



Norwegian University of  
Science and Technology

# Simulation, Control and Visualization of UAS

**Kristoffer Dønnestad**

Master of Science in Engineering Cybernetics

Submission date: June 2011

Supervisor: Thor Inge Fossen, ITK





## **MSC THESIS DESCRIPTION SHEET**

**Name:** Kristoffer Dønnestad  
**Department:** Engineering Cybernetics  
**Thesis Title (English):** Simulation, Control and Visualization of UAS  
**Thesis Title (Norwegian):** Simulering, regulering og visualisering av UAS

**Thesis Description:** The purpose of the thesis is to develop and implement a mathematical model of the Recce D6 unmanned aerial vehicle (UAV) system for simulation and hardware-in-the-loop testing. The model should include the aerodynamic forces and comprise feasible actuator dynamics.

The following items must be considered:

1. Derive and implement the relevant dynamics of the Recce D6 UAV system. Model parameters should be based on data from Odin Aero.
2. The model should be excited using standard aircraft maneuvers such as take-off, landing, steady flight, turning and climbing/diving. Hence, a simple motion control system must also be considered and implemented.
3. Create a visualization environment in Matlab to substantiate the UAV simulator model response.
4. Present your results in a report.

**Start date:** 2011-01-17  
**Due date:** 2011-06-20

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



# Abstract

The *Unmanned Vehicle Laboratory* at the Department of Engineering Cybernetics at the Norwegian University of Science and Technology has the overall goal of developing an unmanned aerial system (UAS). The unmanned aerial vehicle (UAV) to be considered is the Recce D6 delta-wing produced by Odin Aero AS.

A UAS is clearly quite a complex system, and the consequences of system failure during flight are obviously potentially vital. The need for a comprehensive test setup for extensive ground testing is clearly important, and a flight simulator of the UAV is a key component of such a test setup.

A mathematical model of Recce D6 is suggested from a first-principle approach and together with appurtenant actuators implemented in Matlab /Simulink. A simple motion control system with reference models and simple PID controllers are also discussed and implemented. Last, a Matlab-native graphical interface is developed to substantiate the UAV handling.

The simulation model, motion control system and visualization is put together in a nearly 7 minutes long demonstration video of a fully automated flight. The video illustrates a set of aircraft maneuvers and handling issues related to take off, touch and go landings, climbing, descent, banked turn, stall and landings under gusty wind and no wind.

The resulting video reveals a promising basis of a simulator environment for hardware-in-the-loop testing, but further tuning of the model parameters is required to acquire an accurate artificial representation of the aircraft.



# Preface

The hand-in of this thesis finalizes a master's degree in engineering cybernetics from Department of Engineering Cybernetics at the Norwegian University of Science and Technology in Trondheim, Norway. It has been five exacting, but highly profitable years.

With no background from aviation from before, work on this thesis has been a great challenge. The really time consuming parts has turned out to be the somewhat less expected ones. This is not necessarily reflected in the text, as there is really not that much to say when the final result is eventually finalized. Topics like the landing gear dynamics and the heading reference model and the graphical environment are examples of such unexpected, but major time consuming challenges. The graphical environment in particular holds the unfortunate property of being an endless task always subject to further tweaking and development.

Thanks goes out to everyone that have been working at the unmanned vehicle laboratory over the last year for contributing to a good working environment. I wish all of my fellow students the best for the future.

Thanks to my supervisor professor Thor Inge Fossen at the Department of Engineering Cybernetics for establishing the project and the laboratory, and for useful feedback. Also thanks to Helle Slupphaug for help with the English language.

The overall unmanned aerial system project at the university is very interesting. The establishment of a blog or website of some sort for the project progress is encouraged. With a hope that the contributions in this thesis is found useful for further usage, I wish the project and all contributors the best of luck for the future.

Kristoffer Dønnestad





## Abbreviations

AC	Aerodynamic Center
AoA	Angle of Attack
CG	Center of Gravity (Center of mass)
CO	Center of Origin
CP	Center of Pressure
DOF	Degree Of Freedom
EKF	Extended Kalman Filter
GNC	Guidance, Navigation and Control
GNSS	Global Navigation Satellite System
HIL	Hardware-in-the-loop
INS	Inertial Navigation System
MTOW	Maximum Take-Off Weight
NED	North East Down
SA	Stability Axes
NTNU	Norwegian University of Science and Technology
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle



# Contents

<b>Abstract</b>	<b>i</b>
<b>Preface</b>	<b>iii</b>
<b>Abbreviations</b>	<b>v</b>
<b>List of Figures</b>	<b>xi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Theory Basis and Advised Literature . . . . .	3
1.3 Thesis Outline and Contributions . . . . .	4
1.4 Recce D6 Overview . . . . .	4
1.5 Current Overall Project Status . . . . .	5
<b>2 Background</b>	<b>7</b>
2.1 Kinematics . . . . .	7
2.1.1 Vectors . . . . .	7
2.1.2 Coordinate Systems . . . . .	9
2.1.3 Rotations . . . . .	11
2.1.4 Angular Velocity Transformation . . . . .	13
2.2 Kinetics . . . . .	14
2.2.1 Forces and Moments on a Rigid Body . . . . .	14
2.2.2 Rigid Body Dynamics . . . . .	14
2.3 Aerodynamics . . . . .	15

<b>3</b>	<b>Recce D6 Simulation Model</b>	<b>19</b>
3.1	Equations of Motion . . . . .	20
3.1.1	Aerodynamics, $\tau_a$ . . . . .	24
3.1.2	Thrust, $\tau_t$ . . . . .	31
3.1.3	Landing gear, $\tau_{LG}$ . . . . .	32
3.1.4	Tuning the Aerodynamics . . . . .	34
3.2	Propulsion System . . . . .	37
3.3	Flap Dynamics . . . . .	39
<b>4</b>	<b>Recce D6 Control Design Model</b>	<b>41</b>
4.1	Surge Velocity Control . . . . .	44
4.2	Altitude Control . . . . .	44
4.3	Heading Control . . . . .	45
<b>5</b>	<b>Motion Control System, Flight Mode</b>	<b>49</b>
5.1	Guidance System . . . . .	50
5.1.1	Reference Models . . . . .	51
5.1.2	Heading Reference Model . . . . .	51
5.2	Control system . . . . .	53
<b>6</b>	<b>Implementation</b>	<b>55</b>
6.1	Graphic Interface . . . . .	55
6.1.1	3D Visualization . . . . .	55
6.1.2	Artificial Horizon . . . . .	56
6.1.3	Compass . . . . .	58
6.2	Crash Handling . . . . .	59
<b>7</b>	<b>Simulated Flight</b>	<b>61</b>
7.1	Case Study I: Landing Gear Dynamics . . . . .	63
7.2	Case Study II: Take-Off . . . . .	65
7.3	Case Study III: Climbing . . . . .	66
7.4	Case Study IV: Horizontal Steady Flight . . . . .	66
7.5	Case Study V: Dutch Roll . . . . .	67

7.6	Case Study VI: Banked Turn . . . . .	68
7.7	Case Study VII: Descent . . . . .	68
7.8	Case Study VIII: Wind . . . . .	68
7.9	Case Study IX: Touch-and-go, Crosswind . . . . .	70
7.10	Case Study X: Stall . . . . .	71
7.11	Case Study XI: Propulsion System . . . . .	72
7.12	Case Study XII: Touch-and-go, Upwind . . . . .	72
7.13	Case Study XIII: Landing at Low Airspeed . . . . .	73
<b>8</b>	<b>Conclusions</b>	<b>75</b>
8.1	Further Work . . . . .	75
	Bibliography . . . . .	77
<b>A</b>	<b>Case Study video</b>	<b>79</b>
<b>B</b>	<b>Matlab code</b>	<b>83</b>
B.1	RecceD6_init.m (Matlab script) . . . . .	83
B.2	MakeVideo.m (Matlab script) . . . . .	89
B.3	Plot3d.m (Matlab function) . . . . .	98
B.4	aHorizon.m (Matlab function) . . . . .	103
B.5	RecceCompass.m (Matlab function) . . . . .	104
<b>C</b>	<b>Digital attachment</b>	<b>107</b>
<b>D</b>	<b>Recce D6 documentation from Odin Aero</b>	<b>109</b>
<b>E</b>	<b>Servo motor specifications</b>	<b>111</b>



# List of Figures

1.1	An imagined UAS constellation . . . . .	2
1.2	Photo of RECCE D6 . . . . .	5
1.3	Flap deflection angle definitions . . . . .	6
2.1	Illustration of the relevant coordinate systems in question. . . . .	10
2.2	Airfoil definitions . . . . .	15
3.1	Illustration of inertial frame, body frame and center of gravity. . . . .	23
3.2	Simulator model structure . . . . .	24
3.3	The five aerodynamic centers of the simulator implementation. . . . .	26
3.4	The nonlinear lift coefficient . . . . .	30
3.5	The nonlinear pitching moment coefficient. . . . .	31
3.6	Li-pol battery voltage characteristics . . . . .	39
5.1	GNC structure . . . . .	50
5.2	Motion control principle . . . . .	53
6.1	Simulink snapshot of the motion control system . . . . .	56
6.2	An example plot of the Matlab 3D interface . . . . .	57
6.3	The color-map used in the graphical environment. . . . .	57
6.4	An example plot of the Matlab Artificial horizon plot . . . . .	58
6.5	An example plot of the Matlab compass . . . . .	59
7.1	An example snapshot of the simulated flight video window . . . . .	64
7.2	Average wind strength in simulated flight video. . . . .	69





# Chapter 1

## Introduction

The *Unmanned Vehicle Laboratory* at the Department of Engineering Cybernetics at the Norwegian University of Science and Technology (NTNU) has the overall goal of developing an unmanned aerial system (UAS). A UAS is a complete system consisting of all elements needed to support and utilize an Unmanned Aerial Vehicle (UAV). Such elements may be one or more UAV's, on board hardware and software, ground stations, data-links, launching and landing systems and so forth. Today, such systems have wide ranges of military applications including intelligence, surveillance, target acquisition, reconnaissance and target elimination. Civilian applications are currently not as many, but it is easy to imagine such systems assisting in border patrol, search and rescue missions, data acquisition for numerous research disciplines and sea surveillance. Figure 1.1 illustrates an imagined situation where a small fleet of unmanned air vehicles equipped with video camera and infrared cameras assists in a search for missing people after a hazardous storm in an avalanche exposed area.

### 1.1 Motivation

A UAS is clearly quite a complex system, built up by a lot of components created by numerous distinct developers. The consequences of system failure during flight are obviously potentially vital, especially if flying near urban territories. Thus, demands regarding safety, quality and robustness of controllers, data links, hardware, software and human-machine interaction has to be very strict, yet still fault tolerant. In addition to the technical challenges, a project like this faces significant restrictions from aviation authorities. It becomes clear that developing a comprehensive test setup for extensive ground testing is a fundamental requirement for any involved system. A flight simulator that comprises the dynamics of the UAV and its actuators immediately stands out as a key component of such a test setup.

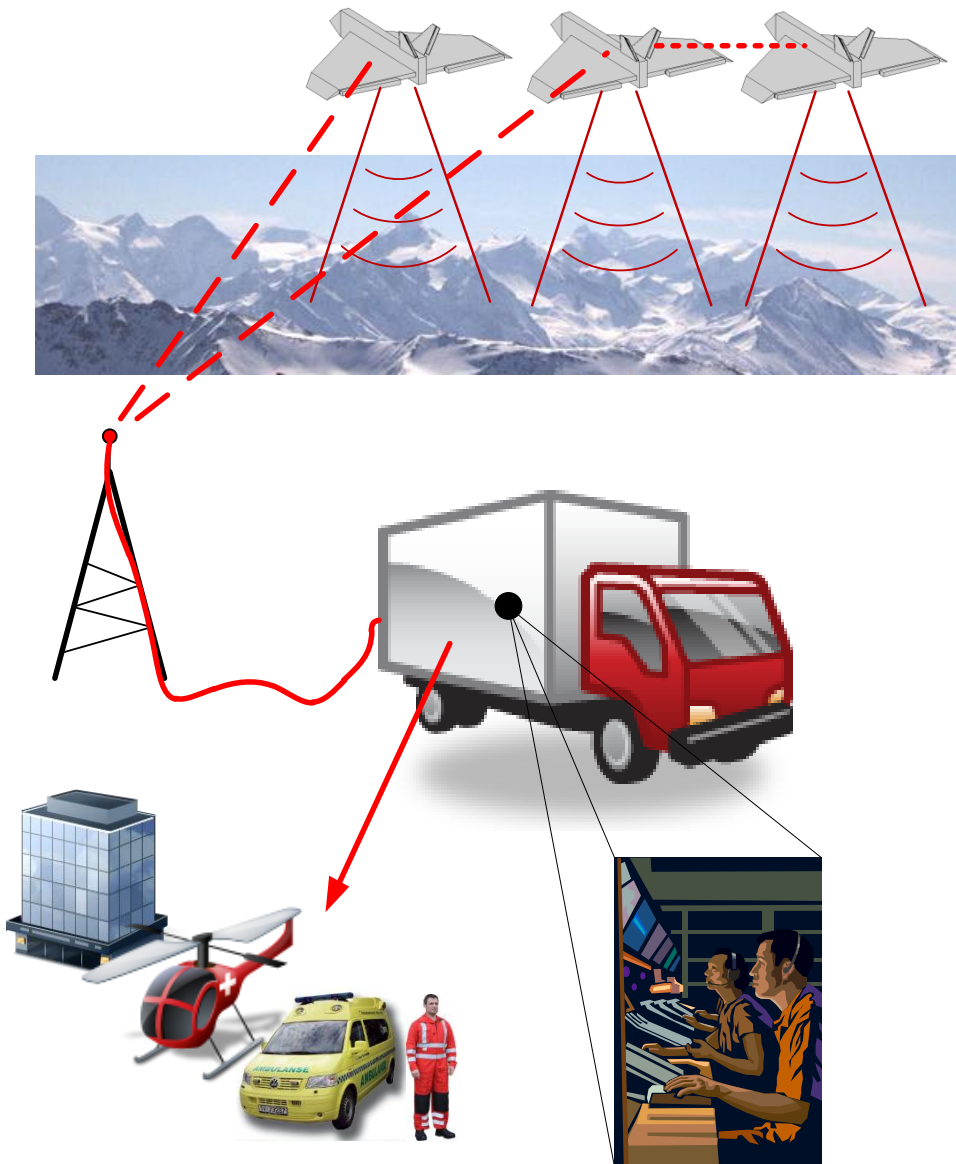


Figure 1.1: An imagined UAS constellation of a search for missing people in a hazardous environment.

Mathematical equations of motion that captures the dynamics of the vehicle in question is the very cornerstone to any flight simulation environment. (Allerton, 2009)

## 1.2 Theory Basis and Advised Literature

Creating an aerial simulator system requires some insight to a wide range of engineering disciplines including classical mechanics, fluid mechanics, programming, mathematics and control engineering. In addition, some insight to aviation terminology, flight visualization and instrumentation should be established.

Classical mechanics (physics) forms the foundation for mathematically describing the dynamic motion (kinetics) and location /orientation (kinematics) of any rigid body. Being one of the first branches within physics, the field has become very rich in results. Useful references in this topic is Egeland and Gravdahl (2003), Fossen (2011b) and Yuan (1988).

Being submerged into viscous air, the aircraft is subject to pressure-induced forces and moments as it moves through it. This is a big and very complicated subject from the field of aerodynamics, a subfield of fluid mechanics. Useful references addressing aerodynamics for aircraft are Katz and Plotkin (2001), and Houghton and Carpenter (2003). More general publications regarding fluid-induced lift and drag are Hoerner and Borst (1985) and Hoerner (1958) respectively. For fluid mechanics in general, White (2008) and Graebel (2007) provide a good supplement to the above mentioned texts in the sense that they summarize many complex subjects in an easily readable manner.

The derivation of automatic flight controllers (autopilots) involves the study of the dynamics of the system, in the agenda of feeding forward and/or feeding back states of a system to a controlled input in order to alter the system dynamics as desired. Clearly a subject with numerous different approaches, all of which have different properties. Basic control engineering literature is Balchen, Andersen and Foss (2003). Often, dynamic systems can be linearized about a certain equilibrium giving a linear time invariant (LTI) system. For such systems, Chen (1999) provides a wealth of results from linear algebra with applications to control engineering. Where linearization is not feasible or desired, Khalil (2002) represents the nonlinear counterpart. Finally Fossen (2011b) provides a summary of all disciplines desired in a motion control system, including state-of-the-art schemes for guidance systems, navigation systems and both linear and nonlinear control systems.

There also exists texts addressing flight simulation, flight control and flight mechanics directly. One should keep in mind that these books usually addresses conventional aircraft and fighters, and comprise much more advanced propulsion, actuation and instrumentation systems compared to a small UAV. Examples are Allerton (2009), McLean 1990), Pratt (2000) and Yechout et al. (2003).

Finally, less advanced books, meant for instance for pilot training in a non-engineering context, is very handy get into airplane, flight and aviation terminology. Such easily read books include Harnard and Philpott (2004) and Nordan Aviation Training Systems (2005)

In general, it has not been achieved to obtain any good texts addressing small delta-wings directly.

### 1.3 Thesis Outline and Contributions

Chapter 2 lists a set of fundamental results related to kinematics, dynamics and aerodynamics of rigid bodies, that is essential to the subsequent chapters and implementations.

Chapter 3 suggests a mathematical module-based simulation model of the UAV, derived from a first-principle approach using basic laws of dynamics and aerodynamics. The model includes actuation models (including power plant) and nonlinear aerodynamic coefficients.

Chapter 4 suggests a set of simplifications to the aerodynamic model from the simulator-model to obtain a model better suitable for control design.

Chapter 6 presents a set of Matlab functions developed to effectively visualize the UAV, and discusses some implementation aspects of the simulator model.

Chapter 7 discusses a six minute flight simulation video, produced in Matlab, using the simulator model derived in chapter 3 and a set of simple controllers. The video comprise a set of 13 distinct case-studies, all of which are meant to illustrate model properties, different aircraft maneuvers and stability considerations.

### 1.4 Recce D6 Overview

The aircraft to be considered is the Recce D6, produced by Odin Aero AS. This is the actual UAV that will be used by the Unmanned Vehicle Laboratory for the overall UAS project. The UAV has not been available for inspection or testing during the work on this particular thesis. A photo of the aircraft is provided courtesy of Odin Aero AS in figure 1.2. All the available documentation of the air vehicle is listed in appendix D.

The main purposes of the aircraft body parts, shown in figure 1.2, can be summarized to

- Wing: Produce lift
- Elevon: (Elevator and Aileron merged) Control roll and pitch
- Elevon servo: Provides the elevon deflection

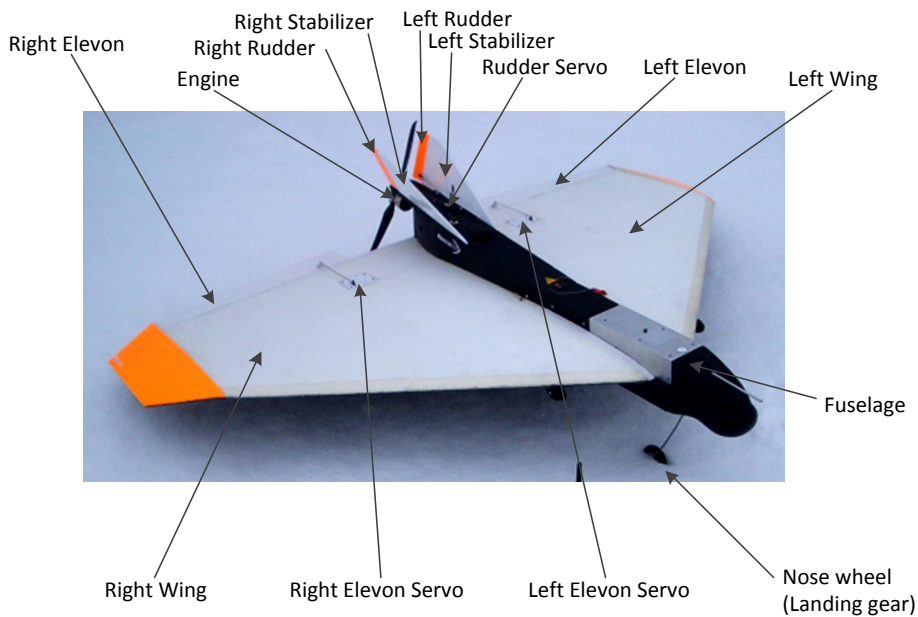


Figure 1.2: Photo of RECCE D6. (Photo: Carl Eric Stephansen, Odin Aero AS)

- Stabilizer and rudder: Control yaw
- Rudder servo: Provide the rudder's deflections. This single servo control both the rudders at the left and right stabilizer.
- Engine: Produce thrust
- Fuselage: Hold things together, carry payloads
- Landing gear: Provide smooth landing and take-off characteristics

The four actuator deflection angles ( $\delta_{el}$ ,  $\delta_{er}$ ,  $\delta_{rl}$ ,  $\delta_{rr}$ ) are defined in figure 1.3.

## 1.5 Current Overall Project Status

The overall project was started autumn 2010, with six students carrying out project work for the course TTK4550 at NTNU. The projects includes

- Inertial Navigation Systems (INS), Observer design and Extended Kalman Filter (EKF)
- Framework for Operating System and Peripheral Interfacing related to UAS

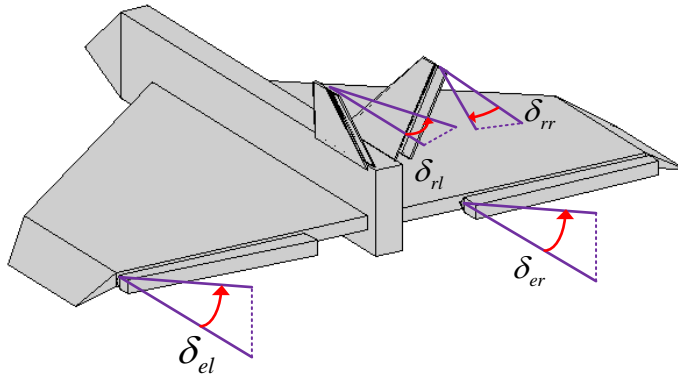


Figure 1.3: Flap deflection angle definitions

- Flight simulator framework, and aircraft modeling

In parallel with work on this thesis, three other students are contributing to the overall UAS project through their master's-thesis at the unmanned vehicle laboratory. The projects includes

- Modeling of Global Navigational Satellite Systems (GNSS) for Hardware-in-the loop (HIL ) testing.
- Development and evaluation of Inertial Navigation Systems schemes.
- Hardware and software integration for the UAV.

No outline of timetable for future work or actual flights are known.

# Chapter 2

## Background

Essential to the study of dynamics are rigid body kinetics and kinematics. For aircraft, whose motion heavily depend on fluid-induced forces and moments from the surrounding air, aerodynamics is also vital. This chapter states the most fundamental aspects of rigid body kinetics, kinematics and aerodynamics used throughout the thesis.

### 2.1 Kinematics

Kinematics is the study of the geometrical aspects of motion, without considering the forces and moments that actually caused the motion. Notations and results are mainly taken from Egeland and Gravdahl (2003), Fossen (2011b) and SNAME (1950).

#### 2.1.1 Vectors

Vectors are heavily used to describe entities such as forces, velocities, positions, torques and so forth. A vector is defined as to have a direction and a magnitude. It will mainly be described in terms of a *coordinate-free vector*, or as components of a predefined Cartesian coordinate system (a *coordinate vector*). The following vector notations are used:

$$\vec{u} = u_1 \vec{a}_1 + u_2 \vec{a}_2 + \dots + u_n \vec{a}_n$$

A coordinate free vector of order  $n$ , described in terms of a linear combination of the orthogonal unit vectors  $\vec{a}_i$ ,  $i \in \{1, 2, \dots, n\}$

$$\mathbf{u}^c = [u_1 \quad u_2 \quad \dots \quad u_n]^T$$

A coordinate vector of order  $n$ , described in terms of components of the Cartesian coordinate system  $c = \{\vec{a}_1, \vec{a}_2, \dots, \vec{a}_n\}$ . The superscript may be omitted if it is evident from the context which coordinate system is referred, or if it is evident that a specific coordinate system reference is superfluous.

$$|\vec{u}|$$

The magnitude of  $\vec{u}$ .  $\vec{u}$  is said to be a *unit vector* if  $|\vec{u}| = 1$

$$\mathbf{u}^\times \text{ or } S(\mathbf{u})$$

The skew-symmetric form of the 3-dimensional coordinate vector  $\mathbf{u}$ .  
 $\mathbf{u}^\times \in \mathbb{R}^{3 \times 3}$

$${}^i \vec{x}$$

A coordinate free unit vector specifying the direction of the x-axis of the coordinate system  $\{{}^i \vec{x}, {}^i \vec{y}, {}^i \vec{z}\} \in i$ .

Vector representations of forces, moments and velocities used are defined by the following examples:

$\vec{v}_{b/n}$	$\in \mathbb{R}^3$	$[m/s]$	Linear velocity of BODY with respect to NED
$\vec{\omega}_{b/n}$	$\in \mathbb{R}^3$	$[rad/s]$	Angular velocity of BODY with respect to NED
$\vec{r}_{b/n}$	$\in \mathbb{R}^3$	$[m]$	Vector defining the position of $o_b$ with respect to $o_n$
$\vec{q}_{b/n}$	$\in \mathbb{R}^4$	$[rad]$	A unit quaternion defining the orientation of $\{b\}$ with respect to $\{n\}$
$\vec{f}_b$	$\in \mathbb{R}^3$	$[N]$	Force with line of action through $o_b$
$\vec{m}_b$	$\in \mathbb{R}^3$	$[Nm]$	Moment about $o_b$

In the following, only vectors in the three-dimensional Euclidean space is considered. That is; vectors of order 3;  $\mathbf{u} \in \mathbb{R}^3$ . The scalar product of two vectors are defined as:

$$\vec{u} \cdot \vec{v} = |\vec{u}| |\vec{v}| \cos \theta = \mathbf{u}^T \mathbf{v} \quad (2.1)$$

where  $\theta$  is the angle between the two vectors. The cross product of two vectors are defined as:

$$\vec{u} \times \vec{v} = \begin{vmatrix} \vec{a}_1 & \vec{a}_2 & \vec{a}_3 \\ u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{vmatrix} = \begin{bmatrix} u_2 v_3 - u_3 v_2 \\ u_3 v_1 - u_1 v_3 \\ u_1 v_2 - u_2 v_1 \end{bmatrix} \quad (2.2)$$

The skew-symmetric form of a coordinate vector  $\mathbf{u}$  is defined as:

$$\mathbf{u}^\times = \mathbf{S}(\mathbf{u}) = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix} \quad (2.3)$$



The cross product may then be written:

$$\mathbf{u} \times \mathbf{v} = \mathbf{u} \times \mathbf{v} = \mathbf{S}(\mathbf{u})\mathbf{v} \quad (2.4)$$

### 2.1.2 Coordinate Systems

Defining reference coordinate systems:

NED	{n}	North East Down coordinate system A coordinate system whose origin is fixed at a certain point on the Earth's surface, and whose $\vec{x}$ -axis points northwards, the $\vec{y}$ -axis points eastwards and the $\vec{z}$ -axis points downwards. Thus, the $\vec{x}$ and the $\vec{y}$ axes are parallel to a tangent plane at the Earth's surface at this point. (This coordinate system may also be referred to as Flat Earth (FE) coordinate system.) This coordinate frame is chosen to be the inertial reference frame throughout this text.
BODY	{b}	Body coordinate system A coordinate system whose origin is fixed in the vehicle, and whose $\vec{x}$ -axis points in the longitudinal direction (positive in the natural forward direction), the $\vec{y}_b$ -axis points in the lateral direction (positive in the natural right-direction), and the $\vec{z}$ -axis is a normal axis to the $\vec{x}\vec{y}$ plane, chosen to have positive direction to complete the right hand rule (natural downwards.)
SA	{s}	Stability coordinate system An intermediate coordinate system rotated an angle $\alpha$ (Angle of Attack, $\alpha$ - AoA) about ${}^b\vec{y}$ .
WIND	{w}	Wind coordinate system A coordinate system whose $\vec{x}$ -axis points in the direction of the vehicle velocity vector relative to the air. It is rotated an angle $-\beta$ ( $\beta$ is sideslip angle) about ${}^s\vec{z}$ .

The coordinate systems are illustrated in figure 2.1.

Defining the characteristic points on an aircraft body:

CG	Center of mass (Also called Center of gravity) The point where the gravitational force acts on the body.
CO	Center of origin. The point where the BODY coordinates has its origin.
CP	Center of pressure. The point where the the pressure induced forces acts on the body.
AC	Aerodynamic center The point where the aerodynamic forces and moments are given about.

Defining positions, velocities and moments along the axis of BODY, NED and WIND:

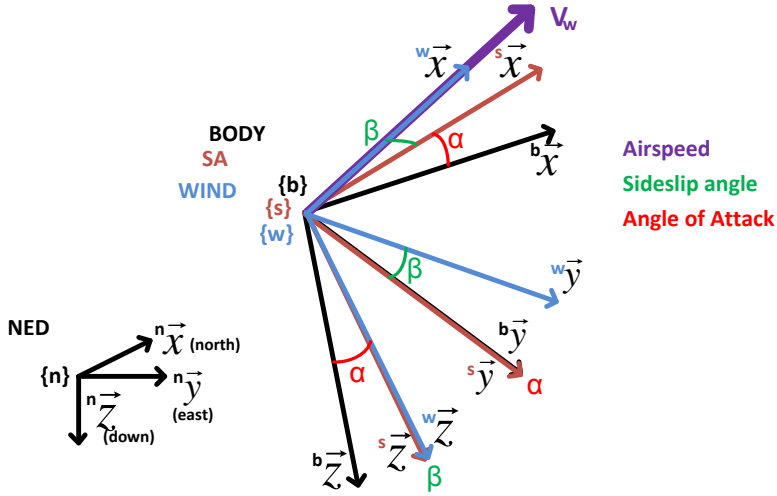


Figure 2.1: Illustration of the relevant coordinate systems in question. It is evident that the airstream velocity vector  $\mathbf{v}_{b/w}^w = [V_w \ 0 \ 0]$  relative to body easily can be defined using  $V_w, \alpha, \beta$ .

Axis	Linear			Angular		
	Position [m]	Velocity [m/s]	Force [N]	Position [rad]	Velocity [rad/s]	Moment [Nm]
$\vec{b}_x$	$x$	$u$	$X$		$p$	$K$
$\vec{b}_y$	$y$	$v$	$Y$		$q$	$M$
$\vec{b}_z$	$z, h$	$w$	$Z$		$r$	$N$
$\vec{n}_x$	$N$	${}^n u$		$\phi$		
$\vec{n}_y$	$E$	${}^n v$		$\theta$		
$\vec{n}_z$	$D$	${}^n w$	$W$	$\psi$		
$\vec{w}_x$			$L$	$\beta$		
$\vec{w}_y$			$D$	$\alpha$		
$\vec{w}_z$						$M_p$

Table 2.1: Nomenclature of positions, orientations, velocities, forces and moments along and about some coordinate frame axis

Note that  $N$  and  $D$  are used two times to denote two different entities respectively. Although it might seem confusing from this table, the context in which it is referred to, usually gains no confusion in which it is used. To keep consistency with other literature it is kept like this. The following nomenclature is often used

$X$	Axial force	$x$	Surge
$Y$	Side force	$y$	Sway
$Z$	Normal force	$z$	Heave
$K$	Rolling moment	$\phi$	Roll
$M$	Pitching moment	$\theta$	Pitch
$N$	Yawing moment	$\psi$	Yaw
$N$	North	$L$	Aerodynamic lift
$E$	East	$D$	Aerodynamic drag
$D$	Down	$M_p$	Aerodynamic pitching moment
$\alpha$	Angle of attack	$W$	Weight
$\beta$	Sideslip angle	$h$	Altitude ( $h = -z$ )

### 2.1.3 Rotations

The orientation of one coordinate system with respect to an other are usually defined in terms of Euler angles, angle axis parametrization, Euler parameters (unit quaternions) or by a rotation matrix. The material is taken from Egeland and Gravdahl (2003).

Euler angles are the most commonly used terminology to describe rotations, possibly due to its intuitive nature. Rotations in terms of Euler angles are three simple rotations about the three principal axes of a coordinate system:

- Roll ( $\phi$ ) - rotation about the  $\vec{x}$ -axis
- Pitch ( $\theta$ ) - rotation about the  $\vec{y}$ -axis
- Yaw ( $\psi$ ) - rotation about the  $\vec{z}$ -axis

Define

$$\Theta := [ \phi \quad \theta \quad \psi ]^T \quad (2.5)$$

The rotation matrix  $\mathbf{R}_n^b \in \mathbb{R}^{3 \times 3}$ , changing the coordinates of  $\mathbf{u}^b$  to  $\mathbf{u}^n$  in terms of Euler angles,  $\mathbf{R}_b^n(\phi, \theta, \psi) = \mathbf{R}_b^n$ , can be written as the product of the above mentioned simple rotations;  $\mathbf{R}_{x,\phi}$ ,  $\mathbf{R}_{y,\theta}$  and  $\mathbf{R}_{z,\psi}$ :

$$\mathbf{R}_b^n = \mathbf{R}_{z,\psi} \cdot \mathbf{R}_{y,\theta} \cdot \mathbf{R}_{x,\phi} \quad (2.6)$$

$$= \underbrace{\begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{R}_{z,\psi}} \cdot \underbrace{\begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}}_{\mathbf{R}_{y,\theta}} \cdot \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}}_{\mathbf{R}_{x,\phi}} \quad (2.7)$$

$$= \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi s\theta c\phi \\ s\psi c\theta & c\psi c\phi + s\psi s\theta s\psi & -c\psi s\phi + s\psi s\theta c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \quad (2.8)$$

This approach is attractive, but further investigation on the resulting rotation matrix reveals loss of rank (singularities) for some Euler-angle combinations. Also, for computer realizations, the computation of a rotation is clearly quite CPU demanding due to the trigonometric terms.

Any rotation in three dimensional space may also be described in terms of a vector  $\vec{k} \in \mathbb{R}^3$  and a simple rotation angle  $\theta$  about this vector. This introduces a four parameter scheme called *angle-axis parametrization* to define rotations in three dimensional space.

The rotation matrix  $\mathbf{R}_a^b = \mathbf{R}_{k,\theta} \in \mathbb{R}^{3 \times 3}$  changing the coordinates of  $\mathbf{u}^a$  to  $\mathbf{u}^b$  in terms of the angle axis parameters is found to be:

$$\mathbf{R}_{k,\theta} = \mathbf{I} + \mathbf{k}^\times \sin \theta + \mathbf{k}^\times \mathbf{k}^\times (1 - \cos \theta) \quad (2.9)$$

This approach is also somewhat intuitive, and attractive in the sense that it does not encounter any singularities. The downside, however, is that this approach is still quite CPU demanding.

The concept of the angle axis parametrization can be further extended to *Euler parameters*; defined in terms of  $\eta \in \mathbb{R}$  and  $\epsilon \in \mathbb{R}^3$  such that

$$\eta = \cos \frac{\theta}{2} \quad (2.10)$$

$$\epsilon = \mathbf{k} \sin \frac{\theta}{2} \quad (2.11)$$

The rotation matrix  $\mathbf{R}_{k,\theta} = \mathbf{R}_e(\eta, \epsilon) \in \mathbb{R}^{3 \times 3}$  changing the coordinates of  $\mathbf{u}^a$  to  $\mathbf{u}^b$  in terms of the Euler parameters is found to be:

$$\mathbf{R}_e(\eta, \epsilon) = \mathbf{I} + 2\eta\epsilon^\times + 2\epsilon^\times\epsilon^\times \quad (2.12)$$

$$= \begin{bmatrix} \eta^2 + \epsilon_1^2 - \epsilon_2^2 - \epsilon_3^2 & 2(\epsilon_1\epsilon_2 - \eta\epsilon_3) & 2(\epsilon_1\epsilon_3 + \eta\epsilon_2) \\ 2(\epsilon_1\epsilon_2 + \eta\epsilon_3) & \eta^2 - \epsilon_1^2 + \epsilon_2^2 - \epsilon_3^2 & 2(\epsilon_2\epsilon_3 - \eta\epsilon_1) \\ 2(\epsilon_1\epsilon_3 - \eta\epsilon_2) & 2(\epsilon_2\epsilon_3 + \eta\epsilon_1) & \eta^2 - \epsilon_1^2 - \epsilon_2^2 + \epsilon_3^2 \end{bmatrix} \quad (2.13)$$

This approach is attractive in the sense that it never encounter singularities, and no trigonometric terms in terms of the Euler parameters, making it quite CPU friendly. Further, defining the vector

$$\mathbf{q} = [ \eta \quad \epsilon^T ]^T \in \mathbb{R}^4 \quad (2.14)$$

and notes that

$$|\mathbf{q}| = 1 \quad (2.15)$$

introduces the ability to treat the vector of Euler parameters (2.14) as a *unit quaternion vector*. Conveniently, it is now possible to apply results from the theory of quaternions, and in particular the theory of unit quaternions.

The quaternion product is denoted  $\otimes$ , and the product of two unit quaternions is defined as

$$\mathbf{q} := \mathbf{q}_1 \otimes \mathbf{q}_2 = \begin{bmatrix} \eta_1 \eta_2 - \epsilon_1^T \epsilon_2 \\ \eta_1 \epsilon_2 + \eta_2 \epsilon_1 + \epsilon_1^\times \epsilon_2 \end{bmatrix} \quad (2.16)$$

Rotating the vector  $\mathbf{u}$  can now be carried out by calculating the unit quaternion product

$$\begin{bmatrix} 0 \\ \mathbf{R}_e(\eta, \epsilon) \mathbf{u} \end{bmatrix} = \begin{bmatrix} \eta \\ \epsilon \end{bmatrix} \otimes \begin{bmatrix} 0 \\ \mathbf{u} \end{bmatrix} \otimes \underbrace{\begin{bmatrix} \eta \\ -\epsilon \end{bmatrix}}_{\bar{\mathbf{q}}} \quad (2.17)$$

where  $\bar{\mathbf{q}}$  denotes the *quaternion conjugate*, or *quaternion inverse*.

Consider two frames whose orientation is defined in terms of  $\Theta_1$  and  $\Theta_2$ . The orientation error

$$\tilde{\Theta} = \Theta_1 - \Theta_2 \quad (2.18)$$

may also be represented in terms of the rotation matrices as

$$\tilde{\mathbf{R}} = (\mathbf{R}_1)^T \mathbf{R}_2 \quad (2.19)$$

or in terms of quaternion error as

$$\tilde{\mathbf{q}} = \bar{\mathbf{q}}_1 \otimes \mathbf{q}_2 \quad (2.20)$$

### 2.1.4 Angular Velocity Transformation

Integrating (2.34) to obtain the body's orientation does not make any physical sense. To relate the rotation velocities of the body ( $\omega_{b/n}^b$ ) to a rate of change in orientation ( $\dot{\mathbf{q}}_{b/n}$ ) an angular velocity transformation is applied: (Fossen, 2011b; Yuan, 1988)

$$\dot{\mathbf{q}}_{b/n} = \mathbf{T}(\mathbf{q}_{b/n}) \omega_{b/n}^b \quad (2.21)$$

$$= \mathbf{U}(\omega_{b/n}^b) \mathbf{q}_{b/n} \quad (2.22)$$

where

$$\mathbf{T}(\mathbf{q}) = \frac{1}{2} \begin{bmatrix} -\epsilon_1 & -\epsilon_2 & -\epsilon_3 \\ \eta & -\epsilon_3 & \epsilon_1 \\ \epsilon_3 & \eta & -\epsilon_1 \\ -\epsilon_2 & \epsilon_1 & \eta \end{bmatrix} \quad 2.23 \quad (2.23)$$

$$\mathbf{U}(\omega) = \frac{1}{2} \begin{bmatrix} 0 & -\omega^T \\ \omega & -\omega^\times \end{bmatrix} \quad (2.24)$$

The same transform can also be carried out using Euler angles (Vik, 2000)

$$\dot{\Theta}_{b/n} = \mathbf{T}(\Theta_{b/n}) \omega_{b/n}^b \quad (2.25)$$

where

$$\mathbf{T}(\Theta_{b/n}) = \frac{1}{\cos \theta} \begin{bmatrix} \cos \theta & \sin \phi \sin \theta & \cos \psi \sin \theta \\ 0 & \cos \phi \cos \theta & -\sin \phi \cos \theta \\ 0 & \sin \phi & \cos \phi \end{bmatrix} \quad (2.26)$$

for  $\theta \neq (2n - 1)\pi$  and  $n$  is some integer.

## 2.2 Kinetics

Kinetics is the study of the resulting motion caused by forces and moments. Notations and results are mainly taken from Fossen (2011b) and Egeland and Gravdahl (2003).

### 2.2.1 Forces and Moments on a Rigid Body

Consider a force  $\vec{f}_a$  with line of action through the point  $a$ , and a moment  $\vec{m}_a$  about the point  $a$ . Then about some other point  $b$  (Egeland and Gravdahl 2003)

$$\vec{f}_b = \vec{f}_a \quad (2.27)$$

$$\vec{m}_b = \vec{m}_a + \vec{r}_{b/a} \times \vec{f}_a \quad (2.28)$$

### 2.2.2 Rigid Body Dynamics

The Newton-Euler rigid body equations of motions are given with respect to an inertial reference frame (here: the NED frame) (Egeland and Gravdahl, 2003; Fossen, 2011b):

$$\vec{f}_g = \frac{{}^n d}{dt} \vec{p}_g = \frac{{}^n d}{dt} m \vec{v}_{b/n} \quad (2.29)$$

$$\vec{m}_g = \frac{{}^n d}{dt} \vec{h}_g = \frac{{}^n d}{dt} \mathbf{I}_g \vec{\omega}_{b/n} \quad (2.30)$$

where the inertia matrix  $\mathbf{I}_g$  (not to be confused with the identity matrix  $\mathbf{I}$ ) is defined as

$$\mathbf{I}_g = \int_{r^b} [(\mathbf{r}^b)^2 \mathbf{I} - \mathbf{r}^b (\mathbf{r}^b)^T] dm \quad (2.31)$$

$$= \int_{r^b} \begin{bmatrix} y^2 & -xy & -xz \\ -xy & x^2 + z^2 & -yz \\ -xz & -yz & x^2 + y^2 \end{bmatrix} dm \quad (2.32)$$

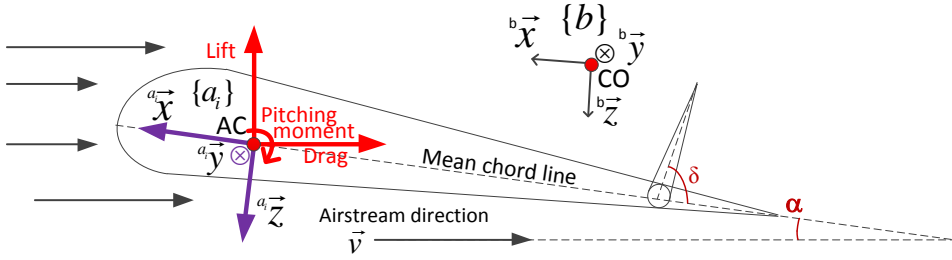


Figure 2.2: Airfoil definitions

Here,  $rb$  denotes the rigid body in question and  $dm = \rho dV$  denotes an infinitesimal mass.

Time differentiation of a vector  $\vec{a}$  in a moving coordinate frame  $\{b\}$  with respect to  $\{n\}$  satisfies (Egeland and Gravdahl, 2003)

$$\frac{{}^n d}{dt} \vec{a} = \frac{{}^b d}{dt} \vec{a} + \vec{\omega}_{b/n} \times \vec{a} \quad (2.33)$$

Inserting (2.33) into the equations (2.29) and (2.30), the rigid body dynamics can conveniently be written (Fossen, 2011b)

$$\underbrace{\begin{bmatrix} m\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_g \end{bmatrix}}_{\mathbf{M}_{RB}} \underbrace{\begin{bmatrix} \dot{\mathbf{v}}_{c/n}^b \\ \dot{\boldsymbol{\omega}}_{n/b}^b \end{bmatrix}}_{\dot{\boldsymbol{\nu}}} + \underbrace{\begin{bmatrix} m(\boldsymbol{\omega}_{n/b}^b)^\times & \mathbf{0} \\ \mathbf{0} & (\boldsymbol{\omega}_{n/b}^b)^\times \mathbf{I}_g \end{bmatrix}}_{\mathbf{C}_{RB}(\boldsymbol{\omega})} \underbrace{\begin{bmatrix} \mathbf{v}_{c/n}^b \\ \boldsymbol{\omega}_{n/b}^b \end{bmatrix}}_{\boldsymbol{\nu}} = \underbrace{\begin{bmatrix} \mathbf{f}_c^b \\ \mathbf{m}_c^b \end{bmatrix}}_{\boldsymbol{\tau}} \quad (2.34)$$

where the following vectors are defined for convenience:

$$\begin{aligned} \boldsymbol{\nu} &= [(\mathbf{v}_{b/n}^b)^T \quad (\boldsymbol{\omega}_{b/n}^b)^T]^T = [u \quad v \quad w \quad p \quad q \quad r]^T \\ \boldsymbol{\eta} &= [(\mathbf{r}_{b/n}^n)^T \quad (\mathbf{q}_{b/n}^n)^T]^T = [x \quad y \quad z \quad \eta \quad \epsilon_1 \quad \epsilon_2 \quad \epsilon_3]^T \\ \boldsymbol{\tau} &= [(\mathbf{f}_b^b)^T \quad (\mathbf{m}_b^b)^T]^T = [X \quad Y \quad Z \quad K \quad M \quad N]^T \end{aligned}$$

## 2.3 Aerodynamics

The material in this chapter is mainly taken from Nordan Aviation Training Systems (2005), Hughton and Carpenter (2003) and Barnard and Philpott (2004) The airfoil in figure 2.2 is a typical cross section of an asymmetric wing along its mean chord line. Note that the orientation of the  $\{a_i\}$  and the  $\{b\}$ -frame does not necessarily coincide. The origin of  $\{a_i\}$  is referred to the Aerodynamic Center (AC).

The two aerodynamic forces lift and drag have lines of action through the airfoil's center of pressure. This point moves at different flight conditions, so it is more convenient to represent the forces in the aerodynamic center. To include the effect

of CP not necessarily coinciding with AC, an aerodynamic pitching moment is introduced to act about AC. The aerodynamic center is a fixed point along the chord where the pitching moment coefficient vary little despite of changes in the lift coefficient. It is usually found a 25% chord length from the leading edge of the airfoil. (Hoerner and Borst, 1985)

Defining the aerodynamic forces and moments about AC (Houghton and Carpenter 2003)

$D$	$[N]$	$\mathbb{R}^1$	Drag force parallel to airstream ${}^w\vec{x}$
$L$	$[N]$	$\mathbb{R}^1$	Lift force perpendicular to airstream; parallel to ${}^w\vec{y}$
$M_p$	$[Nm]$	$\mathbb{R}^1$	Pitching moment about ${}^w\vec{z}$

which is usually defined as

$$L := \frac{1}{2}\rho V^2 A_L C_L \quad (2.35)$$

$$D := \frac{1}{2}\rho V^2 A_D C_D \quad (2.36)$$

$$M_p := \frac{1}{2}\rho V^2 \bar{c} A_{M_p} C_{M_p} \quad (2.37)$$

where

$C_i$	$[\cdot]$	$\mathbb{R}^1$	Aerodynamic coefficient.
$A_i$	$[m^2]$	$\mathbb{R}^1$	Reference area
$\rho$	$[kg/m^3]$	$\mathbb{R}^1$	Density of air
$V$	$[m/s]$	$\mathbb{R}^1$	Airspeed
$\bar{c}$	$[m]$	$\mathbb{R}^1$	Mean chord length

$i \in \{L, D, M_p\}$ . The aerodynamic coefficients are usually obtained by trial flights or performing empiric wind tunnel tests for a specific aircraft or airfoil design about the AC.

### Lift

Lift is defined as the resulting force perpendicular to the airstream direction. As the wing moves relative to the surrounding air, the airspeed along the different parts of the wing is altered. Consequently, the pressure distribution is also altered.<sup>1</sup> Any pressure induced by fluid motion on a body generates a perpendicular force on the surface. By designing a wing such that the relative airspeed on the upper side of the wing is higher than the lower side, the difference in forces due to the pressure distribution generates lift. This is the key phenomena that makes it possible to lift a body with higher density than air.

### Drag

Drag is defined as the resulting force parallel to, but with opposite direction than,

---

<sup>1</sup>The balance between fluid velocity, dynamic pressure and elevation in incompressible and frictionless fluids along a streamline is described by Bernoulli's equation, which is a widely known equation in fluid dynamics. In-depth description of the theorem can for instance be found in White (2008, pp. 183-192).



the relative airstream direction. Drag can be considered to consist of the sum of *parasite drag*, and *induced drag*. The former is the drag component generated by the actual motion of a body in a viscous fluid. It can be further broken down into *form drag*, *skin friction drag* and *interference drag*. The latter is a by-product of the creation of lift. The following short description of the different drag components are taken from Nordan Aviation Training Systems, London metropolitan university (2005):

- **Form drag** (pressure drag) is the result of changed pressure distribution, and thus altering the forces acting on the wing, due to viscous boundary layer flow separation. As the name suggests, this is due to the shape of the immersed body.
- **Skin friction drag** is the drag component induced by fluid displacement along a surface because of its roughness.
- **Interference drag** is generated in the junction between different bodies, for instance in the junction between the fuselage and a wing. The combined drag of two joined bodies turns out to be somewhat higher than the sum of the independent drag of the bodies. An example: To generate lift, the air velocity at the upper side of the wing needs to be higher than the general air velocity of the aircraft. This difference in airspeed results in vortices in the transition between body parts. The continuous acceleration of the fluid particles in the vortices results in a drag force acting on the body.
- **Induced drag** is a bi-product of the generation of lift. Lift is the result of difference in pressure over and under a wing. Considering the tip of a wing in flight; The difference in pressure makes the higher-density air under the wing expand into the lower-density air around the wingtip. This results in an airstream forming a twisting vortex core behind each wing tip. This airstream generates a complicated flow pattern that results in a so-called downwash airstream angle. The lift generated is actually perpendicular to this airstream direction, rather than the relative wind direction. With the lift defined as perpendicular to the relative airstream, the components of the induced lift vector from the downwash airstream needs to be decomposed into a lift component relative to the relative airstream, and the resulting component is considered as induced drag.



## Chapter 3

# Recce D6 Simulation Model

The choice of a mathematical model highly depends on the purpose of it. Creating a model can in it self be a great way of gaining understanding of a system. Furthermore, the models may be used to simulate systems to foresee potential problems, and in the derivation of controllers. One might think that the “best” mathematical model is the mostly advanced - the one that propagates the states “perfectly” according to reality for a global workspace. In reality, a compromise between demand of accuracy for the target application and resources available for deriving and solving the model has to be made.

Many models of conventional aircraft are available in a variety of textbooks. For delta wings, like the Recce D6, out-of-the-box models are considerably harder to dig up. This chapter addresses the derivation of a model used for simulating purposes from rigid body kinetics and basic aerodynamics in 6 DOF. The resulting model includes actuator dynamics (including motor, battery and servo motors) and landing gear dynamics. This is mainly done so that as many unforeseen problems as possible can arise during ground testing on the simulator, but also in order to be able to simulate very specific error-scenarios. The model is build up by defining five bodyparts and one motor, and parametrize the aerodynamic properties of the bodyparts independently. This modular approach facilitates the ability of rapidly changing existing, or putting together new aircraft designs by reusing existing bodyparts.

For model-based controller-derivation within a restricted flight-envelope (that is, the set of nominal flight conditions), a lot of simplifications should be made. This is done in the next chapter.

### 3.1 Equations of Motion

The NED frame is chosen as the inertial frame. The total mathematical model used to simulate Recce D6 about its CG is

$$\dot{\boldsymbol{\eta}} = \begin{bmatrix} \mathbf{R}_b^n & \mathbf{0} \\ \mathbf{0} & \mathbf{T}(\mathbf{q}) \end{bmatrix} \boldsymbol{\nu} \quad (3.1)$$

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\omega})\boldsymbol{\nu} = \boldsymbol{\tau}_g + \boldsymbol{\tau}_a + \boldsymbol{\tau}_t + \boldsymbol{\tau}_{lg} \quad (3.2)$$

where the *system inertia matrix*  $\mathbf{M}$ , and the *Coriolis-Centripetal matrix* might include aerodynamic mass and inertia (also called aerodynamic added mass) in addition to the rigid body mass and inertia (Fossen, 2011a). The added mass term represents an “additional mass” due to fluid displacement and will be treated as a constant in this thesis. For further reading about this topic consult for instance Katz and Plotkin (2001), Graebel (2007), White (2008) and Fossen (2011b).

A short explanation on the forces applied to the equations of motion:

- $\boldsymbol{\tau}_g$  Force and moment due to gravity
- $\boldsymbol{\tau}_a$  Forces and moments due to aerodynamic effects
- $\boldsymbol{\tau}_t$  Force and moment due to thrust
- $\boldsymbol{\tau}_{lg}$  Forces and moments due to wheel displacement

Aircraft is usually designed to be symmetrical about the  ${}^b\vec{x}{}^b\vec{z}$ -plane (McLean, 1990), which implies  $\mathbf{I}_{xy} = \mathbf{I}_{yz} = 0$  giving the rigid body inertia matrix about a point on the  ${}^b\vec{x}$ -axis:

$$\mathbf{I}_{RB} = \begin{bmatrix} I_{xx} & 0 & -I_{xz} \\ 0 & I_{yy} & 0 \\ -I_{xz} & 0 & I_{zz} \end{bmatrix} \quad (3.3)$$

Note that about the center of mass, the rigid body product of inertia  $\mathbf{I}_{xz} = 0$  as well. For simplicity, also assume that this holds for the corresponding aerodynamic added inertia. About CG, the system inertia matrix  $\mathbf{M}$  becomes diagonal:

$$\mathbf{M} = \mathbf{M}_{RB} + \mathbf{M}_F \quad (3.4)$$

$$= \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{xx} & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{yy} & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{zz} \end{bmatrix} - \begin{bmatrix} X_{\dot{u}} & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{\dot{v}} & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_{\dot{w}} & 0 & 0 & 0 \\ 0 & 0 & 0 & L_{\dot{p}} & 0 & 0 \\ 0 & 0 & 0 & 0 & M_{\dot{q}} & 0 \\ 0 & 0 & 0 & 0 & 0 & N_{\dot{r}} \end{bmatrix} \quad (3.5)$$

$$= \begin{bmatrix} m - X_{\dot{u}} & 0 & 0 & 0 & 0 & 0 \\ 0 & m - Y_{\dot{v}} & 0 & 0 & 0 & 0 \\ 0 & 0 & m - Z_{\dot{w}} & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{xx} - L_{\dot{p}} & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{yy} - M_{\dot{q}} & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{zz} - N_{\dot{r}} \end{bmatrix} \quad (3.6)$$

$$:= \begin{bmatrix} \mathbf{M}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_1 \end{bmatrix} \quad (3.7)$$





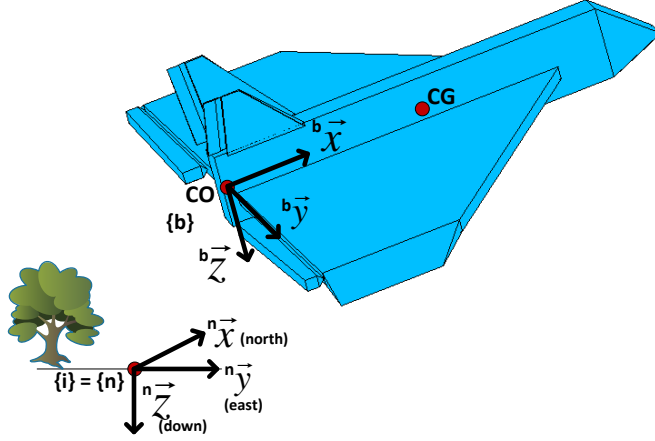


Figure 3.1: Illustration of the inertial frame  $\{n\}$ , body frame  $\{b\}$  and the center of gravity CG (also referred to as the center of mass) and the center of origin CO.  $b_x$  is pointing in the longitudinal direction of the air vehicle,  $b_y$  points in the lateral direction, and  $b_z$  is pointing directly downwards -completing the right hand rule.

In this thesis, the mass matrix is selected as

$$\mathbf{M}_1 = \begin{bmatrix} 2.8 & 0 & 0 \\ 0 & 2.8 & 0 \\ 0 & 0 & 2.8 \end{bmatrix} \text{ kg} \quad (3.19)$$

and the inertia matrix

$$\mathbf{I}_1 = \begin{bmatrix} 0.15 & 0 & 0 \\ 0 & 0.14 & 0 \\ 0 & 0 & 0.29 \end{bmatrix} \text{ kg}\cdot\text{m}^2 \quad (3.20)$$

giving the system inertia matrix

$$\mathbf{M} = \begin{bmatrix} 2.8 & 0 & 0 & & & \\ 0 & 2.8 & 0 & & & \\ 0 & 0 & 2.8 & & & \\ & & & \mathbf{0}_{3 \times 3} & & \\ & & & & 0.15 & 0 & 0 \\ & & & & 0 & 0.14 & 0 \\ & & & & 0 & 0 & 0.29 \end{bmatrix} \quad (3.21)$$

Further, the center of mass is set to

$$\mathbf{r}_{g/b}^b = [0.5 \quad 0 \quad 0]^T \text{ m} \quad (3.22)$$

The overall model structure of the Recce D6 simulator model is shown in figure 3.2

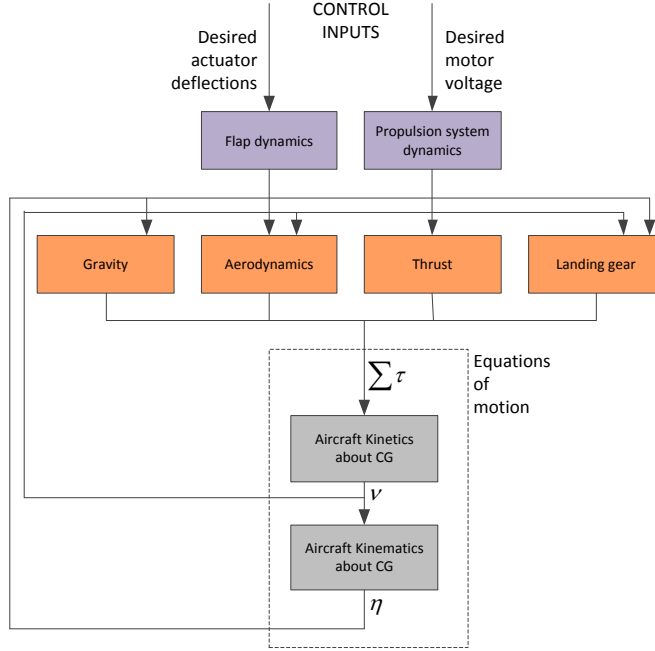


Figure 3.2: Simulator model structure

### 3.1.1 Aerodynamics, $\tau_a$

The aircraft is divided into five bodyparts that induce aerodynamic effects on the motion. To achieve modularity, each body part is modeled with one aerodynamic center of which the induced forces and moments act about. The wings are expected to have their respective aerodynamic centers along the average chord, and at 25% of chord length from the leading edge. The fuselage is simply guesstimated. One coordinate system is defined in each aerodynamic center. The respective position and orientations of the aerodynamic center coordinate systems are guesstimated from the drawings and photo of Recce D6 in appendix D, and are set to

$$\begin{aligned} \text{Fuselage } \{a_f\} \quad \mathbf{r}_{af/b}^b &= [ 0.5 \quad 0 \quad 0.03 ]^T \\ \Theta_{af/b} &= [ 0^\circ \quad 0^\circ \quad 0^\circ ] \\ \Rightarrow \mathbf{q}_{af/b} &= [ 1 \quad 0 \quad 0 \quad 0 ]^T \end{aligned}$$

$$\begin{aligned} \text{Left wing } \{a_{wl}\} \quad \mathbf{r}_{awl/b}^b &= [ 0.35 \quad -0.42 \quad 0 ]^T \\ \Theta_{awl/b} &= [ 0^\circ \quad 0^\circ \quad 0^\circ ] \\ \Rightarrow \mathbf{q}_{awl/b} &= [ 1 \quad 0 \quad 0 \quad 0 ]^T \end{aligned}$$



$$\begin{aligned} \text{Right wing } \{a_{wr}\} \quad \mathbf{r}_{awr/b}^b &= [ 0.35 \quad 0.42 \quad 0 ]^T \\ \Theta_{awr/b} &= [ 0^\circ \quad 0^\circ \quad 0^\circ ] \\ \Rightarrow \mathbf{q}_{awr/b} &= [ 1 \quad 0 \quad 0 \quad 0 ]^T \end{aligned}$$

$$\begin{aligned} \text{Left stabilizer } \{a_{sl}\} \quad \mathbf{r}_{asl/b}^b &= [ 0.07 \quad 0.09 \quad 0 ]^T \\ \Theta_{asl/b} &= [ -60^\circ \quad 0^\circ \quad 0^\circ ] \\ \Rightarrow \mathbf{q}_{asl/b} &= [ \frac{\sqrt{3}}{2} \quad -\frac{1}{2} \quad 0 \quad 0 ]^T \end{aligned}$$

$$\begin{aligned} \text{Right stabilizer } \{a_{sr}\} \quad \mathbf{r}_{asr/b}^b &= [ 0.07 \quad -0.09 \quad 0 ]^T \\ \Theta_{asr/b} &= [ 60^\circ \quad 0^\circ \quad 0^\circ ] \\ \Rightarrow \mathbf{q}_{asr/b} &= [ \frac{\sqrt{3}}{2} \quad \frac{1}{2} \quad 0 \quad 0 ]^T \end{aligned}$$

The orientation and position of the aerodynamic centers are shown in figure 3.3.

Before proceeding, some assumptions are to be made. First, the model will neglect forces due to buoyancy. The aircraft density is large compared to the density of air. Further, low airspeed and low altitudes are assumed. Low airspeed means airspeed far below supersonic flight, which in turn leads to the assumption of constant air density regardless of airspeed. The low altitudes leads to the assumption of constant air density regardless of altitude. In total, this suggests air incompressibility, and thus constant air density for all feasible flight conditions of Recce D6.

### Relative air velocity

The lift and drag components are given as relative to the airstream about the aerodynamic centers;  $\mathbf{v}_{ai/w}^{a_i}$ . It is easy to see that the air velocity of CG can be written as the sum  $\vec{v}_{g/n} = \vec{v}_{g/w} + \vec{v}_{w/n}$ . Expecting the wind velocity to be specified in NED coordinates, the air velocity in BODY coordinates is found to be

$$\mathbf{v}_{g/w}^b = \mathbf{v}_{g/n}^b - \mathbf{R}_n^b \mathbf{v}_{w/n}^n \quad (3.23)$$

The actual air velocity in the aerodynamic centers are then found to be

$$\mathbf{v}_{ai/w}^{a_i} = \mathbf{R}_b^{a_i} \mathbf{v}_{ai/w}^b \quad (3.24)$$

$$= \mathbf{R}_b^{a_i} \left( \mathbf{v}_{g/w}^b + \boldsymbol{\omega}_{g/w}^b \times \mathbf{r}_{ai/g}^b \right) \quad (3.25)$$

By assuming  $\boldsymbol{\omega}_{g/w}^b \approx \boldsymbol{\omega}_{b/n}^b$  we get the air velocity about the aerodynamic centers

$$\mathbf{v}_{ai/w}^{a_i} = \mathbf{R}_b^{a_i} \left( \mathbf{v}_{g/w}^b + \boldsymbol{\omega}_{b/n}^b \times \mathbf{r}_{ai/g}^b \right) \quad (3.26)$$

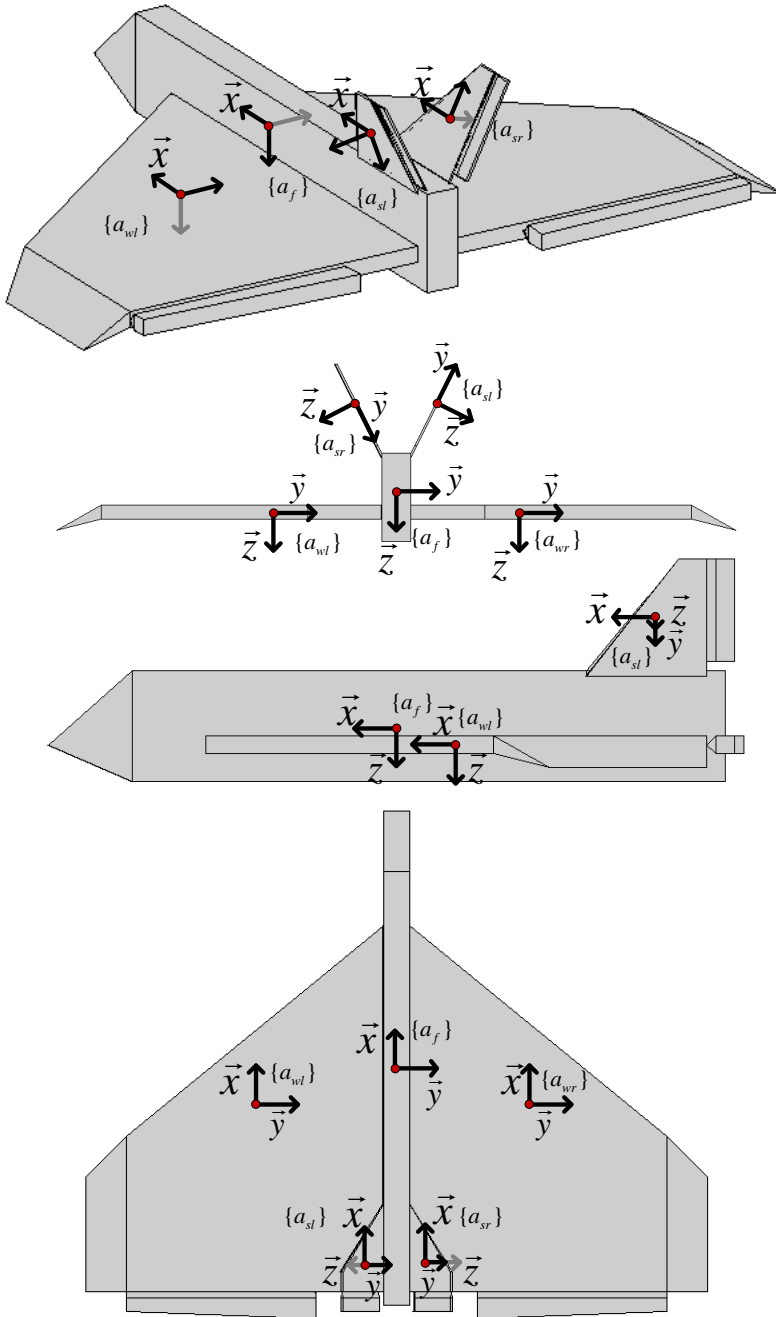


Figure 3.3: The aerodynamics is modeled as five distinct aerodynamic centers.

For aerodynamics, it is useful to decompose this velocity vector into angle of attack  $\alpha$ , sideslip angle  $\beta$  and the total airspeed  $V_w$  according to (Fossen, 2011a)

$$\mathbf{v}_{\alpha_i/w}^{a_i} := \begin{bmatrix} \tilde{u} & \tilde{v} & \tilde{w} \end{bmatrix}^T \quad (3.27)$$

$$V_w = \sqrt{\tilde{u}^2 + \tilde{v}^2 + \tilde{w}^2} \quad (3.28)$$

$$\alpha = \text{atan2}(\tilde{w}, \tilde{u}) \quad (3.29)$$

$$\beta = \text{asin}\left(\frac{\tilde{v}}{V_w}\right) \quad (3.30)$$

### Aerodynamic forces and moments

Lift and pitching moment is modeled as scalar entities restricted to produce force and moment in the longitudinal direction of the respective aerodynamic center frame. This is done because the aircraft is expected to have very high velocity in the longitudinal direction compared to the lateral direction, so the aircraft designer has optimized lift and minimized drag under the same assumption. The result is neglectable lift and moment contributions due to lateral airstream - for instance from wind gusts. Drag, on the other hand, gives significantly more contribution to the motion under transverse motion, so drag force can not be neglected in any directions. However, for simplicity, the drag components are completely decoupled along the three principal axes of the aerodynamic center frames.

In each aerodynamic center, the aerodynamic forces and moments are found according to (Using (2.35),(2.36) and (2.37))

$$\mathbf{L}^{a_i} = \mathbf{R}_{y,-\alpha} \begin{bmatrix} 0 \\ 0 \\ -\frac{1}{2}\rho V_w^2 A_L C_L(\alpha, \delta) \end{bmatrix} \quad (3.31)$$

$$= \begin{bmatrix} \cos \alpha & 0 & -\sin \alpha \\ 0 & 1 & 0 \\ \sin \alpha & 0 & \cos \alpha \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -\frac{1}{2}\rho V_w^2 A_L C_L(\alpha, \delta) \end{bmatrix} \quad (3.32)$$

$$= \frac{1}{2}\rho \begin{bmatrix} V_w^2 A_L C_L(\alpha, \delta) \sin \alpha \\ 0 \\ -V_w^2 A_L C_L(\alpha, \delta) \cos \alpha \end{bmatrix} \quad (3.33)$$

$$= \frac{1}{2}\rho \begin{bmatrix} |\tilde{w}| \tilde{w} A_L C_L(\alpha, \delta) \\ 0 \\ -|\tilde{u}| \tilde{u} A_L C_L(\alpha, \delta) \end{bmatrix} \quad (3.34)$$

$$\mathbf{D}^{a_i} = -\frac{1}{2}\rho \begin{bmatrix} |\tilde{u}| \tilde{u} A_x C_{Dx}(\delta, C_L) \\ |\tilde{v}| \tilde{v} A_y C_{Dy}(\delta, C_L) \\ |\tilde{w}| \tilde{w} A_z C_{Dz}(\delta, C_L) \end{bmatrix} \quad (3.35)$$

$$\mathbf{M}_p^{a_i} = \begin{bmatrix} 0 \\ \frac{1}{2}\rho V_w^2 A_{Mp} \bar{c} C_{Mp}(\alpha, \delta) \\ 0 \end{bmatrix} \quad (3.36)$$

such that

$$\mathbf{f}_{a_i}^{a_i} = \mathbf{L}^{a_i} + \mathbf{D}^{a_i} \quad (3.37)$$

$$\mathbf{m}_{a_i}^{a_i} = \mathbf{M}_p^{a_i} \quad (3.38)$$

or in component form

$$\mathbf{f}_{a_i}^{a_i} = \frac{1}{2}\rho \begin{bmatrix} V_w^2 A_L C_L(\alpha, \delta) \sin \alpha - |\tilde{u}| \tilde{u} A_x C_{Dx}(\delta, C_L) \\ -|\tilde{v}| \tilde{v} A_y C_{Dy}(\delta, C_L) \\ -V_w^2 A_L C_L(\alpha, \delta) \cos \alpha - |\tilde{w}| \tilde{w} A_z C_{Dz}(\delta, C_L) \end{bmatrix} \quad (3.39)$$

$$\mathbf{m}_{a_i}^{a_i} = \begin{bmatrix} 0 \\ \frac{1}{2}\rho V_w^2 A_{Mp} \bar{c} C_{Mp}(\alpha) \\ 0 \end{bmatrix} \quad (3.40)$$

About CG, the aerodynamic forces and moments become

$$\mathbf{f}_g^b = \mathbf{R}_{a_i}^b \mathbf{f}_{a_i}^{a_i} \quad (3.41)$$

$$\mathbf{m}_g^b = \mathbf{R}_{a_i}^b \left( \mathbf{m}_{a_i}^{a_i} + \mathbf{r}_{a_i/g}^b \times \mathbf{f}_{a_i}^{a_i} \right) \quad (3.42)$$

where

$$\mathbf{r}_{a_i/g}^b = \mathbf{r}_{a_i/co}^b - \mathbf{r}_{g/co}^b \quad (3.43)$$

$$= \begin{bmatrix} x_{a_i} - x_g \\ y_{a_i} - y_g \\ z_{a_i} - z_g \end{bmatrix} \quad (3.44)$$

such that the total forces and moments vector applied in (3.2) from all aerodynamic centers become

$$\boldsymbol{\tau}_a = \sum_{i \in \mathcal{A}} \boldsymbol{\tau}_{a_i} \quad (3.45)$$

$$= \sum_{i \in \mathcal{A}} \left[ \mathbf{R}_{a_i}^b \left( \mathbf{m}_{a_i}^{a_i} + \mathbf{r}_{a_i/g}^b \times \mathbf{f}_{a_i}^{a_i} \right) \right] \quad (3.46)$$

$$\mathcal{A} = \{f, wl, wr, sl, sr\} \quad (3.47)$$

The aircraft has not been available for wind tunnel testing or trial flights. Until reasonable aerodynamic coefficients can be obtained by empiric tests, some intermediate assumptions has to be made for simple simulating-purposes:

**Lift coefficient**

The lift coefficient is modeled as

$$C_L(\alpha, \delta) = \begin{cases} C_L(\alpha_{eff}) & \text{if } \alpha_1 \leq \alpha_{eff} \leq \alpha_2 \\ 0 & \text{otherwise} \end{cases} \quad (3.48)$$

$$C_L(\alpha_{eff}) := C_{Lmax} \sin\left(\frac{\pi}{2(\alpha_s - \alpha_0)}(\alpha_{eff} - \alpha_0)\right) \quad (3.49)$$

$$\alpha_1 := \alpha_0 - 2(\alpha_s - \alpha_0) \quad (3.50)$$

$$\alpha_{eff} := \alpha - k_\delta \delta \quad (3.51)$$

$$\alpha_2 := \alpha_0 + 2(\alpha_s - \alpha_0) \quad (3.52)$$

$$k_\delta := \left. \frac{\frac{\partial}{\partial \delta} C_L(\alpha, R_e, \delta)}{\frac{\partial}{\partial \alpha} C_L(\alpha, R_e, \delta)} \right|_{\substack{\alpha = \alpha_0 \\ \delta = 0}} \quad (3.53)$$

where

$\alpha$	Zero angle of attack	[rad]
$\delta$	Flap deflection	[rad]
$C_{Lmax}$	Max coefficient of lift	[.]
$\alpha_s$	Stall angle	[rad]
$\alpha_0$	Zero lift angle	[rad]
$k_\delta$	Flap/AoA ratio	[.]
$\alpha_{eff}$	“Effective” angle of attack	[rad]

The lift coefficient estimator (3.48)-3.53 has been implemented into the project directory as the Matlab function

```
CL = CLift(AoA, delta, CLmax, as, a0, kdelta)
```

and a plot of the the lift coefficient estimator is shown in figure (3.4). This illustration may also be opened in Matlab by running the `CLiftDemo.m` script from the project folder.

**Drag coefficient**

The drag coefficient

$$\mathbf{C}_D = C_{Di} \begin{bmatrix} C_L^2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + C_{Df} + \begin{bmatrix} k_{\delta x} \sin(|\delta|) & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & k_{\delta z} \cos(|\delta|) \end{bmatrix} \quad (3.54)$$

$$= \begin{bmatrix} C_{Di} C_L^2 + C_{Dx} + k_{\delta x} \sin(|\delta|) & 0 & 0 \\ 0 & C_{Dy} & 0 \\ 0 & 0 & C_{Dz} + k_{\delta z} \cos(|\delta|) \end{bmatrix} \quad (3.55)$$

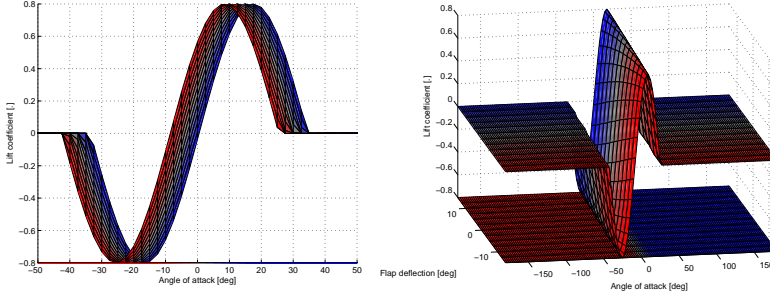


Figure 3.4: Lift coefficient estimator (3.48)-(3.51). The leftmost illustration shows the lift coefficient using the color map in figure 6.3 to color the flap deflection angle. The rightmost illustration shows full picture. The bed is colored according to color of the wing (angle of attack), and the lift coefficient landscape is colored according to color of the flap (the flap deflection angle.)

where

$C_{Di}$	Induced drag coefficient (Hoerner, 1958)	[.]
$C_L$	Lift coefficient	[.]
$\mathbf{C}_{Df} = \text{diag}(C_{Dx}, C_{Dy}, C_{Dz})$	Form drag coefficients	[.]
$\delta$	Flap deflection	[rad]
$k_{\delta x}$	Flap/AoA ratio	[.]
$k_{\delta z}$	Flap/AoA ratio	[.]

### Pitching moment coefficient

Pitching moment coefficient is modeled as

$$C_{Mp} := -C_{Mmax} \sin(\alpha_{eff} - \alpha_{M0}) \quad (3.56)$$

$$\alpha_{eff} := \alpha - k_{\delta} \delta \quad (3.57)$$

$$k_{\delta} := \left. \frac{\frac{\partial}{\partial \delta} C_{Mp}(\alpha, R_e, \delta)}{\frac{\partial}{\partial \alpha} C_{Mp}(\alpha, R_e, \delta)} \right|_{\substack{\alpha = \alpha_0 \\ \delta = 0}} \quad (3.58)$$

where

$C_{Mp}$	Pitching moment coefficient	[.]
$C_{Mmax}$	Max pitching moment coefficient	[.]
$\alpha_{eff}$	“Effective” angle of attack	[rad]
$\alpha_{M0}$	Angle of zero pitching moment	[rad]
$\alpha$	Zero angle of attack	[rad]
$k_{\delta}$	Flap/AoA ratio	[.]
$\delta$	Flap deflection	[rad]
$\alpha_0$	Zero lift angle	[rad]

The lift pitching moment coefficient estimator (3.56)-3.58 has been implemented into the project directory as the Matlab function

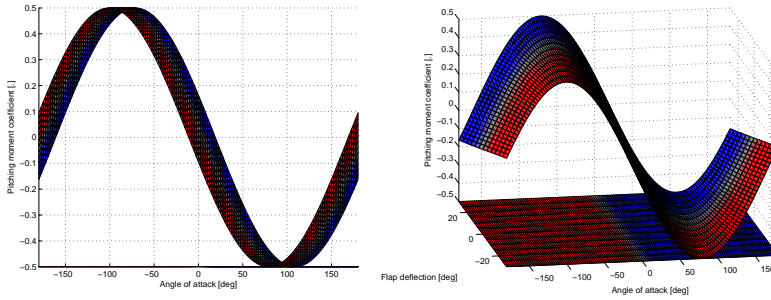


Figure 3.5: Pitching moment coefficient estimator (3.56). The leftmost illustration shows the aerodynamic pitching moment coefficient using the color map in figure 6.3 to color the flap deflection angle. The bed is colored according to color of the wing (angle of attack), and the lift coefficient landscape is colored according to color of the flap (the flap deflection angle.)

```
1 Cm = Cpmoment(AoA,delta,Cmmax,cm0,kdelta)
```

and a plot of the the pitching moment coefficient estimator is shown in figure 3.5. This illustration may also be opened in Matlab by running the `CMomentDemo.m` script from the project folder.

### 3.1.2 Thrust, $\tau_t$

Thrust force is modeled as (Allerton, 2009)

$$f_T = \frac{1}{2} C_T \rho |n| n D^4 \quad (3.59)$$

where

$f_t$	Thrust force	[N]
$n$	Propeller revolutions per minute	[rpm]
$C_t$	Coefficient of thrust	[.]
$\rho$	Density of air	[kg/m <sup>3</sup> ]
$D$	Propeller diameter	[m]

The thrust force direction is expected to be fixed in the longitudinal direction of the aircraft, or in other words in parallel with  ${}^b\vec{x}$ . Hence, no rotations needs to be applied to the thrust force and moment. The thrust force is then  $\mathbf{f}_t^b = [f_t \ 0 \ 0]^T$  With respect to CG, the thrust force is at

$$\mathbf{r}_{t/g}^b = \mathbf{r}_{t/b}^b - \mathbf{r}_{g/b}^b \quad (3.60)$$

$$= \begin{bmatrix} x_t - x_g \\ y_t - y_g \\ z_t - z_g \end{bmatrix} \quad (3.61)$$

It is expected that the moment induced by an accelerating propeller rotation can be neglected. The applied force and moments to the equation of motion is then

$$\boldsymbol{\tau}_t = \left[ \begin{array}{c} \mathbf{f}_t^b \\ \mathbf{m}_t^b + \mathbf{r}_{t/g}^b \times \mathbf{f}_t^b \end{array} \right] \quad (3.62)$$

or in component form

$$\boldsymbol{\tau}_t = \left[ \begin{array}{c} \frac{1}{2} C_T \rho |n| n D^4 \\ 0 \\ 0 \\ m_p \\ -f_t (z_t - z_g) \\ 0 \end{array} \right] \quad (3.63)$$

where  $\mathbf{m}_p^b = [m_p \ 0 \ 0]^T$  is the moment induced to the aircraft from the propeller angular velocity, see section 3.2.

For recce D6, the location of the thrust frame is guesstimated to be located at

Thrust  $\{t\}$   $\mathbf{r}_{t/b}^b = [x_t \ y_t \ z_t]^T = [0.06 \ 0 \ -0.075]^T$   
with orientation coincident with  $\{b\}$ .

### 3.1.3 Landing gear, $\tau_{LG}$

Recce D6 is assumed to have three landing gear wheels; One nose wheel, one left wheel and one right wheel. Forces induced to the aircraft by the landing gear is of obvious importance during landing and take off. The landing gear of Recce D6 is not retractable, so the drag it produces in air is considered to be part of the drag induced by the fuselage. Hence, the landing gear only induces forces to the aircraft when the wheels are displaced, i.e. there is no effect on the aircraft dynamics while in air.

The wheel-frame locations in body are set to

$$\begin{array}{ll} \text{Nose wheel} & \{w_n\} \quad \mathbf{r}_{w_n/b}^b = [-1 \ 0 \ 0.1]^T \\ \text{Left wheel} & \{w_l\} \quad \mathbf{r}_{w_l/b}^b = [0 \ -0.5 \ 0.1]^T \\ \text{Right wheel} & \{w_r\} \quad \mathbf{r}_{w_r/b}^b = [0 \ 0.5 \ 0.1]^T \end{array}$$

The position of the wheel with respect to the center of gravity is

$$\mathbf{r}_{w_i/g}^b = \mathbf{r}_{w_i/b}^b - \mathbf{r}_{g/b}^b \quad (3.64)$$

To calculate the displacement, the position of the wheels in NED is found:

$$\mathbf{r}_{w_i/n}^n = \mathbf{r}_{b/n}^n + \mathbf{R}_b^n \mathbf{r}_{w_i/b}^b \quad (3.65)$$

$$\mathbf{r}_{w_i/n}^n := [x_{w_i} \ y_{w_i} \ z_{w_i}]^T \quad (3.66)$$



Hence,  $z_{w_i}$  defines the wheel displacement  $\Delta z_{w_i}$ , if  $z_{w_i} > 0$  such that

$$\Delta z_{w_i} := \begin{cases} z_{w_i} & \text{if } z_{w_i} > 0 \\ 0 & \text{else} \end{cases} \quad (3.67)$$

The wheels are simply assumed to apply a normal force to the aircraft due to displacement according to

$$\mathbf{f}_{\Delta z}^b = -\mathbf{K}_s \begin{bmatrix} 0 & 0 & \Delta z_{w_i} \end{bmatrix}^T \quad (3.68)$$

where  $\mathbf{K}_s$  is a matrix of spring coefficients of the form  $\mathbf{K}_s = \text{diag} \begin{bmatrix} 0 & 0 & k_{sz} \end{bmatrix}$ .

The landing gear also produce damping during landing and take-off. The velocity of each wheel in NED are found in BODY coordinates to be

$$\mathbf{v}_{w_i/n}^b = \mathbf{v}_{b/n}^b + \mathbf{r}_{w_i/g}^b \times \omega_{b/n}^b \quad (3.69)$$

$$\mathbf{v}_{w_i/n}^b := \begin{bmatrix} u_{w_i} & v_{w_i} & w_{w_i} \end{bmatrix}^T \quad (3.70)$$

In the  ${}^n z$  direction, energy dissipation (damping) is only applied if the wheel is being increasingly displaced. This is because it does not make physical sense that the landing gear produce a damping force that ‘‘holds the aircraft down’’ during take off. The damping is simply modeled as

$$\mathbf{f}_{w_f}^b = - \begin{cases} \mathbf{K}_d \mathbf{v}_{w_i/n}^b & \text{if } \Delta z_{w_i} > 0 \text{ and } w_{w_i} > 0 \\ \check{\mathbf{K}}_d \mathbf{v}_{w_i/n}^b & \text{else if } \Delta z_{w_i} > 0 \\ \mathbf{0}_{3 \times 1} & \text{else} \end{cases} \quad (3.71)$$

where  $\mathbf{K}_d$  is a matrix of damping coefficients of the form  $\mathbf{K}_d = \text{diag} \begin{bmatrix} k_{dx} & k_{dy} & k_{dz} \end{bmatrix}$ , and  $\check{\mathbf{K}}_d = \text{diag} \begin{bmatrix} k_{dx} & k_{dy} & 0 \end{bmatrix}$ .

In CG, the landing gear forces and moments become

$$\mathbf{f}_{w_i}^b = \mathbf{f}_{w_f}^b + \mathbf{f}_{\Delta z}^b \quad (3.72)$$

$$\mathbf{m}_g^b = \mathbf{r}_{w_i/g}^b \times \mathbf{f}_{w_i}^b \quad (3.73)$$

where

$$\mathbf{r}_{w_i/g}^b = \mathbf{r}_{w_i/b}^b - \mathbf{r}_{g/b}^b \quad (3.74)$$

$$= \begin{bmatrix} x_{w_i} - x_g \\ y_{w_i} - y_g \\ z_{w_i} - z_g \end{bmatrix} \quad (3.75)$$

such that the total applied forces and moments vector applied to (3.2) from the landing gear become

$$\boldsymbol{\tau}_{lg} = \sum_{i \in \mathcal{G}} \boldsymbol{\tau}_{w_i} \quad (3.76)$$

$$= \sum_{i \in \mathcal{G}} \begin{bmatrix} \mathbf{f}_{w_i}^b \\ \mathbf{r}_{w_i/g}^b \times \mathbf{f}_{w_i}^b \end{bmatrix} \quad (3.77)$$

$$\mathcal{G} = \{nw, lw, rw\} \quad (3.78)$$

### 3.1.4 Tuning the Aerodynamics

Model validation and tuning is a rather big part of the simulator development. It might in fact reasonably amount to 80% of the overall project. (Allerton, 2009) It usually comprises verification by experienced aircraft pilots and comparison of simulation data from actual measurements. Subsequently, both the dynamics and steady state response of the aircraft is validated. Until actual flight measurements, and experienced pilots are available, the only material available for model tuning is the Recce D6-spec provided by Odin Aero AS, listed in appendix D.

#### Characteristic areas

The characteristic areas  $A_L$ ,  $\mathbf{A}_D = [ A_{Dx} \ A_{Dy} \ A_{Dz} ]$ ,  $A_{Mp}$  and average chord length  $\bar{c}$  for the distinct body parts are guesstimated from the drawings and photo of Recce D6 as

Fuselage	$A_L = A_{Mp} = 79 \cdot 10^{-3}$ $\mathbf{A}_D = [ 12.6 \ 95.8 \ 79 ] \cdot 10^{-3}$ $\bar{c} = 1.06$	$[m^2]$ $[m^2]$ $[m]$	$\mathbb{R}^1$ $\mathbb{R}^3$ $\mathbb{R}^1$
Wings	$A_L = A_{Mp} = 381.5 \cdot 10^{-3}$ $\mathbf{A}_D = [ 18.3 \ 52 \ 381.5 ] \cdot 10^{-3}$ $\bar{c} = 0.62$	$[m^2]$ $[m^2]$ $[m]$	$\mathbb{R}^1$ $\mathbb{R}^3$ $\mathbb{R}^1$
Stabilizers	$A_L = A_{Mp} = 38.15 \cdot 10^{-3}$ $\mathbf{A}_D = [ 1.8 \ 5.2 \ 38.15 ] \cdot 10^{-3}$ $\bar{c} = 0.062$	$[m^2]$ $[m^2]$ $[m]$	$\mathbb{R}^1$ $\mathbb{R}^3$ $\mathbb{R}^1$

where  $A_{Dx}$  is the area of Recce projected into the  ${}^b y^b z^b$ -plane, and similar for  $A_{Dy}$  and  $A_{Dz}$ .

#### Lift coefficient, wing

Cruising speed is typically 65 km/t. It is assumed that the aircraft design is done in a way so that the main wings produces the lift needed to be able to fly horizontally ( $\alpha = 0^\circ$ ) at this airspeed and with no elevon deflection ( $\delta = 0^\circ$ ).

Selecting  $C_{Lmax} = 0.6$ ,  $\alpha_0 = -2.5^\circ$ ,  $\alpha_s = 13^\circ$  and  $k_\delta = 0.15$ . With no actuator deflection ( $\delta = 0^\circ$ ) the lift coefficient becomes

$$C_L(\alpha = 0, \delta = 0) = 0.1504 \quad (3.79)$$

The airspeed needed for the two wings to balance the weight of the airplane is

$$L = W \quad (3.80)$$

$$\frac{1}{2} \rho V^2 (2 \cdot A_L) C_L = mg \quad (3.81)$$

$$\Rightarrow V = \sqrt{\frac{mg}{\rho A_L C_L}} \quad (3.82)$$

where  $g = 9.81m/s^2$ ,  $\rho = 1.29kg/m^3$ . Recce D6 has maximum take-off weight (MTOW) 2.8 kg, where 0.5 kg is payload. Hence, the net weight is 2.3 kg. Using (3.82), the following table defines the absolute minimum airspeed desired for take-off (angle of attack equal to stall-angle) and the airspeed desired for horizontal flight

Takeoff weight	Horizontal flight $\alpha = 0^\circ$	Min. take-off speed $\alpha = \alpha_s = 13^\circ$
2.3kg (net. weight)	62.9km/h	31.5km/h
2.55kg	66.2km/h	33.1km/h
2.8kg (max weight)	69.4km/h	34.7km/h

Table 3.1: Airspeeds necessary to maintain horizontal flight, and minimum take-off airspeeds for minimum, typical and maximum take-off weights.

Without further discussion, the airspeeds in table 3.1 to balance the weight is simply considered to be “fair”, and hence this concludes the choice of lift coefficient parameters for the main wings.

### Drag coefficient, wing

The induced drag coefficient  $C_{Di}$  is estimated from the drag polar of the NACA-2408 airfoil, and set to  $C_{Di} = 0.16$ . Simply expecting a lift/drag ratio of 15 for the wings at  $\alpha, \delta = 0$  gives the balance

$$L = 15 \cdot D \quad (3.83)$$

$$\frac{1}{2}\rho V^2 A_L C_L = 15 \cdot \frac{1}{2}\rho V^2 A_{Dx} (C_{Dx} + C_{Di} C_L^2) \quad (3.84)$$

$$\Rightarrow C_{Dx} = \frac{A_L C_L - 7.5 \cdot A_{Dx} C_{Di} C_L^2}{15 \cdot A_{Dx}} \quad (3.85)$$

$$= \frac{381.5 \cdot 10^{-3} \cdot 0.1504 - 7.5 \cdot 18.3 \cdot 10^{-3} \cdot 0.16 \cdot 0.1504^2}{15 \cdot 18.3 \cdot 10^{-3}} \quad (3.86)$$

$$= 0.207 \quad (3.87)$$

$C_{Dy}$  is chosen to be somewhat higher,  $C_{Dx} < C_{Dy} = 0.3$ .  $C_{Dz}$  is chosen to have drag coefficient like a disk  $C_{Dz} = 1.17$ . (White, 2008)

This concludes the choice of drag coefficient for the main wings. The stabilizers are chosen to have the same coefficients, but with a slightly lower  $C_{Dx}$  since this airfoil is expected to be symmetrical.  $C_{D\delta i}, i \in \{x, z\}$  is chosen from pure guessing.

### Pitching moment coefficient, wing

No information in the Recce D6 spec can be used to extract information about the rotational dynamics of the aircraft. The only parameter available for tuning is then an expectation of zero resultant pitching moment about CG at  $V = 65 \text{ km/h}$ ,  $\alpha =$

$0^\circ, \delta = 0^\circ, \mathbf{r}_{g/b}^b = [0.5 \ 0 \ 0]$ . Then, for  $i \in \{lw, rw\}$ , the aerodynamic forces lift and drag produces the moment about the center of mass

$$\mathbf{M} = \mathbf{r}_{a_i/g}^b \times \mathbf{f}_{a_i}^b \quad (3.88)$$

If  $\mathbf{r}_{a_i/g}^b = [x \ y \ z]^T = [-0.15 \ y \ 0]^T$ , and  $\mathbf{f}_{a_i}^b = [-D \ 0 \ L]^T = [-D \ 0 \ -W]^T$ , extracting the pitching moment in this situation yields

$$M_p = -Wx \quad (3.89)$$

$$\frac{1}{2}\rho V^2 \bar{c} A_{Mp} C_{Mp} = -mgx \quad (3.90)$$

$$\Rightarrow C_{Mp} = \frac{-2mgx}{\rho V^2 \bar{c} A_{Mp}} \quad (3.91)$$

$$= \frac{-2 \cdot 2.8 \cdot 9.81 \cdot (-0.15)}{1.29 \cdot \left(\frac{65}{3.6}\right)^2 \cdot 0.62 \cdot 381.5 \cdot 10^{-3}} \quad (3.92)$$

$$= 0.083 \quad (3.93)$$

so it follows that

$$0.083 = -C_{Mmax} \sin(0^\circ - \alpha_{M0}) \quad (3.94)$$

and after some calculations, choosing  $C_{Mmax} = 2$  and  $\alpha_{MO} = 2.38^\circ$  is found to fulfill this.

### Stabilizers

The stabilizer coefficients are mainly chosen to have the same properties as the main wings, but with some adjustment because the airfoil is expected to be symmetrical and the rudders are proportionally bigger than the elevon is with respect to the main wings.

### Fuselage

Fuselage coefficients of drag are simply chosen by considering the table of various drag coefficients for three-dimensional bodies in White (2008 p. 483), and guesstimating based on the shape of the fuselage.

### Summary

The characteristic areas and aerodynamic coefficients and parameters for the aircraft is set to

			Fuselage	Wings	Stabilizers
Lift	$A_L$	$[m^2 \cdot 10^{-3}]$	79	381.5	38.15
	$C_{Lmax}$	$[.]$	0.1	0.6	0.6
	$\alpha_s$	$[deg]$	13°	13°	15°
	$\alpha_0$	$[deg]$	-2°	-4°	0°
	$k_\delta$	$[.]$	0	0.15	0.3
Drag	$A_{Dx}$	$[m^2 \cdot 10^{-3}]$	12.6	18.3	1.83
	$A_{Dy}$	$[m^2 \cdot 10^{-3}]$	95.8	52	5.2
	$A_{Dz}$	$[m^2 \cdot 10^{-3}]$	79	381.5	38.15
	$C_{Di}$	$[.]$	0.16	0.16	0.16
	$C_{Dx}$	$[.]$	0.5	0.209	0.18
	$C_{Dy}$	$[.]$	1	0.3	0.3
	$C_{Dz}$	$[.]$	1	1.17	1.17
	$k_{\delta x}$	$[.]$	0	0.23	0.4
	$k_{\delta z}$	$[.]$	0	0.05	0.1
	Pitch moment	$A_{Mp}$	$[m^2 \cdot 10^{-3}]$	79	381.5
$\bar{c}$		$[m]$	1.06	0.62	0.062
$C_{Mmax}$		$[.]$	1	2	2
$\alpha_{M0}$		$[deg]$	2.3°	2.3°	0°
$k_\delta$		$[.]$	0	0.15	0.25

## 3.2 Propulsion System

The propulsion of the aircraft is provided by a DC motor in conjunction with a voltage regulator and a Li-polymer battery.

Ignoring the dynamics due to armature inductance, the model for thrust is selected as (Partly from Egeland and Gravdahl (2003))

$$\boldsymbol{\tau}_t = [f_t \ 0 \ 0 \ -m_p \ 0 \ 0]^T \quad (3.95)$$

$$f_t = \frac{1}{2} C_t \rho |n| n D^4 \quad (3.96)$$

$$\dot{\omega}_p = \frac{1}{J_m} (m_s - m_p) \quad (3.97)$$

$$m_p = \frac{1}{2} C_p \rho |\omega_p| \omega_p \quad (3.98)$$

$$m_s = i_m k_e \quad (3.99)$$

$$i_m = \frac{u_m}{R_m} \quad (3.100)$$

where

$\tau_t$	Applied thrust and torques in BODY	
$f_t$	Thrust force	[N]
$m_s$	Shaft moment	[Nm]
$C_T$	Coefficient of thrust	[.]
$\rho$	Density of air	[kg/m <sup>3</sup> ]
$n$	Propeller rotation velocity	[rpm]
$D$	Propeller diameter	[m]
$J_m$	Inertia of shaft and propeller	[kg·m <sup>2</sup> ]
$m_s$	Shaft moment	[Nm]
$m_p$	Propeller moment	[Nm]
$C_p$	Coefficient of propeller moment	[.]
$i_m$	Armature voltage	[A]
$k_e$	Motor torque constant	[.]
$u_m$	Applied motor voltage	[V]
$R_m$	Resistance of armature	[Ω]

The model for voltage regulator is selected as

$$u_m = \max \{ u_d \quad u_b \} \quad (3.101)$$

where

$u_d$	Desired voltage	[V]
$u_b$	Battery voltage	[V]

The model for power plant voltage is selected as

$$u_b = u_{ems} - u_i \quad (3.102)$$

$$u_i = i_m R_i \quad (3.103)$$

$$u_{ems} = u_N \left( 1 - e^{\frac{E_b^2}{bE_N}} \right) (a(E_b - E_N) + 1) \quad (3.104)$$

$$\dot{E}_b = -\frac{10}{36} i_m \quad (3.105)$$

where

$u_{ems}$	Electromotive battery voltage	[V]
$u_i$	Battery inner voltage	[V]
$R_i$	Battery inner resistance	[Ω]
$u_N$	Nominal battery voltage	[V]
$E_b$	Energy left in battery	[mAh]
$E_N$	Nominal battery energy	[mAh]
$a$	Design constant of voltage loss	[.]
$b$	Design constant of voltage loss	[.]

### Tuning the propulsion system

The Recce D6 spec lists  $E_N = 8000$  mAh and  $u_N = 11.1$  V for the battery. Further,

setting  $a = 1.2 \cdot 10^{-5}$  and  $b = 20$  the battery voltage as function of remaining battery capacity (3.104) becomes

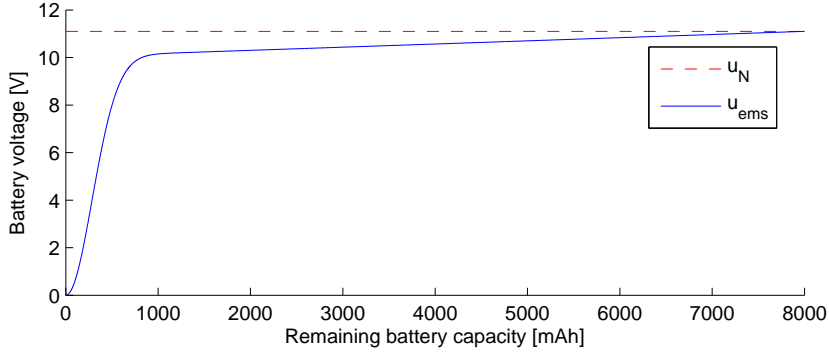


Figure 3.6: Li-pol battery voltage characteristics

which is simply considered to be “fair.”

### 3.3 Flap Dynamics

The geometry of the mechanical connection from servo deflections to flap deflections are unknown. Therefore, the servo motor deflection is modeled to apply flap deflection directly. Assuming saturated first order-like dynamics, the flap dynamics is modeled as

$$\dot{\delta} = \omega_s \quad (3.106)$$

$$\omega_s = \begin{cases} (\delta_d - \delta) k_f & \text{if } |(\delta_d - \delta) k_f| < \omega_{max} \\ \omega_{max} \text{sign}(\delta_d - \delta) & \text{else} \end{cases} \quad (3.107)$$

where

$\omega_s$	Rotation velocity of servo motor	[rad/s]
$\delta_d$	Desired flap deflection angle	[rad]
$\delta$	Actual flap deflection	[rad]
$k_f$	Rotation velocity proportional gain, $k_f > 0$	[·]
$\omega_{max}$	Maximum angular velocity of servo	[rad/s]

The servos are expected to have the same maximum velocity in both directions. From the data sheets of the servo motors, the maximum angular velocities of the flaps are

Right and left wing    Graupner servo C271.  
 Transit speed: 0,12 Sec/40°  
 $\Rightarrow \omega_{max} = \frac{40^\circ}{0.12} = \frac{45^\circ \cdot \pi}{0.12 \cdot 180^\circ} = 7.85[rad/s]$

Stabilizers servo      Robbe Servo FS 500 Mg Micro  
 Transit speed: 0,1 Sec/45°  
 $\Rightarrow \omega_{max} = \frac{45^\circ}{0.1} = \frac{40^\circ \cdot \pi}{0.12 \cdot 180^\circ} = 5.82[rad/s]$

The gain  $k_f$  is set by trial and failure, and set to  $k_f = 10$  for all servos.



## Chapter 4

# Recce D6 Control Design Model

The suggested simulation model from chapter 3 incorporates actuator dynamics, landing gear dynamics, and makes little assumptions on the current flight condition. During normal flight within a restricted flight-envelope, the landing gear forces can be ignored and the aerodynamic coefficients are often treated as constants. For aircraft expected to encounter significantly varying flight conditions, model inaccuracies due to linearization of the coefficients are often compensated for by incorporating gain-scheduling or adaptive control-techniques into the controller. (McLean, 1990) Recce D6 is restricted to quite small speed variations, and in addition low altitudes. Until experience from actual flight prove the opposite, linearization of the coefficients for Recce D6 are expected to be a good assumption for controller deviation.

Re-encountering the Newton-Euler equations of motion for a rigid body (3.2), but without the landing gear forces and moments yields

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\omega})\boldsymbol{\nu} = \boldsymbol{\tau}_g + \boldsymbol{\tau}_a + \boldsymbol{\tau}_t \quad (4.1)$$

where the gravitational force and moment vector is (from (3.12), (3.15))

$$\boldsymbol{\tau}_g = mg \begin{bmatrix} -\sin \theta \\ \cos \theta \sin \phi \\ \cos \theta \cos \phi \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (4.2)$$

and the thrust force and moment vector is (from (3.63) and (3.98))

$$\boldsymbol{\tau}_t = \begin{bmatrix} \frac{1}{2}C_T\rho|n|n \\ 0 \\ 0 \\ \frac{1}{2}C_p\rho|\omega_p|\omega_p \\ -\frac{1}{2}C_T\rho|n|nD^4(z_t - z_g) \\ 0 \end{bmatrix} \quad (4.3)$$

Collecting all constants (and noticing that  $\omega \propto n$ ) gives the thrust force and moment

$$\boldsymbol{\tau}_t = \begin{bmatrix} X_{|n|n}|n|n \\ 0 \\ 0 \\ L_{|n|n}|n|n \\ M_{|n|n}|n|n \\ 0 \end{bmatrix} \quad (4.4)$$

For aerodynamics, some assumptions should be made. A controller should only be expected to be valid for a certain limited set of flight conditions; a restricted flight envelope. For control design, consider the following assumptions of the flight condition:

- All aerodynamics forces and moments can be represented in CG
- Angle of attack and sideslip is zero; WIND and BODY coordinate frames are coincident.
- Linearization of the aerodynamic coefficients is fair for a limited flight envelope.
- The UAV has high velocity compared to the wind velocity.  $\Rightarrow \tilde{u} \approx u, \tilde{w} \approx w$ .
- The left and right rudder deflections are equal in magnitude;  $\delta_{rl} = -\delta_{rr}$ , and the pair of wings and stabilizers with appurtenant actuators are “identical.”

The constant air density assumption from chapter 3 is also retained. Under these assumptions, the aerodynamics can be written

$$\boldsymbol{\tau}_a = \mathbf{A}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{B}(\boldsymbol{\nu})\boldsymbol{\delta} \quad (4.5)$$

$$= \begin{bmatrix} X_{|u|u}|u| & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{|v|v}|v| & 0 & 0 & 0 & 0 \\ Z_{|u|u}|u| & 0 & Z_{|w|w}|w| & 0 & 0 & 0 \\ 0 & 0 & 0 & L_{|p|p}|p| & 0 & L_{|r|r}|r| \\ M_{|u|u}|u| & 0 & 0 & 0 & M_{|q|q}|q| & 0 \\ 0 & 0 & 0 & 0 & 0 & N_{|r|r}|r| \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \end{bmatrix} \\ + |u|u \begin{bmatrix} X_{\delta e} & X_{\delta e} & X_{\delta r} \\ 0 & 0 & Y_{\delta r} \\ Z_{\delta e} & Z_{\delta e} & Z_{\delta r} \\ L_{\delta e} & -L_{\delta e} & L_{\delta r} \\ M_{\delta e} & M_{\delta e} & M_{\delta r} \\ 0 & 0 & N_{\delta r} \end{bmatrix} \begin{bmatrix} \delta_{el} \\ \delta_{er} \\ \delta_{rl} \end{bmatrix} \quad (4.6)$$

where the diagonal elements of  $\mathbf{A}(\boldsymbol{\nu})$  are coefficients representing drag terms,  $Z_{|u|u}|u|u$  is a coefficient for lift,  $M_{|u|u}|u|u$  is a coefficient for pitching moment, and  $L_{|r|r}|r|r$  is a coefficient for roll moment due to yaw rate. In theory, there are couplings between all degrees of freedom, but this model is by engineering judgment expected to retain the most significant properties of the flight dynamics that a controller should capture.

The final nonlinear 6 DOF model is then

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\omega})\boldsymbol{\nu} - \boldsymbol{\tau}_g = \mathbf{A}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{B}(\boldsymbol{\nu})\boldsymbol{\delta} + \boldsymbol{\tau}_t \quad (4.7)$$

or in components

$$\begin{aligned} (m - X_{\dot{u}})(\dot{u} - rv + qw) + mg \sin \theta &= X_{|u|u}|u|u \\ &+ X_{\delta e}(\delta_{el} + \delta_{er})|u|u \\ &+ X_{\delta r}\delta_{rl}|u|u \\ &+ X_{|n|n}|n|n \end{aligned} \quad (4.8)$$

$$\begin{aligned} (m - Y_{\dot{v}})(\dot{v} + ru - pw) - mg \cos \theta \sin \phi &= Y_{|v|v}|v|v \\ &+ Y_{\delta r}\delta_{rl}|u|u \end{aligned} \quad (4.9)$$

$$\begin{aligned} (m - Z_{\dot{w}})(\dot{w} - qu + pv) - mg \cos \theta \cos \phi &= Z_{|u|u}|u|u + Z_{|w|w}|w|w \\ &+ Z_{\delta e}(\delta_{el} + \delta_{er})|u|u \\ &+ Z_{\delta r}\delta_{rl}|u|u \end{aligned} \quad (4.10)$$

$$\begin{aligned} (I_{xx} - L_{\dot{p}})\dot{p} + (I_{zz} - N_{\dot{r}} - I_{yy} + M_{\dot{q}})rq &= L_{|p|p}|p|p + L_{|r|r}|r|r \\ &+ L_{\delta e}(\delta_{el} - \delta_{er})|u|u \\ &+ L_{\delta r}\delta_{rl}|u|u \\ &+ L_{|n|n}|n|n \end{aligned} \quad (4.11)$$

$$\begin{aligned} (I_{yy} - M_{\dot{q}})\dot{q} + (I_{xx} - L_{\dot{p}} - I_{zz} + N_{\dot{r}})rp &= M_{|u|u}|u|u + M_{|q|q}|q|q \\ &+ M_{\delta e}(\delta_{el} + \delta_{er})|u|u \\ &+ M_{\delta r}\delta_{rl}|u|u \\ &+ M_{|n|n}|n|n \end{aligned} \quad (4.12)$$

$$\begin{aligned} (I_{zz} - N_{\dot{r}})\dot{r} + (I_{yy} - M_{\dot{q}} - I_{xx} + L_{\dot{p}})pq &= N_{|r|r}|r|r \\ &+ N_{\delta r}\delta_{rl}|u|u \end{aligned} \quad (4.13)$$

Writing out the components of the kinematics (3.1) of the system using the Euler angles representation for attitude, (2.8) and (2.26) yield (Fossen, 2011b)

$$\begin{bmatrix} \dot{N} \\ \dot{E} \\ \dot{D} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} uc\psi c\theta + v(c\psi s\theta s\phi - s\psi c\phi) + w(s\psi s\phi + c\psi s\theta c\phi) \\ us\psi c\theta + v(c\psi c\phi + s\psi s\theta s\psi) + w(s\psi s\theta c\phi - c\psi s\phi) \\ -us\theta + vc\theta s\phi + wc\theta c\phi \\ p + qs\phi t\theta + rc\phi t\theta \\ qc\phi - rs\phi \\ q\frac{s\phi}{c\theta} + r\frac{c\phi}{s\theta} \end{bmatrix} \quad (4.14)$$

## 4.1 Surge Velocity Control

For surge velocity control, consider equation 4.8, and make the following assumptions

- Only the motor is used for surge control.  $\Rightarrow \delta_{el}, \delta_{er}, \delta_{rl} \approx 0$
- Heave, sway and angular velocities is small compared to surge, so that second order terms are neglectable.  $rv, qw \approx 0$

The surge velocity kinetics become

$$(m - X_{\dot{u}}) \dot{u} + mg \sin \theta - X_{|u|u} |u| u = X_{|n|n} |n| n \quad (4.15)$$

Ignoring the dynamics of the propulsion system, assume that the applied voltage to the motor armature gives a certain motor *rpm* directly;  $u_m \propto \omega_p \propto n$  and the propeller is only used for forward thrust;  $u_m > 0$

$$(m - X_{\dot{u}}) \dot{u} + mg \sin \theta - X_{|u|u} |u| u = X_{u_m^2} u_m^2 \quad (4.16)$$

It is thus seen that the surge velocity control may be achieved by inputting a certain desired armature voltage to the motor. Notice that the  $u_m > 0$  condition implies that the motor can not be used as airbrakes by reversing the propeller rotation direction.

## 4.2 Altitude Control

For altitude control, assume that the surge velocity controller maintains a certain velocity. Next, assume that the aircraft is flying close to horizontally ( $\phi, p = 0$ ) and with constant heading ( $r = 0$ ). From (4.14), the kinematics for altitude ( $h$ ) becomes

$$\dot{h} = -\dot{D} = u \sin \theta - w \cos \theta \quad (4.17)$$

which reveals couplings to surge velocity  $u$ , heave velocity  $w$ , and pitch angle  $\theta$ . Under the same assumption, the pitch kinematics is

$$\dot{\theta} = q \quad (4.18)$$

Hence, the kinetics of surge velocity  $u$ , heave velocity  $w$  and the pitching moment  $q$  is determining the dynamics of altitude. The relevant kinetics are then given by

(4.8), (4.10) and (4.12), reduced to

$$\begin{aligned} (m - X_{\dot{u}}) (\dot{u} + qw) + mg \sin \theta &= X_{|u|u} |u| u \\ &+ X_{\delta_e} (\delta_{el} + \delta_{er}) |u| u + X_{\delta_r} \delta_{rl} |u| u \\ &+ X_{|n|n} |n| n \end{aligned} \quad (4.19)$$

$$\begin{aligned} (m - Z_{\dot{w}}) (\dot{w} - qu) - mg \cos \theta \cos \phi &= Z_{|u|u} |u| u + Z_{|w|w} |w| w \\ &+ Z_{\delta_e} (\delta_{el} + \delta_{er}) |u| u + Z_{\delta_r} \delta_{rl} \end{aligned} \quad (4.20)$$

$$\begin{aligned} (I_{yy} - M_{\dot{q}}) \dot{q} &= M_{|u|u} |u| u + M_{|q|q} |q| q \\ &+ M_{\delta_e} (\delta_{el} + \delta_{er}) |u| u + M_{\delta_r} \delta_{rl} |u| u \\ &+ M_{|n|n} |n| n \end{aligned} \quad (4.21)$$

Consider the following assumptions

- The rudders gives little contribution to pitching moment, and are much more coupled to a yawing motion. Thus, they are not reasonably used for pitch /altitude control.  $\Rightarrow \delta_{rl} \approx 0$
- Elevon deflections give little contribution to lift and drag compared to the pitching moment.  $\Rightarrow X_{\delta_e}, Z_{\delta_e} \approx 0$
- To avoid rolling, the pitch/altitude controller should apply the same deflection to both elevons.  $\delta_e := \delta_{el} + \delta_{er}$ ,  $\delta_{el} = \delta_{er}$ .
- The thrust-induced pitching moment is canceled by the aerodynamic pitching moment,  $M_{|u|u} |u| u + M_{|q|q} |q| q = 0$
- Second order cross product terms are neglectable,  $qw, qu \approx 0$

The resulting kinetics become

$$(m - X_{\dot{u}}) \dot{u} + mg \sin \theta - X_{|u|u} |u| u - X_{|n|n} |n| n = 0 \quad (4.22)$$

$$(m - Z_{\dot{w}}) \dot{w} - mg \cos \theta - Z_{|u|u} |u| u - Z_{|w|w} |w| w = 0 \quad (4.23)$$

$$(I_{yy} - M_{\dot{q}}) \dot{q} = M_{\delta_e} |u| u \delta_e \quad (4.24)$$

and the corresponding kinematics

$$\dot{\theta} = q \quad (4.25)$$

$$\dot{h} = -\dot{D} = u \sin \theta - w \cos \theta \quad (4.26)$$

Where it is seen that an altitude controller only needs to maintain a desired pitch angle  $\theta_d$  by applying  $\delta_e$  in order to achieve climb or descent ( $\dot{h} \neq 0$ ).

## 4.3 Heading Control

The following derivation of banked roll maneuver is inspired by the approaches in McLean (1990), Fossen (2011a) and Nordian Aviation training systems (2005). For

heading control, assume that the pitch /altitude controller ensures constant pitch;  $\theta, q = 0$ , and constant altitude  $\dot{h}, \dot{D}, w = 0$ . The kinematics for altitude (4.14) is reduced to

$$\dot{D} = v \sin \phi = 0 \quad (4.27)$$

It is easily seen that to assure constant altitude, either  $v$  or  $\phi$  must be zero. A common turning maneuver is the banked turn, where  $\phi \neq 0$ , where ailerons (here; elevons) are used to generate rolling rate  $p$ . This suggest a turning maneuver where the rudders are kept at zero deflection;  $\delta_{rl} = 0$ , and an assumption that the altitude controller ensures zero sway velocity ( $v = 0$ ; no sideslip) is established. Equation 4.9 now reads

$$(m - Y_{\dot{v}})ru - mg \sin \phi = 0 \quad (4.28)$$

$$\Rightarrow r = \frac{mg \sin \phi}{u(m - Y_{\dot{v}})} \quad (4.29)$$

From (4.14), the kinematics for heading ( $\psi$ ) become

$$\dot{\psi} = r \cos \phi \quad (4.30)$$

$$= \frac{mg \sin \phi}{u(m - Y_{\dot{v}})} \cos \phi \quad (4.31)$$

which can be linearized to ( $\phi^* = 0$ )

$$\dot{\psi} = \frac{mg}{u(m - Y_{\dot{v}})} \phi \quad (4.32)$$

which reveals a linear relationship between perturbations in roll and yaw rate. Thus, heading control can be ensured by rolling towards the desired heading. Consider the kinetics for roll 4.11 and assume that

- Rudders are not used for roll/heading control;  $\delta_{rl} = 0$
- The moment from the propeller rotation is small,  $L_{|n|n} \approx 0$
- To avoid pitching, the roll/heading controller should apply the opposite deflection to the elevons.  $\Delta\delta_e := \delta_{el} - \delta_{er}$ ,  $\delta_{el} = \delta_{er}$ .
- The roll moment from yaw rate  $r$  is small compared to the roll moment from the elevons.  $L_{|r|r} \approx 0$

The kinetics for roll is then

$$(I_{xx} - L_{\dot{p}})\dot{p} - L_{|p|p}|p|p = L_{\delta_e}|u|u\Delta\delta_e \quad (4.33)$$

and the corresponding kinematics

$$\dot{\psi} = \frac{mg}{u(m - Y_{\dot{v}})} \phi \quad (4.34)$$

$$\dot{\phi} = p \quad (4.35)$$

Where it is seen that an heading controller only needs to maintain a desired roll  $\phi_d$  by applying  $\Delta\delta_e$  in order to achieve turning ( $\dot{\psi} \neq 0$ ).





## Chapter 5

# Motion Control System, Flight Mode

A motion control system can usually be decomposed three different topics, denoted *guidance*, *navigation* and *control* (GNC). The motion control system may comprise these elements both in a tightly coupled way, or as entirely decoupled, independent blocks. Both approaches have advantages and disadvantages. Since this overall project will involve several contributors over a long period of time, in order to facilitate software upgrades and decoupling workloads, a relatively decoupled approach is likely to be the most rational. The following short description of the three blocks are short summaries taken from Fossen (2011b), and adapted to an UAV setting:

- The guidance system computes the reference signal to the control system. This reference is often based on a mission specific control objective, such as set-point regulation, path following or trajectory tracking. It will also often comprise controller modes, such as ground, landing, take-off, emergency or flight mode. It may further be subject to advanced calculations based on for instance obstacle avoidance, weather-optimal routing and energy-optimization. Finally the desired controller reference is subject to reference models so that infeasible requirements to the controller is avoided. Clearly, creating a guidance system can be very easily achieved by simply by defining a set of way-points or set-points without any optimization, or it can be a very comprehensive task of almost unlimited complexity.
- Navigation is the system that reads sensor measurements and produces estimates of relevant aircraft states such as attitude, positions and velocities. The estimates are commonly produced by a Kalman filter or a nonlinear observer.

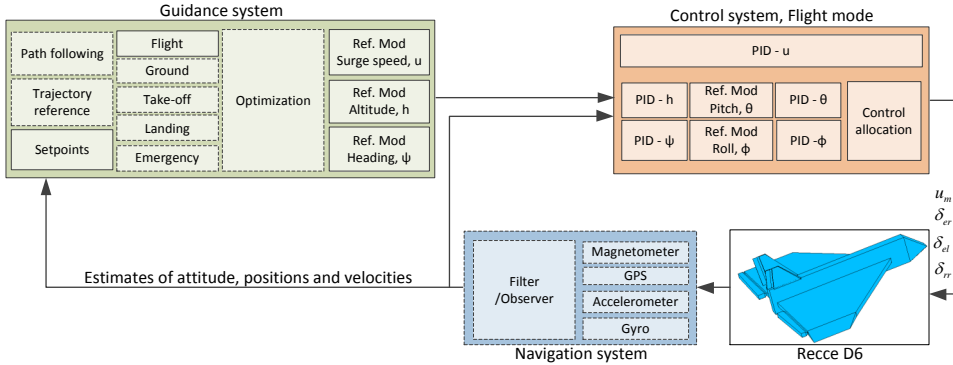


Figure 5.1: The proposed guidance, navigation and control system for Recce D6. The dashed boxes are not considered in this thesis.

- The control system is the system that transforms the reference input from the guidance system to actual commanded actuator deflections based on the control objective, defined by the current controller mode.

A suggestion for a GNC structure for Recce D6 in flight mode is illustrated in figure 5.1. It will be assumed that the control objective is to maintain a certain heading, altitude and surge speed.

The navigation system in figure 5.1 is a rather big and complex topic in it self. In a simulator setting, estimates of positions and velocities are superfluous since all states of the simulated aircraft can be accessed at any time directly. In the remainder of this thesis, the problem of filtering hardware measurements into estimates of positions and velocities are omitted, and all the relevant states are assumed to be perfectly known.

## 5.1 Guidance System

One might imagine a human operated ground station for the UAV where set-points are defined for the UAV during flight. To improve the system response that may arise from controller integrator windup due to infeasible steps in heading and pitch, reference models comprising feasible state propagation is one approach. Combining the human-provided set-points with corresponding reference models completes a suggestion of a simple, easily implementable guidance system.

### 5.1.1 Reference Models

Consider the third-order reference model (Fossen, 2011b) for pitch

$$\theta_r = \frac{\omega_0^3}{(s - \omega_0)^3} \theta_d \quad (5.1)$$

$$= \frac{\omega_0^3}{s^3 + 3\omega_0 s^2 + 3\omega_0^2 s + \omega_0^3} \theta_d \quad (5.2)$$

This may be realized as the linear time-invariant (LTI) system in canonical form

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} -3\omega_0 & -3\omega_0^2 & -\omega_0^3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} \omega_0^3 \\ 0 \\ 0 \end{bmatrix} \theta_d \quad (5.3)$$

$$\theta_r = x_3 \quad (5.4)$$

Note that this approach gives no infeasible discontinuities for the acceleration term  $x_1$ . Further, the acceleration ( $x_1$ ) and velocity terms ( $x_2$ ) may easily be saturated using

$$x_i = \begin{cases} |x_i| \text{sign}(x_i) & \text{if } x_i > x_{imax} \\ x_i & \text{else} \end{cases} \quad (5.5)$$

The same approach may be used for surge velocity, altitude and roll.

The suggested values for the surge speed, altitude, pitch and roll reference models are listed in table 5.1.

### 5.1.2 Heading Reference Model

For all reasonable purposes, pitch (for instance) is restricted to be limited such that

$$\frac{-\pi}{2} \ll \theta_d \ll \frac{\pi}{2} \quad (5.6)$$

which is a handy assumption, because the optimal pitch rate direction is always the “obvious” one;

$$\text{sign} \dot{\theta}_r = \text{sign}(\theta_d - \theta_r) \quad (5.7)$$

Heading on the other hand, may take the values

$$-\pi < \psi \leq \pi \quad (5.8)$$

where a problem of routing the desired heading in the optimal direction arises. This is best illustrated with the example

$$\psi_d = \pi \quad (5.9)$$

$$\psi_r = -\pi + d\psi_r \quad (5.10)$$

$$\Delta\psi = \psi_d - \psi_r \quad (5.11)$$

$$\approx 2\pi \quad (5.12)$$

Variable	$\omega_0$	$x_{1max}$	$x_{3max}$
Surge speed	3.1	15	15
Altitude	1.3	8	5
Pitch	31.4	6.3	6.3
Roll	12.6	3.1	3.1
Heading	0.6	0.3	0.6

Table 5.1: The reference model constants suggested for Recce D6. Found by trial and failure. Heading is set to be somewhat slow for demo purposes.

which clearly reveals that using  $\Delta x$  as input to a reference model yields a very inefficient error representation for heading control, since rotating the opposite direction yields  $-\Delta\psi = \psi_r - \psi_d \approx 0$ . Instead, consider the nonlinear measure of heading error (derived from a simple rotation about  $\vec{z}$  using quaternions) in the optimal heading direction:

$$\tilde{\psi} := \sin\left(\frac{\psi_d}{2} - \frac{\psi_r}{2}\right) \text{sign}\left(\cos\left(\frac{\psi_d}{2} - \frac{\psi_r}{2}\right)\right) \quad (5.13)$$

and the LTI system used to propagate the states of the reference model

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} -3\omega_0 & -3\omega_0^2 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} \omega_0^3 \\ 0 \\ 0 \end{bmatrix} \tilde{\psi} \quad (5.14)$$

$$x_3 = \begin{cases} x_3 & \text{if } x_3 \in (-\pi, \pi] \\ x_3 - 2\pi \text{sign}(x_3) & \text{else} \end{cases} \quad (5.15)$$

$$\psi_r = x_3 \quad (5.16)$$

By limiting  $x_3$  to the interval  $(-\pi, \pi]$ , the reference model is never in danger of producing arithmetic overflow in the target processing device. Further, limiting  $x_3$  does not affect the calculation of  $\tilde{\psi}$ , since both sine and cosine are periodic in  $2\pi$ . Saturating  $x_2$  and  $x_1$  is done as in (5.5). The suggested values for the heading reference model is listed in table 5.1.

The reference models are implemented into Matlab/Simulink, and an animated example using pitch and heading reference models may be run from the project folder by running the `refModelEx.m` script. The same script may also be used to tune the reference models by altering the desired pitch and yaw angles, and trying out different values for the models. In the example animation, the red aircraft illustrates the desired orientation, the green illustrates the reference orientation. Note that by running the script from Matlab, the animation does not run real-time. Therefore, the same example script is also attached in the project folder as the video `refModelEx.avi`.

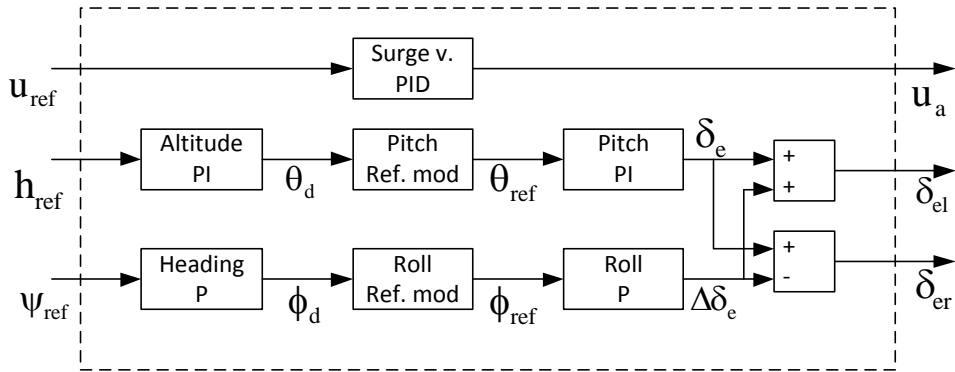


Figure 5.2: The implemented motion controller principle.

## 5.2 Control system

In the simulation environment, all coefficients and states are perfectly known, and advanced controller derivation schemes could fairly be applied to generate superior system responses. This approach has little relevance for the first attempts of automatic flights, as the increased complexity also introduces new sources of errors. It is thus reasonable for the first autopilot-flights to be carried out using as simple controllers as possible.

The following conclusions were drawn in chapter 4:

- Surge velocity control is ensured by applying motor voltage.
- Altitude control is ensured by applying pitch, further by applying elevon deflections.
- Heading control is ensured by applying roll, further by applying differences in elevon deflections.

Thus, the rudders are not used in this particular controller implementation. A modular PID controller approach is implemented as shown in figure 5.2. The controllers are simple PID controllers with saturated outputs (sat PID) and integrators (sat I). The implemented controller gains are simply set by trial and failure, and listed in table 5.2.

Variable	P	I	D	sat I	sat PID
Surge velocity	20	1	-0.02	$\pm 10$	(0, 10)
Altitude	0.1	0.05	-	$\pm 5^\circ$	$\pm 30^\circ$
Pitch	3	0.05	-	$\pm 5^\circ$	$\pm 45^\circ$
Roll	2	-	-	-	$\pm 45^\circ$
Heading	2	-	-	-	$\pm 40^\circ$

Table 5.2: The PID constants used in the controller.



# Chapter 6

## Implementation

The simulator model from chapter 3 is implemented in Matlab/Simulink in the Simulink model file `RecceD6.mdl`. All initialization variables are listed in `RecceD6_init.m`, and may easily be changed here. The outermost Simulink diagram is shown in figure 6.1. Obviously, the Simulink model consist of several subsystems representing actuator dynamics, kinetics, kinematics, gravity, landing gear, thrust and aerodynamics to mention the most prominent ones. The implementation of the Recce D6 block is done according to figure 3.2.

### 6.1 Graphic Interface

A lot of effort has been put into generating graphical interfaces of relevant flight simulation variables to quickly and intuitively grasp and substantiate the resulting effect of actuator deflections, and to easily verify autopilot and guidance systems and so on. Since the simulation is done in Matlab, focus has been on creating Matlab-native interfaces.

#### 6.1.1 3D Visualization

A Matlab function is created to visualize relevant flight states in 3D as

```
i plot3D(fignr,splot3d,p,q,aoa,a_fl,a_fr,a_el,a_er,r_og,ax)
```

and the example script `plot3Dex.m` in the project folder produces the output in figure 6.2. This 3D model is geometrically quite alike the Recce D6, and this plot easily shows the UAV's position and attitude in NED. It also shows the rudder and elevon actuator deflections, and colors them correspondingly to the color-map shown in figure 6.3. The wings are colored according to angle of attack.

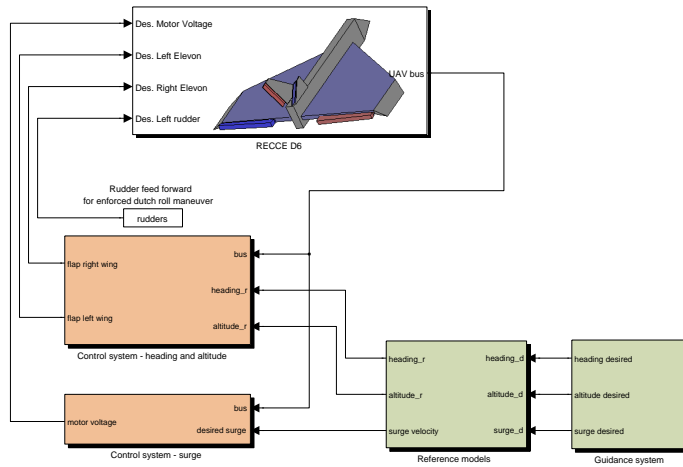


Figure 6.1: The outermost Simulink model of the implemented Recce D6 Motion control system

A demo video showing the UAV coloring may be run by executing the example script *coloringEx.m*. The example output is also stored in the attached project folder as *coloringEx.avi*. For angle of attack illustration in the demo video, imagine an horizontal airstream always hitting the aircraft from the front.

## 6.1.2 Artificial Horizon

A standard flight instrument is an artificial horizon, and in its simplest form shows the current aircraft orientation in NED. To explicitly visualize the UAV's actual, reference and desired orientation an artificial horizon-like plot is implemented into the Matlab function

```
1 aHorizon(fignr, plotHor, 0, 0d, 0r)
```

and the example script *aHorizonEx.m* in the project folder produces the output in figure 6.4. The inputs to the function are triplets of Euler angles in radians respectively, and the figure output the angles in degrees. The orientation input ( $O$ ) to the function is a vector of roll, pitch and yaw angles where the last element represents the current. This way, the pitch and yaw angles history can conveniently be shown in the plot.



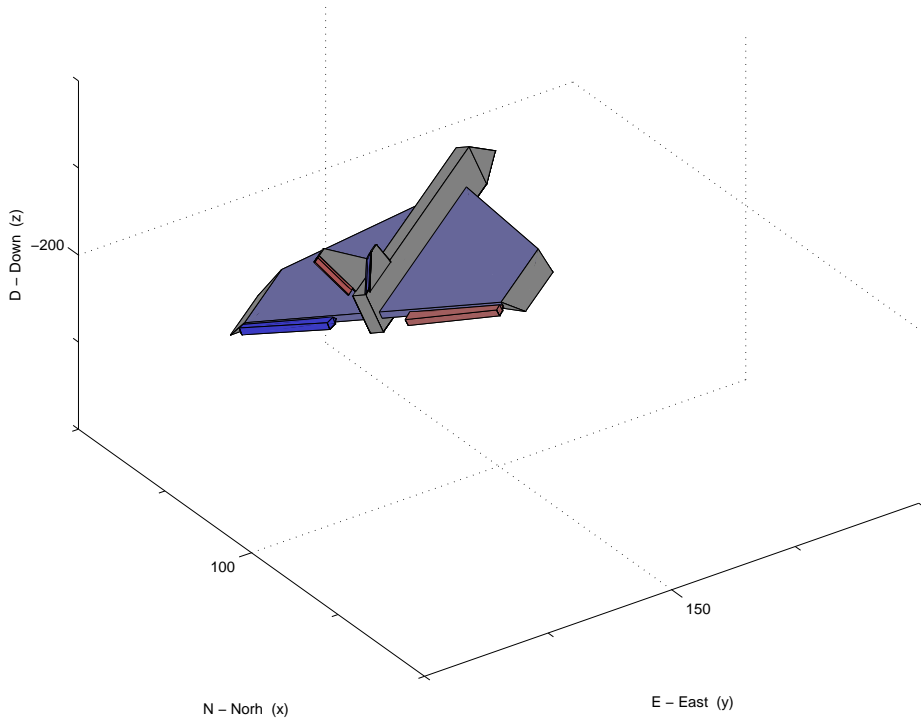


Figure 6.2: An example plot of the 3D interface in Matlab. In addition to coloring, the actuator deflections are actually rotated correctly, although it is not easy to see on this snapshot.

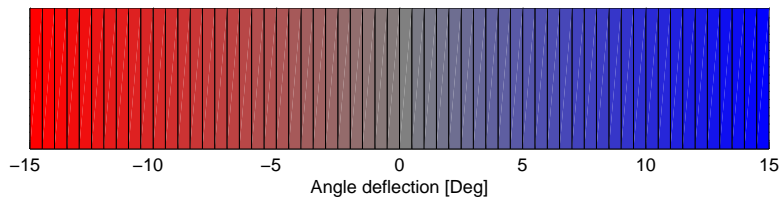


Figure 6.3: The color-map used in the graphical environment. Angles greater than  $15^\circ$  are colored as  $15^\circ$ , and lesser than  $-15^\circ$  are colored as  $-15^\circ$ .

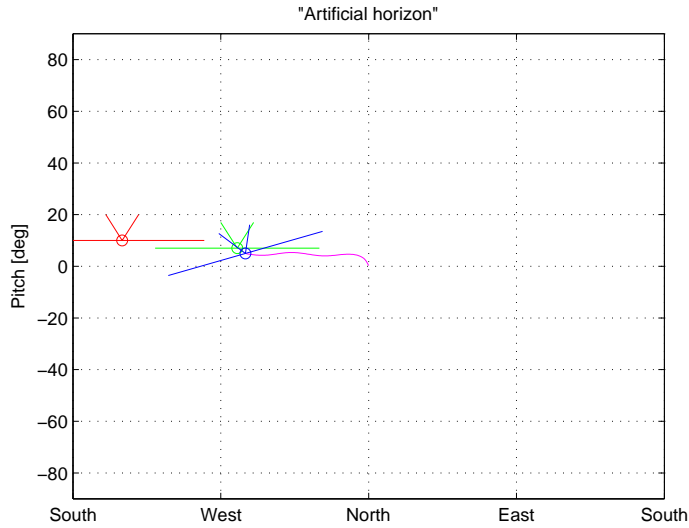


Figure 6.4: An example plot of the artificial horizon interface in Matlab. The red aircraft illustrates the desired orientation, the green illustrates the reference orientation, and the blue illustrates the actual orientation.

### 6.1.3 Compass

Another standard flight instrument is the compass which basically shows heading angle. This information is intuitively embedded into the artificial horizon plot previously discussed, but for aircraft, the wind direction relative to the aircraft heading is also important, and it is hard to imagine a good way of embedding this into the artificial horizon plot. One might imagine that the UAS ground station having a weather station measuring wind direction and speed. A compass plot showing a silhouette of the UAV, wind direction and strength, desired and reference heading is implemented into the Matlab function

```
1 RecceCompass(cheading, desheading, refheading, wind)
```

and the example script *RecceCompassEx.m* in the project folder produces the output in figure 6.5. The center arrow represents wind, where length is proportional to wind strength, and the compass circle equals 7 m/s. The aircraft heading, wind strength and direction is also displayed as text, in addition to current wind strength classification according to the Beaufort scale.

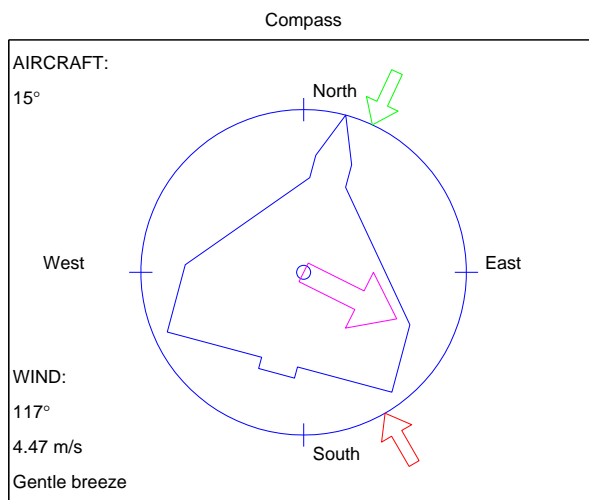


Figure 6.5: Example plot of the compass plot showing actual, reference and desired heading, and wind direction and strength.

## 6.2 Crash Handling

Even if the simulator model from chapter 3 is meant to comprise a widest possible range of relevant flight dynamics, certain extreme situations such as a high speed crash should not be expected to be properly modeled. This is superfluous anyway, there is simply no need to simulate a useless and damaged aircraft. However, the simulation environment should not crash from extreme numerical values.

Recall that the landing gear has a high spring constant. Imagine the situation where the UAV is heading towards the ground at a very high down-velocity. Consider a fixed-step solver (without zero-crossing detection). The last sample before impact, everything is normal and in the next sample the UAV is way “under ground.” This will give an extreme force due to the big wheel displacement, and hence the solver will compute an extreme acceleration upwards throwing the UAV in the air. The simulator-solution to this is to simply ignore these situations by saturating the landing gear-induced forces. Thus, a crash will not destroy the aircraft in the simulation environment, nor throw it in the air, but rather yield a controlled restoring force.

Another outcome is landings upside-down. The landing gear will produce a restoring force, but the fuselage will appear as buried with the landing gear “floating” on the surface. No fix for this is currently implemented. In general, no deviations from a first principle approach is used to influence the behavior of the UAV model implementation.



# Chapter 7

## Simulated Flight

Attached to this thesis, there is a simulated flight video named `CaseStudys.avi`. Embedded into this single flight is a set of scenarios meant illustrate all the dynamics of the simulation model, and the necessary flight maneuvers needed to maintain a certain heading, altitude and surge speed for a fully functional Recce D6 - both subject to no wind, and gusty winds. All case studies discussed in this chapter have an adjacent video-sequence in the same order of appearance. Hence, the reader of this chapter should open the video and follow the simulated flight in parallel with reading the case studies. The video frame comprises a lot of information, and it is very hard to digest it all at real-time playback. A suggestion is to run the video in a video playback software able to play the video in slow motion as necessary<sup>1</sup>.

All details regarding the creation of this video is listed in appendix A. Most comments regarding the handling of the aircraft are based on Nordan Aviation Training Systems (2005) and Barnard and Philpott (2003)

The video is organized as follows:

- Case I: Landing gear dynamics (Time: 0:00)
- Case II: Take-off (Time: 0:10)
- Case III: Climbing (Time: 0:35)
- Case IV: Horizontal steady flight (Time: 0:45)
- Case V: Enforced Dutch roll (Time: 1:00)
- Case VI: Banked turn (Time: 1:20)
- Case VII: Descent (Time: 1:45)

---

<sup>1</sup>An example is the (free) open-source VLC media player available from <http://www.videolan.org/vlc>.

- Case VIII: Wind (Time: 2:05)
- Case IX: Touch-and-go, crosswind (Time: 3:00)
- Case X: Loss of power - stall (Time: 3:50)
- Case XI: Propulsion system (Time: 4:30)
- Case XII: Touch-and-go, upwind (Time: 5:20)
- Case XIII: Landing at low airspeed (Time: 6:10)
- (Video end at 06:40)

A video frame snapshot with shorts comments is shown in figure 7.1. Details regarding the creation of this video is given in appendix A. The layout is organized such that the four inputs to the UAV is included in the three displays under the 3D plot:

- The leftmost graph shows actuator deflections for the left wing and stabilizer, and the rightmost graph shows a corresponding graph for the right side.
- The middle graph shows the powerplant status. The battery voltage and charge is displayed as text, while the total powerplant power consumption is displayed as the graph. Obviously, the motor represent nearly all energy consumption from the powerplant. In reality, the input to the motor is the motor voltage, but total power consumption is proportional to this. The powerplant status is therefore chosen to represent the in input to save precious space in the video frame and to keep it clearly set out.

The column of three displays on the right hand side of the 3D plot includes the control objectives:

- The upper plot shows the compass plot discussed in section 6.1.3.
- The middle plot shows altitude. The green arrow represent reference altitude.
- The lower plot shows the velocity components of BODY, and the current speed. The blue plot, surge speed, is recognized as one of the three primary control objectives. The blue arrow is the reference surge speed. The black arrow is the actual surge air velocity.

The column of three displays to the right contains other relevant flight data:

- The upper plot shows the orientation as Euler angles, where the cyan color (yaw) is recognized as the heading angle. Note that this plot really comprises the same orientation information as the blue airplane in the artificial horizon plot.

- The middle plot shows the N and E positions over the last 20 seconds.
- One might imagine the aircraft being equipped with sensors so measure angle of attack and sideslip angle. The upper plot therefore shows the angle of attack and the sideslip angle of the fuselage where such a sensor might be installed. Keep in mind that these angles are coincident with the corresponding angles for both the main wings. (Unless the craft have a significant rolling rate or yawing rate at high angle of attack.)

A few additional comments comments to the figure:

- The 3D plot discussed in section 6.1.1 is augmented with a 3D arrow showing the current wind direction and strength. If the wind strength is  $\approx 0$  m/s the arrow is not displayed.
- The current time and corresponding case study are included as text in the upper left corner. The simulation starts at time 00:00, displayed in the format minutes:seconds.
- The artificial horizon plot also displays the last 20 seconds actual pitch and heading angle history as the magenta graph.

The case composition is meant to illustrate model properties, and verify some well known flight related issues and maneuvers from real life in the simulator. The first few case studies are mainly meant to illustrate that ordinary flight under perfect conditions is “easy.” The main idea of a simulation environment is to foresee potential problems, and detect problems that otherwise may have been overlooked. So to justify the main purpose of the simulator, the UAV is exposed to gusty wind from case 7.8, and a motor failure in case 7.10.

## 7.1 Case Study I: Landing Gear Dynamics

Purpose of case: Illustrate landing gear dynamics.

	<b>Video time</b>	<b>Surge speed</b>	<b>Altitude</b>	<b>Heading</b>
	<b>Start: 0:00</b>	<i>[m/s]</i>	<i>[m]</i>	<i>[deg]</i>
<i>Previous</i>		-	-	-
New set-point	0:00	0	0	0
<b>Case end</b>	<b>0:10</b>			
<b>Duration</b>	<b>10s</b>			

The simulation is started with the UAV dropped from an altitude of 0.5 meters, slightly rotated with 6 degrees of roll and -3 degrees of pitch initially. Since the landing gear dynamics is highly damped, it quickly converges to an equilibrium

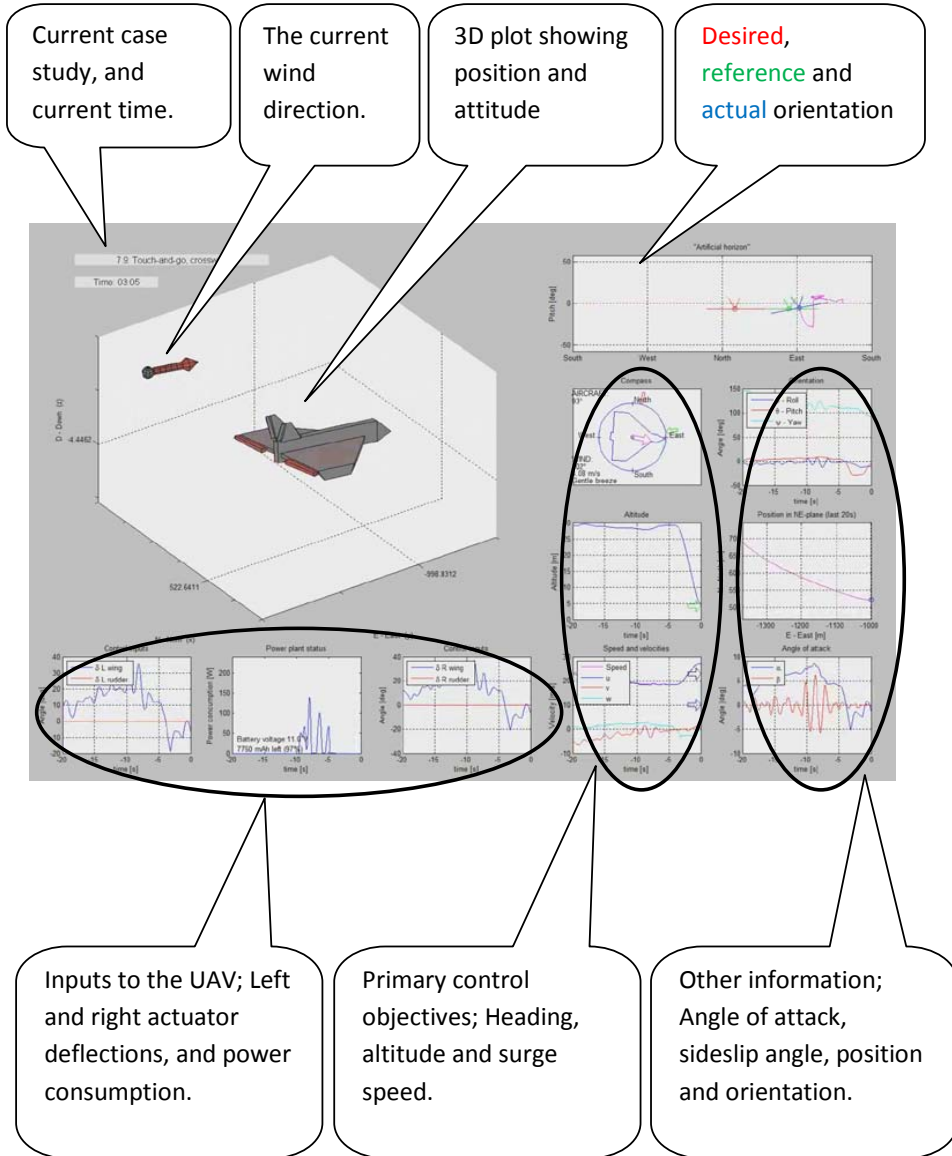


Figure 7.1: An example snapshot of the simulated flight video window.



without significant oscillations. Notice that the UAV converges to a pitch equilibrium at about  $\approx -1^\circ$ . This is because the center of mass for this simulation is exactly between the nose wheel and the rear wheels in the longitudinal direction, giving slightly more weight to the nose wheel than for the two rear wheels. Hence, the nose wheel is slightly more displaced.

Since the UAV has no surge velocity at start, the only airstream is resulting from the falling motion. This results in a quite “crazy” angle of attack coloring of the wings during the first few seconds. It has not been considered as important to solve this particular issue which is obviously only related to the visualization at near zero surge speed.

Notice that both the right and left elevon deflections are increasing. This is because the altitude reference is 0 while the UAV really is at an altitude of 0.07 meters. Also the simulation is started with a little orientation error that results in a desired roll. Hence, the altitude controller wants the UAV to pitch down, and the heading controller wants the UAV to roll. Remember that the UAV is in “flight mode” during the entire simulation. With the UAV standing almost still on the ground, this is a good demo of issues related to non-nominal flight conditions that the ordinary flight-mode controller do not comprise any fix for. If this turns out to be a problem in practice, a separate ground mode controller should be considered.

## 7.2 Case Study II: Take-Off

Purpose of case: Take-off maneuver

	<b>Video time</b>	<b>Surge speed</b>	<b>Altitude</b>	<b>Heading</b>
	<b>Start: 0:10</b>	$[m/s]$	$[m]$	$[deg]$
<i>Previous</i>		$0$	$0$	$0$
New set-point	0:10	<b>20 (+20)</b>	0	0
New set-point	0:15	20	<b>30 (+30)</b>	0
<b>Case end</b>	<b>0:35</b>			
<b>Duration</b>	<b>25s</b>			

Now voltage is applied to the motor giving thrust. Notice that the power consumption from the powerplant is rapidly increased, and at the same time the nose pitches slightly further down. This is because the applied thrust force with line of action over the center of mass in the longitudinal direction gives a negative pitching moment, further displacing the nose wheel. As the UAV accelerates the wings produce more lift, resulting in a positive pitching moment. Since the reference altitude still is zero, the altitude controller outputs elevon deflections to pitch down. The result is small pitch oscillations. Also notice how the increased velocity gives the heading controller better controllability of the motion. The roll requested to turn the aircraft towards the reference heading only results in increased landing gear friction which turns the aircraft on the ground before take-off.

At 00:15, the surge velocity is close to 20 m/s, and the commanded altitude is set to 30 meters. Both elevon deflections are increased, and the nose pitches up. Just as the nose wheel is in the air, notice how the angle of attack is changed, since the airstream now suddenly hits the UAV from below. ( $w$  is increased.) The increased angle of attack gives more lift, and the UAV takes off. When the rear wheels are airborne, the moment induced from the landing gear suddenly disappears. The UAV now converges to the new resulting equilibrium established by the aerodynamic and thrust forces. First now, the UAV can be considered to be operating under nominal flight conditions, and the flight mode controller should be expected to fulfill the control objectives of maintaining altitude, surge speed and heading.

One experience deduced from this case is that the flight mode controller is not very well suited for take off, and hence a separate take-off controller is motivated.

### 7.3 Case Study III: Climbing

Purpose of case: Climbing maneuver

	<b>Video time</b>	<b>Surge speed</b>	<b>Altitude</b>	<b>Heading</b>
	<b>Start: 0:35</b>	$[m/s]$	$[m]$	$[deg]$
<i>Previous</i>		20	30	0
New set-point	0:35	20	<b>100 (+70)</b>	0
<b>Case end</b>	<b>0:45</b>			
<b>Duration</b>	<b>10s</b>			

Climbing is the action of increasing altitude. This is simply done by pitching up and maintaining the same surge speed by applying more thrust. Notice how the power consumption is before and during the climb.

### 7.4 Case Study IV: Horizontal Steady Flight

Purpose of case: Steady flight maneuver

	<b>Video time</b>	<b>Surge speed</b>	<b>Altitude</b>	<b>Heading</b>
	<b>Start: 0:45</b>	$[m/s]$	$[m]$	$[deg]$
<i>Previous</i>		20	100	0
New set-point	0:45	<b>18 (-2)</b>	100	0
<b>Case end</b>	<b>1:00</b>			
<b>Duration</b>	<b>15s</b>			

Steady flight is flight maintaining a certain equilibrium point. Furthermore, horizontal steady flight is flying at a constant altitude. By applying a constant altitude,

surge speed and heading, notice how all inputs and outputs converges to their respective equilibrium's after some time.

The UAV is now cruising at 18 m/s or equivalently 65 km/h. Notice how the velocity components other than surge is almost zero. (Almost zero angle of attack.) Thus, this particular air surge speed in horizontal steady flight minimizes drag. This airspeed in surge should therefore be the desired airspeed in surge for missions where maximum endurance is requested.

Since there is no wind, notice how the air velocity in surge perfectly coincides with the surge velocity of the UAV -as expected.

## 7.5 Case Study V: Dutch Roll

Purpose of case: Dutch roll maneuver

	<b>Video time</b>	<b>Surge speed</b>	<b>Altitude</b>	<b>Heading</b>
	<b>Start: 1:00</b>	[m/s]	[m]	[deg]
<i>Previous</i>		18	100	0
New set-point	-	-	-	-
<b>Case end</b>	<b>1:20</b>			
<b>Duration</b>	<b>20s</b>			

Dutch roll is a classification of an lateral instability phenomena. It is an oscillation in roll, sideslip and yaw; A passenger looking out the window of an aircraft in Dutch roll mode will see the wingtip drawing an elliptic pattern relative to the horizon. Imagine a lateral wind gust increasing the lateral drag force from the stabilizers at the rear of an aircraft. This sudden sideslip will perturb the yaw motion of the aircraft, which in turn produces roll because of distinct air velocity over the two wings. The yawing motion will turn the aircraft such that it starts sideslipping the other way. This goes back and fourth until the perturbed yaw-motion is damped out.

The Dutch roll is typically a consequence of a lateral perturbation from wind. However, a similar response can be provoked by applying rapid steps on the rudders to experimentally grasp the lateral stability properties of the craft. This is what is done in this case; Steps are fed forward to the rudders. At this airspeed, the UAV shows little oscillation and recovers to zero sideslip efficiently. Thus, lateral stability for this particular simulated UAV is verified by simulation. Also notice that the control system immediately rolls to compensate for the heading error.

Since Dutch roll is a lateral stability phenomena, it is worth mentioning that a perturbation in roll also will introduce another lateral stability phenomena called spiral dive. For aircraft not naturally stable in roll this might lead to a dangerous graveyard spiral if proper correcting action is not carried out. Recce D6 is expected to be naturally stable in roll and, being unmanned, the autopilot should automatically compensate for such perturbations.

## 7.6 Case Study VI: Banked Turn

Purpose of case: Banked turn maneuver

	<b>Video time</b>	<b>Surge speed</b>	<b>Altitude</b>	<b>Heading</b>
	<b>Start: 1:20</b>	$[m/s]$	$[m]$	$[deg]$
<i>Previous</i>		18	100	0
New set-point	1:20	18	100	<b>-105 (-105)</b>
<b>Case end</b>	<b>1:45</b>			
<b>Duration</b>	<b>25s</b>			

Change in heading is effectively carried out by rolling the aircraft towards the desired direction of heading. This is often called a banked turn.

## 7.7 Case Study VII: Descent

Purpose of case: Descent maneuver

	<b>Video time</b>	<b>Surge speed</b>	<b>Altitude</b>	<b>Heading</b>
	<b>Start: 1:45</b>	$[m/s]$	$[m]$	$[deg]$
<i>Previous</i>		18	100	-105
New set-point	1:45	18	<b>30 (-70)</b>	-105
<b>Case end</b>	<b>2:05</b>			
<b>Duration</b>	<b>20s</b>			

Descent is the action of reducing altitude. This is simply done by pitching down. Notice how the surge velocity is rapidly increased even if the reference surge velocity is not changed, and no thrust is applied from the motor. The aircraft is not equipped with air-brakes, and the motor is not configured to be able to rotate in the opposite direction. The reference surge velocity is recovered after the descent is over.

## 7.8 Case Study VIII: Wind

Purpose of case: Illustrate the effect of gusty wind on the UAV, upwind, crosswind and tailwind.

	<b>Video time</b>	<b>Surge speed</b>	<b>Altitude</b>	<b>Heading</b>
	<b>Start: 2:05</b>	$[m/s]$	$[m]$	$[deg]$
<i>Previous</i>		18	30	-105
New set-point	2:05	18	30	<b>165 (+270)</b>
<b>Case end</b>	<b>3:00</b>			
<b>Duration</b>	<b>55s</b>			

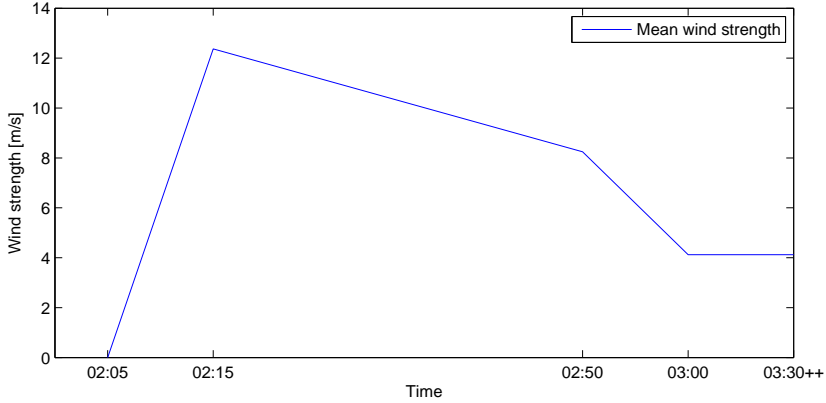


Figure 7.2: The average wind strength with direction towards  $75^\circ$ .

At time 2:05, the aircraft is exposed to wind generated by the simple wind model

$$\ddot{\xi}_i = k_{\delta w_i} \delta(t) - 2\zeta_i \omega_{0i} \dot{\xi} - \xi \omega_{0i}^2 \quad (7.1)$$

$$w_i = \xi_i + \bar{w}_i \quad (7.2)$$

$$i \in \{N, E\} \quad (7.3)$$

where  $w_i$  is the wind in  $[m/s]$  along N and E respectively. (The wind is modeled to be entirely horizontal.)  $\delta(t)$  is unit white noise,  $\bar{w}_i$  is the wind middle value, and  $\omega_0, \zeta, k_{\delta w}$  are design constants set to

$$k_{\delta w N} = 3 \quad (7.4)$$

$$\omega_{0N} = \pi \quad (7.5)$$

$$\zeta_N = 0.3 \quad (7.6)$$

$$\bar{w}_N = -1 \quad (7.7)$$

$$k_{\delta w E} = 0.5 \quad (7.8)$$

$$\omega_{0E} = \pi \quad (7.9)$$

$$\zeta_E = 0.3 \quad (7.10)$$

$$\bar{w}_E = 4 \quad (7.11)$$

The  $k_{\delta_i}$  are dynamically set in the simulation, but the average wind direction is (towards)  $75^\circ$ . The average wind strength in this direction in the simulation is shown in figure 7.2. As seen from this figure, and from the video, the wind strength is rapidly increased towards quite much wind, and then reduced to an average of about 4.1 m/s at 03:00. This average wind strength is unchanged throughout the video.

First, notice that the relative wind velocity and surge velocity no longer coincide. Remember that it is surge speed that is fed back to the surge speed controller,

and not airspeed in surge. The simulated heading is initially towards the wind direction, so the surge speed controller has to apply thrust in order to maintain a certain surge speed relative to NED, similar to a boat in a river going towards the current.

As seen from the model and the video, the wind is quite gusty. Think of this as turbulence. Notice how all states suddenly goes from being smooth, to heavily perturbed. The desired heading is now set to be along the wind direction. The reason for setting the heading reference model to somewhat slow is so that the transition between upwind and along the wind direction is done slowly here. As the UAV turns, the sway speed reaches 10 m/s when the heading angle is  $-175^\circ$  which seem to coincide with the average wind speed. At this heading, the wind hits the aircraft from the side so this is an expected observation.

As the UAV turns further, the wind starts hitting from behind. (Tailwind) Now make the interesting observation of the air surge speed being significantly lower than the UAV surge speed. The aerodynamic forces and moments are quadratic in terms of the airspeed, not the actual surge speed, so this highly affects the maneuverability and stability properties of the UAV. Notice in particular that the low airspeed now gives little damping in yaw, so the UAV actually goes into a Dutch roll mode with significantly more prominent oscillations than in the enforced Dutch roll mode from section 7.5, with the controller much less able to counteract it. As a matter of fact, notice that the aircraft is no longer able to maintain altitude, and descents while the angle of attack oscillates about the stall angle of the main wings. Luckily, the wind strength is reduced from fresh breeze to gentle breeze at this very critical moment, which in turn provide sufficient airspeed in surge to restore maneuverability. Without this lucky turn of events, a crash or ditch (emergency landing on water) would have been inevitable -not because of system failure, but rather because of a system weakness or clumsy or ignorant reference in surge speed.

This case illustrates that the airspeed highly affects the maneuverability and stability properties of the UAV, and that surge speed not necessary provide sufficient feedback to the control system, or guarantee that the UAV stays within its flight envelope. Alternatively, the actual airspeed can be implemented as the reference control objective directly.

## 7.9 Case Study IX: Touch-and-go, Crosswind

Purpose of case: Touch-and-go maneuver

	<b>Video time</b>	<b>Surge speed</b>	<b>Altitude</b>	<b>Heading</b>
	<b>Start: 3:00</b>	$[m/s]$	$[m]$	$[deg]$
<i>Previous</i>		18	30	165
New set-point	3:00	<b>15</b> (-3)	<b>10</b> (-20)	<b>15</b> (+150)
New set-point	3:15	<b>0</b> (-15)	<b>0</b> (-10)	15
New set-point	3:27	<b>20</b> (+20)	0	15
New set-point	3:30	20	<b>100</b> (+100)	15
<b>Case end</b>	<b>3:50</b>			
<b>Duration</b>	<b>50s</b>			

Touch-and-go is a maneuver where the air vehicle performs a controlled landing, but where it takes off again instead of coming to a full stop. This is a common pilot training maneuver. The current wind strength is an average of about 4.1 m/s. The problem of cross wind landings are that the lateral sway velocity comes to a hard stop at touchdown because of high wheel friction. Observe how the strong lateral wind forces the aircraft to turn after touching down, while the heading controller tries to bank the plane in the opposite direction to restore reference heading.

## 7.10 Case Study X: Stall

Purpose of case: Stall

	<b>Video time</b>	<b>Surge speed</b>	<b>Altitude</b>	<b>Heading</b>
	<b>Start: 3:50</b>	$[m/s]$	$[m]$	$[deg]$
<i>Previous</i>		20	100	15
New set-point	3:50	<b>0</b> (-20)	<b>120</b> (+20)	15
New set-point	4:10	<b>18</b> (+18)	<b>100</b> (-100)	15
<b>Case end</b>	<b>4:30</b>			
<b>Duration</b>	<b>40s</b>			

Recall that the stall angle for the main wings were set to  $13^\circ$ . The desired altitude is set to 120 meters, desiring a climb of 20 meters, but the desired surge velocity is set to zero. This is equivalent to a sudden motor failure due to the current propulsion system implementation. Notice how the powerplant produces no thrust at all. The altitude controller outputs a pitch reference, but as the airspeed is reduced the nose is forced to pitch down. The air velocity is then increased from the loss of altitude, and eventually the airspeed gets sufficiently high for the pitch controller to regain maneuverability. Without thrust, the UAV becomes a glider, eventually doomed to crash without regaining thrust. Observe that the angle of attack is oscillating about the stall angle, which is recalled as the airstream angle relative to the wing where lift is maximized.

This case motivates the implementation of some hardware watchdog looking for system failure, and a corresponding emergency response. This response may be a controlled emergency descent, parachute launching or automatic error recovery.

## 7.11 Case Study XI: Propulsion System

Purpose of case: Illustrate powerplant properties

	<b>Video time</b>	<b>Surge speed</b>	<b>Altitude</b>	<b>Heading</b>
	<b>Start: 4:30</b>	$[m/s]$	$[m]$	$[deg]$
<i>Previous</i>		20	100	15
New set-point	4:30	<b>25 (+5)</b>	100	<b>-45 (-60)</b>
New set-point	4:35	<b>20 (-5)</b>	100	-45
New set-point	4:40	<b>25 (+5)</b>	100	-45
New set-point	4:45	<b>20 (-5)</b>	100	-45
New set-point	4:50	<b>25 (+5)</b>	100	-45
New set-point	4:55	<b>18 (-7)</b>	100	-45
<b>Case end</b>	<b>5:20</b>			
<b>Duration</b>	<b>50s</b>			

Now the UAV has been flying for about four and a half minute. Initially, the battery capacity was at 8000 mAh, but now only 7600 mAh (95%) remain. As the reference surge velocity increases and decreases, notice how the battery voltage is declined as the power consumption is high, and vice verca. This voltage drop is due to the inner resistance of the battery, and although the motor only consumes 200W, the total power consumption is close to 225W. Also notice that when there are no power consumption, the battery voltage has decreased from 11.1 to 11.0 volt. This is a consequence of lower battery capacity.

## 7.12 Case Study XII: Touch-and-go, Upwind

Purpose of case: Landing maneuver

	<b>Video time</b>	<b>Surge speed</b>	<b>Altitude</b>	<b>Heading</b>
	<b>Start: 5:20</b>	$[m/s]$	$[m]$	$[deg]$
<i>Previous</i>		18	100	-45
New set-point	5:20	18	30	<b>-65 (-20)</b>
New set-point	5:30	15	10	-65
New set-point	5:45	0	0	-65
New set-point	6:00	15	10	-65
<b>Case end</b>	<b>6:10</b>			
<b>Duration</b>	<b>40s</b>			

The touch-and-go maneuver as in case 7.9. is repeated, only this time upwind. The UAV touches down at higher surge airspeed. Thus, the wings produce more lift at the moment of touchdown, and the landing gear spring results in a somewhat bumpy landing. Never the less, notice that maintaining the reference heading is much better ensured at upwind landings that crosswind.



## 7.13 Case Study XIII: Landing at Low Airspeed

Purpose of case: Landing at low airspeed

	<b>Video time</b>	<b>Surge speed</b>	<b>Altitude</b>	<b>Heading</b>
	<b>Start: 6:10</b>	$[m/s]$	$[m]$	$[deg]$
<i>Previous</i>		15	10	-45
New set-point	6:10	<b>11</b> (-4)	10	<b>-65</b> (-20)
New set-point	6:20	<b>8</b> (-3)	2	-65
New set-point	6:25	0	0	-65
<b>Case end</b>	<b>6:40</b>			
<b>Duration</b>	<b>30s</b>			

It is clear that landings in as little surge velocity as possible is desired to reduce the time and distance needed to come to a full stop. In this case, the surge speed is reduced to a minimum by utilizing information about the wind to set the surge velocity as low as possible. Notice how the lower airspeed results in an increased angle of attack right before touching down. Just as it touches down, the angle of attack is suddenly changed from six degrees to zero. The wings now suddenly produce much less lift at touchdown, and this results in a smoother landing. It should be kept in mind though, that the UAV becomes significantly more vulnerable to perturbations from the wind. Also remember the conclusion of reduced maneuverability at low airspeeds from case 7.8. Hence, a compromise between a minimum airspeed and surge speed has to be made.

The landing maneuver is perhaps the most challenging of all UAV maneuvers. For landings on airstrips, the landings are clearly restricted to be along a certain direction, and wind direction parallel to the strip can of course not be guaranteed. This introduces a requirement of controlled landings in crosswind. A sophisticated landing maneuver is to perform the final approach with heading offset to align the course of the aircraft with the runway direction. (The course direction is the direction of the resulting speed of the aircraft.) Then, right before touching down, a controlled sideslip can be achieved by provoking the Dutch roll mode of the aircraft to align the heading of the craft to the runway heading. The complexity of the landing maneuver suggests that the landings of the UAV should be performed on a field rather than on a runway, where upwind landings always are possible.

To reduce the area needed for landings, a separate breaking procedure by rotating the propellers in the opposite direction can also be considered. However, the possible consequence of reversing the propeller direction in air before actually touching down is a hard impact potentially wrecking the vehicle, so caution should be carried out in such an implementation.



# Chapter 8

## Conclusions

The dynamics of the aircraft derived in chapter 3 is based on a first principle approach comprising aerodynamics, thrust, landing gear and gravity forces. State-of-the-art aircraft systems such as modern fighters and passenger transport vehicles are quite complex instrumentation systems compared to the Recce D6 UAV. This makes the on-board actuation system modeling of Recce D6 quite simple, and limited to three servo motors and one propulsion system. The model is successfully augmented with these systems.

The implementation of the simulation model in Matlab /Simulink is properly tuned to match the most prominent properties of Recce D6 as listed in the specifications. As shown in the simulated flight of chapter 7, the simulator model seem to produce a reasonable response and capture well known aircraft handling issues. However, the model has not been verified by neither experienced pilots nor by comparing logged flight data to the simulator response, and thus the model should not be considered to be accurate.

The implemented motion control system is simple, but provide a useful and relevant basis for simple UAV maneuvers.

The visualization implementation in Matlab is proven very useful to substantiate the simulator model and motion control system response, and may for instance be used to generate videos such as the one attached to this thesis.

### 8.1 Further Work

The model should be tuned to match the dynamics of Recce D6 more accurate than the current simulator can be assumed to be.

The simulation model and visualization is a suitable basis for hardware-in-the-loop testing. This may be achieved by using Simulink Coder<sup>1</sup> to compile the simulator

---

<sup>1</sup>Trademark of The MathWorks, Inc - formerly known as Real-Time Workshop.

logic and visualization to run real-time under Matlab or under a native operative system environment. The on board UAV computer will be a Beagle board<sup>2</sup> running a Linux based distribution named Ångström. Considerations regarding the constellation of on board instrumentation and this operating system is addressed in Skøien (2011), while hardware-in-the-loop implementation considerations interfacing Matlab with external hardware (such as the Beagle board) is considered in Stern (2011).

The model can be augmented with models for hardware such as gyro, magnetometer, accelerometer and GPS. The navigation block discussed in chapter 5 can then be implemented and tested, and further used as reference to the control system.

The model could be used to test more comprehensive guidance schemes and controllers, and different autopilot modes can be tried out.

Last, a set of scenarios can be defined to test robustness of the system such as motor failure, servo motor malfunctions, rapid changes in aerodynamic properties, sudden weight distribution changes or operative system failure.

---

<sup>2</sup>BeagleBoard.org, an all volunteer activity started-up by a collection of passionate individuals.

## Bibliography

- Allerton, D., 2009. *Principles of Flight Simulation*. Chichester: John Wiley & Sons, Ltd
- Barnard, R. H. and Philpott, D.R., 2004. *Aircraft Flight*. Harlow: Pearson Education Limited
- Egeland, O and Gravdahl, J.T., 2003. *Modeling and Simulation for Automatic Control*. Trondheim: Marine Cybernetics AS
- Hoerner, S. F., 1958. *Fluid-dynamic drag*. New Jersey: (Published by the author.)
- Hoerner, S. I. and Borst, H.V., 1985. *Fluid-dynamic lift*. Vancouver: Mrs. Liselotte A. Hoerner
- Fossen, T. I., 2011a. *Mathematical models for control of aircraft and satellites*. Trondheim: Department of Engineering Cybernetics, NTNU.
- Fossen, T. I., 2011b. *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons Ltd.
- Graebel, W. P., 2007. *Advanced Fluid Mechanics*. Elsevier Inc.
- Houghton E.L., Carpenter, P.W., 2003. *Aerodynamics for Engineering Students*. Butterworth-Heinemann.
- McLean, D., 1990. *Automatic Flight Control Systems*. Hemel Hempstead: Prentice Hall International (UK) Ltd.
- Nordian Aviation Training Systems, 2005. *Principles of flight*. London: London metropolitan university
- Skjøien, K., 2011. *A Modular Software and Hardware Framework with Application to Unmanned Autonomous Systems*. M.Sc. Norwegian University of Science and Technology.
- SNAME (The Society of Naval Architects and Marine Engineers), 1950. Nomenclature for Treating the Motion of a Submerged Body Through a Fluid. In: *Technical and Research Bulletin No. 1-5*. New York: The Society of Naval Architects and Marine Engineers

Stern, C., 2011. *HIL Framework for UAV Testing*. M.Sc. Norwegian University of Science and Technology.

Vik, B., 2000. *Nonlinear Design and Analysis of Integrated GPS and Inertial Navigation Systems*. Ph. D. Norwegian University of Science and Technology.

White, F.M., 2008. *Fluid Mechanics*. 6th ed. New York: McGraw-Hill.

Yechout, T.R., Morris, S.L., Bossert D.E. and Hallgren, W.F., 2003. *Introduction to aircraft flight mechanics*. Reston: American Institute of Aeronautics and Astronautics, Inc.

Yuan, Joseph S.-C., 1988. *Closed-Loop Manipulator Control Using Quaternion Feedback*. IEEE Journal of robotics and automation, vol. 4, no.4, August 1988.

# Appendix A

## Case Study video

The attached video showing a simulated flight of Recce D6 is generated in Matlab/Simulink, (trademarks of The MathWorks, Inc) To facilitate the creation of similar videos, this appendix shows how this specific video has been produced.

The hardware and software used throughout this thesis, and to generate the video is

- HP Z200 Workstation
  - Intel Core i5 CPU, 3.20GHz, 2 Cores
  - 8 GB RAM
  - x64
- Microsoft Windows 7 Enterprise
  - Version 6.1.7601 Service Pack 1 Build 7601
  - x64
- Matlab
  - Version 7.10.0.499 (R2010a)
  - 64-bit
  - Simulink Version 7.5 (R2010a)

All relevant Matlab functions are included in the project folder, so no special Matlab toolboxes are required to perform the simulation or generate the video. To perform simulation and generate the same video, complete the following steps: (Keep in mind that the wind is randomly generated, so the response might vary slightly since the wind is not identical for each simulation.)

1. All simulation inputs and variables are defined in the Matlab script `RecceD6_init.m`. To properly initialize the simulation, run this script. Matlab will prompt a question dialog box asking to start Simulink simulation. By clicking Yes, Matlab will run the `RecceD6.mdl` simulation. The current solver for this model is forward Euler; `ode1` (Euler) with 2.1 ms step size. The simulation takes about 15 minutes on the PC it was developed at.
2. When the simulation is finished, all variables needed to generate the video is in the Matlab workspace. To generate the video, open the `MakeVideo.m` script. This script produces all the figures that represents each video frame, captures them and makes an array of video-frames. This script require some explanation:
  - (a) First, the relative path to where the output AVI file should be specified by setting the `avipath` variable. The file name should not be included.
  - (b) The video is generated using the built-in Matlab-function `avifile`. One shortcoming of the development platform is that actually no of the compression parameters for `avifile` are available. Thus, no compression is applied during video generation. Hence, the uncompressed output file size might grow to enormous proportions. The next inconvenience is that `avifile` can not write files larger than 2GB. The workaround to this has been to create many `.avi` files, and limit the number of frames in each file such that the output file does not exceed 2GM in size. This is done by specifying the `framesprvideo` variable on top of the script.
  - (c) Next, specify the desired number of frames per second in the output video by setting the `plotInterval` variable.
  - (d) All the video frames has to be exactly equal in size. All the different plot commands throughout the script might alter the overall figure size depending on the plotting content. The figure size is therefore set manually right before the figure frame is captured and put into the video frame array. Currently, the video size is set relative to the screen resolution of the development machine. This video size is set at the end of the current-plot iteration loop.
  - (e) An extensive workload is carried out by Matlab for each frame, so generating the video frames may take as much as four hours to complete. The output of the case study video is 28 avi-files, with a total size of 56,1GB. The output filename is:  
`Recce_[year]_[month]_[date]_kl_[hour]_[minute]_vnr_[file number].avi`  
 Hence, by sorting the output files alphabetically, the videos are sorted in a chronological order.
3. The output file sizes are clearly very unpractical. Therefore, the files are compressed using the free software Any Video Converter. Video options are Codec: `xvid`, Frame size: `1920x1080`, Bitrate: `2000`. After being compressed, the videos are 109 MB in size.



4. The 28 compressed files are joined to one .avi file by using the free software AVI Joiner. This is the final video.

Keep in mind that the issues related to the big .avi files may be specific to the development platform. Other platforms might accept video compression in Matlab directly. Also other strategies for video creation in Matlab exist, such as “mpg-write” (available from Matlab Central). Neither this did work on the development machine.

All 3rd party software discussed here are included in the digital attachment of this thesis.



# Appendix B

## Matlab code

The Matlab implementation consist of more that 2000 lines of Matlab code, and listing all code in the report would take 34 pages. Further, printing all model diagrams from the Simulink implementation takes 31 pages. (Not all are distinct; the same block for landing gear are used three times for instance.) Printing 65 pages of Matlab code and Simulink models would be a disgrace to the environment, and yield no benefit. The initialization file for the simulation, the make-movie script, the 3d plot function, the artificial horizon function and the compass function are included.

Focus has been to write readable code for a Matlab script editor, not subject to a requirement of printer-friendliness. Thus, the code-listing here is subject to an intentionally disregarded text-wrapping issue at print-out.

### B.1 RecceD6\_init.m (Matlab script)

```
1 % Author: Kristoffer Dønnestad, 17.06.2011
2 % Attachment to M.Sc. thesis
3 % Unmanned Vehicle Laboratory
4 % Department of Engineering Cybernetics
5 % Norwegian University of Science and Technology
6 %
7
8 clc; clear all; close all;
9
10 % DEFINE SIMULATION CONSTANTS
11 fignr = 1; % The figure number to plot into
12 simtime = 405; % Number of seconds to simulate
13 StepSize = 1/(24*20); % Solver step length.
14
15 % DEFINE SIMULATION INPUTS
16 caseindex = [ 1 2 2 3 4 5 6 7 8 8
17 8 9 9 9 9 10 10 11 11 11 11 11 11 12 12 12 12 13
18 13 13 13];
19 ref.time = [ 0 10 15 35 45 60 80 105 125 135
20 170 180 195 207 210 230 250 270 275 280 285 290 295 320 330 345 360
21 370 380 393 simtime];
```



```

69 | g = 9.81; % The acceleration of
    | gravity
70 | rho = 1.2928; % Density of air
71 |
72 |
73 | %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
74 | % FLAP DYNAMICS %
75 | %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
76 |
77 | leftServo.omega_max = 45*pi/(.1*180); % Maximum rotation velocity [rad/s
    | ]
78 | leftServo.maxDeflection = 45*pi/180; % Max deflection
79 | leftServo.minDeflection = -45*pi/180; % Min deflection
80 | leftServo.Kp = 10; %
81 |
82 |
83 | rightServo.omega_max = 45*pi/(.1*180); % Maximum rotation velocity [rad/s
    | ]
84 | rightServo.maxDeflection = 45*pi/180; % Max deflection
85 | rightServo.minDeflection = -45*pi/180; % Max deflection
86 | rightServo.Kp = 10; %
87 |
88 | stabilizerServo.omega_max = 40*pi/(.12*180); % Maximum rotation velocity [rad
    | /s]
89 | stabilizerServo.maxDeflection = 45*pi/180; % Max deflection
90 | stabilizerServo.minDeflection = -45*pi/180; % Max deflection
91 | stabilizerServo.Kp = 10; %
92 |
93 | %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
94 | % PROPULSION SYSTEM %
95 | %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
96 |
97 | thrust.r_t = [-0.06 0 -0.075]'; % Position of center of thrust
98 | thrust.prop_d = 0.3; % Propeller diameter
99 | thrust.Ct = 3.3; % Coefficient of thrust
100 |
101 | battery.Emax = 8000; % Battery capacity [mAh]
102 | battery.Un = 11.1; % Nominal battery voltage
103 | battery.a = 1.2e-005; % Linear voltage drop
104 | battery.b = 20; % Quadratic voltage drop -
    | dominant at low battery charge
105 | battery.Ri = 0.03; % Inner resistance
106 |
107 | motor.Jm = .2; % Motor inertia
108 | motor.La = 1.5e-2; % Motor inductance
109 | motor.R = 0.5; % DC Motor winding resistance. U
    | ^2/P = 10^2/200=0.5
110 | motor.Ke = 0.75; % Shaft torque constant. Shaft
    | torque = Ke*i
111 | motor.Kp = 0.0176; % Propeller torque constant.
    | Propeller torque = w|w|*Kp
112 |
113 | %
114 | %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
115 | % %
116 | % DEFINE CONTROLLER CONSTANTS %
117 | % %
118 | %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
119 |
120 | %%%% SURGE CONTROLLER
121 | PIDs.surge.p = 20;
122 | PIDs.surge.i = 1;
123 | PIDs.surge.d = -.02;
124 | PIDs.surge.imax = 10;
125 | PIDs.surge.imin = -10;
126 | PIDs.surge.max = 10;
127 | PIDs.surge.min = 0;
128 | PIDs.surge.k_au = 1;

```

```

129
130 %%%% HEADING
131 PIDs.roll.p = 2;
132 PIDs.roll.i = 0;
133 PIDs.roll.d = 0;
134 PIDs.roll.imax = leftServo.maxDeflection;
135 PIDs.roll.imin = leftServo.minDeflection;
136 PIDs.roll.max = leftServo.maxDeflection;
137 PIDs.roll.min = leftServo.minDeflection;
138 PIDs.roll.k_auw = 0;
139
140 PIDs.yaw.p = 2;
141 PIDs.yaw.i = 0;
142 PIDs.yaw.d = 0;
143 PIDs.yaw.imax = 40*pi/180;
144 PIDs.yaw.imin = -40*pi/180;
145 PIDs.yaw.max = 40*pi/180;
146 PIDs.yaw.min = -40*pi/180;
147 PIDs.yaw.k_auw = 0;
148
149
150 %%%% ALTITUDE
151 PIDs.pitch.p = 3;
152 PIDs.pitch.i = .05;
153 PIDs.pitch.d = 0;
154 PIDs.pitch.imax = leftServo.maxDeflection;
155 PIDs.pitch.imin = leftServo.minDeflection;
156 PIDs.pitch.max = leftServo.maxDeflection;
157 PIDs.pitch.min = leftServo.minDeflection;
158 PIDs.pitch.k_auw = 0;
159
160 PIDs.altitude.p = .1;
161 PIDs.altitude.i = .05;
162 PIDs.altitude.d = 0;
163 PIDs.altitude.imax = 5*pi/180;
164 PIDs.altitude.imin = -5*pi/180;
165 PIDs.altitude.max = 30*pi/180;
166 PIDs.altitude.min = -30*pi/180;
167 PIDs.altitude.k_auw = 0;
168
169
170 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
171 %                                                                 %
172 %           DEFINE REFERENCE MODEL CONSTANTS                       %
173 %                                                                 %
174 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
175
176
177 refm.altitude.omega = 75*pi/180;
178 refm.altitude.sat1 = 8;
179 refm.altitude.sat2 = 5;
180
181 refm.pitch.omega = 5*360*pi/180;
182 refm.pitch.sat1 = 360*pi/180;
183 refm.pitch.sat2 = 360*pi/180;
184
185 refm.yaw.omega = 36*pi/180;
186 refm.yaw.sat1 = (360/20)*pi/180;
187 refm.yaw.sat2 = 36*pi/180;
188
189 refm.roll.omega = 2*360*pi/180;
190 refm.roll.sat1 = 180*pi/180;
191 refm.roll.sat2 = 180*pi/180;
192
193 refm.surge.omega = 180*pi/180;
194 refm.surge.sat1 = 15;
195 refm.surge.sat2 = 15;
196

```

```

197
198
199 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
200 %
201 %           DEFINE ALL SIMULATOR MODEL PARAMETERS           %
202 %
203 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
204
205
206 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
207 %           LANDING GEAR           %
208 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
209
210 % Intermediate values:
211 Kd = diag([0.01 .35 450]);           % Damping coefficient
212 Ks = diag([0 0 350]);           % Spring coefficient
213
214 r_nw = [1 0 0.1]';           % Position of nose wheel in BODY
215 r_lw = [0 -0.5 0.1]';           % Position of left wheel in BODY
216 r_rw = [0 0.5 0.1]';           % Position of right wheel in BODY
217
218
219 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
220 %           AERODYNAMIC CENTERS           %
221 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
222
223 af.r = [ 0.4  0  0.0 ]';           % Position of fuselage
224     aerodynamic center
225 awl.r = [ 0.35 -0.42  0 ]';           % Position of left wing
226     aerodynamic center
227 awr.r = [ 0.35  0.42  0 ]';           % Position of right wing
228     aerodynamic center
229 asl.r = [ 0.07 -0.09  0 ]';           % Position of left stabilizer
230     aerodynamic center
231 asr.r = [ 0.07  0.09  0 ]';           % Position of right stabilizer
232     aerodynamic center
233
234
235 af.q = [ 1  0  0 0]';           % Orientation of fuselage
236     aerodynamic center axes
237 awl.q = [ 1  0  0 0]';           % Orientation of left wing
238     aerodynamic center axes
239 awr.q = [ 1  0  0 0]';           % Orientation of right wing
240     aerodynamic center axes
241 asl.q = [sqrt(3)/2 -1/2 0 0]';           % Orientation of left
242     stabilizer aerodynamic center axes
243 asr.q = [sqrt(3)/2  1/2 0 0]';           % Orientation of right
244     stabilizer aerodynamic center axes
245
246
247 % DEFINE LIFT PROPERTIES:
248 af.CLmax = 0.1;           % Max lift coefficient of
249     fuselage
250 awl.CLmax = 0.6;           % Max lift coefficient of left
251     wing
252 awr.CLmax = 0.6;           % Max lift coefficient of right
253     wing
254 asl.CLmax = 0.6;           % Max lift coefficient of left
255     stabilizer
256 asr.CLmax = 0.6;           % Max lift coefficient of right
257     stabilizer
258
259
260 af.as = 13*pi/180;           % Stall angle, fuselage
261 awr.as = 13*pi/180;           % Stall angle, left wing
262 awl.as = 13*pi/180;           % Stall angle, right wing
263 asl.as = 15*pi/180;           % Stall angle, left stabilizer
264 asr.as = 15*pi/180;           % Stall angle, right stabilizer
265
266
267 af.a0 = -2*pi/180;           % Zero lift angle, fuselage
268 awl.a0 = -4*pi/180;           % Zero lift angle, left wing

```

```

250 awr.a0 = -4*pi/180; % Zero lift angle, right wing
251 asl.a0 = 0*pi/180; % Zero lift angle, left
    stabilizator
252 asr.a0 = 0*pi/180; % Zero lift angle, right
    stabilizator
253
254 af.A1 = 0.079; % Characteristic lift area,
    fuselage
255 awl.A1 = 0.3815; % Characteristic lift area, left
    wing
256 awr.A1 = 0.3815; % Characteristic lift area, right
    wing
257 asl.A1 = 0.03815; % Characteristic lift area, left
    stabilizator
258 asr.A1 = 0.03815; % Characteristic lift area, right
    stabilizator
259
260 af.kdl = 0; % Flap/AoA ratio, fuselage
261 awl.kdl = 0.15; % Flap/AoA ratio, left wing
262 awr.kdl = 0.15; % Flap/AoA ratio, right wing
263 asl.kdl = 0.3; % Flap/AoA ratio, left
    stabilizator
264 asr.kdl = 0.3; % Flap/AoA ratio, right
    stabilizator
265
266 % DEFINE DRAG PROPERTIES:
267 af.CDp = diag([.5 1 1]); % Parasite drag coefficients,
    fuselage
268 awl.CDp = diag([0.209 0.3 1.17]); % Parasite drag coefficients,
    left wing
269 awr.CDp = diag([0.209 0.3 1.17]); % Parasite drag coefficients,
    right wing
270 asl.CDp = diag([0.18 0.3 1.17]); % Parasite drag coefficients,
    left stabilizator
271 asr.CDp = diag([0.18 0.3 1.17]); % Parasite drag coefficients,
    right stabilizator
272
273 af.CDpd = diag([0 0 0]); % Parasite drag coefficients, due
    to flap deflection fuselage
274 awl.CDpd = diag([0.23 0 0.05]); % Parasite drag coefficients, due
    to flap deflection left wing
275 awr.CDpd = diag([0.23 0 0.05]); % Parasite drag coefficients, due
    to flap deflection right wing
276 asl.CDpd = diag([.1 0 0.1]); % Parasite drag coefficients, due
    to flap deflection left stabilizator
277 asr.CDpd = diag([.1 0 0.1]); % Parasite drag coefficients, due
    to flap deflection right stabilizator
278
279 af.Ad = [0.0126 0.0958 af.A1];
280 awl.Ad = [0.0183 0.052 awl.A1];
281 awr.Ad = [0.0183 0.052 awr.A1];
282 asl.Ad = awl.Ad.*0.1;
283 asr.Ad = awr.Ad.*0.1;
284
285 af.CDi = 0.16; % Induced drag coefficient,
    fuselage
286 awl.CDi = 0.16; % Induced drag coefficient, left
    wing
287 awr.CDi = 0.16; % Induced drag coefficient, right
    wing
288 asl.CDi = 0.16; % Induced drag coefficient, left
    stabilizator
289 asr.CDi = 0.16; % Induced drag coefficient, right
    stabilizator
290
291 % DEFINE PITCHING MOMENT PROPERTIES
292 af.Cmmax = .8; % Max

```



```

293 awl.Cmmax = .8; % Max pitching moment coefficient
    of left wing
294 awr.Cmmax = .8; % Max pitching moment coefficient
    of right wing
295 asl.Cmmax = .8; % Max pitching moment coefficient
    of left stabilizator
296 asr.Cmmax = .8; % Max pitching moment coefficient
    of right stabilizator
297
298 af.c_bar = 1.06; % Max lift coefficient of fuselage
299 awl.c_bar = 0.62; % Max pitching moment coefficient
    of left wing
300 awr.c_bar = 0.62; % Max pitching moment coefficient
    of right wing
301 asl.c_bar = 0.062; % Max pitching moment coefficient
    of left stabilizator
302 asr.c_bar = 0.062; % Max pitching moment coefficient
    of right stabilizator
303
304 af.cm0 = 1*pi/180; % Zero moment angle, fuselage
305 awl.cm0 = 5*pi/180; % Zero moment angle, left wing
306 awr.cm0 = 5*pi/180; % Zero moment angle, right wing
307 asl.cm0 = 0*pi/180; % Zero moment angle, left
    stabilizator
308 asr.cm0 = 0*pi/180; % Zero moment angle, right
    stabilizator
309
310 af.kdp = 0; % Flap/AoA ratio, fuselage
311 awl.kdp = 0.5; % Flap/AoA ratio, left wing
312 awr.kdp = 0.5; % Flap/AoA ratio, right wing
313 asl.kdp = 0.5; % Flap/AoA ratio, left
    stabilizator
314 asr.kdp = 0.5; % Flap/AoA ratio, right
    stabilizator
315
316 %%
317 disp('All variables initialized')
318 button = questdlg('Perform Simulink simulation?');
319 if (strcmp(button,'Yes'))
320     disp('Starting Simulink simulation')
321     sim RecceD6;
322     disp('Simulink simulation finished')
323 end

```

## B.2 MakeVideo.m (Matlab script)

```

1 % Author: Kristoffer Dønnestad, 17.06.2011
2 % Attachment to M.Sc. thesis
3 % Unmanned Vehicle Laboratory
4 % Department of Engineering Cybernetics
5 % Norwegian University of Science and Technology
6 %
7 % GENERATE VIDEO:
8 GENERATE_AVI_VIDEO = 1; % 1 for yes, 0 for no
9 avipath = '..\AVI\'; % Relative path to AVI folder, from current folder
10 framesprvideo = 350; % Maximum number of videoframes for each AVI file
11
12 % Movie settings
13 plotInterval = 1/24; % 1/plotInterval = Frames pr second
14 fromTime = 0; % Start movie time
15 toTime = simtime; % Stop movie time
16
17 % Figure out which indexes from the workspace variables that should be used

```

```

18 % to generate each videoframe (last 20 seconds)
19 timeStepIndexes = zeros(length(p.signals.values),1);
20 cIndex = 0;
21 for n=1:length(p.signals.values)
22     if (p.time(n)-plotInterval*cIndex > plotInterval) && (p.time(n)>fromTime)
23         && (p.time(n)<toTime)
24             cIndex = cIndex + 1;
25             timeStepIndexes(n) = 1;
26     end
27 end
28
29 figurehandle = figure(figure);
30 mainaxes = gca;
31
32 set(figurehandle, 'Visible', 'off');
33 mTextBox = uicontrol('style','text');
34 set(mTextBox, 'Position', [267 960 150 20]);
35 set(mTextBox, 'FontSize', 12);
36
37 cTextBox = uicontrol('style','text');
38 set(cTextBox, 'Position', [267 1000 350 20]);
39 set(cTextBox, 'FontSize', 12);
40 ctimeindex = 1;
41 ccaseindex = 1;
42
43 movieframeindex = 0;
44 totalmovieframeindex = 0;
45 currentvideonr = 1;
46 disp(['Total number of videoframes: ' num2str(cIndex)]);
47 videotime = tic;
48 starttime = clock;
49
50 % Generate a speed vector for plotting
51 speedNED = zeros(length(p.signals.values),1);
52 for n=1:length(p.signals.values)
53     speedNED(n) = norm(v.signals.values(n,1:3));
54 end
55
56 % Adjust axes frames for 3d plot
57 axrange = 1.2; % Big number give small UAV
58 axcenter = [p.signals.values(1,1) p.signals.values(1,2) p.signals.values(1,3)
59 ];
60 ax = [axcenter(1)-axrange % MinX
61       axcenter(1)+axrange % MaxX
62       axcenter(2)-axrange % MinY
63       axcenter(2)+axrange % MaxY
64       axcenter(3)-axrange*2/3 % MinZ
65       axcenter(3)+axrange*2/3]'; % MaxZ
66
67 for n=1:length(p.signals.values)
68     if timeStepIndexes(n)
69
70         plotSec = 20;
71         ctime = p.time(n);
72         stopTime = ctime - plotSec;
73         plotIndexes = p.time<=ctime & p.time>stopTime;
74         timeArray = p.time(plotIndexes)-ctime;
75
76         totsec = floor(ctime);
77         min = floor(totsec/60);
78         sec = floor(totsec - min*60);
79         minStr = num2str(min);
80         secStr = num2str(sec);
81         if(min<10) minStr = ['0' minStr]; end
82         if(sec<10) secStr = ['0' secStr]; end
83         set(mTextBox, 'String', ['Time: ' minStr ':' secStr]);

```

```

84 set(cTextBox,'BackgroundColor','default');
85 if(ctime >= ref.time(ctimeindex+1))
86     ctimeindex = ctimeindex + 1;
87
88     if (ccaseindex ~= caseindex(ctimeindex))
89         ccaseindex = caseindex(ctimeindex);
90         set(cTextBox,'BackgroundColor','red');
91     else
92         end
93
94 end
95 set(cTextBox,'String',casetext(ccaseindex));
96
97
98 % 3D PLOT
99 splot3d = subplot(4,5,[1 2 3 6 7 8 11 12 13]);
100 axtolerance = [axrange/3 axrange/3 axrange/5];
101 if(abs(p.signals.values(n,1)-axcenter(1)) > axtolerance(1))
102     axcenter(1) = p.signals.values(n,1) - axtolerance(1)*sign(p.
103     signals.values(n,1)-axcenter(1));
104     ax(1) = axcenter(1) - axrange;
105     ax(2) = axcenter(1) + axrange;
106 end
107 if(abs(p.signals.values(n,2)-axcenter(2)) > axtolerance(2))
108     axcenter(2) = p.signals.values(n,2) - axtolerance(2)*sign(p.
109     signals.values(n,2)-axcenter(2));
110     ax(3) = axcenter(2) - axrange;
111     ax(4) = axcenter(2) + axrange;
112 end
113 if(abs(p.signals.values(n,3)-axcenter(3)) > axtolerance(3))
114     axcenter(3) = p.signals.values(n,3) - axtolerance(3)*sign(p.
115     signals.values(n,3)-axcenter(3));
116     ax(5) = axcenter(3) - axrange*2/3;
117     ax(6) = axcenter(3) + axrange*2/3;
118 end
119
120 hold off
121 plot3D(fignr,splot3d,p.signals.values(n,:),q.signals.values(n,:),AoA.
122     signals.values(n),flap_lw.signals.values(n),flap_rw.signals.values(
123     n),flap_ls.signals.values(n),flap_rs.signals.values(n),r_co_cg,ax);
124 hold on
125
126 % ADD WIND ARROW
127 % Get and compute relevant wind data
128 wx = wind.signals.values(n,1);
129 wy = wind.signals.values(n,2);
130 wz = wind.signals.values(n,3);
131 air = cartesian2air(wind.signals.values(n,:));
132 Wstr = air(1);
133 Walpha = air(2);
134 Wbeta = air(3);
135
136 if(Wstr > 1e-5)
137     %Define arrow geometry:
138     Sn = 8;
139     ArrowX = meshgrid(0:Wstr/Sn:Wstr);
140     ArrowY = meshgrid(cos(0:2*pi/Sn:2*pi).*3)';
141     ArrowZ = meshgrid(sin(0:2*pi/Sn:2*pi).*3)';
142
143     ArrowY(:,Sn) = ArrowY(:,Sn).*2;
144     ArrowZ(:,Sn) = ArrowZ(:,Sn).*2;
145     ArrowY(:,Sn+1) = ArrowY(:,Sn).*0;
146     ArrowZ(:,Sn+1) = ArrowZ(:,Sn).*0;
147     ArrowX(:,Sn+1) = ArrowX(:,Sn)+2*Wstr/Sn;
148
149     ArrowX = ArrowX(:,2:end);
150     ArrowY = ArrowY(:,2:end);

```

```

147     ArrowZ = ArrowZ(:,2:end);
148
149     % Rotate arrow according to wind data
150     WindRotmatr = rotateMatrQuat(e2q(0,0,Wbeta));
151     [ms ns] = size(ArrowX);
152     for mm=1:ms
153         for nn = 1:ns
154             xyz = WindRotmatr*[ArrowX(mm,nn) ArrowY(mm,nn) ArrowZ(mm,
155                 nn)]';
156             ArrowX(mm,nn) = xyz(1).*0.1 + ax(2) - 0.3;
157             ArrowY(mm,nn) = xyz(2).*0.1 + ax(3) + 0.3;
158             ArrowZ(mm,nn) = xyz(3).*0.1 + ax(5) + 0.3;
159         end
160     end
161
162     % Also add a sphere to the plot to show arrow origin
163     [Sx,Sy,Sz] = sphere(Sn);
164     [ms ns] = size(Sx);
165     for mm=1:ms
166         for nn = 1:ns
167             xyz = WindRotmatr*[Sx(mm,nn) Sy(mm,nn) Sz(mm,nn)]';
168             Sx(mm,nn) = xyz(1).*0.05 + ax(2) - .3;
169             Sy(mm,nn) = xyz(2).*0.05 + ax(3) + .3;
170             Sz(mm,nn) = xyz(3).*0.05 + ax(5) + .3;
171         end
172     end
173
174     % Plot the arrow and the sphere into the figure
175     surf(ArrowX,ArrowY,ArrowZ,ones(size(ArrowX))*(0.5 - Wstr/20));
176     surf(Sx,Sy,Sz,ones(size(Sx)).*0.5);
177
178     end
179
180     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
181     % "ARTIFICIAL HORIZON" PLOT
182     subplot(4,5,[4 5])
183     plot(eulerAngles.signals.values(plotIndexes,3).*(180/pi),eulerAngles.
184         signals.values(plotIndexes,2).*(180/pi),'m');
185     hold on
186
187     r = EulerAnglesRefMod.signals.values(n,1);
188     plane = [cos(r) -sin(r); sin(r) cos(r)]'*[35 -35 -7 0 7; 0 0 14 0
189         14];
190     plot(EulerAnglesRefMod.signals.values(n,3).*(180/pi),
191         EulerAnglesRefMod.signals.values(n,2).*(180/pi),'go');
192     plot(plane(1,1:2)+EulerAnglesRefMod.signals.values(n,3).*(180/(pi)),
193         plane(2,1:2)+EulerAnglesRefMod.signals.values(n,2).*(180/(pi)),'g?')
194     ;
195     plot(plane(1,3:5)+EulerAnglesRefMod.signals.values(n,3).*(180/(pi)),
196         plane(2,3:5)+EulerAnglesRefMod.signals.values(n,2).*(180/(pi)),'g?')
197     ;
198
199     r = eulerAngles.signals.values(n,1);
200     plane = [cos(r) -sin(r); sin(r) cos(r)]'*[35 -35 -7 0 7; 0 0 14 0
201         14];
202     plot(eulerAngles.signals.values(n,3).*(180/pi),eulerAngles.signals.
203         values(n,2).*(180/pi),'bo');
204     plot(plane(1,1:2)+eulerAngles.signals.values(n,3).*(180/(pi)),plane
205         (2,1:2)+eulerAngles.signals.values(n,2).*(180/(pi)),'b');
206     plot(plane(1,3:5)+eulerAngles.signals.values(n,3).*(180/(pi)),plane
207         (2,3:5)+eulerAngles.signals.values(n,2).*(180/(pi)),'b');
208
209     r = EulerAnglesRef.signals.values(n,1);
210     plane = [cos(r) -sin(r); sin(r) cos(r)]'*[35 -35 -7 0 7; 0 0 14 0
211         14];
212     plot(EulerAnglesRef.signals.values(n,3).*(180/pi),EulerAnglesRef.
213         signals.values(n,2).*(180/pi),'ro');

```

```

200 plot(plane(1,1:2)+EulerAnglesRef.signals.values(n,3).*(180/(pi)),
201       plane(2,1:2)+EulerAnglesRef.signals.values(n,2).*(180/(pi)), 'r');
202 plot(plane(1,3:5)+EulerAnglesRef.signals.values(n,3).*(180/(pi)),
203       plane(2,3:5)+EulerAnglesRef.signals.values(n,2).*(180/(pi)), 'r');
204
205 hold off
206 title('Artificial horizon')
207 set(gca,'XTick',[-180 -90 0 90 180]);
208 set(gca,'XTickLabel',{'South' 'West' 'North' 'East' 'South'});
209 ylabel('Pitch [deg]')
210 xlim([-180,180]);
211 ylim([-58,58]);
212 grid on
213
214 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
215 % ORIENTATION PLOT
216 subplot(4,5,10)
217 plot(timeArray,eulerAngles.signals.values(plotIndexes,1).*(180/pi),'b
218       ');
219 hold on
220 plot(timeArray,eulerAngles.signals.values(plotIndexes,2).*(180/pi),'r
221       ');
222 plot(timeArray,eulerAngles.signals.values(plotIndexes,3).*(180/pi),'c
223       ');
224 hold off
225 legend('\phi - Roll','\theta - Pitch','\psi - Yaw','Location','
226       NorthWest')
227 title('Orientation')
228 ylabel('Angle [deg]')
229 xlabel('time [s]')
230 xlim([-plotSec,0]);
231 grid on
232
233 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
234 % ANGLE OF ATTACK /SIDESLIP PLOT
235 subplot(4,5,20)
236 plot(timeArray,AoA.signals.values(plotIndexes).*(180/pi),'b');
237 hold on
238 plot(timeArray,beta.signals.values(plotIndexes).*(180/pi),'r');
239 legend('\alpha','\beta','Location','NorthWest')
240 xlim([-plotSec,0]);
241 xlabel('time [s]');
242 ylabel('Angle [deg]')
243 title('Angle of attack')
244 hold off
245 grid on;
246
247 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
248 % POSITION PLOT
249 subplot(4,5,15)
250 plot(p.signals.values(plotIndexes,2),p.signals.values(plotIndexes,1),
251       'm');
252 hold on
253 plot(p.signals.values(n,2),p.signals.values(n,1),'bo');
254 hold off
255 title('Position in NE-plane (last 20s)')
256 ylabel('N - North [m]');
257 xlabel('E - East [m]');
258 grid on
259 valuesvector = [p.signals.values(plotIndexes,2) p.signals.values(
260       plotIndexes,1)'];
261 if(abs(max(valuesvector)) > 10 )
262     axis equal
263 else
264     axis([-10 10 -10 10]);

```

```

260     end
261
262
263
264     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
265     % ALTITUDE PLOT
266     subplot(4,5,14)
267     title('Altitude')
268     plot(timeArray,-p.signals.values(plotIndexes,3),'b');
269     hold on
270     silarrow = [ 0 0 .6 .6 1 .6 .6 0 ;
271                -.1 .1 .1 .3 0 -.3 -.1 -.1 ]';
272     maxmin = get(gca,'ylim');
273     if(maxmin(1) > altitude_ref.signals.values(n))
274         set(gca,'Ylim',[altitude_ref.signals.values(n) maxmin(2)]);
275     elseif (maxmin(2) < altitude_ref.signals.values(n))
276         set(gca,'Ylim',[maxmin(1) altitude_ref.signals.values(n)]);
277     end
278     maxmin = get(gca,'ylim');
279     plot(silarrow(:,1)*2-2,.2*silarrow(:,2)*(maxmin(2) - maxmin(1))+
280          altitude_ref.signals.values(n),'g');
281     set(gca,'Ylim',maxmin);
282     hold off;
283
284     title('Altitude');
285     xlim([-plotSec,0]);
286     xlabel('time [s]');
287     ylabel('Altitude [m]');
288     grid on
289
290     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
291     % LEFT FLAPS PLOT
292     subplot(4,5,16)
293     plot(timeArray,flap_lw.signals.values(plotIndexes).*(180/pi),'b');
294     hold on
295     plot(timeArray,flap_ls.signals.values(plotIndexes).*(180/pi),'r');
296     hold off
297     legend('\delta L wing','\delta L rudder.','Location','NorthWest');
298     xlim([-plotSec,0]);
299     xlabel('time [s]');
300     ylabel('Angle [deg]');
301     title('Left control inputs')
302     grid on
303
304     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
305     % POWER PLANT PLOT
306     subplot(4,5,17)
307     plot(timeArray,power_consumption.signals.values(plotIndexes,1))
308     hold on
309     voltageStr = num2str(batt_u.signals.values(n,1));
310     if length(voltageStr) > 4
311         voltageStr = voltageStr(1:4);
312     end
313     %text(-19.3,60,['Propeller: ' num2str(round(prop_velocity.signals.
314                values(n)*60/(2*pi))) ' RPM'])
315     text(-19.3,35,['Battery voltage: ' voltageStr ' V'])
316     text(-19.3,10,[' num2str(round(batt_mAh.signals.values(n,1))) ' mAh
317                left ( ' num2str(round(batt_mAh.signals.values(n,1)*100/battery.Emax
318                ) '%)'])
319     xlim([-plotSec,0]);
320     ylim([0,240]);
321     xlabel('time [s]');
322     ylabel('Power consumption [W]');
323     title('Power plant status')
324     hold off
325     %grid on

```

```

324
325
326 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
327 % RIGHT FLAPS PLOT
328 subplot(4,5,18)
329 plot(timeArray, flap_rw.signals.values(plotIndexes).*(180/pi), 'b');
330 hold on
331 plot(timeArray, flap_rs.signals.values(plotIndexes).*(180/pi), 'r');
332 hold off
333 legend('\delta R wing', '\delta R rudder.', 'Location', 'NorthWest');
334 xlim([-plotSec, 0]);
335 xlabel('time [s]');
336 ylabel('Angle [deg]');
337 title('Right control inputs')
338 grid on
339
340
341
342 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
343 % VELOCITIES PLOT
344 subplot(4,5,19)
345 plot(timeArray, speedNED(plotIndexes), 'm');
346 hold on;
347 plot(timeArray, v.signals.values(plotIndexes, 1), 'b');
348 plot(timeArray, v.signals.values(plotIndexes, 2), 'r');
349 plot(timeArray, v.signals.values(plotIndexes, 3), 'c');
350 %plot(timeArray, v.signals.values(plotIndexes, 3), 'r'); %
    INSERT AIRSPEED
351
352 maxmin = get(gca, 'ylim');
353 if(maxmin(1) > v.signals.values(n, 1))
354     set(gca, 'Ylim', [surge_ref.signals.values(n) maxmin(2)]);
355 elseif (maxmin(2) < surge_ref.signals.values(n))
356     set(gca, 'Ylim', [maxmin(1) surge_ref.signals.values(n)]);
357 end
358 maxmin = get(gca, 'ylim');
359 plot(silarrow(:, 1)*2-2, .2*silarrow(:, 2)*(maxmin(2) - maxmin(1))+
    surge_ref.signals.values(n), 'b');
360 plot(silarrow(:, 1)*2-2, .2*silarrow(:, 2)*(maxmin(2) - maxmin(1))+
    airvelocity_body.signals.values(n, 1), 'k');
361 set(gca, 'Ylim', maxmin);
362
363 hold off;
364 title('Speed and velocities')
365 legend('Speed', 'u', 'v', 'w', 'Location', 'NorthWest');
366 xlim([-plotSec, 0]);
367 xlabel('time [s]');
368 ylabel('Velocity [m/s]');
369 grid on
370
371
372 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
373 % WIND PLOT
374 subplot(4,5,9)
375 circle = [sin(0:2*pi/90:2*pi)' cos(0:2*pi/90:2*pi)'];
376 Windstrength = [0.3 1.6 3.4 5.5 8 10.8 13.9 17.2 20.8 23.5 28.5
    32.7];
377 BeaufortScale = {...
378     'Calm'
379     'Light air'
380     'Light breeze'
381     'Gentle breeze'
382     'Moderate breeze'
383     'Fresh breeze'
384     'Strong breeze'
385     'Near gale'
386     'Gale'
387     'Storm'

```

```

388         'Violent storm'
389         'Hurricane'
390     };
391
392     % Define aircraft silhouette:
393     silfus = [ -.8 .8 .8 5 5 .8 .8 0 -.8 -.8 -5 -5 -.8 -.8 ;
394              -4.5 -4.5 -4 -4 -1 4 5 7 5 4 -1 -4 -4 -4.5
395              ]';
396
397     % Define unit wind arrow silhouette:
398     silarrow = [ -.1 .1 .1 .3 0 -.3 -.1 -.1 ;
399                0 0 .6 .6 1 .6 .6 0 ]';
400
401     desreftap = [2.5*silarrow(:,1)'
402                 -2.5*silarrow(:,2)'+9.5]';
403
404     cheading = eulerAngles.signals.values(n,3);
405     desheading = EulerAnglesRef.signals.values(n,3);
406     refheading = EulerAnglesRefMod.signals.values(n,3);
407
408     silfusAct = zeros(size(silfus));
409     for siln = 1:length(silfus)
410         silfusAct(siln,:) = [cos(cheading) sin(cheading) ; -sin(cheading)
411                             cos(cheading)]*silfus(siln,:)' ;
412     end
413
414     silfusDes = zeros(size(desreftap));
415     silfusRef = zeros(size(desreftap));
416     for siln = 1:length(desreftap)
417         silfusDes(siln,:) = [cos(desheading) sin(desheading) ; -sin(
418                             desheading) cos(desheading)]*desreftap(siln,:)' ;
419         silfusRef(siln,:) = [cos(refheading) sin(refheading) ; -sin(
420                             refheading) cos(refheading)]*desreftap(siln,:)' ;
421     end
422
423     for siln = 1:length(silarrow)
424         silarrow(siln,:) = Wstr*[cos(Wbeta) sin(Wbeta) ; -sin(Wbeta) cos(
425                                 Wbeta)]*silarrow(siln,:)' ;
426     end
427
428     plot(circle(:,1)*7,circle(:,2)*7);
429     hold on
430     plot(circle(:,1)*.3,circle(:,2)*.3);
431     plot([0 0],[6.5 7.5]);
432     plot([0 0],[-6.5 -7.5]);
433     plot([6.5 7.5],[0 0]);
434     plot([-6.5 -7.5],[0 0]);
435
436     plot(silfusDes(:,1),silfusDes(:,2),'r');
437     plot(silfusRef(:,1),silfusRef(:,2),'g');
438     plot(silfusAct(:,1),silfusAct(:,2),'b');
439
440     plot(silarrow(:,1),silarrow(:,2),'m');
441
442     hold off
443
444     for ws = 1:length(Windstrength)
445         BeaufortIndex = ws;
446         if Wstr < Windstrength(ws)
447             break;
448         end
449     end
450
451     title('Compass')

```



```

451     text(0.4,7.8,['North']);
452     text(7.8,0.4,['East']);
453     text(-11.2,0.4,['West']);
454     text(0.4,-7.8,['South']);
455
456
457
458     WstrStr = num2str(Wstr);
459     if length(WstrStr) > 4
460         WstrStr = WstrStr(1:4);
461     end
462
463     text(-12.5,-4.5, ['WIND:']);
464     text(-12.5,-6, [ num2str(round(Wbeta*180/pi)) '\circ' ]);
465     text(-12.5,-7.5, [ WstrStr ' m/s' ]);
466     text(-12.5,-9, [ BeaufortScale(BeaufortIndex)]);
467
468     text(-12.5,9, ['AIRCRAFT:']);
469     text(-12.5,7.5, [num2str(round(eulerAngles.signals.values(n,3)*180/pi
470         )) '\circ' ]);
471
472     axis([-10 10 -10 10]);
473     axis equal;
474     set(gca,'ytick',[]);
475     set(gca,'xtick',[]);
476
477     if (GENERATE_AVI_VIDEO)
478         % Create AVI object, and add the current plot to the AVI object:
479         movieframeindex = movieframeindex+1;           % Keep
480         track of number of movieframes in the current AVI
481         totalmovieframeindex = totalmovieframeindex + 1;           % Keep
482         track of total number of movieframes
483
484         % !!!! Make sure all video frames have the exact same size: !!!!
485         scrsz = get(0,'ScreenSize');
486         set(gcf,'Position',[20 50 scrsz(3)-40 scrsz(4)-100]);
487
488         movieframes(movieframeindex) = getframe(fighandle);           % Get the
489         current figure frame, and store it to a movieframe array
490         if(movieframeindex == framesprvideo)           % If
491             maximum number of videoframes pr AVI file is reached
492
493             timeused = toc(videotime);           %
494             Estimate remaining movie generation time
495             videotime = tic;
496             timeleft = (timeused / framesprvideo) * (cIndex -
497                 totalmovieframeindex);
498             hleft = floor(timeleft/(60*60));
499             mleft = floor((timeleft - hleft*60*60)/60);
500             sleft = floor(timeleft - hleft*60*60 - mleft*60);
501             disp(['Videoframe ' num2str(totalmovieframeindex) '/' num2str(
502                 cIndex) ' completed. Estimated time left: ' num2str(hleft)
503                 'h, ' num2str(mleft) 'min, ' num2str(sleft) 'sec.']);
504
505             time = clock;
506             ss = floor(mod(time(6)+sleft,60));
507             mm = floor((time(5)+mleft + floor((time(6)+sleft)/60)));
508             hh = mod(time(4)+hleft,24);
509             disp(['Estimated finishing time at ' num2str(hh) ':' num2str(
510                 mm) ' ' num2str(ss) '.']);
511
512             % Create AVI object
513             aviobj = avifile([avipath 'Recce_' num2str(floor(starttime(1)
514                 )) '_' num2str(floor(starttime(2))) '_' num2str(floor(
515                 starttime(3))) '_kl_' num2str(floor(starttime(4))) '_'
516                 num2str(floor(starttime(5))) '_vnr_' num2str(currentvideonr
517                 )], 'FPS', round(1/plotInterval), 'compression', 'None');

```

```

505         for n=1:length(movieframes)
506             aviobj = addframe(aviobj,movieframes(n));           % Add all
                    matlab figures to the AVI object
507         end
508         aviobj = close(aviobj);                                 % ...
                    close the AVI object
509         clear movieframes;                                     % Dispose
                    the covered videoframes
510         movieframeindex = 0;                                   % Reset
                    the current videoframe indexes
511         currentvideonr = currentvideonr + 1;                   % And
                    increase the number of videos
512         end
513     end % End IF AVI video
514
515     end % End Current plot
516 end % Finished iterating through all plot frames
517
518 % Store all the remaining matlab figures to the last AVI file
519 if (GENERATE_AVI_VIDEO)
520     aviobj = avifile([avipath 'Recce_' num2str(floor(starttime(1))) '_' num2str(
                    floor(starttime(2))) '_' num2str(floor(starttime(3))) '_k1_' num2str(floor(
                    starttime(4))) '_' num2str(floor(starttime(5))) '_vnr_' num2str(
                    currentvideonr)],'FPS',round(1/plotInterval),'compression','None');
521     for n=1:length(movieframes)
522         aviobj = addframe(aviobj,movieframes(n));
523     end
524     aviobj = close(aviobj);
525 end

```

### B.3 Plot3d.m (Matlab function)

```

1 % figh = plot3D(fignr,splot3d,p,q,aoa,a_fl,a_fr,a_el,a_er,r_og,ax)
2 %
3 % Inputs:
4 % fignr - The figure number
5 % splot3d - The subplot handle to plot 3D plot into
6 % p - The position in NED coordinates. [3,1]
7 % q - A quaternion specifying the orientation of BODY in NED. [4x1]
8 % aoa - Angle of attack. (rad)
9 % a_fl - Angle of the left flap (rad)
10 % a_fr - Angle of the right flap (rad)
11 % a_el - Angle of the left elevator (rad)
12 % a_er - Angle of the right elevator (rad)
13 % r_og - A vector specifying the position of CG in CO. [3x1]
14 % This point is the point the aircraft is rotated about
15 % ax - Axis argument, e.g. [XMIN XMAX YMIN YMAX ZMIN ZMAX]
16 %
17 % Returns:
18 % fig - The figure handle
19 %
20 % Author: Kristoffer Dønnestad, 17.06.2011
21 % Attachment to M.Sc. thesis
22 % Unmanned Vehicle Laboratory
23 % Department of Engineering Cybernetics
24 % Norwegian University of Science and Technology
25 %
26 function figh = plot3D(fignr,splot3d,p,q,aoa,a_fl,a_fr,a_el,a_er,r_og,ax)
27 figh = figure(fignr);
28 cla(splot3d)
29 scrsz = get(0,'ScreenSize');
30 set(gcf,'Position',[20 50 scrsz(3)-40 scrsz(4)-100]);
31

```

```

32 maxAngle = 15*pi/180;           % Max coloring angle
33 maxRange = 2*abs(maxAngle);     % The coloring range
34
35 % Define matrices used to draw left and right wing
36 wl(:, :, 2) = [-46, -40, -2
37               -46, -40, -2
38               -46, -40, -2
39               -46, -40, -2
40               -46, -40, -2];
41 wl(:, :, 1) = [ 0, 0, 0
42               17, 23, 54
43               17, 23, 54
44               0, 0, 0
45               0, 0, 0];
46 wl(:, :, 3) = [-2.5, 1, 1
47               -2.5, 1, 1
48               -2.5, -1, -1
49               -2.5, -1, -1
50               -2.5, 1, 1];
51
52
53 wr(:, :, 1) = wl(:, :, 1);
54 wr(:, :, 2) = -wl(:, :, 2);
55 wr(:, :, 3) = wl(:, :, 3);
56
57 aoaNorm = aoa/maxRange+0.5;
58 if aoaNorm > 1
59     aoaNorm = 1;
60 elseif aoaNorm < 0
61     aoaNorm = 0;
62 end
63
64 wlC = [ones(4,1).*0.5 ones(4,1).*aoaNorm];
65 wrC = [ones(4,1).*0.5 ones(4,1).*aoaNorm];
66
67 % Define matrices used to draw the fuselage
68 b(:, :, 2) = [ -2, -2, 2, 2
69               -2, -2, 2, 2
70               -2, -2, 2, 2
71               -2, -2, 2, 2
72               -2, -2, 2, 2
73               -2, -2, 2, 2];
74 b(:, :, 1) = [ -2, -2, -2, -2
75               62, 62, 62, 62
76               71, 71, 71, 71
77               62, 62, 62, 62
78               -2, -2, -2, -2
79               -2, -2, -2, -2];
80 b(:, :, 3) = [-4, 8, 8, -4
81               -4, 8, 8, -4
82               -0, 0, 0, -0
83               -4, -4, -4, -4
84               -4, -4, -4, -4
85               -4, 8, 8, -4];
86 bC = ones(5,3).*0.5;
87
88 % Define matrices used to draw the left and right flap
89 fl(:, :, 2) = [-40, -40, -12, -12
90               -40, -40, -12, -12
91               -40, -40, -12, -12
92               -40, -40, -12, -12
93               -40, -40, -12, -12
94               -40, -40, -12, -12];
95 fl(:, :, 1) = [-3, -3, -4, -4
96               -1, -1, -1, -1
97               0, 0, 0, 0
98               -1, -1, -1, -1
99               -3, -3, -4, -4

```

```

100         -3,   -3,   -4,   -4];
101 fl(:,:,3)=-[ -1,   1,   1,   -1
102             -1,   1,   1,   -1
103              0,   0,   0,   0
104             -1,  -1,  -1,  -1
105             -1,  -1,  -1,  -1
106             -1,   1,   1,  -1];
107
108 fr(:,:,1) = fl(:,:,1);
109 fr(:,:,2) = -fl(:,:,2);
110 fr(:,:,3) = fl(:,:,3);
111
112 % Now rotate the left flap according to the a_fl argument
113 rotMatr = rotateMatrQuat(angleAxis2unitQuat([a_fl, 0, -1, 0]));
114 sfl = size(fl);
115 for i=1:sfl(1)
116     for j=1:sfl(2)
117         fl(i,j,:) = rotMatr*[fl(i,j,1) fl(i,j,2) fl(i,j,3)]';
118     end
119 end
120
121 % Now rotate the right flap according to the a_fr argument
122 rotMatr = rotateMatrQuat(angleAxis2unitQuat([a_fr, 0, -1, 0]));
123 sfr = size(fr);
124 for i=1:sfr(1)
125     for j=1:sfr(2)
126         fr(i,j,:) = rotMatr*[fr(i,j,1) fr(i,j,2) fr(i,j,3)]';
127     end
128 end
129
130 % Now set color on the flaps according to
131 flCnorm = a_fl/maxRange+0.5;
132 if flCnorm > 1
133     flCnorm = 1;
134 elseif flCnorm < 0
135     flCnorm = 0;
136 end
137
138 frCnorm = a_fr/maxRange+0.5;
139 if frCnorm > 1
140     frCnorm = 1;
141 elseif frCnorm < 0
142     frCnorm = 0;
143 end
144
145 flC = ones(5,3).*flCnorm;
146 frC = ones(5,3).*frCnorm;
147
148 % Define matrices used to draw left and right stabilizer
149 el(:,:,2)=[ -2,  -2,  -2,  -2
150            -8.3, -8,   -8,-8.3
151            -8.3,-8.3,-8.3,-8.3
152            -2,  -2,  -2,  -2];
153 el(:,:,1)=[  0,   0,  13,  13
154             0,   0,   3,   3
155             0,   0,   3,   3
156             0,   0,  13,  13];
157 el(:,:,3)=-[ 7.4,  8,   8,  7.4
158             20,  20,  20,  20
159             20,  20,  20,  20
160             7.4, 7.4, 7.4, 7.4];
161
162 er(:,:,1) = el(:,:,1);
163 er(:,:,2) = -el(:,:,2);
164 er(:,:,3) = el(:,:,3);
165
166 elC = ones(3,3)*0.5;
167 erC = ones(3,3)*0.5;

```

```

168
169
170
171 % Define matrices used to draw left and right elevator flap
172 fel(:, :, 2) = [-8.3, -8, -2.5, -2.8
173               -8.3, -8, -2.5, -2.8
174               -8.15, -8.15, -2.65, -2.65
175               -8.3, -8.3, -2.8, -2.8
176               -8.3, -8.3, -2.8, -2.8
177               -8, -8, -2.5, -2.5];
178 fel(:, :, 1) = [-3, -3, -3, -3
179               -1, -1, -1, -1
180               -0, -0, -0, -0
181               -1, -1, -1, -1
182               -3, -3, -3, -3
183               -3, -3, -3, -3];
184 fel(:, :, 3) = [-20, 20, 9, 9
185               20, 20, 9, 9
186               20, 20, 9, 9
187               20, 20, 9, 9
188               20, 20, 9, 9
189               20, 20, 9, 9];
190
191
192 fer(:, :, 1) = fel(:, :, 1);
193 fer(:, :, 2) = -fel(:, :, 2);
194 fer(:, :, 3) = fel(:, :, 3);
195
196
197 % Now rotate the left elevator flap according to the a_el argument
198 %yfix = 26.2/11;
199 rotMatr = rotateMatrQuat(angleAxis2unitQuat([a_el, 0, -5.5, -11]));
200 sfel = size(fel);
201 for i=1:sfel(1)
202     for j=1:sfel(2)
203         fel(i, j, :) = [fel(i, j, 1), fel(i, j, 2)+2.65 fel(i, j, 3)+9]';
204         fel(i, j, :) = rotMatr*[fel(i, j, 1) fel(i, j, 2) fel(i, j, 3)]';
205         fel(i, j, :) = [fel(i, j, 1), fel(i, j, 2)-2.65 fel(i, j, 3)-9]';
206     end
207 end
208
209 % Now rotate the right elevator flap according to the a_er argument
210 rotMatr = rotateMatrQuat(angleAxis2unitQuat([a_er, 0, -5.5, 11]));
211 sfer = size(fer);
212 for i=1:sfer(1)
213     for j=1:sfer(2)
214         fer(i, j, :) = [fer(i, j, 1), fer(i, j, 2)-2.65 fer(i, j, 3)+9]';
215         fer(i, j, :) = rotMatr*[fer(i, j, 1) fer(i, j, 2) fer(i, j, 3)]';
216         fer(i, j, :) = [fer(i, j, 1), fer(i, j, 2)+2.65 fer(i, j, 3)-9]';
217     end
218 end
219
220 felNorm = a_el/maxRange+0.5;
221 if felNorm > 1
222     felNorm = 1;
223 elseif felNorm < 0
224     felNorm = 0;
225 end
226
227 ferNorm = a_er/maxRange+0.5;
228 if ferNorm > 1
229     ferNorm = 1;
230 elseif ferNorm < 0
231     ferNorm = 0;
232 end
233
234 felC = ones(5,3).*felNorm;
235 ferC = ones(5,3).*ferNorm;

```

```

236
237
238 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
239 % Now all actuators are rotated, and colors have been defined. %
240 % Rotate everything according to UAV orientation %
241 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
242
243
244 rotMatrAll = rotateMatrQuat(q);
245
246 downscale = 1.42/92;
247
248 % Right and left wing
249 swl = size(wl);
250 for i=1:swl(1)
251     for j=1:swl(2)
252         wl(i,j,:) = rotMatrAll*[wl(i,j,1)*downscale-r_og(1) ...
253                               wl(i,j,2)*downscale-r_og(2) ...
254                               wl(i,j,3)*downscale-r_og(3)]' + r_og + p';
255         wr(i,j,:) = rotMatrAll*[wr(i,j,1)*downscale-r_og(1) ...
256                               wr(i,j,2)*downscale-r_og(2) ...
257                               wr(i,j,3)*downscale-r_og(3)]' + r_og + p';
258     end
259 end
260
261 sb = size(b);
262 for i=1:sb(1)
263     for j=1:sb(2)
264         b(i,j,:) = rotMatrAll*[b(i,j,1)*downscale-r_og(1) ...
265                               b(i,j,2)*downscale-r_og(2) ...
266                               b(i,j,3)*downscale-r_og(3)]' + r_og + p';
267     end
268 end
269
270 sfl = size(fl);
271 for i=1:sfl(1)
272     for j=1:sfl(2)
273         fl(i,j,:) = rotMatrAll*[fl(i,j,1)*downscale-r_og(1) ...
274                               fl(i,j,2)*downscale-r_og(2) ...
275                               fl(i,j,3)*downscale-r_og(3)]' + r_og + p';
276         fr(i,j,:) = rotMatrAll*[fr(i,j,1)*downscale-r_og(1) ...
277                               fr(i,j,2)*downscale-r_og(2) ...
278                               fr(i,j,3)*downscale-r_og(3)]' + r_og + p';
279     end
280 end
281
282 sel = size(el);
283 for i=1:sel(1)
284     for j=1:sel(2)
285         el(i,j,:) = rotMatrAll*[el(i,j,1)*downscale-r_og(1) ...
286                               el(i,j,2)*downscale-r_og(2) ...
287                               el(i,j,3)*downscale-r_og(3)]' + r_og + p';
288         er(i,j,:) = rotMatrAll*[er(i,j,1)*downscale-r_og(1) ...
289                               er(i,j,2)*downscale-r_og(2) ...
290                               er(i,j,3)*downscale-r_og(3)]' + r_og + p';
291     end
292 end
293
294 sfel = size(fel);
295 for i=1:sfel(1)
296     for j=1:sfel(2)
297         fel(i,j,:) = rotMatrAll*[fel(i,j,1)*downscale-r_og(1) ...
298                               fel(i,j,2)*downscale-r_og(2) ...
299                               fel(i,j,3)*downscale-r_og(3)]' + r_og + p';
300         fer(i,j,:) = rotMatrAll*[fer(i,j,1)*downscale-r_og(1) ...
301                               fer(i,j,2)*downscale-r_og(2) ...
302                               fer(i,j,3)*downscale-r_og(3)]' + r_og + p';
303     end

```

```

304 end
305
306
307 % Draw it all!
308 subplot(splot3d);
309 cmap = [(1:-0.01:0.51)' (0:0.01:0.49)' (0:0.01:0.49)'
310         (0.5:-0.01:0)' (0.5:-0.01:0)' (0.5:0.01:1)'];
311 colormap(cmap);
312 set(gca,'CLim',[0,1])
313
314 hold on;
315 surf(wl(:,:,1), wl(:,:,2), wl(:,:,3), wlC);
316 surf(wr(:,:,1), wr(:,:,2), wr(:,:,3), wrC);
317 surf(b(:,:,1), b(:,:,2), b(:,:,3), bC);
318 surf(fl(:,:,1), fl(:,:,2), fl(:,:,3), flC);
319 surf(fr(:,:,1), fr(:,:,2), fr(:,:,3), frC);
320 surf(el(:,:,1), el(:,:,2), el(:,:,3), elC);
321 surf(er(:,:,1), er(:,:,2), er(:,:,3), erC);
322 surf(fel(:,:,1),fel(:,:,2),fel(:,:,3),felC);
323 surf(fer(:,:,1),fer(:,:,2),fer(:,:,3),ferC);
324
325 if ax(5)<0 && ax(6)>0
326     [X,Y] = meshgrid(ax(1):.3:ax(2),ax(3):.3:ax(4));
327     Z = zeros(size(X));
328     cEarth = ones(size(X)-1).*0.5;
329     surf(X,Y,Z,cEarth);
330 end
331
332 grid on
333 axis equal
334 axis(ax);
335 set(gca,'Xtick',p(1));
336 set(gca,'Ytick',p(2));
337 set(gca,'Ztick',p(3));
338 set(gca,'MinorGridLineStyle','-');
339 set(gca,'XMinorTick','on');
340 set(gca,'YMinorTick','on');
341 set(gca,'ZMinorTick','on');
342 set(gca,'ZDir','rev');
343 set(gca,'YDir','rev');
344 view([-125,30]); % VIEW ANGLE
345 hold off
346 xlabel('N - Norh (x)')
347 ylabel('E - East (y)')
348 zlabel('D - Down (z)')
349 end

```

## B.4 aHorizon.m (Matlab function)

```

1 % Produces an artificial horizon plot
2 % figh = aHorizon(fignr,splotHor,0,0d,0r)
3 %
4 % figh - the figure handle
5 % fignr - the figure handle to plot into
6 % splotHor - the figure subplot to plot into
7 % 0 - UAV orientation vector [n,[roll pitch yaw]]
8 %
9 % A vector of n roll, pitch, and yaw-triplets can be specified
10 % to plot the orientation change of pitch and yaw over time
11 % The current orientation is the last element,
12 % i.e. at [length(0),[roll pitch yaw]]
13 % 0d - UAV desired orientation [roll_d pitch_d yaw_d]
14 % 0r - UAV reference orientation [roll_r pitch_r yaw_r]
15 %

```

```

15 % All angles are specified in [rad]
16 %
17 % Author: Kristoffer Dønnestad, 17.06.2011
18 % Attachment to M.Sc. thesis
19 % Unmanned Vehicle Laboratory
20 % Department of Engineering Cybernetics
21 % Norwegian University of Science and Technology
22 %
23 function figh = aHorizon(fignr,splotHor,0,0d,0r)
24
25 figh = figure(fignr);
26 subplot(splotHor);
27
28 %Plot magenta orientation change history
29 plot(0(:,3).*(180/pi),0(:,2).*(180/pi),'m');
30 hold on
31
32 % Plot current desired orientation
33 r = 0d(1);
34 plane = [cos(r) -sin(r); sin(r) cos(r)]'*[50 -50 -10 0 10; 0 0 20 0 20];
35 plot(0d(3).*(180/pi),0d(2).*(180/pi),'ro');
36 plot(plane(1,1:2)+0d(3).*(180/(pi)),plane(2,1:2)/2+0d(2).*(180/(pi)),'r');
37 plot(plane(1,3:5)+0d(3).*(180/(pi)),plane(2,3:5)/2+0d(2).*(180/(pi)),'r');
38
39 % Plot current reference orientation
40 r = 0r(1);
41 plane = [cos(r) -sin(r); sin(r) cos(r)]'*[50 -50 -10 0 10; 0 0 20 0 20];
42 plot(0r(3).*(180/pi),0r(2).*(180/pi),'go');
43 plot(plane(1,1:2)+0r(3).*(180/(pi)),plane(2,1:2)/2+0r(2).*(180/(pi)),'g');
44 plot(plane(1,3:5)+0r(3).*(180/(pi)),plane(2,3:5)/2+0r(2).*(180/(pi)),'g');
45
46 % Plot current orientation
47 n = length(0);
48 r = 0(n,1);
49 plane = [cos(r) -sin(r); sin(r) cos(r)]'*[50 -50 -10 0 10; 0 0 20 0 20];
50 plot(0(n,3).*(180/pi),0(n,2).*(180/pi),'bo');
51 plot(plane(1,1:2)+0(n,3).*(180/(pi)),plane(2,1:2)/2+0(n,2).*(180/(pi)),'b');
52 plot(plane(1,3:5)+0(n,3).*(180/(pi)),plane(2,3:5)/2+0(n,2).*(180/(pi)),'b');
53
54
55 hold off
56 title('"Artificial horizon"');
57 set(gca,'XTick',[-180 -90 0 90 180]);
58 set(gca,'XTickLabel',{'South' 'West' 'North' 'East' 'South'});
59 ylabel('Pitch [deg]');
60 xlim([-180,180]);
61 ylim([-90,90]);
62 grid on
63 end

```

## B.5 RecceCompass.m (Matlab function)

```

1 % Produces a compass plot showing a silhouette of Recce D6
2 % figh = RecceCompass(cheading,desheading,refheading,wind)
3 %
4 % cheading - The current heading (blue) [rad]
5 % desheading - The desired heading (red) [rad]
6 % refheading - The reference heading (green) [rad]
7 % wind - A vector representing the wind [w_x w_y w_z]
8 %
9 % Outputs the figure handle
10 %
11 % Author: Kristoffer Dønnestad, 17.06.2011

```



```

12 % Attachment to M.Sc. thesis
13 % Unmanned Vehicle Laboratory
14 % Department of Engineering Cybernetics
15 % Norwegian University of Science and Technology
16 %
17 function figh = RecceCompass(cheading, desheading, refheading, wind)
18 figh = figure();
19
20 air = cartesian2air(wind);
21 Wstr = air(1);
22 Wbeta = air(3);
23
24 circle = [sin(0:2*pi/90:2*pi)' cos(0:2*pi/90:2*pi)'];
25 Windstrength = [0.3 1.6 3.4 5.5 8 10.8 13.9 17.2 20.8 23.5 28.5
26 32.7];
27 BeaufortScale = {...
28     'Calm'
29     'Light air'
30     'Light breeze'
31     'Gentle breeze'
32     'Moderate breeze'
33     'Fresh breeze'
34     'Strong breeze'
35     'Near gale'
36     'Gale'
37     'Storm'
38     'Violent storm'
39     'Hurricane'
40     };
41
42 % Define aircraft silhouette:
43 silfus = [ -.8 .8 .8 5 5 .8 .8 0 -.8 -.8 -5 -5 -.8 -.8 ;
44     -4.5 -4.5 -4 -4 -1 4 5 7 5 4 -1 -4 -4 -4.5
45     ]';
46
47 % Define unit wind arrow silhouette:
48 silarrow = [ -.1 .1 .1 .3 0 -.3 -.1 -.1 ;
49     0 0 .6 .6 1 .6 .6 0 ]';
50
51 desreftap = [2.5*silarrow(:,1)'
52     -2.5*silarrow(:,2)'+9.5]';
53
54 silfusAct = zeros(size(silfus));
55 for siln = 1:length(silfus)
56     silfusAct(siln,:) = [cos(cheading) sin(cheading) ; -sin(cheading)
57         cos(cheading)]*silfus(siln,:);
58 end
59
60 silfusDes = zeros(size(desreftap));
61 silfusRef = zeros(size(desreftap));
62 for siln = 1:length(desreftap)
63     silfusDes(siln,:) = [cos(desheading) sin(desheading) ; -sin(
64         desheading) cos(desheading)]*desreftap(siln,:);
65     silfusRef(siln,:) = [cos(refheading) sin(refheading) ; -sin(
66         refheading) cos(refheading)]*desreftap(siln,:);
67 end
68
69 for siln = 1:length(silarrow)
70     silarrow(siln,:) = Wstr*[cos(Wbeta) sin(Wbeta) ; -sin(Wbeta) cos(
71         Wbeta)]*silarrow(siln,:);
72 end
73
74 plot(circle(:,1)*7, circle(:,2)*7);
75 hold on

```

```

74     plot(circle(:,1)*.3, circle(:,2)*.3);
75     plot([0 0],[6.5 7.5]);
76     plot([0 0],[-6.5 -7.5]);
77     plot([6.5 7.5],[0 0]);
78     plot([-6.5 -7.5],[0 0]);
79
80     plot(silfusDes(:,1),silfusDes(:,2),'r');
81     plot(silfusRef(:,1),silfusRef(:,2),'g');
82     plot(silfusAct(:,1),silfusAct(:,2),'b');
83
84     plot(silarrow(:,1),silarrow(:,2),'m');
85
86     hold off
87     axis([-10 10 -10 10]);
88     axis equal;
89     set(gca,'ytick',[]);
90     set(gca,'xtick',[]);
91
92
93     for ws = 1:length(Windstrength)
94         BeaufortIndex = ws;
95         if Wstr < Windstrength(ws)
96             break;
97         end
98     end
99
100    title('Compass')
101
102    text(0.4,7.8,['North']);
103    text(7.8,0.4,['East']);
104    text(-11.2,0.4,['West']);
105    text(0.4,-7.8,['South']);
106
107
108    WstrStr = num2str(Wstr);
109    if length(WstrStr) > 4
110        WstrStr = WstrStr(1:4);
111    end
112
113    text(-12.5,-4.5, ['WIND:']);
114    text(-12.5,-6, [ num2str(round(Wbeta*180/pi)) '\circ' ]);
115    text(-12.5,-7.5, [ WstrStr ' m/s' ]);
116    text(-12.5,-9, [ BeaufortScale(BeaufortIndex)]);
117
118    text(-12.5,9, ['AIRCRAFT:']);
119    text(-12.5,7.5, [num2str(round(cheading*180/pi)) '\circ' ]);

```

# Appendix C

## Digital attachment

Root folder:

- `CaseStudys.avi`  
The video discussed under Case Studies
- `coloringEx.avi`  
An example video showing the coloring scheme of the 3d plot.
- `refModelEx.avi`  
An example video showing the heading reference model.

Avi folder: Empty folder prepared for video output.

Matlab folder:

- All the developed Matlab code, and Simulink model.

3rd Party Software folder:

- `AVI_Joiner`  
The (free) software used for joining multiple .avi files
- `Any Video Converter`  
The (free) software used to compress the .avi files.
- `mpgwrite`  
MATLAB Movie to MPEG Converter. (Free)
- `VLC`  
A (free) media player very much suitable to playback the example videos.



## Appendix D

# Recce D6 documentation from Odin Aero

All the received documentation of Recce D6 provided by Odin Aero is listed here. This is all the information about the air vehicle that has been available through the entire thesis work.

The following documentation is the property of Odin Aero AS, and may not be used or copied without the owners approval.

**Title/Name:** RECCE D6

**Use(s):** Reconnaissance and surveillance

**Manufacturer and Country:** Odin Aero AS, Norway

**Powerplant:** Type: 200W Brushless motor, Powered by LiPol battery

**Performance:** Speed 85km/h, Endurance 1 hour at cruising speed 65km/h, Mission Radius 10km, Ceiling 1000ft.

**Dimensions:** Length 1.06m, Height 0.26, Wingspan 1.42m

**Weights:** MTOW 2.8kg Payload 0.5kg

**Datalink:** RF uplink/downlink, real time video downlink

**Guidance/Tracking:** Remote control and GPS auto-navigation (Neural Network Adaptive Control), laptop computer mapping

**Payload:** CCD video camera, IR camera and other various options

**Electrical Power:** 12 VDC

**Launch:** Hand launch/ unprepared terrain

**Recovery:** Skid landing

**Structure Material:** Composite

**Ground Control Station:** Laptop computer, RF datalink

**Battery:** Thunder Power Li-Polymer 8000mAh/ 11,1V Pro Lite Series.

**Motor:** Plettenberg motor Orbit 10-22

**Propeller:** Graupner CFK Folding prop 13-7"

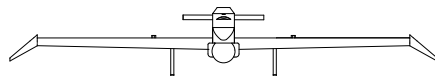
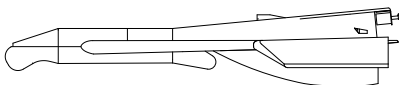
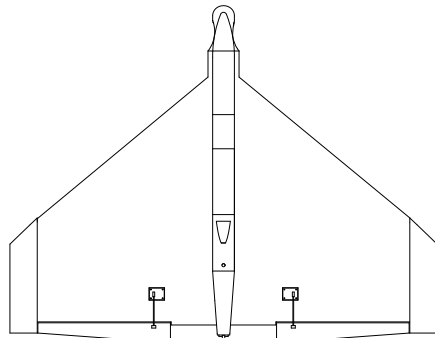
**Speed controller:** Electrify SS-35 Brushless Esc 5V/ 2A BEC 2-4 LiPo

**Elevon servos:** Graupner/ JR C 271 No. 5107

**Rudder servo:** Robbe FS 500 MG No. 8431

**Receiver:** Futaba R 617 FS 2,4GHz

**Transmitter:** Futaba T7C 2,4GHz



## Appendix E

# Servo motor specifications

The servo motor transit speeds are found from some retailer websites.

O N L I N E

**robbe**  
modellSPORT

Deutsch  
English  
Français

Italiano  
Español

Search:


Registration:

[Initial registration](#)

Selection:

- [Important new items](#)
- [Special offers](#)
- [Promotional items](#)
- [Model aircraft](#)
- [Model helicopters](#)
- [Model boats](#)
- [RC cars](#)
- [Radio control systems](#)
- [Servos and gyros](#)
- [Electric accessories](#)
- [Drive motors](#)
- [Motor accessories](#)
- [Finishing materials](#)
- [Materials](#)

[Home](#) [Help](#) [Basket](#) [GIB](#) [Contact](#)



Servo FS 500 Mg Micro  
1-8431

To order:	
€:28.00	
€:23.33	<input type="button" value="add to basket"/>

**Small, fast, high-torque wing-mounting servo of micro size.** Ideal for wing-mounting, even with thin airfoils. **The gearbox is all-metal, from the motor pinion to the output gear.** Ideal for large-span gliders, whose control surfaces tend to destroy the teeth of plastic gears. Also a good choice for trucks and multi-function boats.

**The FS 500 MG micro-servo is supplied with a universal JR connector.**

**Specification:**

Dimensions: 29.0 x 13.5 x 29.5 mm	
Mass/weight: 21.00 pond/g	
Operating voltage: 4.8 - 6 Volts	
Nominal voltage: 4.80 Volts	Nominal voltage: 6.00 Volts
Torque: 24.00 Ncm	Torque: 30.00 Ncm
Transit speed: 0.12 Sec/45°	Transit speed: 0.10 Sec/45°

[CPSnet](#)  
Online-Service

more

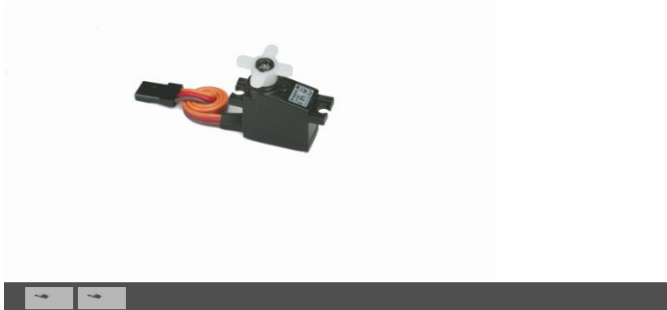




[Home](#) [About Graupner](#) [Products](#) [News](#) [Press](#) [Service](#) [Links](#)

> [Products](#) > [RC range](#) > [Servos, gyros](#) > [Analog servos](#) > [SERVO C271 \(METAL GEAR\)](#)

### SERVO C271 (METAL GEAR)



### SERVO C271 (METAL GEAR)

Order number 5107  
Price: 48,90 €

This item is available



**SERVO C 261 (MICRO POWER)**



**servo C 341 bulk**

#### Characteristics

#### Specification

Dimensions (LxWxH)	21 x 11 x 21 mm
Operating voltage	4,8...6 V
Angular travel incl. trim, approx.	2 x 45 °
Replacement gearbox	5107.2
Gear unit	Getriebe mg
All-up weight, approx.	13 g
Bearings	Lager pb
No-load current drain, approx.	8 mA
Torque at 6.0V, approx.	20 Ncm
Transit speed at 6.0 V, approx.	0,12 Sek/40°
Charging rate at 6.0V approx.	560 mA

[Imprint](#) | [Terms and Conditions](#)

Search

Graupner Shops

[Dealer login](#)  
[Login Repair Service](#)

[Factory outlet - online](#)

Newsletter

[Register for New sletter](#)  
[Cancel New sletter](#)  
[Change profile](#)

Catalogue/Flyer

[Price list](#)  
[New items 2011](#)  
[Tangent Catalogue 2011](#)  
[Main catalogue 51FS 2009](#)  
[GM-Racing RC-Cars 2011](#)  
[GM-Racing Catalogue 2009](#)

Websites

[IFS-2,4 GHz System](#)  
[Tangent Modelltechnik](#)  
[Office Cup](#)  
[GM-Racing](#)  
[Robotics](#)



[Like](#)