BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

DEVELOPMENT OF AUTOPILOT AND FLIGHT DIRECTOR MODES INSIDE A SIMULINK ENVIRONMENT

NÁVRH AUTOPILOTA A LETOVÝCH ŘÍDÍCÍCH MÓDŮ V PROSTŘEDÍ SIMULINK

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR
AUTOR PRÁCE

Bc. JIŘÍ NOVÁK

SUPERVISOR
VEDOUCÍ PRÁCE

doc. Ing. LUDĚK NECHVÁTAL, Ph.D.

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav automatizace a informatiky
Student:	**Bc. Jiří Novák**
Studijní program:	Strojní inženýrství
Studijní obor:	Aplikovaná informatika a řízení
Vedoucí práce:	**doc. Ing. Luděk Nechvátal, Ph.D.**
Akademický rok:	2019/20

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Návrh autopilota a letových řídících módů v prostředí Simulink

Stručná charakteristika problematiky úkolu:

Pozice a orientace letadla pohybujícího se ve vzduchu je popsána šesti pohybovými veličinami (třemi translačními a třemi rotačními). Ty lze (za určitých zjednodušujících předpokladů a dalších vztahů) získat na základě II. Newtonova pohybového zákona jako řešení soustavy obyčejných diferenciálních rovnic. Základní úlohou v dynamice letu je pak umět popsat pohybovou odezvu na poruchu ve výchozích podmínkách, při kterých bylo letadlo v rovnovážném stavu, a tuto odezvu dále řídit z hlediska lepších letových vlastností. To se obvykle realizuje prostřednictvím zpětně–vazebného řízení.
Diplomová práce bude realizována ve spolupráci s firmou Honeywell International, s.r.o., externím školitelem bude M.S. Ondřej Karas.

Cíle diplomové práce:

Cílem práce je vytvoření simulačního prostředí, které by zahrnovalo charakteristiky známého letadla v nějakém letovém režimu (např. cestovním). Dále je pak úkolem navrhnout několik autopilotů a řídících letových módů a vyhodnotit jejich efektivitu. Eventuálně je možné uvažovat i další letové režimy a funkce.

Konkrétní cíle:

1. Vytvoření simulačního prostředí v MATLAB/Simulink využívající letové charakteristiky (fyzikální vlastnosti draku letadla, stabilitní a řídící derivace) vybraného typu letadla. Výstupem bude vizualizace v softwaru FlightGear.
2. Návrh autopilota, který řídí úhel podélného sklonu a úhel příčného náklonu na základě vhodného PID–regulátoru. Součásti návrhu bude vyladění příslušných koeficientů zesílení.
3. Součástí návrhu bude "flight director Altitude Hold and Heading Select", který by posílal úhly stoupání (théta) a náklonu (fí) do modulu autopilota.
4. Vyhodnocení účinnosti charakteristik autopilota a "flight director" (tlumení, zbytková amplituda a frekvence).

Volitelné cíle:

1. Zahrnutí tlumení úhlu vybočení a naměření jeho charakteristik.
2. Návrh dalších "flight director" módů, např. vertikální rychlost, IAS/FLC, LNAV.
3. Zahrnutí jednoduchého servo modelu a návrh Auto Pitch Trim, Auto Yaw Trim.

Seznam doporučené literatury:

COOK, M. V., Flight Dynamics Principles, 3rd ed., Butterworth-Heinemann, 2012. ISBN 978-0080982427.

ETKIN, B., REID, L. D., Dynamics of Flight, Stability and Control, 3rd ed., Wiley & Sons, 1996. ISBN 978-0-471-03418-6.

ROSKAM, J., Airplane Flight Dynamics & Automatic Flight Controls, DARcorporation, 2018. ISBN 978-1884885174.

STEVENS, B. L., LEWIS, F. L., Aircraft Control and Simulation, 2nd ed., Wiley-Interscience, 2003. ISBN 978-0471371458.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2019/20

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D.	doc. Ing. Jaroslav Katolický, Ph.D.
ředitel ústavu	děkan fakulty

Abstract

This thesis is focused on the development of a simulation environment in Matlab/Simulink for a selected aircraft. The position and orientation of the aircraft moving in the air is described by six-degrees-of-freedom equations of motion. The system of translational, rotational and kinematic equations forms a set of nine nonlinear first order differential equations. These equations can be linearized around an equilibrium point referred to as a steady–state flight condition. The simulation environment contains a developed flight control system based on PID controllers. A basic autopilot is capable of holding pitch attitude and roll angle. Flight director modes including altitude hold, heading select, vertical speed mode, flight level change mode, altitude capture mode and navigational mode based on a nonlinear guidance law are presented. A Pareto optimality based optimization tuning process is developed to optimally tune the regulators. The simulation is graphically represented in FlightGear open source software.

Abstrakt

Tato diplomová práce je zaměřena na vývoj simulačního prostředí v Matlab/Simulink zvoleného letadla ve známém letovém režimu. Pozice a orientace letadla pohybujícího se ve vzduchu je popsána pohybovými rovnicemi se šesti stupni volnosti. Soustava translačních, rotačních a kinematických rovnic tvoří soustavu devíti nelineárních diferenciálních rovnic prvního řádu. Tyto rovnice lze linearizovat okolo nějakého rovnovážného stavu, který budeme nazývat letovým režimem. Součástí simulačního prostředí je řídící systém letadla založený na PID regulaci. Základem je návrh autopilota, který řídí úhel podélného sklonu a úhel příčného náklonu. Součástí návrhu jsou takzvané „flight director" módy jako udržení výšky, volba kursu, regulace vertikální rychlosti, změna výšky, zachycení požadované výšky a navigační mód založený na nelineárním navigačním zákonu. Optimalizace regulátorů za použití PSO algoritmu a Pareto optimalitě je využita pro nastavení parametrů PID regulátoru. Simulační prostředí je vizualizováno v softwaru FlightGear.

Keywords

autopilot, flight director, PID control, equations of motion, particle swarm optimization, Pareto optimality, flight control system, nonlinear guidance law, waypoint navigation, trimmed state, longitudinal motion, lateral-directional motion

Klíčová slova

autopilot, flight director, PID řízení, pohybové rovnice, optimalizace hejnem částic, Pareto optimalita, letový řídicí systém, nelineární zákon navádění, waypoint navigace, ustálený stav, podélný pohyb, příčný pohyb

Novák, J.: *Development of Autopilot and Flight Director Modes inside a Simulink Environment*, Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2020. 100 s. Vedoucí diplomové práce doc. Ing. Luděk Nechvátal, Ph.D.

Rozšířený abstrakt

Prvního leteckého autopilota vyvinul Američan Lawrence Sperry a demonstroval jej v roce 1914. Od té doby prošel vývoj leteckých autopilotů značným vývojem. Nejdříve během druhé světové války, která urychlila vývoj letecké techniky obecně a poté vznikem komerční letecké dopravy, která vyžadovala zejména vysokou bezpečnost a komfort. V současné době lze letecký řídicí systém rozdělit na několik kategorií. Systémy zvyšující stabilitu pod zkratkou (SAS) jako je například „Yaw damper" poskytují vhodné tlumení oscilací letadla a zlepšují tak letové vlastnosti a bezpečnost. Práce je zaměřena na klasické autopiloty, jejichž základem je autopilot, který řídí úhel podelného náklonu a příčného náklonu. Do modulu autopilota vstupuje takzvaný „flight director". Módy tohoto systému navrženy v této práci jsou: udržení výšky, volba kursu, řízení vertikální rychlosti, změna výšky, zachycení výšky a navigace po drahách definovaných takzvanými waypointy. Řízení je realizováno pomocí PID regulátorů, které jsou algoritmicky nastaveny na vhodné zesílení. Metoda nalezení optimálního zesílení je založena na konstrukci takzvané multikriterialní účelové funkce, která je funkcí časových charakteristik odezvy jako je překmit, doba náběhu, doba ustálení a pro účely vývoje autopilota také přetížení, které nesmí překročit stanovené limity. Jelikož každá charakteristika má jiný percentuální vliv na hodnotu účelové funkce, jsou tyto charakteristiky váženy z hlediska relativní důležitosti konstrukcí množiny Paretových optimálních řešení. Výsledná účelová funkce je dále prohledávána v nějaké konečné a ohraničené množině parametrů známým algoritmem optimalizace hejnem částic. Vhodná řešení jsou dále otestována na nelineární simulaci letadla.

Práce je členěna následovně. V první kapitole je zrekapitulováno základní odvození pohybových rovnic se šesti stupni volnosti a používané souřadné systémy. Na konci kapitoly je také ukázáno, jak se tento systém devíti nelineárních diferenciálních rovnic implementuje v softwaru Matlab/Simulink. V druhé kapitole je vysvětleno, jak vypadají aerodynamické síly a momenty působící na letadlo. Poskytnutá aerodynamická data jsou pro letový režim přímočarého letu v konstantní výšce. V následující kapitole je využito principu malých odchylek pro linearizaci pohybových rovnic a odvození stavových reprezentací separovaných podélných a příčných rovnic. Jako ověření správnosti linearizovaných systémů je součástí kapitoly i srovnání odezvy pohybových veličin na skokovou změnu na vstupu výškovky, křidélek a směrového kormidla. Následující kapitola je věnována zejména již nastíněné optimalizaci parametrů PID regulátoru pomocí algoritmu optimalizace hejnem částic. Kromě formálního matematického odvození metody je v závěru kapitoly uveden příklad ladění regulátoru pro řízení nutačního úhlu letadla. Jádrem práce je kapitola 6, která je zaměřena na návrh různých letových módů autopilota. Důraz je kladen zejména na změnu výšky, kde kromě dvou nejčastěji používaných módů vertikální rychlosti a změny výšky změnou tahu je také integrováno zachycení výšky po kruhové trajektorii se zvoleným přetížením. Řídicí systém pak musí ve vhodně zvolenou dobu přepnout mezi módem stoupání a módem zachycení a dále v okolí požadované výšky přepnout na mód udržení výšky. Navigace na waypointy a sledování letové trasy je, zejména díky komplexnosti tématu, diskutována v následujicí kapitole. Ve vědeckých publikacích (např. [10]) je sledování tras popsáno zejména pro bezpilotní letouny (UAV) typu dron, atp., tedy pro létající objekty pohybujicí se velmi malou rychlostí oproti, v této práci, uvažovanému Boeingu 747-200. Tento fakt dává určitá omezení co se křivosti tras týče a stabiltě vůči zvoleným parametrům algoritmu navádění, proto byla úloha zjednodušena na navádění na přímé trasy mezi waypointy, což je také běžná forma tras pro velká dopravní letadla. Kapitola 8 doplňuje navržené autopiloty o logickou složku jejich spínání

a přechodů formou konečných automatů. Knihovna Stateflow je integrována do prostředí Matlab/Simulink a umožňuje tvorbu konečných automatů s podmínkami přechodu mezi jejich stavy. Tento systém má značnou výhodu i při vývoji leteckého softwaru, kde by se logika pomocí Switch bloků v Simulinku stala velmi rychle nepřehlednou a neefektivní. V poslední, deváté kapitole nalezneme krátký popis zvoleného open-source vizualizačního programu FlightGear, který oživí simulaci animací letu.

Práce byla realizována ve spolupráci s firmou Honeywell international s.r.o a odborným školitelem M.S. Ondřejem Karasem. Mezi hlavní přínosy práce patří budoucí použití modelu v dalších diplomových pracích a zvýšení spolupráce mezi firmou Honeywell international s.r.o a Fakultou strojního inženýrství v Brně. Jedním z přínosů může být také metodologie nastavení PID regulátorů popsanou optimalizační metodou pro urychlení volby parametrů PID regulátorů. Je třeba podotknout, že parametry se pro každý regulátor autopilota nastavují pro mnoho letových režimů pokrývajicích letovou obálku, tudíž výběr kandidátů použitou optimalizací může být výhodou. Důležitým bodem práce je také implementace Serret–Frenetových transformací do stavového modelu letadla pro příčný pohyb a lineární analýza navigačního algoritmu. Za zajímavý lze také považovat vyvinutý mód zachycení výšky (ASEL), který se v běžně dostupné literatuře na toto téma téměř nevyskytuje.

I declare that I have written the Master's thesis *Development of Autopilot and Flight Director Modes inside a Simulink Environment* on my own and under the guidance of my supervisor doc. Ing. Luďek Nechvátal, Ph.D. using the sources listed in the references.

Bc. Jiří Novák

I would like to express my thanks to doc. Ing. Luďek Nechvátal, Ph.D. and aerospace engineer and pilot M.S. Ondřej Karas from Honeywell s.r.o for patience, kindness, valuable advice and guidance, and the time spent helping me.

Bc. Jiří Novák

Contents

Nomenclature

m	aircraft mass [kg]
U, V, W	aircraft velocities in body axes [$\mathrm{m} \cdot \mathrm{s}^{-1}$]
P, Q, R	aircraft angular velocities in body axes [$\mathrm{rad} \cdot \mathrm{s}^{-1}$]
X, Y, Z	total forces acting on the aircraft in body axes [N]
$\mathcal{L}, \mathcal{M}, \mathcal{N}$	total moments acting on the aircraft [$\mathrm{N} \cdot \mathrm{m}$]
ϕ, θ, ψ	Euler angles [rad]
D, C, L	Aerodynamic forces in the wind frame [N]
a_x, a_y, a_z	aircraft accelerations in body axes [$\mathrm{m} \cdot \mathrm{s}^{-2}$]
α	angle of attack [rad]
γ	flight path angle [rad]
β	sideslip angle [rad]
h	altitude [m]
$\delta_e, \delta_a, \delta_r$	elevator, aileron and rudder deflection angles [rad]
τ	thrust lever deflection [rad]
i_h	horizontal stabiliser incidence angle [rad]
V_{TAS}	true airspeed [$\mathrm{m} \cdot \mathrm{s}^{-1}$]
C_L, C_D	aerodynamic coefficients [$-$]
ρ	fluid density [$\mathrm{kg} \cdot \mathrm{m}^{-3}$]
$\tilde{q}$	dynamic pressure [$\mathrm{N} \cdot \mathrm{m}^{-2}$]
c	local speed of sound [$\mathrm{m} \cdot \mathrm{s}^{-1}$]
$Mach$	Mach number [$-$]
μ, ι	latitude and longitude [deg]
ω	frequency [s^{-1}]
ζ	damping ratio [$-$]
M_p	overshoot [%]
t_r	rise time [s]
t_s	settling time [s]
E_{ss}	steady state error [%]
K_p, K_i, K_d	gains of PID controller [$-$]
σ	skew disk margin parameter [$-$]
α_{max}	disk margin [$-$]
$D(\alpha_{max}, \sigma)$	set of stable gain and phase perturbations [$-$]
V/S	vertical component of the true airspeed [$\mathrm{m} \cdot \mathrm{s}^{-1}$]
$\mathbf{W}$	waypoint matrix [$-$]
$\mathbf{p}$	aircraft position in flat Earth axes [m]
$\kappa(s)$	curvature of a trajectory parametrized by s [$-$]
$\mathbf{p}_d$	path desired to follow [m]
y_s	cross-track error [m]
L_1	a line defined from a vehicle's position to a virtual target point on a desired trajectory [m]
$-_s$	quantity referred to a steady-state condition
$\{\cdot\}_V$	vectors expressed in a local vertical frame
$\{\cdot\}_B$	vectors expressed in a body axis system
$\{\cdot\}_S$	vectors expressed in a stability axis system
$\{\cdot\}_S$	vectors expressed in a wind axis system

1 Introduction

The first autopilot was demonstrated in 1914 by American pilot Lawrence Sperry at the Airplane Safety Competition (*Concours de la Securité en Aéroplane*) held in France. His plane, equipped with a gyroscopic stabilizer, was able to maintain stability even when his mechanic, Emil Cachin, walked 7 feet out onto the right wing while Sperry held his hands over his head. The development of control theory and the start of the Second World War led to further improvements in aircraft and autopilot design. Nowadays, the development of aircraft control systems presents many more challenges. A control engineer must think in terms of guidance and digital control, simulation tools, numerical methods, sensitivity and robust implementation. It is a good practice to approach complex control problems using model-based design methodology. This approach aims to reduce the costs associated with testing. The idea is to generate an embedded code automatically from a model structured in blocks. The key aspect of developing control systems via the use of simulation techniques is the acquisition of system dynamics to model their behavior. There are generally two ways to do that. Since aircraft motion has been well studied and stability derivatives can be measured to a good degree, we can have a detailed understanding of the inner processes of our system. If we are not, for some reason, capable of deriving a mathematical model of the system, we can resort to system identification techniques, which merely require us to measure the inputs and outputs of the system.

Aircraft control systems are naturally based on feedback control. We can separate them into three categories. *Stability augmentation systems* (SAS), such as a yaw damper, have the role of providing suitable damping and natural frequencies to the rotational modes to keep all aircraft maneuvers safe and stable. *Control augmentation systems* (CAS) have the ability to give pilots a particular type of response to their control inputs. One such system is a normal acceleration CAS, which has the ability to control the acceleration generated along the negative z-axis. The third type of aircraft control system is that of *Autopilots*. They have the role of performing certain tasks so that the pilot is not required to devote constant attention to their manual performance. Good examples of autopilots include "Altitude hold" and "Heading hold". The requirements on all of the above–mentioned systems are very high. They have to provide good performance characteristics within a broad range of flight conditions and stochastic events such as poor weather conditions or bad sensor readings. When simulating such systems, what are known as "operating points" are chosen and the aircraft characteristics are measured. Since many quantities are constantly changing during the flight, such as the air density or Mach number, it is unfeasible in practice to work with just one operating point if one wishes to satisfy requirements over the whole flight envelope. As shown in the author's bachelor's thesis [18], the behavior of a nonlinear first order differential equations–based system of aircraft motion is to a large degree similar to that of linearized equations at a given equilibrium point. This enables the use of a large toolbox of control theory techniques based on LTI (linear time invariant) systems. In practice, many operating points are measured, and through the employment of advanced control concepts, such as gain scheduling, controllers achieve the best performance possible for a given set of hardware. This Master's thesis is mainly focused on autopilot development and the mathematical modeling of aircraft motion using Matlab/Simulink software. In this Master's thesis we will focus on just one operating point, developing a range of aircraft control systems and taking into account possible problems with sensors as well as worst-case scenario events. Flight condition data for a large four jet-engine commercial transport airplane, the Boeing

747-200, have been taken from [1] and used in the model. In the first part we will review six-degrees-of-freedom equations of motion and basic mathematical and engineering concepts that will be needed later. The second chapter is devoted to the description of an aircraft's forces and moments. The distinction will be made between steady–state and perturbed forces and moments. In the third chapter, state-space representations based on linearization theorem will be made for decoupled longitudinal and lateral-directional systems. Certain augmentations will be introduced in order to have a better picture of the states. We will study these linearized MIMO systems by means of classical control theory, such as transfer functions, in order to design autopilots and tune them. Several PID controllers variants will be used. The same controllers will then be tested on a nonlinear model with and without disturbances. The fourth chapter reviews the basic notion of PID controller design and expands it by introducing a Pareto optimality–based multiobjective criterion. The objective function as a function of PID gain parameters will weigh time domain performance criteria such as overshoot or rise time so that one measure of PID performance can be established. The solution space will be searched by a known particle swarm optimization algorithm and a set of potentially optimal solutions will be examined. Finally, an analysis of a closed loop system under plant uncertainty variations will be studied via the concept of disk margins, which are typically used in robust control theory to guarantee the stability of a system under combined gain and phase variations. The core of the thesis, which deals with various flight director modes, is in Chapter Five. Starting with simpler autopilots like pitch attitude hold or roll angle hold, the topic will develop to cover more complicated so called flight director modes like vertical speed mode or altitude capture mode, which control the commands for the autopilot. A separate chapter will be devoted to path tracking and waypoint navigation, which will be mathematically described and algorithmically implemented into the simulation. A proven robust nonlinear guidance law will be employed and linear analysis will show the optimal guidance law radius parameter setting. Simulations navigating an aircraft onto straight line segments with multiple radius parameters and initial heading variations will be tested to prove the concept. Some care will be needed to ensure the aircraft will always be navigated in a forward direction as specified by the ordering of the waypoint matrix. The transitions of the straight line segments will be solved by considering the proximity distance for the switching of a line segment and the natural ability of the nonlinear guidance law to cope with various realizations of the path to be traced. Moving to the end of the thesis, the topic of finite state machine modeling in the Stateflow environment will be introduced and a basic overview of the capabilities of such system when applied to flight control systems will be provided. The last chapter introduces the option of visualizing flight simulations using FlightGear open–source flight simulator software. The appendices complement the thesis with data sheets regarding the chosen aircraft and stability derivative definitions. A full derivation of small perturbation equations of motion is also attached for the sake of completeness.

Although this thesis is structured logically from the basics up, the reader is advised to consult the literature in order to obtain a fuller picture of the topic. The main sources of knowledge concerning the dynamics and control of aircraft are [1], [2], [3] and [5].

2 Equations of motion

The mathematical theory of the motion of an aircraft in flight, which is considered as a rigid body with six degrees of freedom, was introduced in its present form by Professor George Hartley Bryan in 1911. His equations are still used today in the analysis and simulation of even most advanced of today's aircraft. Here, we will review the equations of motion of a rigid constant–mass aircraft considering a flat-Earth simplification. From the beginning, no assumptions regarding inertias, body shape or aerodynamics and propulsion models will be considered. This general model is based on Newtonian mechanics and forms a fundamental understanding of the physics behind aircraft motion. It is also closely tied to reference frames, Earth axes, body axis, and wind and stability axes. These will be explained shortly, but one should always pay attention to which reference frame is used in a particular situation. Detailed derivations of 6-DoF equations of motion can be found in, e.g. [1], [2] and [18].

2.1 Reference frames

In aircraft dynamics, it is important to distinguish between the orientation of acting forces and a relative position to Earth. Some variables are naturally expressed in one reference frame (for example thrust force F_T is usually expressed in the body reference frame) and have to be transformed, if used in another coordinate system. The Earth–fixed axes form a coordinate system connected to the surface of the Earth. We will often be using the Earth axes coordinate system in the form of a local vertical frame whose origin is translated to the aircraft centre of gravity. Vectors expressed in this reference frame will be enclosed in curly braces and marked as $\{.\}_V$. Similarly, the body axis system with its origin at the aircraft's centre of gravity will be denoted by $\{.\}_B$. The aircraft manufacturer's aerodynamic data is usually presented using the stability axis system.

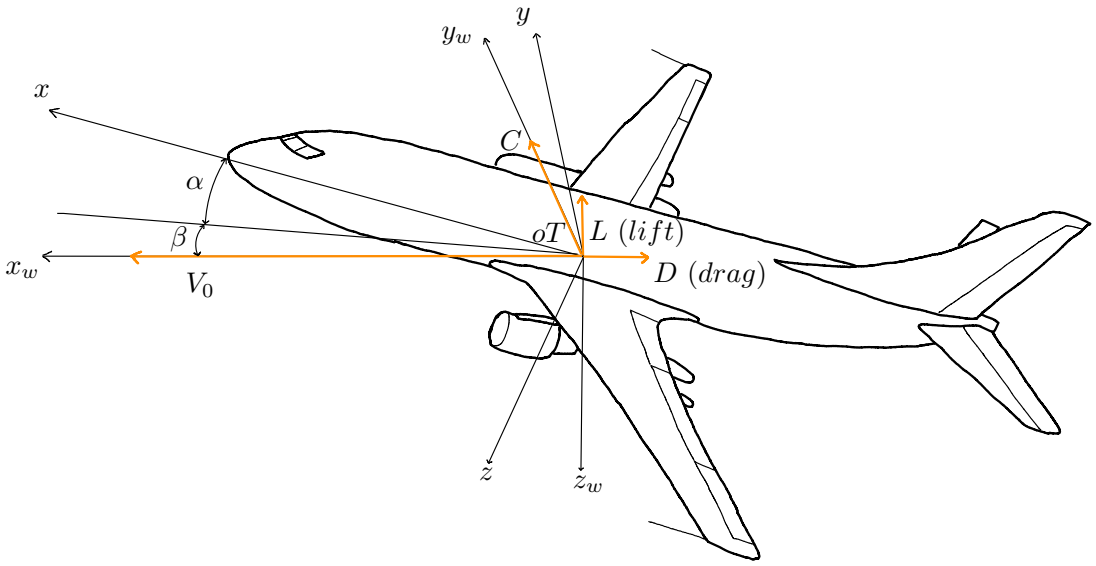

Figure 1: Aircraft reference frames

The x-stability axis is the projection of the velocity vector of the aircraft on the plane of symmetry. It will be denoted as $\{.\}_S$. The angle between the x-stability axis and the x-body axis is called the angle of attack (α). The angle between the velocity vector and the x-stability axis is defined as the side slip angle (β). Rotating the stability axis system by the side slip angle will give us a wind axis system $\{.\}_W$. The velocity vector in the body axis system $\{\mathbf{V_0}\}_B = (U, V, W)$ therefore collapses into x-wind axis $\{\mathbf{V_0}\}_W = (V_{TAS}, 0, 0)$. From a geometric perspective, $\alpha = \arctan\frac{W}{U}$, $\beta = \arctan\frac{V}{U}$ and $V_{TAS} = \sqrt{U^2 + V^2 + W^2}$ (also called true airspeed). We can see the above explained reference frames in Figure 1. In order to describe all the necessary transformations from the local vertical frame, let us introduce a heading angle ψ, an elevation angle θ and a bank angle ϕ. These are called Euler angles and they describe a relative rotation between two systems via consecutive rotations of a coordinate system. Since the rotation matrices have the property that their inverse equals its transpose, we can write $[C_{B\leftarrow V}] = [C_{V\leftarrow B}]^T$ and $[C_{B\leftarrow W}] = [C_{W\leftarrow B}]^T$. The transformation matrix from the body frame to the local vertical frame can be written as

$$[C_{V\leftarrow B}] = \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\cos\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}.$$

Similarly, the transformation matrix from body frame to wind frame in terms of α and β can be expressed as

$$[C_{W\leftarrow B}] = \begin{bmatrix} \cos\alpha\cos\beta & \sin\beta & \sin\alpha\cos\beta \\ -\cos\alpha\sin\beta & \cos\beta & -\sin\alpha\sin\beta \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix}. \tag{2.1}$$

Note that if $\beta = 0$ in (2.1), we obtain transformation matrix $[C_{S\leftarrow B}]$. Lastly, when working with the stability axis system, we have to recompute moment inertias I_{xx}, I_{zz}, I_{xz}, which are usually expressed in the body coordinate frame. The transformation is given by

$$\begin{Bmatrix} I_{xx} \\ I_{zz} \\ I_{xz} \end{Bmatrix}_S = \begin{bmatrix} \cos^2\alpha & \sin^2\alpha & -\sin 2\alpha \\ \sin^2\alpha & \cos^2\alpha & \sin 2\alpha \\ \frac{1}{2}\sin 2\alpha & -\frac{1}{2}\sin 2\alpha & \cos 2\alpha \end{bmatrix} \begin{Bmatrix} I_{xx} \\ I_{zz} \\ I_{xz} \end{Bmatrix}_B.$$

Taking into account the reference frames, six-degrees-of-freedom equations of motion will be reviewed in the next subsection.

2.2 Translational equations

From Newton's second law of motion, a general form of translational acceleration in an inertial reference frame can be written as

$$m\frac{\mathrm{d}\mathbf{V}}{\mathrm{dt}} = \sum \mathbf{F}, \tag{2.2}$$

where m is the aircraft's mass and $\mathbf{V}$ is the aircraft's speed vector. The resulting forces acting on the aircraft can be decomposed into gravity, aerodynamic and thrust forces.

$$\sum \mathbf{F} = \mathbf{F}_G + \mathbf{F}_A + \mathbf{F}_T.$$

It is natural to define gravity force $\mathbf{F}_G$ in a local vertical frame, aerodynamic force $\mathbf{F}_A$ in the wind or stability frame and thrust force $\mathbf{F}_T$ in the body frame.

$$\{\mathbf{F}_G\}_V = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}, \quad \{\mathbf{F}_A\}_W = \begin{bmatrix} -D \\ C \\ -L \end{bmatrix}, \quad \{\mathbf{F}_T\}_B = \begin{bmatrix} X_T \\ Y_T \\ Z_T \end{bmatrix}.$$

System (2.2) can be expanded further via expression in the body axis-frame. By introducing rotation velocity vector $\mathbf{\Omega}$, we can express the velocity derivative in the body axis:

$$m(\{\dot{\mathbf{V}}\}_B + \{\mathbf{\Omega}\}_B \times \{\mathbf{V}\}_B) = [C_{B\leftarrow V}]\{\mathbf{F}_G\}_V + [C_{B\leftarrow W}]\{\mathbf{F}_A\}_W + \{\mathbf{F}_T\}_B \tag{2.3}$$

All the forces must be also transformed into a desired reference frame. The standard notation for elements of velocity and the angular velocity vector is

$$\{\mathbf{V}\}_B = \begin{bmatrix} U \\ V \\ W \end{bmatrix}, \quad \{\mathbf{\Omega}\}_B = \begin{bmatrix} P \\ Q \\ R \end{bmatrix}.$$

The scalar form of system (2.3) can be obtained via the rearrangement and expression of gravitational force. The following system is a general form of differential equations of translational motion. In order to solve them, the acting forces will have to be specified.

$$\begin{aligned} \dot{U} &= \frac{1}{m}(X_A + X_T) - WQ + VR - g\sin\theta, \\ \dot{V} &= \frac{1}{m}(Y_A + Y_T) - UR + WP + g\cos\theta\sin\phi, \\ \dot{W} &= \frac{1}{m}(Z_A + Z_T) - VP + UQ + g\cos\theta\cos\phi. \end{aligned} \tag{2.4}$$

2.3 Rotational equations

In a similar manner, the equation of rotational acceleration can be presented in its vector form as

$$\frac{\mathrm{d}\mathbf{H}}{\mathrm{dt}} = \sum \mathbf{M}. \tag{2.5}$$

Here, $\mathbf{H}$ represents the total angular momentum about the centre of gravity. For a body reference frame, equality (2.5) can be expanded to

$$\{\dot{\mathbf{H}}\}_B + \{\mathbf{\Omega}\}_B \times \{\mathbf{H}\}_B = \sum \{\mathbf{M}\}_B.$$

The angular momentum can be described in a body axis system by the inertia tensor $[\mathbf{I}]_B$ and angular velocity $\{\mathbf{\Omega}\}_B$ as

$$\{\mathbf{H}\}_B = [\mathbf{I}]_B\{\mathbf{\Omega}\}_B + \{\mathbf{H}^*\}_B.$$

where $\{\mathbf{H}^*\}_B$ represents the deformation component that involves rotational elements such as propellers. This will not be relevant in this text, thus $\{\mathbf{H}^*\}_B = \mathbf{0}$. Due to the xz plane symmetry (which is the case with most aircrafts), the inertia tensor can be simplified to

$$[\mathbf{I}]_B = \begin{bmatrix} I_{xx} & 0 & -I_{xz} \\ 0 & I_{yy} & 0 \\ -I_{xz} & 0 & I_{zz} \end{bmatrix}.$$

The moment equation now takes the form

$$[\mathbf{I}]_B\{\dot{\mathbf{\Omega}}\}_B + \{\mathbf{\Omega}\}_B \times ([\mathbf{I}]_B\{\mathbf{\Omega}\}_B) = \sum\{\mathbf{M}\}_B.$$

Let the total momentum $\sum\{\mathbf{M}\}_B$ have the components $\mathcal{L}, \mathcal{M}, \mathcal{N}$ and let it be possible to decompose it into aerodynamic and thrust components, i.e.

$$\sum\{\mathbf{M}\}_B = \begin{bmatrix} \mathcal{L}_A + \mathcal{L}_T \\ \mathcal{M}_A + \mathcal{M}_T \\ \mathcal{N}_A + \mathcal{N}_T \end{bmatrix}.$$

After a few algebraic rearrangements and the computation of the inertia tensor inverse, the scalar form of (2.5) in the body axis frame is

$$\begin{aligned}
\dot{P} =& \frac{1}{I_{xx}I_{zz} - I_{xz}^2}\left[I_{zz}(\mathcal{L} - QR(I_{zz} - I_{yy}) + I_{xz}PQ) + I_{xz}(\mathcal{N} - PQ(I_{yy} - I_{xx}) - I_{xz}QR)\right], \\
\dot{Q} =& \frac{1}{I_{yy}}\left[\mathcal{M} - RP(I_{xx} - I_{zz}) - I_{xz}(P^2 - R^2)\right], \\
\dot{R} =& \frac{1}{I_{xx}I_{zz} - I_{xz}^2}\left[I_{xz}(\mathcal{L} - RQ(I_{zz} - I_{yy}) + I_{xz}PQ) + I_{xx}(\mathcal{N} - PQ(I_{yy} - I_{xx}) - I_{xz}QR)\right].
\end{aligned} \tag{2.6}$$

2.4 Kinematic equations

The force and moment equations do not form a complete system. The connection between a moving reference frame and a fixed reference frame is missing. Specifically, no relationship has been established between Euler angles θ, ψ, ϕ and the angular velocities P, Q, R. A general equality holds for the sum of angular velocities and Euler angle rates in rotating coordinate systems

$$\mathbf{i}P + \mathbf{j}Q + \mathbf{k}R = \dot{\boldsymbol{\psi}} + \dot{\boldsymbol{\theta}} + \dot{\boldsymbol{\phi}}.$$

where $\mathbf{i}, \mathbf{j}, \mathbf{k}$ are standard orthonormal basis vectors in the body axis. The well-known system of gimbal equations solves this issue and completes the set of translational and rotational equations. Their derivation can be found in, e.g. [18].

$$\begin{aligned}
\dot{\phi} &= P + Q\sin\phi\,\mathrm{tg}\,\theta + R\cos\phi\,\mathrm{tg}\,\theta, \\
\dot{\theta} &= Q\cos\phi - R\sin\phi, \\
\dot{\psi} &= Q\frac{\sin\phi}{\cos\theta} + R\frac{\cos\phi}{\cos\theta}.
\end{aligned} \tag{2.7}$$

The system can be also supplemented by the so-called navigational equations which project the velocity in the body axis frame onto the fixed earth axis frame. This can be accomplished simply by $\{\mathbf{V}\}_E \equiv \{\mathbf{V}\}_V = [C_{V\leftarrow B}]\{\mathbf{V}\}_B$. In order to provide a clear picture, the scalar form is again presented:

$$\begin{aligned}
\dot{x}_E &= \cos\theta\cos\psi U + (\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi)V + (\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi)W, \\
\dot{y}_E &= \cos\theta\sin\psi U + (\sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi)V + (\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi)W, \\
\dot{z}_E &= -\sin\theta U + \sin\phi\cos\theta V + \cos\phi\cos\theta W.
\end{aligned}$$

It should be pointed out that a singularity exists for $\theta = \pm\frac{\pi}{2}$. This can be solved by a quaternion formulation, but it is not crucial for simple aircraft simulations.

The set of translational (2.4), rotational (2.6) and gimbal kinematic equations (2.7) forms a set of nine nonlinear first order differential equations. The set of navigational equations can be added to compute the aircraft's position and orientation in the earth reference frame. The system is now in the form $\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x})$. Later, it will be shown that explicit dependence on t vanishes i.e. the system is in fact autonomous. Since $\mathbf{x}$ is a vector function of time and the system is autonomous, we can call the system a dynamical system.

2.5 Matlab/Simulink implementation

Simulink has advanced tools for aircraft motion simulations. Aerospace Blockset is a standard tool in the aerospace industry and has predefined blocks for the equations of motion. On the left side of the 6DOF equations of the motion block in Figure 2, inputs for total forces and moments can be found. The outputs are all quantities needed to describe motions of the aircraft, such as body axis speeds, flat-Earth position, Euler angles, angular velocities and accelerations. Initial conditions and variables can be set in the block settings. For all simulations, a fixed-step automatic solver selection was chosen with a step size of 2 ms. All the data needed for the simulation were stored in a separate Matlab file, data_letadlo.m. We will be using metric units throughout the text (with few exceptions) and in simulations despite the fact that aviation industry has been heavily using imperial units (especially for altitude, distance and speeds). The model will be built upon this

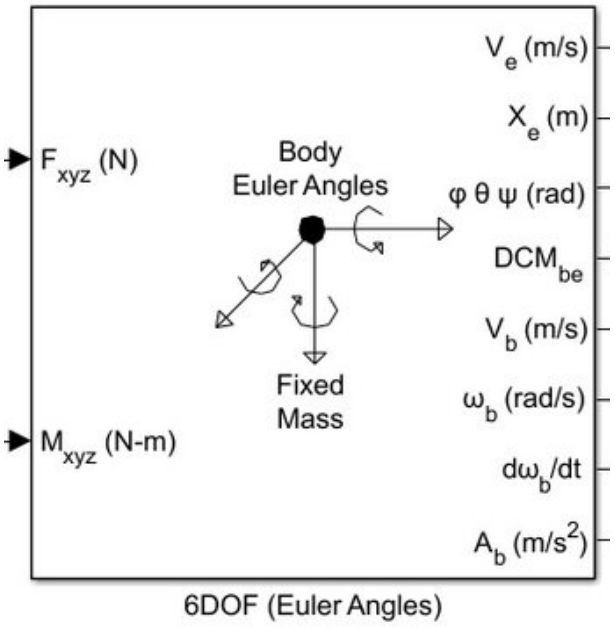

Figure 2: Equations of motion block

block. There are many variations of this block available (quaternion formulation, wind angles, variable mass). The forces and moments will have to be specified by the data and the decomposition of gravitational force based on aircraft orientation. The block assumes that the applied forces are acting at the center of gravity of the body, and that the mass and inertia are constant. Our simulation will be also handling only rigid–body dynamics. If one needs to perform more in-depth analysis and simulation, aeroservoelastic model of the aircraft accounting for flexible modes of the wings has to be computed.

3 Aerodynamic and thrust forces and moments

The purpose of this chapter is to explain the modeling of aerodynamic and thrust forces and moments. In practice, they can be determined in two ways: via experimental methods such as flight tests or wind tunnel tests and via computational methods. In mathematical models the main focus will is on stability and control derivatives. These are usually valid for a certain range around a steady–state point (forces and moments equilibrium), since they are dependent on other variables such as the dynamic pressure or Mach number. The dynamic pressure is the increase in pressure of a moving fluid over its static value due to the motion. It is a quantity which is helpful in the study of stress variations within the structure of an aircraft. Dynamic pressure also relates forces and moments acting on an aircraft to their respective coefficients. It is indicated as $\bar{q}$ and defined as

$$\bar{q} = \frac{\rho V_{TAS}^2}{2}$$

where ρ is the fluid density obtained from the 1976 Committee on Extension to the Standard Atmosphere (COESA) model and V_{TAS} is the true airspeed. The Mach number is a dimensionless quantity representing the ratio of flow velocity past a boundary to the local speed of sound, i.e.

$$Mach = \frac{V_{TAS}}{c}.$$

The local speed of sound c is again obtained from the COESA Model.

The set of data for a certain trim point is called flight condition data. Many flight conditions must be measured in order to cover the whole flight envelope and develop robust controllers. The flight condition data and aircraft data used in this thesis have been taken from [1]. They are also listed in Appendix A. Because it is assumed that a small degree of coupling exists between the longitudinal and lateral-directional variables, the model will be presented as two independent sets of forces and moments (see [18]).

3.1 Longitudinal steady-state forces and moments

Drag force D, lift force L and pitching moment $\mathcal{M}_A$ are considered to be aerodynamic longitudinal variables. Similarly, X_T,Z_T and $\mathcal{M}_T$ are thrust longitudinal variables. They are considered to act in the xz stability plane. The aircraft drag can be computed as

$$D = C_D \bar{q} S,$$

where S is the wing area. If the drag coefficient C_D is further approximated by the first order Taylor polynomial, we obtain

$$D = (C_{D_0} + C_{D_\alpha}\alpha + C_{D_{i_h}} i_h + C_{D_{\delta_e}} \delta_e)\bar{q} S.$$

We can see the dependence of C_D on the angle of attack α, incidence horizontal tail deflection i_h and elevator deflection δ_e. C_{D_0} is the value of drag at a zero angle of attack, incidence angle deflection and elevator deflection. Note that the numerical values of C_{D_0} and C_{D_α} depend on the steady state itself. The simulation should therefore run mostly within the range of linearity of the coefficient.

Coefficients $C_{D_{i_h}}$ and $C_{D_{\delta_e}}$ then represent changes in aircraft drag due to the change in stabilizer incidence angle and elevator deflection angle, respectively. The same process

holds for lift coefficient C_L, therefore

$$L = (C_{L_0} + C_{L_\alpha}\alpha + C_{L_{i_h}} i_h + C_{L_{\delta_e}}\delta_e)\bar{q}S,$$

pitch moment coefficient $C_{\mathcal{M}}$ has to be additionally multiplied by the mean geometric chord $\bar{c}$

$$\mathcal{M}_A = (C_{\mathcal{M}_0} + C_{\mathcal{M}_\alpha}\alpha + C_{\mathcal{M}_{i_h}} i_h + C_{\mathcal{M}_{\delta_e}}\delta_e)\bar{q}S\bar{c}.$$

The coefficients are to be evaluated at constant Mach and Reynolds numbers. The modeling of longitudinal thrust force and moments is more complex. For now, assuming we are in the body axis system, we can write

$$\begin{aligned} X_T &= T\cos(\phi_T), \\ Z_T &= -T\sin(\phi_T), \\ \mathcal{M}_T &= -Td_T, \end{aligned}$$

where ϕ_T and d_T characterize the offset of the thrust line to the x-body axis. Thrust force T can be set by throttle position coefficient τ.

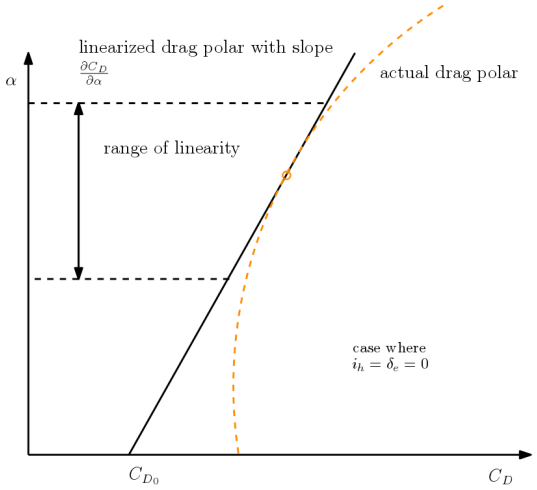

Figure 3: Interpretation of linear drag polar

3.2 Lateral-directional steady-state forces and moments

In general, the steady–state condition with $V \neq 0$ is called side-slipping. A aerodynamic forces and moments acting on the aircraft are in this case the side force C, rolling moment $\mathcal{L}_A$ and yawing moment $\mathcal{N}_A$. In the same manner as before, expanding coefficients into the first order Taylor polynomial allows us to obtain

$$\begin{aligned} C &= (C_{Y_0} + C_{Y_\beta}\beta + C_{Y_{\delta_a}}\delta_a + C_{Y_{\delta_r}}\delta_r)\bar{q}S, \\ \mathcal{L}_A &= (C_{\mathcal{L}_0} + C_{\mathcal{L}_\beta}\beta + C_{\mathcal{L}_{\delta_a}}\delta_a + C_{\mathcal{L}_{\delta_r}}\delta_r)\bar{q}Sb, \\ \mathcal{N}_A &= (C_{\mathcal{N}_0} + C_{\mathcal{N}_\beta}\beta + C_{\mathcal{N}_{\delta_a}}\delta_a + C_{\mathcal{N}_{\delta_r}}\delta_r)\bar{q}Sb. \end{aligned}$$

For the lateral-directional thrust forces and moments, we have to introduce a thrust line offset angle ψ_T. The reason for this angle is to minimize the nacelle drag of engines in their local flow field. Calculating all engines separately, we obtain

$$Y_T = \sum_i T_i \psi_{T_i},$$

$$\mathcal{L}_T = \sum_i T_i(z_{T_i}\psi_{T_i} - y_{T_i}\phi_{T_i}) - T_i y_{T_i}\alpha,$$
$$\mathcal{N}_T = \sum_i T_i(x_{T_i}\psi_{T_i} - y_{T_i}).$$

3.3 Perturbation forces and moments

To accomplish a full simulation, the effects of small changes in variables on forces and moments have to be taken into account. A detailed explanation of individual variable effects is presented in [1]. The adopted linear model of small perturbation forces and moments which was used in the simulation is shown immediately below. The variables for the perturbed forces and moments denoted by lower case letters x, z, m, y, l, n should be added to the steady–state forces and moments. For the longitudinal forces and moments, we have

$$\begin{bmatrix} \frac{x}{\bar{q}S} \\ \frac{z}{\bar{q}S} \\ \frac{m}{\bar{q}S\bar{c}} \end{bmatrix} = \begin{bmatrix} -(C_{D_u} + 2C_{L_1}) & (-C_{D_\alpha} + C_{L_1}) & -C_{D_{\dot{\alpha}}} & -C_{D_q} & -C_{D_{\delta_e}} \\ -(C_{L_u} + 2C_{L_1}) & (-C_{L_\alpha} - C_{D_1}) & -C_{L_{\dot{\alpha}}} & -C_{L_q} & -C_{L_{\delta_e}} \\ (C_{m_u} + 2C_{m_1}) & C_{m_\alpha} & C_{m_{\dot{\alpha}}} & C_{m_q} & C_{m_{\delta_e}} \end{bmatrix} \begin{bmatrix} \frac{u}{V_{TAS}} \\ \alpha \\ \frac{\dot{\alpha}\bar{c}}{2V_{TAS}} \\ \frac{q\bar{c}}{2V_{TAS}} \\ \delta_e \end{bmatrix}.$$

A similar matrix form can be derived for the lateral directional forces and moments. The effect of two control surfaces are considered. The result of the matrix multiplication yields non–dimensional forces and moments which have to be multiplied by a factor to obtain a force.

$$\begin{bmatrix} \frac{y}{\bar{q}S} \\ \frac{l}{\bar{q}Sb} \\ \frac{n}{\bar{q}Sb} \end{bmatrix} = \begin{bmatrix} C_{y_\beta} & C_{y_{\dot{\beta}}} & C_{y_p} & C_{y_r} & C_{y_{\delta_a}} & C_{y_{\delta_r}} \\ C_{l_\beta} & C_{l_{\dot{\beta}}} & C_{l_p} & C_{l_r} & C_{l_{\delta_a}} & C_{l_{\delta_r}} \\ C_{n_\beta} & C_{n_{\dot{\beta}}} & C_{n_p} & C_{n_r} & C_{n_{\delta_a}} & C_{n_{\delta_r}} \end{bmatrix} \begin{bmatrix} \beta \\ \frac{\dot{\beta}b}{2V_{TAS}} \\ \frac{pb}{2V_{TAS}} \\ \frac{rb}{2V_{TAS}} \\ \delta_a \\ \delta_r \end{bmatrix}.$$

3.4 Matlab/Simulink implementation

A working Simulink implementation of aircraft motion was accomplished. Steady–state coefficients C_{L_1}, C_{D_1} and $C_{T_{x_1}}$ produce a steady–state level flight condition. Coefficient $C_{T_{x_1}}$ was discarded and steady state thrust force T was found directly using four turbofan engine systems from Aerospace Blockset by finding steady–state throttle position τ. Control over thrust will be used later. Throttle position $\tau = 0.404$ exerts a similar thrust to the discarded coefficient even though the engine model does not provide a static thrust. It is dependent on the Mach number and altitude. The block parameters of the turbofan engine system in Simulink are set as follows. Initial thrust is set (as stated earlier) to 0.404. The nominal net thrust for a jet engine usually refers to the Sea Level Static (SLS) condition for the international standard atmosphere (ISA). The net thrust naturally decreases with altitude due to the fall in air density. The maximum sea-level static thrust was set to 360 kN. The fastest engine time constant at sea-level static equilibrium condition was set to 9 seconds. In real world the engines reach 63.2 % of maximum thrust after this time. The ratio of installed to uninstalled thrust was set to 0.9. It specifies a decrease in the thrust

of the engine while installed in the aircraft due to the shape of the cowling and other effects. Uninstalled thrust is measured on a test stand under carefully controlled conditions. The pilot's control inputs are highlighted by a background color and set to zero (see Figure 5). The gravity, aerodynamic and thrust forces and moments are assembled in separate subsystems. For the purpose of clarity, the stability and control derivatives are separated into longitudinal and lateral-directional ones. Simulink allows multiple signals to be linked to just one through the mux blocks. They are widely used in the model. All the unused signal ports are closed by terminator blocks. Dynamic pressure $\bar{q}$, true airspeed V_{TAS}, local speed of sound c and air density ρ are updated during the simulation with help of the COESA Atmosphere Model block. The steady–state values are therefore only used as initial values. FlightGear animation is also included in the simulation. It requires a set of Euler angles, altitude, latitude and longitude. The equations of motion block provides us with the aircraft's position. The Flat Earth to LLA block converts a 3-by-1 vector of flat Earth position $[x_E \;\; y_E \;\; z_E]$ into geodetic latitude μ, longitude ι and altitude h. The flat Earth coordinate system assumes the z-axis is downward positive. First, the x_E and y_E coordinates are transformed into North and East coordinates. By defining the angle Ψ between the x-axis and North, the transformation will take the form

$$\begin{bmatrix} N_E \\ E_E \end{bmatrix} = \begin{bmatrix} \cos\Psi & -\sin\Psi \\ \sin\Psi & \cos\Psi \end{bmatrix} \begin{bmatrix} x_E \\ y_E \end{bmatrix}.$$

To convert the North and East coordinates to geodetic latitude and longitude, the radius of curvature in the prime vertical R_N and the radius of curvature in the meridian R_M have to be specified. This can be achieved via the equations

$$\begin{aligned} R_N =& \frac{R}{\sqrt{1-(2f-f^2)\sin^2\mu_0}}, \\ R_M =& R_N \frac{1-(2f-f^2)}{\sqrt{1-(2f-f^2)\sin^2\mu_0}}, \end{aligned}$$

where R represents the equatorial radius of the planet and f is the flattening of the planet. Small changes in the latitude and longitude are approximated from small changes in the North and East positions by

$$\begin{aligned} \mathrm{d}\mu =& \arctan(R_M^{-1})\mathrm{dN_E}, \\ \mathrm{d}\iota =& \arctan((R_N\cos\mu)^{-1})\mathrm{dE_E}. \end{aligned}$$

Values of μ_0 and ι_0 represent the initial latitude and longitude respectively. The final latitude and longitude are the sum of the initial value and the small changes. The altitude is the negative flat Earth z-axis value minus the reference height h_{ref}.

$$\begin{aligned} \mu =& \mu_0 + \mathrm{d}\mu, \\ \iota =& \iota_0 + \mathrm{d}\iota, \\ h =& - z_E - h_{ref}. \end{aligned}$$

Figure 5 shows the connection of all the described blocks. The model has been subsequently encapsulated into a **Boeing 747-200** subsystem, where it is clear what the

inputs and outputs of the system are. In reality, the outputs are all the measured variables. For example, an inertial measurement unit (IMU) provides measurements of a body's specific force angular rates and the orientation of the body using a combination of accelerometers, gyroscopes and magnetometers. When handling the IMU data, one should be ready to handle noise and errors induced in the model. Sensor errors can take several forms. Among the common errors encountered in practice are offset errors, scale factor errors, misalignment errors (for example due to the imperfect mounting of the sensor), noise and environmental sensitivity induced by thermal gradients. Variables which are hard to measure directly, for example sideslip angle β therefore have to be estimated using a variety of methods, such as observers. Figure 4 shows the encapsulation of the model.

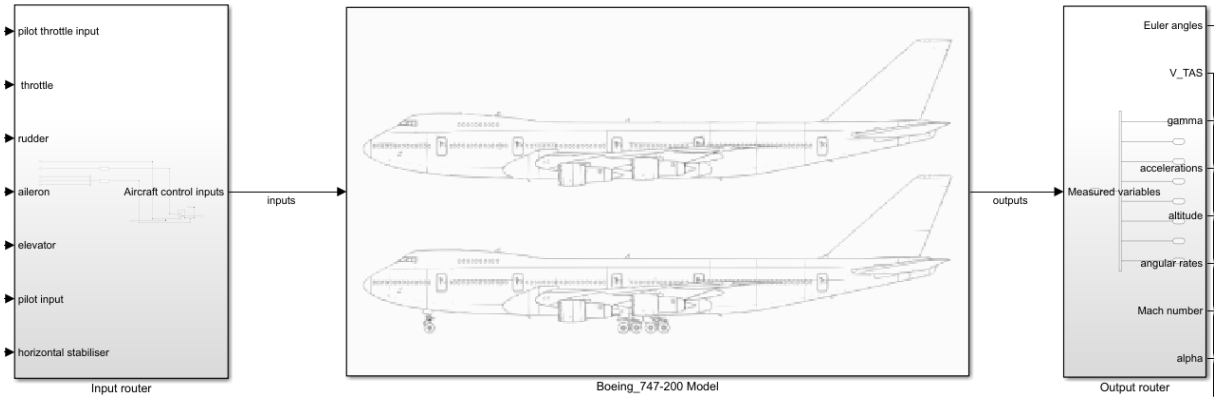

Figure 4: Simulink flight simulation

Among the inputs, we can see that pilot inputs are separated from autopilot inputs. A logic diagram developed later should make sure that pilot inputs do not interfere with the autopilot. All the pilot inputs are set to zero while the autopilot is in control. The maximum range of control surface deflections are stated in Table 1. The control surfaces which are part of the model are highlighted in green and the rest in red. They are provided just for reference. The inboard ailerons are mainly used at higher speeds while the outboard

Table 1: Boeing 747-200 control surface limits

Control surface	**Max. deflection**
Elevators	$-23, +17$ degrees
Horizontal stabilizer	$-12, +3$ degrees
Inboard ailerons	$-20, +20$ degrees
Rudder	$-25, +25$ degrees
Spoilers 1,2,3,4,9,10,11,12	0.25 degrees
Spoilers 5,8	0.20 degrees
Spoilers 6,7 (speedbreakers)	0.20 degrees
Outboard ailerons	$-25, +15$ degrees

ailerons are locked. At low speeds, both pairs of ailerons can be used. The spoilers are the components on the top surface of the wing intended to create a controlled stall by reducing lift. They are often used to increase descent rate without increasing speed, which is very convenient when approaching an airport. They are also used during the landing roll, to increase normal pressure on the wheels and improve braking.

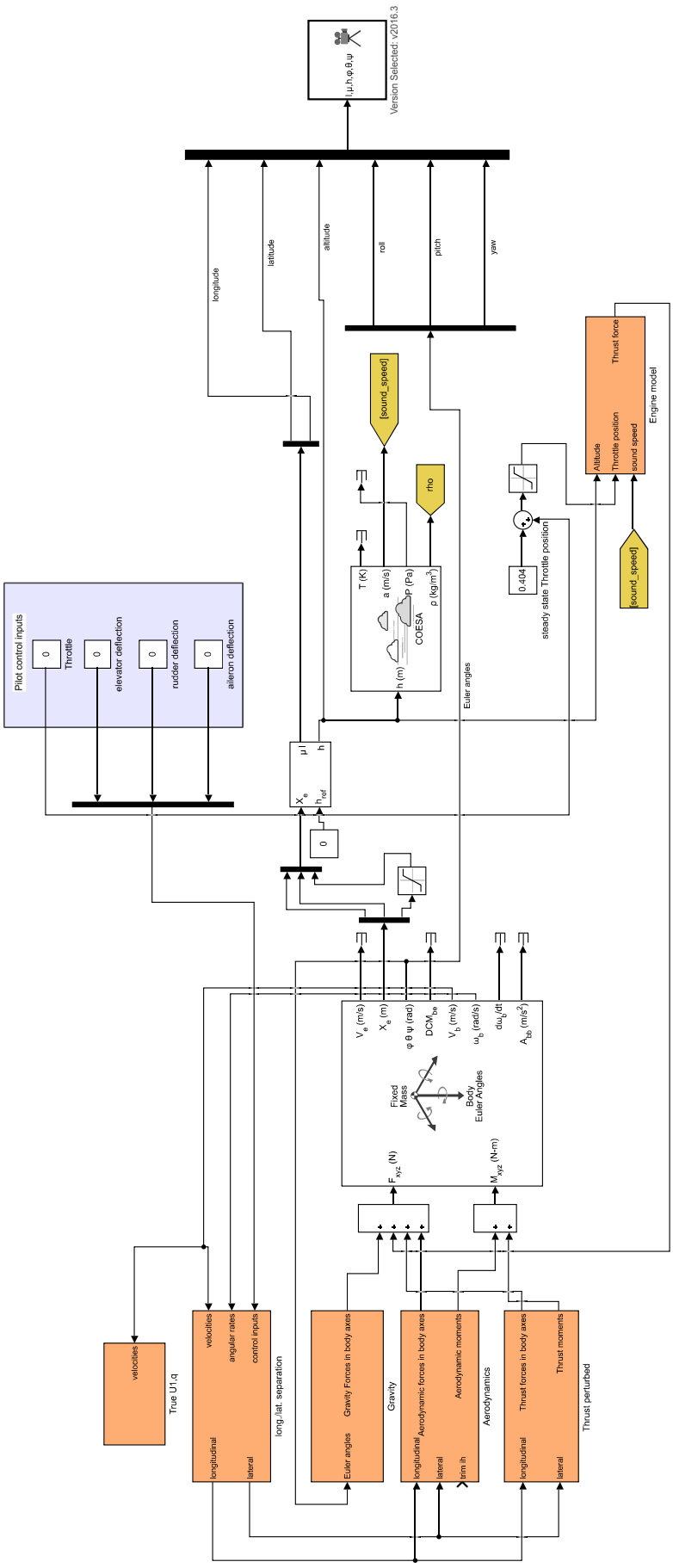

Figure 5: Simulink flight simulation

4 State-space representation of small perturbation equations

In the last chapters, the complete nonlinear dynamics of an aircarft were developed and implemented in Matlab/Simulink. To progress further, we will develop state-space representations of equations of motion based on linearization theory. We will adjust these models to suit our needs and compare them to the nonlinear model to be sure they are close enough. This will form the backbone for further controller development. We can write our system in the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \tag{4.1}$$

where $\mathbf{u}$ is a vector of control inputs. A point $(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})$ is called a steady state (or point of equilibrium) if

$$\mathbf{f}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) = \mathbf{0}.$$

In practice, we usually work with a steady–state straight and level flight, but other types of equilibria also exist. If we define perturbations from the steady state to be $\mathbf{z} = \mathbf{x} - \tilde{\mathbf{x}}$ and $\mathbf{z_u} = \mathbf{u} - \tilde{\mathbf{u}}$, we can define a linear system

$$\dot{\mathbf{z}} = \left(\frac{\partial f_i(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})}{\partial x_i} \right)_{i,j=1}^{n} \mathbf{z} + \left(\frac{\partial f_i(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})}{\partial u_k} \right)_{i,k=1}^{n} \mathbf{z_u}.$$

Hereinafter, the notation for the steady–state vector elements will be

$$\begin{aligned} \tilde{\mathbf{x}} =& [U_s, V_s, W_s, P_s, Q_s, R_s, \theta_s, \phi_s, \psi_s]^T, \\ \tilde{\mathbf{u}} =& [(\delta_e)_s, (\delta_a)_s, (\delta_r)_s, \tau_s]^T. \end{aligned}$$

If we work in the stability axis reference frame, we should keep in mind that $W_e = 0$. In some works from the literature, steady–state velocity $(V_{TAS})_s$ is often denoted U_1. This is because of the chosen reference frame and the steady–state straight flight condition. Perturbed variables will be denoted by lower-case letters in the following way:

$$\begin{aligned} \mathbf{z} =& [u, v, w, p, q, r, \theta, \phi, \psi]^T, \\ \mathbf{z_u} =& [\delta_e, \delta_a, \delta_r, \tau]^T. \end{aligned}$$

Perturbed forces and moments introduced in the previous chapter can be used unchanged in (4.1). However, in order to obtain a better insight into the physical characteristics of the relative importance of the aerodynamic forces and moments, the so-called dimensional stability derivatives are introduced. For example, let us consider the term $C_{\mathcal{M}_\alpha}$. That term will be transformed to

$$\frac{\bar{q}_e S \bar{c} C_{\mathcal{M}_\alpha} \alpha}{I_{yy}} = \mathcal{M}_\alpha \alpha.$$

The newly defined dimensional stability derivative $\mathcal{M}_\alpha$ represents the pitch angular acceleration imparted to the aircraft as a result of a unit change in the angle of attack. The definitions of all stability derivatives can be found in Appendix B.

Since the equations are now fully decoupled, we can represent them via two separate state-space systems. The longitudinal state-space system will be denoted as

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \tag{4.2}$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \tag{4.3}$$

with the initial conditions $\mathbf{x}(0) = \mathbf{x_0}$. $\mathbf{A}$ is the $n \times n$ state matrix, $\mathbf{B}$ is the $n \times m$ input matrix, $\mathbf{C}$ is the $r \times n$ output matrix, $\mathbf{D}$ is the $r \times n$ direct matrix and inequality $r \leq n$ must hold. In practice, we usually want $\mathbf{D}$ to be the zero matrix and $\mathbf{C}$ to be the identity matrix. However, if we cannot measure some of the states, we include that fact in $\mathbf{C}$ and use some estimation methods such as Kalman filters. The longitudinal state vector will be denoted $\mathbf{x} = [u, \alpha, q, \theta]^T$. The same procedure will be applied to lateral-directional equations using the notation

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mathbf{K}\mathbf{x}(t) + \mathbf{L}\mathbf{u}(t),\\ \mathbf{y}(t) &= \mathbf{M}\mathbf{x}(t) + \mathbf{N}\mathbf{u}(t)\end{aligned}$$

with state vector $\mathbf{x} = [v, p, r, \phi, \psi]^T$. It is often advantageous to keep track of angle of attack α and side-slip angle β. They can either be included in the output vector without changing the states, or they can replace the normal velocity w and the side velocity v so that $w = (V_{TAS})_s\alpha$ and $v = (V_{TAS})_s\beta$. For steady–state straight flight it holds that $(V_{TAS})_s = U_s$. Let us now represent the longitudinal equations in matrix form:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & ((V_{TAS})_s - Z_\alpha) & 0 & 0 \\ 0 & -\mathcal{M}_{\dot{\alpha}} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{\alpha} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} =$$

$$\begin{bmatrix} (X_u + X_{T_u}) & X_\alpha & 0 & -g\cos\theta_s \\ Z_u & Z_\alpha & (Z_q + (V_{TAS})_s) & -g\sin\theta_s \\ (\mathcal{M}_u + \mathcal{M}_{T_u}) & (\mathcal{M}_\alpha + \mathcal{M}_{T_\alpha}) & \mathcal{M}_q & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ \alpha \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} X_{\delta_e} \\ Z_{\delta_e} \\ \mathcal{M}_{\delta_e} \\ 0 \end{bmatrix} \delta_e.$$

Via the inverse multiplication of the leftmost matrix we can obtain the state–space form. It will also be necessary to obtain response characteristics for the variables, that are not included in the equations of motion. Provided that the variable of interest can be expressed as a function of the basic aircraft motion variables, it may be obtained by augmenting the state description. Specifically, height rate $\dot{h}(t)$ can be generally computed as

$$\dot{h}(t) = U\theta - V\phi - W. \tag{4.4}$$

Note that $V_s = 0$ for steady–state symmetric flight and since $(V_{TAS})_s = U_s$, (4.4) can be simplified to

$$\dot{h}(t) = (V_{TAS})_s\theta - w = (V_{TAS})_s\theta - (V_{TAS})_s\alpha.$$

The longitudinal state-space model can be then extended simply by adding the corresponding state to the model.

$$\left[\begin{array}{c} \dot{\mathbf{x}}(t) \\ \hdashline \dot{h}(t) \end{array}\right] = \left[\begin{array}{ccccc} & & \mathbf{A} & & \mathbf{0} \\ \hdashline 0 & -(V_{TAS})_s & 0 & (V_{TAS})_s & 0 \end{array}\right] \left[\begin{array}{c} \mathbf{x}(t) \\ \hdashline h(t) \end{array}\right] + \left[\begin{array}{c} \mathbf{B} \\ \hdashline 0 \end{array}\right] \delta_e.$$

It is worth mentioning that thrust input τ has been omitted from the model. This is due to unknown stability derivatives, which are not present in the data. The regulators including thrust changes will have to be implemented directly in the model.

In the same manner we can derive the lateral directional state-space equations:

$$\begin{bmatrix} (V_{TAS})_s & 0 & 0 & 0 & 0 \\ 0 & 1 & -\frac{I_{xz}}{I_{xx}} & 0 & 0 \\ 0 & -\frac{I_{xz}}{I_{zz}} & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\beta} \\ \dot{p} \\ \dot{r} \\ \dot{\phi} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} Y_\beta & Y_p & (Y_r - (V_{TAS})_s) & g\cos\theta_s & 0 \\ L_\beta & L_p & L_r & 0 & 0 \\ (N_\beta + N_{T_\beta}) & N_p & N_r & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$
$$\begin{bmatrix} \beta \\ p \\ r \\ \phi \\ \psi \end{bmatrix} + \begin{bmatrix} Y_{\delta_a} & Y_{\delta_r} \\ L_{\delta_a} & L_{\delta_r} \\ N_{\delta_a} & N_{\delta_r} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix}.$$

Both aileron and rudder control inputs are included in the state–space form. The derivation of both state space forms and the assumptions made can be found in Appendix C. To demonstrate the validity of the state-space forms in practice, the step responses of the Simulink nonlinear model and the state–space forms were compared.

4.1 Stability modes and model analysis

To gain a better insight into longitudinal and lateral–directional state–space equations, a Laplace transform will be used to acquire the transfer function matrix and perform an analysis of the system. This process will be shown in the longitudinal state–space form. Since **A**,**B**,**C**,**D** are matrices of constant coefficients, the Laplace transform of system (4.2), assuming zero initial conditions, is

$$\begin{aligned} s\mathbf{x}(s) &= \mathbf{A}\mathbf{x}(s) + \mathbf{B}\mathbf{u}(s), \\ \mathbf{y}(s) &= \mathbf{C}\mathbf{x}(s) + \mathbf{D}\mathbf{u}(s). \end{aligned}$$

The state equation can be rearranged to

$$\mathbf{x}(s) = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\mathbf{u}(s), \tag{4.5}$$

where **I** is an identity matrix of the same dimension as **A**. By combining the output equation and (4.5), output vector $\mathbf{y}(s)$ is given by

$$\mathbf{y}(s) = [\mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}]\mathbf{u}(s) = \mathbf{G}(s)\mathbf{u}(s),$$

where $\mathbf{G}(s)$ is called a transfer function matrix. In general, the transfer function matrix takes the form

$$\mathbf{G}(s) = \frac{1}{\Delta(s)}\mathbf{N}(s),$$

where $\mathbf{N}(s)$ is a polynomial matrix whose elements are all the transfer function numerators. The term $\Delta(s)$ is a characteristic polynomial common to all transfer functions. The roots of this polynomial are of key importance for stability and control analysis. For aircraft motion applications, the longitudinal and lateral-directional transfer function

matrices may be written as

$$\mathbf{G}(s) = \frac{1}{\Delta(s)} \begin{bmatrix} N^u_{\delta_e} \\ N^\alpha_{\delta_e} \\ N^q_{\delta_e} \\ N^\theta_{\delta_e} \\ N^h_{\delta_e} \end{bmatrix}, \qquad \mathbf{H}(s) = \frac{1}{\Delta(s)} \begin{bmatrix} N^\beta_{\delta_a} & N^\beta_{\delta_r} \\ N^p_{\delta_a} & N^p_{\delta_r} \\ N^r_{\delta_a} & N^r_{\delta_r} \\ N^\phi_{\delta_a} & N^\phi_{\delta_r} \\ N^\psi_{\delta_a} & N^\psi_{\delta_r} \end{bmatrix}.$$

The characteristic polynomial of longitudinal transfer function matrix $\mathbf{G}(s)$ commonly factorizes in two pairs of complex roots which describe longitudinal stability modes that are called *phugoid* and *short-period*. The characteristic equation for $\Delta(s)$ can be written as

$$(s^2 + 2\zeta_p\omega_p s + \omega_p^2)(s^2 + 2\zeta_s\omega_s s + \omega_s^2) = 0.$$

The longitudinal dynamics of an aircraft can be linked to a pair of loosely coupled mass-spring damper systems and the interpretation of motion following a disturbance from a steady–state can be performed via a comparison to a mechanical mass–spring damper. Damping ratios ζ_p, ζ_s and natural frequencies ω_p, ω_s are the only parameters.

The lateral–directional characteristic polynomial commonly factorizes in two real roots and a pair of complex roots. Thus it is also a fourth order polynomial describing a non-oscillatory *spiral mode*, a non-oscillatory *roll subsidence mode* and an oscillatory *Dutch roll mode*. The characteristic equation takes the form

$$(1 + (1/T_s))(1 + (1/T_r))(s^2 + 2\zeta_d\omega_d s + \omega_d^2) = 0.$$

Unlike the case of longitudinal dynamics, the interpretation of lateral-directional dynamics is not as straightforward as the stability modes are not so distinct. Usually there is a much greater degree of mode coupling and interaction.

4.1.1 Longitudinal stability modes

Both the longitudinal stability modes are excited, whenever the aircraft is disturbed from a steady state by either a pilot control input or external atmospheric disturbances such as gusts and turbulence. The short period mode is a highly damped oscillation in pitch about the y-axis. Most of the damping is caused by the motion of the tail. It manifests itself as a classical second-order oscillation with the principal variables being angle of attack α, pitch rate q and pitch attitude θ. The undamped natural frequency falls in a range from 1 rad/s to 10 rad/s and the damping is usually lower than the desired level, although stabilizing. The speed response is negligible at this time scale and is approximately constant. The damping effects are dominated by the aerodynamics of the tailplane, which has

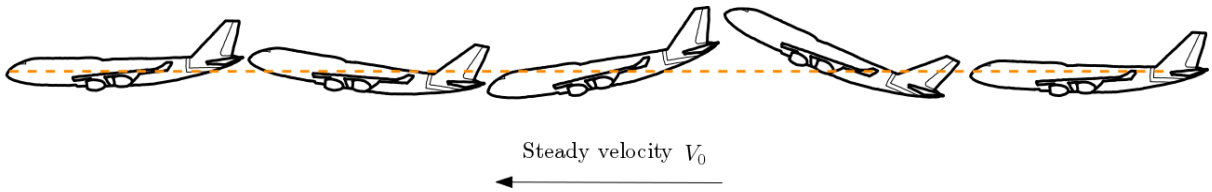

Figure 6: Short period oscillation

a tendency to align itself with the incident flow. There is a negligible change in altitude

and airspeed by the time the mode subsides.

The phugoid mode is in contrast a lightly damped oscillation in speed u, which couples with pitch attitude θ and height h. It has also a low frequency usually in the range from $0.1 rad/s$ to $1 rad/s$. A disturbance resulting in a small decrease in speed has the effect of a change in lift. Since the aircraft is assumed to be in equilibrium, the loss of lift results in a change in height. As the aircraft accelerates downwards, it gains more speed and lift until it steadily pitches up and starts to climb. This process continues in a loop until the motion is damped out by drag effects. The phugoid mode may be thought of as an exchange of potential and kinetic energy. The damping characteristics of the mode may be influenced by power effects.

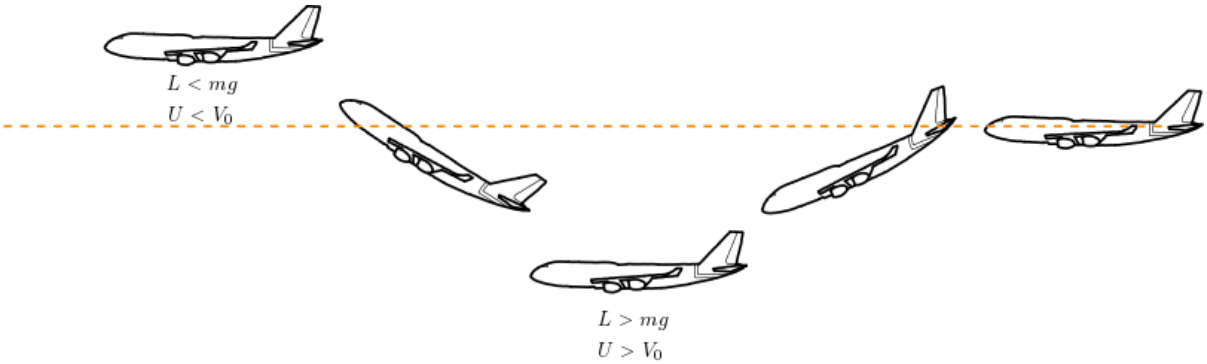

Figure 7: Phugoid period oscillation

4.1.2 Lateral-directional stability modes

Following a disturbance by the ailerons, rudder or external effects, lateral-directional stability modes are excited. The roll subsidence mode described by a single real root with time constant T_r manifests itself as an exponential lag characteristic in rolling motion. After the aircraft experiences a disturbing rolling moment, angular acceleration starts the rolling motion. The change in the incidence of the right and left wing builds up the restoring moment until the aircraft is flying in steady–state flight. From pilot's perspective, the roll mode appears as a lag in response to control inputs.

The spiral mode is represented by the second real root in the lateral-directional

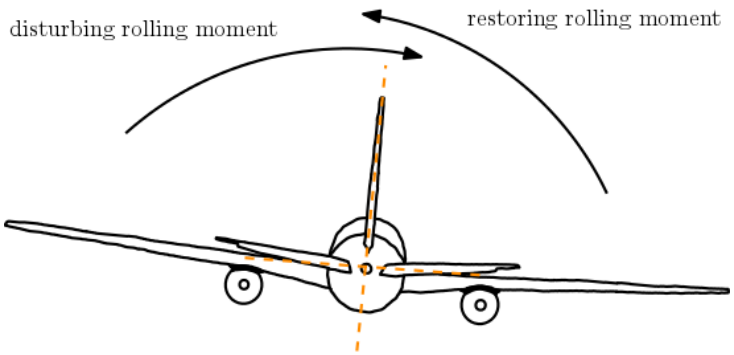

Figure 8: Roll subsidence mode

characteristic equation with time constant T_s. It is slow to develop and involves a coupled motion in roll, yaw and sideslip. The disturbance in sideslip causes an aircraft to roll. Roll

angle ϕ starts to increase slowly. That has the effect of a change in sideslip velocity v, which puts the fin at incidence β. At the same time, the dihedral effect of the wing creates a restoring rolling moment due to the sideslip. The requirements for lateral and directional stability are often made to make the opposing effects nearly equal. When the effect of the fin is greater than that of the dihedral, the spiral mode becomes unstable. Because of its large time constant exceeding 100 seconds, an unstable spiral mode is not that dangerous if spotted by the pilot in time. Still, an the unstable flight path diverges into a spiral descent. The Dutch roll mode is a damped oscillation in yaw around the z-axis, which couples into

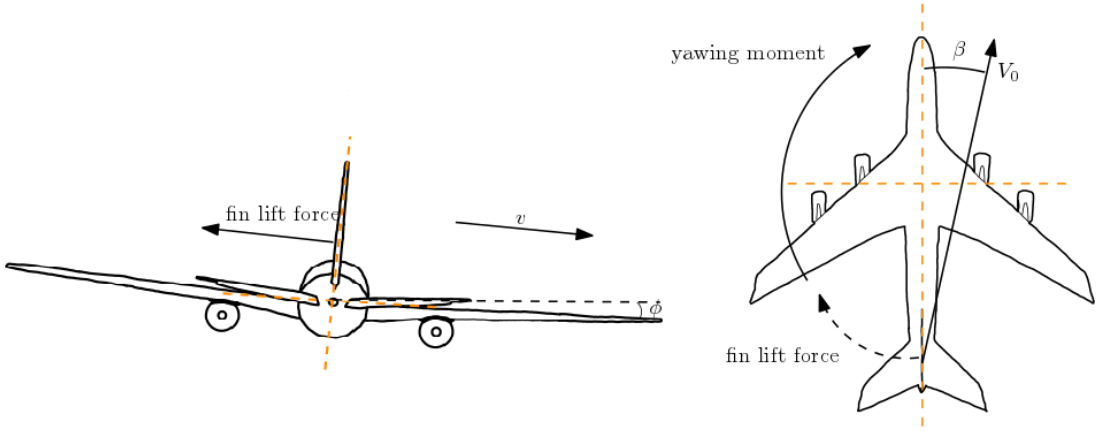

Figure 9: Spiral mode development

roll and sideslip. Since the moment of inertia in pitch and yaw are of a similar order, the frequencies of the Dutch roll mode and the short period mode are also alike. Both the stiffness and damping in yaw are determined by the aerodynamic properties of the fin, and a larger fin is often desirable for a stable Dutch roll mode. An aircraft disturbed from the steady state in yaw starts to oscillate in the xy-plane due to aerodynamic effects. The difference in yaw gives rise to lift and drag perturbations. This coupling results in oscillatory motion in roll, which lags oscillation in yaw by approximately 90 degrees. The motion is easily understood by tracing the path of a wing tip in one Dutch roll cycle. The oscillatory cycle repeats decaying to zero with positive damping. Damping in yaw is often not satisfactory and a yaw damper should therefore be installed.

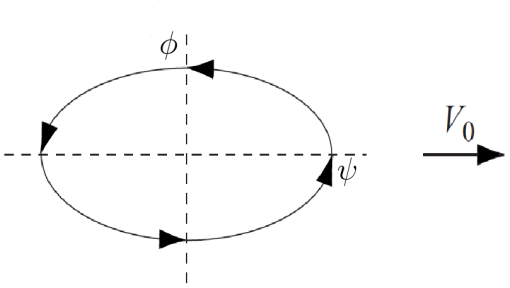

Figure 10: Dutch roll cycle

4.2 Matlab/Simulink verification

This section provides a stability analysis of the developed state-space model as well as a comparison of the step responses. Let us examine the roots of the longitudinal characteristic polynomial. After the creation of the state-space model, the command *roots*() on the characteristic polynomial returns a set of roots. Note that an additional zero pole was found. This increase in the order of the state equation represents the height integration. The zero denominator should cancel with the zero numerator in all numerator polynomials except that describing the height response. Two pairs of complex roots represent a stable short-period and phugoid in that order. The damping ratio of the short period is $\zeta_s = 0.4984$ and the natural frequency is $\omega_s = 1.1789$ rad/s. The phugoid has a much lower damping ratio of $\zeta_p = 0.0205$ and its natural frequency $\omega_p = 0.0684$ rad/s. The

Longitudinal state space

```
s = tf('s');
states = {'u' 'alpha' 'q' 'theta' 'h'};
inputs = {'elevator'};
outputs = {'u' 'alpha' 'q' 'theta' 'h'};
system_lon = ss(Alon,Blon,Clon,Dlon,'statename',...
    states,'inputname',inputs,'outputname',outputs);
roots(charpoly(Alon))
```

```
ans = 5x1 complex
   0.0000 + 0.0000i
  -0.5876 + 1.1022i
  -0.5876 - 1.1022i
  -0.0014 + 0.0684i
  -0.0014 - 0.0684i
```

roots of the lateral-directional characteristic polynomial are also of the fifth order with the addition of a zero root, which indicates neutral stability. This is due to the reference to the body axis system by the yaw angle to the state equation. This can be interpreted as meaning that the dynamics of the aircraft is independent of the yaw angle or heading. The oscillatory Dutch roll mode has a damping ratio of $\zeta_d = 0.1198$ and a natural frequency of $\omega_d = 1.0556$ rad/s. The time constants of the roll subsidence mode and spiral mode are $T_r = 1.0547$ s and $T_s = 58.4795$ s, respectively. Although all of the stability modes are stable, the root of the spiral mode occurs very close to the imaginary axis with a low margin of stability. After acquiring the characteristics of the stability modes, the Simulink nonlinear model should be tested to ascertain, if it fits the state-space model. For that reason, a comparison of the step responses has been made for all control surfaces. The step magnitude has been modified to $\frac{\pi}{180}$ rad to reflect only a small one degree change in control surface angle. The results for elevator response variables over 50 s of measurement can be found in Figure 11. Note that the short period approximation fits very well, but the phugoid shows a high degree of error. This can be caused by the differences between the linear and nonlinear model or by the modeling of engine effects in the simulation. Despite the inconsistency, the state-space model can be used to develop regulation loops,

Lateral-directional state space

```
s = tf('s');
states = {'beta' 'p' 'r' 'phi' 'psi'};
inputs = {'aileron' 'rudder'};
outputs = {'beta' 'p' 'r' 'phi' 'psi'};
system = ss(Alat,Blat,Clat,Dlat,'statename',...
    states,'inputname',inputs,'outputname',outputs);
roots(charpoly(Alat))
```

```
ans = 5x1 complex
   0.0000 + 0.0000i
  -0.1265 + 1.0480i
  -0.1265 - 1.0480i
  -0.9481 + 0.0000i
  -0.0171 + 0.0000i
```

since phugoid is slow to react and regulator efficiency is more dependent on immediate response, which happens to be short period. The comparison of lateral-directional responses showed the higher accuracy of the state-space model. For both aileron and rudder step responses, the error between nonlinear and linear systems was not significant enough to affect the future use of linearized state-space representations for autopilot development. The aileron and rudder responses can be found in Figures 12 and 13. Dark red responses account for linear model and blue responses respresent a nonlinear model. The code to generate the longitudinal and lateral-directional state–space representations has been provided. To achieve a clear arrangement, we can name the inputs and outputs of the system. Matlab provides powerful tools to design and analyze control systems in the *Control System Toolbox*.

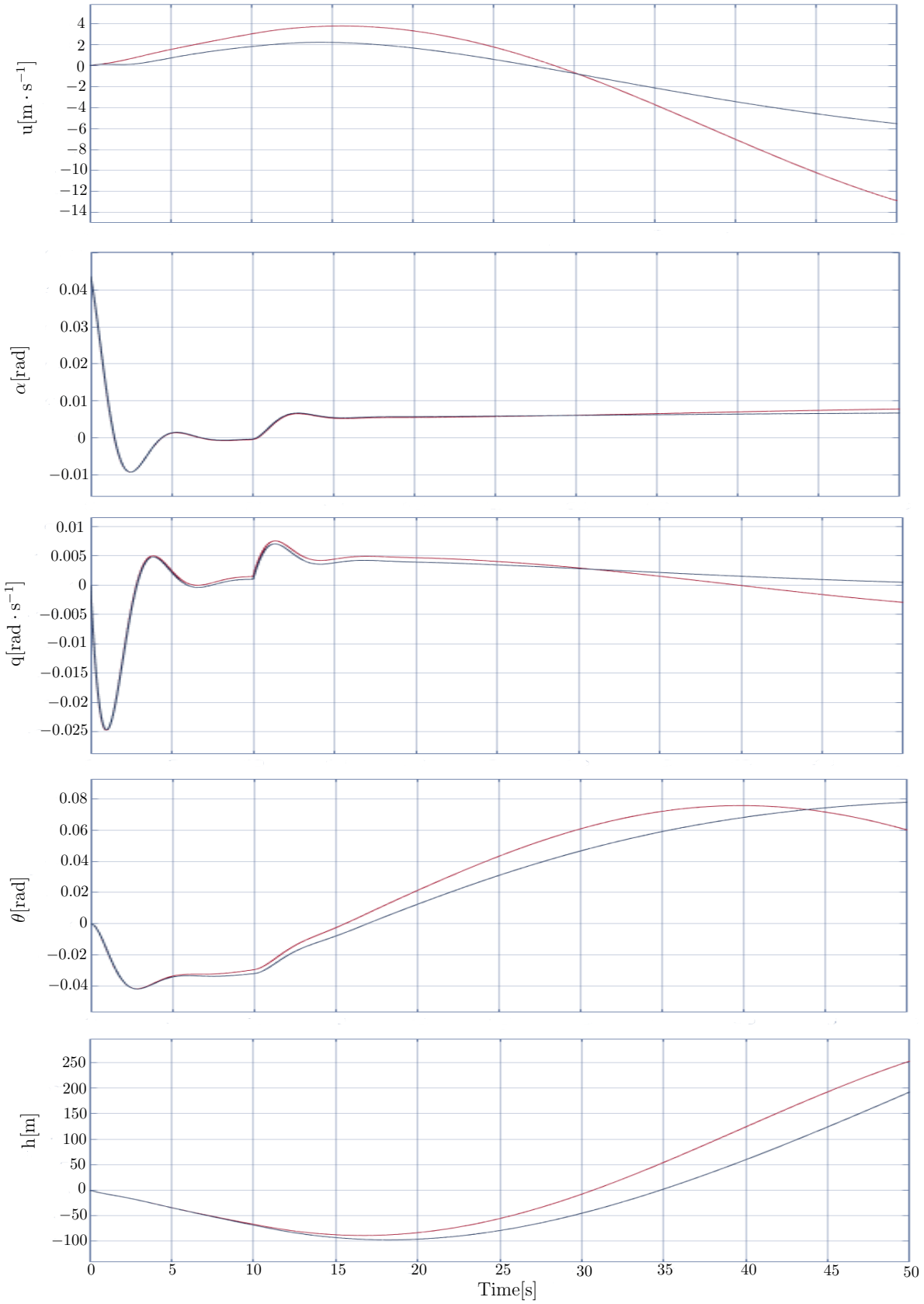

Figure 11: Elevator step response comparison

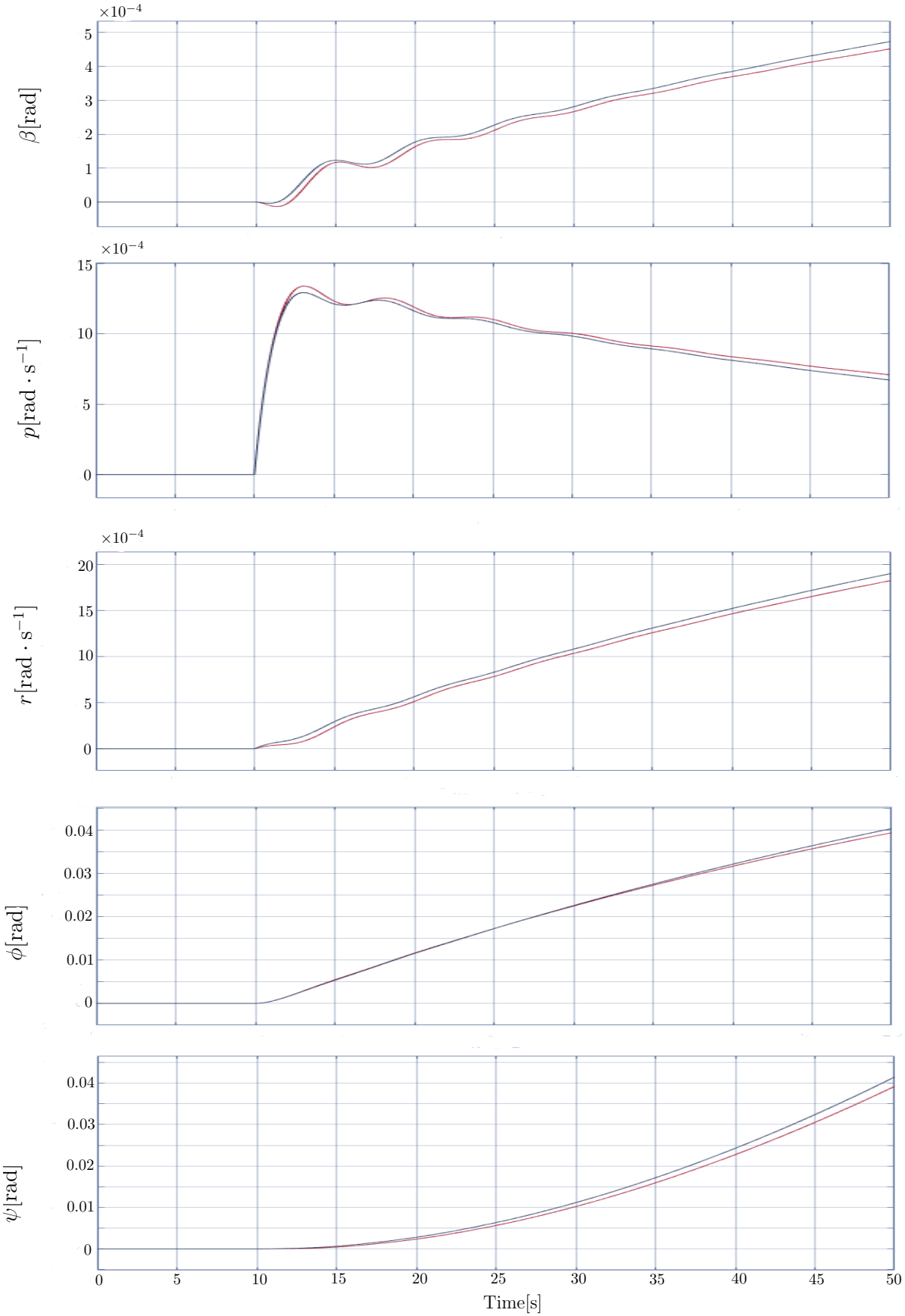

Figure 12: Aileron step response comparison

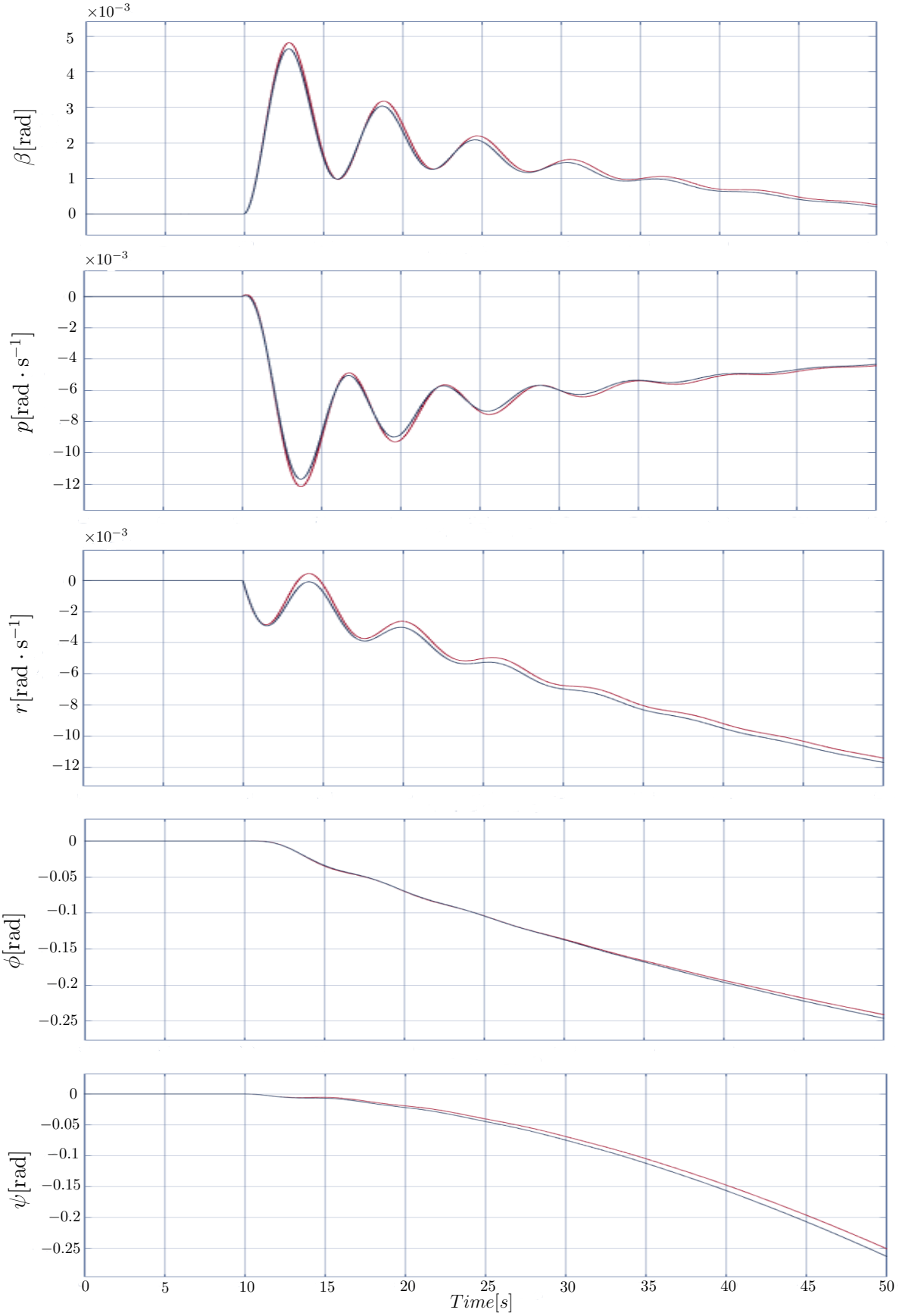

Figure 13: Rudder step response comparison

5 PID control/tuning optimization

In this chapter, the classical PID regulators will be reviewed and an optimization criterion for custom tuning will be presented. The PID regulators have a simple structure and are successfully used in a broad range of applications. Although they are not a universal solution for every regulation problem, they are robust and reliable in most cases.

5.1 Regulator structure

A proportional-integral-derivative controller (PID) utilizes a feedback loop to transform an error $e(t)$ into a correction input $u(t)$. The error value is computed as the difference between a desired setpoint $r(t)$ and a measured value $y(t)$ of a controlled variable $e(t) = r(t) - y(t)$. The controller then multiplies, integrates and takes the derivative of the error to produce a control input. In the parallel form, a PID controller takes the following form

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\mathrm{d}\tau + \mathrm{K_d}\frac{\mathrm{d}}{\mathrm{dt}}\mathrm{e(t)}. \tag{5.1}$$

This time domain formula describes an ideal PID regulator. Every real regulator is affected by some kind of inertia–induced delays. From now on, we will assume the PID regulator is ideal. Taking the Laplace transform of this control rule, we obtain

$$G_R(s) = \frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d s. \tag{5.2}$$

Equations (5.1) and (5.2) contain the terms K_p, K_i, K_d. These are gains, which have to be tuned for every application in order to function properly.

The **proportional** term produces an output signal, which is proportional to the current error value. The larger the gain K_p, the larger the action output of the regulator. Setting K_p too high can result in an unstable system. On the other hand, a low gain often fails to meet regulation time requirements. The biggest disadvantage of proportional control is its inability to compensate for steady–state error. This means that, the setpoint is not reached with a pure proportional controller. It is often described by a proportional band pp, which sets the percentage of input change to change the output by 100%.

$$pp = \frac{100}{K_p}[\%]$$

The **integral** term generates an action based on the integral of the error value over the regulation time. It is therefore slower to react compared to a proportional term, since the output is generated over time. It does have the benefit of driving the steady-state error to zero, however, the action of the controller rises as long there is an error. A pure integral controller leads to a less stable oscillatory response, and therefore combination with a proportional controller is advised. A second disadvantage of the integral term is the so called wind-up effect. If the integral term accumulates a large error during the regulation, it usually leads to high overshoot values. One of the possible solutions is to saturate the integrator output.

The **derivative** term produces an output that is proportional to the rate of change of the error. It responds fast and does not respond when the error stays constant. Derivative action predicts system behavior and improves settling time. Because the derivative term is

prone to exhibiting a dangerous response, when the error changes fast or instantaneously (step change), mean value of error change is sometimes used in discrete version of the controller. To avoid the effects of noise and other high-frequency inputs, low pass filters combined with the derivative term are usually used in real world applications. The step change of the setpoint results in a fast change in the error and a spike for the derivative term. Ramping the setpoint can have good effects on the control loop with regards to the prevention of such a response.

5.2 Multiobjective performance criterion

Finding a set of gains K_p, K_i, K_d that satisfy our requirements can be very challenging. Although basic tuning can be conducted manually with some intuition, it usually does not lead to an optimal solution. To describe the quality of a means of regulation, a performance criterion (or objective function) is often proposed and a solution space is searched. The objective functions can be classified as time or frequency domain–based objectives. The most commonly used time domain performance criteria are integral error performance criteria such as (ITAE) or (ITSE).

$$ITAE = \int_0^\infty t|e(t)|\mathrm{dt} \quad ITSE = \int_0^\infty te^2(t)\mathrm{dt}$$

Minimizing the weighted absolute or square error signals may lead to good results, but it does not guarantee that all the basic evaluation parameters such as overshoot M_p, rise time t_r, settling time t_s and steady–state error E_{ss} will be minimized. Multiobjective optimization takes all the desired parameters into account. A simple multiobjective criterion function may take this form:

$$\mathbf{J}(\mathbf{k}) = w_{M_p} M_p(\mathbf{k}) + w_{t_r} t_r(\mathbf{k}) + w_{t_s} t_s(\mathbf{k}) + w_{E_{ss}} E_{ss}(\mathbf{k}). \tag{5.3}$$

In (5.3), $\mathbf{k}$ represents a vector of PID gains $\mathbf{k} = [K_p, K_i, K_d]$. The biggest problem of such an approach is finding an appropriate set of weighting factors w_{M_p}, w_{t_r} and w_{t_s}. Since the range of each parameter is unknown, its percentual contribution to the fitness value is also unknown. Based on paper [7], the proposed time domain performance criterion evaluates the weighting factors according to their percentual contribution to the fitness value. The method of computing the weighting factors is based on multiobjective Pareto front solutions. The Pareto front is the set of Pareto optimal allocations. Pareto optimality is a concept drawn from game theory. It describes a condition in which we cannot improve the state of one individual without making another one worse off given the resources available. The concept is named after Vilfredo Pareto (1843–1923), the Italian engineer and economist. A Pareto front can easily be illustrated in two dimensions. Points A and B do not strictly dominate each other, but point C is strictly dominated by both A and B, thus C is not a part of the Pareto front. From this perspective, we can see, that dealing with multiobjective criteria does not lead us to one single best solution, but to a set of alternative solutions representing the best possible trade-off between the variables. Generally, the aim is to solve the problem

$$\min \mathbf{f}(\mathbf{k}) = \min[f_1(\mathbf{k}), f_2(\mathbf{k}), \ldots, f_j(\mathbf{k})],$$

where $f_i(\mathbf{k}) : \mathbb{R}^3 \to \mathbb{R}, \quad i = 1, 2 \ldots, j$ are objective functions. Based on application, the set of constraints $g_i(\mathbf{k}) \leq 0$ and $h_i(\mathbf{k}) = 0$ can be constructed in a such way that we

form a boundary on the acceptable solutions. A set of PID gains $\mathbf{k}_m$ is said to dominate $\mathbf{k}_n$, if and only if $\forall i \in \{1 \ldots, j\}$ we have $f_i(\mathbf{k}_m) \leq f_i(\mathbf{k}_n)$ and $\exists i \in \{1, ..., j\}$ so that $f_i(\mathbf{k}_m) < f_i(\mathbf{k}_n)$. If there is no solution $\mathbf{k} \in \mathbb{R}^3$ dominating a feasible solution $\mathbf{k}_m$, it is called a Pareto optimal solution. The set of Pareto optimal solutions will be denoted as $\mathcal{P} = \{\mathbf{k}_{p_1}, \mathbf{k}_{p_2}, ..., \mathbf{k}_{p_l}\}$. In terms of objective functions $f(\mathbf{k})$, the Pareto front is given by the matrix

$$\mathcal{PF} = \begin{Bmatrix} f_1(\mathbf{k}_{p_1}) & f_2(\mathbf{k}_{p_1}) & \ldots & f_j(\mathbf{k}_{p_1}) \\ f_1(\mathbf{k}_{p_2}) & f_2(\mathbf{k}_{p_1}) & \ldots & f_j(\mathbf{k}_{p_1}) \\ \vdots & \vdots & & \vdots \\ f_1(\mathbf{k}_{p_l}) & f_2(\mathbf{k}_{p_1}) & \ldots & f_j(\mathbf{k}_{p_1}) \end{Bmatrix}.$$

The goal of this process is to ensure that the main objective function (5.3) has a uniform contribution from all of its terms. This would be accomplished if all terms were equally

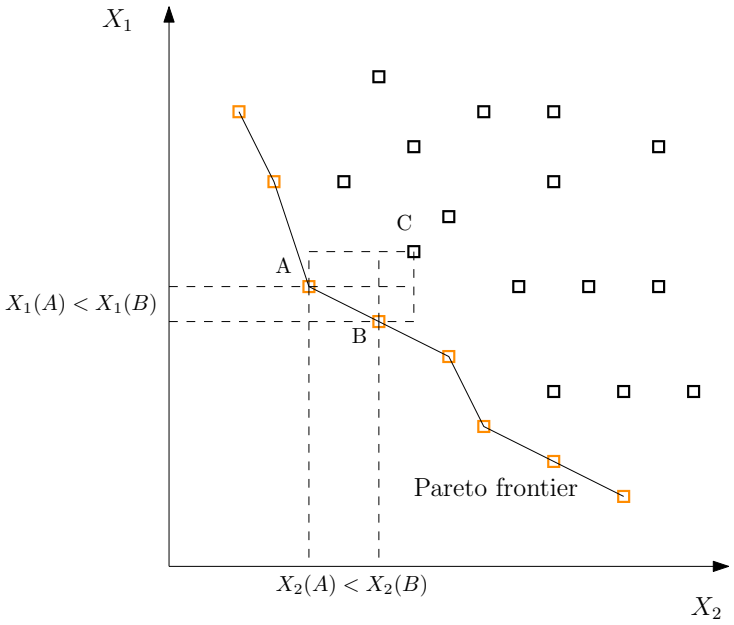

Figure 14: Example of a Pareto front

weighted and all terms had equal standard deviations. Since this is not the case and the standard deviations of E_{ss} and t_s are different in practice, for example $\sigma_{E_{ss}} \ll \sigma_{t_s}$, we have to compensate by setting $w_{E_{ss}} \gg w_{t_s}$. For each term $f_i(\mathbf{k})$, the contribution percentage of the term can be evaluated as

$$\mathcal{CP}\{f_i(\mathbf{k})\} = \frac{\mu_i}{\sum_{n=1}^{l} \mu_n} \tag{5.4}$$

where μ_i is the mean value of the corresponding Pareto solutions column $f_i(\mathbf{k}_{pn})$. The weighting factors are then inversely proportional to the contribution percentage

$$w_i = \frac{1}{\mathcal{CP}\{f_i(\mathbf{k})\} \sum_{n=1}^{j} \frac{1}{\mathcal{CP}\{f_n(\mathbf{k})\}}}.$$

By substituting the contribution percentage from (5.4), we obtain a simple expression for weighting factors

$$w_i = \frac{1}{\mu_i \sum_{n=1}^{j} \frac{1}{\mu_n}}. \tag{5.5}$$

Therefore, the objective function (5.3) weighted by (5.5) can statistically ensure the equivalent contribution of terms in a multiobjective criterion. Importance weights w_{ci} are introduced to provide the option of changing the importance of any objective intentionally. At the end, the general form of the objective function is

$$\mathbf{J}(\mathbf{k}) = \sum_{i=1}^{j} w_{ci} \left[\frac{f_i(\mathbf{k})}{\mu_i \sum_{n=1}^{j} \frac{1}{\mu_n}} \right]. \tag{5.6}$$

Objective function (5.6) maintains the same contribution values of its objectives and allows the custom setting of importance weights. An efficient search algorithm should be used to search for suitable solutions in (5.6). Article [7] recommends the usage of the particle swarm optimization algorithm (PSO).

5.3 PSO algorithm

The particle swarm optimization algorithm is a widely used metaheuristic algorithm. It can search large spaces of candidate solutions without becoming stuck at the local minimum. PSO tries to improve the candidate solution in an iterative manner by having a swarm of agents/particles search the solution space. In order to imitate social behavior found in nature, the agents share their knowledge of the best found solutions and change their movement accordingly. Agents are also affected by their own best found solution. The strength of the tendency towards a global best found solution and the tendency towards the agent's own best found solution is weighted appropriately. To describe the algorithm in mathematical terms, lets assume S to be a number of particles in n-dimensional space and that $\mathbf{X}_i \in \mathbb{R}^n$ is the position of the i-th particle. Let $\mathbf{V}_i \in \mathbb{R}^n$ be the velocity of a particle. Each particle keeps the position of its best found solution in $\mathbf{p}_i$. The best found solution shared among all particles is stored in a vector $\mathbf{g}$. Let the components of position vector $\mathbf{X}$ be bounded so that $x_1 \in \langle b_1, b_2 \rangle$; $x_2 \in \langle b_3, b_4 \rangle \dots x_n \in \left\langle b_{(2n-1)}, b_{(2n)} \right\rangle$ and let the particles be uniformly distributed within the bounds for the first iteration. The particles position will be updated in each iteration according to the equations

$$\begin{aligned} \mathbf{V}_i^{k+1} &= \lambda \mathbf{V}_i^k + c_1 r_{i1}^k (\mathbf{p}_i^k - \mathbf{X}_i^k) + c_2 r_{i2}^k (\mathbf{g}^k - \mathbf{X}_i^k), \\ \mathbf{X}_i^{k+1} &= \mathbf{X}_i^k + \mathbf{V}_i^{k+1}. \end{aligned}$$

In the literature, λ is called the inertia weight factor, which decreases in each iteration by a given value. This results in less movement by the particles and provides a balance between exploration and exploitation, thus requiring less iterations to obtain a sufficiently good solution. r_{i1} and r_{i2} represent random factors in the tendency of particles. Both $r_{i1}, r_{i2} \in \langle 0, 1 \rangle$. the weighting of the stochastic acceleration towards the local best and global best solutions is represented by c_1 and c_2. If c_1 were low enough, the particles would converge rapidly on one spot without searching the rest of the domain. If c_2 were low enough, the particles would not be affected by other particles and they could be considered as independent greedy search agents. Both constants are set from experience to 2. In PID tuning optimization, an appropriate solution space should be chosen in order to find good solutions. Position $\mathbf{X}$ is in this case equivalent to $\mathbf{k} \in \mathbb{R}^3$.

5.4 PID tuning example in Matlab

For this example, let us consider an open-loop transfer function for the aircraft pitch dynamics, where the input is elevator deflection angle δ_e and the output is the aircraft pitch angle θ. From our data, we have

$$P(s) = \frac{\Theta(s)}{\Delta(s)} = \frac{-1.706s^2 - 0.8531s - 0.01005}{s^4 + 1.178s^3 + 1.568s^2 + 0.00998s + 0.007295}.$$

We will implement combinations of proportional K_p, integral K_i and derivative K_d control in the unity feedback architecture in order to achieve the desired system behavior. We will demand these criteria for a step reference of 0.2 radians (magnitude is necessary to specify due to load factor limits). The maximum overshoot should be 15%, rise time should be less than 5 seconds and settling time less than 20 seconds. The maximum load factor should also be measured and compared with the limits for civil aircraft design between $0.05 - 0.4G$. A discrete form of the Pareto front for the multiobjective problem has been obtained. The boundaries were set to be $K_p \in \langle 0.001, 1.5\rangle$, $K_i \in \langle 0.001, 1.0\rangle$ and $K_d \in \langle 0.001, 1.0\rangle$. For all combinations of gain parameters with a step size of 0.2, the following Pareto front solutions were obtained. They are shown in Figure 15 just for reference. While optimizing the objective function via PSO, the step size was set to 0.05 to obtain a better approximation of the percentual contribution. Because of its low

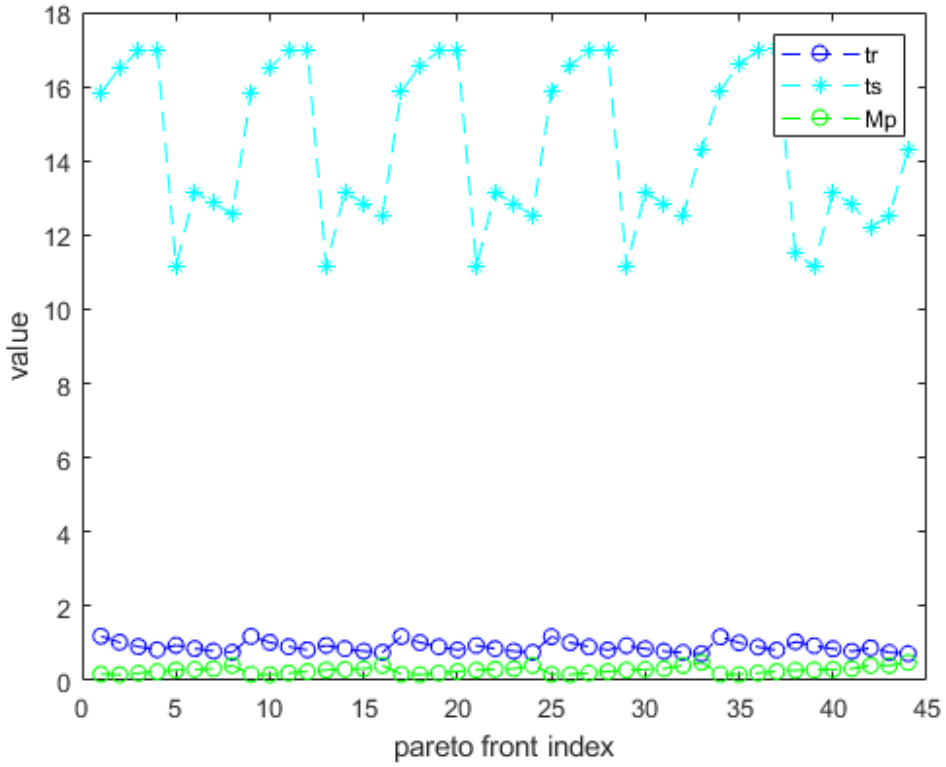

Figure 15: Pareto front results

contribution, steady–state error E_{ss} was completely omitted from the objective function. The final form of the objective function used in tuning was therefore

$$\mathbf{J}(\mathbf{k}) = w_{M_p} M_p(\mathbf{k}) + w_{t_r} t_r(\mathbf{k}) + w_{t_{sts}} t_s(\mathbf{k}).$$

To speed up the process, a constraint was added to eliminate results with a high sum of objective parameters.

$$M_p(\mathbf{k}) + t_r(\mathbf{k}) + t_s(\mathbf{k}) \leq B.$$

B is a predefined constant set to 30. Optimized weights were obtained from the discrete Pareto solution set. The values are $w_{t_r} = 0.2262$, $w_{t_s} = 0.0142$ and $w_{M_p} = 0.7596$. For the PSO run, the number of iterations was set to $N = 50$, swarm size was $S = 30$, the inertia weight factor $\lambda = [0.9 : 0.014 : 0.2]$ was decreasing from 0.9 to 0.2 and the number of trials was $T = 10$. We should therefore obtain 10 unique solutions from PSO. We can

Table 2: Solutions found by PSO

$\mathbf{J(k)}$	K_p	K_i	K_d	t_r	t_s	M_p
0.1580	1.0197	0.0624	1.9182	**0.9941**	27.6589	13.0870
0.1497	0.8299	0.0803	1.3525	1.1899	24.1071	11.7865
0.1552	0.8760	0.2449	1.9344	1.0596	**14.8747**	16.9372
0.1455	0.8220	0.0985	1.6886	1.1896	22.0455	11.7841
0.1422	0.8429	0.1210	1.2576	1.1713	20.1110	11.9907
0.1507	0.8156	0.0766	1.9455	1.1884	24.6524	11.7880
0.1463	0.8379	0.0905	1.4761	1.1782	22.9121	**11.6806**
0.1552	0.8538	0.2233	1.1946	1.1023	15.3255	16.5161
0.1810	0.6092	0.0562	1.7965	1.5475	27.0659	14.4138
0.1508	0.8590	0.1948	1.4281	1.1109	16.3080	15.3175

indeed see some variability in the results. The best found solution in terms of rise time was $t_r = 0.9941$ seconds. The best settling time from the 10 trials was $t_s = 14.8747$ seconds. The lowest overshoot among the candidates was $M_p = 11.6806$ %. Let us also plot

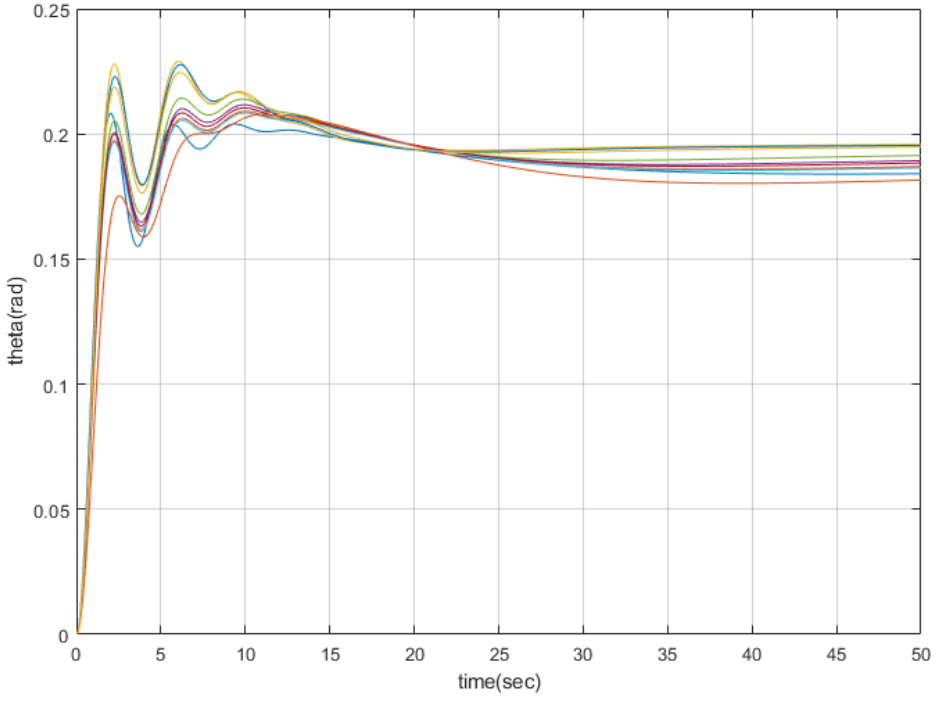

Figure 16: PSO best candidates responses

all the step responses to see how they vary. Depending on the conditions and measured normal acceleration, we may choose suitable PID gains. Some of the responses appear to have large settling times and some responses do not meet our requirements for overshoot. For high reference signal changes such as 0.2 radians, the requirements regarding normal acceleration are not usually met, since the response of the regulator is too fast. Reference tracking in the form of a ramp can suppress this effect. Multiobjective optimization pro-

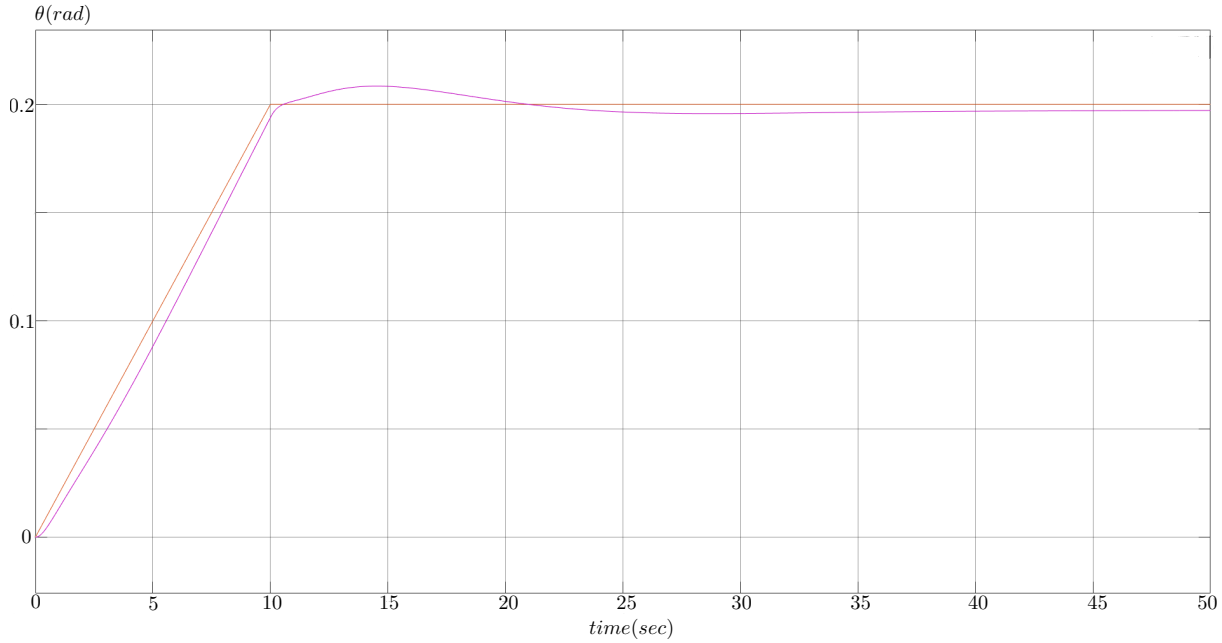

Figure 17: Ramp reference tracking

vides a good way of analyzing PID feedback loops. There are many other possible ways to tune a PID controller, though not all tools provide such a good awareness of the problem and options.

Let us proceed with the proposed optimization tool. As previously mentioned, maximum normal acceleration during the response is an important factor in the PID tuning process. Although easily measured in the simulation developed earlier, a simplified analytical way to compute normal acceleration using transfer functions has not been presented yet. Let us assume we need a normal acceleration response to an elevator assuming we measure the acceleration in the centre of gravity. The simplified equation has the form

$$a_z = \dot{w} - q(U)_s. \tag{5.7}$$

Taking the Laplace transform of (5.7), we obtain

$$a_z(s) = sw(s) - s\theta(s)(U)_s. \tag{5.8}$$

Then, we can express the normal acceleration transfer function in terms of elevator response transfer functions

$$a_z(s) = s\frac{N^w_{\delta_e}(s)}{\Delta(s)}\delta_e(s) - s(U)_s\frac{N^\theta_{\delta_e}(s)}{\Delta(s)}\delta_e(s) = \frac{s(N^w_{\delta_e}(s) - (U)_s N^\theta_{\delta_e}(s))\delta_e(s)}{\Delta(s)}.$$

Assuming pitch angle θ to be in a feedback loop, we can write

$$\mathbf{G}_0(s) = \left(K_p + \frac{K_i}{s} + K_d s\right)\frac{N^\theta_{\delta_e}(s)}{\Delta(s)}.$$

The pitch angle $\Theta(s)$ response to a reference value $\Theta_{ref}(s)$ can be written as

$$\Theta(s) = \frac{G_0(s)}{1 + G_0(s)} \Theta_{ref}(s). \tag{5.9}$$

Substituting equation (5.9) into (5.8) gives a normal acceleration response to the controlled pitch attitude. The elevator input signal can be expressed in terms of reference pitch attitude as

$$\delta_e(s) = \left(\Theta_{ref}(s) - \Theta(s)\right) G_R(s).$$

This developed transfer function forms a so called improper transfer function, where the order of the numerator is the same or higher than of the denominator.

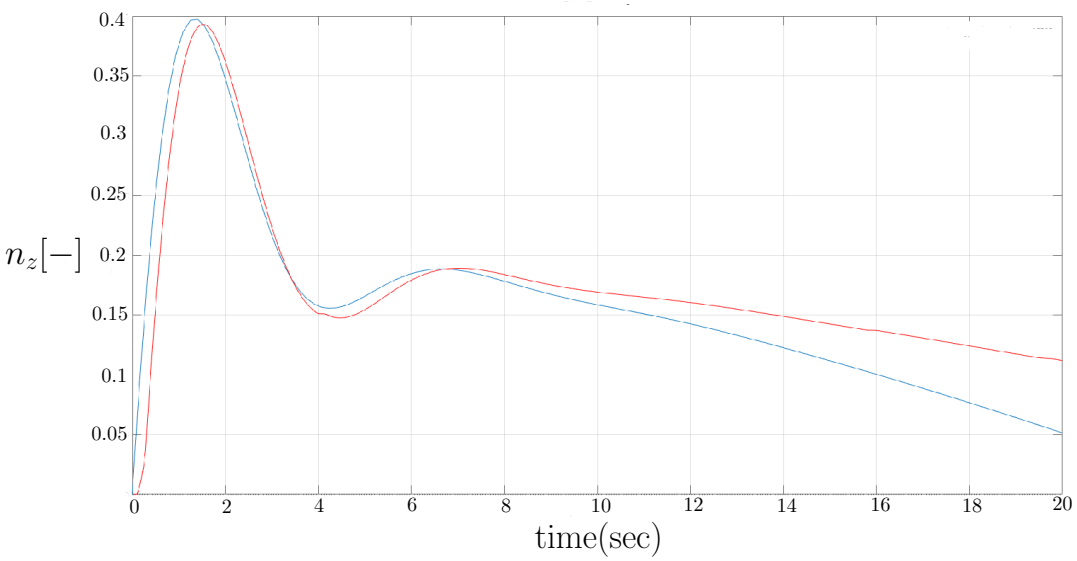

Figure 18: Normal acceleration approximation

For our purposes, the approximation based on circular motion is satisfactory. Because in the decoupled linear equations of motion, $\dot{\theta} = q$, we can approximate the non-dimensional normal acceleration as

$$n_z \approx \frac{\dot{\theta} V_{TAS}}{g}.$$

From a 1 deg step response on the elevator, it can be seen in Figure 18 that the modified pitch attitude transfer function matches the simulation response quite well for our purpose. To utilize the PSO algorithm with the proposed theory further, let us incorporate the normal acceleration requirement in the criterion function. Let $\Gamma(\mathbf{k})$ be the peak normal acceleration during the simulation period and let Γ_{req} be the peak permitted normal acceleration. The objective function is to be modified so that it drives the peak normal acceleration to its maximum allowed value. As shown in Figure 17, the maximum normal acceleration is highly dependent on the slope of the ramp reference signal φ. At the same time, φ highly affects other evaluation criteria. It is therefore natural to incorporate φ as an optimization parameter. The vector of optimization parameters is modified to $\mathbf{k} = [K_p, K_i, K_d, \varphi]$ with

$$\mathbf{J}(\mathbf{k}) = w_{M_p} M_p(\mathbf{k}) + w_{t_r} t_r(\mathbf{k}) + w_{t_{sts}} t_s(\mathbf{k}) + w_\Gamma(\mathbf{k}) |\Gamma_{req} - \Gamma(\mathbf{k})|.$$

Similarly, the constraint inequality was modified to

$$M_p(\mathbf{k}) + t_r(\mathbf{k}) + t_s(\mathbf{k}) + |\Gamma_{req} - \Gamma(\mathbf{k})| \leq B.$$

Several trials were run with the required peak normal acceleration set to $\Gamma_{req} = 0.1$. Other parameters of the optimization algorithm were preserved. Table 3 shows 10 trials of the algorithm. Note that $R_t = \theta_{ref}/\Gamma(\mathbf{k})$ and defines the time when the reference

Table 3: Solutions found to the modified problem

$\mathbf{J}(\mathbf{k})$	K_p	K_i	K_d	R_t	t_r	t_s	M_p	$max[n_z]$
0.0092	1.0653	0.4696	3.9107	4.7353	3.5386	14.1320	10.5362	0.1011
0.0105	1.9889	1.2703	3.2468	11.7568	9.0705	16.0267	4.2987	0.0497
0.0067	2.0835	0.1949	1.1319	5.3346	4.1009	20.6344	5.4133	0.0897
0.0071	1.9912	1.0982	3.4747	6.0955	4.3697	13.1599	6.8170	0.0932
0.0093	2.3604	1.0074	1.4371	8.6400	6.5121	15.1487	5.3939	0.0649
0.0074	2.0598	0.9537	3.5303	6.1323	4.4690	13.2833	6.8092	0.0905
0.0066	2.0792	0.3171	0.8506	5.1855	3.9402	17.5392	6.1706	0.0946
0.0069	2.4537	1.5278	2.8902	5.9658	4.3614	12.7356	7.3758	0.1006
0.0094	2.2242	0.9865	1.1653	7.1934	5.4624	14.1353	7.5081	0.0776
0.0097	2.6491	1.8374	3.5025	9.5924	7.5209	15.7835	5.5658	0.0645

signal hits the desired value. Once again, the table offers several settings, which can be used. The choice can be made simply based on the value of the objective function

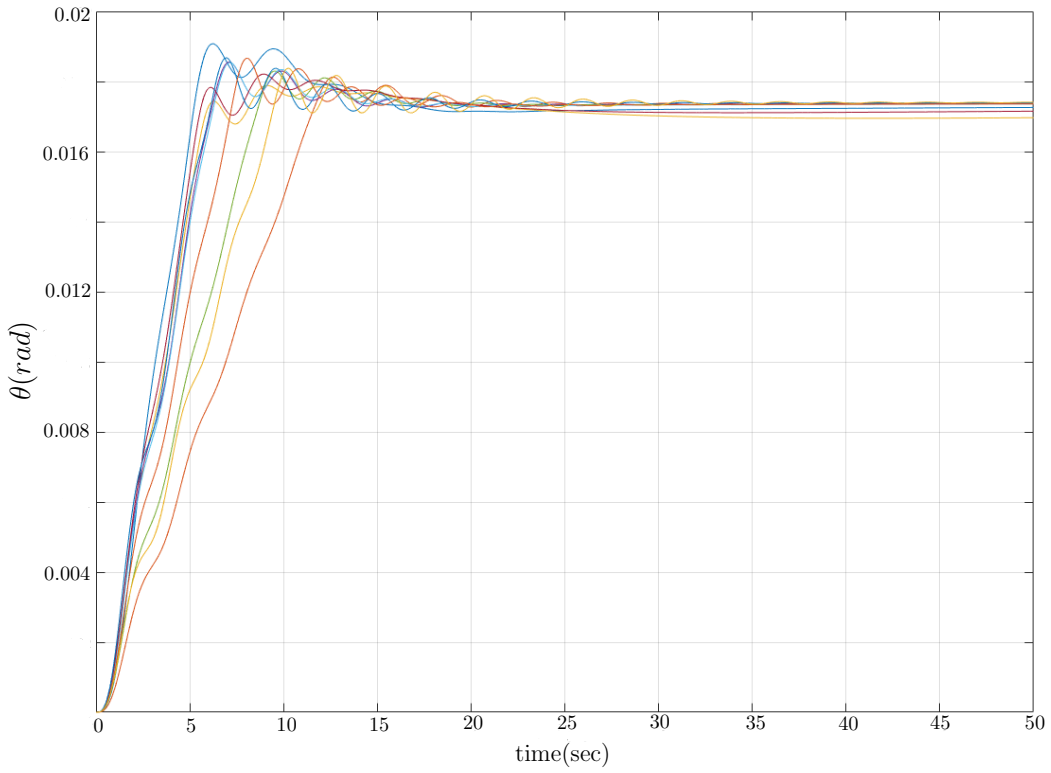

Figure 19: Best PSO candidates

or the preferred evaluation criteria. From another perspective, the resulting responses

can also be compared. The best criterion function value was chosen in this case, and the regulator was tested in the Simulink environment. The quality of response was also improved via the use of the trimmable horizontal stabilizer to trim the aircraft more quickly. The effect of the horizontal stabiliser is explained in the following chapters. Usage of the horizontal stabilizer led to a lower overshoot and lower settling time. Keep in mind, that a horizontal stabiliser has great authority over aircraft pitch but changes its deflection much slower or via pulse due to large forces acting on it. Overall quality was optimized by the criterion function, where the contribution weights were responsible for maintaining an equivalent contribution value for all the objective terms. For greater control over the optimization process, importance weights can be selected according to the design specifications indicated by an importance value. A general criterion function using importance weights would look like

$$\mathbf{J}(\mathbf{k}) = \sum_{i=1}^{j} w_{ci} \left[w_i f_i(\mathbf{k}) \right].$$

For example, we can now specify the importance factor of M_p to $w_{cM_p} = 0.7$ and all the other importance weights to

$$w_{ct_r} = w_{ct_s} = w_{c\Gamma} = \frac{1 - w_{cM_p}}{3}.$$

Playing with importance factors could lead to other well optimized solutions if one of the evaluation criteria is targeted. The incorporation of the normal acceleration requirement and the slope of the ramp function variable led to interesting results. One scenario from the above table was chosen. it was within 6% accuracy of the desired $\Gamma_{req} = 0.1$, specifically $\Gamma(\mathbf{k}^*) = 0.0946$. Although other control methods will be employed, the most important

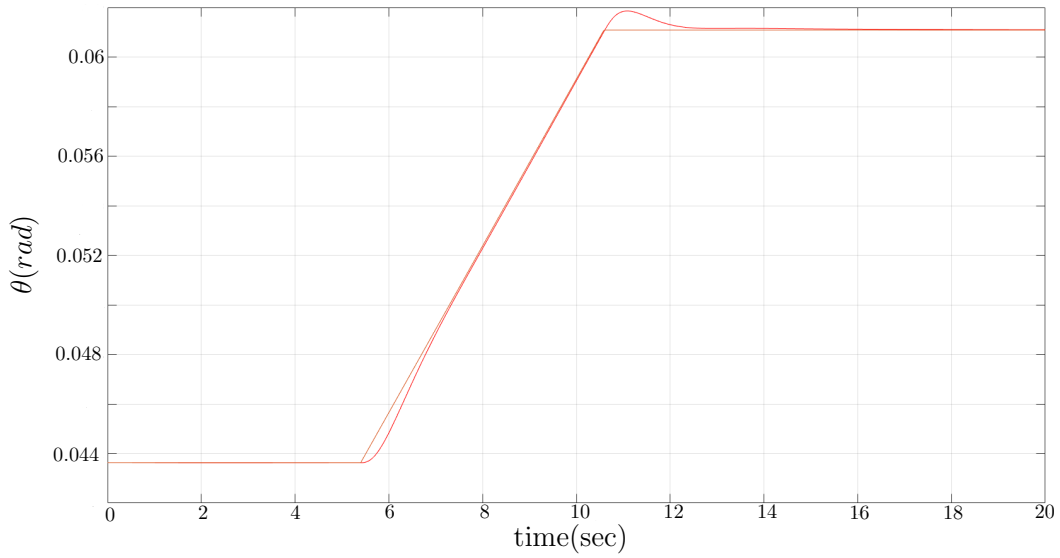

Figure 20: Simulation response with horizontal stabilizer

simple autopilot PID control loops will be tuned by variations of the Pareto optimality optimization approach.

5.5 Disk margin robust stability analysis

Feedback controllers are usually designed to achieve various performance objectives including disturbance rejection. Techniques to account for the mismatch between the plant model used to design the controller and the dynamics of a real system have been developed. Classical margins measure gain and phase perturbations that can be tolerated while retaining the closed-loop stability. In some cases, individual variations in gain or phase margins can be misleading since some systems can be very robust to both gain and phase margins individually but not that robust to a combination of both. A very comprehensive tutorial (found in [14]) on this topic was published by Cornell University and is a source for many statements regarding the theory formulated here.

Gain and phase variations will be modeled as a complex valued multiplicative factor f acting on the open loop system L defining a perturbed loop $L_f = fL$. To represent the gain and phase variations by a disk, we can restrict f to be a part of a parametrized family of disks in the following form

$$f \in D(\alpha, a, b) = \left\{ \frac{1 + a\delta}{1 - b\delta} : \delta \in \mathbb{C}, |\delta| < \alpha \right\},$$

where a, b and α are real parameters defining the set of disk parameters and δ is the normalized uncertainty modeled as an arbitrary complex value in the unit disk. Please, do not be confused by the symbol nomenclature. This subsection will use α as a disk parameter and not as the angle of attack. To avoid degenerate cases and simplify the parametrization, we can substitute for $a = 1/2(1 - \sigma)$ and $b = 1/2(1 + \sigma)$. The new parameter $\sigma \in \mathbb{R}$ will be referred to as a skew parameter, which leads to the simplification

$$f \in D(\alpha, \sigma) = \left\{ \frac{1 + \frac{1-\sigma}{2}\delta}{1 - \frac{1+\sigma}{2}\delta} : \delta \in \mathbb{C}, |\delta| < \alpha \right\}.$$

The most common approach is to select σ and compute the largest value of α for which the closed loop stability is maintained. The disk where all the combinations of phase and gain margins are stable is therefore defined by $\alpha_{max}(\sigma)$ for a given skew σ.

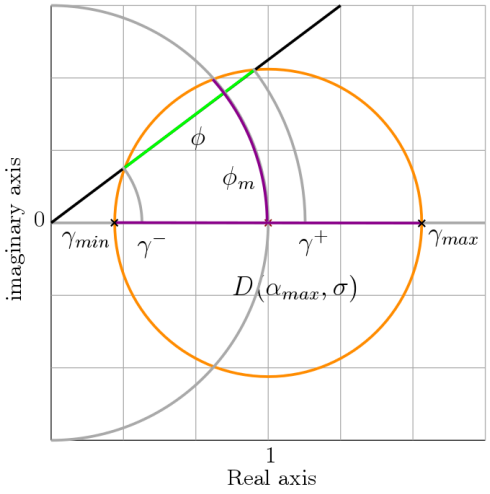

Figure 21: Admissible gain variations for a given phase variation

The disk margin α_{max} can be used to determine the gain and phase margins denoted as $(\gamma_{min}, \gamma_{max})$ and ϕ_m. The closed loop is stable for the portions of the unit circle and

the real axis that intersects the disk $D(\alpha_{max}, \sigma)$. If $D(\alpha_{max}, \sigma)$ contains the whole unit disk, then $\phi_m = \infty$ and the system is stable for all phase variations. We can think of the connection of classical and disk margins in the following way. If we consider a given phase variation ϕ as shown in Figure 21, we can determine a safe range of gain variation denoted as (γ^-, γ^+) for the given phase variation.

Disk margins have a very neat interpretation in the Nyquist plane. If we consider a typical case of $D(\alpha_{max}, \sigma)$ with $0 < \gamma_{min} < 1$ and $1 < \gamma_{max} < \infty$, in order for the system to be stable, for all perturbations $f \in D(\alpha_{max}, \sigma)$ and all frequencies ω, it must be $1 + fL(j\omega) \neq 0$. If rewritten as $L(j\omega) \neq -f^{-1}$, the condition can be interpreted as a Nyquist exclusion region meaning that the Nyquist plot $L(j\omega)$ does not enter the disk defined as $\{-f^{-1} \in \mathbb{C} : f \in D(\alpha_{max}, \sigma)\}$. The exclusion region always contains the critical point $(-1, 0)$ and is tangent to the Nyquist plot at some point. An example of such an exclusion region can be seen in Figure 22 with three different σ values. This approach extends

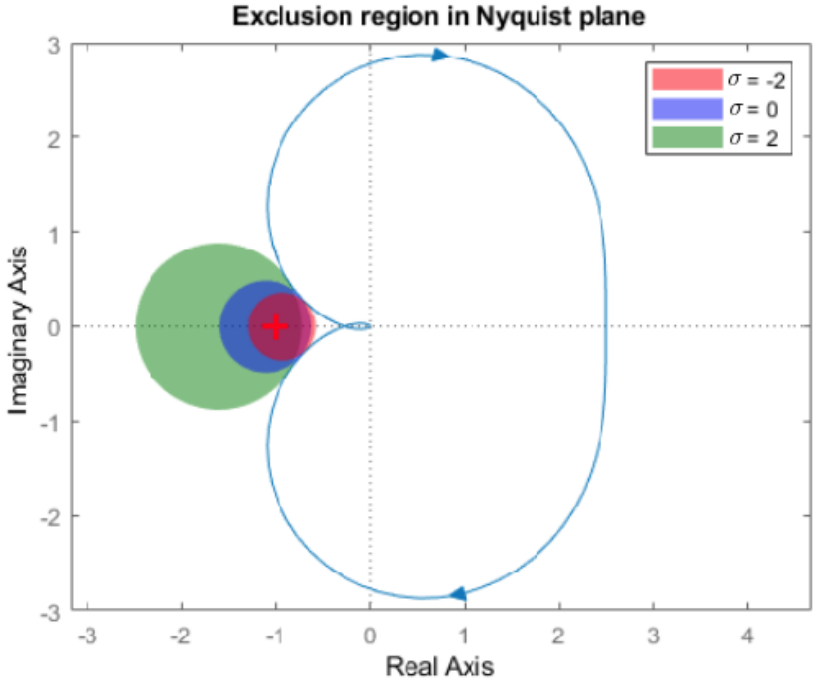

Figure 22: Exclusion region for variations of σ

the classical gain and phase margins and allows us to state more accurate assertions on the system stability. In practice, the only problem might be the unknown region of skew σ which is most relevant to our specific system. In some cases, it is reasonable to loop through σ and determine, how α_{max} changes, and possibly find the lowest α_{max} as the worst-case scenario.

The paragraphs above summarize the topic of disk margins for single-input-single-output systems (SISO). For multiple-input-multiple-output systems (MIMO) such as an aircraft dynamics state space model, we can simply represent uncertainties one channel at a time and proceed in the same way as with (SISO) disk margins. Unfortunately, this can sometimes be too optimistic, since combined uncertainties in multiple channels can have much lower stability margins than the one loop itself. However, it is also possible to model multiple uncertainties in the system at the same time. We can define a disk margin

at each channel as

$$f_j = \left\{ \frac{1 + \frac{1-\sigma}{2}\delta_j}{1 - \frac{1+\sigma}{2}\delta_j} : \delta_j \in \mathbb{C}, |\delta_j| < \alpha \right\}.$$

The model now replaces the open-loop response L with LF where

$$F = \begin{bmatrix} f_1 & 0 & \dots & 0 \\ 0 & f_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & f_N \end{bmatrix}.$$

All perturbations are restricted by the set $D(\alpha, \sigma)$, which is defined by a given skew σ. A common choice is to use symmetric disks of perturbations meaning that $\sigma = 0$. The multi-loop disk margin is given by a single number, α_{max}, which defines the largest disk of simultaneous perturbations $f_1, \dots f_N$ in such a manner that all phase and gain combinations in the disk yield closed loop stability. It is important to note that the individual perturbations might not be the same but vary independently. If we have an $n \times m$ system where n is the number of output states and m is the number of inputs, we can present perturbations at the plant output with $N = n$ perturbations or at the plant input by having $N = m$ perturbations. Alternatively, we can combine plant input and output perturbations by having $N = n + m$. Generally, perturbations introduced at more channels at the same time yield lower disk stability margins than when they are employed independently. It is therefore useful to have some knowledge of such margins when designing a controller. In the development phase of each flight director mode (which is the main topic of the thesis), the disk margins will be examined to prevent the involvement of controllers with low robustness.

Example 5.5.1. In this example, we will examine the closed loop stability of the previously developed SISO pitch attitude hold controller used as an example for multiobjective optimization. The open loop structure was determined to be

$$L(s) = G_R(s)P(s) = \frac{3.584s^4 + 2.422s^3 + 0.3495s^2 + 0.01047s + 7.968 \cdot 10^{-5}}{s^6 + 1.203s^5 + 1.597s^4 + 0.04918s^3 + 0.007544s^2 + 0.0001824s}.$$

The Bode plot of this transfer function would show us that the gain margin is infinite and the phase margin is 23.8 deg . at a frequency of $\omega = 2.08$ rad/s. If we want to compare the results with disk–based margins, we start by choosing the skew $\sigma = 0$ and computing the margins. The minimum gain margin $\gamma_{min} = 0.6658$ and maximum gain margin $\gamma_{max} = 1.5019$, along with the phase margin $\phi_m = \pm 22.6872$ deg . define the maximum range of the disk region. The disk margin $\alpha_{max} = 0.4012$ defines the region of stability around $\sigma = 0$. The gain-only and phase-only disk–based margins are a bit conservative compared to the classical margins, but they guarantee much stronger robustness. The worst–case disk margin is the smallest disk margin that occurs in the range of uncertainties in L and the minimum guaranteed margin over the range of uncertainties. Computing the worst–case disk margin for the pitch attitude hold system yields $\alpha_{max} = 0.4004$, which is not such a decrease from the previously computed margin for skew $\sigma = 0$. We may conclude that the controller is robust to combinations of phase and gain uncertainties to a satisfactory degree.

6 Flight control system development

All modern aircrafts include a variety of automatic control systems that help the pilot with the control and navigation of the aircraft. Aircraft typically operates in a large variety of conditions, which usually depend on Mach number and altitude. Based on the airframe and engine characteristics of the aircraft, we can define the so-called *flight envelope*. The boundaries of this envelope are determined by many factors. At low speeds, a stall speed boundary is present. The slower the aircraft flies, the greater the angle of attack needed to produce lift equal to the aircraft's weight. If the speed decreases further, the needed angle will approach the critical stall angle of attack. The need for an airspeed boundary is thus obvious. At high speeds, the limit follows a constant dynamic pressure contour defined by the aircraft's structural limits. At higher altitudes, speed becomes limited by the maximum thrust the engine can generate. The maximum height is bounded by a service ceiling where the minimum rate of climb cannot be achieved. In order to develop a robust

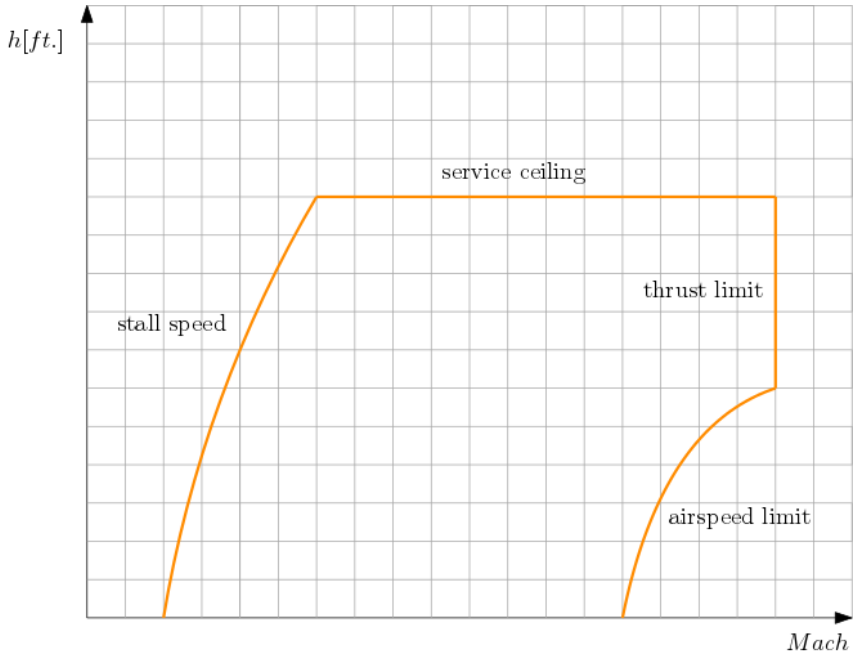

Figure 23: Graphical representation of flight envelope

controller, many linearizations must be performed over a large set of trim points in the defined flight envelope and the stability characteristics must be measured. The smooth transitions of controller parameters during the flight are established by a gain scheduling approach. Another option would be a nonlinear controller (nonlinear dynamic inversion–based for example) which would guarantee stability over the whole flight envelope. More on this topic can be found, e.g., in [8].

Before we start to implement specific controllers for the selected flight condition, let us examine servomechanism effects. While the regulator generates the control input $\mathbf{u}$ as a reaction to current state $\mathbf{x}$, the control signal reaches a servomechanism which generates the control effort. Servomechanisms have the ability to amplify and transform signals to controls by means of electric, hydraulic or pneumatic actuators. In general, the dynamics of a first order proportional servomechanism is described by

$$T_s\dot{\omega}(t) = -\omega(t) + K_{sm}u(t),$$

where ω is the angular velocity of the servo, K_{sm} is servomechanism gain and T_s is a time constant. It is natural that the actuators have some limits and cannot follow extreme changes in the control input instantaneously. The transfer function of this dynamics is then

$$A(s) = \frac{\omega(s)}{u(s)} = \frac{K_{sm}}{T_s s + 1}.$$

For our purposes, $K_{sm} = 1$ and time constant T_s will vary depending on the situation. For example, the elevator servo time constant will be much lower than the trimmable horizontal stabilizer time constant. All the control inputs will also be saturated so that they request only positions which are reachable for the aircraft's control surfaces.

6.1 Pitch-Attitude hold

Although pitch control was shown for the PID optimization example, many technical aspects were omitted in order to focus on the optimization method. Controlled variable θ is usually measured by the attitude reference gyro. The controller does not hold flight path angle γ, because angle of attack α can change with flight conditions. A pitch-attitude control loop architecture, as implemented in Simulink, is shown in Figure 24. The desired

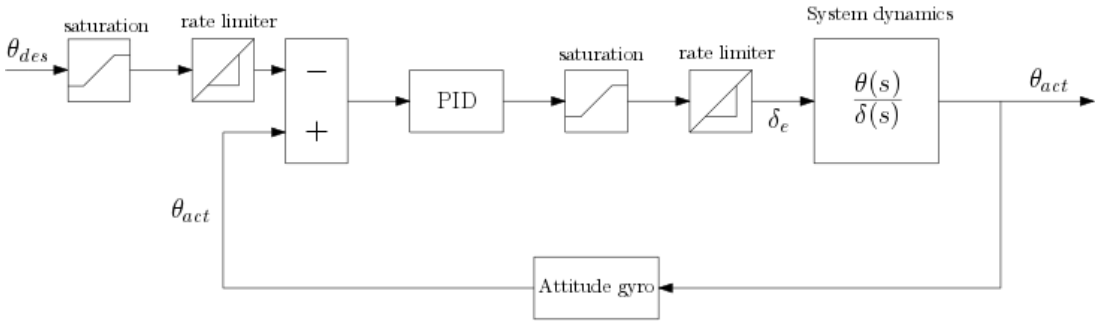

Figure 24: Pitch-attitude hold control loop

pitch angle θ_{des} was restricted to take values in $\langle -25°, 25° \rangle$ and PID controller output (elevator deflection angle) was saturated at $\langle -20°, 20° \rangle$. Exploring the closed-loop frequency characteristics of the designed PID controller provides us with the following Bode diagram: We obtain a phase margin of 29.1 deg at 2.79 rad/s. The closed loop is stable at all frequencies. As mentioned in the previous section, the effect of a trimmable horizontal stabilizer improved the response characteristics of the pitch-attitude hold even further. It features a fully moving horizontal tail surface which adjusts its angle to ensure optimum elevator effectiveness. The trimmable horizontal stabiliser is primarily used to alleviate the load on the elevator and its servo. Its secondary purpose is to enhance the achievable flight envelope of the aircraft, since, on the 747, the pitching moment achievable by the elevator is not sufficient to achieve an equilibrium condition throughout its flight envelope. The fact that the trimmable horizontal stabiliser moves does not actually improve stability of the aircraft. Let us recall that the moment coefficient equation is set to zero.

$$C_{m_0} + C_{m_\alpha}\alpha + C_{m_{i_h}} i_h + C_{m_{\delta_e}} \delta_e = 0.$$

At any point during flight, the stabilizer incidence angle i_h should therefore be set to the value which helps the elevator to remain at a zero deflection angle. From the steady–state

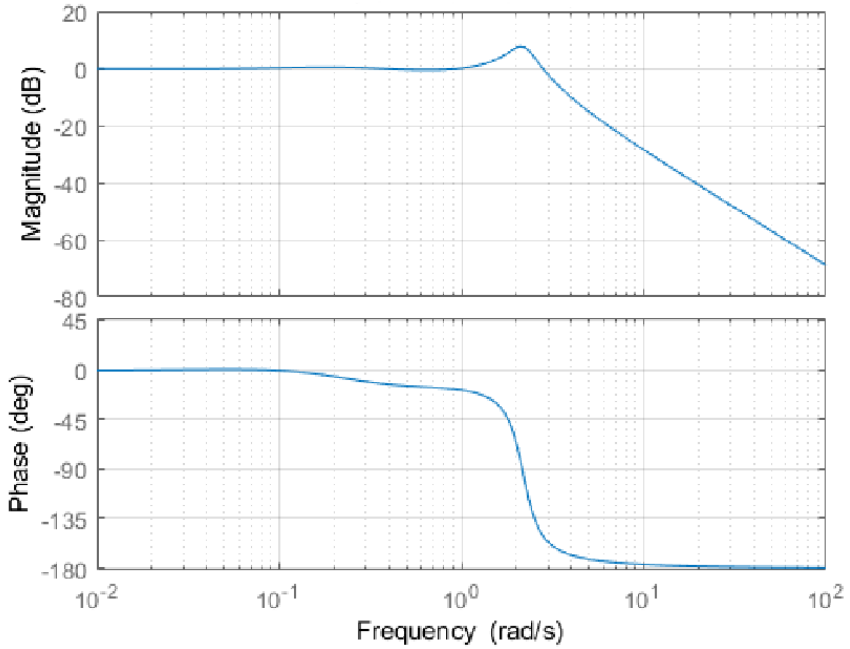

Figure 25: Pitch-attitude hold Bode plot

moment's coefficient equation, we obtain

$$i_h = -\frac{C_{m_0} + C_{m_\alpha}\alpha + C_{m_{\delta_e}}\delta_e}{C_{m_{i_h}}}.$$

The trimmable horizontal stabiliser is too slow to react to sudden demands for pitching moment. Its drive mechanism is not fast enough and does not have the precision required to do that. Therefore the elevator is used to control pitch and the trimmable horizontal stabiliser is used to alleviate loads on the elevator servo, by a series of pulses.

6.2 Altitude hold

Altitude hold allows the aircraft to remain at a fixed altitude. Even under indement weather conditions, altitude hold should hold the aircraft within 100 ft of a specified altitude and warn the pilot if this limit is exceeded. Although altitude hold tracks the error between the actual and desired altitude, it is not designed for smooth transitions when the height difference is too large. We will be using the pitch altitude hold as the inner loop of the design. Because the flight path angle $\gamma = \theta - \alpha$ is required to approach zero once the height feedback error reaches zero, the signal of α is added to the height feedback error. The controller design is depicted in Figure 26. Block G_F represents the effective lag of the pressure-altitude measurement. Experiments have shown that a simple proportional controller is suitable for altitude hold. The need for γ to be zero when the altitude error reaches zero, would not be satisfied by the addition of cumulative or derivative commands. The choice of gain K_p depends on the ability of the controller to hold the aircraft within limits during turns or turbulence. The gain was set to $K_p = 0.001$ after several experiments with the heading select mode. The aircraft experienced a maximum roll angle ϕ of 25 deg . and altitude hold kept the aircraft safely within 100 ft. We can see altitude hold response

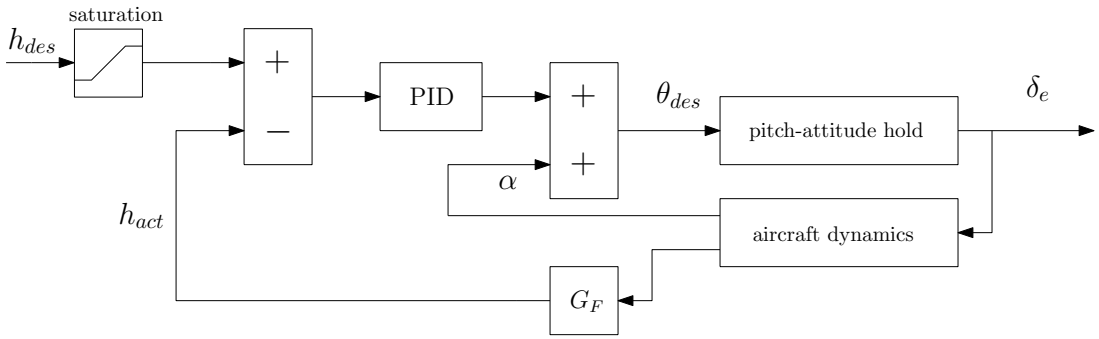

Figure 26: Altitude hold control loop

while heading select is active in Figure 27. Keep in mind that altitude hold should only be

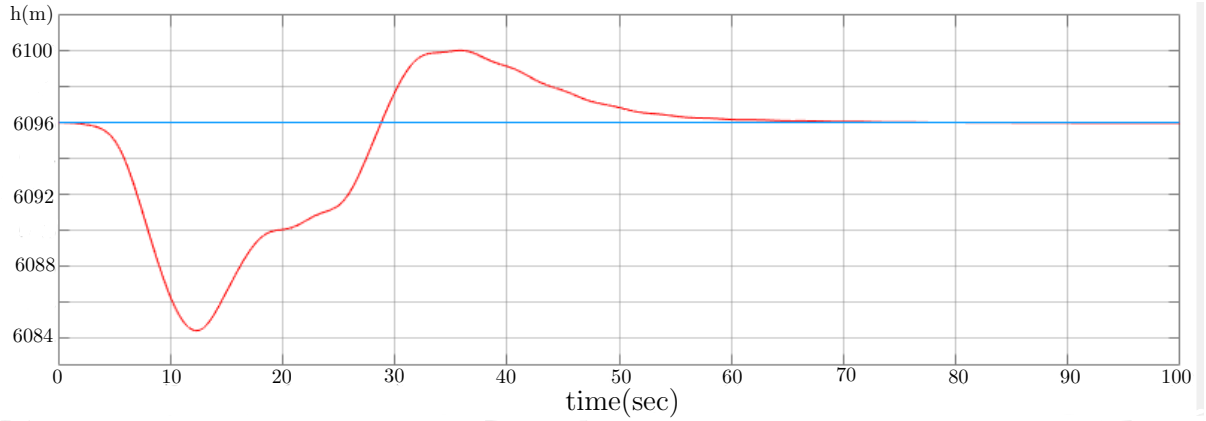

Figure 27: Altitude hold response in a level turn

used when the altitude error is within ±15 m since it is not a mode capable of changing altitude safely. To prove the stability of the developed controller, block diagram algebra has been utilized to analyze this cascaded control loop. Classical gain and phase margins showed that the loop has a 23.1 dB gain margin at 1.31 rad/s frequency and a 64.8 deg . phase margin at 0.201 rad/s frequency. The disk–based margins yielded an $\alpha_{max} = 1.0513$ disk margin with a gain margin interval of $(\gamma_{min}, \gamma_{max}) = (0.3109, 3.2165)$ and a maximum phase margin of $\phi_m = 55.4593$ deg.

6.3 Roll angle hold

Roll angle ϕ hold forms an important inner loop and the basis for other lateral-directional autopilots. A sensor such as a gyroscope is used to measure deviations of the roll angle. The autopilot maintains a wings-level flight by controlling the ailerons. This provides the pilot with relief function and also eliminates the danger of unnoticed development of a spiral descent. If the aircraft is held at some other attitude than wings-level, additional control of pitch rate and sideslip are needed due to the coupling of the variables. An additional control mechanism results in a coordinated turn where the sideslip angle is held at zero. The simplest form of roll angle hold architecture based on a PID controller can be seen in Figure 28.

Let us go deeper into the design of roll angle hold. We will start with an open-loop

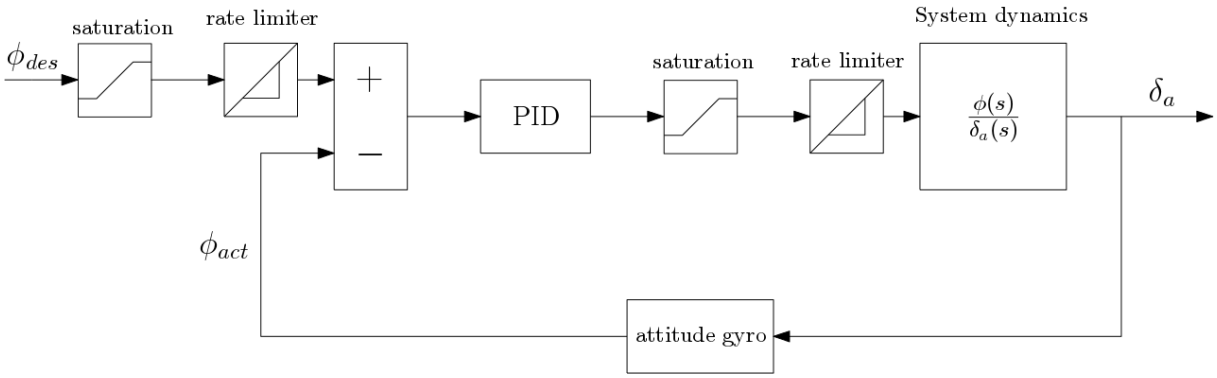

Figure 28: Roll angle hold control loop

roll angle to aileron transfer function

$$R(s) = \frac{\Phi(s)}{\Delta(s)} = \frac{0.2234s^2 + 0.08512s + 0.2628}{s^4 + 1.218s^3 + 1.375s^2 + 1.08s + 0.01807}.$$

A similar strategy as in pitch control was employed. The PSO algorithm was set to find an optimal set of gains and ramp reference slope. Acceleration was omitted in this case. By setting a custom weight for overshoot of $w_{cM_p} = 0.7$, settings resulting in a low overshoot response are preferred. The following performance characteristics of $\mathbf{k}^*$ were measured for one degree change in the target value of ϕ. Overshoot $M_p = 2.7647$ % was among the lowest in the set of admissible solutions. Settling time $t_s = 11.9769$ s and rise time $t_r = 3.4642$ s were sufficient trade-offs for the low overshoot value. Ramp time $R_{\mathrm{t}} = 3.4932$ s is a sufficiently fast run-up for the desired value. Figure 29 shows the response of the Simulink model with implemented gains.

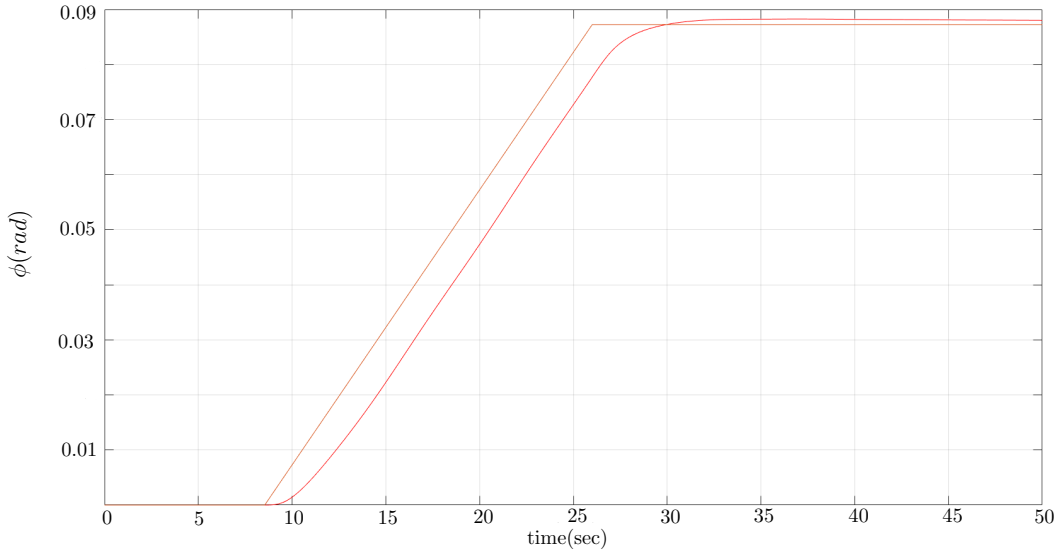

Figure 29: Roll angle hold response

After finding an optimal set of gains $\mathbf{k}^* = [K_p, K_i, K_d, \varphi]$, the closed loop stability analysis should be worked out. The Bode plot shows the minimum stability margin to be a phase margin of 64.4 deg . at a frequency of 0.516 rad/s. The symmetric disk margin

for roll angle hold yields $\alpha_{max} = 0.8524$ with gain margin bounds (0.40232.4854) and $\phi_m = \pm 46.1658$. Although ϕ reference tracking was accomplished, no attention was given

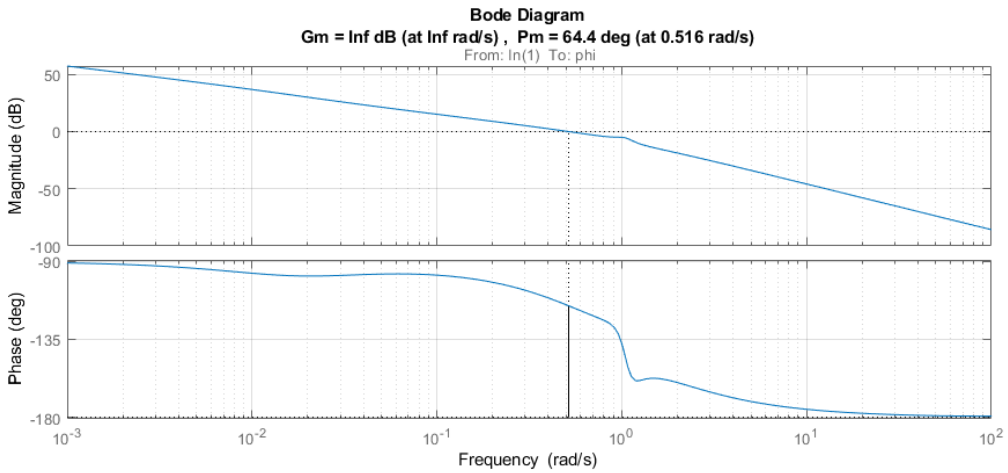

Figure 30: Bode plot for roll angle hold

to coupled variables. An introduction to turn coordination shall now be presented.

A coordinated turn is defined as a turn where zero acceleration in the y-axis of the aircraft's body is observed. Turn coordination is desirable for passenger comfort and better flying qualities. By minimizing sideslip, aerodynamic efficiency is also maintained. During a coordinated turn, the pitch and roll attitude is maintained, but the aircraft's heading changes at a constant rate. With such conditions, the kinematic equations with $\dot{\theta} = \dot{\phi} = 0$ take the form

$$\begin{aligned} P &= -\dot{\psi} \sin\theta, \\ Q &= \dot{\psi} \cos\theta \sin\phi, \\ R &= \dot{\psi} \cos\theta \cos\phi. \end{aligned}$$

Since, by definition, body y-axis acceleration $\dot{V} = 0$ and the forces sum to zero, we can write the motion equation as

$$0 = -UR + WP + g \sin\phi \cos\theta. \tag{6.1}$$

Substituting body angular velocities P and R into equation (6.1) leads to

$$g \sin\phi \cos\theta = -\dot{\psi} \left[U \cos\theta \cos\phi + W \sin\theta \right].$$

Transforming velocities U and W into V_{TAS}, we obtain

$$g \sin\phi \cos\theta = -V_{TAS} \dot{\psi} \cos\beta \left[\cos\alpha \cos\theta \cos\phi + \sin\alpha \sin\theta \right]. \tag{6.2}$$

By expressing the centripetal acceleration as

$$\vartheta = \frac{V_{TAS} \dot{\psi}}{g},$$

we can rewrite equation (6.2) as

$$\sin\phi = \vartheta \cos\beta \left(\sin\alpha \tan\theta + \cos\alpha \cos\phi \right).$$

Via a tedious algebraic work and solving for variables ϕ and θ, a trim algorithm can be used to find a steady state turning condition by varying α and β in addition to the control surfaces. A simplified constraint for $\gamma = 0$ and $\beta = 0$ can be expressed as

$$\tan\phi = \frac{\dot{\psi}}{g}\frac{V_{TAS}}{\cos\theta}.$$

For a specified turn rate $\dot{\psi}$, the required pitch and yaw rates can be calculated from the kinematic equations and the roll rate P can be neglected if $\cos\theta \approxeq 1$. More easily, the required rudder compensation can be approximated from the yawing moment coefficient equation under the assumption that $\beta = 0$ as

$$\delta_r = -\frac{C_{n_{\delta_a}}}{C_{n_{\delta_r}}}\delta_a.$$

In the Simulink model, the turn coordination scheme was applied and rudder input was calculated to obtain a coordinated turn. We can see the resulting rudder input during the test maneuver in Figure 31. To obtain a level steady–state turn, the altitude hold

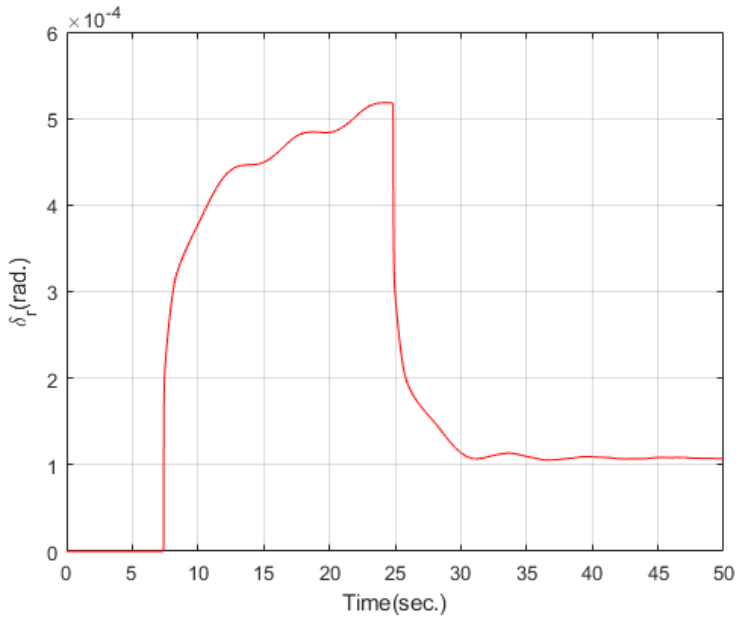

Figure 31: Rudder input for a coordinated turn

autopilot should be active when roll angle hold is triggered. At the same time, sideslip angle β is kept close to zero by a yaw damper developed in the next section.

6.4 Yaw damper

The yaw damper is a stability augmentation system developed to damp yawing oscillations observable in the Dutch-roll mode. The sideslip angle feedback loop will be used to damp the undesired yawing oscillation. The sideslip angle to rudder transfer function has the following form

$$Y(s) = \frac{\beta(s)}{\Delta(s)} = \frac{0.01438s^3 + 0.646s^2 + 0.5494s - 0.008099}{s^4 + 1.218s^3 + 1.375s^2 + 1.08s + 0.01807}.$$

The yaw damper was implemented using **controlSystemDesigner** in Matlab. During

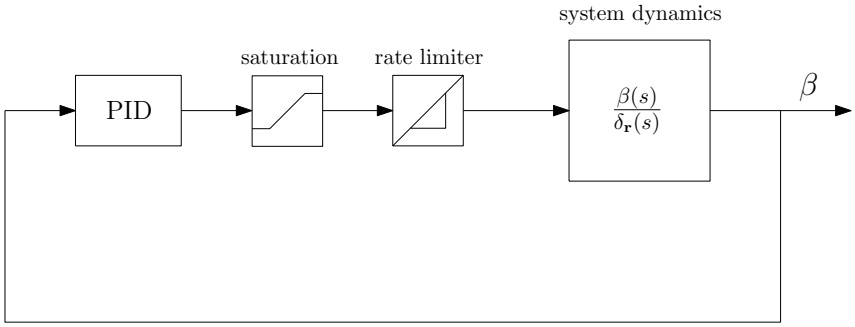

Figure 32: Yaw damper control loop

the test maneuvre, satisfactory damping of sideslip angle development was achieved. We can see the response in Figure 33. For most aircrafts, the yaw damper is active all the time and provides an important stability augmentation. Because β is coupled with yaw rate R and there is usually no reliable way to measure the angle β, an optional yaw damper design based on yaw rate can be used. A tutorial on modern yaw damper design [16] was followed to implement a yaw damper. By using a simple proportional controller with

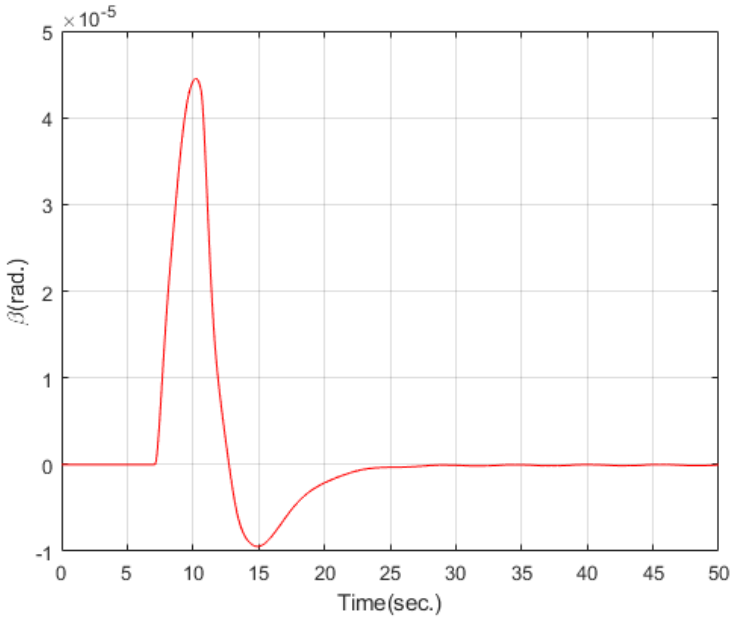

Figure 33: Yaw damper response during a turn

$k = 2.85$, we can see the reduction in yaw oscillations in Figure 34. Unfortunately, when

the ailerons are moved, the aircraft seems not to react properly due to over-stabilization of the spiral mode. The aircraft should bank without constant aileron input. This problem

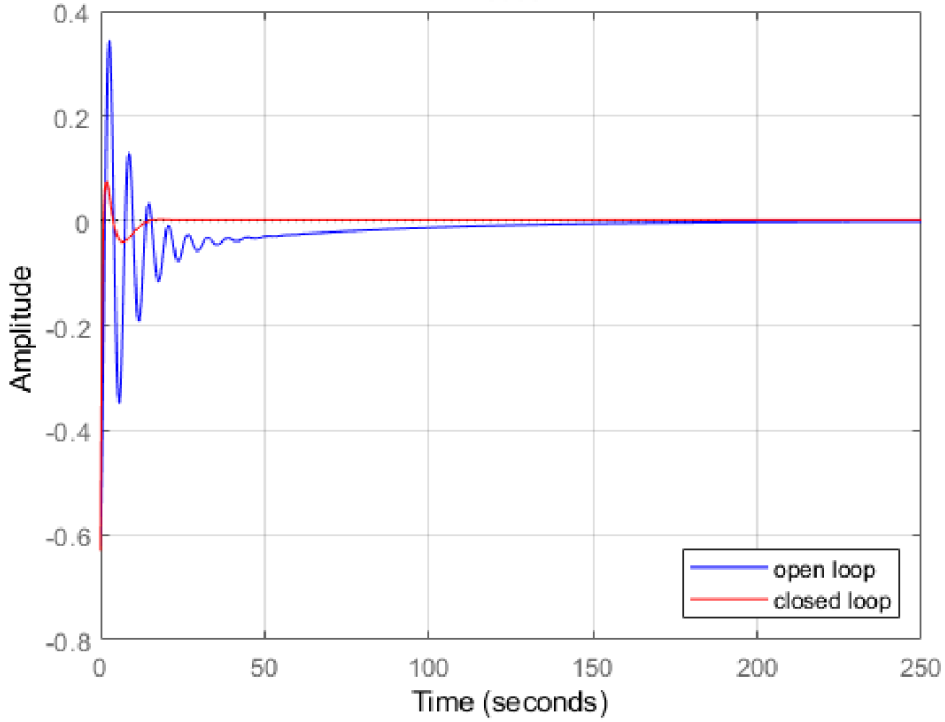

Figure 34: Yaw oscillation reduction

can typically be solved by a washout filter. This filter can be described by a transfer function with the form

$$W(s) = \frac{ks}{s+a}.$$

Using the **SISO design tool** in Matlab, the washout filter was designed with a time

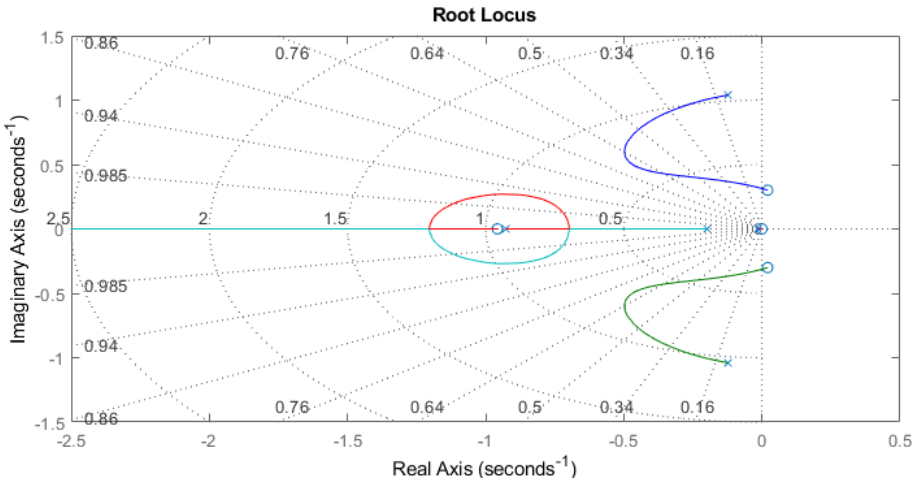

Figure 35: Root locus plot

constant set to $a = 0.2$. To select a correct gain, it is useful to plot a root locus (see Figure 35) with the implemented washout filter. The highest damping ratio of the Dutch roll mode was achieved at $k = 1.56$ with a damping of $\zeta = 0.617$. The washout filter helped to restore the normal bank and turn behaviour when using the ailerons as shown in Figure 36. As expected, the aircraft maintained a non-zero bank angle as a response to aileron impulse. With most aircraft, the yaw damper is suppressed during takeoffs and

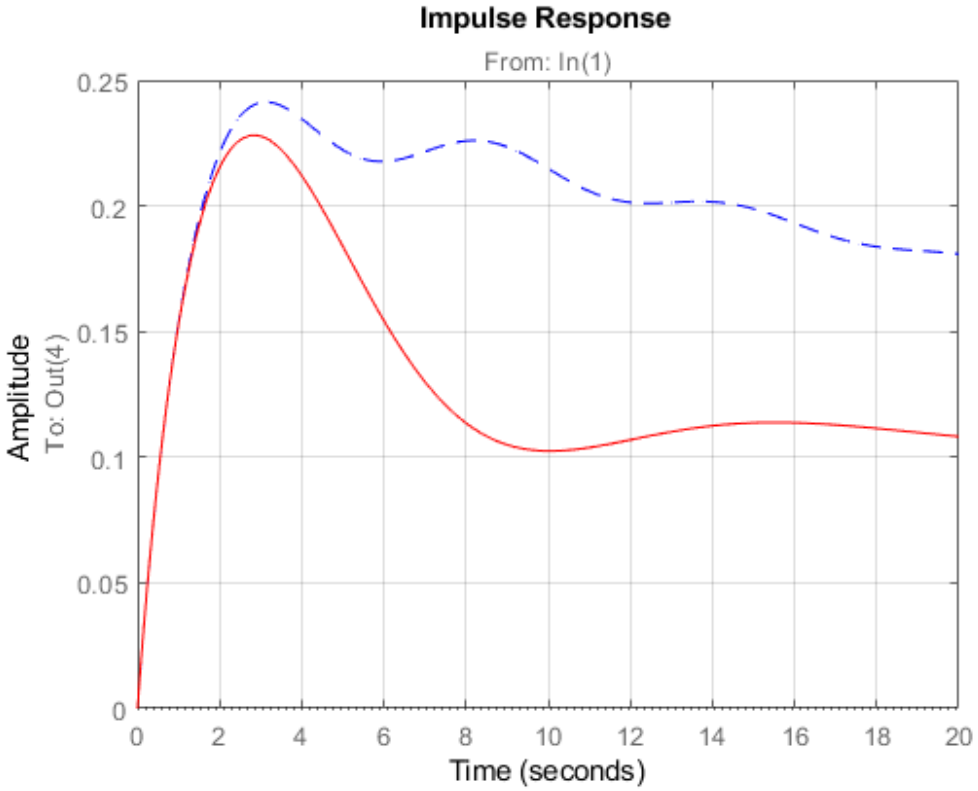

Figure 36: Bank angle development of open and closed loop systems

landings. If it is not, the pilot may end up fighting against the yaw damper to make corrections, which is especially for high crosswind situations. In some cases, automatic deactivation of yaw damper is performed by the flight control system when a particular minimum altitude is reached. The disk margin of the open loop system was computed to be $\alpha_{max} = 1.5819$ for symmetric skew with a gain margin $(0.1167, 8.5674)$ and a maximum phase margin $\phi_m = 76.6849$ deg.

6.5 Heading select

The heading hold autopilot is an important navigational mode designed to hold the aircraft at a selected compass heading. A typical heading hold setting features an extra feedback loop of yaw angle ψ around the developed roll angle hold autopilot with a turn compensation feature. When activated, the heading hold should turn the aircraft to a certain direction in such a way that the turn does not require more than 180 deg. Therefore, there is no fixed positive or negative feedback, but a decision mechanism which picks the sign based on the current and desired yaw angle. A simple scheme of the heading select mode is depicted in Figure 37.

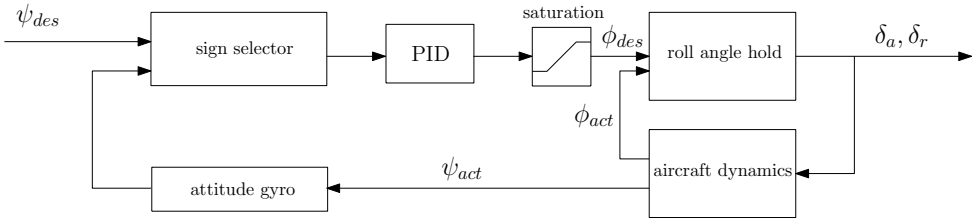

Figure 37: Heading select control loop

The logic behind the sign selector block was developed in Simulink. For every case of actual and desired heading combinations, the controller steers the aircraft in the most logical direction, so that the aircraft does not need to turn more than 180 deg . in yaw. The Simulink scheme of the sign selector is presented in Figure 38. Note that *Psi actual* can reach any value within the range $\langle 0, 2\pi \rangle$. Similarly as with the altitude hold, the hea-

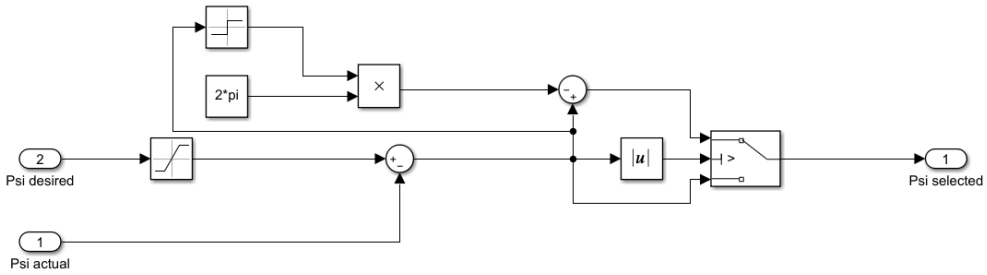

Figure 38: Simulink model of the sign selector

ding select seems to have the best performance characteristics when only K_p gain is used. A gain of $K_p = 2.27771$ was selected and no overshoot in heading was present during turns to the right and left. Several heading responses were simulated for $\psi \in \langle 0, 120 \rangle$ deg . with a step of 30 degrees. The responses can be seen in Figure 39. The rate of heading

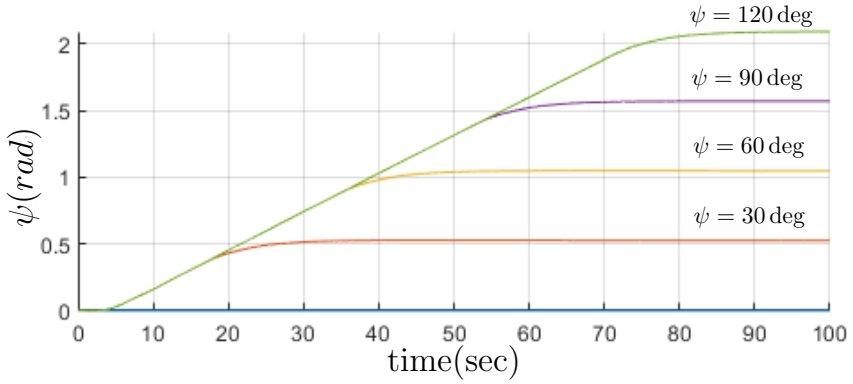

Figure 39: Heading select responses to a sequence of commands

change achieved for this setting is approximately $\dot{\psi} \approx 1.5\,\text{deg}$. Optionally, heading select can be designed in such a way that the pilot can also set the heading rate.

The stability of the designed heading select controller yielded an infinite gain margin

and a 29.7 deg . phase margin at a frequency 1.22 rad/s. The disk–based margins were computed to find $\alpha_{max} = 1.2428$ with a gain margin interval $(\gamma_{min}, \gamma_{max}) = (0.2335, 4.2824)$ and a phase margin of $\phi_m = \pm 63.7121$ deg.

6.6 Vertical speed mode

In vertical speed mode (VS), the flight director maintains the selected vertical speed reference. While using the VS mode is rather convenient if one wishes to predict the duration of a climb or descent, the speed-based climb (FLC) can employ the most economical airspeed. Because the vertical speed mode handles vertical speed by definition, care should be taken during a climb to avoid a dangerous decrease in airspeed and possible stall. Ideally, the VS mode should hold at least some minimum required airspeed. On the other hand, the descent should be also checked so that the aircraft does not overspeed. In both the climb and the descent, the climb or descent rate V/S is the vertical component of the true airspeed. This corresponds to

$$\dot{h} = V/S = V_{TAS} \sin \gamma. \tag{6.3}$$

The airspeed is equal to the groundspeed if there is no tail-wind or head-wind. A positive γ means that the aircraft is climbing while a negative γ indicates a descent. The rate of climb is labeled as *ROC* and rate of descent as *ROD*. It is good to point out that no automatic climb autopilots can be activated until a minimum engagement altitude (usually 500 ft) is reached. The VS mode gives a pitch attitude command to hold the selected V/S, while the throttle position is varied to hold a constant airspeed. Altimeter and vertical airspeed meter measurements are necessary for VS mode implementation. The pilot can check the vertical speed using the vertical speed indicator (VSI), which is one of the basic flight instruments found aboard a plane. After reaching a selected altitude, the vertical speed mode should be disengaged and altitude hold automatically activated. This strategy leads to a large altitude overshoot, since the aircraft reaches the altitude with selected V/S and has to slow down quite rapidly. This problem can be solved by implementing ASEL (altitude capture mode), which disengages the climb mode in advance to control the aircraft's ascent to the selected altitude smoothly, after which the altitude hold is activated.

In this case, the value of control variable γ is computed from (6.3). The error signal is generated by subtracting it from the actual value. It is recommended that the speed hold function is active in order to prevent the aircraft from losing speed while climbing. The simple block diagram in Figure 40 shows a general implementation of the vertical speed mode. Note that it is rather a simplified version due to the complexity of the actual Simulink model's implementation and signal routing. Since γ is the control variable, the controller design can be based on the $\frac{\gamma(s)}{\Delta(s)}$ transfer function. Since γ is not directly a state variable in our linearized model, we can obtain it by extending the output matrix C and using $\gamma = \theta - \alpha$. The controller indirectly controls the required V/S by computing γ_{des} from (6.3).

$$V(s) = \frac{\gamma(s)}{\Delta(s)} = \frac{0.03769s^3 - 0.03068s^2 - 0.8456s - 0.001092}{s^4 + 1.178s^3 + 1.568s^2 + 0.00998s + 0.007295}.$$

The vertical speed mode can be seen in action in Figure 41. In the first phase after VS activation, the aircraft starts to change γ in order to gain the required vertical speed. In

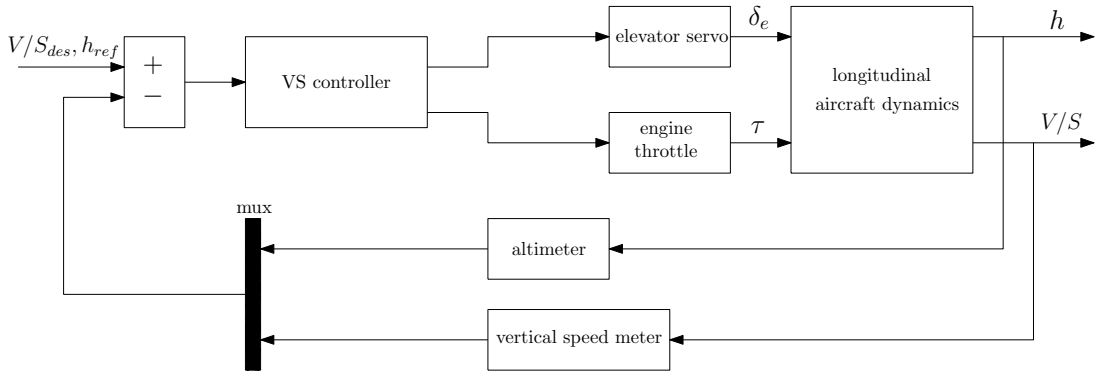

Figure 40: Simplified VS control loop

the second phase, the aircraft changes its altitude with constant γ and constant V/S, if the true airspeed is held constant by the throttle input. In the third phase, just after the aircraft reaches a predefined height error, altitude capture mode is triggered to achieve a smooth transition into altitude hold mode on a circular path. The designed controller

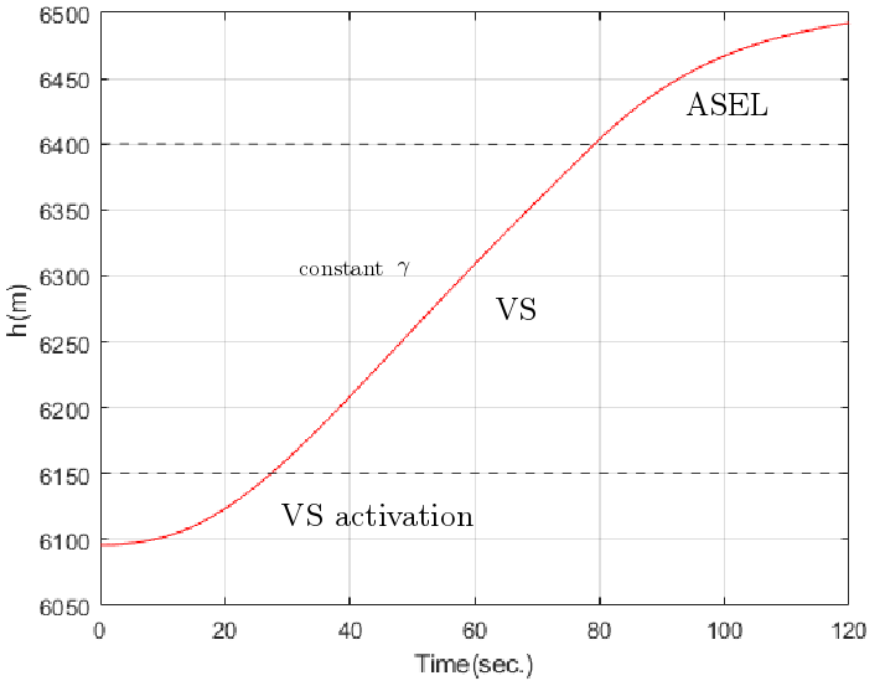

Figure 41: Example of VS change of altitude

with a set of gains $\mathbf{k} = [1.0024, 0.2174, 1.1451]$ yields a classical gain margin of 4.83 dB at a frequency of 1.17 rad/s and a phase margin of 37.8 deg . at 0.684 rad/s. The disk margin was less optimistic, with $\alpha_{max} = 0.4357$ with gain margin bounds $(0.6422, 1.5570)$ and a maximum phase margin of $\phi_m = \pm 24.5796$ deg. The vertical speed mode was tested for various commands. High altitude climb tests were employed as a part of altitude capture mode testing. The focus was placed on shallow climbs with an altitude difference of 400 m. From Figure 42 it can be seen that lower speed commands work as expected. The altitude capture mode in place (which is described later) steers the aircraft to a circular capture path very soon after the start of the climb. We cannot see if the aircraft hits the

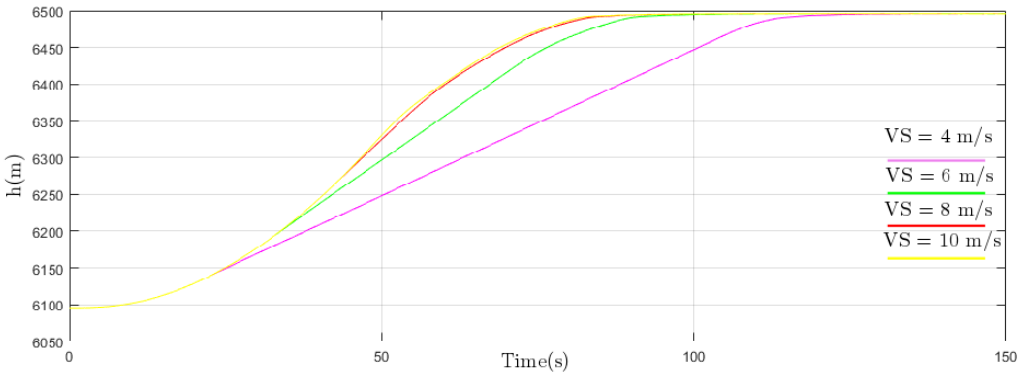

Figure 42: Climb simulations with various VS commands

desired vertical speed since the control logic shuts the VS mode down.

6.7 Flight level change

Flight level change mode (FLC) is a type of automatic climb/descent mode, which is mainly based on airspeed hold. The selected airspeed is controlled by the elevators and the excess power used to change the altitude is set by a climb thrust setting. FLC is claimed to be the best mode to use in a climb since the indicated airspeed is held constant which is considered to be an efficient way to change the altitude. The simplest type of flight level change mode is the indicated airspeed mode (IAS), which blindly holds the airspeed regardless of the desired altitude. This causes the aircraft to descend at the beginning of mode activation in order to gain the required airspeed before pitching up and climbing. The FLC mode is thus a more advanced version of this controller which forbids the aircraft to pitch down if the desired altitude is higher. The aircraft holds the altitude instead before it gains the required airspeed via an increase in thrust. The excess thrust is then used for climbing. The governing equation used to describe the process is

$$T = D + mg\sin(\gamma) + ma_x, \tag{6.4}$$

where T is the total thrust, D is the total drag and a_x is the aircraft acceleration in the x-axis of the body. In (6.4) we can assume that the thrust and drag are equal and any excess thrust is expressed by the aircraft's acceleration. In order to maintain the speed, the acceleration is converted to a change in flight path angle as

$$\Delta\gamma = \arcsin\left(\frac{a_x}{g}\right).$$

The flight level change controller takes the desired change in flight path angle and sums it with α. This allows, the FLC controller to pass θ_{cmd} commands to the pitch attitude hold autopilot. Figure 43 shows a the general block diagram of the flight level change flight director. In order to develop the FLC mode, one needs to use the altitude to elevator transfer function and the velocity to throttle setting transfer function. The thrust has not been assumed in the state-space model yet. Nevertheless, the values for X_τ, Z_τ and M_τ for the assumed condition have been found in [6]. They can be generally computed as

$$X_\tau = \frac{1}{m}\left(\frac{\partial T}{\partial \tau}\right)\cos\phi_T,$$

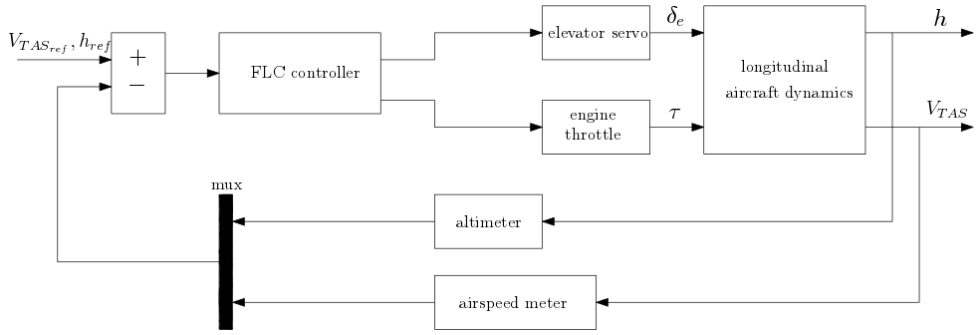

Figure 43: FLC block diagram

$$Z_\tau = -\frac{1}{m}\left(\frac{\partial T}{\partial \tau}\right)\sin\phi_T,$$
$$M_\tau = -\left(\frac{\phi_T}{I_{yy}}\right)\left(\frac{\partial T}{\partial \tau}\right)\cos\phi_T.$$

as shown in [6]. When the throttle setting τ increases, there is a corresponding increase in thrust. The corresponding stability derivatives are $X_\tau = 3.434 \times 10^{-6}$, $Z_\tau = -1.5 \times 10^{-7}$ and $M_\tau = 0.67 \times 10^{-7}$. The response of velocity u to the throttle setting was subsequently derived as

$$\frac{u(s)}{\tau(s)} = \frac{3.434 \cdot 10^{-5}s^3 + 4.019e-05s^2 + 5.008e-05s - 3.615 \cdot 10^{-6}}{s^4 + 1.178s^3 + 1.568s^2 + 0.00998s + 0.007295}.$$

Based on the defined transfer function, a simple airspeed controller can be developed assuming throttle actuators, jet engines and airspeed and accelerometer sensors. The change in thrust is caused by a change in throttle setting which alters the flow of fuel to the engines. This process has a time constant T_E, which specifies the jet engine time constant. The airspeed sensor is represented by a first order transfer function with a time constant T_u. Another important factor is the system's authority over the thrust. We can evaluate K_E based on the percentage of system authority. For example, if maximum thrust $T_{max} = 800$ kN and if only 10 percent authority is allowed, than $K_E = 80$ kN.

6.8 Altitude capture mode

Developed climb modes such as VS mode or FLC mode provide a well controlled apparatus for the automatic climb and descent of an aircraft. Though the aircraft's flight control system tracks the difference between actual and desired altitude, the climb modes do not take into account the smooth capture of the selected altitude. An instant switch from the climb mode to the altitude hold would result in undesirable overshoot. Strictly speaking, the altitude hold is not suitable for such transitions. For this reason, the altitude capture mode (ASEL) was developed to lead the aircraft in a circular path when approaching a selected altitude under desired normal acceleration. This normal acceleration is usually limited to $0.05g$ for passenger comfort. Developing a robust control law for an altitude capture mode seems to be relatively straightforward, but in reality it poses many challenges. First of all, initial conditions at the time of altitude capture activation can be widely different. The aircraft can be in a climb or descent at a range of flight path angles and speeds. A geometrical description of the problem is depicted in Figure 44. The control

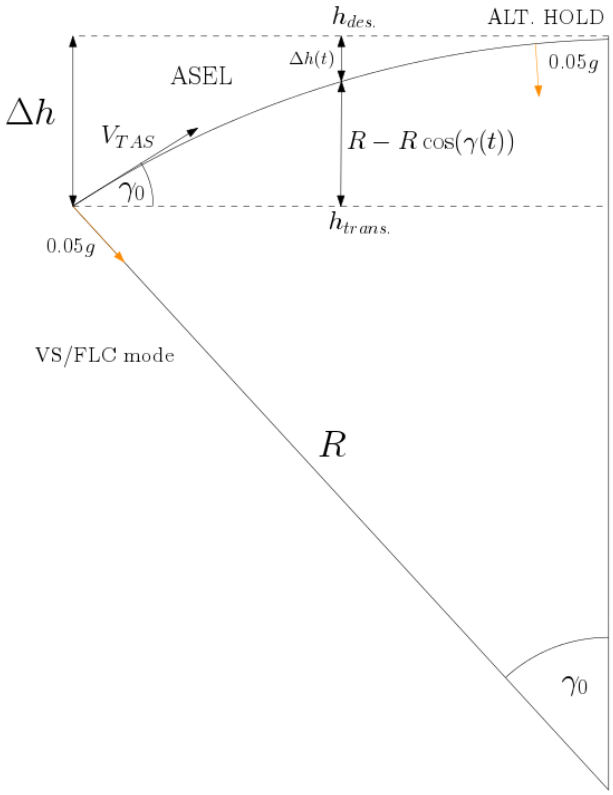

Figure 44: Altitude capture mode

strategy has two phases, one of which is activated when a climb mode such as VS or FLC is active. Based on the current vertical speed $\dot{h}$ and the selected normal acceleration for the capture maneuvre, the time t_c to capture the aircraft is estimated along with the transition altitude error Δh as well as h_{trans}. This estimate has the purpose of selecting an optimal transition altitude h_{trans} so that the smooth capture of the selected altitude under a chosen normal acceleration can be achieved. These values are computed at every step/real time, so any changes in vertical speed are taken into account. The controller logic switches the climb mode to ASEL, whenever

$$|h_{des.} - h(t)| \leq \left|\Delta h(\dot{h})\right|. \tag{6.5}$$

Note that dependence on $\dot{h}$ is assumed in the transition condition equation. The term $\Delta h(\dot{h})$ should be saturated to prevent any unexpected spikes in $\dot{h}$ resulting in the early activation of the mode. Let $\Delta h(\dot{h}) \in \langle -100, 100 \rangle$ m. The transition to ASEL can work well only if certain control input transitions are present. The control input generated by the climb mode at the time of transition, (let us denote it $\delta_e(t_0)$) has to be maintained and summed with the control input generated by ASEL. Switching the control input to zero at the time of transition would result in unstable behavior from the aircraft and the failure of the capture maneuvre. This generally applies to other autopilot transitions as well and it will be mentioned later.

Once the ASEL is activated, initial flight path angle γ_0, initial vertical speed $\dot{h}(t_0)$ and Δh are stored to a current instance of the controller. Practically speaking, the storage of values can be achieved in several ways in Simulink, the most popular being via the usage of **Switch** and **Memory** blocks to store/freeze a value on event. From here, a capture

radius R is computed from the initial values

$$R = \frac{\Delta h}{1 - \cos(\gamma_0)}.$$

With respect to a radius R, $\dot{h}(t_0)$ and required normal acceleration, $\Delta h(t)$ is computed at each step to back-calculate a desired flight path angle γ_{des}. Note that $\Delta h(t)$ is a predicted value needed for a circular altitude capture, and it is not the actual difference between the desired and actual altitude. Ideally, our controller should ensure that the actual altitude difference approaches the estimated one $|h_{des.} - h(t)| \rightarrow \Delta h(t)$. The value γ_{des} linearly decreases to zero and is controlled through a PID regulator commanding elevator. We can write

$$\delta_e(t) = K_p \left(\gamma_{des}(t) - \gamma(t)\right) + K_i \int_{t_0}^{t} \left(\gamma_{des}(\tau) - \gamma(\tau)\right) \mathrm{d}\tau + \mathrm{K_d} \frac{\mathrm{d}}{\mathrm{dt}} \left(\gamma_{\mathrm{des}}(\mathrm{t}) - \gamma(\mathrm{t})\right).$$

To further stabilize the circular flight path tracking, a proportional command feeds the difference of the predicted and actual altitude error, so the PID controller changes the flight path angle in a manner that is accordingly close to the actual altitude error:

$$\Delta\delta_e(t) = K \left(\Delta h(t) - |h_{des.} - h(t)|\right).$$

This control strategy works as long as $\Delta\delta_e(t) \ll \delta_e(t)$. It practically means that the gain K has to be a small number. In our simulation, good performance was achieved with $K = 0.001$. Once the altitude difference becomes small enough for the altitude hold controller to take over, another transition is initiated and ASEL is deactivated. The transition condition for the altitude hold activation is

$$|h_{des.} - h(t)| \leq 10\mathrm{m}.$$

Again, once ASEL is deactivated, its elevator command should smoothly change to an altitude hold command, so its last value is therefore summed with the altitude hold command. Ideally, the commanded flight path angle $[\gamma_{des}]_{ALT.HOLD}$ of altitude hold should be close enough to the commanded flight path angle of ASEL $[\gamma_{des}]_{ASEL}$. This require-

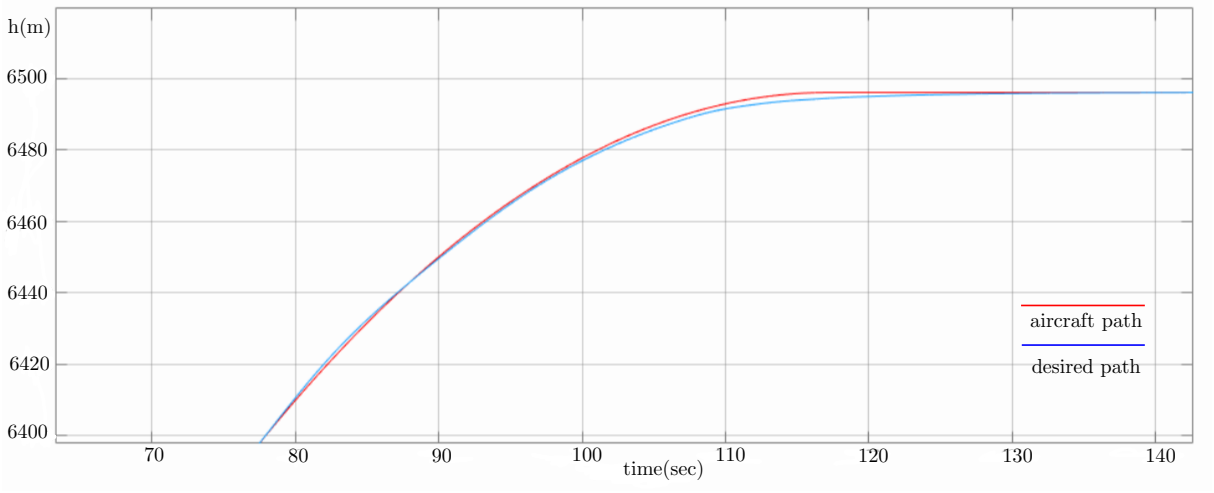

Figure 45: Altitude capture mode tracking of the desired flight path

ment could mean, in some cases, that the transition should be performed at a different altitude difference than the fixed 10 meters. This is not taken into consideration in the implemented design. The logic behind the transitions of modes has been implemented in *Stateflow*, a graphical language based on finite state machines. Details of the *Stateflow* implementation will be mentioned in a separate section. As an example, tracking of the optimal circular flight path (in blue) using the developed altitude capture controller at $V/S = 5$ m/s is presented below in Figure 45. Note that this capture maneuvre was measured under specific conditions so it serves just as an example of desired behavior in the case of a robust design. Additional testing has been performed in different conditions. Our design has been tested for the vertical speed mode (VS) and flight level change mode (FLC) at different speed and vertical speed commands, as well as for climb and descent. Several normal acceleration requirements were also tried out. Figures 46 and 47 show climbs from $h = 6096$ m to $h = 6396$ m at two different vertical speeds. Based on the vertical speed and selected normal acceleration, a different transition altitude error Δh was chosen. It can be seen that choosing higher normal accelerations at lower vertical speeds leads to quite a low transition altitude error Δh that can result in an overshoot since the reaction time of the aircraft has some limits. More simulations were conducted

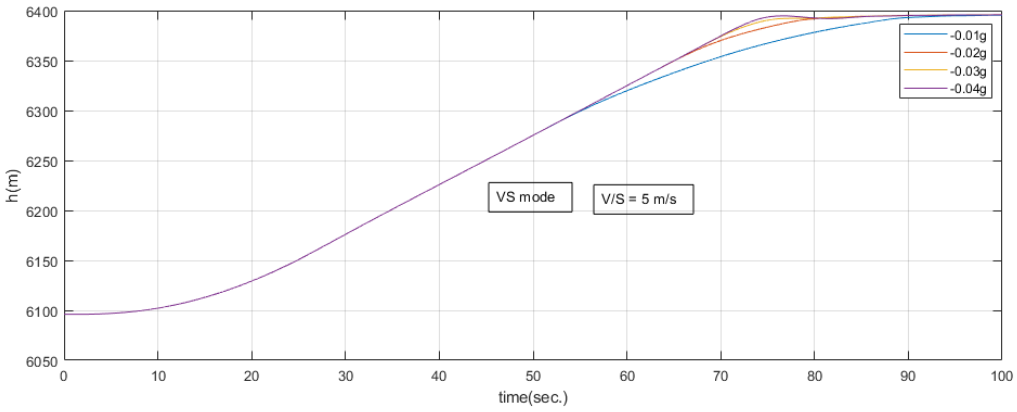

Figure 46: Altitude capture mode with normal acceleration variations at V/S = 5m/s

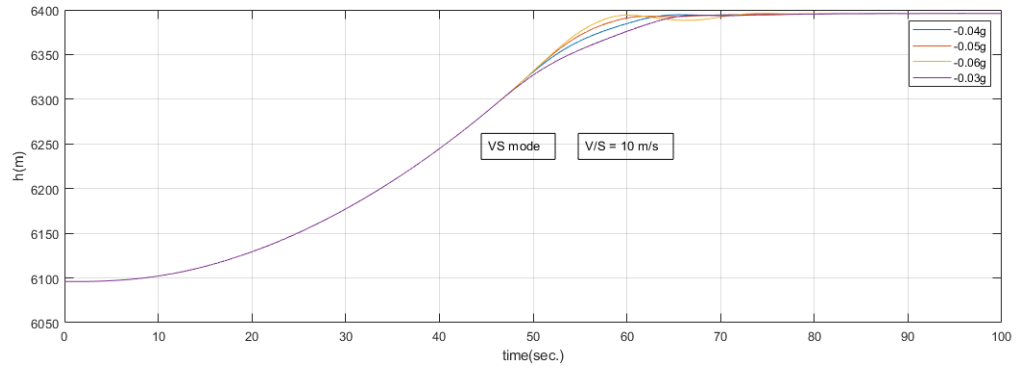

Figure 47: Altitude capture mode with normal acceleration variations at V/S = 10m/s

for a climb to $h = 7096$ m and a descent to $h = 5096$ m. A closer look at the responses, which occurred, while the ASEL mode was active, can be seen in Figures 48 and 49. Improvements to the altitude capture mode could be made. Either a different control law

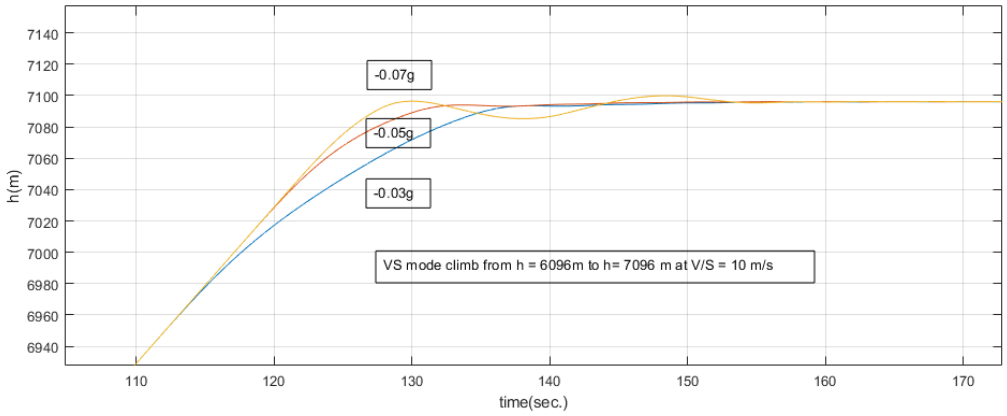

Figure 48: Altitude capture mode with normal acceleration variations at V/S = 10m/s

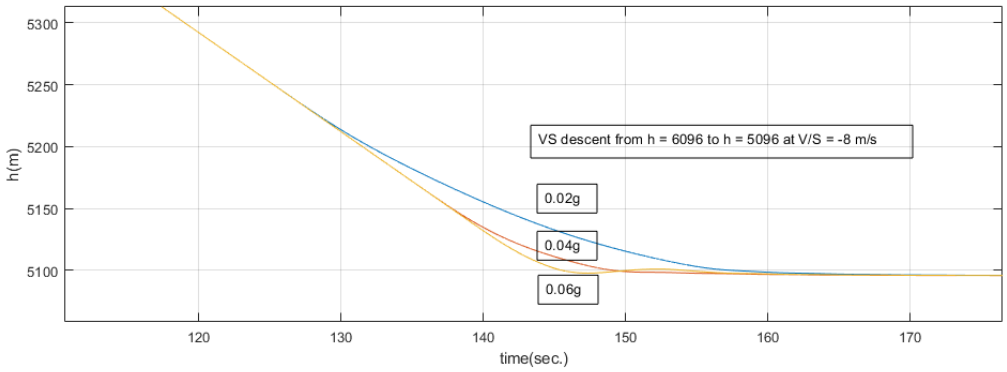

Figure 49: Altitude capture mode with normal acceleration variations at V/S = −8 m/s

could be developed or additional tuning could be done. The normal acceleration of ASEL could also be selected by the autopilot in order to achieve optimal altitude capture.

6.9 Simulink model

In this section, a range of autopilot and flight director modes were implemented and tested in Simulink. The overall structure of the project is displayed in Figure 50. Three segments can be identified. The *Logic* segment contains the input ports of the system and the Stateflow logic diagram controlling the activation of individual flight director modes through **GoTo** tags (magenta colored) and enabled subsystems. The *Flight control system* segment contains the developed autopilot and flight director modes. Each individual mode is encapsulated in an enabled subsystem (blue colored). The *Aircraft model* which was already described earlier, takes the pilot and autopilot control surface commands and outputs the measured variables of aircraft motion. Figure 50 is included just as a reference. Further details can be studied in the corresponding chapters or the Matlab/Simulink files enclosed with the thesis as an attachment.

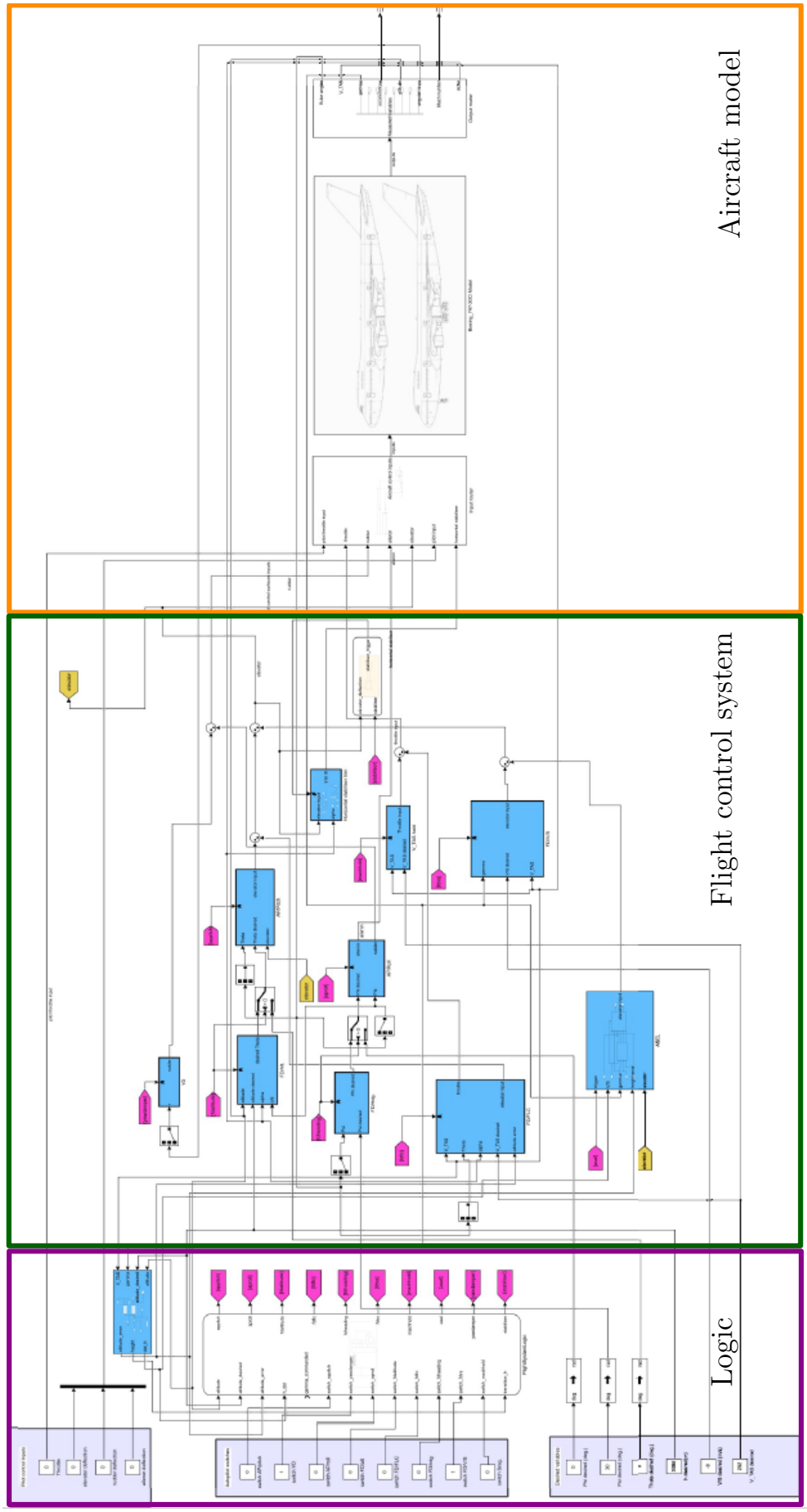

Figure 50: Final model segments

7 Path tracking and waypoint navigation

This chapter focuses on the mathematical background, algorithm development and implementation of waypoint navigation. A waypoint is a geographical location defined to specify the route or flight path of an aircraft. Waypoints can be of two types. A fly-by waypoint requires a certain degree of turn anticipation to allow the smooth tangential interception of the next segment of the route. At fly-over waypoints, the turn to join to the next segment is initiated after the waypoint is reached, not in advance. In this thesis, fly-by waypoints will be considered. The literature often focuses on lightweight UAV applications. We recommend [10], [11], [12] and [13] for a basic overview of the topic.

7.1 Mathematical background

Let us consider a series of waypoints organized in a matrix. Each row corresponds to a waypoint which is defined by a longitude, latitude and altitude

$$\mathbf{W} = \begin{bmatrix} w_{1_x} & w_{1_y} & w_{1_z} \\ w_{2_x} & w_{2_y} & w_{2_z} \\ \vdots & \vdots & \vdots \\ w_{m_x} & w_{m_y} & w_{m_z} \end{bmatrix}.$$

Generally, the reference path can be any simple curve connecting these waypoints. The desired path can be therefore defined by the parametrization

$$\mathbf{p}_d(s) = [x_d(s), y_d(s), z_d(s)]^T.$$

Our requirement on the path $\mathbf{p}_d$ is that, for every waypoint $\mathbf{W}_i$, there exists an $s^* \in \mathbb{R}$, such that $\mathbf{p}_d(s^*) = \mathbf{W}_i$. Let $\mathbf{p}$ be the actual position of the aircraft. The closest point on the desired path to the aircraft can be found by

$$d_{min} = \min_s \left(\| \mathbf{p} - \mathbf{p}_d(s) \| \right).$$

We can construct the so–called Serret–Frenet frame for the parametrized path. Vectors decomposed in this frame will be denoted by $\{.\}_{SF}$. It is assumed that the path is constructed at a constant altitude, which significantly simplifies computations. Figure 51 shows the variables involved. Note that $\mathbf{x}_s$ is the tangential unit vector and $\mathbf{y}_s$ is the unit normal vector. We can also see that the distance of the aircraft to the desired path can be computed as

$$y_s = \int_0^t \dot{y_s} \mathrm{dt} + y_0,$$

where y_0 is the initial distance from the path. The distance y_s is sometimes referred to as the aircraft's cross-track error. We can see that ψ is the flown course (or standard yaw angle) if we assume sideslip angle $\beta = 0$ and ψ_s is the trajectory direction. The relative course will be defined as

$$\tilde{\psi} = \psi - \psi_s.$$

The curvature $\kappa(s)$ binds the path parameter and path course angle rates in the following way:

$$\dot{\psi_s} = \kappa(s)\dot{s}. \tag{7.1}$$

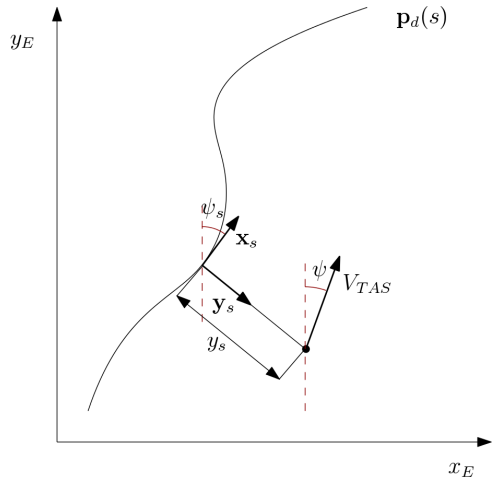

Figure 51: Serret–Frenet frame for the 2D parametrized path

The unit tangent vector $\mathbf{x}_s$ and unit normal vector $\mathbf{y}_s$ can be expressed in vector form as

$$\mathbf{x}_s = [\cos\psi_s, \sin\psi_s]^T, \quad \mathbf{y}_s = [-\sin\psi_s, \cos\psi_s]^T.$$

We can now express the Serret–Frenet formulas in 2-D along with the time dynamics as

$$\begin{aligned}\dot{\mathbf{x}}_s(s) &= \kappa \mathbf{y}_s(s),\\ \dot{\mathbf{y}}_s(s) &= -\kappa \mathbf{x}_s(s),\\ \dot{\psi}_s &= \kappa(s)\dot{s}.\end{aligned}$$

Using a 2D rotational matrix, we can express the body velocities in the Serret–Frenet frame as

$$\{\mathbf{v}\}_{SF} = \begin{bmatrix} \cos\tilde{\psi} & -\sin\tilde{\psi} \\ \sin\tilde{\psi} & \cos\tilde{\psi} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}.$$

An alternative way of expressing the body velocities would be a velocity decomposition using the original parametrization. Let the aircraft be at a distance $\{\boldsymbol{\eta}\}_{SF} = [0,\ y_s]^T$ relative to the Serret–Frenet frame. The velocity of the Serret–Frenet frame can be expressed by the derivative of the parameter s as $\{\boldsymbol{\nu}\}_{SF} = [\dot{s},\ 0]^T$. Because the point along the curve s is always the closest point of the curve to the body-fixed origin, the speed of the body origin relative to the Serret–Frenet frame can be expressed simply by $\{\boldsymbol{\mu}\}_{SF} = [0,\ \dot{y}_s]^T$. Using these formulations, we obtain

$$\{\mathbf{v}\}_{SF} = \{\boldsymbol{\nu}\}_{SF} + \{\boldsymbol{\mu}\}_{SF} + \{\boldsymbol{\omega}\}_{SF} \times \{\boldsymbol{\eta}\}_{SF}, \tag{7.2}$$

where $\{\boldsymbol{\omega}\}_{SF}$ represents the turn rate matrix of the Serret–Frenet frame. Rewriting the equation (7.2) componentwise, we obtain

$$\begin{bmatrix} \cos\tilde{\psi} & -\sin\tilde{\psi} \\ \sin\tilde{\psi} & \cos\tilde{\psi} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \dot{s} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \dot{y}_s \end{bmatrix} + \begin{bmatrix} 0 & -\dot{\psi}_s \\ \dot{\psi}_s & 0 \end{bmatrix} \begin{bmatrix} 0 \\ y_s \end{bmatrix}.$$

This expression can be further transformed using (7.1) and the simplification $\dot{\psi} = r$.

$$\begin{bmatrix} 1-\kappa y_s & 0 & 0 \\ 0 & 1 & 0 \\ \kappa & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{s} \\ \dot{y}_s \\ \dot{\tilde{\psi}} \end{bmatrix} = \begin{bmatrix} \cos\tilde{\psi} & -\sin\tilde{\psi} & 0 \\ \sin\tilde{\psi} & \cos\tilde{\psi} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ r \end{bmatrix}.$$

Because in a level turn coordinated flight $V \approx 0$ and $U \approx V_{TAS}$, we can rewrite these transformations as

$$\begin{bmatrix} \dot{s} \\ \dot{y}_s \\ \dot{\tilde{\psi}} \end{bmatrix} = \begin{bmatrix} \frac{V_{TAS}\cos\tilde{\psi}}{1-\kappa y_s} \\ V_{TAS}\sin\tilde{\psi} \\ r - \frac{V_{TAS}\cos\tilde{\psi}\kappa}{1-\kappa y_s} \end{bmatrix}.$$

For the sake of simplicity and practicality, the waypoints will be connected by straight lines. This means that $\kappa = 0$ at all segments. The Serret–Frenet transformations then simplify further to

$$\begin{bmatrix} \dot{s} \\ \dot{y}_s \\ \dot{\tilde{\psi}} \end{bmatrix} = \begin{bmatrix} V_{TAS}\cos\tilde{\psi} \\ V_{TAS}\sin\tilde{\psi} \\ r \end{bmatrix}. \tag{7.3}$$

The current required course in the xy plane is therefore

$$\chi_c = \arctan\left(\frac{\mathbf{W}_{c_y} - \mathbf{W}_{p_y}}{\mathbf{W}_{c_x} - \mathbf{W}_{p_x}}\right).$$

and the next course is similarly computed as

$$\chi_n = \arctan\left(\frac{\mathbf{W}_{n_y} - \mathbf{W}_{c_y}}{\mathbf{W}_{n_x} - \mathbf{W}_{c_x}}\right).$$

The current waypoint $\mathbf{W}_c$, previous waypoint $\mathbf{W}_p$ and next waypoint $\mathbf{W}_n$ are extracted from the waypoint matrix.

7.2 Nonlinear guidance law algorithm

The nonlinear guidance law (NLGL) is a geometric algorithm based on tracking a point on the desired path. This point is referred to as the virtual target point (VTP). The algorithm is based mainly on [13]. When dealing with a Serret–Frenet frame, the VTP would be considered as a point on the path closest to the aircraft. When using a NLGL algorithm, the VTP is a point on the path at distance L_1, where L_1 is considered to be fixed. If $d_{min} \leq L_1$, the VTP can be found by

$$s_{d_{min}} = \arg_s \left(\| \mathbf{p} - \mathbf{p}_d(s) \| \right),$$

with $\mathbf{p} = (x_E, y_E)$ being the position of the aircraft and $\mathbf{p}_d(s)$ being a point on the path dependent on parameterization s. Once the aircraft is close enough for L_1 to intercept the path, the points are found by

$$s_{VTP} = \arg_s \left(\| \| \mathbf{p} - \mathbf{p}_d(s) \| - L_1 | = 0 \right).$$

Usually, two solutions are found q and q'. To ensure that the aircraft is always moving forward, an intercept condition, $s_{VTP} > s_{d_{min}}$, chooses the right point to track between the solutions. This situation is depicted in Figure 52. Once we select the right intersection to determine the VTP, we can track its coordinates: $q' = (x_t, y_t)$. Also, the angle defined by the connecting line of the aircraft and the VTP will be $\theta_{VTP} = \text{atan2}(y_t - y_E, x_t - x_E)$.

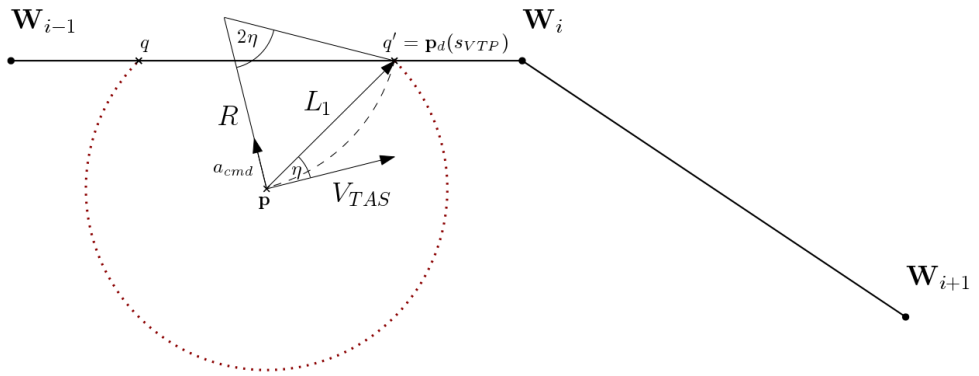

Figure 52: Geometric example of a nonlinear guidance law

If we assume that the heading of the aircraft is equal to its course so that $\beta = 0$, we can define an angle η as the angle between the velocity vector and the connecting line of the virtual target point as $\eta = \theta_{VTP} - \psi$. The simplest algorithm would take advantage of the heading select autopilot to drive the angle $\psi \to \theta_{VTP}$. According to the procedure set out in [8], a circular path to the VTP is determined at each time step and a centripetal acceleration command drives the aircraft to the desired path. The lateral acceleration command is determined from the equation

$$a_{cmd} = \frac{V_{TAS}^2}{R} = 2\frac{V_{TAS}^2}{L_1}\sin\eta.$$

From a geometric perspective, the radius L_1 and circular path radius R are tied together in a simple way.

$$L_1 = 2R\sin\eta.$$

Note that the sign of the commanded acceleration has to be signed properly in order to drive the aircraft to the desired path. During the tracking, the aircraft acceleration sign changes at some point and a smooth transition to the path is achieved. We can now express the commanded centripetal acceleration in the form of yaw rate as

$$\dot{\psi}_{cmd} = \frac{a_{cmd}}{V_{TAS}}.$$

Following the coordinated turn dynamics, we can easily approximate the commanded ϕ_{cmd} angle as

$$\phi_{cmd} = \arctan\left(\frac{a_{cmd}}{g\cos\theta}\right) = \arctan\left(2\frac{V_{TAS}^2}{L_1 g\cos\theta}\sin\eta\right).$$

The nonlinear guidance law proved to have a superior performance for both curved and straight paths. The comparison of several guidance algorithms was carried out in [10], for example. Note that the performance of such guidance algorithms is usually determined by the mean absolute error in terms of distance to the path and or criteria such as the integral of the time-multiplied absolute value of error (ITAE). The ITAE value will be denoted as Q_{ITAE}. Factors having significant effects on the Q_{ITAE} include the velocity V_{TAS} and the radius L_1. The goal is to achieve the best performance by finding an $L_{opt.}$ that enables Q_{ITAE} to be minimized.

Algorithmically, the navigational algorithm obtains the waypoint matrix $\mathbf{W}$. We can express each segment from $\mathbf{W}_i$ to $\mathbf{W}_{i+1}$ parametrically as

$$[\mathbf{p}_d(s)]_j = \mathbf{W}_i + s(\mathbf{W}_{i+1} - \mathbf{W}_i).$$

For each segment, the minimum distance to this segment is computed. The distance from the aircraft position $\mathbf{p}$ to the segment $[\mathbf{p}_d(s)]_j$ is given by the function $X(s) = \|[\mathbf{p}_d(s)]_j - \mathbf{p}\|$. Because X^2 is convex quadratic in s, we will take the derivative of X^2 and set it to zero to find the minimum distance. The parameter s which minimizes the distance is found by

$$\hat{s} = \frac{\langle \mathbf{p} - \mathbf{W}_i, \mathbf{W}_{i+1} - \mathbf{W}_i \rangle}{\|\mathbf{W}_{i+1} - \mathbf{W}_i\|^2},$$

where $\langle \mathbf{u}, \mathbf{v} \rangle$ is the dot product of $\mathbf{u}$ and $\mathbf{v}$. Because s was originally in interval $s \in [0, 1]$, we find the parameter by calculating $s^* = \min(\max(\hat{s}, 0), 1)$ and the minimum distance is $[d_{min}]_j = \|[\mathbf{p}_d(s^*)]_j - \mathbf{p}\|$. If we find the minimum among the line segments defining the route, we find the segment of interest. The only exception occurs when the circle of radius L_1 crosses the next line segment, in which case the route should be switched to the next line segment so that the aircraft makes a smooth transition among the segments effectively. This situation can be seen in the geographic plot in Figure 53 from the experimentally obtained longitude and latitude data while simulating the nonlinear Simulink model. The red lines represent the route segments connecting at $\mathbf{W}_{i+1}$. The blue line represents the aircraft trajectory as it passes the two segments. As described above, the algorithm

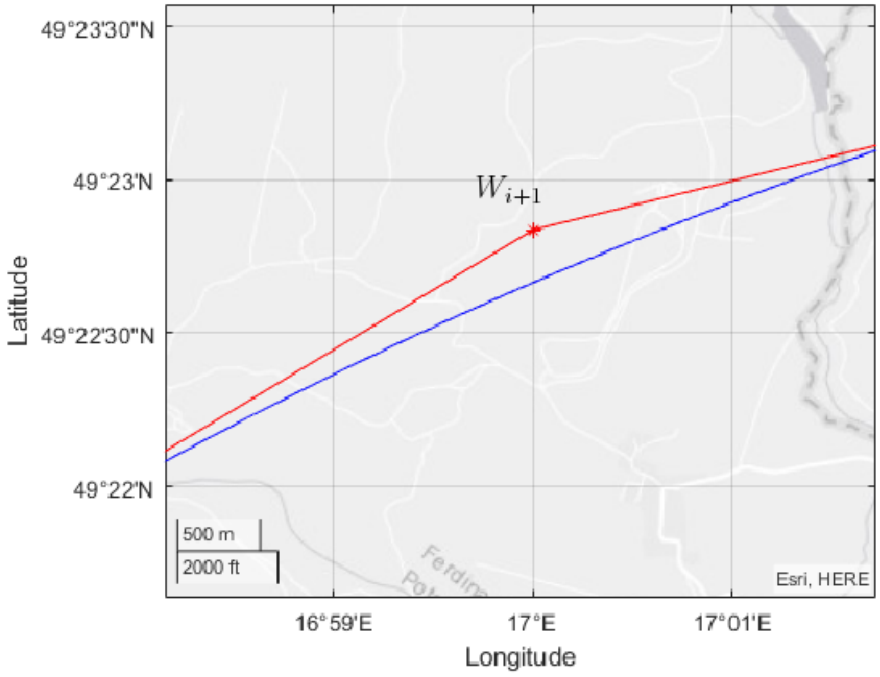

Figure 53: NLGL proximity distance segment switch

searches the intersections between a circle of radius L_1 with its origin at the aircraft's current position and the line segment. In the case that the aircraft is further than L_1, the VTP is chosen as the closest point on the route. The computations are made in Simulink through built-in Matlab functions, which are enumerated every 2 s, so the roll angle hold autopilot receives an updated command within this period. If we let the Matlab functions inherit the sample time, we may have problems with speed of the simulation. In the test seen in Figure 54, the radius parameter L_1 was changed and the response was tested for the same initial conditions. The initial heading of the aircraft was intentionally selected

so that the aircraft has to perform a significant turn to join the desired path. We can

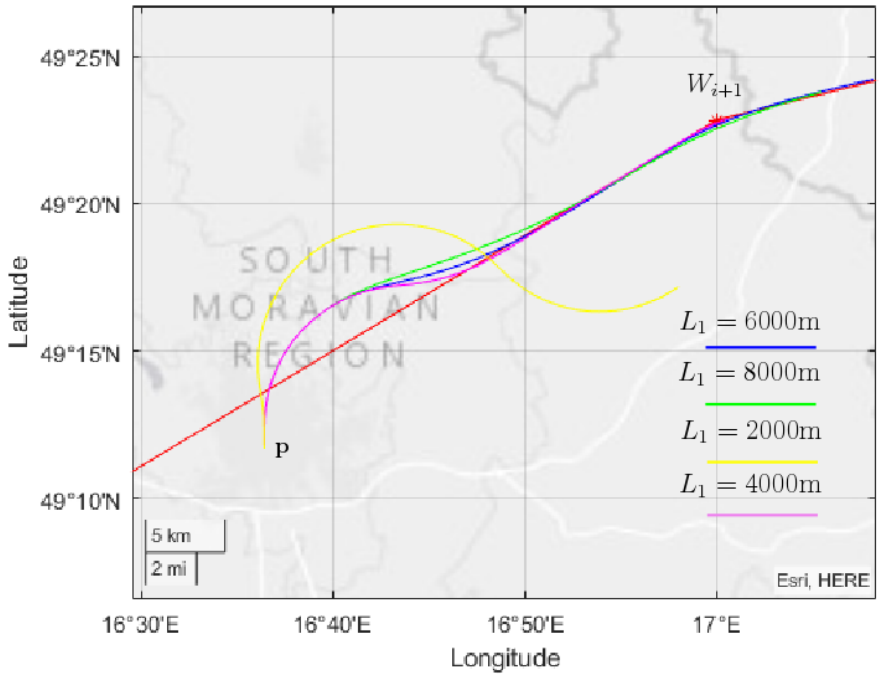

Figure 54: NLGL performance under various L_1

observe that decreasing the radius L_1 brings a performance benefit to a certain point. Due

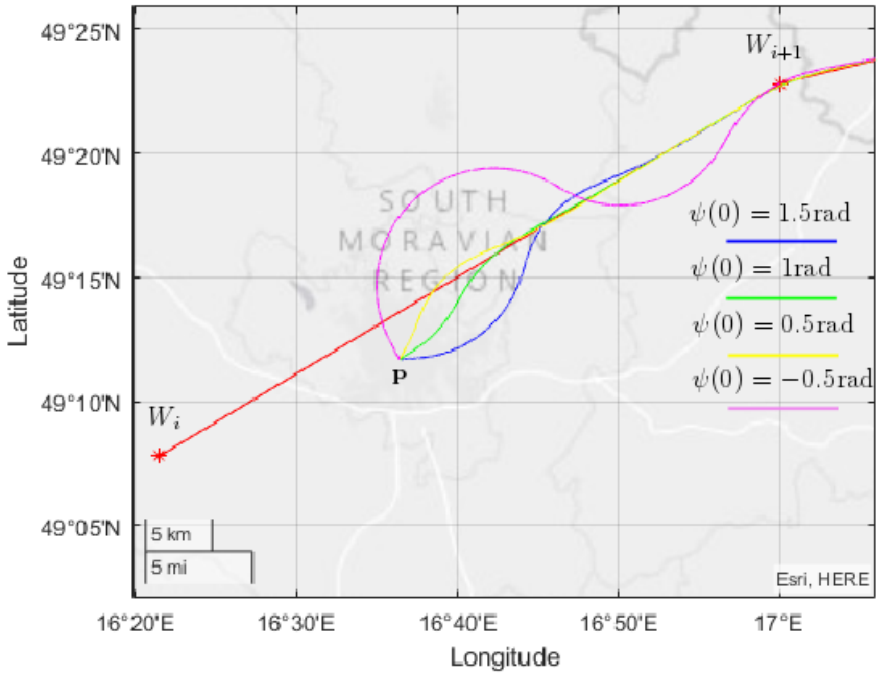

Figure 55: NLGL performance with different initial heading

to limits which were intentionally imposed on the roll angle hold autopilot, the aircraft is

not capable of performing a sharper turn. This leads to instability, as can be seen from the yellow trajectory for $L_1 = 2000$ m. No restrictions on roll angle hold will improve the performance further until approximately $L_1 \approx 1500$ m, as will be proven via a linear analysis. The behavior of the aircraft driven by an NLGL was also tested for variations in the initial heading of the aircraft. The radius was held constant at $L_1 = 4000$ m for this test. The results are presented in Figure 55. As expected, all trajectories eventually follow the path. The pink line representing the heading crossing the path at the highest angle can be seen to follow a distinct circular turn before eventually joining the path. In all the tests, the aircraft altitude was held constant at $h = 6096$ m as well as speed at $V_{TAS} = 205$ m/s. The nonlinear guidance law algorithm was successfully tested and it performed exceptionally well. The stability analysis and further measurements are discussed in Linear Analysis subsection. The guidance presented in 2D can be quite simply expanded to 3D via the following thought process. If we know the altitude difference of the two waypoints and their projected distance, we can easily command the flight path angle γ by setting a relevant vertical speed on the VS flight director. This can be automatically performed by the guidance algorithm or by the pilot. Another approach would be to extend the idea of the nonlinear guidance law to 3D by assuming a ball with radius L_1 instead of a circle. The controller would then act for η_h in the horizontal plane and angle η_v in the vertical plane. The computed commanded acceleration towards the VTP would then be decomposed to a vertical acceleration component a_{cmd_z} and a lateral acceleration component a_{cmd_y}. This

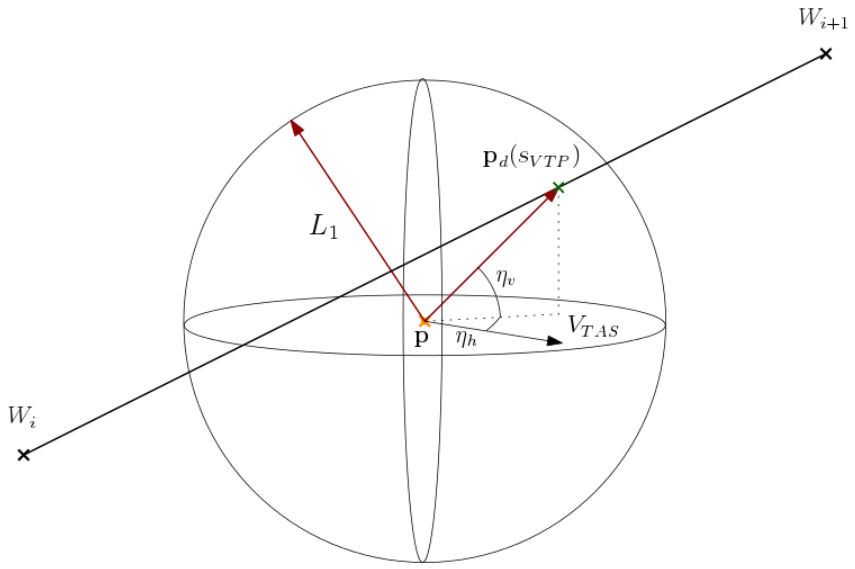

Figure 56: Geometric character of NLGL in 3D

variation of the nonlinear guidance law controller would then take advantage of both the pitch attitude hold and the roll angle hold autopilot. In practice, the waypoint coordinates can be loaded into the flight computer before take-off. It should be mentioned that the coordinates are expressed in terms of longitude and latitude and therefore must first be recomputed to flat Earth coordinates originating at the aircraft position so we can perform all the analytic geometry in the classical Cartesian coordinate system.

7.2.1 Linear analysis

A linear approximation of the navigation control law in the case of straight line tracking is worked out in this section. Surprisingly, the linearization of the control formula contains a PD controller. It utilizes a small angle approximation, i.e. $\sin\eta \approx \eta$ where η yields two components that can be approximated via cross-track error. From the geometric perspective

$$\eta = \eta_1 + \eta_2 = \frac{y_s}{L_1} + \frac{\dot{y}_s}{V_{TAS}}. \tag{7.4}$$

Once we substitute (7.4) into the guidance law, we obtain the following expression (note that the ratio of aircraft speed V_{TAS} to distance L_1 is an important factor that determines the controller gains):

$$a_{cmd} \approx 2\frac{V_{TAS}}{L_1}\left(\dot{y}_s + \frac{V_{TAS}}{L_1}y_s\right).$$

The distance L_1 can be chosen based on a stability analysis of the linearized dynamics of the aircraft and the derived linear controller. The inner loop ϕ controller should be a part of the model in order to implement the control law on top of it. For further analysis of the navigational law, let us augment system (7.3) to the lateral-directional state space representation as shown in (7.5), (again, assuming small angles, we can write $\sin\tilde{\psi} \approx \tilde{\psi}$).

$$\begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dot{\tilde{\psi}}(t) \\ \dot{y}_s(t) \end{bmatrix} = \left[\begin{array}{ccccc:cc} & & \mathbf{K} & & & \mathbf{0} & \mathbf{0} \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & (V_{TAS})_s & 0 \end{array}\right] \begin{bmatrix} \mathbf{x}(t) \\ \tilde{\psi}(t) \\ y_s(t) \end{bmatrix} + \begin{bmatrix} \mathbf{L} \\ \mathbf{0} \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix}. \tag{7.5}$$

Considering (7.5), we can simulate the development of cross-track error along with the proposed PD controller and study its response characteristics. A case study on system

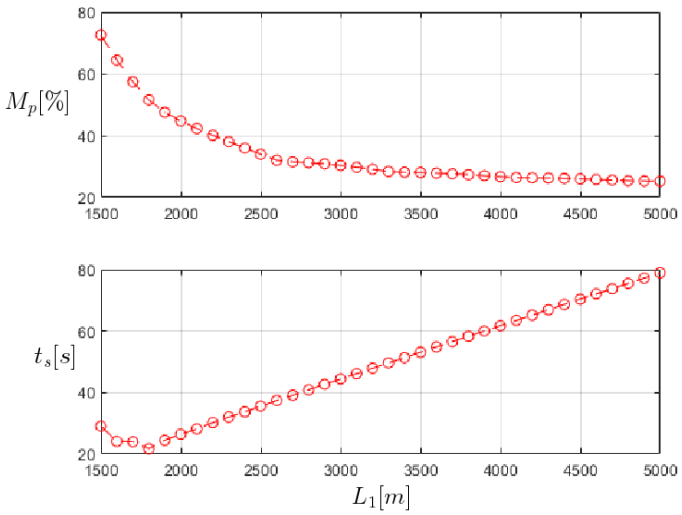

Figure 57: Time domain characteristics based on L_1 variations

(7.5) in a feedback loop featuring an inner loop ϕ controller and a linearized navigational law was performed by variations of parameter L_1. The closed loop system showed itself to be unstable for low values of L_1, and therefore impossible to use. For L_1 values that were too large, the response of the system had too large a settling time t_s. We can see the graph of overshoot M_p and settling time t_s to L_1 in Figure 57. There are certain

limitations that should be pointed out regarding this experiment. The experiment does not take yaw damper effects into account, so we cannot claim that $\beta \approx 0$ during the turn. Suppressing β decreases overshoot significantly. Another important factor which was not yet taken into account is aircraft speed. Variations in constant speed were introduced ranging in the interval $V_{TAS} \in \langle (V_{TAS})_e - 80, (V_{TAS})_e + 80 \rangle$ and radius $L_1 \in \langle 1400, 2200 \rangle$. To evaluate the overall performance of the controller, the integral of the time-multiplied absolute value of error (ITAE) criterion was used. The surface plot in Figure 58 shows the results. We can see that with increasing speed, the optimal radius parameter increases

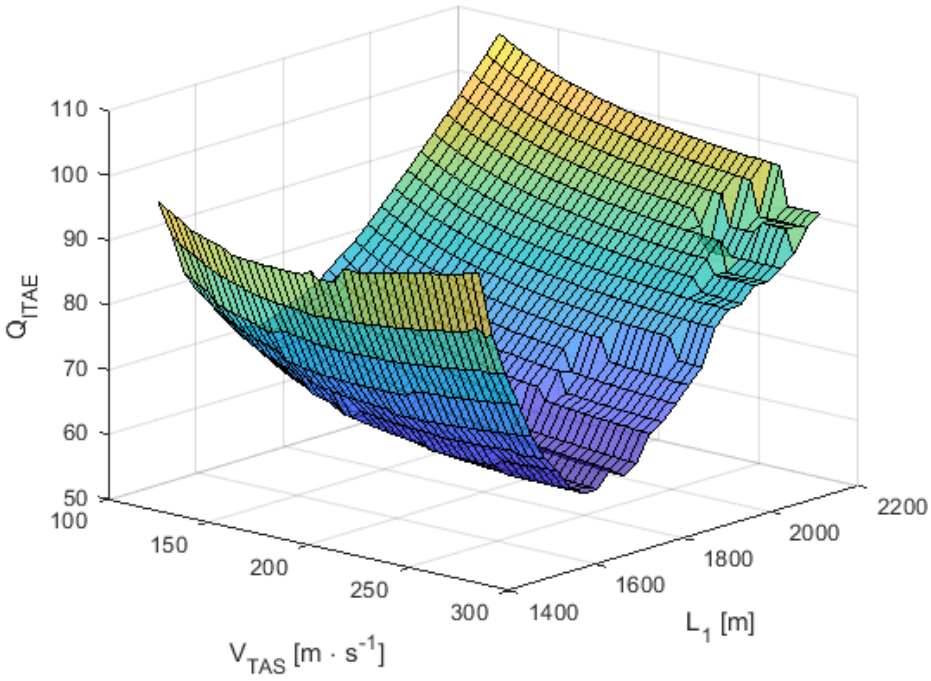

Figure 58: ITAE evaluation of the guidance law under speed and radius variations

slightly. Depending on speed, we can state that the optimal L_1 lies within the range $L_1 \in \langle 1500, 1600 \rangle$ m.

Finally, let us examine the disk margins for the guidance law controller. For $L_1 = 1500$ m we obtain $\alpha_{max} = 0.4165$ with the gain margin lying in the interval (0.65531.5260) and a phase margin of $\phi_m = \pm 23.5265$ deg. For $L_1 = 1600$ m the disk margin increases slightly to $\alpha_{max} = 0.4754$ with the gain margin interval (0.61591.6236) and phase margin $\phi_m = \pm 26.7402$. The disk margins increase further when L_1 is increased. For example, with $L_1 = 3000$ m the disk margins yields as value of $\alpha_{max} = 0.8788$ or $\alpha_{max} = 0.9606$ for $L_1 = 4000$ m. It is therefore a difficult task to choose the radius parameter as we must trade performance for robustness.

8 Finite state machine modeling

A system defined by a finite number of states which are reachable and a set of transition conditions defining the transitions of those states is called a finite state machine. It can be used to model the sequential logic of complicated algorithms that can be later implemented in hardware or software. It has many other applications in mathematics, artificial intelligence games and linguistics. In this thesis, the Stateflow finite state modeling language was used inside a Simulink environment to model automatic flight control logic. For a small number of controllers, the logic behind transitions and activations over them can be easily achieved inside Simulink through **Switch** blocks, but this approach can become messy and cumbersome very quickly if more transitions are needed. Luckily, Stateflow diagrams can be implemented directly into Simulink to complement dynamical system models. First, let us recall some of the basic theory behind finite state machines before diving into Stateflow. More insight to the theory can be found in [15].

8.1 Theory of finite state machines

The formal definition of a deterministic finite automaton (DFA) or finite state machine is described by a five-element tuple $\mathrm{A} = (\mathrm{S}, \sum, \delta, \mathrm{s}_0, \mathrm{F})$, where S is a set of states, $\sum$ is a finite non-empty input alphabet, $\delta : \mathrm{S} \times \sum \to \mathrm{S}$ is a transition function which defines a state transition whenever an input from the alphabet is present, $\underline{s_0} \in \mathrm{S}$ is an initial state and $\mathrm{F} \subseteq \mathrm{S}$ is a set of admissible states.

Example 8.1.1. Consider an automaton $\mathrm{A} = (\mathrm{S}, \sum, \delta, \mathrm{s}_0, \mathrm{F})$ where $\mathrm{S} = \{\mathrm{s}_0, \mathrm{s}_1, \mathrm{s}_2, \mathrm{s}_3\}$ and $\sum = \{a, b\}$ with a transition function The state diagram of the following finite state

δ	a	b
$\underline{s_0}$	s_1	s_2
$\underline{s_1}$	s_1	s_2
$\underline{s_2}$	s_3	s_2
s_3	s_3	s_3

machine can be represented by a diagram, as in Figure 59.

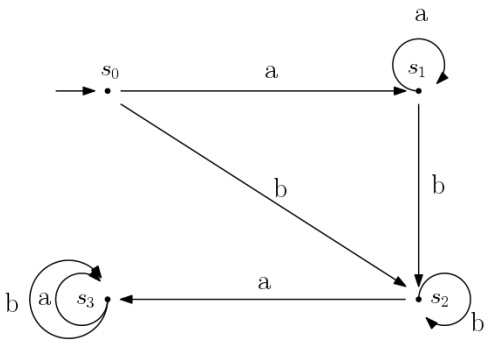

Figure 59: Example 8.1.1. state diagram

Let us introduce several special types of (DFA), the Mealy and Moore machines, which can be implemented in Stateflow and whose behavior can be formally verified. A **Mealy machine** is a finite state machine whose output is defined by both the current state as

well as by current inputs. For each state and input, one transition is possible at most. A formal definition of a Mealy machine is a six-element tuple $\mathrm{M} = (\mathrm{S}, \sum, \delta, \mathrm{s}_0, \Lambda, \mathrm{G})$. Symbols S, $\sum$, δ, s_0 have the same meaning as in simple (DFAs). In addition, Λ is a set called an output alphabet and $\mathrm{G} : \mathrm{S} \times \sum \to \Lambda$ is a mapping called an output function.

Example 8.1.2. Consider a Mealy machine $\mathrm{M} = (\mathrm{S}, \sum, \delta, \mathrm{s}_0, \Lambda, \mathrm{G})$, where $\mathrm{S} = \{\mathrm{s}_0, \mathrm{s}_1, \mathrm{s}_2, \mathrm{s}_3\}$, $\sum = \{a, b\}$ and $\Lambda = \{x_1, x_2, x_3\}$ with a transition function and output function defined by G. The state diagram of this Mealy machine is presented in Figure 60. Note that the

δ,G	a		b	
s_0	s_1	x_1	s_2	x_1
s_1	s_1	x_2	s_3	x_3
s_2	s_3	x_3	s_2	x_1
s_3	s_3	x_3	s_3	x_2

transition alphabet $\sum$ also defines an action or output in curly brackets.

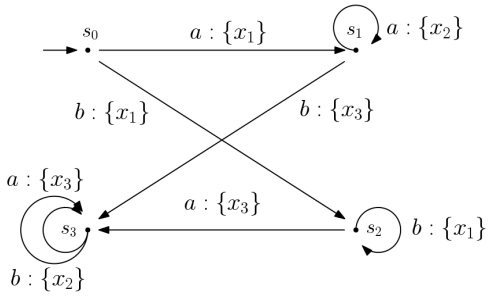

Figure 60: Example 8.1.2. state diagram

Finally a **Moore machine** is a finite state machine, whose output values are determined only by the current state. The formal definition is very similar to that of the Mealy machine. We have a six-element tuple $\mathrm{m} = (\mathrm{S}, \sum, \delta, \mathrm{s}_0, \Lambda, \mathrm{G})$. The only difference is in the mapping or output function, which is by definition $\mathrm{G} : \mathrm{S} \to \Lambda$.

Example 8.1.3. Example 8.1.1 can be easily extended by extending its table. A state

δ	a	b	G
s_0	s_1	s_2	x_2
s_1	s_1	s_2	x_1
s_2	s_3	s_2	x_2
s_3	s_3	s_3	x_3

diagram would be analogous to that of Example 8.1.1, with the only difference being that the output is mapped to each state.

8.2 Stateflow modeling

Stateflow is a graphical programming environment based on finite state machines. It can implement both Mealy and Moore machines as well as more general finite state machines that provide other possibilities for modeling. The basic Stateflow chart is formed by a

finite number of states and transitions. A **state** is the simplest building block of a Stateflow chart. It is labelled by three optional types of action; the *entry* action is executed after the state becomes active; the *during* action evaluates once the state is active and no valid outgoing transitions are enabled; and the *exit* action executes after the state has been exited. States can be hierarchical, so the number of OR or AND states can be nested within one superstate. A **transition** connects a source state to a destination state while defining conditions which have to be satisfied to perform that transition. A transition label has the general form of $E[C]\{cA\}/tA$, where E is an event, C is a condition, cA is a condition action and tA is a transition action. A simple example from the *FlightSystemLogic* chart built on top of the Simulink aircraft simulation environment is presented in Figure 61. A simple Boolean condition engages pitch attitude hold through the switch_appitch variable. A special kind of transition is the default transition but

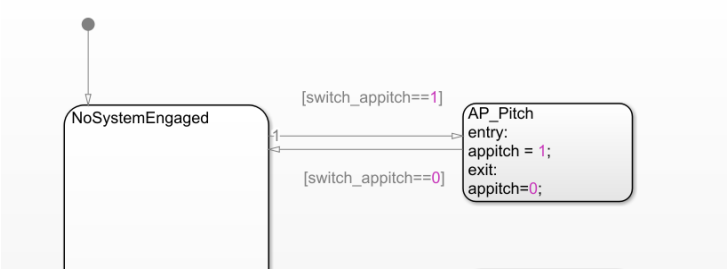

Figure 61: Stateflow pitch attitude hold state

with no source state or specified default target state. Every chart has to have at least one default transition. In a Stateflow chart, one can define **input**, **output**, **local**, **constant** and **parameter** data; switch_appitch is input data and appitch is output data in this case. The controllers themselves are engaged through Enabled subsystems in Simulink diagrams by such Boolean output variables.

Flowcharts are another important construct in Stateflow. A **flowchart** is a composition of connective junctions and transitions. It can be used to construct a decision logic and

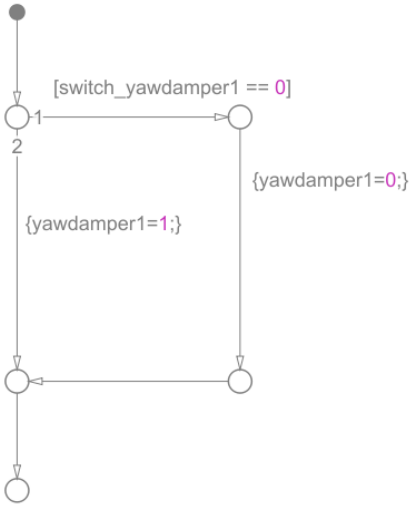

Figure 62: Stateflow simple if-else flowchart segment

form networks as well as to create modular reusable decision and loop logic segments. The

simplest example of a flowchart is presented in Figure 62. It is a simple segment checking if a yaw damper has been manually deactivated. The numbers near the junctions define the order of transition evaluation. Let us point out one of the important applications of Stateflow in this project. While developing the climb modes, the need for conditioned switching based on altitude error emerged. The solution in Simulink seemed to be overly complicated and Stateflow appears to be an ideal solution. Figure 63 shows the implementation of the climb mode states. Both of the climb modes VS and FLC are linked to ASEL and the transition is performed only if the absolute value of altitude difference is lower or equal to the desired transition altitude. ASEL is further linked to the altitude hold flight director at the condition that the aircraft is close enough to the desired altitude. Stateflow has many other capabilities exceeding those of finite state machines. One

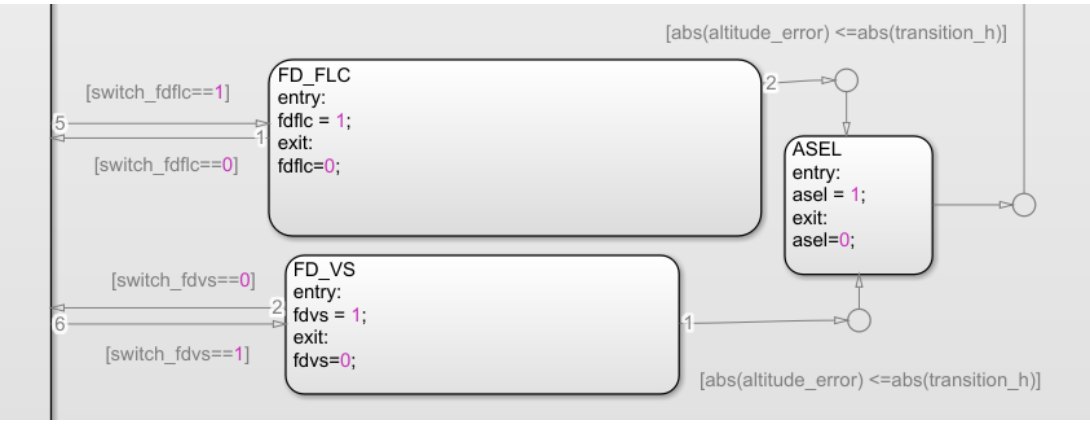

Figure 63: Altitude capture mode state transitions

can implement graphical functions, Matlab functions, truth tables, event broadcasting and much more. The *FlightSystemLogic* chart implemented in this thesis is printed in Figure 64. We can see several hierarchical states in the diagram. The default state that is entered after the simulation start is called a *FlighSystemLogic_TakeOff_Approach*. It is designated for operations at low altitudes and approach control systems. The yaw damper is turned off in this mode. The mode is mostly empty since no low altitude conditions were assumed in this thesis. When satisfying the minimum altitude condition, the state transitions to *FlighSystemLogic_Cruise*. This is the main superstate which is active during the simulation. Based on pilot inputs via the flight computer, the relevant autopilot or flight director modes become active.

The Stateflow modeling language helped to organize the logic in a clear structured way. This project is still not sufficiently extensive to make use of Stateflow's full potential. It is nevertheless a good tool to learn and is often passed over as an option.

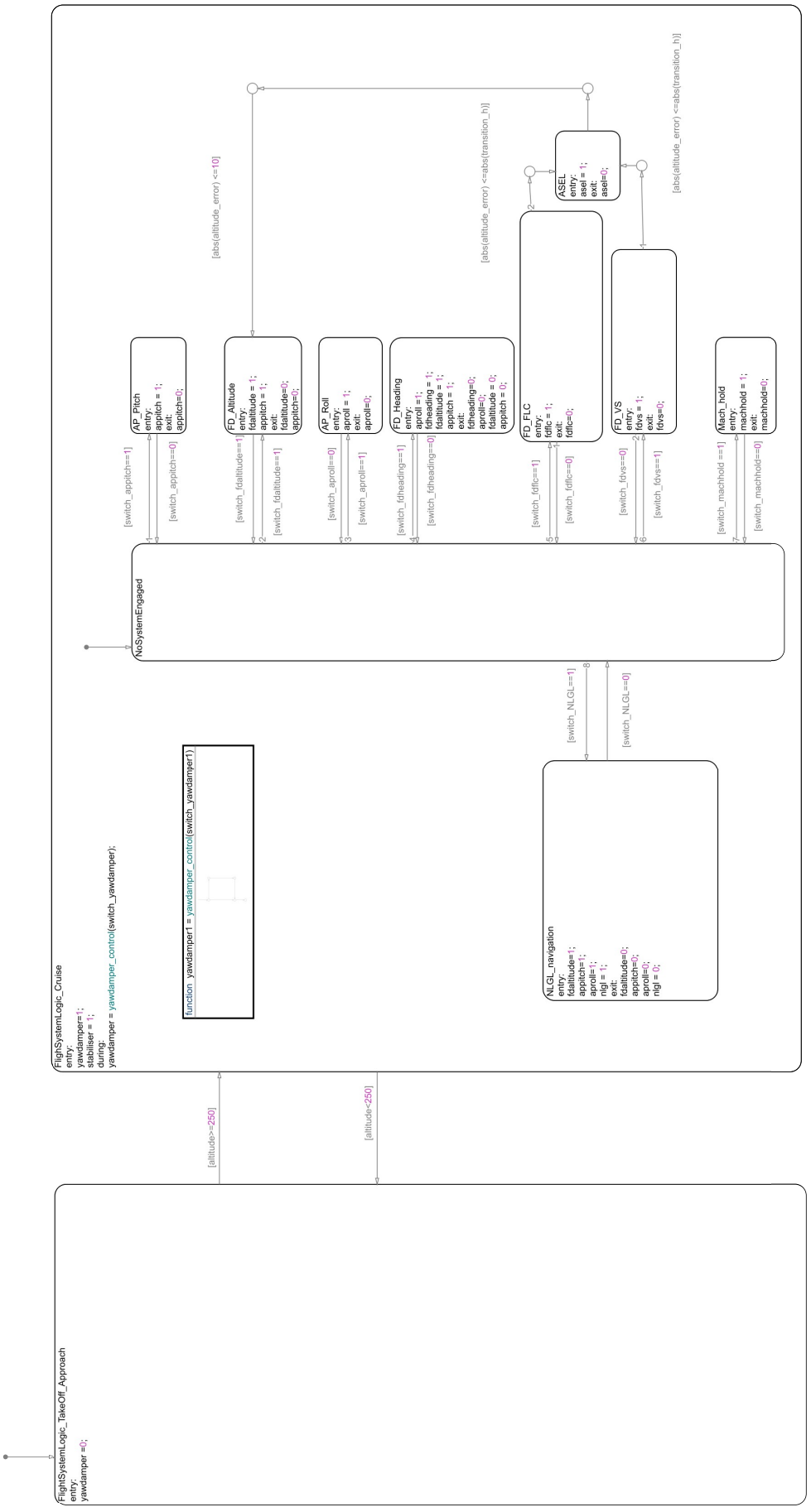

Figure 64: Main logic state diagram

9 FlightGear Flight Simulator

FlightGear is an open-source flight simulator. It is supported by Aerospace Blockset, which allows a real-time simulation to be presented in a graphical way. FlightGear supports a variety of aircraft designs and allows settings for control surface movements. Given a longitude, latitude, altitude and Euler angles, animation can be accomplished by running FlightGear from Matlab. FlightGear has been used in many projects in academia and industry (as well as NASA) because of its variability and open-source character. Even though not critical for scientific purposes, its embodiment of flight simulation provides great aesthetic added value and is a nice way to present results. A good review of the software can be found in, e.g. [17]. It is not so straightforward to connect Simu-

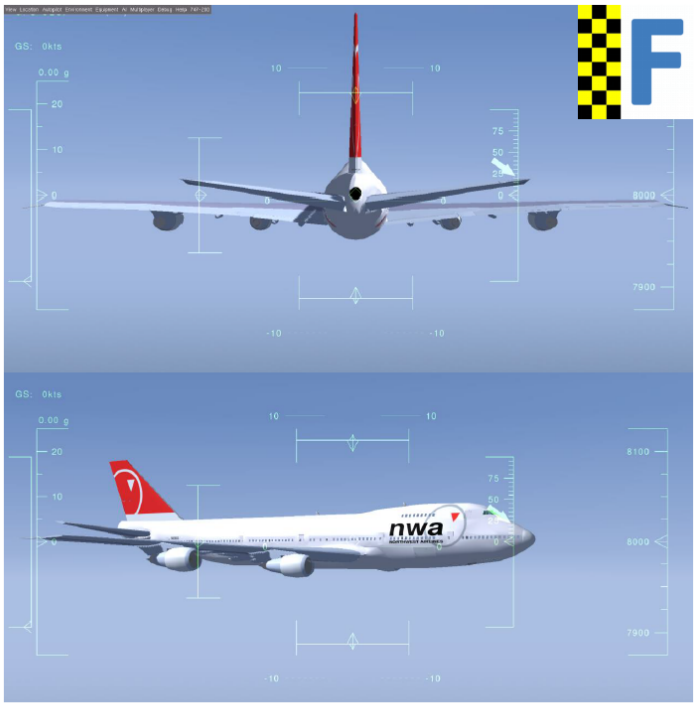

Figure 65: FlightGear simulator animation

link and FlightGear. Apart from the Aerospace blockset animation block, which parses data to FlightGear, the application should be set up internally. This is accomplished by creating a **connect.bat** file. Apart from *FG_ROOT* and *FG_SCENERY* specification, initial conditions including visibility and, location (of the airport if on the ground) can be configured.

```
1  C:
2  cd C:\Program Files\FlightGear 2017.2.1
3  SET FG_ROOT=C:\Program Files\FlightGear 2017.2.1\data
4  SET FG_SCENERY=C:\Program Files\FlightGear 2017.2.1\data\Scenery
5  .\\bin\fgfs --aircraft=747-200 --fdm=null --enable-auto-coordina
```

Figure 66: connect.bat file

10 Conclusions

The aim of this work was to create a simulation environment in Matlab/Simulink for a selected aircraft at a specific flight condition and to develop a set of autopilots for this condition while evaluating their performance. Ideally, the workflow of this project and the algorithms used can be expanded to other flight conditions or other aircraft geometries in order to present a complete flight model. Collaboration with Honeywell international, s.r.o. and its aerospace division led to interesting results and the developed model could later be used as a basis for other scientific works and possibly as the foundation for an upcoming thesis led by Honeywell.

A simulation of Boeing 747-200 dynamics was developed and animated by the FlightGear flight simulator. Pitch attitude hold and roll angle hold autopilots were developed using PID controllers and tuned by the state of the art multiobjective optimization method. The particle swarm optimization algorithm was used to find optimal PID gains that minimize a suitable objective function. The weights of this objective function were computed using Pareto front solutions of the problem. In addition to these autopilots, flight director modes such as altitude hold and heading select were developed. These take the form of the so-called cascaded loops. Other important flight director modes were developed, including vertical speed mode, flight level change mode, altitude capture mode and navigational modes allowing the aircraft to track a series of waypoints. For the extensive testing and tuning of the developed flight control systems, state-space representations of the decoupled longitudinal and lateral–directional linearizations of the selected flight condition were used. Characteristics such as damping ratio, stability margins, overshoot and load factor were measured for the autopilots. The cooperation of several autopilots usually poses many problems. The transitions between the controllers must take place seamlessly so that they not cause a sudden change in control inputs that leads to instability. With the help of the Stateflow programming environment, the logic behind state transitions was developed. Each flight director mode represents a state in a logic diagram which can only be activated under certain conditions. A good example of such transitions is an automatic climb using the vertical speed mode. The transition to altitude capture mode and subsequently to altitude hold mode should be automatically initiated when approaching the desired altitude.

Additionally, the topic of navigation was studied. Even though only straight paths connecting the so-called waypoints were considered in the simulation, the Serret-Frenet transformations were mathematically derived for more general paths in 2D. With the use of the developed climb modes, the nonlinear guidance law algorithm adopted from works considering UAVs was tuned and tested in simulations for the 2D case and a solution to the 3D case was proposed. Linear analysis of the guidance law yielded a proportional and derivative controller so that stability could be easily shown with the inner loop roll angle hold autopilot.

The work contains some mathematical background related to dynamical systems as well as some mathematical tools used in various related areas such as control theory or simulations. Future work could expand the model to include a broader range of flight conditions so as to cover the whole flight envelope of the aircraft. The gain scheduling strategy could later be adopted in order to develop a fully capable controller. As an alternative, linear quadratic regulators and nonlinear dynamic inversion designs are popular today and undoubtedly have a future in control systems and the aerospace industry.

Reference

[1] ROSKAM, J.: *Airplane Flight Dynamics and Automatic Flight Controls*, Lawrence: DAR Corporation, 2003.

[2] STEVENS, B. L., LEWIS, F. L., *Aircraft Control and Simulation, 2nd ed*, Wiley-Interscience, 2003. ISBN 978-0471371458.

[3] COOK, M. V.: *Flight Dynamics Principles*, Arnold, London 1997.

[4] AGOSTINO, De M., EUGENE, L. D., BERNDT, J.S.: *A general solution to the aircraft trim problem*, AIAA Modeling and Simulation Technologies Conference (MST), 2007.

[5] NELSON, R. C.: *Flight Stability and Automatic Control* , McGraw-Hill, 1989.

[6] MCLEAN, D.: *Automatic Flight Control Systems* , Prentice Hall International Series in Systems and Control Engineering, 1990.

[7] SAHIB, M.A., BESTOUN, S.A.: *A new multiobjective performance criterion used in PID tuning optimization algorithms,* Journal of Advanced Research [online]. Elsevier B.V, 2016, 7(1), 125-134 [cit. 2020-02-29]. DOI: 10.1016/j.jare.2015.03.004. ISSN 2090-1232.

[8] LU, Peng, Erik-jan VAN KAMPEN, Cornelis DE VISSER, Qiping CHU: *Aircraft fault-tolerant trajectory control using Incremental Nonlinear Dynamic Inversion,* Control Engineering Practice [online]. Elsevier, 2016, 57, 126-141 [cit. 2020-02-29]. DOI: 10.1016/j.conengprac.2016.09.010. ISSN 0967-0661.

[9] SADRAEY, M..: *Automatic Flight Control Systems* , Morgan & Claypool, 2020. ISBN: 9781681737300

[10] SUJIT, P. B, Srikanth SARIPALLI and J. B SOUSA.: *An evaluation of UAV path following algorithms.* In: 2013 European Control Conference (ECC) [online]. EUCA, 2013, s. 3332-3337 [cit. 2020-04-08]. DOI: 10.23919/ECC.2013.6669680.

[11] RUBÍ, B., A. RUIZ, R. PERÉZ and B. MORCEGO.: *Path-Flyer: A Benchmark of Quadrotor Path Following Algorithms.* , In: IEEE International Conference on Control and Automation, ICCA [online]. IEEE Computer Society, 2019, 2019-, s. 633-638 [cit. 2020-04-27]. DOI: 10.1109/ICCA.2019.8899563. ISBN 9781728111643. ISSN 19483449.

[12] RYSDYK R..: *UAV path following for constant line-of-sight.* , University of Washington, Seattle (2003), 10.2514/6.2003-6626.

[13] PARK J., SANGHYUK H., Deyst J.: *A New Nonlinear Guidance Logic for Trajectory Tracking.* , (2004) 10.2514/6.2004-4900.

[14] SEILER, P., PACKARD, A., GAHINET, P.: *An Introduction to Disk Margins.* ArXiv.org [online]. Ithaca: Cornell University Library, arXiv.org, 2020 [cit. 2020-04-24]. Available from: http://search.proquest.com/docview/2376046874/

[15] ZHAN, N., WANG, S., ZHAO, H.: *Formal Verification of Simulink/Stateflow Diagrams: A Deductive Approach.* Cham: Springer International Publishing, 2017. DOI: 10.1007/978-3-319-47016-0. ISBN 9783319470146.

[16] MATHWORKS.: *YAW DAMPER for a 747 aircraft*, [online], 2020, [cit. 2020-06-24]. Dostupné z: https://www.mathworks.com/help/control/examples/yaw-damper-design-for-a-747-jet-aircraft.html

[17] ANONYMOUS.: *FlightGear*, In: Linux Format [online]. Bath: Future Publishing, 2016, (214), 56 [cit. 2020-04-19]. ISSN 14704234. Dostupné z: http://search.proquest.com/docview/1810277164/

[18] NOVÁK, J.: *Stabilita a řízení dynamických systémů použitých při modelování pohybu letadla*, Bakalářská práce, Fakulta strojního inženýrství, Vysoké učení technické v Brně, Brno, 2018.

11 Appendix A

The chosen aircraft is a typical large four-jet engine commercial transport aircraft, the Boeing 747-200. All relevant data are provided in the table below. All of the stability derivatives are dimensionless and referenced to the stability axes.

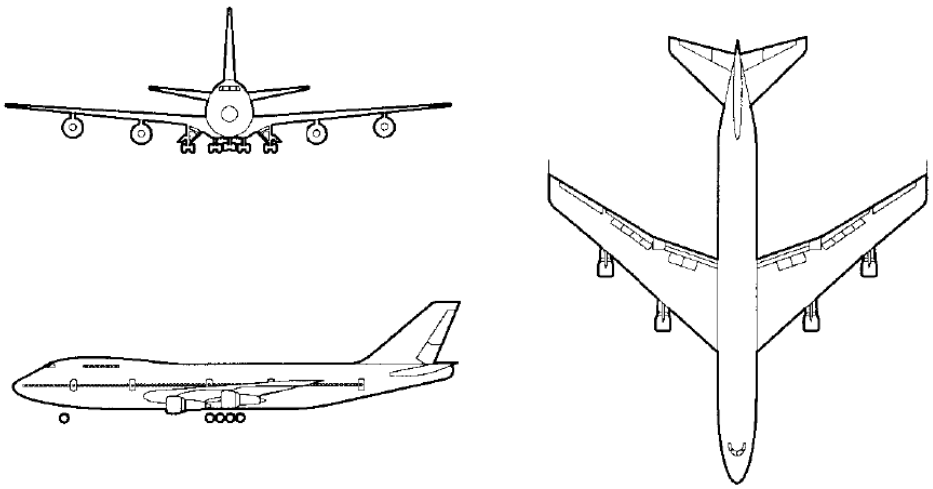

Figure 67: Boeing 747-200

Table 4: Reference Geometry & Mass Data

$S[m^2]$	510,96
$\bar{c}[m]$	8,32
$b[m]$	59,74
$m[kg]$	288773,23
$I_{xx}[kg \cdot m^2]$	24675886,69
$I_{yy}[kg \cdot m^2]$	44877574,145
$I_{zz}[kg \cdot m^2]$	67384152,115
$I_{xz}[kg \cdot m^2]$	1315143,4115

Table 5: Flight Condition Data & Steady State Coefficients

$h[m]$	6096
$Mach$	0,650
$(V_{TAS})_s[m/s]$	205,13
$\bar{q}[N/m^2]$	13888
$C.G.$	0,25
$(\alpha)_s[rad.]$	0,043633
C_{L_1}	0,40
C_{D_1}	0,0250
$C_{T_{x_1}}$	0,0250

Table 6: Longitudinal Coefficients and Stability Derivatrives

C_{D_0}	C_{D_u}	C_{D_α}	$C_{T_{x_0}}$	C_{L_0}	C_{L_u}	C_{L_α}	$C_{L_{\dot{\alpha}}}$	C_{L_q}	C_{m_0}	C_{m_u}	C_{m_α}	$C_{m_{\dot{\alpha}}}$	C_{m_q}
0,0164	0	0,20	-0,055	0,21	0,13	4,4	7,0	6,6	0	0,013	-1,0	-4,0	-20,5

Table 7: Lateral-Directional Stability Derivatives

C_{l_β}	C_{l_p}	C_{l_r}	C_{y_β}	C_{y_p}	C_{y_r}	C_{n_β}	$C_{n_{T_\beta}}$	C_{n_p}	C_{n_r}
0,0164	0	0,20	-0,055	0,21	0,13	4,4	7,0	6,6	0

Table 8: Control and Hinge Moment Derivatives

$C_{D_{\delta_e}}$	$C_{L_{\delta_e}}$	$C_{m_{\delta_e}}$	$C_{D_{i_h}}$	$C_{L_{i_h}}$	$C_{m_{i_h}}$	$C_{l_{\delta_a}}$	$C_{l_{\delta_r}}$	$C_{y_{\delta_a}}$	$C_{y_{\delta_r}}$	$C_{n_{\delta_a}}$	$C_{n_{\delta_r}}$
0	0,32	-1,30	0	0,70	-2,7	0,013	0,008	0	0,120	0,0018	-0,100

12 Appendix B

Each dimensional derivative represents either the linear or angular acceleration imparted to the aircraft as a result of a unit change in it's associated motion or control variable.

Table 9: Longitudinal Dimensional Stability Derivatives

$X_u = \frac{-\bar{q}S(C_{D_u}+2C_{D_1})}{m(V_0)_e}$	$Z_u = \frac{-\bar{q}S(C_{L_u}+2C_{L_1})}{m(V_0)_e}$	$Z_{\delta_e} = \frac{-\bar{q}SC_{L_{\delta_e}}}{m}$
$X_{T_u} = \frac{\bar{q}S(C_{T_{x_u}}+2C_{T_{x_1}})}{m(V_0)_e}$	$Z_\alpha = \frac{-\bar{q}S(C_{L_\alpha}+C_{D_1})}{m}$	$M_u = \frac{\bar{q}S\bar{c}(C_{m_u}+2C_{m_1})}{I_{yy}(V_0)_e}$
$X_\alpha = \frac{-\bar{q}S(C_{D_\alpha}-C_{L_1})}{m}$	$Z_{\dot{\alpha}} = \frac{-\bar{q}S\bar{c}C_{L_{\dot{\alpha}}}}{2m(V_0)_e}$	$M_{T_u} = \frac{\bar{q}S\bar{c}(C_{m_{T_u}}+2C_{m_{T_1}})}{I_{yy}(V_0)_e}$
$X_{\delta_e} = \frac{-\bar{q}SC_{D_{\delta_e}}}{m}$	$Z_q = \frac{-\bar{q}S\bar{c}C_{L_q}}{2m(V_0)_e}$	$M_\alpha = \frac{\bar{q}S\bar{c}C_{m_\alpha}}{I_{yy}}$
$M_{\dot{\alpha}} = \frac{\bar{q}S\bar{c}^2C_{m_{\dot{\alpha}}}}{2I_{yy}(V_0)_e}$	$M_q = \frac{\bar{q}S\bar{c}^2C_{m_q}}{2I_{yy}(V_0)_e}$	$M_{T_\alpha} = \frac{\bar{q}S\bar{c}C_{m_{T_\alpha}}}{I_{yy}}$
$M_{\delta_e} = \frac{\bar{q}S\bar{c}C_{m_{\delta_e}}}{I_{yy}}$		

Table 10: Lateral-Directional Dimensional Stability Derivatives

$Y_\beta = \frac{\bar{q}SC_{y_\beta}}{m}$	$L_\beta = \frac{\bar{q}SbC_{l_\beta}}{I_{xx}}$	$N_\beta = \frac{\bar{q}SbC_{n_\beta}}{I_{zz}}$
$Y_p = \frac{\bar{q}SbC_{y_p}}{2m(V_0)_e}$	$L_p = \frac{\bar{q}Sb^2C_{l_p}}{2I_{xx}(V_0)_e}$	$N_{T_\beta} = \frac{\bar{q}SbC_{n_{T_\beta}}}{I_{zz}}$
$Y_r = \frac{\bar{q}SbC_{y_r}}{2m(V_0)_e}$	$L_r = \frac{\bar{q}Sb^2C_{l_r}}{2I_{xx}(V_0)_e}$	$N_p = \frac{\bar{q}Sb^2C_{n_p}}{2I_{zz}(V_0)_e}$
$Y_{\delta_a} = \frac{\bar{q}SC_{y_{\delta_a}}}{m}$	$L_{\delta_a} = \frac{\bar{q}SbC_{l_{\delta_a}}}{I_{xx}}$	$N_r = \frac{\bar{q}Sb^2C_{n_r}}{2I_{zz}(V_0)_e}$
$Y_{\delta_r} = \frac{\bar{q}SC_{y_{\delta_r}}}{m}$	$L_{\delta_r} = \frac{\bar{q}SbC_{l_{\delta_r}}}{I_{xx}}$	$N_{\delta_a} = \frac{\bar{q}SbC_{n_{\delta_a}}}{I_{zz}}$
$N_{\delta_r} = \frac{\bar{q}SbC_{n_{\delta_r}}}{I_{zz}}$		

13 Appendix C

This section concerns the algebraic derivation of the state space form along with its limitation. Knowledge of the translational, rotational and kinematic equations set out in Chapter 1 is required. In perturbed state flight, all motion variables are defined relative to a known steady state flight condition. The derivation was largely taken from [1]. Let us make the following substitutions referring to steady–state and perturbed variables. Substituting into the general translational equations leads to

$$\begin{array}{ccc}
U = U_s + u & V = V_s + v & W = W_s + w \\
P = P_s + p & Q = Q_s + q & R = R_s + r \\
\Phi = \phi_s + \phi & \Theta = \theta_s + \theta & \Psi = \psi_s + \psi \\
X_A = (X_A)_s + x_A & Y_A = (Y_A)_s + y_A & Z_A = (Z_A)_s + z_A \\
X_T = (X_T)_s + x_T & Y_T = (Y_T)_s + y_T & Z_T = (Z_T)_s + z_T \\
\mathcal{L}_A = (\mathcal{L}_A)_s + l_A & \mathcal{M}_A = (\mathcal{M}_A)_s + m_A & \mathcal{N}_A = (\mathcal{N}_A)_s + n_A \\
\mathcal{L}_T = (\mathcal{L}_T)_s + l_T & \mathcal{M}_T = (\mathcal{M}_T)_s + m_T & \mathcal{N}_T = (\mathcal{N}_T)_s + n_T
\end{array}$$

$$\begin{aligned}
\dot{u} &= \frac{1}{m}((X_A)_s + x_A + (X_T)_s + x_T) - (W_s + w)(Q_s + q) + (V_s + v)(R_s + r) - g\sin(\theta_s + \theta), \\
\dot{v} &= \frac{1}{m}((Y_A)_s + y_A + (Y_T)_s + y_T) - (U_s + u)(R_s + r) + (W_s + w)(P_s + p) + g\cos(\theta_s + \theta)\sin(\phi_s + \phi), \\
\dot{w} &= \frac{1}{m}((Z_A)_s + z_A + (Z_T)_s + z_T) - (V_s + v)(P_s + p) + (U_s + u)(Q_s + q) + g\cos(\theta_s + \theta)\cos(\phi_s + \phi).
\end{aligned}$$

The same applies to to rotational equations and kinematic equations.

$$\begin{aligned}
\dot{p} =& \frac{1}{I_{xx}I_{zz} - I_{xz}^2}[I_{zz}((\mathcal{L}_A)_s + l_A + (\mathcal{L}_T)_s + l_T - (Q_s + q)(R_s + r)(I_{zz} - I_{yy}) + I_{xz}(P_s + p)(Q_s + q)) + \\
& I_{xz}((\mathcal{N}_A)_s + n_A + (\mathcal{N}_T)_s + n_T - (P_s + p)(Q_s + q)(I_{yy} - I_{xx}) - I_{xz}(Q_s + q)(R_s + r))], \\
\dot{q} =& \frac{1}{I_{yy}}\left[(\mathcal{M}_A)_s + m_A + (\mathcal{M}_T)_s + m_T - (R_s + r)(P_s + p)(I_{xx} - I_{zz}) - I_{xz}((P_s + p)^2 - (R_s + r)^2)\right], \\
\dot{r} =& \frac{1}{I_{xx}I_{zz} - I_{xz}^2}[I_{xz}((\mathcal{L}_A)_s + l_A + (\mathcal{L}_T)_s + l_T - (R_s + r)(Q_s + q)(I_{zz} - I_{yy}) + I_{xz}(P_s + p)(Q_s + q)) + \\
& I_{xx}((\mathcal{N}_A)_s + n_A + (\mathcal{N}_T)_s + n_T - (P_s + p)(Q_s + q)(I_{yy} - I_{xx}) - I_{xz}(Q_s + q)(R_s + r))].
\end{aligned}$$

$$\begin{aligned}
\dot{\phi} &= (P_s + p) + (Q_s + q)\sin(\phi_s + \phi)\,\mathrm{tg}(\theta_s + \theta) + (R_s + r)\cos(\phi_s + \phi)\,\mathrm{tg}(\theta_s + \theta), \\
\dot{\theta} &= (Q_s + q)\cos(\phi_s + \phi) - (R_s + r)\sin(\phi_s + \phi), \\
\dot{\psi} &= (Q_s + q)\frac{\sin(\phi_s + \phi)}{\cos(\theta_s + \theta)} + (R_s + r)\frac{\cos(\phi_s + \phi)}{\cos(\theta_s + \theta)}.
\end{aligned}$$

Several assumptions will now be made for the simplification of the equations and their manipulation. The first simplification will be performed using small angle approximations. For angle $\varphi \approx 0$, we can write $\cos\varphi \approx 1$, $\sin\varphi \approx \varphi$ and $\mathrm{tg}\,\varphi \approx \varphi$. This simplification holds quite well for angles up to 15 degrees. We can now represent the simplified trigonometric identities in the following table.

$$\sin(\varphi_s + \varphi) \approx \sin\varphi_s + \varphi\cos\varphi_s,$$

$$\cos(\varphi_s + \varphi) \approx \cos\varphi_s - \varphi \sin\varphi_s,$$
$$\operatorname{tg}(\varphi_s + \varphi) \approx \frac{\operatorname{tg}\varphi_s + \varphi}{1 - \operatorname{tg}\varphi_s \varphi}.$$

The second assumption made will be the elimination of terms representing steady–state motion, which is assumed to be satisfied as the motion is defined relative to a steady–state condition. Finally, the assumption of small perturbation will be made to eliminate the products and cross products of the perturbed variables, which are considered to be negligible. The remaining linear terms form a linearized set of equations of motion.
Translational equations:

$$\begin{aligned} m\left(\dot{u} - V_s r - R_s v + W_s q\right) &= -mg\theta\cos\theta_s + x_A + x_T, \\ m\left(\dot{v} + U_s r + R_s u - W_s p - P_s w\right) &= -mg\theta\sin\phi_s\sin\theta_s + mg\phi\cos\phi_s\cos\theta_s + y_A + y_T, \\ m\left(\dot{w} - U_s q - Q_s u + V_s p + P_s v\right) &= -mg\theta\cos\phi_s\sin\theta_s - mg\phi\sin\phi_s\cos\theta_s + z_A + z_T, \end{aligned}$$

Rotational equations:

$$\begin{aligned} I_{xx}\dot{p} - I_{xz}\dot{r} - I_{xz}\left(P_s q + Q_s p\right) + \left(I_{zz} - I_{yy}\right)\left(R_s q + Q_s r\right) &= l_A + l_T, \\ I_{yy}\dot{q} + \left(I_{xx} - I_{zz}\right)\left(P_s r + R_s p\right) + I_{xz}\left(2P_s p + 2R_s r\right) &= m_A + m_T, \\ I_{zz}\dot{r} - I_{xz}\dot{p} + \left(I_{yy} - I_{xx}\right)\left(P_s q + Q_s p\right) + I_{xz}\left(Q_s r + R_s q\right) &= n_A + n_T. \end{aligned}$$

Kinematic equations:

$$\begin{aligned} p &= \dot{\phi} - \dot{\psi}_s\theta\cos\theta_s - \dot{\psi}\sin\theta_s, \\ q &= -\dot{\theta}_s\phi\sin\phi_s + \dot{\theta}\cos\phi_s + \dot{\psi}_s\phi\cos\theta_s\cos\phi_s - \dot{\psi}_s\theta\sin\theta_s\sin\phi_s + \dot{\psi}\cos\theta_s\sin\phi_s, \\ r &= -\dot{\psi}_s\phi\cos\theta_s\sin\phi_s - \dot{\psi}_s\theta\sin\theta_s\cos\phi_s + \dot{\psi}\cos\theta_s\cos\phi_s - \dot{\theta}_s\phi\cos\phi_s - \dot{\theta}\sin\phi_s. \end{aligned}$$

Now, for most steady–state conditions data measured for the aircraft, wings–level steady–state straight flight is usually assumed so that $V_s = \phi_s = P_s = Q_s = R_s = \dot{\phi}_s = \dot{\theta}_s = \dot{\psi}_s = 0$. The equations simplify to Translational equations:

$$\begin{aligned} m\left(\dot{u} + W_s q\right) &= -mg\theta\cos\theta_s + x_A + x_T, \\ m\left(\dot{v} + U_s r - W_s p\right) &= mg\phi\cos\theta_s + y_A + y_T, \\ m\left(\dot{w} - U_s q\right) &= -mg\theta\sin\theta_s + z_A + z_T, \end{aligned}$$

Rotational equations:

$$\begin{aligned} I_{xx}\dot{p} - I_{xz}\dot{r} &= l_A + l_T, \\ I_{yy}\dot{q} &= m_A + m_T, \\ I_{zz}\dot{r} - I_{xz}\dot{p} &= n_A + n_T. \end{aligned}$$

Kinematic equations:

$$\begin{aligned} p &= \dot{\phi} - \dot{\psi}\sin\theta_s, \\ q &= \dot{\theta}, \end{aligned}$$

$$r = \dot{\psi} \cos \theta_s.$$

Using dimensional stability derivatives from Appendix B, we can now represent the perturbed forces and moments for both the longitudinal and lateral directional equations of motion and substitute the state variable $w = (V_{TAS})_s \alpha$ and $U_s = (V_{TAS})_s$. We also assume $W_s = 0$ so that the set of equations decouple completely. Note that the mass of the aircraft m is incorporated into the stability derivatives.

$$\begin{aligned}
\dot{u} &= -g\theta \cos \theta_s + X_u u + X_{T_u} u + X_\alpha \alpha + X_{\delta_e} \delta_e, \\
V_{TAS}\dot{\alpha} - (V_{TAS})_s \, q &= -g\theta \sin \theta_s + Z_u u + Z_\alpha \alpha + Z_{\dot{\alpha}} \dot{\alpha} + Z_q q + Z_{\delta_e} \delta_e, \\
I_{yy}\dot{q} &= M_u u + M_{T_u} u + M_\alpha \alpha + M_{T_\alpha} \alpha + M_{\dot{\alpha}} \dot{\alpha} + M_q q + M_{\delta_e} \delta_e, \\
q &= \dot{\theta}.
\end{aligned}$$

From there, the matrix format presented in the text can be easily derived. For the lateral directional equations, using substitutions $v = V_{TAS}\beta$

$$\begin{aligned}
\dot{\beta} \, (V_{TAS})_s + (V_{TAS})_s \, r &= g\phi \cos \theta_s + Y_\beta \beta + Y_p p + Y_r r + Y_{\delta_a} \delta_a + Y_{\delta_r} \delta_r, \\
I_{xx}\dot{p} - I_{xz}\dot{r} &= L_\beta \beta + L_p p + L_r r + L_{\delta_a} \delta_a + L_{\delta_r} \delta_r, \\
I_{zz}\dot{r} - I_{xz}\dot{p} &= N_\beta \beta + N_p p + N_r r + N_{\delta_a} \delta_a + N_{\delta_r} \delta_r, \\
p &= \dot{\phi} - \dot{\psi} \sin \theta_s, \\
r &= \dot{\psi} \cos \theta_s.
\end{aligned}$$