Control of Differential Driven Rover by Means of Sensor Fusion

Master Thesis

Andreas Holst Thøgersen & Jesper Skovager Thomsen
Mechatronic Control Engineering
MCE4 - 1025

Aalborg University
Department of Energy Technology

AALBORG UNIVERSITY
STUDENT REPORT

10th Semester
Mechatronic Control Engineering
Department of Energy Technology
Pontoppidanstræde 111
9220 Aalborg
https://www.et.aau.dk/

Title:

Control of Differential Driven Rover by Means of Sensor Fusion

Project Period:

From 01-02-2022 to 15-06-2022

Project Group:

MCE4 - 1025

Authers:

Andreas Holst Thøgersen
Jesper Skovager Thomsen

Supervisors:

Petar Durdevic
Simon Lennart Sahlin

Number of Pages: 78
Appendices: 3

Abstract:

In this report, a differential-driven rover for use in agriculture is analyzed with the end goal of determining and controlling its position.
For determining the rover's position as well as its other states, an extended Kalman filter has been designed combining multiple sensors to achieve higher precision than what would have been achievable by using only one sensor. The sensors used are wheel encoders, an inertial measurement unit, a magnetometer, and a GNSS with RTK capabilities.
The system states are used for designing two different kinds of controllers. The first one is a heading controller, to ensure that the rover keeps the correct heading while driving along at a constant linear velocity. The second is an inertial frame controller, ensuring that the rover reaches predefined positions.
The extended Kalman filter as well as all the sensor input are handled by a microcontroller running FreeRTOS to ensure real-time operation. While the control and interface to the motors are handled by an NVIDIA Jetson Nano running Robot Operating System.

Summary

This master thesis is concerned with the design of a control system for the navigation of a differential-driven rover. The main objective of the thesis is to develop a generic control platform that may be extended in different ways. The control platform is built around an Nvidia Jetson Nano board running Robot Operating System (ROS) as this allows for the incorporation of different premade functionalities developed by the open-source ROS community. The NJN is the main junction between any potential rover expansions, and will for the thesis receive state estimates, generated on a Nucleo development board and use these state estimates for feedback control of the rover, and send the commands to a dual-motor drive unit.

The Nucleo board will read the sensor data coming from the motor encoders, an inertial measurement unit (IMU), a magnetometer as well as a global navigation satellite system board and fuse it by means of an extended Kalman filter (EKF). The Nucleo board is running freeRTOS which is a real-time operating system to ensure consistent timing.

The general accuracy of the EKF cannot be quantified in this report as this requires knowledge of the true system states. Despite this, the performance of the filter in the event of a data fallout of 40 seconds has been simulated and has shown a high similarity to the position estimates recorded by the GNSS.

Two different control structures have been proposed in the report one being an alignment controller and the other an inertial frame controller. The alignment controller will attempt to align the rover heading with a given reference while driving at a constant set linear velocity. This controller is designed as it may be used with camera vision but also serves as part of the control structure for the inertial frame controller. The inertial frame controller is designed to handle the navigation of the rover in the inertial frame and will therefore take in the desired location (x_{ref}, y_{ref}) and navigate the rover to within some distance ϵ from this point. Simulation results of the complete system show that the inertial frame controller has completed the predefined trajectory with a root mean square error of 7 cm and a max deviation from the trajectory of 18 cm. These values are however dependent on the velocity at which the trajectory is completed as higher velocities generally require higher system bandwidth.

Contents

Summary v

Nomenclature ix

1 Problem Formulation 1
- 1.1 Introduction . . . 1
- 1.2 Problem Description . . . 2
- 1.3 Problem Statement . . . 2

2 System Description 3
- 2.1 System Structure . . . 4
- 2.2 Wiring Diagram . . . 5
- 2.3 ROS Implementation on NJN . . . 5

3 System Model 9
- 3.1 Dynamic Model . . . 10
- 3.2 Transformation to Inertial Frame . . . 11

4 Parameter Identification 13
- 4.1 System Parameters . . . 13
- 4.2 Time Delay . . . 15
- 4.3 Discussion . . . 16

5 Sensor Calibration and Interpretation 17
- 5.1 Encoders . . . 17
- 5.2 GNSS . . . 17
- 5.3 Accelerometer . . . 18
  - 5.3.1 Accelerometer Model . . . 21
  - 5.3.2 Accelerometer Testing . . . 21
- 5.4 Gyroscope . . . 23
- 5.5 Magnetometer . . . 24

6 State Observer Design 29
- 6.1 Inertial Navigation Filter . . . 29
- 6.2 Extension of Kalman Filter . . . 32
- 6.3 Scalar Implementation . . . 35
- 6.4 Computational Load of Nucleo Board . . . 36
- 6.5 Validation of EKF . . . 36

7 State Estimation from Measured Sensor Data 41
- 7.1 Test Scenario . . . 41

7.2 Performance Evaluation 42
7.3 Performance Evaluation During Data Fallout 44
7.4 Discussion 44

8 Control **47**
8.1 Alignment Controller 47
8.1.1 Controller Evaluation 50
8.2 Inertial Frame Controller 51
8.2.1 Controller Evaluation 54

9 Discussion **59**

10 Conclusion **61**

11 Future Work **63**

Bibliography **65**

Appendices:

A Component Description **69**

B Determination of Measurement Uncertainty **71**

C State Estimation Plots **75**

Nomenclature

Symbols:

$\Delta\psi$	Angle between rover current heading and reference point	rad
Δd	Distance from current position to d	m
Δl	Distance from current to reference position	m
Δx	Distance from current to reference position in the x direction	m
$\dot{x}$	Inertial frame velocity	m/s
$\dot{y}$	Inertial frame velocity	m/s
$\dot{\omega}$	Rotational acceleration	rad/s^2
$\dot{\phi}_l$	Rotational velocity of left motor	rad/s
$\dot{\phi}_r$	Rotational velocity of right motor	rad/s
$\dot{\phi}_{l,ref}$	Rotational velocity reference for the left motor	rad/s
$\dot{\phi}_{r,ref}$	Rotational velocity reference for the right motor	rad/s
ϵ	Radius of around reference location	m
$\hat{x}$	Estimated state vector	–
μ	Average value for measurement uncertainty	–
ω	Rotational velocity of the rover	rad/s
ω_{meas}	Gyroscope measurement	rad/s
ω_{ref}	Rotational velocity reference	rad/s
ω_{true}	Gyroscope true value	rad/s
ϕ	Wheel position	rad
ψ	Angle between rover position and reference point	rad
$\mathcal{T}_l$	Left motor torque	Nm
$\mathcal{T}_r$	Right motor torque	Nm
θ	Heading of the rover	rad
θ_{IMU}	Angle difference between body and IMU frame	rad
θ_{meas}	Magnetometer measurement	rad
θ_{ref}	Heading reference	rad
θ_{true}	Magnetometer true value	rad
$\tilde{y}$	Error between estimated and measured states	–
$a_{b,x}$	Acceleration in x direction of body frame	m/s^2
$a_{b,y}$	Acceleration in y direction of body frame	m/s^2
A_{bias}	Accelerometer bias	m/s^2
a_{cen}	Centripetal acceleration	m/s^2
$a_{IMU,x}$	Acceleration in x direction of IMU frame	m/s^2
$a_{IMU,y}$	Acceleration in y direction of IMU frame	m/s^2
a_{meas}	Accelerometer measurement	m/s^2
a_{tan}	Tangential acceleration	m/s^2
a_{true}	Accelerometer true value	m/s^2
b	Friction constant	Ns/rad
d	Closest point to the reference position along current heading	m

e_θ	Error in heading	rad
e_s	Error in distance	rad
f	State transition function	m/s
G_{bias}	Gyroscope bias	rad/s
J	Inertia of the rover	kgm^2
K_i	Integral gain for motor controller	–
K_p	Proportional gain for motor controller	–
K_{Den}	Kalman gain denominator	
K_{Num}	Kalman gain numerator	–
$K_{p,\theta}$	Proportional gain for heading controller	–
$K_{p,s}$	Proportional gain for distance controller	–
L	Width of the rover	m
L_{IMU}	Length from center of rotation to IMU	m
lat	latitude of the rover	°
lat_o	latitude of the rover at the origin of the coordinate system	°
lon	longitude of the rover	°
lon_o	longitude of the rover at the origin of the coordinate system	°
m	Mass of the rover	kg
M_{bias}	Magnetometer bias	rad
n_t	Number of ticks from encoder	–
$q_{\dot{x}}$	Velocity in x process noise variance	$(\text{m/s})^2$
$q_{\dot{y}}$	Velocity in y process noise variance	$(\text{m/s})^2$
q_ω	Rotational velocity process noise variance	$(\text{rad/s})^2$
q_θ	Heading process noise variance	$(\text{rad})^2$
$q_{A_{bias}}$	Accelerometer bias process noise variance	$(\text{m/s}^2)^2$
$q_{G_{bias}}$	Gyroscope bias process noise variance	$(\text{rad/s})^2$
$q_{M_{bias}}$	Magnetometer bias process noise variance	$(\text{rad})^2$
q_V	Velocity process noise variance	$(\text{m/s})^2$
q_x	Distance in x process noise variance	$(\text{m})^2$
q_y	Distance in y process noise variance	$(\text{m})^2$
r	Radius of the wheels	m
r_{acc}	Accelerometer measurement uncertainty variance	$(\text{m/s}^2)^2$
r_{GNSS}	GNSS measurement uncertainty variance	$(\text{m})^2$
r_{gyro}	Gyroscope measurement uncertainty variance	$(\text{rad/s})^2$
r_{mag}	Magnetometer measurement uncertainty variance	$(\text{rad})^2$
S	Body frame position	m
S_{ref}	Body frame position reference	m
T_s	Kalman filter sampling time	s
t_{enc}	Time between encoder measurements	s
V	Linear velocity of the rover	m/s
v_a	Accelerometer zero-mean Gaussian noise	m/s^2
v_g	Gyroscope zero-mean Gaussian noise	rad/s
V_{meas}	Velocity measurement	m/s

v_m	Magnetometer zero-mean Gaussian noise	rad
V_{ref}	Linear velocity reference	m/s
x	Inertial frame position	m
x_b	Body frame x direction	–
x_{IMU}	IMU frame x direction	–
$x_{mag,max/min}$	Maximum and minimum raw values in the x direction	–
$x_{mag,o,max}$	Maximum hard iron compensated values in the x direction	–
$x_{mag,os}$	Magnetometer hard and soft iron compensated data in the x direction	–
$x_{mag,o}$	Magnetometer hard iron compensated data in the x direction	–
x_{mag}	Magnetometer raw data in the x direction	–
x_{ref}	Reference point in the inertial frame x direction	m
y	Inertial frame position	m
y_b	Body frame y direction	–
y_{IMU}	IMU frame y direction	–
$y_{mag,max/min}$	Maximum and minimum raw values in the y direction	–
$y_{mag,o,max}$	Maximum hard iron compensated values in the y direction	–
$y_{mag,os}$	Magnetometer hard and soft iron compensated data in the y direction	–
$y_{mag,o}$	Magnetometer hard iron compensated data in the y direction	–
y_{mag}	Magnetometer raw data in the y direction	–
y_{ref}	Reference point in the inertial frame y direction	m
z	Measurement vector	–

Matrices:

$\hat{\boldsymbol{P}}$	Error covariance matrix	–
$\boldsymbol{A_{ss}}$	System matrix of the rover	–
$\boldsymbol{A}$	Generic system matrix	–
$\boldsymbol{B_{ss}}$	Input matrix of the rover	–
$\boldsymbol{B}$	Generic input matrix	–
$\boldsymbol{C}$	Generic output matrix	–
$\boldsymbol{F}$	State transition matrix	–
$\boldsymbol{G_{\dot{\phi}}}$	Rotational velocity transfer function matrix	–
$\boldsymbol{G_{kin}}$	Kinematic transfer function matrix	–
$\boldsymbol{G}$	Control matrix	–
$\boldsymbol{H}$	Observation matrix	–
$\boldsymbol{K}$	Optimal Kalman gain	–
$\boldsymbol{M}$	Kinematic relation between motor rotational velocity and linear/rotational velocity of the rover	–
$\boldsymbol{Q}$	Error covariance matrix	–
$\boldsymbol{R}$	Error covariance matrix	–

Abbreviations:

BMS	Battery Management System
CV	Camera Vision
DoF	Degrees of Freedom
DSP	Digital Signal Processor
EKF	Extended Kalman Filter
ENU	East, North, UP
GNSS	Global Navigation Satellite System
IMU	Inertial Measurement Unit
INF	Inertial Navigation Filter
KF	Kalman Filter
NJN	Nvidea Jetson Nano
PCB	Printed Circuit Board
RMSE	Root Mean Square Error
ROS	Robot Operating System
RPM	Revolutions Per Minute
RTK	Real Time Kinematics
VO	Visual Odometry

Constants:

g	Gravitational acceleration	$9.82\,\mathrm{m/s^2}$
R_{earth}	Radius of the earth	$6371 \cdot 10^3\,\mathrm{m}$

Problem Formulation 1

1.1 Introduction

Agriculture machinery is conventionally large and heavy. The idea of using smaller machines, in larger quantities, proposes some interesting features which might benefit the field of agriculture. Smaller rovers have the benefit of being lighter, and thereby not compacting the soil in the same way larger machinery might. It may also allows for crop management on the level of individual plants rather than on a field level, as a large quantity of small rovers, would be able to comb through a field more efficiently than if done manually. In this report the rover depicted on Figure 1.1 is considered. This is a differential-driven rover with approximate dimensions of 650 mm long, by 450 mm wide, by 330 mm tall without the robotic arm, can be seen on Figure 1.1. The rover is designed for agriculture, where the terrain is rough and sometimes muddy, tracks are fitted instead of conventional wheels, to prevent slip, between the wheels and the ground. The rover is meant to have a robotic arm attached to it, making it a very versatile machine, not only for agriculture, but for many fields of interest.

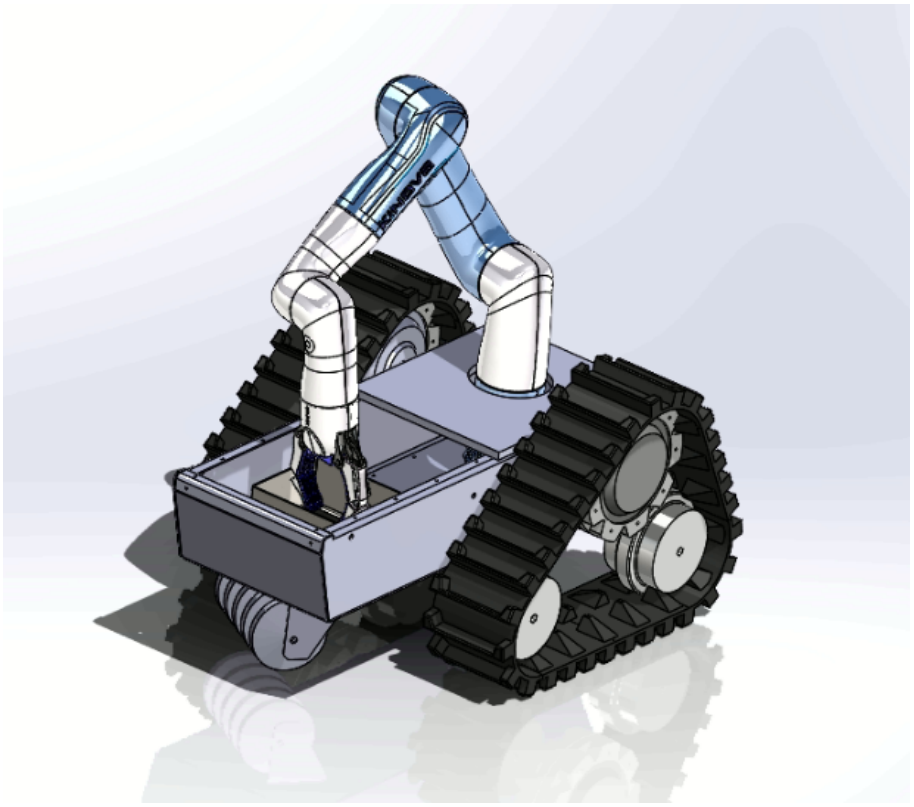


Figure 1.1. Rendering of the rover considered in this project

1.2 Problem Description

The rover will for the project be viewed as a general platform with the main focus being on the navigation and control of the rover. To navigate the rover, the position, velocity, and heading of the rover will have to be estimated. This may either be used directly in a feedback controller or to track the rover. This state estimation will for the report be done by means of an extended Kalman filter where sensor data from motor encoders, an Inertial Measurement Unit (IMU), a magnetometer as well as a Global Navigation Satellite System (GNSS) module will be fused to achieve a better overall state estimation. Using different sensors rather than just relying on e.g. GNSS means that state estimations of the rover may be performed in between the relatively slow update frequency of the GNSS. Another benefit of this strategy is that the system will have some redundancy which means that the states of the rover may be estimated despite events such as a data fallout.

1.3 Problem Statement

How can an extended Kalman filter be designed to estimate system states, by means of sensor fusion, for use in a feedback controller to navigate a differential-driven rover?

System Description 2

The following chapter will be concerned with the components of the rover, and how they interact. A close-up of the rover electronics with annotations can be seen on Fig. 2.1. The electronics of the system may be split up into three different main components being:

- Main Computer
- Motor and motor drive
- Sensor pack

A full description of the different sub-components, as well as their respective capabilities, can be found in Appendix A but a summary of their different system tasks and how they fit into the system will be provided here.

A full picture of the rover can be seen on the left part of Figure 2.1 while the right part is a zoom of the content of the light blue dashed box. In this picture all USB connections for the Nvidia Jetson Nano (NJN) has been removed in order to provide a clear view of the boards. The main computer for the system will be the NJN, seen on the bottom right of the acrylic plate on Figure 2.1, which will be running Robot Operating System (ROS) and will function as the junction between the different components of the system, as well as run the control code. The NJN will then send velocity commands to the motor drive, which will internally handle the individual velocity control loops for the two motors. The NJN is used as it allows for various extensions to the rover capabilities, such as camera vision, control of the robot arm, etc.

The Nucleo board on the top level of the acrylic plate of Figure 2.1 will for the project act as a sensor junction point that will take in data from the IMU, magnetometer, GNSS, and encoders to estimate the rover states by means of an EKF. The estimated rover states will then be transmitted to the NJN over a UART connection. The Nucleo board will be running FreeRTOS which is a real-time operating system to ensure strict timing.

On Figure 2.1 a radio frequency (RF) module can be seen. This has initially been intended for remote control of the rover but is not used in the project as it has been observed to cause problems due to interference of the radio signal from the real-time kinematics signal from the GNSS.

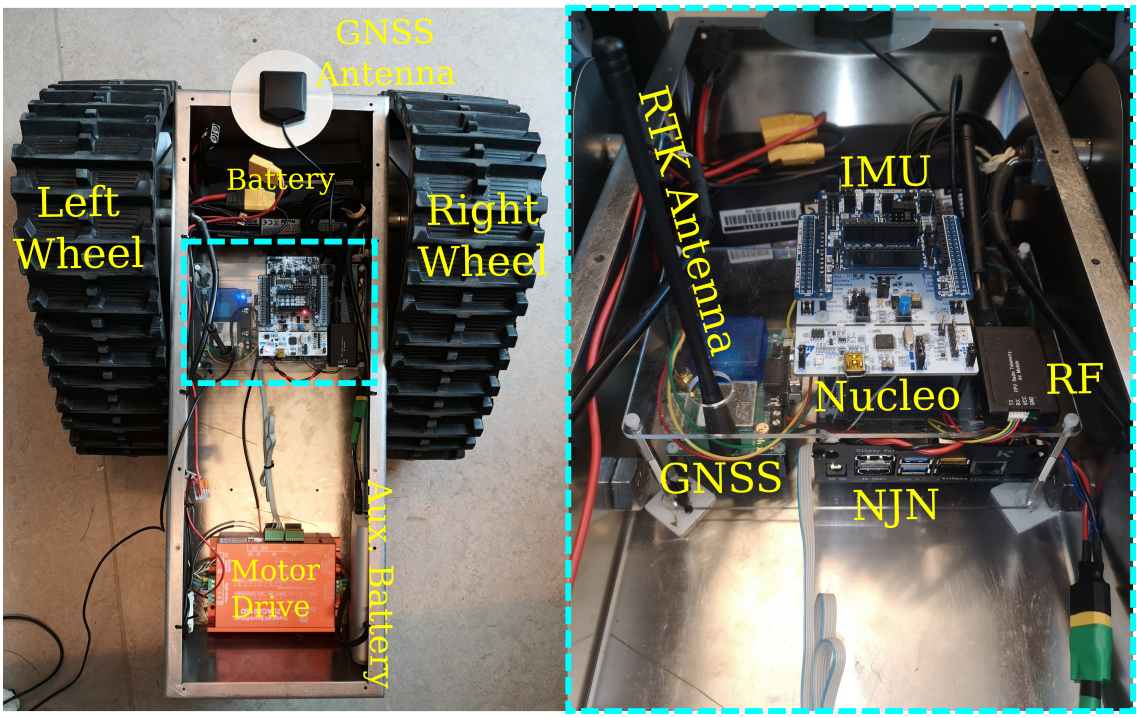


Figure 2.1. Picture of physical setup with annotations

2.1 System Structure

The internal communication of the system can be seen on Figure 2.2, giving an overview of how the system components interact. Starting from the bottom left corner two GNSS modules will be linked together to generate a Real-Time Kinematic (RTK) signal for the positioning of the rover. This RTK signal will, along with the data from the IMU and the wheel encoders, be processed by the Nucleo board after which the estimated rover states are sent to the NJN.

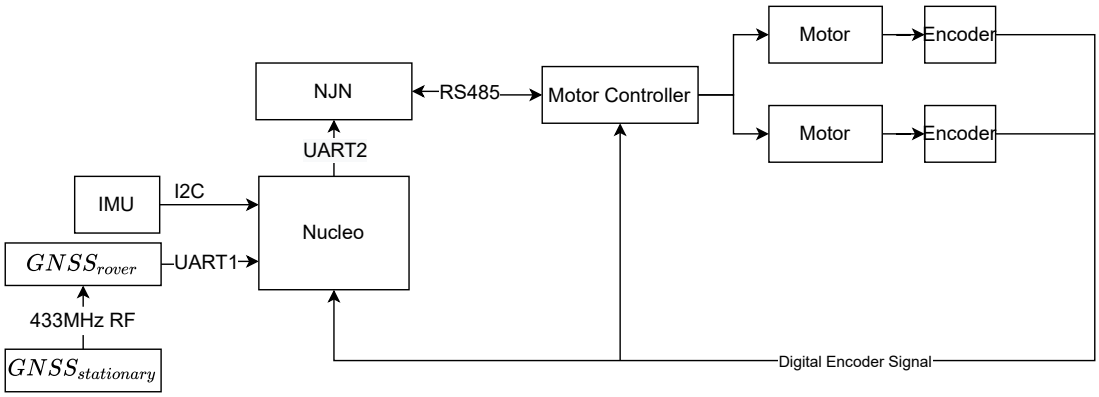


Figure 2.2. High level block diagram of system setup

One bottleneck expected to be encountered in the system is the RS485 connection between the NJN and the motor controller board. During the initial tests of the

motor controller, the board is observed only to be reliable in transmitting data to the NJN at a rate of approximately 10 Hz. This will heavily impact the bandwidth which may be expected from the controller and will serve as a justification as to why the controller may be deployed on the NJN although it does not guarantee real-time operation.

2.2 Wiring Diagram

The wiring diagram of the setup can be seen in Figure 2.3. Where the specific connections between the boards are outlined. As the IMU unit is a direct plugin for the Nucleo board no wiring is necessary between these. The 36V rail is to be supplied by the main battery of the rover while a separate auxiliary 5V battery will be used to supply power for the different modules of the rover.

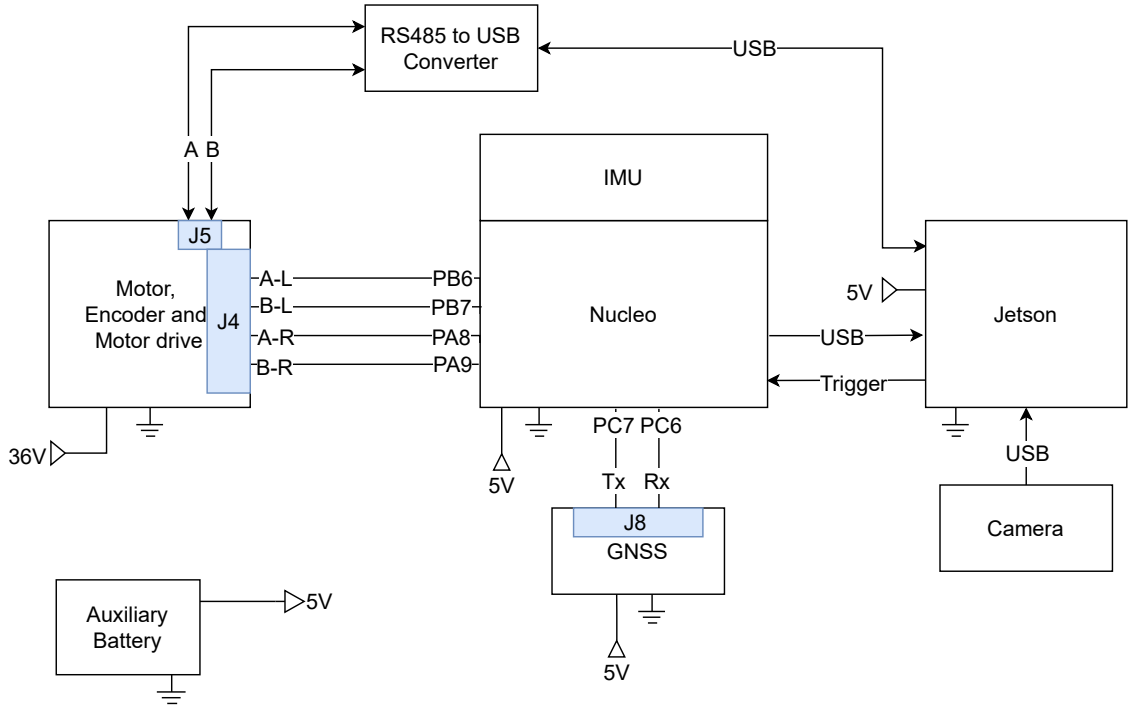


Figure 2.3. Wiring diagram of rover

2.3 ROS Implementation on NJN

In ROS the system is made up of different nodes, running python scripts, all with the possibility of publishing and/or subscribing to different topics. With this, it is possible to run multiple scripts at the same time while exchanging information. An illustration of the ROS structure can be seen on Figure 2.4, and the different nodes will be further elaborated in the following subsections.

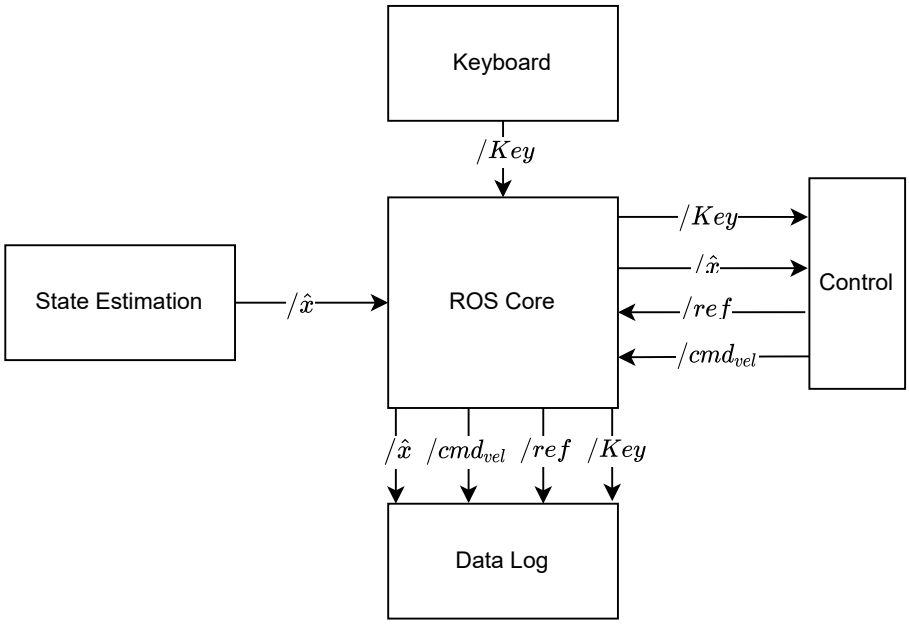


Figure 2.4. Node structure in ROS

State Estimation

This node is intended to read the data coming in on the USB connection from the Nucleo. This data is made up of the system states calculated by the Kalman filter, as well as a timestamp for when the data has been sent from the Nucleo. This information is published to the $/\hat{x}$ topic.

Keyboard

This node is for controlling the rover with a keyboard. It reads what keys are being pressed, and publishes that information to the $/Key$ topic. This node is primarily introduced as it will allow for remote control of the rover to e.g. change mode and perform manual overwrite.

Control

The control node is made up of multiple functionalities. The first functionality is reading a predetermined trajectory from a .CSV file. The second is writing the commanded motor values to the motor controller. The third is the actual controller. This node subscribes to the $/Key$ and $/\hat{x}$ topics. What key is pressed has multiple purposes in this node. Different keyboard keys have been mapped to determine whether the rover should run a trajectory, disable motors, or perform manual

overwrite of the commanded velocities. $/Key$ is also mapped to determine the rover's linear and angular velocity during manual control. The node is publishing to the $/ref$ and $/cmd_{vel}$ topic, where $/cmd_{vel}$ also contains the timestamp for when the commanded velocity has been sent to the motors.

Data log

The data log note subscribes to the topics $/Key$, $/\hat{x}$, $/ref$ and $/cmd_{vel}$. A key is mapped to start and stop the data log. After the datalog has started the node gathers the different topics in a .CSV file.

Topics

A list of the topics and what they contain is listed in Table 2.1. $T_{\hat{x}}$ is the timestamp of when the estimated states is sent from the Nucleo and $T_{/cmd_{vel}}$ is the time stamp of when the velocity references are sent to the motors.

Table 2.1. Table og ROS topics and their contents

$/\hat{x}$	$/ref$	$/cmd_{vel}$	$/Key$
$T_{\hat{x}}$	x_{ref}	$T_{cmd_{vel}}$	key
$\hat{V}$	y_{ref}	$\dot{\phi}_{ref}$	
$\hat{\omega}$	θ_{ref}	$\dot{\phi}_{ref}$	
$\hat{\theta}$	V_{ref}		
$\hat{\dot{x}}$			
$\hat{\dot{y}}$			
$\hat{x}$			
$\hat{y}$			
$\hat{A}_{bias}$			
$\hat{G}_{bias}$			
$\hat{M}_{bias}$			

Expansion

Using ROS enables further expanding the capabilities of the rover. The nodes implemented at the moment are necessary for the basic functionality of the rover. However, more nodes can be implemented to enable features such as CV, without having to change the other nodes of the structure. With ROS being an open-source community-based platform, many different libraries and extensions already exist which means that additional functionality may easily be integrated.

System Model 3

In order to design control for the rover, a model of a differential drive robot is derived, folowing the method in [1].
A simplified illustration of the rover can be seen on Figure 3.1.

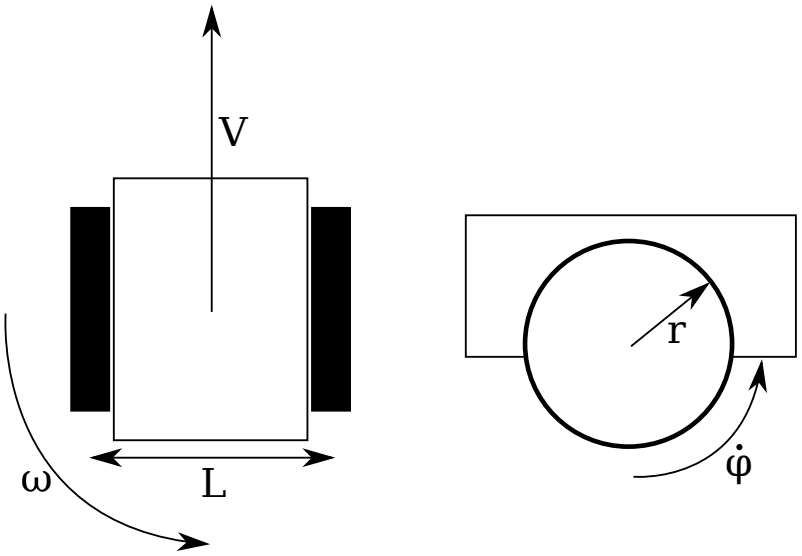


Figure 3.1. Simplified illustration of the rover

The motion can be described by the linear velocity(V) as well as the rotational velocity(ω). For the project the wheels of the rover will be assumed never to slip, this is justified by the fact that the rover will be driving on tracks and is further strengthened by the relative low accelerations expected. The linear velocity of each wheel can be described by the rotational velocity of the wheel multiplied by the radius of the wheel and can be related to the velocity and rotation of the rover as described in Equations (3.1) and (3.2).

$$V + \frac{L}{2}\omega = r\dot{\phi}_r \tag{3.1}$$

$$V - \frac{L}{2}\omega = r\dot{\phi}_l \tag{3.2}$$

Where r is the radius of the wheel, $\dot{\phi}_r$ and $\dot{\phi}_l$ is the angular velocity of the motor on the right and left side of the rover, and L is the length between the wheels.
Solving Equations (3.1) and (3.2) for V and ω respectively results in Equations (3.3) and (3.4).

$$V = \frac{r}{2}(\dot{\phi}_r + \dot{\phi}_l) \tag{3.3}$$

$$\omega = \frac{r}{L}(\dot{\phi}_r - \dot{\phi}_l) \tag{3.4}$$

Which may be represented on matrix form as Equation (3.5), with $\boldsymbol{M}$ being the kinematic relation matrix between wheel rotational velocities and linear/rotational velocity of the rover.

$$\begin{bmatrix} V \\ \omega \end{bmatrix} = \boldsymbol{M} \begin{bmatrix} \dot{\phi}_r \\ \dot{\phi}_l \end{bmatrix} \tag{3.5}$$

$$\boldsymbol{M} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{L} & -\frac{r}{L} \end{bmatrix}$$

3.1 Dynamic Model

The dynamics of the rover can be described by the torque at the two motors ($\mathcal{T}_r$ and $\mathcal{T}_l$), as seen in Equations (3.6) and (3.7), assuming the the center of mass is directly in between the two motors.

$$m\dot{V} = \left(\frac{\mathcal{T}_r}{r} - b\dot{\phi}_r\right) + \left(\frac{\mathcal{T}_l}{r} - b\dot{\phi}_l\right) \tag{3.6}$$

$$J\dot{\omega} = \left(\frac{\mathcal{T}_r}{r} - b\dot{\phi}_r\right)\frac{L}{2} - \left(\frac{\mathcal{T}_l}{r} - b\dot{\phi}_l\right)\frac{L}{2} \tag{3.7}$$

Where m is the mass of the rover, J is the moment of inertia about the center of mass, and b is the friction coefficient, assuming that the friction at both wheels are the same.
This results in the matrix form in Equation (3.8).

$$\begin{bmatrix} \dot{V} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \frac{1}{mr} & \frac{1}{mr} \\ \frac{L}{2rJ} & -\frac{L}{2rJ} \end{bmatrix} \begin{bmatrix} \mathcal{T}_r \\ \mathcal{T}_l \end{bmatrix} + \begin{bmatrix} -\frac{b}{m} & -\frac{b}{m} \\ -\frac{bL}{2J} & \frac{bL}{2J} \end{bmatrix} \begin{bmatrix} \dot{\phi}_r \\ \dot{\phi}_l \end{bmatrix} \tag{3.8}$$

Taking the time derivative of Equation (3.5) and inserting into Equation (3.8), results in the state space model of the system as given in Equation (3.9).

$$\begin{bmatrix} \ddot{\phi}_r \\ \ddot{\phi}_l \end{bmatrix} = \boldsymbol{A_{ss}} \begin{bmatrix} \dot{\phi}_r \\ \dot{\phi}_l \end{bmatrix} + \boldsymbol{B_{ss}} \begin{bmatrix} \mathcal{T}_r \\ \mathcal{T}_l \end{bmatrix} \tag{3.9}$$

$$\boldsymbol{A_{ss}} = \begin{bmatrix} -\frac{b\left(L^2m+4J\right)}{4rmJ} & -\frac{b\left(-L^2m+4J\right)}{4rmJ} \\ -\frac{b\left(-L^2m+4J\right)}{4rmJ} & -\frac{b\left(L^2m+4J\right)}{4rmJ} \end{bmatrix}, \boldsymbol{B_{ss}} = \begin{bmatrix} \frac{L^2m+4J}{4r^2mJ} & \frac{-L^2m+4J}{4r^2mJ} \\ \frac{-L^2m+4J}{4r^2mJ} & \frac{L^2m+4J}{4r^2mJ} \end{bmatrix}$$

3.2 Transformation to Inertial Frame

The rover model represented above is in a body-fixed rover frame. For this project, the rover will be assumed only to be moving in a two-dimensional xy plane. To track the rover in the inertial frame a transformation of the rover states will have to be made. This can be done with simple geometric relations based on Figure 3.2 given the angle of the rover with respect to the inertial frame θ and its velocity V as Equations (3.10) and (3.11).

$$\dot{x} = V\, cos(\theta) \tag{3.10}$$

$$\dot{y} = V\, sin(\theta) \tag{3.11}$$

Where the body-fixed velocity of the rover V is defined in the direction of y_b as the rover is constrained to linear movement in this direction.

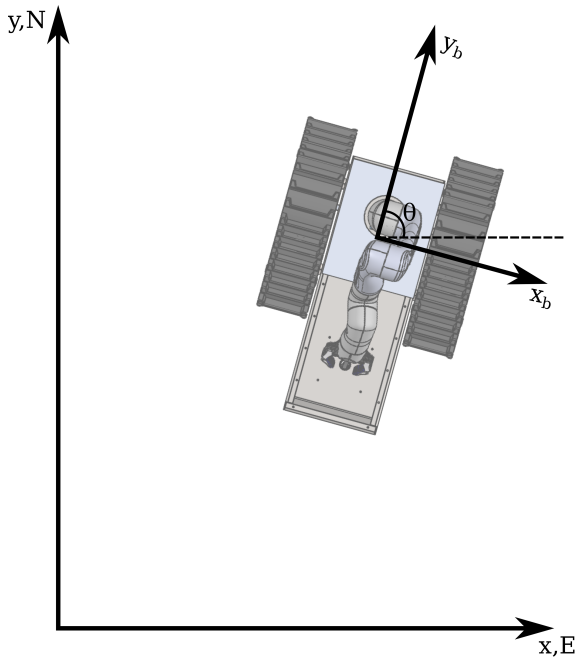


Figure 3.2. Rover body frame inside inertial north east frame

Parameter Identification 4

4.1 System Parameters

To make the model emulate the physical system the parameters of the system have to be determined. As the motor controller is a closed system with no access to set the controller gains. These gains will have to be found experimentally along with the weight of the rover(m), the inertia(J), and the friction coefficient(b). A block diagram of the combined motor drive and rover dynamics can be seen in Figure 4.1.

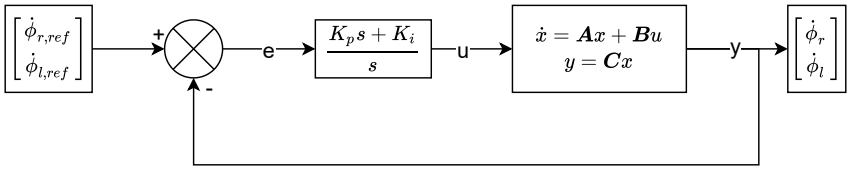


Figure 4.1. combined motor and motor drive system structure

Setting up the model in Simulink, and subjecting it to the same motor velocity reference as used in the physical system, the parameters can be determined using the Parameter Estimation plugin to Simulink. The Parameter Estimation tool, is an optimization tool using the sum of squares cost function on the error between the real values and the simulated values, in order minimize this. The reference is a pseudo-random reference with steps close to the expected working point of the rover. A similar reference on both motors result in the rover moving forwards, whereas different references result in the rover turning. The result of the fitted model compared to the encoder data from the rover can be seen on Figure 4.2, where it is plotted along with the reference.

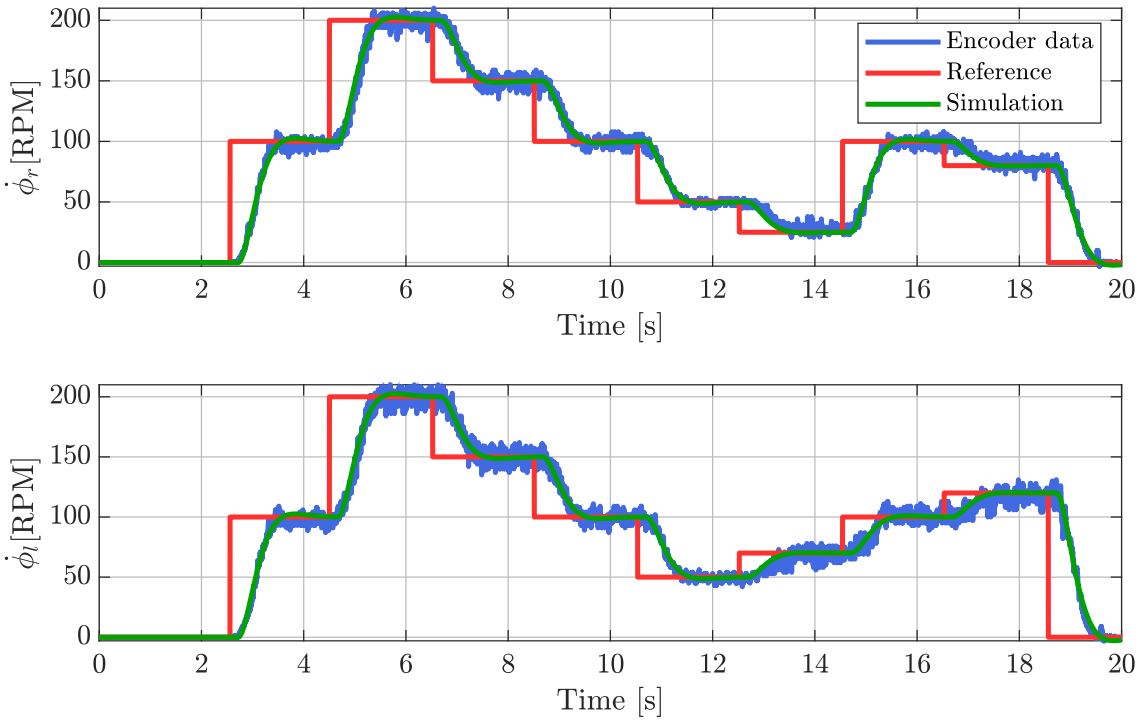


Figure 4.2. Physical and model response to sudo random reference

The fitted parameters are listed in Table 4.1. It should be noted that these parameters, except for radius(r) and width(L), do not represent the actual values of the system, only a combination that makes the model respond similarly to the physical system. Finding the actual parameters is out of scope for this project, as this part of the system is seen as a black box that transforms $\dot{\phi}_{ref} \rightarrow \dot{\phi}$.

Table 4.1. Parameters used for the model

	Symbol	Value	Unit
Radius of wheel	r	0.086	m
Width of rover	L	0.41	m
Mass of rover	m	35	kg
Inertia around center of rotation	J	1.4	kgm^2
Friction Coefficient	b	10	Ns/rad
Proportional Gain	K_p	1e−5	-
Integral Gain	K_i	2.4	-

A time delay from when the NJN sends a reference to when the motors start moving is observed which is compensated in the parameter estimation, but still visible on Figure 4.2. This will be further elaborated in the following section.

4.2 Time Delay

The time delay observed is better illustrated on a zoom of Figure 4.2 as seen on Figure 4.3. The time delay is observed to be approximately 150 ms and is relatively consistent across different steps in reference.

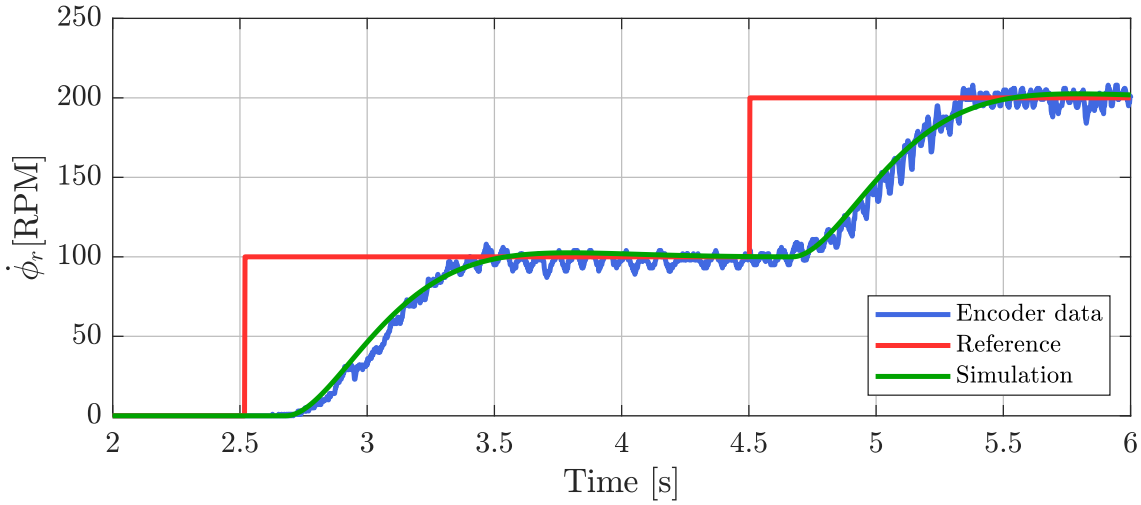


Figure 4.3. Illustration of system time delay

The time delay can come from different places in the system, the main concerns being the possible delay caused by the operating system (OS) on the NJN, and the other being, caused by the motor controller, over which we have no control.
To evaluate a potential delay caused by the NJN, a logic analyzer has been connected to the system. The logic analyzer is connected to the output port of the USB to the RS485 converter, as well as the trigger signal used for making Figure 4.2. This trigger signal is set to high when the reference is set in the code and then set to low when the code is done sending. The result of this can be seen on Figure 4.4.

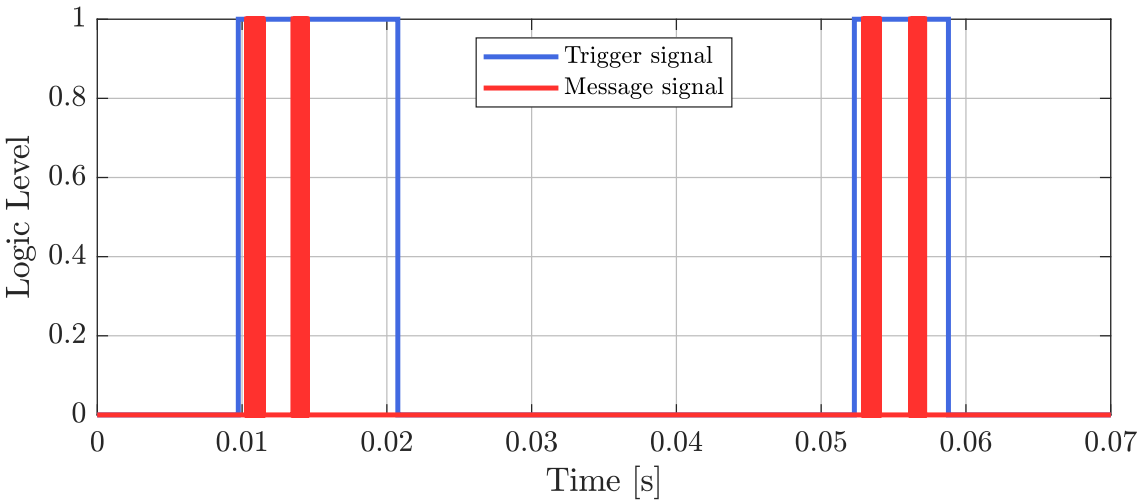


Figure 4.4. Trigger signal compared to the message signal

It is observed that when the reference signal is set, setting the trigger signal to high, a small amount of time passes before the message signal starts coming through. The falling edge of the trigger signal may therefore be seen as a worst-case scenario of the OS delay.
With a total time delay of approximately 150 ms, the OS delay observed on Figure 4.4, is relatively small. This means that the larger part of the time delay comes from the motor drive.

4.3 Discussion

Even though the parameters make the model fit the actual trajectory, it is important to note that the parameters do not represent the actual physical properties of the system. This will potentially cause problems when adding a new feature to the rover such as the robotic arm. In this case, a new parameter fit will have to be done, as the weight of the arm cannot simply be added to the weight, based on the parameter fit, of the rover. Additional work can be done to determine the exact parameters, but since there is no way of changing the controller gains in the motor drive, this is not beneficial for the project.
The time delay in the motor drive is higher than what would be expected. When looking at the step response of the system on Figure 4.3, an approximate settling time of 0.8 seconds can be observed. Based on this the response may be approximated as a first-order system with a time constant of approximately 200 ms, it would be expected that the time delay is much lower than this. This is however not the case with a 150 ms time delay. This will consequently add additional phase to the system which results in lower stability margins. This has to be taken into account when designing the controllers.

Sensor Calibration and Interpretation 5

In the following chapter, the sensors of the system will be presented while calibration and interpretation steps for the individual sensors will be performed to use the data for the state observer. Sensor models will in addition to this be made for use in the simulation. This is done in order for the simulation results to represent the performance that may be expected from the physical setup.

5.1 Encoders

The raw signal from the encoders is fed directly into the microcontroller. The encoder signals are interpreted in the microcontroller as an angular position of the motors. Each encoder outputs two signals, so it is possible to know the direction of rotation. Each signal provides two counts per tick of the encoder. The encoder used in this project has 4096 ticks per revolution, which means the microcontroller gets four times that per rotation. The position of the motor can be determined by $\phi = n_t \frac{2\pi}{4 \cdot 4096}$, where n_t is the recorded number of ticks. Calculating the discrete time derivative of this position yields the rotational velocity of the motor which can be described by Equation (5.1).

$$\dot{\phi} = \frac{\phi_k - \phi_{k-1}}{t_{enc}} \tag{5.1}$$

With subscript k representing the current time step, and t_{enc} representing the size of the time step, which for the encoder in this project is set to 10 ms.

5.2 GNSS

The GNSS module gets signals from the surrounding satellites to get a geographic position on earth, represented by latitude and longitude. It is desirable to get this position represented in a local east north up (ENU) frame, though we are not interested in the up component, as the rover will be assumed only to move in an east north plane. This is done with the Equirectangular projection, which can be used to estimate cartesian coordinates based on the latitude and longitude. The formula used for this is presented in Equation (5.2) [2].

$$x = R_{earth}(lon - lon_o)\cos(lat_o) \tag{5.2}$$

$$y = R_{earth}(lat - lat_o) \tag{5.3}$$

With x being aligned with east, y aligned with north, and R_{earth} being the mean radius of the earth. Subscript o represents the position at the origin of the chosen coordinate system.

This is a simple map projection that projects the earth to a cylindrical shape. This makes it only accurate along the equator line. In order to compensate for this the term $\cos(lat_o)$ is added to the x coordinate.

To compensate for some of the errors associated with using a GNSS, an RTK link between two receivers can be used. This is due to the fact that a signal traveling from a satellite to a receiver on earth is bound to pick up some disturbances on the way. By placing a stationary base module at a known location in the vicinity of the rover, the error between the GNSS position and the actual position can be measured. This error is then transmitted to the rover to compensate for the same error.

5.3 Accelerometer

For the calibration of the accelerometer, a few assumptions have to be made in order to simplify the calculations. It is assumed that the accelerometer is placed parallel to the floor of the rover and the rover is driving on a level surface so that the gravitational effect on the accelerometer is only present in the z direction. This gravitational effect is neglected as it is assumed that the rover only moves in the xy plane. In the case where the rover does not move on a level surface, this assumption does not hold, and the contribution from the gravitational acceleration will not only be limited to the z axis. This potential accelerometer bias will have to be considered when designing the state estimator. It is desirable to rotate the coordinate frame of the accelerometer to align with the coordinate frame of the rover. This potential offset is illustrated on Figure 5.1.

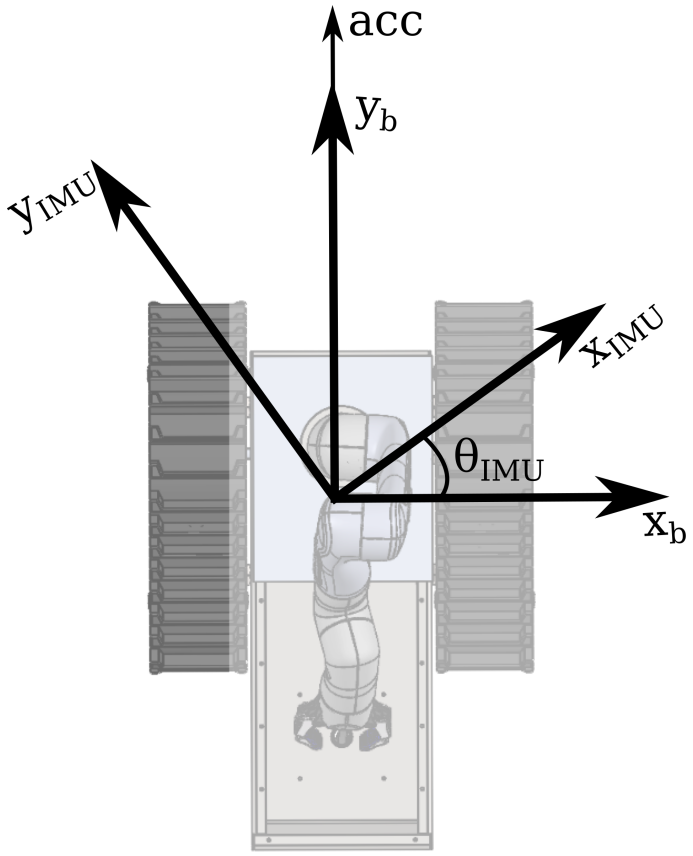


Figure 5.1. Rotation of the accelerometer coordinate frame to the rover body frame

The correction can be calculated by the rotation matrix seen in Equation (5.4).

$$\begin{bmatrix} a_{b,x} \\ a_{b,y} \end{bmatrix} = \begin{bmatrix} \cos\theta_{IMU} & -\sin\theta_{IMU} \\ \sin\theta_{IMU} & \cos\theta_{IMU} \end{bmatrix} \begin{bmatrix} a_{IMU,x} \\ a_{IMU,y} \end{bmatrix} \tag{5.4}$$

As the linear acceleration only happens in the y direction of the body frame, only the calculation of $a_{b,y}$ is necessary. The angle θ_{IMU} will depend on the mounting of the sensor and will have to be found experimentally. This can be done by driving both motors at the same velocity, resulting in the rover moving in a straight line while recording the acceleration values. This should ideally only result in an acceleration in the y direction, and it is possible to calculate θ_{IMU} by using the inverse tangent, as described in Equation (5.5).

$$\theta_{IMU} = 90^\circ - \tan^{-1}\frac{a_{IMU,y}}{a_{IMU,x}} \tag{5.5}$$

As this angle of rotation between the coordinate frames does not change over time, it is not necessary to use computing power on calculating $\cos\theta_{IMU}$ and $\sin\theta_{IMU}$ on the Nucelo as these will be constant, and therefore only need to be calculated once. This calibration of the accelerometer is under the assumption that it is placed directly in the center of rotation of the rover. This is likely not to be the case,

as the ideal mounting point may be obstructed by other components such as the robotic arm. For the case where the accelerometer is not placed ideally, as seen on Figure 5.2, rotation of the rover will also affect the accelerometer readings.

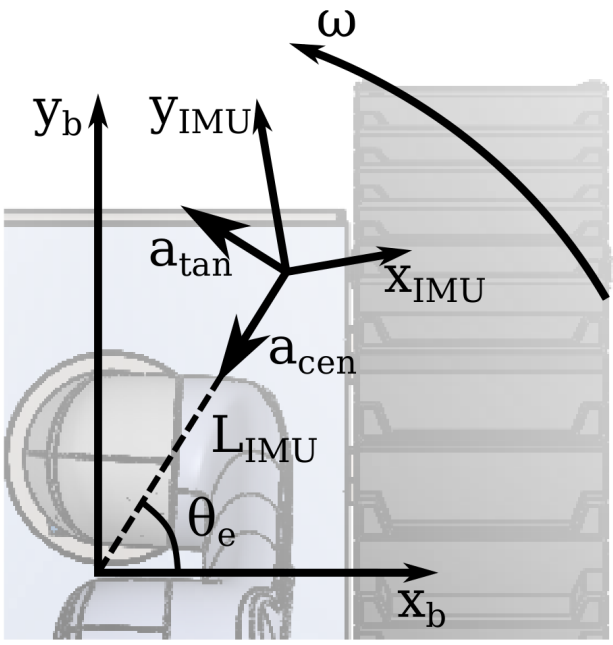


Figure 5.2. Rotation and displacement of the accelerometer coordinate frame to the rover body frame

The first thing to consider is the centripetal acceleration which is excited by the rover rotating, and has an acceleration pointing towards the center of rotation, this acceleration is describe in Equation (5.6).

$$a_{cen} = \omega^2 L_{IMU} \tag{5.6}$$

Where L_{IMU} is the displacement length from the rover's center of rotation to the accelerometer. The second thing to consider is the resulting linear tangential acceleration a_{tan} of the IMU coming from the angular acceleration of the rover, this can be described by Equation (5.7).

$$a_{tan} = \dot{\omega} L_{IMU} \tag{5.7}$$

Both these accelerations will have to be subtracted from the measurements, for the accelerometer to represent the linear acceleration of the vehicle body. Combining Equations (5.5) to (5.7) and rotating the centripetal and rotational component to the vehicle body frame yields Equation (5.8).

$$\begin{bmatrix} a_{b,x} \\ a_{b,y} \end{bmatrix} = \begin{bmatrix} \cos\theta_{IMU} & -\sin\theta_{IMU} \\ \sin\theta_{IMU} & \cos\theta_{IMU} \end{bmatrix} \begin{bmatrix} a_{IMU,x} \\ a_{IMU,y} \end{bmatrix} - \omega^2 L_{IMU} \begin{bmatrix} \cos\theta_e \\ \sin\theta_e \end{bmatrix} - \dot{\omega} L_{IMU} \begin{bmatrix} \sin\theta_e \\ \cos\theta_e \end{bmatrix} \tag{5.8}$$

By measuring the rotational velocity with the gyroscope, and using the previously calculated value for θ_{IMU} it is possible to calculate the constants and compensate for the displacement of the sensor when calculating the body frame linear accelerations.

Even though compensating for the displacement of the accelerometer is possible, it is not ideal as it would mean the accelerometer signals, are dependent on the gyroscope signal. This means that the noise-dependent variance of the accelerometer signal includes the gyroscope noise. This could be a problem since the gyroscope signal is squared, resulting in the noise no longer being Gaussian distributed, which could cause problems in the calculation of the Kalman filter.
To avoid having to compensate for this displacement, it is desirable to place the accelerometer as close as possible to the center of rotation. This would make L small and possibly make the acceleration contribution of the rotation negligible.

5.3.1 Accelerometer Model

To design a model of the rover, it is necessary to make a model of the accelerometer. This is done by taking Equation (5.8) and isolating for the accelerometer signal $a_{acc,x}$ and $a_{acc,y}$. This can be seen in Equation (5.9)

$$\begin{bmatrix} a_{IMU,x} \\ a_{IMU,y} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}^{-1} \left(\begin{bmatrix} a_{b,x} \\ a_{b,y} \end{bmatrix} + \omega^2 L_{IMU} \begin{bmatrix} \cos\theta_e \\ \sin\theta_e \end{bmatrix} + \dot{\omega} L_{IMU} \begin{bmatrix} \sin\theta_e \\ \cos\theta_e \end{bmatrix} \right) \tag{5.9}$$

The accelerometer values are measured in thousands of a g, where g is the gravitational force which approximates to $1\,\mathrm{g} = 9.82\,\mathrm{m/s^2}$. Therefore the output signal from the accelerometer has to be scaled with $9.82 \cdot 10^{-3}\,\mathrm{m/gs^2}$

5.3.2 Accelerometer Testing

Driving the rover forward and backward with 200 RPM on both motors, results in the raw accelerometer data seen on Figure 5.3. The rover stands still from approximately 0-1.5 s, where it starts driving forwards nearly until the 4 s mark, after which it drives backward until around the 7 s mark where it stands still again.

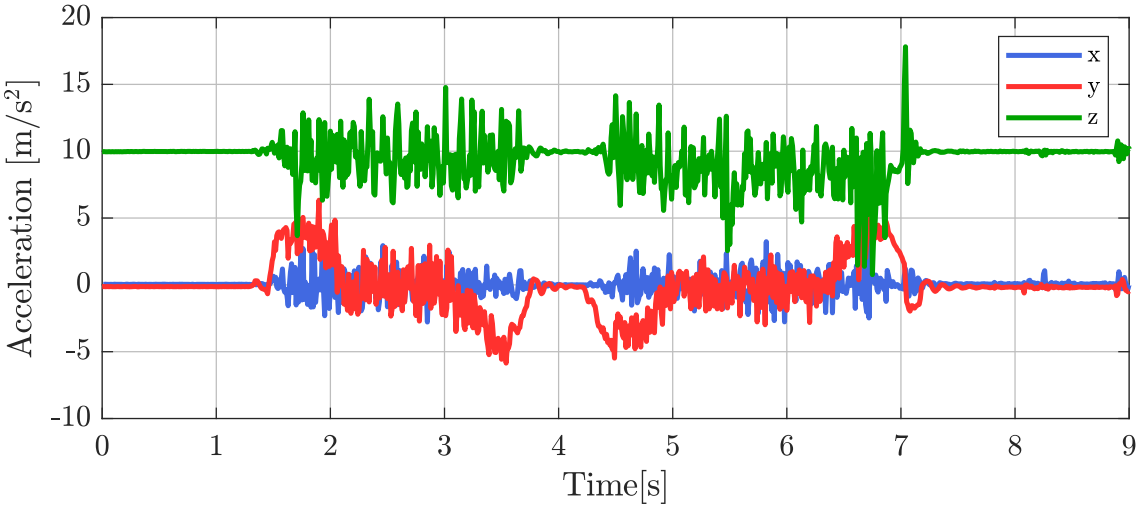


Figure 5.3. Raw accelerometer data from the rover driving forth and back

It is observed that the acceleration in the z direction contains approximately $9.82\,\mathrm{m/s^2}$ at a standstill which is expected, and picks up a lot of disturbances and noise when the rover is driving, especially in reverse where it seems to diverge from its mean. This may be caused by the rover tilting slightly, due to the absence of the trailing wheel, as well as heavy components such as the battery not being strapped down, causing a shift in the weight distribution. The x direction seems to pick up nothing but disturbance and noise. While the y direction picks up the acceleration and deceleration of the rover. This indicates that the IMU frame is aligned with the body frame of the rover.
Taking the acceleration and integrating this over time in order to get the velocity results in Figure 5.4, where it is compared to the velocity calculated from the wheel encoders. The accelerometer data is compensated for the small dc-offset detected before the rover starts moving.

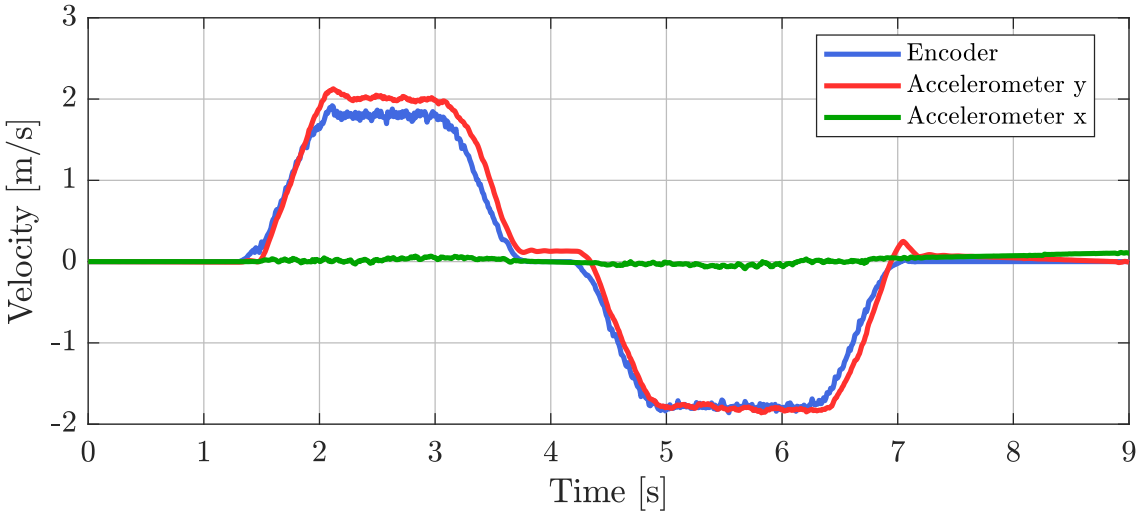


Figure 5.4. Velocity of the rover as determined by the accelerometer and the encoder

The accelerometer-based velocity is seen to compare well against the encoder-based velocity. There is some deviation but overall show the same tendencies. The plot of the integrated acceleration in the x direction is also shown, emphasizing the validity of the assumption that the linear acceleration is only observed in the y direction.
It is observed that while the rover stands still in between going forwards and backward, the accelerometer determined velocity still shows movement. This is likely due to the rover tilting slightly while accelerating, and thereby the y axis picking up some of the gravitational acceleration, resulting in an offset in the velocity.
It is also important to notice that the accelerometer velocity drifts slightly at the end of the trajectory. This is due to the dc offset, which has been compensated for, does not stay the same, and therefore is included in the integration over time causing a drift. Figure 5.4 is a short trajectory, with relatively high velocities and the rover driving on a smooth surface. Driving the rover back and forth multiple times for a longer period on a rough surface reveals the plot seen on Figure 5.5.

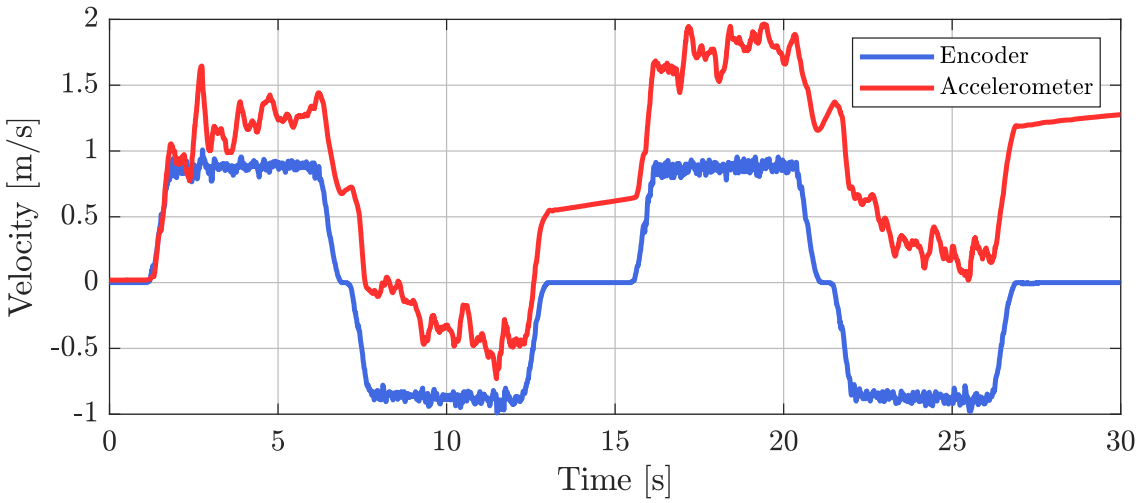


Figure 5.5. Velocity of the rover as determined by the accelerometer and the encoder

On this graph, a much higher drift is observed. This is likely due to multiple reasons, one being the rover running for a longer time and therefore keeps integrating over a small error in the acceleration value. Another reason is the rover moving on rougher terrain and therefore picks up a lot of disturbances. The third reason might be that the rover does not move as fast as in Figure 5.4, and therefore the difference between actual acceleration and disturbances is smaller and harder to distinguish.

5.4 Gyroscope

The gyroscope measures a rotational velocity and does not need any calibration for it to output a useful signal. With this sensor, the only signal that is of interest is the rotation around the z axis. The velocity is measured in thousands of a degree

per second. This has to be converted in order to be used in the calculations. This is done by multiplying it with $\frac{1}{1000}\frac{2\pi}{360}$.
It is assumed that the gyroscope is placed parallel to the ground which eliminates the need for any calibration of this sensor. Looking at the data from the rover rotating around its center of rotation on a level surface, as seen on Figure 5.6, only a noticeable signal on the z axis is observed.

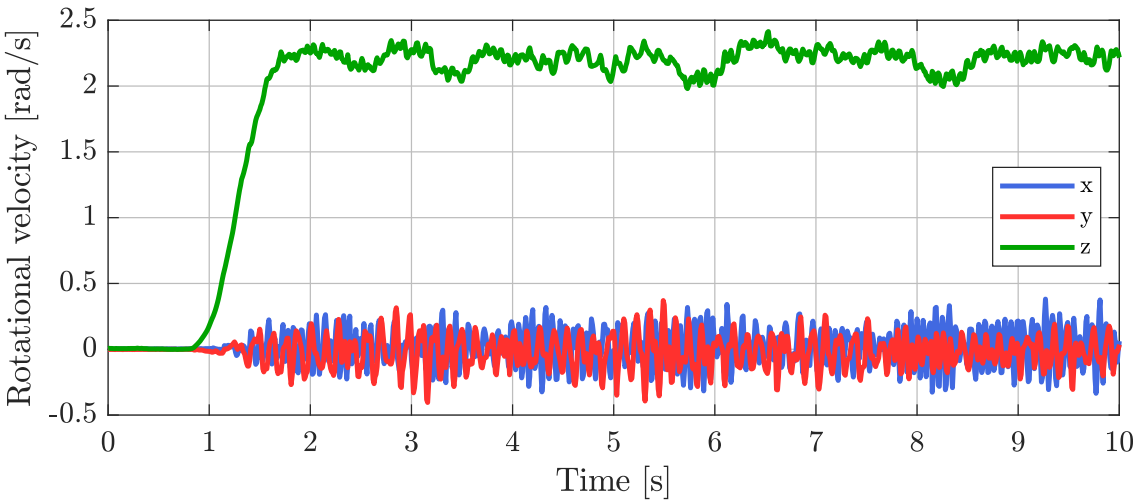


Figure 5.6. Raw data of the gyroscope

5.5 Magnetometer

The magnetometer measures the heading of the rover based on the earth's magnetic field. This does however also mean that other magnetic sources will have an impact on the reading of the magnetometer. The raw data from the magnetometer can be seen on Figure 5.7. This data is made by rotating the motors of the rover in opposite directions on a level surface, resulting in the rover rotating around its center of rotation.

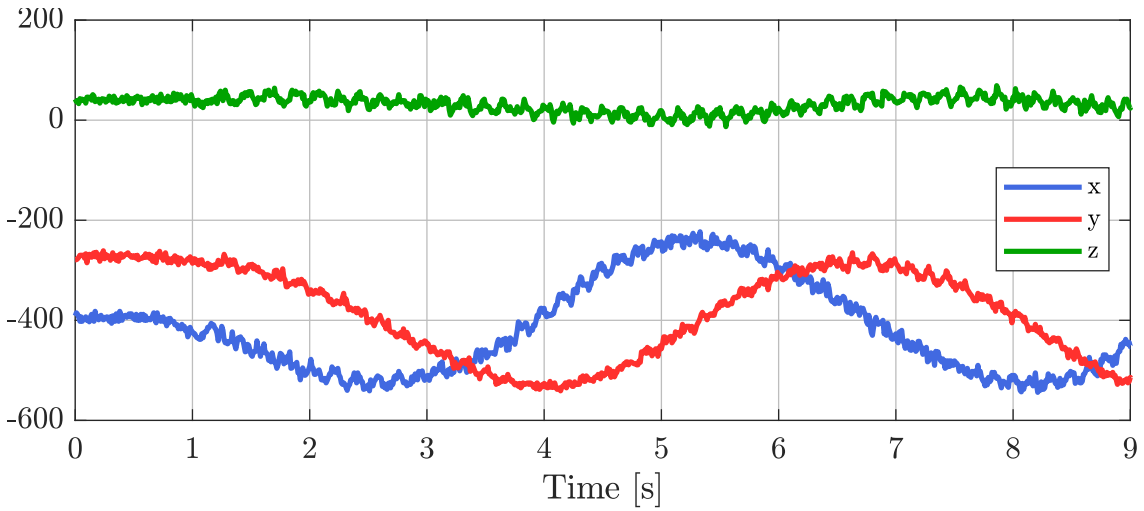


Figure 5.7. Raw data of the magnetometer

Since the rover is assumed to only have planar movement, the z component can be ignored. Plotting the x and y components against each other results in a circle offset from the origin of the coordinate system. This offset is due to hard iron disturbances and can be compensated by taking the average of the maximum and minimum values of the x and y values, and subtracting them from the respective data points, as described in Equation (5.10). The effect of this can be seen in Figure 5.8.

$$\begin{bmatrix} x_{mag,o} \\ y_{mag,o} \end{bmatrix} = \begin{bmatrix} x_{mag} \\ y_{mag} \end{bmatrix} - \frac{1}{2} \begin{bmatrix} x_{mag,max} + x_{mag,min} \\ y_{mag,max} + y_{mag,min} \end{bmatrix} \tag{5.10}$$

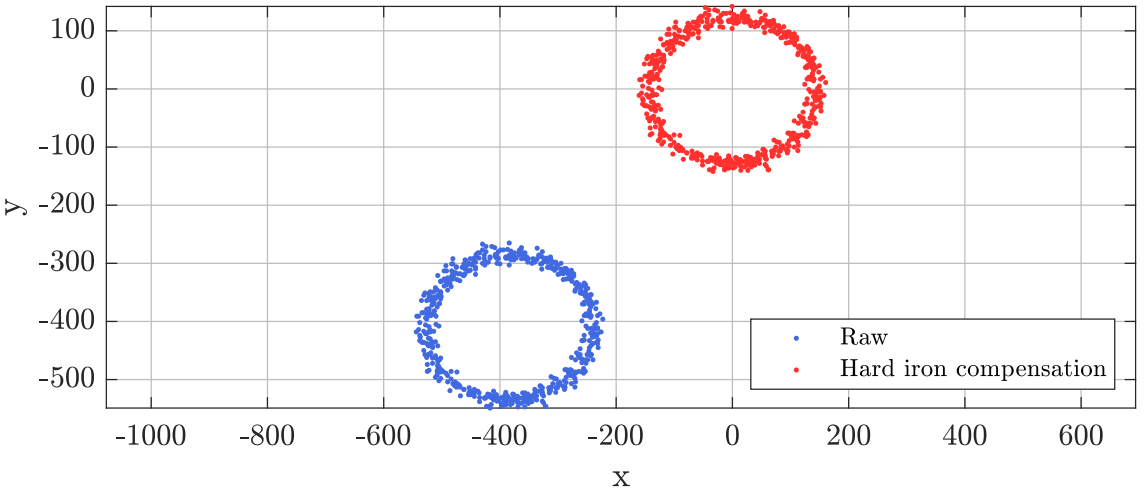


Figure 5.8. Magnetometer values with the hard iron disturbance compensation

Soft iron sources can cause the shape of the data point to be ellipsoidal rather than the desired circular shape. This can be compensated by taking the largest value in the x direction and dividing it by the largest value in the y direction, for

normalization of the data as seen in Equation (5.11). The data in the y direction remains unchanged in this step. The effect of this compensation can be seen on Figure 5.9.

$$\begin{bmatrix} x_{mag,os} \\ y_{mag,os} \end{bmatrix} = \begin{bmatrix} \frac{y_{mag,o,max}}{x_{mag,o,max}} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{mag,o} \\ y_{mag,o} \end{bmatrix} \tag{5.11}$$

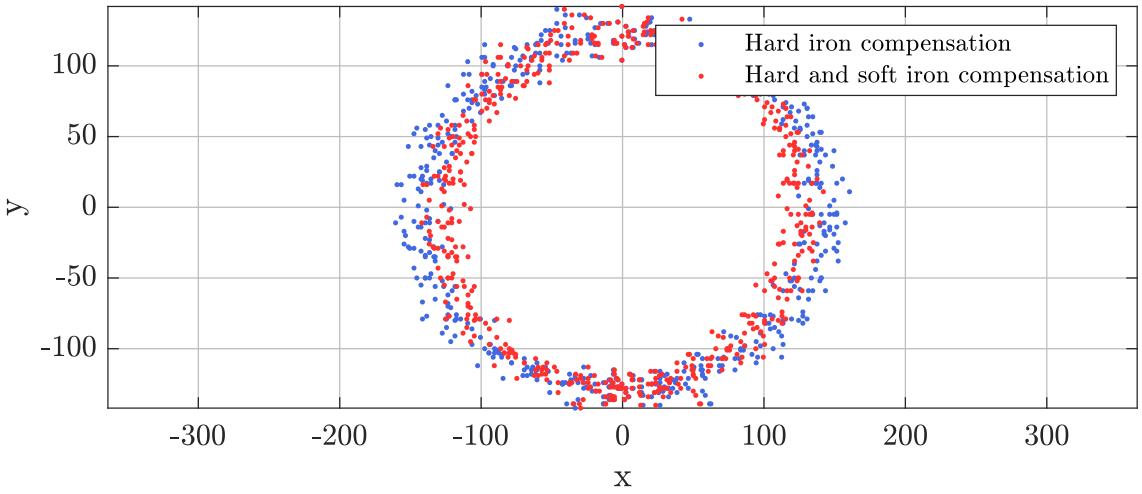


Figure 5.9. Magnetometer values with the hard iron disturbance compensation

It should be noted that when the rover does not move on a level surface compensation does not hold true, as the ellipsoid would look different. This is a bias that should be considered in the design of the state estimator.
With the magnetometer data scaled correctly, it is possible to calculate the angle of the rover in comparison to the inertial frame (θ). This is done with the atan2 function.

$$\theta = \text{atan2}(y_{mag,os}, x_{mag,os}) \tag{5.12}$$

atan2 returns a value in the interval $-\pi > \theta < \pi$, this is not desirable as this makes a non linear jump for every full rotation of the rover. Therefor everytime the rover completes a rotation 2π is added to θ and subtracted when completing a rotation in the opposite direction.
As a sanity check this angle can be plotted as seen on 5.10. Here the rover starts at an arbitrary angle and rotates around its center of rotation.

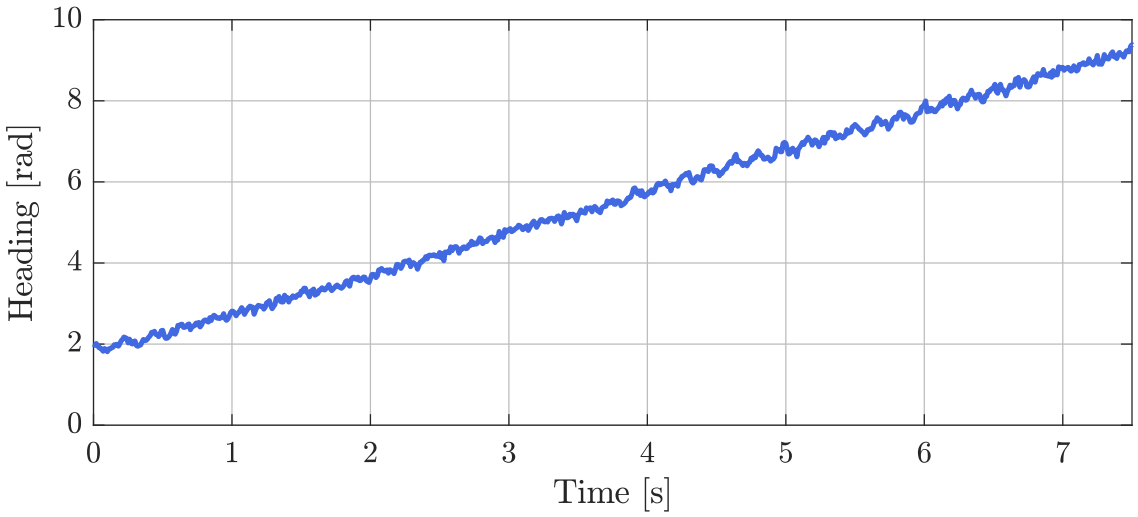


Figure 5.10. Heading of the rover based on the magnetometer readings

With all of the sensors being calibrated the actual values provided from the sensors may now be utilized. To combine the measurements a state observer will therefore be designed in the following chapter to yield the best estimate of the rover states based on the available sensor information.

State Observer Design 6

In order to control the rover it is essential to know its current states. These states will for this project be estimated by means of an Extended Kalman Filter (EKF). This filter will take in data from the different sensors and fuse this into a state estimation for the system. Fusing data in this way will allow for the different sensor measurements to complement each other as different noisy measurements may be combined to establish a better state estimation than either of the individual sensors can.

6.1 Inertial Navigation Filter

As a starting point for the filter design, an inertial navigation filter (INF) will be designed, whereafter the GNSS input will be included to improve the overall accuracy of the state estimations in the following section. The exclusion of the GNSS signal will mean that the INF may be constructed exclusively on in the rover body frame which means that an ordinary linear Kalman filter may be used. The Kalman filter will therefore primarily be constructed and designed as a linear filter whereafter it will be extended to allow for the non-linearities introduced by the transformation from the body frame to an inertial frame.

The Kalman filter (KF) is a recursive prediction correction algorithm and it is therefore essential to be able to describe the system dynamics. The state transition and measurement function of a linear system with no direct feed through may be described as Equations (6.1) and (6.2) [3].

$$\dot{x} = \boldsymbol{A}\, x + \boldsymbol{B}\, u + w \tag{6.1}$$

$$z = \boldsymbol{C}\, x + v \tag{6.2}$$

Where x is the states of the system, z is the output of the system, and u is the system inputs. w and v are zero-mean Gaussian random variables with error covariance $\boldsymbol{Q}$ and $\boldsymbol{R}$ respectively. $\boldsymbol{A}$,$\boldsymbol{B}$, and $\boldsymbol{C}$ are the system, input, and output matrix respectively. Note here that the bold symbols represent matrix quantities.

Discretizing this using the forward euler integration $\dot{x} \approx \frac{x^{k+1}-x^k}{Ts}$ yields Equations (6.3) and (6.4), where the superscript k denotes current time step while $k+1$ denotes the time step one sampling time ahead, and T_s is the interval between each sample.

$$x^k = \boldsymbol{F}\, x^{k-1} + \boldsymbol{G}\, u^k + w^k \tag{6.3}$$
$$z^k = \boldsymbol{H}\, x^k + v^k \tag{6.4}$$

The system matrices are redefined as $\boldsymbol{F} = \boldsymbol{I} + \boldsymbol{A} \cdot T_s$, $\boldsymbol{G} = \boldsymbol{B} \cdot T_s$ and $\boldsymbol{H} = \boldsymbol{C}$ while $\boldsymbol{I}$ is the identity matrix. Based on this the Kalman filter can be implemented as described in [4] with the prediction and correction steps as seen in Table 6.1.

Table 6.1. Prediction and correction equations for linear Kalman filter

Prediction:		
1	Predict states	$\hat{x}_k^- = \boldsymbol{F}\, \hat{x}_{k-1} + \boldsymbol{G}\, u_k$
2	Predict error covariance	$\hat{\boldsymbol{P}}_k^- = \boldsymbol{F}\, \hat{\boldsymbol{P}}_{k-1}\, \boldsymbol{F}^T + \boldsymbol{Q}$
Correction:		
3	Inovation	$\tilde{y}_k = z_k - \boldsymbol{H}\, \hat{x}_k^-$
4	Calculate Kalman gain	$\boldsymbol{K_k} = \hat{\boldsymbol{P}}_k^-\, \boldsymbol{H}^T\, (\boldsymbol{H}\, \hat{\boldsymbol{P}}_k^-\, \boldsymbol{H}^T + \boldsymbol{R})^{-1}$
5	Correct state estimate	$\hat{x}_k = \hat{x}_k^- + \boldsymbol{K_k}\, \tilde{y}_k$
6	Update error covariance	$\hat{\boldsymbol{P}}_k = (\boldsymbol{I} - \boldsymbol{K_k}\, \boldsymbol{H})\, \hat{\boldsymbol{P}}_k^-$

Where $\hat{x}^-$ is the predicted state estimate, $\hat{x}$ is the corrected state estimate while $\hat{\boldsymbol{P}}^-$ is the predicted error covariance and $\hat{\boldsymbol{P}}$ is the corrected error covariance. $\boldsymbol{F}$, $\boldsymbol{G}$ and $\boldsymbol{H}$ are respectively the state transition, control, and observation matrix for the filter. $\tilde{y}$ is the error between the prediction model and the measured states z while $\boldsymbol{K}$ is the optimal Kalman gain for the correction of the state estimate. The superscript T of the table above denotes the transpose of a vector or matrix.

The IMU sensors measurements are assumed to be composed of the true measurement as well as a bias and a zero-mean Gaussian noise as seen in Equations (6.5) to (6.7).

$$a_{meas}^k = a_{true}^k + A_{bias}^k + v_a^k \tag{6.5}$$
$$\omega_{meas}^k = \omega_{true}^k + G_{bias}^k + v_g^k \tag{6.6}$$
$$\theta_{meas}^k = \theta_{true}^k + M_{bias}^k + v_m^k \tag{6.7}$$

Where A_{bias}, G_{bias} and M_{bias} is the accelerometer, gyroscope, and magnetometer bias respectively both acceleration and angular velocity are in the body frame of the rover while the magnetometer measurement is in the inertial frame. The acceleration measurement will however be integrated to convert it into the linear velocity of the rover such that $V_{meas}^k = \hat{V}^{k-1} + a_{meas}^k\, T_s$. This integration also highlights why it is essential to compensate for any sensor bias as it will accumulate error over time.

For the implementation into the system in question the state, input, and measurement vectors will be defined as in Equations (6.8) to (6.10). It should be

noted here that the input vector u for the KF is the measured rotational velocities of the two motors measured by the respective encoders, effectively removing motor dynamics from the KF.

$$\hat{x}^k = [\hat{V}^k\, \hat{\omega}^k\, \hat{\theta}^k\, \hat{A}^k_{bias}\, \hat{G}^k_{bias}]^T \tag{6.8}$$
$$u^k = [\dot{\phi}^k_r\, \dot{\phi}^k_l]^T \tag{6.9}$$
$$z^k = [V^k_{meas},\, \omega^k_{meas},\, \theta^k_{meas}]^T \tag{6.10}$$

The state propagation equations of the INF filter is.

$$\hat{V}^k = \frac{r}{2} \cdot \dot{\phi}^k_r + \frac{r}{2} \cdot \dot{\phi}^k_l \tag{6.11}$$
$$\hat{\omega}^k = \frac{r}{L} \cdot \dot{\phi}^k_r - \frac{r}{L} \cdot \dot{\phi}^k_l \tag{6.12}$$
$$\hat{\theta}^k = \hat{\theta}^{k-1} + T_s \cdot \hat{\omega}^{k-1} \tag{6.13}$$
$$\hat{A}^k_{bias} = \hat{A}^{k-1}_{bias} \tag{6.14}$$
$$\hat{G}^k_{bias} = \hat{G}^{k-1}_{bias} \tag{6.15}$$

Here it should be noted that the magnetometer bias M_{bias} is not included as an estimated state as it cannot be estimated based on known information (at least not without assuming perfect information of initial conditions). The sensor bias is here modeled as random walks as no information is available. The state propagation equations may be split up into the following $\boldsymbol{F}$ and $\boldsymbol{G}$ matrices.

$$\boldsymbol{F} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & T_s & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{G} = \begin{bmatrix} r/2 & r/2 \\ r/L & r/L \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{6.16}$$

The output equations of the model are based on the different sensor models as seen in Equations (6.17) to (6.19).

$$V^k_{meas} = \hat{V}^k + T_s \cdot \hat{A}^k_{bias} \tag{6.17}$$
$$\omega^k_{meas} = \hat{\omega}^k + \hat{G}^k_{bias} \tag{6.18}$$
$$\theta^k_{meas} = \hat{\theta}^k \tag{6.19}$$

Which yield the following $\boldsymbol{H}$ matrix.

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & T_s & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{6.20}$$

In addition to the system matrices, a covariance matrix for both process noise and measurement uncertainty will have to be defined. These will both be defined as diagonal matrices, effectively assuming no cross-correlation between the disturbances in either of the states as well as no correlation between the measurement noise. For the system, the values for the entries of the $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrix will be determined empirically.

$$\boldsymbol{Q} = \begin{bmatrix} q_V & 0 & 0 & 0 & 0 \\ 0 & q_\omega & 0 & 0 & 0 \\ 0 & 0 & q_\theta & 0 & 0 \\ 0 & 0 & 0 & q_{A_{bias}} & 0 \\ 0 & 0 & 0 & 0 & q_{G_{bias}} \end{bmatrix} \tag{6.21}$$

$$\boldsymbol{R} = \begin{bmatrix} r_{acc} & 0 & 0 \\ 0 & r_{gyro} & 0 \\ 0 & 0 & r_{mag} \end{bmatrix} \tag{6.22}$$

6.2 Extension of Kalman Filter

The above INF is exclusively designed to estimate the states of the system in a body-fixed frame. To expand the filter to include the data from the GNSS module the Kalman filter will have to be expanded to allow for the non-linearity from the sine and cosine relations introduced from the coordinate transformation from the body-fixed frame to a global one. To allow for the non-linear dynamics the state propagation equations may be rewritten to the more general form as Equation (6.23) while the measurement function is unchanged as it is linear for the system.

$$x^k = f(x^{k-1}, u^k) + w^k \tag{6.23}$$

The state transition function f may be used in its non-linear form for the state transition, but will have to be linarized locally in order for it to be used for the covariance calculations. The Jacobian of the state transition function will therefore have to be evaluated in the vicinity of the current predicted states as:

$$\boldsymbol{F}_k = \left.\frac{\partial f}{\partial x}\right|_{x^{k-1}, u^k} \tag{6.24}$$

The expanded state, input and measurement vectors may therefore be redefined as:

$$\hat{x}^k = [\hat{V}^k,\, \hat{\omega}^k,\, \hat{\theta}^k,\, \hat{\dot{x}}^k,\, \hat{\dot{y}}^k,\, \hat{x}^k,\, \hat{y}^k,\, \hat{A}^k_{bias},\, \hat{G}^k_{bias},\, \hat{M}^k_{bias}]^T \tag{6.25}$$

$$u^k = [\dot{\phi}^k_r,\, \dot{\phi}^k_l]^T \tag{6.26}$$

$$z^k = [V^k_{meas},\, \omega^k_{meas},\, m^k_{meas},\, x^k_{meas},\, y^k_{meas}]^T \tag{6.27}$$

The state transition function f is then defined by appending the state transition equations for the velocities and positions in the inertial x-y plane as well as the magnetometer bias (Equations (6.28) to (6.32)) to the state transition equations from the INF in Equations (6.11) to (6.15).

$$\hat{\dot{x}}^k = \hat{V}^{k-1} \cdot \cos(\hat{\theta}^{k-1}) \tag{6.28}$$

$$\hat{\dot{y}}^k = \hat{V}^{k-1} \cdot \sin(\hat{\theta}^{k-1}) \tag{6.29}$$

$$\hat{x}^k = \hat{x}^{k-1} + T_s \cdot \hat{\dot{x}}^{k-1} \tag{6.30}$$

$$\hat{y}^k = \hat{y}^{k-1} + T_s \cdot \hat{\dot{y}}^{k-1} \tag{6.31}$$

$$\hat{M}^k = \hat{M}^{k-1} \tag{6.32}$$

Calculating the Jacobian of f as defined in Equation (6.24) yields $\boldsymbol{F}_k$ as:

$$\boldsymbol{F}^k = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & T_s & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c(\hat{\theta}^{k-1}) & 0 & -\hat{V}^{k-1} \cdot s(\hat{\theta}^{k-1}) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s(\hat{\theta}^{k-1}) & 0 & \hat{V}^{k-1} \cdot c(\hat{\theta}^{k-1}) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & Ts & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & Ts & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.33}$$

Where the short hand notation $s(\theta)$ is used to represent $sin(\theta)$ and $c(\theta)$ for $cos(\theta)$. The output equations of the system can then be redefined as:

$$V^k_{meas} = \hat{V}^k + T_s \cdot \hat{A}^k_{bias} \tag{6.34}$$

$$\omega^k_{meas} = \hat{\omega}^k + \hat{G}^k_{bias} \tag{6.35}$$

$$\theta^k_{meas} = \hat{\theta}^k + \hat{M}^k_{bias} \tag{6.36}$$

$$x^k_{meas} = \hat{x}^k \tag{6.37}$$

$$y^k_{meas} = \hat{y}^k \tag{6.38}$$

Which results in a linear measurement matrix $\boldsymbol{H}$

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & T_s & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{6.39}$$

As for the INF the process noise covariance and the measurement uncertainty matrices will be defined as diagonal matrices as:

$$\boldsymbol{Q} = \begin{bmatrix} q_V & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & q_\omega & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & q_\theta & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & q_{\dot{x}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & q_{\dot{y}} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & q_x & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & q_y & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & q_{A_{bias}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & q_{G_{bias}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & q_{M_{bias}} \end{bmatrix} \tag{6.40}$$

$$\boldsymbol{R} = \begin{bmatrix} r_{acc} & 0 & 0 & 0 & 0 \\ 0 & r_{gyro} & 0 & 0 & 0 \\ 0 & 0 & r_{mag} & 0 & 0 \\ 0 & 0 & 0 & r_{GNSS} & 0 \\ 0 & 0 & 0 & 0 & r_{GNSS} \end{bmatrix} \tag{6.41}$$

This concludes the design of the traditional EKF. One problem general for both the KF and the EKF is however the fact that it relies on a matrix inversion of an n_z by n_z matrix where n_z is the dimension of the measurement vector for the calculation of the Kalman gain. Matrix inversion is in general a high computational load, and it may therefore be desirable to implement a different version where the Kalman gain is approximated rather than calculated.

6.3 Scalar Implementation

For this project, the implementation will be done by use of a scalar implementation as described in [5]. In this implementation, the Kalman gain is approximated one line at a time, which effectively reduces the matrix inversion into a scalar inversion. When doing so we define the innovation of the i'th measurement as $\tilde{y}_k^i = z_k^i + H^{(i,:)}\hat{x}_k^+$, where z_k^i is the i'th measurement and $H^{(i,:)}$ is the i'th row of $\boldsymbol{H}$. The Kalman gain denominator can thereafter be calculated based on the i'th diagonal of the measurement uncertainty matrix $\boldsymbol{R}$ which has now been reduced to a scalar value. Here it should be noted that this means that if any cross-correlation between the measurements exists they will be completely disregarded for the calculations.
Other than the lower computational load, scalar implementation also has the benefit that it allows for sensors with different sampling frequencies to be used. For the specific sensors used for this project, this is very essential as the measurements from IMU and the encoders, can be updated with a frequency of 1000 Hz while the magnetometer and GNSS module may be updated at 100 Hz and 2 Hz respectively. By using this scalar implementation it is possible to correct the predictions as new measurements become available while disregarding any unupdated measurements.

Table 6.2. Prediction and correction equations for extended scalar updated Kalman filter

Prediction:		
1	Calculate Jackobian	$\boldsymbol{F}_k = \frac{\partial \boldsymbol{f}}{\partial x}\vert_{\hat{x}_{k-1},u_k}$
2	Predict states	$\hat{x}_k^- = f(\hat{x}_{k-1}, u_k)$
3	Predict error covariance	$\hat{\boldsymbol{P}}_k^- = \boldsymbol{F}_k\hat{\boldsymbol{P}}_{k-1}(\boldsymbol{F}_k)^T + \boldsymbol{Q}$
Correct:		
For $i = 1 : n_z$		
If sensor i has updated		
4	Inovation	$\tilde{y}_k^i = z_k^i - H^{(i,:)}\hat{x}_k^-$
5	Calculate Numerator of Kalman gain	$K_{Num} = \hat{\boldsymbol{P}}_k^-(H^{(i,:)})^T$
6	Calculate Denominator of Kalman gain	$K_{Den} = H^{(i,:)}\,K_{Num} + R^{(i,i)}$
7	Correct state estimate	$\hat{x}_k = \hat{x}_k^- + K_{Num}\,(\tilde{y}_k^i/K_{Den})$
8	Update error covariance	$\hat{\boldsymbol{P}}_k = \hat{\boldsymbol{P}}_k^- - 1/K_{Den}(K_{num}\,K_{num}^T)$
9	Update estimates	$\hat{x}_k^- = \hat{x}_k$, $\hat{\boldsymbol{P}}_k^- = \hat{\boldsymbol{P}}_k$
end		
end		

6.4 Computational Load of Nucleo Board

Although the encoder and IMU may be updated at 1000 Hz implementation on the Nucleo board has proven to be too demanding for the DSP, as it is observed to lose timing. The update frequency of these sensors, as well as the EKF calculations, will therefore have to be reduced to 100 Hz. Doing so results in the loading of the DSP presented in Figure 6.1.

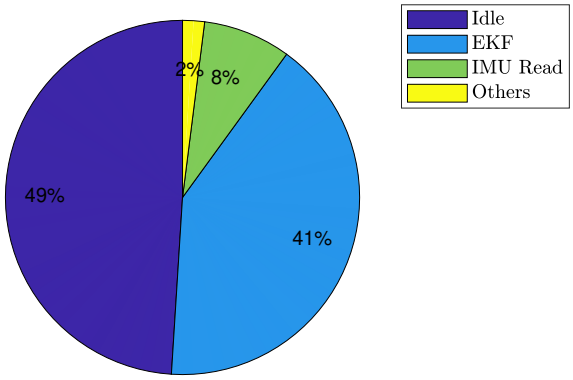


Figure 6.1. Time usage of the DSP during operation at 100 Hz, the category "Others" include the reading of GNSS, encoders as well as some miscellaneous functions

From Figure 6.1 the code on the Nucleo board is seen to run with a time overhead of 49 %. This indicates that some increase in the system frequency may be achieved without compromising real-time operation, but only to a limited extent. The IMU reading function can be seen to occupy 8 % of the DSP time which seems rather excessive but is hard to optimize as it relies on a toolbox provided in the STM32CubeIDE.

6.5 Validation of EKF

In the above sections, a filter has been described and designed. The validation of the designed filter will initially be done on the simulation model as this yields a way of knowing the ground truth of the rover, but will however not include any disturbance to the system states.
For the simulations, the values for the sensor noise will be chosen as seen in Table 6.3 while the process noise uncertainty is chosen as Table 6.4. The method used for the derivation of the $\boldsymbol{R}$ values can be found in Appendix B while the values chosen for $\boldsymbol{Q}$ is the values found in Chapter 7.

Table 6.3. Measurement uncertainty variance

Parameter	Value	Unit
r_{acc}	4e-4	$(m/s^2)^2$
r_{gyro}	6e-3	$(rad/s)^2$
r_{mag}	1e-3	rad^2
r_{GNSS}	2e-3	m^2

Table 6.4. Process noise variance

Parameter	Value	Unit
q_V	1e-3	$(m/s)^2$
q_ω	1e-2	$(rad/s)^2$
q_θ	1e-3	(rad^2)
$q_{\dot{x}}$	1e-2	(m/s^2)
$q_{\dot{y}}$	1e-2	$(m/s)^2$
q_x	1e-5	m^2
q_y	1e-5	m^2
$q_{A_{bias}}$	5e-3	$(m/s^2)^2$
$q_{G_{bias}}$	1e-4	$(rad/s)^2$
$q_{M_{bias}}$	1e-2	rad^2

For the validation of the EKF, the reference for the two motors is chosen as seen in Figure 6.2. This reference is chosen as it will trigger some linear and angular acceleration of the rover and does not require any additional controller to be designed at this stage. For the simulation, all sensor inputs are modeled with an added band-limited white noise therm with a noise power according to their respective measurement uncertainty.

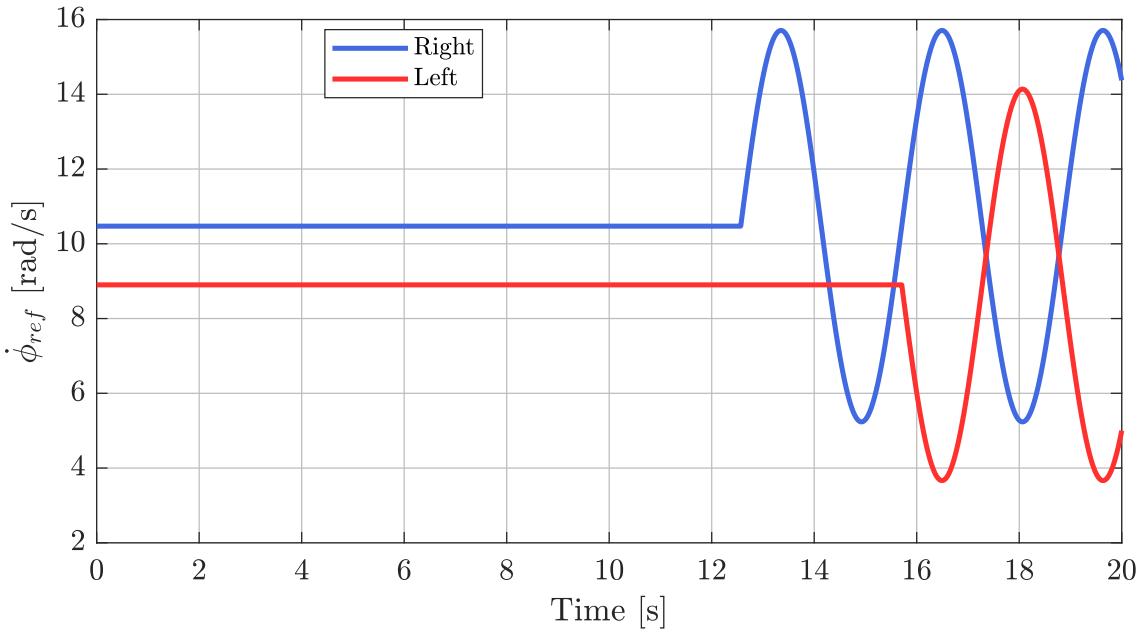


Figure 6.2. Motor references for the validation

The estimated states along with their true states can be seen on Figure 6.3. The sensor bias for the IMU is here initialized as one and then stepped up to a value of two after 10 seconds while the magnetometer bias is stepped from 0.25 to 0.5 rad to test the capabilities of the filter to cope with this. For this simulation, the IMU board is assumed to be placed 20 cm of the center of rotation of the vehicle as this may be assumed to be a worst-case scenario regarding the sensor placement. A noticeable deviation between the actual acceleration bias and the estimated one is

observed as the motor references are changed to sinus curves, which is caused by this sensor misplacement. No additional disturbance has been injected to any of the rover states as no information is available of these.

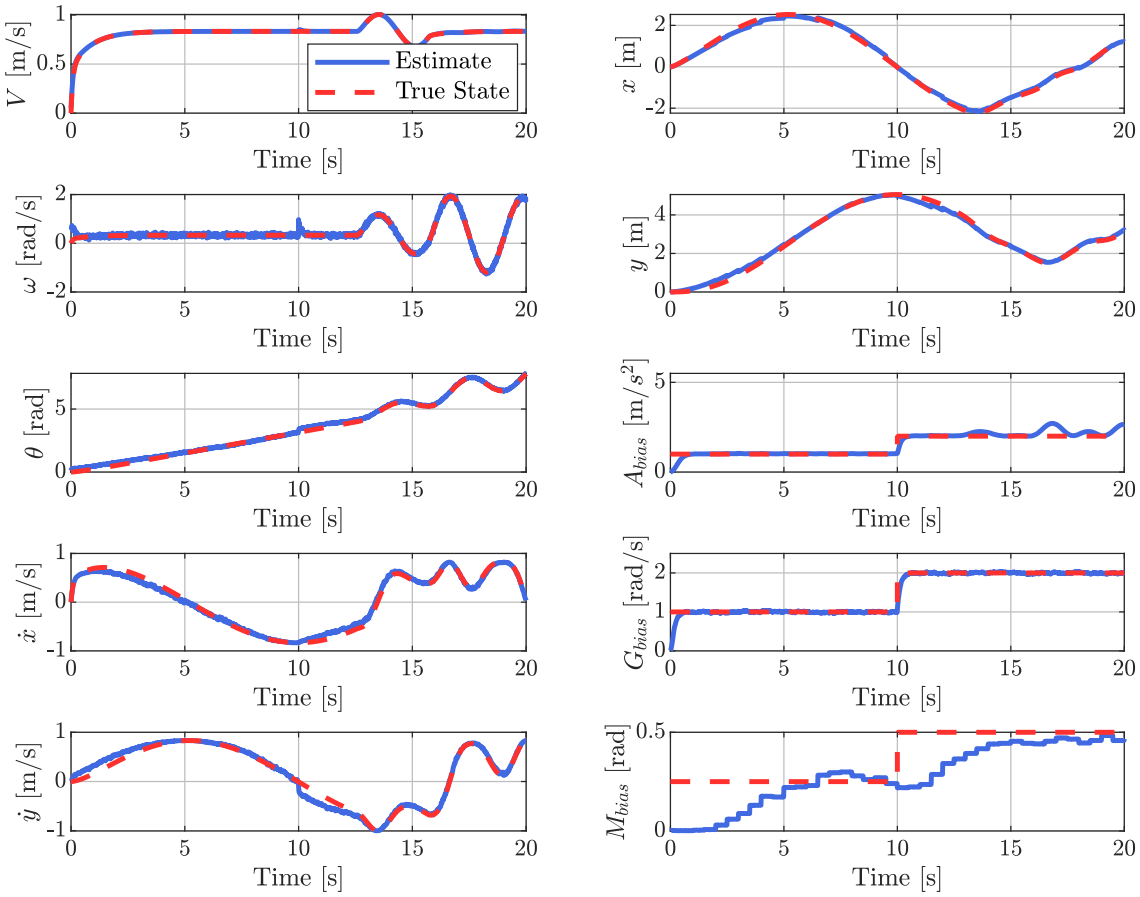


Figure 6.3. Comparison of estimated and true system states

Figure 6.4 illustrates the estimated position of the rover in the inertial frame where the true position, as well as the GNSS data points, is added for illustrative purposes.

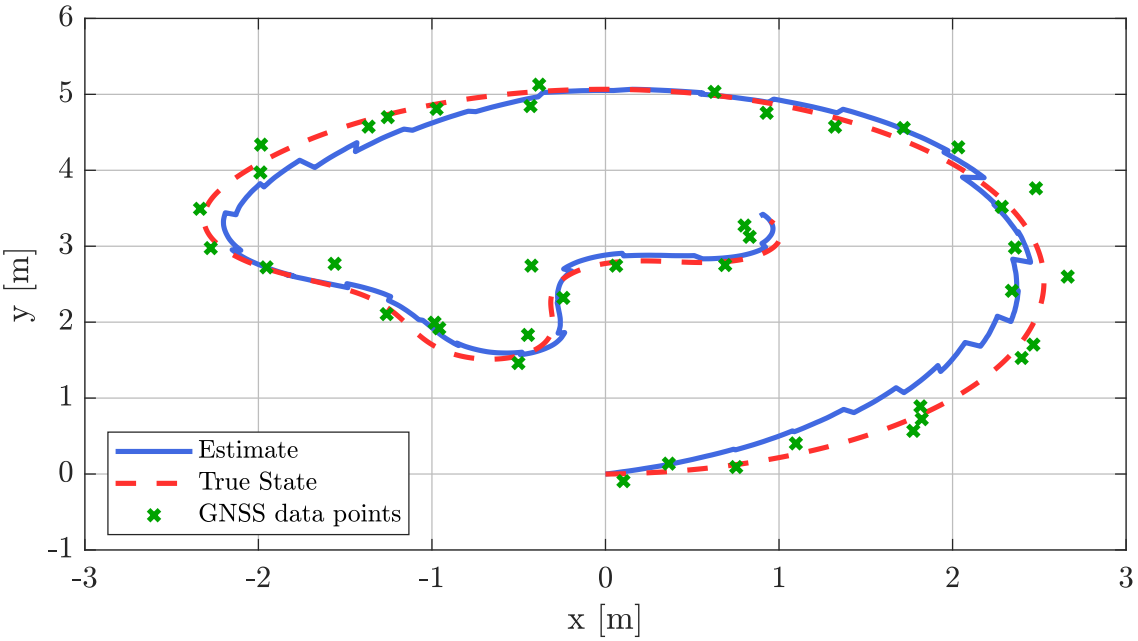


Figure 6.4. Comparison of estimated and actual rover position in the inertial frame

The position estimate is initially seen to drift off due to the magnetometer bias which is poorly estimated at the start but the estimated position is seen to converge towards the true state as more GNSS data is received as this allows for a better magnetometer bias estimation.
With the EKF validated it can be tuned and tested on data from the rover driving around.

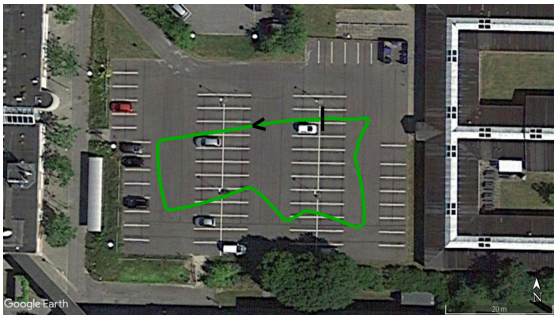
State Estimation from Measured Sensor Data

7

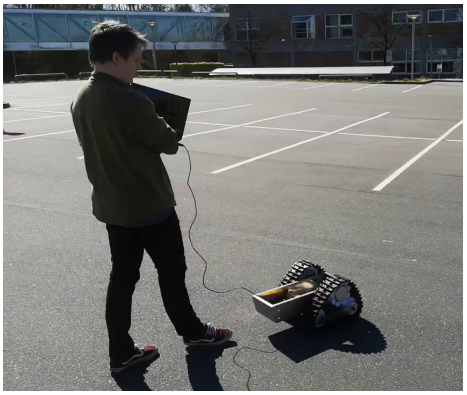
In the following chapter the EKF will be tuned and further validated on measured data from the utilized sensor pack. A general problem for tuning the $\boldsymbol{Q}$ matrix is the fact that the true states of the rover cannot be determined with the available equipment.

7.1 Test Scenario

The test is made by driving around on an empty parking lot in an arbitrary trajectory which is known to have the same starting and ending position and heading as this will allow for an evaluation of the general drift of the filter. The trajectory on the parking lot, as well as a photo of the test conditions, can be seen on Figure 7.1. This parking lot is surrounded by buildings which might affect the GNSS readings. The parking lot is also not level, which will effectively act as a disturbance for the filter as the transformation from the rover body frame to the inertial frame presented in Equations (3.10) and (3.11) will have an error.



(a) Top down view of trajectory on parking lot [6]



(b) Photo of test conditions

Figure 7.1. Test scenario

7.2 Performance Evaluation

For the evaluation the primary focus will be on the x,y (East, North) position of the rover as any small error in either of the other system states will heavily impact the position of the rover. Three different scenarios will be investigated in this section, being a KF only with IMU measurements, a KF with IMU and magnetometer measurements available, and a last scenario GNSS data is fed to the filter in addition to the IMU and magnetometer data.

The first scenario where only IMU measurements are used for correction is shown to highlight the general trend when using a pure dead reckoning approach to estimate the position based on an estimated angular and linear velocity. This approach may be good for a shorter duration of time but will drift with time as any error in either angular or linear velocity will accumulate over time. A plot of the state estimates compared to their measured values is shown in Figure 7.2, where the measured angle from the magnetometer θ, and position from the GNSS x,y are plotted, but not included in the correction step. Similar plots for the scenario including the magnetometer and the GNSS signal can be found in Appendix C.

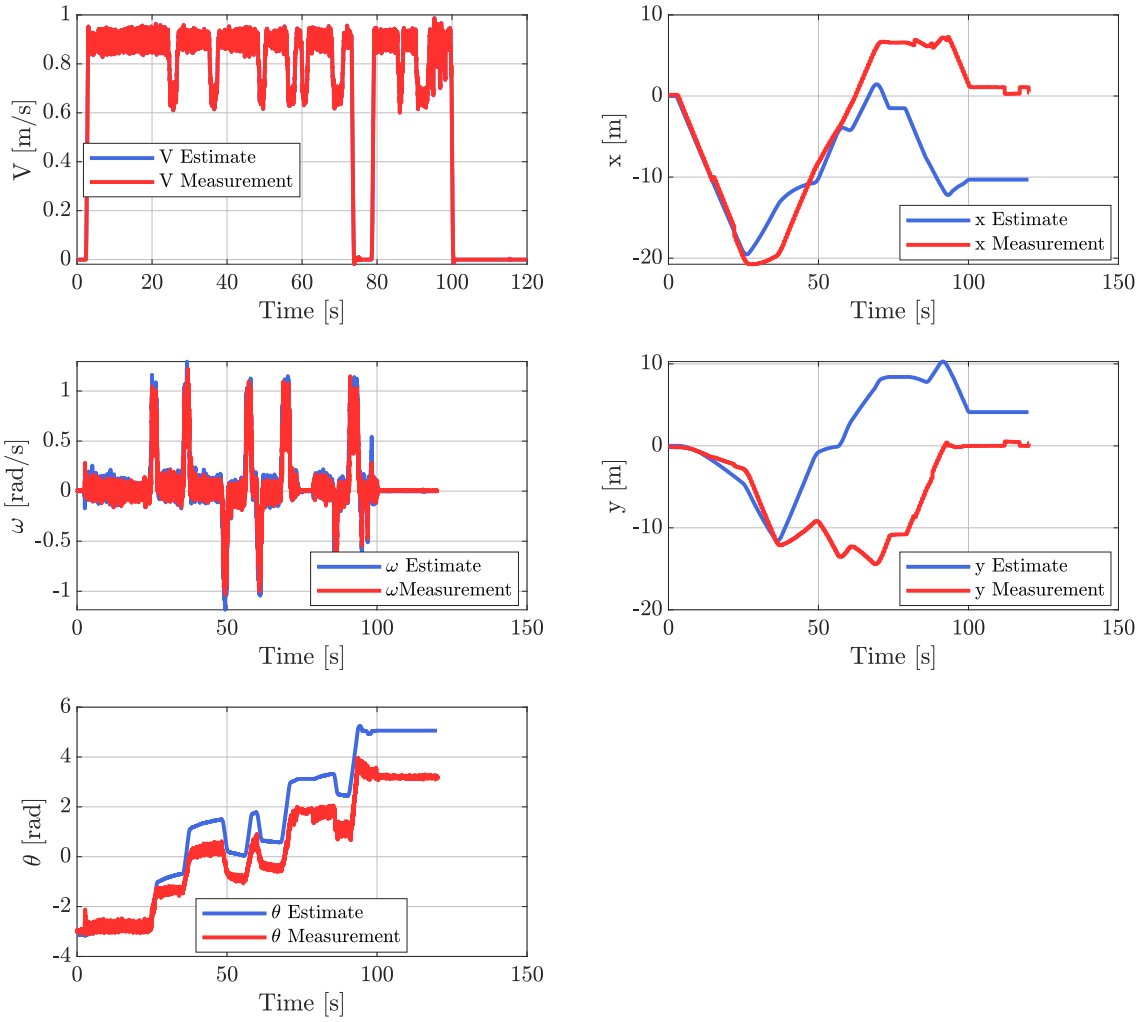


Figure 7.2

A representation of the rover position for all of the above mentioned scenarios can be seen in Figure 7.3. In this figure arrows have been added to indicate direction as well as a full line at approximately [0,0] to indicate the start/stop position. The accuracy of the rover position is observed to increase as the magnetometer measurements are added as this will mean that the heading of the rover can be determined within some span. Any bias of the magnetometer can however still not be determined as this requires measurements from the GNSS. Some drift is however still observed between the GNSS measurements and the estimated position for the second scenario as this still relies on dead reckoning for the estimation of the position.

For the third scenario where all sensor data is used in the EKF. A constant correction from the GNSS eliminates the need for dead reckoning and therefore increases the accuracy of the estimates. Some different artifacts of the GNSS signal can also be observed from the GNSS position data points, being a short data fallout when moving in the western direction (negative x) at the start of the trajectory, and two sudden jumps in the GNSS position when moving north (positive y) towards the ending of the trajectory. The data fallout of the GNSS highlights one of the main benefits of using different sensors for the estimation of the position, as this will allow for state estimations despite missed data. The first jump observed in the GNSS position is caused by the rover GNSS module losing the RTK link to the base GNSS module while the second jump is observed at the point when the GNSS regains RTK information. This will have to be considered as a loss of RTK link will mean that the KF places too much trust in the GNSS data.

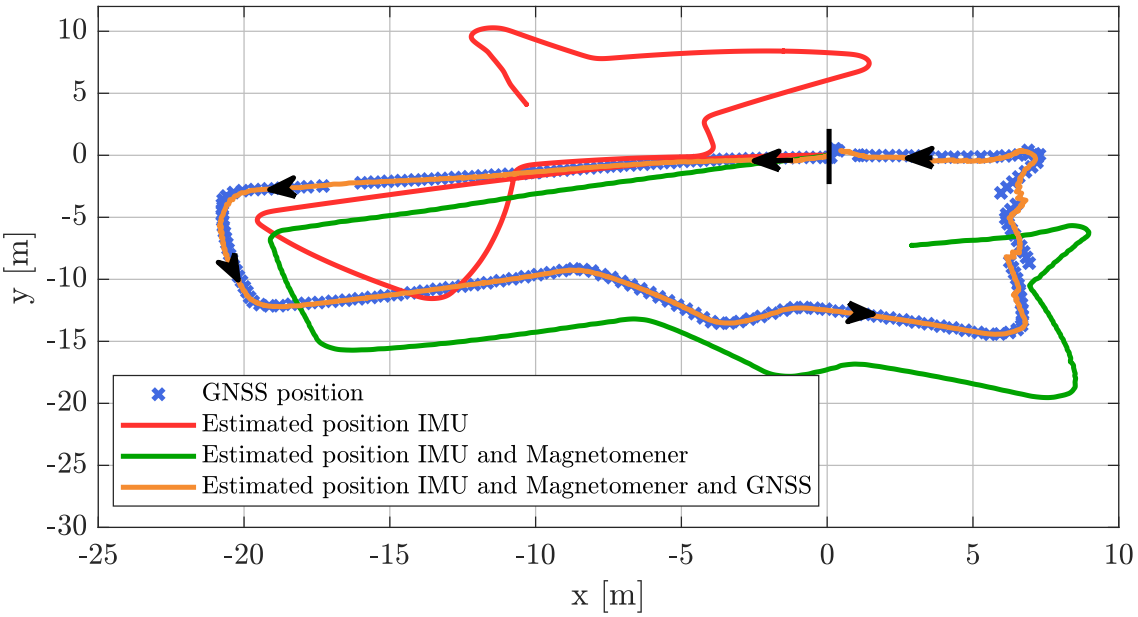


Figure 7.3. Estimated rover position based on different available sensors

7.3 Performance Evaluation During Data Fallout

In the GNSS dataset used in Figure 7.3 A minor data fallout of 2 seconds is observed. To test the navigation system in case of a greater data fallout a test is conducted on the same data set, but where the GNSS signal is assumed lost for 40 seconds represented by the data points marked with red in Figure 7.4.

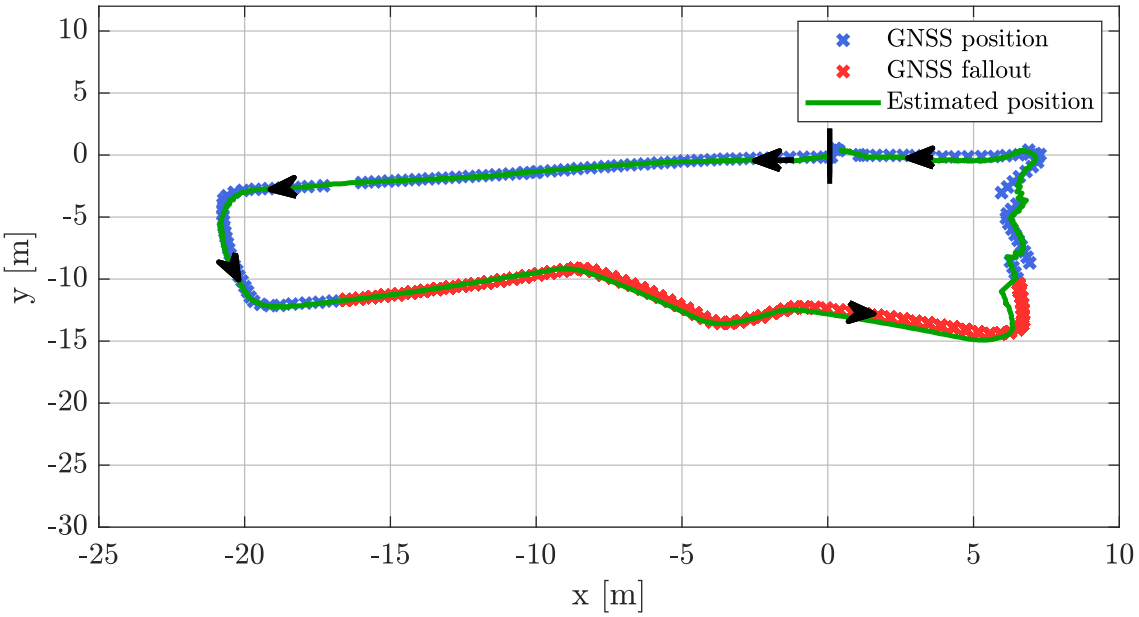


Figure 7.4. Estimated rover position during a data fallout of 40 seconds

The general trend of the position estimates is here observed to follow that of the GNSS well although some deviation is observed towards the end. It should however be pointed out here that the GNSS signal does not represent the ground truth of the rover's position but can be expected to be in the vicinity of it. Another scenario with an 80 seconds data fallout is shown in Appendix C where the GNSS signal is used in the EKF for the first 10 seconds to allow for an estimation of the initial states of the rover where after the data fall out is simulated.

7.4 Discussion

When comparing the measured and estimated states it is clear why a dead reckoning strategy, where states are estimated based on the integral of their derivatives is undesirable. This is highlighted by inspecting Figure 7.2, where both the linear and angular velocity is seen to approximately match their respective measurements while the measured and estimated heading are seen to diverge over time. When adding the magnetometer measurements the accuracy is highly increased as this reduced the extent to which the filter relies on dead reckoning.

The tests performed with the filter have in general not been ideal as they have all

been made in an urban area with tall surrounding buildings which may distort the GNSS signals. This may also have been the cause of both the data fallout observed initially in the trajectory and of the RTK compensation loss observed towards the end. The lost RTK connection may be compensated by changing the measurement uncertainty variance for the GNSS module in the EKF as this will make the filter rely more on the other sensors.

Control 8

Two different control structures will be proposed for the rover. The different control structures will be a heading controller and an inertial frame controller. The reason for proposing both structures is that the heading controller will be feasible for use with CV, while the inertial frame control structure will be feasible if the objective is to navigate the rover to a given position in the inertial frame.

The kinematic transfer function $\boldsymbol{G_{kin}}$ can be defined based on the rotational velocity transfer function $\boldsymbol{G_{\dot{\phi}}}$ which is derived by numerically closing the feedback loop of the motor dynamics and the PI-controller of Figure 4.1 with the parameters found in Table 4.1. $\boldsymbol{G_{kin}}$ is calculated as $\boldsymbol{G_{kin}} = \boldsymbol{M^{-1}}\,\boldsymbol{G_{\dot{\phi}}}\,\boldsymbol{M}$ where the transfer function $\boldsymbol{G_{kin}}$ takes in a linear and angular velocity reference and outputs the actual linear and angular velocity, while $\boldsymbol{M}$ represents the kinematic relation between $\dot{\phi}$ and the rotational/linear velocity. The frequency response of this transfer function is depicted on Fig. 8.1. It is observed that the gain of the off-diagonals of $\boldsymbol{G_{kin}}$ is negligible compared to the diagonals, which indicates that a controller for V_{ref} and ω_{ref} may be designed separately as they can be assumed to be decoupled.

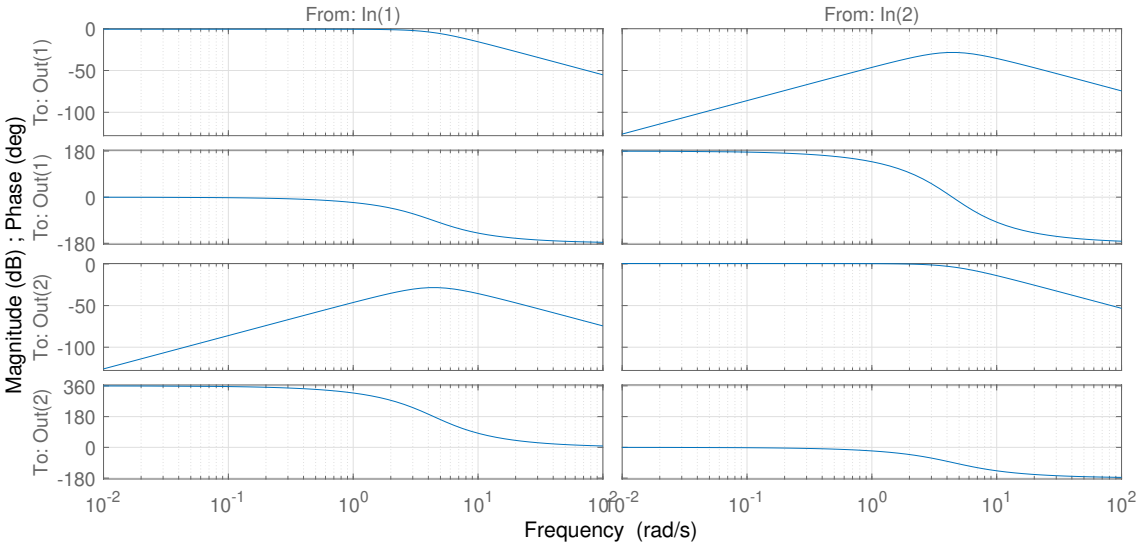


Figure 8.1. Bode plot of the rover kinematic transfer function G_{kin}

8.1 Alignment Controller

The purpose of the alignment controller will be to control the heading of the rover while a constant velocity command will be used. The velocity may then be chosen

based on different factors such as e.g. the required precision of the rover alignment. No additional feedback is here added to the velocity command which effectively means that the rover velocity will purely be based on the calculated velocity from the encoders.

The heading of the rover will be handled by a proportional controller as depicted on Figure 8.2.

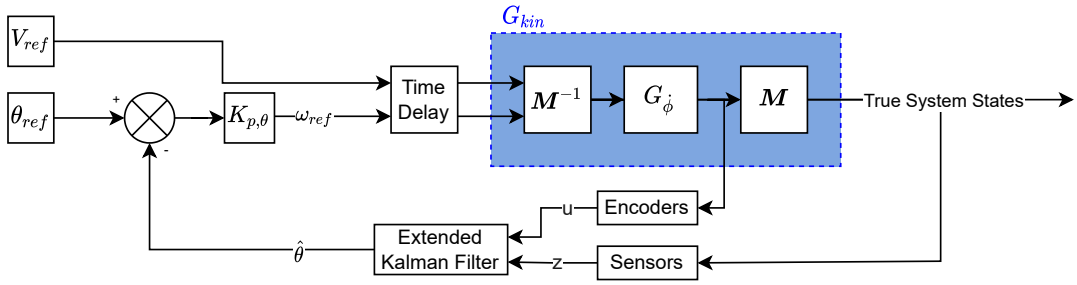


Figure 8.2. Alignment control structure of the rover

A proportional controller is chosen as it is the simplest possible controller and will still yield zero steady-state error for a step input in the rover heading due to the free integrator of the transfer function from $\frac{\theta}{\omega}$.

The time delay observed in Section 4.2 will for the tuning of the controller be approximated as a low pass filter with a time constant equivalent to the length of the time delay. The impact on the frequency response of this assumption compared with a true delay and the non-delayed system can be seen on Fig. 8.3.

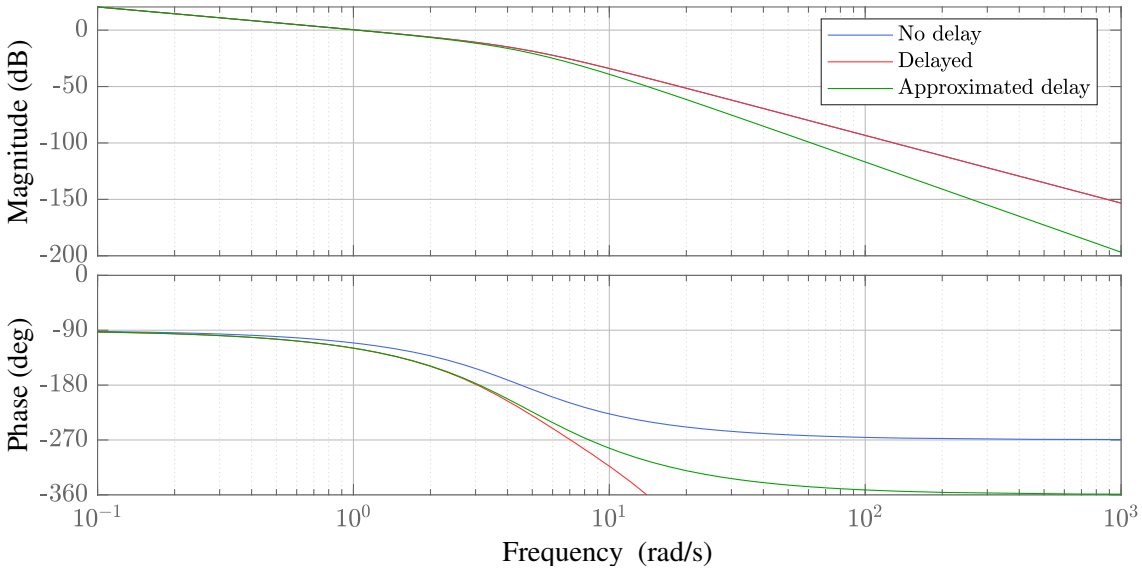


Figure 8.3. Bode plot of $G_{kin}(2,2) \cdot \frac{1}{s}$ with and without time delay as well as with an approximation of the time delay

The approximated transfer function is seen to accurately model the behavior of the true system at frequencies below 4 rad/s where after both phase and magnitude are

seen to diverge. This discrepancy is however expected to be negligible as the system gain at this frequency is low.

The root loci of the open-loop transfer function with the approximated time delay for the alignment controller can be seen on Figure 8.4.

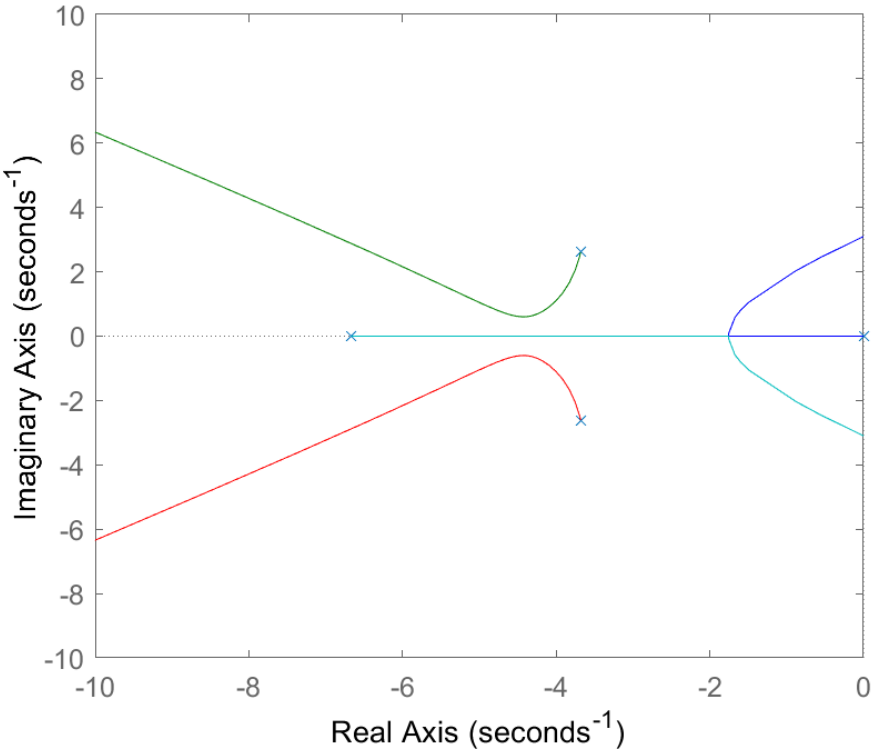


Figure 8.4. Root loci for $\frac{\theta}{\omega_{ref}}$ with the approximated time delay

The controller gain $K_{p,\omega}$ will here be chosen to be 0.94 as this places the dominant pole pair of the combined closed-loop transfer function such that a damping of 0.7 is achieved. This is chosen as it will yield a compromise between fast response time and minimizing overshoot. This results in a bandwidth of 1.3 rad/s for the combined alignment controller and system dynamics. The step response of this can be seen on Figure 8.5.

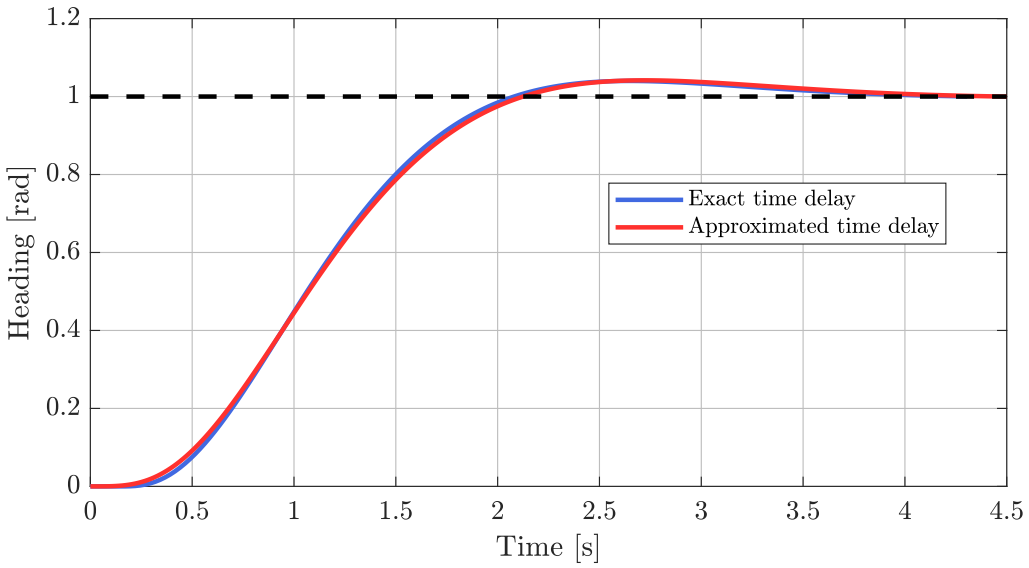


Figure 8.5. Step response of the closed loop time delayed transfer function, the transfer function with the approximated time delay is added for reference

8.1.1 Controller Evaluation

To evaluate the controller performance, on the model of the system with the designed EKF, a constant reference of 1 m/s is given to the velocity, while the heading is stepped from $0 - \pi$ rad after 3 seconds. The result of this is shown on Figure 8.6.

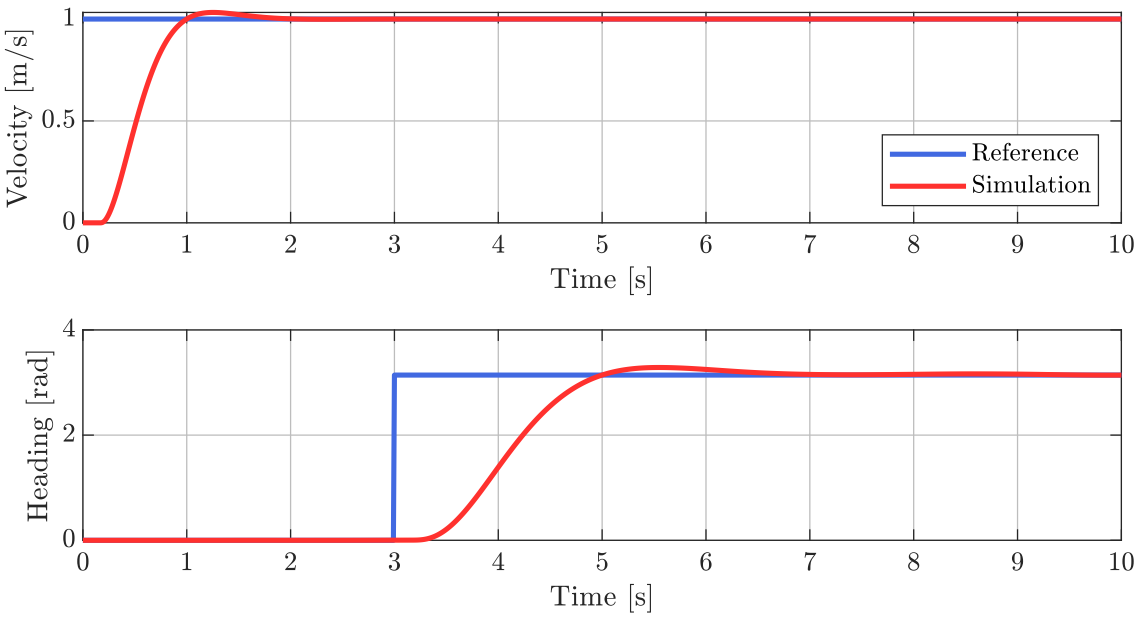


Figure 8.6. Heading of the rover compared to the reference

As there is no feedback control designed for the velocity of the rover, the desired velocity may differ from the actual velocity. The heading controller is observed to behave as anticipated in the design of the controller. It is also observed that the

slight coupling between velocity and rotational velocity, which has been assumed decoupled in the controller design, has no influence on the response of the system.

8.2 Inertial Frame Controller

The alignment controller ensures the correct heading but may offset in the inertial frame as it only ensures that the rover will drive parallel to some line. This section will therefore focus on a way of expanding the control structure such that the rover may navigate to a given reference point.

The inertial controller will however still utilize the above designed heading controller but will in addition to this have a distance control feedback loop designed to control the motor velocity reference V_{ref}.

This distance controller will be designed similarly to the heading controller and will also be using a proportional controller as this will result in a type one system, yielding zero steady-state error to a step input. This controller will generate an error signal based on the body frame position S and a reference for this S_{ref}. The rover distance traveled S could potentially be estimated by the EKF but will not be done as another problem with this control structure is the fact that S_{ref} is meaningless in the inertial frame. This problem will be handled later in this section and S will therefore be assumed known.

The root loci of the system can be sen on Figure 8.7. The S controller will for this case be tuned such that the dominant pole pair of the system is critically damped, to prevent overshoot. Doing so results in a gain for $K_{p,s}$ of 0.74.

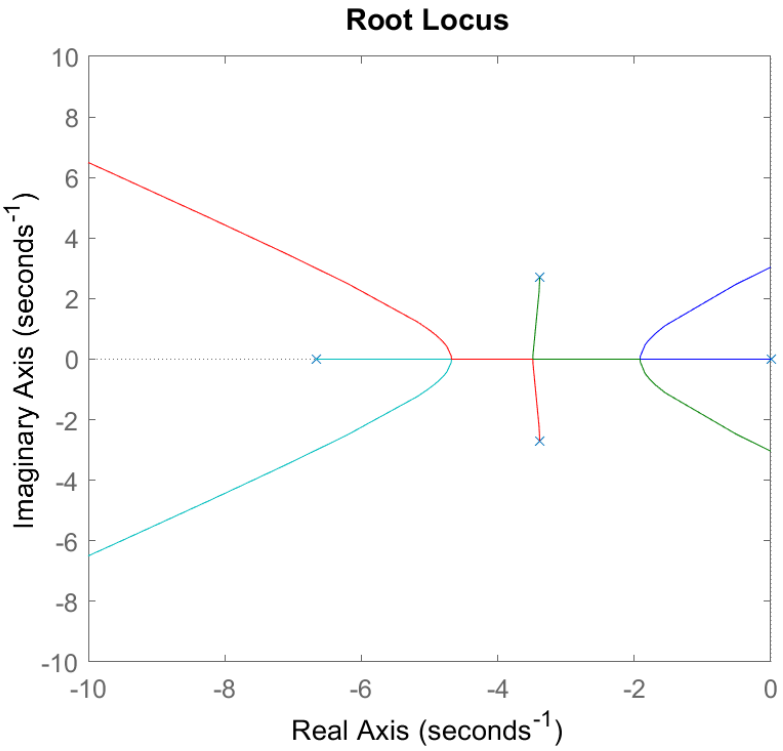


Figure 8.7. Root loci for $\frac{S}{V_{ref}}$ with the approximated time delay

Closing the feedback loop with this controller gain yields the approximated and exact step response seen on Figure 8.8, where a closed-loop system bandwidth of 1.9 rad/s is achieved.

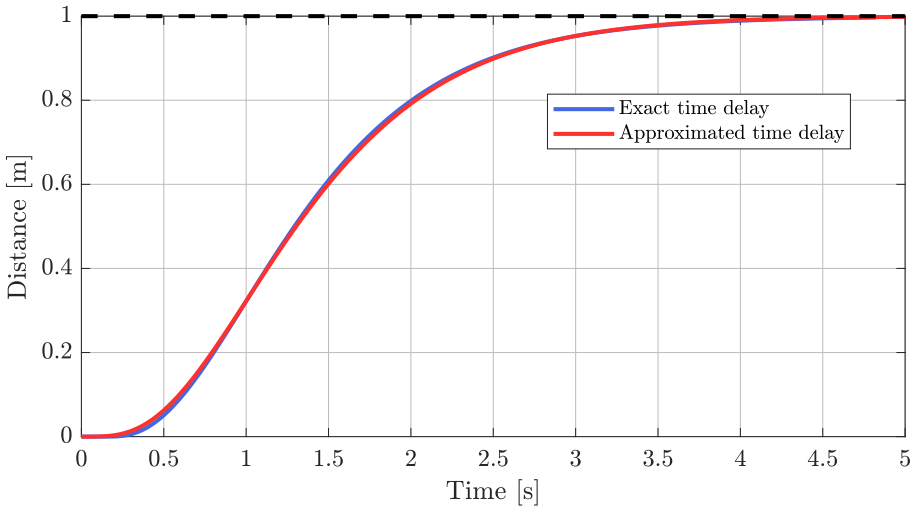


Figure 8.8. Step response of the closed loop time delayed transfer function, the transfer function with the approximated time delay is added for reference

With both the heading and distance controllers of the system designed, a reference for both controllers will have to be determined such that the distance Δl from the

rover to the goal position (x_{ref}, y_{ref}) will approach zero. The robot position control proposed in this section will be based upon the methods presented in [7].

To ensure that $\Delta l \rightarrow 0$ two new variables Δd and ψ will have to be defined where Δd represents the distance to the point d which is defined as the point with the shortest distance to the reference point, along with the rover's current orientation while ψ is the angle between the rover position and the reference point as illustrated on Figure 8.9.

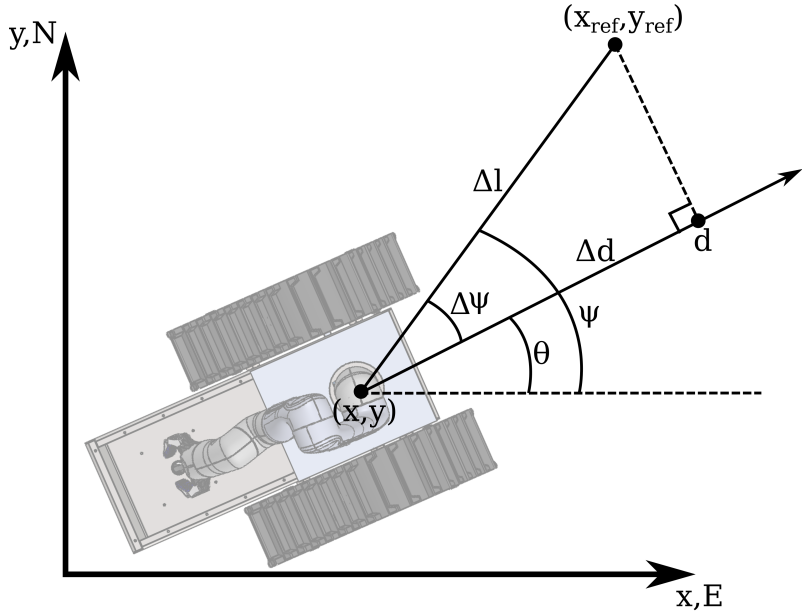


Figure 8.9

$\Delta\psi$ may then be defined as the difference between ψ and θ as: $\Delta\psi = \psi - \theta$. Based on trigonometric relations it can be seen that:

$$\Delta l = \frac{\Delta d}{cos(\Delta\psi)} \tag{8.1}$$

This effectively means that if a controller can be designed to drive both Δd and $\Delta\psi$ to zero then Δl will consequently approach zero. To ensure this we will define the error signals, fed into the heading and distance controllers, as $e_\theta = \Delta\psi$ and $e_s = \Delta d$ respectively. By defining $\theta_{ref} = \psi$ this can be calculated as:

$$\theta_{ref} = \text{atan2}(\Delta y, \Delta x) \tag{8.2}$$

Where Δy and Δx represent the distance from the current to the desired location in the y and x direction. This means that the error signal for the heading controller will be: $e_\theta = \theta_{ref} - \theta$ which is what the control law is designed to handle.

By defining the point d along the orientation of the rover from Figure 8.9 as being the desired reference point, the error is seen to be: $e_s = S_{ref} - S = \Delta d$ which can be calculated, based on Equation (8.1) as:

$$e_s = \Delta l \, cos(\Delta\psi) = \sqrt{\Delta x^2 + \Delta y^2} \cdot cos(\Delta\psi) \tag{8.3}$$

A general problem for non-holonomic constraint systems is that they cannot be completely stabilized by any continuous control law [7]. A circle of radius ϵ will therefore have to be defined around the reference point (x_{ref}, y_{ref}) such that when $\Delta l \leq \epsilon$ the controller errors e_θ and e_s will be set to zero as the reference angle θ_{ref} will be undefined in the case where the actual position is equivalent to the reference point, while any estimation noise in (x, y) will make the heading controller highly oscillatory when close to (x_{ref}, y_{ref}). The rover will for the control strategy of this project be constrained to forward movement, but the reference generation will work for movement in either direction. A block diagram of the designed control strategy, as well as the designed reference generation, is sen on Figure 8.10

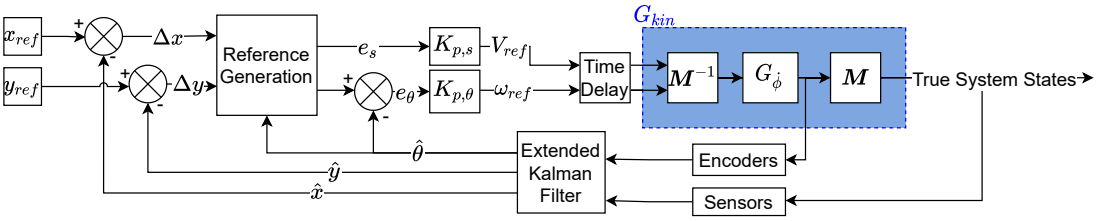


Figure 8.10. Inertial Control structure of the rover

8.2.1 Controller Evaluation

The above designed controller will for the project be evaluated in two different manners. In the first scenario, a simple reference is generated where four different way-points are given to the controller with a fixed time interval in between. For this scenario, the controller will be allowed to reach its reference and will therefore come to a full stop during the validation. For the second test scenario, a reference trajectory will be constructed by calculating way-points along the desired path of the rover. A new way-point will then be fed to the rover at each time step without the rover reaching any other way-points than the last. This effectively means that the controller in the first scenario is free in its choice of the trajectory to a given reference point while the second scenario is more constrained as the rover constantly will be trailing the reference point in a follow the carrot manner.

The generated error signals e_θ as well as the distance to the desired location Δl during the first scenario can be seen on Figure 8.11. A new way-point is here fed to the controller at an interval of 15 seconds to allow the controller to reach the desired location. The controllers are observed to drive the distance to the way-point Δl towards the chosen error tolerance ϵ which for this case is chosen to be 20 cm.

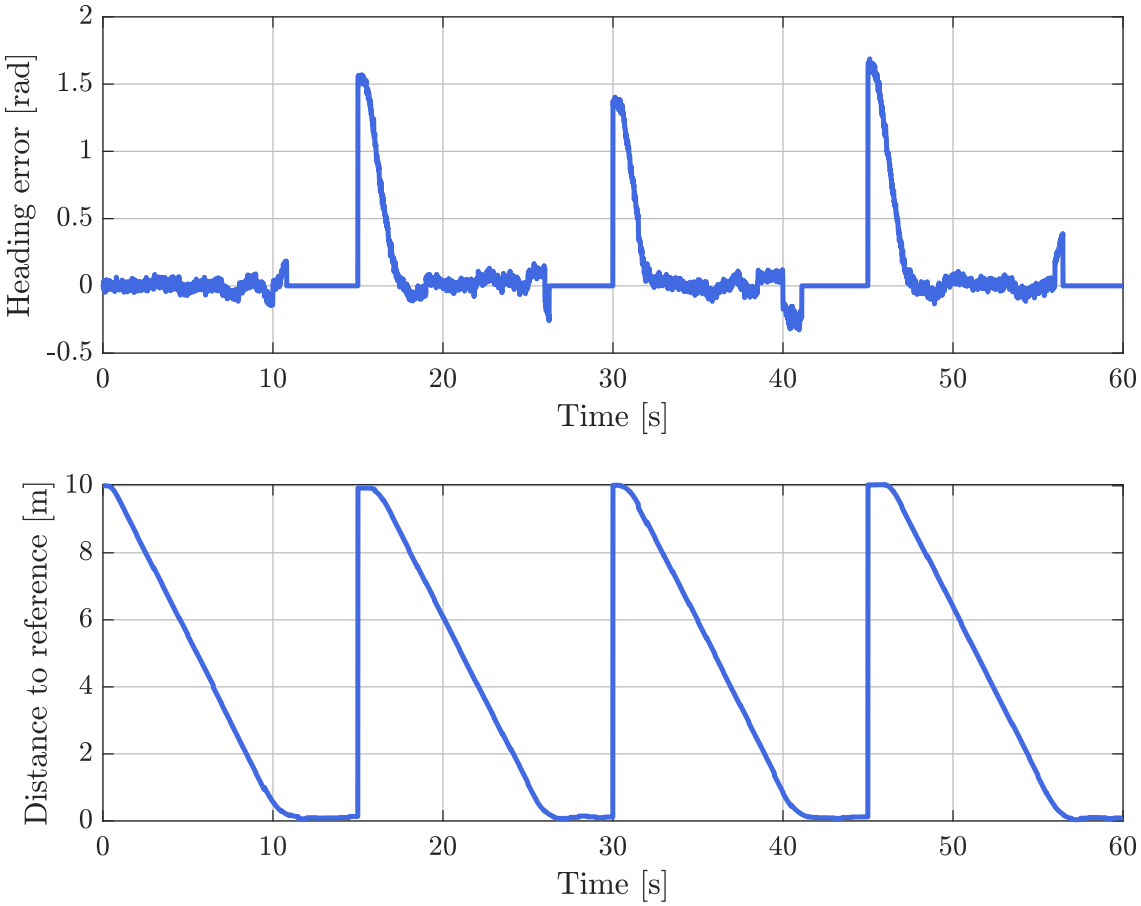


Figure 8.11. Angular error and length to the desired reference point

A visualization of the route taken by the rover can be seen on Figure 8.12. The trajectory will start and stop in (0,0) while the rover is driving in an anti-clockwise direction. The rover is here only given the objective of reaching the separate waypoints and will therefore not follow any predefined trajectory to reach these.

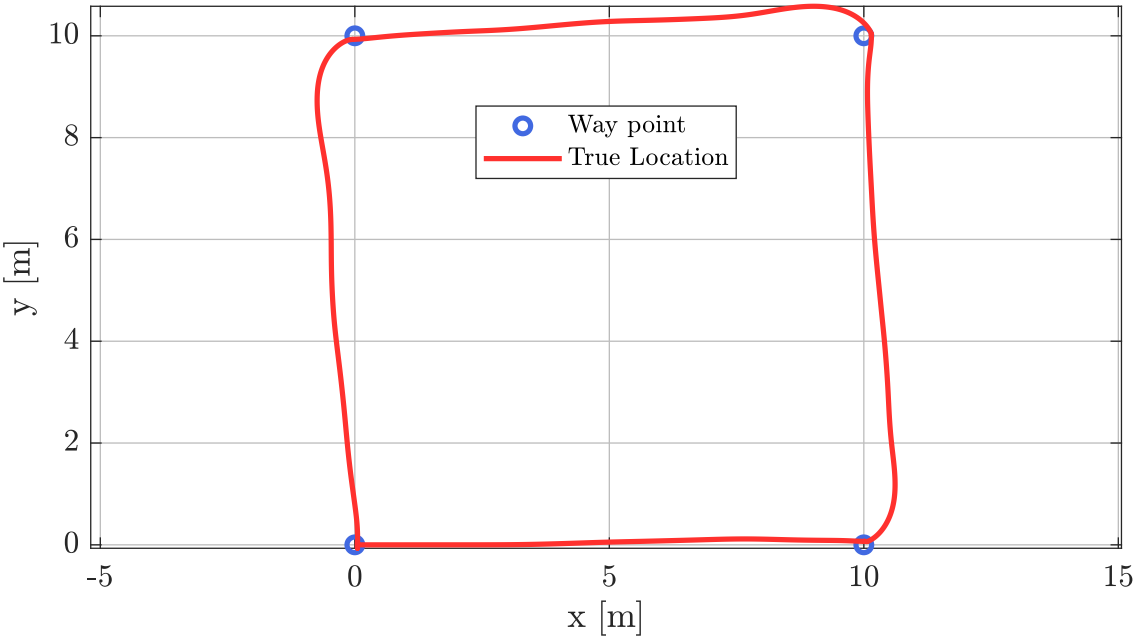


Figure 8.12. Comparison of reference trajectory and true rover location

To test the tracing capabilities of the inertial controller another scenario is constructed where an array of way-points is fed into the controller as this will allow not only to define the end goal of the rover but also the path that the rover should follow to reach the point. The simulated rover position compared to the reference trajectory can be seen on Figure 8.13. This trajectory has been designed to drive with a linear velocity of 0.9 m/s for the straight sections and 0.45 m/s for the curved parts.

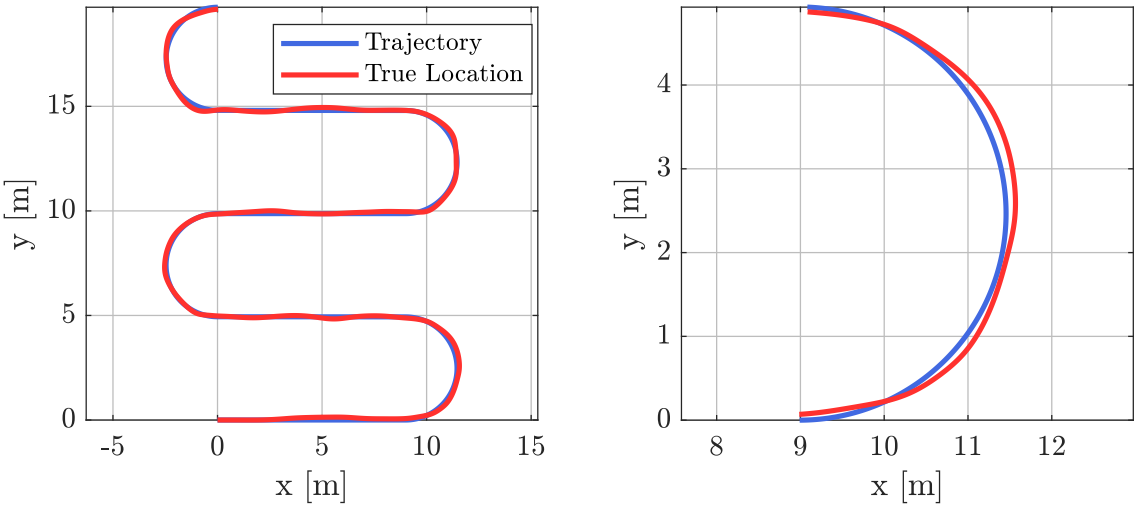


Figure 8.13. Comparison of reference trajectory and true rover location. Left graph shows the full trajectory while the right plot is a zoom of the first turn

During the entire course of the trajectory, the maximum recorded deviation from the intended trajectory is 18 cm while the root mean square error (RMSE) during the

trajectory is 7 cm. Note that these errors are not calculated as the difference between the position and the way-point, but as the difference from the actual position to the closest point on the trajectory. These results are dependent on the velocity during the trajectory and if the linear velocity is increased by a factor of two for the complete trajectory the RMSE and max error are respectively increased to 15 cm and 35 cm.

Discussion 9

In the following chapter, some additional considerations of parts of project content will be discussed.

When driving the rover around, it has been observed that the RTK signal may drop out from time to time. This has a large influence on the estimated states in the EKF, as a high amount of confidence is put into the GNSS readings, which are less accurate without the RTK. This can potentially be mitigated by changing the $\boldsymbol{R}$ matrix whenever an RTK fallout is detected.

The EKF has been designed based on an assumption of planer movement. With the sensors used this basically means that additional information of the rovers states is measured but not used. Using all of the available sensor information may help to get a better estimation of the pose of the rover. This could potentially prove beneficial when driving on rougher terrain where the assumption of the rover driving on a level surface is challenged. This could however also prove to be even harder computationally, as the EKF at the current state already consumes 41 % of the time available on the Nucleo. Implementing more states to the filter could potentially require the filter to run even slower than it already does, to maintain timing.
At the moment the filter is slower than the fastest sensor in the system. The filter runs at 100 Hz while the IMU has the potential of producing data at 1000 Hz, as the fastest sensor. So running the filter at a higher frequency will also be beneficial, though this will require a more efficient way of calculating the filter.

The proposed inertial frame controller of the project is designed with the objective of controlling the rover to some desired location. It may however be feasible to expand this control structure in order not only to control the end position but also the the heading of the rover at the desired destination. This control strategy has been investigated in [7]. Doing so will yield an extra degree of freedom when controlling the rover to reach a certain way-point as the rover in addition to reaching a specific point will do so from a well defined heading.

The heading controller is beneficial when it is combined with CV. The CV algorithm intended for the report has been a line detection algorithm to find rows in a field, as described in Appendix A. This line detection should return an error value between the heading of the rover, and the desired heading in the body frame. This can be projected to the inertial frame, and act as the reference input to the heading controller. This approach can allow the rover to navigate within the crop rows on a field.

The proposed structure of the thesis does require a lot of different boards for the general control of the rover. This has meant that a lot of project time has been consumed by the general programming of these boards as well as on establishing the communication between them. As a consequence of this, it has not been possible, with the given time frame, to implement the proposed control structures on the rover in order to validate the system performance on the physical setup.

A general concern when using the NJN for the implementation of the control structure is that ROS does not guarantee real time operation. This is however only expected to have a minor impact on the system performance due to the low bandwidth of the system. The achieved system bandwidth of the alignment and position controllers are respectively 0.2 Hz and 0.3 Hz. A general rule of thumb is that a discrete controller will approximately behave if it is continuous if it is discretized at a rate five to ten times faster than the desired system bandwidth. This means that an update rate of 3 Hz will be sufficient to ensure that the no significant changes to the system response, which is expected to be well within the capabilities of the NJN.

10 Conclusion

In the report, an extended Kalman filter has been designed to fuse the incoming data streams from the wheel encoders, an IMU, a magnetometer, as well as a GNSS module. Using a large sensor pack as proposed in the project will allow for the estimation of biases in the different sensors, and will in addition to this give the system some redundancy. An example of this system redundancy is seen in Figure 7.4 where a GNSS data fallout has been simulated on recorded sensor data. The filter is in this case observed to closely follow the GNSS data points despite the missing data over a period of 40 seconds. The overall accuracy of the filter can however not be evaluated in the report as this will require knowledge of the true system states.

A feedback controller has also been designed for the rover which utilized the state estimates from the EKF to navigate the rover to a given reference point in the inertial reference frame. During the evaluation of the inertial frame controller on the defined trajectory, an RMSE of 7 cm has been achieved with a maximum deviation from the trajectory of 18 cm. The precision of the controller is however observed to be highly dependent on the velocity as a higher velocity generally will require a higher controller bandwidth.

Future Work 11

Implementation of Camera Vision

As described earlier in the project the rover has initially been intended to navigate through field rows based on an error angle generated from a CV algorithm. Another interesting aspect to investigate would have been Visual Odometry (VO) which may be expected to further increase the accuracy of the EKF. A benefit of VO is that it, like the IMU and magnetometer, does not rely on an external signal and may therefore increase the accuracy of the rover when operating either indoors or in an urban scenario where GNSS may be unavailable.

Power Management and Distribution

The rover platform described in the project is seen to have a large number of different connected components. It would therefore be feasible to create a custom Printed Circuit Board (PCB) to handle all of these different interconnections as this would reduce the chance of any connections coming loose. In addition to distributing power and data connections between the boards, the PCB should also have charging capabilities for the main and auxiliary battery such that a rover may operate and charge itself without any human intervention. The board should also include some battery management system to prevent battery abuse such as deep discharge of the batteries as this will permanently damage lithium batteries.

Live Data Visualization and Communication

Another feature that would increase the overall quality of the rover would be to have full control of the rover from a remote station such that e.g. manual override and changing of the control mode could be accessible online. With online capabilities on the rover, it would also be possible to live stream data such as rover position, battery level from the BMS, and any potential error codes. This would especially be beneficial as the rover is intended to operate in larger fleets where it may be hard to keep track of every individual rover.

Bibliography

[1] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar. *Robot Modeling And Control*. John Wiley & Sons, Ltd, 2nd edition, 2020.

[2] John P. Snyder. *Map projections: A working manual*, 1987.

[3] Charles L. Philips & John M. Parr. *Feedback Control System*. Pearson, 5th edition, 2011.

[4] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005. ISBN 0262201623 9780262201629.

[5] Tucker McClure. *How Kalman Filters Work, Part 2*, 2016. URL `https://www.anuncommonlab.com/articles/how-kalman-filters-work/part2.html`. Visited 19/05/2022.

[6] Google. *Fredrik Bajers Vej 7G, 9220 Aalborg*, 2021. URL `https://earth.google.com/web/@57.01393348,9.9885678,14.88211059a,120.17636259d,35y,0h,0t,0r@`. Visited 30/05/2022.

[7] Frederico C. Vieira., Adelardo A. D. Medeiros., Pablo J. Alsina., and Antônio P. Araújo Jr.. Position and orientation control of a two-wheeled differentially driven nonholonomic mobile robot. In *Proceedings of the First International Conference on Informatics in Control, Automation and Robotics - Volume 2: ICINCO,*, pages 256–262. INSTICC, SciTePress, 2004. ISBN 972-8865-12-0. doi: 10.5220/0001138702560262.

Appendices

Component Description A

IMU and Magnetometer

The IMU used for this project is an X-NUCLEO-IKS01A3, which is a motion MEMS expansion board mounted on a NUCLEO-F401RE development board both manufactured by STMicroelectronics. The expansion board consists of multiple sensors, including a 6DoF accelerometer(3D) and gyroscope(3D), as well as a 3D magnetometer. The expansion board communicates with the development board with I2C. A benefit of the IMU and magnetometer is that they may be updated at 1000 Hz and 100 Hz respectively which means that faster system dynamics can be determined. Another benefit of the IMU and magnetometer is the fact that they are very reliable as they do not rely on external factors such as GNSS which means that they may not be jammed and will also work in an indoor environment.

GNSS

The GNSS used in this project is the NEO-M8P-2 module mounted on the C94-M8P-E application board from U-Blox. With two of the application boards, it is possible to use one as a base station while the other is attached to the rover, enabling the use of Real-Time Kinematic(RTK) and allowing for a more precise rover position measurements. The rover and base station communicate with a radio link over a 433Mhz frequency. Sending data from the rover GNSS module to the mainboard is done over UART using the UBX protocol from U-blox. The GNSS used in this project can update with a frequency of 2 Hz which is slow compared to the IMU but comes with the benefit of being an absolute measurement which effectively means that the position may be determined directly and will therefore not rely on dead reckoning.

Motor/Encoder and controller

The motors and controller are from ZLTECH. The motors are the model ZLLG65ASM250-4096 V2.0, which is a 350W motor with a rated torque of 8Nm and a maximum speed of 300RPM with a build-in magnetic incremental encoder.
The motor driver is the ZLAC8015D. This is a dual servo motor driver, making it possible to supply/control both motors with one motor drive. The drive has a built-in torque and speed control. So the input to the motor drive is a speed reference. The motor drive communicates with RS485 protocol.

Computer

The computer in the system is the Nvidia Jetson Nano (NJN), running Robot Operating System (ROS). The use of the NJN is to run the control for the rover, as well as interpret the camera feed, and extract the necessary lines to use in the control.

Camera Vision

A camera may be attached to the rover to run a camera vision program for line extraction of the crop rows in the field. This camera vision software is not something that will be evaluated in this report. It will be considered as a black box, with an output error angle between the center of the rows and the heading of the rover, as illustrated on Figure A.1

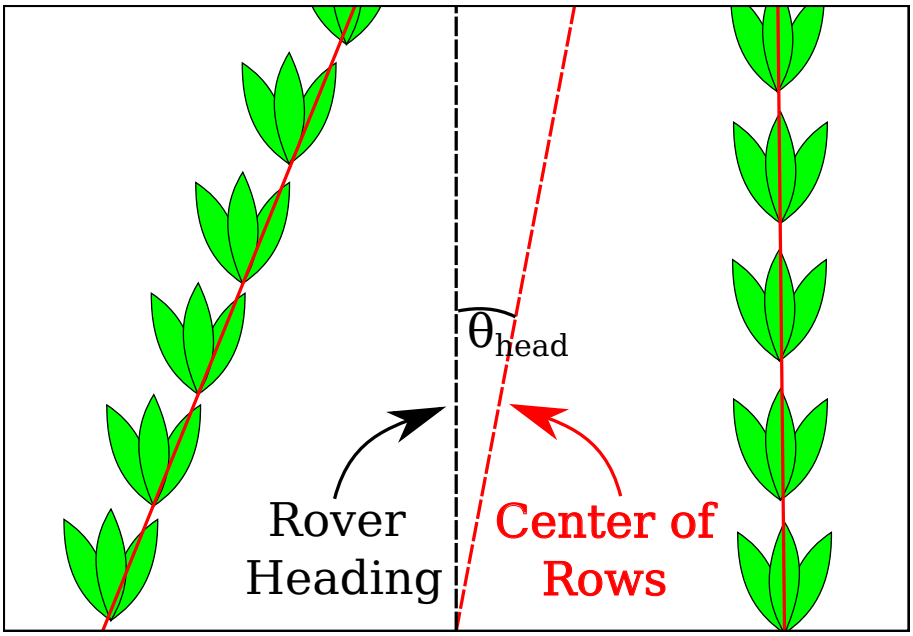


Figure A.1. A sketch of the calculation of the CV error angle, from the camera's point of view

Determination of Measurement Uncertainty

B

The measurement uncertainty of the different sensors will be determined in the following section, based on sampled data during different scenarios. The measured variances of the signals can be found in Appendix B.

Gyroscope

The measurement uncertainty of the gyroscope is determined by commanding the motor controller to drive each wheel of the rover in opposite direction at a constant speed. This results in a rotation of the rover as seen on Figure B.1. To calculate the variance of the signal the transient phase from zero to two seconds has been removed and may be calculated as:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} z_i \tag{B.1}$$

$$r_{gyro} = \frac{1}{n-1}\sum_{i=1}^{n}(z_i - \mu)^2 \tag{B.2}$$

where μ is the average value of the signal, z_i is i'th measurement and n is the total number of observations.

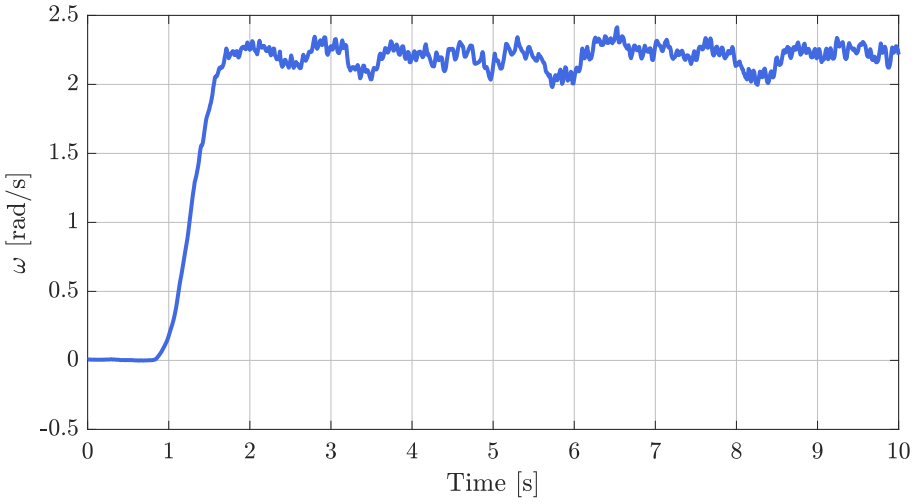


Figure B.1. Angular velocity of rover measured from gyroscope

This approach will include cross-correlation terms between the motor speeds and the gyroscope measurements, but will still be used as measurement uncertainty cross-correlations will have to be neglected to use the scalar implementation of the EKF.

Accelerometer

The accelerometer measurements will for the implementation be integrated to yield the velocity of the rover. For this test, both motors are commanded a constant similar velocity. A steady-state accelerometer bias compensation is here made as any bias will heavily impact the calculated velocity. This may be expected to be valid for a short duration as the bias is expected to drift slowly. The variance of the velocity is calculated as above and the collected data can be seen on Fig. B.2.

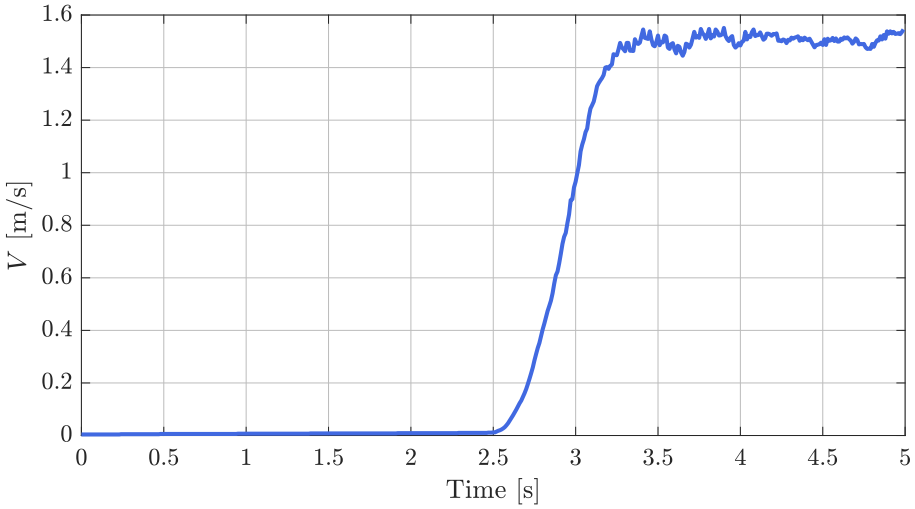


Figure B.2. linear velocity of rover measured from accelerometer

The same argument about the cross-correlation between motor velocity and velocity measurement can be made here as above. An alternative approach would be to measure the uncertainty with no rover movement by is not chosen as this may result in the EKF having too much confidence in the measurements.

Magnetometer

For the measurement of the heading uncertainty, the rover will be stationary while data is collected as small control/actuation errors for the two motors are observed to cause the trajectory of the rover to curve. The obtained measurements can be seen in Figure B.3.

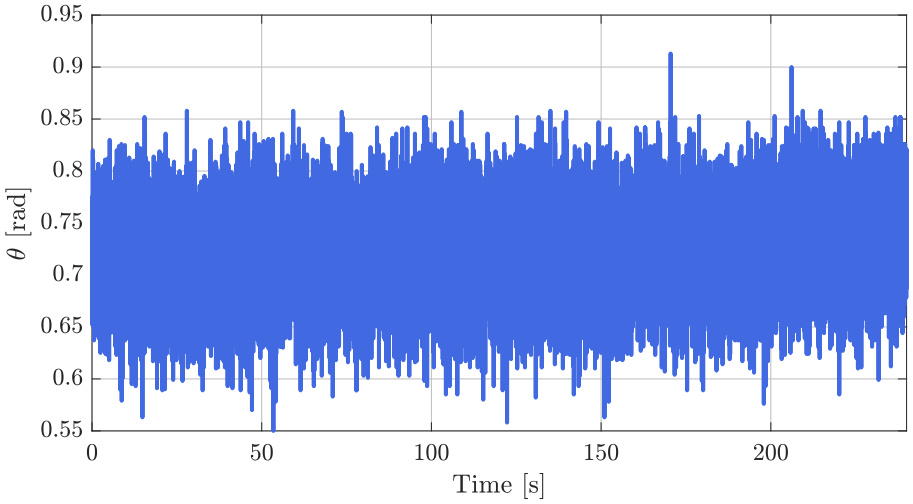


Figure B.3. rover headding measured by magnetometer

RTK GNSS

The same data set as above is used for the measurement uncertainty of the GNSS module. The position of the rover in Figure B.4 is observed to drift forth and back over time and the assumption of a normal distributed white noise error term may therefore not hold. It should however be noted that the test conditions at which this data has been collected may not be ideal as it has been performed in a parking lot enclosed by buildings which may affect the results. As there is quite a substantial difference in the variance of the East and North signals the variance of the east signal will be chosen as this is the largest.

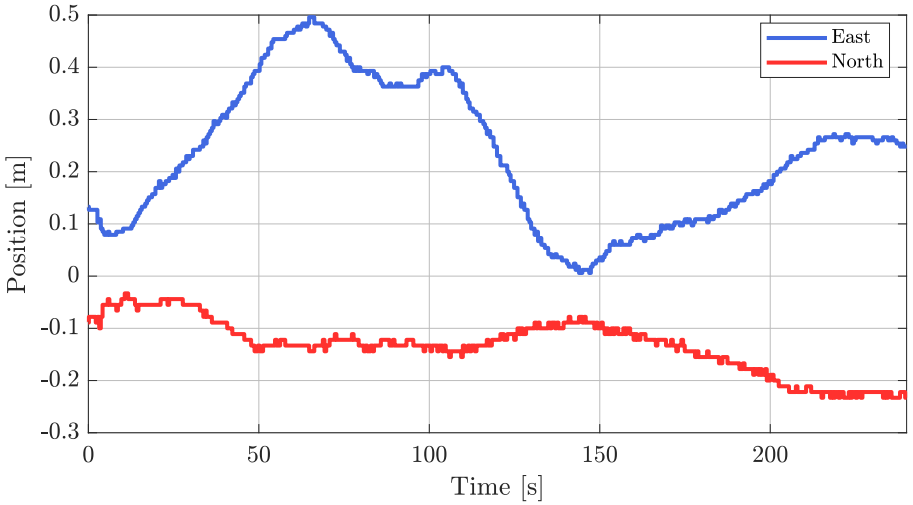


Figure B.4. rover position measured by GNSS with established RTK connection

The measured variance of all the different sensors can be seen in Appendix B

Parameter	Value	Unit
r_{acc}	4e-4	$(m/s^2)^2$
r_{gyro}	6e-3	$(rad/s)^2$
r_{mag}	2e-3	(rad^2)
r_{GNSS}	2e-3	m^2

State Estimation Plots

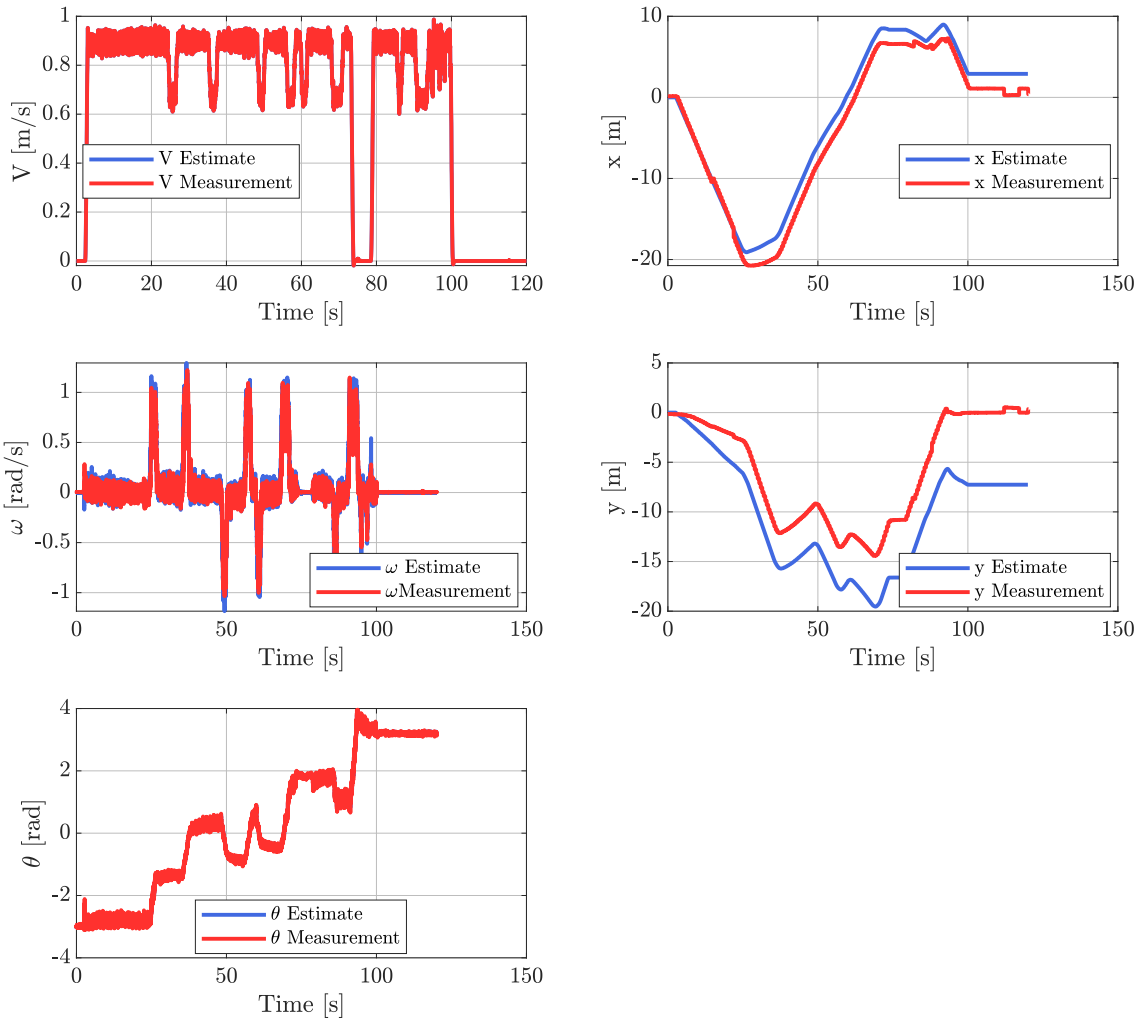


Figure C.1. Comparison of state estimates and measured data for KF using IMU and magnetometer data

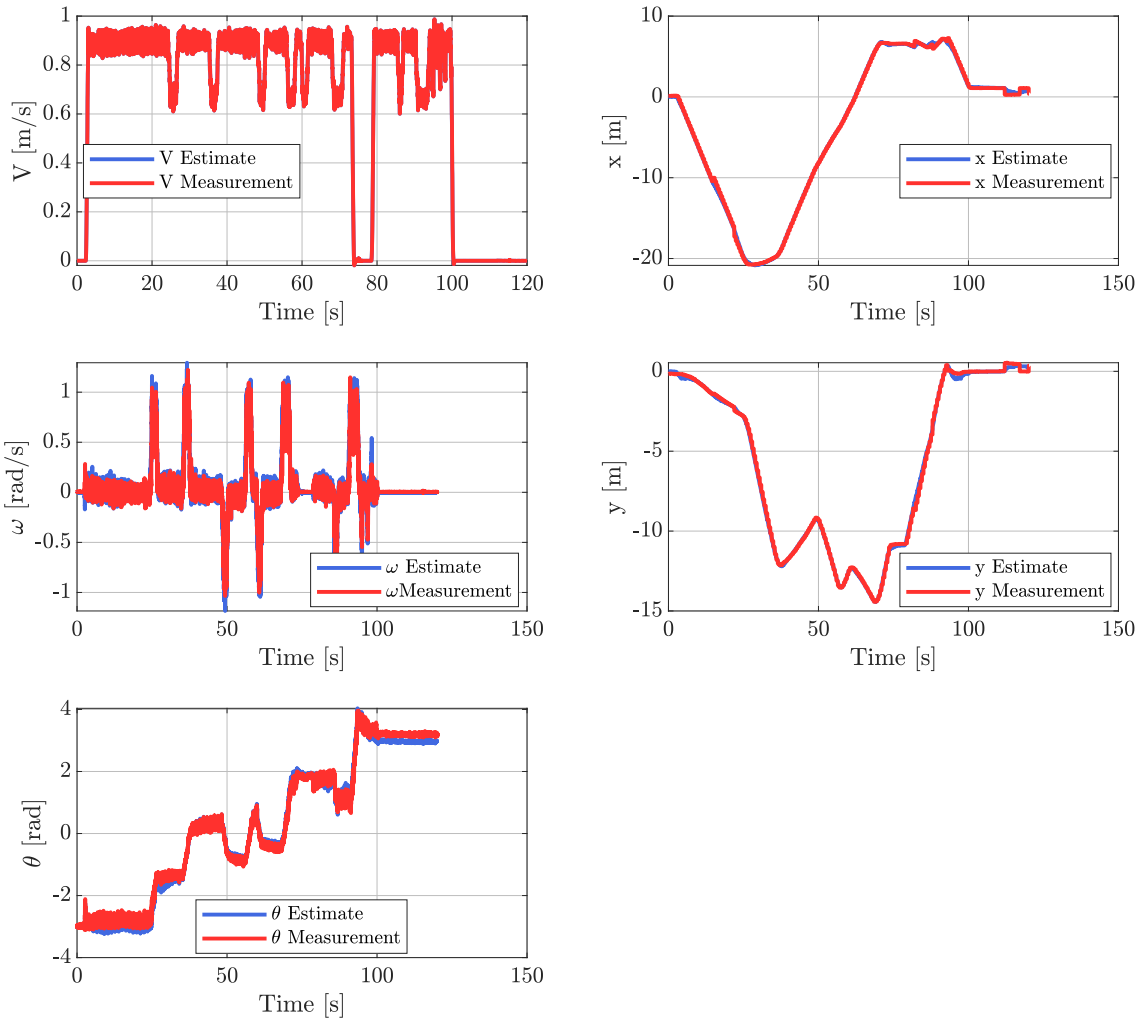


Figure C.2. Comparison of state estimates and measured data for KF using IMU, magnetometer and GNSS data

Figure C.3 shows the estimated rover position at a data fallout of 80 seconds, the filter is here fed the GNSS signals for the first 10 seconds in order to ensure that the initial conditions of the filter at the time of the data fallout is correct.

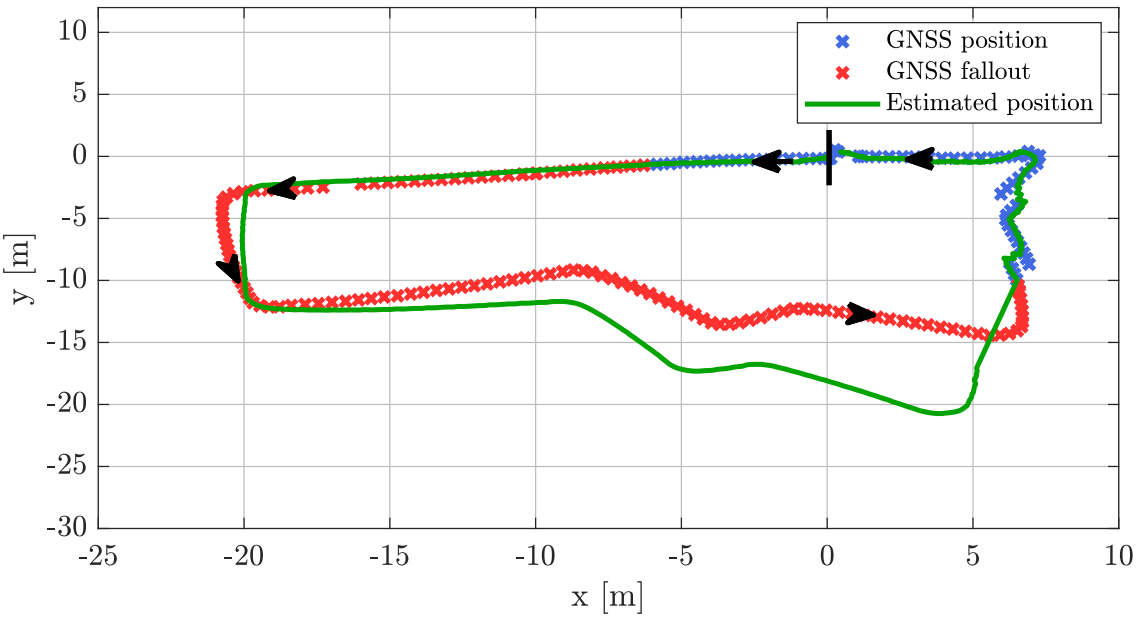


Figure C.3. Estimated rover position during a data fallout of 80 seconds