TÉCNICO LISBOA

Aircraft Altitude Control and Fault Processing

Control of an aircraft's altitude based on its sensor inputs and processing and mitigation of mid-flight system faults

Pedro Afonso **66277**
João Manito **73096**
Daniel de Schiffart **81479**

19th December 2018

Abstract

The objective of the second laboratory for this course was to control an aircraft's altitude and movement based on the input provided by its sensors and the transmission of data throughout the aircraft's onboard systems via its Local Area Network, characterizing the data protocols and preparing data for transmission. The second part of this laboratory had its focus more directed to the possibility of mid-flight faults to various aircraft systems, the handling of these faults and the simulation of data to maintain the control of the aircraft's altitude mentioned in the first part in the presence of faults in any of the crucial systems for this process. The work was to be implemented in MATLAB and its supporting software package *Simulink*.

Contents

Part I
Aircraft Altitude Sensor and On-Board Data Bus

This first part of the report focuses on the gathering of external data, in this case the pressure, through the on-board sensors, modelling of sensor behaviour using gathered data, and the comparison of this theoretical model with the actual data to study the error obtained. This work is followed by the study of the transmission of data through the aircraft's on-board system local network.

1 Modeling The Input Data

For this first section, we were provided a set of gathered voltage data from a sensor at three different temperatures, with each voltage value accompanied by the corresponding static pressure value. In total, a total of 219 lines of data, which means voltage values for 219 different real static pressure values. A sample of the data provided is seen in table 1.

Pressure [mbar]	Output [Volt] at −45°C	Output [Volt] at 25°C	Output [Volt] at 125°C
1.00×10^{1}	9.69×10^{-2}	3.92×10^{-2}	8.05×10^{-2}
⋮	⋮	⋮	⋮

Table 1: Example of the provided data formatting.

The data was to be used to determine a model for the pressure, using the voltage and ambient temperature as model inputs. For this purpose, we used MATLAB's `fit` function from the *Curve Fitting Toolbox*, using a second order polynomial for the temperature axis and a fourth-order polynomial curve for the voltage fitting. This yielded the polynomials found in equation 1.

$$\begin{aligned} p(v,T) = {} & 5.639 + 382.1v - 0.05732T - 69.73v^2 + 0.2646vT + 0.0002634T^2 + 11.53v^3 \\ & -0.05126v^2T - 0.0002063vT^2 - 0.8314v^4 + 0.004329v^3T + 4.644 \times 10^{-5}v^2T^2 \end{aligned} \tag{1}$$

The model was plotted against the provided input data and can be found in figure 3 in page 7.

2 Modelling of a temperature data ADC

For this question we considered the range of values between −45°C and 125°C, which leaves us with a total of 170 values of temperature to be transmitted. With an additional request of an accuracy of 0.10°C, we need ten times as many values. All things considered, we need to transmit any values in a range of 1700 possibilities. Going through the amount of possible values held by a number of binary bits, we look for the amount of bits to meet this requirement.

With this in mind, and considering using an offset of values to eliminate the need for a signal bit (or even converting the temperature to Kelvin), we need at the very least 11 bits to transmit.

$$\begin{array}{ccc} \vdots & = & \vdots \\ 2^{9} & = & 512 \\ 2^{10} & = & 1024 \\ 2^{11} & = & 2048 \\ 2^{12} & = & 4096 \\ \vdots & = & \vdots \end{array}$$

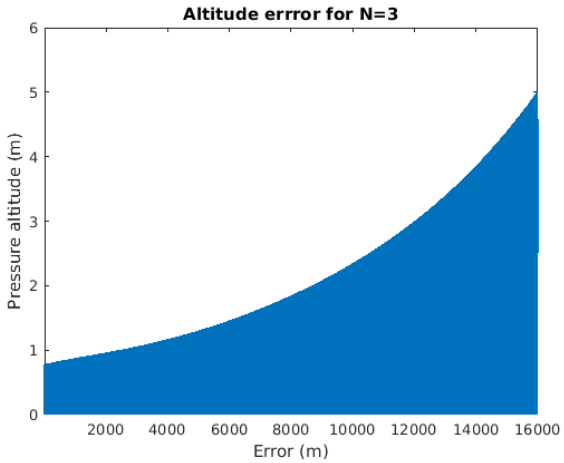

Figure 1: Simulation result error for N=3

3 Evaluation of model error and ADC accuracy in altitude determination

In order to be able to feed the pressure sensor data to the Air Data Computer, a Analog to Digital conversion has to be performed. This in turn leads us to a question: how much resolution is needed for the conversion? Obviously, the minimum ammount of bits necessary must be able to go from 0 to the maximum value necessary. In this case, the ISA mean sea level altitude is defined by a pressure of 1013.25 mbar; As such, at least 10 bits (1024 steps) are necessary to contain the entire range of pressures from sea level to vaccum. However, since there are locations on the planet with pressure altitudes below ISA mean sea level, we decided to require at least 11 bits of resolution, to ensure those locations are correctly included. However, a more accurate pressure determination is required to ensure that pressure altimeter errors are small enough for this system to be used for air travel. For this effect, we defined that the ADC shall have a fixed point output, with 11 integer part bits and N fractional bits. To determine the number of fractional bits required, we created a set of pressure values, simulating the pressure range of the International Standard Atmosphere.

Simulating a pressure sensor with an ADC, we determined the conversion altitude error. The results for N=3 (total 14 bits) and N=4 (total 15 bits) are presented in Figures **??** and **??**. As we can see, both present small errors, in the order of meters. Taking into account that we are working with modeled, ideal sensors, without noise and ignoring a plethora of other factors, we decided to go with N=4 (total 15 bits) as the operational requirement for the number of ADC bits. This allows us to have a larger tolerance for any problems that might arise in the use of the sensor, while still allowing meter-order errors in the altitude determination, enough to ensure a safe vertical separation between aircraft.

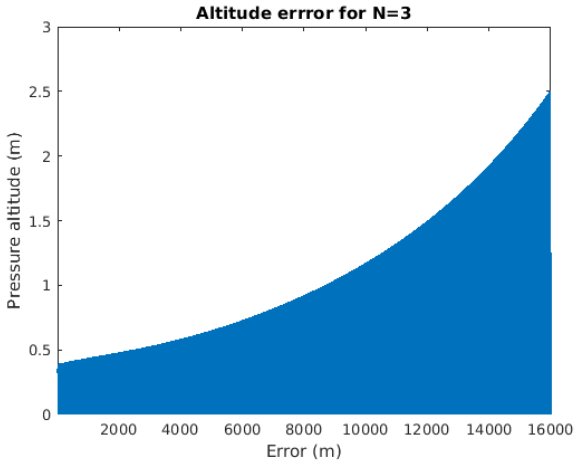

Figure 2: Simulation result error for N=4

4 ARINC 429 Data Bus Sketching

The ARINC 429 Data Bus that was designed in order to provide the Air Data Computer with the needed inputs is presented in Figure 1.

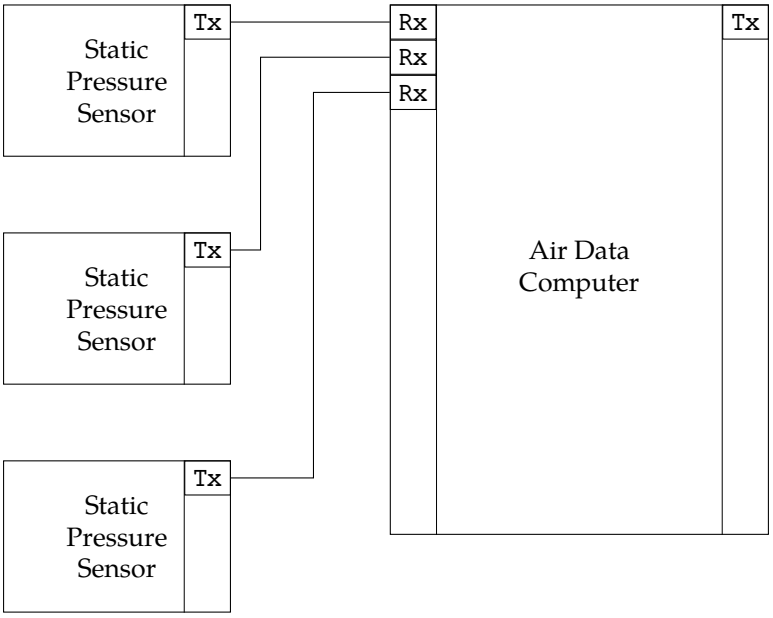

Figure 3: Air Data Computer.

$$h(p) = 145366.45\left(1 - \left(\frac{p_s}{1013.25}\right)^{0.190284}\right)$$

Part II
Dealing With Faults and Errors

For this section of the laboratory project the focus was to analyze and test possible faults within the components of the system and study and implement methods to deal with these

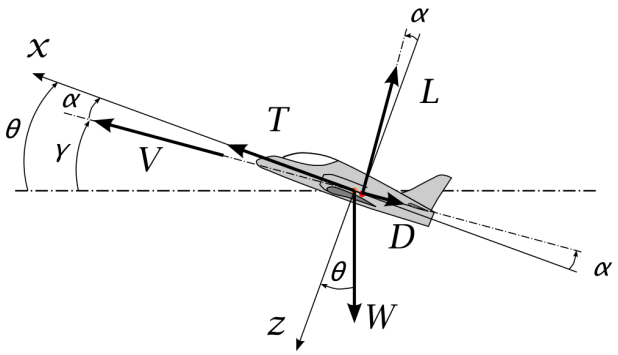

Figure 4: Aircraft body referential and relevant angles used in the description of the aircraft behaviour.

problems. For this very purpose, we needed to design a simple aircraft controller and perform any tests and implementations around it.

5 Modelling The Aircraft's Motion

Per request of the teacher, the objective was to use a simplified model of an aircraft and its behaviour and design and implement an altitude controller. The considered plant variables were the horizontal position x, the altitude h, the horizontal and vertical velocities (or rate of change of both previous variables) $\dot{x}$ and $\dot{h}$, and the flight path angle γ. Also per request, the variables to be controlled were the angle of attack α and the engine power, which we took the liberty of changing to the engine thrust T as a variable of force. The variables were all considered in their correspondent SI units.

The variables and their corresponding units and initial considered values can be found in table 2.

Variable	Description	Unit	Initial Values
x	Horizontal position	m	0
h	Altitude	m	0
$\dot{x}$	Horizontal speed	$m\,s^{-1}$	65.8354
$\dot{h}$	Altitude rate	$m\,s^{-1}$	0
γ	Flight path angle	rad	0
α	Angle of attack	rad	0.052534
T	Thrust force	N	500

Table 2: The variables used in the description of the aircraft behaviour and their respective units.

The dynamic equations were derived from standard aircraft behaviour obtained from other relevant literature. Using the relations found in figure 2, we derived the equations for the basic horizontal movement and altitude rates that can be found in equations 2 and 3.

$$\frac{dh}{dt} = V(t)\sin\gamma(t) \tag{2}$$

$$\frac{dx}{dt} = V(t)\cos\gamma(t) \tag{3}$$

Within these equations we see the first appearance of the V variable, which we will use for the scalar velocity of the aircraft as the Pythagorean sum of both rates of movement referenced in table 2.

Redirecting our attention back to figure 2 we can obtain a value for the resultant force of the aircraft by considering all the forces acting on it in every moment. By considering all the forces in the direction of the movement, using Newton's first law $\mathbf{F} = m\mathbf{a}$, and considering $a = \dot{v}$, we can write down the rate of the aircraft's aforementioned scalar velocity.

$$m\dot{V} = m\frac{dV(t)}{dt} = T - D - mg\sin\gamma \tag{4}$$

Rounding all that up, together with an equation for the rate of change of the flight path angle $\dot{\gamma}$, we obtain the reference equations 5 and 6.

$$\frac{dV(t)}{dt} = \dot{V} = \frac{1}{m}(T - D - mg\sin\gamma) \tag{5}$$

$$\frac{d\gamma}{dt}\dot{\gamma} = \frac{1}{mv}(L - mg\cos\gamma) \tag{6}$$

Further simplifications were taken into account to raise expedience for the controller design and facilitate its implementation. Since we want to focus on the aircraft and the on-board system's behaviour and response to faults and errors, we could assume that the initial undisturbed condition would be stable level flight. This means we can consider the rates of altitude, horizontal velocity and flight path angle to be zero. So,

$$\dot{h} = \dot{x} = \dot{\gamma} = 0$$

This meant that some components in equations 5 and 6 could be cut, which leaves us with equations 7, 8 and 9.

$$\dot{h} = V\sin\gamma = 0 \Rightarrow \begin{cases} \cancel{V = 0} \\ \sin\gamma = 0 \Rightarrow \gamma = 0 \vee \cancel{\gamma = \pi} \end{cases} \tag{7}$$

$$\dot{v} = 0 \Rightarrow 0 = \frac{1}{m}(T - D - mg) \Rightarrow T = D + mg \tag{8}$$

$$\dot{\gamma} = 0 \Rightarrow 0 = \frac{1}{mv}(L - mg) \Rightarrow L = mg \tag{9}$$

On the first equation above (7) we can assume the null value for γ as the other values obtained from the equation have no physical meaning in this simulation.

The aircraft aerodynamic forces for lift and drag can be decomposed into their adimensional equivalents, for which data can be obtained from analogous sources without many adjustments. This way, we can obtain the values for these forces based on current flight status.

$$C_D = \frac{D}{\frac{1}{2}\rho V^2 S} \Rightarrow D = \frac{1}{2}\rho V^2 S C_D \tag{10}$$

$$C_L = \frac{L}{\frac{1}{2}\rho V^2 S} \Rightarrow L = \frac{1}{2}\rho V^2 S C_L \tag{11}$$

Applying this logic to the previous equations we can replace the absolute values of forces with adimensional data.

$$T\cos\alpha = D \Rightarrow T_0\cos\alpha_0 = \frac{C_D\rho V_0^2 S}{2}$$

$$T\sin\alpha + L = mg \Rightarrow T_0\sin\alpha_0 + \frac{C_L\rho V_0^2 S}{2} = mg$$

6 The Controller

To implement the controller we had to define the inputs and their relations with the outputs and the aircraft dynamics, which were mentioned and described in section 3.

For this reason, we define the aircraft's states as the vector **x** and the aircraft's inputs as vector **u**, where a bold typeface represents a non-scalar variable, lowercase for vectors and uppercase for matrices.

$$\mathbf{x} = \begin{bmatrix} h & v & \gamma \end{bmatrix}^T \qquad \mathbf{u} = \begin{bmatrix} \alpha & T \end{bmatrix}^T$$

With this notation we can define the aircraft's behaviour as defined below.

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$

Both the matrices referenced here can be defined by the aforementioned variables and inputs.

$$\mathbf{A} = \begin{bmatrix} \frac{\partial \dot{h}}{\partial h} & \frac{\partial \dot{h}}{\partial v} & \frac{\partial \dot{h}}{\partial \gamma} \\ \frac{\partial \dot{v}}{\partial h} & \frac{\partial \dot{v}}{\partial v} & \frac{\partial \dot{v}}{\partial \gamma} \\ \frac{\partial \dot{\gamma}}{\partial h} & \frac{\partial \dot{\gamma}}{\partial v} & \frac{\partial \dot{\gamma}}{\partial \gamma} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} \frac{\partial \dot{h}}{\partial \gamma} & \frac{\partial \dot{h}}{\partial T} \\ \frac{\partial \dot{v}}{\partial \gamma} & \frac{\partial \dot{v}}{\partial T} \\ \frac{\partial \dot{\gamma}}{\partial \gamma} & \frac{\partial \dot{\gamma}}{\partial T} \end{bmatrix}$$

Referring to the equations 7, 8 and 9, we can write down the matrices with the corresponding derivatives applied, simplified to depend only on the control and input variables (making use of the simplifications on equations 7, 8 and 9, and using the coefficients in equations 10 and 11).

$$\mathbf{A} = \begin{bmatrix} 0 & \sin\gamma_0 & V_0 \cos\gamma_0 \\ 0 & -\frac{C_D \rho V_0 S}{m} & -g\cos\gamma_0 \\ 0 & \square & \frac{g \sin\gamma_0}{V_0} \end{bmatrix}, \quad \square = -\frac{T_0 \sin\alpha_0}{mV^2} + \frac{C_L \rho S}{2m} + \frac{g\cos\gamma_0}{V_0^2} \tag{12}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 0 \\ -\frac{T_0 \sin\alpha_0}{m} & \frac{\cos\alpha_0}{m} \\ \frac{T_0 \cos\alpha_0}{mV_0} & \frac{\sin\alpha_0}{mV_0} \end{bmatrix} \tag{13}$$

Finally, we get the state-space equation in short form.

$$\begin{bmatrix} \dot{h} \\ \dot{V} \\ \dot{\gamma} \end{bmatrix} = \mathbf{A} \begin{bmatrix} h \\ V \\ \gamma \end{bmatrix} + \mathbf{B} \begin{bmatrix} \alpha \\ T \end{bmatrix} \tag{14}$$

Constant values from airplane:

$$C_l = 0.476 \qquad C_d = 0.016 \qquad m = 1518.44\,\text{kg} \qquad \rho(\text{Air density}) = 1.20\,\text{kg}\,\text{m}^{-3}$$
$$S(\text{Wing Area}) = 12\,\text{m}^2 \qquad g(\text{Gravity acceleration}) = 9.80\,\text{m}\,\text{s}^{-2}$$
$$T_0(\text{Thrust in equilibrium}) = 500\,\text{N}$$

7 Possible Faults

Within the simple system implemented in this laboratory, several components can fail or behave unexpectedly. Whether it's the control system or the sensors used, either one can have a fault at some point and the control system needs to be ready to return the control to the pilot should a fault be detected somewhere.

This fault detection will be done by monitoring the flight variables and their behaviour throughout the simulation. The first step should then be define what constitutes a fault. By definition,

A fault is a deviation (of a feature) from the acceptable, standard operational condition.

For sensors, we can consider four types of faults, either sensor bias, drift, loss of accuracy or freeze. The sources for these faults come from either the barometric sensor or the

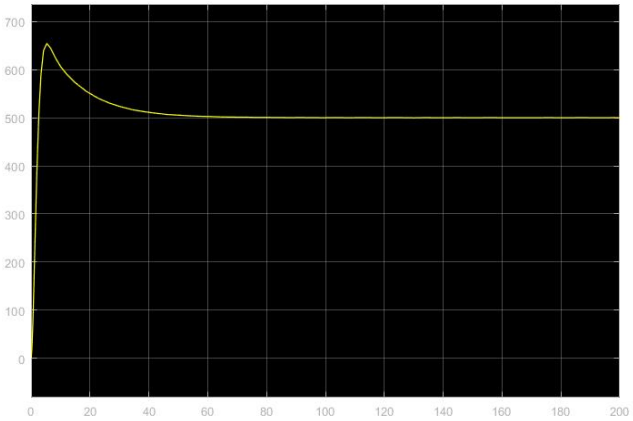

Figure 5: Controller altitude simulation.

temperature A/D conversion. For the bus, we can consider physical loss of the bus and data corruption as our faults.

We then modeled a few failure modes for the controller. While at first glance some seem different from the faults above, the above faults can be considered subsets of these failure modes: signals above a given limit and signals that change too fast ("spikes"). Afterwards, we added these failure mode models to the simulator, ending with an architecture that is designed to detect errors and return the control of the aircraft to the pilot, emulated in this simulation with a simulation stop. We simulated a combination between every type of faults, injected manually, and for each situation in the presence of a fault the simulation was stopped and a control flag set to true corresponding to the failure. This represents the transfer of control to the pilot and simultaneous warning to the pilot of the presence of a fault and its origin. Since this analysis is "binary" in nature, there are no relevant plots to present.

8 Architecture for a fault tolerant controller

For the present lab work, we created a triplex architecture for fault tolerance. This architecture is present in the simulink file triplex.slx.

A Data Plots

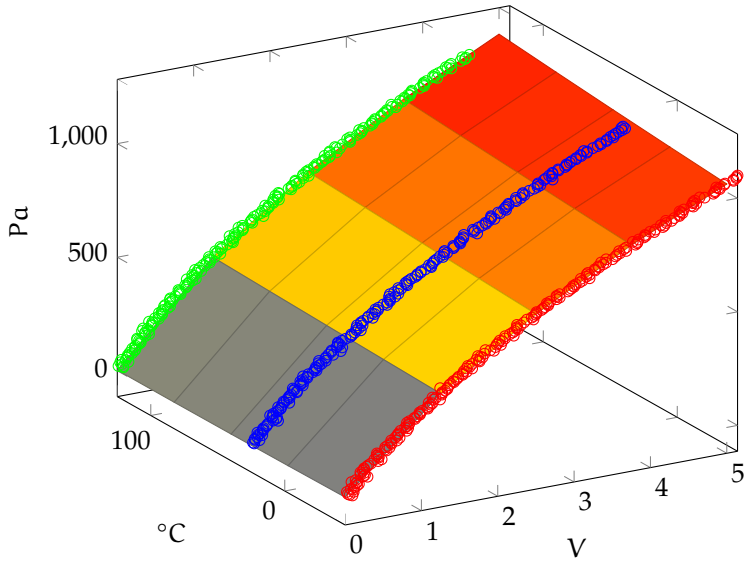

Figure 6: Comparison of the data provided and the modelled polynomial.

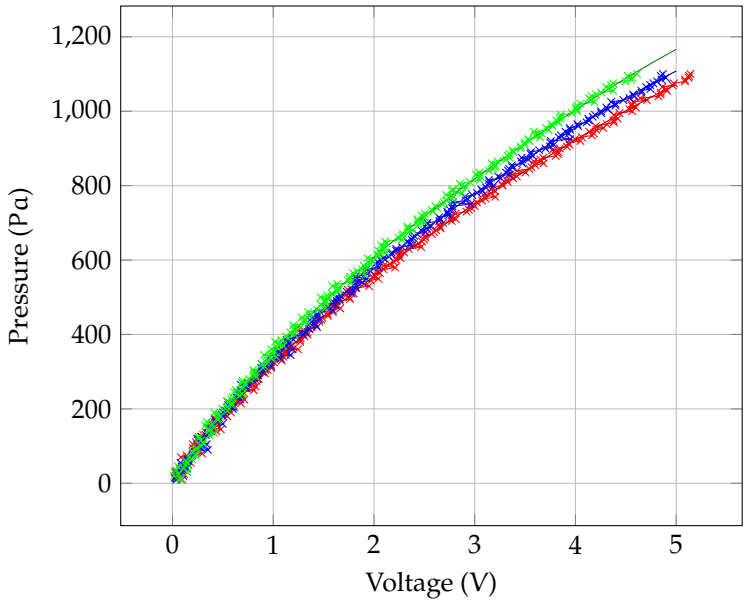

Figure 7: Side-view of the plots versus the modeled version at the same temperature. The modelled plots can be seen behind the data points.