Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Control Engineering

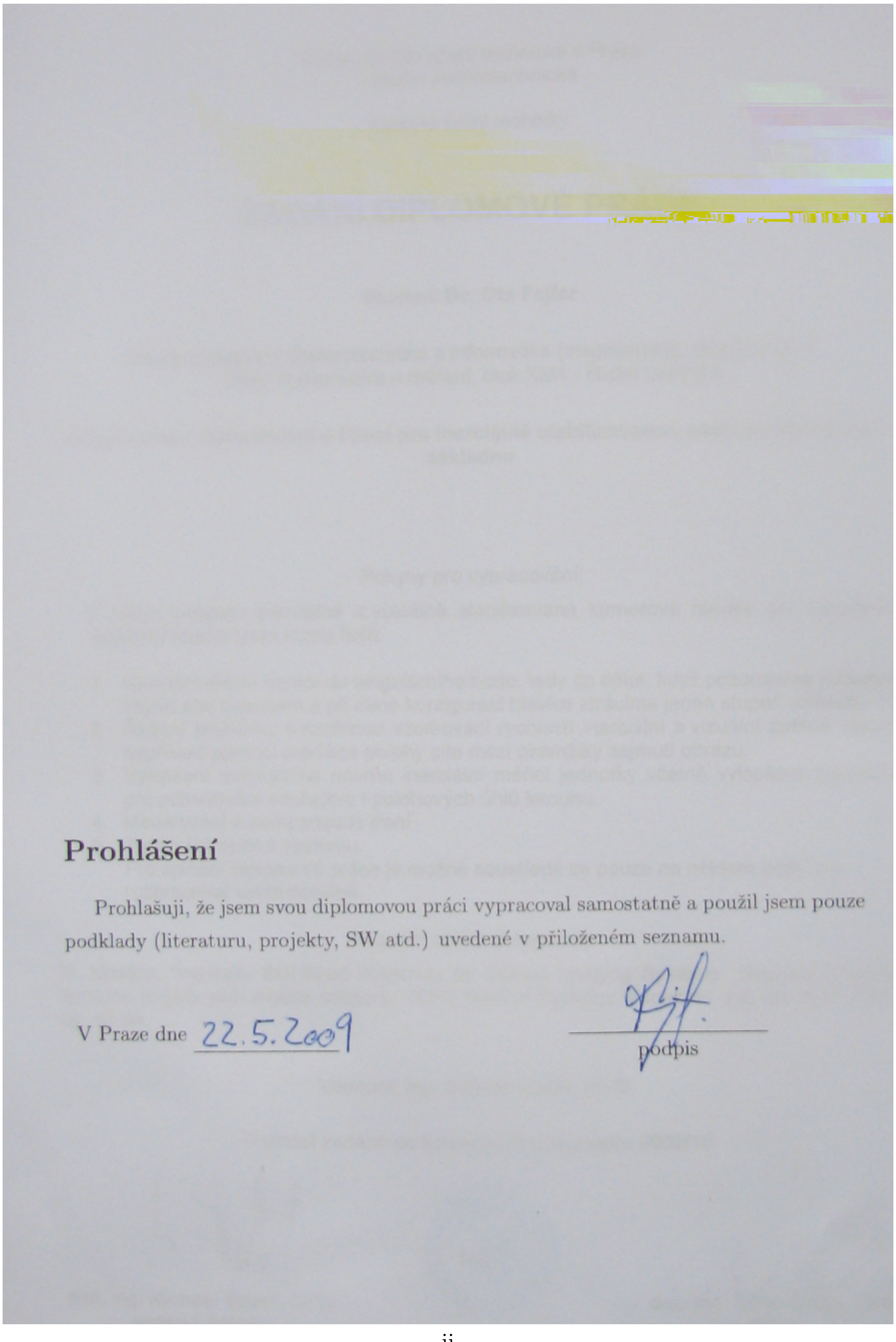
DIPLOMA THESIS

Estimation and control for inertially stabilized airborne camera platform

Prague, 2009

Author: Ota Fejfar

Author:	Bc. Ota Fejfar
Supervisor:	Ing. Zdeněk Hurák Ph.D.
	Department of Control Engineering,
	Faculty of Electrical Engineering,
	Czech Technical University in Prague
year:	2009



Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne 22.5.2009

[illegible]
podpis

Acknowledgements

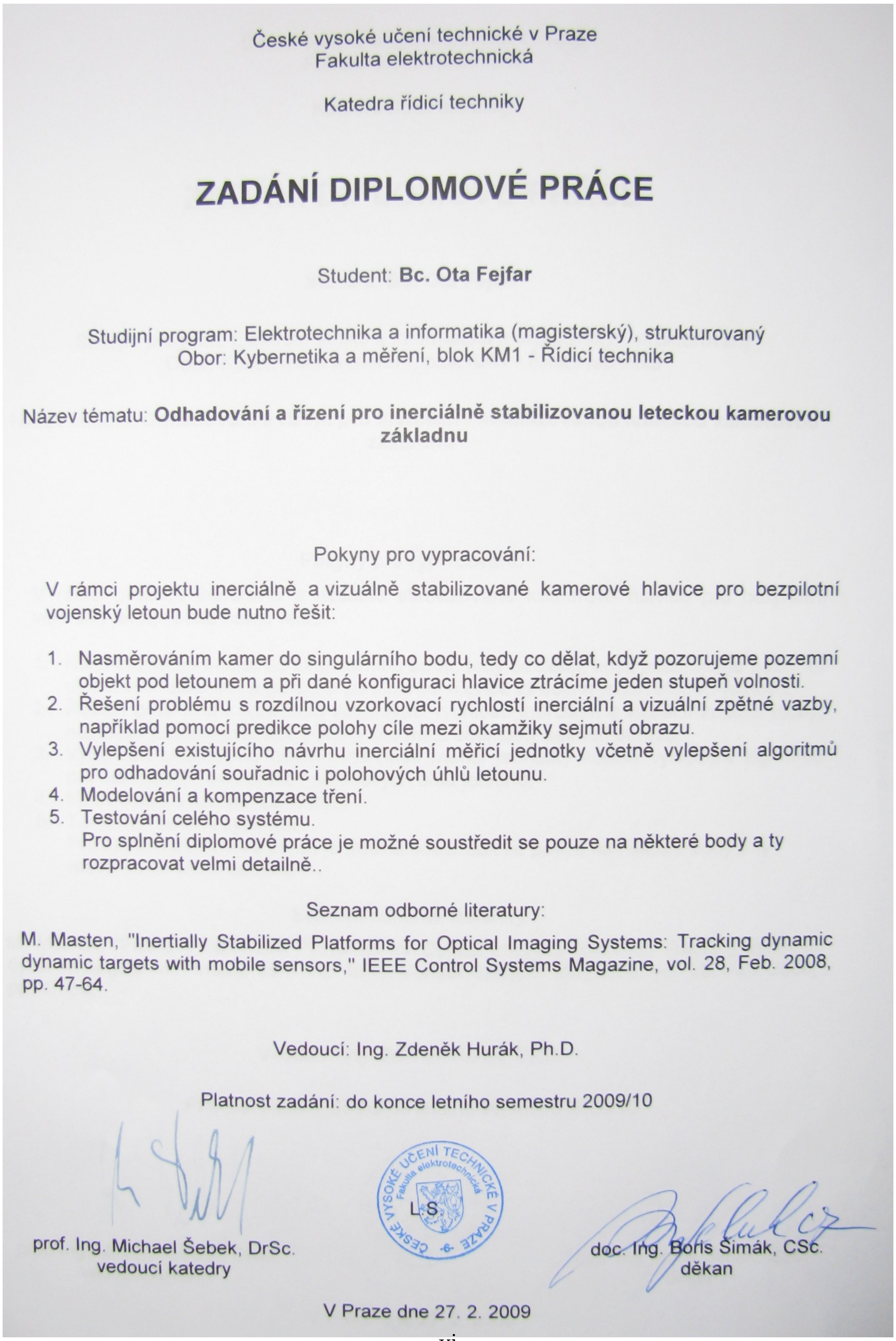
I would like to convey my gratitude to my supervisor Ing. Zdeněk Hurák Ph.D. for his patient and effective consulting of any problem at any time as well as for his great motivation abilities. I would also like to thank Martin Řezáč and Jaroslav Žoha who were always ready to help by a good advice. Finally, a great thanks comes to my family and my girlfriend Eva for their enduring support throughout my whole studies.

Abstract

The Center for Applied Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, solves the project of the stabilized camera platform in contract for Czech Airforce Technological Institute as a part of the project of Czech unmanned aerial vehicle Manta. In order to solve the problem of tracking an inertially moving object, the translational velocity of the platform carrier has to be known. For this purpose, the inertial navigation unit is being developed. It contains three-axis gyroscopes, accelerometers and magnetometers as well as a GPS module. This thesis solves the problem of velocity estimation using the data of the given sensors. To do this the inertial navigation unit simulator is first developed to provide the expected output sensor data as well as the data of all known true state signals. They are used for an extended Kalman filter design that estimates the actual state of the system. The estimated translational velocity is then included in the calculated state. At the end of the paper, the C-code for the used MCU LPC2119 is made and some hardware problems are being discussed.

Anotace

Projekt stabilizované základny pro kamerový systém je zpracováván týmem Centra pro aplikovanou kybernetiku, Fakulty elektrotechnické, Českého vysokého učení technického v Praze. Projekt byl zadán Vojenským technologickým ústavem letectva a je součástí projektu vývoje českého bezpilotního letounu Manta. Aktuálním úkolem je vyvinout řízení směru osy kamery tak, aby bylo možné sledovat objekt, který se vůči vztažné soustavě kamerového systému nějak pohybuje. K tomuto účelu je nutné znát aktuální translační rychlost nosiče kamerového systému v prostoru. Proto je vyvíjena inerciální navigační jednotka obsahující tříosé akcelerometry, gyroskopy, magnetometry a GPS modul. Tato diplomová práce popisuje řešení odhadu translační rychlosti odhadem stavu systému pomocí dat zmiňovaných senzorů. Nejprve je vyvinut simulátor inerciální navigační jednotky, který poskytuje jak očekávaná data výstupů senzorů tak i data všech známých stavů. Ty jsou použity při návrhu rozšířeného Kalmanova filtru, který odhaduje aktuální stav systému, kdy odhadovaná translační rychlost je jeho součástí. Na konci práce je vyvinut C-kód pro použité MCU LPC2119 a jsou diskutovány některé problémy s hardwarem, které se objevily během návrhu.



České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Ota Fejfar**

Studijní program: Elektrotechnika a informatika (magisterský), strukturovaný
Obor: Kybernetika a měření, blok KM1 - Řídicí technika

Název tématu: **Odhadování a řízení pro inerciálně stabilizovanou leteckou kamerovou základnu**

Pokyny pro vypracování:

V rámci projektu inerciálně a vizuálně stabilizované kamerové hlavice pro bezpilotní vojenský letoun bude nutno řešit:

1. Nasměrováním kamer do singulárního bodu, tedy co dělat, když pozorujeme pozemní objekt pod letounem a při dané konfiguraci hlavice ztrácíme jeden stupeň volnosti.
2. Řešení problému s rozdílnou vzorkovací rychlostí inerciální a vizuální zpětné vazby, například pomocí predikce polohy cíle mezi okamžiky sejmutí obrazu.
3. Vylepšení existujícího návrhu inerciální měřicí jednotky včetně vylepšení algoritmů pro odhadování souřadnic i polohových úhlů letounu.
4. Modelování a kompenzace tření.
5. Testování celého systému.
   Pro splnění diplomové práce je možné soustředit se pouze na některé body a ty rozpracovat velmi detailně..

Seznam odborné literatury:

M. Masten, "Inertially Stabilized Platforms for Optical Imaging Systems: Tracking dynamic dynamic targets with mobile sensors," IEEE Control Systems Magazine, vol. 28, Feb. 2008, pp. 47-64.

Vedoucí: Ing. Zdeněk Hurák, Ph.D.

Platnost zadání: do konce letního semestru 2009/10

[illegible]

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
L.S.

[illegible]

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 27. 2. 2009

Contents

1 Introduction **1**

1.1 Recent state of project 2

1.2 Motivation 3

1.3 The goals of thesis 4

2 Inertial navigation unit **5**

2.1 LPC2119 5

2.2 ADIS16350 6

2.3 HMC2003 7

2.4 NL-504ETTL 8

3 Mathematical modelling **9**

3.1 Coordinate frames 9

3.1.1 The ground coordinate frame 10

3.1.2 The plane coordinate frame 10

3.1.3 Transformation of coordinate frames 10

3.1.4 The dynamics of Euler angles 12

3.1.5 The restrictions on usage of Euler angles 13

3.2 Sensor models 13

3.2.1 Rate gyros 13

3.2.2 Accelerometers 14

3.2.3 GPS 15

3.2.4 Magnetometer compass 16

3.3 Extended Kalman filter 17

3.4 Cholesky decomposition 18

3.5 State space model 19

3.5.1 State variables 19

3.5.2 State equations 20
3.5.3 Output equations 22

4 Simulation **23**
4.1 INU simulator 23
4.2 Subsystems 27
4.3 Simulation data 28

5 EKF design **32**
5.1 EKF M-file 32
5.2 EKF tuning and results 35

6 Hardware **37**
6.1 Hardware programming 37
6.1.1 GPS data acquisition 37
6.1.2 Code optimization 39
6.2 Magnetometer data acquisition 40
6.3 Known hardware problems 40

7 Conclusion **42**

Bibliography **I**

A Wiring diagram **II**

B NMEA Protocol **IV**

C State space summary and Jacobians **VIII**
C.1 Nonlinear state space model IX
C.2 Jacobians XI

D Simulator subsystems **XIV**

E EKF and INU simulation results **XIX**
E.1 INU simulation outputs XIX
E.2 EKF simulation results XXI

F CD contents **XXIV**

List of Figures

1.1 UAV Manta 1
1.2 Camera platform 2

3.1 Coordinate frame. 9
3.2 Transformation of the ground coordinate frame into the plane coordinate frame. 11
3.3 State variables 20

4.1 INU Simulator 24
4.2 Block of Euler angles dynamics 27
4.3 Accelerometers subsystem block 28
4.4 Transformation blocks 28
4.5 Input data of the simulation 30
4.6 Output data of the simulation 31

5.1 Progress chart of EKF algorithm 33
5.2 EKF velocity estimation (extended INU data) 36

6.1 Magnetometer signal processing 40

E.1 Accelerometers and gyros outputs of the INU simulator XIX
E.2 Magnetometers and GPS outputs of the INU simulator XX
E.3 EKF results: Euler angles, trans. velocities and gravity vector XXI
E.4 EKF results: GPS, mag. intensity and angular rates XXII
E.5 EKF results: acceleration and offset estimates XXIII

List of Tables

2.1 Main gyroscope characteristics 6
2.2 Main accelerometer characteristics 7
2.3 Important HMC2003 characteristics 7
2.4 Important NL-504ETTL characteristics 8

4.1 Acquisition blocks summary 25
4.2 Simulator parameters 27
4.3 Settings of simulator parameters. 29

Nomenclature

INU	Inertial navigation unit
CAN	Controller-area network
CG	Center of gravity
CPU	Central processing unit
EKF	Extended Kalman filter
MCU	Microcontoller unit, microprocessor
SPI	Serial Peripheral Interface
UART	Universal asynchronous receiver/transmitter
UAV	Unmanned aerial vehicle

Chapter 1

Introduction

Figure 1.1: UAV Manta. (`http://dce.felk.cvut.cz/mamok`)

Manta (see fig. 1.1) is an unmanned aerial vehicle (*UAV*) being developed by Czech Air Force Technological Institute as a substitute for an older UAV Sojka which is bigger and weightier and hence less practical. The inertially stabilized camera platform is the component of UAV Manta. It is being developed by Czech Technical University in Prague in cooperation with Czech company ESSA in contract for Czech Air Force Technological Institute. The platform will be attached to UAV Manta and will have to stabilize the line of sight of its camera in needed direction. That is, no disturbance (wind, aircraft manoeuvring) would affect the line of sight of the camera, except the reference disturbance.

1.1 Recent state of project

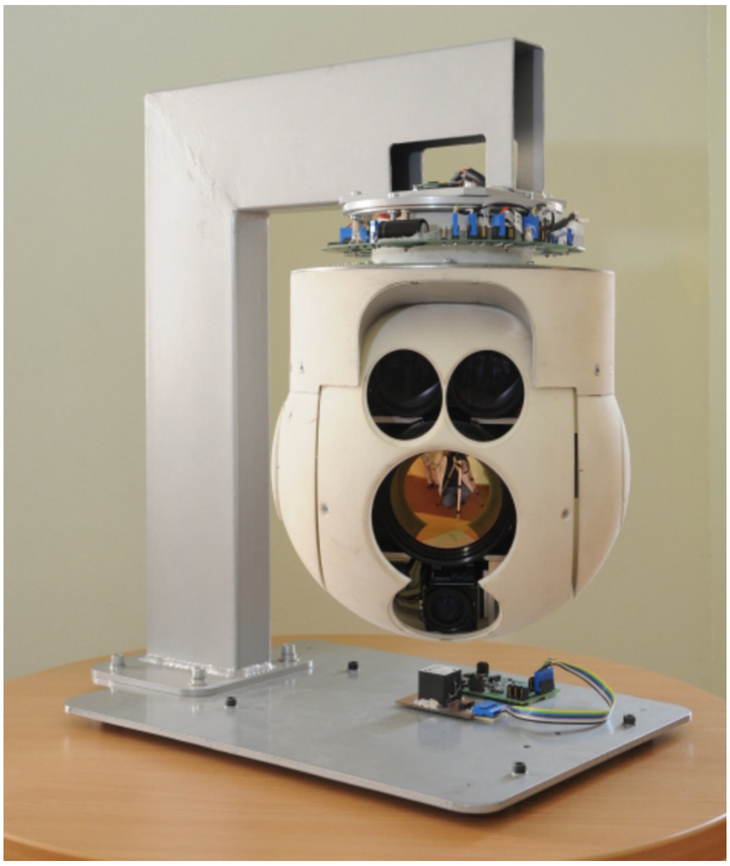


Figure 1.2: Camera platform. (`http://dce.felk.cvut.cz/mamok`)

The actual state of the project of stabilized platform (see fig. 1.2) in the beginning of this work is following:

- **The mechanical part of platform** is double gimbal Azimuth-Elevation (*Az-El*) system with two direct drive DC motors (developed by ESSA company). That is, the platform is two-axes rotational system sparing some space. There is, however, a singularity that must be counted with (solved by a restriction of usage).
- **The electronical system** (HW, basic SW) was completed within the thesis of Martin Řezáč and Jaroslav Žoha (Řezáč, M., 2008; Žoha, J., 2008*b*). Hardware is based on microprocessor LPC2119 processing the data of four gyroscopes. Note that the angular rates of the platform and a plane it is attached to may differ because the platform can rotate independently of the plane. Software processes an LQG controller compensating the effects of undesired disturbances mentioned above.

- **The basic hardware for inertial navigation unit (INU)** has been realized by Martin Řezáč and Jaroslav Žoha. The unit includes three-axes accelerometers, gyroscopes, magnetometers, attached to the plane axes in plane coordinate system (described below), and GPS. It is meant to give some other information for further design procedure.

- C code functions to obtain data from gyroscopes and accelerometers using MCU LPC2119 as well as the functions for matrix operations such as matrix mutiplication, transposition etc.

1.2 Motivation

As was written the camera platform is recently able to compensate the disturbances and hold the line of sight of the camera in constant direction thanks to the gyroscopes giving the angular rates information of camera platform. However, if the goal is to track some static or moving object, the gyros alone are not sufficient. Just imagine the situation when the UAV moves a specific velocity. The camera platform tries to track some object (no matter if it is a static or a moving one) and keep its image in the middle of the screen in headquarters. We can suppose the object image to move out of the screen as the aircraft moves along the real object. At this point we need to decide whether the image movement is caused by the different velocities of the aircraft and the tracked object or by rotations caused by some disturbances such as wind or by combination of the two mentioned causes. However, we have only gyroscopes giving us information about the rotations of camera platform but no information about the velocity of the plane. It is clear that we simply cannot decide the type of ”image movement” based only on the gyros. There is a significant need of obtaining the translational velocities of the aircraft itself.

For this purpose the INU mentioned above has been designed. It contains accelerometers, gyroscopes, magnetometers and GPS. The real velocity of the plane is supposed to be estimated by modern methods. It is because the signal of accelerometers contains some undesired subsignals (e.g. the acceleration of gravity) as well as the true translational acceleration of the aircraft. Offsets of some sensors have to be taken into an account too. A quality estimation design has to be made to acquire the real translational velocity.

Why three more gyroscopes?

As has been noted the existing gyros are attached to the camera platform which is able to rotate independently of the platform. It would be much more difficult to count the angular rates of the aircraft from the gyros of the platform and so the processing time of MCU is supposed to be much longer. The second argument is that the INU with its own gyroscopes will be more autonomous and hence we can use it in some another application.

1.3 The goals of thesis

The main goal of the thesis is to design adequate estimator of the translational velocity using a given INU. This task could be decomposed into a couple of subtasks:

- To model an expectable behavior of given sensors.
- To make a simulator giving data we can expect in the output of the sensors as well as the true data of the simulated motion.
- To design an estimator using simulated data.
- To obtain proper data through some real measurements.
- To adapt the estimator for the real data.
- To make some real navigation experiment using the designed INU.

Chapter 2

Inertial navigation unit

It is mentioned in section 1.1 that the hardware of the inertial navigation unit (INU) includes three-axis accelerometer, gyroscope and magnetometer as well as a GPS unit. The data from the output of those sensors are brought to pins of microprocessor LPC2119. In this section there is an overview of important electronics and used sensors as well as the description of their parameters especially of the important ones that have to be considered during the design of the velocity estimation. The complete information about the INU including a wiring diagram is in (Žoha, J., 2008*a*). The wiring diagram is also to be found in app. A.

2.1 LPC2119

LPC2119 is a 32-bit microcontroller unit (MCU) based on ARM core. The distributor of this MCU is Phillips company. It is a multifunctional control unit with loads of options. The main features that are important for this project are:

- Although the logic is based on 3.3V, there are up to forty-five 5V tolerant input pins.
- Two serial interfaces UART0 and UART1 and two SPIs.
- 128 kB on-chip Flash memory as well as 16 kB on-chip static RAM
- A maximum CPU clock frequency of 60 MHz.
- Two interconnected CAN interfaces

- Four channel 10-bit A/D converter with conversion time as low as 2.44 μs.
- Vectored Interrupt Controller with configurable priorities and vector addresses.
- Two 32-bit timers with prescale options and PWM unit with six outputs

For more information about LPC2119 see its manual (Phi, 2004).

2.2 ADIS16350

The ADIS16350 is a complete three-axis gyroscope and three-axis accelerometer system inertial sensing system. It provides the X-, Y- and Z-axis angular rate, the X-, Y- and Z-axis linear acceleration. All these data are accessible via the SPI port. The main features are:

- gyroscope digital range scaling $\pm 75\ ^o/s$, $\pm 150\ ^o/s$ and $\pm 300\ ^o/s$ settings with 14-bit resolution.
- accelerometer with $\pm 10g$ range and 14-bit resolution
- SPI-compatible serial interface
- supply voltage 4.75V to 5.25V

The important characteristics of embedded sensors are shown in tables 2.1 and 2.2. For more detailed information see (Ana, 2007-2008).

Parameter	**Conditions**	**Typical**	**Unit**
Initial sensitivity	25^{o}C, dynamic range = ±300 ^{o}/s	0.07326	^{o}/s/LSB
	25^{o}C, dynamic range = ±150 ^{o}/s	0.03663	^{o}/s/LSB
	25^{o}C, dynamic range = ±75 ^{o}/s	0.01832	^{o}/s/LSB
Output noise	25^{o}C, = ±300 ^{o}/s range, 2-tap filter setting	0.60	^{o}/s/ rms
	25^{o}C, = ±150 ^{o}/s range, 8-tap filter setting	0.35	^{o}/s/ rms
	25^{o}C, = ±75 ^{o}/s range, 32-tap filter setting	0.17	^{o}/s/ rms
3dB Bandwidth		350	Hz
Sensor Resonant Frequency		14	kHz

Table 2.1: Main gyroscope characteristics

Parameter	**Conditions**	**Typical**	**Unit**
Dynamic range		± 10	g
Initial sensitivity	25^oC	2.522	mg/LSB
Output noise	25^oC, no filtering	35	mg rms
3dB Bandwidth		350	Hz
Sensor Resonant Frequency		10	kHz

Table 2.2: Main accelerometer characteristics

2.3 HMC2003

HMC2003 is a high-sensitivity, three-axis magnetic sensor being distributed by Honeywell company. It is used to measure low magnetic field strengths. The main applications it may be used for are

- Precision compassing
- Navigation systems
- Attitude reference
- Traffic detection
- Proximity detection
- Medical devices

The tab. 2.3 contains the most important characteristics of HMC2003. For more details see (Hon, 2007). The outputs (the X-, Y- and Z-axis geomagnetic field strength) of the sensor is brought to A/D inputs of LPC2119 to be used for further processing.

Parameter	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Sensitivity		0.98	1	1.02	V/gauss
Null field output		2.3	2.5	2.7	V
Resolution			40		μgauss
Field range	Maximum magnetic flux density	-2		2	gauss
Output voltage	Each magnetometer axis output	0.5		4.5	V
Bandwidth			1		kHz

Table 2.3: Important HMC2003 characteristics

2.4 NL-504ETTL

NL-504ETTL is a complete GPS smart antenna receiver of NAVILOCK company. It consists of GPS receiver circuits and an antenna. The antenna is capable of tracking up to 32 satellites and provides

- fast time-to-first-fix
- default one-second navigation update
- low power consumption
- rapid satellite acquisition thanks to build-in micro battery and keeping reserve system data

This unit can be used for personal, automotive and marine positioning and navigation. The main characteristics of GPS receiver are summarized in the tab. 2.4. The output GPS data are in NMEA protocol form. The NMEA protocol consists of GGA, GLL, GSA, GSV, RMC and VTG "subprotocols". The complete NMEA protocol can be found in App. B. The output signal of NL-504ETTL is on TTL level and proceeds to UART1 RX/TX pins of LPC2119. The setting of the UART1 is 9600 bps, 8 data-bits, no parity, 1 stop-bit. For more details of NL-504ETTL see (Nav, 2007).

Frequency	L1 1575.42 MHz, C/A code
Channels	Support 32 channels
Update rate	1 Hz default, up to 5Hz
Acquisition time	Hot start (open sky) - 2s (typical) Cold start (open sky) - 36s (typical)
Position accuracy	Autonomous - 3m (2D RMS) SBAS - 2.5m (depends on accuracy of correction data)
Max. Altitude	$< 18{,}000$ m
Max. Velocity	< 515 m/s
Protocol support	NMEA 0183 ver 3.01

Table 2.4: Important NL-504ETTL characteristics

Chapter 3

Mathematical modelling

In this chapter, we discuss a mathematical preface of this thesis. We first establish coordinate frames and sensor models, then we make a brief introduction into extended Kalman filter and Cholesky decomposition. Finally, the state space model is being set.

3.1 Coordinate frames

There are several coordinate frames that are considered throughout the whole project (e.g. (Řezáč, M., 2008; Žoha, J., 2008*b*; Haniš, T., 2008)). For this part of the project it is not necessary to consider all of them. There are the *ground* and the *plane* coordinate frames being introduced in this section.

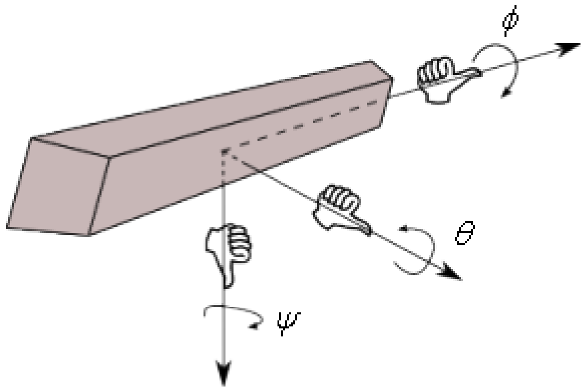


Figure 3.1: Plane coordinate frame.

The coordination frames are set according to the ISO norm (see fig. 3.1). All frames are completely described by their origin and three orthogonal vectors. A compact description is set as *oxyz*. For the positive angle of rotation applies the law of the right hand. That is, being shown in fig. 3.1), when the thumb points in the positive direction of the axis, the flexed fingers show the positive rotation around that axis.

3.1.1 The ground coordinate frame

There are several ways to interpret the ground coordinate frame $o_g x_g y_g z_g$. In this case, as it is supposed to use the GPS measurements, the x_g and y_g axes are identical to the ones used by GPS. Hence we use the frame that has its x_g and y_g axes oriented according to compass while satisfying the ISO norm. That is, the positive x_g axis points in the north direction, the positive y_g axis leads in the east direction and z_g axis is orthogonal to both of them and its positive direction points down into the middle of the Earth. The *longitude*, *latitude* and *altitude* of the origin o_g is 0.

3.1.2 The plane coordinate frame

The origin of the plane coordinate frame *oxyz* is laid in the center of gravity (CG) of the plane. The positive direct axis x leads through the nose of the plane, the positive cross axis y leads in the right wing direction and the third axis z is orthogonal to both x and y axes and its positive direction leads "down" through the belly of the plane. We will assume quietly throughout the whole thesis that the CG of INU and all its axes are the same as the plane ones.

3.1.3 Transformation of coordinate frames

At the beginning it must be said that by the term "*Coordinate frames transformation*" we mean the transformation of some vector $\overline{a}_g$ known in the ground coordinate frame into a vector $\overline{a}$ in the plane coordinate frame and vice versa. That is, as we need to transform a vector, we can assume $o = o_g$[1].

[1]Note that if we want to transform a **position** of some point in the plane frame (e.g. the position of target we observe) into the ground frame, we would have to assume the offset between these two frames $|o - o_g|$. Whenever we need only vector transformation (e.g. the velocity of the plane), the offset between frames is not needed to be assumed.

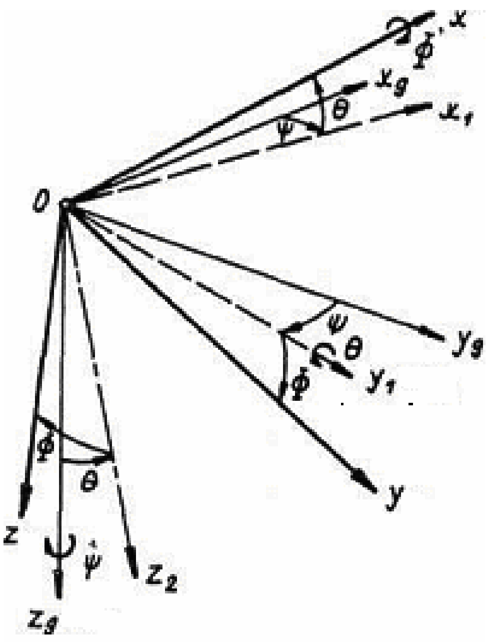


Figure 3.2: Transformation of the ground coordinate frame into the plane coordinate frame. Source: (PECH, Z. AND VĚK, V., 2003)

For the purpose of transformation itself we will use the concept of Euler angles (see fig. 3.2). We will first rotate the ground coordinate frame $G = ox_gy_gz_g$ around its z_g axis into a plane of symmetry[2] of an aircraft to obtain the yaw angle ψ and the new coordinate frame $A = ox_1y_1z_g$. We will now rotate the frame A around its y_1 axis into the direct axis of the aircraft. We have just got the pitch angle θ. The last step is to rotate given frame $B = oxy_1z_2$ around its x axis into the plane coordinate frame $P = oxyz$ to obtain the roll angle ϕ.

We need now to represent some ground coordinate frame vector $\overline{a}_g = [x, y, z]^T$ in the coordinates of the plane frame while we know the Euler angles ϕ, θ and ψ. According to (HURÁK, Z., 2008) we can use rotational matrix

$$\mathbf{R_G^A} = \begin{pmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.1}$$

which can be expressed as "the rotational matrix expressing the coordinates of some vector in ground frame within A frame". To complete the transformation we use yet

[2] *The plane of symmetry of an aircraft* is the plane given by x and z axes in the plane coordinate frame.

following two matrices:

$$\mathbf{R_A^B} = \begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{pmatrix} \tag{3.2}$$

$$\mathbf{R_B^P} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{pmatrix} \tag{3.3}$$

Let's make following substitutions:

$$\begin{aligned} c_\psi &= \cos(\psi), \quad s_\psi = \sin(\psi), \\ c_\phi &= \cos(\phi), \quad s_\phi = \sin(\phi), \\ c_\theta &= \cos(\theta), \quad s_\theta = \sin(\theta), \end{aligned} \tag{3.4}$$

the expression of the given ground frame vector within the plane frame is then

$$\overline{a} = \mathbf{R_G^P} \cdot \overline{a}_g = \mathbf{R_B^P} \cdot \mathbf{R_A^B} \cdot \mathbf{R_G^A} \cdot \overline{a}_g \tag{3.5}$$

$$\mathbf{R_G^P} = \begin{pmatrix} c_\psi c_\theta & s_\psi c_\theta & -s_\theta \\ c_\psi s_\theta s_\phi - s_\psi c_\phi & s_\psi s_\theta s_\phi + c_\psi c_\phi & c_\theta s_\phi \\ c_\psi s_\theta c_\phi + s_\psi s_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi & c_\theta c_\phi \end{pmatrix} \tag{3.6}$$

As this rotation and its inversion are the only rotations we will use, let's simplify the notation: $\mathbf{R_G^P} = \mathbf{R}$.

According to (WWW.MATHPAGES.COM, 2008) an interesting property of rotation matrix is $\mathbf{R^{-1}} = \mathbf{R^T}$. That is why we will use

$$\begin{aligned} \overline{a} &= \mathbf{R} \cdot \overline{a}_g \\ \overline{a}_g &= \mathbf{R}^T \cdot \overline{a}. \end{aligned} \tag{3.7}$$

3.1.4 The dynamics of Euler angles

Assume now, that we have got rotation rates of the aircraft (e.g. an output of ideal gyroscopes). The rotation rates are the roll rate p, the pitch rate q and the yaw rate r. According to (BEARD, R. W., 2007) the computation of Euler angles (depending on rotation rates) leads to three differential equations

$$\begin{aligned} \dot{\phi} &= p + q\sin(\phi)\tan(\theta) + r\cos(\phi)\tan(\theta) \\ \dot{\theta} &= q\cos(\phi) - r\sin(\phi) \\ \dot{\psi} &= q\tfrac{\sin(\phi)}{\cos(\theta)} + r\tfrac{\cos(\phi)}{\cos(\theta)} \end{aligned} \tag{3.8}$$

3.1.5 The restrictions on usage of Euler angles

Looking at eq. (3.8) it is obvious that there are some singularities due to some cosines of pitch angle θ[3] in denominators of some fractions. This singularity applies whenever the x axis is oriented vertically. That is, the nose of aircraft points into the center of the Earth or contrariwise.

We can solve this problem by setting the restriction on the pitch angle near $\pm\pi$. As the UAV Manta is supposed to fly mainly in some horizontal flight with seldom manoeuvres, the restriction should not be difficult to fulfil.

The advantage of Euler angles is definitely an easy imagination of the real aircraft angular position. Thanks to that, the Euler angles are often used as the basic interpretation of the angular position of an observed object. The disadvantage, the singularity, was mentioned above. To solve the singularity problem, the solution may be to use quaternions. The quaternions are insensitive to singularities and allow us to observe the full scale of manoeuvring but the imagination of the angular position is much more difficult. For the purposes of this thesis, the Euler angles will be sufficient.

3.2 Sensor models

This section defines mathematical models of all sensors that are mounted and used within the INU. It is important to do this definition at this point because the sensor models will be mainly used in following chapters. The first great usage is in abstract simulation unit to produce data for further estimator design (see chapter 4). The second purpose of sensor models usage is their use in the state estimator design itself (see sec. 3.5). The sensor models come mainly from (Beard, R. W., 2007) although they may be a little modified to suit the current problem.

3.2.1 Rate gyros

A gyroscope angular rate sensor contains a small vibrating lever. When the lever undergoes an angular rotation, the frequency of vibrations changes due to the Coriolis force. Thanks to this effect, the rotation is detected. Generally the output of a rate gyro

[3]Note that $\tan\theta = \frac{\sin(\theta)}{\cos(\theta)}$.

is:

$$y_{gyro} = k_{gyro}\omega + \beta_{gyro}(T) + \eta_{gyro}, \tag{3.9}$$

where y_{gyro} is the signal that we measure at the output of gyro sensor in Volts, k_{gyro} is a gain, ω is the true angular rate, β_{gyro} is a component that depends on temperature and acts as some kind of offset in the output signal, and η_{gyro} is a zero mean Gaussian process with known variance.

There are three rate gyros being used in this application. They are aligned along the x, y and z axes and support us with the roll, pitch and yaw rate data p, q and r. Assuming eq. (3.9) we have

$$\begin{aligned} y_{gyro,x} &= k_{gyro,x}p + \beta_{gyro,x}(T) + \eta_{gyro,x} \\ y_{gyro,y} &= k_{gyro,y}q + \beta_{gyro,y}(T) + \eta_{gyro,y} \\ y_{gyro,z} &= k_{gyro,z}r + \beta_{gyro,z}(T) + \eta_{gyro,z}. \end{aligned} \tag{3.10}$$

3.2.2 Accelerometers

The principle of an accelerometer lies in a small plate that is flexibly attached to the body of the sensor. When the whole system accelerates or decelerates the inertial properties of the system forces the small plate to change the position against the body of the sensor. The capacitance between the plate and the body is then changed and this change can be detected. The output of an accelerometer is:

$$y_{acc} = k_{acc}a + \beta_{acc}(T) + \eta_{acc}, \tag{3.11}$$

where y_{acc} is in $[V]$, k_{acc} is a gain, a is an acceleration in $[m/s^2]$, β_{acc} is a component that depends on temperature and η_{acc} is a zero mean Gaussian process with known variance.

However, the acceleration a in (3.11) still consists of more components such is the gravity acceleration and centrifugal acceleration. These two more components have also to be considered to acquire the pure translational acceleration. In the three-dimensional system we have

$$\begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} = \frac{1}{m}\left(\overline{F} - \overline{F}_{gravity}\right) = \dot{\overline{\mathrm{v}}} + \overline{\omega} \times \overline{\mathrm{v}} - \frac{1}{m}\overline{F}_{gravity}, \tag{3.12}$$

while the last component of (3.12) is actually the fixed gravitation vector $\overline{g}_g = [0, 0, g]^T$ in ground coordinate frame expressed within the coordinates of the plane coordinate frame.

Using the (3.7) and (3.6) we can write

$$\frac{1}{m}\overline{F}_{gravity} = \overline{g} = \mathbf{R}\overline{g}_g = \begin{pmatrix} -g\sin\theta \\ g\cos\theta\sin\phi \\ g\cos\theta\cos\phi \end{pmatrix}. \tag{3.13}$$

Then, from (3.12) and (3.13), we have the component form of the accelerometer model:

$$\begin{aligned} a_x &= \dot{u} + qw - rv + g\sin\theta \\ a_y &= \dot{v} + ru - pw - g\cos\theta\sin\phi \\ a_z &= \dot{w} + pv - qu - g\cos\theta\cos\phi, \end{aligned} \tag{3.14}$$

where the u, v and w components are the true translational velocities along the x, y and z axes and $\dot{u}$, $\dot{v}$ and $\dot{w}$ are the appropriate true accelerations. From (3.11) the output of the three axes accelerometer is

$$\begin{aligned} y_{acc,x} &= \dot{u} + qw - rv + g\sin\theta + \beta_{acc,x}(T) + \eta_{acc,x} \\ y_{acc,y} &= \dot{v} + ru - pw - g\cos\theta\sin\phi + \beta_{acc,y}(T) + \eta_{acc,y} \\ y_{acc,z} &= \dot{w} + pv - qu - g\cos\theta\cos\phi + \beta_{acc,z}(T) + \eta_{acc,z}. \end{aligned} \tag{3.15}$$

3.2.3 GPS

There is a set of GPS signal errors in (Beard, R. W., 2007) being described in detail. These errors are caused by atmosphere conditions, changing satellite geometry, clock drift, multipath signals and measurement error. For this application we will assume a combined measurement zero mean error η_{gps} instead of the mentioned ones. The output of the GPS will then be

$$\begin{aligned} y_{gps,n} &= p_n + \eta_{gps,n} \\ y_{gps,e} &= p_e + \eta_{gps,e} \\ y_{gps,h} &= h + \eta_{gps,h}, \end{aligned} \tag{3.16}$$

where p_n, p_e and h are the actual earth coordinates and altitude above sea level respectively. In the further design we will also take into an account, that the altitude error $\eta_{gps,h}$ is much greater than position errors $\eta_{gps,n}$ and $\eta_{gps,e}$.

We can also obtain GPS data per degree/minute/second in meters by assuming that 1 latitudinal degree is about 110.9km, one latitudinal minute is about 1849m and one latitudinal second measures 31.82m. The latitude-meters transfer relation is constant. The longitude-meters relation depends on the actual latitude position. That is

$$p_{e,const} = \cos(\alpha)p_{n,const} \tag{3.17}$$

where $p_{e,const}$ is the longitude-meters transfer constant for a given latitude position, α is the actual latitude position and $p_{n,const}$ is the latitude-meters transfer constant. As it is known that the UAV, given by its flight range, will be used more or less on the constant latitude, from now on, we will assume that the data of GPS are acquired in meters.

3.2.4 Magnetometer compass

A sensitive magnetometer can measure the earth's magnetic field intensity. The earth's magnetic field intensity can be represented by three-axes vector $\overline{H} = [H_x, H_y, H_z]^T$ in ground coordinate frame. The intensity may be predicted from the actual position on the earth, but it can be affected, for example, by earth's geological characteristics, e.g. by iron deposits. However, it is almost constant for a given area (tens of kilometers). For a larger area (more than hundreds of kilometers), the current intensity vector should be updated. For this purpose, for example, the magnetic intensity map could be used. As the magnetometer measuring the earth's magnetic field is very sensitive, it should be avoided to use any magnetic field disturbing objects, such as screwdrivers and so on, in magnetometer's surroundings during the precise measurements.

The magnetic intensity vector in Prague on 1st May 2009 was

$$\overline{H}_{g0} = \begin{pmatrix} H_{g0,x} \\ H_{g0,y} \\ H_{g0,z} \end{pmatrix} = \begin{pmatrix} 19,882.1 \\ 976.4 \\ 44,602.2 \end{pmatrix} [nT] = \begin{pmatrix} 0.198821 \\ 0.009764 \\ 0.446022 \end{pmatrix} [gauss] \tag{3.18}$$

To get magnetic vector expressed in the plane coordinate frame we simply use the rotation matrix from (3.7). Hence the output of the magnetometer is expected in the following form:

$$\begin{aligned} y_{mag,x} &= H_{g0,x} c_\psi c_\theta + H_{g0,y} s_\psi c_\theta - H_{g0,z} s_\theta + \beta_{mag,x}(T) + \eta_{mag,x} \\ y_{mag,y} &= H_{g0,x}(c_\psi s_\theta s_\phi - s_\psi c_\phi) + H_{g0,y}(s_\psi s_\theta s_\phi + c_\psi c_\phi) - H_{g0,z}(c_\theta s_\phi) + \beta_{mag,y}(T) + \eta_{mag,y} \\ y_{mag,y} &= H_{g0,x}(c_\psi s_\theta c_\phi + s_\psi s_\phi) + H_{g0,y}(s_\psi s_\theta c_\phi - c_\psi s_\phi) - H_{g0,z}(c_\theta c_\phi) + \beta_{mag,z}(T) + \eta_{mag,z} \end{aligned} \tag{3.19}$$

where $y_{mag,.}$ is in $[V]$, $\beta_{mag,.}$ is a component that depends on temperature and $\eta_{mag,.}$ is a zero mean Gaussian process with known variance.

3.3 Extended Kalman filter

State estimation using Kalman filter is a complex problem. A lot about Kalman filter can be found in literature. In this section, there is a very brief description of the most important part of extended Kalman filter (EKF) that is being used by this project. The main idea to use the EKF in this application comes from (Beard, R. W., 2007), although in this case the EKF estimates **all** mentioned states of current system.

Extended Kalman filter is a method of state estimation of nonlinear stochastic systems described by

$$\begin{aligned} \dot{x}(t) &= f(x,u,t) + \xi(t) \\ y(t) &= g(x,u,t) + \eta(t). \end{aligned} \tag{3.20}$$

We can obtain state space matrices A and C, required for an update of the covariance matrix P, by computing Jacobians of $f(x,u,t)$ and $g(x,u,t)$:

$$\begin{aligned} A(x) &= \tfrac{\partial f(x,u,t)}{\partial x} \\ C(x) &= \tfrac{\partial g(x,u,t)}{\partial x} \end{aligned} \tag{3.21}$$

Before the algorithm starts, the initialization of state estimate $\hat{x}(t)$ and **symmetric** covariance matrices P, Q and R has to be made. The algorithm itself can then be divided into two processes. **The data (filtration) step** is done each time the data come from sensors. This process contains following operations

$$\begin{aligned} C_i &= \tfrac{\partial g_i}{\partial x}(\hat{x}) && --\ \textit{Jacobian} \\ L_i &= PC_i^T(C_iPC_i^T + R)^{-1} && --\ \textit{Kalman's gain} \\ e_i &= y_i - g(\hat{x},u,t) && --\ \textit{estimation error} \\ \hat{x} &= \hat{x} + L_ie_i && --\ \textit{update state estimate} \\ P &= P - PC_i^T(C_iPC_i^T + R)^{-1}CP = P - L_iC_iP && --\ \textit{update covariance matrix P} \end{aligned} \tag{3.22}$$

That is, we first compute Jacobian C, Kalman's gain L and state estimate error e. Then we do the state estimate computation and finish by updating covariance matrix P.

The time (prediction) step is done continuously and depends on the sample period T_{sam}. Hence the system must be discretized for example by

$$x(t+T) = T_{sam}\dot{x} + x(t) \tag{3.23}$$

The operations that are included within an iteration of prediction process are

$$\begin{array}{lll} A_i = \frac{\partial f_i}{\partial x}(\hat{x}) & -- \textit{Jacobian} \\ \hat{x} \;\; = \hat{x} + T_{sam} f(\hat{x}, u, t) & -- \textit{update state estimate} \\ P \;\; = A_i P A_i^T + Q & -- \textit{update covariance matrix } P \end{array} \tag{3.24}$$

That is, we first compute the Jacobian A and state estimate $\hat{x}$ and then we update covariance matrix P.

3.4 Cholesky decomposition

Cholesky decomposition allows us to express some symmetric matrix M as a product

$$M = LL^T, \tag{3.25}$$

where L is a lower triangular matrix. If matrix A is positive-definite, the Cholesky decomposition is unique.

While $i \geq j$, the elements of M matrix are

$$\begin{array}{l} L_{i,i} = \sqrt{M_{i,i} - \sum_{k=1}^{i-1} L_{i,k}^2} \\ L_{i,j} = \frac{1}{L_{j,j}} \left(M_{i,j} - \sum_{k=1}^{j-1} L_{i,k} L_{j,k} \right) \end{array} \tag{3.26}$$

There may be a problem when any of the diagonal elements of L are zero. We will solve this problem by setting whole appropriate column to zero. That is,

$$\texttt{If} \;\; L_{j,j} == 0 \;\; \texttt{than} \;\; L_{i,j} = 0, \forall i > j.$$

Cholesky decomposition can be used to solve

$$xA = B \tag{3.27}$$

for x while $A = LL^T$ is a symmetric matrix. Using Cholesky decomposition we have

$$\begin{array}{r} xLL^T = B \\ yL^T = B \\ xL = y \end{array} \tag{3.28}$$

where we first compute elements of y

$$y_{i,j} = \frac{1}{L_{j,j}} (b_{i,j} - \sum_{k=1}^{j-1} y_{i,k} L_{j,k}) \tag{3.29}$$

and finally we obtain x by

$$x_{i,j} = \frac{1}{L_{j,j}} (y_{i,j} - \sum_{k=1}^{n} x_{i,k} L_{k,j}). \tag{3.30}$$

3.5 State space model

In this section, there is whole state model for EKF being described. We first describe state variables, then we set the model in the (3.20) form, i.e. we will describe the state and output equations. Note that in this section described model has got no inputs because we do not have either any direct information about thrust, ailerons, elevators and rudders of UAV or their models. The following estimation is whole based on the observations of the sensor outputs. Both complete state space model and the appropriate Jacobians can be found in app. C.

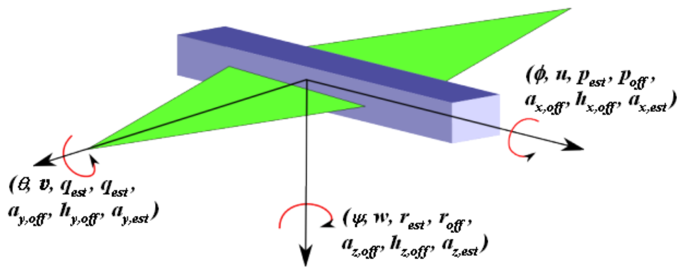
3.5.1 State variables

As has been described in chapter 2, the INU includes three-axis accelerometer, three-axis gyroscope and three-axis magnetometer as well as a GPS unit. That is, we are able to obtain data of translational accelerations, rotation rates and the vector of Earth's magnetic intensity in the aircraft coordinate frame as well as the actual position in ground coordinate frame thanks to GPS. We can also assume that accelerometer, gyroscope and magnetometer data includes some offset that is mentioned in section 3.2 as $\beta(T)$ component of equations (3.9), (3.11) and (3.19) describing general sensor models. Through this information we are able to derive state variables that are listed below:

ϕ = the roll Euler angle [rad],
θ = the pitch Euler angle [rad],,
ψ = the yaw Euler angle [rad],,
u = the x-axis translational velocity [m/s],
v = the y-axis translational velocity [m/s],
w = the z-axis translational velocity [m/s],
p_n = the inertial north position of the UAV [m],
p_e = the inertial east position of the UAV [m],
h = the altitude of the UAV [m],

p_{est}	$=$	the estimation of the true roll rate [rad/s],
q_{est}	$=$	the estimation of the true pitch rate [rad/s],
r_{est}	$=$	the estimation of the true yaw rate [rad/s],
p_{off}	$=$	the estimation of the x-axis gyroscope offset [rad/s],
q_{off}	$=$	the estimation of the y-axis gyroscope offset [rad/s],
r_{off}	$=$	the estimation of the z-axis gyroscope offset [rad/s],
$a_{x,off}$	$=$	the estimation of the x-axis accelerometer offset [m/s^2],
$a_{y,off}$	$=$	the estimation of the y-axis accelerometer offset [m/s^2],
$a_{z,off}$	$=$	the estimation of the z-axis accelerometer offset [m/s^2],
$h_{x,off}$	$=$	the estimation of the x-axis magnetometer offset [gauss],
$h_{y,off}$	$=$	the estimation of the y-axis magnetometer offset [gauss],
$h_{z,off}$	$=$	the estimation of the z-axis magnetometer offset [gauss],
$a_{x,est}$	$=$	the estimation of the true x-axis translational acceleration [m/s^2],
$a_{y,est}$	$=$	the estimation of the true y-axis translational acceleration [m/s^2],
$a_{z,est}$	$=$	the estimation of the true z-axis translational acceleration [m/s^2].

In fig. 3.3 we depict some UAV state variables for a better conception of what axis do the state variables belong to.



$(\phi, u, p_{est}, p_{off}, a_{x,off}, h_{x,off}, a_{x,est})$
$(\theta, v, q_{est}, q_{est}, a_{y,off}, h_{y,off}, a_{y,est})$
$(\psi, w, r_{est}, r_{off}, a_{z,off}, h_{z,off}, a_{z,est})$

Figure 3.3: State variables.

3.5.2 State equations

We will now assemble the state space model. The basics have been set in section 3.2. We are going to extend those terms, make some modifications and complete the state space model that will be used for Kalman filter design. Let's start with **Euler angles**

dynamics. Assuming the eq. (3.8) we have

$$\begin{aligned}\dot{\phi} &= p_{est} + q_{est}\sin(\phi)\tan(\theta) + r_{est}\cos(\phi)\tan(\theta)\\ \dot{\theta} &= q_{est}\cos(\phi) - r_{est}\sin(\phi)\\ \dot{\psi} &= q_{est}\tfrac{\sin(\phi)}{\cos(\theta)} + r_{est}\tfrac{\cos(\phi)}{\cos(\theta)}\,.\end{aligned} \tag{3.31}$$

The dynamics of **translational velocities** u, v, and w can be derived from (3.15) by expressing $\dot{u}$, $\dot{v}$ and $\dot{w}$:

$$\begin{aligned}\dot{u} &= -q_{est}w + r_{est}v - g\sin\theta + a_{x,est}\\ \dot{v} &= -r_{est}u + p_{est}w + g\cos\theta\sin\phi + a_{y,est}\\ \dot{w} &= -p_{est}v + q_{est}u + g\cos\theta\cos\phi + a_{z,est}.\end{aligned} \tag{3.32}$$

As we have the equations for Euler angles and translational velocities, it is straightforward to obtain also the differential equations describing **the position of INU** in ground coordinate frame.

$$\begin{pmatrix}\dot{p}_n\\ \dot{p}_e\\ \dot{h}\end{pmatrix} = \mathbf{R}^T\begin{pmatrix}u\\ v\\ w\end{pmatrix} \tag{3.33}$$

where $\mathbf{R}$ is the rotation matrix from 3.6. Hence the differential equations for position of INU are

$$\begin{aligned}\dot{p}_n &= u(c_\psi c_\theta) + c(c_\psi s_\theta s_\phi - s_\psi c_\phi) + w(c_\psi s_\theta c_\phi + s_\psi s_\phi)\\ \dot{p}_e &= u(s_\psi c_\theta) + v(s_\psi s_\theta s_\phi + c_\psi c_\phi) + w(s_\psi s_\theta c_\phi - c_\psi s_\phi)\\ \dot{h} &= -u(s_\theta) + v(c_\theta s_\phi) + w(c_\theta c_\phi).\end{aligned} \tag{3.34}$$

The remaining state variables, that we have to set differential equations for, are **the estimates of true accelerations and angular rates data as well as offsets of accelerometers, gyroscopes and magnetometers**. The basic idea, allowing us to obtain the true signal and to filter offsets consists in presumption that an output signal of sensor consists of the true and the offset signal, i.e. their sum respectively

$$y_{sens} = y_{true} + y_{off}. \tag{3.35}$$

The motivation for the decomposition of the sensor signal is that we need to obtain the estimation of the true signal $y_{true} = x_{est}$ and thus to eliminate the error of the estimation produced by the offset $y_{off} = x_{off}$. We do not know anything about those two decomposed signals except their sum, but, fortunately, we can presume variances of their noises ξ_{est} and ξ_{off}. Hence the differential equations for x_{est} and x_{off} are

$$\begin{aligned}\dot{x}_{est} &= \xi_{est}\\ \dot{x}_{off} &= \xi_{off}\end{aligned} \tag{3.36}$$

As we know that offset signal either changes very slowly or does not change at all, we can set $\xi_{est} >> \xi_{off}$ or $\xi_{est} \rightarrow 0$ respectively. That is, we can set the offset noise signal ξ_{off} to zero and assume the variance of the estimate signal x_{est} (at the very beginning of the estimator design) to be the same as the variance of sensor output signal y_{sens}. Note that the sensor signal variance could be obtained from the sensor datasheet list.

3.5.3 Output equations

The output of state space model contains all signals, or their combination respectively, that can be measured by INU sensors. As we actually estimate the GPS data directly, the output of GPS estimate is

$$\begin{aligned} y_{p_n} &= p_n \\ y_{p_e} &= p_e \\ y_h &= h \end{aligned} \tag{3.37}$$

The output of gyroscopes and accelerometers consists of two signals as is explained in subsection 3.5.2. The appropriate outputs are then

$$\begin{aligned} y_p &= p_{est} + p_{off} \\ y_r &= q_{est} + q_{off} \\ y_q &= r_{est} + r_{off} \\ y_{a_x} &= a_{x,est} + a_{x,off} \\ y_{a_y} &= a_{y,est} + a_{y,off} \\ y_{a_z} &= a_{z,est} + a_{z,off} \end{aligned} \tag{3.38}$$

And finally the outputs that correspond to the magnetometers are obtained by rotation of the known vector $\overline{H}_{g0}$ from 3.18 using the $\mathbf{R}$ matrix from 3.6

$$\begin{aligned} y_{h_x} &= h_x(c_\psi c_\theta) + h_y(s_\psi c_\theta) - h_z(s_\theta) + h_{x,off} \\ y_{h_y} &= h_x(c_\psi s_\theta s_\phi - s_\psi c_\phi) + h_y(s_\psi s_\theta s_\phi + c_\psi c_\phi) + h_z(c_\theta s_\phi) + h_{y,off} \\ y_{h_z} &= h_x(c_\psi s_\theta c_\phi + s_\psi s_\phi) + h_y(s_\psi s_\theta c_\phi - c_\psi s_\phi) + h_z(c_\theta c_\phi) + h_{z,off}. \end{aligned} \tag{3.39}$$

Chapter 4

Simulation

The simulator of the inertial navigation unit (INU), i.e. the simulator that provides assumed data of the INU depending on the chosen true forces affecting the INU, had to be built in order to speed up whole following design. This chapter provides information about INU simulator design as well as settings of its parameters and choice of assumed true forces for further use in extended Kalman filter design (EKF).

4.1 INU simulator

INU simulator is made using the MATLAB/Simulink interface. The system is made of the basic Simulink blocks. The mathematical model comes from sec. 3.2.

The Simulink scheme of the simulator can be found in fig. 4.1. The green blocks present the system inputs. They are true accelerations and angular rates in plane coordinate frame as well as the geomagnetic intensity[1] in ground coordinate frame. It is possible to shape signal true acceleration and angular rate data in the appropriate input blocks.

The light blue blocks represents subsystems that are to be described later in this chapter. The orange, noise, and red, offset, blocks represent the additive error of the sensor outputs. Their sum with the true simulated sensor data produces the simulated sensor output signal.

[1]Note that there is M_x, M_y, M_z. notation in the scheme being used to represent the magnetic intensity. This notation survived from the earlier versions of the simulator. Let's assume $M_x = H_x, M_y = H_y, M_z = H_z$

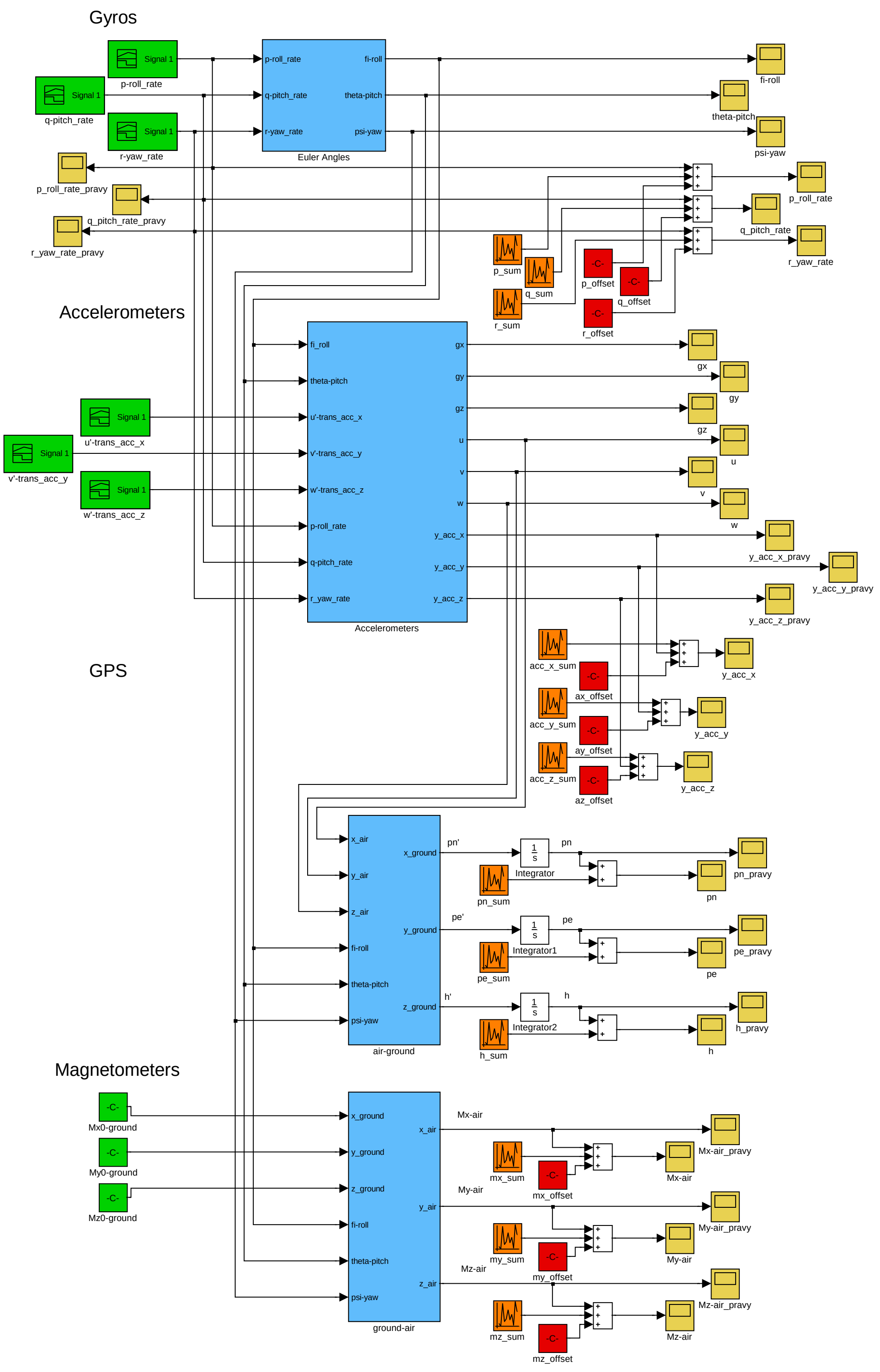


Figure 4.1: INU Simulator

Signal type	**True**	**Output**	**Workspace**	**Description**
Gyroscopes	p_roll_rate_pravy		p_roll_rateData_pravy	true roll angular rate
	q_pitch_rate_pravy		q_pitch_rateData_pravy	true pitch angular rate
	r_yaw_rate_pravy		r_yaw_rateData_pravy	true yaw angular rate
		p_roll_rate	p_roll_rateData	simulated gyro roll output
		q_pitch_rate	q_pitch_rateData	simulated gyro pitch output
		r_yaw_rate	r_yaw_rateData	simulated gyro yaw output
	fi-roll		fi_rollData	roll angle ϕ
	theta-pitch		theta_pitchData	pitch angle θ
	psi-yaw		psi_yawData	yaw angle ψ
Accelerometers	y_acc_x_pravy		y_acc_xData_pravy	true x-axis trans. acceleration
	y_acc_y_pravy		y_acc_yData_pravy	true y-axis trans. acceleration
	y_acc_z_pravy		y_acc_zData_pravy	true z-axis trans. acceleration
		y_acc_x	y_acc_xData	simul. x-axis output acc.
		y_acc_y	y_acc_yData	simul. y-axis output acc.
		y_acc_z	y_acc_zData	simul. z-axis output acc.
	u		uData	trans. x-axis velocity
	v		vData	trans. y-axis velocity
	w		wData	trans. z-axis velocity
	gx		gxData	x-axis gravity
	gy		gyData	y-axis gravity
	gz		gzData	z-axis gravity
GPS	pn_pravy		pnData_pravy	true north position, ground frame
	pe_pravy		peData_pravy	true east position, ground frame
	h_pravy		hData_pravy	true altitude, ground frame
		pn	pnData	simul. north position, ground frame
		pe	peData	simul. east position, ground frame
		h	hData	simul. altitude, ground frame
Magnetometers	Mx-air_pravy		Mx_airData_pravy	true magneto. x
	My-air_pravy		My_airData_pravy	true magneto. y
	Mz-air_pravy		Mz_airData_pravy	true magneto. z
		Mx-air	Mx_airData	simul. magneto. x
		My-air	My_airData	simul. magneto. y
		Mz-air	Mz_airData	simul. magneto. z

Table 4.1: Acquisition blocks summary

The yellow blocks are the data acquisition ones. I chose standard scope blocks to be able to observe the simulation easily when debugging the simulator or generating new data. Scope includes a function of transfering acquired data to workspace which is the most welcome feature. The table 4.1 summarizes all output blocks. In the "True" column, there are names of blocks that collect data in their true shape. That is, the collected data depend only on the true input signal of the green blocks. The "Output" column includes names of blocks colleting the data that are expected at the output of the real system. These data include the true signal as well as a noise and/or an offset. The "workspace" column shows names of the data structures that can be found in MATLAB workspace after the simulation has been performed.

There are several parameters, such as sample period or offset values, that have to be set before the simulation is started. These parameters can easily be set using *init.m* file situated in the *Simulation* directory as well as the simulator Simulink model file *simulace_8_sum_offsetAll_1Hz_GPS_GPSinit.mdl.* Table 4.2 lists all parameters of INU simulator.

T_vystupu	=	sample period of sensor data acquisition
T_gps	=	sample period of sensor data acquisition
g	=	gravity constant
Mx0_ground	=	x-axis component of the geomagnetic intensity in ground coordinate frame, [gauss]
My0_ground	=	y-axis component of the geomagnetic intensity in ground coordinate frame, [gauss]
Mz0_ground	=	z-axis component of the geomagnetic intensity in ground coordinate frame, [gauss]
acc_sum	=	noise variance of accelerometers $[m/s^2]$
gyro_sum	=	noise variance of gyroscopes $[rad/s]$
gps_sum	=	noise variance of GPS $[m]$
magneto_sum	=	noise variance of magnetometers $[gauss]$
p_offset	=	roll gyro offset
q_offset	=	pitch gyro offset
r_offset	=	yaw gyro offset
ax_offset	=	x accelerometer offset
ay_offset	=	y accelerometer offset
az_offset	=	z accelerometer offset

mx_offset	=	x magnetometer offset
my_offset	=	y magnetometer offset
mz_offset	=	z magnetometer offset
pn_init	=	initial position in +north-south axis direction [m]
pe_init	=	initial position in +east-west axis direction [m]
h_init	=	initial position in +down-up axis direction [m]

Table 4.2: Simulator parameters

4.2 Subsystems

As has been mentioned in sec. 4.1, there are some subsystems in the Simulink scheme of the simulator in fig. 4.1. This section describes their function.

The block in fig. 4.2 represents the Euler angles dynamics. That is, it realizes eq. (3.8). The inputs of the system are actual angular rates; the outputs are Euler angles. Whole Simulink scheme can be found in D.

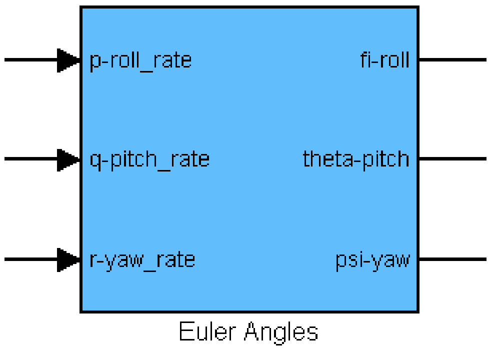


Figure 4.2: Block of Euler angles dynamics

In figure 4.3, there is the accelerometer block . Its task is to count the true translational velocities and true accelerometer data depending on the Euler angles and true translational accelerations that are brought to the inputs. The block also provides gravitation vector in the plane coordinate frame. Actually, the Accelerometers block realizes eq. (3.12). The inside of the block is shown in app. D.

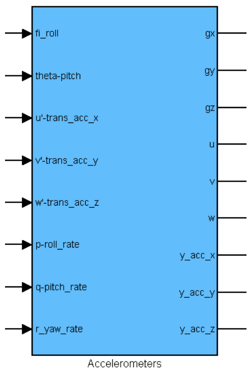


Figure 4.3: Accelerometers subsystem block

Figure 4.4 contains both ground-plain (4.4a) and plain-ground (4.4b) coordinate transformation blocks. The inputs are actual Euler angles and the vector we need to transform. On the output is then the transformed vector in new coordinate frame. In fact, these blocks realize transformation matrix (3.6) and its inverse respectively. Simulink schemes of both blocks can be seen in app. D.

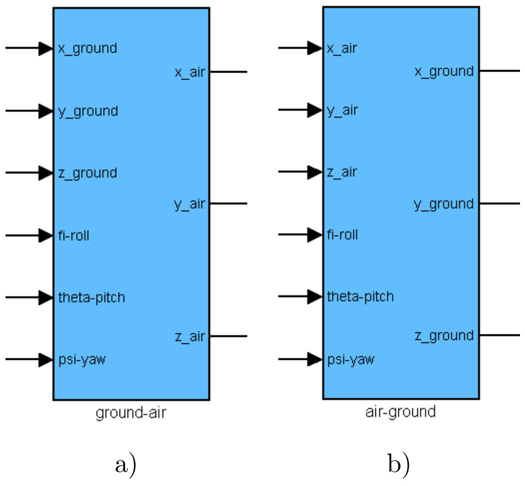


Figure 4.4: Transformation blocks. a) Transformation from the ground coordinate frame to plane one. b) Inverse transformation

4.3 Simulation data

Several different simulations were done through the project. I have chosen one that is in my opinion sufficiently illustrative for the purposes of this thesis. This section describes the details of the simulation.

First of all we set the parameters of INU simulator up. The table 4.3 shows the setting.

parameter	**value**	**unit**	**parameter**	**value**	**unit**
g	9.81	$[m/s^2]$	T_vystupu	0.1	[s]
Mx0_ground	0.198821	[gauss]	T_gps	1	[s]
My0_ground	0.009764	[gauss]	ax_offset	0.2	$[m/s^2]$
Mz0_ground	0.446022	[gauss]	ay_offset	-0.3	$[m/s^2]$
acc_sum	0.15	$[m/s^2]$	az_offset	0.1	$[m/s^2]$
gyro_sum	0.033	$[rad/s]$	mx_offset	0.01	[gauss]
gps_sum	2.5	$[m]$	my_offset	-0.025	[gauss]
magneto_sum	0.002	$[gauss]$	mz_offset	-0.01	[gauss]
p_offset	0.01	$[rad/s]$	pn_init	0	[m]
q_offset	-0.02	$[rad/s]$	pe_init	0	[m]
r_offset	0.015	$[rad/s]$	h_init	0	[m]

Table 4.3: Settings of simulator parameters.

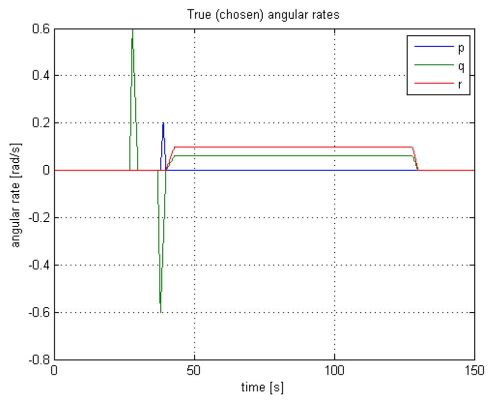
To shape the simulation data according to our needs, we need to set appropriate input data. That is, we shape the true acceleration $\dot{u}, \dot{v}$ and $\dot{w}$ and angular rates p, q and r data by using six "*signal builder*" blocks at the input of the simulator. The detailed description of the chosen signals is following:

0s	- start of the simulation
0-21s	- hold all input signals at zero
21-22s	- linear growth of the true x-axis translational acceleration towards $15[m/s^2]$
22-27s	- hold all signals at their levels
27-28s	- linear growth of the pitch angular rate towards $0.6[rad/s]$
28-30s	- linear fall of the pitch angular rate towards $0[rad/s]$
	- linear fall of the true x-axis translational acceleration towards $0[m/s^2]$
30s	- from now on the signals of the true y-axis and z-axis trans. accelerations are completely random simulating the effects of the surrounding environment such as the wind and so on

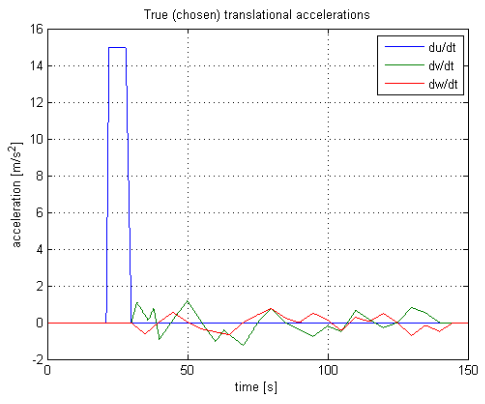
37-38s	-	linear fall of the pitch angular rate towards -0.6$[rad/s]$
38-39s	-	linear growth of the roll angular rate towards 0.2$[rad/s]$
39-40s	-	linear fall of the roll angular rate towards 0$[rad/s]$
38-40s	-	linear growth of the pitch angular rate towards 0$[rad/s]$
40-43s	-	linear growth of the pitch angular rate towards 0.06$[rad/s]$
	-	linear growth of the yaw angular rate towards 0.1$[rad/s]$
128-130s	-	linear fall of the pitch angular rate towards 0$[rad/s]$
	-	linear fall of the yaw angular rate towards 0$[rad/s]$
150s	-	end of the simulation

That is, the aircraft first accelerates in the straightforward direction. Then it raises its nose and flies up under the constant angle. Then it stops accelerating and levels the flight level. During this manoeuvre the environment starts to affect the aircraft that starts to move randomly in the left, right, up and down directions. After the flight level has been leveled, the aircraft rotates a little positively around the x axis and then it remains in the circles, thanks to a rotating around both the z and y axes, until the rotations are stopped a while before the simulation ends.

The described set of moves can be found in fig. 4.5. The appropriate simulated sensor outputs are shown in fig. 4.6. These outputs are used in the next chapter as the data that are to be compared with EKF output.

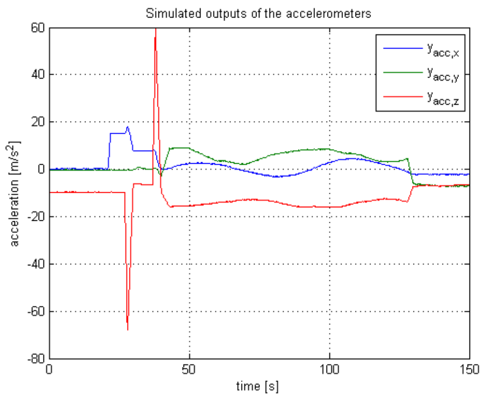


a)

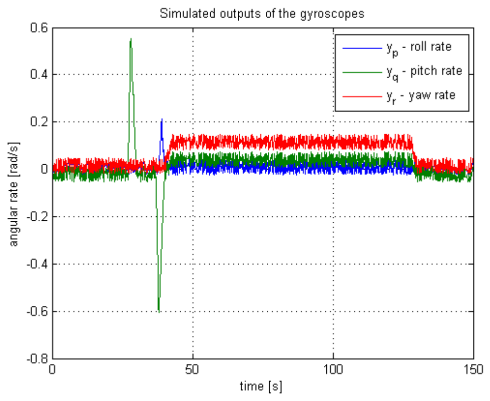


b)

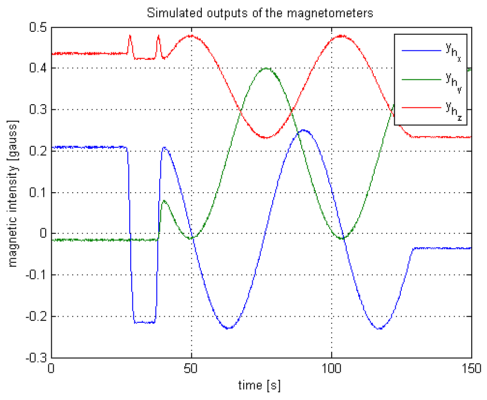
Figure 4.5: Input data of the simulation: a) true translational accelerations, b) true angular rates



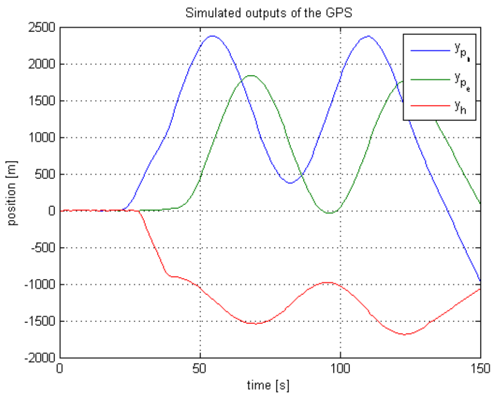
a)



b)



c)



d)

Figure 4.6: Output data of the simulation: a) accelerometers, b) gyroscopes, c) magnetometers, d) GPS

Chapter 5

EKF design

Till now, this thesis has introduced all theory and presumptions we need to design an extended Kalman filter (EKF) for this application. This chapter describes the EKF design while using both the methods and state space model of chapter 3 and the simulated data of chapter 4.

5.1 EKF M-file

The design is being made for the real hardware with the real microprocessor unit LPC2119 (chapter 2). It has been found that we can not use the double precision floating-point variables type for this algorithm simply because the memory capacity of MCU is unsufficient. That is, we must use the single precision $float$ type. In order to simulate this condition, I have decided to use single precision variables in MATLAB too. This can be done by simply retyping all variables in the design M-file by[1]

$$x = single(x) \tag{5.1}$$

Before the estimation starts, we need to set constants up. They are the data acquisition period of GPS $T_{GPS} = 1$, the data acquisition period of the other sensors $T_{sens} = 0.1$, the gravitation constant $g = 9.81$ and the geomagnetic intensity vector in the ground coordinate frame H_{g0} that depends on the actual position in the Earth. As the data simulation has been done with Prague magnetic intensity values, we set intensity

[1]The standard type in MATLAB is *double* and all variables that are not retyped are automaticly *double*.

components according to (3.18). Also the covariance matrices P, Q and R have to be initialized at this point. The P matrix initiates as an identity matrix, the other two matrices are diagonal and their settings will be discussed later.

The EKF algorithm itself is well described in sec. 3.3. We could end here with the desription of EKF structure, however, the mentioned algorithm is "only" a backbone of the used procedure. There are some aspects that have to be yet introduced.

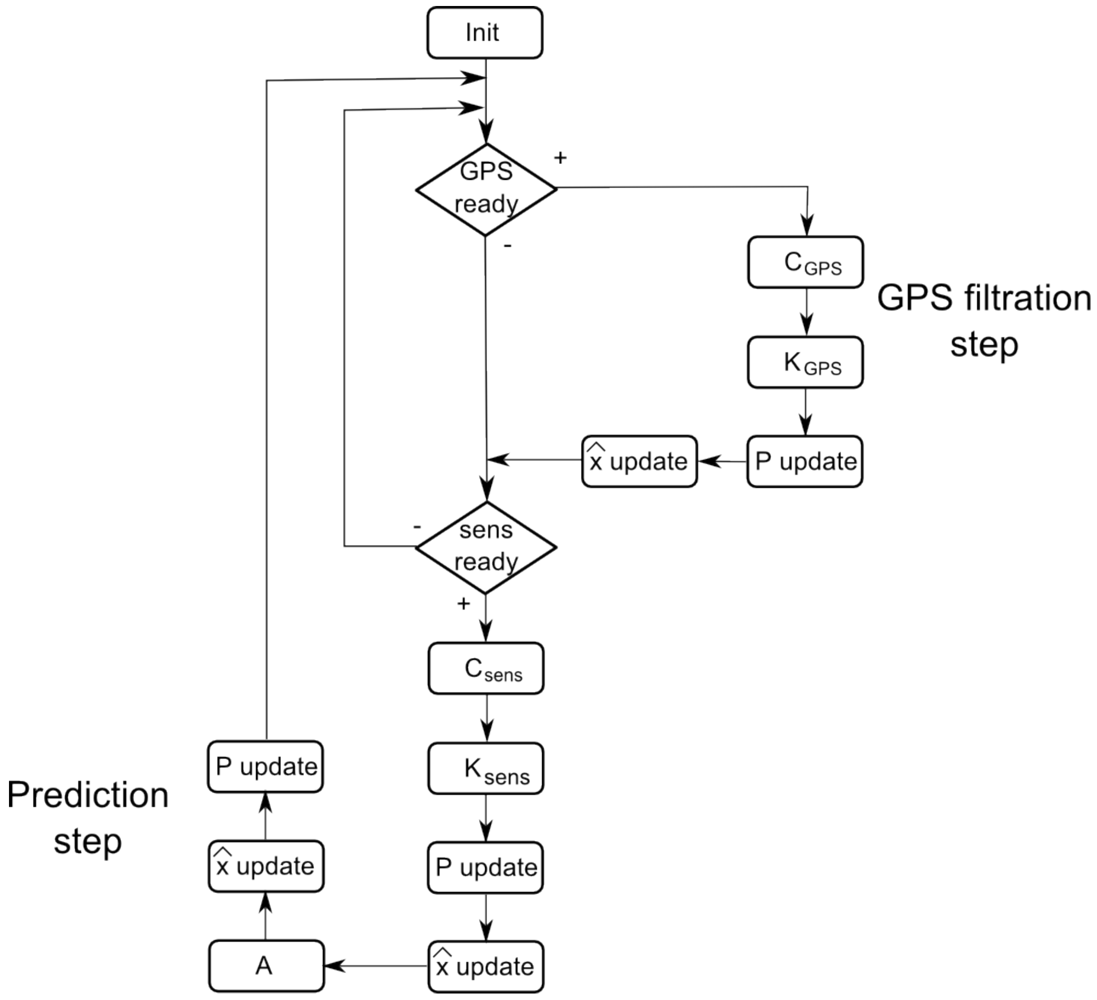


Figure 5.1: Progress chart of EKF algorithm

First of all, we have to deal with the difference between the GPS and the other sensors data acquisition rates. Figure 5.1 shows the progress chart of the used algorithm that solves this problem. At the beginning the initialization is performed. Then the algorithm loop begins. Each time the data from the sensors (accelerometers, magnetometers and gyroscopes) are acquired, the sensor filtration step is made except the situation when the data from GPS are acquired. In that case the GPS filtration step is being performed and then the algorithm returns to the sensor data waiting loop. The prediction step is made always after the sensor filtration step computation.

To obtain Kalman's gain in (3.22), i.e. to solve

$$L_i = PC_i^T (C_i P C_i^T + R)^{-1}, \tag{5.2}$$

we need to find an inverse of $(C_i P C_i^T + R)$ which is not easily done when we want to use the MCU. Fortunately, we can take into an account that this matrix is symmetric and so we can solve

$$L_i (C_i P C_i^T + R) = PC_i^T, \tag{5.3}$$

for L_i using Cholesky decomposition explained in sec. 3.4. This fact is also remembered in the EKF design M-file by using functions

$$\begin{aligned} [L\_chol, diag\_chol] &= Cholesky\_decomp\_single(C * p * C' + R); \\ L &= Cholesky\_solut\_single(L\_chol, diag\_chol, p * C'); \end{aligned} \tag{5.4}$$

instead of standard

$$L = p * C' * inv(C * p * C' + R); \tag{5.5}$$

The functions (5.4) performe the Cholesky method exactly by using single precision arithmetics. While this method has been already detailed in sec. 3.4, the further discussing of this problem is not necessary. The appropriate M-files can be found on the enclosed CD.

It has also been found that, due to the "unprecise" CPU/MCU arithmetics, there may rise an error of the P matrix symmetry. This error grows with the number of performed iterations and the estimation becomes very unpredictable. Some of the elements of P matrix may rise towards infinity. Hence the normalization procedure is being performed each time the P matrix is modified. This procedure symmetrizes P matrix simply by

$$\begin{aligned} P_{i,j} &= \tfrac{1}{2} P_(i,j) P_(j,i) \\ P_{j,i} &= P_{i,j} \end{aligned} \tag{5.6}$$

$\forall i < j$. This simple algorithm is written in the "*normalize_p.m*" M-file and is located on the enclosed CD.

Whole algorithm is transformed into the MATLAB code in *thesis_EKF_simulation.m* that can be found on the enclosed CD.

5.2 EKF tuning and results

The EKF tuning consists in the Q and R covariance matrix choice. The initiate settings came from the sensors and GPS datasheets (Ana, 2007-2008), (Hon, 2007) and (Nav, 2007). The further design was made ad hoc thanks to the simulations in MATLAB that are much more effective than using INU hardware directly. The resulting setting for Q, R_{sens} and R_{GPS} covariance matrices are in a diagonal form. The appropriate diagonals of the covariance matrices are

$$\begin{aligned} Q_{diag} &= \xi\xi^T &&= [.05^2, .05^2, .05^2, .05^2, .05^2, .05^2, 2^2, 2^2, 2^2, 1^2, 1^2, 1^2, \\ & && \quad 0^2, 0^2, 0^2, 0^2, 0^2, 0^2, 0^2, 0^2, 0^2, 1^2, 1^2, 1^2]T_{sens} \\ R_{GPS,diag} &= (\eta_{GPS})(\eta_{GPS})^T &&= [30^2, 30^2, 30^2]T_{GPS} \\ R_{sens,diag} &= (\eta_{sens})(\eta_{sens})^T &&= [.05^2, .05^2, .05^2, 1^2, 1^2, 1^2, 1^2, 1^2, 1^2]T_{sens} \end{aligned} \tag{5.7}$$

The Q matrix represents the state signal noises ξ while R matrices represent the output signal noises η. Actually, we can say that the model being used by EKF is complete now. Note that there is also the period of data acquisition being taken into an account in (5.7).

All results of the designed Kalman filter are in app. E. The main result that we had to obtain is the estimate of the translational velocity. Figure 5.2 depicts the velocity estimation of the extended INU simulation data[2]. In that figure we can see that the estimation goes well for the signal of x-axis translational velocity $\dot{u}$ while the results of the other two velocity estimations are not so good. In addition no estimate, including that of $\dot{u}$, is good until the estimated system makes some movement.

The explanation lies in the GPS properties. According to the simulation settings we made in sec. 4.3, the GPS signal varies $\pm 3m$ in each axis and is acquired only once per second. This choice was made with respect to (Nav, 2007). The velocity signal is affected by both the GPS filtration step and the sensor filtration step using the data of accelerometers. However, there is an integral relation between acceleration and velocity. From the improper integral definition the solution is not unique and depends on the initial conditions of the system. That is why we can filter the velocity signal relevantly only by using the position data of GPS where the derivative unique relation is. If the system makes some movement in any direction, it is obvious that, assuming the variance

[2]The extended in this case means that first 150 seconds of the simulation are the same as those described in sec. 4.3. Then the simulation remains in the state when both translational accelerations and angular rates are zero, until the simulation stops at 500 seconds.

properties of GPS, the relative error will be much lower than in the case the system does not move at all. For an illustartion, the difference between moving the speed[3] of $10m/s$ and $1m/s$ is that in former case the relative position after 1 second returned by GPS is $10 \pm 3m$ and in the second case the position is $1 \pm 3m$. That is, the maximum relative error of the position is in the former case $\pm 30\%$ while in the second one it is $\pm 300\%$. The quality of the estimation depends on the right choice of R_{GPS} but the properties of GPS signal limit it if the system moves either too slowly or not at all.

The conclusion is that this velocity estimation system is recommended to be used for systems which translational velocities are not presumed to be near zero. The estimation of the velocities near zeros using this system is very untrustful.

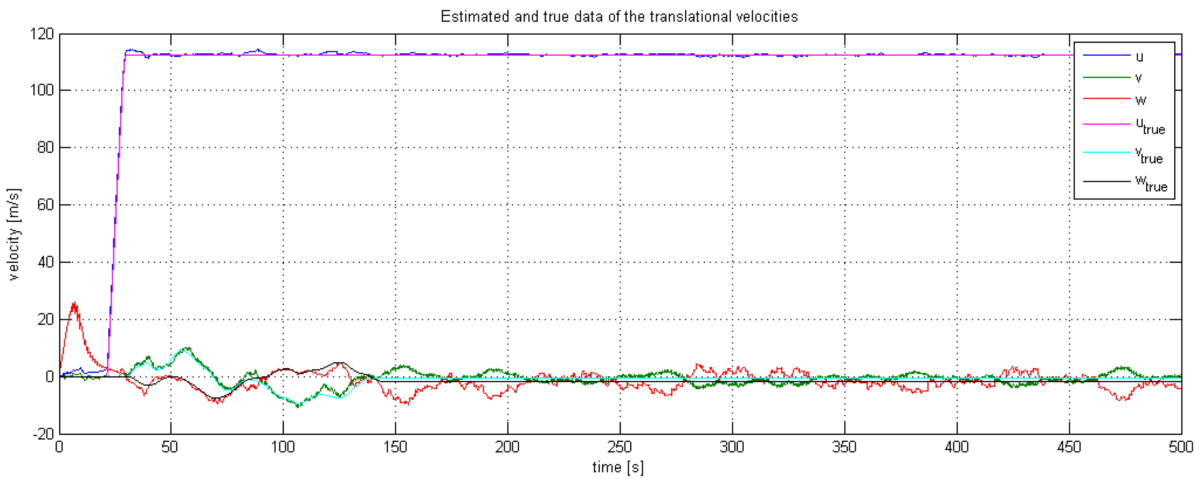


Figure 5.2: EKF velocity estimation (extended INU data)

[3]We assume the one dimensional movement

Chapter 6

Hardware

This chapter discusses hardware programming and some hardware bugs, that have been found during the project, as well as some suggestions of their solution.

6.1 Hardware programming

The backbone of the code comes from Martin Řezáč and implements the accelerometer/gyroscope data collecting using SPI interface, the functions for matrix operations such as matrix multiplication, Cholesky decomposition etc. and the basic template for the extended Kalman filter algorithm. However, this code has to be added and modified a little to suit the current problem. Note that whole code is located in *inerce.c* file on the enclosed CD as well as its manual. It includes all necessary comments and hence this section only introduces either the aspects that are needed to give them a closer look or the interesting ones.

6.1.1 GPS data acquisition

As is written in sec. 2.4, the GPS receiver proceeds its data to the serial port UART1 of LPC2119. The serial link parameters are 9600 bps, 8 data-bits, no parity, 1 stop-bit. To set these parameters for LPC2119 we can include a prepared header file $uart_zen.h$ and then use function

$$UART\_init((uint8\_t)1,(uint32\_t)1600,11); \tag{6.1}$$

where the first argument specifies the UART channel, the second argument sets the baud rate and the third argument is the interrupt priority. Note that the basic LPC2119 clock frequency of $10MHz$ has been multiplied by six using an in-built Current Controlled Oscillator (see (Phi, 2004)). The real baud rate is then actually six times greater than the one that has been specified as the $UART_init$ argument. That is, the function 6.1 specifies the baud rate of $9600bps$.

The GPS data are transferred in NMEA protocol form (see app. B). However, we need to obtain the GPS position and altitude only; may be yet the actual time. This information is located in GGA (Global positioning system fixed data) subprotocol. The interesting part for us is

$$\$GPGGA, time, latitude, N/S\ indicator, longitude, E/W\ indic., \ldots, altitude, \ldots * \tag{6.2}$$

The algorithm for GGA recognition runs in the UART1 interrupt routine. This algorithm only runs some counters that are very quick and hence the usage of this algorithm in interrupt routine is relevant. The algorithm is:

1. initialize a global variable $char\ RXGPGGA[100] = "\$GPGGA * \backslash 0;$
2. wait for "\$" symbol
3. next five characters count the number of received "G", "P" and "A" symbols
4. if $\sum G = 3$, $\sum P = 1$ and $\sum A = 1$, go to 5.; else return to 2.
5. fill any received character in the $RXGPGGA$ until the $*$ symbol is received;
6. put $0x0D$ and $0x0A$ at the end of $RXGPGGA$. GPS data are ready now.
7. return to 2.

Note that the algorithm is prepared in the manner to be able to obtain the other subprotocols of NMEA too. The appropriate parts of the code are currently commented but if the need rises, a simple modification will allow us to gain those subprotocols.

We have just obtained a string $RXGPGGA$ that contains the GGA subprotocol. We need now to obtain the data we need from it. This problem is solved by the function $voidparse\_GPGGA(void)$. This function parses the GGA protocol while knowing its structure. That is, it parses the GGA protocol according to the commas it includes and saves the appropriate data into separate *string* variables. After this procedure, the obtained strings are converted to $float$ type while the "N/S" and E/W indicators affect the positivity/negativity of the latitude and longitude.

6.1.2 Code optimization

The hardware version of the algorithm is actually similar to that being explain in sec. 5.1. Hence I decided not to describe it in this chapter again. However, there is a big difference in the calculation itself. Here comes its description.

The EKF algorithm 3.22 and 3.24 forces us to performe matrix multiplications. As we have the A and P matrices of the order 24 (see C.2), only the $A \cdot P$ multiplication means, using a simple mathematics, $13,824$ float type multiplications and the whole equation $P = APA^T + Q$ means $27,648$ float type multiplications. That is, the matrix multiplications weight on MCU a lot and really slow down the whole calculation process.

If we take a closer look at the A matrix, however, we realize that so many multiplications are not necessary because the A matrix contains many spaces that are zero all the time and hence the multiplications using those elements are always zero. Moreover from the 10th to 24th row there are elements of ones only on the diagonal of A and hence no multiplication is needed too. That is why the new multiplication algorithms have been developed. They are

function	**multiplication count**	**operation**
ApAtQ	3,312	$APA^T + Q$
pCt_gps	0	PC^T_{GPS}
pCt_sens	216	PC^T_{sens}
CpCtR_gps	0	$C_{GPS}PC^T_{GPS} + R$
CpCtR_sens	216	$C_{sens}PC^T_{sens} + R$
$\sum$	3,744	

All functions are to be found both in C-code version in *inerce.c* and MATLAB-code version in an appropriate m-file on the enclosed CD. Note that these code modifications are very specific for the current state space model. If the model changes, these functions will be useless.

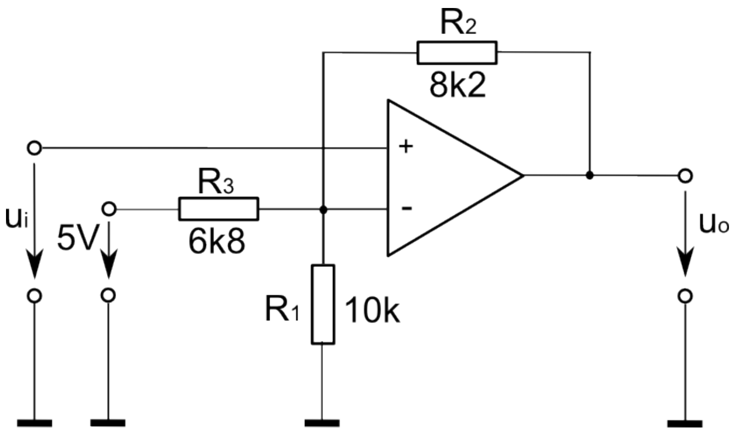


Figure 6.1: Magnetometer signal processing

6.2 Magnetometer data acquisition

According to sec. 2.3 the used magnetometer HMC2003 generates analog signal 0.5-4.5V. The signal processing circuits changes the signal to fit the 3.3V A/D input of LPC2119. To do this, the operational amplifiers are used (see fig. 6.1). I counted the input-output voltage relation as

$$u_o = 3.026u_i - 6.029 \qquad (6.3)$$

where u_i is the magnetometer signal and u_o is the processed signal that is brought to the LPC2119 A/D input. The experiment, when the known voltage was brought to the input of amplifier and the output voltage was measured, confirmed the result (6.3). However, applying this relation to the signals that were transformed by A/D converters brought different results then I expected. That is, the size of measured magnetic intensity vector was too different from the size of vector (3.18). Unfortunately, I was not able to solve this problem within the thesis project due to the shortage of time.

6.3 Known hardware problems

Throughout whole project, it is quietly assumed that altitude data are obtained using GPS measurements. It is true that the used GPS receiver provides this information but, generally, it is very untrustworthy. The design counts with GPS altitude measures simply because there is no other altitude source for the algorithm within the inertial navigation unit (INU).

The solution for this problem is already in progress now but, most likely, will not be completed before the end of this thesis. The solution consists in use of the barometric altitude sensor. The concrete model of the barometric module that has been chosen is MS5534B of the Intersema company which there is some experience with. The main features are

- 10-1100$mbar$ absolute pressure range
- Piezosensitive silicon micromachined sensor
- 15Bit ADC
- 3-wire serial interface
- low power and low voltage consumption

Complete information may be seen in (Int, 2005).

Chapter 7

Conclusion

Within the project of the Stabilized camera platform for UAV Manta, the inertial navigation unit (INU) is being developed. Although the camera platform is recently able to compensate rotation impacts and keep the line of sight of the camera in constant direction, the observation of some concrete stationary or moving object is rather different and more difficult task. Generally, if the picture of an object that we observe in the headquarters moves out of the screen, we do not know whether it is caused due to the rotations of the camera carrier or due to the tranlational movement of the carrier alongside the observed object. This fact forces us to develop navigation system that provides translational velocity of the carrier.

The hardware for INU was developed by Jaroslav Žoha and Martin Řezáč. It contains three-axis gyroscope, three-axis accelerometer, three-axis magnetometer as well as a GPS module. None of these sensors provides the velocity data either directly or in a sufficient period. That is why the three-axis velocity has to be estimated. This thesis solves the problem of the velocity estimation using a given inertial navigation unit.

As the system is nonlinear, the extended Kalman filter method (EKF) has been chosen. The thesis derives the solution, using this method, step by step. First, the mathematical background is provided. It contains basic information on the used coordinate frames, EKF method, Cholesky decomposition and the mathematical models of the used sensors. To use the EKF properly, we need to set a quality state space model for INU. The INU model is derived from the sensor models. To allow the EKF to filter the offsets of the accelerometers, gyroscopes and magnetometers, the model contains separate state variables for them. The INU model order is 24.

The INU simulator has been developed in order to ensure the most effective EKF design. The simulator uses MATLAB/Simulink interface and provides expected sensor output data as well as the true state data to observe the designed EKF properly. Simulated data are saved into the MATLAB workspace where they are prepared for further EKF design.

The EKF design is made with respect to MCU abilities. That is, the Cholesky decomposition is used instead of matrix inverse, the expected sample periods are applied, the single precision floating point arithmetics are used etc. The designed EKF algorithm also solves the problem of the different GPS sample time. The resulting estimation of the velocity is compared to the true one produced by the simulator and discussed. The estimates of the other state variables are also included in an appropriate appendix.

The last chapter discusses the hardware programming and some problems that appeared during it. The hardware algorithm is actually the C-code version of the MATLAB EKF design code. The C-code has been optimized using the exact knowledge of the state matrices A and C. These matrices contain many zero elements all the time and hence it is not necessary to compute whole matrix multiplication according to the definition. For example, the optimization reduces the number of floating point multiplications of the term APA^T, where A and P are square matrices of order 24, from previous 27,648 to optimized 3,312. However, this optimization is very specific for the used state space model. If another model is used, the optimization must be rebuilt.

Unfortunately, I was not able to obtain the proper data of the magnetometers within the thesis project due to the time press. Also, it has been realized that the altitude data provided by the GPS module are too untrustworthy. The solution has been suggested in the last chapter. It consists in the use of a barometric module. The solution is in progress and will not be completed within the thesis project.

Considering the mentioned sensor problems, it is useless to perform any experiments using the programmed hardware. Before the sensors work properly, the computed data are not relevant to make any conclusions.

I presume to go on with the project after the thesis has been finished.

Bibliography

Ana (2007-2008), *High Precision Tri-Axis Inertial Sensor ADIS16350/ADIS16355.* Analog Devices, `http://www.analog.com/static/imported-files/data_sheets/ADIS16350.pdf`.

Hon (2007), *HMC2003, Three-axis Magnetic Sensor Hybrid.* Honeywell, `http://www.ssec.honeywell.com/magnetic/datasheets/hmc2003.pdf`.

Int (2005), *MS5534B, Barometer module.* Intersema, `www.intersema.ch/products/documentation/doc/34/raw/`.

Nav (2007), *Datasheet of GPS smart antenna module, NL-50x MTK series.* Navilock, `http://www.omtec.de/bilder/datenblaetter/navilock/60412_60413_60414_MTK_Manual.pdf`.

Phi (2004), *LPC2119/2129/2194/2292/2294 user manual.* Philips Semiconductors, `http://www.nxp.com/acrobat_download/usermanuals/UM_LPC21XX_LPC22XX_2.pdf`.

Beard, R. W. (2007), *Studies in Computational Intelligence*, `www.springerlink.com`, chapter State Estimation for Micro Air Vehicles, pp. **173–199**.

Řezáč, M. (2008), *Controller design of the stabilization system of the optical axis of the camera system of an unmanned aircraft*, Master's thesis, Faculty of Electrical Engineering, Czech Technical University in Prague.

Haniš, T. (2008), *Stabilized platform for UAV vehicles: signal processing of inertial sensors*, Master's thesis, Faculty of Electrical Engineering, Czech Technical University in Prague.

Havlena, V. and Štecha, J. (1999), '*Modern control theory*', Lecture notes, Faculty of Electrical Engineering, Czech Technical University in Prague.

Hurák, Z. (2008), *Inertial line-of-sight stabilization, pointing and tracking for an airborne double gimbal Az-El system: control design aspects.* Faculty of Electrical Engineering, Czech Technical University in Prague.

Žoha, J. (2008*a*), *Electronic subsystems*, Technical report, Centre for Applied Cybernetics, Czech Technical University in Prague.

Žoha, J. (2008*b*), *Electronical stabilization of the optical axis of the camera system of an unmanned aircraft*, Master's thesis, Faculty of Electrical Engineering, Czech Technical University in Prague.

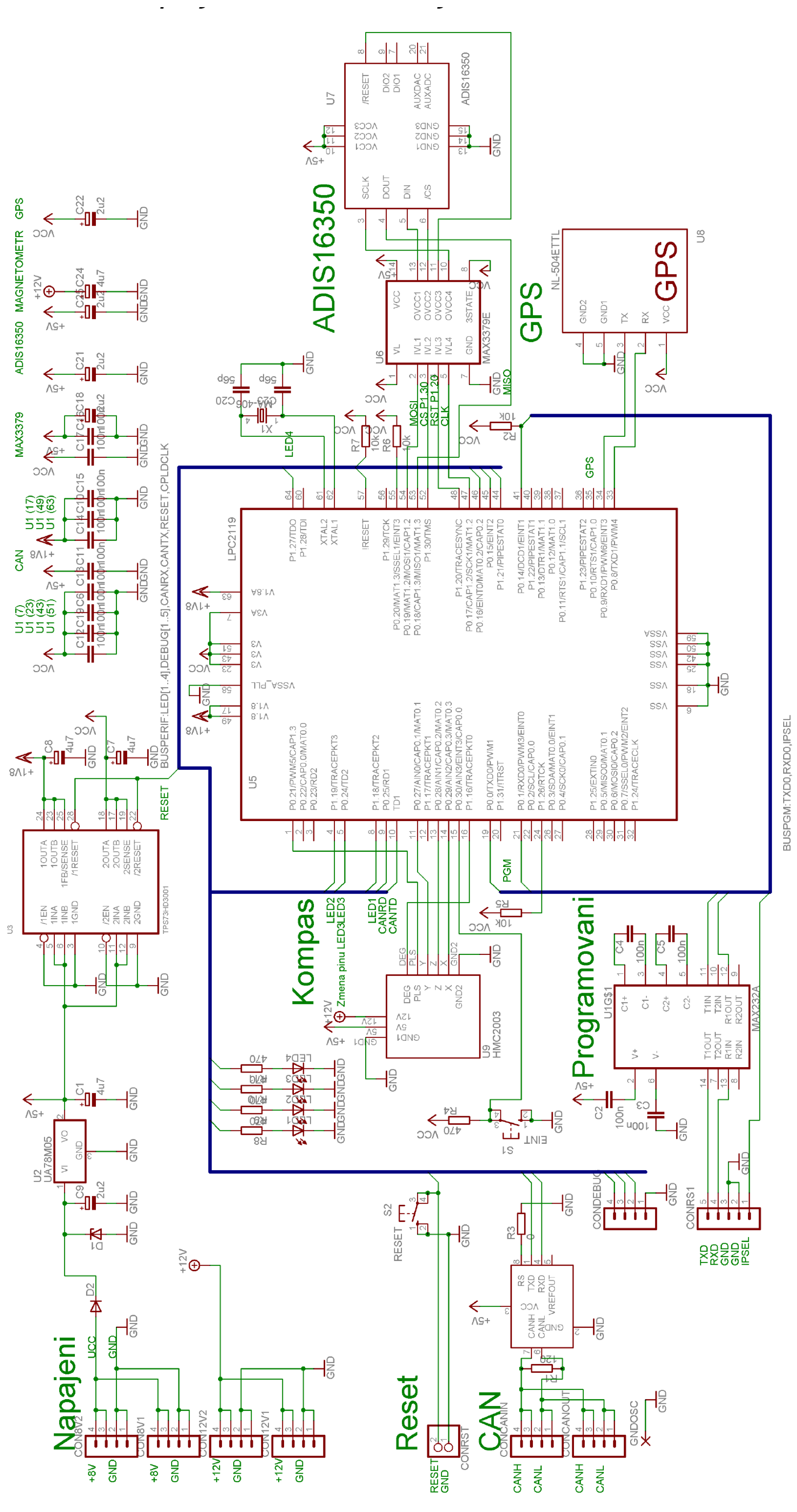
Pech, Z. and Věk, V. (2003), '*Flight control systems*', Lecture notes, Faculty of Electrical Engineering, Czech Technical University in Prague.

Štecha, J. (1999), '*Optimal decision and control*', Lecture notes, Faculty of Electrical Engineering, Czech Technical University in Prague.

www.mathpages.com (2008), '*Rotation Matrices*'. http://www.mathpages.com/home/kmath593/kmath593.htm.

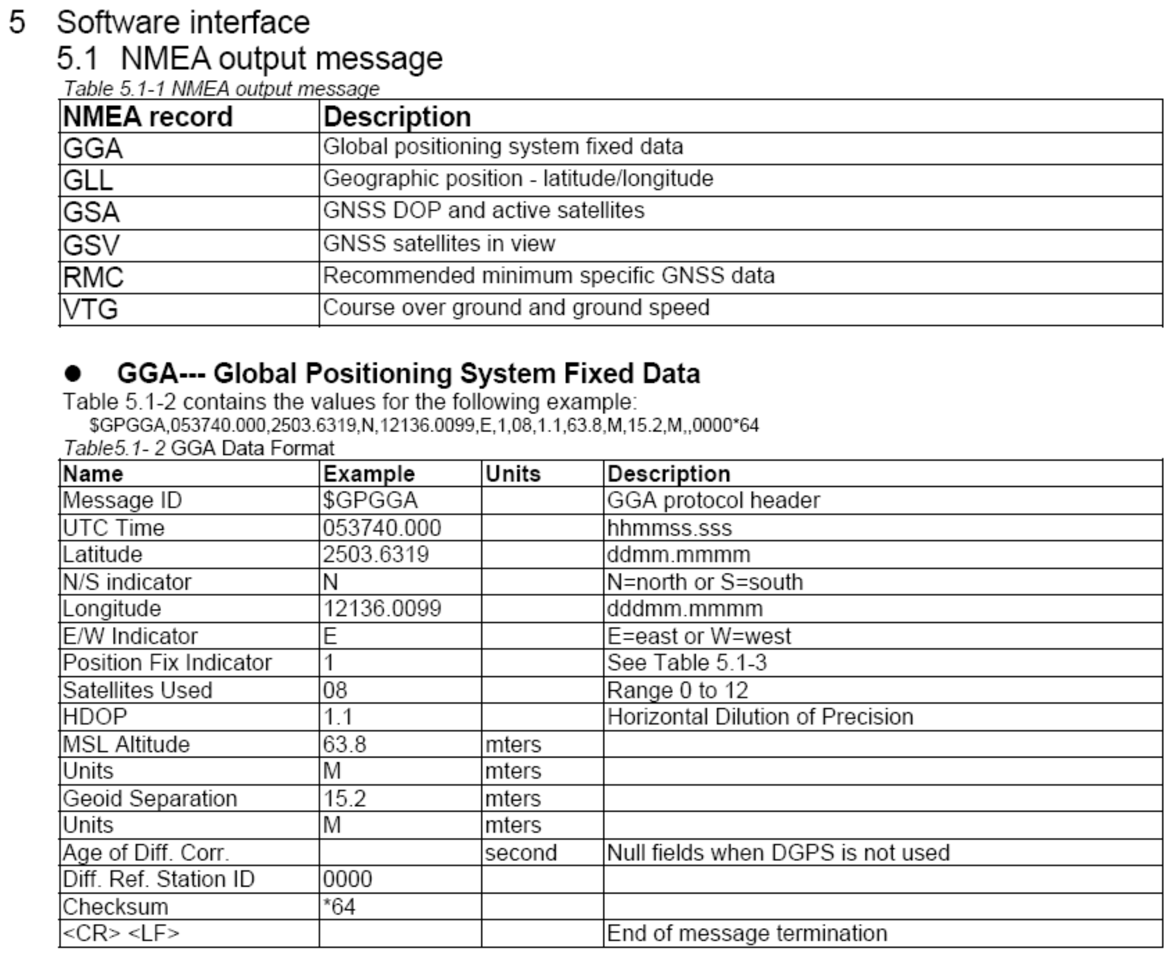
Appendix A

Wiring diagram


Napajeni
Reset
CAN
Kompas
Programovani
ADIS16350
GPS
LPC2119
MAX3379E
HMC2003
MAX232A
NL-504ETTL
TPS73HD301
UA78M05
Zmena pinu LED3

Appendix B

NMEA Protocol



5 Software interface

5.1 NMEA output message

Table 5.1-1 NMEA output message

NMEA record	Description
GGA	Global positioning system fixed data
GLL	Geographic position - latitude/longitude
GSA	GNSS DOP and active satellites
GSV	GNSS satellites in view
RMC	Recommended minimum specific GNSS data
VTG	Course over ground and ground speed

- **GGA--- Global Positioning System Fixed Data**

Table 5.1-2 contains the values for the following example:

$GPGGA,053740.000,2503.6319,N,12136.0099,E,1,08,1.1,63.8,M,15.2,M,,0000*64

Table5.1- 2 GGA Data Format

Name	Example	Units	Description
Message ID	$GPGGA		GGA protocol header
UTC Time	053740.000		hhmmss.sss
Latitude	2503.6319		ddmm.mmmm
N/S indicator	N		N=north or S=south
Longitude	12136.0099		dddmm.mmmm
E/W Indicator	E		E=east or W=west
Position Fix Indicator	1		See Table 5.1-3
Satellites Used	08		Range 0 to 12
HDOP	1.1		Horizontal Dilution of Precision
MSL Altitude	63.8	mters	
Units	M	mters	
Geoid Separation	15.2	mters	
Units	M	mters	
Age of Diff. Corr.		second	Null fields when DGPS is not used
Diff. Ref. Station ID	0000		
Checksum	*64		
<CR> <LF>			End of message termination

Table 5.1-3 Position Fix Indicators

Value	Description
0	Fix not available or invalid
1	GPS SPS Mode, fix valid
2	Differential GPS, SPS Mode, fix valid
3-5	Not supported
6	Dead Reckoning Mode, fix valid

- **GLL--- Geographic Position – Latitude/Longitude**

Table 5.1-4 contains the values for the following example:

$GPGLL,2503.6319,N,12136.0099,E,053740.000,A,A*52

Table 5.1-4 GLL Data Format

Name	**Example**	**Units**	**Description**
Message ID	$GPGLL		GLL protocol header
Latitude	2503.6319		ddmm.mmmm
N/S indicator	N		N=north or S=south
Longitude	12136.0099		dddmm.mmmm
E/W indicator	E		E=east or W=west
UTC Time	053740.000		hhmmss.sss
Status	A		A=data valid or V=data not valid
Mode	A		A=autonomous, D=DGPS, E=DR
Checksum	*52		
<CR> <LF>			End of message termination

- **GSA---GNSS DOP and Active Satellites**

Table 5.1-5 contains the values for the following example:

$GPGSA,A,3,24,07,17,11,28,08,20,04,,,,,2.0,1.1,1.7*35

Table 5.1-5 GSA Data Format

Name	**Example**	**Units**	**Description**
Message ID	$GPGSA		GSA protocol header
Mode 1	A		See Table 5.1-6
Mode 2	3		See Table 5.1-7
ID of satellite used	24		Sv on Channel 1
ID of satellite used	07		Sv on Channel 2
….			….
ID of satellite used			Sv on Channel 12
PDOP	2.0		Position Dilution of Precision
HDOP	1.1		Horizontal Dilution of Precision
VDOP	1.7		Vertical Dilution of Precision
Checksum	*35		
<CR> <LF>			End of message termination

Table 5.1-6 Mode 1

Value	Description
M	Manual- forced to operate in 2D or 3D mode
A	Automatic-allowed to automatically switch 2D/3D

Table 5.1-7 Mode 2

Value	Description
1	Fix not available
2	2D
3	3D

- **GSV---GNSS Satellites in View**

Table 5.1-8 contains the values for the following example:

```
$GPGSV,3,1,12,28,81,285,42,24,67,302,46,31,54,354,,20,51,077,46*73
$GPGSV,3,2,12,17,41,328,45,07,32,315,45,04,31,250,40,11,25,046,41*75
$GPGSV,3,3,12,08,22,214,38,27,08,190,16,19,05,092,33,23,04,127,*7B
```

Table 5.1-8 GSV Data Format

Name	**Example**	**Units**	**Description**
Message ID	$GPGSV		GSV protocol header
Total number of messages[1]	3		Range 1 to 3
Message number[1]	1		Range 1 to 3
Satellites in view	12		
Satellite ID	28		Channel 1 (Range 01 to 32)
Elevation	81	degrees	Channel 1 (Range 00 to 90)
Azimuth	285	degrees	Channel 1 (Range 000 to 359)
SNR (C/No)	42	dB-Hz	Channel 1 (Range 00 to 99, null when not tracking)
Satellite ID	20		Channel 4 (Range 01 to 32)
Elevation	51	degrees	Channel 4 (Range 00 to 90)
Azimuth	077	degrees	Channel 4 (Range 000 to 359)
SNR (C/No)	46	dB-Hz	Channel 4 (Range 00 to 99, null when not tracking)
Checksum	*73		
<CR> <LF>			End of message termination

1. Depending on the number of satellites tracked multiple messages of GSV data may be required.

- **RMC---Recommended Minimum Specific GNSS Data**

Table 5.1-9 contains the values for the following example:

```
$GPRMC,053740.000,A,2503.6319,N,12136.0099,E,2.69,79.65,100106,,,A*53
```

Table 5.1-9 RMC Data Format

Name	**Example**	**Units**	**Description**
Message ID	$GPRMC		RMC protocol header
UTC Time	053740.000		hhmmss.sss
Status	A		A=data valid or V=data not valid
Latitude	2503.6319		ddmm.mmmm
N/S Indicator	N		N=north or S=south
Longitude	12136.0099		dddmm.mmmm
E/W Indicator	E		E=east or W=west
Speed over ground	2.69	knots	True
Course over ground	79.65	degrees	
Date	100106		ddmmyy
Magnetic variation		degrees	
Variation sense			E=east or W=west (Not shown)
Mode	A		A=autonomous, D=DGPS, E=DR
Checksum	*53		
<CR> <LF>			End of message termination

- **VTG---Course Over Ground and Ground Speed**

Table 5.1-10 contains the values for the following example:

```
$GPVTG,79.65,T,,M,2.69,N,5.0,K,A*38
```

Table 5.1-10 VTG Data Format

Name	Example	Units	Description
Message ID	$GPVTG		VTG protocol header
Course over ground	79.65	degrees	Measured heading
Reference	T		True
Course over ground		degrees	Measured heading
Reference	M		Magnetic
Speed over ground	2.69	knots	Measured speed
Units	N		Knots
Speed over ground	5.0	km/hr	Measured speed
Units	K		Kilometer per hour
Mode	A		A=autonomous, D=DGPS, E=DR
Checksum	*38		
<CR> <LF>			End of message termination

Appendix C

State space summary and Jacobians

C.1 Nonlinear state space model

State equations in $\dot{x}(t) = f(x,u,t) + \xi(t)$ form:

$$
\begin{aligned}
\dot{\phi} &= p_{est} + q_{est}\sin(\phi)\tan(\theta) + r_{est}\cos(\phi)\tan(\theta) + \xi_\phi \\
\dot{\theta} &= q_{est}\cos(\phi) - r_{est}\sin(\phi) + \xi_\theta \\
\dot{\psi} &= q_{est}\tfrac{\sin(\phi)}{\cos(\theta)} + r_{est}\tfrac{\cos(\phi)}{\cos(\theta)} + \xi_\psi \\
\dot{u} &= -q_{est}w + r_{est}v - g\sin\theta + a_{x,est} + \xi_u \\
\dot{v} &= -r_{est}u + p_{est}w + g\cos\theta\sin\phi + a_{y,est} + \xi_v \\
\dot{w} &= -p_{est}v + q_{est}u + g\cos\theta\cos\phi + a_{z,est} + \xi_w \\
\dot{p}_n &= u(c_\psi c_\theta) + c(c_\psi s_\theta s_\phi - s_\psi c_\phi) + w(c_\psi s_\theta c_\phi + s_\psi s_\phi) + \xi_{p_n} \\
\dot{p}_e &= u(s_\psi c_\theta) + v(s_\psi s_\theta s_\phi + c_\psi c_\phi) + w(s_\psi s_\theta c_\phi - c_\psi s_\phi) + \xi_{p_e} \\
\dot{h} &= -u(s_\theta) + v(c_\theta s_\phi) + w(c_\theta c_\phi) + \xi_h \\
\dot{p}_{est} &= 0 + \xi_{p_{est}} \\
\dot{q}_{est} &= 0 + \xi_{q_{est}} \\
\dot{r}_{est} &= 0 + \xi_{r_{est}} \\
\dot{p}_{off} &= 0 \\
\dot{q}_{off} &= 0 \\
\dot{r}_{off} &= 0 \\
\dot{a}_{x,off} &= 0 \\
\dot{a}_{y,off} &= 0 \\
\dot{a}_{z,off} &= 0 \\
\dot{h}_{x,off} &= 0 \\
\dot{h}_{y,off} &= 0 \\
\dot{h}_{z,off} &= 0 \\
\dot{a}_{x,est} &= 0 + \xi_{a_{x,est}} \\
\dot{a}_{y,est} &= 0 + \xi_{a_{y,est}} \\
\dot{a}_{z,est} &= 0 + \xi_{a_{z,est}}
\end{aligned}
$$

However, we need a disceretized model. Hence we use 3.23 to obtain

$$
\begin{aligned}
\phi(t+T) &= T(p_{est} + q_{est}\sin(\phi)\tan(\theta) + r_{est}\cos(\phi)\tan(\theta) + \xi_\phi) + \phi \\
\theta(t+T) &= T(q_{est}\cos(\phi) - r_{est}\sin(\phi) + \xi_\theta) + \theta \\
\psi(t+T) &= T(q_{est}\tfrac{\sin(\phi)}{\cos(\theta)} + r_{est}\tfrac{\cos(\phi)}{\cos(\theta)} + \xi_\psi) + \psi \\
u(t+T) &= T(-q_{est}w + r_{est}v - g\sin\theta + a_{x,est} + \xi_u) + u \\
v(t+T) &= T(-r_{est}u + p_{est}w + g\cos\theta\sin\phi + a_{y,est} + \xi_v) + v \\
w(t+T) &= T(-p_{est}v + q_{est}u + g\cos\theta\cos\phi + a_{z,est} + \xi_w) + w \\
p_n(t+T) &= T(u(c_\psi c_\theta) + c(c_\psi s_\theta s_\phi - s_\psi c_\phi) + w(c_\psi s_\theta c_\phi + s_\psi s_\phi) + \xi_{p_n}) + p_n \\
p_e(t+T) &= T(u(s_\psi c_\theta) + v(s_\psi s_\theta s_\phi + c_\psi c_\phi) + w(s_\psi s_\theta c_\phi - c_\psi s_\phi) + \xi_{p_e}) + p_e \\
h(t+T) &= T(-u(s_\theta) + v(c_\theta s_\phi) + w(c_\theta c_\phi) + \xi_h) + h \\
p_{est}(t+T) &= T(\xi_{p_{est}}) + p_{est} \\
q_{est}(t+T) &= T(\xi_{q_{est}}) + q_{est} \\
r_{est}(t+T) &= T(\xi_{r_{est}}) + r_{est} \\
p_{off}(t+T) &= p_{off} \\
q_{off}(t+T) &= q_{off} \\
r_{off}(t+T) &= r_{off} \\
a_{x,off}(t+T) &= a_{x,off} \\
a_{y,off}(t+T) &= a_{y,off} \\
a_{z,off}(t+T) &= a_{z,off} \\
h_{x,off}(t+T) &= h_{x,off} \\
h_{y,off}(t+T) &= h_{y,off} \\
h_{z,off}(t+T) &= h_{z,off} \\
a_{x,est}(t+T) &= T(\xi_{a_{x,est}}) + a_{x,est} \\
a_{y,est}(t+T) &= T(\xi_{a_{y,est}}) + a_{y,est} \\
a_{z,est}(t+T) &= T(\xi_{a_{z,est}}) + a_{z,est}
\end{aligned}
\tag{C.1}
$$

where T is a sample period.

As the GPS output data are acquired less frequently than the output data of the other sensors, we will express the GPS outputs separately. GPS output equations in $y(t) = g(x,u,t) + \eta(t)$ are

$$
\begin{aligned}
y_{p_n} &= p_n + \eta_{y_{gps,n}} \\
y_{p_e} &= p_e + \eta_{y_{gps,e}} \\
y_h &= h + \eta_{y_{gps,h}}
\end{aligned}
$$

and the output equations of the other sensors are

$$\begin{aligned}
y_{h_x} &= h_x(c_\psi c_\theta) + h_y(s_\psi c_\theta) - h_z(s_\theta) + h_{x,off} + \eta_{y_{mag,x}} \\
y_{h_y} &= h_x(c_\psi s_\theta s_\phi - s_\psi c_\phi) + h_y(s_\psi s_\theta s_\phi + c_\psi c_\phi) + h_z(c_\theta s_\phi) + h_{y,off} + \eta_{y_{mag,y}} \\
y_{h_z} &= h_x(c_\psi s_\theta c_\phi + s_\psi s_\phi) + h_y(s_\psi s_\theta c_\phi - c_\psi s_\phi) + h_z(c_\theta c_\phi) + h_{z,off} + \eta_{y_{mag,z}} \\
y_p &= p_{est} + p_{off} + \eta_{y_{gyro,p}} \\
y_r &= q_{est} + q_{off} + \eta_{y_{gyro,q}} \\
y_q &= r_{est} + r_{off} + \eta_{y_{gyro,r}} \\
y_{a_x} &= a_{x,est} + a_{x,off} + \eta_{y_{acc,x}} \\
y_{a_y} &= a_{y,est} + a_{y,off} + \eta_{y_{acc,y}} \\
y_{a_z} &= a_{z,est} + a_{z,off} + \eta_{y_{acc,z}}
\end{aligned}$$

C.2 Jacobians

The size of state matrix **A** is $[24, 24]$ and we express it as

$$\frac{\partial f(x,u,t)}{\partial x} = A(x) = \begin{pmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,24} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,24} \\ \vdots & \vdots & \ddots & \vdots \\ A_{24,1} & A_{24,2} & \cdots & A_{24,24} \end{pmatrix} \tag{C.2}$$

We assume now the substitutions (3.4) and also set

$$t_\theta = \tan(\theta),$$

We also assume the discretized state space model (C.1). The elements of A in (C.2) are

$$\begin{aligned}
A_{1,1} &= T(q_{est}(t_\theta c_\phi) - r_{est}(t_\theta s_\phi)) + 1 & A_{4,2} &= T(-q_{est} c_\theta) \\
A_{1,2} &= T\frac{q_{est}(s_\phi)+r_{est}(c_\phi}{\theta^2}) & A_{4,5} &= T(r_{est}) \\
A_{1,10} &= T & A_{4,6} &= T(-q_{est}) \\
A_{1,11} &= T(s_\phi t_\theta) & A_{4,11} &= T(-w) \\
A_{1,12} &= T(c_\phi t_\theta) & A_{4,12} &= T(v) \\
A_{2,1} &= T(-q_{est} s_\phi - r_{est} c_\phi) & A_{4,22} &= T
\end{aligned}$$

$$
\begin{aligned}
A_{2,11} &= T(c_\phi)\\
A_{2,12} &= T(-s_\phi)\\
A_{3,1} &= T\tfrac{q_{est}c_\phi - r_{est}s_\phi}{c_\theta}\\
A_{3,2} &= T\tfrac{(q_{est}s_\phi + r_{est}c_\phi)s_\theta}{c_\theta^2}\\
A_{3,11} &= T\tfrac{s_\phi}{c_\theta}\\
A_{3,12} &= T\tfrac{c_\phi}{c_\theta}\\
A_{2,1} &= T(-q_{est}s_\phi - r_{est}c_\phi)\\
A_{2,11} &= T(c_\phi)\\
A_{2,12} &= T(-s_\phi)\\
A_{3,1} &= T\tfrac{q_{est}c_\phi - r_{est}s_\phi}{c_\theta}\\
A_{3,2} &= T\tfrac{(q_{est}s_\phi + r_{est}c_\phi)s_\theta}{c_\theta^2}\\
A_{3,11} &= T\tfrac{s_\phi}{c_\theta}\\
A_{3,12} &= T\tfrac{c_\phi}{c_\theta}
\end{aligned}
\qquad
\begin{aligned}
A_{5,1} &= T(gc_\theta c_\phi)\\
A_{5,2} &= T(-gs_\theta s_\phi)\\
A_{5,4} &= T(-r_{est})\\
A_{5,6} &= T(p_{est})\\
A_{5,10} &= T(w)\\
A_{5,12} &= T(-u)\\
A_{5,23} &= T\\
A_{6,1} &= T(-gc_\theta s_\phi)\\
A_{6,2} &= T(-gs_\theta c_\phi)\\
A_{6,4} &= T(q_{est})\\
A_{6,5} &= T(-p_{est})\\
A_{6,10} &= T(-v)\\
A_{6,11} &= T(u)\\
A_{6,24} &= T
\end{aligned}
$$

$$
\begin{aligned}
A_{7,1} &= T(v(s_\psi s_\phi + c_\psi s_\theta c_\phi) + w(s_\psi c_\phi - c_\psi s_\theta s_\phi))\\
A_{7,2} &= T(-uc_\psi s_\theta + vc_\psi c_\theta s_\phi + wc_\psi c_\theta c_\phi)\\
A_{7,3} &= T(-us_\psi c_\theta + v(-s_\psi s_\theta s_\phi - c_\psi c_\phi) + w(c_\psi s_\phi - s_\psi s_\theta c_\phi))\\
A_{7,4} &= T(c_\psi c_\theta)\\
A_{7,5} &= T(c_\psi s_\theta s_\phi - s_\psi c_\phi)\\
A_{7,6} &= T(s_\psi s_\phi + c_\psi s_\theta c_\phi)\\
A_{8,1} &= T(v(s_\psi s_\theta c_\phi - c_\psi s_\phi) + w(-s_\psi s_\theta s_\phi - c_\psi c_\phi))\\
A_{8,2} &= T(-us_\psi s_\theta + vs_\psi c_\theta s_\phi + ws_\psi c_\theta c_\phi)\\
A_{8,3} &= T(uc_\psi c_\theta + v(c_\psi s_\theta s_\phi - s_\psi c_\phi) + w(s_\psi s_\phi + c_\psi s_\theta c_\phi))
\end{aligned}
$$

$$
\begin{aligned}
A_{8,4} &= T(s_\psi c_\theta) & A_{9,2} &= T(-uc_\theta - vs_\theta s_\phi - ws_\theta c_\phi)\\
A_{8,5} &= T(c_\psi c_\phi + s_\psi s_\theta s_\phi) & A_{9,4} &= T(-s_\theta)\\
A_{8,6} &= T(s_\psi s_\theta c_\phi - c_\psi s_\phi) & A_{9,5} &= T(c_\theta s_\phi)\\
A_{9,1} &= T(vc_\theta c_\phi - wc_\theta s_\phi) & A_{9,6} &= T(c_\theta c_\phi)
\end{aligned}
$$

$$A_{2,2} = A_{3,3} = \ldots = A_{24,24} = 1$$

The remaining components of $\mathbf{A}$ that are not in the list above are equal to zero.

We will now express matrix C_{GPS} for GPS data, which size is $[3, 24]$, and matrix C_{sens}, which size is $[9, 24]$, in the same way as (C.2)

$$\frac{\partial g_{GPS}(x,u,t)}{\partial x} = C_{GPS}(x) = \begin{pmatrix} C_{GPS,1,1} & C_{GPS,1,2} & \cdots & C_{GPS,1,24} \\ C_{GPS,2,1} & C_{GPS,2,2} & \cdots & C_{GPS,2,24} \\ C_{GPS,3,1} & C_{GPS,3,2} & \cdots & C_{GPS,3,24} \end{pmatrix} \tag{C.3}$$

$$\frac{\partial g_{sens}(x,u,t)}{\partial x} = C_{sens}(x) = \begin{pmatrix} C_{sens,1,1} & C_{sens,1,2} & \cdots & C_{sens,1,24} \\ C_{sens,2,1} & C_{sens,2,2} & \cdots & C_{sens,2,24} \\ \vdots & \vdots & \ddots & \vdots \\ C_{sens,9,1} & C_{sens,9,2} & \cdots & C_{sens,9,24} \end{pmatrix} \tag{C.4}$$

The elements of C_{GPS} corresponding to GPS outpus in (C.3) are

$$C_{GPS,1,7} = 1, \quad C_{GPS,2,8} = 1, \quad C_{GPS,3,9} = 1$$

All other elements of C_{GPS} are equal to zero. Finally, the elements of C_{sens} are

$$\begin{aligned}
C_{sens,1,2} &= -h_{g0,x}c_\psi s_\theta - h_{g0,y}s_\psi s_\theta - h_{g0,z}c_\theta \\
C_{sens,1,3} &= -h_{g0,x}s_\psi c_\theta + h_{g0,y}c_\psi c_\theta \\
C_{sens,1,19} &= 1 \\
C_{sens,2,1} &= h_{g0,x}(s_\psi s_\phi + c_\psi s_\theta c_\phi) + h_{g0,y}(s_\psi s_\theta c_\phi - c_\psi s_\phi) + h_{g0,z}c_\theta c_\phi \\
C_{sens,2,2} &= h_{g0,x}c_\psi c_\theta s_\phi + h_{g0,y}s_\psi c_\theta s_\phi - h_{g0,z}s_\theta s_\phi \\
C_{sens,2,3} &= h_{g0,x}(-s_\psi s_\theta s_\phi - c_\psi c_\phi) + h_{g0,y}(-c_\psi s_\theta s_\phi - s_\psi c_\phi) \\
C_{sens,2,20} &= 1 \\
C_{sens,3,1} &= h_{g0,x}(s_{ps}c_{fi} - c_\psi s_\theta s_\phi) + h_{g0,y}(-s_\psi s_\theta s_\phi - c_\psi c_\phi) - h_{g0,z}c_\theta s_\phi \\
C_{sens,3,2} &= h_{g0,x}c_\psi c_\theta c_\phi + h_{g0,y}s_\psi c_\theta c_\phi - h_{g0,z}s_\theta c_\phi \\
C_{sens,3,3} &= h_{g0,x}(c_\psi s_\phi - s_\psi s_\theta c_\phi) + h_{g0,y}(s_\psi s_\phi + c_\psi s_\theta c_\phi) \\
C_{sens,3,21} &= 1
\end{aligned}$$

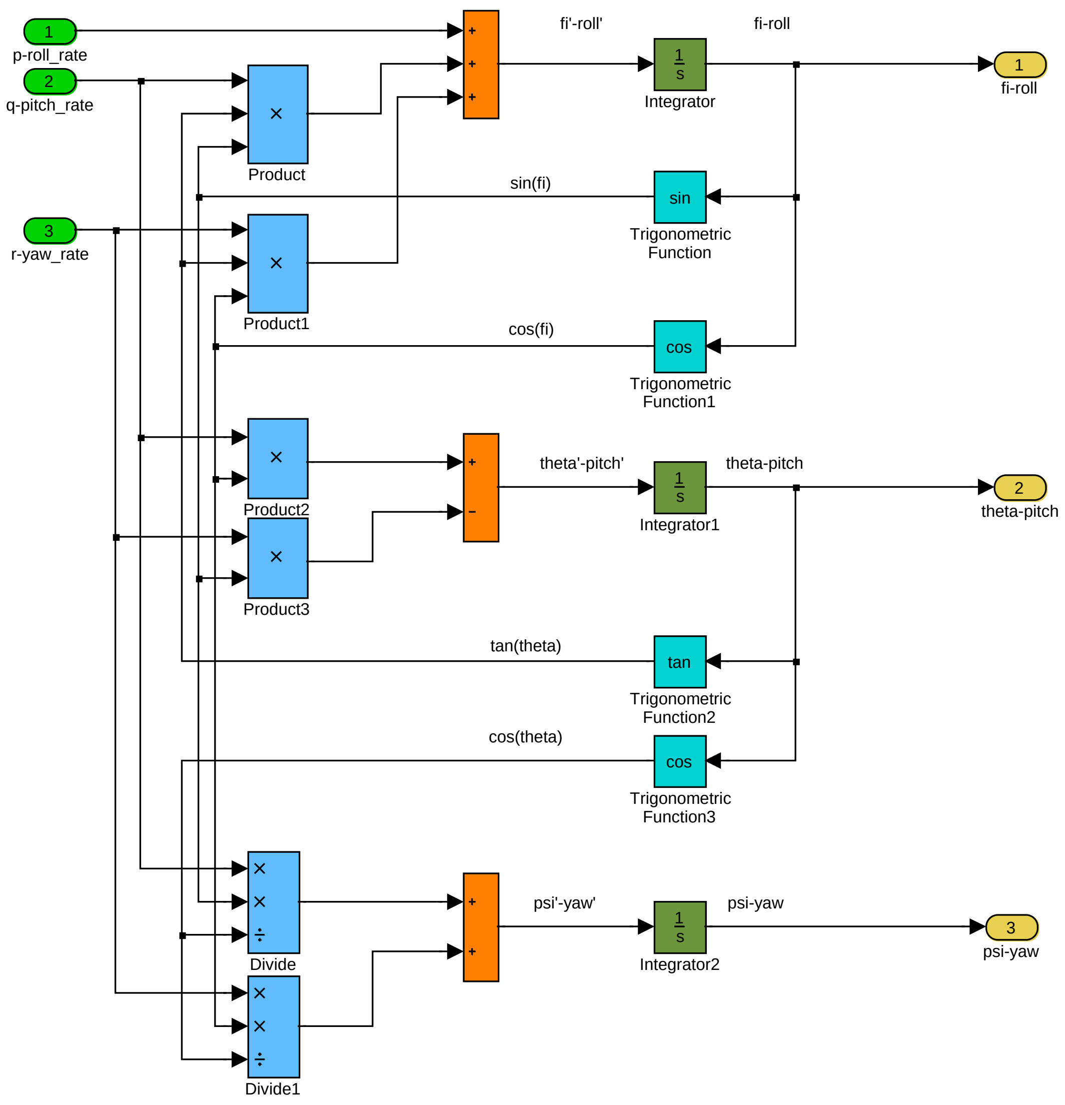
$$\begin{aligned}
C_{sens,4,10} &= 1, \quad C_{sens,4,13} = 1 \\
C_{sens,5,11} &= 1, \quad C_{sens,5,14} = 1 \\
C_{sens,6,12} &= 1, \quad C_{sens,6,15} = 1 \\
C_{sens,7,16} &= 1, \quad C_{sens,7,22} = 1 \\
C_{sens,8,17} &= 1, \quad C_{sens,8,23} = 1 \\
C_{sens,9,18} &= 1, \quad C_{sens,9,24} = 1
\end{aligned}$$

The elements of C_{sens} that have not been mentioned are equal to zero.

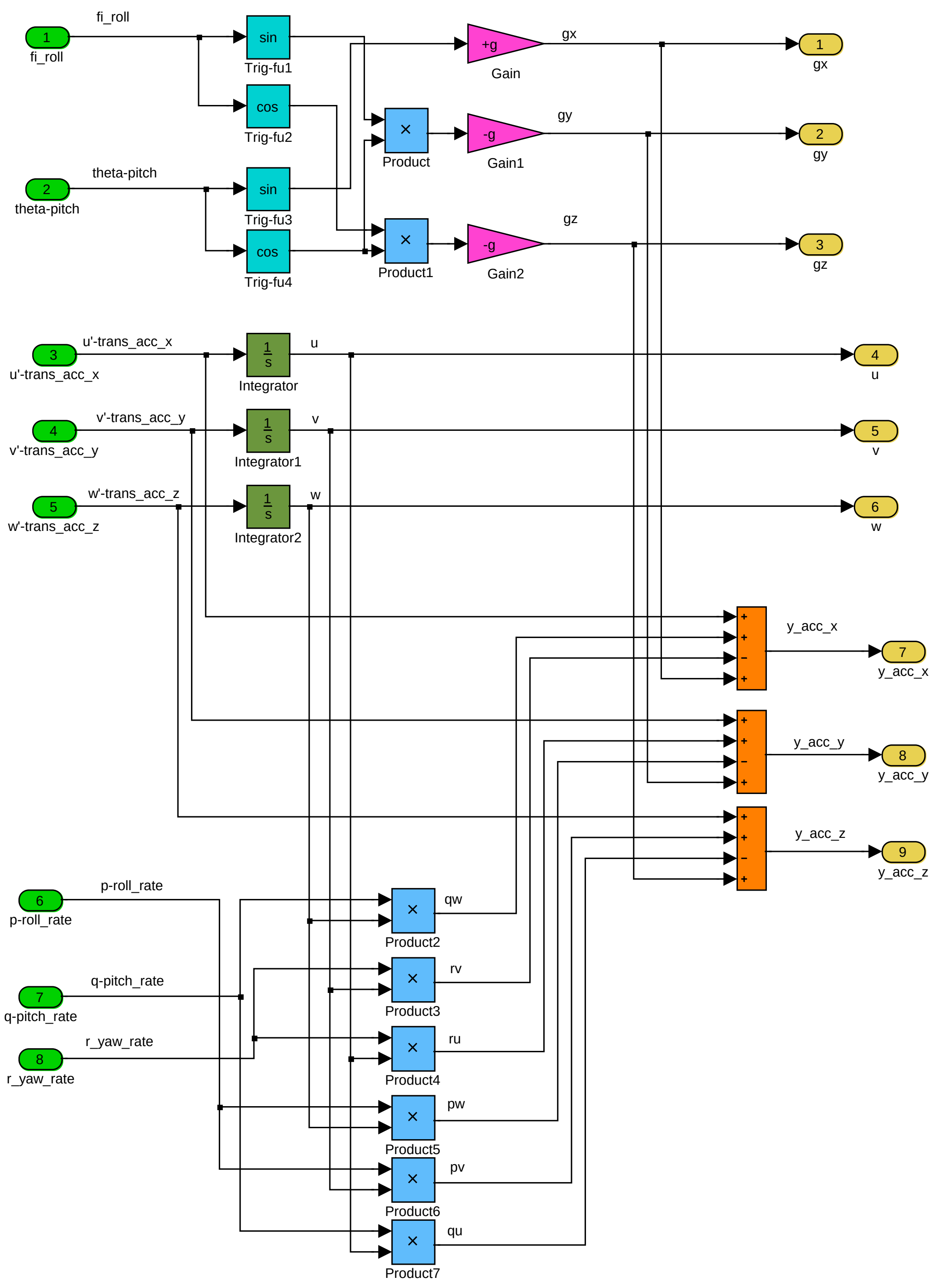
Appendix D

Simulator subsystems

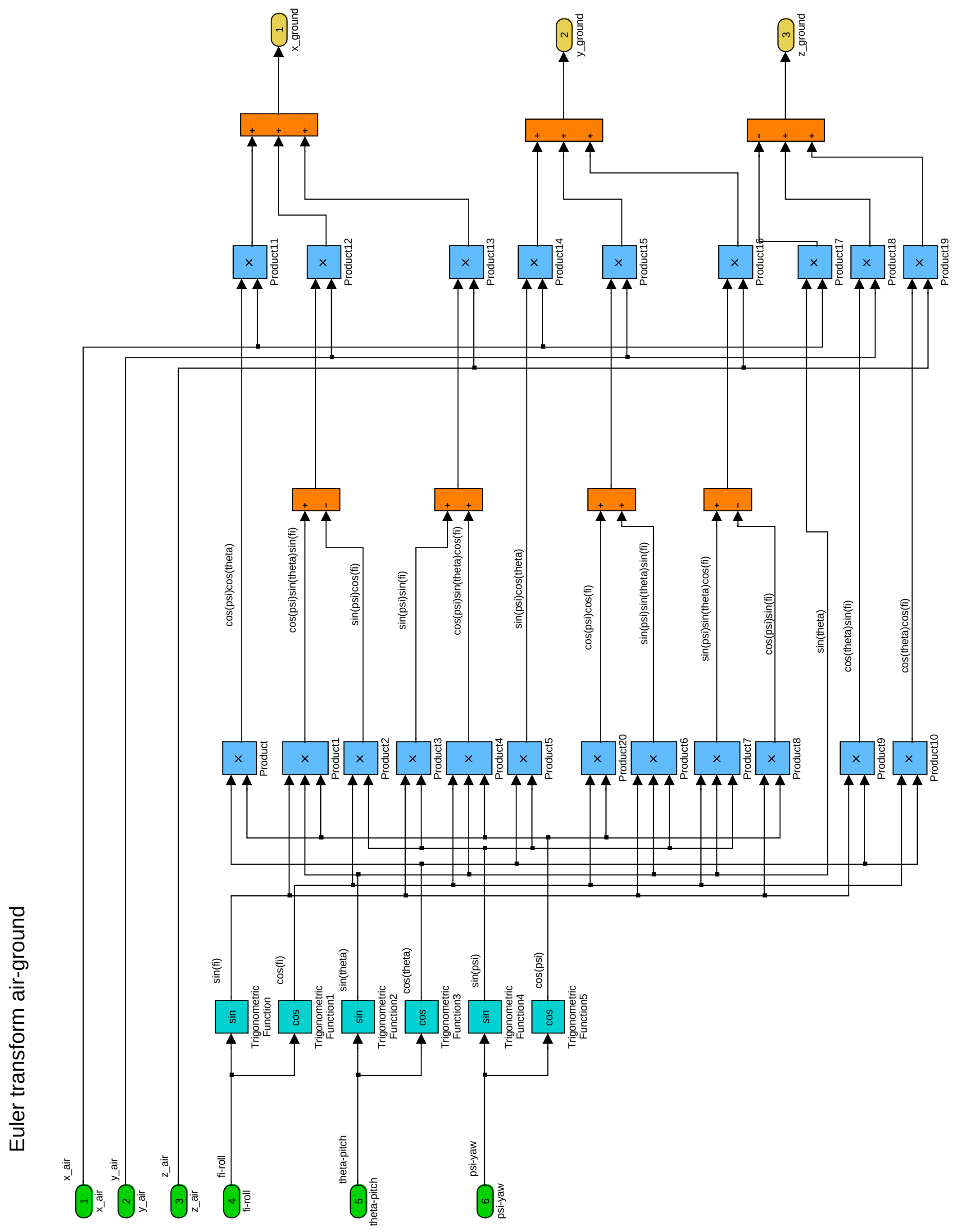
Euler angles



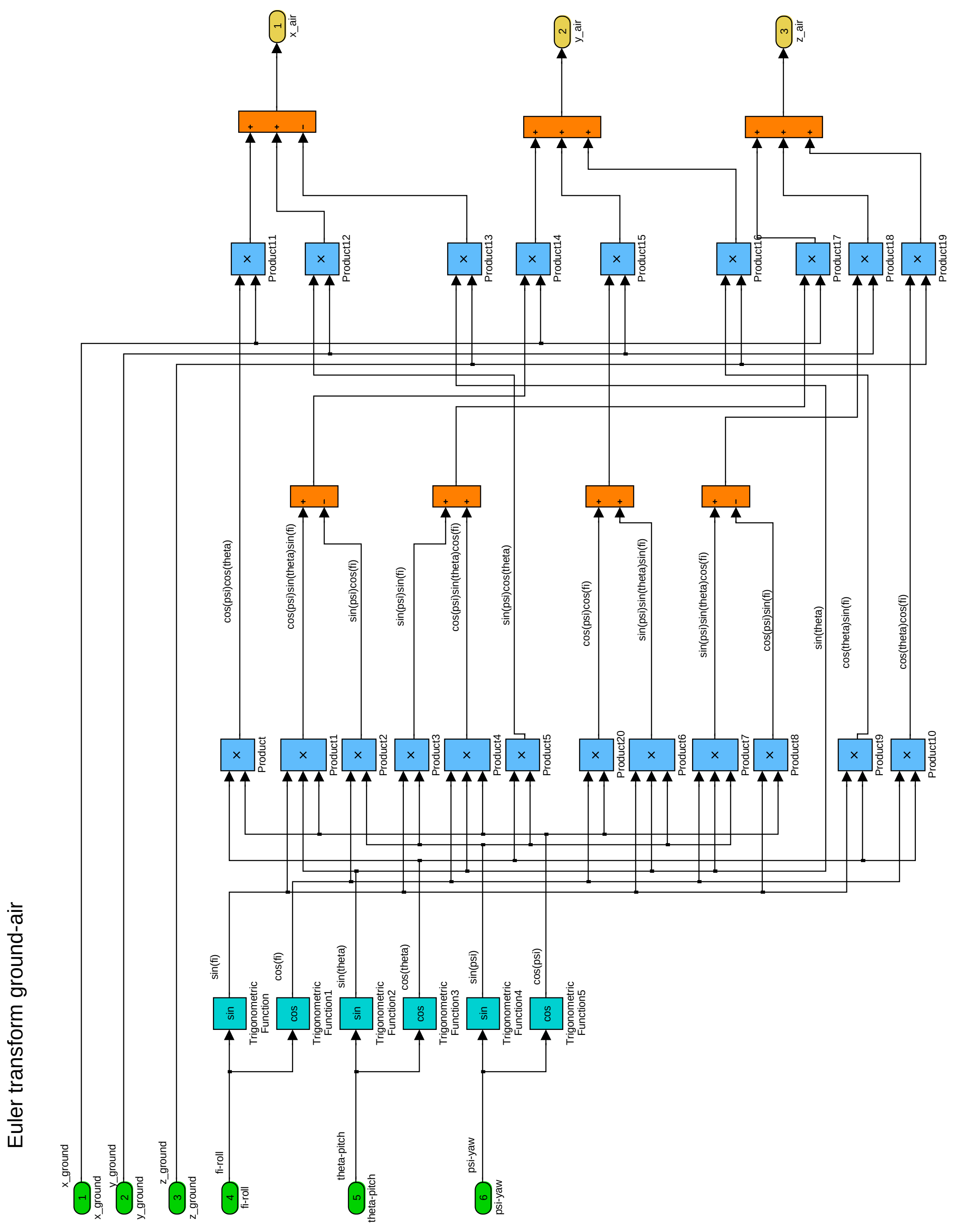
Accelerometers



Euler transform air-ground



Euler transform ground-air



Appendix E

EKF and INU simulation results

E.1 INU simulation outputs

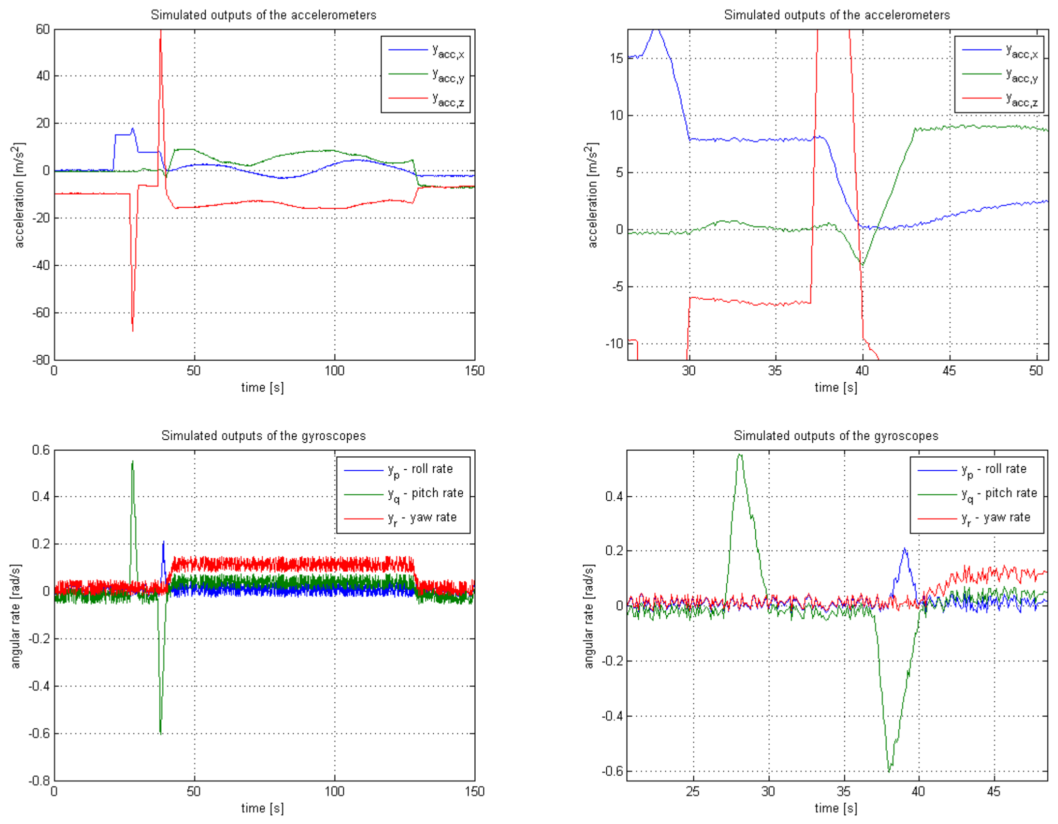


Figure E.1: Accelerometers and gyros outputs of the INU simulator to the left and the detail to the right

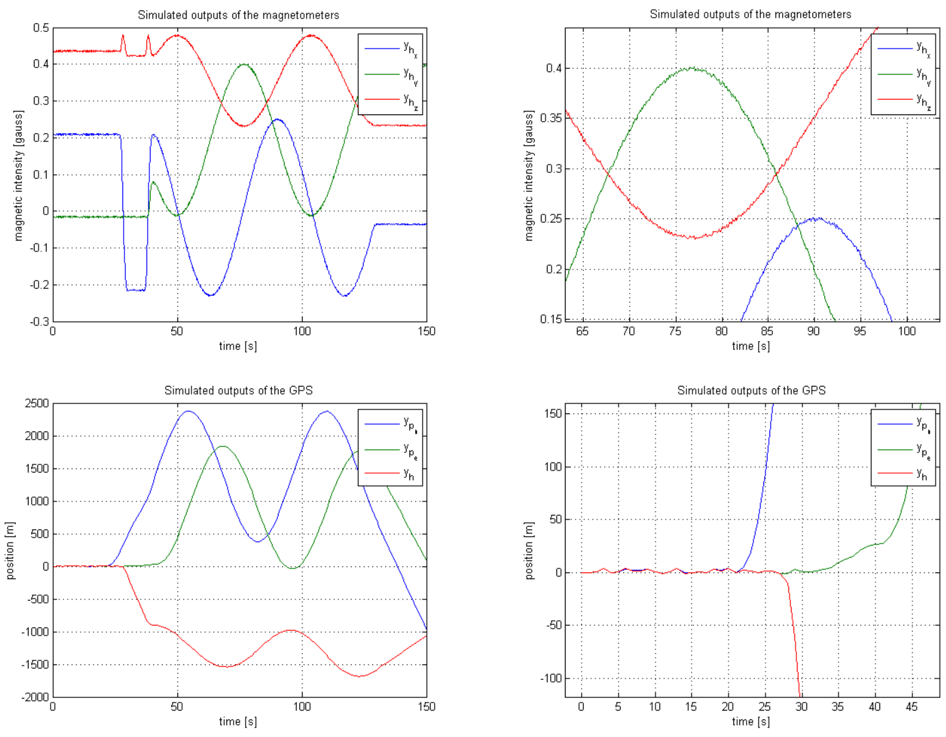


Figure E.2: Magnetometers and GPS outputs of the INU simulator to the left and the detail to the right

E.2 EKF simulation results

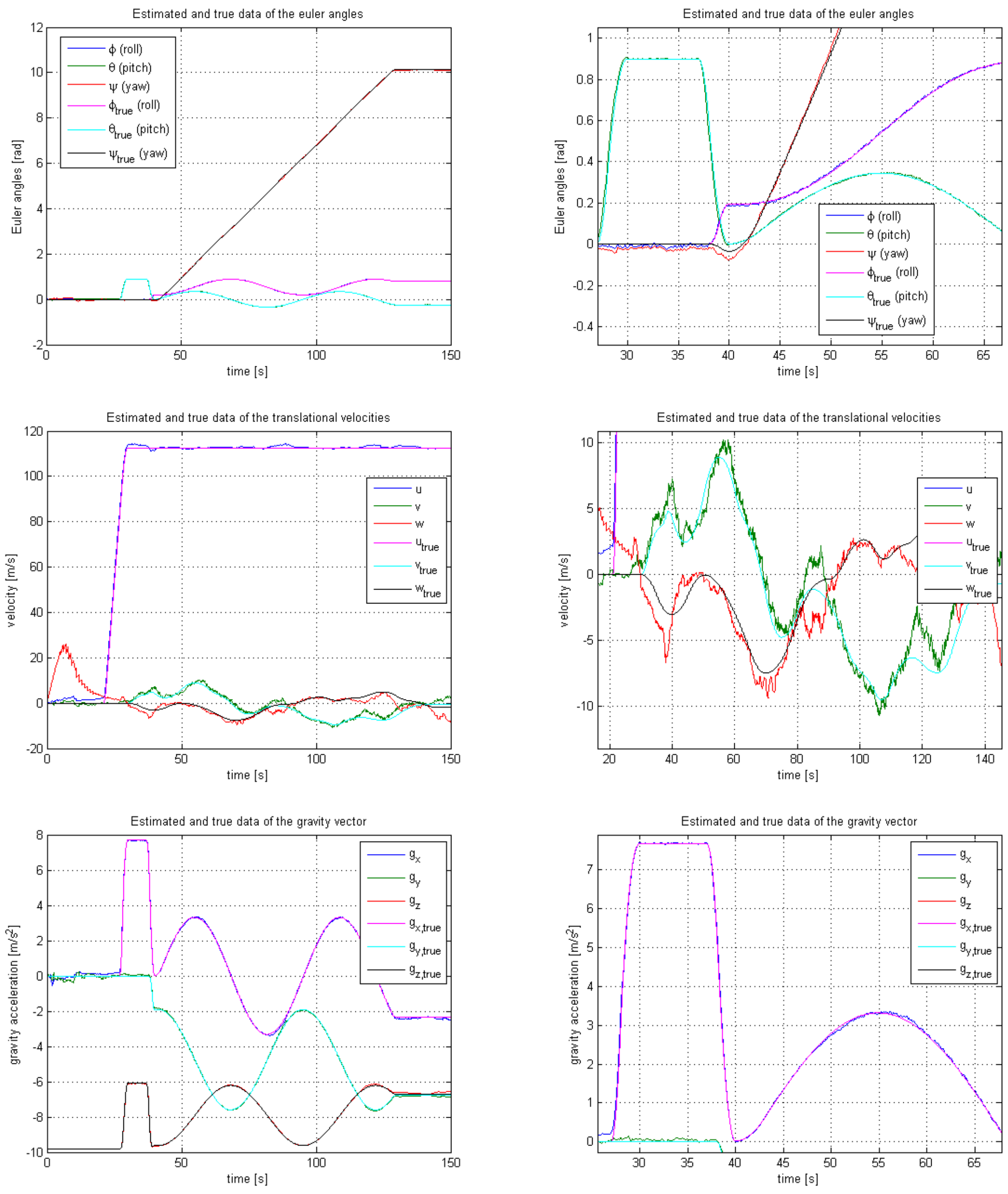


Figure E.3: EKF results: Euler angles, trans. velocities and gravity vector

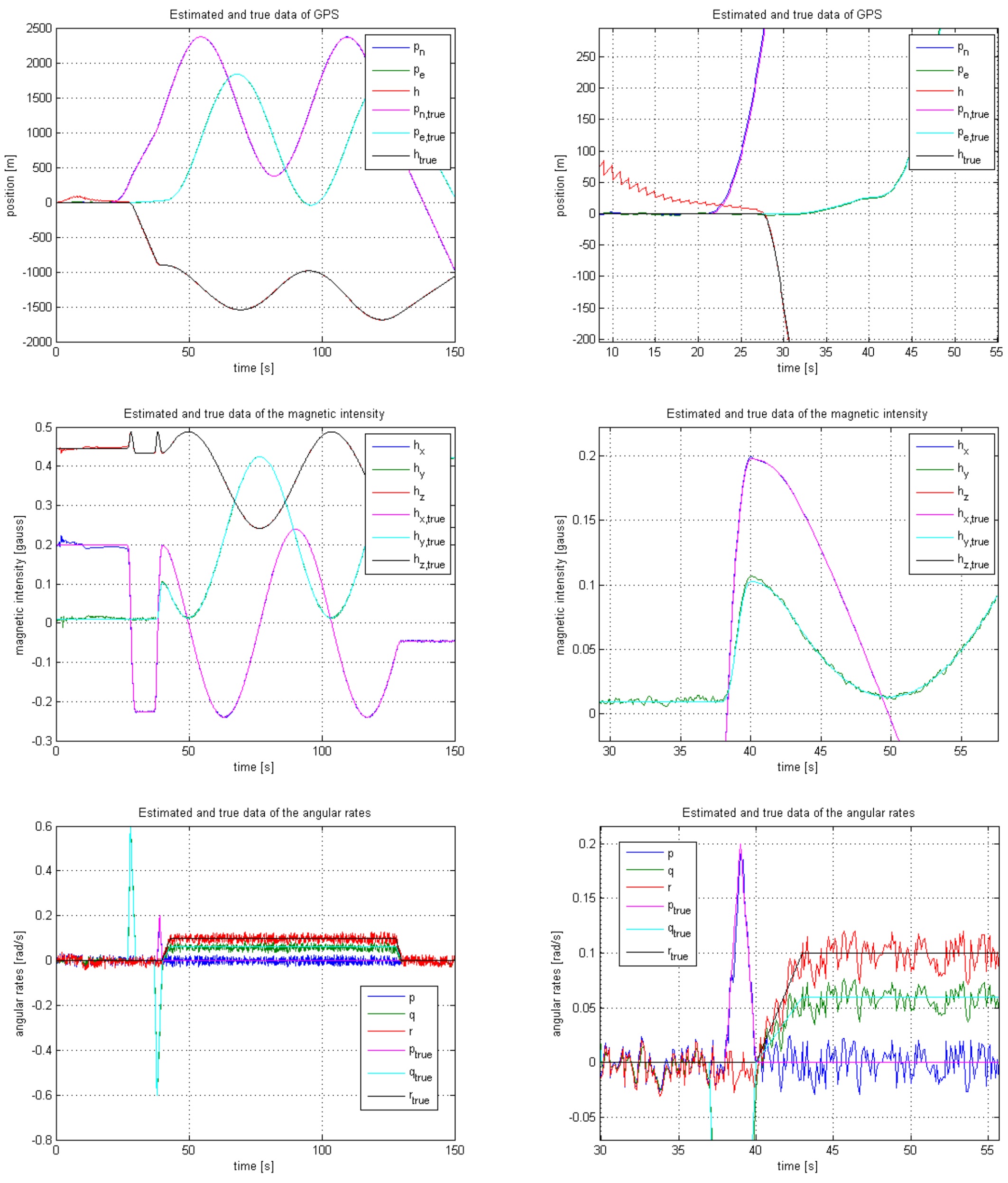


Figure E.4: EKF results: GPS, mag. intensity and angular rates

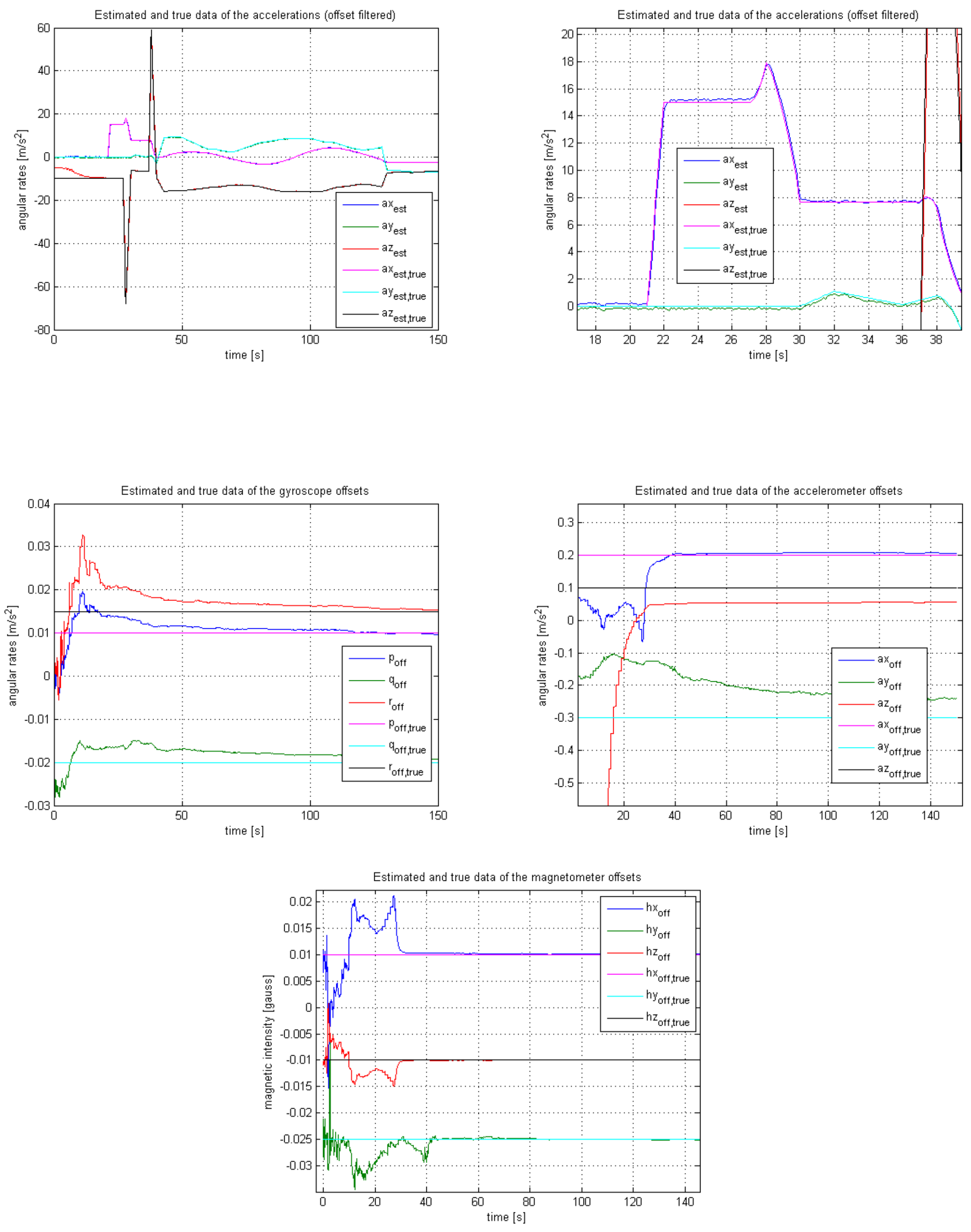


Figure E.5: EKF results: acceleration and offset estimates

Appendix F

CD contents

- this text in **PDF** format
- datasheets of the used sensors and MCU
- MATLAB/Simulink
  - INU simulator Simulink file as well as the simulator initialization m-file
  - simulated data in workspace form
  - EKF design code
  - the MATLAB code containing the algorithm for MCU code optimization
- C-code for MCU and its manual