac{\bar{c}}{V}} \end{bmatrix}$$

The eigenvalues of A are

$$\begin{aligned}\lambda_1 &= -0.6094 + 3.4845i \\ \lambda_2 &= -0.6094 - 3.4845i\end{aligned}$$

The eigenvalues calculated by Matlab without the assumptions are $\lambda_{1/2} = -1.0881 \pm 3.4380i$. The complex part fits well but the real part is a little off. This is again due to the strong assumptions made. Roll angle and roll rate are not that close to zero. That causes the error.

Spiral Since the spiral is a very slow motion (at least for the Cessna Citation) the accelerations can be neglected. Also $C_{Y_p} \approx 0$ and $C_{Y_r} \ll \mu_b$. By that matrix (65) simplifies to matrix (69)

$$\begin{bmatrix} C_{Y_{\beta}} & C_L & 0 & -4\mu_b \\ 0 & -\frac{1}{2}D_b & 1 & 0 \\ C_{\ell_{\beta}} & 0 & C_{\ell_p} & C_{\ell_r} \\ C_{n_{\beta}} & 0 & C_{n_p} & C_{n_r} \end{bmatrix} \begin{bmatrix} \beta \\ \varphi \\ \frac{pb}{2V} \\ \frac{rb}{2V} \end{bmatrix} \quad (69)$$

$$C_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -\frac{\bar{c}}{2V} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$C_2 = \begin{bmatrix} C_{Y_{\beta}} & C_{\ell} & 0 & -2\mu_b \\ 0 & 0 & 1 & 0 \\ C_{\ell_{\beta}} & 0 & C_{\ell_p} & C_{\ell_r} \\ C_{n_{\beta}} & 0 & C_{n_p} & C_{n_r} \end{bmatrix}$$

Since matrix C_1 is singular the same procedure as for the Phugoid must be applied. Doing the calculations the eigenvalues of A^{-1} will be

$$\begin{aligned}\lambda_a &= 49.1808 \\ \lambda_b &= 0\end{aligned}$$

Which result in one relevant eigenvalue for A

$$\lambda = 0.0203$$

The eigenvalue calculated by Matlab without the assumptions is $\lambda = 0.0199$. Hence the approximation is very close.

In total all eigenvalues of the numerical model are at least in the same order of magnitude as the analytically derived eigenvalues. This verifies the numerical model to a certain extend.

4.4 Model Response to Inputs and Disturbances

4.4.1 Response to Control Inputs

Phugoid The plots below are created using the linear simulation model and are therefore used to verify the numerical model. The next step is to establish whether these plots make physical sense. At $t = 5s$, there is an input in the pitch rate as seen in figure 11. This is a positive pitch rate and one would expect that a positive pitch rate will lead to a positive increase in angle of attack, which in turn will lead to a positive increase in body angle. With a greater angle of attack, the velocity will decrease as less speed is necessary to create the same amount of lift. This is the exact behavior Figure 11 shows. Figure 11, also shows the correct continuation of this positive pitch rate. As shown in the figure the pitch rate decreases after the sudden increase at $t = 5s$, all the way to a negative pitch rate at $t = 17s$. As expected the body angle decreases, the angle of attack slowly increases and so does the velocity. This leads to the conclusion that this numerical model is verified.

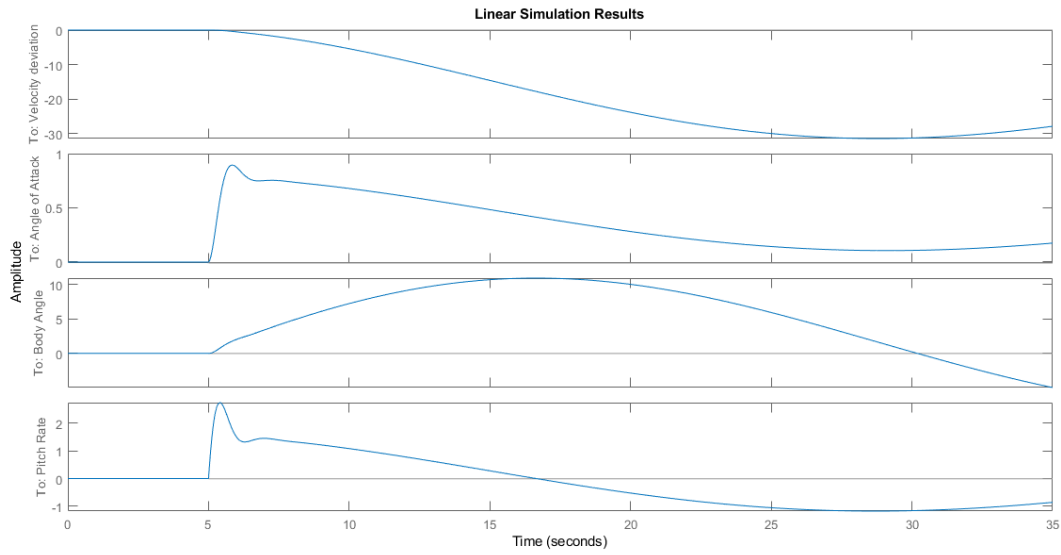


Figure 11: Aircraft response to input - Phugoid

Short period Figure 12 shows the aircraft's simulated response to a control input causing a short period. All parameters are as expected for a short period and they all dampen out at their trim settings. Except for the speed. The speed keeps on increasing over time which is not normal. Therefore the short period is not fully verified.

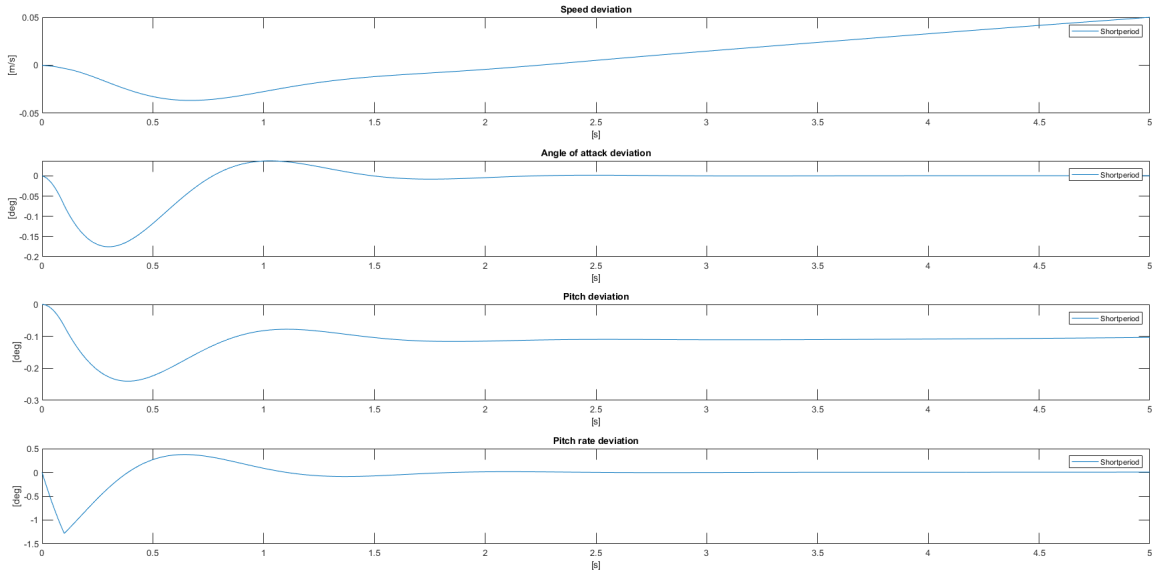


Figure 12: Aircraft response to input - Dutch Roll

Dutch Roll Figure 13 shows the Aircraft's response to an induced Dutch Roll. As evident from the plot the control input leads to large increase in side slip angle and a small increase in bank angle. The bank angle turns negative very quickly and then oscillates and stabilizes around -0.45 . It must be noted that the bank angle does continue to decrease very slightly. The effects of this are more noticeable when a larger time is taken. The other plotted values (side slip angle, roll rate and yaw rate) all dampen out around zero.

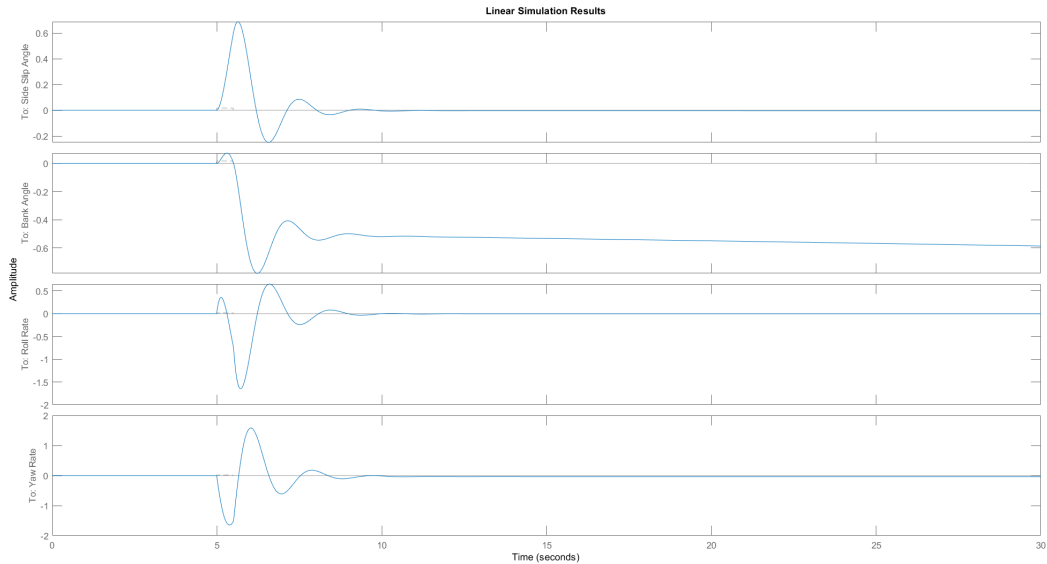


Figure 13: Aircraft response to input - Dutch Roll

Spiral As shown in Figure 14, the control input for the spiral is started at $t = 5sec$. The control input causes a negative spike in the roll rate. This peak disappears after 1 second and the roll rate remains 0 for the rest of the manoeuvre. The yaw rate on the other hand oscillates and dampens out at -1 . Also the side slip angle oscillates, but dampens out at -0.1 . The bank angle, on the other hand, goes from 0 to -15 degrees. Unlike the other parameters the bank angle does not stay constant at -15 degrees but decreases further, but at a very slow rate. This leads to the side slip angle decreasing slightly too (as they are connected to each other).

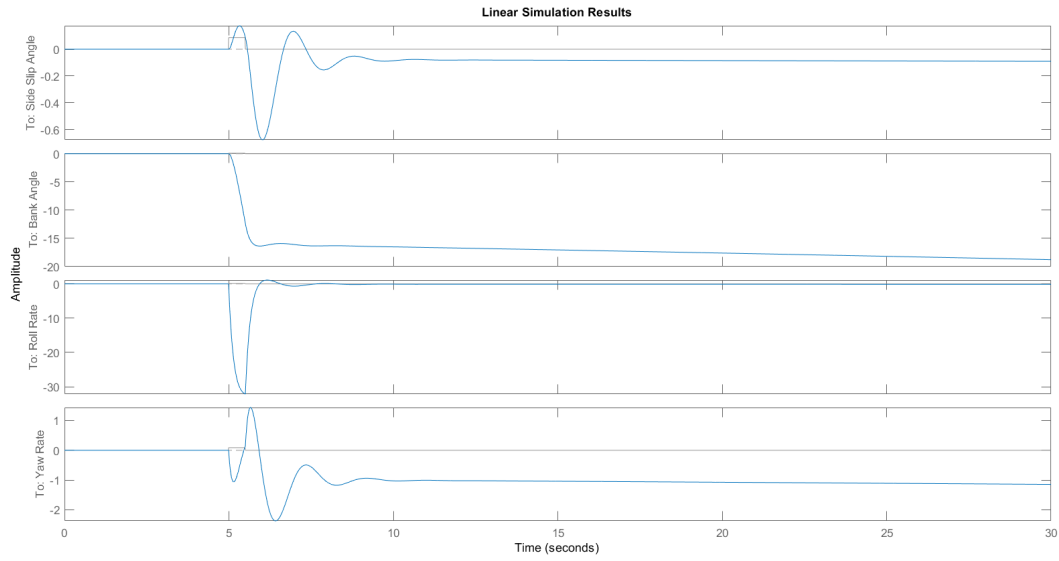


Figure 14: Aircraft response to input - Spiral

4.4.2 Response to Disturbances

Phugoid Figure 15 shows the aircraft's simulated response to a disturbance in one of the states. More specifically, it shows how the aircraft responds to a disturbance in the pitch. As visible in the figure the pitch angle rapidly decreases from 2 to about 0.25. This leads to a phugoid motion in the aircraft as visible by the angle of attack which oscillates around the 0 degrees axis, and the velocity deviation which also shows a correct oscillation. In the 200 s time frame, it is also already noticeable that the phugoid motion dampens which is what is expected from a stable aircraft.

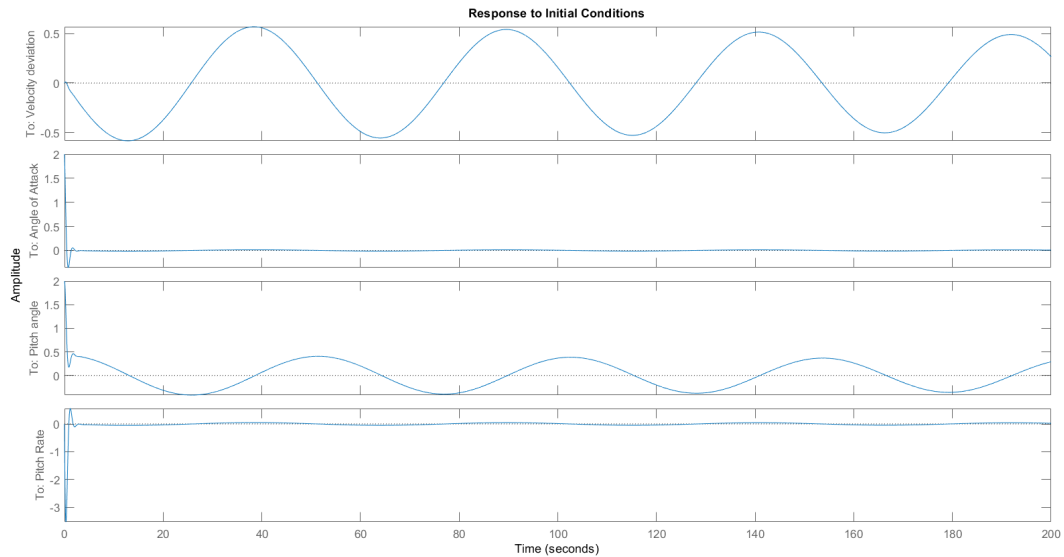


Figure 15: Aircraft Simulated Response to Disturbance - Phugoid

Short period The short period disturbance plot is shown in Figure 16. As shown in the plot the oscillations dampen out very quickly and return to their trim positions within 2 seconds. As this is the short period this is as expected and therefore this plot for the short period can be considered verified.

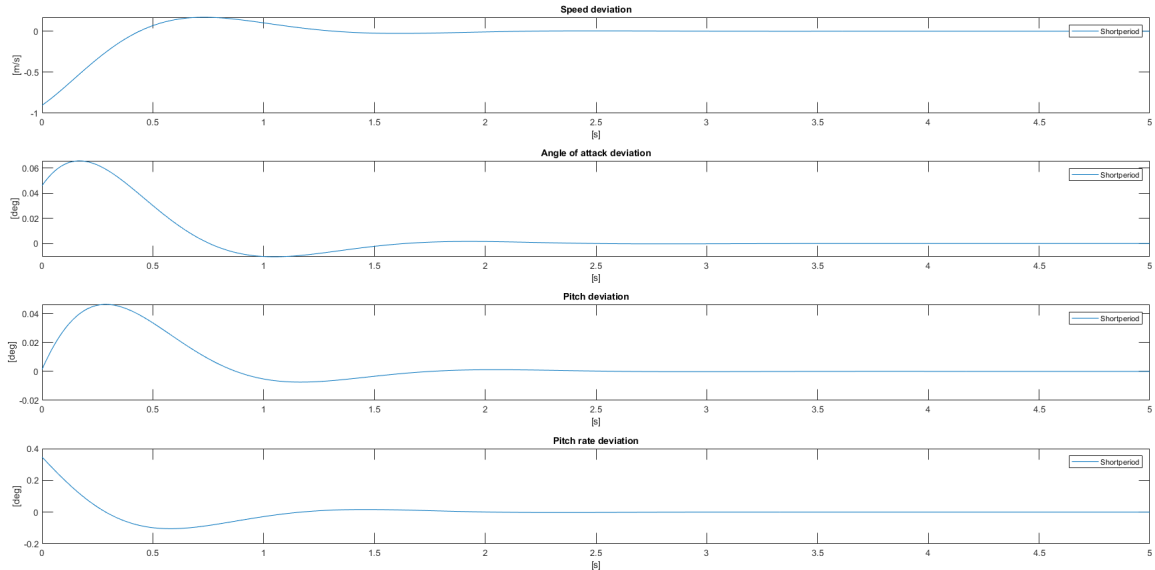


Figure 16: Aircraft Simulated Response to Disturbance - Short Period

Dutch Roll Figure 18 shows the effects of the aircraft to a Dutch Roll causing disturbance. The disturbance in question is a 5 degree change in side slip angle that could theoretically be caused by a side gust. As evident from the curves the aircraft will first experience pitch angles and angles of attack very similar to those of a Dutch Roll caused by the response of an input (Figure 13). After the Disturbance the simulated aircraft seems to stabilize with a roll rate of 0, a yaw rate of 0 and a side slip angle of 0. The bank angle, however, flattens out at -0.9.

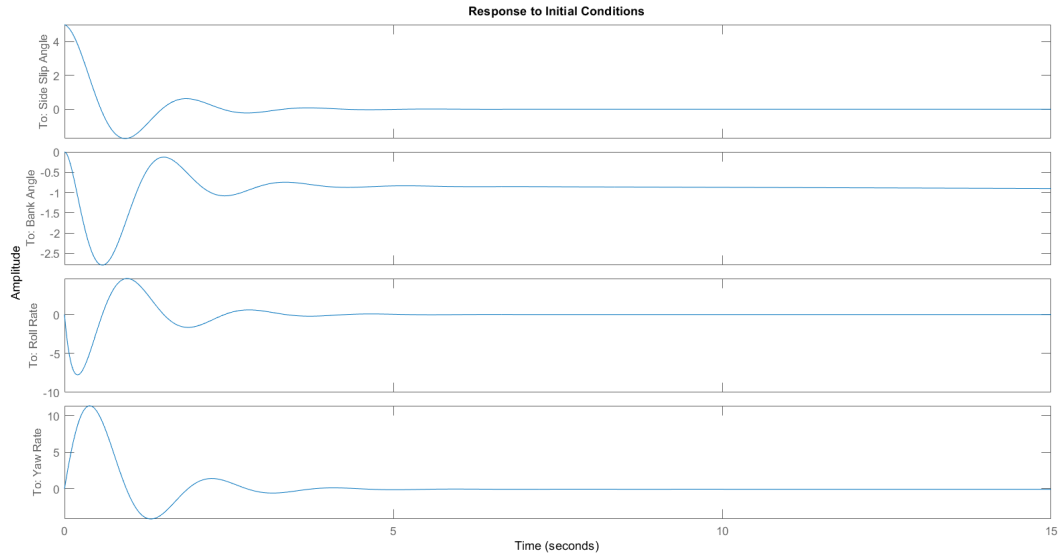


Figure 17: Aircraft Simulated Response to Disturbance - Dutch Roll

Spiral The last disturbance that needs to be evaluated is a disturbance causing a spiral. As disturbance a 5 degree bank angle has been chosen. As visible from the plot the simulation does as expected. The bank angle first decreases a little but then increases at a fast pace (0.036 degrees per second). The side slip angle oscillates and dampens out at 0.025. The roll rate changes according to the bank angle and changes sign in the first 2 seconds but then remains constant as the bank angle linearly increases.

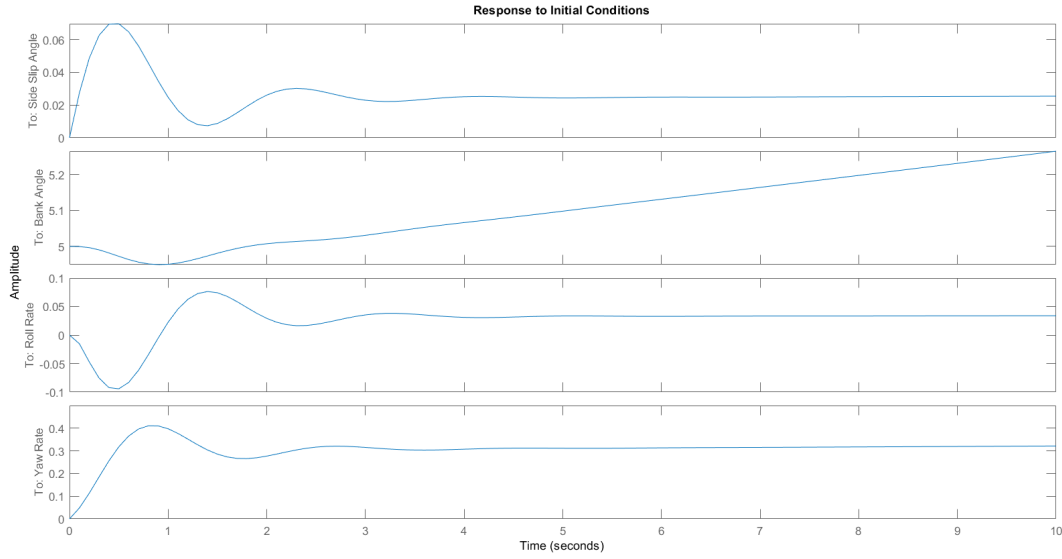


Figure 18: Aircraft Simulated Response to Disturbance - Spiral

Concluding remarks on Model Response to Inputs and Disturbances

By looking at the plots that show the models response to the control input and disturbances it was possible to verify most of the simulation model. All of the plots showed the behavior they were supposed to show. Except for the Short Period response to input plot, where the velocity kept increasing even though it shouldn't. Overall it is possible to say that all responses are verified except for the Short Period.

5 Validation

In this chapter the numerical model will be compared to the data obtained from the flight test. For this the model that was perfected by the verification model will be used. First in Section 5.1 the recorded control inputs are fed to the simulation model its response is compared to the measured aircraft response. Than in Section 5.2 the eigenmotions during the flight are plotted and the eigenvalues that are computed in the verification process are validated. Lastly in Section 5.3 some alterations to the model are made to match the data if arguments can be found why the model wasn't correctly fitting the data before.

5.1 Simulated vs Measured Responses

Symmetric Movement The 2 symmetric motions analysed during the flight test were the phugoid as well as the short period which are shown in Figure 21 and Figure 19 respectively. The response of the aircraft as well as the response of the state space model are plotted as deviations from the stable state of the aircraft. In order to simulate the response of the aircraft, the measured control surface deflections are used, which are also taken as deviation from the stable condition.

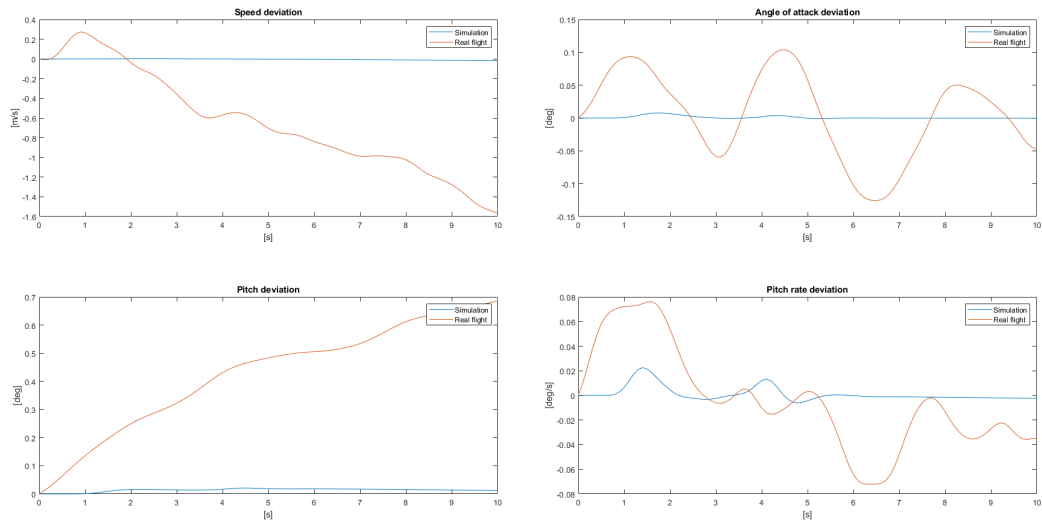


Figure 19: Short Period

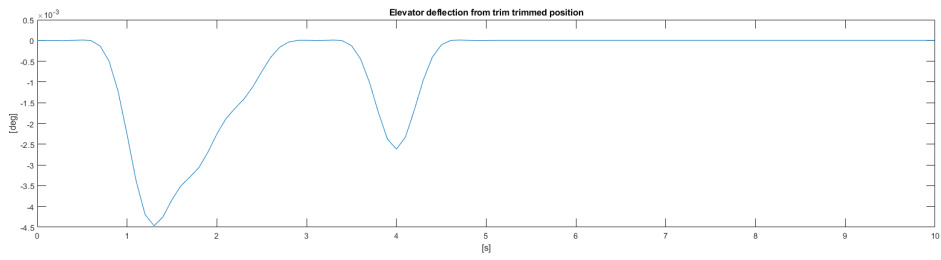


Figure 20: Elevator input during the shortperiod

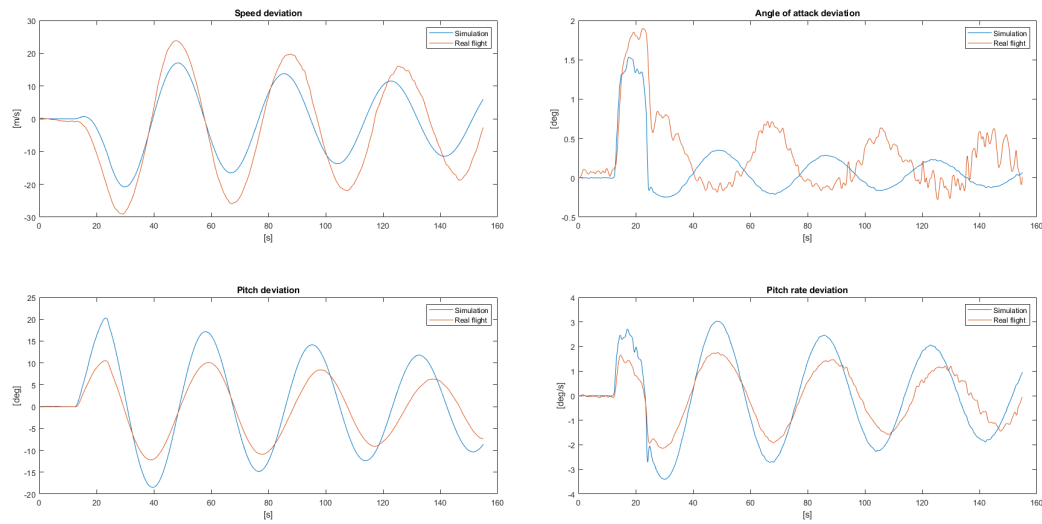


Figure 21: Phugoid



Figure 22: Elevator input during the phugoid

It can be seen that the response of the linear system shows similarities for the phugoid, however for the short period response, the linear response does not represent the observed behaviour at all. For the phugoid, the period as well as the halving time can be calculated for the different output variables. The period for the numerical solution is 37.8 [s], while the period for the real flight is significantly smaller with 38 [s], which is a near exact fit. The halving time for the numerical model is calculated to be 165 [s]. while the real flight data shows a halving time of 169 [s], which is an error of 2.4%. Furthermore, an amplitude error can be observed, which varies with the different outputs, this could be explained, by an error in the elevator effectiveness. In the comparison of angle of attack between the numerical and real solution, a phase shift of π can be observed, which can be caused due to the fact that the sensor for the angle of attack is not located at the cg around which the aircraft rotates and thus encounters additional airstream components due to the pitch rate of the aircraft.

Asymmetric Movement For the asymmetric motion, the dutch roll and spiral were analysed, which can be found in Figure 23 and Figure 24 respectively. The plotting and handling of inputs are handled as described in the symmetric motion. However, as sideslip was not recorded during the flight no comparison between the numerical model and the real flight could be performed.

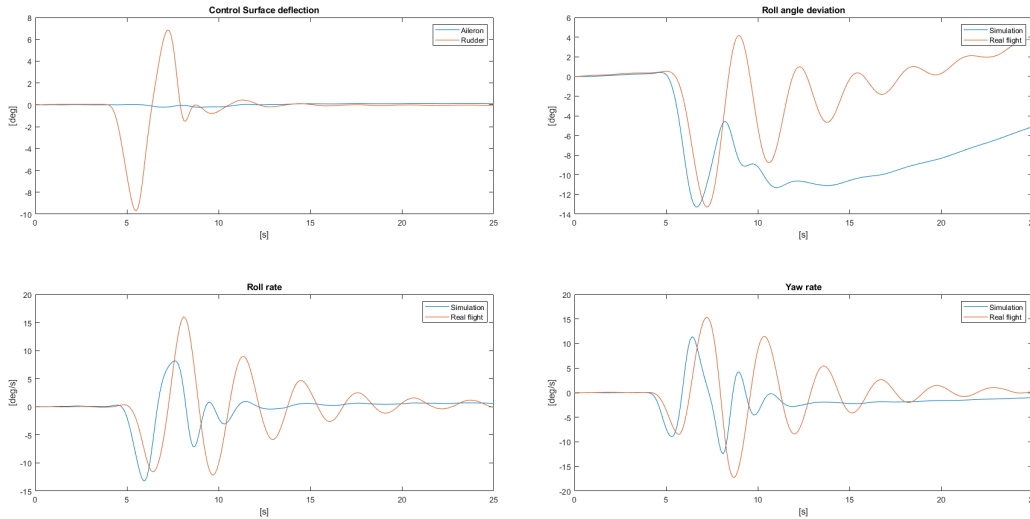


Figure 23: Dutch Roll

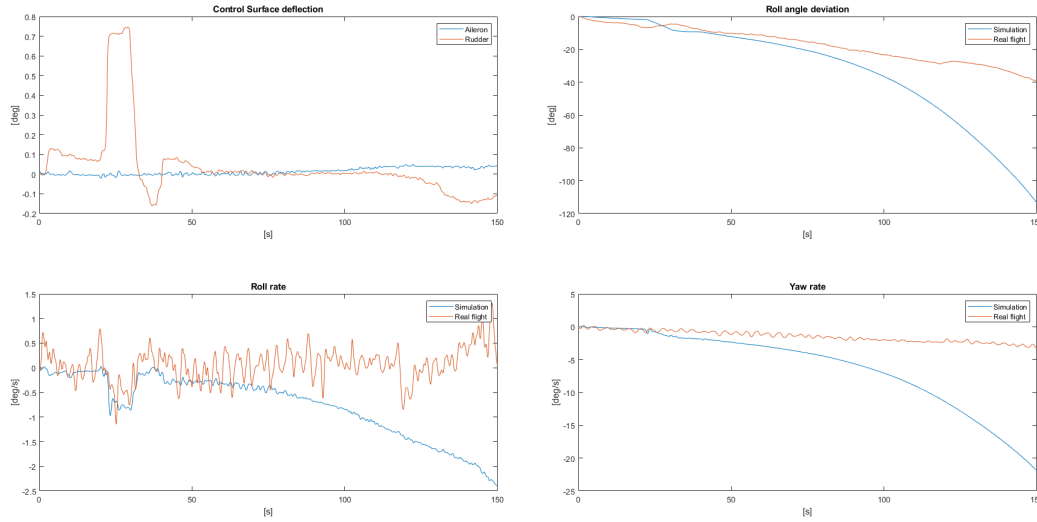


Figure 24: Spiral

From the figures it can be seen that the spiral is not well simulated, while the dutch roll response displays the characteristic motion. The period for the numerical model can be found to be 1.8 [s], while the real period is 2.4 [s]. This is an error of 28.5%. The halving time for the numerical model can be found to be 0.75 [s], while the real halving time is 3.1 [s]. This difference is very large and is probably caused by errors in the lateral stability derivatives.

5.2 Eigenmotions and Eigenvalues

In 4.2 the eigenmodes and eigenvalues of the simulated response were given. Now this will be validated with the actual aircraft response and eigenmodes plotted in the previous graphs. The dynamic data gathered during the flight does not directly provide any numerical values for eigenvalues. Nonetheless the response still allows for checking if the simulated eigenvalues show the correct sign and magnitude.

Symmetric Movement The short period motion of the aircraft has eigenvalues 1 and 2 in Table 22. However, none of their behaviour is shown in Figure 19. The eigenvalues have a very negative real part, so an exponential decay should be visible. This does not occur. The short period cannot be simulated using the numerical model because it does not represent the physical flight data at all.

The phugoid has the eigenvalues 3 and 4 in Table 22. They are complex conjugates of each other, because if you take a linear combination of the solutions of the equations of motion you should obtain a real non-complex result. Their real part is negative and very close to zero. Therefore hardly any damping should be observed. Indeed, the test flight results show very slow damping for the phugoid motion. The complex part of the eigenvalues are small meaning the motion has a small natural frequency. That's completely in-line with the observations since the phugoid is an eigenmode with a large period.

Asymmetric Movement The Dutch roll has the eigenvalues 2 and 3 in Table 23. They are complex conjugates with a large negative real part and a large complex part. Therefore the motion should be well-damped and have a high natural frequency (and thus small period). This is indeed what is observed in Figure 23.

The spiral eigenmode has a different behaviour in the simulation compared to the dynamic flight data. The roll rate and yaw rate see a faster and larger response in the simulated model. Eigenvalue 4 in Table 23 is positive and near zero. The behaviour is therefore expected to diverge, however, this is not visible in the flight test results. The spiral can therefore not be modelled correctly with the linearized equations of motion to accurately represent real data.

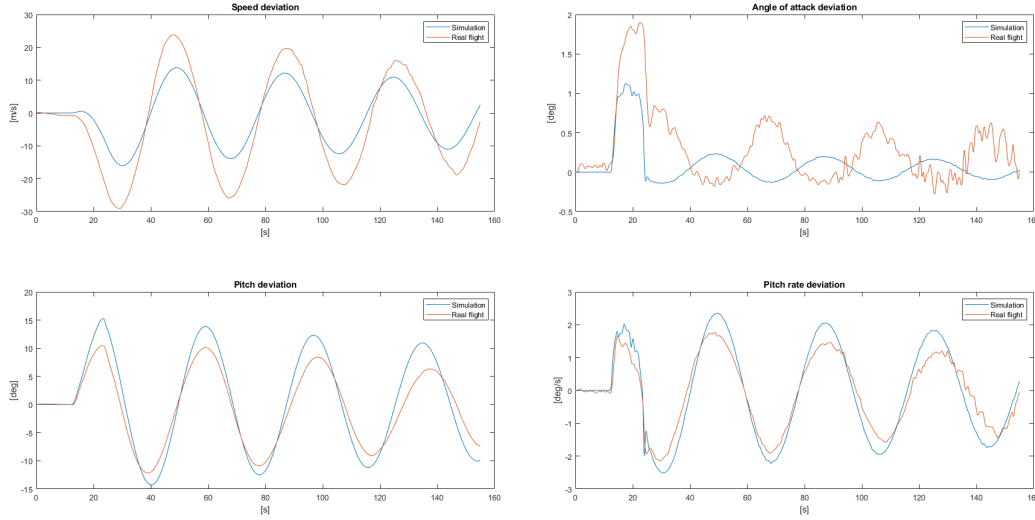


Figure 25: Fitted motion of the phugoid

5.3 Model Matching

Symmetric Motion In order to better fit the numerical model to the real flight data, model matching was performed. The first observation in comparing the numerical model to the real data, is the amplitude error in pitch angle and pitch rate. This can be compensated by either decreasing elevator effectiveness $C_{m_{\delta e}}$ or increasing longitudinal stability $C_{m_{\alpha}}$. The drawback in changing $C_{m_{\alpha}}$, however is the change in natural frequency and damping ratio, on the other hand $C_{m_{\delta e}}$ was determined during the flight and thus is sufficiently accurate, thus $C_{m_{\alpha}}$ is decreased by 0.35. This better matches the amplitude of pitch and pitch rate, however increases the overall damping of the motion and decreases the amplitude of airspeed deviation significantly. This discrepancy however could not be resolved.

Asymmetric Motion After fitting most of the coefficients for the symmetric motion only few parameters were still open for change. These parameters were CYb , CYb_{dot} , CYp , CYr , $CYda$, $CYdr$, Clb , Clp , Clr , Cl_{da} , Cl_{dr} , Cnb , Cnb_{dot} , Cnp , Cnr , $Cnda$, $Cndr$. Altering the value of these parameters should make it possible to make the dutch roll motion fit the flight test data. To approach the fitting smartly it is noted that the initial responses seem to be almost correct, but all the motions seem to damp out to quickly. This behavior is somewhat expected due to it being a linear model, but should be corrected for none the less. In an attempt to limit the damping the change in the real part of the eigenvalues was observed since these should increase for the damping to become less, but should never become more than zero. While possibilities were found to make either the yaw rate or the roll rate behave better, no set of parameter were found in which they both fitted the data.

6 Conclusion

In conclusion, it has been found that the developed numerical model can represent the characteristic motions of the Cessna Citation II sufficiently accurate for small timescales. Especially the phugoid is represented accurately in both frequency and damping. However, the short period of the aircraft is unable to be induced in the model by a steering input even though the corresponding eigenvalues and eigenmotions can be found in the state space model. In the asymmetric Motion, the dutch roll is clearly identifiable in the developed model, however the frequency and damping of the motion deviates from the real flight data. This can be attributed to the more complex equations of motion as well as nonlinear effects in the motion. Similarly, the numerical model can predict the beginning of the spiral with sufficient accuracy, however, the solutions quickly diverge as nonlinear effects take over the real aircraft. These can not be modeled with a state space model.

For further research it is recommended to develop a more sophisticated numerical model, which is able to take nonlinear effects into account, or is able to provide correction terms to correct for nonlinearities in the operational scope of the aircraft. The model could furthermore be improved by obtaining higher accuracy stability derivatives, which are obtained as a function flight parameters.

References

- [1] T.J. Mulder A.C. in 't Veld. Flight dynamics assignment ae3212-ii, February 2020.
- [2] Forecast International. Pratt whitney canada jt15d, 2012. online, date accessed 17/03/2020.
- [3] J.C. van der Vaart E. de Weerdt C.C. de Visser A.C. in 't Veld E. Mooij J.A. Mulder, W.H.J.J. van Staveren. Flight dynamics, 2013. article number 06917720009.

A Appendix B

aircraft: Cessna Citation II (C550) configuration: Clean cruise (flaps up, gear up)		
Longitudinal force derivatives	Normal force derivatives	Pitch moment derivatives
$C_{X_u} = -0.0279$	$C_{Z_u} = -0.3762$	$C_{m_0} = 0.0297$
$C_{X_\alpha} = -0.4797$	$C_{Z_\alpha} = -5.7434$	$C_{m_u} = 0.0699$
$C_{X_{\dot{\alpha}}} = 0.0833$	$C_{Z_{\dot{\alpha}}} = -0.0035$	$C_{m_\alpha} = -0.5626$
$C_{X_q} = -0.2817$	$C_{Z_q} = -5.6629$	$C_{m_{\dot{\alpha}}} = 0.1780$
$C_{X_\delta} = -0.0373$	$C_{Z_\delta} = -0.6961$	$C_{m_q} = -8.7941$
		$C_{m_\delta} = -1.1642$
		$C_{m_{\eta_c}} = -0.0064$

Figure 26: Longitudinal stability coefficients

aircraft: Cessna Citation II (C550) configuration: Clean cruise (flaps up, gear up)		
Lateral force derivatives	Roll moment derivatives	Yaw moment derivatives
$C_{Y_\beta} = -0.7500$	$C_{l_\beta} = -0.1026$	$C_{n_\beta} = 0.1348$
$C_{Y_{\dot{\beta}}} = 0$	$C_{l_p} = -0.7108$	$C_{n_{\dot{\beta}}} = 0$
$C_{Y_p} = -0.0304$	$C_{l_r} = 0.2376$	$C_{n_p} = -0.0602$
$C_{Y_r} = 0.8495$	$C_{l_{\delta_a}} = -0.2309$	$C_{n_r} = -0.2061$
$C_{Y_{\delta_a}} = -0.0400$	$C_{l_{\delta_r}} = 0.0344$	$C_{n_{\delta_a}} = -0.0120$
$C_{Y_{\delta_r}} = 0.2300$		$C_{n_{\delta_r}} = -0.0939$

Figure 27: Lateral stability coefficients

B Matlab Code

```

1 clear all;
2 close all;
3 clc;
4
5 run('refdataone.m');
6 run('flightdataone.m')
7
8 A_measurementref = [1157 5010 249 1.7 798 813 360 12.5;
9 1297 5020 221 2.4 673 682 412 10.5;
10 1426 5020 192 3.6 561 579 447 8.8;
11 1564 5030 163 5.4 463 484 478 7.2;
12 1787 5020 130 8.7 443 467 532 6;
13 1920 5110 118 10.6 474 499 570 5.2];
14
15 A_measurementflight = [1130 5030 251 1.6 770 806 367 10.2;
16 1258 5030 221 2.4 638 670 422 7.8;
17 1398 5020 190 3.7 533 579 456 5;
18 1530 5040 161 5.6 444 488 492 4;
19 1664 5040 134 8.5 420 450 521 2.2;
20 1763 5030 121 10.4 420 450 545 1.5]; %look in to the temperatures
21
22 T0 = [273.15;273.15;273.15;273.15;273.15;273.15];
23 T02 = [288.15;288.15;288.15;288.15;288.15;288.15];
24
25 Hpf = A_measurementflight(:,2)*0.3048; %m
26 Machf = M(A_measurementflight);
27 fuelleftf = A_measurementflight(:,5)*0.000125998; %kg/s
28 fuelrightf = A_measurementflight(:,6)*0.000125998;%kg/s
29 aewf = (ones(6,1)+0.2*Machf.^2);
30 Delta_Tf = (A_measurementflight(:,8)+T0)./aewf - (T02 - 0.0065*Hpf); %
    degrees
31
32 Hpr = A_measurementref(:,2)*0.3048; %m
33 Machr = M(A_measurementref);
34 fuelleftf = A_measurementref(:,5)*0.000125998; %kg/s
35 fuelrightf = A_measurementref(:,6)*0.000125998;%kg/s
36 aewr = (ones(6,1)+0.2*Machr.^2);
37 Delta_Tr = (A_measurementref(:,8)+T0)./aewr - (T02 - 0.0065*Hpr); %degrees
38
39
40
41 %Flight data processing
42 A = (15.911^2)/30.00;
43 clf = cl(pounds_ZFMflight,pounds_FuelStartflight,A_measurementflight);
44 clf2 = clf.^2;
45 cdf = cd(Hpf,Machf,Delta_Tf,fuelleftf,fuelrightf,A_measurementflight);
46
47 alphaf = A_measurementflight(:,4)*pi/180;
48 Xf = [ones(6,1) alphaf];
49 clafq = mldivide(Xf,clf);
50 cl0f = clafq(1,1);
51 claf = clafq(2,1);
52
53 Mf = [ones(6,1) clf2];
54 usef = mldivide(Mf,cdf);
55 cd0f = usef(1,1);

```

```

56 ef = 1/(A*pi*(usef(2,1)));
57
58
59
60 %Reference data processing
61 A = (15.911^2)/30.00;
62 clr = cl(pounds_ZFMr,pounds_FuelStartr,A_measurementref);
63 clr2 = clr.^2;
64 cdr = cd(Hpr,Machr,Delta_Tr,fuelleftr,fuelrighttr,A_measurementref);
65
66
67 alphas = A_measurementref(:,4)*pi/180;
68 Xr = [ones(5,1) alphas(1:5,1)];
69 clarq = mldivide(Xr,clr(1:5,1));
70 cl0r = clarq(1,1);
71 clar = clarq(2,1);
72
73 Mr = [ones(5,1) clr2(1:5,1)];
74 user = mldivide(Mr,cdr(1:5,1));
75 cd0r = user(1,1);
76 er = 1/(A*pi*(user(2,1)));
77
78
79 % subplot(1,2,1);
80 % scatter(A_measurementref(:,4),clr);
81 % hold on;
82 % ar = -2:0.1:12;
83 % plot(ar,(cl0r+clar*ar*pi/180));
84 % title("C_L_\alpha reference data");
85 % ylabel("C_L [-]");
86 % xlabel("\alpha [deg]");
87 % dim = [0.25 0.2 0.35 0.1];
88 % str1 = {'Cruise configuration','Mach range 0.1925-0.4081','Reynolds number
      range 7.9064-16.8090 (10^6)'}; % 7906400.177-16808993.42
89 % annotation('textbox',dim,'String',str1,'FitBoxToText','on');
90 %
91 % subplot(1,2,2);
92 % scatter(A_measurementflight(:,4),clf);
93 % hold on;
94 % af = -2:0.1:12;
95 % plot(af,(cl0f+claf*af*pi/180));
96 % title("C_L_\alpha flight data");
97 % ylabel("C_L [-]");
98 % xlabel("\alpha [deg]");
99 % dim = [0.675 0.2 0.775 0.1];
100 % str2 = {'Cruise configuration','Mach range 0.1972-0.4115','Reynolds number
      range 8.1178-16.9395 (10^6)'}; %8117768.57-16939461.29
101 % annotation('textbox',dim,'String',str2,'FitBoxToText','on');
102
103
104 % subplot(1,2,1);
105 % plot(cdr,clr);
106 % title("Lift over drag curve reference data");
107 % ylabel("C_L [-]");
108 % xlabel("C_D [-]");
109 % dim = [0.25 0.2 0.35 0.1];
110 % str1 = {'Cruise configuration','Mach range 0.1925-0.4081','Reynolds number
      range 7.9064-16.8090 (10^6)'}; % 7906400.177-16808993.42

```

```

111 % annotation('textbox',dim,'String',str1,'FitBoxToText','on');
112 %
113 % subplot(1,2,2);
114 % plot(cdf,clf);
115 % title("Lift over drag curve flight data");
116 % ylabel("C_L [-]");
117 % xlabel("C_D [-]");
118 % dim = [0.675 0.2 0.775 0.1];
119 % str2 = {'Cruise configuration','Mach range 0.1972-0.4115','Reynolds number
range 8.1178-16.9395 (10^6)'}; %8117768.57-16939461.29
120 % annotation('textbox',dim,'String',str2,'FitBoxToText','on');
121
122
123 % 'cla flight data [1/rad]'
124 % 'cla reference data [1/rad]'
125 % cl0 flight data [-]
126 % cl0 reference data
127 % 'Oswald efficiency factor flight data [-]'
128 % 'Oswald efficiency factor reference data [-]'
129 % 'CD0 flight data [-]'
130 % 'CD0 reference data [-]'
131 Jonasdat = [claf;clar;cl0f;cl0r;ef;er;cd0f;cd0r]
132
133 function [W_kg] = W_loc(N_m,pounds_ZFM,pounds_FuelStart,A_measurement)
134 F_used = A_measurement(N_m,7);
135 W_lbm = pounds_ZFM + pounds_FuelStart - F_used;
136 W_kg = W_lbm*0.453592;
137 end
138
139 function [M_TAS] = M(A_measurement)
140 M_ans = zeros(length(A_measurement(:,1)),1);
141 %V_conv =
142 % IAS2CAS = griddedInterpolant(V_conv(:,1),V_conv(:,2));
143 for i = 1:length(A_measurement(:,1))
144     gamma = 1.4; % constant
145     p0 = 101325; % pressure at sea level [Pa]
146     rho0 = 1.2250; % air density at sea level [kg/m^3]
147     lambda = -0.0065; % temperature gradient in ISA [K/m]
148     T0 = 288.15; % temperature at sea level in ISA [K]
149     R = 287.05; % specific gas constant [m^2/sec^2K]
150     g0 = 9.81; % [m/sec^2] (gravity constant)
151     S = 30.00; % wing area [m^2]
152     V_cas = (A_measurement(i,3)-2)*0.514444; % calibrated airspeed [m/s]
153     p = p0*(1+(lambda*(A_measurement(i,2)*0.3048))/T0)^(-g0/(lambda*R));
154     Ms1 = 1 + ((gamma-1)/(2*gamma))*(rho0/p0)*(V_cas)^2;
155     Ms2 = Ms1^(gamma/(gamma-1));
156     Ms3 = Ms2 - 1;
157     Ms4 = 1 + (p0/p)*Ms3;
158     Ms5 = Ms4^((gamma-1)/gamma);
159     Ms6 = Ms5 - 1;
160     Ms7 = (2/(gamma-1))*Ms6;
161     M = sqrt(Ms7);
162     M_ans(i,1) = M;
163 end
164 M_TAS = M_ans;
165 end
166
167 function [Thrustlr] = Thrustcalc(Hp,Mach,Delta_T,fuelleft,fuelright)

```

```

168 %Hp [m],Mach [-],Delta_T[deg],fuelleft[kg/s],fuelright[kg/s]
169 %writing to file
170 array= [Hp; Mach;Delta_T;fuelleft;fuelright];
171 fileID = fopen('matlab.dat','w');
172 for i = 1:length(array)
173 fprintf(fileID,"%d\n",array(i,1));
174 end
175 fclose(fileID);
176
177
178 system('thrust.exe &');
179
180 %output
181 load("thrust.dat");
182 Thrustlr = thrust;
183 end
184
185 function [V_EAS] = VE(A_measurement)
186 V_ans = zeros(length(A_measurement(:,1)),1);
187 %V_conv =
188 % IAS2CAS = griddedInterpolant(V_conv(:,1),V_conv(:,2));
189 for i = 1:length(A_measurement(:,1))
190     gamma = 1.4; % constant
191     p0 = 101325; % pressure at sea level [Pa]
192     rho0 = 1.2250; % air density at sea level [kg/m^3]
193     lambda = -0.0065; % temperature gradient in ISA [K/m]
194     T0 = 288.15; % temperature at sea level in ISA [K]
195     R = 287.05; % specific gas constant [m^2/sec^2K]
196     g0 = 9.81; % [m/sec^2] (gravity constant)
197     S = 30.00; % wing area [m^2]
198     V_cas = (A_measurement(i,3)-2)*0.514444; % calibrated airspeed [m/s]
199     p = p0*(1+(lambda*(A_measurement(i,2)*0.3048))/T0)^(-g0/(lambda*R));
200     Ms1 = 1 + ((gamma-1)/(2*gamma))*(rho0/p0)*(V_cas)^2;
201     Ms2 = Ms1^(gamma/(gamma-1));
202     Ms3 = Ms2 - 1;
203     Ms4 = 1 + (p0/p)*Ms3;
204     Ms5 = Ms4^((gamma-1)/gamma);
205     Ms6 = Ms5 - 1;
206     Ms7 = (2/(gamma-1))*Ms6;
207     M = sqrt(Ms7);
208     T = (A_measurement(i,8)+273.15)/(1+((gamma-1)/2)*M^2);
209     a = sqrt(gamma*R*T);
210     rho = p/(R*T);
211     V_ans(i,1) = M*a*sqrt(rho/rho0);
212 end
213 V_EAS = V_ans;
214 end
215
216 function [V_TAS] = VT(A_measurement)
217 V_ans = zeros(length(A_measurement(:,1)),1);
218 rho_ans = zeros(length(A_measurement(:,1)),1);
219 %V_conv =
220 % IAS2CAS = griddedInterpolant(V_conv(:,1),V_conv(:,2));
221 for i = 1:length(A_measurement(:,1))
222     gamma = 1.4; % constant
223     p0 = 101325; % pressure at sea level [Pa]
224     rho0 = 1.2250; % air density at sea level [kg/m^3]
225     lambda = -0.0065; % temperature gradient in ISA [K/m]

```

```

226     T0 = 288.15; % temperature at sea level in ISA [K]
227     R = 287.05; % specific gas constant [m^2/sec^2K]
228     g0 = 9.81; % [m/sec^2] (gravity constant)
229     S = 30.00; % wing area [m^2]
230     V_cas = (A_measurement(i,3)-2)*0.514444; % calibrated airspeed [m/s]
231     p = p0*(1+(lambda*(A_measurement(i,2)*0.3048))/T0)^(-g0/(lambda*R));
232     Ms1 = 1 + ((gamma-1)/(2*gamma))*(rho0/p0)*(V_cas)^2;
233     Ms2 = Ms1^(gamma/(gamma-1));
234     Ms3 = Ms2 - 1;
235     Ms4 = 1 + (p0/p)*Ms3;
236     Ms5 = Ms4^((gamma-1)/gamma);
237     Ms6 = Ms5 - 1;
238     Ms7 = (2/(gamma-1))*Ms6;
239     M = sqrt(Ms7);
240     T = (A_measurement(i,8)+273.15)/(1+((gamma-1)/2)*M^2);
241     a = sqrt(gamma*R*T);
242     rho_ans(i,1) = p/(R*T);
243     V_ans(i,1) = M*a;
244 end
245 V_TAS = [V_ans rho_ans];
246 end
247
248 function [CD] = cd(Hp,Mach,Delta_T,fuelleft,fuelright,A_measurement)
249 wer = VT(A_measurement);
250 V_t = wer(:,1);
251 rho = wer(:,2);
252 CD_ans = zeros(length(A_measurement(:,1)),1);
253 for i = 1:length(A_measurement(:,1))
254     g0 = 9.81; % [m/sec^2] (gravity constant)
255     S = 30.00; % wing area [m^2]
256     rho0 = 1.2250; % air density at sea level [kg/m^3]
257     A = Thrustcalc(Hp(i,1),Mach(i,1),Delta_T(i,1),fuelleft(i,1),fuelright(i,1));
258     Th = A(1,1)+A(1,2);
259     CD_ans(i,1) = Th/(0.5*(rho(i,1))*(V_t(i,1)^2)*S);
260 end
261 CD = CD_ans;
262 end
263
264
265 function [CL] = cl(pounds_ZFM,pounds_FuelStart,A_measurement)
266 V_e = VE(A_measurement);
267 CL_ans = zeros(length(A_measurement(:,1)),1);
268 for i = 1:length(A_measurement(:,1))
269     g0 = 9.81; % [m/sec^2] (gravity constant)
270     S = 30.00; % wing area [m^2]
271     rho0 = 1.2250; % air density at sea level [kg/m^3]
272     CL_ans(i,1) = (W_loc(i,pounds_ZFM,pounds_FuelStart,A_measurement)*g0)/(0.5*
        rho0*(V_e(i,1)^2)*S);
273 end
274 CL = CL_ans;
275 end

```

```

1 run("statespace_asymmetric.m");
2 load("ours.mat");
3
4
5 aoa = flightdata.vane_AOA.data;
6 t = flightdata.time.data;
7 sampling_rate = 1/(t(2)-t(1));

```

```

8  t_start = 36500;
9  t_end = t_start+1500;
10 t = t(t_start:t_end)-t(t_start);
11
12 hp = flightdata.Dadc1_alt.data(t_start)
13
14
15 V_0 = flightdata.Dadc1_cas.data(t_start)
16
17 ail_0 = flightdata.delta_a.data(t_start);
18 rud_0 = flightdata.delta_r.data(t_start);
19
20 ail = flightdata.delta_a.data(t_start:t_end)-ail_0;
21 ail = ail/180*pi;
22 rud = flightdata.delta_r.data(t_start:t_end)-rud_0;
23 rud = rud/180*pi;
24
25 r_0 = flightdata.Ahrs1_bRollRate.data(t_start);
26 y_0 = flightdata.Ahrs1_bYawRate.data(t_start);
27 roll_0 = flightdata.Ahrs1_Roll.data(t_start);
28
29 r = flightdata.Ahrs1_bRollRate.data(t_start:t_end)-r_0;
30 ya = flightdata.Ahrs1_bYawRate.data(t_start:t_end)-y_0;
31 roll = flightdata.Ahrs1_Roll.data(t_start:t_end)-roll_0;
32
33 y = lsim(asymmetric, [-ail, -rud], t);
34
35
36 tiledlayout(2,2);
37 % nexttile
38 % plot(t, input/pi*180);
39 % title("Elevator deflection from trim trimmed position");
40 % ylabel("[deg]");
41 % xlabel("[s]");
42 nexttile
43 plot(t, ail./pi.*180, t, rud./pi.*180);
44 legend("Aileron", "Rudder");
45 title("Control Surface deflection");
46 ylabel("[deg]");
47 xlabel("[s]");
48 nexttile
49 plot(t, -y(:,2), t, roll);
50 legend("Simulation", "Real flight");
51 title("Roll angle deviation");
52 ylabel("[deg]");
53 xlabel("[s]");
54 nexttile
55 plot(t, -y(:,3), t, -r);
56 legend("Simulation", "Real flight");
57 title("Roll rate");
58 ylabel("[deg/s]");
59 xlabel("[s]");
60 nexttile
61 plot(t, -y(:,4), t, ya);
62 legend("Simulation", "Real flight");
63 title("Yaw rate");
64 ylabel("[deg/s]");
65 xlabel("[s]");

```



```

1
2
3 Asimp_sp = zeros(2,2);
4
5 Dc = c/V0;
6
7 Asimp_sp(1,1)=CZa/(2*muc*Dc);
8 Asimp_sp(1,2)=1/Dc;
9 Asimp_sp(2,1)=(CZa*Cmadot/(4*muc^2*KY2*Dc))+(Cma/(2*muc*KY2*Dc));
10 Asimp_sp(2,2)=(Cmadot+Cmq)/(2*muc*KY2*Dc);
11
12 Asimp_sp
13 eig(Asimp_sp)
14
15 Asimp_dr = zeros(2,2);
16
17 Db = b/V0;
18
19 Asimp_dr(1,1) = CYb/(2*mub*Db);
20 Asimp_dr(1,2) = -2/Db;
21 Asimp_dr(2,1) = Cnb/(4*mub*KZ2*Db);
22 Asimp_dr(2,2) = Cnr/(4*mub*KZ2*Db);
23
24 Asimp_dr
25 eig(Asimp_dr)
26
27 C1_dr = zeros(2,2);
28 C1_dr(1,1)=-2*mub*Db;
29 C1_dr(2,2)=-4*mub*KZ2*Db;
30
31 C2_dr = zeros(2,2);
32
33 C2_dr(1,1)=CYb;
34 C2_dr(1,2)=-4*mub;
35 C2_dr(2,1)=Cnb;
36 C2_dr(2,2)=Cnr;
37
38 Asimp_dr2 = -inv(C1_dr)*C2_dr
39 Asimp_dr
40 eig(Asimp_dr)
41
42
43 C1_ph = zeros(3,3);
44 C1_ph(1,1) = -2*muc*Dc;
45 C1_ph(3,2)=-Dc;
46
47 C2_ph = zeros(3,3);
48 C2_ph(1,1)=CXu;
49 C2_ph(1,2)=CZ0;
50 C2_ph(2,1)=CZu;
51 C2_ph(2,3)=2*muc;
52 C2_ph(3,3)=Cmq;
53
54 Asimp_ph_inv = -C1_ph*inv(C2_ph)
55 eigenv=eig(Asimp_ph_inv)
56 eigenv.^(-1)

1 % Citation 550 – Linear simulation
2

```

```

3 % xcg = 0.25*c
4
5 % Stationary flight condition
6
7 hp0      = (6.6539e+03)*0.304;           % pressure altitude in the stationary
      flight condition [m]
8 V0       = 159.5755;                   % true airspeed in the stationary flight
      [m/sec]
9 alpha0   = 5.3936/180*pi;              % angle of attack in the stationary flight
      condition [rad]
10 th0     = 4.3298/180*pi;              % pitch angle in the stationary flight
      condition [rad]
11
12 % Aircraft mass
13 m        = 6317;                       % mass [kg]
14
15 % aerodynamic properties
16 %e        = 0.8;                       %cit_par      % Oswald factor [ ]
17 e         = 0.9670;                   %flight
18 %e        = 0.8860;                   %reference
19
20 %CD0      = 0.04; %cit_par              % Zero lift drag coefficient [ ]
21 CD0       = 0.0220; %flight
22 %CD0      = 0.0225; %reference
23
24 %CLa      = 5.084; %cit_par              % Slope of CL-alpha curve [ ]
25 CLa       = 4.583; %flight
26 %CLa      = 4.713; %reference
27
28 % Longitudinal stability
29 Cma       = -0.6453; %-0.5626;          % longitudinal stability [ ]
30 Cmde      = -1.4227; %-1.4335;          % elevator effectiveness [ ]
31
32 % Aircraft geometry
33
34 S         = 30.00;                     % wing area [m^2]
35 Sh        = 0.2*S;                     % stabiliser area [m^2]
36 Sh_S      = Sh/S;                      % [ ]
37 lh        = 0.71*5.968;                % tail length [m]
38 c         = 2.0569;                     % mean aerodynamic cord [m]
39 lh_c      = lh/c;                      % [ ]
40 b         = 15.911;                     % wing span [m]
41 bh        = 5.791;                     % stabiliser span [m]
42 A         = b^2/S;                      % wing aspect ratio [ ]
43 Ah        = bh^2/Sh;                    % stabiliser aspect ratio [ ]
44 Vh_V      = 1;                          % [ ]
45 ih        = -2*pi/180;                  % stabiliser angle of incidence [rad]
46
47 % Constant values concerning atmosphere and gravity
48
49 rho0      = 1.2250;                     % air density at sea level [kg/m^3]
50 lambda    = -0.0065;                     % temperature gradient in ISA [K/m]
51 Temp0     = 288.15;                     % temperature at sea level in ISA [K]
52 R         = 287.05;                     % specific gas constant [m^2/sec^2K]
53 g         = 9.81;                       % [m/sec^2] (gravity constant)
54
55 rho       = rho0*((1+(lambda*hp0/Temp0)))^(-(g/(lambda*R))+1)); % [kg/m^3] (air
      density)

```

```

56 W          = m*g;                                     % [N]          (
    aircraft_weight)

57
58 % Constant values concerning aircraft inertia
59
60 muc        = m/(rho*S*c);
61 mub        = m/(rho*S*b);
62 KX2        = 0.019;
63 KZ2        = 0.042;
64 KXZ        = 0.002;
65 KY2        = 1.25*1.114;
66
67 % Aerodynamic constants
68
69 Cmac       = 0;                                     % Moment coefficient about the aerodynamic
    centre [ ]
70 CNwa      = CLa;                                     % Wing normal force slope [ ]
71 CNha      = 2*pi*Ah/(Ah+2);                         % Stabiliser normal force slope [ ]
72 depsda    = 4/(A+2);                                % Downwash gradient [ ]
73
74 % Lift and drag coefficient
75 CL0 = 0.0807;
76
77 CL = CL0+CLa*alpha0;%2*W/(rho*V0^2*S);              % Lift coefficient [ ]
78 CD = CD0 + (CL)^2/(pi*A*e); % Drag coefficient [ ]
79
80 % Stability derivatives
81
82 CX0       = CL*sin(th0);
83 CXu       = -2*CL*sin(th0); %-0.095;
84 CXa       = CL+(2*CLa)/(pi*A*e)*CL; %+0.47966;
85 CXadot    = +0.08330;
86 CXq       = -0.28170;
87 CXde      = -0.03728;
88
89 CZ0       = -W*cos(th0)/(0.5*rho*V0^2*S);
90 CZu       = -2*CL*cos(th0); %-0.37616;
91 CZa       = -CLa-CD; %-5.74340;
92 CZadot    = -0.00350;
93 CZq       = -4.545; %-5.66290;
94 CZde      = -0.69612;
95
96 Cmu       = +0.06990;
97 Cmadot    = +0.17800;
98 Cmq       = -8.295; %-8.79415;
99
100 CYb       = -0.7500;
101 CYbdot    = 0;
102 CYp       = -0.0304;
103 CYr       = +0.8495;
104 CYda      = -0.0400;
105 CYdr      = +0.2300;
106
107 Clb       = -0.10260;
108 Clp       = -0.71085;
109 Clr       = +0.23760;
110 Clda      = -0.23088;
111 Cldr      = +0.03440;

```

```

112
113 Cnb      =  +0.1348;
114 Cnbdot   =   0      ;
115 Cnp      =  -0.0602;
116 Cnr      =  -0.2061;
117 Cnda     =  -0.0120;
118 Cndr     =  -0.0939;

1  %This script calculates the elevator control derivative Cm_de
2
3  % Aircraft Data
4  g = 9.81;
5  S = 30;
6  c = 2.0569;
7  b = 15.911;
8
9  % Inputs
10 TAT = 5
11 hp = 6610*0.3048
12 p0=101325
13 R = 287
14 gamma = 1.4
15 rho0 = 1.225
16 m = 6285.5; % result is independent of mass
17 %rho = 0.6601;
18 vc =160*0.51444 ;    % TAS=113 , IAS=82
19 %TAS calculation
20 p = p0*((1-(0.0065*hp)/273.15)^(-9.81/(-0.0065*R)));
21 M025 = 1 + ((gamma-1)/(2*gamma)) * (rho0/p0) * vc.^2;
22 M05 = M025.^(gamma/(gamma-1));
23 M1 = M05 - 1;
24 M2 = 1 + (p0./p).*M1;
25 M3 = M2.^((gamma-1)/gamma);
26 M4 = M3 - 1 ;
27 M5 = (2/(gamma-1))*M4 ;
28 M6 = sqrt(M5);
29
30 Tlocal = (TAT+273.15)/(1+(gamma-1)/2*M6^2);
31
32 asound = sqrt(gamma*R*Tlocal);
33 rho = p/(R*Tlocal);
34 TAS = M6*asound;
35 %TAS
36 %calculation
37 x_cg = 261.56*0.0254+0.25*c ;
38 m_pax = 99;
39 x1 = 288*0.0254;
40 x2 = 134*0.0254;
41 de1 = 0*(pi/180);
42 de2 = -0.5*(pi/180);
43
44 Cm_de = calcCm_de(g,c,m,rho,TAS,S,x_cg,m_pax,x1,x2,de1,de2)
45
46 function [Cm_de] = calcCm_de(g,c,m,rho,TAS,S,x_cg,m_pax,x1,x2,de1,de2)
47
48 Cn = m*g/(0.5*rho*TAS^2*S);
49
50 DeltaCm = -Cn / c * -1.666057659*0.0254 %m_pax*(x2-x1)/(m*g);
51 DeltaDe = de2-de1;

```

```

52
53 Cm_de = DeltaCm / DeltaDe;
54
55 end

1 run("statespace_asymmetric.m");
2 load("ours.mat");
3
4
5 aoa = flightdata.vane_AOA.data;
6 t = flightdata.time.data;
7 sampling_rate = 1/(t(2)-t(1));
8 t_start = 33950;
9 t_end = t_start+250;
10 t = t(t_start:t_end)-t(t_start);
11
12 hp = flightdata.Dadc1_alt.data(t_start);
13
14
15 V_0 = flightdata.Dadc1_cas.data(t_start);
16
17 ail_0 = flightdata.delta_a.data(t_start);
18 rud_0 = flightdata.delta_r.data(t_start);
19
20 ail = flightdata.delta_a.data(t_start:t_end)-ail_0;
21 ail = ail/180*pi;
22 rud = flightdata.delta_r.data(t_start:t_end)-rud_0;
23 rud = rud/180*pi;
24
25 r_0 = flightdata.Ahrs1_bRollRate.data(t_start);
26 y_0 = flightdata.Ahrs1_bYawRate.data(t_start);
27 roll_0 = flightdata.Ahrs1_Roll.data(t_start);
28
29 r = flightdata.Ahrs1_bRollRate.data(t_start:t_end)-r_0;
30 ya = flightdata.Ahrs1_bYawRate.data(t_start:t_end)-y_0;
31 roll = flightdata.Ahrs1_Roll.data(t_start:t_end)-roll_0;
32
33 y = lsim(asymmetric, [ail, rud], t);
34
35
36 tiledlayout(2,2);
37 % nexttile
38 % plot(t, input/pi*180);
39 % title("Elevator deflection from trim trimmed position");
40 % ylabel("[deg]");
41 % xlabel("[s]");
42 nexttile
43 plot(t, ail./pi.*180, t, rud./pi.*180);
44 legend("Aileron", "Rudder");
45 title("Control Surface deflection");
46 ylabel("[deg]");
47 xlabel("[s]");
48 nexttile
49 plot(t, -y(:,2), t, roll);
50 legend("Simulation", "Real flight");
51 title("Roll angle deviation");
52 ylabel("[deg]");
53 xlabel("[s]");
54 nexttile

```

```

55 plot(t, -y(:,3), t, r);
56 legend("Simulation", "Real flight");
57 title("Roll rate");
58 ylabel("[deg/s]");
59 xlabel("[s]");
60 nexttile
61 plot(t, -y(:,4), t, ya);
62 legend("Simulation", "Real flight");
63 title("Yaw rate");
64 ylabel("[deg/s]");
65 xlabel("[s]");

1 clear all;
2 close all;
3 clc;
4 % For this program to work you need to uncommment either the first or the
5 % second two beneath. First two refer to the reference data and last two to
6 %"our" flight
7
8 % load('matlab.mat')
9 % run('refdataone.m');
10
11 load('ours.mat')
12 run('flightdataone.m')
13
14
15 A = [100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400 1500 1600 1700
16       1800 1900 2000 2100 2200 2300 2400 2500];
17
18 B = [298.16 591.18 879.08 1165.42 1448.40 1732.53 2014.80 2298.84 2581.92
19       2866.30 3150.18 3434.52 3718.52 4003.23 4287.76 4572.24 4856.56 5141.16
20       5425.64 5709.90 5994.04 6278.47 6562.82 6846.96 7131.00];
21
22 C = [2600 2700 2800 2900 3000 3100 3200 3300 3400 3500 3600 3700 3800 3900 4000
23       4100 4200 4300 4400 4500 4600 4700 4800 4900 5008];
24
25 D = [7415.33 7699.60 7984.34 8269.06 8554.05 8839.04 9124.80 9410.62 9696.97
26       9983.40 10270.08 10556.84 10843.87 11131.00 11418.20 11705.50 11993.31
27       12281.18 12569.04 12856.86 13144.73 13432.48 13720.56 14008.46 14320.34];
28
29 A_fuel = [transpose(A) transpose(B); transpose(C) transpose(D)];
30
31 Wlh = flightdata.lh_engine_FMF.data/3600; %pounds/s
32 Wrh = flightdata.rh_engine_FMF.data/3600; %pounds/s
33 time = flightdata.time.data; %s
34
35 cgmin = cg(9.1, A_payloadflight, BEM, pounds_FuelStartflight, pounds_ZFMflight,
36           A_fuel, Wlh, Wrh)
37 cgmax = cg(5434.9, A_payloadflight, BEM, pounds_FuelStartflight, pounds_ZFMflight,
38           A_fuel, Wlh, Wrh)
39
40 function [Wans] = W(t, pounds_ZFM, pounds_FuelStart, Wlh, Wrh)
41 %This function calculates  $W(t) = \text{ramp mass} - \int_0^t (\text{fuelflow} * dt)$ 
42 %The answer is in [lbs kg]
43 time = 9:0.1:t;
44 Wlhu = Wlh(1:length(time), 1);
45 Wrhu = Wrh(1:length(time), 1);

```

```

40 F_used = trapz(time,Wlhu)+trapz(time,Wrhs);
    %needs to look at time in
41 %measurement data and give F_used for that t —> FTIS
42 W_lbm = pounds_ZFM + pounds_FuelStart - F_used;
43 W_kg = W_lbm*0.453592;
44 Wans = [W_lbm W_kg];
45 end
46
47
48
49 function [cgans] = cg(t,A_payload,BEM,pounds_FuelStart,pounds_ZFM,A_fuel,Wlh,Wrh
    )
50 %This function calculates the cg location and gives at in
51 %[inch meter fractionofMAC]
52 time = 9:0.1:t;
53 Wlhu = Wlh(1:length(time),1);
54 Wrhs = Wrh(1:length(time),1);
55 F_used = trapz(time,Wlhu)+trapz(time,Wrhs); %FTIS must
    give this
56 Fuel = pounds_FuelStart - F_used; %pounds
57 s = transpose(A_payload(:,2));
58 b = A_payload(:,1);
59 Fuel2M = griddedInterpolant(A_fuel(:,1),A_fuel(:,2)*100); %pound-inch
60 Mtot = s*b + Fuel2M(Fuel) + BEM(1,1)*BEM(1,2);
61 a = W(t,pounds_ZFM,pounds_FuelStart,Wlh,Wrhs);
62 cg_in = Mtot/a(1,1) - 261.56; %inch from the leading edge of the MAC
63 cg_m = cg_in*0.0254;
64 cg_pMAC = cg_in/80.98;
65 cgans = [cg_in; cg_m; cg_pMAC];
66 end
67
68
69
70
71
72 %inputs in pounds unless other unit is stated behind it
73
74
75 pounds_seat1 = 80/0.453592;
76 pounds_seat2 = 102/0.453592;
77 pounds_seat3 = 71/0.453592;
78 pounds_seat4 = 77/0.453592;
79 pounds_seat5 = 76/0.453592;
80 pounds_seat6 = 64/0.453592;
81 pounds_seat7 = 74/0.453592;
82 pounds_seat8 = 99/0.453592;
83 pounds_seat10 = 60/0.453592;
84 pounds_nose = 0;
85 pounds_aft1 = 0;
86 pounds_aft2 = 0;
87 pounds_FuelStartflight = 4100;
88
89
90 %seats contain 9x2 matrix with [x_cg_fromfront[inches],mass[pounds]] per
91 %seat
92 seatsflight = [131 pounds_seat1; 131 pounds_seat2; 214 pounds_seat3; 214
    pounds_seat4; 251 pounds_seat5; 251 pounds_seat6; 288 pounds_seat7; 288
    pounds_seat8; 170 pounds_seat10];
93
94
95 baggageflight = [74 pounds_nose; 321 pounds_aft1; 338 pounds_aft2];
96
97 A_payloadflight = [seatsflight; baggageflight];
98
99 pounds_payloadflight = sum(A_payloadflight(:,2));

```

```

27
28 BEM = [9165.0 291.65];           % [pounds inches]
29
30 pounds_ZFMflight = pounds_payloadflight + BEM(1,1);

1  % Ap = 4*muc^2*KY2;
2  % Bp = -2*muc*(KY2*CZa+Cmadot+Cmq);
3  % Cp = CZa*Cmq -2*muc*Cma;
4
5  Ap = 8*mub^2*KZ2;
6  Bp = -(4*mub*KZ2*CYb+2*mub*Cnr);
7  Cp = CYb*Cnr +4*mub*Cnb;
8
9  syms lam;
10
11 p = Ap*lam^2+Bp*lam+Cp;
12 R = solve(p,lam);
13 Rnumeric = vpa(R)

1  run("statespace_symmetric.m");
2  load("ours.mat");
3
4
5  aoa = flightdata.vane_AOA.data;
6  t = flightdata.time.data;
7  sampling_rate = 1/(t(2)-t(1));
8  t_start = 29980;
9  t_end = 31530;
10 t = t(t_start:t_end)-t(t_start);
11
12 hp = flightdata.Dadc1_alt.data(t_start)
13
14
15 V_0 = flightdata.Dadc1_cas.data(t_start)
16 alpha_0 = aoa(t_start)
17 elev_0 = flightdata.delta_e.data(t_start);
18 q_0 = flightdata.Ahrs1_bPitchRate.data(t_start);
19 th_0 = flightdata.Ahrs1_Pitch.data(t_start)
20
21
22 input = flightdata.delta_e.data(t_start:t_end)-elev_0;
23 input = input/180*pi;
24 tas = flightdata.Dadc1_cas.data(t_start:t_end)-V_0;
25 aoa = aoa(t_start:t_end)-alpha_0;
26 q = flightdata.Ahrs1_bPitchRate.data(t_start:t_end)-q_0;
27 th = flightdata.Ahrs1_Pitch.data(t_start:t_end)-th_0;
28
29 y = lsim(symmetric, input, t);
30 eig(symmetric.A)
31
32
33 title("Phugoid")
34 tiledlayout(2,2);
35 % nexttile
36 % plot(t, input/pi*180);
37 % title("Elevator deflection from trim trimmed position");
38 % ylabel("[deg]");
39 % xlabel("[s]");
40 nexttile

```



```

41 plot(t, y(:,1), t, tas);
42 legend("Simulation", "Real flight");
43 title("Speed deviation");
44 ylabel("[m/s]");
45 xlabel("[s]");
46 nexttile
47 plot(t, y(:,2), t, aoa);
48 legend("Simulation", "Real flight");
49 title("Angle of attack deviation");
50 ylabel("[deg]");
51 xlabel("[s]");
52 nexttile
53 plot(t, y(:,3), t, th);
54 legend("Simulation", "Real flight");
55 title("Pitch deviation");
56 ylabel("[deg]");
57 xlabel("[s]");
58 nexttile
59 plot(t, y(:,4), t, q);
60 legend("Simulation", "Real flight");
61 title("Pitch rate deviation");
62 ylabel("[deg/s]");
63 xlabel("[s]");
64
65 % clc
66 % close all
67 % plot(t, input/pi*180);
68 % title("Elevator deflection from trim trimmed position");
69 % ylabel("[deg]");
70 % xlabel("[s]");

1 pounds_seat1r = 95/0.453592;
2 pounds_seat2r = 92/0.453592;
3 pounds_seat3r = 66/0.453592;
4 pounds_seat4r = 61/0.453592;
5 pounds_seat5r = 75/0.453592;
6 pounds_seat6r = 78/0.453592;
7 pounds_seat7r = 86/0.453592;
8 pounds_seat8r = 68/0.453592;
9 pounds_seat10r = 74/0.453592;
10 pounds_noser = 0;
11 pounds_aft1r = 0;
12 pounds_aft2r = 0;
13 pounds_FuelStartr = 4050;
14
15 %seats contain 9x2 matrix with [x_cg_fromfront[inches],mass[pounds]] per
16 %seat
17 seatsr = [131 pounds_seat1r; 131 pounds_seat2r; 214 pounds_seat3r; 214
            pounds_seat4r; 251 pounds_seat5r; 251 pounds_seat6r; 288 pounds_seat7r; 288
            pounds_seat8r; 170 pounds_seat10r];
18
19 baggager = [74 pounds_noser; 321 pounds_aft1r; 338 pounds_aft2r];
20
21 A_payload = [seatsr; baggager];
22
23 pounds_payloadr = sum(A_payload(:,2));
24
25 BEM = [9165.0 291.65]; % [pounds inches]
26

```

```

27 pounds_ZFMr = pounds_payloadr + BEM(1,1);

1 clear all;
2 close all;
3 clc;
4
5 %run("Cit_par.m")
6
7 %parameters\
8 R = 287;
9 Cmdelta = -1.4227;% Also change in functions
10 Ws= 60500 ;%N
11 M_empty = 9165*0.45359 ;%kg
12 d = 0.686;
13 rho0 = 1.225;
14 gamma = 1.4;
15 p0 = 101325;
16 %Payload+fuel
17 Wet_mass0= 2562.7; %kg
18 %Initial mass
19 W0 = (6329.8)*9.81; %N
20
21
22
23 %hp,IAS,a,de,detr,Fe,FFl,FFr,Fused,TAT
24 data = [6510      161      5.4      0.1      3.3      0      454      500      636
1.8;
25 6780      150      6.5      -0.4      3.3      -20      450      496      662      0.2;
26 7200      140      7.4      -1      3.3      -34      440      488      697      -0.5;
27 7540      130      8.6      -1.3      3.3      -45      437      484      721      -1.8;
28 6900      170      4.8      0.4      3.3      23      455      500      745      1.5;
29 6540      180      4      0.7      3.3      33      460      505      764      2.5;
30 6050      190      3.5      0.9      3.3      64      469      515      790      4];
31
32
33 hp = transpose(data(:,1))*0.3048; %m
34 Vc = (transpose(data(:,2))-2)*0.51444;%kts to m/s
35 alpha = transpose(data(:,3))/180*pi;%deg to rad
36 de = transpose(data(:,4))/180*pi;%deg to rad
37 Fe = transpose(data(:,6)); %N
38 FFl = transpose(data(:,7))*1.25997e-4; %lbs/hr to kg/s
39 FFr = transpose(data(:,8))*1.25997e-4; %lbs/hr to kg/s
40 TAT = transpose(data(:,10));
41 Fe_reduced = [];
42 Fuel_standard = 0.048
43
44
45 %Reducing the Force
46 for a=1:length(Fe) ;
47     temp = Fe(a)*Ws/(W0-data(a,9)*4.45);
48     Fe_reduced = [Fe_reduced temp];
49 end
50
51
52 %Reducing Velocity
53 V_reduced = [];
54 Mach = [];
55 TAS = [];
56 EAS = [];

```

```

57 Tcorrect = [];
58 for a=1:length(Vc);
59 p = p0*((1-(0.0065*hp(a))/288.15)^(-9.81/(-0.0065*R)));
60 M025 = 1 + ((gamma-1)/(2*gamma)) * (rho0/p0) * Vc(a).^2;
61 M05 = M025.^(gamma/(gamma-1));
62 M1 = M05 - 1;
63 M2 = 1 + (p0./p).*M1;
64 M3 = M2.^((gamma-1)/gamma);
65 M4 = M3 - 1;
66 M5 = (2/(gamma-1))*M4;
67 M6 = sqrt(M5);
68 Mach = [Mach M6];
69 Tlocal = (TAT(a)+273.15)/(1+(gamma-1)/2*M6^2);
70 Tcorrect = [Tcorrect Tlocal];
71 asound = sqrt(gamma*R*Tlocal);
72 rho = p/(R*Tlocal);
73 TASl = M6*asound;
74 TAS = [TAS TASl];
75 EASl = TASl*sqrt(rho/rho0);
76 EAS = [EAS EASl];
77 temp = EASl*sqrt(Ws/(W0-data(a,9)*4.45));
78 V_reduced = [V_reduced temp];
79 end
80
81 %Reducing Elevator
82 de_reduced = []
83 for b=1:length(Vc);
84 Delta_T = Tcorrect(b)-(288.15-0.0065*hp(b))
85 T = Thrustcalc(hp(b),Mach(b),Delta_T,FFl(b),FFr(b));
86 T_stand = Thrustcalc(hp(b),Mach(b),Delta_T,Fuel_standard,Fuel_standard);
87 detemp = elreduced(de(b),sum(T),sum(T_stand),EAS(b),rho0,d);
88 de_reduced = [de_reduced detemp];
89 end
90
91 %\delta_{e_{e q}}^{*} \text{ versus } \tilde{V}_{e}
92 %plotting
93 %elevator deflection
94 xq = min(V_reduced):0.01:max(V_reduced);
95 yq = interp1(V_reduced, de_reduced,xq,"spline");
96 figure();
97 hold on;
98 scatter(V_reduced,de_reduced);
99 plot(xq,yq);
100 hold off;
101 xlabel('V_{e}');
102 ylabel('\delta_{e_{e q}}^{*} (rad)');
103 grid on;
104 ax = gca;
105 set(ax, 'Ydir', 'reverse');
106 ax.XAxisLocation = 'origin';
107 ax.YAxisLocation = 'origin';
108
109 %Stick force
110 xq2 = min(V_reduced):0.01:max(V_reduced);
111 yq2 = interp1(V_reduced, Fe_reduced,xq,"spline");
112 figure();
113 hold on;
114 scatter(V_reduced,Fe_reduced);

```

```

115 plot(xq2,yq2);
116 hold off;
117 xlabel('V_{e}');
118 ylabel('(\bar{F}_{e}^{\ast})(N)');
119 grid on;
120 ax2 = gca;
121 set(ax2, 'Ydir', 'reverse');
122 ax2.XAxisLocation = 'origin';
123 ax2.YAxisLocation = 'origin';
124
125
126
127 [r,m,b] = regression(alpha,de);
128
129
130
131 x = 0:0.01:0.2;
132 y = m*x+b;
133
134 figure();
135 scatter(alpha,de);
136 hold on;
137 plot(x,y);
138 hold off;
139 xlabel("\alpha (rad)");
140 ylabel("\delta_{e} (rad)");
141
142
143 grid on;
144 ax2 = gca;
145 set(ax2, 'Ydir', 'reverse');
146 ax2.XAxisLocation = 'origin';
147 ax2.YAxisLocation = 'origin';
148
149
150 Cmalpha = -Cmdelta*m;
151 m;
152
153
154 function [Thrustlr] = Thrustcalc(Hp,Mach,Delta_T,fuelleft,fuelright)
155 %Hp [m],Mach [-],Delta_T[deg],fuelleft[kg/s],fuelright[kg/s]
156 %writing to file
157 array= [Hp, Mach,Delta_T,fuelleft,fuelright];
158 fileID = fopen('matlab.dat','w');
159 for i = 1:1:length(array);
160 fprintf(fileID,"%d\n",array(i));
161 end
162 fclose(fileID);
163
164
165 system('thrust.exe &');
166 close all;
167
168 %output
169 load("thrust.dat");
170 Thrustlr = thrust;
171 end
172

```

```

173 function [dereduced] = elreduced(de,T,T_stand,Vlocal, rho0,d);
174
175 CmT = -0.0064;
176 Cmdelta = -1.4227;
177 Tcs = T_stand/(0.5*rho0*Vlocal^2*d^2)
178 Tc = T/(0.5*rho0*Vlocal^2*d^2) % N ;
179 dereduced = de - 1/Cmdelta*CmT*(Tcs-Tc)
180
181 end

1 run("statespace_symmetric.m");
2 load("ours.mat");
3
4
5 aoa = flightdata.vane_AOA.data;
6 t = flightdata.time.data;
7 sampling_rate = 1/(t(2)-t(1));
8 t_start = 28100+18;
9 t_end = t_start+100;
10 t = t(t_start:t_end)-t(t_start);
11
12 hp = flightdata.Dadc1_alt.data(t_start)
13
14
15 V_0 = flightdata.Dadc1_cas.data(t_start)
16 alpha_0 = aoa(t_start)
17 elev_0 = flightdata.delta_e.data(t_start)
18 q_0 = flightdata.Ahrs1_bPitchRate.data(t_start);
19 th_0 = flightdata.Ahrs1_Pitch.data(t_start);
20
21
22 input = flightdata.delta_e.data(t_start:t_end)-elev_0;
23 input = input/180*pi;
24 tas = flightdata.Dadc1_cas.data(t_start:t_end)-V_0;
25 aoa = aoa(t_start:t_end)-alpha_0;
26 q = flightdata.Ahrs1_bPitchRate.data(t_start:t_end)-q_0;
27 th = flightdata.Ahrs1_Pitch.data(t_start:t_end)-th_0;
28
29 y = lsim(symmetric, input, t);
30
31
32 title("Phugoid")
33 tiledlayout(2,2);
34 % nexttile
35 % plot(t, input/pi*180);
36 % title("Elevator deflection from trim trimmed position");
37 % ylabel("[deg]");
38 % xlabel("[s]");
39 nexttile
40 plot(t, y(:,1), t, tas);
41 legend("Simulation", "Real flight");
42 title("Speed deviation");
43 ylabel("[m/s]");
44 xlabel("[s]");
45 nexttile
46 plot(t, y(:,2), t, aoa);
47 legend("Simulation", "Real flight");
48 title("Angle of attack deviation");
49 ylabel("[deg]");

```

```

50 xlabel("[s]");
51 nexttile
52 plot(t, y(:,3), t, th);
53 legend("Simulation", "Real flight");
54 title("Pitch deviation");
55 ylabel("[deg]");
56 xlabel("[s]");
57 nexttile
58 plot(t, y(:,4), t, q);
59 legend("Simulation", "Real flight");
60 title("Pitch rate deviation");
61 ylabel("[deg/s]");
62 xlabel("[s]");
63
64 clc
65 close all
66 plot(t, input/pi*180);
67 title("Elevator deflection from trim trimmed position");
68 ylabel("[deg]");
69 xlabel("[s]");

1 clear all;
2 close all;
3 clc;
4
5 run("Cit_par.m")
6
7 % Asymmetric Motion
8 M = eye(4);
9 M(3,3) = 2*V0/b;
10 M(4,4) = 2*V0/b;
11 M = inv(M);
12
13 Db = b/V0;
14
15 C1 = zeros(4,4);
16 C1(1,1) = (CYb-2*mub)*Db;
17 C1(2,2) = -0.5*Db;
18 C1(3,3) = -4*mub*KX2*Db;
19 C1(4,4) = -4*mub*KZ2*Db;
20 C1(4,1) = Cnb*Db;
21 C1(4,3) = 4*mub*KXZ*Db;
22 C1(3,4) = 4*mub*KXZ*Db;
23
24 C2 = zeros(4,4);
25 C2(1,1) = CYb;
26 C2(1,2) = CL;
27 C2(1,3) = CYP;
28 C2(1,4) = CYr-4*mub;
29
30 C2(2,1) = 0;
31 C2(2,2) = 0;
32 C2(2,3) = 1;
33 C2(2,4) = 0;
34
35 C2(3,1) = Clb;
36 C2(3,2) = 0;
37 C2(3,3) = Clp;
38 C2(3,4) = Clr;

```

```

39
40 C2(4,1) = Cnb;
41 C2(4,2) = 0;
42 C2(4,3) = Cnp;
43 C2(4,4) = Cnr;
44
45 C1 = C1*M;
46 C2 = C2*M;
47
48 C3 = [CYda, CYdr;
49       0, 0;
50       Clda, Cldr;
51       Cnda, Cndr];
52
53 A = -1*inv(C1)*C2;
54 B = -1*inv(C1)*C3;
55
56 C = eye(4);
57 C(1,1) = 180/pi;
58 C(2,2) = 180/pi;
59 C(3,3) = 180/pi;
60 C(4,4) = 180/pi;
61 D = [0, 0;
62       0, 0;
63       0, 0;
64       0, 0];
65
66 asymmetric = ss(A,B,C,D);
67 asymmetric.OutputName = ["Side Slip Angle", "Bank Angle", "Roll Rate", "Yaw Rate
68                             "];
69
70 eig(asymmetric.A)
71 t = 0:0.01:30;
72 [P,D] = eig(asymmetric.A)
73 first = P(:,1) .* [exp(1).^(D(1,1)*t)];
74 second = P(:,2) .* [exp(1).^(D(2,2)*t)];
75 third = P(:,3) .* [exp(1).^(D(3,3)*t)];
76 fourth = P(:,4) .* [exp(1).^(D(4,4)*t)];
77
78 tiledlayout(2,2);
79 nexttile
80 plot(t, real(first(1,:)), t, real(second(1,:)), t, real(third(1,:)), t, real(
81     fourth(1,:)))
82 legend("First Eigenmotion", "Second Eigenmotion", "Third Eigenmotion", "Fourth
83     Eigenmotion")
84 title("Sideslip")
85 ylabel("[deg]");
86 xlabel("[s]");
87
88 nexttile
89 plot(t, real(first(2,:)), t, real(second(2,:)), t, real(third(2,:)), t, real(
90     fourth(2,:)))
91 legend("First Eigenmotion", "Second Eigenmotion", "Third Eigenmotion", "Fourth
92     Eigenmotion")
93 title("Roll angle deviation");
94 ylabel("[deg]");
95 xlabel("[s]");
96
97
98
99

```

```

92 nexttile
93 plot(t, real(first(3,:)), t, real(second(3,:)), t, real(third(3,:)), t, real(
    fourth(3,:))
94 legend("First Eigenmotion", "Second Eigenmotion", "Third Eigenmotion", "Fourth
    Eigenmotion")
95 title("Roll rate");
96 ylabel("[deg/s]");
97 xlabel("[s]");
98
99 nexttile
100 plot(t, real(first(4,:)), t, real(second(4,:)), t, real(third(4,:)), t, real(
    fourth(4,:))
101 legend("First Eigenmotion", "Second Eigenmotion", "Third Eigenmotion", "Fourth
    Eigenmotion")
102 title("Yaw Rate");
103 ylabel("[deg]");
104 xlabel("[s]");
105
106 % u = zeros(size(t, 2), 2);
107 % u(5/0.01:7.5/0.01, 1) = 1 * pi/180;
108 % u(7.5/0.01:10/0.01, 1) = -1 * pi/180;
109 % lsim(asymmetric, u, t)

1 clear all;
2 close all;
3 clc;
4
5 run("Cit_par.m")
6
7 % Symmetric Motion
8 M = eye(4);
9 M(1,1) = V0;
10 M(4,4) = V0/c;
11 M = inv(M);
12
13 Dc = c/V0;
14
15 C1 = zeros(4,4);
16 C1(1,1) = -2*muc*Dc;
17 C1(2,2) = (CZa-2*muc)*Dc;
18 C1(3,3) = -Dc;
19 C1(4,2) = Cma*Dc;
20 C1(4,4) = -2*muc*KY2*Dc;
21
22 C2 = zeros(4,4);
23 C2(1,1) = CXu;
24 C2(1,2) = CXa;
25 C2(1,3) = CZ0;
26 C2(1,4) = CXq;
27
28 C2(2,1) = CZu;
29 C2(2,2) = CZa;
30 C2(2,3) = -CX0;
31 C2(2,4) = (CZq+2*muc);
32
33 C2(3,4) = 1;
34
35 C2(4,1) = Cmu;
36 C2(4,2) = Cma;

```



```

37 C2(4,4) = Cmq;
38
39 C1 = C1*M;
40 C2 = C2*M;
41
42 C3 = [CXde; CZde; 0; Cmde];
43
44 A = -1*inv(C1)*C2;
45 B = -1*inv(C1)*C3;
46
47 C = eye(4);
48 C(2,2) = 180/pi;
49 C(3,3) = 180/pi;
50 C(4,4) = 180/pi;
51 D = [0;0;0;0];
52
53 symmetric = ss(A,B,C,D);
54 symmetric.OutputName = ["Velocity deviation", "Angle of Attack", "Pitch angle",
    "Pitch Rate"];
55
56 A
57 t = 0:0.01:5;
58 in = zeros(size(t));
59 in(1:10) = 0.5/180*pi;
60 [P,D] = eig(symmetric.A)
61 first = lsim(symmetric, in, t);P(:,1) .* [exp(1).^(D(1,1)*t)];
62 second = P(:,2) .* [exp(1).^(D(2,2)*t)];
63 third = P(:,3) .* [exp(1).^(D(3,3)*t)];
64 fourth = P(:,4) .* [exp(1).^(D(4,4)*t)];
65 first = first';
66 tiledlayout(4,1);
67 nexttile
68 plot(t, real(first(1,:))%, t, real(second(1,:)), t, real(third(1,:)), t, real(
    fourth(1,:))
69 %legend("First Eigenmotion", "Second Eigenmotion", "Third Eigenmotion", "Fourth
    Eigenmotion")
70 legend("Shortperiod")
71 title("Speed deviation")
72 ylabel("[m/s]");
73 xlabel("[s]");
74
75 nexttile
76 plot(t, real(first(2,:))%, t, real(second(2,:)), t, real(third(2,:)), t, real(
    fourth(2,:))
77 legend("First Eigenmotion", "Second Eigenmotion", "Third Eigenmotion", "Fourth
    Eigenmotion")
78 legend("Shortperiod")
79 title("Angle of attack deviation");
80 ylabel("[deg]");
81 xlabel("[s]");
82
83 nexttile
84 plot(t, real(first(3,:))%, t, real(second(3,:)), t, real(third(3,:)), t, real(
    fourth(3,:))
85 legend("First Eigenmotion", "Second Eigenmotion", "Third Eigenmotion", "Fourth
    Eigenmotion")
86 legend("Shortperiod")
87 title("Pitch deviation");

```

```

88 ylabel("[deg]");
89 xlabel("[s]");
90
91 nexttile
92 plot(t, real(first(4,:))%, t, real(second(4,:)), t, real(third(4,:)), t, real(
    fourth(4,:))
93 legend("First Eigenmotion", "Second Eigenmotion", "Third Eigenmotion", "Fourth
    Eigenmotion")
94 legend("Shortperiod")
95 title("Pitch rate deviation");
96 ylabel("[deg]");
97 xlabel("[s]");
98 %
99 %% u = -0.5/180*pi*ones(size(t));
100 %% u(1:5/0.01) = 0;
101 %% u(7/0.01:end) = 0;
102 %% lsim(symmetric, u, t)
103 %% initial(symetric, [0, 2/180*pi, 2/180*pi, 0], t)
104 %% impulse(symmetric, 100)

```