# Position Estimation Using an Inertial Measurement Unit Without Tracking System

**Johnny Leporcq**

**School of Electrical Engineering**

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 25.05.2018

**Thesis supervisor and advisor:**

Prof. Risto Wichman

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Johnny Leporcq

Title: Position Estimation Using an Inertial Measurement Unit Without Tracking System

Date: 25.05.2018 Language: English Number of pages: 7+57

Department of Signal Processing and Acoustics

Professorship: Signal processing

Supervisor and advisor: Prof. Risto Wichman

This thesis aims to estimate the position of an inertial measurement unit (IMU) without any tracking device such as GPS. The work includes the calibration of the accelerometer with particle swarm optimization (PSO) to solve the equation, the gyrometer with the extended Kalman filter (EKF) and the magnetometer also with EKF. The calibration is realized with the data from the sensors and Matlab. When the calibration is done, the acceleration is obtained from the accelerometer and the gyrometer. The algorithm employs mostly rotation matrix theory.
The performance of the algorithm depends on the success of the calibration. A small error in the estimation of the acceleration leads to a wrong result. This was, nevertheless, to be expected as a double integration with respect to time of a signal with remaining traces of bias is doomed to fail without any correction algorithms. Unfortunately, a working algorithm could not be achieved, pointing out that it may be difficult to realize one without external devices such as GPS.

Keywords: Inertial measurement unit (IMU), Extended Kalman filter (EKF), Rotation matrix, Orthogonalization, Particle swarm optimization (PSO), Calibration

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Johnny Leporcq

Työn nimi: Sijainnin estimointi inertiamittausyksiköllä ilman paikannusjärjestelmää

Päivämäärä: 25.05.2018 Kieli: Englanti Sivumäärä: 7+57

Signaalinkäsittely laitos

Professuuri: Signaalinkäsittely

Työn valvoja: Prof. Risto Wichman

Työn ohjaaja: Prof. Risto Wichman

Tässä työssä estimoidaan inertiamittausyksikön (IMU) sijaintia käyttämättä GPS-laitetta. Työ sisältää kiihtyvyysanturin kalibroinnin hiukkasparvioptimointialgorithmilla (PSO), gyroskoopin laajennetulla Kalmanin suodattimella (EKF) ja kompassin EKF:lla. Kalibrointi on suoritettu vain anturien arvoilla ja Matlab-sovelluksella. Anturin kiihtyvyys saa kalibroiduilta kiihtyvyysanturilta ja kompassilta. Algorithmi käyttää rotaatiomatriisin teoria.
Algorithmi tehokkuus riippuu kalibroinnista. Pienikin estimointivirhe aiheittaa väärän tuloksen.Työn tulokset voitiin ennustaa koska tuplaintegrointi pienellä virhellä johtaa helposti ja nopeasti tulokset väärään suuntaan. Työn algoritmi vaatii korjausalgoritmin joka pystyisi poistamaan integroinnin virheen. Valitettavasti toimivaa algoritmia ei löydettu, joka viittaa siihen, että sen toteutaminen saattaa olla vaikeaa ilman apulaitetta, kuten GPS-laitetta.

Avainsanat: Inertiamittausyksikkö (IMU), Laajennettu Kalmanin suodatin (EKF), Rotaatiomatriisi, Ortogonalisoiminen, Hiukkasparvioptimointi (PSO), Kalibrointi

# Preface

I want to thank Professor Wichman for the subject and my girlfriend Vanessa for her patience.

Helsinki, 25.05.2018

Johnny Leporcq

# Contents

# Symbols and abbreviations

## Symbols

| | |
|---|---|
| $\vec{g}$ | Gravitational vector |
| $\omega$ | Angular speed |
| $\vec{a}$ | Acceleration vector |
| $\vec{v}$ | Speed vector |
| $\vec{p}$ | Position vector |
| $\vec{\omega}$ | Angular velocity |
| $\mathbf{I}$ | Matrix identity |
| $\mathbf{J}$ | Jacobian matrix |
| $\phi$ | Pitch |
| $\theta$ | Roll |
| $\psi$ | Yaw |

## Operators

| | |
|---|---|
| $\vec{.}$ | column vector |
| $\|.\|$ | Module of a vector |
| E[.] | The expectation operator |
| $\frac{\mathrm{d}}{\mathrm{d}t}$ | derivative with respect to variable $t$ |
| $\frac{\partial}{\partial t}$ | partial derivative with respect to variable $t$ |
| $\sum_i$ | sum over index $i$ |
| $^{\mathrm{T}}$ | Transpose |
| $\times$ | Cross product |
| $\cdot$ | dot product |
| $\int$ | Integral |

## Abbreviations

| | |
|---|---|
| ADC | Analog Digital Converter |
| EKF | Extended Kalman filter |
| EEMD | Ensemble Empirical Mode Decomposition |
| GPS | Global Positioning System |
| IMU | Inertial Movement Unit |
| MEMS | Micro-Electro-Mechanical Systems |
| MMSE | Minimum Mean Square Error |
| PDF | Probality Density Function |
| PSO | Particle Swarm Optimization |

# 1 Introduction

## 1.1 Motivation

The Inertial Measurement Unit (IMU) are these days frequently used in robotics, in vehicles and, of course, in mobile phones. An IMU is often composed of an accelerometer, a gyrometer and a magnetometer. The latter two sensors are typical navigation instruments that are used in boats or planes. The compass is roughly a special case of magnetometer. The compass only shows the direction of the local magnetic field while a magnetometer measures also its strength. These bulky and very expensive equipment are nowadays replace by MEMS (Micro-Electro-Mechanical Systems) version because of their size and their low cost. Their price goes from 2 $ to over 2000 $ while an high end gyrometer could cost over 100000 $. There are a large number of MEMS IMU manufacturer, such as Xsens, Analog devices, Honeywell or MEMSIC which provide high end (also called tactical grade) IMU and STMicroelectronics, mCube, Bosch and TDK InvenSense which provides cheap IMU that are used, for instance, in mobile phone. In this work, the manufacturer of the IMU is TDK Invensense and the board manufacturer is CJMCU. The price of the board was about 9 $. A tactical grade IMU are sold fully calibrated while the low cost ones are sold with some factory calibrations which are stored in the IMU registers. The calibration only covers the scaling factors of each axes.

The motivation behind this thesis resides in obtaining the position of the sensor without tracking system nor expensive calibration equipment. The position estimation using an IMU has been obviously studied in numerous papers but there is in most cases a tracking system that corrects any deviation of the position estimated by the IMU. The tracking device is usually a GPS [1] if outdoor and indoor it can be WiFi network [2] or RFID (Radio Frequency Identification) [3] or a camera [4]. The reason of this thesis is too see if it is possible to estimate the position without heavy correction of the position by an external sensor. The word "heavy" was used in the previous sentence because in many papers it feels that the tracking system alone would give the same results because the position estimated by the IMU does drift and is at least every second corrected by the tracking system.

## 1.2 Structure

In the first part, a global view of the sensors available on the electronic board will be given. It covers briefly the theory of the physical phenomenas that make the sensors operate. In the following chapter, the mathematical theory on rotation matrix needed to understand the work. The theory on rotation matrices is long and tedious but, in this paper, it will be only briefly described. The theory is followed by the theoretical presentation of the algorithms present in the work. Afterward, in the fifth chapter, a simulation accompanies and test the theory. The simulation is realized using synthetic signals that simulate the data signals of the sensors. In the sixth chapter, a brief description of the electronic equipment used in this work is

given. The functioning and programming of the $I^2C$ bus will only be briefly examined. In the same chapter, the experiments on the data obtained by the sensor will be discussed. In the final chapter, there will be a discussion on the results with some possible future improvements that could help to improve this work.

# 2 Sensors

In this part, the general knowledge on the sensors present in this work will be examined.

## 2.1 Accelerometer

An accelerometer is a device that measures the acceleration of a movement [5]. The MEMS sensor senses the magnitude of the change caused by the acceleration and converts it into a electrical value using capacitive, piezoeletric or piezoresistive technique. In any case, the sensor senses the acceleration due to the Earth's gravity but the deviation from free fall. In other words, when an accelerometer is in free fall situation, it will measure a magnitude of $0 \text{m.s}^{-2}$.

An accelerometer can be described by a mechanical combination of a proof mass,

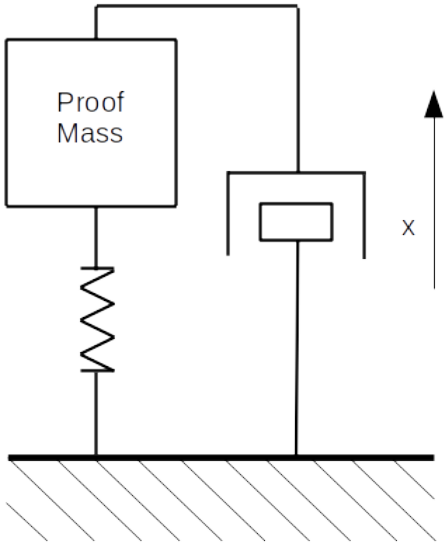


Figure 2.1: Mechanical model of an one-axis accelerometer

a spring and a damping system. This is more or less a seismometer (fig.2.1). The damping system is necessary to avoid oscillations. The system is in equilibrium state when there is no acceleration present on the sensor. This means that only the acceleration acting on the proof mass is the Earth's gravitational acceleration $\vec{g}$. In a static state, the accelerometer in figure 2.1 will give a value of $-g$. When the sensor moves, the system moves from the equilibrium state to enter into a non-static state. The difference in values of those two states is the actual acceleration applied to the sensor. In theory, the acceleration $a(t)$ on the sensor can be written as

$$a(t) = a_m(t) - g \tag{2.1}$$

where $a_m(t)$ is the value measured by the accelerometer.

In this work, a MEMS-version of the accelerometer is utilized. It operates the same way as the mechanical description above. The sensor are usually constructed from

a silicon wafer (Si). The silicon is a good structural material which has a small expansion coefficient and a Young's modulus suitable and no mechanical hysteresis. The sensor is often realized with a proof mass held with four support beams. The beams, or cantilevers, work as spring but also as a damping system. For the capacitive-type, the silicon wafer is sealed between two glass wafers with aluminum sheets on the inner sides. This forms a capacitor with a capacitance that varies with the movement of the proof mass. Three degree of freedom requires three accelerometer, one for each of the three axes.
Accelerometers are used in numerous applications such as monitoring motor fault situation[6], flow rate in a given pipe [7] using the changes in vibrations. The accelerometer data is the main part of this work because the position is determined by integrating the deviation of the acceleration two times with respect to time. It will be explained later that the accelerometer data for the purpose of this work cannot be employed alone.

## 2.2 Gyrometer

The gyrometer is a device that measures the angular speed of the sensor. The basic form of the gyroscope is made of a spinning wheel whom axis is free to take any position. When the wheel is rotated, its axis positions itself according to the conservation of the angular momentum. This type is also called spinning-mass gyroscope. The gyrometer has several designs such as ring laser or the interferometric fiber-optic, quantum or vibratory. In this work, only the latter will be discussed. Other gyrometers are meant for great precision in laboratories.
The vibratory gyroscope is the technology used in the MEMS-technology. The sensor is composed of a vibrating part which can be a string, a beam, a ring or two proof masses which undergo a simple harmonic motion. The two proof masses construction is, in most case, used in MEMS. The MEMS gyrometer is, like the accelerometer, realized from a silicon wafer. Silicon is used for the same reason as it is discussed in the accelerometer section 2.1, i.e. excellent Young's modulus and no mechanical hysteresis. The silicon wafer is then sealed between two glass wafers covered with aluminum foil.
The two proof masses are on the same plane and aligned on the same axis. They vibrate in the direction of their axis at the exact same frequency but anti-phased. When the sensor is rotated, a force is created at the proof masses. This force is called the Corialis force and is given by

$$\vec{F} = -2m\vec{v} \times \vec{\Omega} \tag{2.2}$$

where $m$ is the mass of the proof mass, $\vec{v}$ the velocity of the proof mass and $\vec{\Omega}$ is the rotation rate of the system. Because the masses are vibrating in an anti-phase movement, the force on themselves will always be in opposite direction compared to the other one. These forces incline the proof masses whose tilt is sensed by a capacitor. The capacitor plates are above and under the proof masses which form the capacitor.
The three dimensional MEMS-gyrometer works the same way. It is composed of four

proof masses where two masses on the same axis measure the pitch while the two others measure the roll and the yaw. The pitch and the roll have the capacitor plates placed under and over the masses while the yaw has its capacitor plates on the axes of the proof masses.
The gyrometer are used in many applications but in most of these applications it is never used alone. The gyrometer is most of the time used to determine the attitude of an object [8].

## 2.3 Magnetometer

The magnetometer measures the direction and the magnitude of the Earth's local magnetic field. In other words, it is a compass which measures also the magnitude of the field.
Generally, MEMS magnetometers work using the Hall effect. The Hall effect is a phenomena which consists of the variation of voltage across an electrified conductor when plunged in a magnetic field.
When a piece of conductor is connected to a constant voltage source, the charges within the conductor are uniformly arranged across that conductor. It results to a zero voltage between the extremities of the conductor. When the conductor is placed in a magnetic field, a force is created and its equation is

$$\vec{F_L} = q\left(\vec{E} + (\vec{v} \times \vec{B})\right) \tag{2.3}$$

where $q$ is the electron charge, $\vec{E}$ the electric field, $\vec{B}$ the magnetic field and $\vec{v}$ the velocity of the electron. This force is also know as Lorentz force. This force will change the course of the electron. This change will modify the distribution of the charges across the conductor which will change the voltage across the conductor. Without entering into too much detail, the Hall voltage $V_H$ is given by

$$V_H = \frac{I\|\vec{B}\|}{qNd} \tag{2.4}$$

where $I$ is the current going through the conductor, $N$ the carrier density across the conductor and $d$ the depth of the conductor. For more details, the document [9] explained the Hall effects in sensors.
The voltage will be proportional to the magnitude of the magnetic field $\vec{B}$ and can be determined, assuming that $I$, $N$ and $d$ are known and constant.
The three axes magnetometer is composed of three Hall effect MEMS sensor placed in the direction of each axes $\vec{x}, \vec{y}, \vec{z}$ which are orthogonal to each others. A magnetic flux concentrator is placed in the top of the three sensors to increase the sensitivity of the magnetometer. The magnetometer is subject to data corruption when an other source of magnetic field is near. it can be a piece of iron which is magnetized, a speakers or most electronic devices around the house. The magnetometer works best outside while inside it should be used with care.

## 2.4 Barometric pressure sensor

The barometric pressure sensor will not be used in this work due to its lack of sensitivity. But because it is present on the sensor module, a few words about it with be given.
The barometric pressure sensor measures the atmospheric pressure. As we go higher away from the Earth's surface, the pressure drops. It is on this principle that the pressure sensor can be used also for height estimation. The MEMS pressure sensor can be designed with a Hall effect sensor like for the magnetometer but also it can be designed with a piezoelectric crystal. The latter is used in most low-cost MEMS. This type of sensor works with the piezoelectric effect. This effect refers to the production of electricity when a piezoelectric material is subject to volume changes. In the pressure sensor, the piezoelectric crystal is tied to a membrane that is subject to atmospheric pressure. And according to the pressure, the voltage will vary. Unfortunately, this sensor has in theory one meter error which empirically becomes about three meters. The sensor is better for measuring large distance to compensate the GPS-measurement error [10]. The GPS is not precise in altitude measurement.

## 2.5 Sensor fusion

Sensor fusion is a method which uses several sensors to retrieve data that would be otherwise difficult or impossible to obtain [11].
Sensor fusion is also referred as data fusion, information fusion, multi-sensor data fusion. In this work, the fusion of the data is realized from the accelerometer and the gyrometer. This fusion is done to obtain the position which would be hardly possible by just using the accelerometer.

# 3 Mathematical basics

In this section, the theory which is used in this work, is briefly described. Theses theories are useful for a better understanding of what was done to complete this work.

## 3.1 Orthonormal basis

The concept of orthonormal basis is important to understand as it is used all along this work. All the basis of the utilized sensors data must be orthonormal to achieve correct calculation when the data fusion is used. First the concept of basis must be described.

### 3.1.1 Basis

A vector basis of a vector space $\mathcal{E}$ of dimension $n$ is defined as a subset of vectors $\{\vec{e_1}, \vec{e_2}, \vec{e_3}, ..\vec{e_n}\} \in \mathcal{E}$ which are linearly independent and span all other vectors in $\mathcal{E}$. In mathematical form,

$$\forall v \in \mathcal{E}, \exists\{a_1, a_2, a_3, .., a_n\} \in \mathbb{R}^n, \vec{v} = a_1\vec{e_1} + a_2\vec{e_2} + a_3\vec{e_3} + ... + e_n\vec{e_n} \tag{3.1}$$

In this thesis, the dimension of the vector bases will always be three, i.e. $n = 3$.

### 3.1.2 Orthonormality

A basis is orthonormal if its vectors are of unit norm and orthogonal to each others. In a vector space $\mathcal{E}$, if the vectors subset $\{\vec{e_1}, \vec{e_2}, \vec{e_3}, ..., \vec{e_n}\}$ composes the orthonormal basis $\mathcal{B}$, then

$$\forall i, j \in \{1, 2, 3, ..., n\}(i \neq j \Rightarrow \vec{e_i} \perp \vec{e_j}), (\|\vec{e_i}\| = \|\vec{e_j}\| = 1) \tag{3.2}$$

where the operator $\|.\|$ is the norm of a vector.

**NOTE**: In this work, the orthonormal basis is always of dimension three. The vectors composing this basis will be named $\vec{e}_x, \vec{e}_y$ and $\vec{e}_z$, which are respectively the unit vectors of the axis $x, y$ and $z$, and following the right hand rule.
The right hand rule can be described mathematically as follows:
assuming that the following vectors $\vec{e}_x, \vec{e}_z$ and $\vec{e}_z$ form an orthonormal basis $\mathcal{B}$, it can then be written that

$$\vec{e}_x \times \vec{e}_y = \vec{e}_z, \quad \vec{e}_y \times \vec{e}_z = \vec{e}_x, \quad \vec{e}_x \times \vec{e}_z = -\vec{e}_y$$

where the operator $\times$ represents the cross product. The right hand rule is the most common rule used to organize a vector basis in engineering.

## 3.2 Rotation matrix

The rotation matrix is one of the tools used to represent the rotations in an Euclidean space. When it comes to IMU, the most common tool is nevertheless the quaternion transformation. The quaternion is a representation of the attitude angles into a three-dimensional complex space. The quaternion has some advantages over the representation of a rotation, as it only needs four scalars while the rotation matrix requires nine scalars. It is also less subject to numerical inaccuracies in floating point calculation [12]. The main disadvantage, besides the non-intuitive calculation, is the computation of rotation which requires more calculation than with matrices.
In this work, only the matrix representation is utilized. This is mostly because it is more intuitive and the work has no limitation in computation memory.
It can be noticed that the data of an ideal sensor is given in three dimensional orthonormal Cartesian coordinate system, which follows the right-hand rule.

### 3.2.1 Properties

In an Euclidean space spanned by an orthonormal basis, a rotation matrix $\mathbf{R}$ is an orthogonal matrix which satisfies the equation

$$\mathbf{R}^{\mathrm{T}}\mathbf{R} = \mathbf{I} \quad \text{with} \quad \det(\mathbf{R}) = 1 \tag{3.3}$$

where $\mathbf{I}$ represents the identity matrix in the respective basis.
In an orthonormal basis $\mathcal{B}$ spanned by the vectors $\vec{e}_x, \vec{e}_z$ and $\vec{e}_z$, a counterclockwise rotation of angle $\phi$ around the axis $x$ can be represented by

$$\mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \tag{3.4}$$

A counterclockwise rotation of angle $\theta$ around the axis $y$ can be represented by the following matrix

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{3.5}$$

A counterclockwise rotation of angle $\psi$ around the axis $z$ can be represented by

$$\mathbf{R}_z(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin\psi & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

with

$$\det(\mathbf{R}_x) = \det(\mathbf{R}_y) = \det(\mathbf{R}_z) = 1 \tag{3.7}$$

The angles $\phi, \theta$ and $\psi$ are also named Euler angles and are respectively called, the roll, the pitch and the yaw. These terms are widely used in navigation to describe the attitude of an object.

### 3.2.2 Rotation of a vector

A rotation of a vector $\vec{u}$ in an orthonormal basis $\mathbf{B}$ is calculated with the following formula

$$\vec{v} = \mathbf{R}\vec{u}, \quad (\vec{u}, \vec{v}) \in \mathcal{B}^2 \tag{3.8}$$

where $\mathbf{R}$ is a rotation matrix and $\vec{v}$ the new vector.
A rotation not around the axes of the basis $\mathcal{B}$ is defined as a series of rotation around the axes. In mathematical terms, a rotation of a vector $\vec{u}$ can be described as six possible combinations of the rotation matrices $\mathbf{R}_x(\phi)$, $\mathbf{R}_y(\theta)$ and $\mathbf{R}_z(\psi)$. For instance, considering the following three of the six rotations possible

$$\vec{v}_1 = \mathbf{R}_x(\phi)\mathbf{R}_y(\theta)\mathbf{R}_z(\psi)\vec{u}, \quad (\vec{u}, \vec{v}) \in \mathcal{B}^2, \tag{3.9}$$

$$\vec{v}_2 = \mathbf{R}_y(\theta)\mathbf{R}_x(\phi)\mathbf{R}_z(\psi)\vec{u}, \quad (\vec{u}, \vec{v}) \in \mathcal{B}^2 \tag{3.10}$$

and

$$\vec{v}_3 = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi)\vec{u}, \quad (\vec{u}, \vec{v}) \in \mathcal{B}^2 \tag{3.11}$$

Since the matrix product is generally not commutative,

$$\vec{v}_1 \neq \vec{v}_2 \neq \vec{v}_3$$

Thus, the order of multiplication in this case is critical and must be known to complete the inverse rotation. This property leads to complications because the order of rotation of a sensor moving randomly is not obvious to determinate.
A way to overcome this problem is to consider each rotation angle to be relatively small. This assumption is solid if the sensor is not prone to rapid rotational movements. The MEMS-sensor has a sampling frequency of about 230 samples per second which is enough to obtain small angles. Considering the rotation matrices 3.4, 3.5 and 3.6, and the Euler angles being small, it can be written that,

$$\mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \phi \\ 0 & -\phi & 1 \end{bmatrix} \tag{3.12}$$

and

$$\mathbf{R}_x(\phi) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & -\theta \\ 0 & 1 & 0 \\ \theta & 0 & 1 \end{bmatrix} \tag{3.13}$$

and

$$\mathbf{R}_x(\phi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin\psi & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \approx \begin{bmatrix} 1 & \psi & 0 \\ -\psi & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

because the cosine of a small angle is approximately one and the sinus of a small angle is the angle itself. Now introducing the matrices 3.12, 3.13 and 3.14 in the equations 3.9, 3.10 and 3.11.

$$\vec{v}_1 = \mathbf{R}_x(\phi)\mathbf{R}_y(\theta)\mathbf{R}_z(\psi)\vec{u} \approx \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \phi \\ 0 & -\phi & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -\theta \\ 0 & 1 & 0 \\ \theta & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & \psi & 0 \\ -\psi & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$\vec{v}_2 = \mathbf{R}_y(\theta)\mathbf{R}_x(\phi)\mathbf{R}_z(\psi)\vec{u} \approx \begin{bmatrix} 1 & 0 & -\theta \\ 0 & 1 & 0 \\ \theta & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \phi \\ 0 & -\phi & 1 \end{bmatrix} \begin{bmatrix} 1 & \psi & 0 \\ -\psi & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and

$$\vec{v}_3 = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi)\vec{u} \approx \begin{bmatrix} 1 & \psi & 0 \\ -\psi & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -\theta \\ 0 & 1 & 0 \\ \theta & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \phi \\ 0 & -\phi & 1 \end{bmatrix}$$

After calculation they become

$$\vec{v}_1 \approx \begin{bmatrix} 1 & \psi & -\theta \\ \phi\theta - \psi & \phi\theta\psi + 1 & \phi \\ \theta + \phi\psi & \theta\psi - \phi & 1 \end{bmatrix} \vec{u}, \tag{3.15}$$

$$\vec{v}_2 \approx \begin{bmatrix} 1 - \phi\theta\psi & \psi + \phi\theta & -\theta \\ -\psi & 1 & \phi \\ \theta + \phi\psi & \theta\psi - \phi & 1 \end{bmatrix} \vec{u} \tag{3.16}$$

and

$$\vec{v}_3 \approx \begin{bmatrix} 1 & \psi + \phi\theta & \phi\psi - \theta \\ -\psi & 1 - \phi\theta\psi & \phi + \theta\phi \\ \theta & -\phi & 1 \end{bmatrix} \vec{u} \tag{3.17}$$

Because the angles $\phi$,$\theta$ and $\psi$ are considered small, their multiplications with each other is approximately zero. In other words,

$$\vec{v}_1 \approx \vec{v}_2 \approx \vec{v}_3 \tag{3.18}$$

The result will be the same with the other three remaining rotations.
The roll, pitch and yaw will now be sums of these small angles. The small angles are calculated from the variation of the Euler angles at instant $t$ to $t-1$. Then, the rotation can be approximated as

$$\vec{v}(t) \approx \mathbf{R}_x(\phi(t) - \phi(t-1))\mathbf{R}_y(\theta(t) - \theta(t-1))\mathbf{R}_z(\psi(t) - \psi(t-1))\vec{u}(t), \quad (\vec{u}, \vec{v}) \in \mathcal{B}^2 \tag{3.19}$$

The variation of the Euler angles can be obtained from the angular speed measurement as

$$\begin{aligned} \phi(t) - \phi(t-1) &= \omega_x(t)\Delta t \\ \theta(t) - \theta(t-1) &= \omega_y(t)\Delta t \\ \psi(t) - \psi(t-1) &= \omega_z(t)\Delta t \end{aligned} \tag{3.20}$$

where $\omega_i$ is the angular velocity around the axis $i$ and $\Delta t$, the time between two samples.
Using the equation 3.20, the equation 3.19 becomes

$$\vec{v}(t) \approx \mathbf{R}_x(\omega_x(t)\Delta t)\mathbf{R}_y(\omega_y(t)\Delta t)\mathbf{R}_z(\omega_z(t)\Delta t)\vec{u}(t) = \mathbf{R}(t)\vec{u}(t), \quad (\vec{u}, \vec{v}) \in \mathcal{B}^2 \tag{3.21}$$

This approximation also implies that the rotation inverse to retrieve the vector $\vec{u}(t)$ from $\vec{v}(t)$ knowing the matrices $\mathbf{R}_x$,$\mathbf{R}_y$ and $\mathbf{R}_z$ can be done in any order. Without that approximation it would very difficult to obtain the attitude of the sensor. This approximation will obviously increase the calculation errors.

## 3.3 Attitude and Gimbal Lock

The attitude is characterized by the Euler angles, roll $\phi$, pitch $\theta$ and yaw $\psi$ which describe respectively the rotations angles around the axis $x, y$ and $z$. The rotation itself is done by using a rotation matrix which uses the Euler Angles. It is explained in section 3.2.
Since matrix multiplication order is critical, there are two convention groups which define the rotation matrix order (or sequence) of multiplication. The groups with their sequences are

- Proper Euler angles: $xyx$, $xzx$, $yxy$, $yzy$, $zxz$ and $zyz$
- Tait-Bryan angles: $xyz$, $xzy$, $yxz$, $yzx$, $zxy$ and $zyx$

The Tait-Bryan angles are also called classic Euler angles.
In this work, only the Tait-Bryan angles sequence $xyz$, also called $1-2-3$ sequence, will be used. All the sequences above are valid and equivalent in their results. The Tait-Brian angles sequences are more intuitive to use but have the same result as the Proper Euler angles. The Euler angles are obtained straight from the gyrometer's value of the angular speed integrated over time. The Euler angles will be associated with the vector $\vec{\Omega}$ with

$$\vec{\Omega}(t) = \begin{bmatrix} \phi(t) \\ \theta(t) \\ \psi(t) \end{bmatrix} = \int \vec{\omega}(t) dt \tag{3.22}$$

where $\vec{\omega}(t)$ is the angular speed given by the gyrometer.
With the three Euler angles, the attitude of a rigid object can be described in three dimensional space.
The rotation matrix representation normally presents a problem that is the gimbal lock. The gimbal lock appears at certain singularity angles which makes the system lose one degree of freedom.
It can be verified the following way: assuming the following rotation

$$\mathbf{R}(\phi, \theta, \psi) = \mathbf{R}_x(\phi)\mathbf{R}_x(\theta)\mathbf{R}_x(\psi) \tag{3.23}$$

and using equations 3.12, 3.13 and 3.14 of Section 3.2, we obtain

$$\mathbf{R}(\phi, \theta, \psi) = \begin{bmatrix} \cos(\theta)\cos(\psi) & \cos(\theta)\sin(\psi) & -\sin(\theta) \\ -\cos(\phi)\sin(\psi) & \cos(\phi)\cos(\psi) & \\ +\sin(\phi)\sin(\theta)\cos(\psi) & +\sin(\phi)\sin(\theta)\sin(\psi) & \sin(\phi)\cos(\theta) \\ \sin(\phi)\sin(\psi) & -\sin(\phi)\cos(\psi) & \\ +\cos(\phi)\sin(\theta)\cos(\psi) & +\cos(\phi)\sin(\theta)\sin(\psi) & \cos(\phi)\cos(\theta) \end{bmatrix} \tag{3.24}$$

If the pitch $\theta = \frac{\pi}{2}$, the matrix becomes after simplifications

$$\mathbf{R}(\phi, \theta = \frac{\pi}{2}, \psi) = \begin{bmatrix} 0 & 0 & 1 \\ \sin(\phi+\psi) & \cos(\phi+\psi) & 0 \\ -\cos(\phi+\psi) & \sin(\phi+\psi) & 0 \end{bmatrix} \tag{3.25}$$

It can noticed that whatever the value of the roll and yaw, the rotation will remain around the axis $z$. In other words, one degree of freedom is lost. The same phenomena happens if $\theta = -\frac{\pi}{2}$. But as explained in Section 3.2, the rotation matrices are approximated from small angles which keeps the gimbal lock from happening.

## 3.4 Change of basis

The transition matrix is an important tool to represent the change of basis. It is also called a transformation matrix. The change of basis is required due to the construction of the sensor. The accelerometer, gyrometer and magnetometer do not have the same basis as they are physically in different locations. So to use a sensor fusion approach, all the values should be given in the same basis. Ideally, their data should be given in the exact same orthonormal basis.
Given two bases $\mathcal{A}$ and $\mathcal{B}$, a vector $\vec{u}$ in the basis $\mathcal{A}$ can be written in the basis $\mathcal{B}$ as

$$\vec{u}|_{\mathbf{B}} = \mathbf{T}_{\mathbf{A}}^{\mathcal{B}} \vec{u}|_{\mathbf{A}} \tag{3.26}$$

where $\mathbf{T}_{\mathbf{A}}^{\mathcal{B}}$ is the transition matrix from $\mathcal{A}$ to $\mathcal{B}$. The inverse change of basis is simply

$$\vec{u}|_{\mathbf{A}} = {\mathbf{T}_{\mathbf{A}}^{\mathcal{B}}}^{-1} \vec{u}|_{\mathbf{B}} \tag{3.27}$$

where the superscript $^{-1}$ represent the inverse operation of the matrix. It can be noted that

$${\mathbf{T}_{\mathbf{A}}^{\mathcal{B}}}^{-1} \mathbf{T}_{\mathbf{A}}^{\mathcal{B}} = \mathbf{I} \tag{3.28}$$

where $\mathbf{I}$ represents the identity matrix in the respective basis.

## 3.5 Numerical integration

The integration is a required part of this work. As a fact, to obtain the position, the acceleration needs to be integrated two times with respect to time.
In theory, the integration of a continuous function $f$ with respect to time between the instant $a$ and $b$ is given by

$$I = \int_a^b f(t)dt = F(b) - F(a) \tag{3.29}$$

where $F$ is the antiderivative of the function $f$.
When the data is sampled, the theoretical integration is no longer possible. It is approximated by the numerical integration methods. The numerical integration can be calculated with different methods. Their efficiency to approximate the integration depends mostly on the function representing the signal.
In the following part, only four methods (or quadrature rules) will be examined. The performance of divers integration algorithms, included the four which were chosen, are described in [13, 14]

### 3.5.1 Rectangle rule

The rectangle rule is the most basic rule of the three.
Assuming that the distance $[a, b]$ can be divided in $N$ points, $t_1, t_2, t_3, ...t_N$, with $t_1 = a$ and $t_N = b$. Using the rectangle rule, the equation 3.29 can be rewritten as

$$I = \int_a^b f(t)dt \approx \sum_{i=1}^{N-1} f(t_i)(t_{i+1} - t_i) \tag{3.30}$$

The rectangle rule can be slightly upgraded to become the midpoint rule.

### 3.5.2 The midpoint rule

The mid point rule approximates the integration of equation 3.29 as

$$I = \int_a^b f(t)dt \approx \sum_{i=1}^{N-1} f\left(\frac{t_{i+1} - t_i}{2}\right)(t_{i+1} - t_i) \tag{3.31}$$

This rule and the previous one are preferred when calculation resources are limited and when the great precision is not required.

### 3.5.3 Trapezoidal rule

The trapezoidal rule is generally efficient and required small computation resources. The approximation becomes then

$$I = \int_a^b f(t)dt \approx \sum_{i=1}^{N-1} \left(\frac{f(t_{i+1}) - f(t_i)}{2}\right)(t_{i+1} - t_i) \tag{3.32}$$

### 3.5.4 Simpson's rule

The rule consists of introducing middle point between the points already existing. The extra point is equidistant from the two others. Then, an interpolation of 1st or 2nd order is done between the three given points. The calculation of the interpolating polynomial can be done by different algorithms (Lagrange, Spline or Newton). Using the Lagrange polynomial interpolation, it can be written that between three points,$t_i$, $\frac{t_{i+1}+t_i}{2}$, $t_{i+1}$, it exists a polynomial

$$\begin{aligned} p(t) = f(t_i)\frac{(t-m)(t-t_{i+1})}{(t_i-m)(t_i-t_{i+1})} + f(m)\frac{(t-t_i)(t-t_{i+1})}{(m-t_i)(m-t_{i+1})} + \\ f(t_{i+1})\frac{(t-m)(t-t_i)}{(t_{i+1}-m)(t_{i+1}-t_i)}, \forall t \in [t_i, t_{i+1}] \end{aligned} \tag{3.33}$$

where $m = \frac{t_i+t_{i+1}}{2}$.
Using this polynomial interpolation to approximate the equation 3.29,

$$I = \int_a^b f(t)dt \approx \sum_{i=1}^{N-1} \frac{t_{i+1} - t_i}{6}\left[f(t_i) + f(t_{i+1}]) + 4f\left(\frac{t_i + t_{i+1}}{2}\right)\right] \tag{3.34}$$

The complexity of the Simpson's rule depends on the interpolation algorithms.
The Simpson's rule can be ported further by adding more points in between the already existing points. The points are added in a adaptive way which will depends on the function to be integrated and the error criterion that the user desires. If the error is larger that the criterion, the section $[t_i, m, t_{i+1}]$ will be divided into $[t_i, r, m]$ and $[m, s, t_{i+1}]$ with $r$ and $s$ the new middle points. The Simpson rule is efficient for this work because the acceleration, speed and angular speed are continuous without any "jump" which can be described with a polynomial fitting.

# 4 Mathematical models of the sensors and the noise

The following models are simple and robust models describing the sensor data. These models are common models used in most scientific papers which examine IMU solutions. The noise model, present in every model of this work, is examined first. Afterward, a full description of the models of sensors is given.

## 4.1 Noise model

In all sensors, the data is corrupted by noise. The origin of the noise is not always obvious to determine. The most common noise, present in sensors using digitalization, is the one coming from the analog-to-digital converter (ADC). This type of noise, also called quantization noise, appears when converting analog values, which are theoretically infinite, into a finite number of values. It is a rounding error. The quantization noise can be, in most cases, approximated with additive white Gaussian noise (or white noise).
The probability density function (PDF) of the white Gaussian noise is the normal density. The Gaussian PDF is given by

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), x \in \mathbb{R} \tag{4.1}$$

where $\sigma$ is the standard deviation and $\mu$ the mean value. And the white Gaussian noise PDF is the Gaussian density with an average value of zero and of unit variance, i.e. $\mu = 0$ and $\sigma = 1$. With those conditions, the equation of the Gaussian PDF becomes

$$f(x) = \frac{1}{\sqrt{2\pi}} exp\left(-\frac{x^2}{2}\right), x \in \mathbb{R} \tag{4.2}$$

The noise is assumed to be additive which means that, any signal $y(t)$ which is corrupted by white noise can be written as

$$y(t) = z(t) + n(t) \tag{4.3}$$

where $z(t)$ is the original signal and $n(t)$ the Gaussian white noise. The expected value of the signal from equation 4.3 is

$$E[y(t)] = E[z(t)] + E[n(t)] = E[z(t)] \tag{4.4}$$

because the noise $n(t)$ is considered white Gaussian.

**NOTE**: The noise for the accelerometer will be referred as $\vec{n_a}(t)$, for the gyrometer $\vec{n}_g(t)$ and for the magnetometer $\vec{n}_c(t)$ ($c$ like compass).

## 4.2 Accelerometer model

The accelerometer measures the difference between the free fall state and the current state. In free fall, the acceleration measured is

$$\vec{a}_m(t) = \vec{n}_a(t) \tag{4.5}$$

where $\vec{n}_a(t)$ is the noise present in the reading. The acceleration is represented by a vector to point out that the values from the sensor are given in a three dimensional Cartesian coordinate system. If the sensor is static, the measured acceleration vector become

$$\vec{a_m}(t) = -\vec{g}(t) + \vec{n}_a(t) \tag{4.6}$$

Where $\vec{g}(t)$ is the instantaneous value of gravitation vector. When the accelerometer rotates, the gravitation vector changes direction.
When the sensor is moving, the equation 4.6 becomes

$$\vec{a_m}(t) = \vec{a}(t) - \vec{g}(t) + \vec{n}_a(t) \tag{4.7}$$

where $\vec{a}(t)$ is the acceleration sensed by the sensor and also the value needed to compute the position.
This model would be enough in a perfectly calibrated sensor, but like it will be shown in chapter 6, it is far from being the case. The accelerometer has a bias which slowly changes over time due to temperature fluctuation. The bias will be considered as constant, as it can be assumed that the measurements are done within a short period of time in which the temperature does not evolve. The equation 4.7 becomes

$$\vec{a_m}(t) = \vec{a}(t) - \vec{g}(t) + \vec{\text{bias}}_a + \vec{n}_a(t) \tag{4.8}$$

The next step is to model the errors due to the scaling of the axes. The scaling or sensitivity is a factor that amplifies or reduces the measured value to obtain the real value. The equation 4.7 becomes

$$\vec{a_m}(t) = \mathbf{S}_a(\vec{a}(t) - \vec{g}(t)) + \vec{\text{bias}}_a + \vec{n}_a(t) \tag{4.9}$$

where the sensitivity of the axes are represented by $\mathbf{S}_a$, a diagonal square matrix 3×3.

$$\mathbf{S}_a = \begin{bmatrix} S_{a,x} & 0 & 0 \\ 0 & S_{a,y} & 0 \\ 0 & 0 & S_{a,z} \end{bmatrix} \tag{4.10}$$

In the case of an ideal accelerometer, the axes would form an orthonormal basis. In practice, each axis has its own module which are mounted next to each other to keep the size of the MEMS-sensor to a bare minimum. For this reason, the measurement axes cannot be viewed as orthogonal. This is known as the misalignment error. Adding this error to the equation 4.9, it becomes

$$\vec{a_m}(t) = \mathbf{S}_a\mathbf{E}_a(\vec{a}(t) - \vec{g}(t)) + \vec{\text{bias}}_a + \vec{n}_a(t) \tag{4.11}$$

where $\mathbf{E}_a$ is a matrix representing the misalignment of the axes and can be written as

$$\mathbf{E}_a = \begin{bmatrix} 1 & E_{a,xy} & E_{a,xz} \\ E_{a,yx} & 1 & E_{a,yz} \\ E_{a,zx} & E_{a,zy} & 1 \end{bmatrix} \tag{4.12}$$

Changing $\mathbf{S}_a\mathbf{E}_a$ to $\mathbf{T}_a$, the equation 4.11 becomes

$$\vec{a_m}(t) = \mathbf{T}_a(\vec{a}(t) - \vec{g}(t)) + \vec{\text{bias}}_a + \vec{n}_a(t) \tag{4.13}$$

with the error matrix to

$$\mathbf{T}_a = \begin{bmatrix} T_{a,x} & T_{a,xy} & T_{a,xz} \\ T_{a,yx} & T_{a,y} & T_{a,yz} \\ T_{a,zx} & T_{a,zy} & T_{a,z} \end{bmatrix} \tag{4.14}$$

The diagonal terms are the scaling of the axes while the other terms are called cross-coupling error.
When the sensor is not subject to movement, i.e. $\vec{a}(t) = 0$, the equation becomes

$$\vec{a_m}(t) = -\mathbf{T}_a\vec{g}(t) + \vec{\text{bias}}_a + \vec{n}_a(t) \tag{4.15}$$

This equation is needed for the calibration.

## 4.3 Gyrometer model

The gyrometer is used to obtain the attitude of the sensor and also the direction of the Earth gravity vector knowing its original position. The gyrometer can be modeled the same way as the accelerometer.
When the sensor is ideal and stationary, the measured values are

$$\vec{\omega_m}(t) = \vec{0} \tag{4.16}$$

and for a non-static sensor, the measured angular speed at an instant t is

$$\vec{\omega_m}(t) = \vec{\omega}(t) \tag{4.17}$$

where $\omega(t)$ is the value describing the angular velocity of the sensor. The model would be perfect for an ideal gyrometer. As ideal sensor does not exist, noise ,$\vec{n}_g$, and bias, $\vec{\text{bias}}_g$, have to be added to the model. The bias would be including the Earth's angular velocity but is likely to be too low to be sensed by a low-grade MEMS gyroscope.
The sensor being static gives the equation

$$\vec{\omega_m}(t) = \vec{\text{bias}}_g + \vec{n}_g(t) \tag{4.18}$$

and when the sensor is in motion,

$$\vec{\omega_m}(t) = \vec{\omega}(t) + \vec{\text{bias}}_g + \vec{n}_g(t) \tag{4.19}$$

The gyrometer suffers from the same problems as the accelerometer. The sensitivity of each axes is different which leads to non-normality of the basis of the measured values and the misalignment errors are also present. The mathematical model from equation 4.19 becomes

$$\vec{\omega}_m(t) = \mathbf{T}_g\vec{\omega}(t) + \vec{\text{bias}}_g + \vec{n}_g(t) \tag{4.20}$$

where

$$\mathbf{T}_g = \begin{bmatrix} T_{g,x} & T_{g,xy} & T_{g,xz} \\ T_{g,yx} & T_{g,y} & T_{g,yz} \\ T_{g,zx} & T_{g,zy} & T_{g,z} \end{bmatrix} \tag{4.21}$$

The construction of the gyrometer is different from the accelerometer as it is a single piece for the three axes. It means that the misalignment errors should be very close to zero.

## 4.4 Magnetometer

The magnetometer measures the local Earth's magnetic field which means that every magnetic objects will interact with the good functioning of the device.
When the magnetometer is ideally calibrated, the three dimensional vector $\vec{m}_m(t)$ will follow a sphere with a radius equal to the magnitude of the local magnetic field. In this case, the measured values are

$$\vec{m}_m(t) = \vec{m}(t) + \vec{n}_c(t) \tag{4.22}$$

where $m_m(t)$ is the measured value from the magnetometer, $n_c(t)$ the noise and $m(t)$ the real value. Like the other sensors, the magnetometer has a bias, $\vec{\text{bias}}_c$. The bias is caused by the permanent magnetization of the magnetic elements around the sensor. The bias of the magnetometer is also called hard iron effect [15]. Thus, the equation 4.22 can be rewritten to

$$\vec{m}_m(t) = \vec{m}(t) + \vec{\text{bias}}_c + \vec{n}_c(t) \tag{4.23}$$

The magnetometer is mounted on a printed circuit board which has components that are magnetized by an external field. The sphere, that would normally describe the magnitude of the measured vector, will become an ellipsoid. The equation 4.23 becomes

$$\vec{m}_m(t) = \mathbf{S}_c\vec{m}(t) + \vec{\text{bias}}_c + \vec{n}_c(t) \tag{4.24}$$

The matrix $\mathbf{S}_c$ characterizes the soft iron effect [15]. The sensor is composed of three separate Hall effect sensors, which induces misalignment errors. The equation 4.24 becomes

$$\vec{m}_m(t) = \mathbf{E}_c\mathbf{S}_c\vec{m}(t) + \vec{\text{bias}}_c + \vec{n}_c(t) \tag{4.25}$$

Because it is not important to know separately the matrices $\mathbf{E}_c$ and $\mathbf{S}_c$, their product can be replaced by the matrix $\mathbf{T}_c$. The equation becomes

$$\vec{m}_m(t) = \mathbf{T}_c\vec{m}(t) + \vec{\text{bias}}_c + \vec{n}_c(t) \tag{4.26}$$

and more precisely

$$\vec{m}_m(t) = \begin{bmatrix} m_{m,x} \\ m_{m,y} \\ m_{m,z} \end{bmatrix} = \begin{bmatrix} T_{c,x} & T_{c,xy} & T_{c,xz} \\ T_{c,yx} & T_{c,y} & T_{c,yz} \\ T_{c,zx} & T_{c,zy} & T_{c,z} \end{bmatrix} \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} + \begin{bmatrix} \mathrm{b}_{c,x} \\ \mathrm{b}_{c,y} \\ \mathrm{b}_{c,z} \end{bmatrix} + \begin{bmatrix} \mathrm{n}_{c,x} \\ \mathrm{n}_{c,y} \\ \mathrm{n}_{c,z} \end{bmatrix} \tag{4.27}$$

where the diagonal elements represent the sensitivity of the axes which leads to non-normality of the basis.

# 5 Algorithms

In this section will be described the algorithms present in this work. The complete algorithm is described in the figure 5.1. It is important to notice that the algorithms

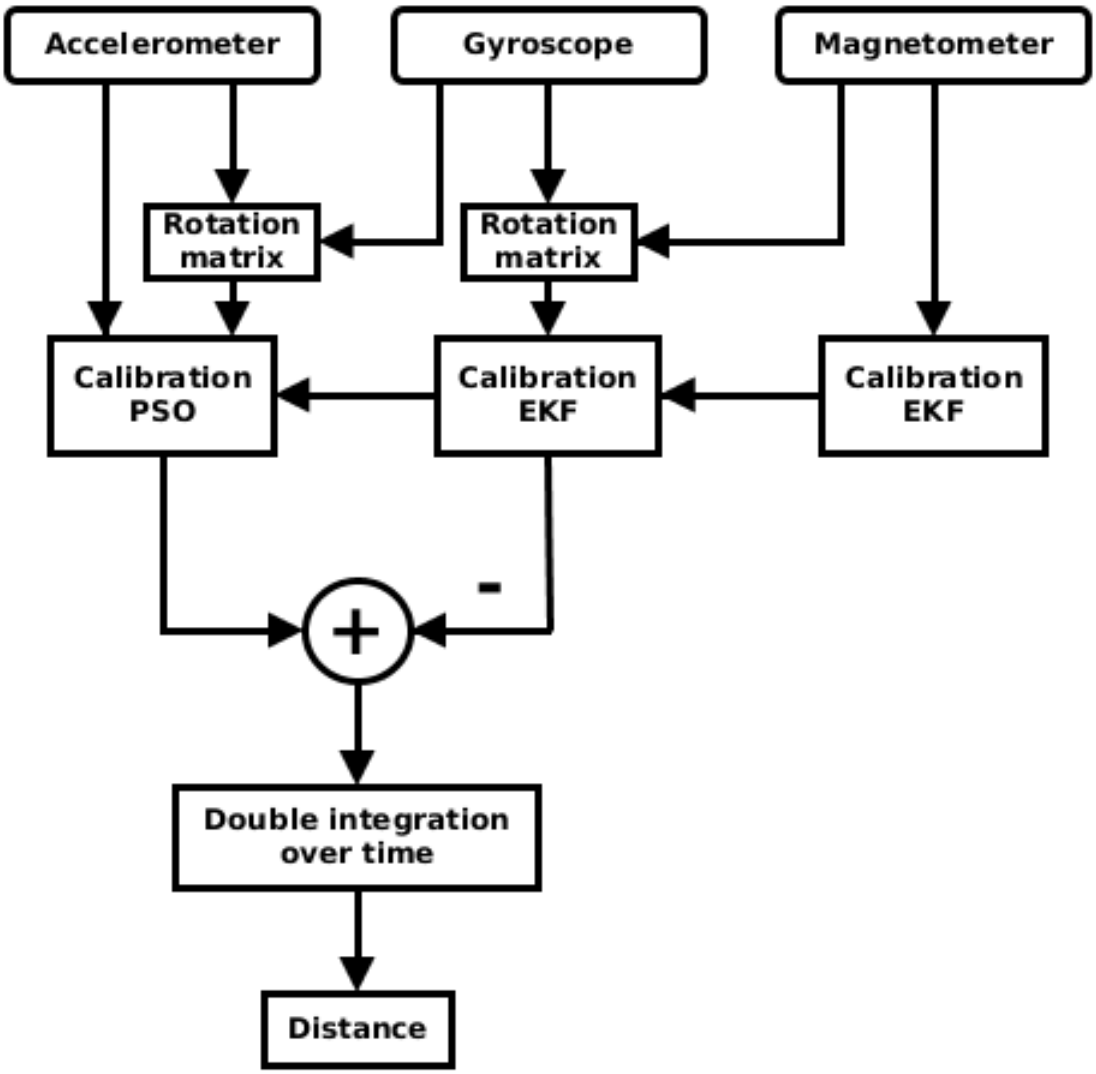


Figure 5.1: Block diagram representing the order of the algorithms

are not chosen to be computationally efficient and that the Gram-Schmidt process is included in the rotation matrix.

## 5.1 Calibration

The calibration is an important step to obtain the right values from the sensors. The manufacturer is usually providing factory values which represent the corrected scaling values. For low-cost products, these calibration values are rough because it would be too expensive to calibrate well a sensor that only cost few dollars.

The calibration has to be done every time the sensor is switched on because the biases of the axes change. It is especially true for the accelerometer, while the other sensor are not significantly affected. The sensors are also sensitive to temperature changes [16]. In article [17], the authors propose a temperature compensation model using neural network. However, estimating a temperature model requires very expensive equipment to obtain stabilized temperature. Thus, in this work, it is assumed that the temperature remains constant during the measurements. That is, the temperature compensation will fall into the calibration model.

### 5.1.1 Accelerometer

The calibration of the accelerometer is usually done by setting the sensor in exactly known positions and correcting the values of $\vec{g}$ measured by the sensor accordingly. The accelerometer is said to be calibrated when the norm of the measured acceleration vector $\vec{a}_m$ is always equal to one $g$ in static state and the sphere it described is centered at the origin. This refers to the equation 4.6 of Section 4.2. To obtain this equation from the model of equation 4.15, the transformation matrix $\mathbf{T}_a$ needs to be determined. The gravity vector can be written from equation 4.15 as

$$\mathbf{T}\vec{g}(t) = \vec{\text{bias}}_a(t) - \vec{a}_m(t) + \vec{n}_a(t) \tag{5.1}$$

and using the expectation value

$$\text{E}[\mathbf{T}\vec{g}(t)] = \text{E}[\vec{\text{bias}}_a(t)] - \text{E}[\vec{a}_m(t)] + \text{E}[\vec{n}_a(t)] \tag{5.2}$$

Because $\mathbf{T}$ is composed of constants, the equation becomes

$$\mathbf{T}\text{E}[\vec{g}(t)] = \text{E}[\vec{\text{bias}}_a] - \text{E}[\vec{a}_m] \tag{5.3}$$

Knowing the magnitude of the gravity vector $\vec{g}(t)$, it can be written

$$\|\text{E}[\vec{g}(t)]\| = g = \|\mathbf{T}^{-1}(\text{E}[\vec{\text{bias}}_a] - \text{E}[\vec{a}_m])\| \tag{5.4}$$

where $g$ is the magnitude of the local Earth gravity vector and equal approximatively to $9,81\text{m.s}^{-2}$.
Expanding the equation 5.4, it becomes

$$\left\| \begin{bmatrix} T_{a,x} & T_{a,xy} & T_{a,xz} \\ T_{a,yx} & T_{a,y} & T_{a,yz} \\ T_{a,zx} & T_{a,zy} & T_{a,z} \end{bmatrix}^{-1} \left( \begin{bmatrix} b_{a,x} \\ b_{a,y} \\ b_{a,z} \end{bmatrix} - \begin{bmatrix} \text{E}[a_{m,x}] \\ \text{E}[a_{m,y}] \\ \text{E}[a_{m,z}] \end{bmatrix} \right) \right\| = g \tag{5.5}$$

The equation 5.5 becomes

$$\|\mathbf{T}(\text{E}[\vec{\text{bias}}_a] - \text{E}[\vec{a}_m])\| = g \tag{5.6}$$

The equation 5.6 has twelve unknown variables

$$\Theta = [T_{a,x}, T_{a,y}, T_{a,z}, T_{a,xy}, T_{a,xz}, T_{a,yx}, T_{a,yz}, T_{a,zx}, T_{a,zy}, b_{a,x}, b_{a,y}, b_{a,z}] \tag{5.7}$$

That means that at least four values of $\mathrm{E}[\vec{a}_m]$ (the vector contains 3 values $a_{m,x}, a_{m,y}$ and $a_{m,z}$) are needed. The values are taken when the sensor is not in motion and placing it in four different position $i$. This equation will be solved using swarm particle optimization [18] where the cost function is

$$\min_{\Theta} \left( \sum_{i=1}^{N} \|\mathbf{T}(\mathrm{E}[\vec{\mathrm{bias}}_a] - \mathrm{E}[\vec{a}_m[i]])\|^2 - g^2 \right), \quad i \in 1, 2, .., N \tag{5.8}$$

where $\mathrm{E}[\vec{a}_m[i]]$ is the average value of the measured acceleration over a few thousand samples taken at a position $i$.

### 5.1.2 Magnetometer

The calibration of a magnetometer is usually done with Helmholtz coil which creates a nearly uniform magnetic field. Setting the magnetometer in different positions within the coils field, the magnetometer values can be corrected accordingly.
The magnetometer is said to be calibrated when the measured vector, representing the magnitude and the direction of the local magnetic field, has a magnitude equal to the one of the local magnetic field and describing a sphere centered in the origin.
The calibration of the magnetometer will be done with the Extended Kalman filter (EKF) and mostly according to the article [19]. The difference resides in the calculation which lead to different results.
To use the EKF, the state variable $\vec{x}(t)$ and the observation $\vec{z}(t)$ must be defined.
The state variable will be defined as the inverse solution of the calibration by setting $\mathbf{T}_c^{-1} = \mathbf{K}$, i.e.

$$\vec{x}(t) = [K_{c,x}, K_{c,y}, K_{c,z}, K_{c,xy}, K_{c,xz}, K_{c,yx}, K_{c,yz}, K_{c,zx}, K_{c,zy}, b_{c,x}, b_{c,y}, b_{c,z}]^{\mathrm{T}} \tag{5.9}$$

The reason for this replacement is for displaying the result of the Jacobian which would take more that a page to display.
The time derivative of the state variable will be

$$\dot{\vec{x}}(t) = \frac{d\vec{x}(t)}{dt} = \vec{n}_p(t-1) \tag{5.10}$$

with $\vec{n}_p$ the process noise.
The observation will be defined as the strength of the local magnetic vector, i.e.

$$\vec{z}(t) = \|\vec{m}_m(t)\|^2 = \vec{m}_m(t)^{\mathrm{T}} \vec{m}_m(t) \tag{5.11}$$

Assuming that the bias does not change during the calibration, i.e. $\vec{\mathrm{bias}}_c(t) = \vec{\mathrm{bias}}_c =$ constant and using the equation 4.26 to extract the true magnetic field $\vec{m}$,

$$\vec{m}(t) = (\mathbf{T}_c)^{-1} (\vec{m}_m(t) - \vec{\mathrm{bias}}_c) + \vec{n}_c(t) = \mathbf{K}(\vec{m}_m(t) - \vec{\mathrm{bias}}_c) + \vec{n}_c(t)$$

As a reminder, $\mathbf{T}_c^{-1} = \mathbf{K}$ with

$$\mathbf{K} = \begin{bmatrix} K_{c,x} & K_{c,xy} & K_{c,xz} \\ K_{c,yx} & K_{c,y} & K_{c,yz} \\ K_{c,zx} & K_{c,zy} & K_{c,z} \end{bmatrix} \tag{5.12}$$

$\vec{m}(t)$ is unknown but the product $\vec{m}(t)^{\mathrm{T}}\vec{m}(t)$ corresponds to the squared Earth's local magnetic field. In Helsinki, it is equal to $52,0854\mu\mathrm{T}$ according to the website Magnetic-deviation [20]. This value is calculated based on the Earth's magnetic field model defined in [21].

$$\vec{m}(t)^{\mathrm{T}}\vec{m}(t) = \left(\mathbf{K}(\vec{m}_m(t) - \vec{\mathrm{bias}}_c) + \vec{n}_c(t)\right)^{\mathrm{T}} \left(\mathbf{K}(\vec{m}_m(t) - \vec{\mathrm{bias}}_c) + \vec{n}_c(t)\right) \tag{5.13}$$

Expending and reorganizing the equation, it becomes

$$\begin{aligned}\vec{m}(t)^{\mathrm{T}}\vec{m}(t) = \vec{m}_m(t)^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}\mathbf{K}\vec{m}_m(t)^{\mathrm{T}} - 2\mathrm{bias}_c^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}\mathbf{K}\vec{m}_m(t) + \mathrm{bias}_c^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}\mathbf{K}\mathrm{bias}_c + \\ 2\vec{n}_c(t)^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}\mathbf{K}\vec{m}_m(t) - 2\mathrm{bias}^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}\mathbf{K}\vec{n}_c(t) + \vec{n}_c(t)^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}\mathbf{K}\vec{n}_c(t)\end{aligned}$$

and rewritten as

$$\vec{m}(t)^{\mathrm{T}}\vec{m}(t) = \vec{m}_m(t)^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}\mathbf{K}\vec{m}_m(t)^{\mathrm{T}} - 2\mathrm{bias}_c^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}\mathbf{K}\vec{m}_m(t) + \mathrm{bias}_c^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}\mathbf{K}\mathrm{bias}_c + \vec{n}_o(t) \tag{5.14}$$

where $\vec{n}_o(t) = 2\vec{n}_c(t)^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}\mathbf{K}\vec{m}_m(t) - 2\mathrm{bias}^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}\mathbf{K}\vec{n}_c(t) + \vec{n}_c(t)^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}\mathbf{K}\vec{n}_c(t)$ is the observation noise and is assumed to be close enough to Gaussian noise.
After introduction of $\vec{m}_m(t)^{\mathrm{T}}\vec{m}_m(t)$ in the equation 5.14, the observation is

$$\begin{aligned}\vec{z}(t) = \vec{m}_m(t)^{\mathrm{T}}\vec{m}_m(t) = \vec{m}(t)^{\mathrm{T}}\vec{m}(t) - \vec{m}_m(t)^{\mathrm{T}}\left(\mathbf{K}^{\mathrm{T}}\mathbf{K} - \mathbf{I}\right)\vec{m}_m(t)^{\mathrm{T}} + \\ 2\mathrm{bias}_c^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}\mathbf{K}\vec{m}_m(t) - \mathrm{bias}_c^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}\mathbf{K}\mathrm{bias}_c + \vec{n}_o(t)\end{aligned} \tag{5.15}$$

The equation can be rewritten as

$$\vec{z}(t) = f(\vec{x}(t)) + \vec{n}_o(t) \tag{5.16}$$

with the function

$$\begin{aligned}f(\vec{x}(t)) = \vec{m}(t)^{\mathrm{T}}\vec{m}(t) - \vec{m}_m(t)^{\mathrm{T}}\left(\mathbf{K}^{\mathrm{T}}\mathbf{K} - \mathbf{I}\right)\vec{m}_m(t)^{\mathrm{T}} + \\ 2\mathrm{bias}_c^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}\mathbf{K}\vec{m}_m(t) - \mathrm{bias}_c^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}\mathbf{K}\mathrm{bias}_c\end{aligned} \tag{5.17}$$

Because the observation equation $\vec{z}(t)$ is not linear, i.e. the function $f$ is not linear, the extended version of the Kalman filter (EKF) must be used. For that, the observation equation will be linearized using the Jacobian matrix $\mathbf{J}_m$.

$$\vec{z}(t) \approx \mathbf{J}_c(t)\vec{x}(t) + \vec{n}_o(t) \tag{5.18}$$

where

$$\mathbf{J}_c(t) = \left[\frac{\partial f(t)}{\partial K_{c,x}} \frac{\partial f(t)}{\partial K_{c,y}} \frac{\partial f(t)}{\partial K_{c,z}} \frac{\partial f(t)}{\partial K_{c,xy}} \frac{\partial f(t)}{\partial K_{c,xz}} \frac{\partial f(t)}{\partial K_{c,yx}} \frac{\partial f(t)}{\partial K_{c,yz}} \frac{\partial f(t)}{\partial K_{c,zx}} \frac{\partial f(t)}{\partial K_{c,zy}} \frac{\partial f(t)}{\partial b_{c,x}} \frac{\partial f(t)}{\partial b_{c,y}} \frac{\partial f(t)}{\partial b_{c,z}}\right] \tag{5.19}$$

Despite the change $\mathbf{T}_c^{-1} = \mathbf{K}$, The result of the Jacobian remains too large to be displayed here. The result can be found in Appendix A. Finally, the equations for the discrete-time Extended Kalman filter are:

$$\begin{cases}\dot{\vec{x}}(t) = \vec{n}_p(t-1) \\ \vec{z}(t) \approx \mathbf{J}_c(t)x(t) + \vec{n}_o(t)\end{cases} \Rightarrow \begin{cases}\vec{x}[k] = A\vec{x}[k-1] + \vec{n}_p[k] \\ \vec{z}[k] = J[k]\vec{x}[k] + n_o[k]\end{cases} \tag{5.20}$$

Where $\mathbf{A}$ is a $1 \times 12$ unity matrix.

### 5.1.3 Gyrometer

Ideally, the calibration of the gyrometer could be done using the Earth's rotation velocity. Unfortunately, the sensor in this work is not sensitive enough to sense it. The Earth's rotational speed is about $7,27 \times 10^{-5}$rad.s$^{-1}$ [22] while the sensor's smallest theoretical (without noise) angular velocity measurable is $13,3 \times 10^{-5}$ rad.s$^{-1}$.
The calibration is usually done with a turntable from which the angular velocity is known. Then, the values of the sensor are corrected accordingly. In this work, the calibration is done without additional equipment.
The gyrometer is here the most difficult sensor to calibrate among the three sensors. The problem resides in the non-availability of a direct measured value which can be used for the calibration. Without any external equipment, it is not a simple task to measure the angular velocity. The angular velocity could be obtained by using a calibrated magnetometer. Using simple geometrical tangent relationship, the Euler angles can be extracted. The derivative of these angles would give values that could be compared to the value measured by the gyrometer. In theory, this process would work. In practice, the data of the magnetometer is very noisy which would induce instability of the function tangent "tan" when the angle is close to $\pm\pi/2$ (or $\pm\pi$ if the function "tan2" is used). This problem can be somehow solved with an approximation of the angle which would lead to more stable values. Another problem comes along with the numerical derivation of noisy data which requires "denoising" algorithm such as smoothing filters or polynomial fitting. These algorithms remove data from the signal if the signal is unknown like in this work.
One other way to do this is to use calculated values from the gyrometer to obtain the rotation matrix. In other words, the equation 3.21 is utilized. Assuming that the magnetometer is calibrated and the magnetic filed is not disturbed by electric devices or magnetized material, the vector describing the magnetic field present can be written as

$$\vec{m}_m(t) = \mathbf{S}_g\mathbf{R}(t)\vec{m}_m(t-1) + \vec{n}_g(t) \tag{5.21}$$

where $\mathbf{S}_g$ is the correction matrix which includes scaling, basis change and orthogonal rectification

$$\mathbf{S}_g = \begin{bmatrix} S_{g,x} & S_{g,xy} & S_{g,xz} \\ S_{g,yx} & S_{g,y} & S_{g,yz} \\ S_{g,zx} & S_{g,zy} & S_{g,z} \end{bmatrix}, \tag{5.22}$$

and $\mathbf{R}(t)$ the rotation matrix at the instant $t$ and $\vec{n}_g$ the noise. The equation 5.21 represents the rotation of an initial vector at an instant $t$.
Since the calibration of magnetometer sensor is not perfect it is wise to add a bias to the equation. It becomes

$$\vec{m}_m(t) = \mathbf{S}_g\mathbf{R}(t)(\vec{m}_m(t-1) + \vec{\text{bias}}_g) + \vec{n}_g(t) \tag{5.23}$$

The extended Kalman filter is then utilized to obtain a solution to $\mathbf{S}_g$ and the bias. The state variable is defined as

$$\vec{x}(t) = [S_{g,x}, S_{g,y}, S_{g,z}, S_{g,xy}, S_{g,xz}, S_{g,yx}, S_{g,yz}, S_{g,zx}, S_{g,zy}, b_{g,x}, b_{g,y}, b_{g,z}]^{\mathrm{T}} \tag{5.24}$$

It represents the solution of the calibration.
The equation 5.23 can be written as

$$\vec{m}_m(t) = f(\vec{x}(t)) + \vec{n}_g(t) \tag{5.25}$$

with the function $f$ being $\mathbf{S}_g\mathbf{R}(t)(\vec{m}_m(t-1) + \vec{\text{bias}}_g)$.
The state equations are

$$\begin{cases} \dot{\vec{x}}(t) = \vec{n}_g(t) \\ \vec{z}(t) \approx \mathbf{J}_g(t)x(t) + \vec{n}_o(t) \end{cases} \Rightarrow \begin{cases} \vec{x}[k] = \mathbf{A}\vec{x}[k-1] + \vec{n}_p[k] \\ \vec{z}[k] = \mathbf{J}_g[k]\vec{x}[k] + n_o[k] \end{cases} \tag{5.26}$$

where $\mathbf{J}_g$ is the jacobian of $\vec{m}_m(t)$

$$\mathbf{J}_g(t) = \left[\frac{\partial f(t)}{\partial S_{g,x}} \frac{\partial f(t)}{\partial S_{g,y}} \frac{\partial f(t)}{\partial S_{g,z}} \frac{\partial f(t)}{\partial S_{g,xy}} \frac{\partial f(t)}{\partial S_{g,xz}} \frac{\partial f(t)}{\partial S_{g,yx}} \frac{\partial f(t)}{\partial S_{g,yz}} \frac{\partial f(t)}{\partial S_{g,zx}} \frac{\partial f(t)}{\partial S_{g,zy}} \frac{\partial f(t)}{\partial b_{g,x}} \frac{\partial f(t)}{\partial b_{g,y}} \frac{\partial f(t)}{\partial b_{c,z}}\right]^{\mathrm{T}} \tag{5.27}$$

The solution of the Jacobian is given in Appendix B.

## 5.2 Gravity vector computation

The Earth's gravitational vector $\vec{g}$ can be obtained by an accelerometer and the fusion of a gyrometer with an accelerometer.
When the sensor is at rest and calibrated, using the equation 4.6 of section 4.2,

$$\vec{a_m}(t) = -\vec{g}(t) + \vec{n}_a(t)$$

The gravity vector can be obtained by just reading the average value of the accelerometer at rest. The average value of white Gaussian noise is zero and when the accelerometer is still and perfectly calibrated, it can be written that

$$\mathrm{E}[\vec{a_m}(t)] = -\mathrm{E}[\vec{g}(t)] \tag{5.28}$$

The gyrometer is used to compute the direction of the gravity vector $\vec{g}(t)$. The computation of the direction of $\vec{g}(t)$ from the gyrometer requires an initial value of the vector $\vec{g}(t)$ obtained from the accelerometer before the sensor is in motion. It can be then written that the gravity vector at an instant $t$ is

$$\vec{g}_g(t) = \mathbf{R}(t)\vec{g}(t-1)|_g + \vec{n}_g(t) \tag{5.29}$$

This value has to be transfered to the basis of the gyrometer to be able to use the equation 5.29. And the equation becomes

$$\vec{g}_g(t) = \mathbf{R}(t)\mathbf{T}_a^g\vec{g}(t-1)|_a + \vec{n}_g(t) \tag{5.30}$$

where $\mathbf{T}_a^g$ is the transformation matrix representing the change of basis from accelerometer to gyrometer.

## 5.3 The Kalman filter

The Kalman filter is usually associated to the IMU and data fusion in many works [24, 25, 26]. The Kalman filter is a generalization of the Wiener filter which is restricted to stationary scalar signal and noises [23]. The filter can be seen as a sequential minimum mean square error (MMSE) estimator of a signal in noise.

### 5.3.1 The Kalman Filter for linear system

For the Kalman filter, the state and observation models must be known. They must also be Gauss-Markov models. Assuming that the signal are digital, ie. $k = nT$ with $n \in \mathbb{N}$ and $T$ the sampling period. The state model is written as:

$$\vec{x}[k] = \mathbf{A}\vec{x}[k-1] + \mathbf{B}\vec{u}[k] + \vec{n}_p[k], \quad \forall k > 0 \tag{5.31}$$

where the matrices $\mathbf{A}$ and $\mathbf{B}$ are known matrices and respectively the state-transition model and the control input model. The process noise is written as $\vec{n}_p$ and defined as white Gaussian.
In the following part the equations of the Kalman filter will be written without much detail. The reference book [23], is a good source of information to understand where these equations come from.
The observation model is given by

$$\vec{z}[k] = \mathbf{H}\vec{x}[k|k] + \vec{n}_o[k], \quad \forall k > 0 \tag{5.32}$$

The prediction:

$$\vec{\hat{x}}[k|k-1] = \mathbf{A}\vec{\hat{x}}[k-1|k-1] + \mathbf{B}\vec{u}[k] \tag{5.33}$$

The minimum prediction MSE matrix or error covariance matrix:

$$\mathbf{M}[k|k-1] = \mathbf{A}\mathbf{M}[k-1|k-1]\mathbf{A}^{\mathrm{T}} + \mathbf{Q}[k-1] \tag{5.34}$$

Where $\mathbf{Q}$ is the system noise covariance matrix. It defines how the uncertainty increases with time due to noise sources.
The Kalman gain is

$$\mathbf{K}[k] = \frac{\mathbf{M}[k|k-1]\mathbf{H}[k]^{\mathrm{T}}}{\mathbf{H}[k]\mathbf{M}[k|k-1]\mathbf{H}[k]^{\mathrm{T}} + \mathbf{R}[k]} \tag{5.35}$$

It defines the weight between the observation and the state model.
The correction or update:

$$\vec{\hat{x}}[k|k] = \vec{\hat{x}}[k|k-1] + \mathbf{K}[k](\vec{z}[k] - \mathbf{H}^{\mathrm{T}}[k]\vec{\hat{x}}[k|k-1]) \tag{5.36}$$

Minimum MSE Matrix update:

$$\mathbf{M}[k|k] = (\mathbf{I} - \mathbf{K}[k]\vec{h}^{\mathrm{T}}[k])\mathbf{M}[k|k-1] \tag{5.37}$$

Where the mean square error matrices are defined as

$$\mathbf{M}[k|k] = \mathrm{E}[(\vec{x}[k] - \hat{\vec{x}}[k|k])(\vec{x}[k] - \hat{\vec{x}}[k|k])^{\mathrm{T}}] \tag{5.38}$$

and

$$\mathbf{M}[k|k-1] = \mathrm{E}[(\vec{x}[k] - \hat{\vec{x}}[k|k-1])(\vec{x}[k] - \hat{\vec{x}}[k|k-1])^{\mathrm{T}}] \tag{5.39}$$

The equation above are the equations required for the Kalman filter.

### 5.3.2 Extended Kalman Filter

In the previous case, the filter was designed for linear systems. When the system is not linear, like often in practice, the extended Kalman filter can be used. The equations remains the same but the transition $\mathbf{A}$ and/or the observation $\mathbf{H}$ must be linearized.
For a non-linear systems, the equations 5.31 and 5.32 can be written as

$$\vec{x}[k] = f(\vec{x}[k-1]) + \vec{n}_p[k-1] \quad \forall k > 0 \tag{5.40}$$

$$\vec{z}[k] = H\vec{x}(k) + \vec{n}_o[k], \quad \forall k > 0 \tag{5.41}$$

where $f$ and $h$ are functions that needs to be linearized to be able to use the Kalman filter. The linearization introduced errors due to the approximation of the functions around the concerned point. The filter is not optimal any longer. The approximation is usually done at the first order but can be done also at higher order. The 1st order linearization is realized using the Jacobian. The transition matrix becomes

$$\mathbf{A} = \frac{\partial f(\vec{x})}{\partial \vec{x}} \tag{5.42}$$

and the observation matrix becomes

$$\mathbf{H} = \frac{\partial h(\vec{x})}{\partial \vec{x}} \tag{5.43}$$

### 5.3.3 Algorithm of the Kalman filter

The Kalman filter required initial values for the state variable $\vec{x}$, the error covariance matrix $\mathbf{M}$, the covariance noise matrices $\mathbf{R}$ and $\mathbf{Q}$, respectively $\vec{x}[0]$, $\mathbf{M}[0]$, $\mathbf{R}[0]$ and $\mathbf{Q}[0]$ . The initial value for the state variable, $\vec{x}[0]$, is not important and can be set to zero as it will be updated correctly afterward. The covariance matrices are important to determine. Unfortunately, it is not an easy task and usually is realized empirically. When the state and measurement models are known, the algorithm goes as follows [27]:

1. Obtain the transition, matrix $\mathbf{A}[k-1]$
2. Obtain the system noise covariance matrix, $\mathbf{Q}[k-1]$
3. Propagate the state vector estimate from $\hat{\vec{x}}[k|k]$ to $\hat{\vec{x}}[k|k-1]$

4. Propagate the error covariance matrix from $\mathbf{M}[k|k]$ to $\mathbf{M}[k|k-1]$
5. Calculate the measurement matrix $\mathbf{H}[k]$
6. Calculate the measurement noise covariance matrix, $\mathbf{R}[k]$
7. Calculate the Kalman gain, $\mathbf{K}[k]$
8. Formulate the measurement vector, $\vec{z}[k]$
9. Update the state vector estimate from $\vec{\hat{x}}[k|k-1]$ to $\vec{\hat{x}}[k|k]$
10. Update the covariance matrix from $\mathbf{M}[k|k-1]$ to $\mathbf{M}[k|k]$

## 5.4 Obtaining the position

To obtain the position $\vec{p}(t)$, the acceleration $\vec{a}(t)$ has to be integrated with respect to time, first to become the velocity $\vec{v}(t)$. The velocity at the instant $t_1$ from an initial instant $t_0$ is given by

$$\vec{v}(t_1) = \int_{t_0}^{t_1} \vec{a}(t)dt \tag{5.44}$$

The position is the integration of the velocity $\vec{v}$ with respect to time. It is written as

$$\vec{p}(t_1) = \int_{t_0}^{t_1} \vec{v}(t)dt \tag{5.45}$$

These integrals are done numerically using one of the quadrature in Section 3.5. The accelerometer does measure the combination of the gravitational acceleration with the actual acceleration of the sensor. To obtain the actual acceleration $\vec{a}(t)$, the gravitational acceleration vector must be removed from the readings. The only way to do it without external equipment is to use the gyrometer. Using the equation 5.6, the actual acceleration is

$$\vec{a}(t)|_g = \mathbf{T}_a^g \vec{a}_m|_a - \mathbf{R}(t)\mathbf{T}_a^g \vec{g}(t-1)|_a + \vec{n}_g(t) \tag{5.46}$$

When the acceleration $\vec{a}(t)$ is obtained, the position can be easily computed.

## 5.5 The modified Gram-Schmidt process

The Gram-Schmidt process is a method used for correcting orthogonality of a set of vectors. This method is needed to correct the rotation matrix obtained from the gyrometer. The classical Gram-Schmidt process is numerically unstable[28]. This means that applied to digital values, the algorithm may not converge nor give the correct values.
The rounding error or the noise modifies the actual value randomly which changes the rotation matrix into a non-orthogonal matrix. The non-orthogonal matrix induces errors in the Euler angles calculation.
The classical gram-Schmidt process works the following way:

Assuming a $\mathbb{R}^3$ vectorial space $\mathcal{V}$ spanned with a non-orthogonal set of vectors $\vec{v}_1, \vec{v}_2$ and $\vec{v}_3$. The orthogonal projection of a vector $\vec{v}$ on an unidimensional subspace $\vec{u}$ is defined as

$$\text{proj}_u(\vec{v}) = \frac{\vec{u} \cdot \vec{v}}{\vec{u} \cdot \vec{u}} \tag{5.47}$$

The algorithm goes as follows,

1. Set the first vector from which the other will be corrected

$$\vec{u}_1 = \vec{v}_1$$

2. The second vector is obtained from the previous vector

$$\vec{u}_2 = \vec{v}_2 - \text{proj}_{u_1}(\vec{v}_2)$$

3. The last vector is then obtained from the the two preceding vector

$$\vec{u}_3 = \vec{v}_3 - \text{proj}_{u_1}(\vec{v}_3) - \text{proj}_{u_2}(\vec{v}_3)$$

And to finally obtain the orthonormal basis, $\vec{e}_1, \vec{e}_2$ and $\vec{e}_3$, each vector are divided by their norm such as,

$$\vec{e}_i = \frac{\vec{u}_i}{\|\vec{u}_i\|}, \quad \forall i \in \{1, 2, 3\} \tag{5.48}$$

The modified Gram-Schmidt is a version that handles the small errors such ADC rounding error or noise. Only few changes appear and the algorithm becomes

1. Set the first vector from which the other will be corrected

$$\vec{u}_1 = \vec{v}_1$$

2. The second step also remains the same,

$$\vec{u}_2 = \vec{v}_2 - \text{proj}_{u_1}(\vec{v}_2)$$

3. The last is little different and refers to the same computation as before, but indirectly

$$\vec{u}_3 = \vec{v}_3 - \text{proj}_{u_1}(\vec{v}_3) - \text{proj}_{u_2}(\vec{v}_3 - \text{proj}_{u_1}(\vec{v}_3))$$

and using the equation 5.48, the system is orthonormalized and numerically stable. For matrices reorthogonalization, the process is the same assuming that the columns of the matrix to be orthogonized are vectors. Writing a matrix $\mathbf{A}$ as $[\vec{v}_1, \vec{v}_2, \vec{v}_3]$, the algorithm can be applied.

## 5.6 Particle swarm optimization

The particle swarm optimization (PSO) is a population based stochastic method for solving optimization problem. The PSO is similar to a genetic algorithm because they are both initialized with random population of random solution. However, unlike a genetic algorithm, the particle swarm system has a memory and randomized velocity that simulate a flow of the swarm [29].
The aim of the algorithm is to find suitable value to minimize a defined cost function. The solution to optimization problem is obtained by moving a swarm of particles in a given space till the cost function is minimized. The PSO has four parameters that are, the position, the velocity, the personal best and the global best. The personal best corresponds to the best result obtained by a particle while the global best is the swarm's best, i.e. the best result obtained by all the particles. So in other words, each particles has its own velocity, position and its position that optimizes best the cost function. The velocity $\vec{v}_i$ and position $\vec{p}_i$ of a particle $i$ are updated at the instant $t+1$ the following way:

$$\begin{cases} \vec{v}_i(t+1) = w\vec{v}_i(t) + c_1 r_1(\vec{p}_{i,\text{best}} - \vec{p}_i(t)) + c_2 r_2(\vec{g}_{i,\text{best}} - \vec{p}_i(t)) \\ \vec{p}(t+1) = \vec{p}_i(t) + \vec{v}_i(t) \end{cases} \tag{5.49}$$

where $w$ is the inertial weight, $c_1$ the cognitive parameter, $c_2$ the social parameter, $r_1$ and $r_2$ are two uniform random numbers in the range of $[0, 1]$. $\vec{p}_{i,\text{best}}$ is the particle's personal best position and $\vec{g}_{i,\text{best}}$ is the swarm best position. The evolution of the speed and position of the particles in the swarm, described in equation 5.49, can be modified [30].
The algorithm can be summarized as follow:

1. Generate the swarm population with zero velocity and random position within defined bounds
2. Evaluate the cost function for each particles
3. Update the particles' personal best and the swarm global best
4. Update the velocity and position of each particle, except the one who has got the global best, using equations 5.49
5. Return to step 2 until the predefined number of iteration is fulfilled
6. The solution to the optimization problem is given by the global best

There exists several PSO algorithms which are obtained by modifying the weights, $w, c_1, c_2, r_1, r_2$, as a function to improve the accuracy of the result. The basis PSO has all the weights constant.

# 6 Experiments

## 6.1 Simulation

The simulation is done to check the validity of the algorithms by using known synthetic signals.

## 6.2 The construction of the simulation data

The signals are continuous and made from piecewise polynomial functions from degree zero up to degree five. The polynomial function can used to describe most movements that are applied to the sensor. The signal was constructed by first realizing the angle signal, then from it was obtained the angular speed as

$$\vec{\Omega}(k) = \vec{\Theta}(k) - \vec{\Theta}(k-1) \tag{6.1}$$

In this case, the derivation is simple as the signal does not contain any noise.
Afterward, from the angular speed signal, the rotation matrices characterizing the position of the sensor can be obtained. Like explained in the theory, the rotation matrices are obtained by multiplying small angle matrices because of the matrix order of multiplication. Small angles are obtained directly from the derivative of the synthetic angles signal.
For simulating the magnetometer data, the rotation matrices obtained previously are multiplied at $k = 1$ by a made-up initial vector as

$$\vec{\text{mag}}(k) = \mathbf{R}(k)\vec{\text{mag}}(k-1) \tag{6.2}$$

The acceleration synthetic data is obtained the same was as the magnetometer data.

$$\vec{a}(k) = \mathbf{R}(k)\vec{a}(k-1) \tag{6.3}$$

To all the previous signals are added Gaussian white noise of SNR of at least 20 dB.

## 6.3 Test of the calibration algorithms

The tests of each calibration are done using the synthesis data.

### 6.3.1 The calibration of the magnetometer

This calibration of the magnetometer is done using only the synthetic magnetometer data. An calibrated magnetometer has its data describing a sphere at the origin. A calibrated magnetometer gives an ellipsoid, which can be in the worst cases scenario, rotated away from the basis axes. The calibration makes use of the Extended Kalman filter as described in the chapter 5.1.2. The following transformation matrix $\mathbf{K}$ and bias $\vec{\text{bias}}_m$ is used to simulate the misalignment, scaling error and the bias of the magnetometer.

$$\mathbf{K} = \begin{bmatrix} 1,376 & 0,01 & -0,04 \\ 0,01 & 0,98 & -0,09 \\ 0,04 & -0,03 & 0,8123 \end{bmatrix} \qquad \vec{\text{bias}}_m = \begin{bmatrix} -21,5 \\ 11 \\ -50,3 \end{bmatrix}$$

The simulation is done on 20000 samples of data that is iterated many times due to the slow convergence of the Extended Kalman filter algorithm. To obtain a convergence of the values, the process noise covariance matrix and the observation noise matrix and the initial value of the state estimate must be properly set. If not, the state values will not converge to the right values. For the simulation data, $\mathbf{R} = 1 \times 10^{6}$ while the $\mathbf{Q} = 1 \times 10^{-8}\mathbf{I}$ and $\vec{x}[k = 0] = [1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0]$. The initial state value is intuitive to determine while other were obtained empirically.

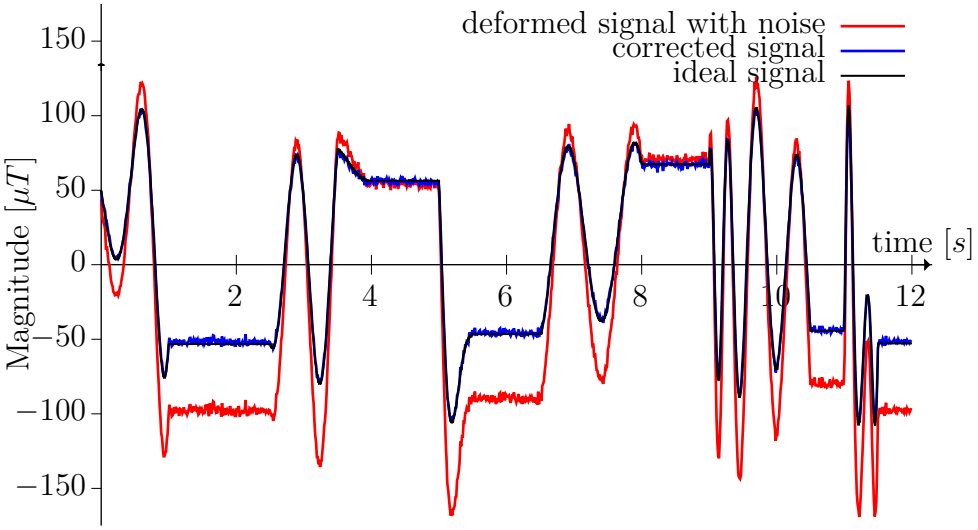


Figure 6.1: Magnetometer's synthetic data $x$

In the figure 6.1, the corrected signal matches almost perfectly the ideal signal.

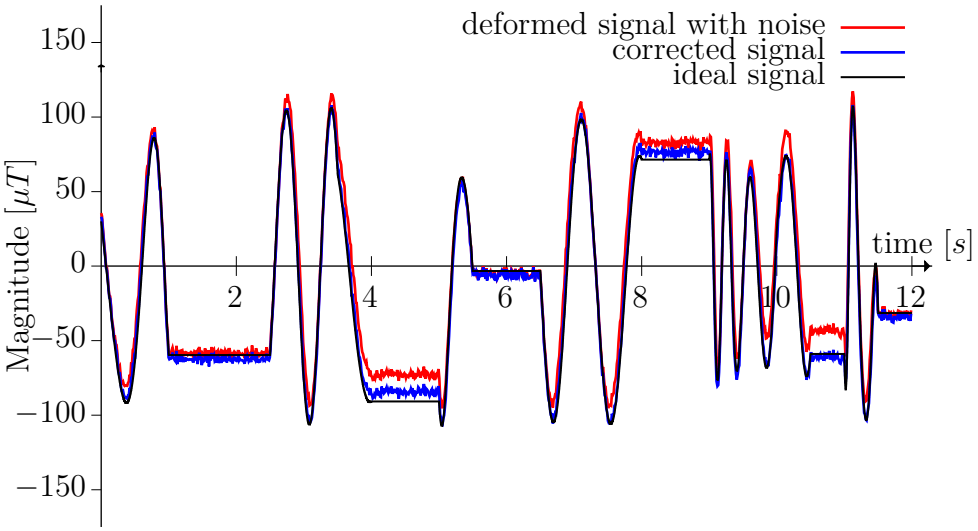


Figure 6.2: Magnetometer's synthetic data $y$

In the figure 6.2, the corrected signal does not match the ideal signal as well as in the previous case.

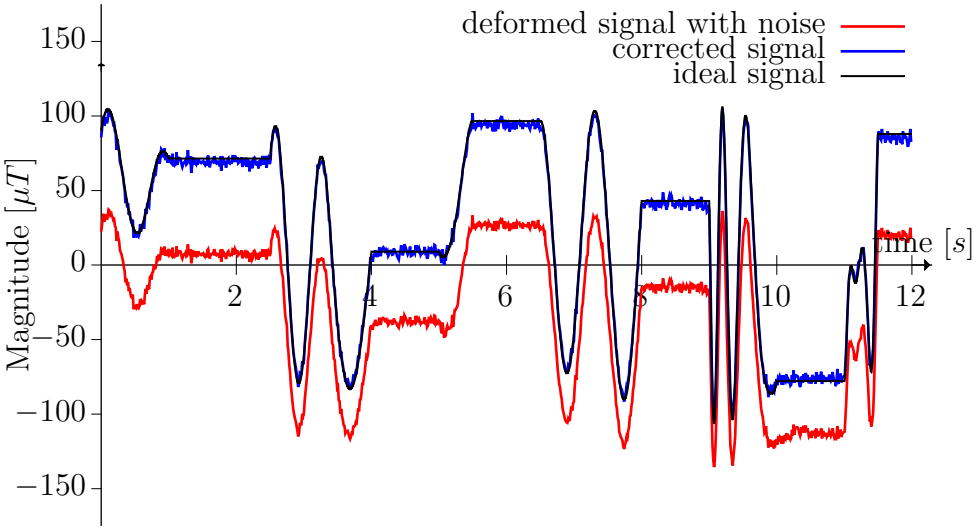


Figure 6.3: Magnetometer's synthetic data $z$

In the figure 6.3, the corrected signal matches the ideal data.
The results obtained in the figures 6.1, 6.2 and 6.3 were possible after 300 iterations. The bias values converge quickly (number of iterations $< 50$) but the matrix values require 300 iterations. The estimated matrix is

$$\mathbf{K}_{est} = \begin{bmatrix} 1,389 & 0,014 & -0,048 \\ 0,015 & 0,985 & -0,094 \\ 0,041 & -0,028 & 0,832 \end{bmatrix} \qquad \vec{\text{bias}}_m = \begin{bmatrix} -21,658 \\ 10,859 \\ -49,891 \end{bmatrix}$$

It is noticeable that the bias and the scaling values are very close to the right values while the other values corresponding to the misalignment are not so well estimated. The misalignment values influence greatly the form of the data. The error in these values should be relatively small.

### 6.3.2 The calibration of the gyrometer

The following calibration is obtained by using the calibrated magnetometer and comparing it with the data obtained from the gyrometer. The gyrometer calibration is not direct as the magnetometer was. It is because the rotation matrices corresponding to the attitude of the sensor are used to match the values of the magnetometer. This calibration algorithm also uses the Extended Kalman filter
The following matrix and vector will simulate the error in the measurement basis and the bias,

$$\mathbf{S}_g = \begin{bmatrix} 1,76 & 0,01 & -0,04 \\ -0,09 & 0,98 & 0,09 \\ 0,4 & 0,03 & 0,7123 \end{bmatrix} \qquad \vec{\text{bias}}_g = \begin{bmatrix} 2,5 \\ -1,1 \\ -3,1 \end{bmatrix}$$

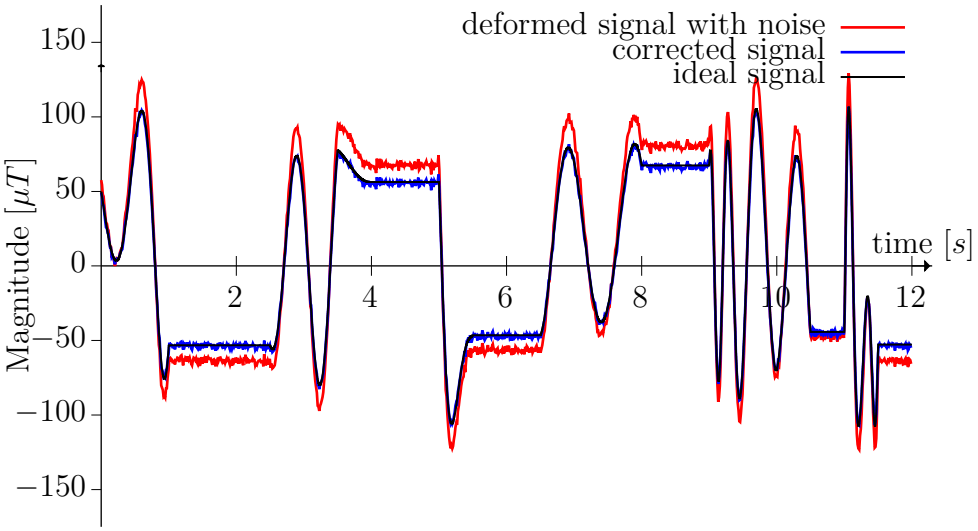


Figure 6.4: Gyro's synthetic data $x$

In the figure 6.4, the results matches closely the ideal signal.

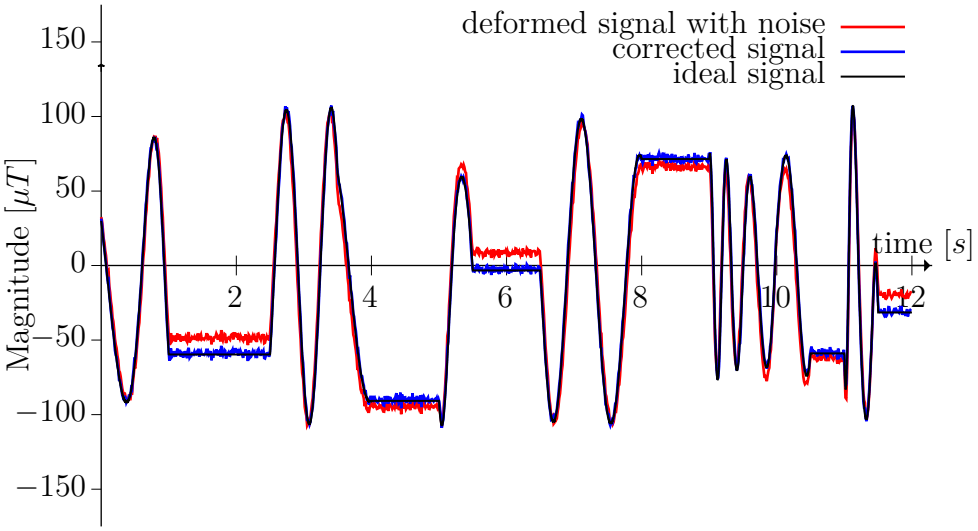


Figure 6.5: Gyro's synthetic data $y$

In the figure 6.5, the results also matches closely the ideal signal.

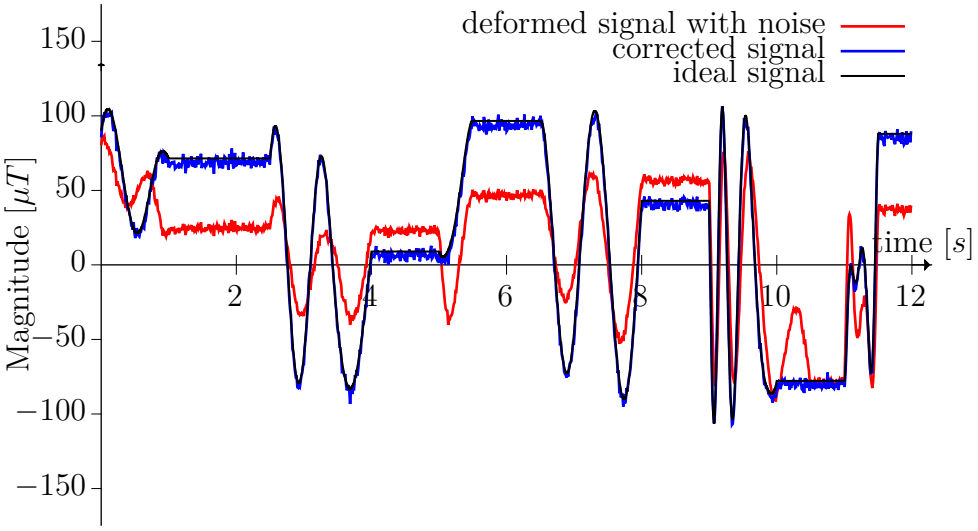


Figure 6.6: Gyro's synthetic data $z$

In the figure 6.6, the results does not match the ideal signal as well as in the previous cases.
The simulation was done on 20000 samples of data that was iterated a large number of times due to the slow convergence of the EKF. The convergence also requires good initial value for the initial state vector and the covariance matrices. In this case, the bias converges to its final values after 1000 iterations while the other values converge before 30 iterations. The initial values were $\mathbf{R} = 1 \times 10^{8}\mathbf{I}_{3\times3}$, $\mathbf{Q} = 1 \times 10^{-10}\mathbf{I}_{12\times12}$ and

the initial state value $\vec{x}[k=0] = [1,0,0,0,1,0,0,0,1,0,0,0]$. The results estimated by the algorithm are as follows

$$\mathbf{S}_g = \begin{bmatrix} 1,1785 & 0.0091 & -0,0404 \\ -0,091 & 0,9824 & 0,0894 \\ 0,3996 & 0,0297 & 0,7129 \end{bmatrix} \qquad \vec{\text{bias}}_g = \begin{bmatrix} 2,4952 \\ 1,0556 \\ -3,1352 \end{bmatrix}$$

It is again noticeable that the values representing the scaling, misalignment and bias are close to the ideal case. A slight deformation can be noticed in the graph of axis $z$ (fig. 6.6)

### 6.3.3 The calibration of the accelerometer

This calibration is the last one and is done with the help of the gyrometer. This calibration is also indirect as the rotation matrix obtained from the gyrometer are utilized. The purpose of the algorithm is to minimize the cost function given in chapter 5.1.1 that is obtained from the acceleration and rotation matrix obtained in six different positions. In this case, the values are taken when the sensor is simulated to be at rest, i.e. the vector remains stable for a short period of time. The equation is solved using a swarm particles algorithm. The number of particles in the swarm is 1000. The matrices representing the error in the sensor are

$$\mathbf{T} = \begin{bmatrix} 1,76 & 0,01 & -0,04 \\ -0,09 & 0,98 & 0,09 \\ 0,4 & 0,03 & 0,7123 \end{bmatrix} \qquad \vec{\text{bias}}_a = \begin{bmatrix} 2,5 \\ -1,1 \\ -3,1 \end{bmatrix}$$

The synthetic signals contains 20 dB SNR white Gaussian noise.

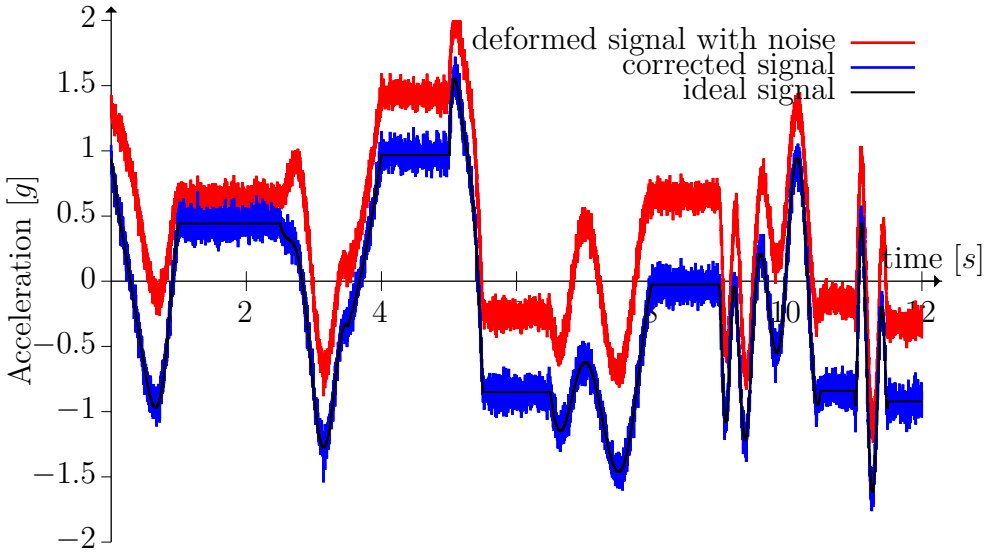


Figure 6.7: accelerometer's synthetic data $x$

In the figure 6.7, the estimated values are closely matched.

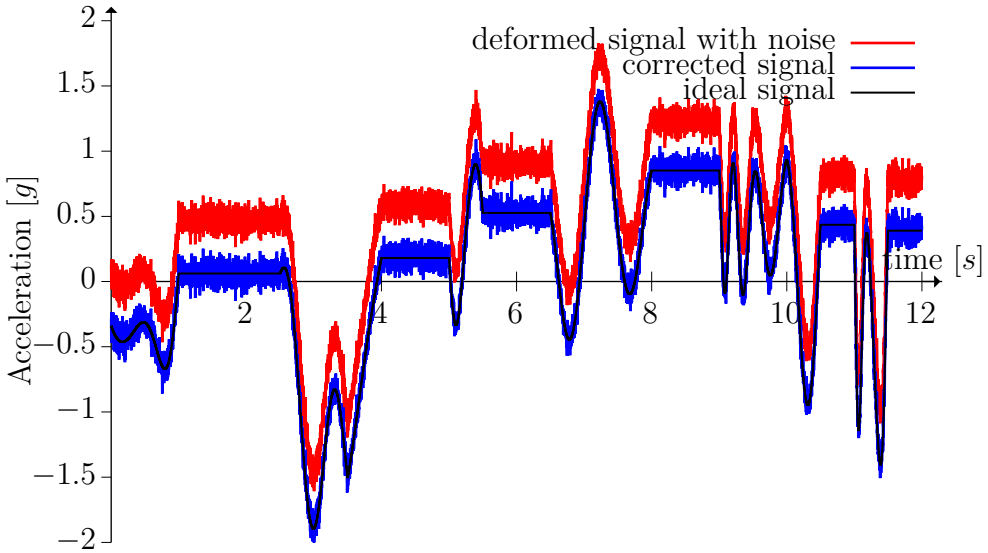


Figure 6.8: accelerometer's synthetic data $y$

In the figure 6.8, the estimated values are also closely matched.

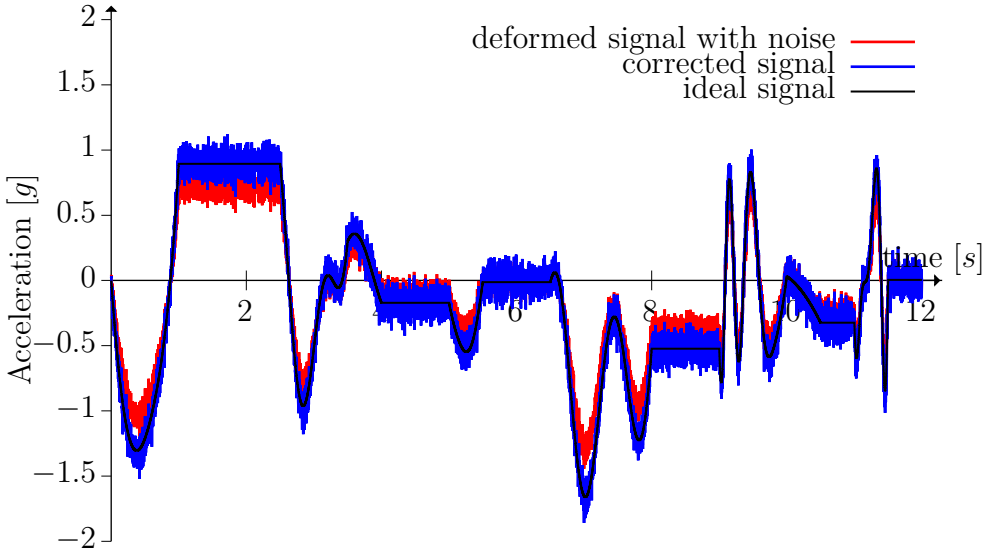


Figure 6.9: accelerometer's synthetic data $z$

In the figure 6.9, the estimated values are closely matched. The swarm particle algorithm converge quickly when good initial values are given. The initial values were $\vec{x}[k=0] = [1,0,0,0,1,0,0,0,1,0,0,0]$ with the solutions area to be for all variable between $-2$ and 2. The results are

$$\mathbf{T} = \begin{bmatrix} 1,1785 & 0.0091 & -0,0404 \\ -0,091 & 0,9824 & 0,0894 \\ 0,3996 & 0,0297 & 0,7129 \end{bmatrix} \qquad \vec{\text{bias}}_a = \begin{bmatrix} 2,4952 \\ 1,0556 \\ -3,1352 \end{bmatrix}$$

The results are close to the ideal case with only six different positions.

## 6.4 Distance

The distance is obtained by removing the gravity vector, obtained from the gyrometer, from the calibrated accelerometer data. Afterwards, the acceleration is integrated two times with respect to the time to obtain the position. In this case, the accelerometer and the gyrometer are calibrated.

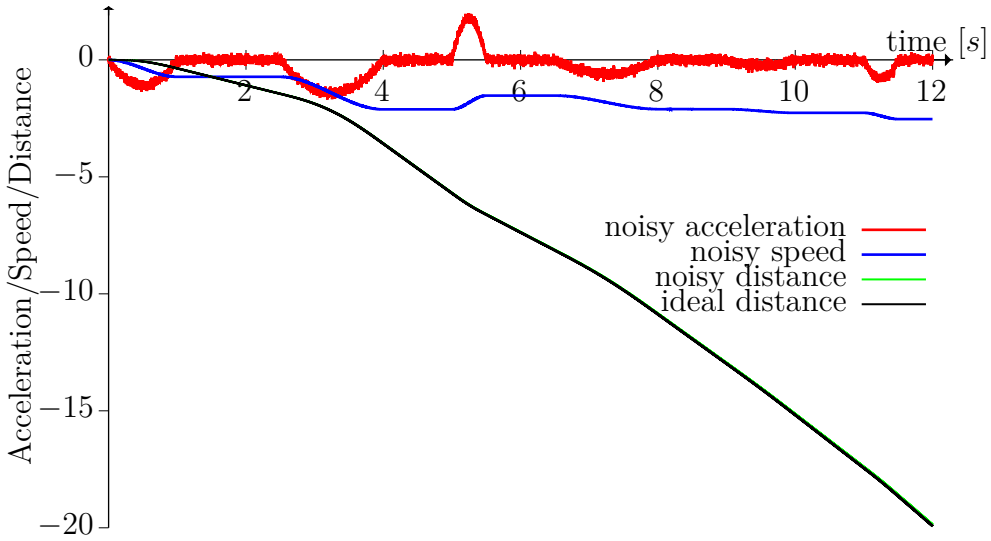


Figure 6.10: distance's synthetic data $x$

In the figure 6.10, the ideal distance is perfectly matched.

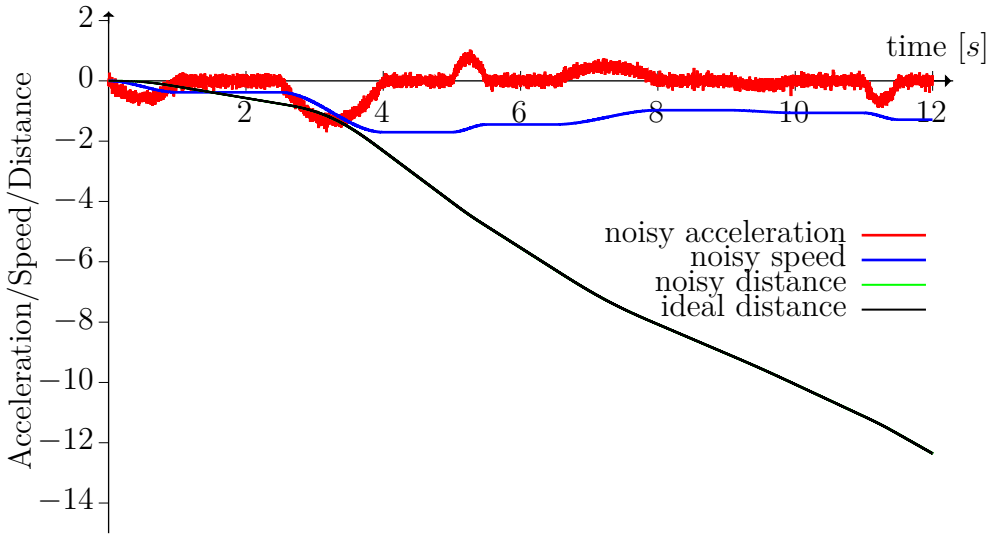


Figure 6.11: distance's synthetic data $y$

In the figure 6.11, the ideal distance is perfectly matched.

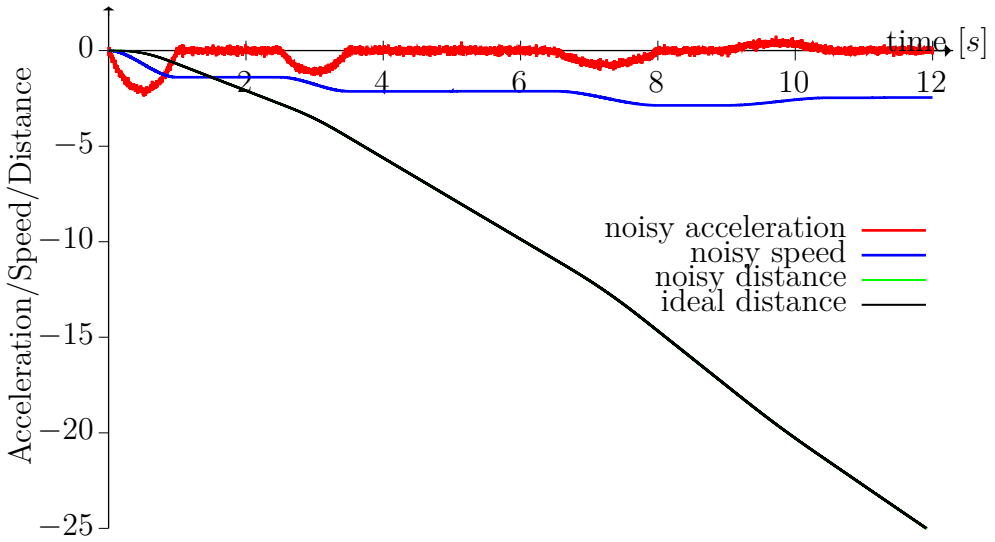


Figure 6.12: distance's synthetic data $z$

In the figure 6.12, the ideal distance is perfectly matched.
It is noticeable that the integrated white Gaussian noise behaves well. It will not be the same case in practice as the integrated noise is victim of the random walk. This algorithm performs almost flawlessly in simulation.

## 6.5 Equipment

This project was completed using a ten degree of freedom sensor, the CJMCU10DOF. The sensor board was connected via a micro-controller, the TM4C123GH6PM [31], connected to a computer via a serial connection USB. The received data is then processed by Matlab r2017b. The sensor module is composed of a barometric sensor BMP-280 and a chip MPU9250 [32] which contains an accelerometer, a gyrometer and a magnetometer, AK8963.

### 6.5.1 TM4C123GH6PM

The TM4C123GXL is a Texas Instruments evaluation board from the Tiva C series which is based on a 32-bit 80MHz ARM cortex M4F micro-controller. In this work, the purpose of the board is to retrieve the data from the CJMCU10DOF sensor board and transmit it to the computer.
The micro-controller was programmed in C language to obtain the data from the registers of the module [33] via $I^2C$ connection. The $I^2C$ is a master- multi-slave connection. In this case, only one slave is in use. The connection requires only two wires: a clock and a data wire, which means that it is half-duplex. In other words, the micro-controller sends a command which is writing bits in a register of the sensor. After few clock sequences, the sensor sends the response corresponding to the command. Each commands requires a certain number of clock cycles. The $I^2C$ bus has a theoretical maximal speed of 400kbit/s but the transfer speed is limited by the sensors updating rate. Here, the slower sensor is the magnetometer. All the words written in the USB-bus are signed char 16-bit except for the magnetometer. It sends 14 bits length words. Each update sends:

1. The delay between two updates because it is not exactly constant*. One word
2. The accelerometer data from the three axes. Three words
3. The gyrometer data from the three axes. Three words
4. The magnetometer data from the three axes. Three words
5. The temperature data. One word

* The time period between two samples is variable. During a movement of the sensor, the period can become about 100$\mu$s longer. The source of this phenomena was not found.
The total of words sent is 8 words of 16 bits and 3 of 14 bits per update. Theoretically, the maximum update would be 2352 updates per second. Counting all the delays from the sensors, the USB connection and Matlab, the update rate will be limited to about 230 updates per second.

### 6.5.2 MPU9250

The MPU-9250, manufactured by InvenSense, is composed of the MPU-6500 chip and the AK8963, manufactured by Asahi Kasei Microdevices. The AK8963 is largely used in mobile phones and is functioning using Hall effect. The MPU-6500 includes the accelerometer and the gyrometer. The AK8963 [34] chip is a magnetometer. While the accelerometer and the gyrometer are constructed on the same plane, the magnetometer AK8963, larger, is placed on top of them. This leads to each of the sensor having their own basis.

Table 1: Characteristics of the MPU-9250 [32, 34]

| Accelerometer | | | | |
|---|---|---|---|---|
| Parameters | Min | Typ | Max | Unit |
| Scale range | 2 | - | 16 | g |
| ADC word length | - | 16 | - | bits |
| Sensitivity change Vs. Temperature (−40 to 80°C) | - | $\pm 0,026$ | - | %/°C |
| Non-linearity (regression fitting) | - | $\pm 0,5\%$ | - | % |
| Cross-axis sensitivity | - | $\pm 2$ | - | % |
| Zero-G initial calibration tolerance x,y | - | $\pm 60$ | - | mg |
| Zero-G initial calibration tolerance z | - | $\pm 80$ | - | mg |
| Zero-G level change Vs. Temperature (−40 to 80°C) | - | $\pm 1,5$ | - | mg/°C |
| Noise power spectral density | - | 300 | - | $\mu g/\sqrt{\text{Hz}}$ |
| Output data rate (low noise) | 4 | - | 4000 | Hz |
| Output delay without filter | - | 0,74 | ms | |
| Gyrometer | | | | |
| Scale range | 250 | - | 2000 | °C/s |
| ADC word length | - | 16 | - | bits |
| Sensitivity scale factor variation over temperature (−40 to 80°C) | - | $\pm 4$ | - | %/°C |
| Non-linearity (regression fitting) | - | $\pm 0,5\%$ | - | % |
| Cross-axis sensitivity | - | $\pm 2$ | - | % |
| Rate noise power spectral density | - | 0.01 | - | $°C/s/\sqrt{\text{Hz}}$ |
| Output data rate (Normal mode) | 4 | - | 8000 | Hz |
| Output delay without filter | - | 0,064 | ms | |
| Magnetometer | | | | |
| Scale range | - | $\pm 4800$ | - | $\mu T$ |
| ADC word length | - | 14 | - | bits |
| Time for measurement | - | 7,2 | 9 | ms |

A major drawback of the sensor is the lack of documentation. For instance, the sensor possesses its own temperature sensor which has incomplete documentation. The registers of the sensors are also very shortly described and let place to time consuming guesses. The sensor proposes also factory calibration which are available for the gyrometer and accelerometer. The values in the register are obtained easily

but the documentation relative to how to use these values is not available. Moreover, the customer service of the company of the board or the chip are not interested in replying emails. This is most likely a compensation for the low price of the sensor.

### 6.5.3 Pressure sensor BMP-280

The pressure sensor, BMP-280, is manufactured by BOSCH Sensortec. Like mentioned earlier, it is not used because its sensitivity is too low to measure short distance in height. The datasheet [35] mentions a sensitivity of ±1m but in reality it is ±3m. Despite this, the temperature is obtained from it.

## 6.6 Experiments on real data

In this section, the algorithms are tested on the sensor MPU-9250. There are obvious differences between the synthetic signal and the signal from the sensor:

1. The accelerometer, gyrometer (table 1) and magnetometer have different delays. In other words, the data of a sensor will have different delay than the other one. This will cause problem when comparing two sensor data if not properly compensated. This compensation can be realized by delaying one signal to match the other using the values of the datasheets. Unfortunately, there are other varying delays appearing in the measurements of the sensors whose sources could not be explained.

2. The programming of the micro-controller is sequential which means that the data from the sensors will be retrieved one at a time leading to even larger delays.

3. The noise is not purely white Gaussian. This will cause noise random walks when the signal is integrated to obtain the position.

The calibration is done the same way as in the simulations but the results are difficult to verify without any additional equipment. All is known is that the sensor, if well constructed, should be close to being calibrated. In other words, the scaling factor should be very close to one and the misalignment factors should be close to zero.

### 6.6.1 Calibration of magnetometer

The magnetometer is highly noisy as it can be noticed in the following figures 6.13, 6.14, 6.15 and 6.16.

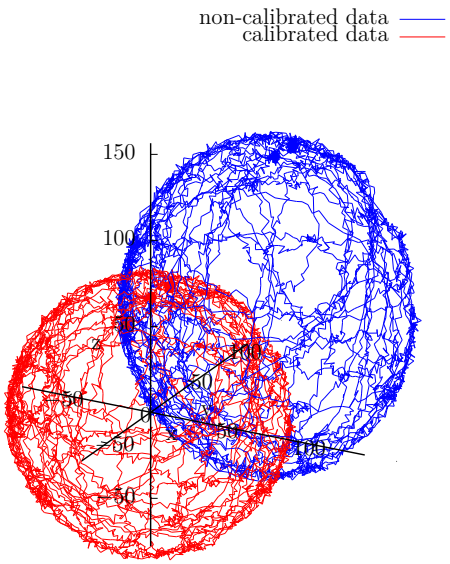


Figure 6.13: Magnetometer data

The results obtained are

$$S = \begin{bmatrix} 0,9926 & 0.0141 & 0.0179 \\ 0.0148 & 0,8046 & 0,0016 \\ 0,0207 & 0,0051 & 1,1685 \end{bmatrix} \qquad \vec{\text{bias}}_m = \begin{bmatrix} 47.4149 \\ 46,7079 \\ 54,496 \end{bmatrix}$$

The results are roughly checked by the form of the three-dimensional graph of the data (fig. 6.13). If it is still an ellipsoid, the calibration did not work. If it is a sphere at the origin like in the case, the scaling factor and bias are estimated correctly. In this case, it is a sphere. For the misalignment factors, they cannot be verified without proper equipment. It is just known that they should be small if the chip is well constructed.

### 6.6.2 Calibration of gyrometer

The calibration of the gyrometer uses the data of the calibrated magnetometer. The results are difficult to verify without any external equipment.

$$K = \begin{bmatrix} 0,9689 & -0,028 & -0.0182 \\ -0,0016 & 0,9477 & 0,011 \\ -0,0402 & -0,0018 & 1,028 \end{bmatrix} \qquad \vec{\text{bias}}_m = \begin{bmatrix} 0.0003 \\ 0.0171 \\ 0,0377 \end{bmatrix}$$

The same assumption as for the magnetometer can be made. The scaling value should be close to one and the other elements should be close to zero.

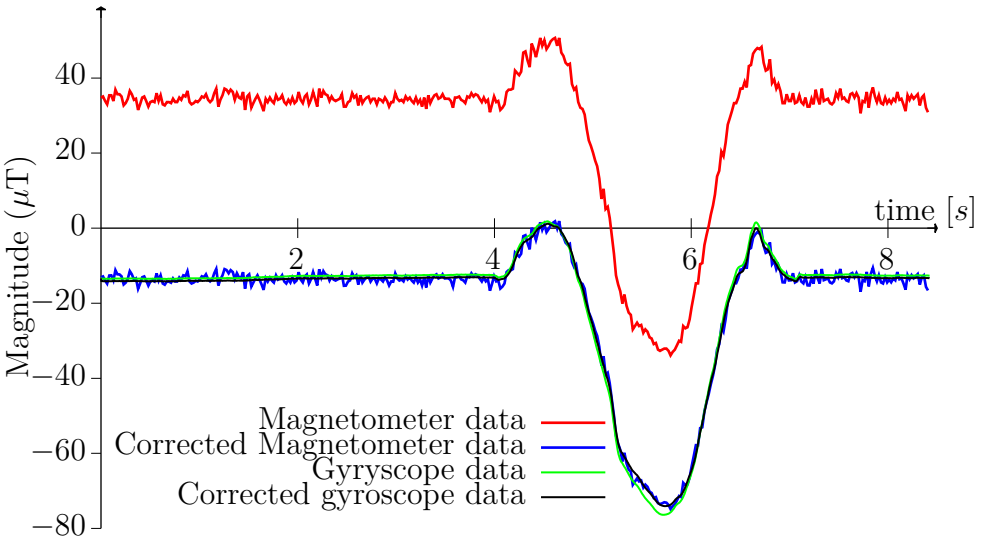


Figure 6.14: gyrometer and magnetometer data $x$

In the figure 6.14, the corrected data obtained from the gyrometer is close to the magnetometer data.

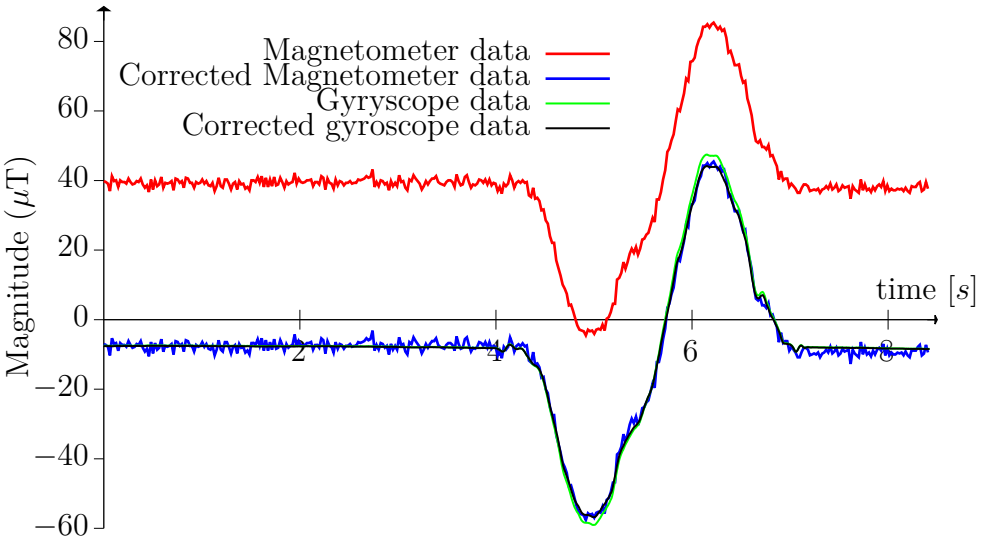


Figure 6.15: gyrometer and magnetometer data $y$

In the figure 6.15, the corrected data obtained from the gyrometer is close to the magnetometer data.

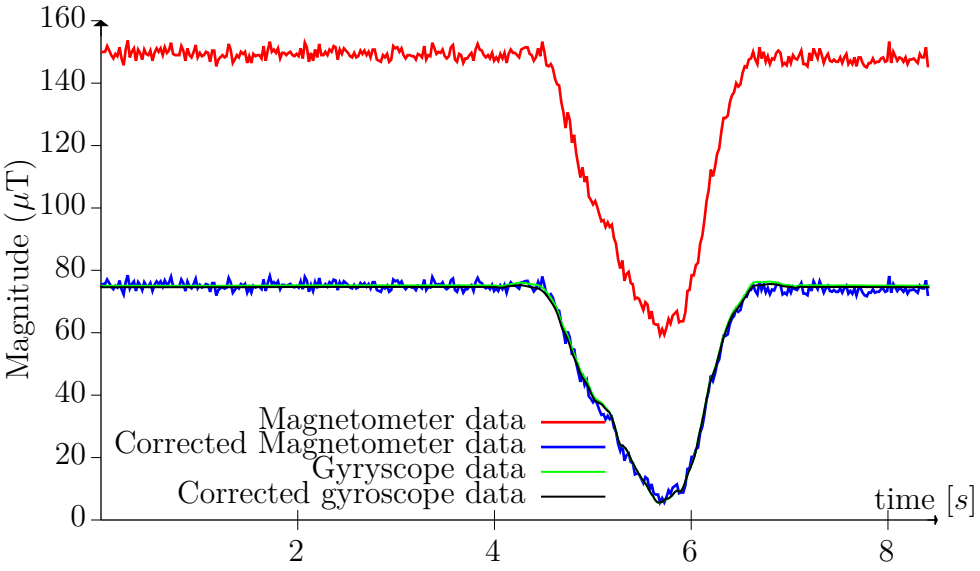


Figure 6.16: gyrometer and magnetometer data $z$

In the figure 6.16, the corrected data obtained from the gyrometer is close to the magnetometer data.
As wanted, the magnetic field vector obtained from the gyrometer matches very closely the vector measured by the calibrated magnetometer.

### 6.6.3 Calibration of accelerometer

The calibration of the acceleration is also done with the gyrometer. Only the scaling and the bias can be verified as the magnitude of the acceleration vector should be equal to $1g$.

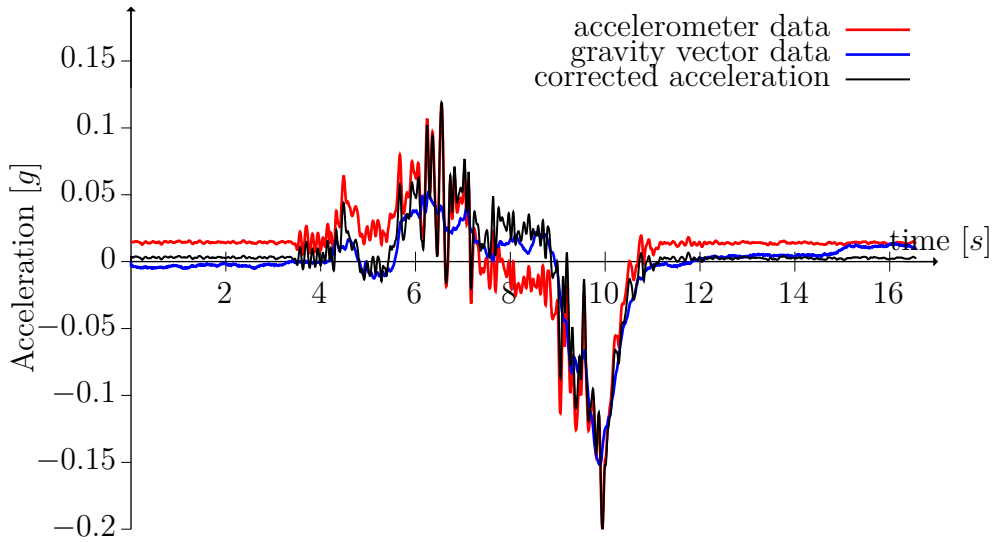


Figure 6.17: Accelerometer and gyrometer data $x$

In the figure 6.17, the calibrated data of the accelerometer is closer to the calibrated data of the gyrometer.

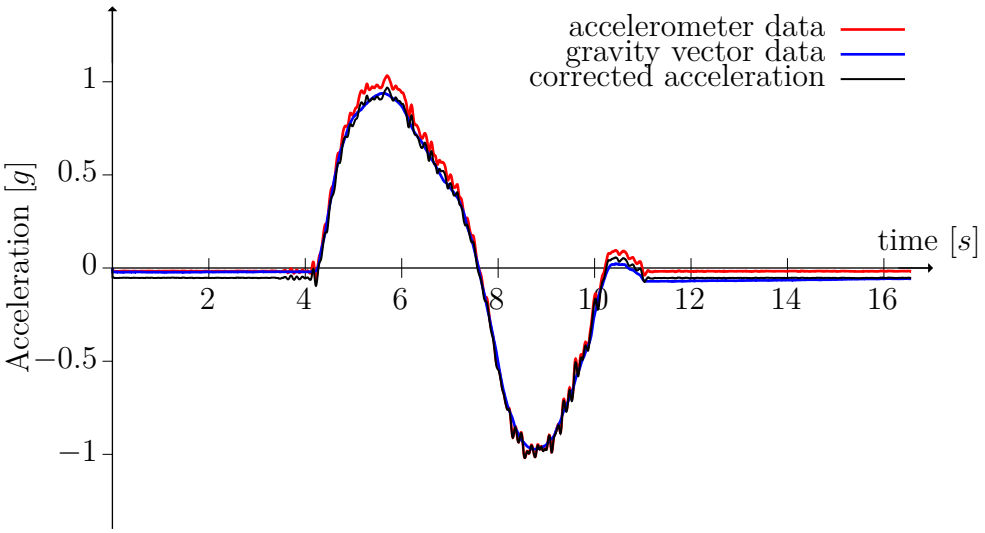


Figure 6.18: Accelerometer and gyrometer data $y$

In the figure 6.18, the calibrated data of the accelerometer is closer to the calibrated data of the gyrometer. In the figure 6.19, the calibrated data of the accelerometer is

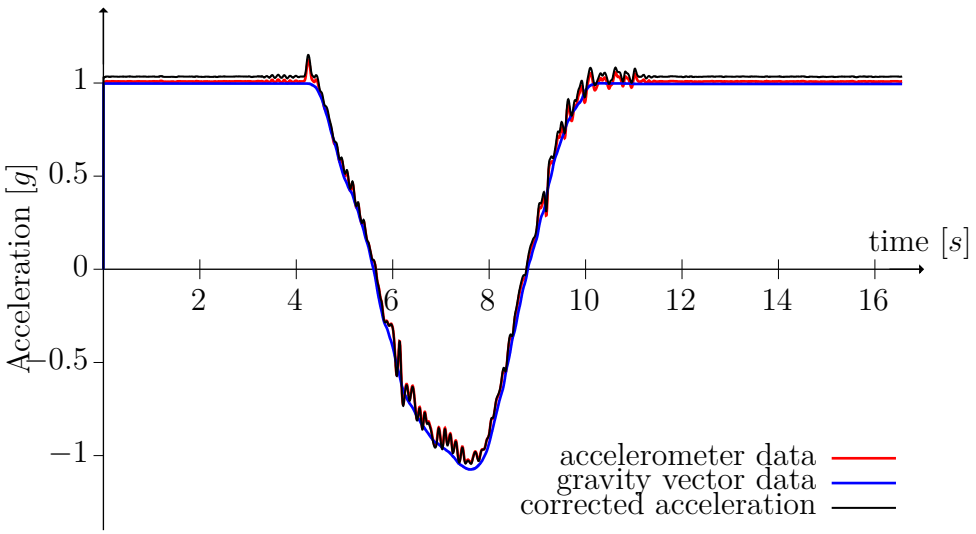


Figure 6.19: Accelerometer and gyrometer data $z$

not any closer to the calibrated gyrometer than uncalibrated data of the gyrometer. The calibration of the accelerometer does not work very well. This can be due to the accumulation of error from all the calibration altogether.

### 6.6.4 Results of the position estimation

The estimation of the position is done by obtaining the acceleration imposed by the movement of the sensor. Because the accelerometer measures the acceleration of the sensor including the gravitational acceleration, the useful data must be separated. The simplest way is the one explained in the chapter 5.4. The gravitational vector is estimated with the help of the gyrometer and is then subtracted from the data measured from the accelerometer.
The following experiment is realized by moving by hand the IMU along a straight line in the xy-plane.

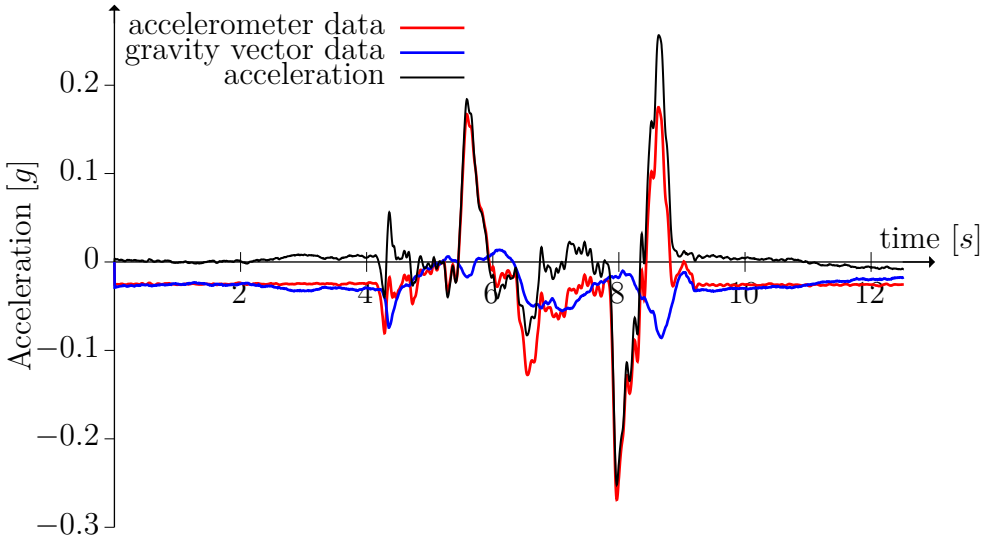


Figure 6.20: distance's data $x$

In the figure 6.20, the acceleration is close to zero when it needs to be.

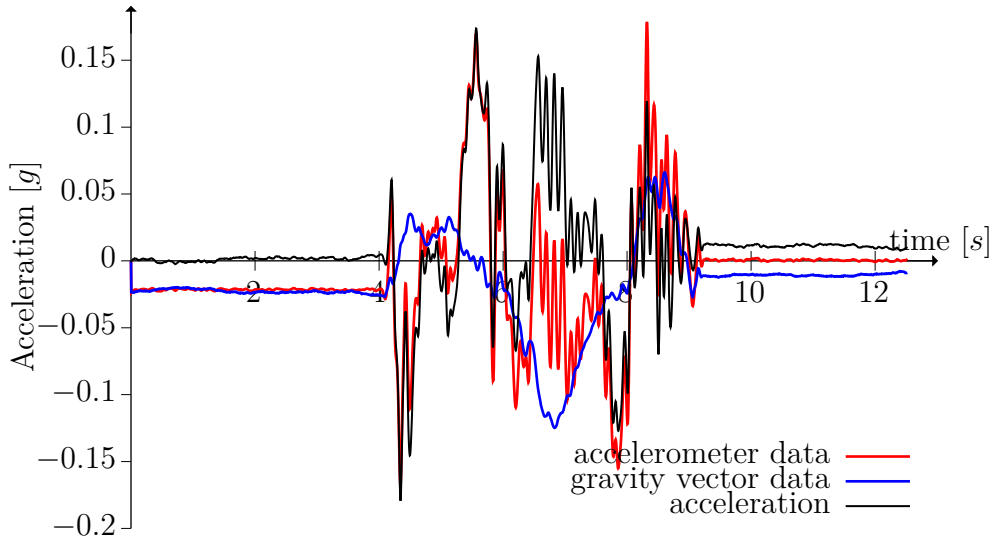


Figure 6.21: Acceleration's data $y$

In the figure 6.21, the acceleration is, at the end, distant from zero when it should be exactly zero.

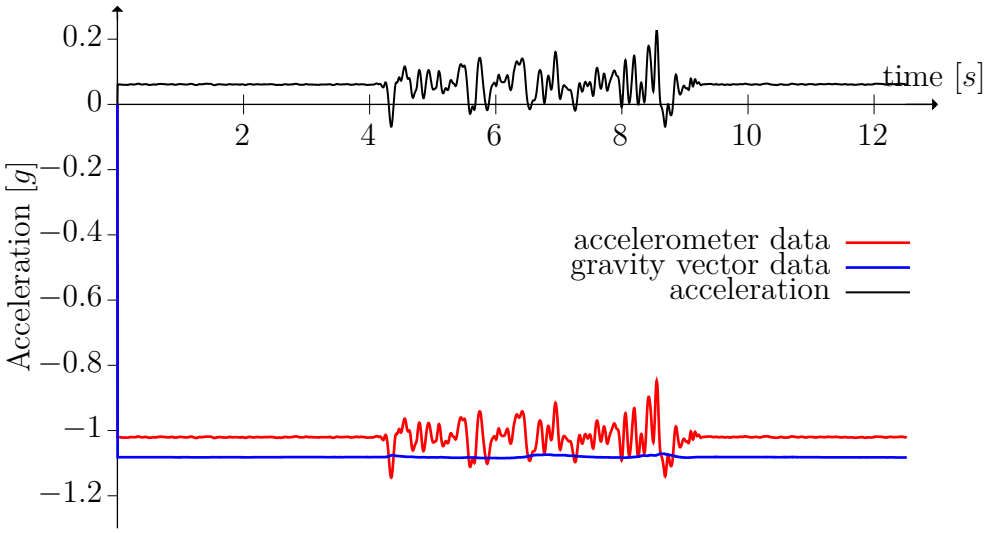


Figure 6.22: Acceleration's data $z$

In the figure 6.22, the acceleration is completely distinct from zero when it should be exactly zero.

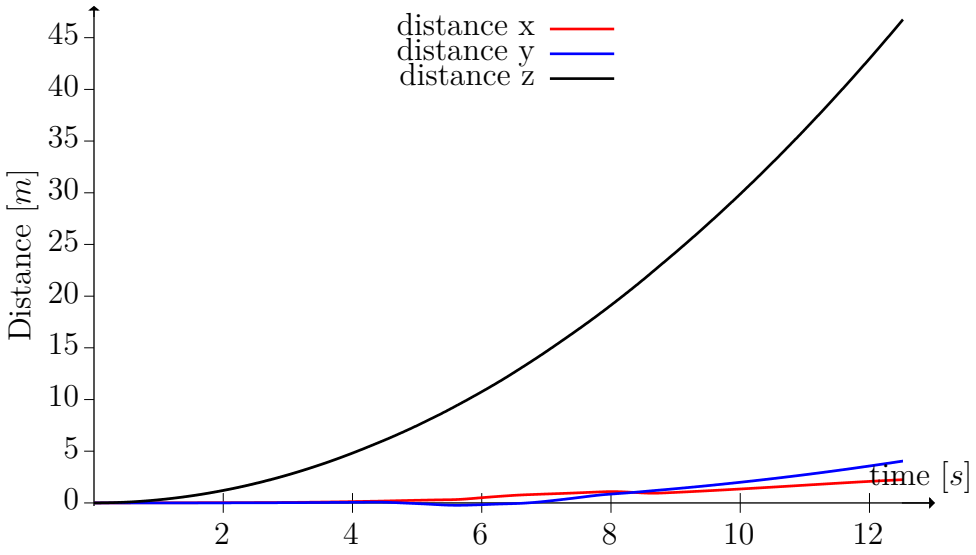


Figure 6.23: distance's data $z$

As expected from the figures 6.20, 6.21 and 6.22, the estimation of the position, shown in figure 6.23, will be completely off the real value. The bias on the z-axis was poorly removed which, when integrated two times, increases improperly the distance estimated. It can be noticed that the curve according to the axis z follows a $t^2$-form which point to a two-time integration of a constant. The constant is the bias which was not properly estimated. The two other distances distinctly follow the $t^2$-form towards the end because the bias was more accurately estimated and its effects are less apparent in the beginning. Even a small error in the estimation of the bias will leads to a wrong estimation of the distance.

# 7 Discussion and conclusion

The estimation of the position without algorithm that would correct the estimation seems hardly possible. This would require a perfect calibration for the accelerometer and the gyrometer. The gyrometer would have to perfectly match the gravity vector. This is because any residue of bias or noise or mismatch between the accelerometer gravitational vector and gyrometer gravitational vector will just cause the distance estimation to be unreliable. For instance, if the error in the acceleration estimation would be $1\mu g$, it would only take 1414 samples or about 6 seconds to be one meter off and one minute to be off hundred meter. One $\mu g$ would not even be viable because the noise standard deviation itself is larger that one $\mu g$.
The sensor needs first better calibration techniques. The EKF works fairly well but it is slow to converge and requires good initial values for the covariance matrices. An improvement could be done by using the Unscented Kalman filter which approximates non-linear functions at the distribution level using sigma point and not using Taylor series approximation. The sensor itself seems to suffer from delays that are difficult to estimate. It is most noticeable when comparing the magnetometer with the gyrometer. When the direction changes abruptly, the gyrometer changes faster that the magnetometer. This phenomena spoils the estimation of the error matrix. The magnetometer is not used other than for the calibration because it is highly unreliable indoor due to the local magnetic field being influenced by any electrical equipment nearby.
Obtaining the position the same way as in this thesis could work if a correction algorithm at the acceleration estimation was added. In most papers, a GPS-tracking system or local position estimation (WIFI, Bluetooth, sensor) is utilized to compensate the drifting of the position. But this makes the purpose of the IMU sensor quite questionable as the position is barely influenced by the IMU measurement. In this work, the noise was not removed because at this point, it was not the issue. But when the precision is enough, using adaptive spline type or EEMD (ensemble empirical mode decomposition) filtering techniques could be a good choice as the acceleration can be easily described as a piecewise polynomials where the interval time would be adaptive.

To conclude, in this thesis an algorithm to estimate the position of an IMU sensor without any tracking device was proposed. The algorithm reveals that the estimation of the position using only the IMU is difficult to achieve as the slightest error in the estimation of the acceleration of the sensor will lead to wrong estimation of the position. The calibration is done without any external equipment, and therefore the results are impossible to verify precisely. The error in calibration accumulates from calibration to calibration. Finally, the algorithm does not work properly and requires at least a correction algorithm at the integral level for the speed and then the distance.

# References


[1] Rezaei S. and Sengupta R., Kalman Filter-Based Integration of DGPS and Vehicle Sensors for Localization, *IEEE Transactions on Control System Technology*, 2007. ISSN: 1558-0865, DOI: 10.1109/TCTST.2006.886439

[2] Benini A., Mancini A. and Longhi S., An IMU/UWB/Vision-based Extended Kalman Filter for Mini-UAV Localization in Indoor Environment using 802.15.4a Wireless sensor Network, *J Intell Robot Syst (2013) 70:461-476, Springer science+Business Media B.V.2012*, DOI: 10.1007/s10846-012-9742-1

[3] Jiménez A. R., Seco F., Zampella F., Prieto J. C. and Guevara J., Indoor Localization of Persons in ALL Scenarios Using an Inertial Measurement Unit (IMU) and the Signal Strength (SS) from RFID tags, *Evaluating AAL Systems Through Competitive Benchmarking, International Competitions and Final Workshop EvAAL 2012, Springer*, 2013, DOI: 10.1007/978-3-642-37419-7, ISBN: 978-3-642-37419-7

[4] Hesch J. A., Kottas D. G., Bowman S. L. and Roumeliotis S., Camera-IMU-based localization: Observability analysis and consistency improvement, *The international Journal of Robotics research*, 2014, DOI:10.1177/0278364913509675

[5] LE PETIT LAROUSSE illustré 2011, 2010 Paris, ISBN: 978-2-03-584088-2

[6] Raj, V. P., Natarajan, K. and Girikumar, T., G. Induction motor fault detection and diagnosis by vibration analysis using MEMS accelerometer *Emerging Trends in Communication, Control, Signal Processing & Computing Applications (C2SPCA)*,2013 International Conference. ISBN: 978-1-4673-5222-2, DOI: 10.1109/C2SPCA.2013.6749391

[7] Dinardo, L., Fabbiano, E. and Vacca, G. Fluid flow rate estimation using acceleration sensors *IEEE Sensing Technology (ICST), 2013 Seventh International Conference*. ISBN: 978-1-4673-5222-2, DOI: 10.1109/ICSensT.2013.6727646.

[8] Zhu R., Sun D., Zhou Z. and Wang D., A linear fusion algorithm for attitude determination using low cost MEMS-based sensors. *Measurement, ELSEVIER*, 2006, DOI:10.10.12/j.measurement.2006.05.020

[9] Cai, W., Chan J. and Garmire, D. 3-Axes MEMS Hall-Effect Sensor *Sensors Application Symposium (SAS), 2011 IEEE*. ISBN: 978-1-4244-8064-7, DOI:10.1109/SAS.2011.5739801.

[10] Zhang Z., Edwan E., Zhou J., Chai W. and Loffeld O., Performance investigation of barometer aided GPS/MEMS-IMU integration, *Position Location and Navigation Symposium (PLANS)*, 2012, DOI: 10.1109/PLANS.2012.6236933. ISBN: 978-1-4673-0387-3

[11] Wald L. Some Terms of Reference in Data Fusion, *IEEE Transactions on Geoscience and Remote Sensing*, 1999, DOI: 10.1109/36.763269, ISSN: 1558-0644

[12] Goldman R. Rethinking Quaternions: Theory and computation, *Morgan & Claypool Publisher*, 2010, ISBN: 978-1-6084-5420-4

[13] Davis J. P. and Robinowitz P. Methods of Numerical Integration, Computer Science and Applied Mathematics, *Academic Press, 2nd edition*, 2014, ISBN: 978-1-4832-6428-8

[14] Robinson I. A comparison of numerical integration programs, *Journal od Computational and Applied Mathematics, volume 5*, 1979, DOI:10.1016/0377-0427(79)90006-2

[15] Kok M., Hol J., D., Schön B., T., Gustafsson F. and Luinge H. Calibration of a magnetometer in combination with inertial sensors, *Information Fusion (Fusion), 2012 15th International Conference*, 2012, ISBN: 978-0-9824438-4-2

[16] LaFond, P. H. Modeling for error reduction in vibrating beam accelerometer *Position Location and Navigation Symposium, 1992. Record. 500 years After Colombus - Navigation Challenges of Tomorrow. IEEE PLANS '92., IEEE.* ISBN: 0-7803-0468-3, DOI:10.1109/PLANS.1992.185830.

[17] Qi, B., Cheng, J. and Zhao, L. A novel temperature drift error model for MEMS capacitive Accelerometer *Advanced Information Technology, Electronic and Automation Control Conference (IAEAC),2017 IEEE 2nd.* ISBN: 978-1-4673-8979-2, DOI:10.1109/IAEAC.2017.8054002.

[18] Dhalwar, S., Kottah, R., Kumar, V., Raj, A., N., J., and Poddar, S. Adaptive Parameter based Particle Swarm Optimization for Accelerometer calibration, *1st IEEE International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)*, 2016, ISBN: 978-1-4673-8587-9

[19] Ge Z., Liu S., Li G., Huang Y., Wang Y., Error model of geomagnetic-field measurement and extended Kalman-filter based compensation method, *PloS ONE 12(4):e0173962*, 2017, DOI:10.1371/journal.pone.0173962

[20] Magnetic Declination http://www.magnetic-declination.com, Accessed: 26.04.2018

[21] Chulliat A. and Macmillan S., The US/UK World Magnetic Model for 2015-2020, *National Geophysical Data Center, NOAA*, 2015, DOI: 10.7289/V5TB14V7

[22] Jewett, J., W. and Serway, R., A. Physics for Scientific and Engineers with Modern Physics, *Central Learning EMEA*, 2008, ISBN: 978-0-4951-1240-2

[23] Kay, S., M. Fundamentals of Statistical Signal Processing Estimation Theory *Prentice Hall, Inc, 1993 New Jersey.* ISBN: 0-13-345711-7.

[24] Wu, Y. and Pei, L. Gyroscope Calibration via Magnetometer, *IEEE Sensors journal Vol. 17, NO. 12*, 2017, DOI: 10.1109/JSEN.2017.2720756

[25] Beravs T., Podobnik, J. and Munih M. Three-Axial Accelerometer Calibration Using Kalman Filter covariance Matrix for Online Estimation of Optimal Sensor Orientation, *IEEE Transaction on Instrumentation and Measurement, Vol. 61, NO. 9*, 2012, DOI: 10.1109/TIM.2012.2187360

[26] Lu, X. and Liu, Z. IEKF-based Self-Calibration Algorithm for Triaxial Accelerometer, *3rd International Conference on Information Science and Control Engineering*, 2016, DOI: 10.1109/ICISCE.2016.213, IBSN: 978-1-5090-2535-0

[27] Groves, P., D. Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems *Artech House, 2008 Boston*. ISBN: 978-1-58053-255-6.

[28] Giraud, L., Langou, J. and Rozloznik, M. The Loss of Orthogonality in the Gram-Schmidt Orthogonalization Process *Computer and Mathematics with Application Volume 50, Issue 7, 2005*. DOI:10.1016/j.camwa.2005.08.009.

[29] Kennedy, J. and Eberhart, R. A new optimizer using particle swarm theory *Micro Machine and Human Sciences, 1995. MHS '95., Proceedings of the Sixth International Symposium on*. DOI:10.1109/MHS.195.494215. ISBN:0-7803-2676-8

[30] Masuda, K., Ishikawa, K., Sekozawa, T. and Kurihara, K. A Multiple Optimal Solution Search Method by Using a Particle Swarm Optimization Algorithm Utilizing the Distribution of Personal Bests, *IEEE Congress on Evolutionary Computation*, 2013, ISBN: 978-1-4799-0454-9

[31] Tiva™ TM4C123GH6PM Microcontroller, Data Sheet, *DS-TM4C123GH6PM-15842.2741, SPMS376E*, 2014, http://www.ti.com/lit/ds/spms376e.pdf

[32] MPU-9250 Product Specification Revision 1.1, *PS-MPU-9250A-01*, 2016, http://www.invensense.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf, Accessed: 27.04.2018.

[33] MPU-9250 Register Map andDescription Revision 1.6, *RM-MPU-9250A-00*, 2015, http://www.invensense.com/wp-content/uploads/2017/11/RM-MPU-9250A-00-v1.6.pdf, Accessed: 27.04.2018.

[34] AK8963, 3-axis Electronic Compass, *MS1356-E-02*, 2013, https://www.akm.com/en/file/datasheet/AK8963C.pdf, Accessed: 27.04.2018

[35] Data sheet BMP280, Digital Pressure Sensor, *BST-BMP280-DS001-19 r.1.19*, 2018, https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BMP280-DS001-19.pdf, Accessed: 27.04.2018

# A Appendix A

Result of the Jacobian calculated in the section 5.1.2 for the EKF.

$$
\mathbf{J}_c(t) = 2 \begin{bmatrix}
(m_{m,x}(t) - b_{c,x})(b_{c,x}K_{c,x} + b_{c,y}K_{c,xy} + b_{c,z}K_{c,xz} - K_{c,x}m_{m,x}(t) - K_{c,xy}m_{m,y}(t) - K_{c,xz}m_{m,z}(t)) \\
(m_{m,y}(t) - b_{c,y})(b_{c,x}K_{c,x} + b_{c,y}K_{c,xy} + b_{c,z}K_{c,xz} - K_{c,x}m_{m,x}(t) - K_{c,xy}m_{m,y}(t) - K_{c,xz}m_{m,z}(t)) \\
(m_{m,z}(t) - b_{c,z})(b_{c,x}K_{c,x} + b_{c,y}K_{c,xy} + b_{c,z}K_{c,xz} - K_{c,x}m_{m,x}(t) - K_{c,xy}m_{m,y}(t) - K_{c,xz}m_{m,z}(t)) \\
(m_{m,x}(t) - b_{c,x})(b_{c,x}K_{c,yx} + b_{c,y}K_{c,y} + b_{c,z}K_{c,yz} - K_{c,yx}m_{m,x}(t) - K_{c,y}m_{m,y}(t) - K_{c,yz}m_{m,z}(t)) \\
(m_{m,y}(t) - b_{c,y})(b_{c,x}K_{c,yx} + b_{c,y}K_{c,y} + b_{c,z}K_{c,yz} - K_{c,yx}m_{m,x}(t) - K_{c,y}m_{m,y}(t) - K_{c,yz}m_{m,z}(t)) \\
(m_{m,z}(t) - b_{c,z})(b_{c,x}K_{c,yx} + b_{c,y}K_{c,y} + b_{c,z}K_{c,yz} - K_{c,yx}m_{m,x}(t) - K_{c,y}m_{m,y}(t) - K_{c,yz}m_{m,z}(t)) \\
(m_{m,x}(t) - b_{c,x})(b_{c,x}K_{c,zx} + b_{c,y}K_{c,zy} + b_{c,z}K_{c,z} - K_{c,zx}m_{m,x}(t) - K_{c,zy}m_{m,y}(t) - K_{c,z}m_{m,z}(t)) \\
(m_{m,y}(t) - b_{c,y})(b_{c,x}K_{c,zx} + b_{c,y}K_{c,zy} + b_{c,z}K_{c,z} - K_{c,zx}m_{m,x}(t) - K_{c,zy}m_{m,y}(t) - K_{c,z}m_{m,z}(t)) \\
(m_{m,z}(t) - b_{c,z})(b_{c,x}K_{c,zx} + b_{c,y}K_{c,zy} + b_{c,z}K_{c,z} - K_{c,zx}m_{m,x}(t) - K_{c,zy}m_{m,y}(t) - K_{c,z}m_{m,z}(t)) \\
m_{m,x}(t)(K_{c,x}^2 + K_{c,yx}^2 + K_{c,zx}^2) - b_{c,x}(K_{c,x}^2 + K_{c,yx}^2 + K_{c,zx}^2) - b_{c,y}(K_{c,x}K_{c,xy} + K_{c,yx}K_{c,y} + K_{c,zx}K_{c,zy}) - b_{c,z}(K_{c,x}K_{c,xz} + \\
K_{c,yx}K_{c,yz} + K_{c,zx}K_{c,z}) + m_{m,y}(t)(K_{c,x}K_{c,xy} + K_{c,yx}K_{c,y} + K_{c,zx}K_{c,zy}) + m_{m,z}(t)(K_{c,x}K_{c,xz} + K_{c,yx}K_{c,yz} + K_{c,zx}K_{c,z}) \\
m_{m,y}(t)(K_{c,xy}^2 + K_{c,y}^2 + K_{c,zy}^2) - b_{c,y}(K_{c,xy}^2 + K_{c,y}^2 + K_{c,zy}^2) - b_{c,x}(K_{c,x}K_{c,xy} + K_{c,yx}K_{c,y} + K_{c,zx}K_{c,zy}) - b_{c,z}(K_{c,xy}K_{c,xz} + \\
K_{c,y}K_{c,yz} + K_{c,zy}K_{c,z}) + m_{m,x}(t)(K_{c,x}K_{c,xy} + K_{c,yx}K_{c,y} + K_{c,zx}K_{c,zy}) + m_{m,z}(t)(K_{c,xy}K_{c,xz} + K_{c,y}K_{c,yz} + K_{c,zy}K_{c,z}) \\
m_{m,z}(t)(K_{c,xz}^2 + K_{c,yz}^2 + K_{c,z}^2) - b_{c,z}(K_{c,xz}^2 + K_{c,yz}^2 + K_{c,z}^2) - b_{c,x}(K_{c,x}K_{c,xz} + K_{c,yx}K_{c,yz} + K_{c,zx}K_{c,z}) - b_{c,y}(K_{c,xy}K_{c,xz} + \\
K_{c,y}K_{c,yz} + K_{c,zy}K_{c,z}) + m_{m,x}(t)(K_{c,x}K_{c,xz} + K_{c,yx}K_{c,yz} + K_{c,zx}K_{c,z}) + m_{m,y}(t)(K_{c,xy}K_{c,xz} + K_{c,y}K_{c,yz} + K_{c,zy}K_{c,z})
\end{bmatrix}
$$

# B Appendix B

Result of the Jacobian $\mathbf{J}_g(t)$ calculated in the section 5.1.3 for the EKF.

$$
\begin{bmatrix}
R_x(t)(m_{m,x}(t-1)-b_{g,x})+R_{xy}(t)(m_{m,y}(t-1)-b_{g,y})+ & 0 & 0 \\
R_{xz}(t)(m_{m,z}(t-1)-b_{g,z}) & & \\
R_{yx}(t)(m_{m,x}(t-1)-b_{g,x})+R_y(t)(m_{m,y}(t-1)-b_{g,y})+ & 0 & 0 \\
R_{yz}(t)(m_{m,z}(t-1)-b_{g,z}) & & \\
R_{zx}(t)(m_{m,x}(t-1)-b_{g,x})+R_{zy}(t)(m_{m,y}(t-1)-b_{g,y})+ & 0 & 0 \\
R_z(t)(m_{m,z}(t-1)-b_{g,z}) & & \\
0 & R_x(t)(m_{m,x}(t-1)-b_{g,x})+R_{xy}(t)(m_{m,y}(t-1)-b_{g,y})+ & \\
 & R_{xz}(t)(m_{m,z}(t-1)-b_{g,z}) & 0 \\
0 & R_{yx}(t)(m_{m,x}(t-1)-b_{g,x})+R_y(t)(m_{m,y}(t-1)-b_{g,y})+ & \\
 & R_{yz}(t)(m_{m,z}(t-1)-b_{g,z}) & 0 \\
0 & R_{zx}(t)(m_{m,x}(t-1)-b_{g,x})+R_{zy}(t)(m_{m,y}(t-1)-b_{g,y})+ & \\
 & R_z(t)(m_{m,z}(t-1)-b_{g,z}) & 0 \\
0 & 0 & R_x(t)(m_{m,x}(t-1)-b_{g,x})+R_{xy}(t)(m_{m,y}(t-1)-b_{g,y})+ \\
 & & R_{xz}(t)(m_{m,z}(t-1)-b_{g,z}) \\
0 & 0 & R_{yx}(t)(m_{m,x}(t-1)-b_{g,x})+R_y(t)(m_{m,y}(t-1)-b_{g,y})+ \\
 & & R_{yz}(t)(m_{m,z}(t-1)-b_{g,z}) \\
0 & 0 & R_{zx}(t)(m_{m,x}(t-1)-b_{g,x})+R_{zy}(t)(m_{m,y}(t-1)-b_{g,y})+ \\
 & & R_z(t)(m_{m,z}(t-1)-b_{g,z}) \\
S_{g,x}R_x(t)+S_{g,xy}R_{yx}(t)+S_{g,xz}R_{zx}(t) & S_{g,yx}R_x(t)+S_{g,y}R_{yx}(t)+S_{g,yz}R_{zx}(t) & S_{g,zx}R_x(t)+S_{g,zy}R_{yx}(t)+S_{g,z}R_{zx}(t) \\
S_{g,x}R_{xy}(t)+S_{g,xy}R_y(t)+S_{g,xz}R_{zy}(t) & S_{g,yx}R_{xy}(t)+S_{g,y}R_y(t)+S_{g,yz}R_{zy}(t) & S_{g,zx}R_{xy}(t)+S_{g,zy}R_y(t)+S_{g,z}R_{zy}(t) \\
S_{g,x}R_{xz}(t)+S_{g,xy}R_{yz}(t)+S_{g,xz}R_z(t) & S_{g,yx}R_{xz}(t)+S_{g,y}R_{yz}(t)+S_{g,yz}R_z(t) & S_{g,zx}R_{xz}(t)+S_{g,zy}R_{yz}(t)+S_{g,z}R_z(t)
\end{bmatrix}
$$