NTNU – Trondheim
Norwegian University of
Science and Technology

Autopilot Design for Unmanned Aerial Vehicles

Ingrid Hagen Johansen

Master of Science in Engineering Cybernetics
Submission date: June 2012
Supervisor: Thor Inge Fossen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

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

MSC THESIS DESCRIPTION SHEET

Name:	Ingrid Hagen Johansen
Department:	Engineering Cybernetics
Thesis title (Norwegian):	Autopilot design for ubemannede fly
Thesis title (English):	Autopilot Design for Unmanned Aerial Vehicles

Thesis Description: The purpose of the thesis is to investigate different methods for unmanned aerial vehicles (UAV) autopilot design. This includes path-generation, path-following control and regulation. Computer simulations should be used to evaluate the performance of the different guidance-controller systems.

The following items should be considered:

1. Literature study on UAV flight control systems. Give an overview of different methods found in the literature for takeoff, altitude control and turning control. Both decoupled and MIMO design techniques should be reviewed.
2. Define the scope of the thesis and clarify what your contributions are.
3. Choose an UAV model for computer simulations and graphical visualization in X-plane.
4. Design autopilot systems for automatic takeoff, altitude control and turning control.
5. Design guidance system for path-following.
6. Investigate how good the autopilot system is for coupled maneuvers, varying payload and wind disturbances.
7. Conclude your results.

Start date:	2012-01-16
Due date:	2012-06-11

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

Abstract

This thesis will present a design of a guidance and control system to use on aircrafts, primarily on UAVs. One control method for heading control and two for pitch and altitude control will be investigated. The control methods are Proportional-Integral-Derivative (PID) and sliding mode control. PID will be tested on both heading and pitch and altitude control, while sliding mode will only be applied to pitch and altitude.

There will be presented a path-following method, Line of Sight, for heading guidance and a kinematic controller for altitude reference.

The presented methods are implemented in Matlab Simulink while the aircraft model used comes from the flight simulator X-Plane. X-Plane is also used to visualize the performance of the autopilot design.

PID and sliding mode control are tested in four different scenarios to investigate which controller who has the best performance. After the simulations, it was observed that the PID had better performances than sliding mode control.

Sammendrag

I denne oppgaven vil det bli presentert et styrings- og kontrol-system for bruk i en autopilot for fly, først og fremst for ubemannede fly. Det vil være en regulatormetode for posisjon og to metoder for høyderegulering. Reguleringsmetodene som vil bli presentert er Proporsjonal-Integral-Derivasjons(PID)-regulering og sliding mode-regulering. PID vil bli brukt for både posisjon og høyde regulering, mens sliding mode kun vil bli brukt for høyde.

Et styringssystem er designet for å planlegge en bane og en høydereferanse som autopiloten skal følge. Line of Sight er brukt for å planlegge en bane i det horisontale plan, nord-øst, mens en kinematisk kontroller gir høydereferanse.

Metodene vil så bli implementert i Matlab Simulink, mens flymodellen kommer fra flysimuleringsprogrammet X-Plane. X-Plane blir også brukt for å visualisere oppførselen til autopiloten.

PID og sliding mode vil til slutt bli testet i fire ulike scenarioer for å se hvilken regulator som har den beste oppførselen. Etter simuleringene kommer det fram at PID har en bedre oppførsel enn sliding mode.

Acknowledgements

This thesis, and the work it presents, is the culmination of my masters degree at the Department of Engineering Cybernetics of the Norwegian University of Science and Technology (NTNU). I would like to thank my supervisor Thor I. Fossen at the Department of Engineering Cybernetics for his patience and his guidance for this report and in the field of Guidance, Navigation and Control Systems.

Also a great thank you to Kjetil Hope Tufteland and Kåre Vistnes for good discussions, input and feedback. Thanks goes also to all the members of the unmanned vehicle laboratory the last couple of months for making a good working environment. Finally, I must thank to Morten Wollert Nygren for corrective reading, and Torleif Hagen for being such a good help when it comes to flight details for Cessna.

Contents

List of Abbreviations

AUV	Autonomous Underwater Vehicle
GNC	Guidance, Navigation and Control
HIL	Hardware in the Loop
LOS	Line of Sight
LP	Low Pass
NED	North-East-Down
NED	North-East-Down
PID	Proportional-Integral-Derivative
SMC	Sliding Mode Control
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
UDP	User Datagram Protocol

List of Figures

List of Tables

Chapter 1

Introduction

There are many different opinions of what an Unmanned Aerial Vehicle (UAV) is. Many believe that they are only used for military purposes, but this is far from the truth. The most important definition of an UAV is that it is an aerial vehicle without a pilot, Unmanned Aerial Vehicle System Association [2012b]. Without pilot means that the aerial vehicle do not have a pilot on board nor a pilot on the ground in a control center, referred to as a Ground Control Station. If the aerial vehicle has a pilot on ground who can communicate with the vehicle, it is referred to as a Unmanned Aerial System (UAS). The UAV is preprogrammed and is supposed to do the operation and come back without human interference, while an UAS is remotely operated during the operation.

In the dictionary, the UAV is defined as: *A powered, aerial vehicle that does not carry a human operator, uses aerodynamic forces to provide vehicle lift, can fly autonomously, can be expendable or recoverable, and can carry a lethal or nonlethal payload. Ballistic or semiballistic vehicles, cruise missiles, and artillery projectiles are not considered unmanned aerial vehicles. Also called UAV.* The Free Dictionary [2012].

1.1 Motivation

Originally the UAVs were designed for military operations, but in these days they are designed for nonmilitary operations as well. The UAVs can be used to:

1. Monitoring the oceans, e.g. how the ice is moving nearby an oil platform
2. Search and rescue operations
3. Dangerous, risky and dull operations

and many other task where it is not necessary to have a human pilot on board. The UAV technology will not be applied in commercial aviation industry in the near future as it is hard for people to trust a machine without having a human pilot observing.

There are many advantages for using UAVs, such as:

1. Low cost
2. No need for qualified pilots
3. Save time by using two or more UAVs at the same time
4. No need to make human considerations when designing the UAV
5. Can operate in areas that are dangerous for humans
6. Have long operation time without loosing precision

Unmanned Aerial Vehicle System Association [2012a]. Figure 1.1 illustrates how an UAV can be used for searching in open seas. By having many UAVs working together in a search operations, lives can be saved.

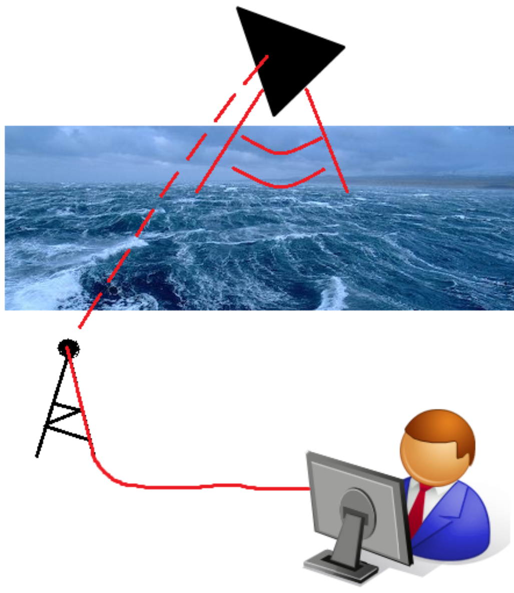

Figure 1.1: UAV used in search and rescue operations, Johansen [2011]

However, there are some disadvantages of using an UAV and some of them are listed here:

1. Limitation on payload
2. Can loose contact with ground base, must have a backup plan
3. Easy to crash
4. A pilot can monitor wider areas

1.2 What has been done

UAV

In wartime, many great inventions are made. This holds for UAVs too. During the American Civil War (1861-1865) the first unmanned aerial vehicle was tested. It was a balloon that carried explosives and dropped its payload after a time-delay. Wind and weather made it difficult to determine the time-delay and the balloon was never a success.

Various radio-controlled unmanned aircrafts were tested during World War I, but did not pass the testing phase before the war ended.

The British Royal Navy used a primitive radio-controlled UAV, the Queen Bee, as aerial target practice in the 1930s. It could be landed and reused and reach a speed of 160km/h.

During World War II, the Nazis developed an UAV that could reach a speed of 804km/h, carry 907kg of explosives and travel 241km, the Revenge Weapon 1. It was used to attack nonmilitary targets and killed more than 900 civilians while injuring 35.000.

In the 1960s and 70s, the United State developed UAVs that were launched from a plane and remotely controlled by operators within the plane. Later in the 1970s and 80s, Israel developed smaller UAVs. They could transmit live video with a 360-degree view. Since they were small they were inexpensive to produce and difficult to shoot down.

Although the UAV technology has gone through a huge development throughout the 20th century, it was first in the 1990s that it got its big breakthrough with the Predator made by the U.S. Department of Defense, How Stuff Works [2012].

The first UAVs was programmed to fly in a straight line or a circle until it ran out of fuel and fell down, much the same as drones do now. Later they got radio communications and could remotely operate the UAV, and now they are preprogrammed with onboard control and guidance systems. The goal is to create UAVs that make decisions by themselves, without human interference.

Autopilot

To make an UAV fly it needs an autopilot. There are different kinds of autopilots, from the simple ones used in small private boats to more complex systems used on for instance submarines, oil tankers and aircrafts. The first attempt at an autopilot was a ship and airplane stabilizer in 1914 by Elmer Sperry (1860-1930) and his company Sperry Gyroscope Company, IEEE Global History Network [2012]. Sperry and Nicholas Minorsky (1855-1970), who formulated the *Proportional-Integral-Derivative* control Bennett [1984], worked together and maid a huge contribution to the autopilot design with their steering mechanism and further works.

Autopilots are today used in simple operations such as to keep a heading, and for more complex operations such as turning, docking, and keeping control of unstable vessels like submarines and some big oil tankers. They are used in boats, submarines, torpedoes, missiles, rockets, spaceships etc. and is an important part of everything which is supposed to be unmanned.

Guidance

The guidance system is an important part of an autopilot. It determines a path to follow based on commanded signals, such as waypoints, altitude, speed, etc. given by an operator.

Guidance methods made for marine crafts can easily be applied to aerial vehicles, especially methods for autonomous underwater vehicles (AUV) since they too operate in 6 degrees of freedom. Børhaug and Pettersen [2005] uses Line of Sight method for cross-track control for underactuated autonomous vehicles on an AUV and Breivik and Fossen [2008] proposes different guidance laws for AUVs. These are all based on straight lines between commanded waypoints. If the path does not consists of straight lines, the path has to be parametrized and a kinematic controller can be applied, Skjetne et al. [2003] and Fossen [2011b].

Control

When it comes to the control system in an autopilot, the choices are many. The most used controller in the industry is the three term, *proportional-integral-derivative (PID)*, controller which dates back to 1890s, with the first practical example from 1911 by Elmer Sperry, Bennett [1984]. This is a linear controller, but it has been used on nonlinear systems such as for an UAV quadrotor Salih et al. [2010] and for an UAV fixed-wing Albaker and Rahim [2011]. These are decoupled controllers, giving you different controllers for speed, altitude, pitch angle, heading angle, etc. For basic PID theory see Balchen et al. [2003] and PID control for marine craft Fossen [2011b].

Sliding mode control is another widely used controller. This is a nonlinear method and will be more robust when used on a nonlinear system.

In UAV autopilot design, sliding mode, decoupled and coupled design techniques have been used; Carletti et al. [2011], Healey and Lienard [1993] and Jones et al. [2009]. Basic theory of sliding mode can be found in Utkin [1992] and Khalil [2002], and applied on marine craft in Fossen [2011b].

1.3 My Contribution

In this thesis I will propose an autopilot design for automatic takeoff, altitude control and turning control. The autopilot will consist of a guidance system and two different controllers which will be tested against each other. The first will be a PID controller, much used in the industry, and the second a Sliding Mode controller. The last controller is nonlinear and more robust than the PID. Sliding mode will only be used for pitch and altitude control, while PID will take care of turning control as well as pitch and altitude. The autopilot design is supposed to handle varying payload and wind. The guidance system will make a path for the UAV to follow based on waypoints given.

The outline of this thesis is as follows: In Chapter 2 the theory of guidance system will be presented. Control system theory is described in Chapter 3, while the design of the overall guidance, navigation and control system is in Chapter 4. Simulation studies and results for the different control strategies proposed is presented in Chapter 5. Conclusion and suggestions for further work will be found in Chapter 6.

Chapter 2

Guidance Theory

To make an autopilot, it is necessary to have a good guidance system. There are many ways of making a guidance system and it is advantageous to know what the autopilot is supposed to do when choosing the method. Is it supposed to track a target which is moving or standing still, is it supposed to track a trajectory or just follow a predefined path? Tracking a target or a trajectory is often time dependent, while a path-following method is time independent.

In this thesis a path-following method for guidance is used. Since this autopilot is supposed to control an aerial vehicle, the guidance system must handle three dimensions. It will be decoupled into two parts, one for horizontal motion, North-East, and one for vertical motion, altitude. The guidance system will be making the path to follow in North-East-Down (NED) coordinates, where x is North, y is East and z is down, or altitude, see Appendix A. The altitude is the negative of down.

2.1 Line of Sight Guidance for Path-Following

Line of Sight (LOS) is a guidance method which can be used for all three scenarios presented above; target tracking, trajectory tracking and path-following. The method is classified as a three-point guidance scheme, since it involves a reference point, the UAV and a target or a setpoint the UAV is supposed to go to. The name *Line of Sight* comes from the principle of the method. It is supposed to follow the line, LOS vector, from the reference point to the target, illustrated in Figure 2.1. LOS is being used for guidance in horizontal plane, North-East, in this thesis.

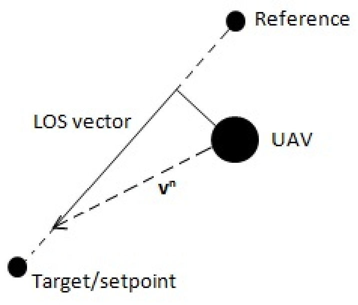

Figure 2.1: Illustration of Line of Sight principle

Consider two waypoints $p_k = [x_k \quad y_k]^\top \in \mathbb{R}^2$ and $p_{k+1} = [x_{k+1} \quad y_{k+1}]^\top \in \mathbb{R}^2$ and a straight line between them, Figure 2.2. The aim is to make the aircraft follow this straight line, by making the cross-track error e as small as possible:

$$\lim_{t\to\infty} e(t) = 0 \tag{2.1}$$

where the cross-track error is given by:

$$e(t) = -[x(t) - x_k]\sin(\alpha_k) + [y(t) - y_k]\cos(\alpha_k) \tag{2.2}$$

and the angle, α_k, used to rotate the north-axis in the path-fixed reference frame with origin in $\mathbf{p}_k^n$, in Figure 2.2, is written as:

$$\alpha_k := \text{atan2}(y_{k+1} - y_k, x_{k+1} - x_k) \in \mathbb{S} := [-\pi, \pi] \tag{2.3}$$

To ensure that the cross-track error $e(t) \to 0$ for both cases, enclosure-based or lookahead-based steering guidance principles can be used. Since the lookahead-based method has several advantages over the enclosure-based method, Fossen [2011b], a lookahead-based approach will be used in this thesis.

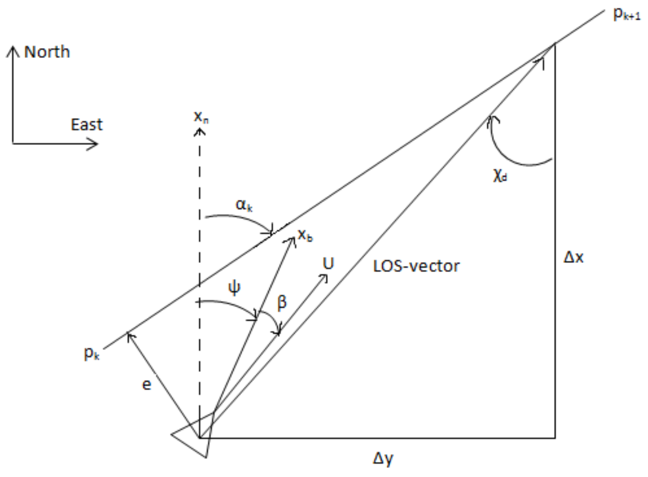

Figure 2.2: Line of sight, Johansen [2011]

Lookahead-Based steering

The lookahead-based approach is a very simple approach and is combined of two parts:

$$\chi_d(e) = \chi_p + \chi_r(e) \tag{2.4}$$

where

$$\chi_p = \alpha_k \tag{2.5}$$

from Equation 2.3. χ_p is called the *path-tangential angle*. The *velocity-path relative angle* $\chi_r(e)$ can be implemented as a control law with only a proportional action:

$$\chi_r(e) = \arctan(-K_p e) \tag{2.6}$$

where $K_p(t) = 1/\Delta(t) > 0$ and Δ is the lookahead distance.

The control system uses heading angle ψ instead of course angle χ and it is necessary to transform the course angle to the heading angle by:

$$\psi_d = \chi_d - \beta \tag{2.7}$$

where β is the sideslip angle which can be computed by:

$$\beta = \arcsin(\frac{v}{U}) \tag{2.8}$$

if the velocities of the aircraft are measured. v is the velocity in East direction and

$$U = \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2} \geq 0 \tag{2.9}$$

where x and y corresponds to north and east.

Circle of Acceptance

Circle of acceptance is a method for knowing when to switch to the next waypoint, from $[x_k \quad y_k]^\top \in \mathbb{R}^2$ to $[x_{k+1} \quad y_{k+1}]^\top \in \mathbb{R}^2$, where x is north and y is east from the NED coordinate frame. When the aircraft is inside a circle with radius $R_{k+1} \in \mathbb{R}^1$ around the point $[x_{k+1} \quad y_{k+1}]^\top$, as can be seen in Figure 2.3, the guidance system should change to the next waypoint. The position of the aircraft has to satisfy:

$$[x_{k+1} - x(t)]^2 + [y_{k+1} - y(t)]^2 \leq R_{k+1}^2 \tag{2.10}$$

at time t to change the waypoint, Fossen [2011b].

Figure 2.3 shows the circle of acceptance principle in the two dimensional plane.

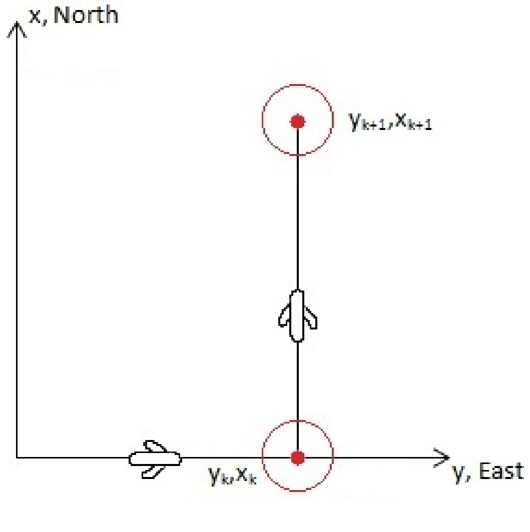

Figure 2.3: Navigation in xy plane with circle of acceptance

2.2 Kinematic Control for Altitude Guidance

Guidance control for altitude is based on the differential equation for altitude control, Fossen [2011a] :

$$\begin{aligned}\dot{h} &= U\gamma \\ &= U(\theta - \alpha)\end{aligned} \tag{2.11}$$

where h is the altitude, U is the speed, either calculated as in Equation 2.9 or measured, α is the angle of attack, measured or calculated as $\alpha = \tan^{-1}(w/u)$ and θ is measured pitch angle. $\gamma = (\theta - \alpha)$ is called the *flight path*. Do not mix this α with the α_k in LOS.

Transforming Equation 2.11 to the desired signal we get:

$$\dot{h}_d = U(\theta_d - \alpha) \tag{2.12}$$

The aim is to reduce the difference between measured altitude and desired altitude to zero, $h_d - h \to 0$, by feeding the control system desired pitch angle. This can be done by applying a P-controller, more described in Chapter 3, such that the desired pitch angle θ_d becomes:

$$\theta_d = \frac{1}{U}(\dot{h}_d - K_p(h_d - h)) + \alpha \tag{2.13}$$

K_p is a controller gain and is given by $K_p = 2\lambda > 0$ where λ is a design parameter. h_d is given by the reference model which will be described in Section 4.1. This kinematic controller is illustrated in a block diagram in Figure 2.4.

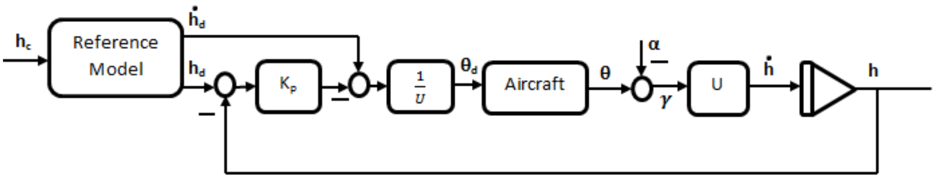

Figure 2.4: Block diagram of a kinematic controller

Chapter 3

Control Theory

An UAV is to be controlled in this master thesis. In the simulation program being used, X-Plane, there are no UAV flight model. In stead, the control system designed in this master thesis is applied on a Cessna 172SP. This Cessna has four control surfaces given in Table 3.1 and illustrated in Figure 3.1.

Table 3.1: Control surfaces

Control Surface	**Action**	
Motor	Speed forward	Surge
Aileron	Banked Turn	Roll
Elevator	Takeoff	Pitch
Rudder	Turning	Yaw

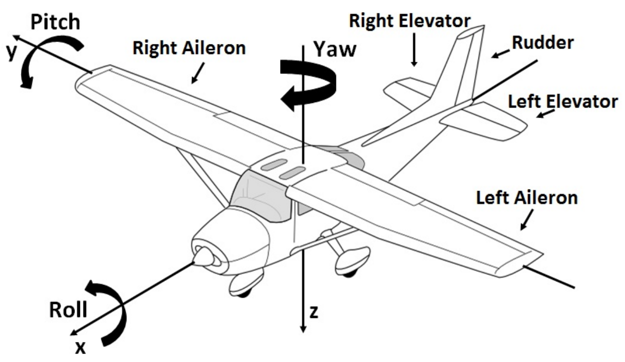

Figure 3.1: Control surfaces on Cessna 172SP

3.1 PID Control

In this thesis two different control systems will be designed. The first one is a regular Proportional-Integral-Derivative (PID) controller. PID is the most widely used controller in the industry because it is easy to implement and maintain. The controller is linear and is here applied to a highly nonlinear system, but it will work nonetheless.

The aim of a PID controller is to make the error of the signal, the difference between wanted signal and actual signal, as small as possible, i.e. go to zero, by making control signals to the process:

$$\lim_{t\to\infty} e = \lim_{t\to\infty} x_d - x \to 0 \tag{3.1}$$

This can be done with four different controllers, Balchen et al. [2003]

- Proportional (P) controller
- Proportional-Integral (PI) controller
- Proportional-Derivative (PD) controller
- Proportional-Integral-Derivative (PID) controller

All these proposed controllers can be used, based on what kind of behavior that is wanted.

Mathematically, a PID controller can be described as:

$$\tau = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{\mathrm{d}}{\mathrm{d}t} e(t) \tag{3.2}$$

where K_p, K_i and K_d represent proportional, integral and derivative gains respectively. By setting one or two of these to zero, you get a P, PI or PD controller. τ is the control signal, which will be sent to the process.

The proportional term gives an output that is proportional to the error. Too high proportional gain K_p can give an unstable process.

The integral term is proportional to both the duration of the error and the magnitude of it. The integral term deals with steady-state error by accelerate the movement of the process towards setpoint. It can contribute to an overshoot because it responds to accumulated error from the past which can be solved by adding the derivative term.

The derivative term slows down the rate of change of the control signal and makes the overshoot smaller. The combined controller-process stability is improved by the derivative term, but it could make the process unstable because it is sensitive to noise in the error signal, Wikipedia - PID [2012].

Figure 3.2 shows a block diagram of a regular PID controller.

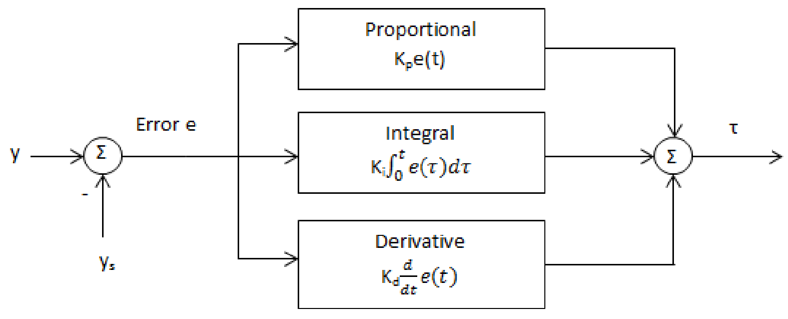

Figure 3.2: Block diagram of a PID controller, Johansen [2011]

Stability

Stability of a PID controller is maintained by tuning the K_p, K_i and K_d gains properly. They are to be tuned such that the error converges to zero. To find this values for K_p, K_i and K_d, test the control system with the process it is supposed to control, but without the guidance system, and have a constant desired signal. When the error converges to zero within a reasonable time and without to high overshoot and oscillations, the controller is stable.

The Figure 3.3 shows different stabilities of PID.

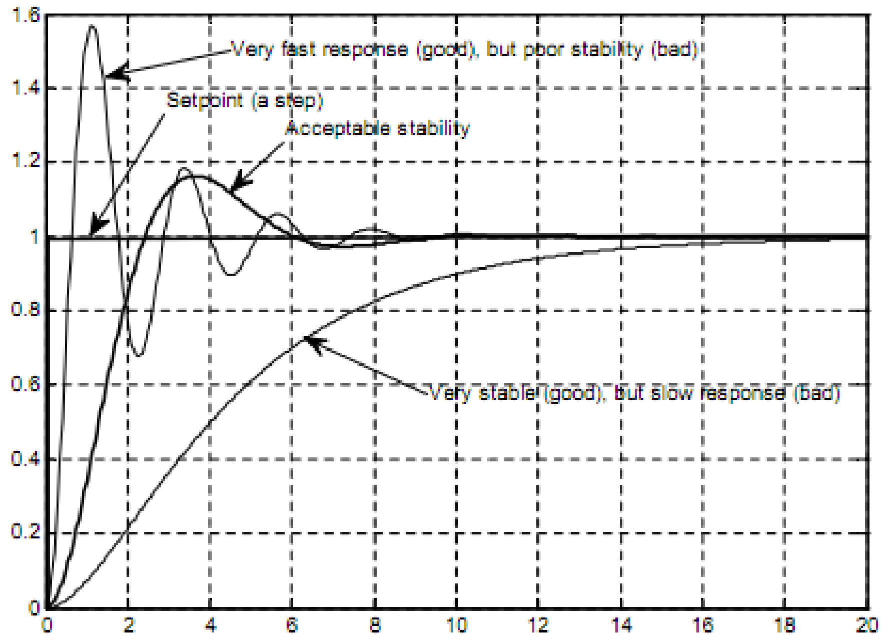

Figure 3.3: Stability of PID controller, Johansen [2011]

3.2 Sliding Mode Control

Sliding Mode Control (SMC) is a robust-nonlinear controller, much used on marine vehicles, Fossen [2011b]. Since marine vehicles and aerial vehicles can be described by the same equations, there should be no problem in using SMC in this autopilot, based on equations in Fossen [2011b].

Sliding mode control in this thesis is proposed as the eigenvalue decomposition method. This method is based on the linearized maneuvering model:

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{N}(u_0)\nu_{\mathbf{r}} = \mathbf{b}\delta \tag{3.3}$$

To explain sliding mode control consider the state-space model:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u + \mathbf{f}(\mathbf{x},t) \tag{3.4}$$

where $\mathbf{x}$ is selected states from $\eta = \begin{bmatrix} N & E & D & \phi & \theta & \psi \end{bmatrix}^T \in \mathbb{R}^6$ and $\nu = \begin{bmatrix} U & V & W & P & Q & R \end{bmatrix}^T \in \mathbb{R}^6$, dependent of what should be controlled. u is the control signal; motor, rudder, aileron or elevator, δ_M, δ_R, δ_A or δ_E respectively. $\mathbf{f}(\mathbf{x},t)$ is a nonlinear function describing the deviation from linearity in terms of disturbances and unmodeled dynamics, Fossen [2011b].

Let $x = \begin{bmatrix} Q & \theta & Z \end{bmatrix}^T$, where $Z = D$, and $u = \delta_E$, A and B matrix becomes:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & 0 \\ 1 & 0 & 0 \\ 0 & U_0 & 0 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} b_1 \\ 0 \\ 0 \end{bmatrix} \tag{3.5}$$

for pitch and altitude control. The feedback control law is written as:

$$u = -\mathbf{k}^\top \mathbf{x} + u_0 \tag{3.6}$$

where $\mathbf{k} \in \mathbb{R}^3$ is feedback gain vector, computed by pole placement. By substituting Equation 3.6 into Equation 3.4 we get:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}(-\mathbf{k}^\top \mathbf{x} + u_0) + \mathbf{f}(\mathbf{x},t) \\ &= (\mathbf{A} - \mathbf{B}\mathbf{k}^\top)\mathbf{x} + \mathbf{B}u_0 + \mathbf{f}(\mathbf{x},t) \\ &= \mathbf{A_c}\mathbf{x} + \mathbf{B}u_0 + \mathbf{f}(\mathbf{x},t) \end{aligned} \tag{3.7}$$

To find a good control law, define the sliding surface

$$s = \mathbf{h}^\top \tilde{\mathbf{x}} \tag{3.8}$$

with $\mathbf{h} \in \mathbb{R}^3$ as a design vector, chosen to make $s \to 0$, which implies that $\tilde{\mathbf{x}} = \mathbf{x} - \mathbf{x_d} \to \mathbf{0}$ and its derivative:

$$\begin{aligned} \dot{s} &= \mathbf{h}^\top \dot{\tilde{\mathbf{x}}} = \mathbf{h}^\top (\dot{\mathbf{x}} - \dot{\mathbf{x}}_\mathbf{d}) \\ &= \mathbf{h}^\top \dot{\mathbf{x}} - \mathbf{h}^\top \dot{\mathbf{x}}_\mathbf{d} \\ &= \mathbf{h}^\top (\mathbf{A_c x} + \mathbf{B} u_0 + \mathbf{f}(\mathbf{x},t)) - \mathbf{h}^\top \dot{\mathbf{x}}_\mathbf{d} \\ &= \mathbf{h}^\top \mathbf{A_c x} + \mathbf{h}^\top \mathbf{B} u_0 + \mathbf{h}^\top \mathbf{f}(\mathbf{x},t) - \mathbf{h}^\top \dot{\mathbf{x}}_\mathbf{d} \end{aligned} \tag{3.9}$$

by applying Equation 3.7 to the derivative of s.

Assuming $\mathbf{h}^\top \mathbf{B} \neq 0$, choose the nonlinear control law as:

$$u_0 = \mathbf{h}^\top \mathbf{B}^{-1} [\mathbf{h}^\top \dot{\mathbf{x}}_\mathbf{d} - \mathbf{h}^\top \hat{\mathbf{f}}(\mathbf{x},t) - \eta \mathrm{sgn}(s)] \quad \eta > 0 \tag{3.10}$$

where $\hat{\mathbf{f}}(\mathbf{x},t)$ is the estimate of $\mathbf{f}(\mathbf{x},t)$ and sgn(s) is the signum function:

$$\mathrm{sgn}(s) = \begin{cases} 1, & s > 0 \\ 0, & s = 0 \\ -1, & s < 0 \end{cases} \tag{3.11}$$

This gives the feedback control law:

$$\begin{aligned} u &= -\mathbf{k}^\top \mathbf{x} + u_0 \\ &= -\mathbf{k}^\top \mathbf{x} + \mathbf{h}^\top \mathbf{B}^{-1} [\mathbf{h}^\top \dot{\mathbf{x}}_\mathbf{d} - \mathbf{h}^\top \hat{\mathbf{f}}(\mathbf{x},t) - \eta \mathrm{sgn}(s)] \end{aligned} \tag{3.12}$$

How to choose k and h

When the state-space model and sliding surface is found, there remains to determine the vectors **k** and **h**. **k** is the feedback gain vector found by pole placement, as mentioned, and is found by placing the poles such that the state-space model in Equation 3.7 is stable. The poles can be assigned arbitrarily if and only if (A, B) in Equation 3.4 is controllable, Chen [1999], see Appendix A for definition of controllability. When determining the poles, it is important to be aware of that the multiplicity of the poles can not be greater than the rank of B. **h** can be chosen as the right eigenvector of A_c, which corresponds to $\lambda = 0$ where $\lambda = \lambda(A_c^\top)$ is the eigenvalue of A_c.

Stability of Sliding Mode

Stability, ensuring that s from Equation 3.8 converges to zero in finite time, can be given by Lyapunov stability analysis. Take the Lyapunov function:

$$V = \frac{1}{2}s^2 \geq 0 \tag{3.13}$$

where $V \geq 0$ means that V is positive semidefinite.

From the control law in Equation 3.10, the derivative of sliding surface in Equation 3.9 becomes:

$$\dot{s} = \mathbf{h}^\top \mathbf{A_c}\mathbf{x} - \eta \mathrm{sgn}(s) + \mathbf{h}^\top \Delta\mathbf{f}(\mathbf{x},t) \tag{3.14}$$

where $\Delta\mathbf{f}(\mathbf{x},t) = \mathbf{f}(\mathbf{x},t) - \hat{\mathbf{f}}(\mathbf{x},t)$. Rewriting the first term with:

$$\mathbf{h}^\top \mathbf{A_c}\mathbf{x} = \mathbf{x}^\top \mathbf{A_c^\top} \mathbf{h} = \lambda \mathbf{x}^\top \mathbf{h} \tag{3.15}$$

where λ represent the eigenvalue corresponding to the eigenvector $\mathbf{h}$ of $\mathbf{A_c^\top}$ such that:

$$\mathbf{A_c^\top}\mathbf{h} = \lambda \mathbf{h} \tag{3.16}$$

This leads to:

$$\dot{s} = \lambda \mathbf{x}^\top \mathbf{h} - \eta \mathrm{sgn}(s) + \mathbf{h}^\top \Delta\mathbf{f}(\mathbf{x},t) \tag{3.17}$$

$\dot{s}$ can be reduced to:

$$\dot{s} = -\eta \mathrm{sgn}(s) + \mathbf{h}^\top \Delta\mathbf{f}(\mathbf{x},t) \tag{3.18}$$

by having $\lambda \mathbf{x}^\top \mathbf{h} = 0$ for $\lambda = 0$.

From this follows:

$$\begin{aligned} \dot{V} &= s\dot{s} \\ &= -\eta \mathrm{sgn}(s)s + s\mathbf{h}^\top \Delta\mathbf{f}(\mathbf{x},t) \\ &= -\eta|s| + s\mathbf{h}^\top \Delta\mathbf{f}(\mathbf{x},t) \leq 0 \end{aligned} \tag{3.19}$$

$\dot{V} \leq 0$, negative semidefinite, by selecting η as:

$$\eta > ||\mathbf{h}|| \cdot ||\Delta\mathbf{f}(\mathbf{x},t)|| \tag{3.20}$$

where $||X||$ detonates the norm of X.

By applying Barbalat's lemma, Appendix A, this means that s converges to zero in finite time if η is chosen properly. If η is chosen large enough, the response of the system will be governed by the response of the sliding surface s and its parameters, even with modeling uncertainty, nonlinear terms and disturbances, Healey and Lienard [1993].

It is important to be aware of chattering in sliding mode due to uncertainties and delays in the system, Khalil [2002]. The sliding mode will then behave like in the Figure 3.4. Instead of having a sliding mode-motion when the trajectory reaches the sliding surface, $s = 0$, in a, it begins to drift away due to delay between the time sign to s switches and the controller switches. When the controller switches, the trajectory reverses and the motion will be towards the sliding surface again.

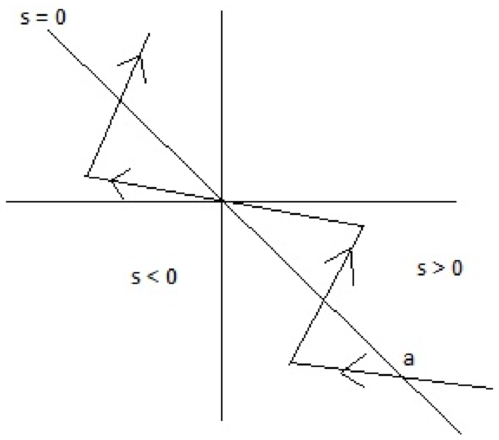

Figure 3.4: Sliding mode with chattering due to delay in controller, Khalil [2002]

Chattering can lead to low control accuracy, wear and tear of actuators and in worst case lead to unmodeled high-frequent dynamics which can lead to worse performance of the system and unstability.

Chattering can be removed by changing sgn(s), Equation 3.11, with

$$\text{sat}(s) = \begin{cases} \text{sgn}(s) & \text{if} |s/\phi| > 1 \\ s/\phi & \text{otherwise} \end{cases} \tag{3.21}$$

where $\phi > 0$ is the parameter for sliding surface boundary layer thickness, see Figure 3.5, Fossen [2011b].

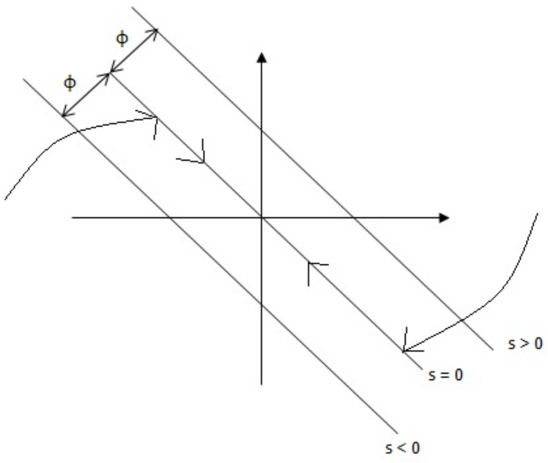

Figure 3.5: Phase portrait of sliding mode with boundary layer, Fossen [2011b]

The aim of sliding mode control is to make a control law that ensures $s \to 0$ in finite time, Figure 3.5.

This is the basic idea behind sliding mode, but there are many different approaches of using sliding mode in autopilot design for an UAV.

Chapter 4

Design of GNC System

An autopilot consist of a Guidance, Navigation and Control (GNC) System, Figure 4.1. Guidance takes care of input to the system, inputs as waypoints and desired speed, and determine the desired path from the current location of the aircraft to the desired waypoint. Guidance is often decoupled onto reference model and guidance system where reference model deals with commanded signals and guidance determines the path. Navigation determines the location and altitude of the aircraft at a given time. The control system ensures that the aircraft follows the desired path and altitude by manipulating the control surfaces.

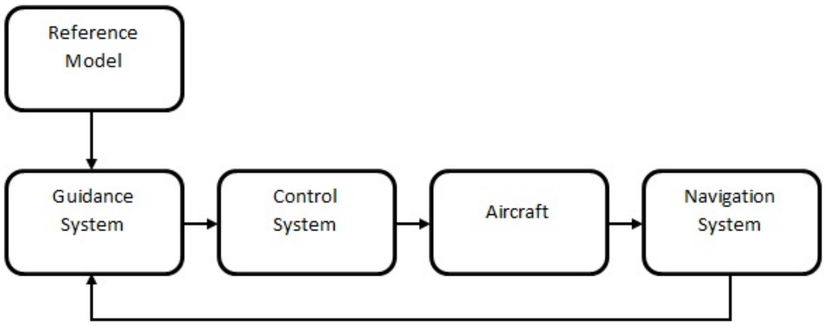

Figure 4.1: Block diagram of a Guidance, Navigation and Control System

4.1 Reference Model and Guidance System

The guidance system has been designed as described in Chapter 2. LOS is used for calculating desired heading angle, the kinematic controller has been used for desired pitch angle and a 1st order LP filter for speed, see Figure 4.2 for block diagram of guidance system.

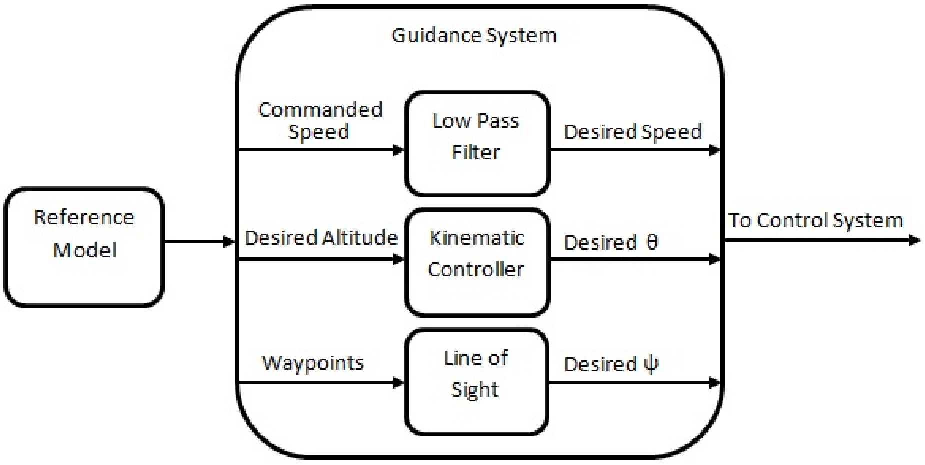

Figure 4.2: Block diagram of guidance system

4.1.1 Reference Model

Signals into the guidance system are given by the reference model. An operator determines where the UAV is supposed to go in North-East coordinates, at which altitude and speed. These are commanded signals. Since the guidance system for altitude does not tolerate steps as inputs, the reference model has to smooth out this signals. This can be done by a third-order Low Pass (LP) filter with the structure:

$$\frac{x_d}{r}(s) = h_{lp}(s) \tag{4.1}$$

where x_d is desired state, r is the reference signal given by operator and h_{lp} is the LP filter. The choice of filter should be based on the physics of the system it is supposed to work on, and for marine craft and aerial vehicles, it is used a model based on *mass-damper-spring* systems so that the LP filter becomes:

$$h_{lp}(s) = \frac{\omega^2}{s^2 + 2\zeta\omega s + \omega^2} \tag{4.2}$$

where ζ is the *relative damping ratio* and ω is the *natural frequencies*, Fossen [2011b].

By adding a 1st order LP filter with time constant $T = 1/\omega$, the altitude LP filter will be:

$$\frac{h_d}{h_c} = \frac{\omega^3}{s^3 + (2\zeta + 1)\omega s^2 + (2\zeta + 1)\omega^2 s + \omega^3} \tag{4.3}$$

where h_c is the commanded altitude and h_d is the desired altitude which will be sent to the guidance system.

The commanded altitude is set by an operator by specifying which altitude is desired at what time. Under takeoff, the altitude is set to zero the first seconds, because the UAV needs a certain speed to be able to liftoff.

Position in NED and speed is sent directly to guidance, position as waypoints and speed as a vector with commanded speed wanted at different times, as for altitude.

4.1.2 Guidance System

LOS for North-East

The LOS guidance method for calculating desired heading angle ψ was designed as described in Section 2.1. Waypoints are commanded by an operator in the simulation file SimGNCSystem.m, Appendix B. To switch from one waypoint to the next, a circle of acceptance test is applied, as mentioned. To find the radius of the circle, test simulation has been done and the value has been set to $R_{k+1} = 1600$ in Equation 2.10.

When calculating the desired heading angle from course angle given by the LOS method, Equation 2.7, it is necessary to have the sideslip angle β. This angle is perfectly measured in the simulation program X-plane, and does not need to be calculated.

Kinematic Control for Altitude

The kinematic controller for altitude guidance is implemented as described in Section 2.2 with a few changes. Since the altitude h in Equation 2.13 is measured in X-plane, this altitude will be used instead of taking the integral of the calculated altitude in $\dot{h} = U(\theta - \alpha)$. This means that the block diagram in Figure 2.4 changes to Figure 4.3. The angle of attack, α, together with the true airspeed U is also measured in X-plane and sent to guidance. From the reference model come desired altitude h_d and the derivative of desired altitude $\dot{h}_d$.

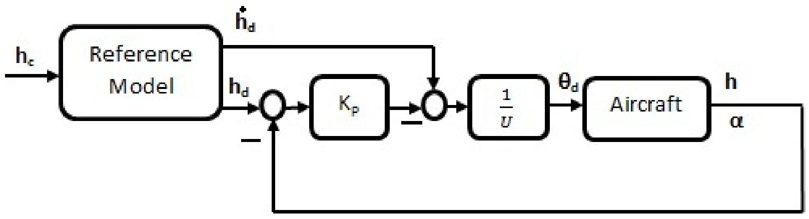

Figure 4.3: Block diagram of implemented kinematic control for altitude guidance

LP for Speed

Since the UAV does not handle steps as input, the commanded speed needs to be smoothed. This can be done by applying a 1st order low pass filter:

$$\frac{U_d}{U_c} = \frac{1}{1 + Ts} \tag{4.4}$$

where U_d is the desired speed given to the control system, U_c is commanded speed and T is the time constant given by $T = 1/\omega > 0$. The LP filter is implemented as in Figure 4.4.

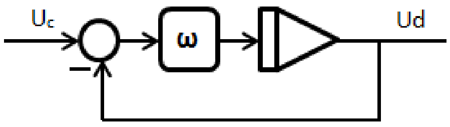

Figure 4.4: 1st order LP filter

Desired roll, pitch and yaw angles and rudder, aileron and elevator deflections are saturated due to physical limitations on the aircraft. Aileron, elevator and rudder does not have 360 degrees of operation area.

4.2 Navigation

The navigation system is supposed to determine the position of the aircraft at a given time t. It could include a filter for filtering measurement noise caused by for instance noise in sensors, waves, current and wind. Often an observer is used for filtering and state estimation to estimate unmeasurable signals or estimate states if the signals drops out.

In this thesis however, perfect measured signals are assumed and an observer is therefore not necessary.

4.3 Control System

The control system has been implemented first with PID control, then with sliding mode control. The PID controller has been made to test the guidance system, handle turning operations and to compare performance with sliding mode control.

4.3.1 PID

The first controller, PID controller, was implemented as described in Section 3.1. The control system was decoupled into four parts, for speed control, roll control, pitch control and yaw control, see Figure 4.5.

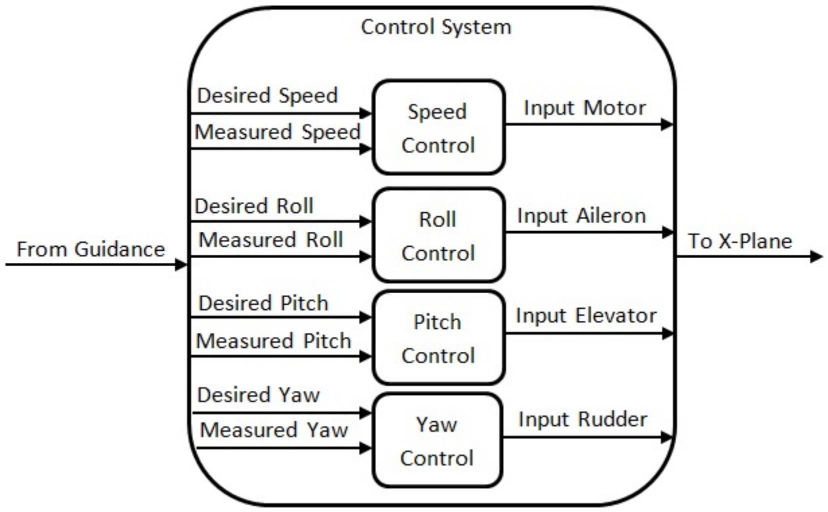

Figure 4.5: Block diagram of control system

With $\delta_M, \delta_A, \delta_E, \delta_R$ as input to motor, aileron, elevator and rudder, respectively, and U, ϕ, θ, ψ as speed, roll, pitch and yaw, respectively, we get the control equations in Equation 4.5:

$$\begin{aligned}
\delta_M &= K_p(U_d - U(t)) + K_i \int_0^t (U_d - U(\tau))d\tau + K_d \frac{\mathrm{d}}{\mathrm{d}t}(U_d - U(t)) \\
\delta_A &= K_p(\phi_d - \phi(t)) + K_i \int_0^t (\phi_d - \phi(\tau))d\tau + K_d \frac{\mathrm{d}}{\mathrm{d}t}(\phi_d - \phi(t)) \\
\delta_E &= K_p(\theta_d - \theta(t)) + K_i \int_0^t (\theta_d - \theta(\tau))d\tau + K_d \frac{\mathrm{d}}{\mathrm{d}t}(\theta_d - \theta(t)) \\
\delta_R &= K_p(\psi_d - \psi(t)) + K_i \int_0^t (\psi_d - \psi(\tau))d\tau + K_d \frac{\mathrm{d}}{\mathrm{d}t}(\psi_d - \psi(t))
\end{aligned} \tag{4.5}$$

Since X-plane have perfect measurements there is no need to take the derivative of the signals and the last term in the equations above, except for speed, will be:

$$\begin{aligned} &K_p(\dot{\phi}_d - P) \\ &K_p(\dot{\theta}_d - Q) \\ &K_p(\dot{\psi}_d - R) \end{aligned} \tag{4.6}$$

where P, Q, R are roll, pitch and yaw rate, respectively.

The gains, K_p, K_i, K_d was tuned manually. There are different methods for tuning a PID controller, but if a range is known, it is easy to tune them manually by simulating different cases and find what values that will give a satisfying behavior.

Anti-Windup

Anti-windup is an important part of PID control. It is applied to PID control to prevent overshoot made by the integral term in Equation 4.5. The overshoot may occur when a large change in set point occurs, then the integral term accumulates a significant error during the rise, also referred to as windup. It then overshoots and continues to increase as the accumulated error is offset by errors in the other direction.

To prevent this a saturation is applied on the controller's output and this is subtracted from the integral action signal. With anti-windup, Figure 3.2 is changed to Figure 4.6.

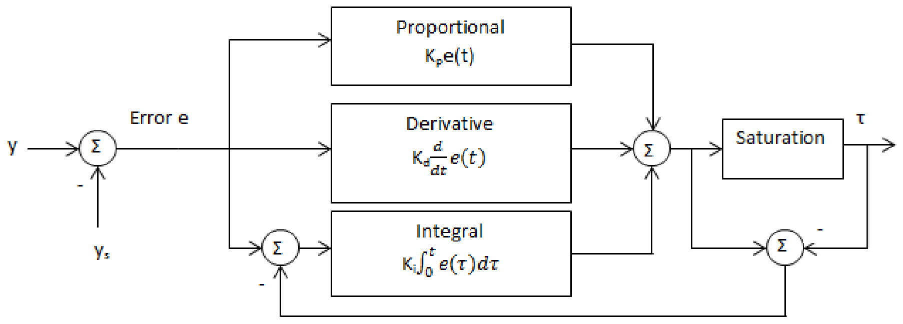

Figure 4.6: Block diagram of a PID controller with anti-windup

4.3.2 Sliding Mode

When designing a sliding mode controller, it is important to have a good model of the UAV. To simplify, the system has been decoupled into three parts, pitch and altitude, heading and speed. Pitch and altitude will be applied sliding mode control, while heading and speed will be controlled by PID.

Pitch and Altitude Control

For pitch and altitude control the state-space system is, Fossen [2011a]:

$$\begin{bmatrix}\dot{Q}\\ \dot{\theta}\\ \dot{Z}\end{bmatrix} = \begin{bmatrix}a_{11} & a_{12} & 0\\ 1 & 0 & 0\\ 0 & U_0 & 0\end{bmatrix}\begin{bmatrix}Q\\ \theta\\ Z\end{bmatrix} + \begin{bmatrix}b_1\\ 0\\ 0\end{bmatrix}\delta_E + \begin{bmatrix}0\\ 0\\ -U_0\alpha\end{bmatrix} \tag{4.7}$$

from Equation 3.4.

The equation for $\dot{Q}$ is found by system identification and

$$\begin{aligned}\dot{\theta} &= P\cos(\phi) - \sin(\phi) \approx Q && (4.8)\\ \dot{h} &= -U_0\sin(\theta) + V\cos(\theta)\sin(\psi) + W\cos(\theta)\cos(\psi) \approx U_0\theta - W && (4.9)\end{aligned}$$

where $h = Z$, $V = P = 0$, $W = -U_0\alpha$ and assumed small values of ϕ and θ.

Sliding surface from Equation 3.8 becomes:

$$s = h_1(Q_{-}Q_d) + h_2(\theta - \theta_d) + h_3(Z - Z_d) \tag{4.10}$$

From Equation 3.6:

$$\begin{aligned}u &= -\mathbf{k}^\top\mathbf{x} + u_0\\ &= -\mathbf{k}^\top\mathbf{x} + \mathbf{h}^\top\mathbf{B}^{-1}[\mathbf{h}^\top\dot{\mathbf{x}}_\mathbf{d} - \mathbf{h}^\top\hat{\mathbf{f}}(\mathbf{x},t) - \eta\mathrm{sgn}(s)]\end{aligned} \tag{4.11}$$

the pitch and altitude control law for $u = \delta_E$ is:

$$\delta_E = -k_1Q - k_3Z + \frac{1}{h_1b_1}[h_1\dot{Q}_d + h_2\dot{\theta}_d + h_3\dot{Z}_d - h_3f_3 - \eta\mathrm{sat}(s)] \tag{4.12}$$

sgn(s) is replaced by sat(s) to avoid chattering, as explained in Section 3.2.

The sliding mode control does not handle steps as input, so the desired pitch needs to be a smooth signal, done with a 1st order LP filter, as has been applied to speed. This is done in the guidance system.

The same has been done on elevator deflection δ_E to avoid wear and tear on the control surface since sliding mode control has fast varying output.

System Identification with Ordinary Least Square

To do system identification the ordinary least square method will be used, where unknown parameters in a linear regression model will be estimated. Consider k independent variables $x_1, x_2, \cdots, x_k$ and n observations $y_1, y_2, \cdots, y_n$, this give the multiple linear regression model:

$$y_i = \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_k x_{ki} + \epsilon_i \quad i = 1, 2, \cdots, n \text{ and } n > k \tag{4.13}$$

which will be estimated as:

$$\hat{y}_i = b_1 x_{1i} + b_2 x_{2i} + \cdots + b_k x_{ki} + e_i \tag{4.14}$$

where ϵ_i and e_i is random error and residual associated with the response of y_i while b_i is the estimate of β_i estimated from samples of data by applying least square system identification method.

The multiple linear regression model can be written on a more compact form as:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{4.15}$$

where:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} x_{11} & x_{21} & \cdots & x_{k1} \\ x_{12} & x_{22} & \cdots & x_{k2} \\ \vdots & \vdots & & \vdots \\ x_{1n} & x_{2n} & \cdots & x_{kn} \end{bmatrix}, \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_k \end{bmatrix} \tag{4.16}$$

Now we want to find a $\mathbf{b}$ that minimizes:

$$\begin{aligned} \text{SSE} = \sum_{i=1}^{n} e_i^2 &= \sum_{i=1}^{n} (y_1 - b_1 x_{1i} - b_2 x_{2i} - \cdots b_k x_{ki})^2 \\ &= (\mathbf{y} - \mathbf{X}\mathbf{b})^\top (\mathbf{y} - \mathbf{X}\mathbf{b}) \end{aligned} \tag{4.17}$$

This is done by solving for $\mathbf{b}$ in:

$$\frac{\partial}{\partial \mathbf{b}}(\text{SSE}) = \mathbf{0} \tag{4.18}$$

which can be solved by solving for $\mathbf{b}$ in:

$$(\mathbf{X}^\top \mathbf{X})\mathbf{b} = \mathbf{X}^\top \mathbf{y} \tag{4.19}$$

The equation for the regression coefficient can then be written as:

$$\mathbf{b} = (\mathbf{X}^\top \mathbf{X})^{-\mathbf{1}} \mathbf{X}^\top \mathbf{y} \tag{4.20}$$

Walpole [2002].

For pitch the unknown parameters are a_{11}, a_{12} and b_1 in Equation 4.7 such that the multiple linear regression model becomes:

$$\dot{Q} = \begin{bmatrix} Q & \theta & \delta_E \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ b_1 \end{bmatrix} \tag{4.21}$$

where Q is pitch rate, θ is pitch angel and δ_E is desired elevator deflection.

4.4 X-Plane

To simulate the autopilot, it is necessary to have a good aircraft model. There are many methods to get an aircraft model; make one in Matlab Simulink based on equations of motion for an aircraft, use a model already made in Matlab Simulink or, as has been done in this thesis, use a flight simulator program. X-Plane, X-Plane 9 [2012], is a simulator mainly used to simulate flights by use of pedals and joystick and to design new aircrafts. In this thesis the flight simulator has been used to simulate the designed autopilot. To do that, the data from the autopilot system has been sent to the flight simulator by a User Datagram Protocol (UDP) block in Matlab Simulink and an inbuilt plugin in X-Plane.

Before the signals enters X-Plane, it has been allocated to fit input for X-Plane. X-Plane is setting motor, left and right aileron, elevator and rudder, but the control system is given out motor, aileron, elevator and rudder. This means that the signals for aileron deflection has to be split into two signals. Left and right ailerons is set opposite to get desired behavior, when turning. This leads to this thrust allocation:

$$\begin{bmatrix} \delta_M \\ \delta_{LA} \\ \delta_{RA} \\ \delta_E \\ \delta_R \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & -1 & 0 & -1 \\ 0 & 0 & -1 & 0 \\ 0 & 0.02 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_M \\ \delta_A \\ \delta_E \\ \delta_R \end{bmatrix} \tag{4.22}$$

where δ_{LA} means left aileron and δ_{RA} means right aileron. The rudder is weighted low since it is desirable to use ailerons for turn.

4.4.1 Cessna 172SP

As mentioned in Chapter 3, a Cessna, not an UAV, has been used as an aircraft model in this thesis, Figure 4.7. The Cessna has the control surfaces: aileron on both wings and rudder on the trailing edge of the vertical stabilizer used for heading, elevator back on the trailing edge of the horizontal stabilizer for pitch and a motor for forward thrust. Specifications for Cessna can be seen in Table 4.1.

Table 4.1: Cessna Specifications, Cessna Aircraft Company [2012]

Length	8.28 m
Height	2.72 m
Wingspan	11.00 m
Wing Area	16.20 m^2
Weight	779 kg
Max Takeoff Payload	378 kg
Max Takeoff Weight	1157 kg

Figure 4.7: Illustration of Cessna 172sp in X-plane

Chapter 5

Case Study

To make hardware in the loop (HIL) testing of the guidance and control system design, the flight simulator X-Plane has been used. The guidance and control system is designed in Matlab Simulink and the control signals are sent to X-Plane by a X-Plane plugin, see Figure 5.1 for X-Plane communication. From X-Plane, the measured signals are sent to the guidance and control system in Simulink, see Figure 5.2.

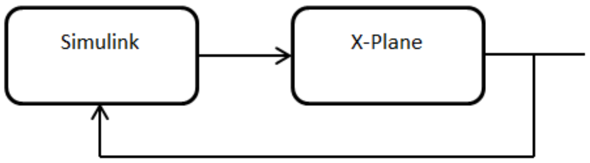

Figure 5.1: Simulink X-Plane communication

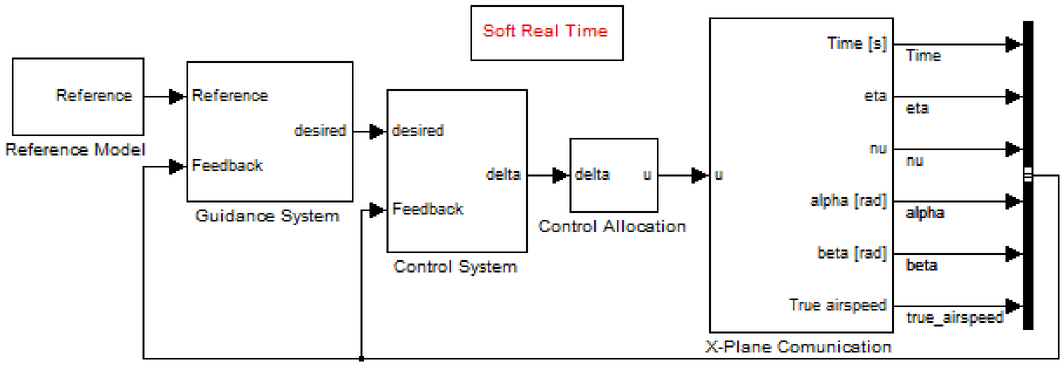

Figure 5.2: Simulink block diagram, communication with X-Plane

The communication with X-Plane is done by UDP protocol. This makes it possible to run the simulation in Matlab Simulink and X-Plane on two different computers. However, in this thesis it all run on one computer. The simulation is done in real time.

Since X-Plane is used as flight model and visualization, the guidance and control system has been tested on a Cessna instead of an UAV. The X-Plane simulator had no UAV available. As long as the control signals sent from the control system get distributed to the right control surfaces on the aircraft, the guidance and control system designed in this thesis could be applied to any UAVs or regular aircrafts.

The purpose of this thesis was to investigate different methods for UAV autopilot design. Two different controllers were chosen, Proportional-Integral-Derivative (PID), and sliding mode control. The PID is a linear controller which does not need model parameter information, while the sliding mode is based on the aircraft model. Sliding mode is more robust that can handle model uncertainties. PID is widely used and is a perfect controller to compare behavior with other control methods.

An UAV must be able to operate under certain conditions. It will be exposed to wind and it has to be able to carry a payload such as cameras, sensors and communication equipment. The guidance and control system will be tested in the following four scenarios:

- Without disturbances and payload, Section 5.1
- With payload, no wind, Section 5.2

- With wind, 20 knots, no payload, Section 5.3
- With wind, 40 knots, no payload, Section 5.4

with both PID control for pitch, altitude and heading, and sliding mode for pitch and altitude with PID for heading control.

The aircraft is initialized such that it starts in position $[0, 0, 0]$ in NED convention. In every case the aim is to takeoff and ascend to an altitude of 500 meters. Ascending is to increase the altitude and is done by varying the pitch angle. The aircraft will takeoff straight east, make a turn north when reaching 8000 meters east, and then make a turn east again at 10000 meters north and 8000 meters east. When reaching an altitude of 500 meters, it will remain at 500.

When the simulation starts, desired altitude is set to zero so that the speed controller is the only one working. This is because the aircraft needs a certain speed to be able to takeoff and this condition holds for all four test cases. The takeoff speed is set to 80 knots, while the cruise speed is set to 48 knots.

For each case, seven plots will be presented; position in North-East vs desired, measured altitude vs desired, measured roll, pitch and yaw angle vs desired, angle of attack and the sideslip angle.

The simulations have been done in Matlab Simulink and X-plane interface has been used as aircraft model and visualization, Section 4.4. To simulate, use the mini guide in Appendix C for computer setup and run the attached *SimGNCSystem.m* file. Make sure all files listed in Appendix B are in the same folder.

5.1 Without disturbances and payload

First the UAV has been simulated with perfect conditions, without disturbances such as wind and without payload.

5.1.1 PID

The PID controller is tuned manually, this is repeated for all later cases. The controller is tuned to handle strong wind, these tuning values are used in all four simulation cases.

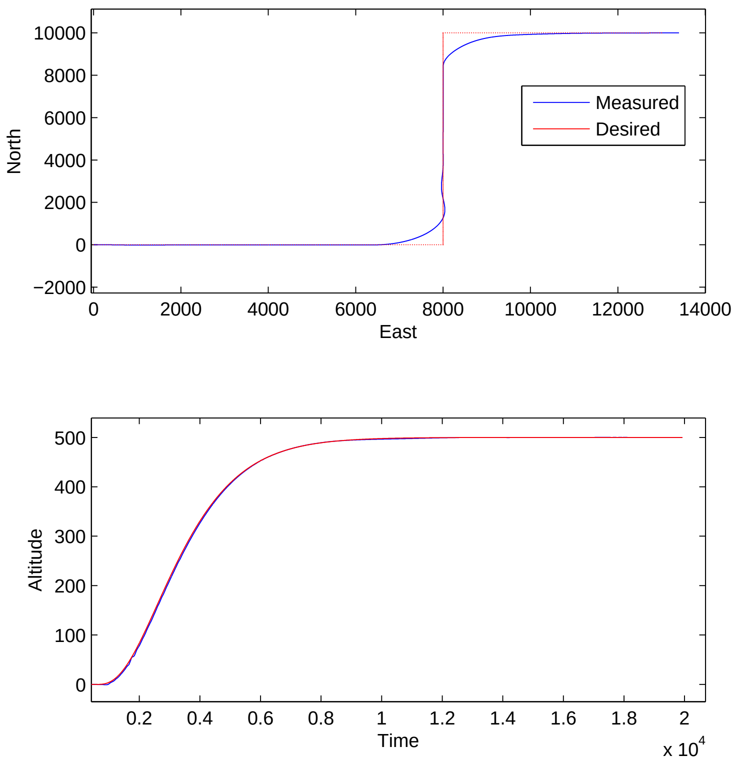

Figure 5.3: North-East and Altitude [m] without disturbances and payload, measured vs desired

As can be seen from Figure 5.3, the control system follows the desired path. There is a small overshoot of 40 meters in heading at the first turning maneuver, but this is considered small enough to not cause problems.

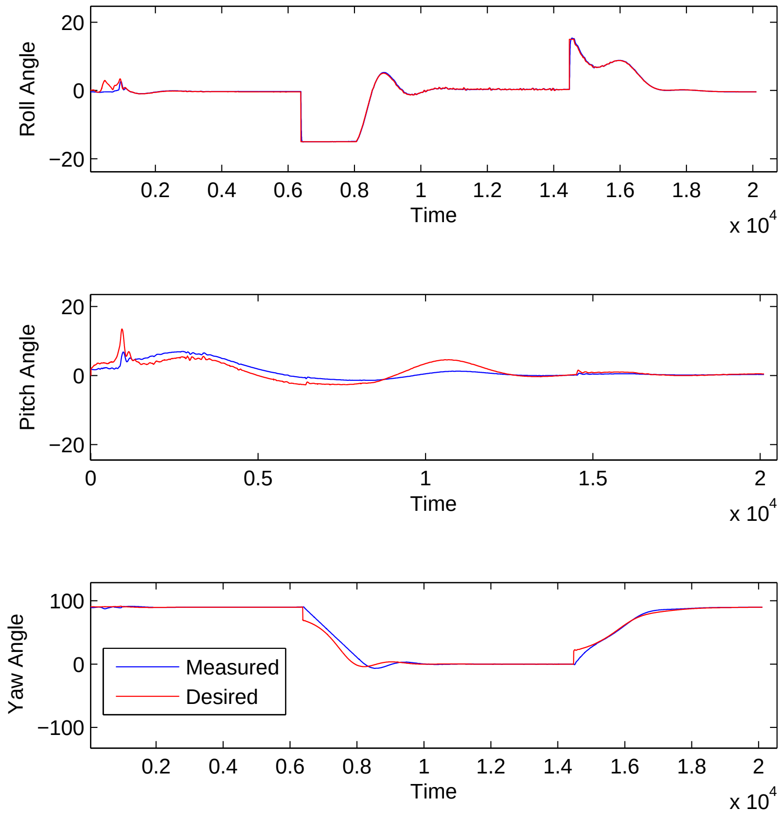

Figure 5.4: Roll, pitch and yaw angle [deg] without disturbances and payload, measured vs desired

From Figure 5.4 it is seen that the control system follows the desired roll, pitch and yaw angles satisfactorily. All angles are in degrees.

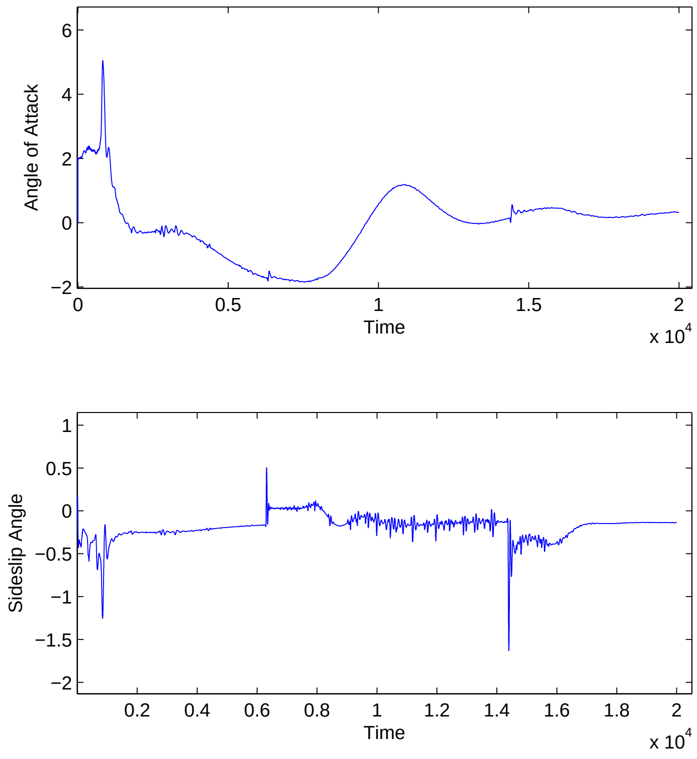

Figure 5.5: Angle of attack and side slip angle [deg] without disturbances and payload, measured vs desired

Figure 5.5 shows the angle of attack and sideslip angle. Angle of attack is the angle between a reference line on the aircraft and the vector representing the relative motion. Sideslip angle is the angle between the aircraft centerline and the vector of relative wind working on the aircraft. The angle of attack is varying as the aircraft ascends and then stabilizes when the aircraft is at the cruising altitude. The sideslip angle is small, close to zero with small spikes.

5.1.2 Sliding Mode

The sliding mode is tuned by pole placement. After much tuning the poles where to find close to the origin. All poles were real, not imaginary.

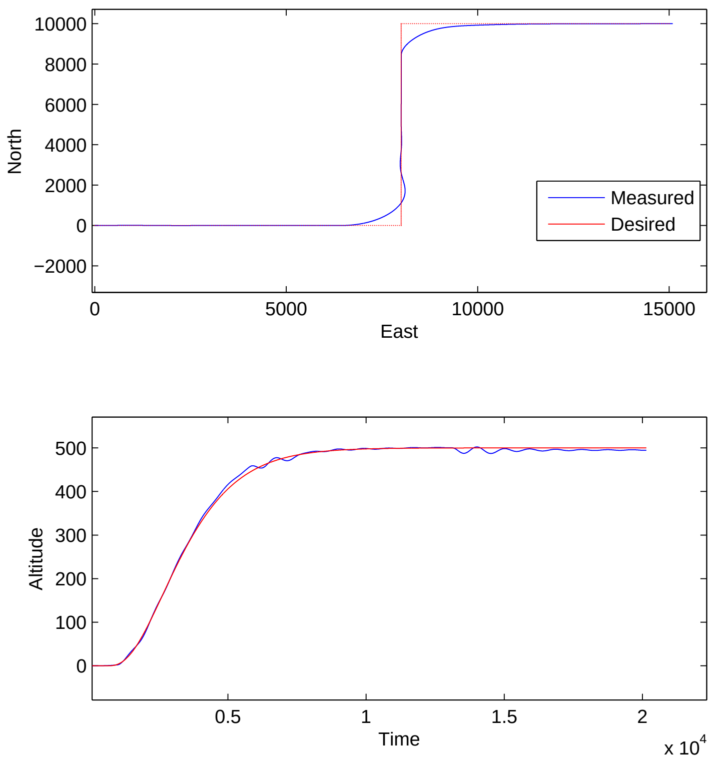

Figure 5.6: North-East and Altitude [m] without disturbances and payload, measured vs desired

In Figure 5.6, the plot for altitude is the only one of interest, because the PID controllers takes care of heading. The sliding mode controller for pitch and altitude control is able to follow desired altitude satisfactorily. At the end of the simulation, small oscillations occur.

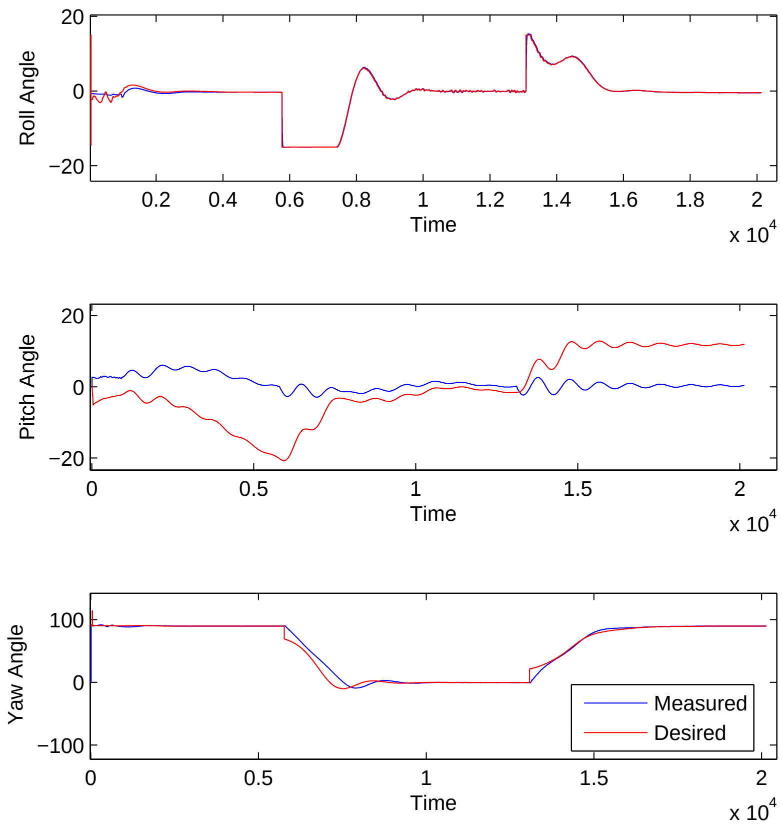

Figure 5.7: Roll, pitch and yaw angle [deg] without disturbances and payload, measured vs desired

Roll and yaw angle on Figure 5.7 are controlled by PID control and have the same behavior as in Figure 5.4. The pitch angle has a relatively big error in the beginning and at the end. In the middle the measured pitch follows the desired pitch.

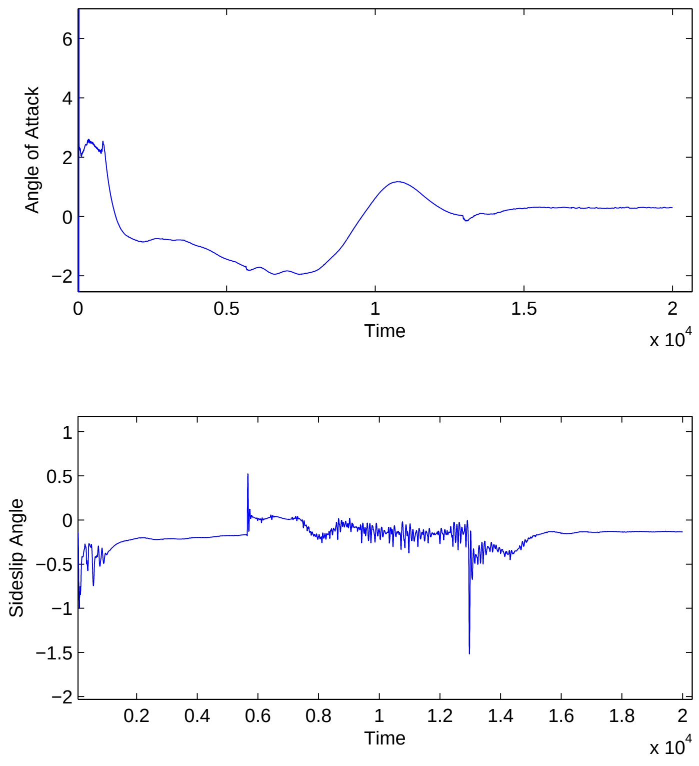

Figure 5.8: Angle of attack and side slip angle [deg] without disturbances and payload, measured vs desired

The angle of attack in Figure 5.8 varies in the beginning and then stabilizes when the aircraft reaches its cruising speed, as it did for PID control. The sideslip angle is the same as for PID control.

5.2 With payload

The GNC system has been tested with a certain amount of payload. An UAV may carry sensors, cameras, communication equipment, etc. and the GNC system has to handle this weight. For a Cessna the maximum payload is 378 kg.

5.2.1 PID

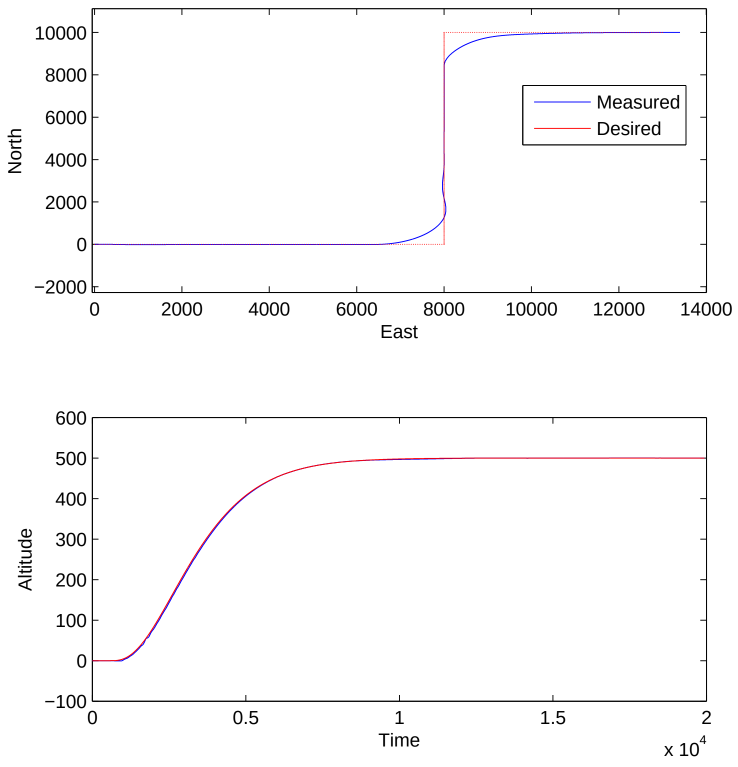

Figure 5.9: North-East and Altitude [m] with payload, measured vs desired

The PID control has no problems following the desired path or altitude when payload is added, as can be seen from Figure 5.9. The overshoot at the first turning maneuver is the same as for simulation without disturbances and payload.

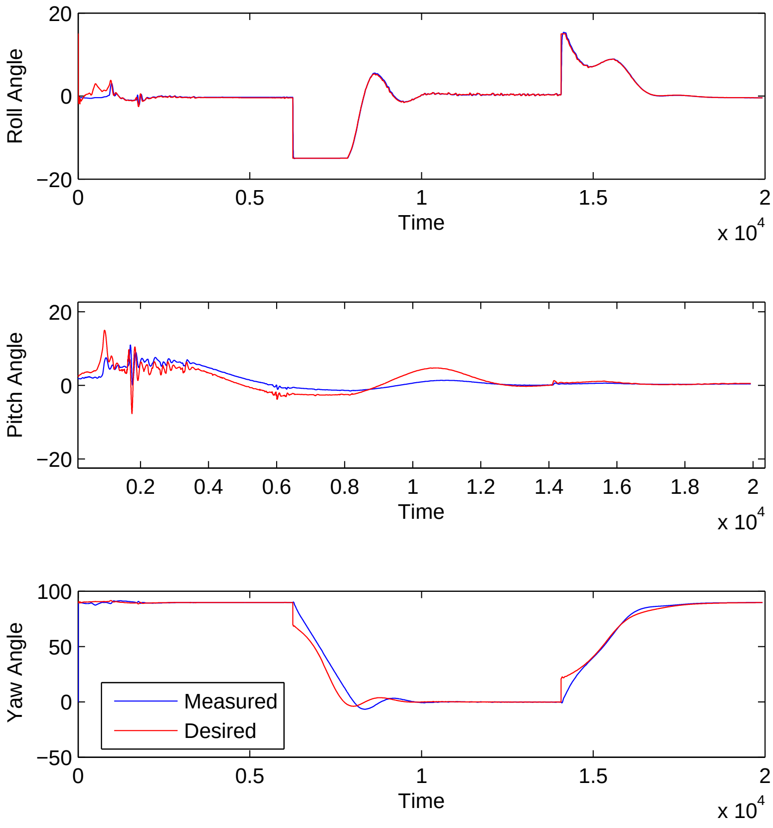

Figure 5.10: Roll, pitch and yaw angle [deg] with payload, measured vs desired

From Figure 5.10 it can be seen that the pitch control do need to work harder when simulating with payload than without. This yields only during takeoff, and not after the aircraft has ascended to cruise altitude.

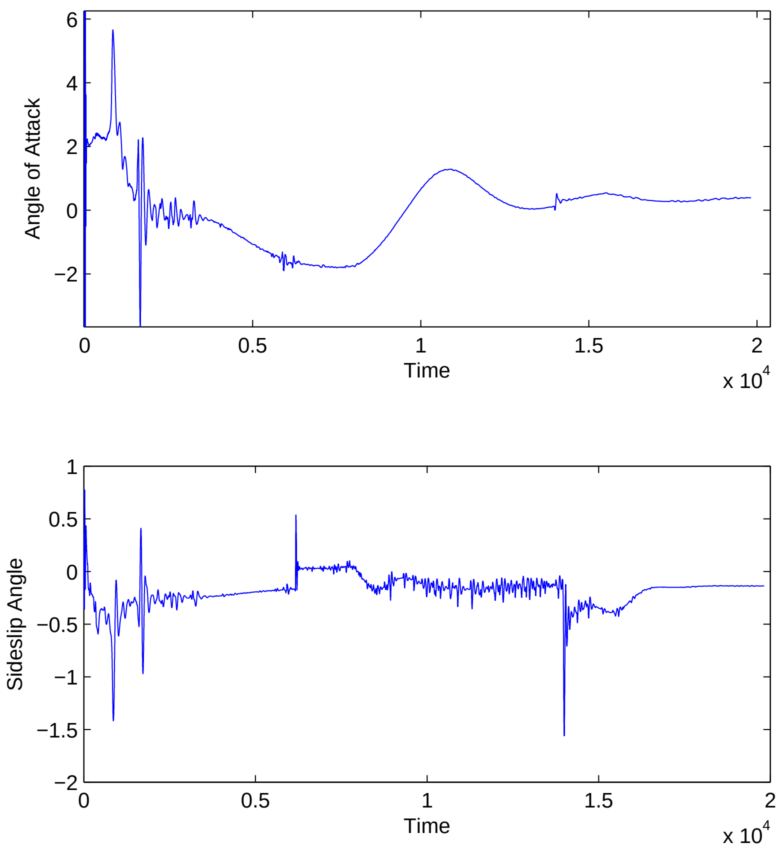

Figure 5.11: Angle of attack and side slip angle [deg] with payload, measured vs desired

There are more spikes in both angle of attack and the sideslip angle when payload is added to the aircraft. During takeoff and ascending, the angle of attack varies more and have greater spikes than the previous case.

5.2.2 Sliding Mode

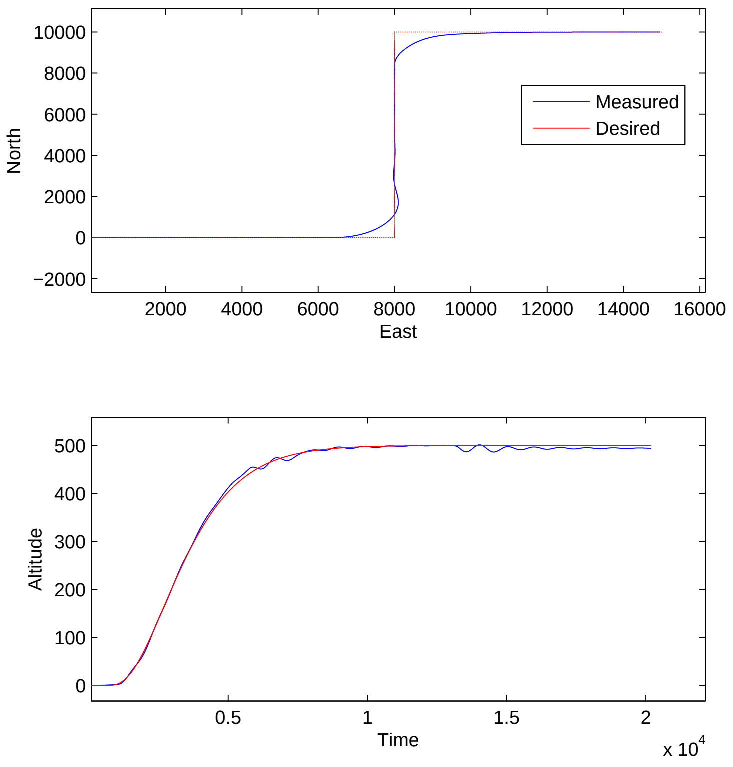

Figure 5.12: North-East and Altitude [m] with payload, measured vs desired

When applied payload, the sliding mode control oscillates a bit more, Figure 5.12. It follows the desired altitude as it ascends, but show some oscillations when reaching the altitude of 500 meters. When turning, at time 1000 to 1500, it is able to keep the altitude without oscillations. This is the second turn, while the first is during the climbing. After finish turn, it has some oscillations that decreases.

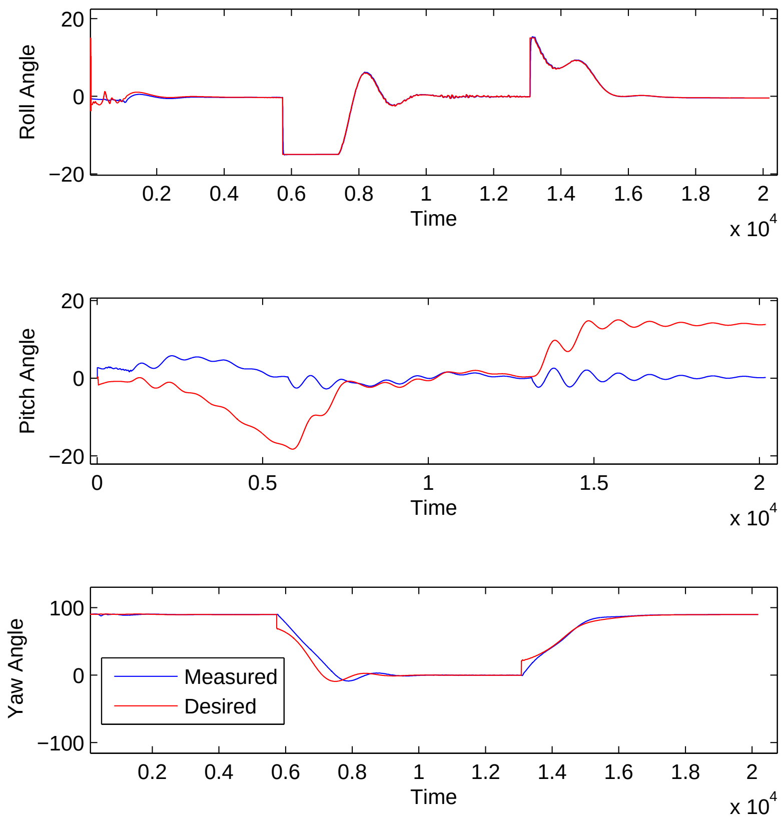

Figure 5.13: Roll, pitch and yaw angle [deg] with payload, measured vs desired

Again, the sliding mode control is not able to follow the desired pitch angle in the beginning and at the end, Figure 5.13. Roll and yaw angle follow desired angle.

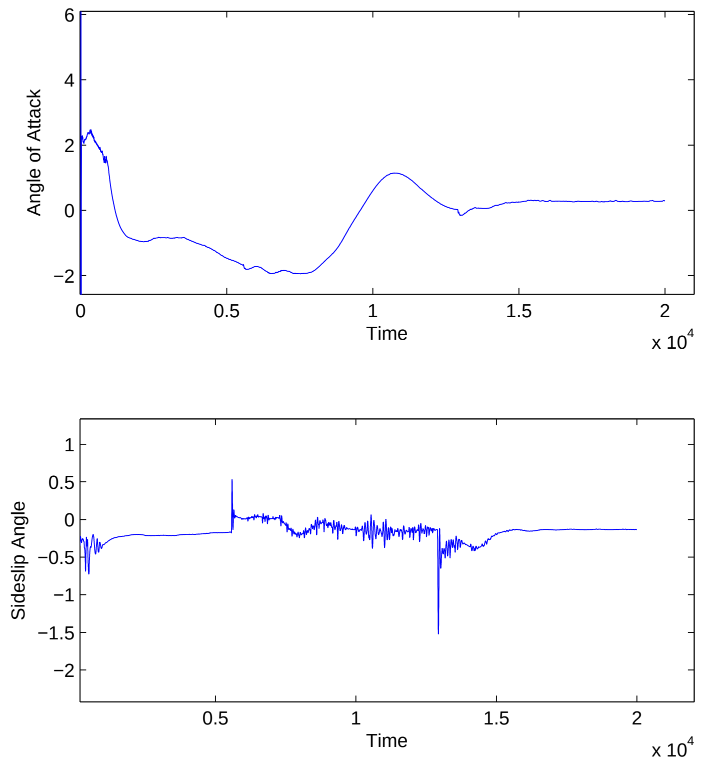

Figure 5.14: Angle of attack and side slip angle [deg] with payload, measured vs desired

The angle of attack and the sideslip angle, Figure 5.14, are the same as for simulation without payload and wind disturbances.

5.3 With Wind 20 knots

Under takeoff it is preferable to have direct headwind. The runway is directed West-East so the applied wind is coming from the east. The applied wind is set after discussion with an experienced Cessna pilot.

A Cessna can handle relatively strong wind, the limitations are mainly caused by maximum speed which is 100 knots. To have a comfortably flight, the wind should not exceed 35 knots. Based on this, the GNC system has been tested in 20 knots, presented in this section, and 40 knots, presented in Section 5.4. It should be notated that making the flight comfortable is not a high priority since it is not supposed to have a human operator onboard.

5.3.1 PID

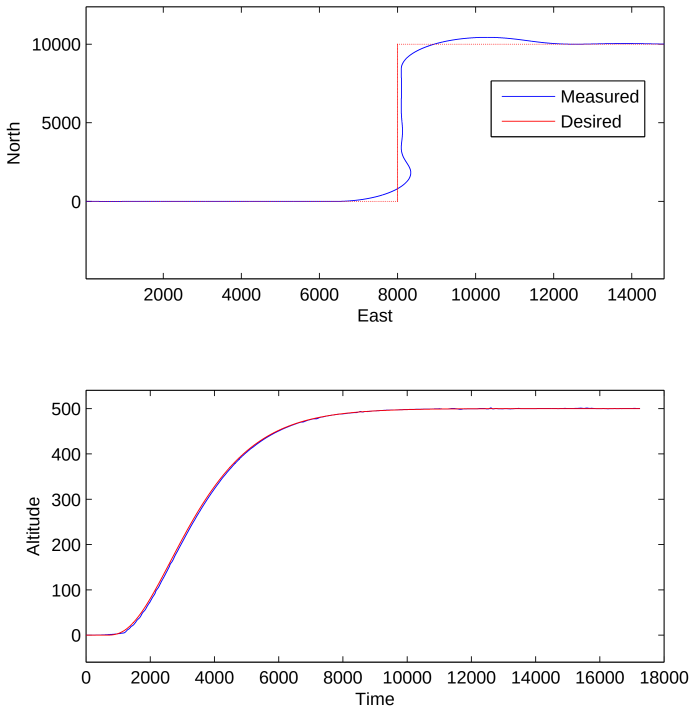

Figure 5.15: North-East and Altitude [m] with 20kt wind, measured vs desired

With wind, 20 knots coming from east, the PID controller in pitch and altitude is working fine, Figure 5.15. In heading we get an overshoot with an error of 350 meters, and it is not able to reach the desired path between waypoint $[N, E] = [0, 8000]$ and $[N, E] = [10000, 8000]$. The error is of approximately 100 meters between these waypoints. After turning east again the overshoot is 400 meters, before the measured position converges to desired path.

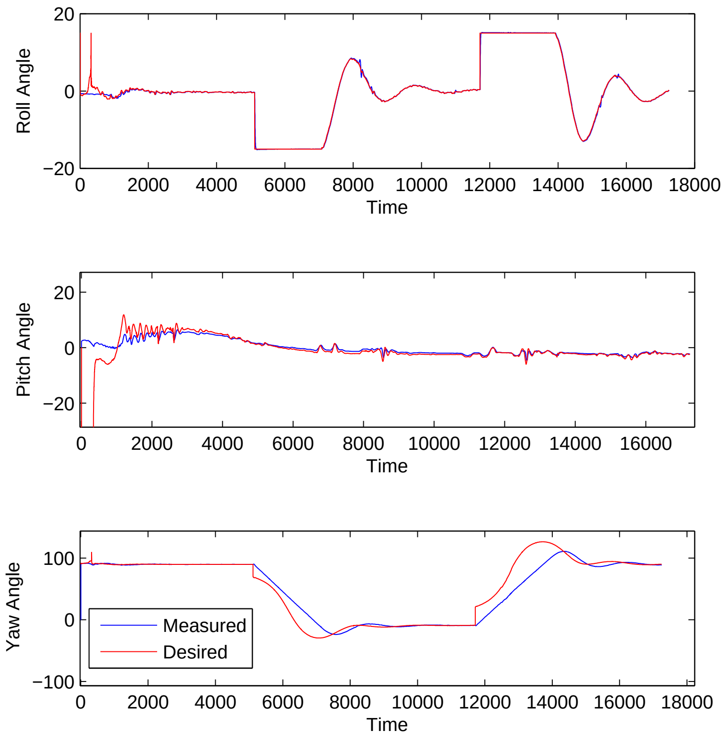

Figure 5.16: Roll, pitch and yaw angle [deg] with 20kt wind, measured vs desired

As can be seen from Figure 5.16, the pitch controller has some spikes because of the wind, but the measured follows the desired signal successfully. In the beginning there is a huge error due to initialization.

The yaw controller has some problems in following the reference, which corresponds to the error in North-East position in Figure 5.15.

The spikes in desired roll is due to initialization.

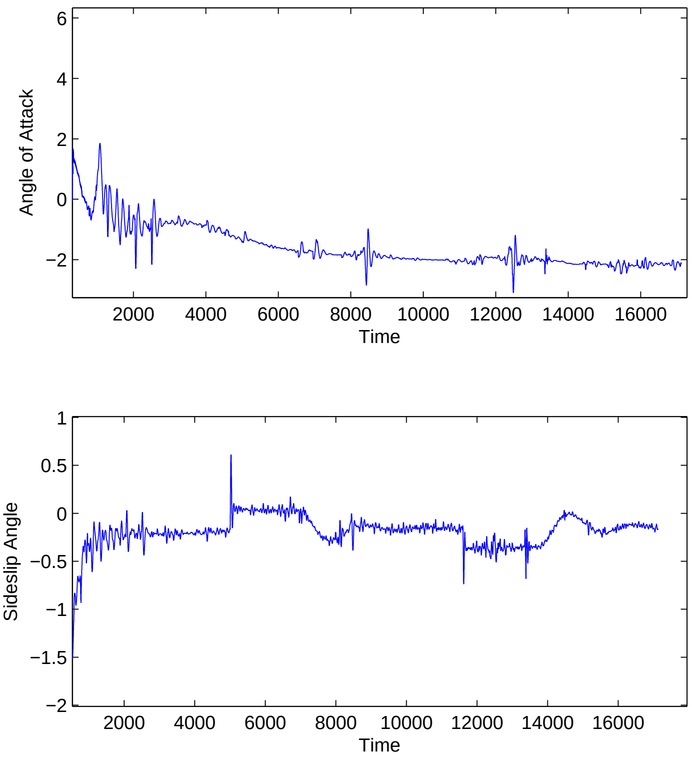

Figure 5.17: Angle of attack and side slip angle [deg] with 20kt wind, measured vs desired

There are a lot of spikes in the measured angle of attack and the sideslip angle in Figure 5.17. The angle of attack starts positive during takeoff, but when cruising it gets negative. This is because when wind is coming from straight ahead or in from the side, the nose has to point down to prevent lift due to wind.

The sideslip angle has spikes during the whole simulation and not only when flying north, as has been the case in the previous simulations.

5.3.2 Sliding Mode

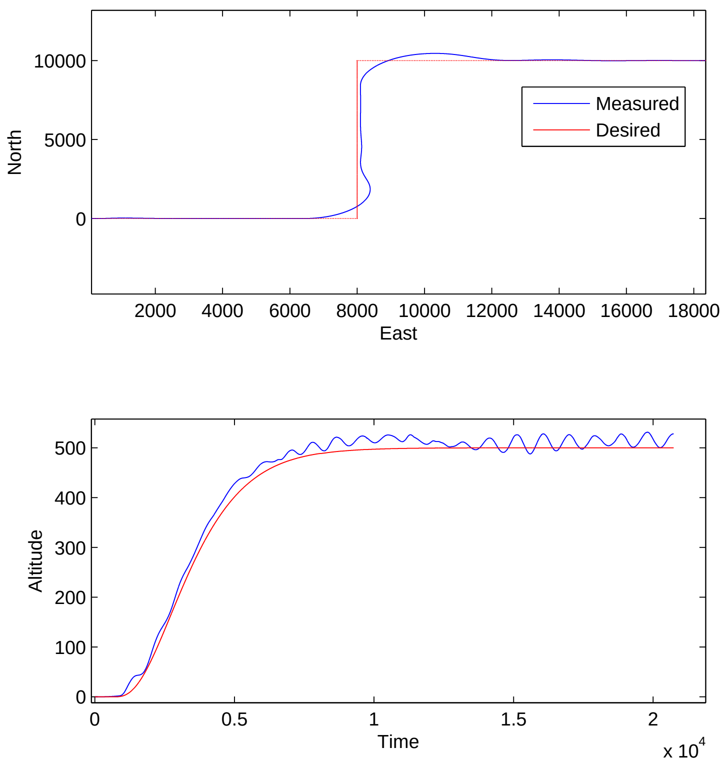

Figure 5.18: North-East and Altitude [m] with 20kt wind, measured vs desired

When wind is applied the sliding mode for pitch and altitude control behaves badly, Figure 5.18. It never reaches the desired altitude and has an error of approximately 30 meters through the whole ascension. At cruising altitude, the aircraft struggles to keep the reference and has oscillations with an amplitude of 20 meters at most.

The North-East position is the same as for PID, which is correct since it is a PID controller in heading.

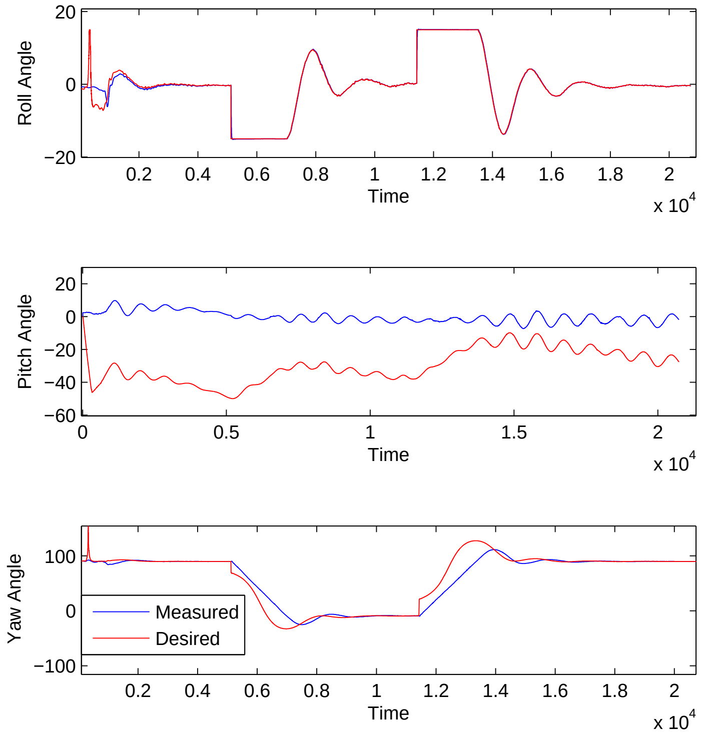

Figure 5.19: Roll, pitch and yaw angle [deg] with 20kt wind, measured vs desired

The only plot of interest is here the pitch angle in Figure 5.19. It never reaches its reference, but gets closer in the end. The desired pitch angle wants a angle of -40 degrees, but the aircraft has physical limitations on the elevators. The elevators have an operating area of -20 to 20 degrees, which is why the measured pitch never reaches desired pitch when desired is below -20 degrees.

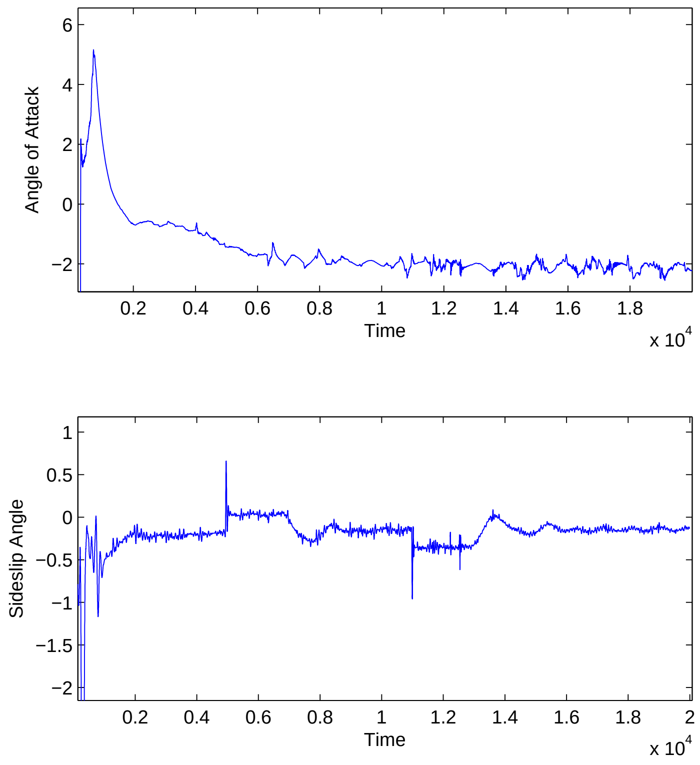

Figure 5.20: Angle of attack and side slip angle [deg] with 20kt wind, measured vs desired

Angle of attack and the sideslip angle, in Figure 5.20, behaves the same as for pure PID control in Figure 5.17. Only change is a somewhat bigger angle of attack in the beginning, at the start of takeoff. The huge spikes in the beginning is due to startup and initialization of the simulation.

5.4 With Wind 40 knots

Now the wind is increased to 40 knots. This is 5 knots above desired wind when flying a Cessna.

5.4.1 PID

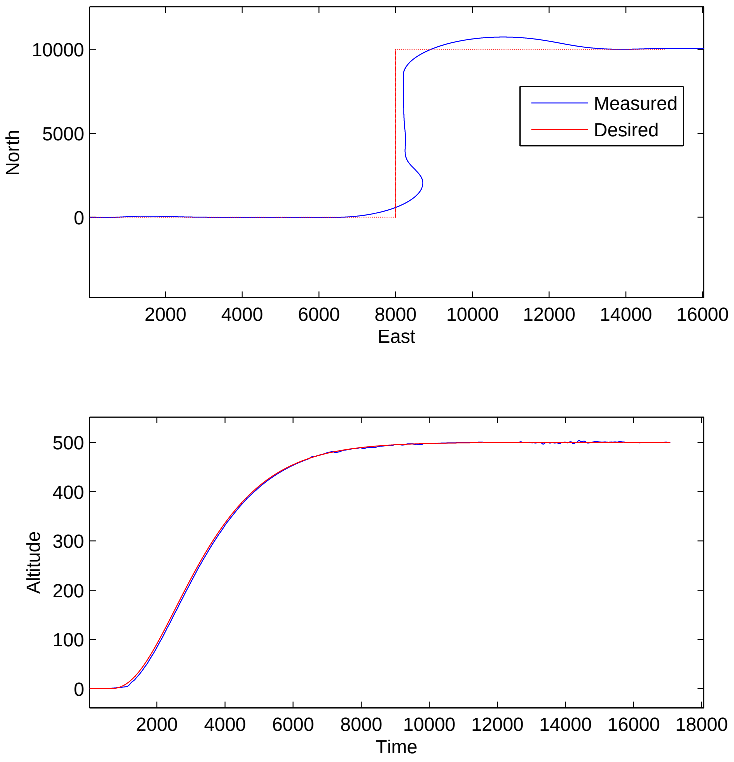

Figure 5.21: North-East and Altitude [m] with 40kt wind, measured vs desired

For simulation with 40 knots wind coming from east, the position gets an overshoot of 700 meters at both turning maneuvers, Figure 5.21. The error between desired path and measured position is approximately 200 meters when flying straight north. At the beginning it is following the desired path satisfactorily, small error of 50 meters at most, and in the end of the simulation it converges to the reference.

The altitude has small spikes of a couple of meters when reaching cruising altitude.

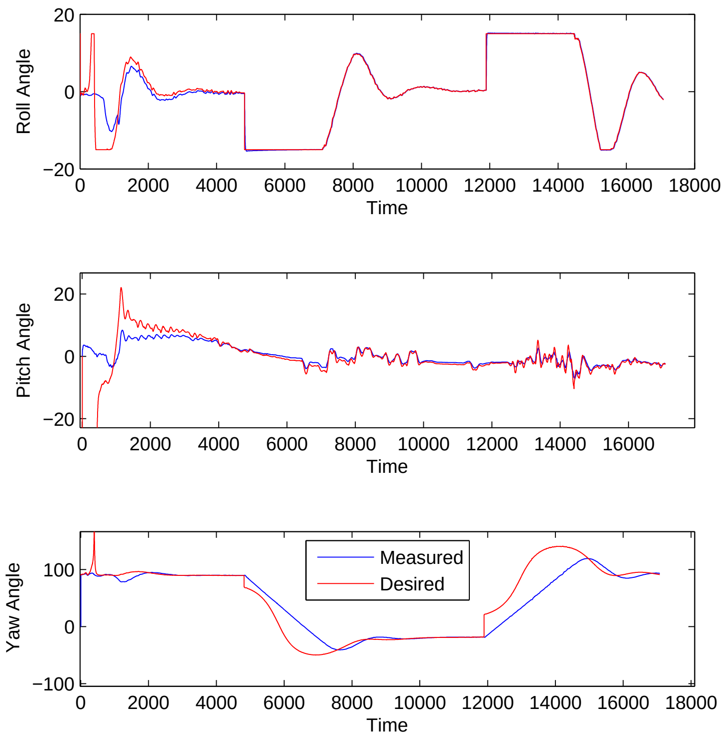

Figure 5.22: Roll, pitch and yaw angle [deg] with 40kt wind, measured vs desired

In the beginning of the simulation, there is an error between desired pitch angle and measured, Figure 5.22. After some time the measured follows the desired perfectly, with small spikes due to the wind disturbance.

The roll and yaw angle have the same behavior as for simulation with 20 knots wind disturbance, the error is a little bit bigger, which is natural because of the strong wind, but the controller gains are the same.

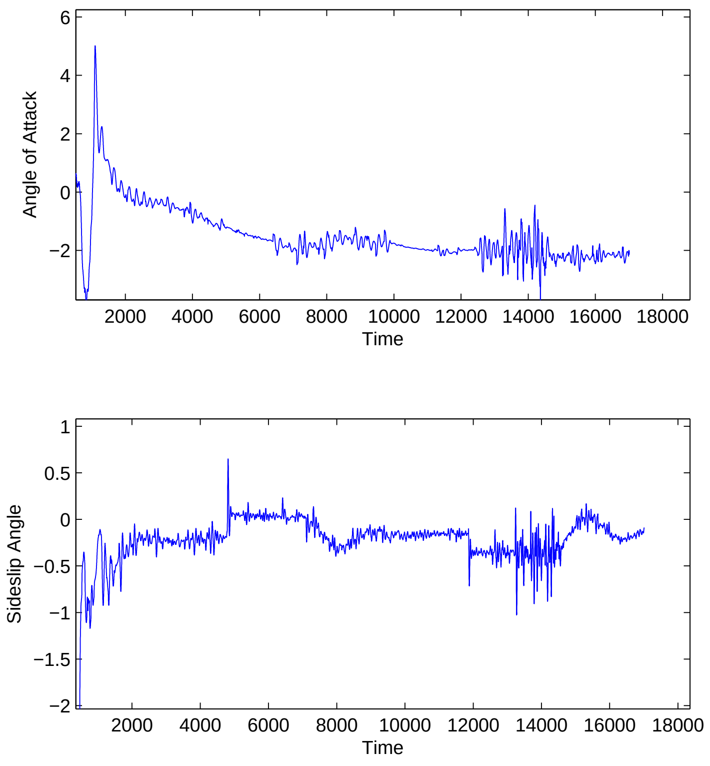

Figure 5.23: Angle of attack and side slip angle [deg] with 40kt wind, measured vs desired

The angle of attack is slightly bigger at the start of the takeoff and decreases as the aircraft hits it cruising speed, Figure 5.23, when the wind is increased with 20 knots.

The sideslip angle has bigger spikes for simulation with 40 knots wind than with 20 knots wind disturbance.

5.4.2 Sliding Mode

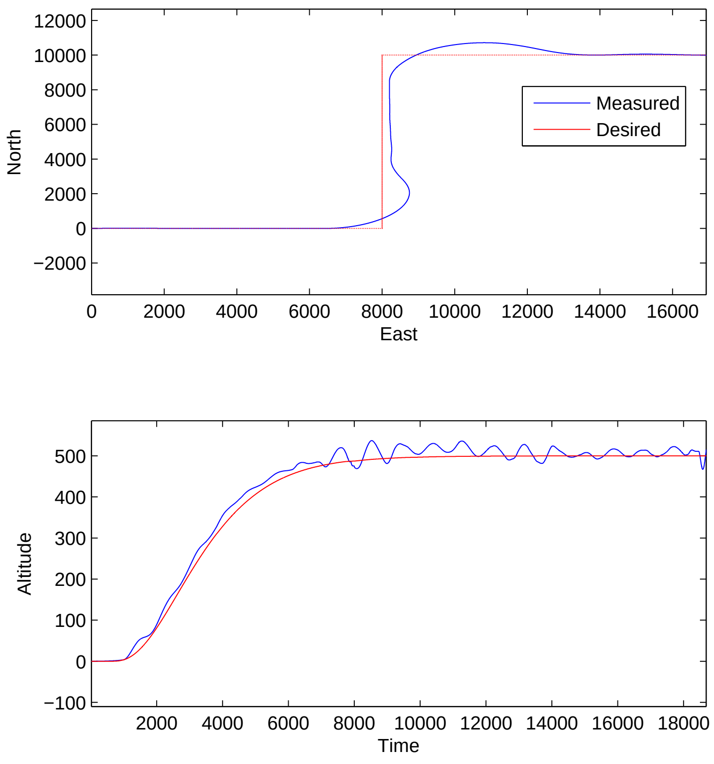

Figure 5.24: North-East and Altitude [m] with 40kt wind, measured vs desired

The sliding mode controller for pitch and altitude does not seem to get affected by the increased wind. From Figure 5.24 it can be seen that the altitude has the same behavior for simulation with 40 knots wind as for 20 knots wind, Figure 5.18.

The PID control in heading does not get affected by sliding mode control in pitch and altitude, and is the same as in Figure 5.21.

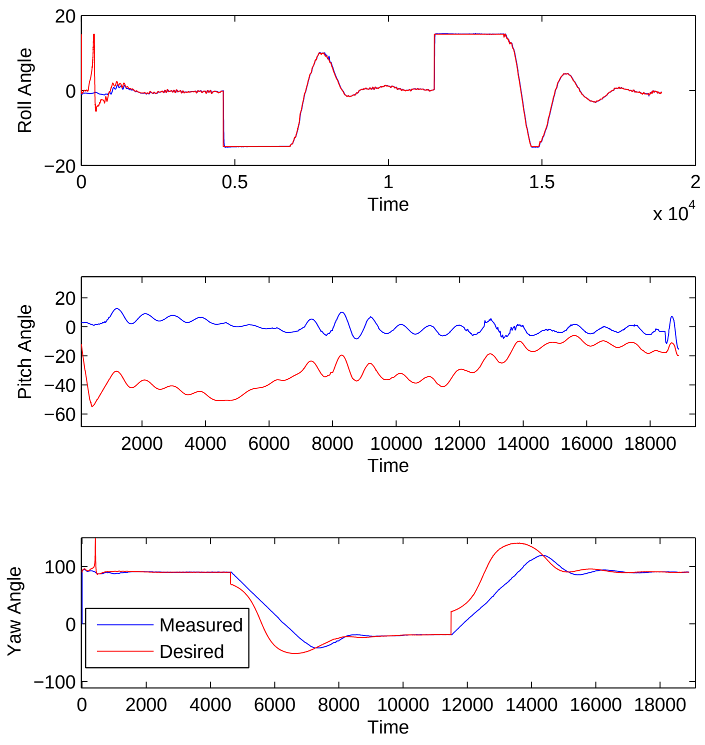

Figure 5.25: Roll, pitch and yaw angle [deg] with 40kt wind, measured vs desired

With 40 knots wind disturbance the pitch controller, Figure 5.25, seems to behave better than it did with 20 knots wind, Figure 5.19. It still has an huge error in the beginning due to the physical limitations, but the error is smaller in the end. However, it is still not successful.

Roll and yaw are controlled by PID and have naturally the same behavior as in Figure 5.22 for pure PID control.

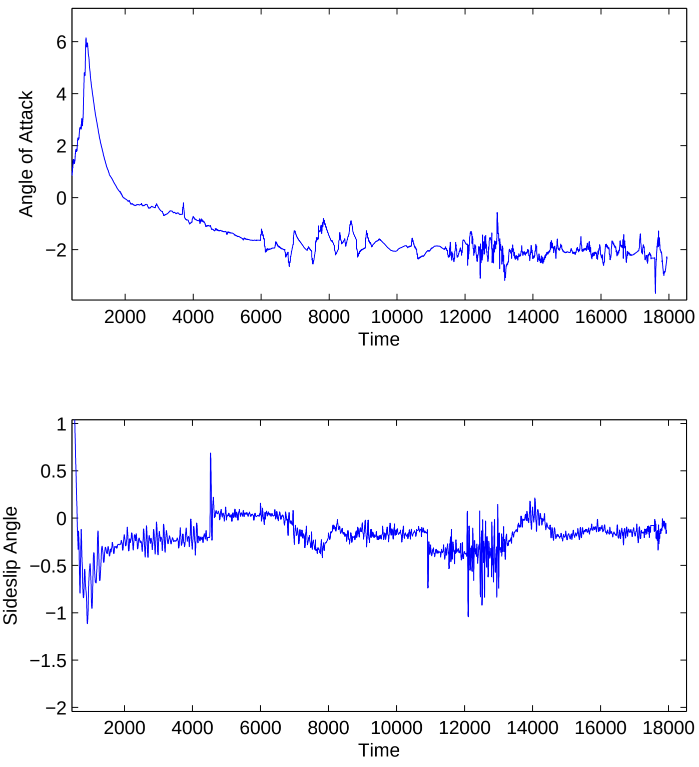

Figure 5.26: Angle of attack and side slip angle [deg] with 40kt wind, measured vs desired

For sliding mode control, the angle of attack, Figure 5.26, is smoother during takeoff than it is for PID control, Figure 5.23. It starts positive and converges to a constant negative angle as the aircraft reaches cruising altitude, with only small spikes occurring.

The sideslip angle is the same as for PID control. Sideslip angle corresponds to heading control which is preformed by PID control.

5.4.3 Discussion

Heading control is performed by PID while pitch and altitude control are performed by both PID and sliding mode. The following list summarize the most important observations made during the simulations:

- PID has good performances in simulation without wind disturbances and payload and with payload.
- PID can not follow desired path in North-East when exposed to wind, has an error of 100 and 200 meters when flying straight north at wind speed of 20 knots and 40 knots from the east, respectively.
- In heading, PID control, there is an overshoot of 40 meters from desired path when turning in simulations without wind disturbances and payload and with payload. The overshoot increases to approximately 400 meters with 20 knots wind speed and 700 with 40 knots wind speed.
- PID has small oscillations in altitude when exposed to wind.
- PID has slightly larger oscillations when exposed to 40 knots wind speed than when exposed to 20 knots wind speed.
- Sliding mode has oscillations in altitude in all four cases.
- Sliding mode has same performances when exposed to 20 knots wind speed as when exposed to 40 knots wind speed.
- PID follows desired roll, pitch and yaw angle nicely in all four cases, worst performance in yaw when exposed to wind.
- Sliding mode almost never reaches desired pitch. Desired pitch is sometimes greater than the physical limitations on the aircraft.
- Angle of attack is smoother for sliding mode than for PID control.
- Sideslip angle is close to zero during all four cases, but it has some spikes.

Based on these observations the PID controller will work fine in the autopilot design. With a better tuning, the sliding mode may have a better performance. Both controllers follow desired altitude, PID with some better result.

Chapter 6

Conclusions and Further Work

6.1 Conclusions

In this thesis a Guidance, Navigation and Control (GNC) system for use in autopilot design for an UAV has been designed and tested with a Cessna 172SP flight model in X-Plane flight simulator. The flight model has been unknown, a black box, during the simulation so that the guidance and control system had to operate without any flight model information. The aim for the GNC system was to takeoff from a runway, ascend to desired altitude and make a controlled turn.

The GNC system has been implemented with PID control in heading and PID or sliding mode control in pitch and altitude control. The controllers have been tested without wind disturbances and payload, with payload and with two different wind speeds.

From the comments and figures in Chapter 5 it can be shown that the PID controller was more successful than the sliding mode controller in all four cases. It was able to keep the desired altitude and follow the desired path in North-East satisfactorily. With harder tuning, especially when exposed to wind, the behavior would be even better. The PID controller has the advantages that it does not need model parameter information to control the system.

The sliding mode controller in pitch and altitude control followed desired altitude, but had oscillations with an amplitude of 20 meters at most. The biggest amplitude was when it was exposed to wind. The PID controller had a little worsening in pitch and altitude control when the wind speed increased, while the sliding mode had the same behavior.

The sliding mode controller was not able to follow the desired pitch angle, but this is because the desired pitch did not have any limitations, which the measured pitch had.

With a better aircraft model and a better method of tuning, the sliding mode controller would have had a better performance. The low pass filter makes the tuning even harder, but it is necessary to prevent wear and tear of the elevator.

Based on the simulations and observations in this thesis, the PID controller is preferable.

6.2 Further Work

To improve the GNC system used for for this autopilot, a better model of the aircraft is needed for use in sliding mode control. With more experience with sliding mode, the tuning could be done in a better way for further improvement.

Sliding mode control for heading should be tested with the improved aircraft model and a good tuning method. This may lead to a sliding mode with a better performance than the PID control.

When this GNC system will be applied to a real UAV, it will no longer have perfectly measured signals, they may drop out and measurement noise will occur. The GNC system do need an observer included to handle this.

During the simulations with wind, the wind has been set to only one direction and not varying in speed. For a good and safe performance on a real UAV, the GNC system do needs to handle gust wind and be able to takeoff with side wind.

Bibliography

B. M. Albaker and N. A. Rahim. Flight path PID controller for propeller-driven fixedwing unmanned aerial vehicles. 2011.

J. Balchen, T. Andresen, B. Foss, and N. teknisk-naturvitenskapelige universitet Institutt for teknisk kybernetikk. *Reguleringsteknikk*. Institutt for teknisk kybernetikk, 2003. ISBN 8247151472.

S. Bennett. Nicolas Minorsky and the Automatic Steering of Ships. 1984.

M. Breivik and T. I. Fossen. Guidance Laws for Autonomous Underwater Vehicles. 2008.

E. Børhaug and K. Y. Pettersen. Cross-track control for underactuated autonomous vehicles. 2005.

C. Carletti, A. Gasparri, S. Longhi, and G. Ulivi. Simultaneous Roll Damping and Course Keeping via Sliding Mode Control for a Marine Vessel in Seaway. 2011.

Cessna Aircraft Company. Cessna 172sp specifications, 2012. URL http://www.cessna.com/single-engine/skyhawk/skyhawk-performance.html.

C. Chen. *Linear System Theory and Design*. The Oxford Series In Electrical And Computer Engineering. Oxford University Press, 1999. ISBN 9780195117776. URL http://books.google.no/books?id=Fqi4oRgmOocC.

T. Fossen. Mathematical models for control of aircraft and satellites. 2011a.

T. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control.* John Wiley & Sons, 2011b. ISBN 9781119991496.

A. J. Healey and D. Lienard. Multivariable Sliding-Mode Control for Autonomous Diving and Steering of Unmanned Underwater Vehicles. 1993.

How Stuff Works. A Brief History of UAVs , 2012. URL `http://science.howstuffworks.com/reaper1.htm`.

IEEE Global History Network. Biography of Elmer A. Sperry, 2012. URL `http://www.ieeeghn.org/wiki/index.php/Elmer_A._Sperry`.

I. H. Johansen. UAV Autopilot Design for Recce D6. 2011.

L. J. N. Jones, C. P. Tan, Z. Man, and R. Akmeliawati. Preliminary Design of Sliding Mode Controller for Angualr Positional Tracking of an Aircraft. 2009.

H. Khalil. *Nonlinear systems.* Prentice Hall, 2002. ISBN 9780130673893. URL `http://books.google.no/books?id=t_d1QgAACAAJ`.

A. L. Salih, M. Moghavvemi, H. A. F. Mohamed, and K. S. Gaeid. Flight PID controller design for a UAV quadrotor. 2010.

R. Skjetne, T. I. Fossen, and P. V. Kokotovic. Robust output maneuvering for a class of nonlinear systems. 2003.

The Free Dictionary. The Free Dictionary - UAV, 2012. URL `http://www.thefreedictionary.com/unmanned+aerial+vehicle`.

Unmanned Aerial Vehicle System Association. Unmanned aerial vehicle system association - advantages, 2012a. URL `http://www.uavs.org/advantages`.

Unmanned Aerial Vehicle System Association. Unmanned aerial vehicle system association - uav or uas, 2012b. URL `http://www.uavs.org/advantages`.

V. Utkin. *Sliding Modes in Control and Optimization*. Communications and Control Engineering Series. Springer-Verlag, 1992. ISBN 9783540535164. URL http://books.google.no/books?id=uzNmQgAACAAJ.

R. Walpole. *Probability and statistics for engineers and scientists*. Prentice Hall, 2002. ISBN 9780130415295. URL http://books.google.no/books?id=CgNEAQAAIAAJ.

Wikipedia - PID. Pid controller — wikipedia, the free encyclopedia, 2012. URL http://en.wikipedia.org/w/index.php?title=PID_controller&oldid=490210451. [Online; accessed 3-May-2012].

X-Plane 9. X-plane 9 flight simulator, 2012. URL http://www.x-plane.com/desktop/landing/.

Appendix A

Definitions and Lemmas

North-East-Down (NED) coordinate frame, Fossen [2011b]

Definition 1. *NED, North-East-Down (NED) , reference frame* $\{n\} =$ *(*x_n*,* y_n*,* z_n*) has its origin in* o_n *and is defined relative to the Earth's reference ellipsoid. Usually it is defined as the tangent plane on the surface of the Earth moving with the craft and with the x-axis pointing North, y-axis pointing East and the z-axis pointing down to the center of Earth, vertical position.*

Controllability, Chen [1999]

Definition 2. *The state equation* $\dot{\boldsymbol{x}}=\boldsymbol{Ax}+\boldsymbol{Bu}$ *or the pair (*$\boldsymbol{A},\boldsymbol{B}$*) is said to be* controllable *if for any initial state* $\boldsymbol{x}(0) = \boldsymbol{x}_0$ *and any final state* $\boldsymbol{x}_1$*, there exist an input that transfers* $\boldsymbol{x}_0$ *to* $\boldsymbol{x}_1$ *in a finite time. Otherwise the state equation or (*$\boldsymbol{A},\boldsymbol{B}$*) is said to be* uncontrollable*.*

Barbalat's lemma, Khalil [2002]

Lemma 1. *Let* $\phi : R \to R$ *be a uniformly continuous function on* $[0,\infty)$*. Suppose that* $\lim_{t\to\infty} \int_0^t \phi(\tau)d\tau$ *exists and is finite. Then,*

$$\phi(t) \to 0 \quad as \quad t \to \infty$$

Appendix B

Attachment Description and Matlab Code

B.1 Attachment Description

Attached to this thesis is a folder named *Matlab*, where all the Matlab code and Simulink models are included, and a folder named *X-plane plugin*, where the plugin for simulation with X-Plane is included. Some of the Matlab code can be found in Appendix B.2. To set up your computer for X-Plane simulation, follow the small guide in Appendix C.

The code is structured as in Figure B.1 and organized as:

.m-files

- SimGNCSystem.m - The main run file, simulates the different case studies.
- initGNC.m - Initializes the aircraft and the GNC system.
- x_plane_interface.m - Initializes the X-Plane communication block.
- systemIdentification.m - System identification for finding a model to use in sliding mode control.
- ordlsq.m - Ordinary least square system identification function.

- SM_delta_e.m - Sliding mode for pitch and altitude.
- plots.m - Plots the simulated case.

.mat-files

- measuredSignals.mat - Storage of data for use in system identification, measured from X-Plane.
- delta.mat - Storage of data for use in system identification, aileron, elevator, rudder and thrust measurements.

.mdl-files

- GNCSystem_PID.mdl - The whole GNC system and X-Plane interface with PID control.
- GNCSystem_PID_SI.mdl - The whole GNC system and X-Plane interface with PID control and system identification measurements.
- GNCSystem_SM.mdl - The whole GNC system and X-Plane interface with sliding mode control.

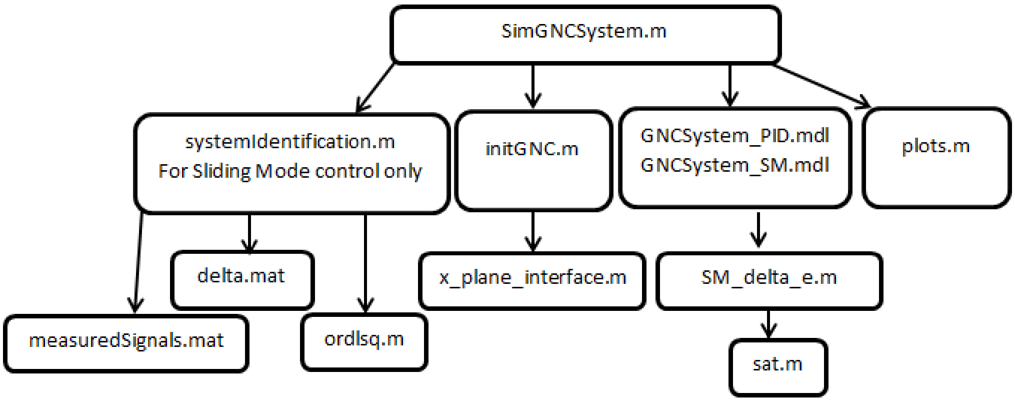

Figure B.1: Code Structure

B.2 Matlab Code

B.2.1 Simulation of GNC System

```
%% Simulation of simulink model GNCSystem_XXX.mdl
%
%
%
%   Author: Ingrid Hagen Johansen
%   Master Thesis Spring 2012
%   Date: 26.05.2012
%
%   Thanks to: Thor I. Fossen
%
%

clc;
clear all;
close all;


%% Initialize
%   Set waypoints [North East Altitude]
%   The Waypoints are selected for take
%   at Vaernes (ENVA), Norway.
%
    Way_points = [0 0 0;
                  0 8000 500;
                  10000 8000 500;
                  10000 32000 500];


    % Without payload
     payload = 0; % [kg]

    % With payload
    % payload = 378; % [kg]

    % Speed
    takeoff_speed = 80; %[kt]
    cruise_speed = 48;  %[kt]

    % If wind, +30 kt
```

```
      cruise_speed = cruise_speed + 30; %[kt]

     % System Identification
     systemIdentification
     poler = [0 -.00006 -0.00003];

%    Initialize

     initGNC


%% Simulation

%     Simulation with PID control
     sim('GNCSystem_PID')
%
%     Simulation with Sliding Mode Control
%     for pitch and altitude
% sim('GNCSystem_SM_pitch')

%% Plot

     plots
```

B.2.2 Initialize GNC System

```
%% Info
%
%    Initialization file for the Guidance,
%    Navigation and Control System (GNC)
%
%
%    Aircraft used in simulation is the Cessna 172SP
%        Length:     27 ft  2 in    (8.28 m)
%        Height:      8 ft 11 in    (2.72 m)
%        Wingspan:   36 ft  1 in   (11.00 m)
%        Wing area:  174 sq ft     (16.20 m^2)
%        Weight:     1,717 lb      (779 kg)
%
%        Max Takeoff Weight: 2,550 lb    (1 157 kg)
%        Max Payload Weight:   833 lb    (378 kg)
%
```

```
%
%   Author: Ingrid Hagen Johansen
%   Master Thesis Spring 2012
%   Date: 26.05.2012
%
%   Thanks to: Thor I. Fossen
%
%
%%  Initialize simulation
%
%

% Initialize simulation variables

    sim_start = 0.0;
    sim_stop = 400;

% Real time setings
    realtime.scalingfactor = 0.05;


% Physics

    % Payload
    init_payload = [payload 0 0 0 0 0;
                    0 payload 0 0 0 0;
                    0 0 payload 0 0 0;
                    0 0 0 0 0 0;
                    0 0 0 0 0 0;
                    0 0 0 0 0 0];

    physics.g = 9.81;
    mass = [743.44 0 0 0 0 0;
            0 743.44 0 0 0 -996.21;
            0 0 743.44 0 996.21 0;
            0 0 0 2008.76 0 0;
            0 0 996.21 0 3827.04 0;
            0 -996.21 0 0 0 5408.32]+ init_payload;



%% X-plane Interface

    x_plane_interface

```

```


%% Reference Model

    % Commanded Speed [kt]
    reference.commandedSpeed.time = [0 150 150 sim_stop];
    reference.commandedSpeed.signals.values = [takeoff_speed;
                                                takeoff_speed;
                                                cruise_speed;
                                                cruise_speed];
    reference.commandedSpeed.signals.dimension = 1;


    % Commanded Altitude
    reference.commandedHeight.time = [0 10 10 sim_stop];
    reference.commandedHeight.signals.values=[-Way_points(1,3);
                                              -Way_points(1,3);
                                              -Way_points(2,3);
                                              -Way_points(2,3)];
    reference.commandedHeight.signals.dimension = 1;

    reference.lp.height.omega = 0.05;


%% Guidance


    % Kp Kinematic Control Altitude
    guidance.altitude.Kp = 1;

    % LOS Guidance
    % Calculate waypoint matrix for North-East

   for i = 1:length(Way_points)
       for j = 1:1
           guidance.waypoints(i,j) = Way_points(i,j);
           guidance.waypoints(i,j+1) = Way_points(i,j+1);
       end
   end

    % Kp = 1 / Lookahead Distance
    guidance.los.Kp = 1/600;

    % Natural Frequency Altitude LP
    guidance.alt.lp.omega = 0.05;

    % Natural Frequency Speed LP
    guidance.speed.lp.omega = 0.2;


%%  Control System


    % Speed controller
        % PID gains
        control.speed.kp = 0.01;
        control.speed.ki = 0.002;
        control.speed.kd = 0.01;

        % Saturation
        control.speed.uppersaturation = 1;
        control.speed.lowersaturation = 0.2;


    % Roll stabilizer controller
        % PID gains
        control.RollAngle.kp = 0.6;
        control.RollAngle.ki = 0.02;
        control.RollAngle.kd = 0;


        % Saturation
        control.RollAngle.uppersaturation = 30*pi/180;
        control.RollAngle.lowersaturation = -30*pi/180;

        control.ailron.uppersaturation = 30*pi/180;
        control.ailron.lowersaturation = -30*pi/180;


    % Pitch angle controller
        % PID gains
        control.pitchAngle.kp = 0.5;
        control.pitchAngle.ki = 0.01;
        control.pitchAngle.kd = 0.01;

        % Saturation
        control.pitchAngle.uppersaturation = 20*pi/180;
        control.pitchAngle.lowersaturation = -20*pi/180;

        control.elevatorAngle.uppersaturation = 20*pi/180;
```

```
            control.elevatorAngle.lowersaturation = -20*pi/180;

            % Natural Frequency for Sliding Mode,
            % Deflection Elevator LP
            control.delta_e.lp.omega = 0.09;

        % Yaw angle controller
            % PID gains
            control.YawAngle.kp = 1;
            control.YawAngle.ki = 0.1;
            control.YawAngle.kd = 0.01;

            % Saturation
            control.YawAngle.uppersaturation = 15*pi/180;
            control.YawAngle.lowersaturation = -15*pi/180;



%%  Control Allocation

    allocation.B = [1 0 0 0;
                    0 1 0 1;
                    0 -1 0 -1;
                    0 0 -1 0;
                    0 0.02 0 0];
```

B.2.3 Plot Simulation

```
%% Plots result from simulation of GNCSystem_XX.mdl
%
%
% Author: Ingrid Hagen Johansen
% Master Thesis spring 2012
% Date: 26.05.2012
%
%
%

close all;

% Load Waypoints
WP = guidance.waypoints;
```

```


%% Desired vs Measured
figure(2)

% North−East
subplot(2,1,1)
%Measured
plot(eta(2),eta(1))
xlabel('East')
ylabel('North')

hold on;
%Desired
x = WP(1,1):50:WP(2,1);
y = WP(1,2):50:WP(2,2);
plot(y,x,'r')
hold on;
x = WP(2,1):50:WP(3,1);
y = WP(2,2):50:WP(3,2);
plot(y,x,'r')
hold on;
x = WP(3,1):50:WP(4,1);
y = WP(3,2):50:WP(4,2)−3000;
plot(y,x,'r')
legend('Measured','Desired', 'Location', 'Best')

% Altitude
subplot(2,1,2)
%Measured
plot(−eta.signals.values(:,3))
hold on
%Desired
plot(−reference.signals.values(:,2),'r')
xlabel('Time')
ylabel('Altitude')




% Roll, Pitch, Yaw
figure(3)
subplot(3,1,1)
plot(180*eta.signals.values(:,4)/pi)
hold on
```

```
plot(180*phi_desired.signals.values/pi,'r')
xlabel('Time')
ylabel('Roll Angle')


subplot(3,1,2)
plot(180*eta.signals.values(:,5)/pi)
hold on
plot(180*desired.signals.values(:,2)/pi,'r')
xlabel('Time')
ylabel('Pitch Angle')

subplot(3,1,3)
plot(180*eta.signals.values(:,6)/pi)
hold on
% plot(180*desired.signals.values(:,3)/pi,'r') %PID
plot(180*desired.signals.values(:,6)/pi,'r')   %Sliding Mode
xlabel('Time')
ylabel('Yaw Angle')

legend('Measured','Desired', 'Location', 'Best')

%% Angle of Attack and Sideslip Angle
figure(4)
for i = 1:length(alpha)
    alphaPlot(i) = alpha(:,:,i);
end

for i = 1:length(beta)
    betaPlot(i) = beta(:,:,i);
end

subplot(2,1,1)
plot(alphaPlot)
xlabel('Time')
ylabel('Angle of Attack')

subplot(2,1,2)
plot(betaPlot)
xlabel('Time')
ylabel('Sideslip Angle')
```

B.2.4 System Identification for Sliding Mode

```
%% System Identification for Pitch and Altitude
%
%
% Author: Ingrid Hagen Johansen
% Master Thesis spring 2012
% Date: 26.05.2012
%
%
%

%% Load Measured Data

measuredSignals = load('measuredSignals');
delta = load('delta');


%% Pitch and Altitude

% q_dot = [q, theta, delta_e]
y_q = measuredSignals.ans.data(:,13);
phi_q = [measuredSignals.ans.data(:,12);
         measuredSignals.ans.data(:,2);
         delta.ans.data(:,3)];

b_q = ordlsq(y_q,phi_q);
```

B.2.5 Sliding Mode Control for Pitch and Altitude Control

```
%% Sliding Mode Control for Pitch and Altitude Control
%
%
% Author: Ingrid Hagen Johansen
% Master Thesis spring 2012
% Date: 26.05.2012
%
%
%%

function delta_e = SM_delta_e(y)
%#codegen

```

```
b_q = y(1:3);
poler = y(4:6);
theta_d = y(7);
theta_d_dot = y(8);
q_d = y(8);
q_d_dot = y(9);
h_d = y(10);
h_d_dot = y(11);
U_0 = y(12);
q = y(13);
theta = y(14);
z = y(15);
alpha = y(16);


x = [q theta z]';
x_d_dot = [q_d_dot theta_d_dot h_d_dot]';

if U_0 == 0
    U_0 = 0.00002;
end


A = [b_q(1) b_q(2) 0; 1 0 0; 0 U_0 0];
b = [b_q(3) 0 0]';
p = [poler(1) poler(2) poler(3)];
f = [0 0 -U_0*alpha]';

k = place(A,b,p);
Ac = A-b*k;
[V,D] = eig(Ac');

for i = 1:length(b)
    hi = V(:,i);
    if norm(hi.'*Ac) < 1e-10; h = hi; end
end

k = [k(1) 0 k(3)]';
% k(2) is set to zero since it is one pure integrator in pitch

s = h(1)*(q-q_d) + h(2)*(theta-theta_d) + h(3)*(z-h_d);
eta = 0.1;

delta_e =  -k'*x + (1/(h'*b))*(h'*x_d_dot -h'*f -eta*sat(s));
```

Appendix C

How to set up X-Plane Interface

Here is a small guide on how to set up your computer for simulation with X-Plane:

1. Install X-Plane as described in the X-Plane installation guide. If sceneries of Værnes, Norway, is wanted, make sure to install them separately. Make sure that it gets installed in the folder "*c:\Program Files*".

2. Make sure that Matlab and Simulink are installed. Simulink must have Signal Processing Blockset and Marine System Simulator (MSS) toolbox in addition to the regular toolboxes. The MSS toolbox is developed by Prof. Thor I. Fossen and Prof. Tristan Perez at Norwegian University of Science and Technology (NTNU) and is available for free at `http://www.marinecontrol.org`.

3. Install X-Plane plugin "*SimulinkInterface.xpl*", made by Kjetil Hope Tufteland, from the X-Plane plugin folder attached. Put the plugin in the folder "*c:\Program Files\X-plane\Resources\plugins*". The plugin will run its installation at startup of the simulator.

Start the Simulation

1. Open SimGNCSystem.m from the Matlab folder.

2. Make sure to have the right IP- address in the X-Plane Communication block in Simulink models GNCSystem_PID.mdl, GNCSystem_PID_SI.mdl and GNCSystem_SM.mdl.

3. Start up X-Plane with CD 1 inserted into the DVD-drive.

4. Choose the *Cessna 172SP* under *Aircraft → Open Aircraft.*

5. Choose *Vaernes* under *Location → Select Global Airport.*

6. Press "*a*" to view the aircraft from outside and press "*v*" to release breaks.

7. Click *Plugins → Simulink interface via UDP → Start UDP communication.* X-plane will now freeze until the simulation starts.

8. Start Simulink simulation from SimGNCSystem.m.

Stop the Simulation

1. Click *Plugins → Simulink interface via UDP → Stop UDP communication.*

2. Stop the Simulink simulation.

If X-Plane is not stopped before Simulink, some sockets may not be closed and will cause problems for further simulations. If this is the case, restart the computer to make sure all sockets are properly closed.