NTNU
Norwegian University of
Science and Technology

Tight, online integration of INS/GNSS based on RTKLIB

Øyvind Aukrust Rones

Master thesis in Guidance, Navigation, and Control
Department of Engineering Cybernetics
Faculty of Information Technology and Electrical Engineering
Norwegian University of Science and Technology

Submission date: 9th of April 2019
Supervisor: Tor Arne Johansen
Co-Supervisor: Kristoffer Gryte

NTNU
Norwegian University of Science and Technology

Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Engineering Cybernetics

MASTER THESIS DESCRIPTION SHEET

Name:	Øyvind Aukrust Rones
Department:	Engineering Cybernetics
Thesis title:	Navigation for Automatic landing system for fixed-wing UAVs

Thesis Description:

An important step towards more autonomous fixed-wing UAVs is to make the landing process more automatic. In the development of such an automatic landing system, accurate online estimation of position, velocity and attitude is of utmost importance. The purpose of this project is twofold; investigate the quality of the position estimates from state-of-the-art low-cost real-time kinematic (RTK) Global Navigation Satellite Systems (GNSS), and compare this to position estimates from an estimator that integrates GNSS and inertial measurements in a tight manner.

The following items must be considered:

1. Present an overview of relevant theory on GNSS and estimators for navigation using tight integration of pseudorange- and inertial navigation measurements.
2. Present the most promising low-cost GNSS systems, and perform a comparison of these based on metrics that are found relevant, to establish a benchmark.
3. Consider challenges with online estimation, including interpretation of GNSS ephemeris data.
4. Plan an online estimator for position and velocity estimation.
5. Implement the estimator in DUNE (Unified Navigational Environment) on the UAV payload computer.
6. Plan and execute tests of the estimator, running online the UAV payload computer using real measurements, where the accuracy should be compared to that of the current solution.
7. Present the results in a report and discuss the weaknesses of the system and how these can be resolved.

Start date:	2018-09-06
Due date:	2019-04-09

Thesis performed at:	Department of Engineering Cybernetics, NTNU
Supervisor:	Professor Tor Arne Johansen, Dept. of Eng. Cybernetics, NTNU
Co-Supervisor:	Kristoffer Gryte, Dept. of Eng. Cybernetics, NTNU

Abstract

With UAVs being used for an ever increasing number of operations, there is a demand for high-precision PVA-estimation. This is a complicated task, and while many systems have been shown to work in simulations, a lack of estimators providing real-time solutions and running on actual hardware is seen. Some commercial systems, such as the PixHawk flight controllers, provides readily available PVA-estimates with limited setup, and can also be programmed and used as a test platform for different systems. However, no framework is provided for this, and significant setup is required. This thesis seeks to provide a platform that simplifies the setup of future PVA estimation systems. A tightly coupled extended Kalman filter estimating position and velocity by integrating pseudo-range and Doppler measurements with acceleration measurements is presented. It is implemented in the real time navigation framework DUNE and an interface with the open source program package RTKLIB is implemented as well, giving access to a host of GNSS related functionality. The system is loaded onto and run on an embedded computer, BeagleBone Black, running a miniaturized Linux distribution. The results show that the system is able to provide PV estimates in real time, but with a somewhat heavy computational load.

Sammendrag

Ettersom UAV-er blir brukt i stadig flere sammenhenger, er det etterspørsel etter høypresisjons-PVA-estimering. Dette er en komplisert oppgave, og selv om mange system har vist seg å fungere i simuleringer, har det blitt observert en mangel på estimatorer som gir løsninger i sanntid og kjører på faktisk hardware. Enkelte kommersielle system, som for eksempel PixHawk flight controller, gir lett tilgjengelige PVA-estimater, krever begrenset oppsett og kan programmeres og brukes som testplattform for forskjellige system. Imidlertid er det ikke satt opp noe rammeverk for dette, og en betydelig mengde oppsett kreves. Denne avhandlingen ønsker å lage en plattform som forenkler oppsettet av fremtidige estimeringssystem. Et tett koblet "extended Kalman filter" som estimerer posisjon og hastighet gjennom integrering av pseudo-range- og Doppler-målinger med akselerasjonsmålinger er presentert. Det er implementert i sanntidsnavigasjonsrammeverktet DUNE, og et grensesnitt med open source programpakken RKTLIB er også implementert, som gir tilgang til en rekke funksjoner relatert til GNSS. Systemet lastes over til en embedded maskin, BeagleBone Black, som kjører en minimal Linux distribusjon. Resultatene viser at systemet gir PV-estimat i sanntid, men på bekostning av en noe stor belastning av CPU-en.

Preface

This master's thesis concludes my master's degree in Cybernetics and Robotics at hte Norwegian University of Science and Technology in Trondheim. It is a continuation of the project performed in 2017 [23], focusing on some of the practical details of position estimation based on raw pseudo-range measurements. This thesis seeks to improve on the project by implementing a tightly coupled integration filter, running online on an embedded computing platform.

Some of the background theory presented in this thesis is taken from the background of the project [23], with some modifications. Notably, the description ephemeris, multipath and atmospheric errors in section 2.1.4, and the ephemeris equations from section 2.1.3.

I would like to thank my co-supervisor, Kristoffer Gryte, for counseling throughout the work and for reading through the unfinished report several times.

Contents

1 Introduction **1**

1.1 Existing implementations 1
1.2 Thesis contributions 2

2 Theoretical background **3**

2.1 Global Navigation Satellite Systems 3
2.1.1 Signal composition 5
2.1.2 Observables 6
2.1.3 Ephemeris calculations 8
2.1.4 Error sources 11
2.2 Integrating INS and GNSS 17
2.2.1 Architectures 17
2.2.2 Extended Kalman Filter 18
2.2.3 Attitude parametrizations 19
2.3 Software packages 20
2.3.1 The LSTS toolchain 20
2.3.2 RTKLIB 22

3 Implementation **23**

3.1 Obtaining low level GNSS signals 23
3.1.1 Stand alone task 23
3.1.2 PyUblox 23
3.1.3 RTKLIB 24
3.2 Extended Kalman filter 24
3.2.1 Equations 24
3.2.2 Model description 25
3.3 DUNE implementation 29
3.3.1 RTKLIB interface 30
3.3.2 Extended Kalman filter 32
3.4 Embedded Platform 33
3.5 Testing 35
3.5.1 Simulator 36
3.5.2 UBX log 37
3.5.3 Serial device 38

3.5.4 IMC log 38

4 Results **41**
4.1 Stationary tests 42
4.2 Dynamic tests 46
4.2.1 The tightly coupled system 46
4.2.2 Multi-GNSS 47

5 Discussion **51**
5.1 Convergence 51
5.2 Tight coupling 52
5.3 Multi GNSS 52
5.4 CPU usage 53
5.5 Result uncertainties 54
5.5.1 Reference solutions 54
5.5.2 PixHawk and ArduPilot 54
5.6 Future work 55

6 Conclusion **57**

Appendices **59**

A Reference frames **61**
A.1 ECI 61
A.2 ECEF 61
A.3 NED 61
A.4 Body 61
A.5 Transformations between frames 62
A.5.1 NED and ECEF 62
A.5.2 Body and NED 62

B Atmospheric models **65**
B.1 The Klobuchar ionospheric model 65
B.2 Saastamoinen tropospheric model 66

List of Figures

2.1 Triangulation principle 4
2.2 Cross-correlation between a PRN code shifted by 15 bits and the original unshifted sequence. 5
2.3 GPS signal propagation to Earth 13
2.4 Relation between atmospheric errors and elevation 14
2.5 Illustration of earth rotation during signal transmission. 15
2.6 A loosely coupled, closed loop system 17
2.7 A tightly coupled, direct integration filter 18
2.8 The Neptus-IMC-Dune toolchain 21

3.1 Block diagram of the implemented system 30
3.2 Block diagram of the hardware package 34
3.3 A beaglebone with a cape (midlertidig) 34
3.4 A NEO-M8T receiver (midlertidig) 35
3.5 Visibility plot of simulated satellites 37
3.6 Simulator block diagram 38
3.7 The Ublox LEA-M8T multi-GNSS receiver 39

4.1 Satellite image of the test area 41
4.2 Cloud plot of the position error under stationary conditions 42
4.3 Position error during stationary conditions in three dimensions. . 43
4.4 Velocity errors of the stationary systems 44
4.5 Receiver clock bias 44
4.6 Receiver clock bias rate of the tightly coupled system when stationary. 45
4.7 Estimated accelerometer bias with uncorrected acceleration measurements 45
4.8 Position estimates with averaged velocity vectors. 46
4.9 Histogram showing the distribution of the position error during testing. 47
4.10 Histogram showing the distribution of the position error of the estimate based solely on GNSS measurements. 48
4.11 Estimated receiver clock bias rate with and without acceleration measurements. 48

4.12 Number of tracked GLONASS and GPS satellites during testing . 49
4.13 Position error of the multi-GNSS setup with different configurations. 50
4.14 Position estimates with averaged velocity vectors from the solution based only on GNSS measurements. 50

A.1 NED and ECEF frames 62

B.1 Klobuchar model 67

Tables

2.1 Ephemeris data 9
2.2 GNSS typical error magnitudes 12

3.1 GNSS IMC messages and relevant members 31
3.2 Imu and State IMC messages with relevant members 32
3.3 Course of testing 36

4.1 RMSE values with different GNSS receiver configurations. 49

Nomenclature

GNSS

c	Speed of light
ρ_i	Pseudorange from satellite i
R_i	True satellite range
τ	Receiver clock bias
$\mathbf{p_{si}}$	Satellite position
$\mathbf{p}$	Receiver position

Kalman filter

η	The real scalar part of the quaternion
$\boldsymbol{\varepsilon}$	The vector part of the quaternion

Abbreviations

AHRS	- Attitude Heading Reference System
IMU	- Inertial Measurement Unit
INS	- Inertial Navigation System
GNSS	- Global Navigation Satellite System
GPS	- Global Positioning System
SV	- Satellite Vehicle
PVA	- Position-Velocity-Attitude
EKF	- Extended Kalman Filter

Chapter 1

Introduction

Over the last decade the price of common Inertial measurement units (IMU) and global navigation satellite system (GNSS) receivers has decreased exponentially, while seeing an increase in precision and accuracy. Correspondingly, a range of different devices have come to rely on these sensors for position, velocity and attitude (PVA) estimation, but perhaps most interesting is the use of these in unmanned vehicles. Accurate PVA estimation systems are necessary for autonomous vehicles as any operator will have limited control during a live run. However, this can not be reliably provided by the IMU or GNSS measurements alone.

An inertial navigation system (INS) provides PVA estimates based on the specific force and angular velocity measurements of an IMU. Estimates are updated at relatively high frequencies and the IMU measurements tend to contain little noise. The INS is therefore considered to accurately capture the high dynamics of the unmanned vehicle in its estimates. However, as these estimates are based on integrated measurements they are bound to drift over time. GNSS provides position and velocity estimates at a lower frequency based on measurements that tend to be far noisier than those of the IMU, but with great long-term accuracy. Thus, the INS and GNSS based estimates are complimentary in nature and can be combined to have the best of both worlds, with the GNSS measurements eliminating the drift of the INS and the INS providing smooth PVA estimates. There is also the added benefit that the INS can provide *dead reckoning* estimates during GNSS signal outtages.

1.1 Existing implementations

The great performance of INS and GNSS integration has made it a popular choice for PVA estimation, and a considerable amount of examples are found in the literature. In [12] inertial measurements are integrated with GNSS measurements based on a real-time kinematic (RTK) approach, and tested on data logged in the field. [11] adds ultra wideband range measurements with a similar setup for increased robustness during GNSS dropouts and tests the implementation with a simulator. A loosely coupled observer is implemented in [9], with attitude estimates based on a rotation matrix with nine degrees of freedom.

For estimating position and velocity (PV) of an unmanned vehicle in real-time, many systems already exist. The open source software RTKLIB is a popular choice for this task as it is highly configurable and requires minimal setup. Its code base is quite extensive and offers functionality related to most GNSS related operations, such as measurement correction models, DGPS and RTK positioning and can read data from a range of different streams. RTKLIB offers no support for integrating data from other sensors, although these may be integrated with the the RTKLIB PV estimates externally. Another popular choice is to rely on an external flight controller for state estimates, such as those of the PixHawk family. This combines both IMU, magnetometer and GNSS measurements to estimate and output both position, velocity and attitude, and is designed to require minimal setup. However, both of these approaches are examples of a loosely coupled approach to INS/GNSS integration, whereas a tightly coupled integration of raw measurements tends to perform better [6, 10].

1.2 Thesis contributions

From a study of existing integration filter implementations, a lack of online estimators is seen. Moreover, although a real-time integration filter requires significant setup there is little mention of code reuse. This thesis therefore focuses on creating an online, tightly coupled INS/GNSS integration filter using the real-time environment DUNE: Unified Navigation Environment (DUNE). An interface between DUNE and RTKLIB is also added to access raw GNSS measurements and the full RTKLIB code base, as this is believed to aid future implementations GNSS dependent systems. The thesis thus contributes with the following

- A simple direct state extended Kalman filter for integrating acceleration measurements with pseudo-range and Doppler shift measurements in a tightly coupled approach, implemented in the real time navigation environment DUNE.
- An interface between the codebase of the open source library RTKLIB and DUNE.
- A hardware platform running the DUNE implementation in real-time.
- Comparison of position and velocity estimates to those of an onboard pixhawk flight controller, as well as a post processed RTK solution from RTKLIB.

Chapter 2

Theoretical background

This chapter presents the background information that this thesis is built on. An overview of GNSS is given, followed by a look into the integration of INS and GNSS and ending with a quick overview of the major software packages used in the implemented system.

2.1 Global Navigation Satellite Systems

A Global Navigation Satellite System, is a navigation system that provides position solutions from radio signals transmitted by orbiting satellites [10]. This is a collective term for several existing systems, as of March 2019, the American GPS, the Russian GLONASS, the European Galileo and the Chinese BeiDou. The two latter of which are still under development, expected to be operational in 2020, but currently offers partial coverage [2, 1]. Additionally, several regional augmentation systems that are compatible with GNSS exist. The following chapter will focus on GPS as it is arguably the most well known system.

GNSS is split into three *segments*:

- Space segment: Composed of all satellites, also called satellite vehicles (SV), in orbit transmitting GNSS data.
- Control segment: Monitors and updates the clock correction and orbital parameters of each satellite.
- User segment: Composed of all GNSS-receivers. Both military, civilian and commercial.

The satellites of each GNSS keep a medium Earth orbit, with the exception of some geostationary Beidou satellites in a geostationary orbit. These orbits gives a better signal geometry, yielding a more precise navigation solution, and better coverage in polar regions [10]. Each satellite is closely monitored by a number

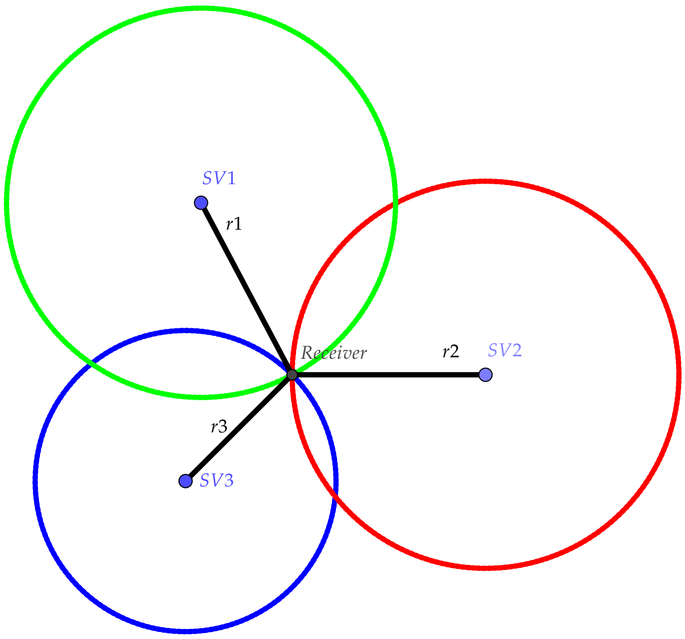

Figure 2.1: This figure illustrates the concept of triangulation from set of satellites, neglecting any measurement errors. SV1-SV3 denotes a set of satellites, while r1-r3 denotes the distance between them and the receiver. Circles with radii given by the ranges are also shown, and the receiver position is given by the intersection of these.

of control stations spread around the planet, that periodically uploads precise orbital parameters to the satellites. In turn the satellites continuously broadcast said parameters to the receivers of the user segments, along with the time of transmission. The receiver may then calculate the satellite's position from the orbital parameters, as well as the range to the transmitting satellite as signal travel time multiplied by the speed of light. From this it is possible to estimate the user's position as indicated in figure 2.1. However, as errors are multiplied by the speed of light, this is only made possible by a set of extremely precise atomic clocks onboard each satellite. The triangulation concept is shown in figure 2.1.

2.1.1 Signal composition

GPS is currently undergoing a modernization program, replacing the old, *legacy*, signals [10, 6]. However, as these have not been fully introduced yet, this thesis will focus on the legacy civillian L1 C / A signal. This signal consists of three distinct parts; the carrier frequency of 1575.42 MHz, a *pseudo-random noise* sequence and a navigation message [19].

Pseudo random noise codes

The pseudo-random noise, also known as a PRN code or C/A code for the L1 signal, is a sequence of 1023 bits, repeated each millisecond. They are unique to each satellite, and have interesting properties with regard to both auto- and cross-correlation as distinct codes are nearly orthogonal to each other. In other words, the cross-correlation of two distinct, arbitrarily shifted, PRN-codes is nearly zero, meaning that satellites can broadcast simultaneously at the same frequency with no risk of interfering with each other. This is known as code-division multiple access (CDMA) [10]. The other property is that the auto-correlation, correlating a PRN code with itself, peaks for an unshifted code sequence. This allows a receiver to align a replica of a PRN code with an incoming one, and thus establish a common time reference. Figure 2.2 illustrates this concept.

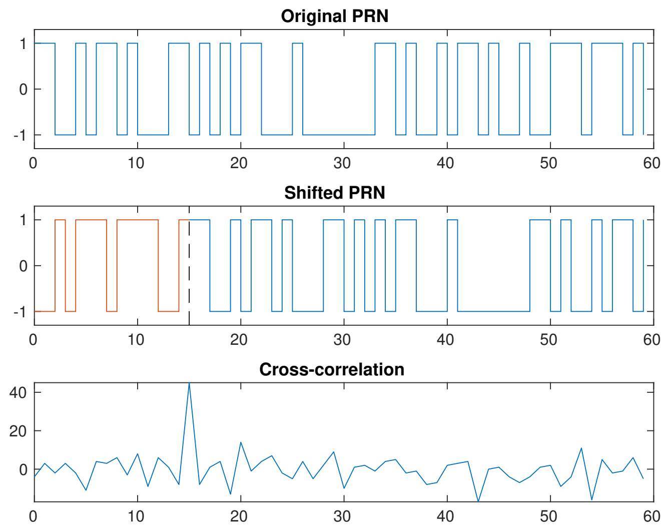

Figure 2.2: Cross-correlation between a PRN code shifted by 15 bits and the original unshifted sequence.

Navigation message

The navigation message includes orbital parameters for calculating satellite position and velocity and is usually called the *ephemeris*. It also contains satellite clock error parameters and the *almanac*, a coarse ephemeris that allows a receiver to calculate the approximate positions of all satellites. This is used to predict when a satellite might be visible and aids rapid signal acquisition [19], see section 2.1.2. The almanac has a longer lifespan than the ephemeris, estimating satellite positions to an accuracy of 3600 meters up to two weeks after the initial upload [10], and is often stored in the receiver between shutdowns. Additionally, the almanac includes parameters from the Klobuchar ionospheric model, described in appendix B.1. The ephemeris and clock error equations are described in detail in section 2.1.3.

2.1.2 Observables

Pseudo-range

When a receiver detects the signal of a new satellite, it will try to lock on to it. This is the process of initially aligning the replica PRN code to that of an incoming signal, and is known as signal *acquisition*. When a match is found for a given signal, and hence the signal phase determined, it is assigned a channel in the receiver, and the *code tracking* process can begin. This is were the pseudo-range measurements comes from; an incoming acquired signal will have lagged behind the replica PRN during the time of transmission, and multiplying the lag time by the speed of light, the pseudo-range is found.

The measurement is called *pseudo*-range as it incorporates numerous errors with the true range, and are explained in more detail in section 2.1.4. This includes both satellite and receiver clock errors, atmospheric delays and relativistic effects as well as others. The pseudo-range between the receiver and a satellite can be modeled as follows [6]

$$P = \rho + \beta - c\Delta t_{sv} + I_r + T_r + \delta p_{sagnac} + \vartheta \tag{2.1}$$

$$\tag{2.2}$$

Where ρ is the geometric range, β is the receiver clock error scaled by the speed of light, Δt_{svi} the satellite clock error, I_{r_i} the ionospheric delay, T_r the tropospheric delay and ϑ are the unmodeled errors. The geometric range is given by

$$\rho = \|\boldsymbol{p}_r - \boldsymbol{p}_s\|_2 = \sqrt{(x_r - x_s)^2 + (y_r - y_s)^2 + (z_r - z_s)^2} \tag{2.3}$$

With $\boldsymbol{p}_r$ being the receiver position at the time of reception and $\boldsymbol{p}_s$ the satellite position at the time of transmission.

$$\tag{2.4}$$

Doppler shift

During satellite acquisition, the high velocity difference of satellite and receiver introduces a Doppler shift on the carrier wave frequency. Hence, if the Doppler shift is unknown or cannot be otherwise estimated, the acquisition process requires a search along frequency in addition to phase. After acquisition, a rough estimate of the Doppler shift has been found and the receiver starts tracking further changes in the incoming carrier wave frequency. Taking advantage of the fact that the relative motion of satellite and receiver is far below the wave propagation speed, the received frequency is approximated as ([16, 10])

$$f_{received} = 1 - \frac{(\boldsymbol{v}_{is}^e - \boldsymbol{v}_{ir}^e) \cdot \boldsymbol{l}_{r,s}^e}{c} f_{transmitted} \tag{2.5}$$

Where $\boldsymbol{l}_{r,s}^e$ is the line of sight vector between satellite and receiver. Consequently the Doppler shift is

$$\Delta f = f_{received} - f_{transmitted} = -\frac{(\boldsymbol{v}_{is}^e - \boldsymbol{v}_{ir}^e) \cdot \boldsymbol{l}_{r,s}^e}{c} f_{transmitted} \tag{2.6}$$

Multiplying by the transmitted signal wavelength yields the relationship between the range rate and the Doppler shift as

$$\dot{r} = -\lambda \Delta f = (\boldsymbol{v}_{is}^e - \boldsymbol{v}_{ir}^e) \cdot \boldsymbol{l}_{r,s}^e \tag{2.7}$$

Where λ is the wavelength of the transmitted signal. The model for the pseudo-range rate is

$$\dot{P} = \dot{\rho} + \dot{\beta} - c\Delta \dot{t}_{sv} + \dot{I}_r + \dot{T}_r + \delta \dot{p}_{sagnac} + \dot{\vartheta} \tag{2.8}$$

However, Δt_{sv}, I_r and T_r are slowly time varying [6], simplifying the model to

$$\dot{P} = \dot{r} + \dot{\beta} + \delta \dot{p}_{sagnac} + \dot{\vartheta} \tag{2.9}$$

where $\dot{\vartheta}_i$ represents the drift of any unmodeled errors.

As the doppler measurement is dependent on receiver velocity, it can be applied to the navigation filter to yield a directly observed receiver velocity estimate. It is also dependent on the line of sight vector, and therefore position, but this dependence is weak [19, 10], and Doppler positioning is rarely used in GNSS position. A weighted least squares approach to PV-estimation is shown in [16], based on pseudo-range and doppler measurements. [25, 26] makes an analytical analysis of the number of required measurements for PV-estimation in two dimensions, but states that it is easily generalized to three, albeit with the assumption that measurements are unbiased.

Carrier Phase

The carrier phase measurement will only be mentioned briefly in this thesis, as it has not been used explicitly in the implemented navigation filter. Nonetheless, as these measurements were readily available with the pseudo-range and doppler shift measurements, they were still included in the internal message bus of the system, and is considered a possibility for future additions.

The low noise of the carrier phase measurement can be taken advantage of by being integrated into a pseduo-range filter. This is known as carrier smoothing and one example is the recursive *Hatch* filter [19]

$$\bar{P}_n = \frac{1}{M} P_n + \frac{M-1}{M} (\bar{P}_{n-1} + \Phi_n - \Phi_{n-1}) \tag{2.10}$$

Where $\bar{P}_n$ is the smoothed pseudo-range of iteration n and Φ_n is the carrier phase measurement given in meters.

Carrier phase can also be used to improve velocity estimates. By using time-differenced carrier phase measurements, velocity can be estimated within the order of mm/s, as opposed to the Doppler approach with a resolution in the order of cm/s. This approach is explained further in [8]. [12] also includes this measurement with a nonlinear observer.

2.1.3 Ephemeris calculations

The ephemeris and clock correction parameters as well as the accompanying equations are shown in table 2.1 and equation set 2.11, respectively. Satellite velocities are needed to estimate receiver velocity from the doppler shift, but the equations are not shown here. [35] derives equations for calculating satellite velocity, but concludes that a simple two point numerical differentiation is precise to the order of millimeters. As opposed to the other GNSS, GLONASS does not broadcast any Keplerian parameters, rather broadcasting ECEF position, velocity and acceleration directly. [10]

Ephemeris equations

Several parameters are needed to calculate clock error and satellite position. Here as shown in [6] and [10]. All of the parameters from the ephemerides are underlined.

The measured satellite broadcast time contains small errors due to satellite clock bias and relativistic effects. The first step is therefore to apply the correction

$$\Delta t_{sv} = \underline{a_{f0}} + \underline{a_{f1}}(t - \underline{t_{oc}}) + \underline{a_{f2}}(t - \underline{t_{oc}})^2 + \Delta t_r \tag{2.11a}$$

Time

Parameters	
t_{sv}	Satellite broadcast time
w_n	Satellite week
t_{ow}	Seconds in GPS week
t_{gd}	Group delay
t_{oc}	Clock data reference time
t_{oe}	Ephemeris reference time

Corrections	
a_{f2}	Satellite clock correction 2
a_{f1}	Satellite clock correction 1
a_{f0}	Satellite clock correction 0

Orbital

Parameters	
M_0	Mean anomaly at reference time
i_0	Inclination angle at reference time
Ω_0	Longitude of ascending node of orbit
ω	Argument of perigee
e	Eccentricity
$\sqrt{A}$	Square root of semi-major axis
$\dot{\Omega}$	Rate of right ascension
$\dot{i}$	Rate of inclination angle

Corrections	
C_{us}	Lateral correction
C_{uc}	parameters
C_{rc}	Radius correction
C_{rs}	parameters
C_{is}	Inclination correction
C_{ic}	parameters
Δn	Mean motion difference

Validity

Validity	
IODC	Clock data validity
IODE	Ephemeris validity

Table 2.1: Ephemeris data

Where the relativistic correction Δt_r is

$$\Delta t_r = F\underline{e\sqrt{A}}sin(E) \tag{2.11b}$$

Where $F = -4.442807633E^{-10}$ is a constant and E is calculated in equation 2.11h

and the corrected satellite time is

$$t = t_{sv} - \Delta t_{sv} \tag{2.11c}$$

Note that equation 2.11a and 2.11c are coupled. However, it has been found that t_{sv} can approximate t in this case without any notable lack in precision [10].

The mean motion is then computed as $n_0 = \sqrt{\frac{\mu}{\underline{A^3}}}$ and corrected

$$n = n_0 + \underline{\Delta n} \tag{2.11d}$$

With the time of signal transmission relative to the ephemeris reference time

$$\Delta t = t_{st} - \underline{t_{oe}} \tag{2.11e}$$

Mean anomaly, M, is then found

$$M = \underline{M_0} + \left(n_0 + \underline{\Delta n}\right)\Delta t \tag{2.11f}$$

To find the eccentric anomaly, E, Kepler's equation is solved iteratively

$$M = E_k - \underline{e_0}sin(E_k) \tag{2.11g}$$

Performing 20 iterations should give centmetric accuracy, while 22 iterations yields millimetric accuracy [10].

From the eccentric anomaly, the true anomaly can be found

$$\nu = tan^{-1}\left(\frac{\sqrt{1 - \underline{e_0}^2}sin(E)}{cos(E) - \underline{e_0}}\right) \tag{2.11h}$$

Expressing position in polar coordinates, the argument of latitude is found as

$$\phi = \underline{\omega} + \nu \tag{2.11i}$$

The orbital radius varies with the eccentric anomaly however, harmonic perturbations are present here and in the argument of latitude as well as the inclination. Thus, the corrections of argument, radius and inclination, respectively, is found

$$\delta u = \underline{C_{us}} sin(2\phi) + \underline{C_{uc}} cos(2\phi) \tag{2.11j}$$
$$\delta r = \underline{C_{rs}} sin(2\phi) + \underline{C_{rc}} cos(2\phi) \tag{2.11k}$$
$$\delta i = \underline{C_{is}} sin(2\phi) + \underline{C_{ic}} cos(2\phi) \tag{2.11l}$$

Applying these yields the corrected expressions

$$u = \phi + \delta u \tag{2.11m}$$
$$r = \underline{A}(1 - \underline{e_0} cos(E)) + \delta r \tag{2.11n}$$
$$i = \underline{i_0} + \delta i + \underline{\dot{i}} t \tag{2.11o}$$

The transmitted longitude of the ascending node, $\underline{\Omega_0}$, is transmitted with respect to the week epoch. With respect to the reference time this is given by

$$\Omega = \underline{\Omega_0} - w_{ie}(\Delta t + \underline{t_{oe}}) + \underline{\dot{\Omega}} \Delta t \tag{2.11p}$$

Where $w_{ie} = 7.2921151467E^{-5}$ is the rotation rate of the Earth. Satellite position in the orbital plane is then found as

$$X = rcos(u) \tag{2.11q}$$
$$Y = rsin(u) \tag{2.11r}$$

Finally, satellite position in the ECEF frame is given by

$$x = Xcos(\Omega) - Ycos(i)sin(\Omega) \tag{2.11s}$$
$$y = Xsin(\Omega) + Ycos(i)cos(\Omega) \tag{2.11t}$$
$$z = Ysin(i) \tag{2.11u}$$

It should also be noted that the equations should handle week crossovers, as the GPS time will be reset at this point. This is done by adding ± 604800 to the corrected GPS time so that it stays in the intervall $[0, 302400]$.

2.1.4 Error sources

Several errors have to be handled when dealing with GNSS positioning. The errors considered in this thesis are given in table 2.2 with typical magnitudes as stated in [24, 19, 21, 10, 18], and will be further explained.

Error sources	
Ionospheric	16-27 m
Tropospheric	2.5-5 m
Ephemeris	1-5 m
Multipath	statistic error, but up to 100 m in extreme cases
Group delay	1-2.5 m
Earth rotation	10-40 m
Relativistic	0-13 m

Table 2.2: GNSS typical error magnitudes

Atmospheric errors

When the GPS signals are transmitted through the atmosphere, the GPS's assumption that the signals travel in a straight line at constant speed no longer holds. The change in signal speed is much more significant than the effect on the signal path [6] and is represented by the refractive index:

$$\eta = \frac{c}{v}$$

Where c is the speed of light and v is the actual signal speed. When the refraction index of a medium frequency dependent, the medium is said to be *dispersive*.

To model the atmospheric delay the atmosphere is divided into two layers. The ionosphere is the section of the atmosphere between 50 and 1000 km from the surface of the earth, while the troposphere is the lower layer between receiver and the ionosphere. The magnitude of the atmospheric errors each vary with the SV elevation angle as the length of exposure through the atmosphere increases inversely with elevation angle [6, 10]. This is due to the fact that a lower elevation angle results in a longer travel path and by extension a longer exposure to the warping effects of the atmosphere. Figure 2.3 demonstrates this.

Typical ionospheric and tropospheric delays with respect to elevation angle are shown in figure 2.4.

Ionospheric error The ionospheric error is a result of free ions and positively charged molecules in the ionosphere. The level of ionization depends on solar activity, seasons and time-of-day which directly affects the speed and travel time of GPS-signals, thereby resulting in an error in the range measured at the receiver. There are several ways of combating this, however. One approach is to take advantage of the fact that the ionosphere is dispersive by using a dual-frequency receiver. The residual ionospheric error in a dual-frequency receiver is in the order of 0.1 m [10], and is the main reason the GPS employs two frequencies. Most receivers only use the L1 frequency however and must estimate the error through a secondary receiver or by mathematical models, such as the NeQuick

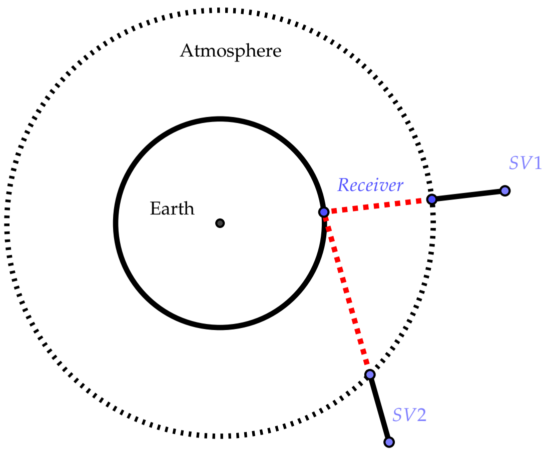

Figure 2.3: GPS signal propagation to Earth. SV2 is an example of a satellite with a low elevation angle, while SV1 has a high one. The dashed red line is the path the signal follows to the receiver and is longer for SV2.

[4] or Klobuchar models [15]. The latter is used by the GPS, and the external parameters needed by the receiver is transmitted in the almanac, while also being a function of the receiver position, satellite azimuth and elevation. Galileo uses the former while GLONASS does not broadcast any ionospheric model parameters [10]. The Klobuchar ionospheric model is shown in appendix B.1.

Tropospheric error The tropospheric error is usually the smallest of the atmospheric errors, but it is also much more sensitive to low elevation angles, with a worst case error of up to 30m [6]. As the troposphere is non-dispersive, delays are consistent for L1 and L2 and both single- and dual-frequency receivers must therefore rely on models to apply corrections. Normally, average values for the receiver's location is employed, although meteorological instruments can aid the receiver to obtain more precise results [6].

Gases like nitrogen and oxygen, normally called the dry components, and water vapor, called the wet component, affect the refraction index differently and must therefore be modeled separately. The dry components account for around 90% of the tropospheric delay and is fairly easy to model as the parameters often are stable for a given area. The wet components are harder to model accurately as they are local and highly varying, but only accounts for about 10% of the delay, [6, 10].
There are several approaches to dealing with the tropospheric delay, where this

thesis relies on the the well known Saastamoinen tropospheric model found in appendix B.2. [13] offers an interesting approach to a cell based modeling of the troposphere based on a dense collection of GPS receivers. [22] investigates how heavy rainfall affects GNSS signals and [3] investigates seasonal variations of the troposphere with respect to long term climate monitoring.

Figure 2.4: Atmospheric errors as a function of satellite elevation angle. (From [10])

Ephemeris error

The ephemeris model is estimated through a curve fit to the measured orbit by the control segment and variations might therefore occur, [6]. The result is a difference between actual and estimated SV position and the associated angle between actual and estimated signal path leads to a range error due to the long distance between receiver and SV. In fact, the sensitivity of position to the angular parameters in the ephemeris is in the order of 10^8 deg, while the angular rate parameters has a sensitivity in the order of $10^{12} deg/s$.

Multipath

Multipath errors occur when a satellite signal reach the receiver through multiple paths due to the signal reflecting off some surface. Signals that are reflected will arrive later than the signal taking the direct route and can usually be detected by the receiver if the delay is large enough. If the reflected signals arrive before the PRN code has been correlated however, interference can occur and shift the correlation peak. As the correlation peak is used to measure the delay from transmission to receival, pseudorange errors will occur. This also means that higher data rate signals are less susceptible to multipath errors [10].

Signals from satellites at low elevations travel nearly parallel to the surface of the Earth and the chance of the signal being reflected off the ground is therefore substantially higher. Thus, discarding data from these satellites might improve the GPS solution [6, 10].

Group delay

The clock correction polynomial, equation 2.11c, is incorporates the so called group delay of the GNSS equipment. This is the delay in the hardware of the satellite, from the initiation of a transmission until a signal is actually transmitted. However, the polynomial is tuned to dual frequency receivers, so single frequency receivers require an additional correction parameter [27]. This is broadcast in the navigation message and applied as follows

$$(\Delta t_{sv})_{L1C/A} = \Delta t_{sv} - T_{gd} \tag{2.12}$$

For the modernized GPS, the civillian signals will include an additional correction called the *inter-signal correction* [28].

Earth rotation: the Sagnac effect

When working with the non-inertial ECEF frame, it is important to consider earth rotation during signal transmission. The ephemeris equations of 2.1.3 calculates satellite positions as a function of time, in the ECEF frame of that instant, while the receiver, on the other hand, estimates its position with respect to the ECEF frame at the time of signal reception. In other words, as illustrated in figure 2.5, the positions are expressed with respect to two separate coordinate frames. These must be aligned, as neglecting this may otherwise result in a significant east-west error of as much 41 meters at the equator [10].

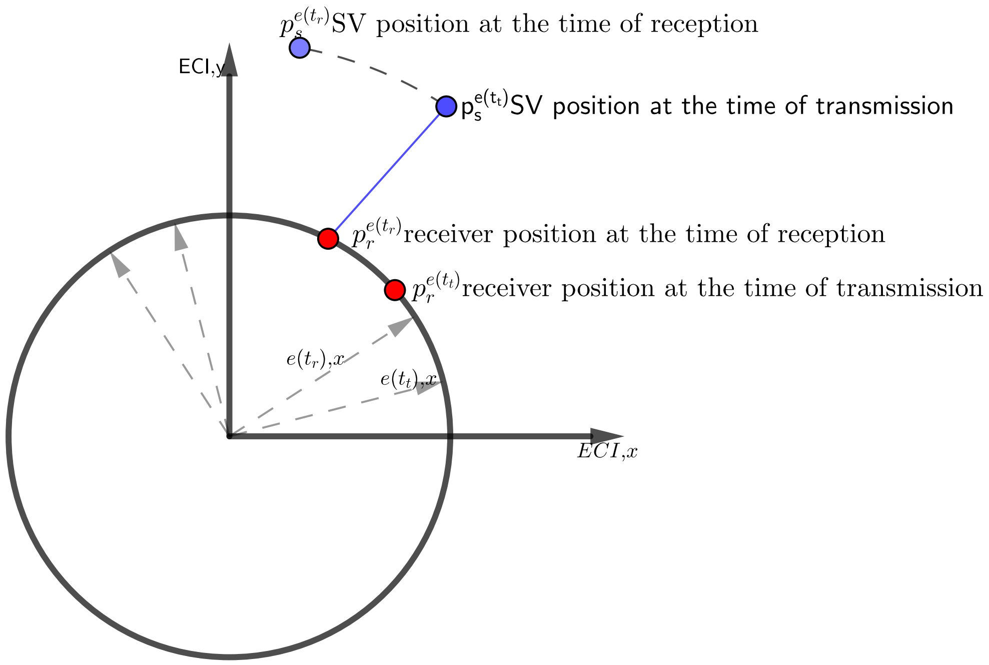
$p_s^{e(t_r)}$SV position at the time of reception
$p_s^{e(t_t)}$SV position at the time of transmission
$p_r^{e(t_r)}$receiver position at the time of reception
$p_r^{e(t_t)}$receiver position at the time of transmission
ECI,y
ECI,x
$e(t_r),x$
$e(t_t),x$

Figure 2.5: Illustration of earth rotation during signal transmission.

Aligning either of the two frames with the other will be equivalent to using a single intertial frame [10]. The true range is then expressed as

$$\rho = \left\| \boldsymbol{R}_{e(t_t)}^{e(t_r)} \boldsymbol{p}_s^{e(t_t)} - \boldsymbol{p}_r^{e(t_r)} \right\|_2 \tag{2.13}$$

where the t_r and t_t denotes the time of reception and transmission respectively, $e(t_x)$ is the ECEF frame at time t_x with respect to a common inertial frame and $\boldsymbol{R}^{e(t_r)}_{e(t_t)}$ is the rotation matrix from the satellite frame to the receiver frame. As the different frames share both origin and a single axis, the rotation matrix is given by

$$\boldsymbol{R}^{e(t_r)}_{e(t_t)} = \begin{bmatrix} cos(w_{ie}(t_r - t_t)) & sin(w_{ie}(t_r - t_t)) & 0 \\ -sin(w_{ie}(t_r - t_t)) & cos(w_{ie}(t_r - t_t)) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

However, given the low rotation speed of the earth, in the order of 10^-5, it is reasonable to apply the small angle approximation, $sin(\theta) = \theta$ and $cos(\theta) = 1$

$$\boldsymbol{R}^{e(t_r)}_{e(t_t)} \approx \begin{bmatrix} 1 & w_{ie}(t_r - t_t) & 0 \\ -w_{ie}(t_r - t_t) & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & w_{ie}\frac{\rho}{c} & 0 \\ -w_{ie}\frac{\rho}{c} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.15}$$

It is also possible to approximate this effect as a correction term, often called the sagnac correction

$$\rho \approx \left\| \boldsymbol{p}_s^{e(t_t)} - \boldsymbol{p}_r^{e(t_r)} \right\| + \delta\rho_{sagnac} \tag{2.16}$$

With the correction approximated to

$$\delta\rho_{sagnac} \approx \frac{\omega_{ie}}{c}(y_s^{e(t_t)} x_r^{e(t_r)} - x_s^{e(t_t)} y_r^{e(t_r)}) \tag{2.17}$$

The non-inertial frame will also affect the range rate measurements with an error of up to $2\frac{mm}{s}$ [10]. This correction term is found by differentiating equation 2.17.

$$\delta\dot{\rho}_{sagnac} \approx \frac{\omega_{ie}}{c}(v_s^{e(t_t)} x_r^{e(t_r)} + u_r^{e(t_r)} y_s^{e(t_t)} - u_s^{e(t_t)} y_r^{e(t_r)} - x_s^{e(t_t)} v_r^{e(t_r)}) \tag{2.18}$$

Where u and v denote the x and y component of receiver and satellite velocity vector.

Relativistic errors

Due to the high speeds and different gravity potential from the high orbit of the satellites, relativistic effects influences the satellite clock frequencies. This effect can be split into a constant frequency component from the satellite's nominal orbit, modified in factory, and a periodic frequency component due to the eccentricity of the orbit. The periodic component must be compensated for by the receiver, as described in section 2.1.3, or it could lead to a worst case error of close to 13 meters, [19].

Without the constant frequency correction applied, the difference in observed frequencies would accumulate to a clock error of approximately $38\mu s$ a day, equivalent to a navigation error of approximately 11 km [21]

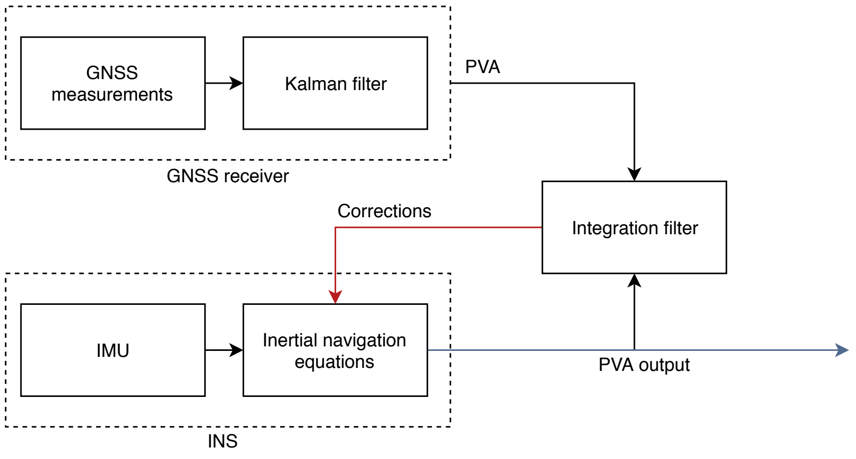

Figure 2.6: A loosely coupled, closed loop system

2.2 Integrating INS and GNSS

2.2.1 Architectures

When integrating GNSS and INS data, the concept of *coupledness* is important to consider. How coupled an integration architecture is describes its complexity with respect to which measurements are used [10, 20]. Although different terms are found in the literature, the most widely used are *loosely coupled*, *tightly coupled* and *deep integration*, as described in [10]. The integration filter output does not necessarily represent the actual state. In the indirect, or error state, filter error estimates of the INS solution is output instead. Further, if these error estimates are applied directly to the INS state estimate, the integration filter is said to be *open loop*. The alternative is to supply the corrections as input to the INS, and is known as a *closed loop* system.

Loosely coupled

A loosely integrated system favors modularity and simplicity for GNSS/INS integration, as shown in figure 2.6. In this scheme, GNSS state estimates are used as measurement inputs to the integration filter, along with either raw IMU measurements or an INS state estimate.

The main drawback of this approach is that it tends to lead to cascaded Kalman filters. Because the estimates of a Kalman filter is based on propagation of earlier filter parameters, errors will be correlated in time. When the integration filter is implemented as a Kalman filter, it assumes that measurements are uncorrelated with respect to time, which is violated if the GNSS receiver implements a Kalman filter itself. This can significantly slow down the estimation of INS errors [10].

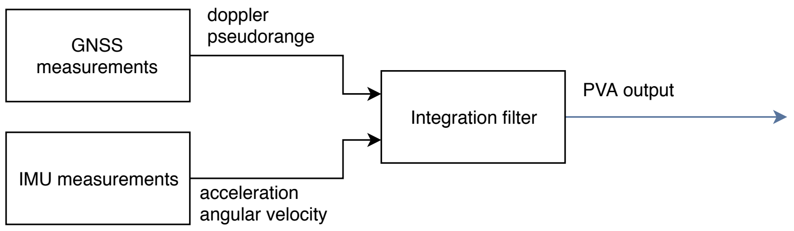

Figure 2.7: A tightly coupled, direct integration filter

Another problems lies in the fact that the GNSS receiver will output no solution when the effective number of satellites fall below four. However, during normal operation the redundancy offered by the separate INS and GNSS solutions will maintain valid estimates even if the integration filter should fail.

Tightly coupled

The tightly integrated approach has no stand-alone GNSS solution, instead utilizing raw measurements from both an INS and a GNSS receiver in a single integration function. This approach does not suffer from the measurement bias introduced in a loosely coupled system, and the GNSS can still aid the INS even when fewer than four satellites are available. This approach tends to perform significantly better than the loosely coupled one, even when based on the same hardware [5, 31, 10].

Figure 2.7 shows a tightly coupled integration architecture. Note that no Kalman filter is connected to the GNSS except for the integration filter, which utilizes the GNSS observables directly.

Deeply coupled

With access to a GNSS receivers firmware integration can be done *ultra-tight*. This further couples the INS and GNSS receiver by using the INS to aid the receiver in signal tracking. See [14] for a detailed explanation with an example implementation.

2.2.2 Extended Kalman Filter

The linear Kalman filter assumes that the error covariance follows a normal distribution. This assumption fails for non-linear systems. The Extended Kalman Filter solves this problem by applying linearized models to the covariance equations, while maintaining the nonlinear model for prediction and residual calculations. It should be noted that the extended case is not optimal, and the filter may diverge for inaccurate process models or poor initial state or error covariance estimates.

Indirect implementation

Instead of having the filter estimate a state vector directly, it can be beneficial to integrate high frequency IMU measurements into a nominal state outside of the filter. This state vector is nominal in the sense that it ignores model imperfections and noise terms, which will eventually accumulate. Gaussian distributions of the accumulated errors can then be estimated in the Kalman filter, and their means injected into the nominal state. If the error state can be estimated and injected reasonably fast compared to the system dynamics, it can be assumed that accumulated errors are small in each correction step. In other words, the error state will operate close to the origin. and second order terms can be neglected, leading to simpler models and calculations [29].

When estimating the attitude of a body that may experience any orientation, a globally non-singular representation should be used for the nominal state. However, as the error state is assumed to operate close to the origin, far from any singularities, a minimal representation can be applied. This is beneficial in the case of the quaternion, as the redundant representation leads to a rank-deficient covariance, due to the unit constraint [17]. Another benefit of the indirect filter, is that all the large-signal dynamics are integrated in the nominal state and corrections can therefore be applied at a lower rate than predictions [29]. For GNSS/INS integration, this is usually the case, where GNSS observables, used for corrections, arrive at a much lower frequency than INS measurements.

Multiplicative Extended Kalman Filter

The MEKF is a special implementation of an indirect EKF, used for PVA estimation. Error estimates are injected into the state estimate through through addition for all estimates except attitude, as adding the attitude error to the attitude estimate violates the quaternion unit constraint. By creating an error quaternion and injecting through the quaternion product operator, the quaternion estimate will only deviate from the unit constraint as numerical errors grow.

2.2.3 Attitude parametrizations

There are numerous ways of representing attitude, a few interesting options will be looked into.

Rotation vector Following from Euler's theorem, the rotation of a rigid body can be described as the rotation by an angle ϕ about some axis.

Euler angles An alternative to the vector parametrization approach are the Euler angles. Instead of using a single axis, the euler angles describe rotation as three *simple rotations*; Three consecutive rotations about the basis vectors of some coordinate system.

Quaternions It has been proved that no three dimensional parametrization of rotation can be globally nonsingular [30]. The unit quaternion has the lowest dimensionality required for a globally nonsingular parametrization [17], and is represented as a four dimensional vector. It is also numerically efficient as the rotation of any vector is achieved through linear matrix multiplication .

The rotation matrix of a quaternion $\boldsymbol{q} = \begin{bmatrix}\eta & \boldsymbol{\varepsilon}\end{bmatrix}^T$ can be written as [7]

$$R_{\eta,\varepsilon} = \boldsymbol{I}_{3x3} + 2\eta \boldsymbol{S}(\boldsymbol{\varepsilon}) + 2\boldsymbol{S}^2(\boldsymbol{\varepsilon}) \tag{2.19}$$

Using the relationship $\boldsymbol{S}(-\boldsymbol{\varepsilon}) = -\boldsymbol{S}(\boldsymbol{\varepsilon})$, it is evident that quaternions over-parametrize the rotation, as both $\boldsymbol{q}$ and $-\boldsymbol{q}$ represent the same rotation matrix. For this reason, it is customary to set the restriction $\eta > 0$ [17].

Gibbs vector The Gibbs vector is a projection of the quaternion space defined as

$$g \equiv \frac{\boldsymbol{\varepsilon}}{\eta} \tag{2.20}$$

This is again a 2-1 mapping, where the Gibbs vector maps $\boldsymbol{q}$ and $-\boldsymbol{q}$ to the same point. However, because of the quaternion overparametrization, the resulting mapping is 1-1 for all rotations in the Euclidean space E3. The Gibbs vector introduces a singularity for $\eta = 0$, tending to infinity around this point, corresponding to a rotation of 180°.

Modified Rodrigues parameters Fairly similar to the Gibbs vector, the modified Rodrigues parameters are defined as

$$p \equiv \frac{\boldsymbol{\varepsilon}}{\eta + 1} \tag{2.21}$$

2.3 Software packages

2.3.1 The LSTS toolchain

The Underwater Systems and Technology Laboratory of Porto university offers an extensive toolchain for the control and operation of unmanned air, ground and surface vehicles. This thesis has used the Neptus-IMC-Dune software toolchain shown in figure 2.8.

DUNE

Dune: Unified Navigation Environment (DUNE) is the software package running on the vehicle, providing a framework for a number of tasks. It is a highly modular thread based system, splitting execution into separate blocks known as *tasks*. Tasks can be responsible for low level operations, such as interaction with sensors and actuators, or complex higher level operations such as plan execution

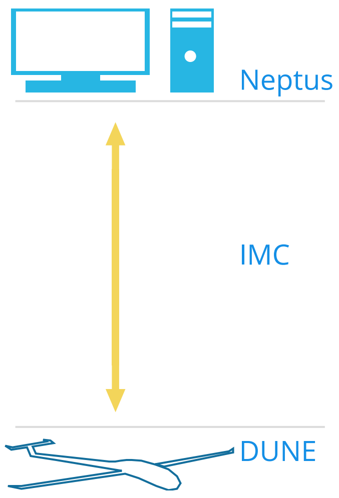

Figure 2.8: The Neptus-IMC-Dune toolchain

and vehicle supervision. Inter-thread communication is realised through the IMC protocol, where each task can bind itself to any IMC message types and and handle them as desired. Dune focuses on code reuse, offering a large codebase for a number of applications, while also offering scripts for easily creating new tasks, all of which are configured through a set of configuration files.

IMC

The Inter-Module Communication (IMC) protocol implements seamless inter- and intra-vehicle communication. It offers custom serialization methods and can therefore transmit data independently of network medium. IMC message types are generated from and described in an xml file.

Neptus

Neptus is a command and control software for the operation of all types of vehicles. It serves as an interface between operator and vehicles at ground control for several phases of a mission; planning, simulation, execution and post-mission analysis. It employs the IMC protocol to communicate with platforms running Dune.

GLUED

GNU/Linux Uniform Environment Distribution (GLUED) is a minimal linux distribution designed to run on an embedded system. It is designed to be as lightweight as possible, and is easily updated with cross compiled packages. It is also highly configurable.

2.3.2 RTKLIB

RTKLIB is an open-source program package for GNSS-related operations created by Tomoji Takasu. It supports all currently existing GNSS and some augmentation systems. Various positioning modes are supported and both input and output is configurable to be in the form of a number of different protocols. Further, it contains a substantial code base for positioning operations.

Chapter 3

Implementation

This chapter presents a tightly coupled extended Kalman filter estimating position and velocity from acceleration, pseudo-range and Doppler shift measurements. A mathematical derivation of the filter model is shown first, followed by an overview of the DUNE implementation. The hardware running the implementation is then presented before the chapter ends with an overview of the testing and tuning process.

3.1 Obtaining low level GNSS signals

Many GNSS receivers can be configured to output raw measurements and ephemerides. However, to access these, a parser must be implemented. There are several alternatives.

3.1.1 Stand alone task

The first option to be considered was to implement a parser task from scratch. This was quickly discarded however, as the ephemeris calculations from section 2.1.3 and measurement corrections would have to be implemented as well. This implementation would be specific to the system of this thesis, making it difficult to use with other systems, and new would have to be added for every new GNSS receiver.

3.1.2 PyUblox

The next to be considered was PyUblox, a python program created by Andrew Tridgell [32]. It can parse several kinds of GNSS receiver output messages, provide corrections and estimate position. It also supports differential GNSS over UDP. However, the python program can not communicate directly with DUNE and would require some different solution, such as communication over UDP. This would require the pyUblox to depend on a UDP task to dispatch measurements, which breaks code abstraction and makes the system difficult to maintain in the future. Also, as the system is to be run on an embedded computer, the additional overhead of employing an interpreter based language should be considered. Lastly, the PyUblox code has not been maintained for some years.

3.1.3 RTKLIB

RTKLIB was the chosen system in the end. It is written in C, which makes it a fairly simple matter to interface it with DUNE, written in C++. This means that a tidy code abstraction can be maintained as a dedicated DUNE task can be implemented for the GNSS related operations of the system, and functions from the RTKLIB code base can be called directly. This is one of the biggest benefits of interfacing RTKLIB. The extensive code base of RTKLIB offers a myriad of different GNSS related functionality, RTK positioning among them, which is believed to significantly aid future work in this area. It is also highly configurable and data can be input from a range of different streams, such as serial devices, log files or TCP servers. The program can configure connected serial devices as well. As all configurations are described in files, it is simple to switch from one configuration from another, and these files can easily be shared with other implementations. There is also a large community situated around RTKLIB, so it has been tested extensively and is still maintained, making an interface with RTKLIB a prime candidate for future implementations. The only downfall is that this setup requires a fair understanding of the RTKLIB source code, which is far from readable with few comments.

3.2 Extended Kalman filter

The chosen integration filter of this thesis is a simple direct state extended Kalman filter. Before the implementation is presented, the EKF equations are shown.

3.2.1 Equations

The discrete time extended Kalman filter method can be divided into a prediction step, based on the state model, and a measurement step as follows.

Prediction

The next state estimate is predicted based on the system model

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}) \tag{3.1}$$

The error covariance is propagated a priori based on the linearized model and process noise covariance

$$\boldsymbol{P}_k^- = \boldsymbol{\Phi}_k \boldsymbol{P}_{k-1} \boldsymbol{\Phi}_k^T + \boldsymbol{Q}_k \tag{3.2}$$

Measurement

The measurement step estimates an incoming set of measurements and calculates a measurement residual, $\boldsymbol{z}_k$, and its covariance, $\boldsymbol{S}_k$

$$\boldsymbol{z}_k = \boldsymbol{y}_k - \boldsymbol{h}(\hat{\boldsymbol{x}}_k^-) \tag{3.3}$$

$$\boldsymbol{S}_k = \boldsymbol{H}_k \boldsymbol{P}_k^- \boldsymbol{H}_k^T + \boldsymbol{R}_k \tag{3.4}$$

The posterior state estimate, $\hat{\boldsymbol{x}}_k$ is calculated based on the average of the priori state and measurement residual weighted by the Kalman gain $\boldsymbol{K}_k$

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{H}_k^T \boldsymbol{S}_k^{-1} \tag{3.5}$$

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k \boldsymbol{z}_k \tag{3.6}$$

The error covariance is then updated. This form of the update is known as the Joseph form and avoids potential problems with numerical stability

$$\boldsymbol{P}_k = (\boldsymbol{I}_{n\times n} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{P}_k^- (\boldsymbol{I}_{n\times n} - \boldsymbol{K}_k \boldsymbol{H}_k)^T + \boldsymbol{K}_k \boldsymbol{R}_k \boldsymbol{K}_k^T \tag{3.7}$$

3.2.2 Model description

The implemented model estimates position and velocity in the ECEF frame of reference, accelerometer bias and receiver clock bias and receiver clock bias rate. Note that all GNSS constellations keep a different time system, and it is therefore necessary to add additional bias estimates when working with a multi-GNSS setup. This thesis works with measurements from both GPS and GLONASS, and hence the offset bias between the two is estimated as well. In that order, this leads to the state space vector

$$\boldsymbol{x} = \begin{bmatrix} \hat{\boldsymbol{p}}_{be}^e \\ \hat{\boldsymbol{v}}_{be}^e \\ \hat{\boldsymbol{b}}_a \\ \hat{\beta}_{gps} \\ \hat{\beta}_d \\ \hat{\beta}_{glonass} \end{bmatrix} \tag{3.8}$$

Note that the attitude is left unmodeled in the filter, although the prediction model depends on the rotation matrix from the body frame to the ECEF frame.

Therefore, the attitude is modeled separately by a different system and assumed to be a perfect representation. It also depends on the gravitational vector decomposed in the ECEF frame. This is found by rotating the *plumb bob* gravity vector, defined in the NED frame as

$$\mathbf{g}_n = \begin{bmatrix} 0 & 0 & 9.80665 \end{bmatrix}^T [\frac{m}{s^2}] \tag{3.9}$$

Measurement model

The EKF measurement vector is

$$y = \begin{bmatrix} P & \dot{P} \end{bmatrix}^T \tag{3.10}$$

composed of the pseudo-range and pseudo-range rate respectively. The measurement model is based on equations 2.2 and 2.9. However, all corrections, except receiver clock bias, are applied outside of the Kalman filter, and are therefore not included in the EKF measurement model. This yields the model

$$P = r + \beta + \vartheta \tag{3.11}$$

$$\dot{P} = \dot{r} + \dot{\beta} + \dot{\vartheta} \tag{3.12}$$

The linearized measurement matrix can be split into a pseudo-range and a pseudo-range rate part as follows

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{H}_P \\ \boldsymbol{H}_{\dot{P}} \end{bmatrix} \tag{3.13}$$

Both will now be derived.

Pseudo-range

To derive the linearized measurement matrix for the pseudo-range measurements, $\boldsymbol{H}_P$, the model is first differentiated with respect to the receiver position

$$\begin{aligned} \frac{\partial P}{\partial \hat{\boldsymbol{p}}^e_{be}} &= \frac{\partial}{\partial \hat{\boldsymbol{p}}^e_{be}} \|\boldsymbol{p}^e_s - \hat{\boldsymbol{p}}^e_{be}\|_2 \\ &= -\left(\frac{\boldsymbol{p}^e_s - \hat{\boldsymbol{p}}^e_{be}}{\|\boldsymbol{p}^e_s - \hat{\boldsymbol{p}}^e_{be}\|_2} \right)^T = \boldsymbol{l}^{e^T} \end{aligned} \tag{3.14}$$

Where $\boldsymbol{l}^{eT}$ is the line of sight vector between receiver and satellite. Differentiating with respect to the receiver clock bias and GLONASS offset is trivial

$$\begin{aligned} \frac{\partial P}{\partial \hat{\beta}_{gps}} &= 1 \\ \frac{\partial P}{\partial \hat{\beta}_{glonass}} &= 1 \end{aligned} \tag{3.15}$$

The model does not depend on any other state variables. Given m distinct pseudo-range measurements the matrix form becomes

$$\boldsymbol{H}_P = \begin{bmatrix} -\boldsymbol{l}_1^{e^T} & \boldsymbol{0}_{1\times 3} & \boldsymbol{0}_{1\times 3} & 1 & 0 & 0 \\ -\boldsymbol{l}_2^{e^T} & \boldsymbol{0}_{1\times 3} & \boldsymbol{0}_{1\times 3} & 1 & 0 & 0 \\ & \vdots & & & & \\ -\boldsymbol{l}_m^{e^T} & \boldsymbol{0}_{1\times 3} & \boldsymbol{0}_{1\times 3} & 1 & 0 & 0 \end{bmatrix} \tag{3.16}$$

Where measurements from the GLONASS satellites will add a one to the last column to include the offset estimate.

Pseudo-range rate

Similarly, the pseudo-range rate model is linearized too. From equation 2.7 it is seen that it depends on both position, velocity and receiver clock bias rate. However, because of the Doppler shift's weak dependence on position, only the velocity and receiver clock bias rate terms are used.

$$\frac{\partial \dot{P}}{\partial \hat{\boldsymbol{v}}_{be}^e} = \frac{\partial}{\partial \hat{\boldsymbol{v}}_{be}^e}(\boldsymbol{v}_s^e - \hat{\boldsymbol{v}}_{be}^e)\boldsymbol{l}^{e^T} = -\boldsymbol{l}^{e^T} \tag{3.17}$$

$$\frac{\partial P}{\partial \hat{\beta}_d} = 1 \tag{3.18}$$

The measurement matrix for GPS satellites are thus

$$\boldsymbol{H}_P = \begin{bmatrix} \boldsymbol{0}_{1\times 3} & -\boldsymbol{l}_1^{e^T} & \boldsymbol{0}_{1\times 3} & 0 & 1 & 0 \\ \boldsymbol{0}_{1\times 3} & -\boldsymbol{l}_2^{e^T} & \boldsymbol{0}_{1\times 3} & 0 & 1 & 0 \\ & \vdots & & & & \\ \boldsymbol{0}_{1\times 3} & -\boldsymbol{l}_m^{e^T} & \boldsymbol{0}_{1\times 3} & 0 & 1 & 0 \end{bmatrix} \tag{3.19}$$

Prediction model

The acceleration measurement, $\boldsymbol{f}_{IMU}^b$, is the input vector of the system and is modeled in the body frame as

$$\begin{aligned} \boldsymbol{f}_{IMU}^b &= \boldsymbol{f}_{ib}^b + \hat{\boldsymbol{b}}_a + \boldsymbol{w}_a \\ \boldsymbol{f}_{ib}^b &= \boldsymbol{f}_{IMU}^b - \hat{\boldsymbol{b}}_a - \boldsymbol{w}_a \end{aligned} \tag{3.20}$$

where $\boldsymbol{w}_a$ is considered Gaussian white noise.

The time derivative of position is simply defined as the velocity

$$\dot{\hat{\boldsymbol{p}}}^e_{be} = \hat{\boldsymbol{v}}^e_{be} \tag{3.21}$$

$$\tag{3.22}$$

Rotating the acceleration measurement 3.20 and adding a Coriolis term to account for earth rotation, the time derivative of the velocity can be written

$$\dot{\hat{\boldsymbol{v}}}^e_{be} = \boldsymbol{R}^e_b(\boldsymbol{f}^b_{IMU} - \hat{\boldsymbol{b}}_a) + \boldsymbol{g}^e - 2\boldsymbol{S}(\boldsymbol{\omega}^e_{ie})\hat{\boldsymbol{v}}^e_{be} \tag{3.23}$$

$$\tag{3.24}$$

Where the acceleration measurement noise is discarded as it has expected value 0 and the rotation matrix $\boldsymbol{R}^e_b$ is estimated outside of the filter. The acceleration bias term is assumed constant

$$\dot{\boldsymbol{b}}_a = \boldsymbol{0} \tag{3.25}$$

The GPS receiver clock bias and its rate is trivial. The GLONASS offset is assumed constant in the prediction model

$$\dot{\hat{\beta}}_{gps} = \hat{\beta}_d \tag{3.26}$$

$$\dot{\hat{\beta}}_d = 0 \tag{3.27}$$

$$\dot{\hat{\beta}}_{glonass} = 0 \tag{3.28}$$

As the filter does not estimate the attitude, this prediction model can be written on state space form

$$\dot{\boldsymbol{x}} = \boldsymbol{F}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}^b_{ib} \tag{3.29}$$

Where $\boldsymbol{x}$ is the state vector and $\boldsymbol{u}$ is the acceleration measurement vector. The transition and input matrix is

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & -2\boldsymbol{S}(\boldsymbol{\omega}^e_{ie}) & \boldsymbol{R}^e_b \\ \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{1\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{1\times3} & \boldsymbol{0}_{1\times3} & \boldsymbol{0}_{1\times3} \\ \boldsymbol{0}_{1\times3} & \boldsymbol{0}_{1\times3} & \boldsymbol{0}_{1\times3} \end{bmatrix} \tag{3.30}$$

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{0}_{3\times3} \\ \boldsymbol{R}^e_b \\ \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{1\times3} \\ \boldsymbol{0}_{1\times3} \\ \boldsymbol{0}_{1\times3} \end{bmatrix} \tag{3.31}$$

Where the rotation matrix, $\boldsymbol{R}^e_b$, is calculated from the attitude received from a separate system.

Discretization

Because the filter is implemented on a computer, the dynamics should be discretized. Defining the acceleration vector

$$\boldsymbol{a} := \boldsymbol{R}_b^e(\boldsymbol{f}_{IMU}^b - \hat{\boldsymbol{b}}_a) + \boldsymbol{g}^e - 2\boldsymbol{S}(\boldsymbol{\omega}_{ie}^e)\hat{\boldsymbol{v}}_{be}^e \tag{3.32}$$

the prediction model becomes

$$\hat{\boldsymbol{p}}_{be}^e \leftarrow \hat{\boldsymbol{p}}_{be}^e + \hat{\boldsymbol{v}}_{be}^e \Delta t + \boldsymbol{a}\frac{\Delta t^2}{2} \tag{3.33}$$

$$\hat{\boldsymbol{v}}_{be}^e \leftarrow \hat{\boldsymbol{v}}_{be}^e + \boldsymbol{a}\Delta t \tag{3.34}$$

Discretization of the process noise covariance matrix normally requires a rather involved integral to be solved. However, the Van Loan method described by algorithm 1 can be used to find both the discretized state transition matrix and process noise covariance matrices [34]

Algorithm 1: Van Loan's discretization method

1. First, form the $2n \times 2n$ matrix $\boldsymbol{M}$:

$$\mathbf{M} := \begin{bmatrix} -\mathbf{F} & \mathbf{Q} \\ \mathbf{0} & \mathbf{F}^\top \end{bmatrix} \tag{3.35}$$

2. Taking the matrix exponential of $\boldsymbol{M}$ yields the following relationship to the discretized system

$$e^{\mathbf{M}\Delta t} = \begin{bmatrix} \cdots & \mathbf{\Phi}_k^{-1}\mathbf{Q}_k \\ 0 & \mathbf{\Phi}_k^\top \end{bmatrix} \tag{3.36}$$

3. $\boldsymbol{Q}_k$ is then found from the upper right matrix as

$$\boldsymbol{Q}_k = \mathbf{\Phi}_k \mathbf{\Phi}_k^{-1} \boldsymbol{Q}_k \tag{3.37}$$

3.3 DUNE implementation

For the system running on hardware, two DUNE tasks have been implemented. One task interfaces the RTKLIB code base, while the other implements the EKF described in section 3.2. Additionally, several preimplemented DUNE tasks are included in the full system. Most notably, an Ardupilot task for connecting DUNE and Ardupilot, and a logging task for storing IMU messages. The Ardupilot task is necessary for two reasons. Firstly, it provides the system with IMU measurements from the Pixhawk IMU and, secondly, it estimates and dispatches the vehicle attitude, which is unmodeled in the EKF. The logging task is necessary for testing, as mentioned in section 3.5. A block diagram of the system is shown in figure 3.1.

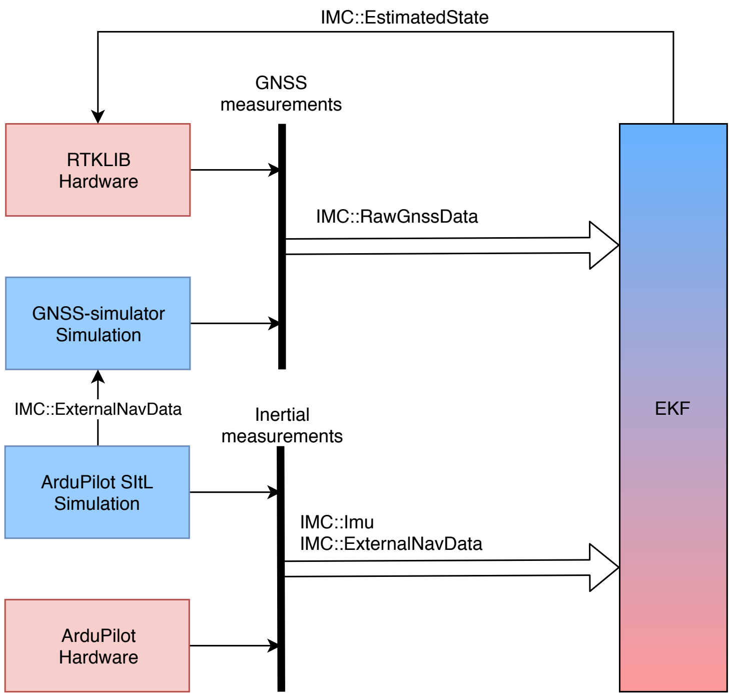

Figure 3.1: Overview of the EKF and the tasks it communicates with. Each box is a distinct task. The tasks are color coded as follows; The blue boxes are tasks used for the simulator (section 3.5.1), while the red ones run on hardware. The EKF task depends only on the IMC-messages and runs regardless of either profile.

3.3.1 RTKLIB interface

The main purpose of the RTKLIB task is to read GNSS measurements from a stream and dispatch these on the IMC bus. It is based on the RTKLIB command-line program rtkrcv, and keeps much of the same functionality, with a few notable exceptions. The task is configured through a configuration file, also called a conf file, and can configure a connected serial device based on receiver configuration file, also called a CMD file. Measurements can be read from different sources, such as log files or serial devices, and, if specified in the conf file, they can be logged in different file formats as well. Unlike the rtkrcv program however, the task does not estimate any state variables, it does not run separately in a terminal and can not output any kind of status screens.

The RTKLIB code base can be accessed in DUNE by adding it as a third party vendor library. However, RTKLIB builds its programs with makefiles only, while DUNE relies on CMake. Therefore, a CMake file is added to the vendor, based on the makefile for rtkrcv. To enable multi-GNSS functionality a flag must be set for each enabled constellation. However, this changes the value of several internal preprocessor macros, which in turn is used to set the size of RTKLIB's internal storage structs. To avoid memory errors it is therefore necessary that all DUNE tasks working with RTKLIB structs add the same flags to their own CMake files.

Message name	**Message members**
RawGNSSdata	• Measurement time (GPS week and time) • Satellite observables (List of IMC message)
RawGNSSsatObs	• GNSS constellation ID • Pseudo-range, doppler-shift and carrier phase • Satellite position and velocity • Measurement standard deviations

Table 3.1: GNSS IMC messages and relevant members

Algorithm 2: RTKLIB interface, pseudocode

```
Input: EstimatedState
Output: RawGNSSdata

Read conf file

while DUNE not stopped do
   data ← Read data from receiver

   if data is ephemeris then
      store data if not already stored
   else if data is measurement and ephemeris stored for satellite then
      calculate satellite position and velocity
      calculate measurement corrections
      dispatch corrected measurements, satellite position and velocity
      store corrected measurements in file
   end if
```

Algorithm 2 describes the rtklib-interface. After configuration files have been read and storage structs have been allocated, the task enters its main execution loop. This reads data from the input stream, which will eventually store an ephemeris. For each available ephemeris, satellite position and velocity is calculated. Measurements are then collected, corrected for atmospheric, sagnac and satellite clock errors and finally dispatched, with satellite position and velocity, as a RawGNSSdata IMC message, shown in table 3.1. The corrections are a function of the receiver position, which the RTKLIB task receives by binding the EstimatedState IMC message, which in this implementation is dispatched from the EKF. If set in the configuration file, the task stores both measurements and receiver position in a UBX and a POS file, respectively, each iteration. Note that the name *RawGNSSdata* is somewhat misleading, as corrections have been applied to the measurements. This IMC message type is simply used because it was implemented prior to this thesis and readily available.

3.3.2 Extended Kalman filter

This task estimates position, velocity, receiver clock bias and receiver clock bias rate based on acceleration and GNSS measurements. It implements the filter described in section 3.2. The measurements are received from the IMU and RawGNSSdata messages, attitude comes from ExternalNavData and finally position and velocity is output in the EstimatedState message as seen in tables 3.1 and 3.2.

Message name	**Message members**
EstimatedState	• Vehicle position (longitude, latitude, height) • Vehicle attitude (between body and NED) • Vehicle velocity (of body, with respect to NED)
ExternalNavData	• Estimated state (IMC message) • Nav Data Type (Status of estimated state)
Imu	• Acceleration (IMC message) • AngularVelocity (IMC message)
Acceleration	• Device time • X, Y and Z acceleration components
AngularVelocity	• Device time • X, Y and Z angular velocity components

Table 3.2: Imu and State IMC messages with relevant members

Each time a set of GNSS measurements are received, they run the measurement step of the EKF. The rate of the measurements depend on how the receiver is configured, and has been set to 10 Hz in this thesis. Similarily, each IMU measurement runs the prediction step. To sufficiently capture the dynamics of the system, these measurements arrive at a higher frequency than the GNSS measurements, at about 30 Hz, although this can vary. As each prediction step discretizes the system, the timestep between measurements should be as exact as possible, and a timestamp is therefore added to each measurement.

A separate state estimate is received from the ArduPilot task in the ExternalNav-Data message. Although only the attitude is used during normal operation, the whole state is kept for both redundancy, and to have a reference during testing and tuning. The external state estimate represents position by latitude, longitude and height, while the attitude denotes the rotation between the body and NED frame. However, as the EKF works with the ECEF frame of reference, this state estimate must be transformed to the ECEF frame as well. The rotation matrix from the body frame to the ECEF frame is calculated as follows

$$\mathbf{R}_b^e = \mathbf{R}_n^e \mathbf{R}_b^n \tag{3.38}$$

where the matrices on the right hand side are the rotation matrix from the NED frame to the ECEF frame and from the body frame to the NED frame. The first is calculated from latitude and longitude by equation A.1, while the second comes from equation A.2, from appendix A.5.1. Equations 3.24 and 3.22 assumes the gravitational vector in the ECEF frame to be known, which is found by rotating the vector defined by 3.9.

3.4 Embedded Platform

The DUNE implementation runs on a BeagleBone Black Industrial (BBB), pictured in figure 3.3. A cape is connected to the BeagleBone, in turn connected to both a GNSS receiver, and a Pixhawk 2.1 (also called The Cube flight controller). The Pixhawk has another GNSS receiver and a magnetometer connected to it. The full payload sits conveniently in a peli case. A simple block diagram illustrates this in figure 3.2

BeagleBone

The BeagleBone Black is a powerful embedded computer board capable of running several different operating systems, such as Ubuntu. It contains both USB, HDMI mini and Ethernet ports, as well as pin headers with both UART, SPI and I2C. By default the BeagleBone comes with a stripped down linux distribution called Angstrom, but it has been flashed with the Glued linux distribution through the onboard SD-card. The BeagleBone can be controlled by SSH over the ethernet port, where the user can start or stop processes, modify configuration files or transfer data to or from the BeagleBone. The router in the case provides the user with additional ethernet ports and a DHCP server to simplify the con-

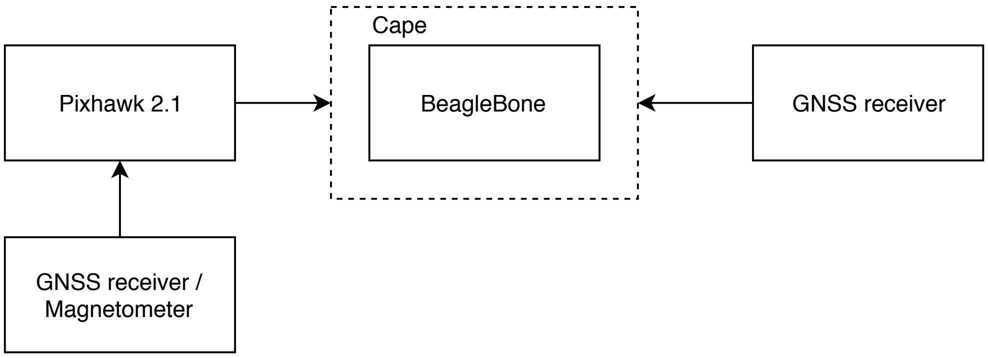

Figure 3.2: Block diagram of the hardware package

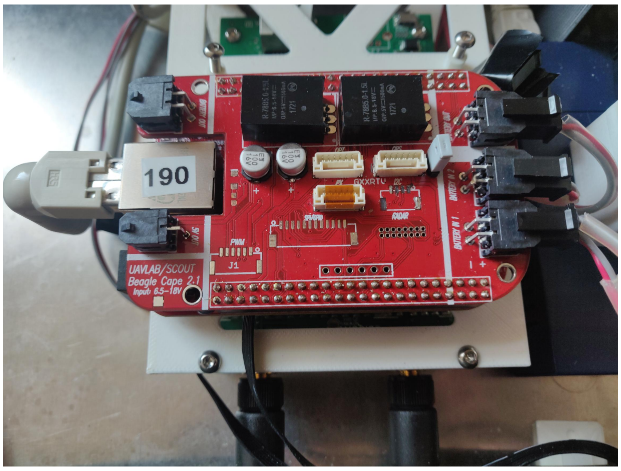

Figure 3.3: A beaglebone with a cape (midlertidig)

nection process.

The cape is a design of the UAVlab at NTNU and was soldered by hand. There are several reasons for including this in the payload. Firstly, it contains connector ports for a GNSS receiver, a pixhawk, two SPI ports, an I2C port and a PWM port, greatly reducing clutter from cables. Secondly, it also contains four molex power connectors, two for voltage input and two for voltage output. It can therefore be used as a power hub for the whole system. Both the pixhawk, receiver and even the ethernet router. Since there are two voltage inputs, power sources can also be *hot-swapped*, keeping the system from powering down. This is convenient when switching batteries or for keeping the BeagleBone running in post-processing. Another useful feature of the cape is its real time clock, powered by a backup battery. With this, the BeagleBone will keep track of time even when turned off, which is especially important for logging purposes as logs are automatically named according to date and time and a name conflict will result in the original log being overwritten. Without the real time clock, the logs would usually be overwritten during startup as all would be named with respect to the BeagleBone's default epoch.

Sensors

The GNSS receiver connected to the BeagleBone is an UBlox NEO-M8T mounted on an InCase series NEO-M8T board. The LEA-M8T breakout board was also used for early testing on a computer as it comes with a built in USB port, but the two receivers are otherwise much the same. Both support GPS, GLONASS, BeiDou and Galileo, as well as the QZSS and SBAS augmentation systems [33]. The receiver uses a Tallysman TW4421-00 Wideband Dual Feed antenna.

Figure 3.4: A NEO-M8T receiver (midlertidig)

Imu measurements are read from the pixhawk flight controller. It contains three separate nine-axis IMU's (3-axis gyroscope, accelerometer and magnetometer) for redundancy, two of which are mechanically vibration-isolated. Additionally, an external GPS and magnetometer is connected. It runs the ArduPlane flight stack and keeps an internal state estimate that is dispatched to the DUNE system, as explained in section 3.3.2.

3.5 Testing

To verify that the whole system was working as intended different methods were used, listed in table 3.3. The EKF alone was first tested with the simulator described in section 3.5.1. Next, the RTKLIB task was tested by configuring it to read real world measurements from a UBX-log file. The system was then tested on real world GNSS data by connecting a GNSS receiver over USB. Finally, the full system was tested and tuned by a log of IMC messages gathered from a test run of the hardware package.

Testing method	Tested system
Simulator	EKF only
UBX log	RTKLIB and EKF (without IMU)
Stationary receiver	Reading and configuration of serial device
IMC log	The full system

Table 3.3: Course of testing

3.5.1 Simulator

Vehicle simulation

With software in the loop (SITL), ardupilot can be run without any hardware. The vehicle is simulated through a flight dynamics model (FDM) which communicates with a flight simulator. The flight simulator simulates measurements and dispatches these to ardupilot. Ardupilot is configured to work with the FDM JSBsim. It is cross platform and contains models for a range of different vehicles, although this implementation simply uses the default model of the Rascal110 RC plane.

GNSS

A simple GNSS simulator has been implemented in DUNE. The focus of the simulator is to provide pseudorange and doppler measurements to a receiver, and as such no ephemeris calculations are made for the simulated satellite positions or velocity. For simplicity the satellites are kept stationary with zero velocity. Positions are taken from a snapshot of visible GPS satellites in the Trondheim area at 13:00 the 13th of November 2018. A visibility plot shows the satellites with respect to elevation angle and azimuth in figure 3.5.

Each satellite of the simulator runs as a separate task on the system. The output of the tasks comes in the form of the IMC message RawGNSSdata, containing satellite position and range and doppler shift measurements. The input is another IMC message, ExternalNavData, containing the true state of the system, dispatched by the ardupilot task. As all the satellites in the constellation run as separate tasks, the true state will be received at the same time for all satellites. In other words, the satellites dispatch messages separately while sharing inputs. As a result, GNSS-messages will be dispatched approximately simultaneously, resulting in messages being sent in *bursts*, resulting in an excessively small time step between measurements. This is not ideal for a discretized system and should

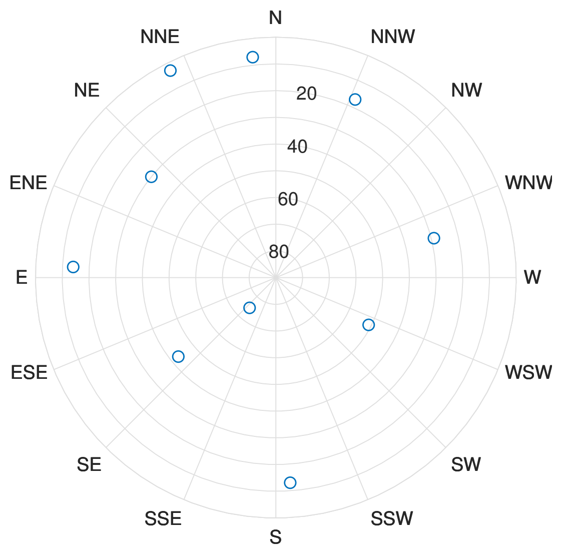

Figure 3.5: Visibility plot of the satellites in the simulator from Trondheim. The user position is in the middle, while the dots denote satellites above the horizon. The distance between user and satellites in the plot is the given by the elevation angle. The angle from the north axis is the azimuth.

be handled. Distinct delays between the satellite tasks are therefore introduced, which spread the dispatching of GNSS messages evenly. Another option would be to buffer the GNSS messages at the integration filter and run all measurements together, but this introduces an unnecessary layer of complexity for the filter.

Full simulator system

Neptus is configured to interface with Ardupilot and can be used to control the simulated vehicle. Not only does this provide the operator with a visual interface for executing maneuvers, but it also offers extensive plotting functionality of messages from the IMC bus shared with DUNE, which is helpful for debugging. DUNE is also configured to run the FlighGear task that offers a IMU measurements with a higher resolution than those received from arduplane. A block diagram of the full simulator is shown in figure 3.6

3.5.2 UBX log

To test both the RTKLIB and EKF tasks together, reading GNSS measurements from a log file is beneficial. Testing can begin immediately after updating code as there is no need to wait for signal acquisition, nor for the ephemeris to download. Additionally, the same log can easily be run with external software, such as RTKLIB, to provide a navigation solution that can be kept as a reference. It

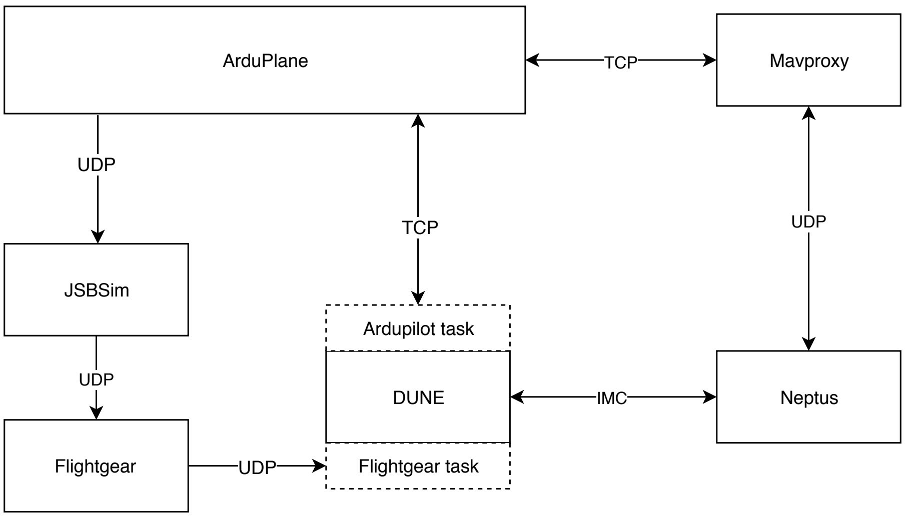

Figure 3.6: Simulator block diagram

is also beneficial as a verification that the integration filter works as intended on real world, noisy data. RTKLIB supports several input protocols, one of which is, as mentioned, the ubx log format. For the dune task equivalent this functionality was included. However, in RTKLIB, this is not designed to work in real time, in that log files are handled as fast as possible. A buffer for storing the measurements of the logfile was therefore added, periodically pushing its contents to the IMC bus according to the original time stamps of each measurement.

3.5.3 Serial device

As the system was tested on a computer, a ublox LEA-M8T multi-GNSS receiver can easily be connected by USB, pictured in figure 3.7. The receiver is configured to output raw measurements by the RTKLIB task, specified by a CMD file. This tests the ability to change the input source based on a configuration file, configuration of a connected serial device and the reading of serial data. The receiver output is *message based*, meaning that it can output several distinct message types with different information. U-center lists the message types being output by a connected receiver, and can therefore be used to verify the configuration.

3.5.4 IMC log

While the UBX log and live serial device is great for testing both the RTKLIB task and the measurement step of the EKF, it does not contain any IMU measurements, and consequently the dynamics of the prediction step is not properly tested. Testing and tuning the full EKF directly on the hardware is impractical as it can only be updated by the somewhat lengthy process of cross-compilation.

Figure 3.7: The Ublox LEA-M8T multi-GNSS receiver

However, DUNE has the ability to log any of the message types on the IMC bus and replay it on any other DUNE system. In other words, replaying the log on a computer, the system will virtually behave as if running with the full hardware package. This is invaluable when tuning the Kalman filter as DUNE can be updated directly and efficiently, there will be no unexpected problems related to real world testing and estimates can easily be compared to a reference.

Chapter 4

Results

This chapter presents the results obtained from the system described in section 3, with the goal of investigating performance of the EKF with real, raw data. Although the DUNE implementation was running during testing, an IMC-log was stored so the test could be rerun on a computer, to simplify the process of fine tuning as well as plotting.

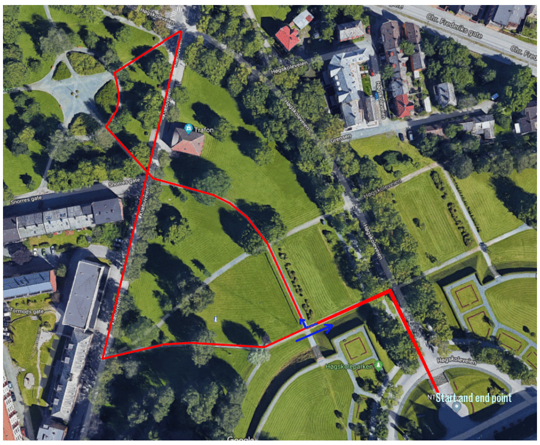

Figure 4.1: Satellite image of the test area

All of the results presented were gathered from tests performed in the area near the Norwegian university of science and technology. All tests were carried out on foot, with the hardware in hand. Figure 4.1 shows a satellite image of the area where the test was undertaken, the planned path in red, and the blue arrows indicating the direction walked.

A post-processed RTK solution has been used as a reference for position and velocity, where a base station log was acquired online from the Norwegian Mapping authority. The base station, TRDS, is positioned approximately 6.5 km away from the testing area, which is assumed to be close enough to obtain a reliable solution.

4.1 Stationary tests

To allow the EKF estimates to converge, the system was kept stationary for 300 seconds before testing was initiated. Results from this process will now be presented.

Position and velocity

The stationary position error of the EKF estimate before full convergence and the RTK solution is shown in figure 4.2. Note that the RTK solution did not have a fix at this time, and its accuracy can not be trusted completely. Regardless, it does seem to have a better precision than the EKF estimate. After convergence some noise is present, with a precision of approximately 2 meters in the east direction and 5 meters in the the north direction. The EKF is shown to converge after approximately 150 seconds in figure 4.3. Note the different scale of the axes.

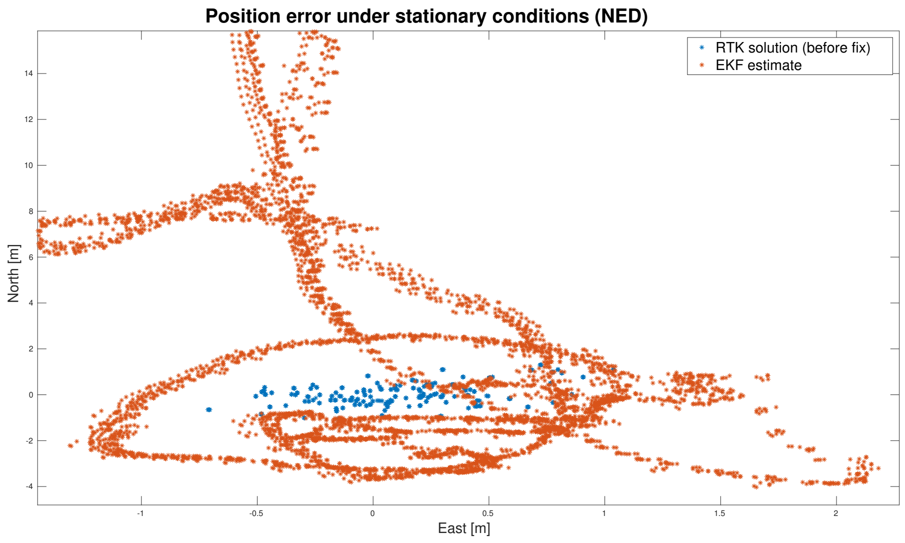

Figure 4.2: Cloud plot of the position error under stationary conditions

The velocity errors of running the EKF with and without acceleration measurements are shown in figure 4.4. The solution based solely on GNSS contains

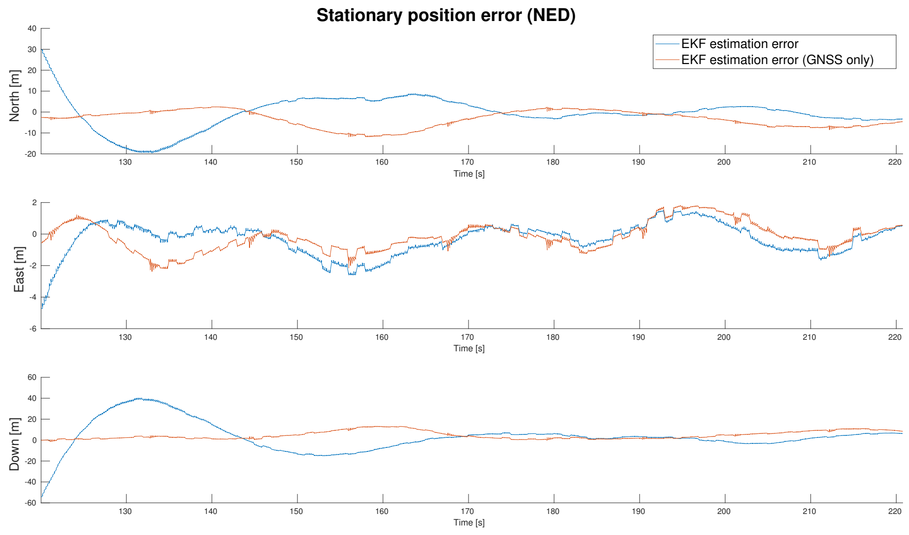

Figure 4.3: Position error during stationary conditions in three dimensions.

some high frequency noise components, and has a root mean square error of $\begin{bmatrix}0.3701 m/s & 0.1398 m/s & 0.4053 m/s\end{bmatrix}$ for north, east and down directions, respectively. Correspondingly, the full system has a root mean square error of $\begin{bmatrix}1.3830 m/s & 0.1637 m/s & 2.9501 m/s\end{bmatrix}$.

Biases

It is important to have a good estimate of the receiver clock bias to get any meaningful results from the GNSS. This estimate is shown in figure 4.5 and, as expected, is shown to have a constant slope after convergence. The estimated rate of the receiver clock bias is shown in figure 4.6 with and without acceleration measurements. While the solution based on GNSS only contains low frequency spikes, the tightly coupled estimate contains some higher frequency noise components.

It was found that the PixHawk IMU measurements contained fairly significant biases, important to estimate to achieve reliable results. This estimate is plotted together with acceleration measurements in figure 4.7. Note that the measurements are corrected for gravity. It is seen from the plot that the EKF makes a fair approximation of the bias.

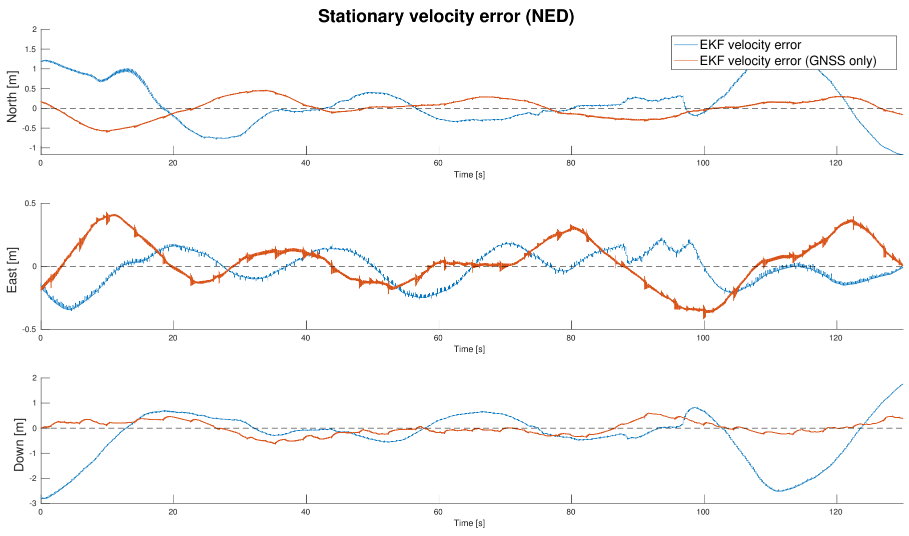

Figure 4.4: Velocity errors of the stationary systems

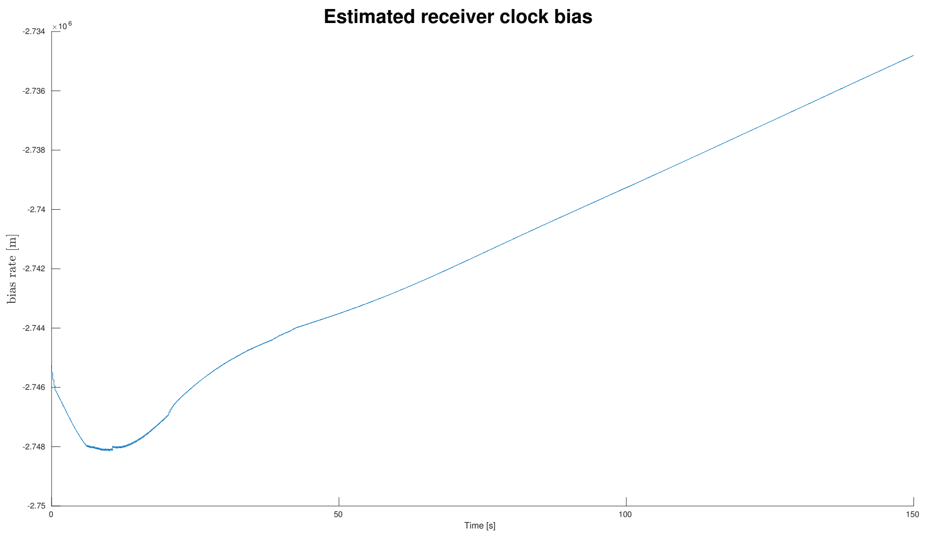

Figure 4.5: Receiver clock bias

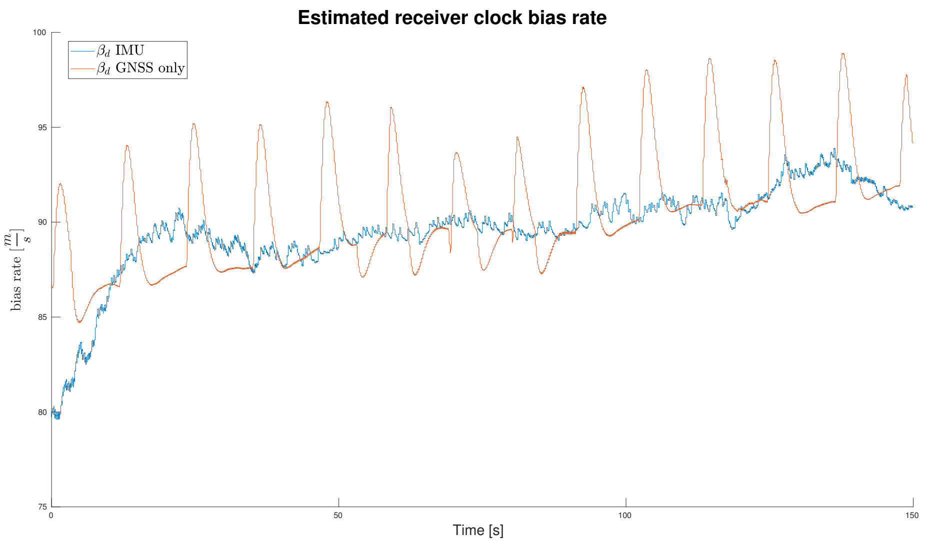

Figure 4.6: Receiver clock bias rate of the tightly coupled system when stationary.

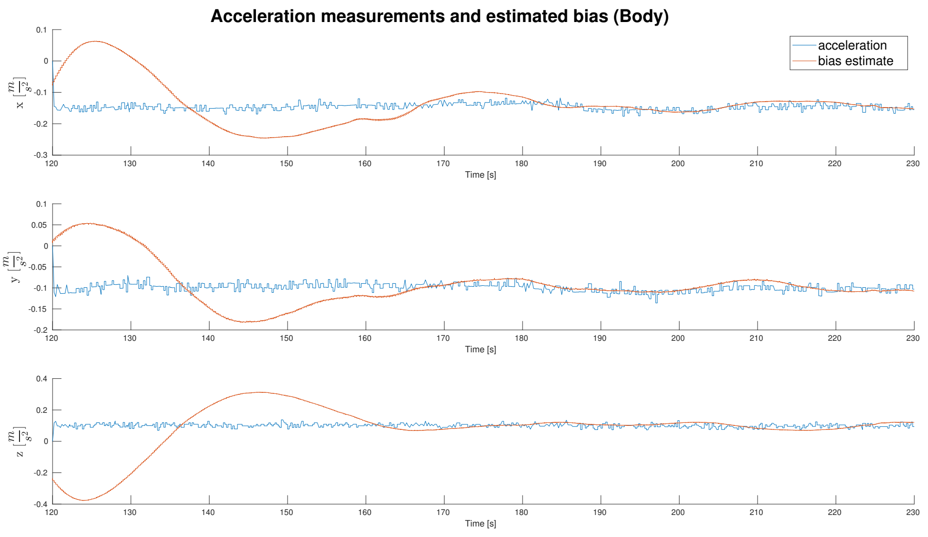

Figure 4.7: Estimated accelerometer bias with uncorrected acceleration measurements

4.2 Dynamic tests

This section covers the part of testing where the system was in motion. To further look into the the benefits of applying acceleration measurements to the EKF, the planned test was performed twice under different conditions. First, the full EKF was run with acceleration and GPS measurements at a frequency of 5 Hz, with results presented in section 4.2.1. Second, the GNSS receiver was configured to output both GPS and GLONASS measurements at 10 Hz, while the acceleration measurements were discarded, leading to state estimates based on GNSS only. These results are presented in section 4.2.2.

4.2.1 The tightly coupled system

Figure 4.8 shows the latitude and longitude estimates with velocity vectors averaged over approximately a second each. The vectors in the lower left corner have a smaller magnitude as this part of the path goes uphill. Assuming course and heading to be equal, the velocity vectors would ideally go tangential to the reference, but as can be seen, this is not always the case.

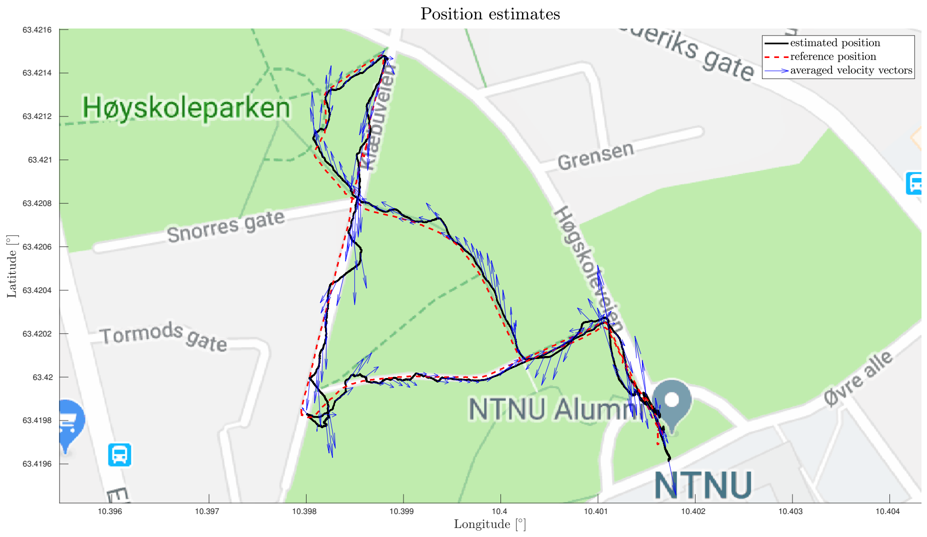

Figure 4.8: Position estimates with averaged velocity vectors.

The distributions of the position error of both setups are represented as histograms in figures 4.9 and 4.10. The tightly coupled solution was found to have a root mean square error (RMSE) of $\begin{bmatrix} 5.1192m & 2.5267m & 7.1108m \end{bmatrix}$ in the north, east and down direction, respectively, while the purely GNSS based solution had a lower RMSE of $\begin{bmatrix}2.4492m & 1.3123m & 6.3648m\end{bmatrix}$. However, the former is seen to roughly retain a normal distribution, as opposed to the GNSS based estimation errors. At this point, it is also interesting to note how the noise of the receiver clock bias estimate increases significantly when the acceleration measurements are discarded, as shown in figure 4.11.

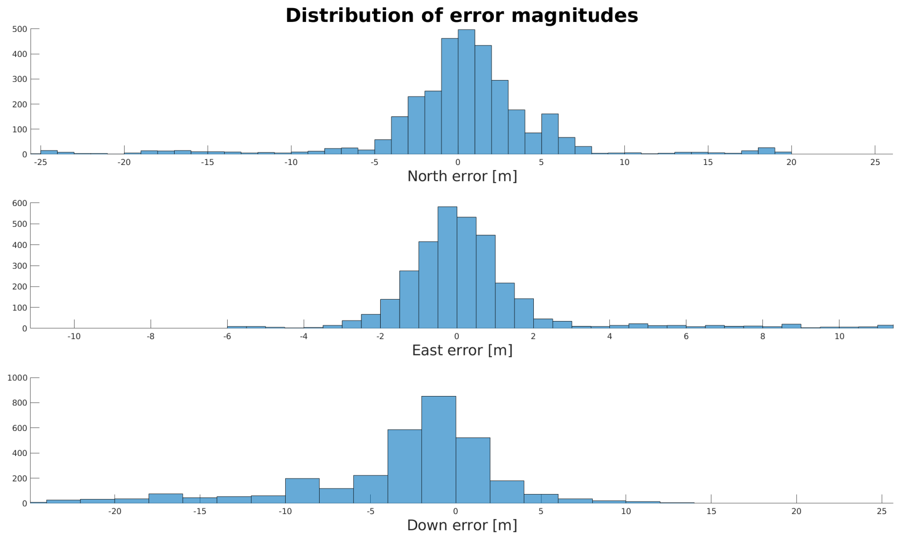

Figure 4.9: Histogram showing the distribution of the position error during testing.

4.2.2 Multi-GNSS

To investigate the benefits of multi-GNSS, three different setups were tested for the EKF using GNSS only. Nameley, GPS and GLONASS at 10 Hz, only GPS at 10 Hz and lastly, both GPS and GLONASS at 5 Hz.

It is interesting to note the increase in the amount of available satellites from adding GLONASS measurements. This is shown in figure 4.12. Note that this is the number of tracked satellites after the elevation mask is applied. In this case, the number of available satellites are more than doubled, and the tracking of the GLONASS satellites is more stable.

The RMSE of the position errors of all three setups are shown in table 4.1. Configuring the GNSS receiver to employ both GPS and GLONASS seems to result in a slightly lower RMSE. The postion errors, with respect to the RTK reference, are shown in figure 4.13. Figure 4.14 shows the estimated latitude and longitude of

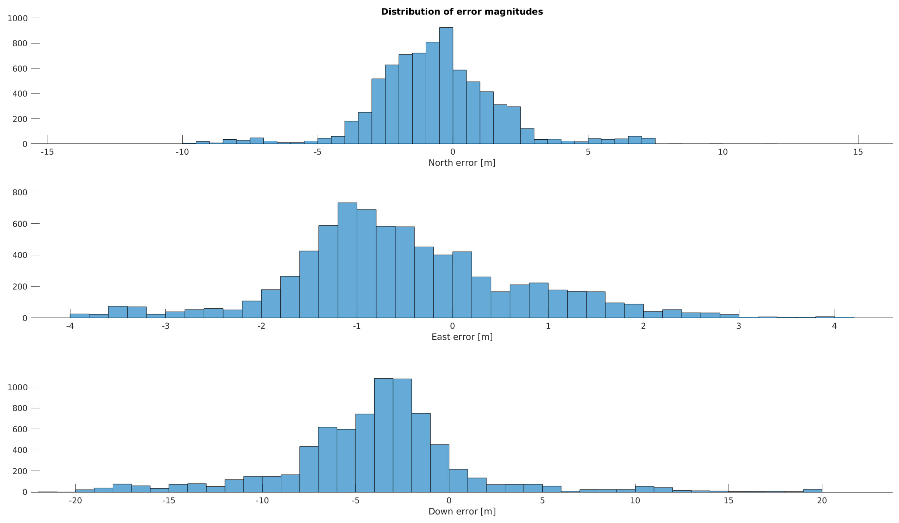

Figure 4.10: Histogram showing the distribution of the position error of the estimate based solely on GNSS measurements.

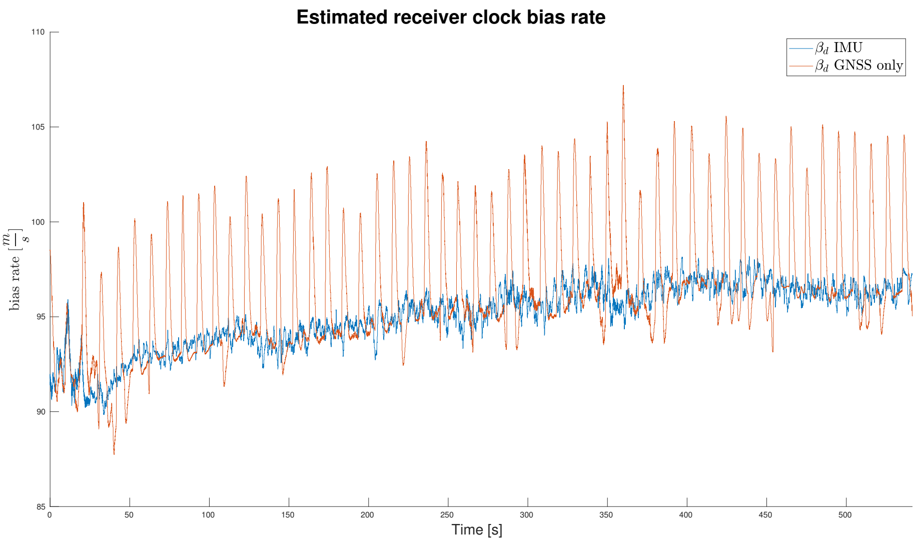

Figure 4.11: Estimated receiver clock bias rate with and without acceleration measurements.

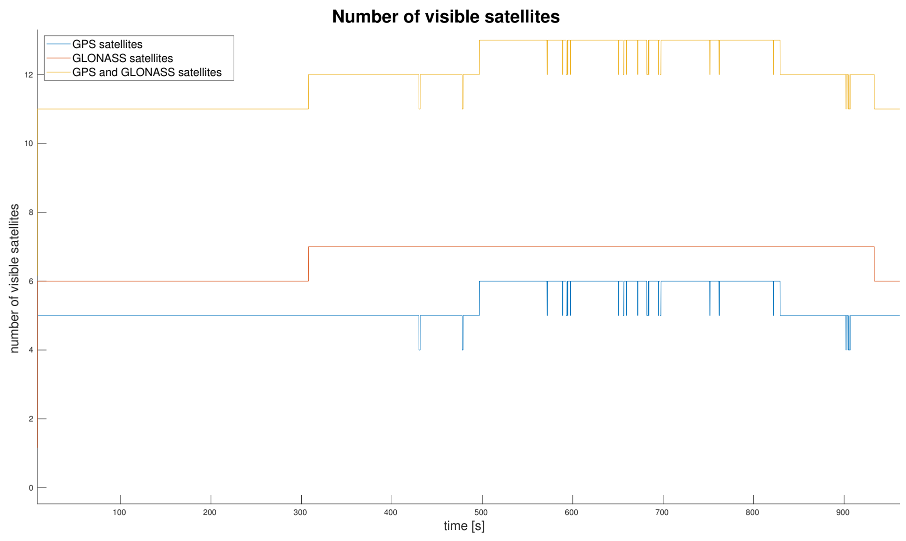

Figure 4.12: Number of tracked GLONASS and GPS satellites during testing

the solution obtained from employing both GPS and GLONASS measurements at 10 Hz.

Setup	**North** [m]	**East** [m]	**Down** [m]
GPS and GLONASS 10 Hz	2.3733	1.3300	6.4983
GPS 10 Hz	3.4710	1.5723	6.8607
GPS and GLONASS 5 Hz	2.5865	1.3967	6.9490

Table 4.1: RMSE values with different GNSS receiver configurations.

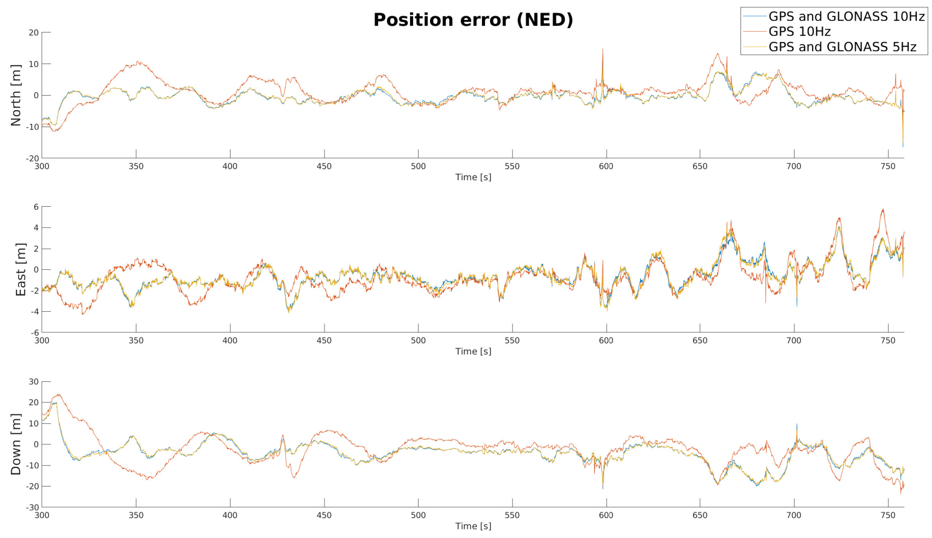

Figure 4.13: Position error of the multi-GNSS setup with different configurations.

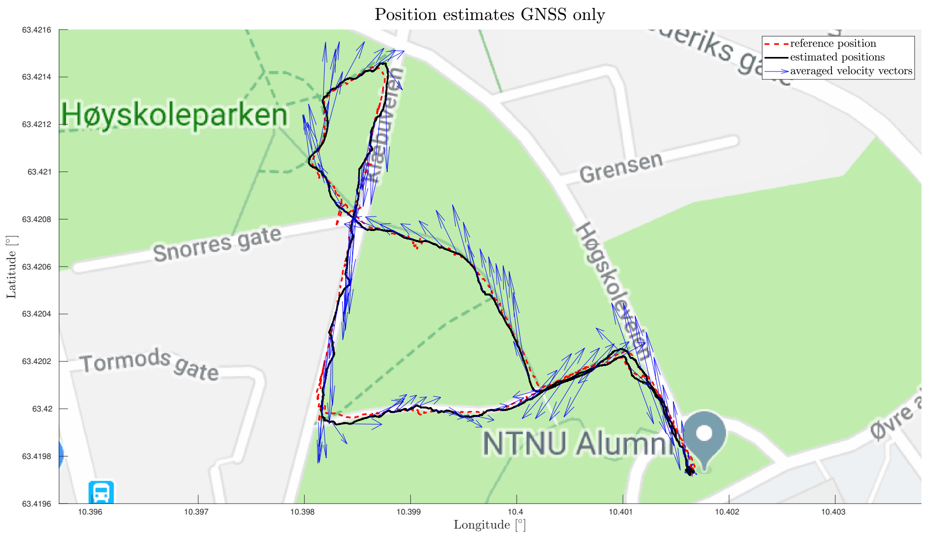

Figure 4.14: Position estimates with averaged velocity vectors from the solution based only on GNSS measurements.

Chapter 5

Discussion

This chapter discusses the system described in chapter 3 in light of the results presented in chapter 4.

5.1 Convergence

The stationary tests revealed the EKF to be fairly slow to converge, requiring approximately 150 seconds before position estimates converged to within ten meters in the north direction and five meters in the east direction. By varying the initial state, it was found that the EKF was particularly dependent on proper initial estimates of both position and receiver clock bias, otherwise deviating significantly after just a few iterations. Although high initial values were added for the variances of these state variables, it proved to have little effect. This may indicate that the measurement covariance for the pseudo-range measurements were too low, resulting in the high value of the initial pseudo-range residuals affecting the state estimates overly much. It should however be noted that the tests performed for this thesis was done by foot, where dynamics are considered fairly low.

As it is known that the quality of GNSS measurement In this thesis, the measurement was simply set as a function of the responsible satellite's elevation angle. However, the convergence and initialization could be improved by adding an extra calibration mode to the EKF, where the assumption of no motion is incorporated, both through the measurements and through the measurement covariance matrix. In other words, a the measurement step is run with a direct measurement of the velocity, set to zero, with a very low measurement covariance as the system is known to be stationary.

5.2 Tight coupling

Comparing the state estimates from the tightly coupled EKF and the estimates based only on GNSS measurements, it is seen that the acceleration measurements contributes to provide smoother estimates, working as a low pass filter. This is especially clear when comparing the different receiver clock bias estimates, where only the decoupled estimate contains significant periodic spikes.

Interestingly, this was also seen to aid the estimation errors in maintaining a normal distribution, and keep the expected value of the errors centered at zero. For the second approach however, the distributions were skewed and biased, violating one of the main assumptions of the Kalman filter estimator, as they can not be described by mean and covariance alone. This is probably the result of stochastic multipath errors. According to the bias present in the error distributions, the reflecting surface would be situated roughly south-west of the testing route, which can be explained from the image in figure 4.1, where several buildings are shown to lay south west of route in the lower left corner. In this test, the redundancy offered by integrating acceleration measurements served to reduce the severity of the multipath significantly, yielding a far more robust system.

While the system was kept stationary, the tightly coupled EKF was found to have significantly higher RMSE values in the north and east direction than the decoupled system, while also being slower to converge. This is because the stationary case does not provide any information with regards to acceleration, and from equation 3.29, discarding the acceleration integration can be interpreted as the acceleration measurement being zero. Thus, in the stationary case, the tightly coupled filter will only integrate the IMU errors, while the second system can be considered to employ perfect acceleration measurements for this scenario. As the acceleration measurements in these tests contained significant bias components, the acceleration measurement errors will be significant until the bias estimate has converged. In figures 4.6 and 4.11 the acceleration measurement noise is seen to have propagated the estimate of the receiver clock bias rate, introducing high frequency noise, yet smoothing the more extreme, periodic transients seen in the decoupled estimate.

Given the close connection between receiver clock bias rate, range-rate and acceleration measurements and velocity, it would have been interesting to investigate the improvement in velocity estimates with respect to a reference, but as mentioned earlier, this was ultimately found to be unavailable.

5.3 Multi GNSS

By adding GLONASS measurements to the EKF, the amount of tracked satellites more than doubled. The effect of this can be seen in table 4.1. In the north direction, this corresponds to an improvement of over one meter, with less significant improvements in the east and down directions. Experiments showed that the tracking of GLONASS satellites was more stable. This is possibly due to GLONASS satellites maintaining orbits with a higher inclination angle, offering better coverage at higher latitudes than GPS. GLONASS measurements were also found to be available several seconds earlier than GPS, as there is no need to wait for an ephemeris, rather broadcasting satellite position, velocity and acceleration directly.

Adding additional satellites improves geometry, and increased precision is expected. By employing both GPS and GLONASS, RMSE values in the north direction improved by more than a meter, with less impressive increase in the east and down directions of 0.2 and 0.4 meters, respectively. The position estimate also seems to be less susceptible to transient errors.

5.4 CPU usage

Running the system on the embedded platform was found to have a fairly large computational footprint, steadily utilizing approximately 80% of the CPU during testing in nominal conditions, with both GLONASS and GPS measurements. As expected, some of the load seemed to be proportional to the amount of available satellites, due to the two sets of ephemeris calculations required to compute satellite positions and velocities. In the RTKLIB DUNE-task, satellite position and velocity is calculated every time a measurement is received, before any satellites are masked out based on their elevation angle with respect to the receiver. However, to determine whether a satellite should be masked out, it is unnecessary to precisely calculate its position. During testing, approximately 30% of available satellites were masked out, and it can thus be beneficial to improve the masking process, especially if additional GNSS are employed.

Assuming the elevation angles to vary slowly, a more computationally effective approach could blacklist satellites after they were masked out, and poll them for updated elevation angles at a low frequency. It would also be possible to use the less precise almanac data to provide a rough, but computationally effective, satellite position estimate. By taking advantage of the fact that satellite orbits are highly stable, it should also be possible to predict satellite positions through Euler integration of the velocity.

It should be noted that the GLONASS satellites do not suffer from this large computational footprint, as was mentioned earlier. It would be interesting to compare the difference in computational load between employing GLONASS and GPS.

5.5 Result uncertainties

Testing revealed some unexpected results, among them the reference systems were found to be slightly lacking.

5.5.1 Reference solutions

There are some uncertainties in the accuracy of the post-processed reference RTK solution. Neither of the two references managed to acquire a proper fix of the integer ambiguities, and the reference for the second test, based on GLONASS and GPS measurements, is even seen to contain sudden jumps in position, as opposed to the smooth path of the one based on GPS only. It is unexpected that the former would provide a less precise estimate as tracking of between six and seven additional satellites should have resulted in a better estimate. However, in post processing, this reference was acquired by only doing a *forward* processing of the measurement log, where the second was acquired from a combined forward and *backward* run. It should also be noted that both of these tests were performed on separate days, so it is possible that conditions were more preferable, with regards to outside factors such as weather or multipath. Another factor could be that the ArduPilot DUNE-task crashed during this test, partially shutting down DUNE and stopping the IMC-log after just a few seconds. The RTKLIB-task did not shut down however, and a log of the raw measurements could still be acquired after testing. This can have affected the measurement log, such as making the RTKLIB-task hang, possibly skipping measurements and forcing the carrier phase tracking to be reset.

5.5.2 PixHawk and ArduPilot

The prediction model of the EKF relied on attitude to integrate IMU measurements. Both attitude estimates and IMU measurements were supplied to the EKF, through the ArduPilot DUNE-task, from an onboard PixHawk, both of which were found to be fairly unreliable. The acceleration measurements contained a significant bias component, which remained even after calibration, and its position estimates were unusable with an RMSE in the order of 10^6 when testing the system in motion. However, it should be noted that applying acceleration measurements was shown to improve state estimates, but further testing is required to ascertain the precision of the attitude estimates as no reference was available for comparison in this regard. It was initially planned to use the PixHawk as a reference for velocity as RTKLIB only outputs position. However, due to the unexpected behaviour of the PixHawk it was decided that these estimates could not be relied upon. Therefore, the estimated velocity of the system could not be properly tested.

5.6 Future work

- **Position and velocity estimates based on RTK**: The main reason an interface between RTKLIB and DUNE was implemented, was so future DUNE systems would easily have access to its large codebase GNSS related functions, most interestingly, its support for RTK based positioning. The implemented RTKLIB-task has already been configured to dispatch carrier phase with this improvement in mind. It would also be interesting to examine the improvement in state estimates from employing carrier-smoothed pseudoranges in the filter.

- **Improved integration filter**: The implemented extended Kalman filter is fairly simple, and can be improved to acquire better state estimates. It is believed that an indirect multiplicative extended Kalman filter with attitude estimation should provide a better performance. Another possibility would be to implement a non-linear observer. This has been shown to have a lighter computational footprint than a Kalman filter, which can be beneficial as CPU usage was found to be fairly high when running the implemented system.

- **Full testing of a multi-GNSS based system**: Due to time constraints, and what is believed to be a faulty pixhawk, the full system was not tested with a multi-GNSS setup. It would be interesting to see how the tightly coupled would improve when aided by additional satellites.

Chapter 6

Conclusion

A tightly coupled extended Kalman filter estimating position and velocity integrating Doppler and pseudo-range measurements with acceleration measurements has been presented. It was implemented in the real time navigation environment DUNE. An interface between the open source program package RTKLIB and DUNE was implemented as well, and was shown to seamlessly provide the DUNE implementation with pseudo-range and range-rate measurements corrected for atmospheric and sagnac errors and satellite clock bias. A hardware setup running the implementation was presented as well.

The results show that integrating acceleration and GNSS measurements provided smoother state estimates than those achieved by employing GNSS measurements alone, and that it may abate the effects of unmodeled stochastic GNSS measurement errors. A multi-GNSS setup was tested as well, showing a fair improvement in estimated position as the number of available satellites increased. However, testing revealed some overhead in the implementation, utilizing around 80% of the CPU during nominal conditions.

Appendices

Appendix A

Reference frames

A.1 ECI

The Earth-centered inertial (ECI) frame has its origin at the center of the Earth and z-axis intersecting the geographic North Pole. It is denoted by {i}.

A.2 ECEF

The Earth-centered, Earth-fixed (ECEF) frame is another coordinate system with origin at the center of mass of the Earth. Its z-axis is also shared with the ECI reference frame, but the x- and y-axes intersects fixed point on the surface of the earth, meaning that it rotates with the Earth about the z-axis. As such, ECEF is not an inertial frame, but for slow moving craft it may be considered as such [7]. In this thesis, the ECEF frame of reference is denoted by {e}.

A.3 NED

The north-east-down frame is a location dependent system tangential to the surface of the Earth. The z-axis points down, along the normal of the Earth ellipsoid, while x- and y-axes points north and east, respectively. The NED frame is denoted {n}.

A.4 Body

The body-fixed reference frame is a moving coordinate system fixed to some craft. Both origin and orientation of the coordinate system is chosen freely . The body system is denoted {b}

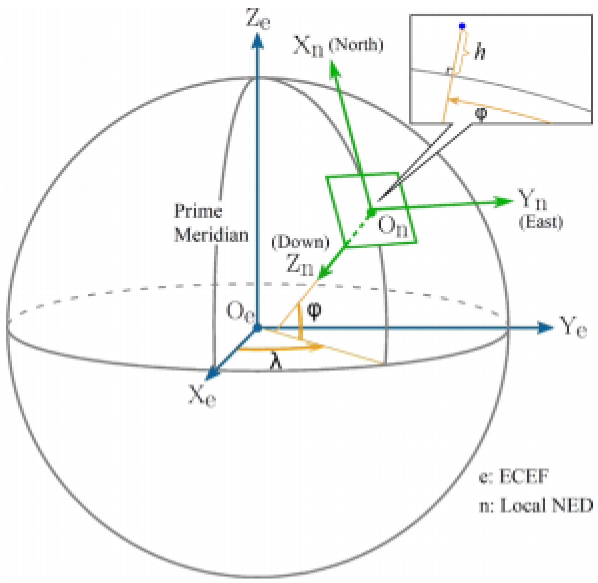

Figure A.1: NED and ECEF frames

A.5 Transformations between frames

A.5.1 NED and ECEF

According to [7], the rotation matrix from NED to ECEF is given by

$$R_n^e = \begin{bmatrix} -cos(l)sin(\mu) & -sin(l) & -cos(l)cos(\mu) \\ -sin(l)sin(\mu) & cos(l) & -sin(l)cos(\mu) \\ cos(\mu) & 0 & -sin(\mu) \end{bmatrix} \tag{A.1}$$

where l denotes latitude and μ denotes longitude.

A.5.2 Body and NED

Given a set of Euler angles $\begin{bmatrix}\phi & \theta & \psi\end{bmatrix}$ denoting the rotation from body to NED, a rotation matrix can be calculated as

$$\mathbf{R}_b^n = \begin{bmatrix} c(\psi)c(\theta) & -s(\psi)c(\phi)+c(\psi)s(\theta)s(\phi) & s(\psi)s(\phi)+c(\psi)c(\phi)s(\theta) \\ s(\psi)c(\theta) & c(\psi)c(\phi)+s(\phi)s(\theta)s(\psi) & -c(\psi)s(\phi)+s(\theta)s(\psi)c(\phi) \\ -s(\theta) & c(\theta)s(\phi) & c(\theta)c(\phi) \end{bmatrix} \tag{A.2}$$

Where c and s is shorthand for the cosine and sine functions, respectively. Alternatively, the rotation matrix can be calculated from the quaternion $\begin{bmatrix}\eta & \varepsilon_1 & \varepsilon_2 & \varepsilon_3\end{bmatrix}$

as

$$\mathbf{R}_b^n = \begin{bmatrix} 1-2(\varepsilon_2^2+\varepsilon_3^2) & 2(\varepsilon_1\varepsilon_2-\varepsilon_3\eta) & 2(\varepsilon_1\varepsilon_3+\varepsilon_2\eta) \\ 2(\varepsilon_1\varepsilon_2+\varepsilon_3\eta) & 1-2(\varepsilon_1^2+\varepsilon_3^2) & 2(\varepsilon_2\varepsilon_3-\varepsilon_1\eta) \\ 2(\varepsilon_1\varepsilon_3-\varepsilon_2\eta) & 2(\varepsilon_2\varepsilon_3+\varepsilon_1\eta) & 1-2(\varepsilon_1^2+\varepsilon_2^2) \end{bmatrix} \tag{A.3}$$

Appendix B

Atmospheric models

B.1 The Klobuchar ionospheric model

Given the user position in geodetic latitude ϕ_u, longitude λ_u, elevation angle E and azimuth A of the observed satellite, as well as model parameters α_n and β_n received from the almanac, the Klobuchar model computes the ionospheric delay I_{GPS} as follows.

$$\psi = \frac{0.0137}{E + 0.11} - 0.022 \tag{B.1a}$$

$$\phi_I = \phi_u + \psi A \tag{B.1b}$$

if $\phi_I > +0.416$, then $\phi_I = +0.416$

if $\phi_I < -0.416$, then $\phi_I = -0.416$

$$\lambda_I = \lambda_u + \frac{\psi sin(A)}{cos(\phi_I)} \tag{B.1c}$$

$$\phi_m = \phi_I + 0.064 cos(\lambda_I - 1.617) \tag{B.1d}$$

$$t = 43200\lambda_I + t_{GPS} \tag{B.1e}$$

Where $0 \leq t \leq 86400$. Therefore, if $t \geq 86400$, subtract 86400, if $t < 0$ add 86400

$$A_I = \sum_{n=0}^{3} \alpha_n \phi_m^n \tag{B.1f}$$

if $A_I < 0$, then $A_I = 0$

$$P_I = \sum_{n=0}^{3} \beta_n \phi_m^n \tag{B.1g}$$

if $P_I < 72000$, then $P_I = 72000$

$$X_I = \frac{2\pi(t - 50400}{P_I} \tag{B.1h}$$

$$F = 1.0 + 16.0(0.53 - E)^3 \tag{B.1i}$$

$$I_{GPS} = \begin{cases} (5 * 10^{-9} + \sum_{n=0}^{3} \alpha_n \phi_m^n)\left(1 - \frac{X_I^2}{2} + \frac{X_I^4}{24}\right)F & , |X_i| \leq 1.57 \\ 5 * 10^{-9} F & , |X_i| \geq 1.57 \end{cases} \tag{B.1j}$$

B.2 Saastamoinen tropospheric model

Given the user position in geodetic latitude ϕ_u, longitude λ_u, height h, elevation angle E, humidity hr and temperature T (in Kelvin). The tropospheric error, δ_t calculated by the Saastamoinen model is

if $h < -100$ or $h > 10^4$, $\delta_t = 0$

$$p = 1013.25(1 - 2.2557e^{-5})^{5.2568} \tag{B.1ka}$$

$$e = 6.108hr \exp^{\frac{17.15T - 4684}{T - 38.45}} \tag{B.1kb}$$

$$z = \frac{\pi}{2.0} - E \tag{B.1kc}$$

$$t_{dry} = \left(\frac{0.0022768p}{1 - 0.00266cos(2\phi_u)} - 0.00028h10^{-3}\right)/cos(z) \tag{B.1kd}$$

$$t_{wet} = \frac{2.857635e}{(T + 0.05)cos(z)} \tag{B.1ke}$$

$$\delta_t = t_{dry} + t_{wet} \tag{B.1kf}$$

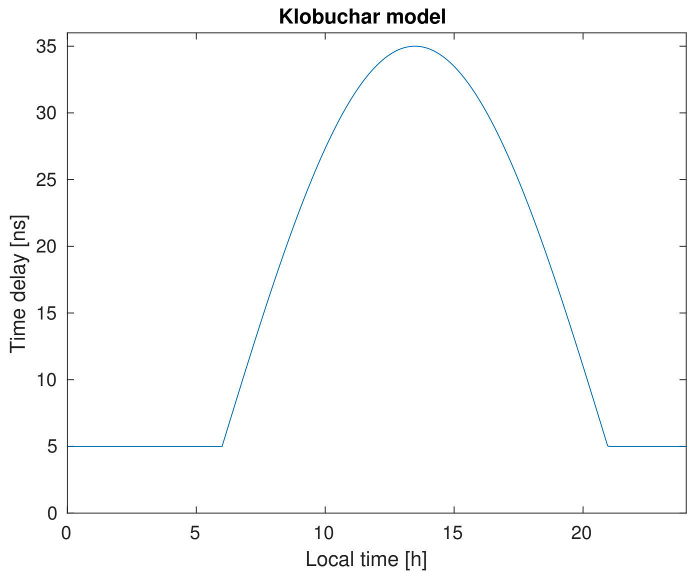

Figure B.1: This figure shows the concept of the klobuchar model. Note that this is not exact, but rather a simple approximation to show the concept of the model.

Bibliography

[1] The bds-3 preliminary system is completed to provide global services. `http://en.beidou.gov.cn/WHATSNEWS/201812/t20181227_16837.html`. Accessed: 2019-03-20.

[2] Galileo faqs. `https://www.gsc-europa.eu/helpdesk/faqs`. Accessed: 2019-03-20.

[3] Z. Baldysz, G. Nykiel, A. Araszkiewicz, M. Figurski, and K. Szafranek. Comparison of gps tropospheric delays derived from two consecutive epn reprocessing campaigns from the point of view of climate monitoring. *Atmospheric Measurement Techniques*, 9(9):4861, 2016.

[4] G. Di Giovanni and S. Radicella. An analytical model of the electron density profile in the ionosphere. *Advances in Space Research*, 10(11):27–30, 1990.

[5] G. Falco, M. Pini, and G. Marucco. Loose and tight gnss/ins integrations: Comparison of performance assessed in real urban scenarios. *Sensors*, 17(2):255, 2017.

[6] J. Farrell. *Aided navigation: GPS with high rate sensors*. McGraw-Hill, Inc., 2008.

[7] T. I. Fossen. *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.

[8] P. Freda, A. Angrisano, S. Gaglione, and S. Troisi. Time-differenced carrier phases technique for precise gnss velocity estimation. *GPS Solutions*, 19(2):335–341, 2015.

[9] H. F. Grip, T. I. Fossen, T. A. Johansen, and A. Saberi. Globally exponentially stable attitude and gyro bias estimation with application to gnss/ins integration. *Automatica*, 51:158–166, 2015.

[10] P. D. Groves. *Principles of GNSS, inertial, and multisensor integrated navigation systems*. Artech house, 2013.

[11] K. Gryte, J. M. Hansen, T. Johansen, and T. I. Fossen. Robust navigation of uav using inertial sensors aided by uwb and rtk gps. In *AIAA Guidance, Navigation, and Control Conference*, page 1035, 2017.

[12] J. M. Hansen, T. A. Johansen, N. Sokolova, and T. I. Fossen. Nonlinear observer for tightly coupled integrated inertial navigation aided by rtk-gnss measurements. *IEEE Transactions on Control Systems Technology*, (99):1–16, 2018.

[13] K. Hirahara. Local gps tropospheric tomography. *Earth, planets and space*, 52(11):935–939, 2000.

[14] H.-S. Kim, S.-C. Bu, G.-I. Jee, and C. G. Park. An ultra-tightly coupled gps/ins integration using federated kalman filter. In *ION GPS*, 2003.

[15] J. A. Klobuchar. Ionospheric time-delay algorithm for single-frequency gps users. *IEEE Transactions on aerospace and electronic systems*, (3):325–331, 1987.

[16] L. Li, J. Zhong, and M. Zhao. Doppler-aided gnss position estimation with weighted least squares. *IEEE Transactions on Vehicular Technology*, 60(8):3615–3624, 2011.

[17] F. L. Markley. Attitude error representations for kalman filtering. *Journal of guidance, control, and dynamics*, 26(2):311–317, 2003.

[18] D. Matsakis. The timing group delay (tgd) correction and gps timing biases. In *Proceedings of the 63rd Annual Meeting of the Institute of Navigation, Cambridge, MA,(Apr. 2007)*, 2007.

[19] P. Misra and P. Enge. Global positioning system: signals, measurements and performance second edition. *Massachusetts: Ganga-Jamuna Press*, 2006.

[20] A. Noureldin, T. B. Karamat, and J. Georgy. *Fundamentals of inertial navigation, satellite-based positioning and their integration*. Springer Science & Business Media, 2012.

[21] J.-F. Pascual-Sánchez. Introducing relativity in global navigation satellite systems. *Annalen der Physik*, 16(4):258–273, 2007.

[22] E. Priego, J. Jones, M. Porres, and A. Seco. Monitoring water vapour with gnss during a heavy rainfall event in the spanish mediterranean area. *Geomatics, Natural Hazards and Risk*, 8(2):282–294, 2017.

[23] A. Rones. Implementation aspects of a gnss based navigation system. 2017.

[24] T. Schuler. On ground-based gps tropospheric delay estimation. *Doctor's Thesis, Studiengang Geodsie und Geoinformation, Universitt der Buundeswehr Munchen*, 73, 2001.

[25] I. Shames, A. N. Bishop, M. Smith, and B. D. Anderson. Analysis of target velocity and position estimation via doppler-shift measurements. In *2011 Australian Control Conference*, pages 507–512. IEEE, 2011.

[26] I. Shames, A. N. Bishop, M. Smith, and B. D. Anderson. Doppler shift target localization. *IEEE Transactions on Aerospace and Electronic Systems*, 49(1):266–276, 2013.

[27] N. G. J. P. O. (SMC/GP). Navstar gps space segment/navigation user interfaces. Revision d, GPS JOINT PROGRAM OFFICE, 2006. https://www.gps.gov/technical/icwg/IS-GPS-200D.pdf.

[28] N. G. J. P. O. (SMC/GP). Navstar gps space segment/user segment l5 interfaces. Revision e, GPS JOINT PROGRAM OFFICE, 2018. https://www.gps.gov/technical/icwg/IS-GPS-705E.pdf.

[29] J. Sola. Quaternion kinematics for the error-state kalman filter. *arXiv preprint arXiv:1711.02508*, 2017.

[30] J. Stuelpnagel. On the parametrization of the three-dimensional rotation group. *SIAM review*, 6(4):422–430, 1964.

[31] Y. Tawk, P. Tomé, C. Botteron, Y. Stebler, and P.-A. Farine. Implementation and performance of a gps/ins tightly coupled assisted pll architecture using mems inertial sensors. *Sensors*, 14(2):3768–3796, 2014.

[32] A. Tridgell. pyublox. `https://github.com/tridge/pyUblox`, 2013.

[33] Ublox. *NEO/LEA-M8Tu-blox M8 concurrentGNSS timingmodules*, 06 2016. Rev. 3.

[34] C. Van Loan. Computing integrals involving the matrix exponential. *IEEE transactions on automatic control*, 23(3):395–404, 1978.

[35] J. Zhang, K. Zhang, R. Grenfell, and R. Deakin. Gps satellite velocity and acceleration determination using the broadcast ephemeris. *The Journal of Navigation*, 59(2):293–305, 2006.