NTNU
Norwegian University of
Science and Technology

Dead Reckoning System for UAV Using RSSI and Extremum Seeking Control

Daniel Røyland

Master of Science in Cybernetics and Robotics
Submission date: June 2016
Supervisor: Tor Arne Johansen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Abstract

Motivated by the vulnerability of Unmanned Aerial Vehicles (UAVs) to loss of GNSS, this master thesis investigates the development of a backup navigation system. The system can be used by the UAV to navigate back to the base station that the communication signal is transmitted from. The backup system is based on an extremum seeking controller that takes the RSSI and the current estimated heading as input, and outputs the heading that steers the UAV back to the origin of the communication signal, i.e. the base station. The extremum seeking controller is implemented in the DUNE framework and Software-in-the-loop tested with the JSBSim Flight Dynamics Model. Under the assumption made, the SIL tests show that the controller can indeed steer the UAV back to the vicinity of the base station. Flight tests conducted with the Skywalker X8 UAV verifies that the extremum seeking controller's embedded observer is able to estimate the RSSI accurately under real flight conditions. The results are discussed, and focus areas for further work are suggested.

Sammendrag

Motivert av såbarheten ubemannede fly har til GNSS-svikt, undersøker denne oppgaven utviklingen av et backup navigasjonssystem. Systemet kan brukes av det ubemannede flyet til å navigere tilbake til basestasjonen som kommunikasjonssignalet sendes fra. Backup-systemet er basert på en extremum seeking-kontroller som regner ut retningen som styrer det ubemannede flyet tilbake til basestasjonen, basert på RSSI og nåværende retning. Extremum seeking-kontrolleren er implementert i DUNE-rammeverket og Software-in-the-loop-testet med JSBSim Flight Dynamics Model. Med de antagelser som har blitt gjort, viser SIL-testene at extremum seeking-kontrolleren kan styre flyet tilbake til et lite område rundt basestasjonen. Testflyving utført med et Skywalker X8 ubemannet fly verifiserer at extremum seeking-kontrollerens innebygde observator er i stand til å estimere RSSIen nøyaktig under reelle forhold. Resultatene er diskutert, og fokusområder for fremtidig arbeid er foreslått.

Preface

This thesis is submitted in partial fulfillment of the requirements for the Master of Science degree at the Norwegian University of Science and Technology. The thesis is a continuation of my project thesis from autumn 2015.

I would like to thank my supervisor Tor Arne Johansen for inspiring ideas and valuable feedback. I would also like to thank my co-supervisors Arthur Zolich for his help with experimental testing, and Mark Haring, for his help on theoretical issues related to the extremum seeking controller. Kristian Klaussen should also be thanked for his feedback and help related to software development in the DUNE framework. My fellow MSc. students, both at room G232 and at the UAV-lab also desirves to be thanked. They have all contributed to making my final year here at NTNU more valuable, both socially and academically. Finally, I would like to thank my family and my wife for always supporting and encouraging me.

Daniel Røyland
Trondheim, June 5, 2016

Table of Contents

List of Tables

List of Figures

Abbreviations

BLOS	=	Beyond Line Of Sight
ES	=	Extremum Seeking
FDM	=	Flight Dynamics Model
GNSS	=	Global Navigation Satellite System
GPS	=	Global Positioning system
IMU	=	Inertial Measurement Unit
INS	=	Inertial Navigation System
lat	=	Latitude
lon	=	Longitude
RSSI	=	Radio Signal Strength Indicator
SIL	=	Software-in-the-loop
SITL	=	Software-in-the-loop
UAV	=	Unmanned Aerial Vehicle
VLOS	=	Visual Line Of Sight

Part I

Introduction and Background Theory

Chapter 1

Introduction

1.1 Motivation

The use of unmanned aerial vehicles (UAVs) is increasing rapidly, with both military and civilian application areas. Potential civil application areas include environmental monitoring, wildfire monitoring, ad hoc communication networks and search and rescue (SAR) (Beard and McLain (2012)). UAVs are also expected to play an increasingly important role in arctic monitoring, and the number of UAV-augmented science missions in the Arctic has gradually increased over the past decade (Marshall et al. (2012)). UAVs are providing data that would be difficult, impossible, risky or expensive to gather in any other manner, and measurements done with the use of UAVs can, amongst other uses, fill gaps in knowledge about weather, sea ice, ocean currents, pollution, marine mammals and fish (Marshall et al. (2012)).

UAVs rely on GNSS to determine their global position. This dependency increases the vulnerability of UAV operations, since GNSS can become unavailable due to a number of reasons, including receiver failure, spoofing, jamming, multipath, and in general, operating in areas where GNSS reception is poor. Because of the increasing availability of GNSS jamming technology, UAVs that rely heavily on GNSS are vulnerable to malicious systems (Conte and Doherty (2009)). Navigation systems that can cope with permanent GNSS outages are therefore necessary.

Assuming the communication link between the base station and the UAV is still operative, the UAV can exploit the information in the RSSI. The RSSI is among other factors related to the UAVs distance from the base station, and by flying in the direction that increases the signal strength of the communication signal, the UAV should end up back at the base station.

The goal of this project is to implement and simulate a backup navigation system that the UAV can use to navigate safely back to its base station in the case of permanent loss of GNSS, through the use of RSSI and extremum seeking control.

1.2 Previous work

One approach taken by researchers to solve the dead reckoning problem is to estimate the current position based on combining the INS state estimates with information from a vision system, and only use GNSS data opportunistically (Conte and Doherty (2009)). The vision system replaces the GNSS signal, combining position information from visual odometry and geo-referenced imagery. Geo-referenced satellite or aerial images must be available on-board UAV beforehand or downloaded in flight. The vision-aided navigation system developed is capable of providing high-rate and drift-free state estimation for UAV autonomous navigation without GNSS.

When it comes to simulated vehicle navigation based on measuring signal strength and extremum seeking, Cochran and Krsic (2007) have considered the problem of seeking the source of a scalar signal using an autonomous vehicle modeled as the nonholonomic unicycle. The unicycle did not have the capability to estimate its own position or the position of the source, but it was capable of measuring the signal originating from the source. The signal field was assumed to decay away from the source, but other than that, the unicycle had no knowledge of the functional form of the field. Extremum seeking control was applied to steer the vehicle to the source. They proved local exponential convergence to an “orbit-like” attractor around the source.

1.3 Contributions

The main contributions of this thesis is:

- Development of a software implementation of the extremum seeking controller described in (Haring and Johansen (2015)).
- Verification of the implementation using a unicycle simulator.
- Software-in-the-loop testing of the implementation with the JSBSim Flight Dynamics Model.
- Experimental testing of the embedded observer of the extremum seeking controller, including flight testing with the Skywalker X8 UAV.

1.4 Assumptions

In this section, the assumptions made in the following chapters are stated.

- **Assumption 1:** Assume that the communication line between the base station and the UAV is operative.
- **Assumption 2:** Assume that the attitude of the UAV has negligible influence on the RSSI.
- **Assumption 3:** Assume that the autopilot of the UAV can estimate the Euler angles roll, pitch and yaw accurately.

1.5 Structure of the thesis

This thesis is divided into 10 chapters and 1 appendix.

- Chapter 2 presents the background theory that is needed in the subsequent chapters. This includes in introduction to extremum seeking control and a description of the extremum seeking controller applied in this thesis, some background theory on path loss and RSSI and a description of the coordinate frames used.
- Chapter 3 describe the system components used in the thesis, including both software and hardware.
- Chapter 4 describes the software implemented, including the extremum seeking controller.
- In chapter 5, the simulation setups are presented and explained. The extremum seeking controller is tested with both a simple unicycle simulator and a well-known Flight Dynamics Model, namely the JSBSim FDM.
- In chapter 6, the experimental work done is described. This includes a concept test performed at campus, where the UAV fuselage is carried around, and a proper flight test performed at the airfield at Agdenes. In both experiments, the measured and estimated RSSI is logged for later analysis and discussion.
- The results are presented in chapter 7 and 8, where chapter 7 presents the results from the simulations performed and chapter 8 presents the results from the experimental testing performed.
- Chapter 9 discusses the results presented in Chapter 7 and 8. Further, recommendations for future work are given.
- Chapter 10 sums up the thesis with a closing conclusion.
- Appendix A shows an example configuration file and lists the commands needed to run DUNE and JSBSim.

Chapter 2

Theory

This chapter explains the background theory and information that is needed to understand the material that is presented throughout the thesis.

2.1 Extremum seeking control

Extremum seeking is a form of adaptive control that optimizes the steady-state output of a cost function, where a model of the cost function is unavailable. While most adaptive controllers deal only with regulation to known set points, extremum seeking control can be used to optimize performance in the cases where the optimal set point in unknown. The unknown cost function can be written

$$y(t) = F(\boldsymbol{x}(t)) \tag{2.1}$$

where e.g. $\boldsymbol{x}(t)$ can be position and $y(t)$ can be radio signal strength around a base station. Extremum seeking requires that the input-output characteristics of the cost function is available and that it has an extremum (Tan et al. (2010)), as can be seen in Figure 2.1. In Figure 2.1, $\mathbf{x}$ is the input to the system and y is the output. d is an unknown disturbance.

Extremum seeking is a gradient based optimization method, and relies on a sufficient exploration of the cost function to be able to provide an estimate of the gradient, and hence move towards the extremum. Extremum seeking control is a high-level control scheme, i.e. its goal is to maximize performance, not to stabilize the plant.

Extremum seeking methods have been applied to a large variety of applications, and have increased in popularity the last 15 year. Applications include autonomous vehicles (Zhang et al. (2007)), process control (Tan et al. (2010)), brake system control and solar cell and radio telescope antenna adjustment to maximize the received signal and blade adjustment in water turbines and wind mills to maximize the generated power (Krstic and W (2000)).

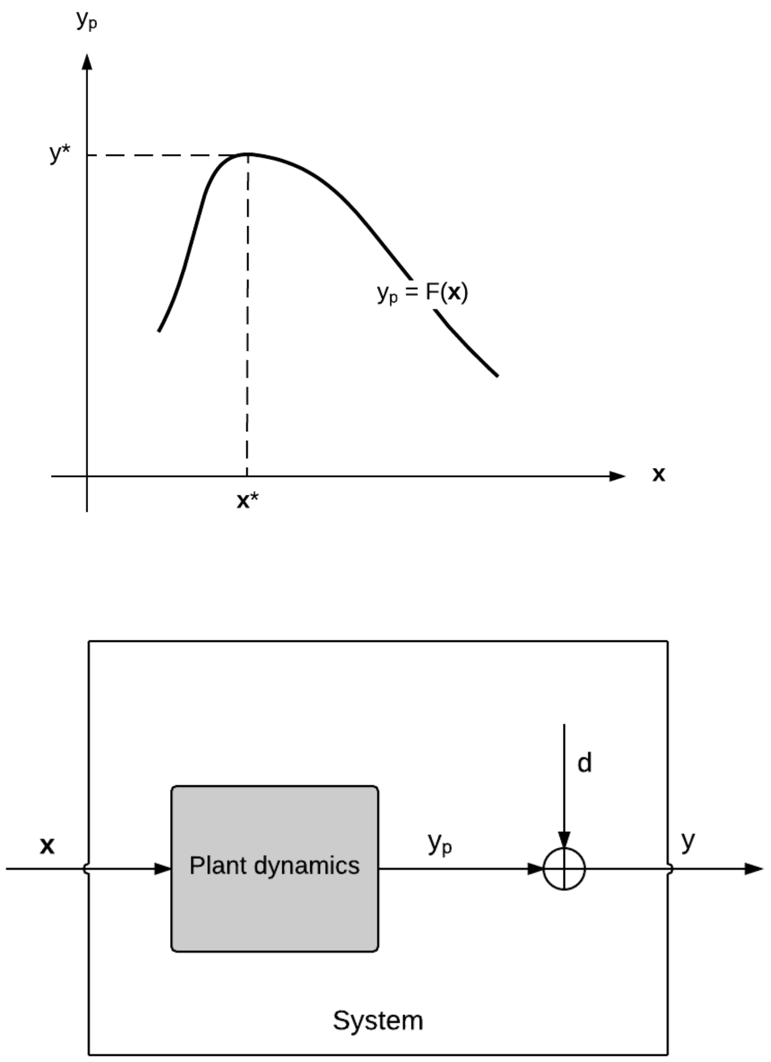


Figure 2.1: Input-output system

2.2 The extremum seeking controller applied in this thesis

The extremum seeking controller applied in this project is a discrete time version of the extremum seeking controller described in (Haring and Johansen (2015)). The input for the controller is the discrete RSSI measurement $y_k \in \mathbb{R}$ and the output is the heading $\psi_k \in \mathbb{R}$ that steers the UAV to the base station, e.g. where $y_k = y_{max}$. More specifically, the heading ψ_k will lead to horizontal circling motions that converge to the maximum of the unknown cost function $F(\boldsymbol{x}(t))$. Here, $F(\boldsymbol{x}(t))$ is the RSSI as a function of horizontal position. It is assumed that F has a global maximum for an unknown position $\boldsymbol{x}(t) = \boldsymbol{x}^\star$. In Eq. (2.1), $\boldsymbol{x}(t)$ is the horizontal position of the UAV, and $y(t)$ is the RSSI value.

The controller consists of an observer part and an optimization part.

Observer.
The observer takes the following form:

$$\hat{\boldsymbol{z}}_{k|k} = \hat{\boldsymbol{z}}_{k|k-1} + \boldsymbol{L}_k(y_k - \boldsymbol{C}\hat{\boldsymbol{z}}_{k|k-1}) \tag{2.2}$$

$$\boldsymbol{P}_{k|k} = \frac{1}{\lambda}(\boldsymbol{I} - \boldsymbol{L}_k\boldsymbol{C})\boldsymbol{P}_{k|k-1}(\boldsymbol{I} - \boldsymbol{L}_k\boldsymbol{C})^T + \frac{1}{1-\lambda}\boldsymbol{L}_k\boldsymbol{L}_k^T \tag{2.3}$$

where $\hat{\boldsymbol{z}}_{(\cdot)} \in \mathbb{R}^3$ is the state vector and $\boldsymbol{P}_{(\cdot)} > 0 \in \mathbb{R}^{3x3}$ is a positive definite matrix. The first element of $\hat{\boldsymbol{z}}_{(\cdot)}$ is the estimate of the cost function value y_k, and the second and third elements are estimates of the gradient of the map F. The matrices $\mathbf{C}$ and $\boldsymbol{L}_k$ are defined as

$$\boldsymbol{C} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{2.4}$$

$$\boldsymbol{L}_k = \frac{1}{\lambda}\boldsymbol{P}_{k|k-1}\boldsymbol{C}^T\left(\frac{1}{1-\lambda} + \frac{1}{\lambda}\boldsymbol{C}\boldsymbol{P}_{k|k-1}\boldsymbol{C}^T\right)^{-1} \tag{2.5}$$

The notation $(\cdot)_{k|k}$ means the value of $(\cdot)_k$ given measurement y_k and $(\cdot)_{k|k-1}$ means the value of $(\cdot)_k$ given the previous measurement y_{k-1}. λ is a tuning parameter which satisfies $0 < \lambda < 1$.

The prediction step of the observer is:

$$\hat{\boldsymbol{z}}_{k+1|k} = \boldsymbol{A}_k\hat{\boldsymbol{z}}_{k|k} \tag{2.6}$$

$$\boldsymbol{P}_{k+1|k} = \boldsymbol{A}_k\boldsymbol{P}_{k|k}\boldsymbol{A}_k^T \tag{2.7}$$

with

$$\boldsymbol{A}_k = \begin{bmatrix} 1 & \omega T\boldsymbol{R}^T(\psi_{k+1}) \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{2.8}$$

Optimizer.
The optimal heading is calculated according to the following update law:

$$\psi_{k+1} = \psi_k + \omega T(1 - \boldsymbol{R}^T(\psi_k + \frac{\omega T}{2})\frac{\kappa\eta \boldsymbol{D}\hat{\boldsymbol{z}}_{k|k}}{\eta + \kappa||\boldsymbol{D}\hat{\boldsymbol{z}}_{k|k}||}) \tag{2.9}$$

where

$$\mathbf{D} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

and

$$\mathbf{R}(\psi) = \begin{bmatrix} \cos(\psi) \\ \sin(\psi) \end{bmatrix} \tag{2.11}$$

$\kappa, \omega > 0$ and $0 < \eta < 1$ are tuning parameters. T is the sampling time. It can be seen that $\mathbf{D}\hat{\mathbf{z}}$ is an estimate of the gradient of F.

Tuning parameters: ω is inversely proportional to the radius of the circle motions in the UAV flight trajectory; A large ω will result in small circles and a small ω will result in large circles. While large circling makes it more likely that the cost function output measurements are rich enough to estimate the gradient, it could also increase the total UAV flight distance. λ is known as the forgetting factor and can be interpreted as the memory of the observer. If λ is close to zero, old measurements will be forgotten fast, and if λ is close to one, old measurements will be used for a long time. η determines the maximal speed in which $\mathbf{x}$ moves towards $\mathbf{x}^*$. The closer η is to one, the higher the maximal speed. κ is related to the convergence of $\mathbf{x}$ to $\mathbf{x}^*$. If κ is too large, large overshoot can occur; if κ is too small, the convergence will be slow.

2.3 Dead reckoning

Dead reckoning deals with the problem of estimating an object's current position any velocity, and predicting a future position by projecting the course and speed from a previously known position (Bowditch and Logan (1914)).

2.4 Path loss and RSSI

When an electromagnetic wave propagates through space, a loss in its power density is experienced. Path loss is influenced by several different factors such as free-space path loss, absorption, reflection, terrain contours, distance between, and height of, transmitter and receiver (Standard (1996)).

RSSI (Received Signal Strength Indication) is a measurement of the power present in a received radio signal (Sauter (2010)). RSSI is defined in IEEE 802.11 as the relative received signal strength in a wireless environment. The 802.11 standard does not define a specific unit for RSSI, nor a relationship between RSSI and power level in mW or dBm. In practice however, RSSI is often measured in dBm, which is the power ratio in decibels (dB) of the measured power referenced to one milliwatt (mW), i.e. 0 dBm is equivalent to 1 mW (Standard (1996)).

2.5 Coordinate frames

Earth-centered reference frames

ECI: The Earth-centered inertial (ECI) frame $\{i\} = (x_i, y_i, z_i)$ with origin o_i can be considered as an inertial frame for terrestrial navigation, e.g. a non-accelerating reference frame where Newton's laws of motion apply. The origin o_i is located in the center of the Earth (Fossen (2011)).

ECEF: The Earth-centered Earth-fixed (ECEF) frame $\{e\} = (x_e, y_e, z_e)$ also has its origin o_e in the center of the Earth, but it rotates relative to the ECI frame around the z-axis with the Earth rotation ω_e. The ECEF frame is thus not an inertial frame (Fossen (2011)).

Geographic reference Frames

NED: The North-East-Down (NED) frame $\{n\} = (x_n, y_n, z_n)$ with origin o_n is defined relative to the Earth's reference ellipsoid (World Geodetic System, 1984). The z-axis points downwards normal to the Earth's surface, the x-axis points towards true north and the y-axis points towards east to complete the orthogonal coordinate system. The location of {n} relative to {e} is determined by using the two angles μ (latitude) and l (longitude) (Fossen (2011)).

BODY: The body-fixed frame $\{b\} = (x_b, y_b, z_b)$ with origin o_b is fixed in the vehicle, meaning that it moves and rotates with the vehicle. The x-axis points in the forward direction, the y-axis to the right and the z-axis downwards. The Body-frame is related to the NED-frame through the Euler angles roll, pitch and yaw (Fossen (2011)).

Chapter 3

System components

The goal of this chapter is to give the reader the sufficient background knowledge about the different system components, so that he or she can make full sense of the work described in the following chapters.

3.1 Software

This section describes the software that has been used during the work with the thesis.

3.1.1 Ardupilot

Ardupilot is an open source autopilot, that includes both software, firmware and hardware. Ardupilot has a rich history and a great online community. It supports fixed-wing planes, multirotors, helicopters and rovers. Ardupilot provides low-level attitude control, state estimation with Extended Kalman Filter and dual IMU support. It also provides high-level mission execution with waypoints and automatic landing and takeoff (ArduPilot (2016)).

3.1.2 JSBSim Flight Dynamics Model

In order to do realistic SIL tests, a decent Flight Dynamics Model, or FDM, is necessary. In this thesis, the JSBSim FDM is used. JSBSim is an open source, 6 DOF, platform independent that is popular in the UAV community (Sourceforge (2016)).

3.1.3 LSTS toolchain

LSTS (Laboratório de Sistemas e Tecnologia Subaquática) is an interdisciplinary research laboratory established in 1997. LSTS specializes on design, construction and operation of underwater, surface and air vehicles. LSTS has developed a toolchain that is extensively used at the UAV-lab. The toolchain is described in the following sections.

GLUED

GLUED (GNU/Linux Uniform Environment Distribution) is a minimal Linux distribution targeted at embedded systems (LSTS (2016c)). It is the operating system used at most of the UAVs at the UAV-lab.

IMC

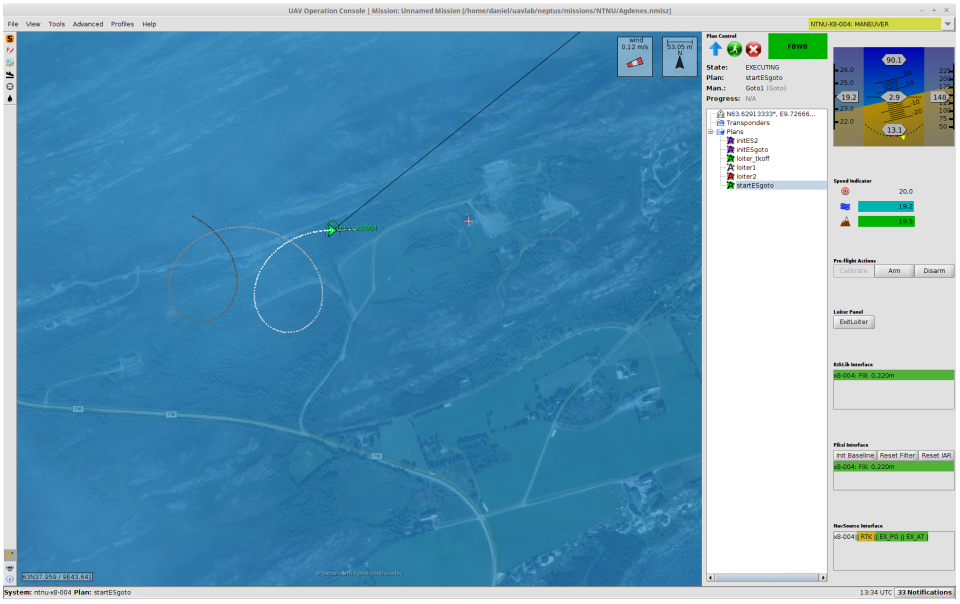
IMC, or Inter-Module Communication (IMC) protocol is a message-oriented protocol designed to build interconnected systems of vehicles, sensors and human operators (LSTS (2016d)). It is designed to be used together with the rest of the LSTS toolchain. IMC does not assume any specific software architecture for client applications.

DUNE

DUNE: Unified Navigation Environment (DUNE) is an open-source on-board software running on the vehicle. It is a runtime environment used to write generic embedded tasks in C++. Tasks are relatively small programs that runs in separate treads of execution. The tasks communicate over the IMC bus through the commands *consume* and *dispatch*. This setup allows for a high degree of modularity. DUNE is responsible for interaction with sensors, payload and actuators. DUNE is also responsible for communication, navigation, control, maneuvering plan execution and vehicle supervision. It is both CPU and operating system independent (LSTS (2016a)).

Neptus

Neptus is an open-source command and control software used to command and monitor unmanned vehicles. It is written in Java and is currently running on both Linux and Windows operating systems (LSTS (2016e)). It supports the different phases of a typical mission life cycle: planning, simulation, execution and mission review analysis. Neptus communicates with DUNE through IMC messages. A screenshot of the Neptus console is shown in Figure 3.1.



State: EXECUTING
Plan: startESgoto
Man.: Goto1 (Goto)
Progress: N/A

x8-004: FIX: 0,220m

x8-004: FIX: 0,220m

x8-004 || RTK || EX_PO || EX_AT |

System: ntnu-x8-004 Plan: startESgoto

Figure 3.1: Neptus Console

3.2 Hardware

This section describes the hardware that has been used during the work with the thesis.

3.2.1 Skywalker X8

The Skywalker X8 is a fixed wing UAV, meaning that is has no tail. It has no clear distinction between its fuselage and wing. The specifications of the X8 is given in Table 3.1 (LSTS (2016f)).

Wingspan	2120 mm
Length	600 mm
Weight	From 1.5 kg to 3.5 kg
Endurance	Up to 60 min
Wind tolerance	Mean 14 knots, Max 18 knots

Table 3.1: X8 specifications

The X8 is very popular for experimental work at the UAV-lab at the Department of Engineering Cybernetics, due to its low cost, durability and large community. In addition, the glider shape and relatively small size allows for a logistically simple setup with short preparation time (LSTS (2016f)). A picture of the X8 is shown in Fig. 3.2 (Institute of Engineering Cybernetics (2016)).

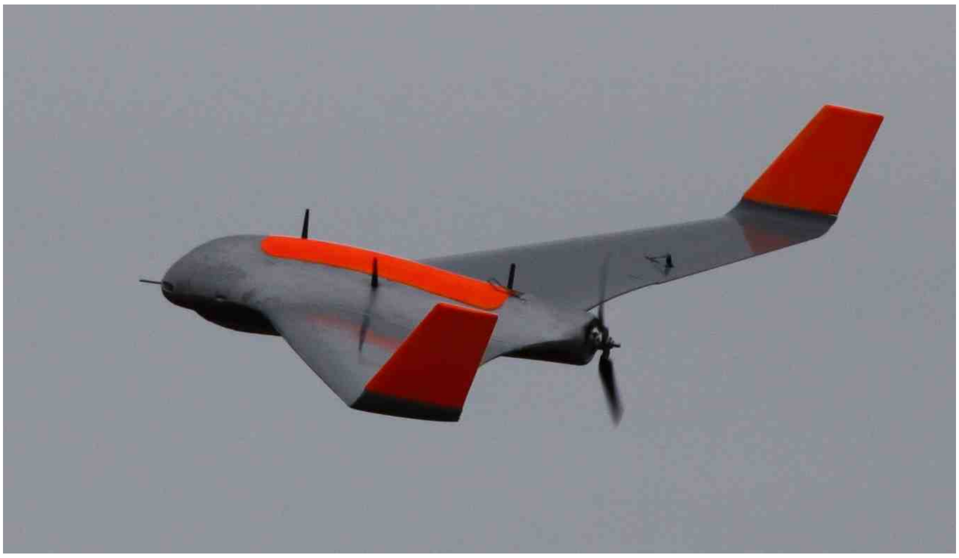


Figure 3.2: Skywalker X8

3.2.2 3DRobotics Pixhawk autopilot

For the field experiments conducted in this thesis, the 3DRobotics Pixhawk was used as the autopilot of the UAV. It is designed by the PX4 open-hardware project and it features processor and sensor technology from ST Microelectronics and a NuttX real-time operating system (Beagleboard (2016)). The firmware consists of two parts, the middleware made by PX4 and the flightstack. At the UAV-lab, the Ardupilot flightstack is used.

3.2.3 BeagleBone Black embedded computer

In order to test the software developed in this thesis, an embedded computer that runs the LSTS toolchain was needed. In addition, since the fuselage of the X8 has limited space, the embedded computer needed to be relatively small. The BeagleBone Black embedded computer fulfilled these requirements and has therefore been adopted by the UAV-lab. It is a low-cost, small development platform that runs Linux (Glued). The operating system is on a memory card and it has a 1 GHz, 32-Bit Sitara AM335x ARM Cortex-A8 Microprocessor. For more information, see (3DRobotics (2016)).

3.2.4 Communication link

At the UAV-lab, the main communication link used between the X8 and the base station is the Ubiquiti Rocket M5, 5.8 GHz. It runs Ubiquiti's Time Division Multiple Access (TDMA) AIRMax protocol. It can be used with a directional antenna with tracker or and omnidirectional antenna. The Rocket M5 can be used for 50+ km distances, depending on the antenna used. It also provides a speed of 150+ Mbps (data-alliance (2016)).

The X8 also has a VHF long range radio which is mainly used both for backup and for BLOS operations.

Part II

Methods

Chapter 4

Implementation

This chapter describes the tasks developed in this thesis, namely the RSSI simulator, the unicycle simulator, the RSSIGradient and the RSSIExtremum tasks. For completeness, the task BankToTurn, which has not been made by the author, is briefly described. The tasks were implemented in the DUNE framework using the C++ programming language. All tasks are configurable through a vehicle-specific configuration file. The configuration file defines which tasks should run and what the task parameters should be, see Appendix A.1. For the commands needed to run DUNE together with Ardupilot, see Appendices A.2 and A.3.

4.1 DUNE Tasks

4.1.1 RSSI simulator task

In a relatively homogeneous landscape, the main factor influencing the RSSI is the distance from the transmitter to the receiver. Because of this, only this distance was considered when the RSSI simulator task was implemented.

The task binds to the following IMC message:

- *IMC::EstimatedState*

And dispatches the following IMC message:

- *IMC::RSSI*

The position of the base station is stored in the task and, as it can be seen in Figure 5.1, the task listens to the *IMC::EstimatedState* message on the IMC bus to acquire the current position of the vehicle. The RSSI value is then computed based on these two positions and then dispatched to the IMC bus using the *IMC::RSSI* message. Since the unit of the *IMC::RSSI* value is %, the computations are done so that the resulting answer is a value between 0 and 100. The RSSI is computed with the following formulas:

$$dist = \sqrt{x_p^2 + y_p^2} \tag{4.1}$$

$$RSSI = 100 \cdot e^{-dist \cdot 0.001} \tag{4.2}$$

where x_p and y_p is the East-North displacement of the vehicle, relative to the base station. It can be seen that the expression for the RSSI value was chosen so that the value decreases exponentially when the vehicle is moving away from the base station. In addition, the expression for the RSSI is chosen so that the RSSI is at 50 $\%$ of maximum at around 700 m and at 10 $\%$ after around 2300 m.

To make the simulated RSSI more realistic, a random number generator was implemented. For every simulated RSSI value, a random number was added to the value, to simulate white noise. Further, on average 10 percent of the RSSI values dispatched to the IMC bus were set to zero, to simulate a temporary lost connection.

4.1.2 Unicycle task

In the task, the vehicle is represented as a simple unicycle, i.e. a point with a position in NED, a velocity and a current heading. Height above ground is assumed constant.

The task binds to the following IMC message:

- *IMC::DesiredHeading*

And dispatches the following IMC message:

- *IMC::EstimatedState*

The simple unicycle equations are used to simulate the behavior of the vehicle. These equations are given in Equations 4.3 and 4.4.

$$\dot{x}_p = V \cdot \cos \psi_p \tag{4.3}$$

$$\dot{y}_p = V \cdot \sin \psi_p \tag{4.4}$$

As seen in Figure 4.1, (x_p, y_p) is the position of the unicycle and ψ_p is the heading. V is the velocity.

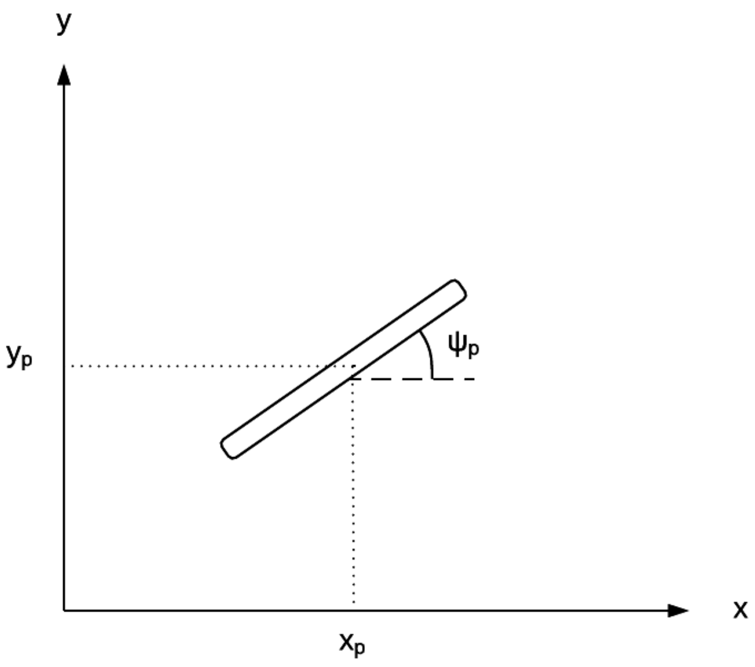


Figure 4.1: Unicycle Illustration

In the task, it is assumed that the desired heading coming from the RSSIExtremum task equals the current heading of the vehicle, i.e. $e_\psi = \psi_d - \psi_p = 0$. Equations 4.3 and 4.4 are integrated using the Forward Euler integration method to obtain Equation 4.5.

$$\boldsymbol{x}_{k+1}^p = \boldsymbol{x}_k^p + T \cdot V \cdot \boldsymbol{R}(\psi_k^p) \tag{4.5}$$

where

$$\boldsymbol{x}_i^p = \begin{bmatrix} x_i^p \\ y_i^p \end{bmatrix} \tag{4.6}$$

and

$$\boldsymbol{R}(\psi_k^p) = \begin{bmatrix} \cos(\psi_k^p) \\ \sin(\psi_k^p) \end{bmatrix} \tag{4.7}$$

and T is the time step. The task runs periodically with $T = 0.01$ s and $V = 20$ m/s. The updated vehicle position and heading is stored in the *IMC::EstimatedState* message and dispatched to the IMC bus.

4.1.3 RSSIGradient task

In this task, the observer described in Section 2.2 was implemented. This task is responsible for computing the state vector $\hat{z}_{k|k}$, from Eq. 2.2, based on the new RSSI value, the heading of the UAV and the prediction $\hat{z}_{k|k-1}$.
The task binds to the following IMC messages:

- *IMC::RSSI*
- *IMC::EstimatedState*

And dispatches the following IMC message:

- *IMC::NavigationData*

The IMC::EstimatedState message is consumed in order to obtain the current heading of the UAV. When the tasks consumes a new RSSI measurement, it computes a new state vector $\hat{z}_{k|k}$ Since the interval T between each received RSSI measurement is generally unknown, this interval is computed every time $\hat{z}_{k|k}$ is computed.

In addition to the equations described in Section 2.2, functionality that detects and disregards an RSSI measurement that suddenly drops to zero was implemented. The reason for implementing this functionality was that during an early RSSI logging performed, it was observed that it occurred regularly that RSSI values sent over the IMC bus were zero, even though the distance between the two communication nodes was relatively short.

$\hat{z}_{k|k}$ is stored in a *IMC::NavigationData* message and then dispatched to the IMC bus.

4.1.4 RSSIExtremum task

In this task, the optimizer described in Section 2.2 was implemented. This task is responsible for calculating the optimal desired heading based on the state vector $\hat{z}_{k|k}$ coming from the RSSIGradient task. In the same manner as the RSSIGradient task, the interval T is computed every time a new state vector is received.
The task binds to the following IMC message:

- *IMC::NavigationData*

And dispatches the following IMC message:

- *IMC::DesiredHeading*

In addition, RSSIExtremum inherits the BasicUAVAutopilot class, see LSTS (2016b) The main reason for this is to let the RSSIExtremum task be activated from Neptus through the use of the *IMC::ControlLoops* message.

4.1.5 BankToTurn task

This task has not been made by the author of this thesis, but since the task has a key role in the SIL setup, a brief description is given here.
The task binds to the following IMC messages:

- *IMC::DesiredHeading*
- *IMC::EstimatedState*
- *IMC::ControlLoops*

And dispatches the following IMC message:

- *IMC::DesiredRoll*

The task translates a desired heading angle to a desired roll angle. This is done with a PID controller that computes the roll angle that minimizes the difference between the estimated heading and the desired heading. This translation is necessary since the autopilot requires a desired roll angle. The reason for this becomes clear if one studies the shape of the X8 UAV. Since the X8 has no tail, and thus no rudder, it uses only its ailerons to steer. The difference between the two aileron angles determines the roll angle.

Chapter 5

Simulations

This chapter describes the simulations done. In both simulation setups, all the work was done in DUNE and Neptus was used as the ground control center. The difference between the two setups described is that in the first setup, the unicycle model was used to simulate the behavior of the UAV, while in the second setup, the JSBSim FDM was used. The main motivation behind the first setup was that debugging and coarse tuning could be done a lot faster than if the JSBSim FDM was used.

The SIL testing is a very important stage of the development phase, since the code used for the SIL testing essentially is the same as the code used in real flights.

5.1 Unicycle test system simulations

To simplify debugging and to make sure that the extremum seeking controller implemented did conceptually what it should do, a simplified test system was implemented. The test system consists of four DUNE tasks, namely the Unicycle task, the RSSI task, RSSIgradient task and the RSSIExtremum task. A block diagram of the setup is shown in Figure 5.1.

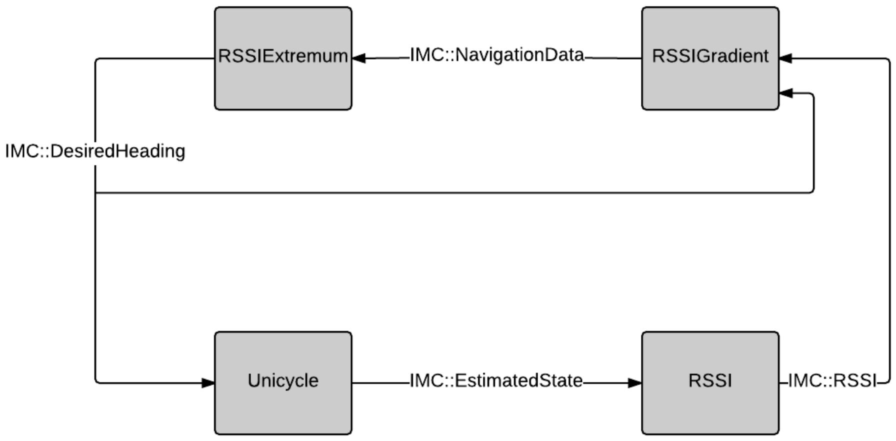


Figure 5.1: Unicycle test setup block diagram

With this setup, it was not necessary to run an external FDM, since the vehicle simulator was simply another DUNE task. This made debugging the extremum seeking controller much easier, since potential errors in the autopilot, the FDM or, in particular, the communication between the autopilot and DUNE, were ruled out.

5.2 Software-in-the-loop simulations

For the SIL testing, the JSBSim simulator was used instead of the Unicycle Simulator from Section 5.1. Figure 5.2 shows how the different parts of the SIL setup communicates. Here, the emphasis is to show how the different parts made in the thesis communicate, not the entire Ardupilot/DUNE/Neptus communication flow. In the figure, StartES is a command from Neptus to initiate the RSSIExtremum controller, the four gray boxes are the DUNE tasks described in Section 4.1, and the yellow box is the JSBSim FDM and the Ardupilot autopilot.

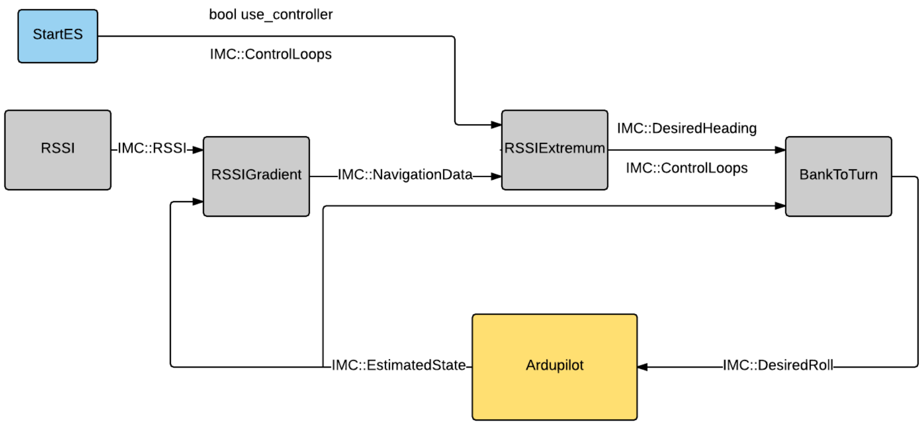


Figure 5.2: Simplified SIL test setup block diagram

Different flight scenarios were simulated, including varying initial distance between the UAV and the base station. A longer distance is more relevant when it comes to actual use of the extremum seeking system, but not very practical for experimental testing. For experimental tests, it is desired that the UAV stays within a VLOS area, so that a pilot easily can take control of the UAV if it turns out that something is not working as it should.

In addition, simulations are done for different intervals between each received RSSI measurement. In the field experiments, an interval of 2 seconds was used, but this is not always the case.

Chapter 6

Experimental work

This chapter describes the experimental testing of the Extremum Seeking controller. Due to limited time, only the observer was tested. Two experiments were conducted. The first experiment at the university campus at Gløshaugen, Trondheim and the last experiment at the airfield at the Agdenes airfield, located about 90 km south-west of Trondheim. While the experiment conducted at Gløshaugen was performed to verify that all the hardware and software worked as it should, the experiment conducted at Agdenes was performed to get a more realistic view of what the measured and estimated RSSI actually looked like during a flight.

6.1 Concept test, Gløshaugen

In order to verify that the task RSSIGradient, described in Section 4.1.3, was able to estimate the RSSI appropriately when used with the intended hardware, a field experiment was performed. An additional purpose of the experiment was to observe the characteristics of the RSSI of the Rocket M5 communication link, and how it varied with distance between base station and UAV.

6.1.1 Experiment equipment

The following equipment was used in the experiment.

- X8 Fuselage
- Beaglebone Black (BBB) Embedded Computer
- Pixhawk Autopilot
- Ubiquiti Rocket M5 5.8 GHz Communication Link
- Batteries for embedded computer, autopilot and communication link

- Base Station (Nest)
- Laptop with Neptus

A picture of the equipment used is shown in Figure 6.1.

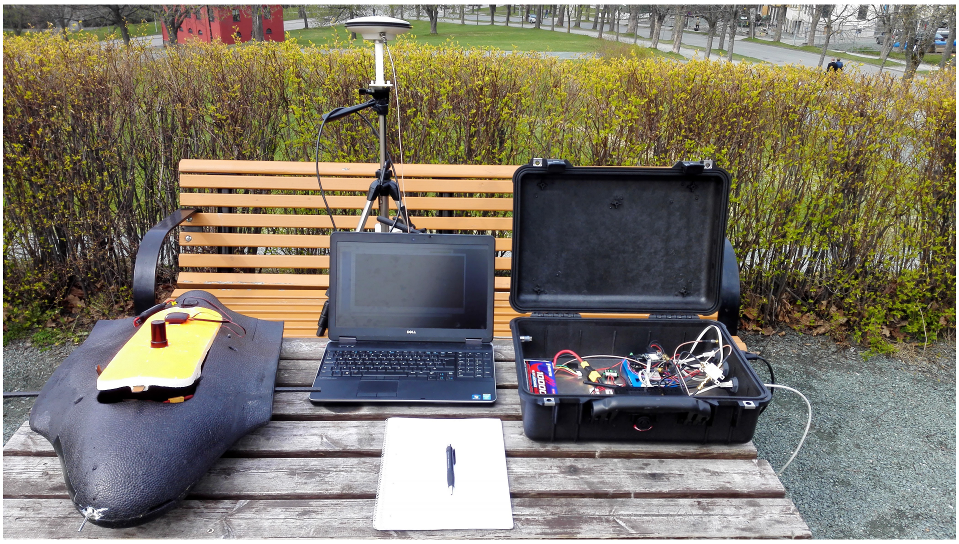


Figure 6.1: Field experiment test setup

6.1.2 Preparations for the experiment

Prior to the experiment, the configuration (.ini) file had to be updated with all the tasks that should run and the appropriate DUNE branch had to be cross-compiled and sent to the Beaglebone Black.

6.1.3 Conduction of experiment

In the experiment, the X8 fuselage with payload was carried around in different patterns. In the first experiment, the UAV was carried while walking in a straight line first away from and then back to the base station. In the second experiment, the UAV was carried while walking in a zigzag pattern away from the base station. In the third experiment, the UAV was carried while walking in a circling pattern towards the base station.

A picture of the X8 fuselage with the payload installed is shown in Figure 6.2.

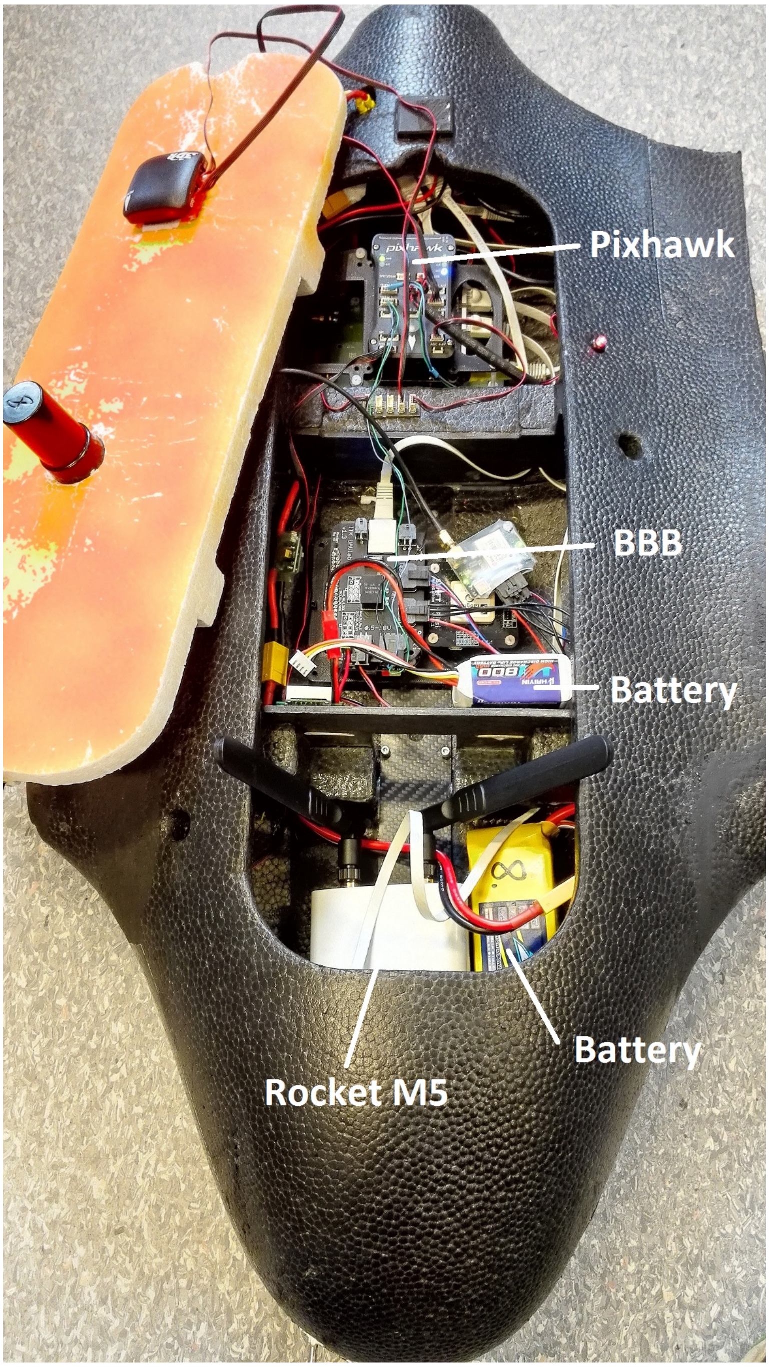


Figure 6.2: X8 fuselage with payload

6.2 Test flight, Agdenes airfield

The purpose of the experiment was essentially the same as in the first experiment, only this time the goal was to see what the measured and estimated RSSI looked like during an actual flight.

6.2.1 Experiment equipment

The equipment used the the experiment was essentially the same as in Section 6.1, with some differences: The complete Skywalker X8 was used, not just the fuselage. A communication antenna with longer range was used, in addition to the necessary equipment needed to take off and fly the UAV in manual mode and to monitor the UAV during flight.

6.2.2 Preparations for the experiment

Prior to the experiment, the configuration (.ini) file had to be updated with all the tasks that should run and the appropriate DUNE branch had to be cross-compiled and sent to the Beaglebone Black. A pre-flight check list was used to verify that the UAV was ready for take-off, and that it could be controlled in manual mode by a ground operator. This was performed by the pilots that led the test flight.

6.2.3 Conduction of experiment

In the experiment, the X8 flew in a circling, descending motion about 200 meters south-east of the base station, starting at 170 meters above ground and finishing at 100 meters above ground. The X8 flew autonomously, following a precomputed flight plan, while the pilot stood by to take over the UAV if needed.

Part III

Results

Chapter 7

Simulation results

This chapter presents the main results from the simulations, both the unicycle simulation results and the SIL simulation results. In order to gain a deeper understanding of how the system performs under different conditions, multiple scenarios were simulated.

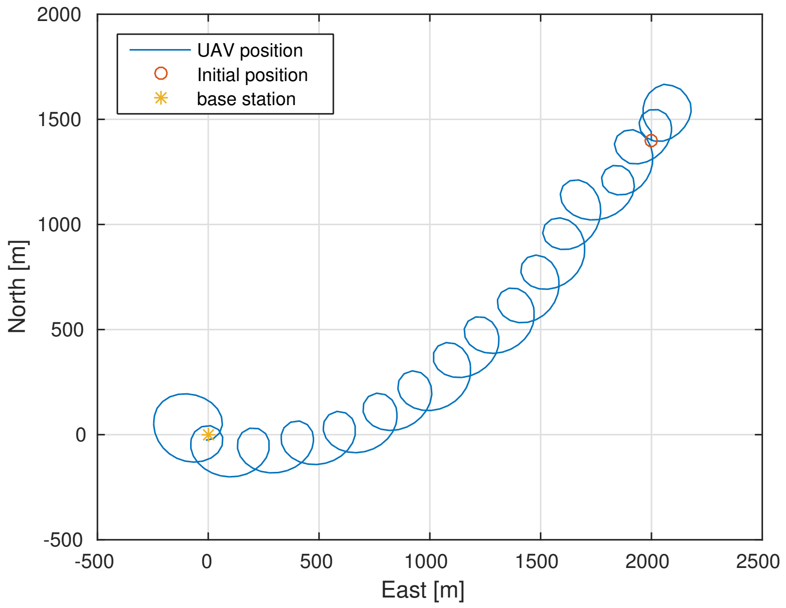
7.1 Unicycle simulation results

In this section, the unicycle simulation results are presented. 3 different parameter configurations were simulated, in order to verify that the extremum seeking controller was working the way it should. In all simulations, the RSSI frequency, λ, η and V_d were kept fixed according to Table 7.1, while ω and κ were varied. Configuration 1 is presented more in-depth, while configurations 2 and 3 focuses more on the difference from configuration 1.

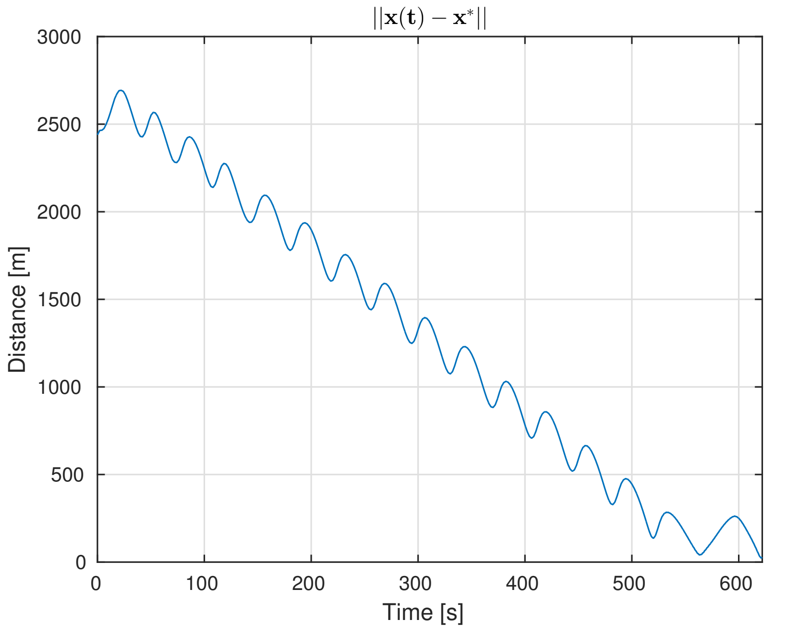
Parameter	Value
f_{RSSI}	0.50 Hz
V_d	20.0 m/s
λ	0.90
η	0.50

Table 7.1: Parameters, unicycle simulations

Configuration 1
In this configuration, ω was set to 0.20 and κ set to 100. The initial position of the vehicle was set to 2000 meters east and 1500 meters north of the base station. Figure 7.1a shows the flight trajectory of the vehicle and Figure 7.1b shows the euclidean distance between the vehicle and the base station as a function of time. Further, Figure 7.2a shows the measured and estimated RSSI and Figure 7.2b shows the estimated gradient of the unknown cost function, both as a function of time.

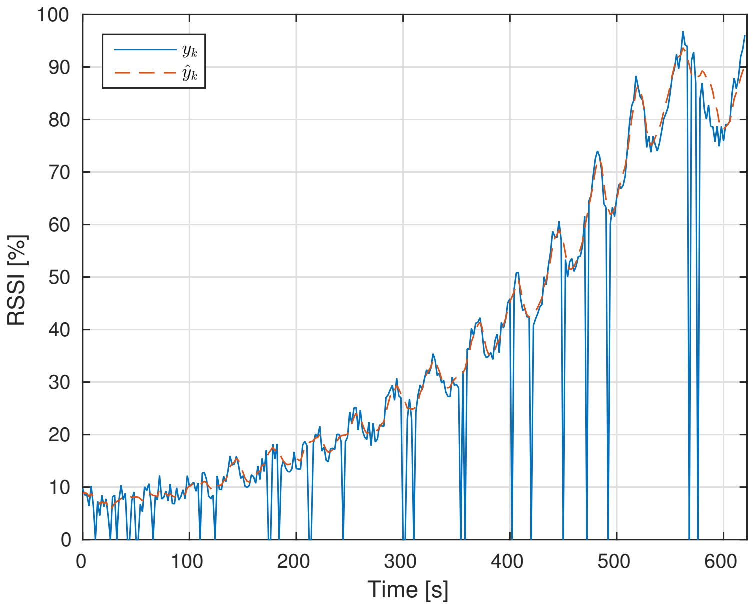


(a) UAV flight trajectory

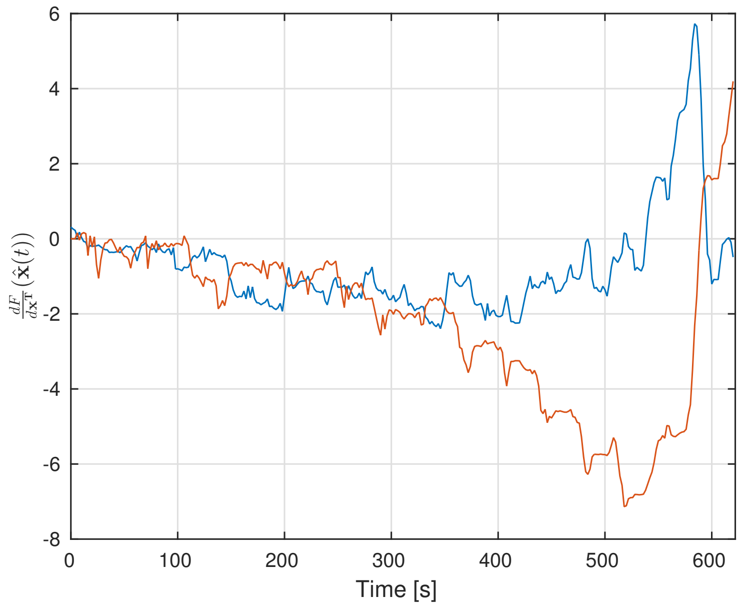


(b) Convergence of solution

Figure 7.1: Unicycle simulations



(a) Measured vs. estimated RSSI



(b) Estimated cost function gradient

Figure 7.2: RSSI and observer plots

Configuration 2

In this configuration, ω was set to 0.20 and κ set to 0.10. The initial position of the UAV was set to 600 meters east and 1000 meters north of the base station. Figure 7.3 shows the flight trajectory of the vehicle.

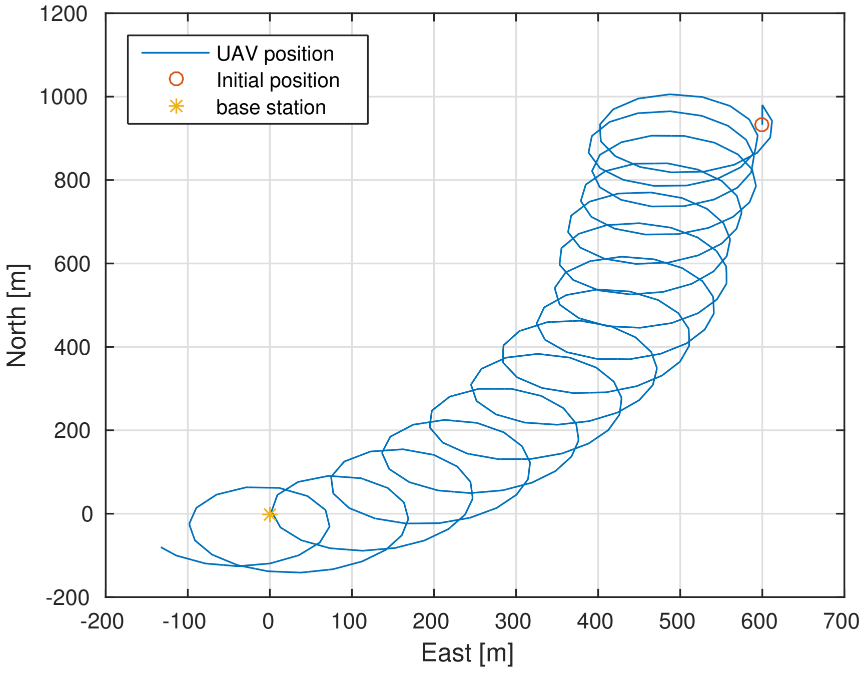


Figure 7.3: UAV flight trajectory, configuration 2

Configuration 3

In this configuration, ω was set to 0.10 and κ set to 100. The initial position of the UAV was set to 600 meters east and 800 meters north of the base station. Figure 7.4 shows the flight trajectory of the vehicle.

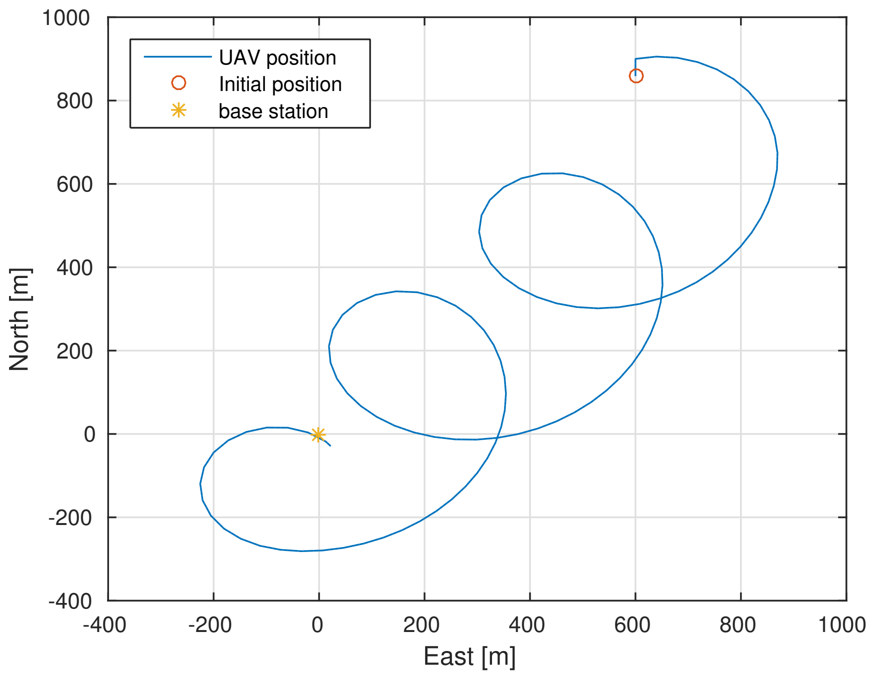


Figure 7.4: UAV flight trajectory, configuration 3

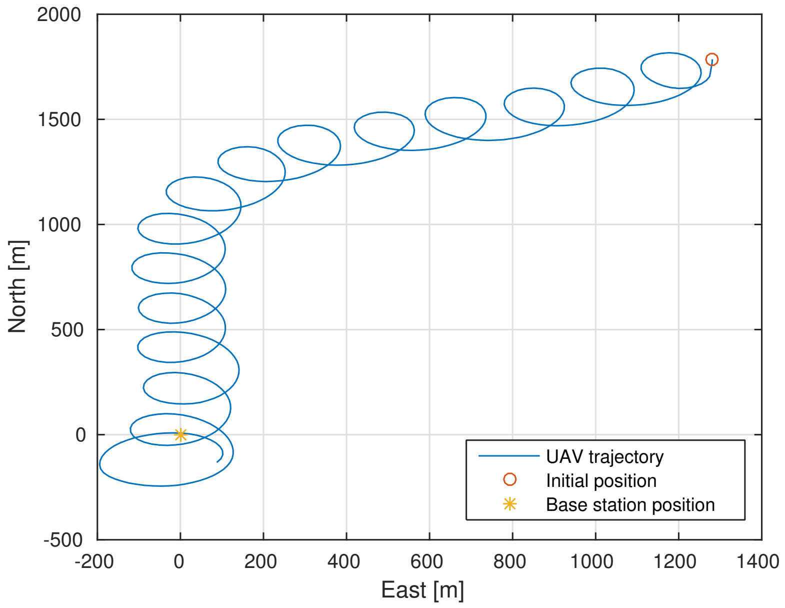
7.2 JSBSim FDM SITL simulation results

This section presents the main results from the Software-In-The-Loop simulations. 3 different configurations are presented. In the different configurations, ω, η and the desired velocity V_d were kept fixed according to Table 7.2, while the initial distance between the UAV and the base station, the forgetting factor λ, as well as the frequency the RSSI measurements arrive with (f_{RSSI}) were varied.

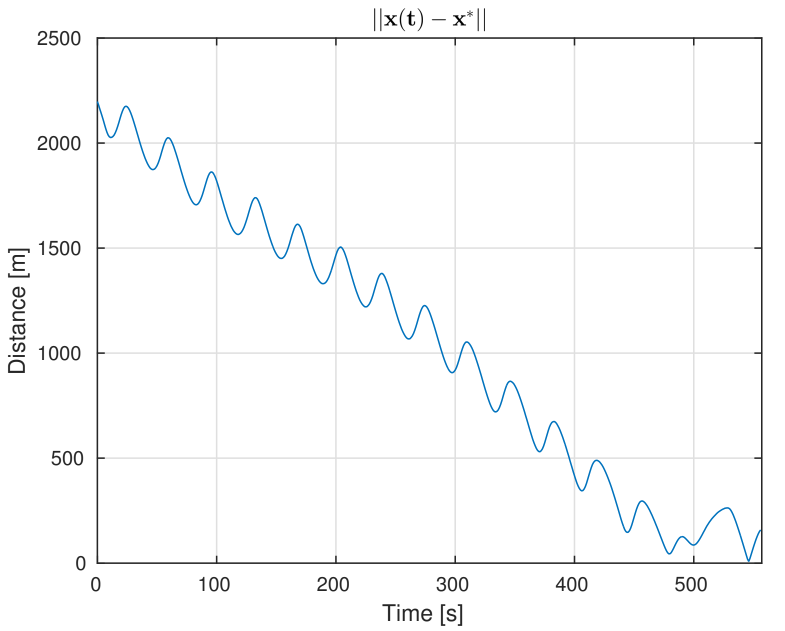
Parameter	Value
ω	0.20
η	0.50
V_d	20.0 m/s

Table 7.2: Parameters, SIL simulations

Configuration 1
In this configuration, f_{RSSI} was set to 0.5 Hz, λ was set to 0.90 and the initial position of the vehicle was set to 2000 meters east and 1500 meters north of the base station. Figure 7.5a shows the flight trajectory of the vehicle and Figure 7.5b shows the euclidean distance between the vehicle and the base station as a function of time. Further, Figure 7.6a shows the measured and estimated RSSI and Figure 7.6b shows the estimated gradient of the unknown cost function, both as a function of time. The measured RSSI is denoted y_k and the estimated RSSI is denoted $\hat{y}_k$. Figure 7.7 shows the estimated and desired heading as a function of time. The desired heading is denoted ψ_d and the measured heading is denoted ψ_m.

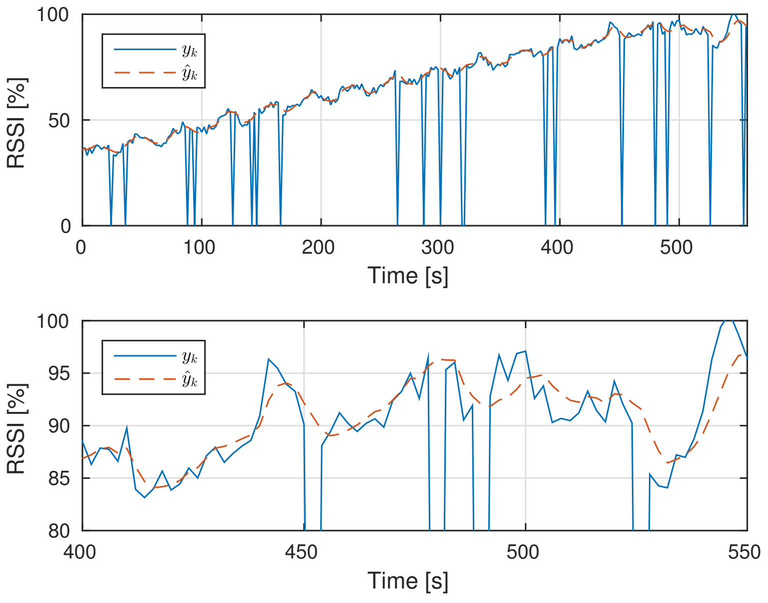


(a) UAV flight trajectory

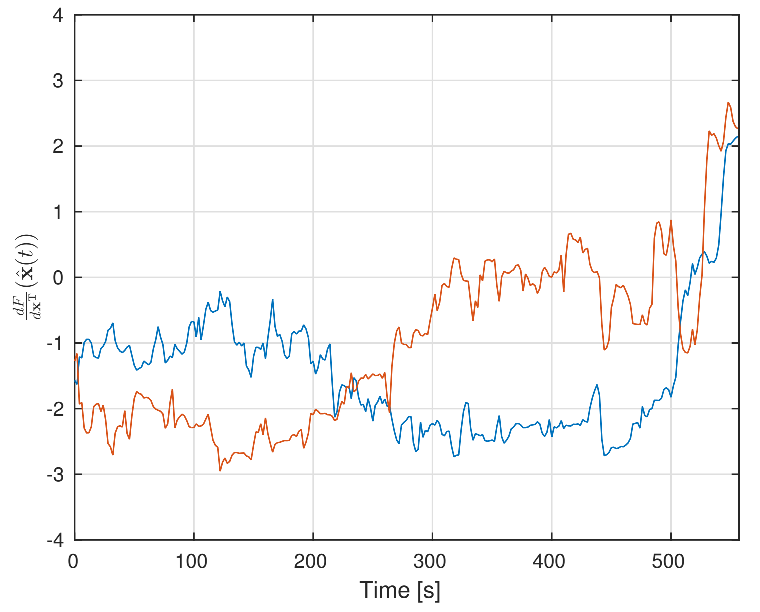


(b) Convergence of solution

Figure 7.5: Configuration 1 flight simulation



(a) Measured vs. estimated RSSI



(b) Estimated cost function gradient

Figure 7.6: RSSI and observer plots

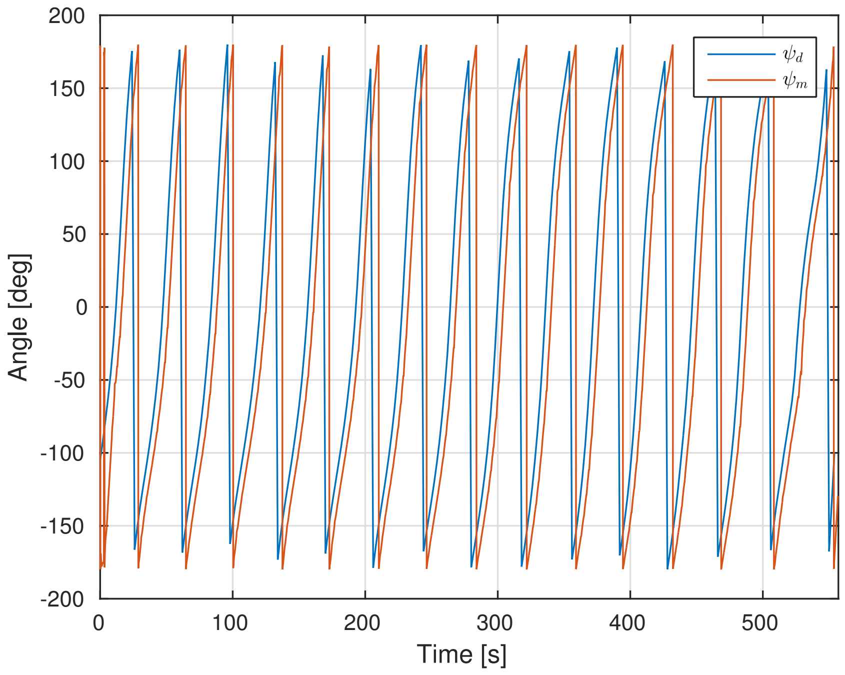
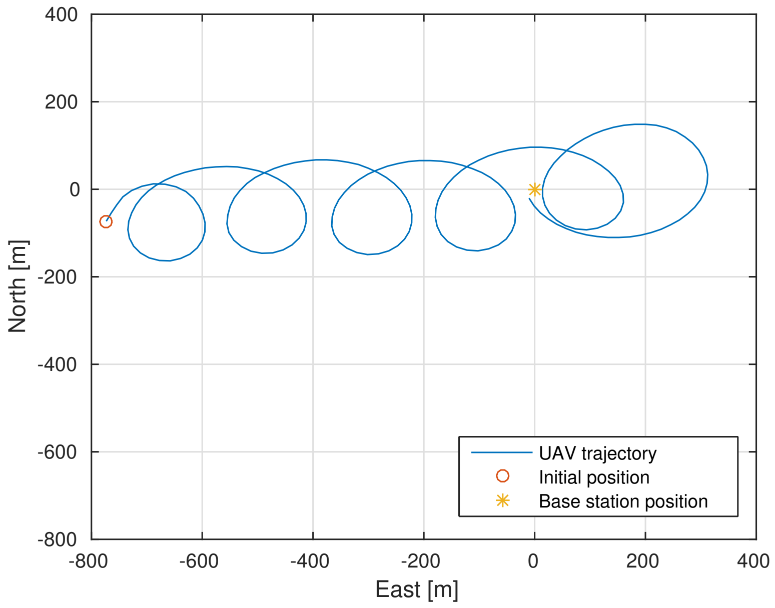


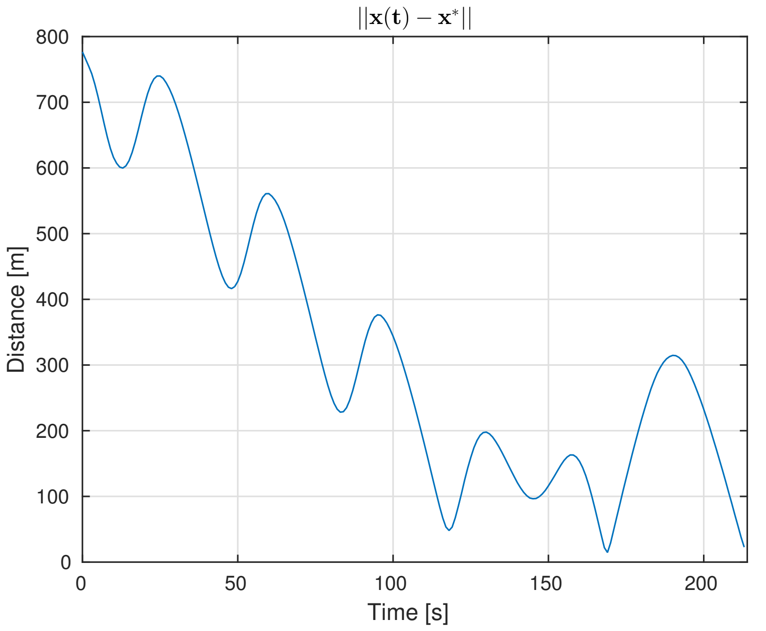
Figure 7.7: Desired heading angle vs. estimated heading angle

Configuration 2

In this configuration, f_{RSSI} was set to 0.5 Hz, λ was set to 0.90 and the initial position of the vehicle was set to 800 meters west and 100 meters south of the base station. Figure 7.8a shows the flight trajectory of the vehicle and Figure 7.8b shows the euclidean distance between the vehicle and the base station as a function of time. Further, Figure 7.9a shows the measured and estimated RSSI and Figure 7.9 shows the estimated gradient of the unknown cost function, both as a function of time. The measured RSSI is denoted y_k and the estimated RSSI is denoted $\hat{y}_k$. Figure 7.10 shows the estimated and desired heading as a function of time. The desired heading is denoted ψ_d and the measured heading is denoted ψ_m.

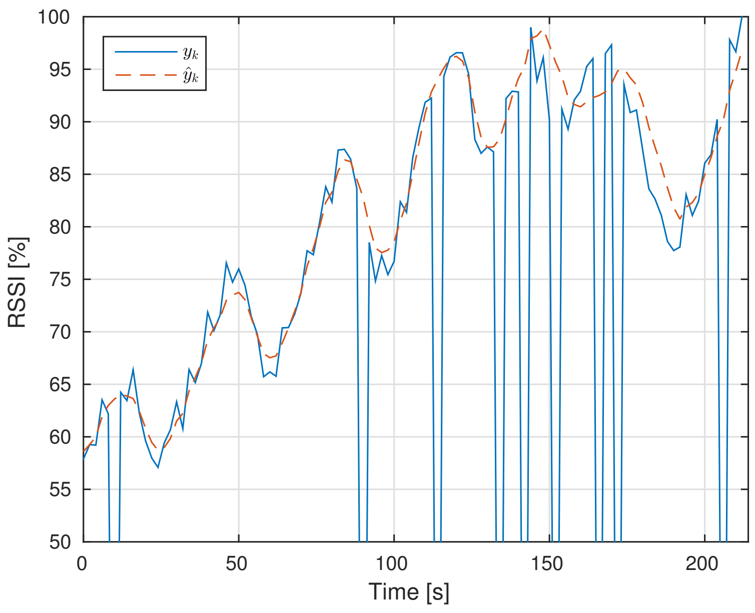


(a) UAV flight trajectory

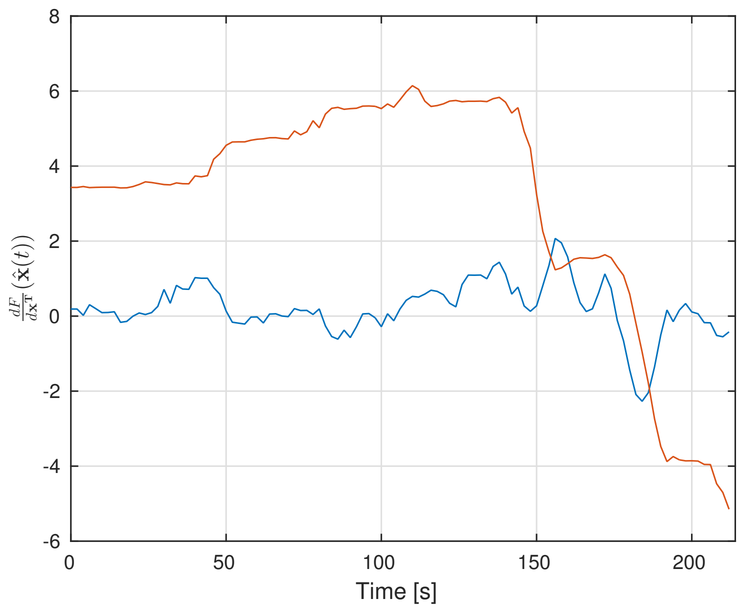


(b) Convergence of solution

Figure 7.8: Configuration 2 flight simulation



(a) Measured vs. estimated RSSI



(b) Estimated cost function gradient

Figure 7.9: RSSI and observer plots

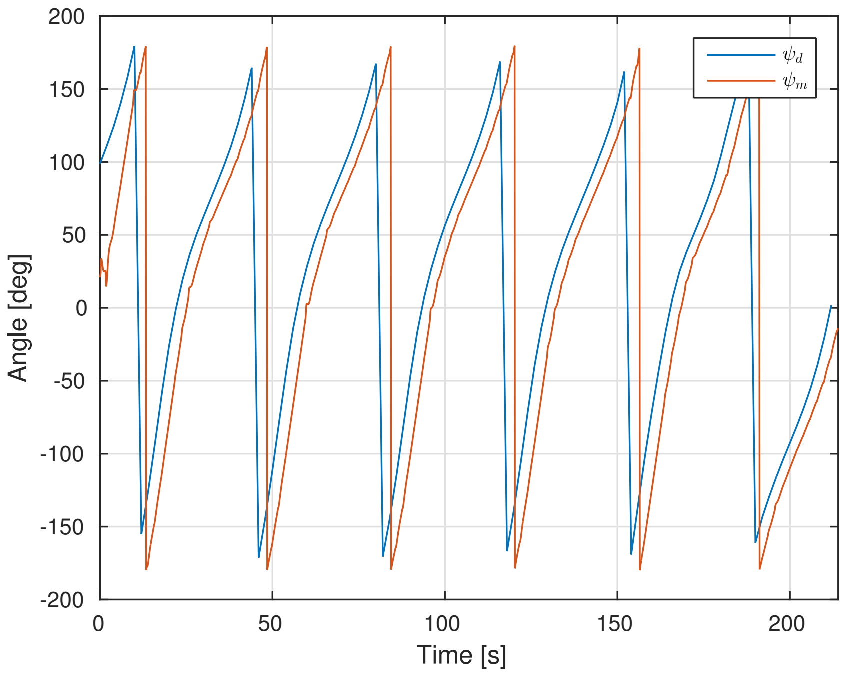
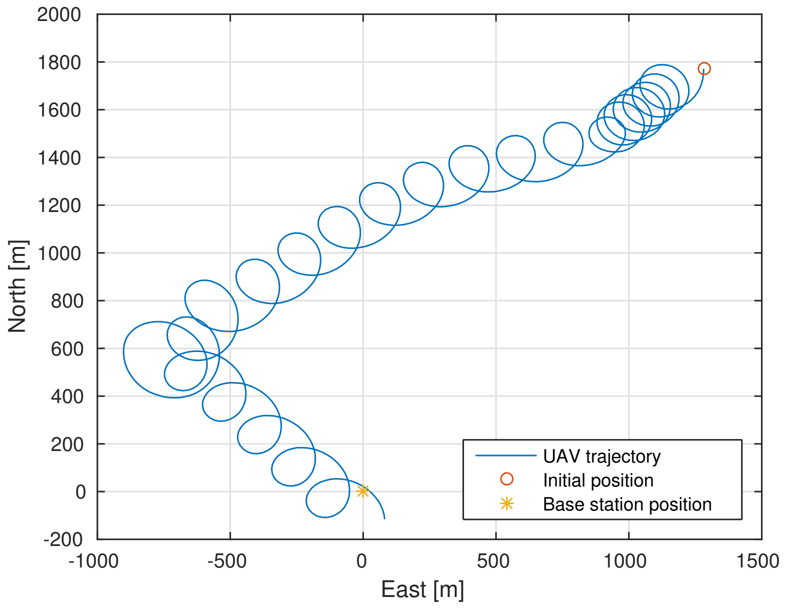
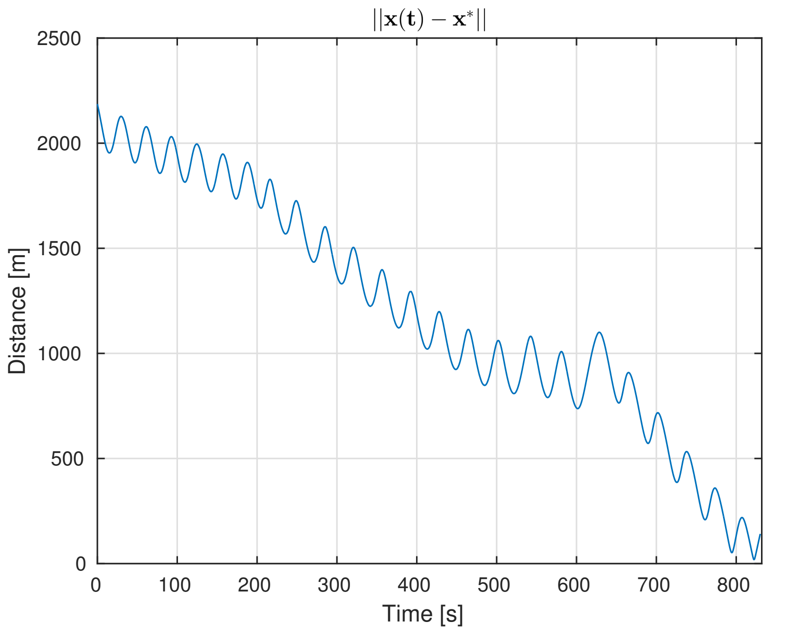


Figure 7.10: Desired heading angle vs. estimated heading angle

Configuration 3
In this configuration, f_{RSSI} was set to 5.0 Hz, λ was initialized to 0.90 and changed to 0.99 after approx. 200 seconds, and the initial position of the vehicle was set to 800 meters west and 100 meters south of the base station. Figure 7.11a shows the flight trajectory of the vehicle and Figure 7.11b shows the euclidean distance between the vehicle and the base station as a function of time. Further, Figure 7.12 shows the measured RSSI together with the estimated RSSI from the observer. The measured RSSI is denoted y_k and the estimated RSSI is denoted $\hat{y}_k$. Figure 7.13 shows the estimated gradient of the unknown cost function, and Figure 7.14 shows the estimated and desired heading, both as a function of time. The desired heading is denoted ψ_d and the measured heading is denoted ψ_m.

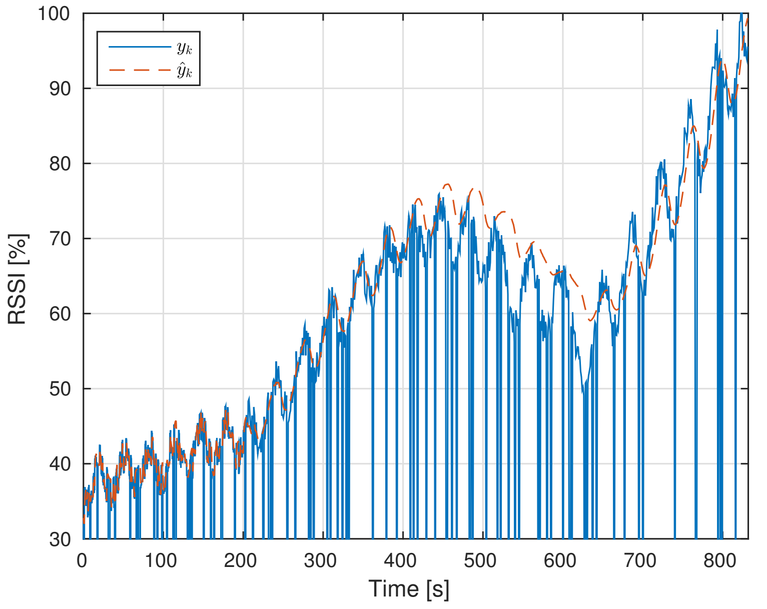


(a) UAV flight trajectory

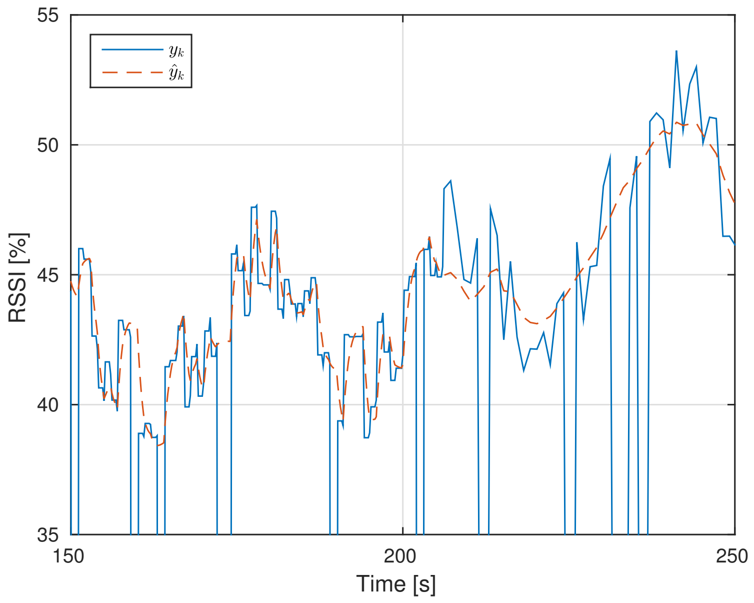


(b) Convergence of solution

Figure 7.11: Configuration 3 flight simulation



(a) Measured vs. estimated RSSI



(b) Measured vs. estimated RSSI, zoomed in

Figure 7.12: Measured vs. estimated RSSI plots

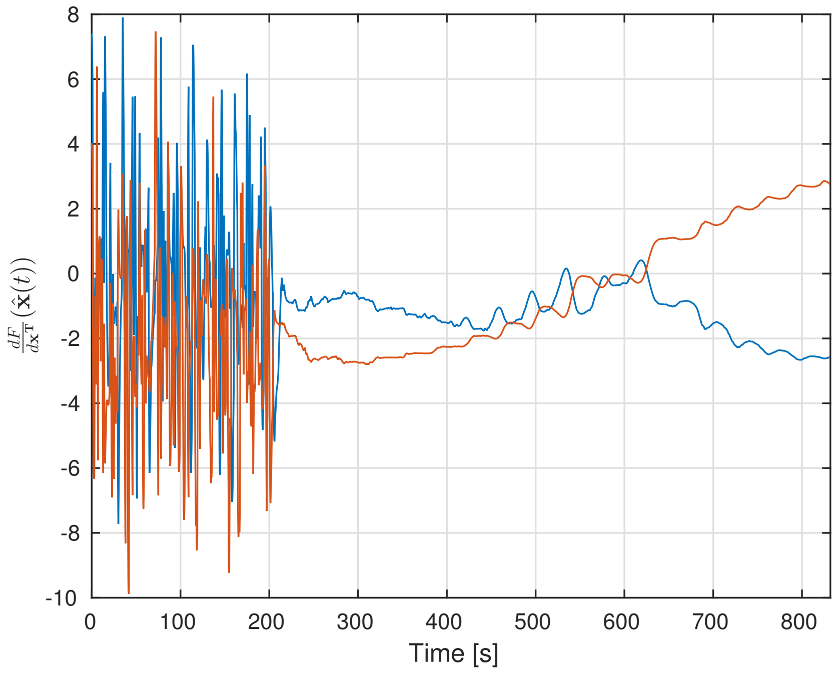


Figure 7.13: Estimated cost function gradient

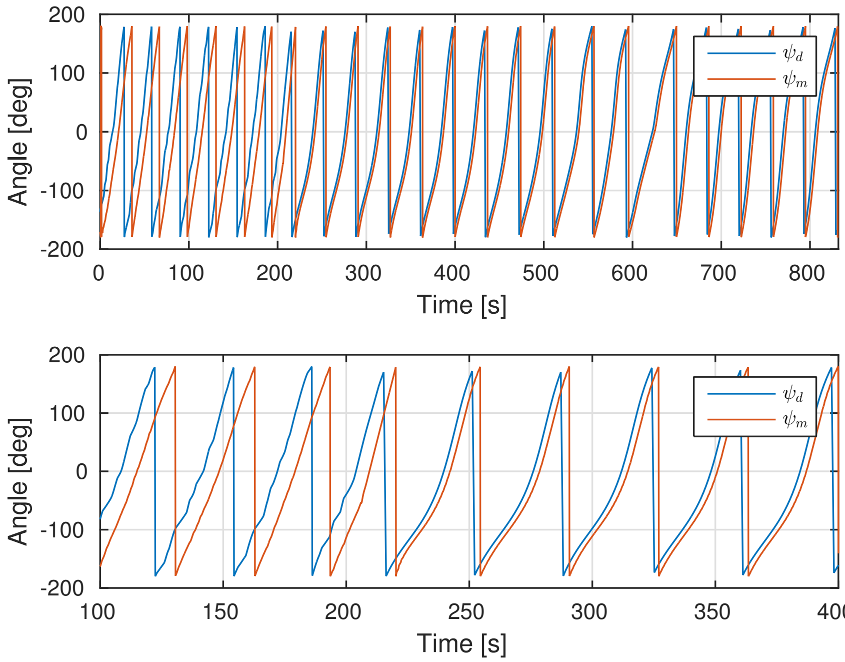


Figure 7.14: Desired heading angle vs. estimated heading angle

Chapter 8

Experimental results

This chapter presents the experimental results obtained. The chapter contains results from both the concept test performed at campus, along with the results from the flight test performed at the airfield at Agdenes.

8.1 Concept test results

The plots show the RSSI together with the estimated RSSI from the observer. In addition, a plots of the position of the UAV are shown. The time interval between every received RSSI measurement and the numerical values of the tuning parameters used is given in Table 8.1. It should be noted that the values of the parameters η, ω and κ are not important, since the UAV was carried around manually. They are only included for completeness.

Parameter	**Value**
f_{RSSI}	0.50 Hz
λ	0.90
η	0.50
ω	0.20
κ	100

Table 8.1: Parameters, experiment

Experiment 1
In this experiment, the UAV was carried while walking in a straight line, first away from and then back to the base station. Figure 8.2 shows the position of the UAV in a East-North plot, along with the euclidean distance from the UAV to the base station. Figure 8.2a shows the RSSI together with the estimated RSSI from the observer.

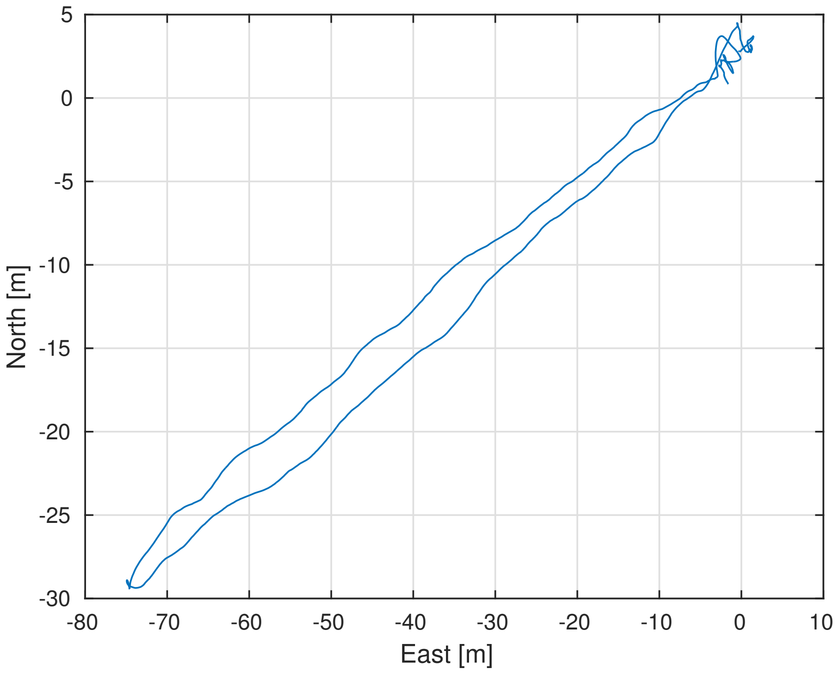
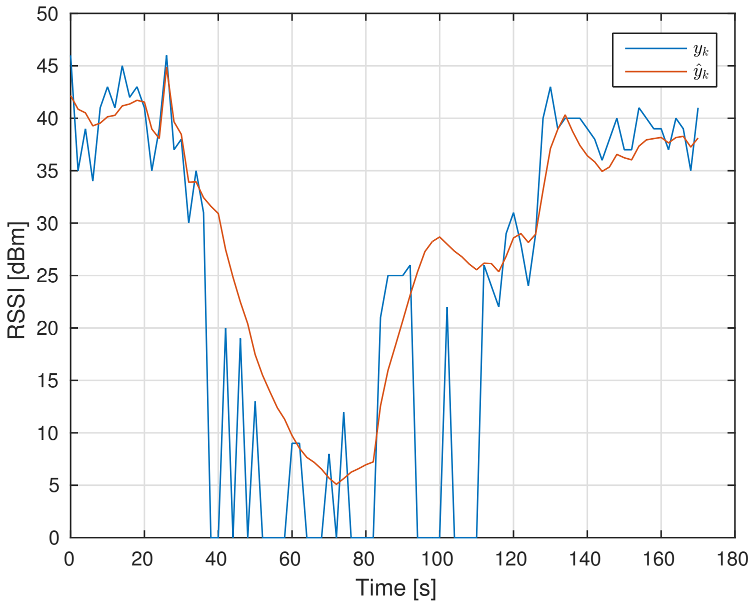
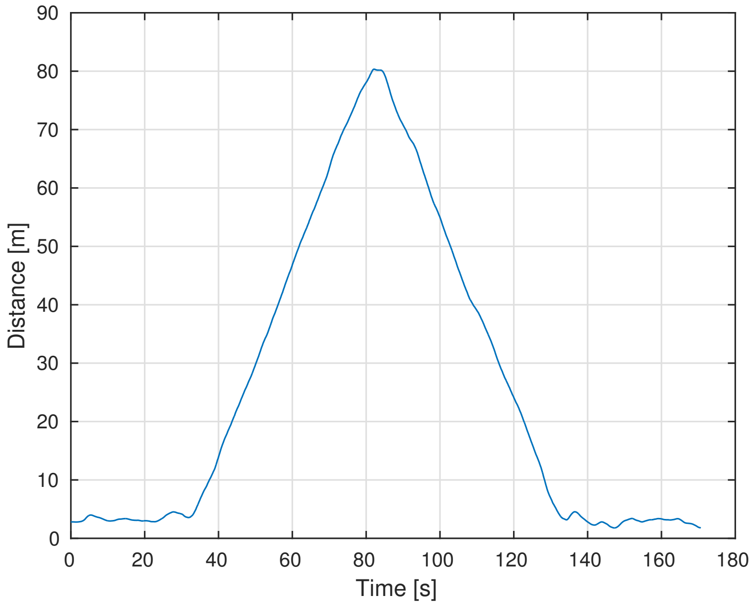


Figure 8.1: Position in the East-North plane



(a) Measured RSSI vs. Estimated RSSI



(b) Distance between base and UAV

Figure 8.2: Experiment 1 position and distance plots

Experiment 2

In this experiment, the UAV was carried while walking in a zigzag pattern away from the base station. Figure 8.4 shows the position of the UAV in a East-North plot, along with the euclidean distance from the UAV to the base station. Figure 8.4a shows the RSSI together with the estimated RSSI from the observer.

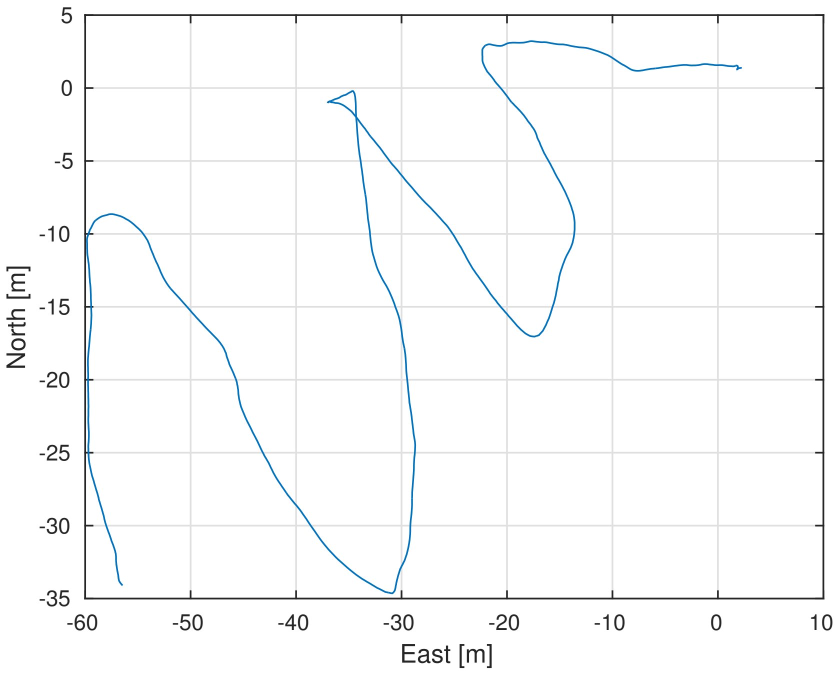
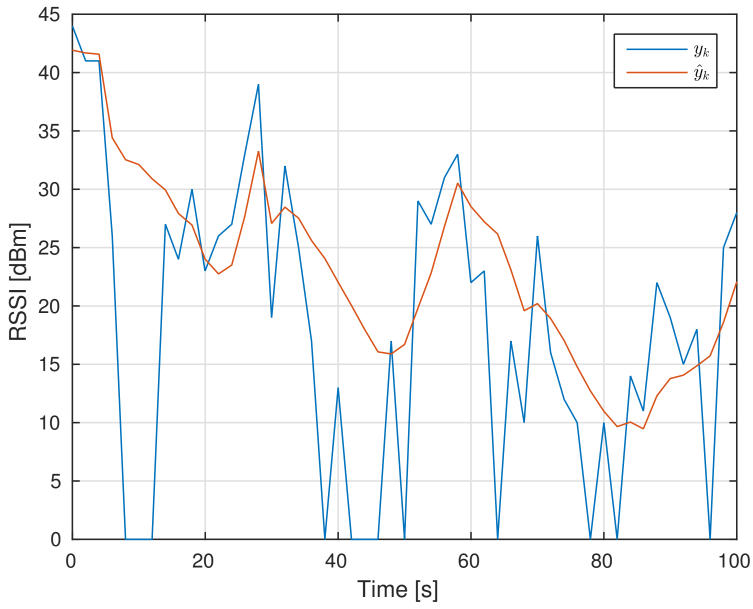
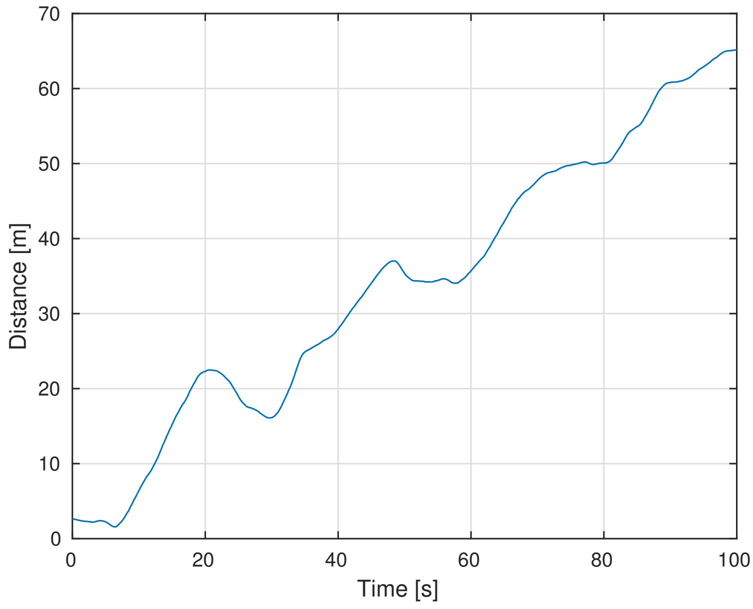


Figure 8.3: Position in the East-North plane



(a) Measured RSSI vs. Estimated RSSI



(b) Distance between base and UAV

Figure 8.4: Experiment 2 position and distance plots

Experiment 3

In this experiment, the UAV was carried while walking in a circling pattern towards the base station. Figure 8.6 shows the position of the UAV in a East-North plot, along with the euclidean distance from the UAV to the base station. Figure 8.6a shows the RSSI together with the estimated RSSI from the observer.

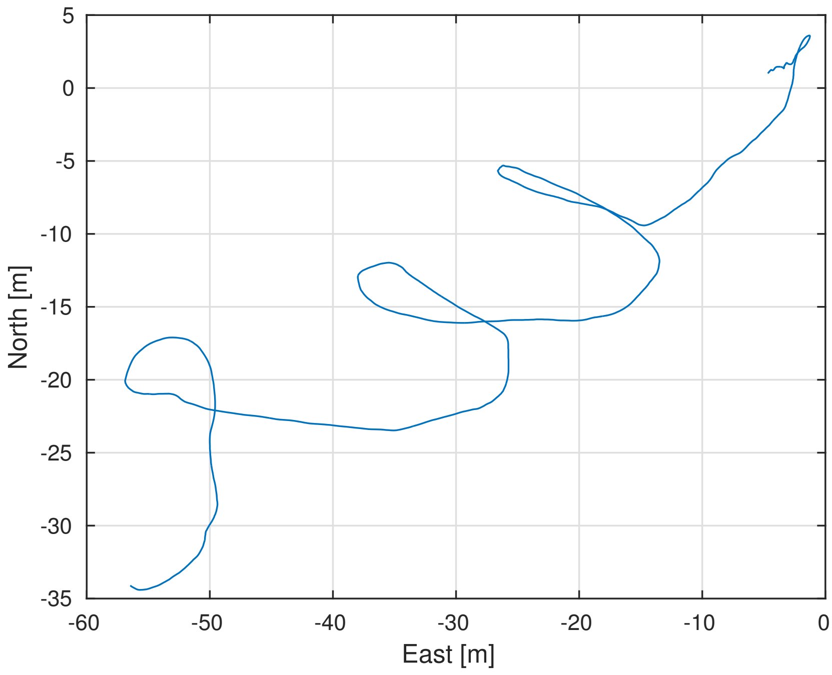
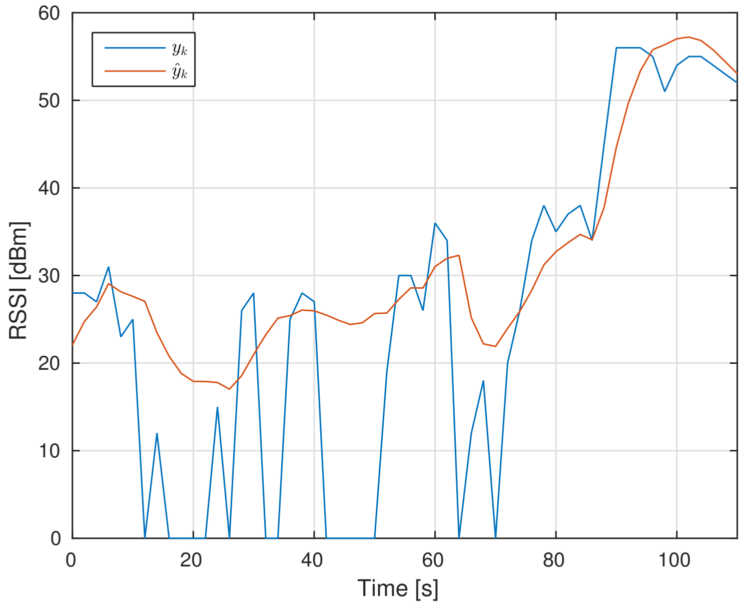
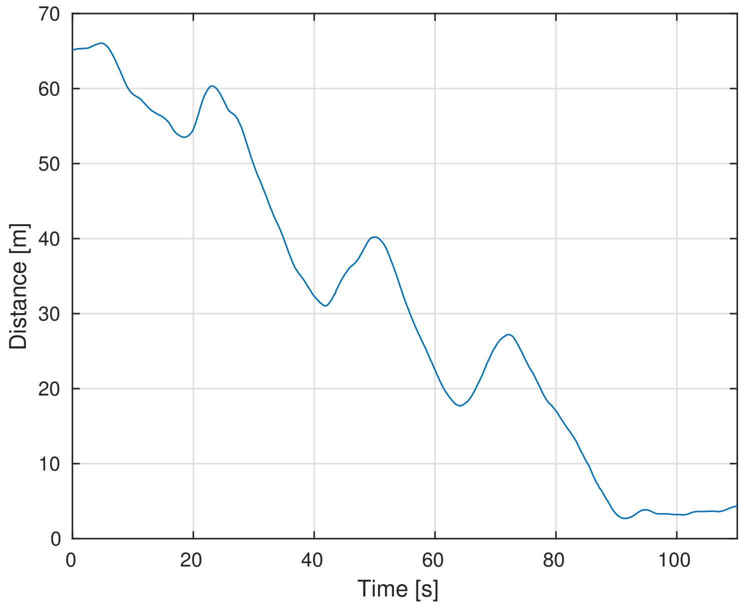


Figure 8.5: Position in the East-North plane



(a) Measured RSSI vs. Estimated RSSI

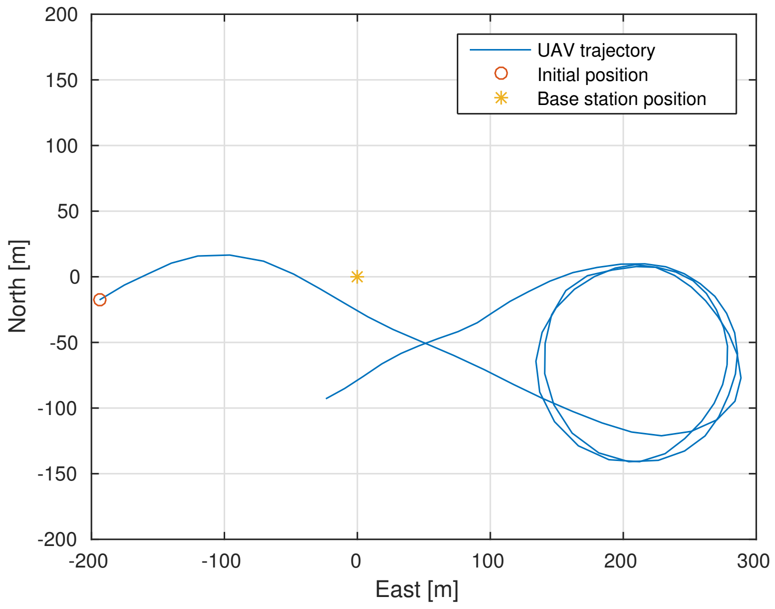


(b) Distance between base and UAV

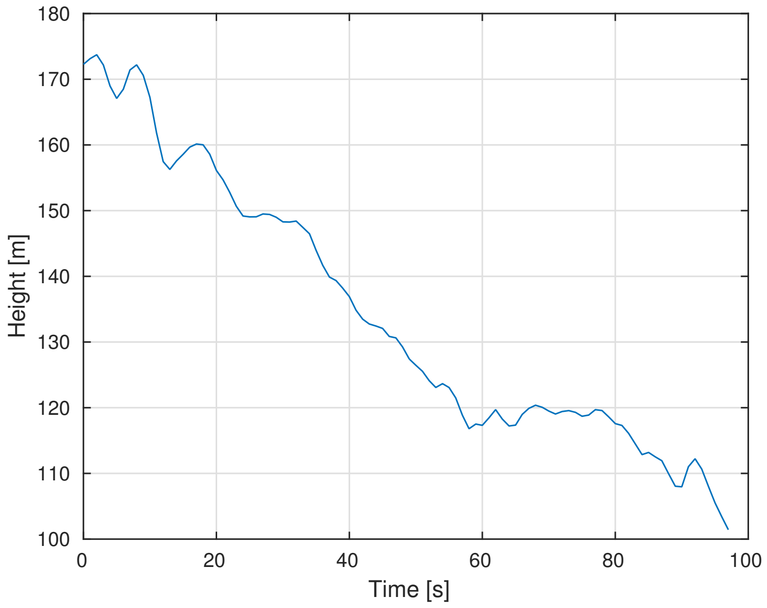
Figure 8.6: Experiment 3 position and distance plots

8.2 Agdenes test flight results

Figure 8.7 shows the position of the UAV in a East-North plot, along with the height above ground of the UAV. Figure 8.8a shows the RSSI together with the estimated RSSI from the observer. Figure 8.8b shows the euclidean distance from the UAV to the base station.

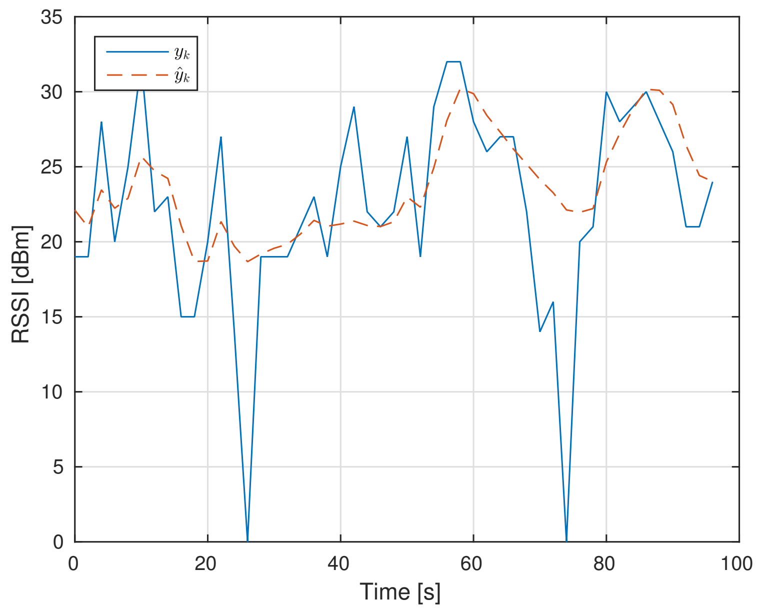


(a) Position in the East-North plane

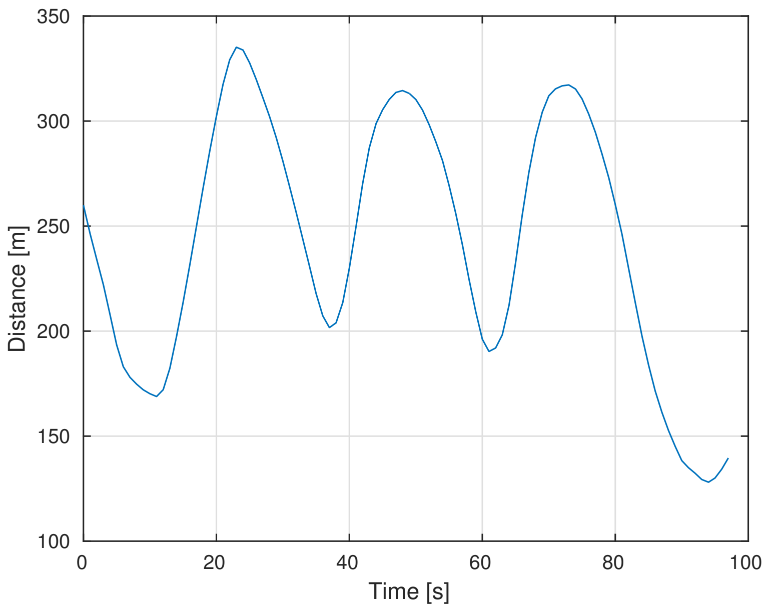


(b) Height above ground

Figure 8.7: Agdenes test flight position and distance plots



(a) Measured RSSI vs. Estimated RSSI



(b) Distance between base and UAV

Figure 8.8: Agdenes test flight position and distance plots

Part IV

Discussion and Conclusion

Chapter 9

Discussion

This chapter discusses and compares some of the results obtained. Finally, recommendations for future work are given.

9.1 Discussion of simulations

Unicycle Simulations
In the unicycle simulations, it is seen that the extremum seeking controller performs conceptually the way it should. It can be seen in Figure 7.1 that the vehicle flight trajectory is a circling motion that converges to the base station, i.e. where the RSSI is highest. Further, it can be seen from Figure 7.2a that the observer is capable of filtering out most of the noise in the RSSI measurements, also including the periods where the communication is lost (RSSI = 0). In Figure 7.2, it can be seen that the estimated gradient is negative in both east and north directions when the vehicle is located south-west of the base station. This makes sense, since east and north are chosen to be positive directions.

Compared to configuration 1, it is seen in Figure 7.3 that a larger κ influences the performance of the controller in the way it is suppose to, i.e. faster convergence. In the same manner, a smaller ω results in larger circles. In Figure 7.1a, the diameter of the circles can be seen to be around 180 meters and in Figure 7.4 the diameter of the circles can be seen to be around 360 meters. This makes sense, since ω is twice as big in configuration 3.

SIL Simulations
When comparing the SIL simulations with the unicycle simulations, it is observed that both system behave similarly. This suggests that the desired heading computed by the extremum seeking controller is well within the limits of what a UAV autopilot is able to follow, as can be seen in Figures 7.7, 7.10 and 7.14. Again, the UAV flight trajectory converges to the base station (Figure 7.5), and the observer does what is should (Figure 7.6). Configuration 2 was included to give an idea of what a VLOS flight test could look like. As could be expected, the UAV converges directly to the base station, without any unwanted transient behaviour. A small overshoot is, however, observed in the vicinity of

the base station. This is not considered a big problem, as a pilot can take control of the UAV at any time.

One of the uncertainties when it comes to using the extremum seeking controller in practice, is the time interval between every RSSI measurement. Because of this, a SIL test where the RSSI frequency was relatively high was conducted. It is seen in Figures 7.11 and 7.13 that when $\lambda = 0.90$ (first 200 seconds), the convergence is slow and the estimated cost function gradient is almost completely contaminated by noise. It can also be seen that there is a relatively large delay between the desired and measured heading (Figure 7.14). When λ is changed to 0.99 (Time $>$ 200 s), the behaviour became more similar to the behaviour seen in configurations 1 and 2. This change in behaviour is likely due to the fact that the RSSI is contaminated by noise, and a larger λ is needed to "filter" out this noise. If Figure 7.12a is studied, it can be seen that for Time $<$ 200 s, the estimated RSSI keeps most of the noise from the measured RSSI, while for Time $>$ 200 s, the estimated RSSI is much more smooth. There is, however, a trade-off here. When λ is larger, the observer gain $\boldsymbol{L}_k$ becomes smaller, as can be seen from Equations (2.2) and (2.5). This means that new RSSI measurement y_k are weighted less, i.e. the a priori estimate $\hat{\boldsymbol{z}}_{k|k-1}$ is weighted more. One consequence of this is that the observer becomes less capable of detecting rapid changes or trend changes in the RSSI. An example of this can be seen in Figure 7.12a, for Time $>$ 400 s. Here, the RSSI stops increasing, and actually starts decreasing. Although the observer did a good job filtering out the noise, it partly failed to follow the decreasing trend in the RSSI. This resulted in that the UAV flew past the base station, and the total flight distance the UAV flew became longer.

One major simplification made in both the unicycle and the SIL simulations, is that the RSSI simulator described in Subsection 4.1.1 was used. In this simulator, the only factor influencing the RSSI other than noise, is the distance between the UAV and the base station. In reality, there are many other factors that come into play. Different terrain can reflect or absorb parts of the communication signal in different ways, and/or result in multipath. This could lead to e.g. local maximums where the UAV could get stuck. One way to reduce the influence of the terrain could be to fly at a higher altitude, but this may not always be possible.

9.2 Discussion of field experiments

Concept Test, Gløshaugen

The experiment was conducted in order to study the characteristics of the RSSI and the quality of the observer estimate. In the experiment, the antenna used at the base station was significantly smaller than the antennas used in real UAV flights. Thus, the experiment must be considered a scaled down setup compared to a real UAV flight setup.

Figures 8.2a, 8.4a and 8.6a suggests that the RSSI is decreasing exponentially when moving away from the base station, just as assumed. Another observation made from Figures 8.2a, 8.4a and 8.6a is that a significant number of the RSSI measurements were 0, especially when the distance between the UAV and the base station increased. This is most likely due to the fact that the connection between the UAV was lost temporary. Most likely, this is due to the fact that an antenna with a short range was used in the experiment. However, a temporary loss of connection can of course occur in larger systems as well. In the cases where the RSSI were zero, it is seen that the observer detects this, and disregards this measurements, so that they do not influence the estimate. This was expected, as the observer also did this in the SIL tests.

Test Flight, Agdenes

The flight test was conducted in order to study the characteristics of the RSSI and the observer performance under realistic conditions. When studying Figure 8.8a and Figure 8.8b, it can be seen that the measured RSSI increased where the distance to the base station decreased and decreased where the distance to the base station increased. This can be seen e.g. after approx. 10 seconds, where the distance is small and the RSSI is high or after approx. 70 seconds, where the distance is large and the RSSI is low. In some time periods however, the distance/RSSI relationship was not as clear. After approx. 35 seconds, the UAV has passed the point on the orbit that is closest to the base station, and was again flying away from it. Here, it could be expected that the RSSI would decrease as the distance increased. This, however, was not the case. The reason for this is not easy to decide, it could be due to one of the other factors affecting the RSSI other than distance, or it could be simply measurement error.

It can also be seen that the connection between the base station and the UAV was lost at two occasions, after approx. 25 seconds and after approx. 75 seconds. This could be expected, based on the results from the first experiment. However, the connection was lost significantly less frequent in the test flight. This is most likely due to the larger and higher quality antenna used.

It is seen in Figure 8.8a that the estimated RSSI captured most of the essence in the measured RSSI. In order to extract more of the information in the measured RSSI, the forgetting factor λ could have been slightly smaller, but this could also make the estimated RSSI less smooth and possibly make the estimated cost function gradient more noisy, as in the first 200 seconds of Figure 7.13, which again could lead to slow convergence.

Comparison of SIL tests and experiments

Here, the RSSI measurements and observer results from the VLOS SIL test (configuration 2) and the flight test performed at Agdenes are compared. One obvious difference between the real and simulated RSSI is the fact that the unit of the real RSSI is dBm, while the unit of the simulated RSSI is percent, where 100 percent is the RSSI at the base station. The reason for making this choice of unit for the RSSI simulator was that the unit of the *IMC::RSSI* message was percent. The different units are however not a big issue, since the extremum seeking controller does not aim for a specific RSSI value, it just aims for where the RSSI is the highest.

It can be seen from Figure 8.8a that the RSSI measurements are less smooth than the simulated RSSI, and that the RSSI value varies a lot from one measurement to the next. To make the RSSI simulator more realistic, the magnitude of the random noise added to the signal could have been larger, thus making it harder for the observer. Even though there are some dissimilarities between the simulations and the flight test, the essence is still the same. Based on this, it is considered likely that the extremum seeking controller would be able to steer the UAV to the vicinity of the base station based on an RSSI that has characteristics similar to the one plotted in Figure 8.8a.

9.3 Future work

The next step should be to do a proper test flight in a VLOS area, where a pilot is standing by to take control of the UAV if needed.

A problem that should be solved, is to determine how much influence the attitude of the UAV has on the RSSI. If this influence is significant, it should be compensated for, either by changing the physical antenna placement on the UAV or computationally in the observer.

Another problem that should be solved is to investigate the performance of the on-board autopilot in the absence of GNSS. If the performance issues are significant, improvements on the local measurements on the UAV must be made.

An interesting question is how the extremum seeking controller should be combined with some type of collision avoidance system. In the case where there is e.g. a mountain between the base station and the UAV, it could be anticipated that the controller would compute a heading that takes the UAV around, and not through, the mountain, since this is where the RSSI would be highest. A more complicated situation is where the UAV could hit e.g. a mountainside on its way around its circling motion. In cases like this, a collision avoidance system is needed.

What must also be taken into consideration, is the total distance flown by the UAV. If one for instance considers Configuration 1 in the SIL test. Here, the shortest distance between the initial position of the UAV and the base station is approximately 2 km. The total distance flown by the UAV however, is closer to 10 km, i.e. 5 times the straight line distance.[1] Clearly, this can be a problem when flying over larger distances, as batteries have limited capacity. Flight patterns that lead to a shorter total distance flown could be considered. It is, however, very important that the excitation/perturbation pattern provides a large enough data set, so that the cost function gradient can be estimated accurately. These are all topics that will have to be assessed in another thesis.

[1] In this example, the total distance flown is approximated by the straight line distance plus the circumference of the number of circles flown, i.e. $2000\text{ m} + 14 \cdot \pi \cdot 180\text{ m} \approx 10000\text{ m}$.

Chapter 10

Conclusion

This master thesis has presented and explained an extremum seeking controller implemented in the DUNE framework. The purpose of the controller is to steer a UAV back to its base station in the absence of GNSS. The input to the controller is the RSSI of the communication signal and the current estimated heading of the UAV. Based on this, the controller calculates the optimal heading that steers the UAV back to its base station. The controller has been SIL tested with the JSBSim Open Source Flight Dynamics Model. Since the JSBSim FDM does not include a simulated RSSI, an RSSI simulator was implemented in DUNE.

Under the assumption made, the SIL tests have shown that the controller can indeed steer the UAV back to the vicinity of the base station. The SIL tests have shown that the performance of the controller is highly dependent of the numerical values of the tuning parameters and the interval between each RSSI measurement. In particular, the forgetting factor λ, which heavily influences the magnitude of the observer gain, and thus the weighting between the a priori state estimate and the new RSSI measurement, must be tuned in relation to the interval between each RSSI measurement.

Experimental work conducted verifies that the extremum seeking controller's embedded observer is able to estimate the RSSI accurately under real flight conditions. The experimental work conducted also verifies the close relation between RSSI and distance between transmitter and receiver.

The author hopes that the work done in this thesis can serve as a basis for further work on the topic and inspire others working with vehicle navigation and control in the absence of GNSS.

Part V

Appendices

Appendix A

Running the simulations

A.1 Minimal configuration file

The following code presents a minimal configuration (.ini) file that defines which tasks should run, and how they should be configured, including how to run the RSSIExtremum task, located in the Control/UAV folder.

```
[Require uav/arduplane.ini]

[General]
Vehicle                 = ntnu-x8-004

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

[Control.UAV.RSSIExtremum]
Enabled                 = Always
Entity Label            = RSSI Extremum Optimizer
Kappa                   = 100
Eta                     = 0.5
Omega                   = 0.2
```

A.2 Running Ardupilot with JSBSim

The following command runs Ardupilot with the JSBSim FDM.

```
$home/ardupilot/ArduPlane $ sim_vehicle.sh -f jsbsim:X8
--console --map --aircraft test
```

A.3 Running DUNE

The following command runs DUNE with the configuration file ntnu-x8-004.ini with the ArduPlane software-in-the-loop (AP-SIL) profile. DH refers to the DUNE home directory.

```
DH/build $ ./dune -c ntnu-x8-004 -p AP-SIL
```

Bibliography

3DRobotics, 2016. 3dr pixhawk. `https://store.3dr.com/products/3dr-pixhawk`, [Online; accessed 10-May-2016].

ArduPilot, 2016. Ardupilot. `http://ardupilot.org/ardupilot/index.html`, [Online; accessed 4-June-2016].

Beagleboard, 2016. Beaglebone black. `https://beagleboard.org/black`, [Online; accessed 10-May-2016].

Beard, R. W., McLain, T. W., 2012. Small unmanned aircraft: Theory and practice. Princeton university press.

Bowditch, N., Logan, G. W., 1914. American practical navigator: an epitome of navigation and nautical astronomy. No. 9. Govt. Print. Off.

Cochran, J., Krsic, M., 2007. Source seeking with a nonholonomic unicycle without position measurements and with tuning of angular velocity part i: Stability analysis. In: Decision and Control, 2007 46th IEEE Conference on. IEEE, Proceedings of the 46th IEEE Conference on Decision and Control, pp. 6009–6016.

Conte, G., Doherty, D., 2009. Vision-based unmanned aerial vehicle navigation using georeferenced information. EURASIP Journal on Advances in Signal Processing 2009, 10.

data-alliance, 2016. Rocket M5 5GHz: Ubiquiti Long-Range Bridge/BaseStation 2x2 MIMO Air Max. `http://www.data-alliance.net/rocket-m5-5ghz-ubiquiti-long-range-bridge-basestation-2x2-mimo-air-ma`
[Online; accessed 4-June-2016].

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

Haring, M., Johansen, T. A., 2015. Extremum-seeking control for nonlinear plants by least-squares gradient estimation.

Institute of Engineering Cybernetics, N., 2016. Unmanned Aerial Vehicles Laboratory (UAV-Lab). http://itk.ntnu.no/english/lab/unmanned, [Online; accessed 10-May-2016].

Krstic, M., W, H. H., 2000. Stability of extremum seeking feedback for general nonlinear dynamic systems. Automatica 36 (4), 595–601.

LSTS, 2016a. DUNE Unified Navigation Environment. http://www.lsts.pt/toolchain/dune, [Online; accessed 6-April-2016].

LSTS, 2016b. DUNE Unified Navigation Environment Documentation. http://lsts.pt/docs/dune/dune-2.6.1/, [Online; accessed 30-May-2016].

LSTS, 2016c. GLUED GNU/Linux Uniform Environment Distribution. http://www.lsts.pt/toolchain/glued, [Online; accessed 6-April-2016].

LSTS, 2016d. IMC Inter-Module Communication Protocol. http://www.lsts.pt/toolchain/imc, [Online; accessed 6-April-2016].

LSTS, 2016e. Neptus Command and Control Software. http://www.lsts.pt/toolchain/neptus, [Online; accessed 6-April-2016].

LSTS, 2016f. X8 X8 Flying Wing. http://www.lsts.pt/vehicles/x8, [Online; accessed 8-April-2016].

Marshall, D., Liu, P., Abrahamsson, M., Raustein, M., Weatherhead, E., la Cour-Harbo, A., Mulac, B., Johansen, K., Eppi, R., Crowe, W., et al., 2012. Enabling science use of unmanned aircraft systems for arctic environmental monitoring. Tech. rep., Arctic Monitoring and Asessment Programme (AMAP).

Sauter, M., 2010. From GSM to LTE: an introduction to mobile networks and mobile broadband. John Wiley & Sons.

Sourceforge, 2016. An open source, platform-independent, flight dynamics & control software library in c++. http://jsbsim.sourceforge.net/, [Online; accessed 9-May-2016].

Standard, F., 1996. 1037c. telecommunications: Glossary of telecommunication terms. Institute for Telecommunications Sciences 7.

Tan, Y., Moase, W., Manzie, C., Nešić, D., Mareels, I., 2010. Extremum seeking from 1922 to 2010. In: Control Conference (CCC), 2010 29th Chinese. IEEE, pp. 14–26.

Zhang, C., Arnold, D., Ghods, N., Siranosian, A., Krstic, M., 2007. Source seeking with non-holonomic unicycle without position measurement and with tuning of forward velocity. Systems & control letters 56 (3), 245–252.