High Angle of Attack Landing of an Unmanned Aerial Vehicle

Kristoffer Gryte

Master of Science in Cybernetics and Robotics
Submission date: July 2015
Supervisor: Thor Inge Fossen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

NTNU **Norwegian University of** **Science and Technology**	**Faculty of Information Technology,** **Mathematics and Electrical Engineering** **Department of Engineering Cybernetics**

MSC THESIS DESCRIPTION SHEET

Name:	Kristoffer Gryte
Department:	Engineering Cybernetics
Thesis title (Norwegian):	Landing av ubemanneded fly ved bruk av stor angrepsvinkel
Thesis title (English):	High Angle-of-Attack Landing of an Unmanned Aerial Vehicle

Thesis Description: The purpose of the thesis is to investigate how a control strategy based on high angle-of-attack (AOA) can be used to automatically land an unmanned aerial vehicle (UAV) in a net. This involves mathematical modeling of the X8 fixed-wing UAV and simulation of high AOA maneuvers. Furthermore, the X8 model should be used to design and test an automatic control system for landing.

The following items should be considered:

1. Define the scope of the thesis and clarify what your contributions are.
2. Development of a flight simulator the X8 UAV, which is valid for high AOA. The simulator should also include wind loads on the fuselage.
3. Development of a software-in-the-loop simulator for the X8 using JSBSim interfaced with the ArduPlane autopilot.
4. Formulate landing as an optimal control problem, that has low terminal velocity as the objective. Use this to investigate how the terminal velocity is affected by different initial conditions and constraints.
5. Extend the optimal control problem to a model predictive controller, and use this in landing simulations.
6. Conclude your results.

Start date:	2015-01-05
Due date:	2015-07-01

Thesis performed at:	Department of Engineering Cybernetics, NTNU
Supervisor:	Professor Thor I. Fossen, Dept. of Eng. Cybernetics, NTNU

Abstract

Motivated by the limited landing space on board a ship, this thesis investigates the landing of the Skywalker X8 fixed wing unmanned aerial vehicle (UAV) in a net. Further motivated by the way birds abruptly, and equally elegantly, reduce their velocity when landing on a perch or the branch of a tree, a perched landing strategy utilizing the increased drag experienced for large angles of attack is used to minimize the velocity at impact with the net. This is accomplished by expressing landing as an optimal control problem, taking advantage of a nonlinear model of the X8 that is valid for high angles of attack. At this stage, the optimal control problem is only concerned with the three longitudinal degrees-of-freedom. It is solved in an open source optimization framework, using a nonlinear interior point method. Further, a nonlinear model predictive controller (NMPC) that can control the X8 throughout the landing is developed. At the heart of the optimal control problem lies a linear model, blended with a flat plate model to increase its validity for high angles of attack in lift, drag and pitch moment. The linear model is developed using an easy-to-use computational fluid dynamics (CFD) modeling software. In addition a six degrees-of-freedom software-in-the-loop (SITL) simulator is developed for future use in testing of hardware-near implementations of the controller, and to allow for validation of the model through pilot testing. Comparison of six different landing scenarios yield a wide range of landing velocities, depending on the constraints of the scenario. Simulations with the developed NMPC show that the same performance is achievable through control of the X8 under minor environmental disturbances. From this it is concluded that the perched landing strategy will lead to a considerable reduction in terminal absolute velocity, compared to a low angle of attack approach. It is found to be advantageous to start from a low altitude, landing into an elevated net. However, whether an equally large reduction is possible in practice depends on the capabilities of the real-time implementation and the validity of the model, particularly the propeller model, the pitching moment and drag coefficients. Finally it depends on how the lateral degrees-of-freedom are affected by the high angle of attack flight.

Samandrag

(*Norwegian Translation of the Abstract*)
Motivert av den avgrensa landingsplassen ombord på skip, undersøkjer denne oppgåva nettlanding av det ubemanna flyet Skywalker X8. Vidare motivert av den snøgge nedbremsinga fuglar elegant oppnår ved landing på greiner, ser denne oppgåva nærare på ein landingsstrategi som utnyttar den drastiske auken i luftmotstand, som finn stad for store angrepsvinklar, til å minimere hastigheita i landingsaugneblinken. Dette blir gjort ved å formulere landinga som eit optimaliseringsproblem som nyttar seg av ein ulineær modell av X8 som er gyldig for store angrepsvinklar. Optimaliseringsproblemet har blitt forenkla til dei tre fridomsgradene i longitudinalplanet, og blir løyst ved hjelp av eit optimaliseringsrammeverk basert på open kjeldekode som nyttar ein ulineær indre-punkt løysar. Vidare har ein ulineær modell-prediktiv regulator blitt utvikla for å kunne kontrollere X8 gjennom landinga. Ein sentral del av optimaliseringsproblemet er ein eigenutvikla lineær modell, der løft-, luftmotstand- og stampkoeffisientane gradvis går over i ein flat plate modell for å auke gyldigheita ved store angrepsvinklar. Den lineære modellen har blitt utvikla ved hjelp av eit enkelt program for numerisk strøymingsanalyse. I tillegg har det blitt utvikla ein programvare-i-sløyfa simulator i seks fridomsgrader for å kunne nyttast i testing av maskinvarenære implementeringar av regulatoren, samt for å legge til rette for modellvalidering gjennom pilottesting. Samanlikning av seks ulike landingsscenario, med ulike avgrensingar, gav ulike landingshastigheiter. Ei simulering med den modell-prediktive regulatoren har vist at tilsvarande ytelsar er oppnåelege ved regulering av X8 under ytre påverknader. Basert på desse resultata blir det konkludert at ein slik landingsstrategi, med stor angrepsvinkel, vil føre til ein stor reduksjon i absolutthastigheita i landingsaugneblinken, samanlikna med å lande med ein låg angrepsvinkel. Det har synt seg føremonleg å starte landinga frå ei låg høgde, samt å lande i eit heva nett. Men om ein like stor reduksjon er mogleg i praksis avhengjer av sanntidsimplementasjonen av systemet og gyldigheita av modellen, herunder særskildt propellmodellen, stampemoment- og luftmotstandskoeffisientane. Det vil også vere avhengig av korleis fridomsgradene i lateralplanet vert påverka av flyging med stor angrepsvinkel.

Preface

This thesis is submitted in partial fulfillment of the requirements for the Master of Science degree at the Norwegian University of Science and Technology.

The work presented in this thesis has been carried out at the Department of Engineering Cybernetics, and would not have been possible without the inputs and contributions from many of its associates. First, I would like to thank Professor Thor Inge Fossen for his inspiring ideas and guidance into the academic world. The team at the UAV-lab deserve to be acknowledged; especially Artur Zolich, who has been an important sparring partner in my aerodynamic endeavors, and Siri H. Mathisen, whom I have worked closely with on optimization and model predictive control. I would also like to thank my fellow MSc. students from room G232B. They have given me invaluable feedback in a competitive, yet friendly environment. Finally I would like to thank my parents and brother for believing in me and encouraging me when times get tough.

Kristoffer Gryte
Trondheim, June 2015

Table of Contents

Abstract **i**

Abstract in Norwegian **iii**

Preface **v**

Table of Contents **x**

List of Tables **xi**

List of Figures **xiv**

List of Code Snippets **xv**

Nomenclature **xvi**

I Introduction and Background Material **1**

1 Introduction **3**
1.1 Skywalker X8 3
1.2 Previous work 4
1.3 Contributions 5
1.4 Structure of this thesis 6

2 Theory **9**
2.1 UAV coordinate systems and models 10
2.1.1 Body-axis 10
2.1.2 Stability-axis 10
2.1.3 UAV models 11
2.1.4 Mapping from elevon to aileron-elevator 12

2.2 Aerodynamics 12
2.2.1 Parametrization of the aerodynamical coefficients 12
2.2.2 Flat plate model 13
2.3 Rotations between frames of reference 14
2.4 Wind model 15
2.4.1 The Dryden turbulence model 16

II Methods **17**

3 Model **19**
3.1 From a physical UAV to aerodynamic coefficients 19
3.2 Choosing model structure 20
3.3 Model extension 20
3.3.1 Drag depending on sideslip 20
3.3.2 Nonlinear extension to lift and drag 21
3.3.3 Flat plate pitch moment 21
3.4 Final model structure 24
3.5 Adaptation of coefficient 24

4 Simulator **25**
4.1 Simulink simulator 25
4.1.1 Simulink structure 26
4.1.2 FlightGear 27
4.1.3 Calculating trim conditions 29
4.2 Software-in-the-loop simulator 29
4.2.1 ArduPlane 30
4.2.2 Flight dynamics model 31
4.2.3 DUNE 31
4.2.4 Mission-centric software-in-the-loop simulator 32
4.2.5 Pilot-centric software-in-the-loop simulator 32

5 Optimal landing **33**
5.1 What is optimal? 33
5.2 Achieving low terminal speed 34
5.2.1 Increasing the potential energy 34
5.2.2 Increasing drag 34
5.3 Characteristics of the high angle of attack region 36
5.4 Optimal landing trajectory in the high angle of attack region 39
5.4.1 Problem setup 39
5.4.2 Continuous-time optimal control 40
5.4.3 Continuous-time optimal control problem formulation 43
5.5 Control in the high angle of attack region 45
5.5.1 Nonlinear Model Predictive Control 45
5.6 Simulation setup in CasADi 46
5.6.1 IPOPT 46

III Results **49**

6 Model **51**
6.1 Inertial parameters 51
6.1.1 Mass and center of gravity 51
6.1.2 Moments- and products of inertia 52
6.2 Aerodynamic parameters 52
6.2.1 Nonlinear pitching moment coefficient 52

7 Landing **55**
7.1 Scenario 1: Normal landing 57
7.2 Scenario 2: Fixed distance and initial height 58
7.3 Scenario 3: Free initial height 59
7.4 Scenario 4: Free distance 60
7.5 Scenario 5: No throttle 61
7.6 Scenario 6: Surface landing 62
7.7 Nonlinear model predictive control for scenario 2 in gust wind 63

IV Discussion and Conclusions **65**

8 Discussion **67**
8.1 Optimistically low terminal velocity 67
8.1.1 Too demanding trajectories 67
8.1.2 Too much drag 69
8.2 Symmetric stall assumption 69
8.2.1 Spin prevention 70
8.2.2 Spin modeling 70
8.3 Non-convex optimization 70
8.4 Future work 71
8.4.1 Real-time implementation 71
8.4.2 Model validation and expansion 71

9 Conclusion **73**

V Appendices **75**

A Submitted abstracts **77**
A.1 Submitted conference abstract for *SciTech 2016* 78
A.2 Submitted conference abstract for *2016 IEEE Aerospace Conference* . . . 80

B Model structure **83**

C XFLR5 **85**
C.1 XFLR assumptions and flow theory 86
C.1.1 Reynolds number 86

C.1.2 Incompressible flow . . . 86
C.1.3 Boundary layer . . . 86

D Model **89**
D.1 Importing a physical UAV into a CFD program . . . 89
D.1.1 Digitalizing the airfoils . . . 90
D.1.2 3D model in XFLR . . . 91
D.2 CFD analysis and interpretation . . . 93
D.2.1 Airfoil analysis . . . 93
D.2.2 Plane analysis . . . 94

E Airfoil analysis **97**
E.1 Wing chord and offset . . . 97
E.2 Airfoils . . . 97
E.3 XFLR analysis raw data . . . 101

F Moment of inertia and center of gravity calculations **105**
F.1 Pendulum method: Finding I_{ij} . . . 105
F.2 Finding center of gravity . . . 107
F.3 Calculations and intermediate results . . . 110

G Running the simulators **113**
G.1 Simulink simulator . . . 113
G.1.1 Command to run FlightGear . . . 113
G.2 Pilot-centric software-in-the-loop simulator . . . 113
G.2.1 Command to run FlightGear . . . 114
G.2.2 Command to run ArduPlane . . . 114
G.2.3 Command to run JSBSim . . . 114
G.2.4 Command to run DUNE . . . 114
G.2.5 Command to run mavproxy . . . 115
G.3 Mission-centric software-in-the-loop simulator . . . 115

H Collection of parameters **117**
H.1 Motor parameters . . . 117
H.2 Battery parameters . . . 118

Bibliography **119**

List of Tables

6.1 Center of gravity . . . 51
6.2 Mass of different X8 components . . . 52
6.3 Inertia results . . . 52
6.4 Model parameters for the Skywalker X8 . . . 53

7.1 Terminal velocities for the different scenarios . . . 56

D.1 Airfoil analysis settings . . . 94
D.2 Plane analysis settings . . . 94
D.3 Stability analysis settings . . . 95

E.1 Wing and chord measurements from 3d scan (in millimeters) . . . 100

F.1 Inertia results . . . 111
F.2 Finding T_p and T_f for I_{xx} . . . 111
F.3 Finding T_p and T_f for I_{yy} . . . 111
F.4 Finding T_p and T_f for I_{zz} . . . 111
F.5 Finding T_p and T_f for I_{xz}. nose up . . . 111
F.6 Finding T_p and T_f for I_{xz}, nose down . . . 111

H.1 Motor parameters for the Hacker A40-12S V2 14-Pole . . . 117
H.2 Battery parameters for the ZIPPY Compact 5000mAh 4S 25C Lipo Pack . 118

List of Figures

1.1 The Skywalker X8 in flight 4

2.1 Sideslip β and angle of attack α 9

3.1 X8 with x_{cg} coordinate 22
3.2 X8 with x_{cg} coordinate 22

4.1 The top level structure of the control system simulator 28
4.2 Structure of the X8 block 28
4.3 The actuator model block 28

5.1 Deceleron configuration 35
5.2 Crow configuration 35
5.3 Deep stall and perched landing 36
5.4 General lift, drag and pitching moment coefficients for large α 38
5.5 Shifting of the pitching moment coefficient 38

6.1 Blend of linear and flat plate lift, drag and pitch moment coefficients . . . 54

7.1 Normal landing 57
7.2 Landing with fixed distance to the net and fixed initial height 58
7.3 Landing with fixed distance to the net and free initial height 59
7.4 Landing with free distance to the net and fixed initial height 60
7.5 Landing without throttle 61
7.6 Landing at $h = 1$m 62
7.7 NMPC with gust 64

D.1 Unscaled airfoils imported to XFLR 91
D.2 The airfoil splines on the 3D surface 92
D.3 Model assembled in XFLR 93

E.1 Visual comparison of X8 foils from center until 450mm 98
E.2 Visual comparison of X8 foils from 475mm until wingtip 99

F.1 Pendulum of mass M and inertia I_{ij} hanging from a massless rod 106
F.2 Setup for performing the pendulum test for finding I_{xz} 108
F.3 The setup for the pendulum test for finding I_{xx} and I_{yy} 108
F.4 The setup for the pendulum test for finding I_{zz} 109
F.5 The minimal payload of the X8 with which the pendulum tests were performed 110

G.1 The structure of the PID controller in Simulink 116

Listings

E.1 XFLR output:Elevons as ailerons 101
E.2 XFLR output:Elevons as elevators 103
G.1 Command to start FlightGear with Simulink simulator 113
G.2 Command to start FlightGear for SITL 114
G.3 Command to start ArduPlane 114
G.4 Command to start JSBSim 114
G.5 Command to start DUNE 115
G.6 Command to start Mavproxy 115
G.7 Minimal DUNE configuration file for running SITL 116

Nomenclature

Aerodynamics

$\bar{q}$	Dynamic pressure: $\bar{q} = 1/2\rho V_a^2$
ρ	Air density
C_D	Dimensionless aerodynamic drag coefficient: $C_D = D/(\bar{q}S)$
C_L	Dimensionless aerodynamic lift coefficient: $C_L = L/(\bar{q}S)$
C_l	Dimensionless aerodynamic rolling moment coefficient: $C_l = l/(\bar{q}bS)$
C_m	Dimensionless aerodynamic pitching moment coefficient: $C_m = m/(\bar{q}cS)$
C_n	Dimensionless aerodynamic yawing moment coefficient: $C_n = n/(\bar{q}bS)$
C_Y	Dimensionless aerodynamic side force coefficient: $C_Y = Y/(\bar{q}S)$
$C_{D,flatplata}$	Drag force coefficient based on flat plate theory
C_{D_0}	Drag force coefficient when $\alpha = q = \delta_e = 0$
$C_{L,flatplata}$	Lift force coefficient based on flat plate theory
C_{L_0}	Lift force coefficient when $\alpha = q = \delta_e = 0$
C_{l_0}	Rolling moment coefficient when $\beta = p = r = \delta_a = \delta_r = 0$

Body frame variables	x	y	z
Position	x	y	z
Attitude	θ	φ	ψ
Linear velocity	v	u	w
Angular velocity	p	q	r
Force	F_x	F_y	F_z
Moment	l	m	n

C_{m_0}	Pitching moment coefficient when $\alpha = q = \delta_e = 0$
C_{n_0}	Yawing moment coefficient when: $\beta = p = r = \delta_a = \delta_r = 0$
C_{prop}	Dimensionless propeller constant for adjusting the propeller model, depending on e.g. propeller blade pitch.
C_{Y_0}	Side force coefficient when $\beta = p = r = \delta_a = \delta_r = 0$
D	Aerodynamic drag force
e	Oswalds efficiency number, a correction factor representing the difference in drag for a given 3D wing and a "perfect" wing with elliptical lift distribution. It is possible to prove that an elliptical lift distribution yields the minimal downwash, resulting in minimal drag.
k_{motor}	Motor constant representing the relationship between throttle and propeller discharge velocity at zero airspeed. $k_{motor} = V_d / \delta_t$
L	Aerodynamic lift force
l	Aerodynamic rolling moment
m	Aerodynamic pitching moment
n	Aerodynamic yawing moment
S_{prop}	Area swept by the propeller.
S_{wing}	Area of the wing
u_r	Relative velocity along the body x-axis: $u_r = u - u_w$
u_w	Wind velocity along the body x-axis
V_a	Airspeed: $V_a = \sqrt{u_r^2 + v_r^2 + w_r^2}$
V_d	Discharge velocity from the propeller
v_r	Relative velocity along the body y-axis: $v_r = v - v_w$
v_w	Wind velocity along the body y-axis
w_r	Relative velocity along the body z-axis: $w_r = w - w_w$
w_w	Wind velocity along the body z-axis
Y	Aerodynamic side force
Aileron	Control surfaces on the wings used to control roll. δ_a is used to represent its angular deflection.
Airfoil	A cross-section of a wing, parallel to the xz-plane. A lot of the aerodynamical characteristics of a plane can be found from analyzing the airfoils.

Angle of attack	The angle α between the chord line and velocity vector of the incoming air. Measured in degrees or radians.
Aspect Ratio	Ratio between the length and chord of a wing. Higher aspect ratio leads to lower drag. $\text{Æ} = b^2/S$
b	Wingspan
Camber	The asymmetry between the upper and lower surface on an airfoil. A related expression is the camber line which is the line going through the midpoints between the upper and lower surface. The camber of an airfoil is defined by the wing profile, but can be changed in-flight by changing the flaps and control surfaces of the wing. Changing the camber leads to shift of the $C_l$vs α curve, at the cost of small changes in dragMeshia (2008)
Chord	The length c of the straight line between the airfoil leading- and trailing egde
Dihedral	The upward angle between the wing and the horizontal plane.Measured in degrees or radians.
Elevator	Control surface used primarily to control the pitch. δ_e is used to represent its angular deflection.
Elevon	When aileron are used both as ailerons and as an elevator they are referred to as elevons. δ_{er} and δ_{el} is used to represent its right and left angular deflection respectively.
NACA	Abbreviation for National Advisory Committee for Aeronautics, a former U.S. federal agency for aeronautical research. They developed a series of airfoils, naming them NACA followed by a four digit number[1]. The first digit describes the maximum camber as a percentage of the chord, whereas the second number describes the position of this maximum camber position in tens of percent of the chord. The last two digits describe the maximum thickness, also expressed in percentage of the chord. One alternative system for parametrising airfoils is the MH-series.
Rudder	Control surface primarily used to control yaw. δ_r is used to represent its angular deflection.
Sideslip	The angle β between the direction the nose is pointing and the velocity vector of the plane. Measured in degrees or radians.
Sweep	The angle between the y-axis and the wings.
VLM	Vortex Line Method.
Wing area	The planform area S of the wings. Measured in sqare meters.

[1]The four-series NACA foils are the most common one, but other exists.

Winglet	A wing configuration where the tip of the wing generally is pointing upwards.
Wingspan	The length b of the straight line between the wing tips. Measured in meters.
XFLR	An easy-to-use, open source VLM CFD program.

Abbreviations

CAD	Computer-Aided Design
CFD	Computational Fluid Dynamics
CFD	Computational Fluid Dynamics
FDM	Flight Dynamics Model
HIL	Hardware-in-the-loop
IPOPT	Interior Point OPTimizer
JSBSim	Jon S. Berndt Simulator
KKT	Karush-Kuhn-Tucker
LQR	Linear Quadratic Controller
NLP	Nonlinear Programming Problem
NMPC	Nonlinear Model Predictive Control
OCP	Optimal Control Problem
ODE	Ordinary Differential Equation
PID	Proportional, Derivative, Integral
RPM	Revolutions Per Minute
SITL	Software-in-the-loop
SQP	Sequential Quadratic Programming
UAV	Unmanned Aerial Vehicle

Part I

Introduction and Background Material

Chapter 1

Introduction

Like other robots, UAVs are often used in operations that are dull, dirty or dangerous. Many of these operations require intervention in remote locations where access to infrastructures, such as runways, is limited. For applications that require longer range than the typical rotary wing UAV, fixed-wings UAVs are used, with the disadvantage that landing becomes more of a challenge. Part of the research community focus on Vertical Take-Off and Landing (VTOL) design, making trade-offs between the properties of rotary wing and fixed-wing UAVs. This often results in large structural changes, both physically and in the control architecture, and in a UAV that has worse performance and increased weight compared to a conventional fixed-wing UAV. This motivates the research on Short Take-Off and Landing (STOL) UAVs, typically with a catapult launcher for take-off and landing in an arrest system. To further reduce the landing velocity, and thus the impact with the arrest system, this thesis focus on landing techniques that utilize the increased drag at high angles of attack. The inspiration comes from the animal kingdom, where birds, flying squirrels and geckos have exploited this phenomenon for millions of years.

The planning and control of the landing motivates the development of a model of the Skywalker X8. A good model is important in model based control design, and will increase the accuracy of observers needed to estimate the states of the UAV. Examples include GPS dead-reckoning, assisting in solving the RTK-GPS integer ambiguity, and wind estimation. In addition, a good model makes HIL(Hardware-in-the-loop) and SITL(Software-in-the-loop) testing more realistic. This is important since it enables the developer to discover bugs in the system at an earlier stage, and it may reduce the time needed for flight tests.

1.1 Skywalker X8

The Skywalker X8 is a fixed wing UAV in a flying wing configuration. This means that it has no tail and no clear distinction between the wing and the fuselage. The X8 was originally designed for FPV (First Person View) flights, but has since its release been widely accepted by the model plane community. At the UAV-lab at the Department of Engineering Cybernetics, the X8 is a popular UAV for experimental missions since it is durable, cheap, easily available and has a large community. Figure 1.1 shows a picture of one of the UAV-lab's X8s in flight[1].

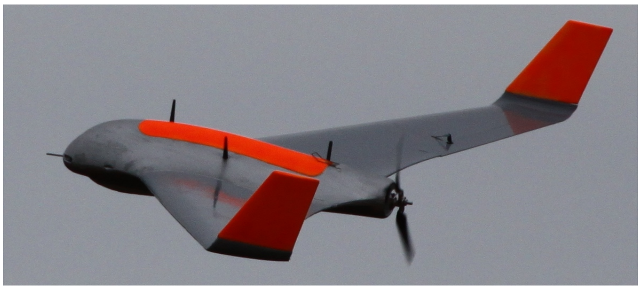


Figure 1.1: The Skywalker X8 in flight

1.2 Previous work

Not surprisingly, airplane control is well covered by the literature. However, most textbooks are only concerned with the linear models in the low angle of attack region (Stevens and Lewis (2003); Etkin and Reid (1996)). Stengel (2004) includes good descriptions of flight at high angle of attack, while Anderson (1989) is a good source on general aerodynamics.

The literature reports many different approaches to the landing control problem. Gautam et al. (2014) is a good summary of current UAV landing techniques. Beard and McLain (2012) gives a thorough description of UAV transfer functions and PID design. However, Rao and Hiong (2014) concludes that the sliding mode control outperforms the PID. Another nonlinear landing controller is presented in Prasad B. and Pradeep (2007), where a feedback linearization controller is used to land an F-16 fighter. This article also gives a detailed description of the normal landing approach for manned aircraft, and points out the criticality of landing; 60% of the accidents in general aviation are related to landing. Skulstad et al. (2015) lands a Skywalker X8 in a net, without considering velocity reduction during impact.

None of the aforementioned controllers are used in the high angle of attack domain. Crowther and Prassas (1999) reviews current methods of UAV retrieval, and outlines a post stall landing implementation. By applying a local time-varying linear controller to perched landing, Roberts et al. (2009) finds that controllability and robustness is very limited, calling for more advanced landing schemes. Further, Pointner and Kotsis (2011) successfully implements a hybrid controller to conduct a deep stall landing for a delta wing

[1]In courtesy of the UAV-lab

UAV. This design also includes recovery from missed approaches. In order to improve the accuracy of the aerodynamic model in the high angle of attack region, this article use a flat plate approximation of the wing blended with the lift and drag coefficients based on linear theory. This nonlinear extension is also used in deep stall landing of a tailed UAV by Taniguchi (2008), and by Mathisen et al. (2015) which use nonlinear model predictive control to track a constant path angle while minimizing velocity.

Highly inspired by birds, Lussier Desbiens et al. (2011) makes a small UAV attach to a vertical wall by the use of a special landing gear consisting of arrays of microspines. The UAV flies towards the wall, sensing the horizontal distance with an ultrasonic range finder. Close to the wall, the UAV pitches up to achieve a near-ballistic, perched flight the final distance towards the wall. The height at which the UAV attach to the wall is not emphasized in this research. A similar landing scheme is presented in Moore et al. (2014), however this analysis is more focused on the nonlinear control design. Through the use of a motion capture system providing high accuracy position and attitude measurements at a high frequency, a small UAV lands on a perch. The optimal landing trajectories for many different initial conditions are calculated offline, and are tracked online using a time-varying linear quadratic controller in combination with a Lyapunov based method to extend the region of attraction around these trajectories. The model used in this article is a flat plate model which has been improved through system identification via the motion capture system (Cory and Tedrake (2008), Hoburg and Tedrake (2009)).

Airfoil analysis of model airplanes is used by many RC-pilots, see Meshia (2008). Within modelling, Jodeh (2006) and Paw (2009) have derived both the aerodynamic and mechanical parameters of UAVs, but none of them have considered the Skywalker X8 in particular. Working with autopilots for landing, Devesa et al. (2004) has a lot of useful information as to how simulators are structured and used, and how model identification is done. However, this is related to identification of ground effects for full size airplanes and have to be adapted to UAVs. Finally, Beard and McLain (2012) provide a good framework for working with UAV simulation and control, through exercises and online video tutorials. This is based on another UAV, but provides a good reference for the X8.

1.3 Contributions

This thesis has two main concerns: the development of a software-in-the-loop simulator for the Skywalker X8 that is valid for large angles of attack, and landing of the X8 utilizing the increased drag for large angles of attack. Through this work, the following contributions are made:

- A **six degrees-of-freedom model of the Skywalker X8, valid for large angles of attack** is developed through the use of a 3D scanning, and the XFLR computational fluid dynamics software. See Chapter 3.
- As a joint venture with Jostein Furseth, a **nonlinear pitching moment coefficient** is suggested in Section 3.3.3. This is based on the flat plate approximation of an airfoil, but takes the large sweep of the X8 into account, as opposed to being based

on the mean aerodynamic chord. No use of nonlinear pitching moment coefficient have been reported in the literature on high angle of attack landing.

- With the aforementioned model as a starting point, a **software-in-the-loop simulator** is developed using the JSBSim flight dynamics model, ArduPlane autopilot and DUNE runtime environment, to get a close resemblance to real mission flights. The details are found in Section 4.2. As a by-product, a Simulink simulator is developed in Section 4.1. This is suitable for early-stage development in Matlab/Simulink. Both simulators have already been adopted by other members of the UAV-lab team, for use in their projects.
- The model also enabled the **creation and evaluation of different optimal landing trajectories**, where the increased drag in the high angle of attack region to acheive low terminal velocity, see Chapter 7. The main results have been submitted to the 2016 IEEE Aerospace Conference in Big Sky, Montana, U.S., see Appendix A.2.
- The work behind this thesis is also contributing to Mathisen et al.(Submitted), found in Appendix A.1, where a six degrees-of-freedom deep stall landing scheme is developed using quaternions.
- Finally a **nonlinear model predictive controller is developed** using a direct multiple shooting method in the CasADi optimization framework with the IPOPT solver. This is explained in Section 5.5.1.

1.4 Structure of this thesis

This thesis is divided into nine chapters and eight appendices.

- Chapter 2 presents basic theory and background information that is needed in the subsequent chapters. This includes notation, coordinate frames and basic aerodynamics.
- The development of the mathematical model of the X8 is described in Chapter 3. Part of the calculations, and the setup for the analysis in XFLR is placed in Appendix D.
- In Chapter 4, the mathematical model is used to develop two simulators. A Simulink simulator simulator is presented in Section 4.1, while Section 4.2 covers the development of a SITL simulator that is interfaced with the actual control system of the X8. Appendix G includes a summary of commands used to run the different simulators.
- Chapter 5 describes the optimal landing problem, by characterization of the high angle of attack flight region; both the advantages and what should be avoided. Finally the problem is formulated as an optimal control problem.
- Chapter 6 proposes the model of the X8, while Chapter 7 describes different landing scenarios and present their outcome.

- Chapters 8 and 9 finalize the thesis through discussion and conclusion.
- Appendix A include the abstract of two articles, partially based on this thesis, that have been submitted to the AIAA SciTech 2016 and the 2016 IEEE Aerospace conferences. This work has been in collaboration with Siri H. Mathisen.
- Appendix B includes the final model structure for the aerodynamic, propulsion and gravitational forces in the developed X8 model.
- Appendices C-F include previous work on how the X8 was analyzed using the XFLR CFD program, as well as a method for calculating the inertia of the X8.
- Appendix G includes startup commands for the simulators.
- Appendix H contains motor and battery parameters.

Chapter 2

Theory

This chapter will explain some of the basic theory and background information that is needed to comprehend the material that is presented throughout this thesis. In order to make sense of the equations in the subsequent chapters, the reader is urged to familiarize with the nomenclature given on page xvi.

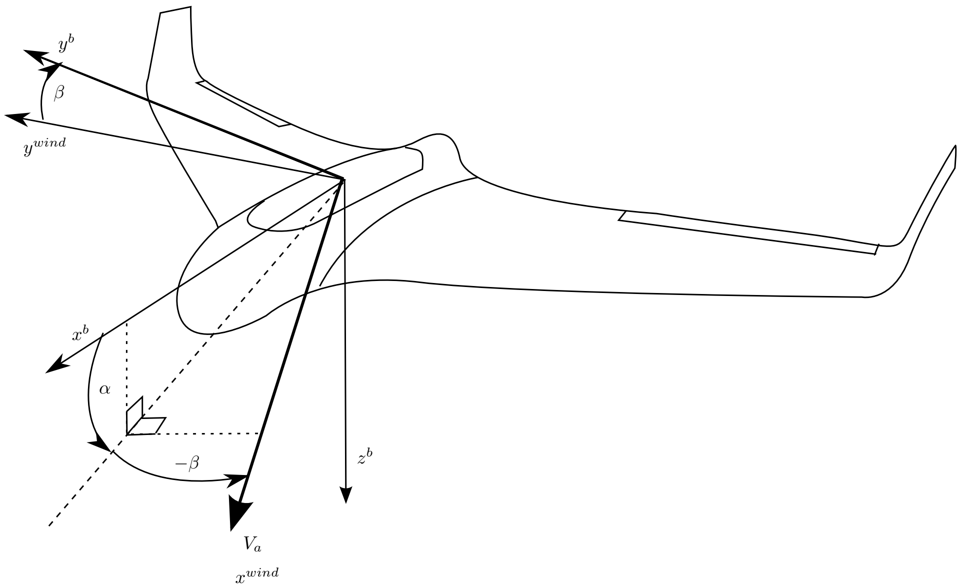


Figure 2.1: Sideslip β and angle of attack α

2.1 UAV coordinate systems and models

There are many different coordinate systems used to describe UAV kinematics and kinetics. This also makes it possible to express the UAV equations of motion in many different ways, see e.g. Etkin and Reid (1996), Beard and McLain (2012) or Stevens and Lewis (2003). The models are essentially the same, as they are based on the same physics and referenced to the same inertial system. Transforming from one to another is a matter of rotation and/or translation, assuming the models are in their complete, non-simplified form. Two of the most common coordinate systems are explained in the following sections.

2.1.1 Body-axis

The body frame is attached to the UAV body, and is perhaps more intuitive when thinking about body motion since forces are expressed directly in X-,Y- and Z- direction. This makes the linking from forces to body movement straight forward, making it attractive to express the higher layers of UAV simulators in body frame, as opposed to stability frame. The X-axis is pointing forward through the nose of the UAV, the Y-axis along the right wing, while the Z-axis points through the belly of the UAV completing a right-handed system. It should be apparent that the Z-axis is not necessarily pointing down towards the center of the earth, as the plane might be rolling or pitching. The aerodynamic moments

$$\begin{aligned} l &= \frac{1}{2}\rho V_a^2 SbC_l(\beta, p, r, \delta_a, \delta_r) \\ m &= \frac{1}{2}\rho V_a^2 SC_m(\alpha, q, \delta_e) \\ n &= \frac{1}{2}\rho V_a^2 SbC_n(\beta, p, r, \delta_a, \delta_r) \end{aligned} \tag{2.1}$$

as well as the lateral force

$$F_y = \frac{1}{2}\rho V_a^2 SC_Y(\beta, p, r, \delta_a, \delta_r) \tag{2.2}$$

are normally defined in body coordinates. Here ρ is the density of air in kg/m^3, and C_i are nondimensional aerodynamic coefficients that will be explained in Section 2.2.

2.1.2 Stability-axis

The stability frame depends on how the UAV moves with respect to the surrounding air, and is perhaps more intuitive when considering the origin of the aerodynamic forces affecting the UAV. The X-axis is pointing in the direction of motion relative to the wind. If we compare to the body-axes, this means that first the X-axis rotated by the angle of attack around the Y-axis, and then the Y-axis is rotated by the sideslip angle around the current Z-axis. This is illustrated in Figure 2.1. The aerodynamic lift and drag forces, defined in

(2.5), are naturally defined in stability-axis.

$$F_{lift} = \frac{1}{2}\rho V_a^2 S C_L(\alpha, q, \delta_e) \tag{2.3}$$

$$F_{drag} = \frac{1}{2}\rho V_a^2 S C_D(\alpha, q, \delta_e) \tag{2.4}$$

$$\tag{2.5}$$

2.1.3 UAV models

Beard and McLain (2012, Equation 4.18-20) states the total forces acting on a UAV in body frame. This can be reformulated to show the relationship to the aerodynamic coefficients expresed in stability frame:

$$\begin{aligned}
\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} &= -\frac{1}{2}\rho V_a^2 S \mathbf{R}_{stability}^{body}(\alpha, \beta) \begin{bmatrix} C_D(\alpha, q, \delta_e) \\ 0 \\ C_L(\alpha, q, \delta_e) \end{bmatrix} + \frac{1}{2}\rho V_a^2 S \begin{bmatrix} 0 \\ C_Y(\beta, p, e, \delta_a, \delta_r) \\ 0 \end{bmatrix} \\
&\quad + \mathbf{R}_{NED}^{body}(\varphi, \theta) \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + \frac{1}{2}\rho S_{prop} C_{prop} \begin{bmatrix} V_d\,(V_d - V_a) \\ 0 \\ 0 \end{bmatrix} \\
\begin{bmatrix} l \\ m \\ n \end{bmatrix} &= \frac{1}{2}\rho V_a^2 S \begin{bmatrix} bC_l(\beta, p, r, \delta_a, \delta_r) \\ C_m(\alpha, q, \delta_e) \\ bC_n(\beta, p, r, \delta_a, \delta_r) \end{bmatrix} + \begin{bmatrix} -k_{T_p}\,(k_\Omega \delta_t)^2 \\ 0 \\ 0 \end{bmatrix}
\end{aligned} \tag{2.6}$$

The final elements of both the forces and moments in Equation (2.6) represent the propeller forces and moments. This is given by Beard (2014), and illustrates that the thrust given by the propeller decreases with increasing airspeed and constant δ_t. Here $V_d = V_a + \delta_t(k_{motor} - V_a)$ is the propeller discharge velocity, i.e. the velocity of the air leaving the propeller, which is not necessarily equal to the airspeed V_a.

$\mathbf{R}_j^i$ is a rotation matrix that, when left-multiplying a vector, rotates the vector from frame j to frame i, see Section 2.3. This can easily be combined with a vectorial representation of the 6 DOF rigid-body kinetics, like the one found in Fossen (2011, eq. 3.42), given in (2.7).

$$\begin{gathered}
\dot{\boldsymbol{\eta}} = \mathbf{J}_\mathbf{q}(\boldsymbol{\eta})\boldsymbol{\nu} \\
\mathbf{M}_{rb}\dot{\boldsymbol{\nu}} + \mathbf{C}_{rb}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}_{rb}
\end{gathered} \tag{2.7}$$

Here $\mathbf{J}_\mathbf{q}$ is given by Equation (2.26),

$$\boldsymbol{\tau} = \left[F_x, F_y, F_z, l, m, n\right]^T \tag{2.8}$$

is the force- and torque vector.

$$\mathbf{M}_{rb} = \begin{bmatrix} m \cdot I_{3\times3} & -m \cdot S(\mathbf{r}_{cg}^b) \\ -m \cdot S(\mathbf{r}_{cg}) & I_{cg} \end{bmatrix} \tag{2.9}$$

$$\tag{2.10}$$

is the rigid body inertia matrix, whereas

$$\mathbf{C}_{rb}(\nu) = \begin{bmatrix} 0 & -m \cdot S(\boldsymbol{\nu}_1) - m \cdot S(\boldsymbol{\nu}_2)S(\mathbf{r}^b_{cg}) \\ -m \cdot S(\boldsymbol{\nu}_1) + m \cdot S(\mathbf{r}^b_{cg})S(\boldsymbol{\nu}_2) & -S(\mathbf{I}_b\boldsymbol{\nu}_2) \end{bmatrix} \tag{2.11}$$

$$\tag{2.12}$$

is one possible parametrization of the coriolis-centripetal matrix. See Fossen (2011, eq. 3.55-57)

Another UAV model is the linear model by Etkin and Reid (1996), which is restated in Equation (B.2).

2.1.4 Mapping from elevon to aileron-elevator

As the X8 has an elevon configuration, whereas most models are expressed with ailerons and an elevator, it is necessary to convert the control signals between the two (Beard and McLain, 2012). This is a simple, linear mapping showed below. Due to this mapping it makes little sense to talk about aileron and elevator deflection on the X8 in degrees, and they are therefore scaled to a range from -1 to 1.

$$\begin{bmatrix} \delta_e \\ \delta_a \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \delta_{er} \\ \delta_{el} \end{bmatrix} \tag{2.13}$$

2.2 Aerodynamics

This section will go through some common representations of the aerodynamic coefficients that are present in the aforementioned UAV models. First, the linear approximation of Beard and McLain (2012) is presented, followed by an approximation for higher angles of attack.

2.2.1 Parametrization of the aerodynamical coefficients

The aerodynamical coefficients C_i are in general nonlinear, multidimensional surfaces. However, in order to simplify the analysis, they are often parameterized in more linear terms(Beard and McLain, 2012).

$$\begin{aligned} C_D(\alpha, q, \delta_e) &\approx C_D(\alpha) + C_{D_q}\frac{c}{2V_a}q + C_{D_{\delta_e}}\delta_e \\ C_L(\alpha, q, \delta_e) &\approx C_L(\alpha) + C_{L_q}\frac{c}{2V_a}q + C_{L_{\delta_e}}\delta_e \\ C_Y(\beta, p, r, \delta_a, \delta_r) &\approx C_Y(\beta) + C_{Y_p}\frac{b}{2V_a}p + C_{Y_r}\frac{b}{2V_a}r + C_{Y_{\delta_a}}\delta_a + C_{Y_{\delta_r}}\delta_r \end{aligned} \tag{2.14}$$

Similarly, (2.1) can be parametrized by making the following assumptions on the moment coefficients:

$$\begin{aligned} C_l(\beta,p,e,\delta_a,\delta_r) &= C_l(\beta) + C_{l_p}\frac{b}{2V_a}p + C_{l_r}\frac{b}{2V_a}r + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r \\ C_m(\beta,p,e,\delta_a,\delta_r) &= C_m(\alpha) + C_{m_q}\frac{b}{2V_a}q + C_{m_{\delta_e}}\delta_e \\ C_n(\beta,p,e,\delta_a,\delta_r) &= C_n(\beta) + C_{n_p}\frac{b}{2V_a}p + C_{n_r}\frac{b}{2V_a}r + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r \end{aligned} \tag{2.15}$$

If an even more simplified, linear model is sought, the simplification of (2.16) can be made. In general a more nonlinear model is more accurate, but this assumes that all the coefficients are known. Identification of the coefficients in a nonlinear model is far more complex than for the linear case.

$$C_L(\alpha) \approx C_{L_0} + C_{L_\alpha}\alpha \tag{2.16a}$$

$$C_Y(\beta) \approx C_{Y_0} + C_{Y_\beta}\beta \tag{2.16b}$$

$$C_l(\beta) \approx C_{l_0} + C_{l_\beta}\beta \tag{2.16c}$$

$$C_m(\alpha) \approx C_{m_0} + C_{m_\alpha}\alpha \tag{2.16d}$$

$$C_n(\beta) \approx C_{n_0} + C_{n_\beta}\beta \tag{2.16e}$$

A similar approximation can be done with the drag coefficient, but it is also very common to express drag as a sum of parasitic drag C_{D_p} and lift-induced drag. The lift-induced drag is approximated with the fraction in Equation (2.17), where e is the Oswalds efficiency factor and Æ is the aspect ratio.

$$C_D(\alpha) \approx C_{D_0} + C_{D_\alpha}\alpha \approx C_{D_p} + \frac{(C_{L_0} + C_{L_\alpha})^2}{\pi e \text{Æ}} \tag{2.17}$$

2.2.2 Flat plate model

Whereas the linear model has proved to be accurate for low angles of attack, it becomes invalid for angles of attack higher than the stall angle. A model that depicts lift and drag reasonably well in the stalled region, particularly for $\alpha > 60°$ (Stengel, 2004), is the flat plate model. It is a very simplified model, as it is purely geometrical and does not take the airfoil camber into account.

According to Anderson (1989), the flat plate model[1] is based on the assumption that the normal force on a surface, which by Newton's 2nd law is equal to the time derivative of momentum from particles hitting the surface, is equal to the product of mass flow of the particles and the change in the velocity's normal component. See Figure 3.1.

$$N = (\rho V_a S_{wing}\sin(\alpha))(V_a\sin(\alpha)) = \rho V_a^2 S_{wing}\sin^2(\alpha) \tag{2.18}$$

[1]Also known as the Newtonian flow model, as the principles were published by Newton in the second book of *Principia*(Anderson, 1989).

From this, the nondimensional normal force coefficient is found:

$$C_N = \frac{N}{\frac{1}{2}\rho V_a^2 S_{wing}} = 2\sin^2(\alpha) \tag{2.19}$$

From Figure 3.1 it is clear that $L = N\cos(\alpha)$ and $D = N\sin(\alpha)$, resulting in the following flat plate lift and drag coefficients:

$$\begin{aligned} C_{D,flatplate}(\alpha) &= 2\,\text{sign}(\alpha)\sin^3(\alpha) \\ C_{L,flatplate}(\alpha) &= 2\,\text{sign}(\alpha)\sin^2(\alpha)\cos(\alpha) \end{aligned} \tag{2.20}$$

where $\text{sign}(\alpha)$ is included to extend the model to negative values of α. Unfortunately, the flat plate model fails to predict the behavior for small angles of attack. This makes sense since for small angles of attack the flow is attached, such that the shape of the airfoil affects the sufficiently. However, once the airfoil stalls, the flow is no longer attached to the airfoil, thereby less influenced by it.

2.3 Rotations between frames of reference

As the UAV model involves forces acting in different reference frames and rotations about different axes, it is important to have an understanding of how forces can be translated from one frame to another. There are several different ways this can be acheived, with Euler angles, rotation matrices and unit quaternions being the most common. However due to the singularity and low computational efficiency of the Euler angles, as well as the over-parameterized rotation matrices, this section will focus on the unit quaternions. For a thorough review of rotations in general, see e.g. Egeland and Gravdahl (2002) or Fossen (2011).

Unit quaternions

A unit quaternion is a complex number, with one real part and three imaginary parts, of unit length.

$$\mathbf{q} = \begin{bmatrix} \eta \\ \boldsymbol{\varepsilon} \end{bmatrix} \in \mathcal{Q} = \left\{ \mathbf{q} : \mathbf{q}^T\mathbf{q}, \mathbf{q} = [\eta, \boldsymbol{\varepsilon}]^T, \eta \in \mathcal{R}, \boldsymbol{\varepsilon} \in \mathcal{R}^3 \right\} \tag{2.21}$$

If the unit quaternion $\mathbf{q}$ represents a rotation from i to frame j, then the vector $\mathbf{v}^i$ can be rotated into frame j by the following operation:

$$\mathbf{v}^j = \mathbf{q} \otimes \mathbf{v}^i \otimes \mathbf{q}^{-1} \tag{2.22}$$

Here $\mathbf{q}^{-1} = [\eta, -\boldsymbol{\varepsilon}]^T$, and the three dimensional vector $\mathbf{v}$ has been converted into a quaternion with real part equal to zero.The quaternion rotation applies the quaternion product $\otimes$, defined as follows (Egeland and Gravdahl, 2002):

$$\mathbf{u} \otimes \mathbf{v} = \begin{bmatrix} \eta_\mathbf{u}\eta_\mathbf{v} - \boldsymbol{\varepsilon}_\mathbf{u}^T\boldsymbol{\varepsilon}_\mathbf{v} \\ \eta_\mathbf{u}\boldsymbol{\varepsilon}_\mathbf{v} + \eta_\mathbf{v}\boldsymbol{\varepsilon}_\mathbf{u} + \boldsymbol{\varepsilon}_\mathbf{u} \times \boldsymbol{\varepsilon}_\mathbf{v} \end{bmatrix} \tag{2.23}$$

Since the notation of quaternion rotation is arguably less intuitive, (2.22) is often rewritten as a rotation matrix (Fossen, 2011).

$$\mathbf{v}^j = \mathbf{R}(\mathbf{q})\mathbf{v}^i = \begin{bmatrix} 1-2(\varepsilon_2^2+\varepsilon_3^2) & 2(\varepsilon_1\varepsilon_2-\varepsilon_3\eta) & 2(\varepsilon_1\varepsilon_3+\varepsilon_2\eta) \\ 2(\varepsilon_1\varepsilon_2+\varepsilon_3\eta) & 1-2(\varepsilon_1^2+\varepsilon_3^2) & 2(\varepsilon_2\varepsilon_3-\varepsilon_1\eta) \\ 2(\varepsilon_1\varepsilon_3-\varepsilon_2\eta) & 2(\varepsilon_2\varepsilon_3+\varepsilon_1\eta) & 1-2(\varepsilon_1^2+\varepsilon_2^2) \end{bmatrix} \mathbf{v}^i \tag{2.24}$$

Finally, the transformation from angular velocity to a quaternion attitude representation is needed to express the change in the attitude with time.

$$\dot{\mathbf{q}} = \mathbf{T}_q(\mathbf{q})\boldsymbol{\omega}^b_{b/n} = \frac{1}{2}\begin{bmatrix} -\varepsilon_1 & -\varepsilon_2 & -\varepsilon_3 \\ \eta & -\varepsilon_3 & \varepsilon_2 \\ \varepsilon_3 & \eta & -\varepsilon_1 \\ -\varepsilon_2 & \varepsilon_1 & \eta \end{bmatrix} \boldsymbol{\omega}^b_{b/n} \tag{2.25}$$

Combining Equations (2.24) and (2.25), it is possible to state the relationship between NED and body frame where $\mathbf{J_q}$ of Equation (2.7) is defined according to Equation (2.26).

$$\mathbf{J_q} = \begin{bmatrix} \mathbf{R}(q) & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{T}(q) \end{bmatrix} \tag{2.26}$$

2.4 Wind model

In order to simulate the UAV in a world-like environment, a realistic wind model is needed. Wind is typically modeled with a steady part, defined in NED frame, and a gust part defined in body frame. The steady part is usually modeled as constant in NED frame, or as a function of height to mimic the wind shear (MathWorks, 2015d). It is given by an absolute value, as shown in Equation (2.27), and a direction.

$$V_{w_{steady}} = W_{20}\frac{\ln\left(\frac{h}{z_0}\right)}{\ln\left(\frac{20}{z_0}\right)} \tag{2.27}$$

Here W_{20} is the wind speed at 20 feet, which for light, moderate and severe turbulence typically is 15, 30 and 45 knots respectively. h is the altitude of the UAV, while z_0 is a constant depending on the flight phase that should be set to 0.15 feet for terminal flight phase. The gust part is modeled as a stochastic process, specifically; white noise passed through a linear shaping filter. The linear time-invariant filters given by the von Kármán turbulence model (Diedrich and Drischler, 1957) are known to fit well with experimental results (Stengel, 2004). However, as stated by Beard and McLain (2012), these filters can not be represented by rational transfer function. Therefore the von Kármán model is approximated by the Dryden model (Liepmann, 1952).

2.4.1 The Dryden turbulence model

The Dryden model consists of six transfer functions, H_u,H_v,H_w,H_p,H_q,H_r, that define how the linear and rotational velocities u,v,w,p,q,r, of the UAV are affected by wind gusts. There exits many different adaptations of the Dryden model, depending on the flight conditions. In the following, the definition from U.S. Department of Defense (1980), is used. Other good sources are Beard and McLain (2012); MathWorks (2015b). This model, valid for altitudes below 1000 feet and low speeds, is given in Equation (2.28).

$$\begin{aligned}
H_u(s) &= \sigma_u\sqrt{\frac{2L_u}{\pi V_a}}\frac{1}{1+\frac{L_u}{V_a}s} & H_p(s) &= \sigma_p\sqrt{\frac{0.8}{V_a}}\frac{\frac{\pi}{4b}^{\frac{1}{6}}}{L_w^{\frac{1}{3}}\left(1+\frac{4b}{\pi V_a}s\right)} \\
H_v(s) &= \sigma_v\sqrt{\frac{L_v}{\pi V_a}}\frac{1+\frac{\sqrt{3}L_v}{V_a}s}{\left(1+\frac{L_v}{V_a}s\right)^2} & H_q(s) &= -\frac{\frac{s}{V_a}}{\left(1+\frac{4b}{\pi V_a}s\right)}\cdot H_w(s) \\
H_w(s) &= \sigma_w\sqrt{\frac{L_w}{\pi V_a}}\frac{1+\frac{\sqrt{3}L_w}{V_a}s}{\left(1+\frac{L_w}{V_a}s\right)^2} & H_r(s) &= \frac{\frac{s}{V_a}}{\left(1+\frac{3b}{\pi V_a}s\right)}\cdot H_v(s)
\end{aligned} \tag{2.28}$$

The parameters of Equation (2.28) are defined in U.S. Department of Defense (1980), and are restated in equations 2.29-2.30 for completeness.

$$L_w = h \qquad L_u = L_v = \frac{h}{(0.177+0.000823h)^{1.2}} \tag{2.29}$$

$$\sigma_w = 0.1W_{20} \qquad \sigma_u = \sigma_v = \frac{1}{(0.177+0.000823h)^{0.4}} \tag{2.30}$$

The linear velocities of the wind gusts can then be expressed in the frequency domain as a product of the transfer function and white noise: $i_{w_g}(s) = H_i(s)w(s)$ for $i \in \{u, v, w\}$. By converting the i_{w_g}'s to time frame, the linear wind velocity in body frame can be expressed as Equation (2.31). Similarly the rotational influence of the wind gust is given by $k_{w_g}(s) = H_k(s)w(s)$ for $k \in \{p, q, r\}$.

$$\mathbf{V}_w^b = \begin{bmatrix} u_w \\ v_w \\ w_w \end{bmatrix} = \mathbf{R}_{NED}^b \begin{bmatrix} w_{n_s} \\ w_{e_s} \\ w_{d_s} \end{bmatrix} + \begin{bmatrix} u_{w_g} \\ v_{w_g} \\ w_{w_g} \end{bmatrix} \tag{2.31}$$

From this, the relative velocities are found:

$$\begin{aligned}
u_r &= u - u_w & p_r &= u - p_{w_g} \\
v_r &= v - v_w & q_r &= q - q_{w_g} \\
w_r &= w - w_w & r_r &= r - r_{w_g}
\end{aligned} \tag{2.32}$$

Part II

Methods

Chapter 3

Model

A good model forms the basis for a UAV simulator and is important in model based control and estimation. This section will explain how an aerodynamic model of an existing, physical UAV can be created, by identification of the aerodynamic coefficient and by choosing an appropriate model structure. Then the model is extended to better fit the nonlinear, high angle of attack region. The final model structure is stated in Equation (B.4), which has been put in the appendices for reasons of space.

3.1 From a physical UAV to aerodynamic coefficients

To find the aerodynamic coefficients of the Skywalker X8 is an operation involving many steps. Details are found in Appendix D, but a summary of the steps is given below.

1. **Make a 3D surface** of the UAV. This is done by importing a 3D scan into a CAD program, after it has been converted to the proper format.
2. **Extract the airfoil** shape from the intersection between the 3D surface and different planes throughout the wing. Convert these curves to a dat file.
3. **Import airfoils to XFLR** for CFD analysis. Reconstruct the UAV using the airfoils and Table E.1.
4. **Perform a static XFLR analysis**. This gives $C_L(\alpha)$, $C_Y(\beta)$ and $C_D(\alpha)$ within the linear flight region.
5. **Perform a stability XFLR analysis** by in turn perturbing the elevons as elevators and ailerons. This will result in the linear coefficients of Equation (B.2).

XFLR is an easy-to-use computational fluid dynamics (CFD) program used for airfoil analysis, valid within the linear flight region. An explanation of XFLR and its assumptions is given in Appendix C.

3.2 Choosing model structure

As explained in Section 2, there are many different UAV models available. This section will present a the model that has been chosen for this thesis, and will explain the reasoning behind the choice.

The different variations of the model given in Beard and McLain (2012) all have the same basic structure, but vary in how they approximate the nonlinear functions $C_i(\alpha, \beta, p, q, r, \delta_j)$. When all these nonlinear functions are identified correctly, the full nonlinear model Equation (2.6) depicts the behavior of the UAV in a good manner. However, the identification process becomes very complex. The linear models of Equation (B.2) and Equation (2.6) with the linear coefficients of equations 2.14, 2.15 and 2.16, on the other hand, can more easily be identified using linear airfoil theory; XFLR outputs the complete six degrees-of-freedom(DOF), linear model in the form of Equation (B.2) directly. The downside is that these models becomes invalid outside the linear region, i.e. for large values of α and β. Since this thesis both needs the model to be identifiable and correct in the high-α region, a compromise between the two extrema is made by basing the X8 model on Equation (2.6) with linear lift, drag and side force as given by Equations (2.14), (2.16a), (2.16b), (2.17), and linear moment coefficient as given in Equations (2.15) and (2.16c-2.16e). Then this model is extended, as explained in the next section.

3.3 Model extension

Even though the aforementioned model covers the most important aerodynamic effects on a UAV, some effects are neglected. This section will explain some of them, how they can be identified, and why they have been included back into the model. To sum up the model, this section also includes an overview of the aerodynamic coefficients used through the rest of this thesis.

3.3.1 Drag depending on sideslip

Many other aerodynamic models (see e.g. Stevens and Lewis (2003)) include a portion of the drag component that is proportional to the sideslip angle β. This is also very natural from a physical perspective, since a sideslip angle results in a larger projected area towards the wind. Since XFLR provides an easy way of analyzing the drag over a range of β, $C_{D_{\beta^2}}$ is easily found by fitting the data from XFLR to a second order model using the Matlab function `polyfit`. This is included partially because the data is readily available, and partially because increasing sideslip might be a way to increase drag before landing.

$$C_D(\beta) \approx C_{D_{\beta^2}} \beta^2 + C_{D_\beta} \beta \tag{3.1}$$

3.3.2 Nonlinear extension to lift and drag

In trying to get both the simplicity of the linear model and the high angle of attack validity of the nonlinear model, e.g. Pointner and Kotsis (2011); Beard and McLain (2012) merge the linear model of Equation (2.16) with the flat plate model from Equation (2.20).

$$\begin{aligned} C_D(\alpha) &\approx C_{D_p} + (1-\sigma(\alpha))\frac{\left[C_{L_0} + C_{L_\alpha}\alpha\right]^2}{\pi e \mathcal{R}} + \sigma(\alpha)\left[2\,\mathrm{sign}(\alpha)\sin^3(\alpha)\right] \\ C_L(\alpha) &\approx (1-\sigma(\alpha))\left[C_{L_0} + C_{L_\alpha}\alpha\right] + \sigma(\alpha)\left[2\,\mathrm{sign}(\alpha)\sin^2(\alpha)\cos(\alpha)\right] \end{aligned} \tag{3.2}$$

Here $\sigma(\alpha) \in [0,1]$, given by Equation (3.3), is a sigmoid function with cutoff at the stall angle $\pm\alpha_0$ and transition rate M, used to smoothen the transition from the linear- to the flat plate model. However, as the flat plate model is not exact, this method often requires adjustment to real world data (Moore et al., 2014).

$$\sigma(\alpha) = \frac{1 + e^{-M(\alpha-\alpha_0)} + e^{M(\alpha+\alpha_0)}}{\left(1 + e^{-M(\alpha-\alpha_0)}\right)\left(1 + e^{M(\alpha+\alpha_0)}\right)} \tag{3.3}$$

3.3.3 Flat plate pitch moment

Due to the importance of a nonlinear $C_m(\alpha)$, as will become apparent in Section 5.3, this section presents a possible nonlinear extension of the pitching moment coefficient valid for high angles of attack, inspired by the flat plate theory extension of Equation (3.2).

To find the pitch moment for the entire UAV, the normal force, represented by the normal force coefficient of Equation (2.19), is integrated over a flat plate representation of the surface. For each section in y, as illustrated by Figure 3.1, the contribution to the pitch moment is given by Equation (3.4).

$$M_{\Delta y} = -\int_{LE(y)}^{TE(y)} \bar{q} C_N (x - x_{cg}) dx \tag{3.4}$$

where we integrate from the leading edge to the trailing edge, and where $\bar{q}$ is the dynamic pressure.

The total moment is found by integrating over all y-sections, remembering that the leading- and trailing edges are functions of y. See Figure 3.2.

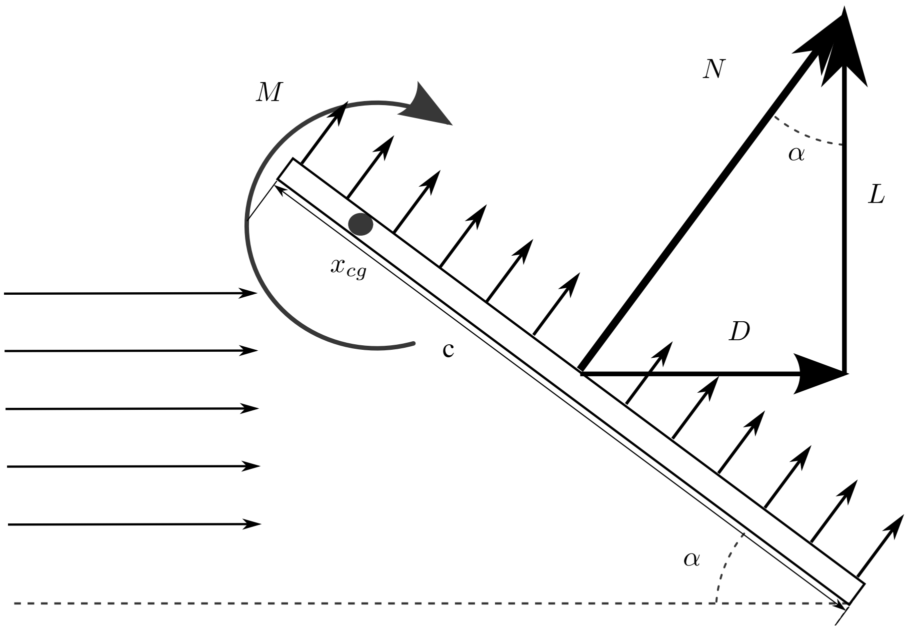


Figure 3.1: X8 with x_{cg} coordinate

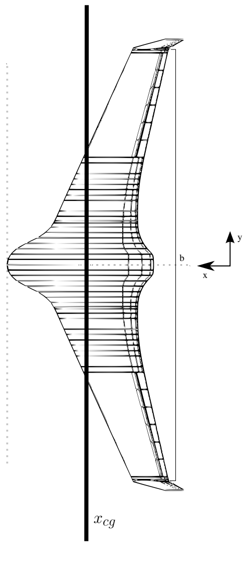


Figure 3.2: X8 with x_{cg} coordinate

$$
\begin{aligned}
M &= \int_{-\frac{b}{2}}^{\frac{b}{2}} M_{\Delta y} dy && (3.5)\\
&= 2\int_{0}^{\frac{b}{2}} M_{\Delta y} dy && (3.6)\\
&= -2\int_{0}^{\frac{b}{2}} \int_{LE(y)}^{TE(y)} \bar{q} C_N (x - x_{cg}) dx dy && (3.7)\\
&= -2 C_N \bar{q} \int_{0}^{\frac{b}{2}} \left[\frac{1}{2}x^2 - x x_{cg}\right]_{LE(y)}^{TE(y)} dy && (3.8)\\
&= -C_N \bar{q} \int_{0}^{\frac{b}{2}} TE(y)^2 - 2TE(y)x_{cg} - LE(y)^2 + 2LE(y)x_{cg} dy && (3.9)\\
& && (3.10)
\end{aligned}
$$

Exploiting that $TE = LE + c$ yields

$$
M = -C_N \bar{q} \int_{0}^{\frac{b}{2}} 2LE(y)c(y) + c(y)^2 - 2c(y)x_{cg} dy \tag{3.11}
$$

For straight wings, the chord is constant and without loss of generality it can be assumed that the leading edge is located at $x = 0$. Then the pitching moment coefficient about the $\frac{1}{4}c$ line is:

$$
\begin{aligned}
M &= -C_N \bar{q} \int_{0}^{\frac{b}{2}} c^2 - 2c x_{cg} dy \\
&= -C_N \bar{q} \frac{b}{2} \left(c^2 - 2c \cdot \frac{c}{4}\right) \\
&= -C_N \bar{q} \frac{bc^2}{4}
\end{aligned} \tag{3.12}
$$

From $C_m = \frac{M}{\bar{q}Sc}$ this yields:

$$
C_{m,flatplate,straight} = -\frac{1}{2}\operatorname{sign}(\alpha)\sin^2(\alpha) \tag{3.13}
$$

However, as the flat plate essentially calculates the area moment of the wings, it is import to consider the sweep of the wing. To find an approximation of the pitching moment for the X8 the integral of Equation (3.11) is rewritten as a sum so that the leading edge and chord length data from Table E.1 can be applied. The definition of the pitch moment coefficient then results in the following

$$\begin{aligned} C_{m,flatplate} &= \frac{M}{\bar{q}Sc} \\ &= \frac{2\sin^2(\alpha)}{S}\sum_{y=0}^{\frac{b}{2}}(2LE(y)c(y) + c(y)^2 - 2c(y)x_{cg})\Delta y \\ &= \frac{2\sin^2(\alpha)}{S}\sum_{y=0}^{\frac{b}{2}}(2LE(y)c(y) + c(y)^2 - 2c(y)x_{cg})\Delta y \end{aligned} \tag{3.14}$$

From this it is apparent that the flat plate pitch moment coefficient can be written as

$$C_{m,flatplate} = C_{m,fp}\sin^2(\alpha) \tag{3.15}$$

where $C_{m,fp}$ is a constant determined by the shape of the wing.

By arguing that the shape of the airfoil has more influence on the aerodynamic forces and moments for low angles of attack, whereas the flat plate is a decent approximation for larger angles, the linear model is blended with the flat plate model in a similar manner as with Equation (3.2).

$$C_m(\alpha) = (1 - \sigma(\alpha))[C_{m_0} + C_{m_\alpha}\alpha] + \sigma(\alpha)\left[C_{m_{fp}}\operatorname{sign}(\alpha)\sin^2(\alpha)\right] \tag{3.16}$$

3.4 Final model structure

The final model structure that will be used throughout this thesis is stated in Equation (B.4) for completion.

3.5 Adaptation of coefficient

Since the coefficients of Equation (B.4) are slightly different than what is given by XFLR, some analysis is needed to get the desired results. As can be seen from the XFLR analysis output given in Listing E.1, the coefficients from the XFLR stability analysis are given in body coordinates. Luckily both the side force coefficients C_{Y_*}, and all the moment coefficients C_{l_*},C_{m_*} and C_{n_*} are the same for both body and stability frame. While $C_L(\alpha)$ and $C_D(\alpha)$ are given by the regular XFLR analysis, the remaining lift and drag coefficients can be found from the body frame coefficients by utilizing the relationship:

$$\begin{bmatrix} C_D \\ C_L \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) \\ -\sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} -C_x \\ -C_z \end{bmatrix} \tag{3.17}$$

Chapter 4

Simulator

Once the basis for a mathematical UAV model has been made, the equations can be put into a simulation framework to provide a realistic pilot training platform, and can also be used for hardware-in-the-loop testing. In addition, simulators are important in design of complex airplane systems(Devesa et al., 2004). The simulator can identify bugs that have to be fixed, and potential problem areas that has to be further tested in future physical tests. Specifically in the case of high-α landing of the X8, a good simulator is needed to test the robustness of the controller prior to actual flight tests. The implementation of the simulator is split in two stages, which is reflected in the structure of this section. First, a Matlab/Simulink model is presented in Section 4.1. For this thesis, this simulator will be important in the initial testing of the model, while in the long run it will provide an important framework for early-stage control and estimation development. After this simulator has been thoroughly tested, an implementation-near simulator is implemented. This simulator is described in Section 4.2.

4.1 Simulink simulator

The initial simulator was created in Matlab/Simulink to facilitate the experimentation with different setups, and to easily adjust the model parameters. It is also important to have a good simulator in Simulink/Matlab since these tools are very common in early stages of controller development. In the longer run, this simulator framework will also be used to simplify the comparison of the simulated outputs from the mathematical model with actual data from X8 flight logs.

4.1.1 Simulink structure

This section will describe how the Simulink model is structured. The general consideration in the implementation has been to let Simulink take care of the signal flow, while Matlab handles the computations. Examples from MathWorks (2015c); Beard and McLain (2012) have been used as inspiration. Figure 4.1 shows the top level architecture with a guidance module that feeds the autopilot with setpoints based on the position and attitude of the X8, as well as a more long term goal such as reaching a waypoint or following a path. The autopilot then seeks to reach these setpoints by sending control commands to the X8. These commands cause a response in the X8, changing its position and attitude. To close the loop, the new position and attitude are fed back to the guidance and autopilot modules, as well as being sent to FlightGear(Section 4.1.2) for visualization. It should be noted that the guidance and autopilot modules are only included to yield an overview of how the X8 model would fit in a typical control system design. A simple PID controller in height, airspeed and course angle has been implemented based on Beard and McLain (2012), but this has not been the focus of this work. The structure of the PID controller can be seen in Figure G.1. Since the purpose of the simulator at this stage has been to experiment with different configurations, and find the resemblance with the X8, the majority of the tests have been performed using a simple USB interfaced RC joystick[1]. Through the `Control select` block, the input source is selected to be either the autopilot, the joystick or simply flying at trim conditions, as calculated in Section 4.1.3.

Actuator models

Within the `X8` module, shown in Figure 4.2, the demanded control is first fed through an actuator model as depicted in Figure 4.3. Here the dynamic and static limitation of the actuators are taken into account by limiting their magnitude and rate of change, using rate limited and saturated second-order integrators similar to Prasad B. and Pradeep (2007)

$$\frac{\delta_{er}}{\delta_{er,command}} = \frac{\omega_0^2}{s^2 + 2\zeta\omega s + \omega_0^2} \tag{4.1}$$

As in Prasad B. and Pradeep (2007), $\omega_0 = 100$ rad/s and $\zeta = \frac{\sqrt{2}}{2}$. Additionally, the rate limit and saturation of the servos are set to ± 3.4907 rad/s[2] and $(-30°, 35°)$[3] respectively.

It is important to note that while both the commanded control from the autopilot and the input to the aerodynamic model is in elevator, rudder and aileron deflection, the physical limitations are on the elevon control surfaces of the X8. Therefore the control signal has to be mapped back and forth from the elevon representation, using Equation (2.13).

The motor dynamics are modeled by a simple first order model according to Equation (4.2),

[1]The E-Sky 0905A

[2]This corresponds to $60°$ in 0.30 seconds. This is based on the delay of a typical mini servo of 0.10 seconds and an assumed control surface delay of 0.20 seconds

[3]Based on measurements on the X8

where n is the motor speed and $\tau = 0.2$ is the time constant.

$$\frac{n}{n_{command}} = \frac{\frac{1}{\tau}}{s + \frac{1}{\tau}} \tag{4.2}$$

Forces and Dynamics submodules

Both the `Forces` and the `Dynamics` modules are wrappers for Matlab functions. This allows for implementation of more complex mathematics in an arguably more intuitive way than through Simulink blocks. Here the `MATLAB Function` block is chosen over the `Interpreted MATLAB Function` since it is compiled rather than interpreted, resulting in a faster simulator. In addition it is capable of handling Simulink Buses, which makes for an arguably cleaner and more intuitive structure, as opposed to vectors of data and muxing/demuxing. Both `airdata` [4] and `states`[5], are implemented as Simulink Buses. In the `Forces` module the aerodynamic, gravitational and propeller forces/-torques, as well as elements of the `airdata` bus, are calculated. Forces and torques are passed through the `Dynamics` block. This block implements the 6 degrees-of-freedom rigid body kinetic model using unit quaternions to represent the attitude, as given by Equation (2.7). Quaternions are used to avoid the singularity experienced at $\theta = 90^\circ$ when using Euler angles. Unfortunately, since the current Matlab-FlightGear interface assumes that attitude is given in Euler angles, the quaternion is transformed to Euler angles, a transformation that is invalid at the aforementioned singularity.

Wind submodule

As UAVs operate in many different wind conditions, the simulator should facilitate a realistic environment with respect to wind. This is achieved by using the Dryden wind model for body frame wind gusts, affecting both linear and rotational velocities, combined with a wind shear model for simulation of steady NED frame winds. See Section 2.4. Luckily both the Dryden wind model and the wind shear model are already implemented in the Matlab Aerospace toolbox.

4.1.2 FlightGear

Both in debugging of the simulator and in validation of the model, it is of utmost importance to visualize the response of the X8. Even though general XY plots can provide useful information about the value of certain variables, a full 3D view from a flight simulator strengthens the use of physical institution in the debugging and validation process. The open source flight simulator FlightGear (Basler et al., 2015) was chosen to visualize the output from the simulator. XPLANE, another popular flight simulator was also considered.

[4]The `airdata` bus includes the norm of the relative velocity V_a, α,β and the body wind velocities

[5]The `states` bus includes NED position, attitude quaternion, attitude Euler angles, accelerations, linear and angular velocities

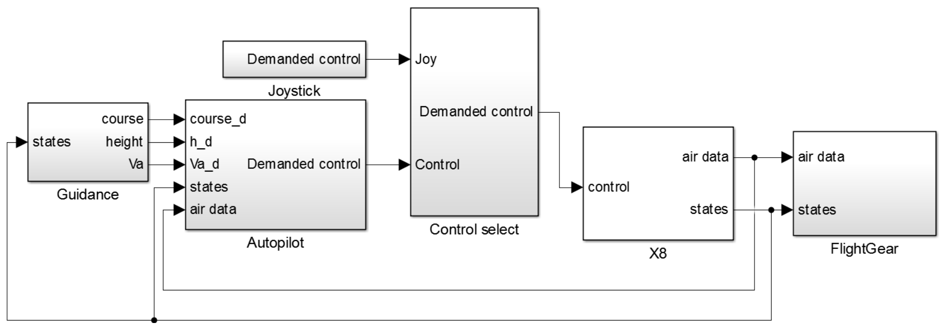


Figure 4.1: The top level structure of the control system simulator

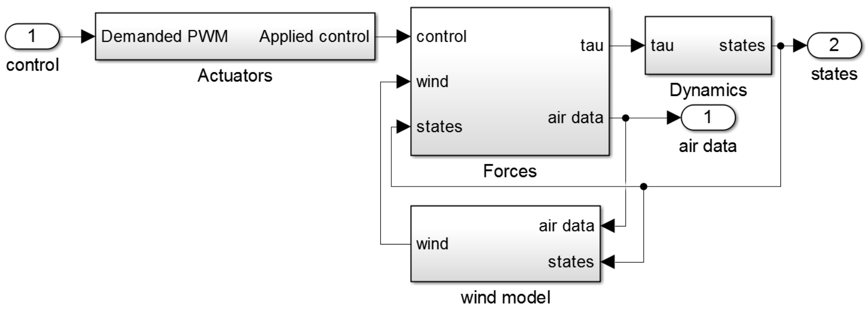


Figure 4.2: Structure of the X8 block

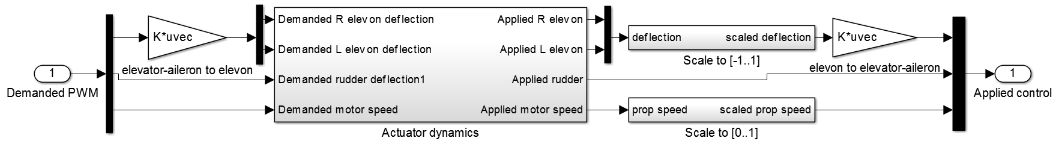


Figure 4.3: The actuator model block

However, FlightGear was favoured due to its open source policy, enabling full configuration of the source code if needed. In its original configuration, FlightGear simulates flights by interacting with a separate simulator(JSBSim), but to gain full overview over the simulations this feature is overridden by passing position and heading [6] information directly from Simulink to FlightGear over UDP. The command used to start FlightGear from the command line is found in Appendix G.1.1.

As the Simulink simulator not necessarily runs at real time, a `Set Pace` block from the Aerospace toolbox is added to the simulator. This ensures that one second in the simulator corresponds to one second in FlightGear, i.e. the real world.

4.1.3 Calculating trim conditions

As many traditional airplane controllers rely on the trim conditions, a scheme to calculate these are made. It has also proved useful to have the trim conditions when experimenting with the simulator. By forcing the UAV to fly at the trim conditions during the first few seconds of the flight, a proper startup is ensured. While MathWorks (2015a); Beard and McLain (2012) explains this flawlessly, only the outlines are presented here.

First, a reduced version of the Simulink simulator is made. This should have the controls δ_i as input. The outputs are set to V_a, α and β. Then, by specifying what internal states, outputs and inputs that should be constant, and what states that should have constant derivatives, the Matlab `trim` function finds the trim conditions. The derivatives of the down position is locked to $-V_a \sin(\gamma)$, while the derivative of the quaternion and the derivative of the linear and rotational velocities are set to zero. None of the inputs are fixed, while $V_a = 18$m/s and $\beta = 0°$. The `trim` function will then find the equilibrium point of the dynamic system that is closest to the initial guess of trim conditions, given by an arbitrary initial height, an initial velocity $u = V_a = 18$m/s and a quaternion corresponding to level, forward flight.

Then both the calculated trim conditions and the reduced Simulink model can be sent in as arguments for the Matlab `linmod` function. This function linearizes the model around the equilibrium point and returns the resulting A,B,C and D matrices to form a linear state space model.

4.2 Software-in-the-loop simulator

After some testing, the lessons learned from flying the Simulink simulator was exploited to include the mathematical model in a hardware-in-the-loop(HIL)/software-in-the-loop(SITL) simulator. This step is important since HIL/SITL is much closer to the real flight of the

[6]FlightGear requires position to be given in latitude, longitude and height, and attitude to be given in roll, pitch and yaw angles.

UAV, the code being essentially the same. A proper HIL/SITL simulator will also help improve the mathematical model itself, since an experienced UAV pilot can give important feedback as to how the behavior of the simulated UAV differs from the UAV.

SITL versus HIL

A SITL and a HIL simulator are very similar. They both run the actual autopilot code that would run on the UAV during real flight, and they both rely on a Flight Dynamics Model (FDM) to simulate the interactions with the real world. The only principal difference is that while for HIL, the autopilot is running on its intended hardware platform, for SITL all hardware, corresponding software, and hardware-software interaction, is emulated in software. This thesis focus on the development of SITL. However, as there already exist a HIL simulator based on the same principles as the SITL, once the JSBSim configuration file has been created for SITL, running HIL is a matter of plugging in the hardware and starting the right scrips. The reason for choosing to work with SITL is that HIL introduces more delay to the system, which often requires very different PID gains from what is used in real flight.

All that is needed to transform this generic UAV simulator to an X8 simulator is a configuration file for JSBSim. In order to run SITL the following software has to be started:

1. The actual autopilot code. The NTNU UAV-lab use the ArduPlane autopilot, further explained in Section 4.2.1.
2. A Flight Dynamics Model that calculates the forces acting on the UAV, see Section 4.2.2
3. Visualization. Here, either FlightGear (Section 4.1.2) or Neptus (Section 4.2.4) can be used.

This thesis involves two slightly different SITL setups; a pilot-centric setup, used for model validation and pilot practice, and a mission-centric setup, used in further control system development. They are both built around the Ardupilot autopilot, and both use JSBSim as FDM. These similarities will be explained first, followed by sections that point out the differences.

4.2.1 ArduPlane

ArduPlane (ArduPilot, 2015) is a popular, open-source autopilot for fixed-wing UAVs, and is the primary autopilot used at the UAV-lab. It is a part of the Ardupilot family that among other things makes running SITL with a flight dynamics model easy.

Mavproxy

The Ardupilot family also involves a simple, command line based ground control station called Mavproxy. From within Mavproxy the modes of the autopilot can be changed, the

PID can be tuned on-line, waypoints can be set and configuration files can be loaded. Mavproxy was chosen due to its simplicity and intuitive command line interface, however more demanding ground control stations such as the APM Planner could have been used.

4.2.2 Flight dynamics model

Testing the autopilot is not useful unless it is interfaced with a proper flight dynamics model (FDM) that can simulate the behavior of a UAV in real flight is needed. By default, the ArduPlane autopilot is configured to interact with an FDM program over UDP ports 5501 and 5502. A variety of FDM programs exists, including CRRCSim, Last_letter and JSBSim.

JSBSim[7] was chosen since it is the most common, and currently the most extensive, FDM program for ArduPlane. Each JSBSim aircraft configuration contains models of mass balance, ground reactions, propulsion, aerodynamics, buoyant forces, external forces, atmospheric effects and/or gravity. These models are defined in user configured XML files, that can be described in both body- and wind axes. The standard JSBSim for ArduPlane comes with only one configuration file; the Rascal110. So in order to make a SITL solution for the X8, a new JSBSim configuration file had to be created. The Rascal110, as well as other airplane models from the JSBSim web page, was used as a basis for the model. The aerodynamic coefficients used in the JSBSim file are exactly the same as those used in the Simulink simulator, with the exception of the motor parameters. JSBSim already has a preconfigured way to parameterize motors, with the use of table lookup, that differ from the motor model implemented in Simulink, as seen in Equation (2.6). However, the JSBSim web page include a simple script, Aeromatic, that calculates the necessary tables and motor parameters based on the motor power rating, the propeller diameter and maximum RPM ratings. Data for the motor on the X8 can be found in Table H.1.

To make the X8 JSBSim configuration file, the configurations of the Rascal110 were gradually changed with the corresponding configurations for the X8. At the same time, the PID controller of the ArduPlane was retuned to fit the new flight characteristics. The tuned parameters of the X8 PID were then stored so that they can be loaded by the autopilot, through Mavproxy.

4.2.3 DUNE

The software framework for the X8, and many other platforms at the UAV-lab, is based on DUNE[8](University of Porto; Underwater Systems and Technology Laboratory, 2015a). DUNE is a portable, runtime environment for simplifying the inclusion and running of generic embedded C++ code on-board a vehicle, such as a UAV. The DUNE philosophy is to split the large operations to be performed into small, effective, generally interrupt triggered tasks. Typical DUNE tasks are control, communication and I/O access.

[7]JSB are the initials of the creator of JSBSim, Jon S. Berndt

[8]DUNE is a recursive acronym for DUNE Unified Navigation Environment

The communication between different tasks, both within the same UAV, between multiple UAVs/AUVs, and from a UAV to a ground station, is based on the Inter-Module Communication protocol (IMC). In addition to streamlining the process of communication, IMC also simplifies logging and timestamping.

4.2.4 Mission-centric software-in-the-loop simulator

Both the mission-centric and the pilot-centric SITL simulators use the aforementioned programs. However, due to the different needs, they vary in how they visualize the simulation. The use of DUNE, and the configuration of ArduPlane, is also different.

The mission-centric SITL will typically be used during implementation and testing of a new controller scheme. The controller will be implemented as a DUNE task, another existing DUNE task will communicate with the ArduPlane autopilot for low level control, and a third task communicates with the ground control station Neptus, used for visualization and issuing commands for the UAV to follow. The commands used to start the mission-centric SITL simulator, as well as a minimal DUNE configuration file needed for interaction with ArduPlane, are given in Appendix G.3.

Neptus

Another feature of the DUNE-family is the Neptus graphical ground station used for mission planning, execution, review and analysis (University of Porto; Underwater Systems and Technology Laboratory, 2015b). The UAV can e.g. be commanded to fly a pre-planned path that has been drawn on a map within the Neptus interface, while live log data such as attitude and height can be displayed. Neptus also includes an extensive framework for analysis and playback of log files.

4.2.5 Pilot-centric software-in-the-loop simulator

Since the primary purpose of the pilot-centric SITL simulator is pilot practice and model testing, it is usually run with ArduPlane in manual mode. This means that ArduPlane merely forwards the input RC values from the RC stick to the servos. For HIL the RC stick can easily be connected to the hardware platform that is running ArduPlane, but for SITL this is a slightly more involved process where the RC values needs to be polled from the stick and sent to ArduPlane over UDP port 5501. This is solved by another existing DUNE task.

For visualization the pilot-centric SITL relies on the first-person view of FlightGear, as explained in Section 4.1.2. The commands used to start the pilot-centric SITL simulator are given in Appendix G.2.

Chapter 5

Optimal landing

This section will explain how the X8 can land in an optimal manner by first declaring what is defined as optimal. Then it is explained how this can be achieved, both physically and in terms of controller design. Finally a simulation setup, where different landing scenarios can be tested and compared, is implemented.

5.1 What is optimal?

In order to land the X8 in an optimal manner, it is necessary to define *optimal* by declaring objectives that should be met. Such objectives could include e.g. low power consumption or low risk, but for simplicity, and to cut the problem to its core, only the following objectives are considered in this thesis:

Landing in the desired position UAVs involved in operations in remote locations are often faced with limited landing space and little established infrastructure for landing, such as a runway. For this reason arrest systems, such as nets or airbags[1], are used to safely land the UAVs. Regardless of the category landing spot, whether it is in an arrest system or a on conventional runway, it is of utmost importance that landing occurs exactly in that spot.

Flyable, safe path For the path to cohere, it must be flyable and safe (Tsourdos et al., 2011). Flyable refers to the fact that the path should satisfy the dynamic and kinematic constraints of the UAV, e.g.its maximum pitch rate. A safe path is one avoiding obstacles. Since collision avoidance is beyond the scope of this thesis, it is assumed that there are no obstacles in the near vicinity of the UAV and the net.

[1]The airbag is located on the ground and is inflated, making it soft to land on top of.

Low velocity For a conventional landing, reduced speed allows for landing on shorter runways. When landing in some arrest system, the motivation for minimizing the impact speed is to reduce the structural strain on the UAV from the arrest system.

The two first objectives mentioned above are simple, in the sense that they are fulfilled by simply following a straight line into the net. Therefore the remainder of this section is focused on how to achieve a low impact speed for the UAV, while keeping the other two objectives in mind.

5.2 Achieving low terminal speed

Since velocity is closely related to kinetic energy, reducing the impact velocity boils down to reducing the kinetic energy of the UAV at the point of impact. This can be achieved by either transforming some kinetic energy into potential energy, or by increasing the dissipation of energy by increasing the drag.

5.2.1 Increasing the potential energy

A brief look at the energy balance of the UAV reveals that the decrease in speed is rather insignificant for the case considered. By assuming that the X8 flies along the surface at trim speed ($h_i = 0$ meters,$v_i = 18$m/s), and that the net is positioned at $h_f = 5$ meters, Equation (5.1) shows that the speed is only reduced to $v_f \approx 15$m/s.

$$\begin{aligned} \frac{1}{2}mv_i^2 + mgh_i &= \frac{1}{2}mv_f^2 + mgh_f \\ v_f &= \sqrt{v_i^2 - 2gh} \approx 15\text{m/s} \end{aligned} \tag{5.1}$$

The velocity decrease is small due to the quadratic velocity term in the formula for kinetic energy. However it is worth noting that if the initial speed already has been lowered, the additional decrease by converting kinetic energy into potential is larger. Therefore the importance of potential energy is pointed out, before looking into other measures for minimizing the impact speed.

5.2.2 Increasing drag

When it comes to dissipating the kinetic energy by increasing drag, the solutions can be divided into two categories: Increased drag from special control surface configurations, and increased drag caused by special maneuvers.

The most common control surface configurations used to decrease speed are:

Speed brakes are hidden inside a slot, usually on the topside of the wing, during normal flight. When engaged, a small, vertical wall pop up, disturbing the airflow and creating extra drag. This also requires some extra control in pitch, since it also will influence the moment balance of the UAV.

Deceleron is an aileron that can be split in two to act as an air brake. The two parts can also work together as a normal aileron (Neihouse and Lee, 1951). See Figure 5.1.

Flaps are control surfaces on the trailing edge of the wing. Lowering them lead to an increase in parasitic drag and an increase in drag.

Crow/butterfly configuration controls the flaps of the airplane down while the ailerons are used as a flap in the upward direction, leading to an increase in drag. See Figure 5.2.

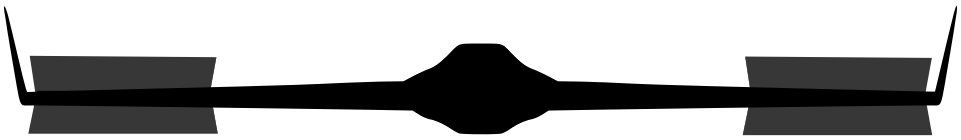


Figure 5.1: Deceleron configuration

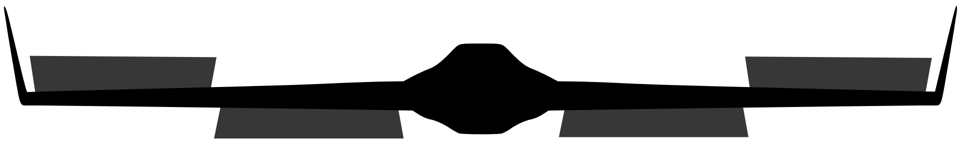


Figure 5.2: Crow configuration

However, as these techniques require structural changes, they are not considered any further in this thesis. Furthermore, due to the small area of the control surfaces, they generally also cause less drag than some of the maneuvers presented next.

Forward slip is a landing technique used general aviation when the plane comes in too high for landing. By applying left rudder and right aileron, usually followed by right rudder, left aileron, the drag is increased by increasing sideslip as the plane zig-zags towards the runway.

Deep stall landing utilizes the large increase in drag experienced in the high angle of attack region, leading to a rapid descent rate. See Mathisen et al. (2015); Taniguchi (2008); Pointner and Kotsis (2011). See Figure 5.3.

Perched landing is inspired by the landing of birds. It also operate in the high angle of attack region, but is characterized by a rapid pitching motion towards the end of the flight. This leads to an abrupt increase in drag, loss in lift, and an almost ballistic flight (Glassman et al., 2012; Moore et al., 2014). See Figure 5.3.

Since the model was extended to include $C_{D_{\beta^2}}$ in Section 3.3.1, forward slip could have proved a viable landing technique also in the simulator. However, as the X8 has no rudder, inducing a sideslip will be very difficult. The high angle of attack landing methods will be investigated more thoroughly in the next section, when the physics behind deep stall and perching is explained.

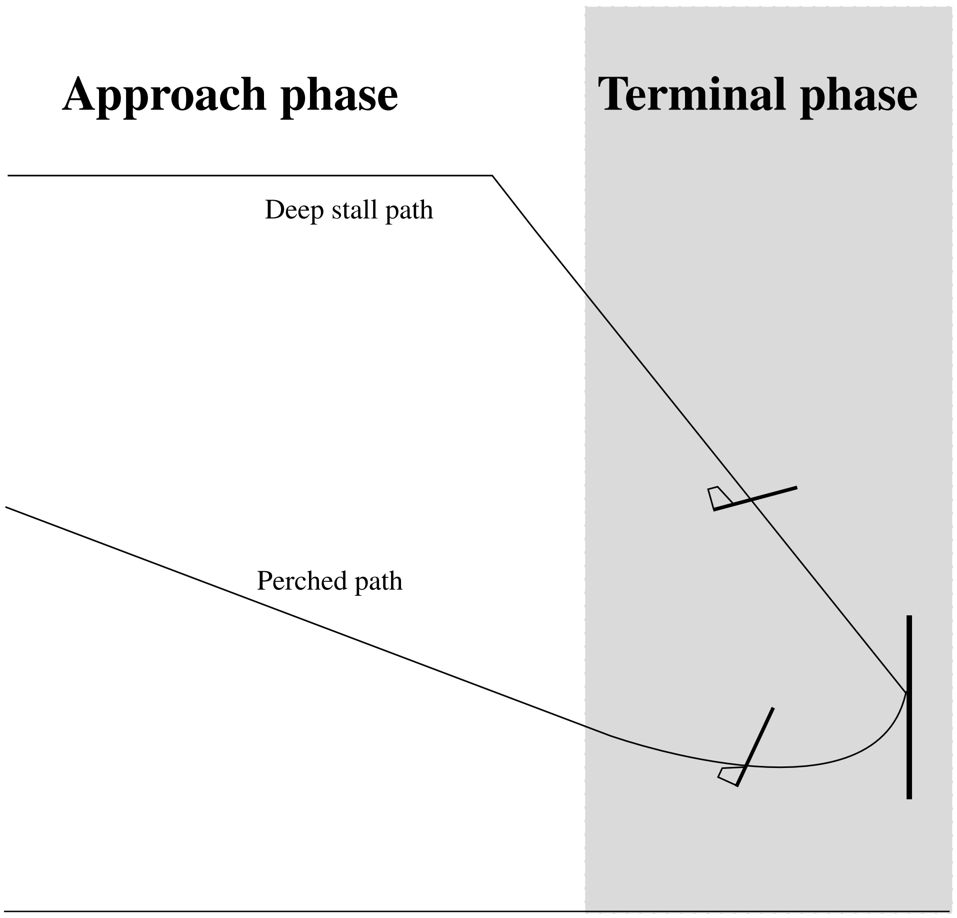


Figure 5.3: Deep stall and perched landing (Paths are not to scale)

5.3 Characteristics of the high angle of attack region

As mentioned in the previous section, both perched and deep stall landing operate in the high angle of attack region. In this section, the characteristics of the high angle of attack region will be explained; both its advantages and its drawbacks. Some of these characteristics are highly nonlinear, thus adding complexity to the mathematical model. These model expansions, as given in Section 3.3, are also explained.

In order to understand both the desirabilities and the dangers of the high angle of attack landing techniques, it is necessary to understand a few basic aerodynamical principles. This section will first explain the ideal stall by looking at typical curves for lift and drag coefficients. Then some of the risks will be presented, from a more applied point of view.

As showed in Equation (B.4), the dynamics of an UAV are heavily dependent on the angle of attack. Up until this point little has been said about how the aerodynamical coefficients change with the angle of attack. For illustrative purposes plots of fictive, but reasonable, lift, drag and pitching moment coefficients are depicted in figures 5.4. These curves are different for different wing configurations[2], but some general characteristics can be pointed out. By looking at Figure 5.4 it becomes apparent that

- The lift increases linearly with α up to stall, then a substantial loss of lift is encountered before a second peak in lift is reached.
- The pitching moment decrease linearly with α up to stall, then it increases to a small peak before it decreases again.
- Drag increases with α. The increase is substantially larger for large α

This shows that the linear approximation in Equation (2.16) fit well with the data for small α. However, the high-α region require more analysis.

As the elevator moves up, the C_m-curve is shifted up, see Figure 5.5. This will cause the C_m-curve to have three zero-crossings, causing the UAV to have three equilibrium points in pitch. Due to the slope of the curve, the first and third are stable while the second is an unstable equilibrium[3]. During deep stall, the UAV seeks the third equilibrium point. This illustrates the importance of a nonlinear parameterization of $C_m(\alpha)$ explained in Section 3.3.3, since that will require far less control actuation to stabilize the plane at a large alpha.

Dangers of high angle of attack flight

The dangers of high-α flight is highly dependent on the configuration of the UAV. However, this section will present the main risk of high-α flight for the X8, mainly based on (Stengel, 2004, Chapter 7.4). If only the longitudinal aerodynamics are considered, high-α flight only have a few concerns:

Less effective control surfaces When the wing stalls, the flow separates from the wing leading to less flow across the control surfaces, and consequently less effective control surfaces. This can be somewhat mitigated if the control surfaces are located in the slipstream of the propeller, thus increasing the airflow.

Deep-stall lock occurs when the effect of the elevator is too low compared to the amplitude of the "hump" in the C_m curve. Consider an UAV operating at the third equilibrium on the pitch moment curve, i.e. at about $\alpha = 38^\circ$ on the green curve in

[2]Different airfoils, mass balance, sweep, dihedral etc.

[3]Assuming $\dot{\alpha} \approx -q$ an increase in α will lead to a negative pitch moment, which again will lead to a decrease in α.

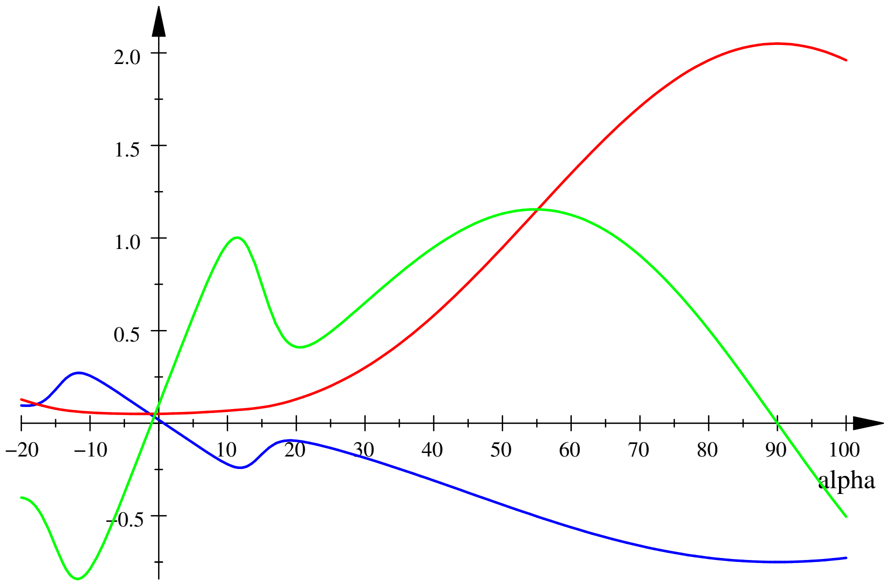


Figure 5.4: General lift(green), drag(red) and pitching moment(blue) coefficients for large α, adapted from (Stengel, 2004, fig. 7.4-1,p. 751)

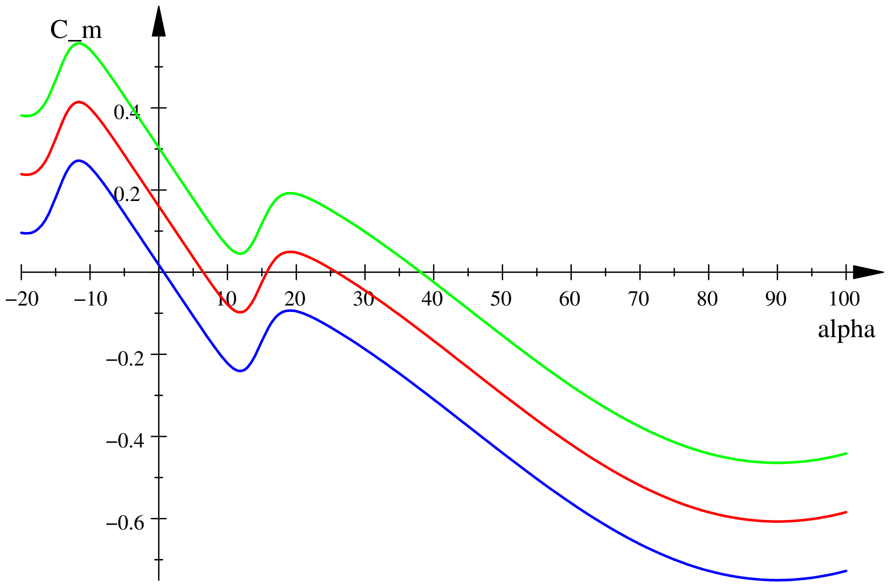


Figure 5.5: Shifting of the pitching moment coefficient

Figure 5.5. If the effect of the elevator is lower than the peak of the C_m-curve, the UAV can not be controlled out of deep stall.

Seeing these two effects as a whole, it apparent that they could amplify each other; the reduced efficiency of the stalled elevator makes it impossible to conquer the C_m peak.

When the lateral model is considered as well, the situation is not so fortunate. If one wing has a slightly larger angle of attack than the other, e.g. from construction asymmetries or due to wind, the stall will be asymmetric. The wing with the highest angle of attack will stall first, and will experience the drop in lift and increase in drag, leading to a wing drop and yaw towards the dropping wing. If not corrected, the UAV will enter a spin. In a steady spin, the center of mass falls along a vertical helical path with constant fall and yaw rate, with the nose pointing towards the center. Spins are classified as either flat or steep, upside-down or straight, depending on the orientation of the UAV. Some aircraft are known to be *impossible* to recover from spin [4]. From this it should be apparent that asymmetric stall should be avoided at all cost, and since recovery can be difficult the focus of the control system should be on spin departure resistance and early recovery NATO (1976).

5.4 Optimal landing trajectory in the high angle of attack region

The rest of this thesis will look further into perched landing. The reason for this is that, due to all the dangers involved with high-α flight, it is interesting to investigate if a late, but abrupt stall yields a landing involving less risk. The primary concern of this thesis is to find an optimal landing trajectory, and then to control the UAV towards the landing point in an optimal manner.

5.4.1 Problem setup

In order to establish a setup for the computation of the optimal landing trajectory, it is necessary to define what scenarios that should be further investigated. The long term goal is to design a complete landing autopilot that takes the UAV optimally from normal flight, into an arrest system at a low velocity. However, due to the aforementioned dangers involved with high-α flight, it is key to keep risk at an acceptable low level. This is the focus of Gryte et al.(Submitted) (see appendix A.2) where development of a two-phase landing approach, that considers the risk before going in to the final, high-α phase, is introduced. This thesis is a first step on the way towards the two-phased approach, where only the final phase is considered, see Figure 5.3. It is assumed that the UAV is able to get to a low altitude in the near vicinity of the landing target. At $t = 0$ the UAV enters the second phase at a distance Δ meters from the landing target, at an altitude of h_0 meters,

[4]NATO (1976) reports that instructions for the P-111 aircraft is to eject if spin is not recovered before reaching 15 000 ft

flying in trimmed conditions. The focus of this thesis is then to minimize the velocity on impact with the landing target. The cases considered in the following assume a net that is positioned in the aft of a ship, 5 meters above the sea level. As you can approach them from any side, airbags are considered more robust against changes in wind direction Gautam et al. (2014). However by assuming that the ship is weathervaning[5], thus heading into the wind, the amount of crosswinds are minimized and are not considered. Without loss of generality it is assumed that the mean wind direction is from the north, and that the ship on which the UAV is landing is heading north. In order to investigate the optimal trajectory, it is of interest to let the UAV operate freely with as little constraints as possible. The constraints are summarized in the following list:

- Constraints are put on the position to hold the UAV above the surface, and to make it hit the net.
- Constraints to make the UAV to enter the net from the correct side.
- The control inputs are constrained to be within the physical limitations of the UAV.
- Since UAVs generally are designed for belly landings, the final attitude is constrained to the equivalent of $80^\circ \leq \theta \leq 100^\circ$. To achieve this, and at the same time avoid the Euler angle singularity at $\theta = 90^\circ$, quaternions are used for attitude representation.
- The trajectory is constrained to the dynamics of the UAV, in order to make the path flyable.

The objective of the optimization problem is to minimize the velocity at impact, more specifically the relative velocity between the UAV and the net. Assuming that the net is stationary, the relative velocity is equivalent with the NED velocity of the UAV. Additionaly the variation in control inputs should be minimized, to ensure flyable paths and to reduce wear and tear on the control surfaces.

5.4.2 Continuous-time optimal control

As this thesis investigates optimal landing trajectories, an introduction to optimization is needed. This section gives a brief introduction to the continuous-time optimal control problem(OCP), as given in e.g. Diehl et al. (2002, 2009). The general continuous-time OCP can be stated as follows:

$$\min_{\mathbf{x}(\cdot), \mathbf{u}(\cdot)} \quad \int_{t_0}^{t_0+T} L(\mathbf{x}(t), \mathbf{u}(t))dt + E(\mathbf{x}(T)) \tag{5.2}$$

[5] A common technique in dynamically positioned ships to reduce the power consumption by heading into the main direction of the environmental forces, see Fossen (2011)

Subject to:

$$\mathbf{x}(t_0) - \mathbf{x}_0 = 0 \tag{5.3a}$$
$$\dot{\mathbf{x}}(t) - f(\mathbf{x}(t), \mathbf{u}(t)) = 0 \quad \forall t \in [t_0, t_0 + T] \tag{5.3b}$$
$$h(\mathbf{x}(t), \mathbf{u}(t)) \leq 0 \quad \forall t \in [t_0, t_0 + T] \tag{5.3c}$$
$$r(\mathbf{x}(T)) \leq 0 \tag{5.3d}$$

Equation (5.2) represents the cost function, where L is the integral cost contribution and E corresponds to the terminal cost. The minimization problem is subject to the constraints in Equation (5.3), where Equation (5.3a) and Equation (5.3d) represents the constraints on initial and terminal values respectively. Equation (5.3b) constrains the path to the dynamics of the system, given by the right-hand side of the ordinary differential equation (ODE), $f(\mathbf{x}(t), \mathbf{u}(t))$. Finally Equation (5.3c) represents the path constrains of the system, including the upper and lower limits on $\mathbf{x}(t)$ and $\mathbf{u}(t)$.
There are several different approaches to solving Equation (5.2)(Diehl et al., 2006). While state-space approaches, including the Hamilton-Jacobi-Bellman equation, have methods to compute solution approximation, they are limited to small state-spaces since the size of the problem scales badly with the dimension of the state space. The *indirect* approaches are based on the calculus of variations. They use Pontryagins maximum principle to describe the necessary optimality conditions for optimal control in continuous time, and use this to eliminate the control inputs from the problem. This results in a boundary value problem that can be solved numerically. For this reason the indirect approaches are referred to as "optimize, then discretize".

Direct approaches, on the other hand, are referred to as "discretize, then optimize"; they discretize the continuous time problem into a nonlinear programming problem (NLP) of finite dimension. Today the direct approaches are the most popular, since they easily handle inequality constraints (Diehl et al., 2006). The three most important direct approaches are direct collocation, direct single and direct multiple shooting.

Direct single shooting

In order to discretize the system, the direct shooting methods divide the control function $\mathbf{u}(t)$ into N constant control parameters $\mathbf{q}_i \in \mathcal{R}^{n_u}$ such that $\mathbf{u}(t;\mathbf{q}) = \mathbf{q}_i \forall t \in [t_i, t_{i+1}]$. Then the controls $\mathbf{u}(t;\mathbf{q})$ and the initial state $\mathbf{x}_0$ are forward integrated over the entire horizon $t \in [0, T]$ to produce the dependent state variables $\mathbf{x}(t)$, denoted as $\mathbf{x}(\mathbf{q};t)$. The path constraints are also discretized, usually over the same grid as $\mathbf{u}(t)$. Consequently the OCP (5.2)-(5.3) is translated into the NLP (5.4)-(5.5), which can be solved by a dense NLP solver.

$$\min_{\mathbf{q} \in \mathcal{R}^{N \cdot n_u}} \quad \int_0^T L(\mathbf{x}(t;\mathbf{q}, \mathbf{u}(t;\mathbf{q}))dt + E(\mathbf{x}(T;\mathbf{q})) \tag{5.4}$$

Subject to:

$$h(\mathbf{x}(t_i;\mathbf{q}), u(t_i;\mathbf{q})) \leq 0 \qquad \forall i \in [0, N-1] \tag{5.5a}$$

$$r(\mathbf{x}(T;\mathbf{q})) \leq 0 \tag{5.5b}$$

Direct multiple shooting

Direct multiple shooting also starts with discretization of the control function. However instead of solving one ODE for the entire horizon, it solves one on each discrete interval simultaneously, and is therefore categorized as a "simultaneous" approach (Diehl et al., 2009). To facilitate this, the artificial initial values $\mathbf{s}_i$ are introduced. The initial value problem

$$\begin{aligned} \dot{\mathbf{x}}_i(t;\mathbf{s}_i,\mathbf{q}_i) &= f(\mathbf{x}_i(t;\mathbf{s}_i,\mathbf{q}_i),\mathbf{q}_i) \quad \forall t \in [t_i, t_{i+1}] \\ \mathbf{x}_i(t_i;\mathbf{s}_i,\mathbf{q}_i) &= \mathbf{s}_i \end{aligned} \tag{5.6}$$

can then be solved to obtain $\mathbf{x}_i(t;\mathbf{s}_i,\mathbf{q}_i)$, a piece of the trajectory. This leads to the introduction of the continuity constraint Equation (5.9b), forcing the terminal condition of one interval to coincide with the initial condition of the next. In a similar manner the integral cost is discretized and computed for each interval, as showed in Equation (5.7).

$$l_i(\mathbf{s}_i,\mathbf{q}_i) = \int_{t_i}^{t_{i+1}} L(\mathbf{x}_i(t_i;s_i,q_i),q_i)dt \tag{5.7}$$

This causes the total cost integral to change to a sum over all interval costs, $l_i(\mathbf{s}_i,\mathbf{q}_i)$. The resulting NLP is seen in (5.8)-(5.9), and should be solved by a sparsity exploiting NLP solver.

$$\min_{\mathbf{s},\mathbf{q}} \quad \sum_{i=0}^{N-1} l_i(\mathbf{s}_i,\mathbf{q}_i) + E(\mathbf{s}_N) \tag{5.8}$$

Subject to:

$$\mathbf{x}_0 - \mathbf{s}_0 = 0 \tag{5.9a}$$

$$\mathbf{x}_i(t_{i+1};\mathbf{s}_i,\mathbf{q}_i) - \mathbf{s}_{i+1} = 0 \qquad \forall i \in [0, N-1] \tag{5.9b}$$

$$h(\mathbf{s}_i,\mathbf{q}_i) \leq 0 \qquad \forall i \in [0, N] \tag{5.9c}$$

$$r(\mathbf{s}_N) \leq 0 \tag{5.9d}$$

Direct multiple shooting have better local convergence properties than direct single shooting, especially for open-loop unstable systems. Another advantage is that the entire state trajectory can be initialized, through the artificial initial condition variable s. Finally, direct multiple shooting makes it easier to include hard constraints in the state variables, as they are included as optimization variables(Mathisen et al., 2015).

Direct collocation

Direct collocation is also a simultaneous, direct approach, however it discretizes both state and control. This is accomplished by approximating the path by a polynomial p_k of order k, instead of integrating the system forward. This polynomial should fit the actual dynamics of the system for all the points $(t_k^{(1)}, t_k^{(m)})$ on the interval $[t_k, t_{k+1}]$. Then the continuation constraints of Equation (5.3b) can be replaced by a set of equality constraints $c_k(s_k, v_k, q_k) = 0$, where v_k is the coefficients of the polynomial.

However, as stated by Diehl et al. (2002), direct multiple shooting has many advantages over collocation. The most prominent being that *adaptive discretization error control* needs regridding, which will increase the dimensions of the NLP, see Diehl et al. (2006).

5.4.3 Continuous-time optimal control problem formulation

This section will form the aforementioned problem setup description into a mathematical continuous-time optimal control problem, by combining the direct multiple shooting NLP in equation 5.8-5.9 with UAV model from Equation (B.4). To simplify the analysis, while focusing on the core of the stall landing, the problem is reduced to a three degrees-of-freedom longitudinal formulation. This includes rotation about the body frame y-axis, and translational motion along the body z- and x-axes. The direct multiple shooting approach was chosen due to the advantages pointed out in Section 5.4.2.

The state vector used in the OCP is given by

$$\mathbf{x} = \begin{bmatrix} N & D & u & w & \varepsilon & \eta_1 & \eta_2 & \eta_3 & q & \alpha & v_N & v_D \end{bmatrix}^T \tag{5.10}$$

with the corresponding initial condition

$$\mathbf{x}_0 = \mathbf{s}_0 = \begin{bmatrix} 0 & 5 & 18 & 0 & 0.9997 & 0 & 0.0231 & 0 & 0 & 0 & 17.9806 & -0.8349 \end{bmatrix}^T \tag{5.11}$$

This coincides with the initial condition constraint of Equation (5.9a). It should be noted that while including velocities both in NED and in body frame increases the dimensions of the problem, thus its complexity, it facilitates constraining the UAV to enter the net from the south side without putting unnecessary constraints on the attitude. In the event that it is optimal to fly backwards into the net, this option should not be ruled out by design considerations. In addition, the control vector is given in the following:

$$u(t) = q_i = \begin{bmatrix} \delta_{e,i} & \delta_{t,i} \end{bmatrix}^T \quad \text{for } t \in [t_i, t_{i+1}] \tag{5.12}$$

The continuation constraint of Equation (5.9b) is enforced by first propagating the inital state $\mathbf{s}_i$, control variable $\mathbf{q}_i$ and current wind through the system equations (2.7) and (B.4). Through numerical integration by the Runge-Kutta method of 5th order[6], the state at the

[6]For information on the Runge-Kutta method of 4th order, as well as other methods for numerical integration see e.g. Egeland and Gravdahl (2002)

end of interval i, $\mathbf{x}_i\left(t_{i+1};\mathbf{s}_i,\mathbf{q}_i\right)$, is found. The initial state for the next interval, s_{i+1}, is then constrained to equal $\mathbf{x}_i\left(t_{i+1};\mathbf{s}_i,\mathbf{q}_i\right)$ for all $i \in [0, N-1]$.

To comply with the limitations on the controls of the UAV are set according to equations 5.13a-5.13b. Similarly, the path of the UAV is constrained to above the surface according to Equation (5.13c)[7]. All these constraints are easily converted to the form $h\left(\mathbf{s}_i,\mathbf{q}_i\right) \leq 0$ of Equation (5.9c).

$$-1 \leq \delta_{e,i} \leq 1 \tag{5.13a}$$

$$0 \leq \delta_{t,i} \leq 1 \tag{5.13b}$$

$$-15 \leq D \leq -1 \tag{5.13c}$$

The final set of constraints are the terminal constraints on the form of Equation (5.9d), to ensure that the UAV land in the net, approaching from the south side with the belly towards the net. Specifically the final position should be $N = 30$,$D = -5$, with a non-negative north velocity.

$$\begin{aligned} N_N &= 30 \\ D_N &= -5 \\ 0 \quad &\leq v_{N,N} \leq 40 \end{aligned} \tag{5.14}$$

As we are only considering the three longitudinal degrees-of-freedom, the attitude constraint can be simplified to:

$$0.6428 \leq \varepsilon_{2,N} \leq 0.7660 \tag{5.15}$$

In addition to the constraints, a cost function should be added to the problem to ensure that the solution is optimal in some manner, and not just feasible. It should be apparent that the cost function should reflect the objectives defined in Section 5.1. However, as most of these objectives are fulfilled by the constraints, the cost function for this problem is rather small, including only the terminal velocity and the difference between consecutive control values. The differences between consecutive control values are minimized to reduce the strain on the control surfaces and to make sure it is flyable. This results in the following cost function:

$$\min_{\mathbf{s},\mathbf{q}} \quad \sum_{i=0}^{N-1} R_1\left(\mathbf{q}_{i+1} - \mathbf{q}_i\right) + R_2\sqrt{v_{N,N}^2 + v_{D,N}^2} \tag{5.16}$$

This concludes the general setup needed to find the optimal landing trajectory for a UAV. In Chapter 7 some specific landing scenarios are presented, which will enforce minor changes to the preceding problem.

[7]The height is limited above to 15 meters above ground in order to reduce the problem size, and below to 1 meter above ground to have some distance to the surface.

5.5 Control in the high angle of attack region

After an optimal landing trajectory is established, means to follow this trajectory are investigated and implemented. As mentioned in Section 1.2, the literature reports many different control strategies for landing UAVs. This thesis follows an approach similar to Moore et al. (2014), where an landing trajectory is found by solving a continuous-time optimal control problem. In Moore et al. (2014), optimal trajectories from many different starting points are calculated in advance. These paths are subsequently tracked using a time-varying linear quadratic controller (LQR), choosing the path closest to the current position of UAV to increase the local convergence of LQR. Similarly to Mathisen et al. (2015), this thesis is using Nonlinear Model Predictive Control (NMPC) in the high-α region. However, while Mathisen et al. (2015) is using NMPC to track a reference in path angle, this thesis is using NMPC to recalculatethe entire optimal trajectory for each timestep to account for e.g. changes in wind.

5.5.1 Nonlinear Model Predictive Control

The main motivation for using NMPC to track the optimal landing trajectory is that the development of the NMPC overlaps a lot with finding the optimal trajectory; in principle NMPC recalculates the optimal trajectory in every time step. In addition, NMPC has the advantage that it handles the nonlinearities experienced in high-α flight well. The physical constraints of the UAV control surfaces can easily be added. In addition, other constraints can be added to experiment with how the UAV behaves under different levels of freedom.

Finally some disadvantages with NMPC should be mentioned. There is no point in a UAV controller that can not run on UAV-applicable hardware, so the controller should be able to run in real-time. For this reason Moore et al. (2014) abandon the NMPC, in favor of LQR-trees. However no real-time considerations are made in this thesis, and it should be stressed that the primary interest is in finding the optimal trajectory. Having established this optimal trajectory, it can be tracked using another type of controller should the NMPC approach prove to be unfit for real-time applications. Another disadvantage with using NMPC on this problem is that, due to the non-convex nature of this problem, the optimal solution is not guaranteed to be global. Advanced solvers, such as IPOP, are aware of this problem, and try to mitigate it. Nevertheless, the problem should be recognized.

For the implementation of this NMPC, a receding horizon NMPC have been chosen. This way, each iteration of the NMPC can easily be warm started by guessing that the optimal trajectory is the same from one step to the next. The details of the implemented algorithm are outlined below. The startup is exactly the same as for Section 5.4.3, but instead of finding the optimal trajectory once, the following steps are performed N times.

1. Find the optimal trajectory as in Section 5.4.3.
2. Apply the first element of the calculated optimal control vector.
3. Shorten the optimization horizon by one. This naturally also leads to a reduction in continuation constraints (Equation (5.9b)) equal to the number of states, and a

reduction in path constraints (Equation (5.9c)) by three[8].

4. Simulate/measure states and wind. This is done by the Runge-Kutta method of 4th order. The NED wind is held constant, while the Dryden wind model is used for gusts in body x and z direction. The Dryden model is implemented as a state space model.
5. Lock the initial condition of the next iteration $\mathbf{s}_0$ to the simulated/measured state.

It should be noted that the optimal trajectory calculated in each step assumes that the current experienced wind will remain constant throughout the horizon, so some changes in the calculated path is expected when the wind has a large gust component.

5.6 Simulation setup in CasADi

In order to simulate the UAV landing, a simulation setup using CasADi (Andersson, 2013) is developed. CasADi is an open-source symbolic framework for numerical optimization and automatic differentiation. It is written in C++, but has a python interface with similar performance. CasADi and its accompanying python interface is chosen since it is a simple platform to get started with optimization, despite its highly efficient implementation.

5.6.1 IPOPT

In addition to many self-developed solvers, CasADi also includes integration with the third party solver “Interior-Point OPTimizer” (IPOPT). Along with active-set sequential quadratic programming (SQP) methods Interior-point is the primary method for solving large nonlinear optimization problems (Nocedal and Wright, 2006). However interior-point methods are generally more attractive than active set methods for problems with many inequality constraints, which is why it is chosen for this problem. IPOPT is a primal-dual interior-point filter line-search method. A complete understanding of the internals of IPOPT is beyond the scope of this thesis, but can be found in Wächter and Biegler (2006). However, the main steps are drafted below for a system on the form given by equations 5.17 - 5.18, based on Nocedal and Wright (2006).

$$\min_{\mathbf{x},\mathbf{s}} \quad f(\mathbf{x}) \tag{5.17}$$

[8]Two constraints corresponding to the reduction of the control vector, and one corresponding to the D component of the state vector.

Subject to:

$$c_E(\mathbf{x}) = 0 \tag{5.18a}$$
$$c_I(\mathbf{x}) - s = 0 \tag{5.18b}$$
$$s \geq 0 \tag{5.18c}$$
$$\tag{5.18d}$$

Here, $\mathbf{x}$ and $\mathbf{s}$ represent the variables and slack variables, while $c_E(\mathbf{x})$ and $c_I(\mathbf{x})$ represent the equality and inequality constraints of the problem, respectively. Through application of Newtons method on the Karush-Kuhn-Tucker (KKT) conditions, a system of linear equations are obtained:

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L} & 0 & -A_E^T(\mathbf{x}) & -A_I^T(\mathbf{x}) \\ 0 & \mathbf{Z} & 0 & \mathbf{S} \\ A_E^T(\mathbf{x}) & 0 & 0 & 0 \\ A_I^T(\mathbf{x}) & -\mathbf{I} & 0 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_s \\ p_y \\ p_z \end{bmatrix} = - \begin{bmatrix} \nabla f(\mathbf{x}) - A_E^T(\mathbf{x})\mathbf{y} - A_I^T(\mathbf{x})\mathbf{z} \\ \mathbf{Sz} - \mu\mathbf{e} \\ c_E(\mathbf{x}) \\ c_I(\mathbf{x}) - \mathbf{s} \end{bmatrix} \tag{5.19}$$

Where $\mathcal{L}$ is the Lagrangian, $A_E(\mathbf{x})$ and $A_I(\mathbf{x})$ are the Jacobians of $c_E(\mathbf{x})$ and $c_I(\mathbf{x})$, while $\mathbf{y}$ and $\mathbf{z}$ represent the equality and inequality Lagrange multipliers respectively. $\mathbf{S}$ and $\mathbf{Z}$ are diagonal matrices of $\mathbf{s}$ and $\mathbf{z}$, $\mathbf{I}$ is an identity matrix of appropriate size, and $\mathbf{e} = [1, 1, \ldots, 1]$. The barrier parameter μ, along with the step size α_s, are the main design parameters of interior point methods, and are beyond the scope of this thesis. From solving Equation (5.19) for the step directions p_x,p_s,p_y and p_z, the next state variable $\mathbf{x}$, slack variable $\mathbf{s}$, and Lagrange multipliers y and z are found from a line search method according to

$$x_{new} = x + \alpha_s p_x \tag{5.20}$$

Being a filter approach, as opposed to the less effective merit function approach, means that IPOPT will accept trial points x_{new} that *either* improve the objective function *or* improve the violation of constraints.

Part III

Results

Chapter 6

Model

This chapter will list the aerodynamic parameters for the Skywalker X8 model, Equation (B.4), and the rigid body kinetics of Equation (2.7), resulting from the analysis described in Chapter 3 and Appendix F.

6.1 Inertial parameters

In the following, the mass, inertia and center of gravity resulting from Appendix F is presented.

6.1.1 Mass and center of gravity

The center of gravity is referenced to a coordinate system with origin at the nose of the X8 with the x-axis backwards along the centerline and z-axis downwards. $z = 0$ is located at the joint between the upper- and lower part that the X8 body is made of. The data can be found in Table 6.1.The measured mass of the X8 is found in Table 6.2

Table 6.1: Center of gravity

x_{cg}	440mm
y_{cg}	0mm
z_{cg}	0mm

Table 6.2: Mass of different X8 components

Left wing covered with foil, including winglet	464g
Right wing covered with foil, including winglet	454g
Empty, painted body	439g
Assembled plane with minimal payload	3364g

Table 6.3: Inertia results

	X	Y	Z	XZ
I	1.2290	0.1702	0.8808	0.9343

6.1.2 Moments- and products of inertia

By following the procedure in Appendix F on the experimental data from Tables F.2-F.6, the moments- and products of inertia can be computed. The results are found in Table 6.3.

6.2 Aerodynamic parameters

Table 6.4 contains the aerodynamic parameters for the X8 model, Equation (B.4). This results from the XFLR and Matlab analysis presented in Chapter 3, and by pilot testing in the simulators[1]. It should be noted that $C_{D_{\beta_1}}$ and $C_{D_{\beta_0}}$ are practically zero, meaning that the drag from sideslip fits a second order model well. Also the lateral coefficients C_{Y_0}, C_{l_0} and C_{n_0} are practically zero.

6.2.1 Nonlinear pitching moment coefficient

By plugging the leading edge and chord length data from Table E.1 into Equation (3.14), $C_{m,fp}$ is found:

$$C_{m,flatplate} = \frac{2\sin^2(\alpha)}{S}\sum_{y=0}^{\frac{b}{2}}(2LE(y)c(y) + c(y)^2 - 2c(y)x_{cg})\Delta y \tag{6.1}$$

$$C_{m,fp} = -\frac{0.1626}{S} = -0.2168 \tag{6.2}$$

The resulting, blended $C_m\left(\alpha\right)$ curve can be seen in Figure 6.1.

[1]The changes made from the pilot tuning added damping in yaw (C_{n_β} and C_{n_r}) and reduced the coupling from yaw to roll C_{l_r}.

Table 6.4: Model parameters for the Skywalker X8

Parameter		**Parameter**	
b	2.1000	$C_{m_{\delta_e}}$	-0.4857
$\bar{c}$	0.3571	$C_{m_{fp}}$	-0.2168
e	0.9935	C_{m_q}	-1.3047
M	50	C_{Y_0}	$3.2049e-18$
α_0	0.2670	C_{Y_p}	-0.1172
S_{prop}	0.1018	C_{Y_r}	0.0959
S_{wing}	0.7500	$C_{Y_{\delta_a}}$	-0.0696
k_{motor}	40	$C_{Y_{\delta_r}}$	0
k_{T_P}	0	C_{Y_β}	-0.1949
k_Ω	0	C_{l_0}	$1.1518e-18$
C_{L_α}	4.0191	C_{l_p}	-0.4018
C_{L_0}	0.0254	C_{l_r}	0.0250
C_{L_q}	3.8954	$C_{l_{\delta_a}}$	0.2987
$C_{L_{delta_e}}$	0.5872	$C_{l_{\delta_r}}$	0
$C_{D_{delta_e}}$	0.8461	C_{l_β}	-0.0765
C_{D_p}	0.0102	C_{n_0}	$-2.2667e-07$
$C_{D_{\beta_2}}$	0.0671	C_{n_p}	-0.0247
$C_{D_{\beta_1}}$	$-2.0864e-07$	C_{n_r}	-0.1252
$C_{D_{\beta_0}}$	$7.7235e-05$	$C_{n_{\delta_a}}$	0.0076
C_{D_q}	0	$C_{n_{\delta_r}}$	0
C_{m_α}	-0.2524	C_{n_β}	0.0403
C_{m_0}	0.0180	C_{prop}	0.5

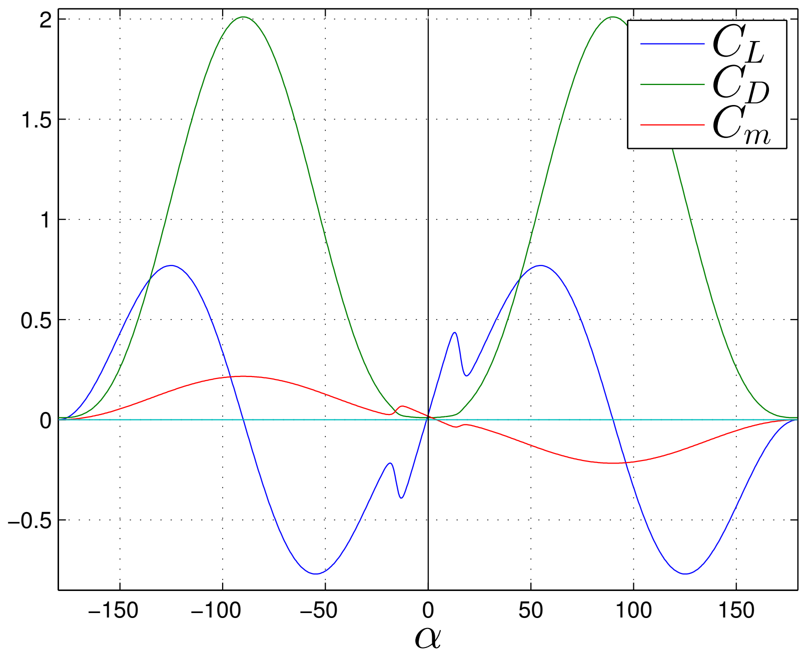


Figure 6.1: Blend of linear and flat plate lift, drag and pitch moment coefficients

Chapter 7

Landing

The following sections present the optimal trajectories found for 6 different scenarios, where initial height, distance to the net, elevation of the net and use of the throttle is varied. The reasoning behind including each scenario is outlined below:

Scenario 1: Normal landing. How fast will the X8 fly when constrained to the within the linear region? This scenario has the angle of attack constrained to well within the linear region, i.e. $-10^\circ \leq \alpha \leq 10^\circ$ is added to the path constraints in Equation (5.9c). It is included as a reference to the other scenarios.

Scenario 2: Fixed distance and initial height. How slow can the UAV land, given a fixed starting height h_0 and a fixed distance to the net Δ? For this scenario, the terminal constraint on the north position in Equation (5.14) is set to 15 meters, and the initial down position D_0 in Equation (5.11) is set to -5 meters. This is considered the basis scenario, on which the subsequent scenarios are based.

Scenario 3: Free initial height. How much can the terminal speed be reduced by entering the terminal phase at the optimal height? Here, the constraints are removed from the initial down position D_0, while the distance to the net is still 15 meters.

Scenario 4: Free distance. How long distance is needed to brake optimally? Now the constraint on the distance to the net is removed from Equation (5.14), while the initial height is 5 meters.

Scenario 5: No throttle. How dependent is the optimal approach on propeller thrust? Intuitively, adding energy to the system should lead to a larger end velocity, but this has to be investigated.

Scenario 6: Surface landing. By placing the net closer to the ground, the X8 is constrained from converting its kinetic energy into potential energy. This case investigates how this affect the terminal velocity and the planned trajectory. The setup for this scenario is the same as for scenario 2, with the exception of the final down position D_N of Equation (5.14), which is set to 1 meter.

NMPC This case looks at the performance of the NMPC described in Section 5.5.1, applied to scenario 2.

The results are presented in four plots; one showing the calculated trajectory in north and down positions with attitude, a plot of the angle of attack, one showing the planned control, and finally a plot of the north, down and absolute value velocities.

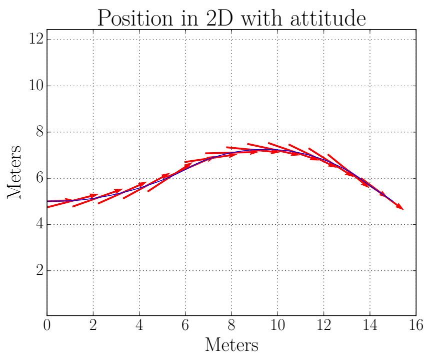
The terminal velocities for all the scenarios are presented in table 7.1 to simplify comparison. For all these scenarios, both R_1 and R_2 of the cost function in Equation (5.16) are set to 1 as this gave acceptable smooth control trajectories and low velocities. For the six path generation scenarios $N = 100$, while for the NMPC $N = 50$.

Table 7.1: Terminal velocities for the different scenarios

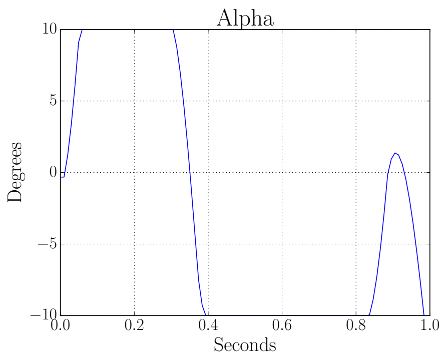
	Velocities [m/s]				
	Absolute	North	Down	Comment	Figure
Scenario 1	13.78	11.32	7.86	$-10° \leq \alpha \leq 10°$	7.1
Scenario 2	1.93	0.85	1.73		7.2
Scenario 3	1.10	1.10	-0.01	$h_0 = 1.01$	7.3
Scenario 4	0.27	0.12	0.24	$\Delta = 47.82$ m	7.4
Scenario 5	4.65	4.23	1.93	No throttle	7.5
Scenario 6	3.77	2.86	2.45	$h_N = 1.0$m	7.6
NMPC1, gust only	1.89	0.95	1.63		7.7

7.1 Scenario 1: Normal landing

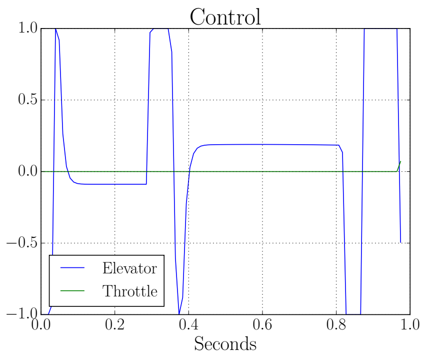
The trajectories for this scenario are found in Figure 7.1. From Figure 7.1b it is clear that the constraints put on X8 are limiting, as they are active for most of the flight. This is also supported by Figure 7.1a where the attitude of the UAV is consistent with the direction it is moving, i.e. $\theta \approx \alpha$. The effects of the small α is seen in Figure 7.1d where the velocity is rather large, with a terminal absolute value of 13.78 m/s.



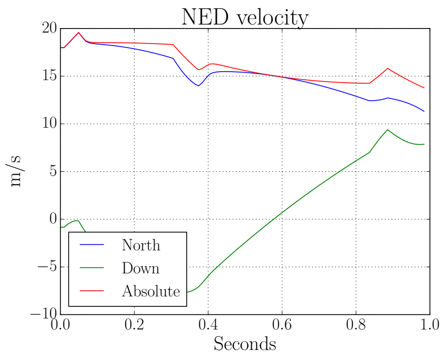
(a) North-Down plot with attitude



(b) Angle-of-attack plot



(c) Elevator and throttle plot

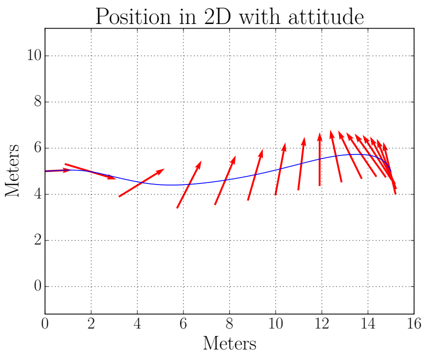


(d) North-Down velocity plot

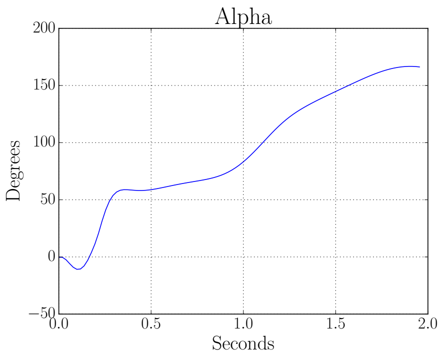
Figure 7.1: Normal landing

7.2 Scenario 2: Fixed distance and initial height

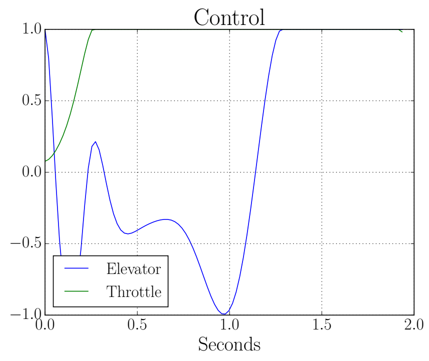
Once the limitations on the angle of attack is removed, it varies greatly throughout the flight, see Figure 7.2b. It is interesting to note that the angle of attack remains around 60° for some time. This coincides with the second top of the C_L curve, as seen in Figure 6.1. After an initial, short nose-down maneuver, the X8 pitches up and increases its angle of attack throughout the flight. The larger α contribute to a significantly lower absolute terminal velocity of 1.93 m/s. From Figure 7.2c it is clear that the propeller is applied for most of the landing. Pairing this with the attitude seen in Figure 7.1a, it becomes apparent that the thrust is applied upwards. As the UAV drops while full thrust is applied upwards, it is not capable of a full hover, which is also the case with the actual X8. The X8 flies with little change in height throughout the scenario, which is natural given the short time interval.



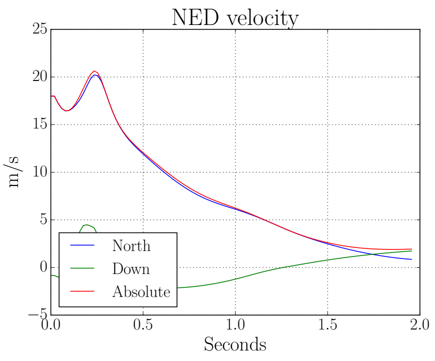
(a) North-Down plot with attitude



(b) Angle-of-attack plot



(c) Elevator and throttle plot

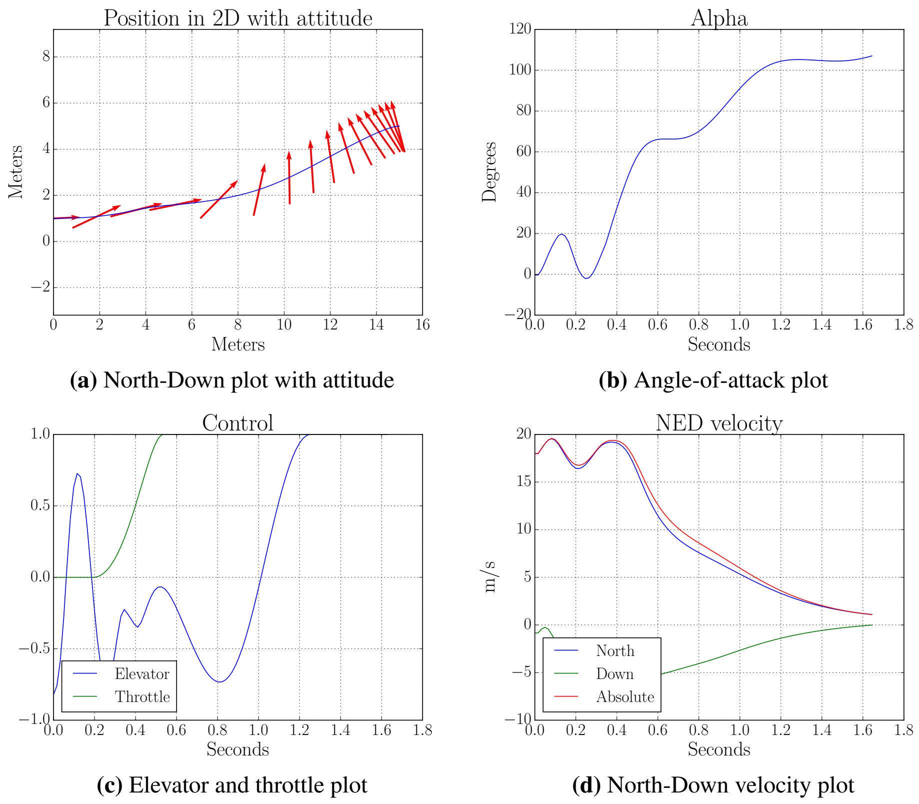


(d) North-Down velocity plot

Figure 7.2: Landing with fixed distance to the net and fixed initial height

7.3 Scenario 3: Free initial height

The calculated trajectories in Figure 7.3 follow the same tendencies as the trajectories in Figure 7.2 from the previous case. However, by letting the initial height be a free variable the terminal absolute velocity is slightly reduced to 1.10 m/s. The optimal starting height is found to be 1.01 meter above the surface, which coincides with the lower constraint on the height.

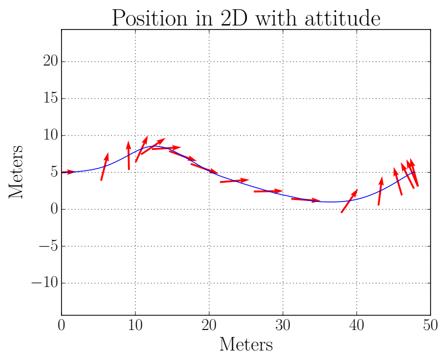


(a) North-Down plot with attitude
(b) Angle-of-attack plot
(c) Elevator and throttle plot
(d) North-Down velocity plot

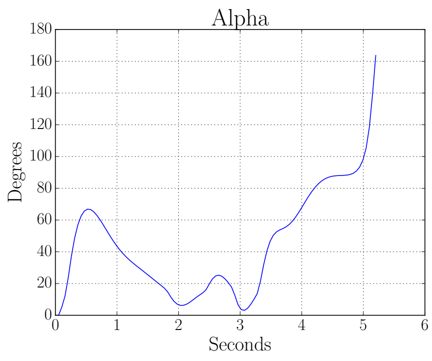
Figure 7.3: Landing with fixed distance to the net and free initial height

7.4 Scenario 4: Free distance

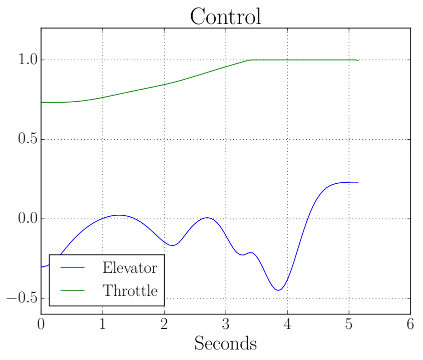
By letting the distance Δ to the net be a free variable, the absolute terminal velocity is further reduced to 0.31 m/s, see Figure 7.4d. The distance found to be optimal is considerably longer than in the previous cases. $\Delta = 40.89$m, see Figure 7.4a. The angle of attack remains positive for the entire scenario. Hovering is also attempted in this case.



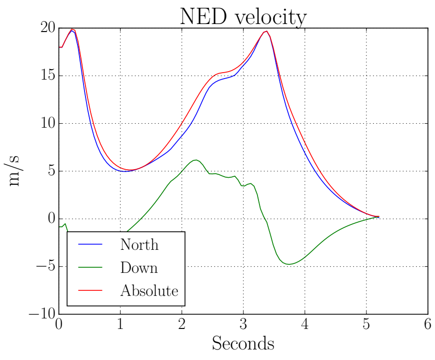
(a) North-Down plot with attitude



(b) Angle-of-attack plot



(c) Elevator and throttle plot

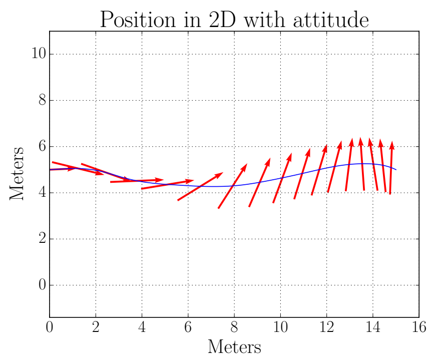


(d) North-Down velocity plot

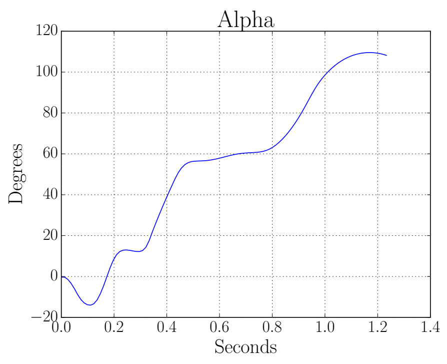
Figure 7.4: Landing with free distance to the net and fixed initial height

7.5 Scenario 5: No throttle

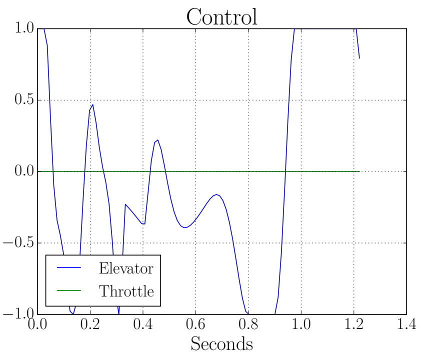
The tendencies both in the path of Figure 7.5a are the same as in Section 7.2. However, it is interesting to note that the angle of attack remains almost constant at around 15° at 0.3s and around 60° from 0.5s to 0.8s, which coincides with the first and second lift peak in Figure 6.1. From the attitude it is seen that the pitch angle barely exceeds 90°, whereas the other scenarios introduce a larger pitch angle. This makes sense, as a pitch angle over 90° cause a component of the thrust force to slow down the UAV. The biggest difference in this scenario is the increase in absolute terminal velocity, to 4.65m/s.



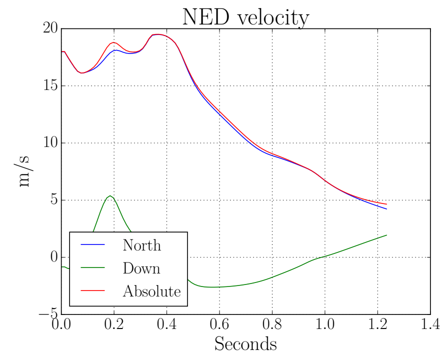
(a) North-Down plot with attitude



(b) Angle-of-attack plot



(c) Elevator and throttle plot

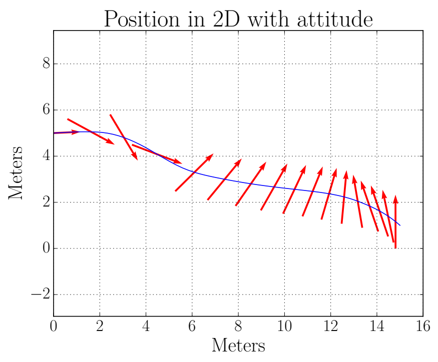


(d) North-Down velocity plot

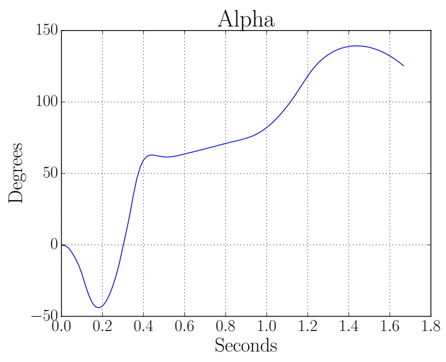
Figure 7.5: Landing without throttle

7.6 Scenario 6: Surface landing

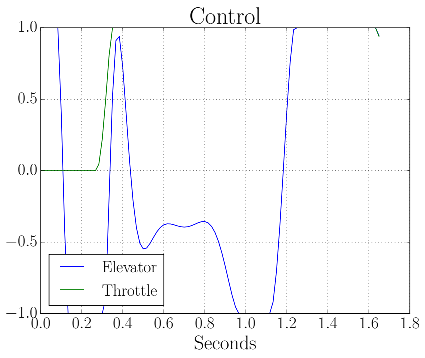
To investigate how much is gained from having an elevated net, the net is moved closer to the surface. Compared to Section 7.2 this yields a slight increase in absolute terminal velocity to 3.77m/s. To quickly loose altitude, while having a large drag, the X8 initiates an abrupt nose-down maneuver. This yields a large negative angle of attack. When a downward velocity is obtained, the nose is pulled back up to control the decent. The angle of attack is steady at values around the second top of the lift curve, but then increases further for the final part of the flight. The throttle is initially zero, but is maxed out as soon as the angle of attack is positive.



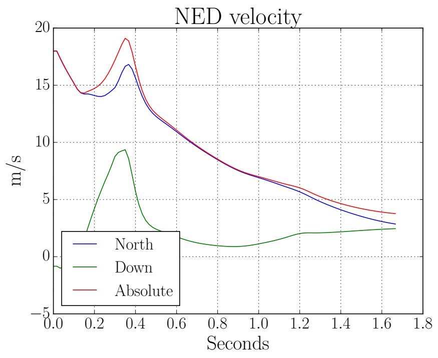
(a) North-Down plot with attitude



(b) Angle-of-attack plot



(c) Elevator and throttle plot

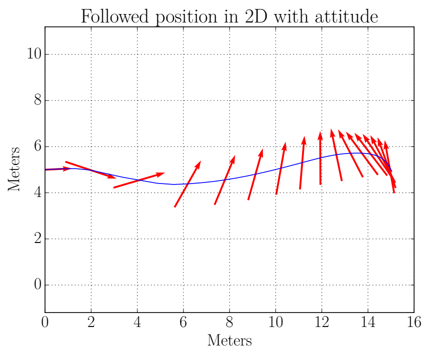


(d) North-Down velocity plot

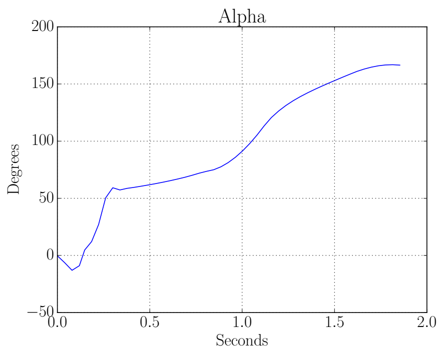
Figure 7.6: Landing at $h = 1\text{m}$

7.7 Nonlinear model predictive control for scenario 2 in gust wind

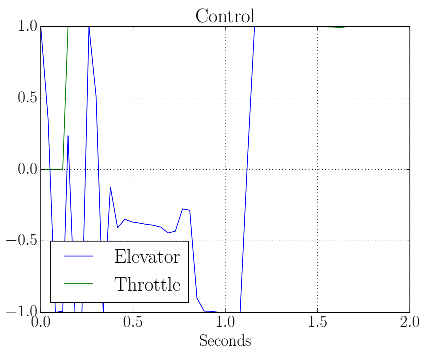
Figures 7.7a-7.7d show the actual path the X8 followed through the simulation, where each of the N elements come from different iteration of the NMPC. All the intermediately calculated paths, plotted in Figure 7.7e, are fairly similar when the steady wind component is zero. The demanded elevator in Figure 7.7c also follows the same tendencies as Figure 7.2c, but is a lot more rugged. The gust component is plotted in Figure 7.7f. For this case, the terminal absolute velocity is very similar to that of scenario 2: 1.89m/s.



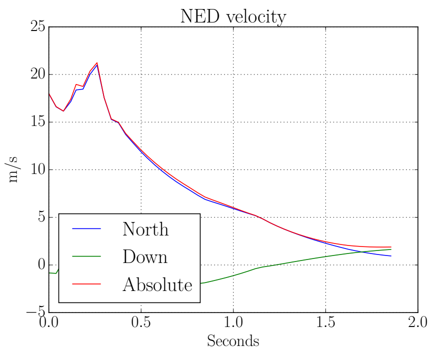
(a) North-Down plot with attitude



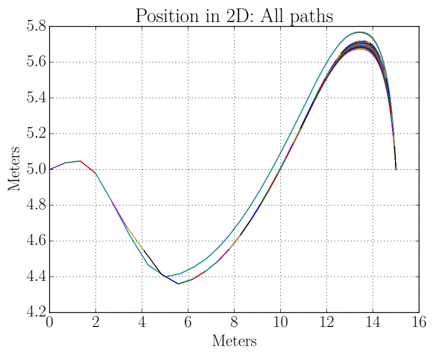
(b) Angle-of-attack plot



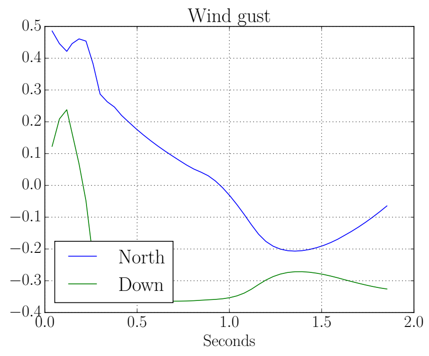
(c) Elevator and throttle plot



(d) North-Down velocity plot



(e) All calculated paths



(f) Wind gust component in NED

Figure 7.7: NMPC with gust

Part IV

Discussion and Conclusions

Chapter 8

Discussion

The results have showed that the landing velocity can be considerably reduced by utilizing the increased drag for large angles of attack. This chapter will discuss some of the results, and will assess its validity and limitations by investigating the assumptions made. Finally, remarks on the implications of this work, as well as possible future focus areas, are considered.

8.1 Optimistically low terminal velocity

This section will point the two main reasons that could cause the predicted terminal velocity to be lower than what the X8 acually can manage; a too demanding trajectory for the X8 to follow, and a model that predicts too much drag.

8.1.1 Too demanding trajectories

If the planned trajectories are too demanding for the X8 to follow, the followed path will naturally be less optimal, resulting in a higher terminal velocity. From the results it is seen that the trajectories require rapid changes in pitch, as well as the ability to almost hover the X8. This section investigates how realistic these demands are.

Fast pitch dynamics

The pitch dynamics are essentially affected by two factors; the moment of inertia around the y-axis, I_y, and the pitch moment coefficient $C_m(\alpha, q, \delta_e)$. If I_y is too low, or any of the components of $C_m(\alpha, q, \delta_e)$ differ from their actual values, the planned trajectories might involve pitching motions too rapid for the X8 to follow.

Beard and McLain (2012) includes the aerodynamic models of the Aerosonde, a 35kg, larger, tailed UAV, and the 1.56kg Zagi flying wing. I_y for the X8 is about $1/10$th that of the Aerosonde, and about twice that of the Zagi. This makes the moment of inertia seem reasonable. Nevertheless, it is worth noting that the formulas for calculating the inertia are very sensitive in the pendulum length L. In order to know the pendulum length, the center of gravity has to be precisely determined. During the tests, slight perturbations in φ were noticed. This will also affect the accuracy of the results as it indicates that some potential energy is converted into rotational energy. As with any other energy transformation, some loss is expected. One possible way to improve this is to attach the X8 to a rigid frame, together forming the pendulum. Then the strings could be attached to the frame to ensure $\dot{\varphi} = 0$, as non-zero rotation of the X8 with respect to the pendulum string is one source of error.

$C_{m_{\delta_e}}$ of the X8 is also in the same ballpark as with the Aerosonde and the Zagi, but it should be mentioned that the efficiency of the elevator during stall is a big uncertainty. C_{m_q} for the X8 and the Zagi are both about $1/3$rd of the Aerosonde. However, the pitch moment dependency on the angle of attack $C_m(\alpha)$ is smaller, which is an inherent problem with the flying wing configuration. Not only is the linear coefficient C_{m_α} smaller for the X8, but through the flat plate extension in pitch the X8 has considerably less damping in pitch for large angles of attack. This is believed to be the main explanation of the rapid pitching moment. This is also supported by the fact that the north-down plots show that the X8 very quickly positions itself athwart of the path, i.e. it is inducing a large α that is not opposed by $C_m(\alpha)$. If the pitch moment damping is larger than anticipated, the X8 might require a longer distance to land.

Even if $C_{m_{\delta_e}}$ is correct, too rapid movement in δ_e could also result in optimistic pitch dynamics by inducing rapid changes in $C_m(\alpha, q, \delta_e)$. At the current stage this is only mitigated by adding cost to the difference in δ_e from one time step to the next. A better solution would be to add the complete actuator model, with correct limits on the rates of the elevons. While this approach will lead to a trivial cost function, the downside is that it will call for a larger state-space, leading to a more computational demanding optimal control problem. This is not an issue for the offline path planning, but will be an issue for the online NMPC.

Hovering capabilities

From the results it is clear that a close-to-hover approach yields a much lower terminal velocity than the no-throttle case. One of the flaws here is that the motor model only has been manually tuned to resemble the actual X8 from the UAV lab, which is almost powerful enough to hover. Given the low speeds at which landing takes place, and thus the little influence from V_a, a more accurate motor model should be simple to make. This will require a rig that can measure the thrust produced by the motor at given control inputs. Regardless of the accuracy of the propulsion model, the no-throttle case provides an upper bound on the landing velocity. Still, it should be mentioned that given the severely underactuated X8 with a nose-up attitude will be extremely sensitive to the surroundings, possibly putting unrealistically high demands on the speed of the control loop, and the accuracy in the sensing and estimation of the states of the X8 and its surrounding environment. One argument for using throttle in the landing phase is that it will cause more air to flow over the ailerons, which might help mitigate the feared reduction of controllability in high angle of attack flight. On the other hand it might be desirable to have *some* speed in the landing moment, to overcome e.g. last-minute environmental disturbances.

8.1.2 Too much drag

As increased drag is the primary method to reduce the velocity of the X8, it is essential to investigate its validity. From Figure 6.1 it is clear that the drag force coefficient has a large peak at $\alpha = 90°$, which is reasonable considering that $\alpha = 90°$ is when all the air is flowing perpendicular to the wing. However, the amplitude of the peak is dubious, as it is purely based on the geometrical assumptions of the flat plate theory. The flat plate theory does not take three-dimensional effects, more specifically aspect ratio corrections, into account. The effects of a finite wingspan is clearly visualized in Ostowari and Naik (1985), where wind tunnel data of the lift, drag, and pitching moment coefficients for the NACA44XX series are plotted for different aspect ratios. Comparing e.g. the NACA4418 airfoil shows a $C_{D,max} \approx 2.05$ for $\mathcal{R} = \infty$ and $C_{D,max} \approx 1.33$ for $\mathcal{R} = 6$[1]. This justifies the belief that the drag force peak is larger than it should be. However, as the airfoil of the X8 is different from the NACA44XX series, the exact reduction is difficult to establish and should be further investigated. It is worth noting that Pointner and Kotsis (2011) successfully lands a flying wing UAV using the flat plate extended drag model in its uncorrected form. While Moore et al. (2014) adapts the flat plate model by using measurements from a motion capture system. The aspect ratio correction should also be investigated for C_L and C_m, as they show the same tendencies in Ostowari and Naik (1985).

The low-α part of the drag coefficient originates from the XFLR analysis, and should also be investigated. Contrary to the flat plate model this part is believed to be lower than in reality. This is evident through the close-to-unity Oswald efficiency number.

Despite the fact that the nonlinear region always has considerably more drag, a consequence of the aforementioned changes is that the relative increase in drag in the high-α region compared to the low-α region, is lower in reality than what is depicted in the model. This could lead to a change of the optimal trajectory.

8.2 Symmetric stall assumption

As this analysis has reduced the landing problem to the longitudinal aerodynamics, it also automatically assumes that the lateral and longitudinal aerodynamic can be decoupled also during stall. Despite this, it is apparent that coupled motions, such as spin, can happen, as they are encountered in the real world. This section suggests how the asymmetric stall

[1]The X8 has $\mathcal{R} \approx 5.8$

assumption can be scrutinized, and how the longitudinal and lateral aerodynamics can be kept uncoupled.

8.2.1 Spin prevention

As indicated by NATO (1976), the recommended way of dealing with spin is to prevent it rather than recovering from it. Similarly, spin can be avoided by preventing asymmetric stalls from occurring. By keeping the wings symmetrical with respect to the incoming wind, they will have the same angle of attack and therefore stall symmetrically without going into a spin. Instead of the normal roll PID controller, which stabilize the UAV around its *body* x-axis, one possible improvement could be to stabilize the UAV around the *stability* x-axis. By assuming that the lateral PID controller maintain both wings at the same angle of attack, lateral and longitudinal axis can still be considered decoupled.

However, as the X8 only has elevons and no rudder, this seems difficult in practice without adding control surfaces capable of producing yaw moment[2]. This approach will also require good estimates of the angle of attack.

8.2.2 Spin modeling

By investigating spin behavior and including this information in the model, spins can be circumvented by avoiding or detecting situations where they occur. This is somewhat similar to what general aviation pesimistically do today, by avoiding stall altogether. Unfortunately, the way spin is represented in the aerodynamic models of today is arguably not very intuitive with respect to the physical principles behind it. Besides, data from spinning UAVs have to be collected from real flights, as spinning in a wind tunnel is impractical.

8.3 Non-convex optimization

Another issue that should be discussed is the non-convex nature of the optimal control problem. Because of this, there is no way of knowing whether the found solution is globally optimal, or if it is just a local minimum. Mitigation of this problem is obviously a key part of the algorithms of advanced nonlinear solvers, such as IPOPT. Nevertheless, this problem was encountered in the early stages of this work, when sometimes a *less* constrained problem yielded a *higher* terminal velocity. One possible way to diminish this problem is to opt for a more advanced, commercial solver. However, this is an inherent problem with nonlinear optimal control.

[2]As a adding a rudder to the X8 is impractical from a constructional perspective, decelerons seem like the most viable option.

8.4 Future work

First and foremost the work in this thesis has laid the groundwork for Gryte et al.(Submitted). Nevertheless, some areas that should be investigated in the future, in addition to what has already been mentioned in this chapter, are the implementation of the controller on the X8, as well as further validation and expansion of the model.

8.4.1 Real-time implementation

As the long term goal of this project is to land the physical X8, some of the remaining steps are pointed out. To be able to operate in the physical world, the controller and simulations should be extended to six degrees-of-freedom. This might be in the form of a 6DOF NMPC as in Mathisen et al.(Submitted) or with a 3DOF NMPC as in this thesis in combination with the aforementioned lateral PID controller, possibly in combination with the two-phased controller scheme presented in Gryte et al.(Submitted). Another possibility is to use the paths from this thesis as reference trajectories for a Lyapunov-based controller, similar to Moore et al. (2014).

The controller should also be adapted to the hardware platform of the X8. As the BeagleBone Black, which currently controls the UAV lab X8, is significantly slower than the average desktop computer, this has to be considered during the implementation. A first step on the way is to implement the NMPC as a DUNE task written in C++[3]. Then the controller can be interfaced with the SITL/HIL simulator for testing. Also the setup of the NMPC should be simplified. For instance the state space can be reduced as there is no need for both the NED and body frame velocities. The search space can be further reduced by enforcing slightly stricter constraints, rejecting solutions that clearly can not be optimal.

Finally, the stability properties of the NMPC should be investigated and possibly enhanced by applying the MPC stability theory from Grüne and Pannek (2011); Rawlings and Mayne (2009).

8.4.2 Model validation and expansion

Since the only validation that has taken place for the aerodynamic model is through pilot SITL testing, one of the key tasks for the future should be to validate the model with flight log data. The development of model-free angle of attack estimator in Johansen et al. (2015), as well as the UAV lab's recent purchase of an Aeroprobe 5-hole sensor for measurement of airspeed, angle of attack and sideslip angle, will be important tools in this work. The model should also be expanded to include details on

Effects of stall, primarily the effectiveness of control surfaces, lateral effects, and validation of the already included lift, drag and pitching moment coefficients.

[3] ACADO(Houska et al., 2011) seems to be a good choice for optimal control in C++.

Nonlinear roll damping should be investigated, as the X8 is inherently extremely well damped in roll.

Moment induced by the propeller. Applying thrust from the propeller will in principle induce moments around all three axis.

Chapter 9

Conclusion

This thesis has investigated perched landing for the Skywalker X8 fixed wing UAV that utilize the increased drag experienced in the high angle of attack region, through a comparison of six different landing scenarios in three degrees-of-freedom. By formulating landing as an optimal control problem through the Python CasADi optimization framework using the IPOPT nonlinear interior point method, six different landing trajectories are constructed. These optimal trajectories produce a wide range of landing velocities when varying the initial height, distance to and elevation of the net, and whether the throttle is used or not.

The low angle of attack reference scenario landed at 13.78m/s, when landing is initiated 15 meters from the net elevated 5 meters above the ground. When removing the constraints on the angle of attack, the landing velocity was reduced to 4.65m/s. This was further reduced to 1.93m/s by using the throttle actively throughout the landing. Further reductions in the terminal velocity was achieved by allowing the initial height to be a free variable. This resulted in a terminal velocity of 1.10m/s, starting from an inital height of 1.01 meters. Seeking a low initial height seems natural from a potential energy perspective. The lowest terminal velocity was attained by setting the distance to the net as a free variable in the optimization. This caused the X8 to fly 47.82 meters before landing at 0.27m/s. Lowering the net to 1 meter above the ground lead to an increased terminal velocity of 4.65m/s.

The conclusion drawn from this is that a perched landing strategy will lead to a reduction in terminal absolute velocity, compared to a conventional landing with a conservative angle of attack. Regarding the setup of the landing, it is found to be advantageous to start from a low altitude, landing into an elevated net. Simulations with the developed receding horizon NMPC have showed that the same performance is achievable through control of the X8 under minor environmental disturbances, in the form of wind gusts.

However, some of these terminal velocities seem unreasonably slow. Model uncertainties, especially in the propeller model, the pitch moment damping and in the drag coefficient, are pointed out as the most probable flaws, along with fast actuator dynamics. Whether an equally large reduction of terminal velocity is possible in practice, on the physical X8, also depends on the capabilities of the real-time implementation and on how the lateral degrees-of-freedom are affected by the high angle of attack flight, subjects that are considered briefly in this thesis.

Part V

Appendices

Appendix A

Submitted abstracts

This appendix includes two abstracts that have been submitted for the AIAA Science and Technology Forum and Exposition SciTech 2016 in San Diego, U.S., and the 2016 IEEE Aerospace Conference in Big Sky, Montana, U.S, respectively.

Non-linear Model Predictive Control for Longitudinal and Lateral Guidance of a Small Fixed-Wing UAV in Precision Deep Stall Landing

Siri Holthe Mathisen, Kristoffer Gryte, Thor I. Fossen and Tor A. Johansen

To recover a small UAV in a small space without a runway, belly landing on a soft surface or a simple an arrest system like a recovery net can be used. It is desirable to ease the impact by minimizing the speed at which the UAV meets the landing target. One method for doing this is to land the UAV with the help of a deep stall. In this case, the UAVs angle of attack needs to be beyond the stall angle, where the drag coefficient of the UAV increases and the lift coefficient decreases. This makes the UAV loose height while at the same time losing speed in the horizontal direction. It is difficult to control a UAV in a deep stall, and side winds might cause the UAV to spin and lose control. It is also important to be able to accurately guide the UAV at the same time as it is being decelerated, to be able to accurately hit the landing target.

Earlier work with net landing is described by in the article by Skulstad et al. [1], where an X8 flying wing UAV is successfully guided into a recovery net with 1 meter vertical and lateral accuracy using a low-cost single-frequency RTK GNSS navigation system. Deep stall landing of small vehicles has been studied by [Sim, 1984], who presented a study of the flight characteristics of a manned deep stall. In his article from 2008 [2], Spera presented useful data from various stalled and unstalled airfoils in wind tunnels and wind turbines, while Taniguchi in his article [3] presented an analysis of a deep stall landing for UAV with longitudinal dynamics, including flight simulations and flight tests.

To control a UAV, model predictive control (MPC) can be used. MPC is a method that tries to optimize an objective function constrained by the systems dynamics and physical and operational constraints. This is done by producing a sequence of controls over a time horizon based on the state of the system, and then applying the first control action to the system. The feedback from the system is then fed into the controller, which produces a new set of controls. The 6 degrees of freedom (6-DOF) dynamics of a UAV is highly non-linear, which benefits from a non-linear MPC (NMPC) to control it. While a linear MPC has an optimization problem that can be solved with convex quadratic programming, the NMPC is usually solved by transforming the control problem into a non-linear program (NLP) and then optimize it.

In this article, we employ an NMPC on the 6-DOF dynamics of a fixed-wing UAV in quaternion representation, to guide it in a deep stall while at the same time landing in a given location. The model for the fixed-wing UAV is described in [4, p. 156] and consists of north-east-down positions relative to the inertial frame, body velocity, quaternions for representation of attitude and angular velocity in the body frame. The UAV model used in this article is controlled by three control surfaces and throttle. The NLP of the NMPC consists of an objective function, which should be minimized over a time horizon of N discrete time intervals, and constraints, including an initial value and a continuity constraint. The initial value maintains a continuity between each optimization of the NMPC, and the continuity constraint keeps each discrete control interval connected to the next, saying that the optimization variable representing the state of the next time step should equal the right hand side of the discretized differential equation for the model.

The deep stall landing of the UAV follows an initial path consisting of two straight paths: The first is horizontal and the next is a sloping final approach from the UAVs cruising position to the landing position. To guide the UAV along this path, its course angle and path angle is controlled by the NMPC. To achieve this, the difference between the path angle and the desired path angle, and the difference between the course angle and the desired course angle, are minimized by including them in the NLPs objective function. The last contribution to the objective function is the speed of the UAV relative to the ground, which should also be minimized. The contributions from these three terms are weighted in the objective function to make the UAV follow the course to satisfaction, and at the same time fly with minimal speed. Both the states of the UAV model and the control variables are included as optimization variables in the NLP. Bounds are placed on both states and inputs to ensure a realistic simulation. The angle of the control surfaces and throttle are limited by the physical saturations.

This work is a continuation of the work done by Mathisen et al. [5], where the NMPC guided deep stall landing of a longitudinal model of a fixed-wing UAV was studied. In this article, the lateral dynamics are also included, considering the side wind influence on the UAV. Through simulations done in python with the open-source software package CasADi, it is investigated how the landing precision and minimal speed is influenced by wind gusts and crosswinds. The forces that act upon the UAV are the aerodynamic force, the gravitational force and the propulsion force. The aerodynamic force is highly dependent on the UAVs relative speed, making changes in the wind important to the model. The wind is modelled as a sum of a steady component in the inertial frame and a gust component in the body frame. To simulate time changing wind, the Dryden turbulence is used to model the gust component, though the steady component is assumed constant for the same altitude. The steady component is assumed known to the model predictive controller, but to simulate time-delayed estimates of the wind gust, the last time steps gust measurement is used in the controller. The gust therefore introduces a time varying disturbance, testing the robustness of the controller. At low speed the control surfaces are less effective, and particular consideration is required by the NMPC to optimally exploit the forces that can be generated by the control surfaces or compensate for winds and adjust the course and flight path. The NMPCs ability to effectively handle non-linear dynamic subject to such constraints is studied in the simulations.

References

[1] R. Skulstad, C. L. Syversen, M. Merz, N. Sokolova, T. I. Fossen, and T. A. Johansen, “Net Recovery of UAV with Single-Frequency RTK GPS,” in *Proc. of the IEEE Aerospace Conference, Big Sky, Montana*, 2015.

[2] D. A. Spera, “Models of lift and drag coefficients of stalled and unstalled airfoils in wind turbines and wind tunnels,” *Nasa/CR*, 2008.

[3] H. Taniguchi, “Analysis of deepstall landing for UAV,” vol. 3, pp. 2498–2503, 2008.

[4] R. Beard and T. McLain, *Small unmanned aircraft: Theory and practice*. Princeton University Press, 2012.

[5] S. H. Mathisen, T. I. Fossen, and T. A. Johansen, “Non-linear model predictive control for guidance of a fixed-wing uav in precision deep stall landing,” in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2015.

Low-Velocity Precision Landing of Fixed-Wing UAV Using Stall and Nonlinear Model Predictive Control

Kristoffer Gryte, Siri H. Mathisen, Tor A. Johansen and Thor I. Fossen

Center for Autonomous Marine Operations and Systems (AMOS),
Department of Engineering Cybernetics,
Norwegian University of Science and Technology, Trondheim, Norway

Several applications of small fixed-wing UAVs require that the UAV can be recovered without access to a runway. Such applications include operation from ships or other space-restricted landing sites. It also includes cases when the deployment is time-critical and there is no time to establish an ideal operating area, such as search and rescue or other emergency response operations. We note that fixed-wing UAVs are often preferred over UAVs with VTOL (Vertical Takeoff and Landing) capability in such applications due to their significantly larger endurance and/or speed.

The recovery of UAVs in such cases usually have two objectives that are typically conflicting: Arriving at the landing target with very high precision within the limited space available, and arriving at the landing target with minimum speed in order to reduce the risk for damage when being caught by the arrest system (landing net, line/hook, soft surface, or similar [6, 2]) or missing the landing target. In order to satisfy these objectives at the same time, the trajectory of the fixed-wing UAV should be chosen to balance the increase in vertical velocity resulting from the loss of lift due to decreasing the horizontal velocity, and the loss of effectiveness of the control surfaces to respond to wind gusts and turbulence when operating at low speed and possibly in stall conditions.

In order to optimally control a small fixed-wing UAV to landing with these operational constraints, we propose a two-phase strategy where

1. First, the UAV approaches the landing target at relatively high speed in order to maintain the necessary control authority to achieve high precision in the final approach and to be able to efficiently abort landing if this is made necessary by too large tracking control error resulting from disturbances due to wind, or the landing target have moved (e.g. ship in motion).

2. Then, as close to the landing target as possible, the UAV is controlled into a deep stall in order to quickly minimize speed upon impact with the arrest system, and at the same time arrive at the landing target position with the required attitude not to cause damage. The deep stall landing utilizes the increased drag at high angles of attack in order to reduce the horizontal velocity, similar to perched landing systems inspired by birds [5]. This is phase is challenging from a control point of view, since the control authority is significantly reduced during this highly dynamic maneuver such that when arriving at the landing target there is very little control authority left, and there is usually no way to abort landing or make significant corrections for unexpected disturbances during this final phase. Moreover, the dynamics in stall and post-stall may be highly nonlinear, and the control actuator limitations must be taken into consideration.

This article presents the above landing control scheme. Due to the increased risk involved with high angle of attack operations[7], risk is included in the desicion process when switching to, and executing, the second phase. To achieve this, the control is split into two levels of abstraction; a low level PID controller[1], and a higher level decision-making Nonlinear Model Predictive Controller (NMPC), similar to [3, Fig. 2(b)]. After switching to the optimal trajectory, it is tracked using NMPC, since this is found to be an effective method to deal with the nonlinearities and control constraints during deep stall landing, [4]. The low level control is tracking a straight path to the landing target for the UAV to arrive at the desired position, however with a rather large velocity and thus a risk of damaging the UAV. While the PID is bringing the UAV closer to the landing target, the high level control evaluates the opportunity for an alternative low speed trajectory towards the arrest system that utilize the increase in drag for large angles of attack to reduce the terminal velocity. However, this trajectory is only accepted if the calculated risk is at an acceptably low level. The risk is evaluated by faster-than-real-time predictive simulation within the NMPC by considering the experienced wind gusts, sensor integrity, motion of the landing target, and the current position and velocity relative to the landing target. This risk predicts the future behavior of the UAV under the prevailing conditions, all to prevent the UAV from going into a spin, stalling the control surfaces, or in any other way lead to an incident. If the predicted risk is not favorable, the system may either proceed with the landing at relatively high speed to avoid a missed landing target incident, or automatically abort the landing and execute an evasive maneuver, if admissible.

The nonlinear control problem is solved by transforming it into a nonlinear program (NLP) and applying direct multiple shooting. Direct multiple shooting divides the control horizon into discrete control intervals and uses piecewise constant control. To increase robustness against wind, the most recent estimate of wind velocity mean and variance is included in the NMPC. The robustness is investigated by using different qualities of wind measurements, since body frame wind can be difficult to estimate in practice.

The proposed control scheme is tested through simulations of nonlinear UAV model in the CasADi python framework. CasADi is a symbolic framework developed to solve nonlinear numerical optimization. In particular the effects of wind gusts and turbulence on the landing performance is evaluated by Monte Carlo simulations, considering the position error, kinetic energy and attitude when arriving at the landing target.

References

[1] Randal W. Beard and Timothy W. McLain. *Small Unmanned Aircraft.* Princeton University Press, 2012.

[2] Alvika Gautam, P.B. Sujit, and Srikanth Saripalli. A survey of autonomous landing techniques for UAVs. *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1210–1218, May 2014.

[3] Tor A Johansen. Toward Dependable Embedded Model Predictive Control. *IEEE SYSTEMS JOURNAL*, pages 1–12, 2014.

[4] Siri Holthe Mathisen, Thor I Fossen, and Tor A Johansen. Non-linear Model Predictive Control for Guidance of a Fixed-Wing UAV in Precision Deep Stall Landing. In *International Conference on Unmanned Aircraft Systems*, Denver, 2015.

[5] Joseph Moore, Rick Cory, and Russ Tedrake. Robust post-stall perching with a simple fixed-wing glider using LQR-Trees. *Bioinspiration & biomimetics*, 9(2):025013, June 2014.

[6] Robert Skulstad, Christoffer Lie, Syversen Mariann, Merz Nadezda, Sokolova Thor, and I Fossen Tor. Net Recovery of UAV with Single-Frequency RTK GPS. In *Proc. of the IEEE Aerospace Conference*, number 978, Big Sky, Montana, 2015.

[7] Robert F. Stengel. *Flight Dynamics.* Princeton University Press, 2004.

Appendix B

Model structure

This chapter includes the complete equation for the aerodynamic, propulsion and gravitational forces acting on the X8, as developed in Chapter 3. In addition, the linear model from Etkin and Reid (1996) is included for completeness. It should be noted that Etkin and Reid (1996) use X, Y, Z, L, M, N to represent forces and moments, and that $\frac{\delta X}{\delta u} = X_u$.

$$\begin{bmatrix}\Delta\dot{u}\\ \dot{w}\\ \dot{q}\\ \Delta\dot{\theta}\end{bmatrix} = \begin{bmatrix} \frac{X_u}{m} & \frac{X_w}{m} & 0 & -g\cos(\theta_0) \\ \frac{Z_u}{m-Z_{\dot{w}}} & \frac{Z_w}{m-Z_{\dot{w}}} & \frac{Z_q+mu_0}{m-Z_{\dot{w}}} & \frac{-mg\sin(\theta_0)}{m-Z_{\dot{w}}} \\ \frac{1}{I_y}\left[M_u + \frac{M_{\dot{w}}Z_u}{m-Z_{\dot{w}}}\right] & \frac{1}{I_y}\left[M_w + \frac{M_{\dot{w}}Z_w}{m-Z_{\dot{w}}}\right] & \frac{1}{I_y}\left[M_q + \frac{M_{\dot{w}}(Z_q+mu_0}{m-Z_{\dot{w}}}\right] & -\frac{M_{\dot{w}}mg\sin(\theta_0)}{I_y(m-Z_{\dot{w}}} \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix}\Delta u\\ w\\ q\\ \Delta\theta\end{bmatrix} + \begin{bmatrix} \frac{\Delta X_c}{m} \\ \frac{\Delta Z_c}{m-Z_{\dot{w}}} \\ \frac{\Delta M_c}{I_y} + \frac{M_{\dot{w}}\cdot\Delta Z_c}{I_y(m-Z_{\dot{w}}} \\ 0 \end{bmatrix} \tag{B.1}$$

$$\begin{bmatrix}\dot{v}\\ \dot{p}\\ \dot{r}\\ \dot{\varphi}\end{bmatrix} = \begin{bmatrix} \frac{Y_v}{m} & \frac{Y_p}{m} & \frac{Y_r}{m} - u_0 & g\cos(\theta_0) \\ \frac{L_v}{I'_x} + I'_{zx}N_v & \frac{L_p}{I'_x} + I'_{zx}N_p & \frac{L_r}{I'_x} + I'_{zx}N_r & 0 \\ I'_{zx}L_v + \frac{N_v}{I'_z} & I'_{zx}L_p + \frac{N_p}{I'_z} & I'_{zx}L_r + \frac{N_r}{I'_z} & 0 \\ 0 & 1 & \tan(\theta_0) & 0 \end{bmatrix} \begin{bmatrix}v\\ p\\ r\\ \varphi\end{bmatrix} + \begin{bmatrix} \frac{\Delta Y_c}{m} \\ \frac{\Delta L_c}{I'_x} + I'_{zx}N_c \\ I'_{zx}\Delta L_c + \frac{\Delta N_c}{I'_z} \\ 0 \end{bmatrix} \tag{B.2}$$

$$\begin{aligned} I'_x &= \frac{I_xI_z - I_{zx}^2}{I_z} \\ I'_z &= \frac{I_xI_z - I_{zx}^2}{I_x} \\ I'_{zx} &= \frac{I_zx}{I_xI_z - I_{xz}^2} \end{aligned} \tag{B.3}$$

$$\begin{aligned} \begin{bmatrix}F_x\\ F_y\\ F_z\end{bmatrix} &= -\frac{1}{2}\rho V_a^2 S\mathbf{R}_{stability}^{body}(\alpha,\beta) \begin{bmatrix} C_{D_p} + (1-\sigma(\alpha))\frac{[C_{L_0}+C_{L_\alpha}\alpha]^2}{\pi e AR} + \sigma(\alpha)\left[2\,\mathrm{sign}(\alpha)\sin^3(\alpha)\right] + C_{D_q}\frac{c}{2V_a}q + C_{D_{\beta^2}}\beta^2 + C_{D_\beta}\beta + C_{D_{\delta_e}}\delta_e \\ 0 \\ (1-\sigma(\alpha))\left[C_{L_0} + C_{L_\alpha}\alpha\right] + \sigma(\alpha)\left[2\,\mathrm{sign}(\alpha)\sin^2(\alpha)\cos(\alpha)\right] + C_{L_q}\frac{c}{2V_a}q + C_{L_{\delta_e}}\delta_e \end{bmatrix} \\ &+ \frac{1}{2}\rho V_a^2 S \begin{bmatrix} 0 \\ C_{Y_0} + C_{Y_\beta}\beta + C_{Y_p}\frac{b}{2V_a}p + C_{Y_r}\frac{b}{2V_a}r + C_{Y_{\delta_a}}\delta_a + C_{Y_{\delta_r}}\delta_r \\ 0 \end{bmatrix} + \mathbf{R}_{NED}^{body}(\varphi,\theta)\begin{bmatrix}0\\0\\mg\end{bmatrix} + \frac{1}{2}\rho S_{prop}C_{prop}\begin{bmatrix}V_d(V_d - V_a)\\0\\0\end{bmatrix} \\ \begin{bmatrix}l\\ m\\ n\end{bmatrix} &= \frac{1}{2}\rho V_a^2 S \begin{bmatrix} b(C_{l_0} + C_{l_\beta}\beta + C_{l_p}\frac{b}{2V_a}p + C_{l_r}\frac{b}{2V_a}r + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r) \\ (1-\sigma(\alpha))[C_{m_0} + C_{m_\alpha}\alpha] + \sigma(\alpha)[C_{m_{fp}}\,\mathrm{sign}(\alpha)\sin^2(\alpha)] + C_{m_q}\frac{b}{2V_a}q + C_{m_{\delta_e}}\delta_e \\ b(C_{n_0} + C_{n_\beta}\beta + C_{n_p}\frac{b}{2V_a}p + C_{n_r}\frac{b}{2V_a}r + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r) \end{bmatrix} + \begin{bmatrix} -k_{T_p}(k_\Omega\delta_t)^2 \\ 0 \\ 0 \end{bmatrix} \end{aligned} \tag{B.4}$$

Appendix C

XFLR5

XFLR5 can run analysis based on the following three principles for calculating aerodynamical forces and moments:

Lifting Line Theory(LLT) is the most theoretically supported analysis, see e.g. Prantl (1921). However it assumes low sweep, low diheral and high aspect ratio. The method represents the airfoil as a lifting line located at $\frac{1}{4}$ of the chord. By applying the Kutta-Joukowski theorem, $F_L(y) = \rho V \Gamma(y)$, it transforms the problem into finding the circulation Γ instead of the lift force F_L. When considering the airflow onto an airfoil, and the slight deflection of the outgoing flow, circulation is essentially the flow around the airfoil that is superpositioned with the inflow to achieve this deflection. Circulation can be found more easily from geometric considerations. Sivells and Neely (1947) explains a nonlinear extension to LLT that is used by XFLR.

Vortice Lattice Method(VLM) extends LLT, by adding panels in the chordwise direction. VLM does not consider viscous effects, turbulence, dissipation and boundary layers effects (Minsaas and Steen, 2012), meaning that it assumes potential flow, incompressible, inviscid and irrotational flow field, thin lifting surfaces, as well as small angle of attack. The extension from LLT leads to a system of linear equations.

3D-panel is added to XFLR to enable modelling of fuselages, give more insight in the pressure distribution across the surface of the wing, and try to improve results by taking the thickness of the wings into account(Deperrois, 2013). However at this point the method is not arguably better than VLM/LLT(Deperrois, 2009)

Since the X8 has a rather large sweep, low aspect ratio and no fuselage, the focus here will be on VLM.

C.1 XFLR assumptions and flow theory

The following section will look at some of the assumptions made by XFLR(VLM) and show how these are applicable to the analysis of the X8.

C.1.1 Reynolds number

XFLR is made for “Analysis of foils and wings operating at low Reynolds numbers”(Deperrois, 2013), typically in the range below ten million. The definition of the Reynolds number is

$$Re_x = \frac{\text{inertial forces}}{\text{viscous forces}} = \frac{\rho V x}{\mu} \tag{C.1}$$

where ρ is the density of air, V is the velocity, x is the position along the chord (distance travelled by the air over the surface) and μ is the dynamic viscosity of air. For a rough upper bound on Re an extreme condition is considered: $V = 30\text{m/s}$, $x = 1\text{m}$, $\rho = 1.164\frac{\text{kg}}{\text{m}^3}$ and $\mu = 1.886 \cdot 10^{-5}\frac{\text{kg}}{\text{m}\cdot\text{s}}$ at 30° Celsius, yielding $Re < 2 \cdot 10^6$ which is well within limitations.

C.1.2 Incompressible flow

Incompressible flow refers to flow of a fluid with constant density. As a result of this, the volume of incompressible fluid remains constant. In airfoil analysis, the dimensionless quantity Mach number is often useful. It is defined as the ratio between the velocity of the object and the velocity of air:

$$M = \frac{v_{object}}{c} = \frac{v_{object}}{\sqrt{\gamma R T}} \tag{C.2}$$

Here γ,T and R are the *ratio of specific heat*, ideal gas constant and temperature of the surrounding gas, respectively. (Anderson, 1989, Equation (4.75)) gives the following expression for the relative density for compressible, isentropic flow:

$$\frac{\rho_0}{\rho} = \left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{1}{\gamma - 1}} \tag{C.3}$$

For normal air we have $\gamma = 1.4$, which yields $\frac{\rho_0}{\rho} \approx 1$ for $M < 0.3$. Thus it is safe to assume the flow as incompressible for low Mach numbers.

C.1.3 Boundary layer

To understand stalling it is key to understand the effects of the boundary layer around an airfoil. Due to the friction along the airfoil the flow passing the airfoil is slowed down, while in the freestream region outside the boundary layer the velocity is assumed unaffected by the friction. As the air moves over the foil gradually more air is affected by friction, leading to an increase in the boundary layer thickness δ towards the tailing edge of the foil.

Naturally the boundary layer is tightly linked to calculation of drag for an airfoil. Viscous flow, i.e. flow where friction affects the fluid motion significantly, is split in two different categories: laminar flow moving along smooth streamlines, and more irregular turbulent flow. They represent two different domains of fluid dynamics and have to be considered separately. A big challenge within the field of CFD is the transition between laminar and turbulent flow, and the detection of such a transition. This problem is eluded by XFLR by assuming inviscid flow. Inviscid calculations will give the correct pressure distribution on the outside of the boundary layer. Anderson (1989) states that the pressure along the normal of an airfoil, for a slender bodied airfoil, is constant through the boundary layer. Then the inviscid pressure can be extended into the viscous boundary layer. However, viscous effects are not negligible for slow flying UAVs, so XFLR estimates this by interpolation of XFoil pregenerated polars (Deperrois, 2013).
The tight link between boundary layer effects and stalling, and the fact that XFLR neglect boundary layer effects, also makes XFLR assume low angle of attack α. This is unfortunate for the high-α landing application of the X8 model.XFLR also assumes small sideslip, β, so this has to be taken into consideration for the analysis.

Appendix D

Model

This chapter will explain how a physical UAV can be imported into XFLR and be analyzed.

D.1 Importing a physical UAV into a CFD program

The aerodynamic forces acting on the wing are highly dependent on the shape of the airfoil, the cross-section of the airplane wing. In order to identify the forces properly a good approximation of the airfoil is needed. Obviously any serious manufacturer knows what airfoil their planes have, but they might be kept secret or the data might be unavailable, the latter being the case with the X8.

Since the X8 is a flying wing UAV, extra care has to taken when approximating its wing. To be able to fly in level flight[1], a UAV needs to be designed such that the lift coefficient C_L is positive for a zero pitch moment C_m (Meshia, 2008). In other words; it should be possible to balance the plane longitudinally while acheiving an upward lift force. The tail on a conventional airplane is designed to counteract the pitching moment from the main wing. For instance; to counteract a negative pitch from the main wing, the tail can be constructed to have negative lift Deperrois (2010). Without a tail to make such corrections, a flying wing has to be constructed differently to guarantee pitch stability. There are several ways this can be done:

- Using a self-stable airfoil that produces positive lift at zero pitching moment sounds tempting, but they generally have worse performance (lower $\frac{C_l}{C_d}$) than normal foils.
- For a swept wing one could also rotate the outmost airfoils, or use different airfoils closer to the tip, to produce a negative lift to counteract the moment from the center

[1]Without any active stabilization from a control system.

of the wing. This is referred to as a negative washout.

Whether the X8 has washout or not is not known in advance, thus the airfoils throughout the wing needs to be investigated.

D.1.1 Digitalizing the airfoils

This section will present how the airfoils were digitalized and prepared for analysis in XFLR. In order to digitalize the airfoils of the X8 throughout the wing there are two apparent options. One could cut out cross-sections of the wing and digitalize them by taking pictures of them. This is a simple method since the X8 is made of styrofoam, but would inevitably destroy the X8. Another possible method is to use a 3D model of the X8, and cut the cross-sections digitally in a CAD program. This is a more complex and time consuming procedure, but it was chosen since it will leave the X8 intact and is easily transferable to other, more complex and expensive UAVs.

Generation of airfoils from a physical model through a 3D CAD model

The process of generating airfoils from a physical object is summarized in the following:

1. **Generate a 3D scan from images of the object.** For this project the large X8 community proved useful, as a complete 3D scan was found online[2]. In principle a similar 3D scan can be created from any 3D scan software, such as Agisoft Photo-Scan[3].

2. **Convert the 3D scan to a surface.** To do further analysis a proper 3D surface is needed, while the output from a 3D scan is merely a point cloud or an STL file. A point cloud describes an object by a set of points in cartesian 3D space, while an STL file describes the object by a set of tiny surfaces decomposed as their normal and vertecies. The process of converting a point cloud or stl to a surface proved quite simple once the right tools are available: NX, a CAD program from Siemens, imports stl straight to a 3D surface[4]

3. **Aquire airfoil spline from surface model.** Once the surface model is aquired, cutting a cross-section in the X8 is acheived by creating a plane parallel to the XZ-plane at a desired distance along the Y-axis, and then creating an "Intersection Curve" between this plane and the surface. See Figure D.2. Creating multiple, parallel airfoils is a matter of generating multiple planes for NX to project the surface onto[5].

[2]`http://diydrones.com/profiles/blogs/x8-3d-scanning?id=705844%3ABlogPost%3A1405554&page=2#comments`

[3]See `http://www.agisoft.com`

[4]This was also tried with both SolidWorks and different AutoCad programs without success due to the large resolution needed for the 3D model to be accurate, limitations of the programs and possibly limited knowledge to the programs

[5]For a video tutorial on how to perform this step, see `http://nxportalen.com/wp-content/uploads//2014/Videos/ForumAnswers/ProfileFromScanSTL/wingprofile.mp4`

4. **Export spline to dat file.** Now that the airfoil spline is available, it is only a matter of converting it to a format accepted by XFLR. This is acheived by a plugin to NX that converts a curve to a txt-file of coordinates[6]. This txt-file is then converted to an XFLR readable dat-file by a simple python script. The airfoils where named e.g. “X8 20 475mm”, which means that it is airfoil number 20, located at 475mm away from the center of the X8 along the y-axis. For XFLR to read the dat-file the points have to start at the trailing edge of the airfoil towards the leading edge along the upper and then back along the lower wing surface. For a complete list of format requrements the reader is referred to Deperrois (2013).

5. **XFLR reworking.** Now the airfoil can be opened by XFLR, but before further analysis can be made all the foils should be on the same format. This will simplify comparison of the different foils, and will enable us to reconstruct a 3D plane in XFLR. By normalizing all the airfoils, they are all scaled to the same size: x coordinates ranging from 0 to 1. Another key point is that the camber line should be aligned with the x-axis. This is acheived by “de-rotating” the foil in XFLR. It should be noted that de-rotating may lead to a de-normalized foil, whereas normalizing may lead to a rotated the foil. To account for this, the process of normalizing/de-rotating/normalizing should be repeated until a acceptable compromise is found. To make it possible to include elevons on the plane, the foils covering the elevons sould be made with flaps in XFLR. See Figure D.1.

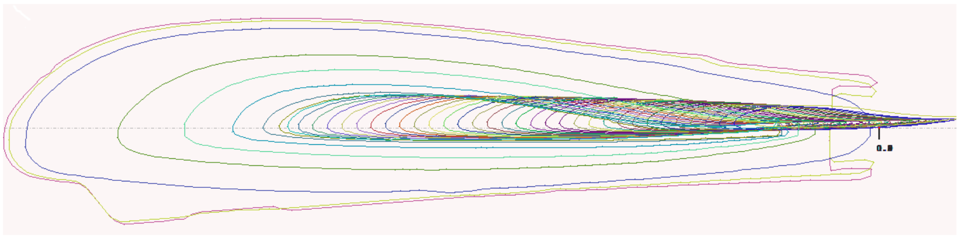


Figure D.1: Unscaled airfoils imported to XFLR

Following this procedure 41 airfoils, sepparated by 25mm from each other, were generated. In addition three foils from the winglets were made. Now that the airfoils are available in a format accepted by XFLR we are, in principle, ready to run the analysis on all the foils. However by building the 3D model first, making it possible to see exactly what analyses are needed, a lot of time in analysis can be saved.

D.1.2 3D model in XFLR

A plane in XFLR consists of one/two wings, a body, an elevator and one/two fins. Since there is no distinction between the wing and the body of the X8, it was modeled as a single wing. The winglets were also modeled as a part of the wing, with a dihedral of 76.5°. To reconstruct the 3D model from the 2D airfoils, the chord length and offset from the tip of

[6]See `http://www.eng-tips.com/viewthread.cfm?qid=272761`

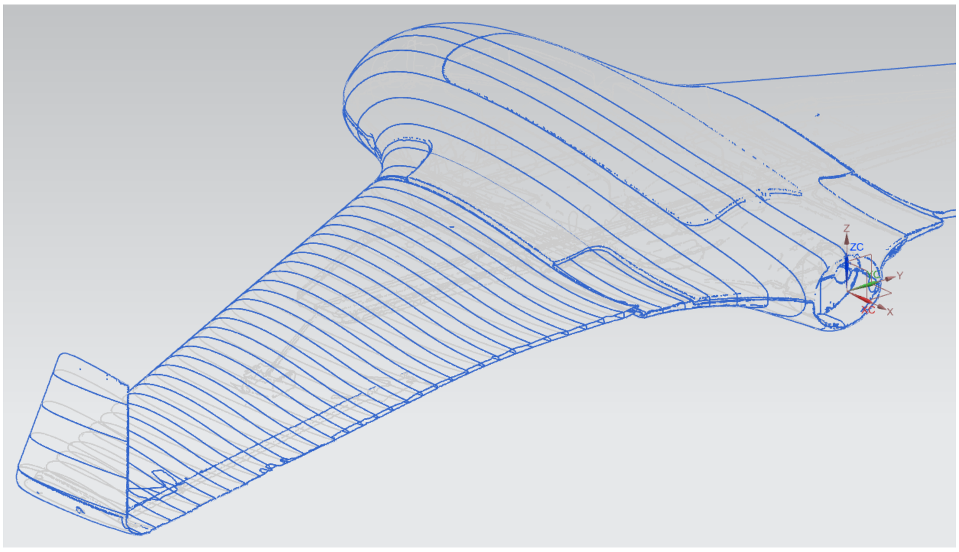


Figure D.2: The airfoil splines on the 3D surface

the plane is neededfor all the foils. These values can be found in Appendix E.1. Originally the wing was generated with one foil section for every 25mm, similar to Figure D.2. This makes the model accurate, but very complex. However there are two simplifications that can be made:

1. Find similar foils and replace them by a common foil
2. Make wider foil sections by joining adjacent sections with the same foil

These simplifications will reduce the complexity of the analysis by a lot because the number of foils that need to be analyzed is vastly reduced. Considering all the different elevons settings that should be analyzed, representing the elevon by one foil reduces the problem size dramatically. By comparison to the nearby foils it was found that the foil “X8 20 475mm” was a good representation of foils nine through 41, i.e. from the root to the tip of the wing. While the body is represented by eight different foils. By visual inspection supported by regression in Excel showed that the both the leading- and trailing edge is linear over the section of the wing covered by the elevon. This implies that these sections can be combined to a single section. After these simplifications the assembled model looks like Figure D.3. Ideally a foil that is chosen to represent many other foils should be chosen in a more scientific manner than by simple visual inspection. A possible solution is to generate an “average foil” by using least square estimation over all the foils. However this proved difficult as the x-coordinates of the different foils not necessarily in accordance. Table E.1 includes an overview of what airfoils are used in the XFLR model. Another advantage of trying to analyse the assembled model before each foil is that when XFLR fails in analysing the model (because the foil data available is insufficient/non-existing), XFLR shows what range of Reynolds numbers it lacks to complete the analysis. A lot of

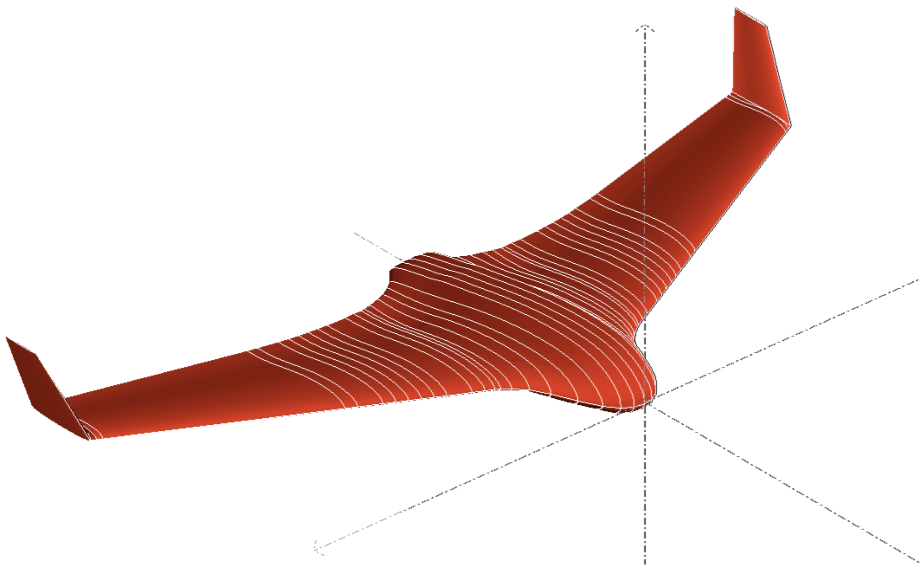


Figure D.3: Model assembled in XFLR

time can be saved by analysing the foils only at these Reynolds numbers.

D.2 CFD analysis and interpretation

This section will briefly describe the analysis setup in XFLR for both airfoil and plane analysis.

D.2.1 Airfoil analysis

The airfoils were analysed in batches of multiple foils, Reynolds numbers and α values. The settings used can be seen in Table D.1. The value for mach, NCrit and both transition locations are set to the default XFLR values. The maximum number of iterations was lowered to 100 to save time in analysis, since seemingly no analysis that had not converged after 100 iterations had converged after 200 iterations.

Table D.1: Airfoil analysis settings

Analysis type	Type 1
Reynolds number	$[10000, 1300000]$
Mach	0.00
NCrit	9.00
Top transition location	1.00
Bottom transition location	1.00
Alpha range	$[-3, 25]$
Initialize the boundary layer after unconverged points	Yes
Initialize the boundary layer after each polar calculation	No
Store OpPoints	Yes
Max iterations	200

Table D.2: Plane analysis settings

Type	1
Analysis method	Ring vortex
Viscous	Yes
Tilt.Geom.	No
Use plane inertia	Yes
Ref. dimensions for aero coefficients	Wing Planform
ρ	$1.225kg/m^3$
ν	$1.5e-05m^2/s$
Ground effects	No

D.2.2 Plane analysis

There are two main options for analysing the plane as a whole in XFLR. The *Analysis*[7] and *Stability Analysis*.

Nonlinear C_L and C_D analysis

The analysis belonging to the *Analysis* family typically analyse the plane for a range of αs, and computes the lift, drag and moment coefficients for this range of α values. These coefficients are in principle nonlinear in α. However as the assumptions of XFLR are not valid in the stall region, the analysis will not converge near stall causing the coefficients to only be known in the below-stall region. This results in almost linear C_L, whereas C_D is highly nonlinear. The typical configuration settings for this type of analysis can be found in Table D.2. The weight, center of mass, moments of inertia is found in Table 6.1, Table 6.2 and Table F.1. The mass of the wing was set equal to the weight of the styrofoam

[7]The family of analysis are simply called Analysis, assumably for historical reasons as the Stability analysis is of newer date

Table D.3: Stability analysis settings

Analysis method:	Ring vortex
Viscous analysis:	No [as the combination with elevons is impossible]
β	0.0
φ	0.0
Use plane inertia	No. [Mass, CoG, inertia according to Table F.1]
Ref.dimensions	Wing planform projected on xy plane
Wing flap angle gain 1	1.00
Wing flap angle gain 2	1.00
ρ	1.225 kg/m^3
ν	$1.5 \cdot 10^{-5} \text{m}^2/\text{s}$
Sequence	Yes, $[-4, 4]$

of the X8, while the remaining mass was placed as point masses to get the center of mass and moments of inertia as close to the measured values as possible.

Stability analysis

Another type of analysis in XFLR is the stability analysis, which is necessary to find the dynamic coefficients that depend on the angular rates and control surface deflection. It is also useful for analyzing characteristics such as phugoid-,short-period-, and Dutch roll mode. Prior to this analysis, elevons should be added to the X8 as this will provide crucial information about the control derivatives. The rest of the analysis settings are found in Table D.3. XFLR then runs through the analysis by looping through the different control inputs from the sequence provided, effectively changing the elevon deflection angle. Since XFLR only has one input for the analysis, the problem of deflecting multiple control surfaces are solved by assigning different gains to the different control surfaces. The control surfaces are numbered from left to right, so “Wing flap angle gain 1” in Table D.3 corresponds to the left elevon. Running an analysis with the gains set equal effectively reduces the elevons to elevators, while opposite gains result in pure ailerons.
Based on the required lift for the given configuration, the velocity V_{Inf} and angle of attack α are found. Once these static conditions are established, the aerodynamic derivatives are estimated. This is done by slightly perturbing the system states, one at the time, and measure the effect this has on the forces and moments. The coefficients are plugged into a slightly simplified version[8] of the longitudinal and lateral state/control matrices (B.2) from Etkin and Reid (1996). The system is now on the linearized form given by (D.3) as expressed in Deperrois (2013, p.65), where $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{F}(t)$ are the state matrix, control

[8]Assumes that $M_{\dot{w}} = Z_{\dot{w}} = 0$

matrix, and input respectively.

$$\begin{bmatrix}\dot{u}\\ \dot{w}\\ \dot{q}\\ \dot{\theta}\end{bmatrix} = \mathbf{A}_{long}\begin{bmatrix}u\\ w\\ q\\ \theta\end{bmatrix} + \mathbf{B}_{long}\cdot\mathbf{F}(t) \tag{D.1}$$

$$\begin{bmatrix}\dot{v}\\ \dot{p}\\ \dot{r}\\ \dot{\varphi}\end{bmatrix} = \mathbf{A}_{lat}\begin{bmatrix}v\\ p\\ r\\ \varphi\end{bmatrix} + \mathbf{B}_{lat}\cdot\mathbf{F}(t) \tag{D.2}$$

$$\tag{D.3}$$

As the characteristics of the system can be found from the characteristic equation, the rest is just simple linear algebra. Passing the state matrices to the Matlab function *damp(A)* yields the poles, damping ratios, frequencies and time constants for all the modes of the system.
The control matrices describe how the different states of the system react to inputs. Since XFLR only can change the control surfaces in pure elevator or pure aileron configuration, the analysis must be run twice. One time in “elevator mode” with the gains as in Table D.3, and then again with the flap angle gains set to 1 and -1 making it an aileron analysis. Naturally these two analysis will have similar dynamics, but very different control matrices since one control matrix corresponds to elevators the other to ailerons.
For the model to be useful, the two models should be merged into one. Since the dynamics of the two models are assumed to be of similar characteristics, simply choosing one of them might provide a decent basis for further investigation. If it is assumed that the X8 will operate in level flight for most of the time, the most valid dynamics will be those from the elevator model. When it comes to finding the control matrices for the final model, the control matrices from the elevator and aileron model can simply be stacked as in Equation (D.4). This is valid both for the longitudinal and lateral matrices. This comes from the fact that in the pure elevator and aileron models, $\mathbf{F}(t)$ is only a single input; elevator or aileron respectively. However in the final, merged model the $\mathbf{F}(t)$ will be two-dimensional, including both elevator and aileron.

$$\mathbf{B}_{final} = [\mathbf{B}_{elevator}, \mathbf{B}_{aileron}] \tag{D.4}$$

Appendix E

Airfoil analysis

E.1 Wing chord and offset

Table E.1 gives the values for chord length and offset for the X8 that has been used to build the XFLR model. Both the trailing and leading edges of the wing are approximately linear from $425mm$ and to the wingtip. To simplify inclusion of elevons in the XFLR model, no foils between the start and the end of the elevons are included. Since this increases the statistical importance of the start and end of the elevons, the measurements for these foils were verified by comparison to neighboring foils. Approximations are due to bad accuracy of the 3d scan in some regions. The table also shows what foils were used in the actual XFLR model, and what foils that where chosen to represent the foils that are not used. At 200mm there was a slight gap in the 3D model, so foil number 9 was replaced by “Wingroot,body” at 195mm and “Wingroot,wing” at 205mm.

E.2 Airfoils

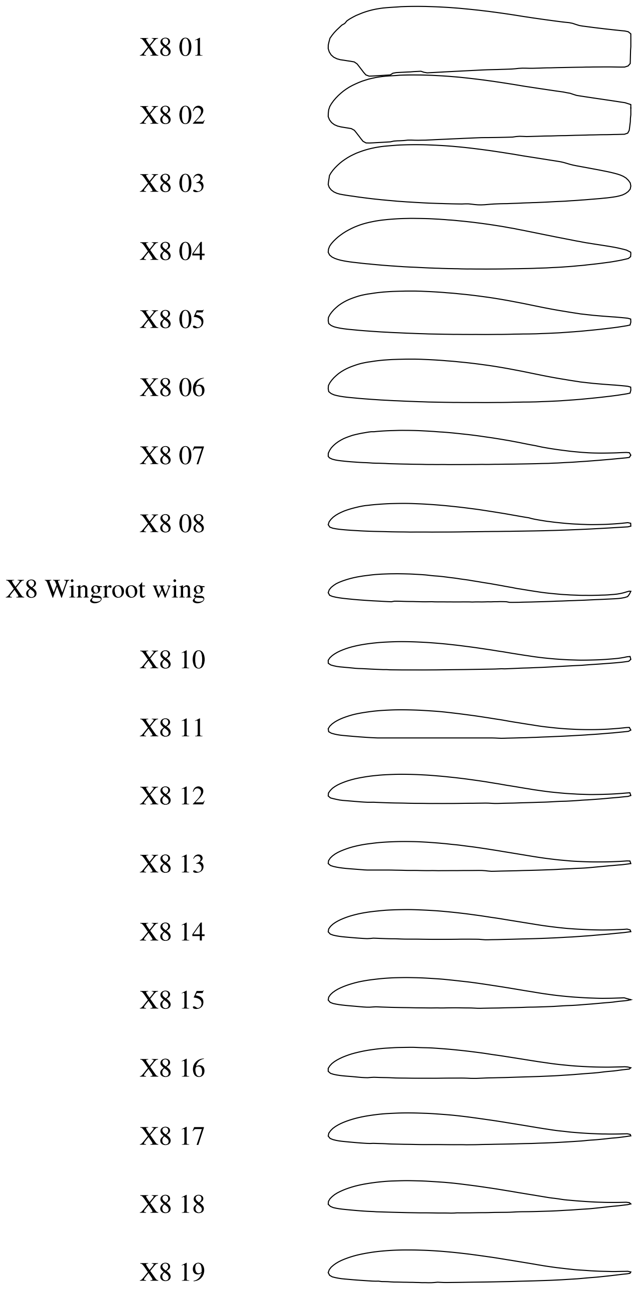


Figure E.1: Visual comparison of X8 foils from center until 450mm

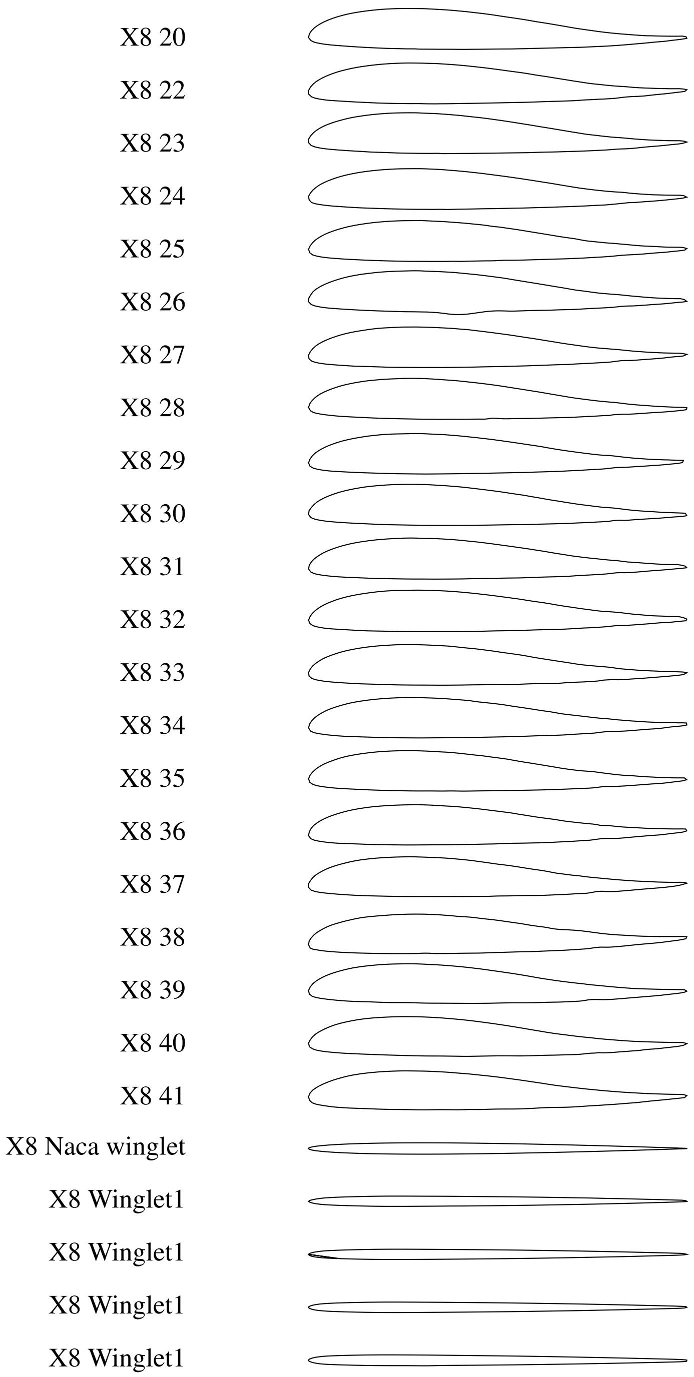


Figure E.2: Visual comparison of X8 foils from 475mm until wingtip

Table E.1: Wing and chord measurements from 3d scan (in millimeters)

Airfoil number	Dist. from center	Chord	LE Offset	Comment	Repr. by
1	0	779,000	0,000		3
2	25	774,600	4,400	Estimated chord	3
3	50	753,348	19,700		3
4	75	692,599	49,900		4
5	100	620,702	102,180		5
6	125	547,641	161,267		6
7	150	494,808	204,268		7
8	175	461,720	231,875		8
Wingroot,body	195	445,829	245,837		20
Wingroot,wing	205	436,871	253,573		20
10	225	428,979	262,843		20
11	250	414,192	275,748		20
12	275	403,205	288,663		20
13	300	391,341	301,457		20
14	325	380,609	314,544		20
15	350	370,033	327,433		20
16	375	360,294	340,223		20
17	400	350,901	353,161		20
18	425	343,200	366,155		20
19	450	335,133	379,176		20
20	475	327,885	392,087		20
21	500	320,584	405,083	Start of elevon	20
22	525	314,024	418,015		20
23	550	308,711	430,978		20
24	575	301,146	443,946		20
25	600	293,704	456,911		20
26	625	286,423	469,913		20
27	650	282,237	482,769		20
28	675	275,947	495,724		20
29	700	268,653	508,637		20
30	725	260,842	521,634		20
31	750	253,779	534,430		20
32	775	249,922	547,322	Estimated chord	20
33	800	243,120	560,308		20
34	825	236,956	573,171		20
35	850	230,093	586,125		20
36	875	222,698	599,141		20
37	900	217,288	611,909		20
38	925	210,883	624,862		20
39	950	203,884	637,884		20
40	975	196,868	650,844		20
	987,5			End of elevon	20
41	1000	190,113	663,906		20

E.3 XFLR analysis raw data

Output from the different XFLR stability analysis can be seen in Listings E.1 and E.2. Please note that, of space considerations, this is merely an excerpt from the actual output.

Listing E.1: XFLR output:Elevons as ailerons

```
Launching the 3D Panel Analysis....
Plane Name
Type 7 — Stability polar
Ref. area  =      0.743 m
Ref. span  =  2099.371 mm
Ref. chord =   408.333 mm

Counted  616 panel elements

   Mass=        3.364 kg

   ___Center of Gravity Position — Body axis____
    CoG_x=       0.4400 m
    CoG_y=      −0.0000 m
    CoG_z=       0.0000 m

   ___Inertia — Body Axis — CoG Origin____
    Ibxx=        1.223 kg.m2
    Ibyy=       0.1702 kg.m2
    Ibzz=       0.8808 kg.m2
    Ibxz=       0.9343 kg.m2


   Solving the problem...

      Calculation for control position −4.00
         Rotating the flap by  4.00 , total angle is  4.00
         Rotating the flap by −4.00 , total angle is −4.00
      Creating the unit RHS vectors...
      Creating the influence matrix...
      Performing LU Matrix decomposition...
      Solving the LU system...
      Searching for zero−moment angle... Alpha=4.12775
      Creating source strengths...
      Calculating doublet strength...
      Calculating speed to balance the weight... VInf = 15.05709 m/s

      ___Inertia — Stability Axis — CoG Origin____
      Isxx=        1.087
      Isyy=       0.1702
      Iszz=        1.017
      Isxz=       0.9492

      Calculating the stability derivatives
         Creating the RHS translation vectors
         Creating the RHS rotation vectors
         LU solving for RHS
         Calculating forces and derivatives
      Calculating the control derivatives

      Longitudinal derivatives
      Xu=    −0.073907          Cxu=    −0.010788
      Xw=       1.3293          Cxa=      0.19404
      Zu=      −4.3849          Czu= −0.00021247
      Zw=      −28.099          CLa=       4.1015
      Zq=      −5.4499          CLq=       3.8964
      Mu=  3.8184e−08           Cmu=    1.365e−08
      Mw=     −0.70396          Cma=     −0.25165
      Mq=     −0.75106          Cmq=       −1.315
      Neutral Point position=   0.46505 m


      Lateral derivatives
      Yv=      −1.2878          CYb=     −0.18798
      Yp=     −0.84288          CYp=     −0.11721
      Yr=      0.68872          CYr=     0.095773
      Lv=        −1.54          Clb=     −0.10708
      Lp=      −6.0671          Clp=     −0.40188
      Lr=       1.5072          Clr=     0.099832
      Nv=      0.34958          Cnb=     0.024306
      Np=     −0.37352          Cnp=    −0.024741
      Nr=     −0.18908          Cnr=    −0.012524

      Control derivatives
      Xde=      0.83621         CXde=   0.0081065
      Yde=       −4.395         CYde=   −0.042606
      Zde=     −0.20068         CZde=  −0.0019455
      Lde=      −39.554         CLde=    −0.18265
```

```
Mde=    -0.018689          CMde=  -0.00044371
Nde=       1.0025          CNde=     0.0046291


-----State matrices----------
 Longitudinal state matrix
      -0.0219699          0.395168               0             -9.81
        -1.30347          -8.35273          13.437                 0
    2.24304e-07          -4.13525        -4.41191                 0
               0                 0               1                 0
 Lateral state matrix
       -0.382816         -0.250558        -14.8524              9.81
        -6.04239          -31.9415         6.62472                 0
        -5.29721          -30.1873         5.99873                 0
               0                 1               0                 0

-----Control Matrices----------
 Longitudinal control matrix
      0.2485753
  -0.05965534
    -0.1097854
             0

 Lateral control matrix
      -1.306466
     -192.261
    -178.5048
             0



---Longitudinal modes----

Eigenvalue:      -6.409+    -7.209i   |      -6.409+     7.209i   |    0.01602+   -0.7536i   |    0.01602+    0.7536i
                 ----------------------------------------------------------------------------------------------------
Eigenvector:          1+          0i   |          1+          0i   |          1+          0i   |          1+          0i
                  12.25+     -15.86i   |      12.25+      15.86i   |   -0.06312+ 0.004058i   |   -0.06312+-0.004058i
                 -6.644+     -8.865i   |     -6.644+      8.865i   |    0.05792+ 0.006068i   |    0.05792+-0.006068i
                  1.144+    0.09585i   |      1.144+   -0.09585i   |  -0.006415+   0.07699i   |  -0.006415+  -0.07699i



---Lateral modes----

Eigenvalue:      -28.79+          0i   |     0.0442+          0i   |      1.209+    -2.727i   |      1.209+     2.727i
                 ----------------------------------------------------------------------------------------------------
Eigenvector:          1+          0i   |          1+          0i   |          1+          0i   |          1+          0i
                  1.939+          0i   |   0.09316+          0i   |    -0.2081+   0.01132i   |    -0.2081+  -0.01132i
                  1.835+          0i   |      1.362+          0i   |    -0.1246+    0.1423i   |    -0.1246+   -0.1423i
               -0.06737+          0i   |      2.108+          0i   |   -0.03175+  -0.06224i   |   -0.03175+   0.06224i

Calculating aerodynamic coefficients in the far field plane
  Calculating point    4.13 ....
Computing On-Body Speeds...
Computing Plane for alpha=   4.13
 Calculating aerodynamic coefficients...
   Calculating wing...Plane Name_Wing


------Finished operating point calculation for control position -4.00--------
```

Listing E.2: XFLR output:Elevons as elevators

```
Launching the 3D Panel Analysis....
Plane Name
Type 7 — Stability polar
Ref. area  =      0.743  m
Ref. span  =  2099.371  mm
Ref. chord =   408.333  mm

Counted  616 panel elements

   Mass=        3.364  kg

   ___Center of Gravity Position — Body axis____
    CoG_x=       0.4400  m
    CoG_y=      −0.0000  m
    CoG_z=       0.0000  m

   ___Inertia — Body Axis — CoG Origin____
    Ibxx=        1.223  kg.m2
    Ibyy=       0.1702  kg.m2
    Ibzz=       0.8808  kg.m2
    Ibxz=       0.9343  kg.m2


   Solving the problem...

      Calculation for control position −4.00
         Rotating the flap by −4.00 , total angle is −4.00
         Rotating the flap by −4.00 , total angle is −4.00
      Creating the unit RHS vectors...
      Creating the influence matrix...
      Performing LU Matrix decomposition...
      Solving the LU system...
      Searching for zero−moment angle... Alpha=8.75399
      Creating source strengths...
      Calculating doublet strength...
      Calculating speed to balance the weight...VInf = 10.96556 m/s

      ___Inertia — Stability Axis — CoG Origin____
      Isxx=       0.9339
      Isyy=       0.1702
      Iszz=         1.17
      Isxz=       0.9425

      Calculating the stability derivatives
         Creating the RHS translation vectors
         Creating the RHS rotation vectors
         LU solving for RHS
         Calculating forces and derivatives
      Calculating the control derivatives

      Longitudinal derivatives
      Xu=      −0.18719            Cxu=     −0.037519
      Xw=         1.828            Cxa=       0.36639
      Zu=       −6.0217            Czu=  −0.00055009
      Zw=       −19.947            CLa=         3.998
      Zq=       −4.0157            CLq=        3.9423
      Mu=   3.6255e−09             Cmu=   1.7796e−09
      Mw=      −0.53017            Cma=      −0.26024
      Mq=      −0.54315            Cmq=       −1.3058
      Neutral Point position=    0.46658 m


      Lateral derivatives
      Yv=      −0.88649            CYb=      −0.17768
      Yp=      −0.38129            CYp=     −0.072806
      Yr=       0.55918            CYr=       0.10677
      Lv=       −1.3941            Clb=       −0.1331
      Lp=       −4.3536            Clp=      −0.39597
      Lr=        1.7302            Clr=       0.15737
      Nv=       0.25111            Cnb=      0.023974
      Np=      −0.71782            Cnp=     −0.065288
      Nr=      −0.14136            Cnr=     −0.012858

      Control derivatives
      Xde=        −1.2321          CXde=     −0.022521
      Yde=   1.1804e−08            CYde=   2.1576e−10
      Zde=        −33.958          CZde=      −0.62069
      Lde=   1.5865e−08            CLde=   1.3813e−10
      Mde=        −6.6255          CMde=      −0.29658
      Nde=  −2.3307e−09            CNde=  −2.0293e−11


      _____State matrices__________
       Longitudinal state matrix
           −0.0556452           0.543397              0           −9.81
             −1.79005           −5.92948        9.77184               0
          2.12971e−08           −3.11436       −3.19058               0
                    0                  0              1               0
```

```
  Lateral state matrix
        -0.263524          -0.113344           -10.7993             9.81
         -6.82684           -28.2506            9.25866                0
         -5.28565           -23.3749            7.33876                0
                0                  1                  0                0

-----Control Matrices----------
  Longitudinal control matrix
      -0.3662678
     -10.09439
      -38.9196
             0

  Lateral control matrix
    3.509013e-09
  8.012121e-08
  6.256032e-08
             0



---Longitudinal modes----

Eigenvalue:      -4.629+   -5.393i  |      -4.629+    5.393i  |     0.04131+     -1.04i  |     0.04131+      1.04i
                 -------------------------------------------------------------------------------------------------
Eigenvector:         1+        0i  |          1+        0i  |          1+        0i  |          1+        0i
                 7.061+     -8.4i  |      7.061+      8.4i  |    -0.1217+  0.01431i  |    -0.1217+ -0.01431i
                -3.513+   -5.014i  |     -3.513+    5.014i  |     0.1103+   0.0217i  |     0.1103+  -0.0217i
                0.8573+  0.08446i  |     0.8573+ -0.08446i  |   -0.01663+   0.1068i  |   -0.01663+  -0.1068i



---Lateral modes----

Eigenvalue:      -23.36+       0i  |    0.06423+        0i  |      1.059+   -2.545i  |      1.059+    2.545i
                 -------------------------------------------------------------------------------------------------
Eigenvector:         1+        0i  |          1+        0i  |          1+        0i  |          1+        0i
                 2.425+        0i  |    0.06933+        0i  |    -0.2863+  0.02292i  |    -0.2863+ -0.02292i
                 2.019+        0i  |     0.9494+        0i  |    -0.1627+   0.1513i  |    -0.1627+  -0.1513i
               -0.1038+        0i  |      1.079+        0i  |   -0.04757+ -0.09268i  |   -0.04757+  0.09268i

Calculating aerodynamic coefficients in the far field plane
  Calculating point    -0.70 ....
Computing On-Body Speeds...
Computing Plane for alpha=   8.75
  Calculating aerodynamic coefficients...
    Calculating wing... Plane Name_Wing


------Finished operating point calculation for control position -4.00--------
```

Appendix F

Moment of inertia and center of gravity calculations

Both the moment of inertia and the center of gravity are important properties when it comes to the stability of an airplane. From Newton's second law of motion applied to rotation,$\tau = I\alpha$, it is clear that moment of inertia relates the torque around an axis to the angular acceleration around the same axis. The arm between the center of gravity and the center of pressure is proportional to the moments acting on the airplane, so the location of the center of gravity is key in stability analysis.

F.1 Pendulum method: Finding I_{ij}

To identify the moment of inertia I_{ij}, the X8 is tied to a string of known length L attached to the ceiling, forming a pendulum of unknown inertia, see Figure F.1. If the string is assumed massless and of constant length, that the plane does not rotate relative to the string ($\dot{\varphi} = 0$) the moment of inertia can be calculated using the Lagrange formalism, see

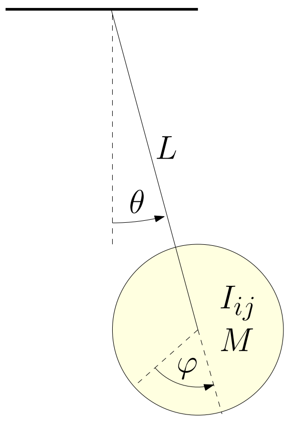


Figure F.1: Pendulum of mass M and inertia I_{ij} hanging from a massless rod

e.g. Goldstein et al. (2002):

$$\mathcal{K} = \frac{1}{2}ML^2\dot{\theta}^2 + \frac{1}{2}I\left(\dot{\theta}+\dot{\varphi}\right)^2 = \frac{1}{2}\left(ML^2+I\right)\dot{\theta}^2 \tag{F.1}$$

$$\mathcal{P} = -MLg\cos\theta \tag{F.2}$$

$$\mathcal{L} = \mathcal{K} - \mathcal{P} = \frac{1}{2}\left(ML^2+I\right)\dot{\theta}^2 - MLg\cos\theta \tag{F.3}$$

$$\frac{\partial\mathcal{L}}{\partial\theta} = -MLG\sin\theta \tag{F.4}$$

$$\frac{\partial\mathcal{L}}{\partial\dot{\theta}} = \left(I+ML^2\right)\dot{\theta} \tag{F.5}$$

$$\tag{F.6}$$

It is then assumed that the energy of the pendulum is dissipated by the dissipative function $\mathcal{F}$

$$\mathcal{F} = \frac{1}{2}Kv^2 = \frac{1}{2}KL^2\dot{\theta}^2 \tag{F.7}$$

$$\frac{\partial\mathcal{F}}{\partial\dot{\theta}} = KL^2\dot{\theta} \tag{F.8}$$

$$\frac{d}{dt}\frac{\partial L}{\partial\dot{\theta}} - \frac{\partial L}{\partial\theta} + \frac{\partial\mathcal{F}}{\partial\dot{\theta}} = 0 \tag{F.9}$$

$$\left(I+ML^2\right)\ddot{\theta} + MLg\sin\theta + KL^2\dot{\theta} = 0 \tag{F.10}$$

$$\tag{F.11}$$

For small θ, $\sin\theta \approx \theta$

$$\left(I + ML^2\right)\ddot{\theta} + MLg\theta + KL^2\dot{\theta} = 0 \tag{F.12}$$

$$\ddot{\theta} + \frac{KL^2}{I + ML^2}\dot{\theta} + \frac{MLg}{I + ML^2}\theta = 0 \tag{F.13}$$

$$\ddot{\theta} + 2\zeta\omega_n\dot{\theta} + \omega_n^2\theta = 0 \tag{F.14}$$

(F.14) can be solved by e.g. Laplace. Assuming $\dot{\theta}(0) = 0$ yields the general solution

$$\theta(t) = Ae^{-\zeta\omega_n t}\sin(\sqrt{1-\zeta^2}\omega_n t) = Ae^{-\frac{t}{\tau}}\sin(\omega_d t) \tag{F.15}$$

Here $\omega_n\zeta = \frac{1}{\tau}$ and $\omega_d = \sqrt{1-\zeta^2}\omega_n$, yelding $\omega_n^2 = (\omega_n\zeta)^2 + \left(\omega_n\sqrt{1-\zeta^2}\right)^2 = \frac{1}{\tau}^2 + (\omega_d)^2$. A is a constant depending on ω_n, ζ and the initial value of the system. By measuring the average time T_p for one oscillations over e.g. 10 periods, the damped frequency can be found from

$$\omega_d = \frac{2\pi}{T_p} \tag{F.16}$$

Since $e^{-5} \approx 0$ it can be assumed that the oscillations of the pendulum have died out after $t = 5\tau$. The undamped frequency can be found from:

$$\frac{1}{\tau} = \frac{5}{T_f} \tag{F.17}$$

(F.13) and (F.14) results in

$$\omega_n^2 = \frac{MLg}{I + ML^2} = \frac{1}{\tau}^2 + \omega_d^2 \tag{F.18}$$

$$I = ML\frac{1}{\omega_n^2}g - ML^2 \tag{F.19}$$

This procedure can be used to find the moments of inertia I_{xx}, I_{yy} and I_{zz} by aligning the respective axis of the UAV with the rotation axis of θ. Following the same principle, the products of inertia $I_{ji} = I_{ij}$ can be found by aligning the axis of rotation with an axis halfway between the i and j axis, see Teimourian and Firouzbakht (2013). See figures F.2, F.3 and F.4 to see how the plane was hung for the I_{xz}, I_{xx}/I_{yy} and I_{zz} experiments respectively. For more pictures on how to perform the pendulum test, the reader is referred to Jodeh (2006) or Paw (2009). We note that in the experiments it is important that the assumptions made in this derivation is followed. Particularly keeping $\dot{\theta} = \dot{L} = 0$.

F.2 Finding center of gravity

Since flying wings are known to be very sensitive to the location of the centre of gravity, special care should be taken when identifying it. Therefore, information from three different sources is considered:

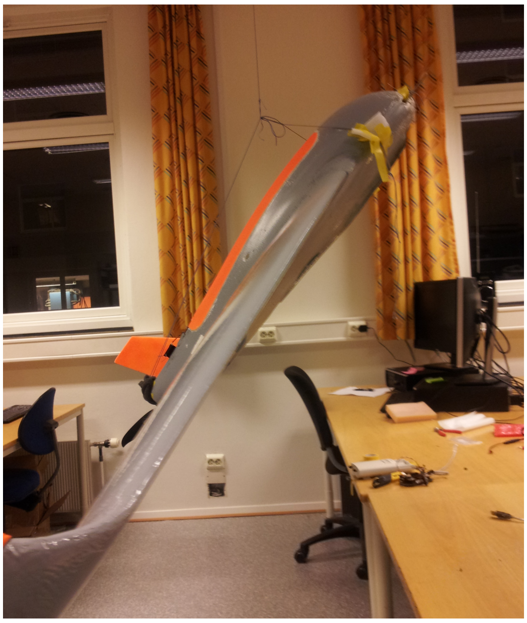


Figure F.2: Setup for performing the pendulum test for finding I_{xz}

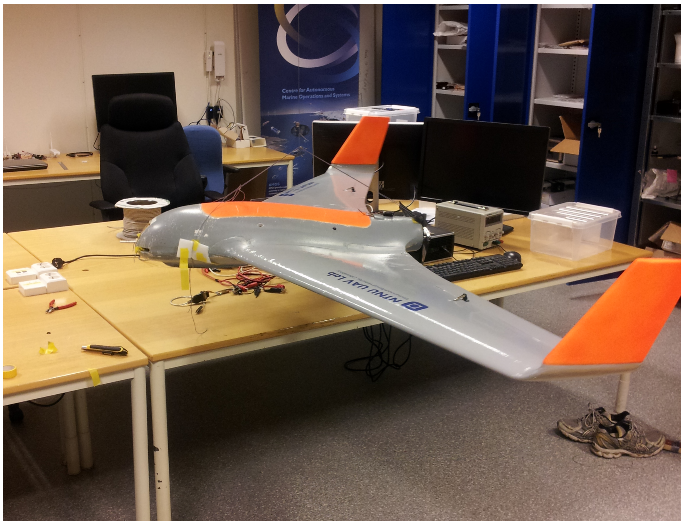


Figure F.3: The setup for the pendulum test for finding I_{xx} and I_{yy}

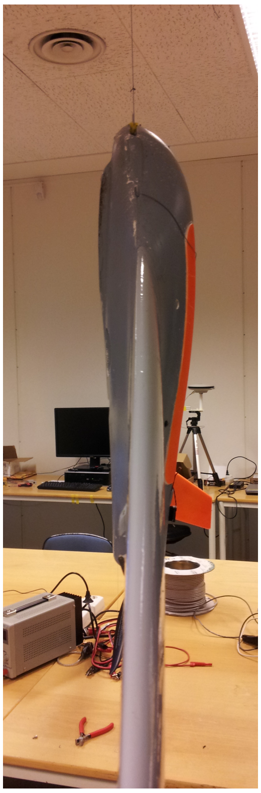


Figure F.4: The setup for the pendulum test for finding I_{zz}

Experimental data The pilot at the UAV-lab needs to make the X8 stable prior to each flight, so good knowledge about the location of the center of gravity is expected.

The pendulum test If the X8 is hung correctly, the center of gravity should be located along a straight line extending the pendulum.

3D modelling Upon construction of the X8, XFLR provides estimates of the center of gravity from geometric considerations. It is also possible to add point masses to better represent the mass distribution of the X8

During the pendulum tests the X8 was loaded with a minimal payload consisting of

- Pixhawk with standard u-blox GPS module and 3DR power module
- Hacker Master Spin 66 pro ESC
- Two Zippy Compact LiPo 5000mAh 25C batteries
- Hacker A40 12S V2 14-pole motor
- RC receiver
- SOM9331; router based on Atheros AR9331 Wi-Fi System-On-Chip

Also see Figure F.5. This setup has the center compartment of the payload bay free. This bay is located close to the center of gravity which means that extra, task specific payload can be added with minimal influence on the inertia. However the extra payloads influence on the inertia is something that should be taken into account for each individual hardware setup.

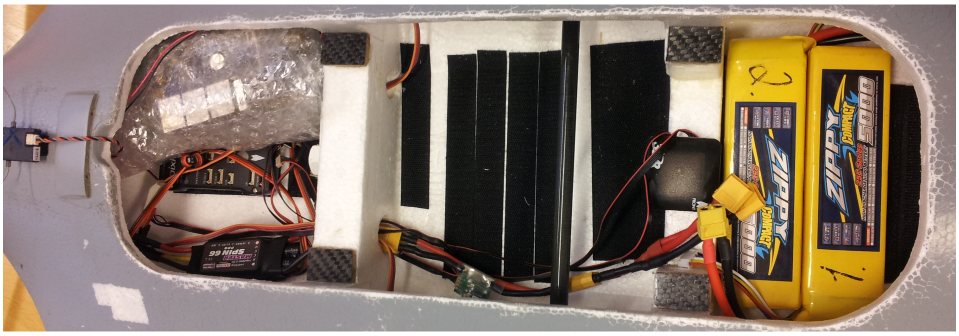


Figure F.5: The minimal payload of the X8 with which the pendulum tests were performed

F.3 Calculations and intermediate results

Values relating to I_{xz} are found by averaging data from the nose up and -down experiments.

Table F.1: Inertia results

	X	Y	Z	XZ
ω_d	2.1262	2.2084	2.4268	2.3200
ω_{ud}	0.0113	0.0010	0.0110	0.0147
ω_n	2.1262	2.2083	2.4268	2.3199
I	1.2290	0.1702	0.8808	0.9343

Table F.2: Finding T_p and T_f for I_{xx}

$T_p \cdot N$	N	T_p	T_f	L
295.76	100	2.9576	433	1.99
296.05	100	2.9605	440	1.99
294.73	100	2.9473	451	1.99
Average		2.95513	441.33	

Table F.3: Finding T_p and T_f for I_{yy}

$T_p \cdot N$	N	T_p	T_f	L
83.37	30	2.779	510	1.99
86.31	30	2.877	505	1.99
230.36	80	2.8795	495	1.99
Average		2.845167	503.33	

Table F.4: Finding T_p and T_f for I_{zz}

$T_p \cdot N$	N	T_p	T_f	L
26.6	10	2.66	483.57	1.49
100.76	40	2.519	450	1.49
77.65	30	2.5883	426.45	1.49
Average		2.5891	453.34	

Table F.5: Finding T_p and T_f for I_{xz}. nose up

$T_p \cdot N$	N	T_p	T_f	L
107.71	40	2.6928	315	1.625
110.28	41	2.6898	345	1.625
Average		2.6913	330	

Table F.6: Finding T_p and T_f for I_{xz}, nose down

$T_p \cdot N$	N	T_p	T_f	L
190.34	70	2.7191	370	1.685
136.62	50	2.7324	390.00	1.685
Average		2.7258	350.00	

Appendix G

Running the simulators

G.1 Simulink simulator

G.1.1 Command to run FlightGear

The following command was used to launch FlightGear : The argument for aircraft is the

Listing G.1: Command to start FlightGear with Simulink simulator

```
fgfs --aircraft=HL20 --fdm=null
    --native-fdm=socket,in,30,78.91.7.39,5502,udp
    --native-ctrls=socket,out,30,129.241.154.164,5505,udp --fog-fastest
    --disable-clouds --start-date-lat=2004:06:01:09:00:00 --disable-sound
    --in-air --enable-freeze --airport=KSFO --runway=10L --altitude=7224
    --heading=113 --offset-distance=4.72 --offset-azimuth=0
```

name of the model that should be displayed. This will load the corresponding configuration file `*model*-set.xml` from `flightgearroot/data/Aircraft/*model*`

G.2 Pilot-centric software-in-the-loop simulator

In order to run the pilot-centric SITL, the following programs should be started in the given order:

1. FlightGear

2. ArduPlane
3. JSBSim
4. DUNE
5. Mavproxy

G.2.1 Command to run FlightGear

Listing G.2: Command to start FlightGear for SITL

```
fgfs --native-fdm=socket,in,10,,5503,udp --fdm=external
    --aircraft=Rascal110-JSBSim --fg-aircraft="./aircraft" --airport=KSFO
    --geometry=1900x1080 --bpp=32 --disable-anti-alias-hud
    --disable-hud-3d --disable-horizon-effect --timeofday=noon
    --disable-sound --disable-fullscreen --disable-random-objects
    --disable-ai-models --fog-disable --disable-specular-highlight
    --disable-anti-alias-hud
```

G.2.2 Command to run ArduPlane

Listing G.3: Command to start ArduPlane

```
/tmp/ArduPlane.build/ArduPlane.elf
```

G.2.3 Command to run JSBSim

Listing G.4: Command to start JSBSim

```
ardupilot/Tools/autotest/jsbsim/runsim.py
    --home=37.621313,-122.378955,5,0 --script=jsbsim/X8_test.xml
    --aircraft=X8 --fgout=127.0.0.1:5503
```

G.2.4 Command to run DUNE

The following command will start up DUNE in ArduPlane software-in-the-loop mode (`AP-SIL`) with the configurations of `ntnu-x8-001.ini`, that can be located in either

DH/dune/etc, DH/dune/user or any of their subfolders. DH refers to the DUNE home directory.

Listing G.5: Command to start DUNE

```
DH/build/dune -c ntnu-x8-001 -p AP-SIL
```

G.2.5 Command to run mavproxy

Listing G.6: Command to start Mavproxy

```
mavproxy.py --master tcp:127.0.0.1:5762
```

G.3 Mission-centric software-in-the-loop simulator

In order to run the mission-centric SITL, the following programs should be started in the given order with the given adjustments from Appendix G.2:

ArduPlane

JSBSim no longer needs the --fgout argument given in Listing G.4

DUNE

Mavproxy should use port 5763 instead of 5762

In addition, Neptus should be run by executing neptus.sh from within the Neptus folder. A minimal DUNE configuration file that connects ArduPlane with Neptus is given in Listing G.7.

Listing G.7: Minimal DUNE configuration file for running SITL

```
[Require ../../etc/uav/ardupilot.ini]

 [General]
 Vehicle                                   = x8-01

 [Control.UAV.Ardupilot/AP-SIL]
 Debug Level = None

 [Control.UAV.Ardupilot/AP-SIL]
 Ardupilot Tracker                         = True
```

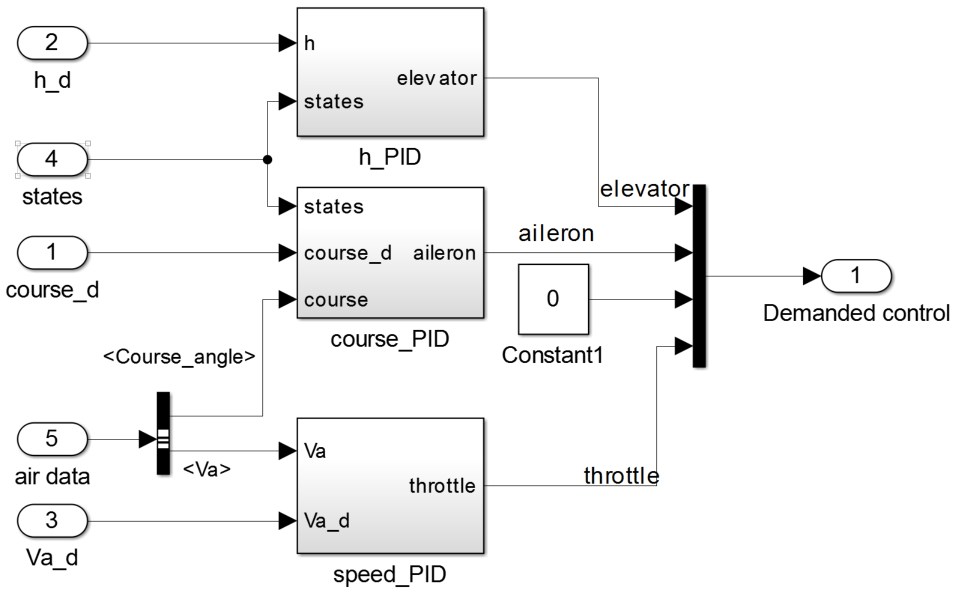


Figure G.1: The structure of the PID controller in Simulink

Appendix H

Collection of parameters

H.1 Motor parameters

The motor parameters are given in Table H.1[1]

Table H.1: Motor parameters for the Hacker A40-12S V2 14-Pole

Powerange	max. 900W (15 sec.)
Idle Current @ 8,4V	2,0A
Resistance (Ri)	0,017 Ω
RPM/Volt (kv)	610 U/min-1
Weight	208g
Diameter	41,7 mm
Length	42 mm
Poles	14
recom. Speedcontroler	40A to 70A Brushless
recom. Timing	$20° - 25°$
Shaft Diameter	5 mm

[1]Source: `http://www.hacker-motor-shop.com/e-vendo.php?shop=hacker_e&a=article&ProdNr=33726606&t=3&c=310&p=310`

H.2 Battery parameters

The battery parameters are given in Table H.2[2]. The assumed configuration has two of these batteries.

Table H.2: Battery parameters for the ZIPPY Compact 5000mAh 4S 25C Lipo Pack

Capacity	5000mAh
Voltage	4S1P / 4 Cell / 14.8V
Discharge	25C Constant / 35C Burst
Weight	488g (including wire, plug & case)
Dimensions	162x29x46mm
Balance Plug	JST-XH

[2]Source: `http://www.hobbyking.com/hobbyking/store/__21371__ZIPPY_Compact_5000mAh_4S_25C_Lipo_Pack.html`

Bibliography

Anderson, J. D. (1989). *Introduction to Flight*. 3 edition.

Andersson, J. (2013). *A General-Purpose Software Framework for Dynamic Optimization*. Phd thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium.

ArduPilot (2015). ArduPlane web page. `http://plane.ardupilot.com/`. Accessed: 23.06.2015.

Basler, M., Spott, M., Buchanan, S., Berndt, J., Buckel, B., Moore, C., Olson, C., Perry, D., Selig, M., and Walisser, D. (2015). The FlightGear Manual.

Beard, R. W. (2014). UAVBOOOK Supplement: Additional thoughts on propeller thrust model. `http://uavbook.byu.edu/lib/exe/fetch.php?media=shared:propeller_model.pdf`.

Beard, R. W. and McLain, T. W. (2012). *Small Unmanned Aircraft*. Princeton University Press.

Cory, R. and Tedrake, R. (2008). Experiments in Fixed-Wing UAV Perching. *AIAA Guidance, Navigation and Control Conference and Exhibit*, pages 1–12.

Crowther, W. J. and Prassas, K. (1999). Post Stall Landing for Field Retrieval of UAVs. In *14th Bristol International unmanned air vehicle systems conference*, number 1999.

Deperrois, A. (2009). About XFLR5 calculations and experimental measurements. `http://www.xflr5.com/docs/Results_vs_Prediction.pdf`.

Deperrois, A. (2010). F. A. Q: Why do I get the message " Point is out of the flight envelope ?". `http://www.xflr5.com/docs/Point_Out_Of_Flight_Envelope.pdf`. Accessed: 25.09.2014.

Deperrois, A. (2013). XFLR5 Analysis of foils and wings operating at low reynolds numbers, 2009. `http://sourceforge.net/projects/xflr5/files/Guidelines.pdf/download`.

Devesa, B., Jourdan, C., and Marc, C. (2004). Ground-Effect Identification and Autoland System Validation from Flight Data. *Journal of Aircraft*, 41(4):730–734.

Diedrich, F. W. and Drischler, J. A. (1957). Effect of Spanwise Variations in Gust Intensity on the Lift Due to Atmospheric Turbulence. Technical report, National Advisory Committee for Aeronautics, Washington.

Diehl, M., Bock, H., Schlder, J. P., Findeisen, R., Nagy, Z., and Allgwer, F. (2002). Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4):557–585.

Diehl, M., Bock, H. G., Diedam, H., and Wieber, P. (2006). Fast Direct Multiple Shooting Algorithms for Optimal Robot Control. In Diehl, M. and Mombaur, K., editors, *Fast Motions in Biomechanics and Robotics*, pages 65–93. Springer Berlin Heidelberg.

Diehl, M., Ferreau, H. J., and Haverbeke, N. (2009). Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation. In *Nonlinear Model Predictive Control - Towards New Challenging Applications*, pages 391–417. Springer.

Egeland, O. and Gravdahl, J. T. (2002). *Modeling and simulation for automatic control*. Marine Cybernetics Trondheim, Norway.

Etkin, B. and Reid, L. D. (1996). *Dynamics of Flight: Stability and Control*, volume 12. John Wiley & Sons, INC., New York, 3rd edition.

Fossen, T. I. (2011). *HANDBOOK OF MARINE CRAFT HYDRODYNAMICS AND MARINE CRAFT HYDRODYNAMICS AND*.

Gautam, A., Sujit, P., and Saripalli, S. (2014). A survey of autonomous landing techniques for UAVs. *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1210–1218.

Glassman, E., Desbiens, A. L., Tobenkin, M., Cutkosky, M., and Tedrake, R. (2012). Region of attraction estimation for a perching aircraft: A Lyapunov method exploiting barrier certificates. *2012 IEEE International Conference on Robotics and Automation*, pages 2235–2242.

Goldstein, H., Poole, C. P., and Safko, J. L. (2002). *Classical Mechanics*. Addison Wesley, 3 edition.

Grüne, L. and Pannek, J. (2011). *Nonlinear Model Predictive Control: Theory and Algorithms*. Communications and Control Engineering. Springer London.

Gryte, K., Mathisen, S. H., Johansen, T. A., and Fossen, T. I. (Submitted). Low-Velocity Precision Landing of Fixed-Wing UAV Using Stall and Nonlinear Model Predictive Control. In *IEEE Aerospace Conference*, volume 2, Big Sky, Montana.

Hoburg, W. and Tedrake, R. (2009). System Identification of Post Stall Aerodynamics for UAV Perching. *AIAA Infotech@Aerospace Conference*, pages 1–9.

Houska, B., Ferreau, H. J., and Diehl, M. (2011). ACADO toolkit An open-source framework for automatic control and dynamic optimization. (May 2010):298–312.

Jodeh, N. M. (2006). *Development of autonomous unmanned aerial vehicle research platform: Modeling, simulation, and flight testing*. PhD thesis, Air force institute of technology.

Johansen, T. A., Cristofaro, A., Sø rensen, K. L., Hansen, J. M., and Fossen, T. I. (2015). On estimation of wind velocity , angle-of-attack and sideslip angle of small UAVs using standard sensors. In *International Conference on Unmanned Aircraft Systems*, Denver.

Liepmann, H. W. (1952). *On the Application of Statistical Concepts to the Buffeting Problem*, volume 19. Douglas Aircraft.

Lussier Desbiens, a., Asbeck, a. T., and Cutkosky, M. R. (2011). Landing, perching and taking off from vertical surfaces. *The International Journal of Robotics Research*, 30(3):355–370.

Mathisen, S. H., Fossen, T. I., and Johansen, T. A. (2015). Non-linear Model Predictive Control for Guidance of a Fixed-Wing UAV in Precision Deep Stall Landing. In *International Conference on Unmanned Aircraft Systems*, Denver.

Mathisen, S. H., Gryte, K., Fossen, T. I., and Johansen, T. A. (Submitted). Non-linear Model Predictive Control for Longitudinal and Lateral Guidance of a Small Fixed-Wing UAV in Precision Deep Stall Landing. In *AIAA SciTech*.

MathWorks (2015a). Airframe Trim and Linearize. `http://se.mathworks.com/help/aeroblks/examples/airframe-trim-and-linearize-1.html#zmw57dd0e1168`. Accessed: 15.06.2015.

MathWorks (2015b). Dryden Wind Turbulence Model (Continuous). `http://se.mathworks.com/help/aeroblks/drydenwindturbulencemodelcontinuous.html?refresh=true`. Accessed: 16.06.2015.

MathWorks (2015c). NASA HL-20 Lifting Body Airframe. `http://se.mathworks.com/help/aeroblks/nasa-hl-20-lifting-body-airframe.html`. Accessed: 15.06.2015.

MathWorks (2015d). Wind Shear Model. `http://se.mathworks.com/help/aeroblks/windshearmodel.html`. Accessed: 10.06.2015.

Meshia, F. (2008). Model analysis with XFLR5. *Soaring Digest*, 25(2):27–51.

Minsaas, K. and Steen, S. (2012). Lecture notes TMR4220 Naval Hydrodynamics; Foil Theory.

Moore, J. L., Cory, R., and Tedrake, R. (2014). Robust post-stall perching with a simple fixed-wing glider using LQR-Trees. *Bioinspiration & biomimetics*, 9(2):025013.

NATO (1976). *STALL/SPIN PROBLEMS OF MILITARY AIRCRAFT*. Number June. Neuilly sur Seine, France.

Neihouse, A. I. and Lee, H. A. (1951). Investigation to Determine the Effectiveness of a Split-Aileron Type Emergency Spin-Recovery Device for the Northrop XF-89 Airplane. Technical report, National Advisory Committee for Aeronautics. Langley Aeronautical Lab, Langley Field, VA, United States.

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer, 2nd edition.

Ostowari, C. and Naik, D. (1985). Post-Stall Wind Tunnel Data for NACA 44XX Series Airfoil Sections. Technical Report 4807, Solar Energy Research Institute, Golden, Colorado.

Paw, Y. C. (2009). *Synthesis and Validation of Flight Control for UAV*. Phd thesis, University of Minnesota.

Pointner, W. and Kotsis, G. (2011). USING FORMAL METHODS TO VERIFY SAFE DEEP STALL LANDING OF A MAV. In *Digital Avionics Systems Conference*, pages 1–10.

Prantl, L. (1921). *Application of modern hydrodynamics to aeronautics*. U.S. Government Printing Office.

Prasad B., B. and Pradeep, S. (2007). Automatic Landing System Design using Feedback Linearization Method. In *AIAA Infotech@Aerospace 2007 Conference and Exhibit*, Infotech@Aerospace Conferences. American Institute of Aeronautics and Astronautics.

Rao, D. M. K. K. V. and Hiong, T. (2014). Automatic landing system design using sliding mode control. *Aerospace Science and Technology*, 32(1):180–187.

Rawlings, J. B. and Mayne, D. Q. (2009). *Model Predictive Control: Theory and Design*. Nob Hill Pub.

Roberts, J. W., Cory, R., and Tedrake, R. (2009). On the controllability of fixed-wing perching. *2009 American Control Conference*, pages 2018–2023.

Sivells, J. and Neely, R. (1947). Method for calculating wing characteristics by lifting-line theory using nonlinear section lift data. Technical Report April, Langley Memorial Aeronautical Laboratory.

Skulstad, R., Syversen, C. L., Merz, M., Sokolova, N., Fossen, T. I., and Johansen, T. A. (2015). Net Recovery of UAV with Single-Frequency RTK GPS. In *Proc. of the IEEE Aerospace Conference*, number 978, Big Sky, Montana.

Stengel, R. F. (2004). *Flight Dynamics*. Princeton University Press.

Stevens, B. L. and Lewis, F. L. (2003). *Aircraft control and simulation*. John Wiley \& Sons, Hoboken, New Jersey, 2 edition.

Taniguchi, H. (2008). Analysis of deepstall landing for uav. In *INTERNATIONAL CONGRESS OF THE AERONAUTICAL SCIENCES*, pages 1–6, Anchorage, Alaska.

Teimourian, A. and Firouzbakht, D. (2013). A Practical Method for Determination of the Moments of Inertia of Unmanned Aerial Vehicles. (September):9–12.

Tsourdos, A., White, B., and Shanmugavel, M. (2011). *Cooperative Path Planning of Unmanned Aerial Vehicles*. Wiley.

University of Porto; Underwater Systems and Technology Laboratory (2015a). DUNE web page. `http://lsts.fe.up.pt/toolchain/dune`. Accessed: 28.06.2015.

University of Porto; Underwater Systems and Technology Laboratory (2015b). Neptus web page. `http://lsts.fe.up.pt/neptus/`. Accessed: 29.06.2015.

U.S. Department of Defense (1980). U.S. Military Specification MIL-F-8785C. Technical report, Department of Defense Military Specifications and Standards, Philadelphia, Pennsylvania.

Wächter, A. and Biegler, L. T. (2006). *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*, volume 106.