UNIVERSITY OF OSLO
Department of Physics

MEMS Inertial Navigation System

Paal Alexander Nerholm

December 19, 2011

Summary

In this thesis a inertial navigation system is in focus where the basis for this system is an Micro-Electro-Mechanical sensor. Micro-Electro-Mechanical sensor are today widely available and affordable providing inertial measurements of various types. Among them are accelerometer, and gyroscopes which are the most important sensors in an inertial navigation system. Modern smart phones have gyroscopes magnetometers accelerometers and GPS sensors integrated providing various interface and navigation features. Being able to utilize these sensors for navigation purposes both for autonomous robots and humans are becoming a reality as MEMS technology improves.

In this thesis the performance of basis navigation algorithm with magnetometer as an aiding sensor is reviewed. First an analysis of mathematical solutions to the rotation matrix is done.This part considers the approaches using quaternions, 321 euler and 9-element rotation matrix. Further more the integration methods, euler forward and Heuns is analysed where the Heuns method combined with the 9-element rotation matrix yielded the lowest error. As a result the Kalman filter is able to reduce the stationary drift from about 400m to about 2m. Though some indications of errors in the implementation were discovered.

When using MEMS inertial sensors the drift results are high and unstable in the long term. Short therm navigation in the seconds region yields acceptable results which points to the necessity of long term stable aiding measurement for position and attitude.

Preface

This thesis is the final work done to achieve a masters degree at the faculty of physics at the University of Oslo. The assignment was given by Devotek AS, i have been sitting approximately one third of my time in Kongsberg and the rest at Kjeller. The assignment was conducted with two supervisors, namely Oddvar Hallingstad and Gunnar Holm. I would like to thank Gunnar for his contribution of the sensor and facilities for mechanical tasks together with many helpful discussions.

Especial gratitude goes to my lecturer and supervisor Oddvar Hallingstad. He has supported my work once every week for half a year, without him the assignment would have been much harder.

Paal Alexander Nerholm
Kjeller, 19 Des 2011

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In modern context, navigation is a term used for finding your way. Every single day we rely on the most basic form of navigation. The most basic type of navigation relies on the observation and recognition of known features or fixed objects in our surroundings and moving between them. These objects may be mountains, buildings, monuments or other land fixed objects that are specific for a geographical location. An extension of this type of navigation is following directions using a map. More advanced forms of navigation have been developed over the centuries. An example is the ancient and well established technique of using the stars as the relative object. Tools such as the marine sextant has been used by navigators to explore the world by sea.

After the introduction of inertial measurements, navigation has become a task for a computer. The laws of classical mechanics as formulated by Sir Isaac Newton are important in inertial navigation systems. Newtons laws describe the relationship between the motion of a body and external forces. Successive mathematical integration of accelerometers produce a measurement of the objects velocity and position, which is the target variables in an INS. Accelerometers usually consist of tree orthogonal measurement axis able to measure acceleration. In order to navigate with respect to the inertial frame the orientation of the accelerometers axis must be known. Gyroscopes measure rotation of the same three axis as the accelerometer and mathematical integration of these measurement yield the desired orientation. Inertial Navigation Systems (INS) are therefore based on accelerations and rotation velocities.

Modern sensor technology provides the means for measuring many environmental phenomena such as, acceleration, magnetic flux, pressure, temperature and rotation velocity. Expensive mechanical sensor platforms has been developed for inertial navigation systems in the aerospace and military community. Micro-Electro-Mechanical-Systems (MEMS) are today an

emerging technology that has the potential for a multitude of uses. These sensors are low cost high volume production units that are easily available. This has introduced a new wave of applications for inertial navigation systems as the automotive industry and hobbyist are able to afford it. There are many internet communities with hobbyist from all over the world developing autonomous quad copters planes and boats. These low cost, mechanical, silicone based sensors are prone to errors due to the manufacturing process. This is a problem that needs to be addressed to be able to use them.

Because of the successive mathematical integration involved in the navigation equations small errors in the measurement cause large errors over time in the position and attitude. This means that the system has a sort-term error stability. Aiding sensors such as magnetometer and GPS are long-term stable and may be used to stabilize the system. Combining these two types of measurements it is possible to achieve long term stability with rapid positioning updates. Because of the availability and low cost of the MEMS sensors and the increasing power available in microcontrollers the task at hand is to review how the basis system performs. Where this system will consist of accelerometers, gyroscopes and magnetometers.

1.1 Problem Description

Kongsberg Devotek performs advanced product development in many business areas and some of the systems to be developed will need - or benefit of usage of an inertial navigation system. Until recently, inertial navigation systems have been reserved sophisticated and high cost products, but is today more actual in low cost solutions as MEMS technology develops, and the accuracy of the available sensors are improving. Objectives of the thesis:

1. Study and describe strap down inertial navigation algorithms
2. Study and describe INS error model (1st order)
3. Study and describe INS error model (1st order)
4. Study and describe performance parameters of the accelerometer and gyroscope of the ADIS 16407 and relate these to inertial sensor error models.
5. Make a prototype IMU by the above identified sensors and perform calibration.
6. Analyse sensor readout and compare with specification

7. implementation of Kalman filter for alignment and possible sensor bias estimation
8. Study and discuss various aiding sensor and principles
9. Implement sensor error models and inertial error model in matlab for analysis of navigation accuracy

Chapter 2

Mathematical Background

In this chapter fundamental mathematical knowledge for this thesis is presented. Most of the material presented in this chapter is found in [6].

2.1 Vectors

A vector is is a mathematical term which describe length and direction. A vector can both be described as geometrical or algebraic. When a vector is algebraic it is represented as a column matrix. Given the tree dimensional coordinate system with basis vectors b_i, the geometric vector can be represented as a linear combination of the basis vectors in the space $\Re^n$:

$$\vec{v} = \sum_{i=1}^{3} v_i \vec{b_i} \tag{2.1}$$

Where the algebraic vector is represented as a column matrix:

$$\vec{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \tag{2.2}$$

2.1.1 Euclidean Norm

Length of a vector is defined as the eucledian norm:

$$||v|| = \sqrt{v_1^2 + v_2^2 + v_3^2} \tag{2.3}$$

2.1.2 Scalar Product

Scalar product is also called the inner product or dot product. The scalar product of two geometric vectors is defined as:

$$< \vec{v}, \vec{u} >= |\vec{v}||\vec{u}|cos\angle\vec{v}\vec{u} \tag{2.4}$$

The scalar product can be used to determine orthogonality, the scalar product becomes zero in that case.

2.1.3 Cross product

In geometrical form the cross product between two vectors are defined as:

$$|\vec{v} \times \vec{u}| = n|\vec{v}||\vec{u}|sin\angle\vec{v}\vec{u} \tag{2.5}$$

Where n is a unit vector perpendicular to the plane that spans out from $\vec{v}$ and $\vec{u}$.

2.1.4 Skew symmetric form

The skew symmetric form is found by calculating the scalar products of:

$$s_{ij}^{p} =< (\omega_1\vec{P}_1 + \omega_2\vec{P}_2 + \omega_3\vec{P}_3) \times \vec{P}_j, \vec{P}_i > \tag{2.6}$$

Where P is the basis vectors that create the cartesian (orthogonal) coordinate system. The skew symmetric form of a vector then becomes:

$$S(\underline{\omega}^P) = \underline{\omega}^P\times = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \; where\; \vec{\omega} = \omega_1\vec{P}_1 + \omega_2\vec{P}_2 + \omega_3\vec{P}_3 \tag{2.7}$$

It is possible to rearragne this to:

$$\underline{a} \times \underline{b} = -\underline{b} \times \underline{a}$$
$$S(\underline{a})\underline{b} = -S(\underline{b})\underline{a}$$

2.2 Euler angles

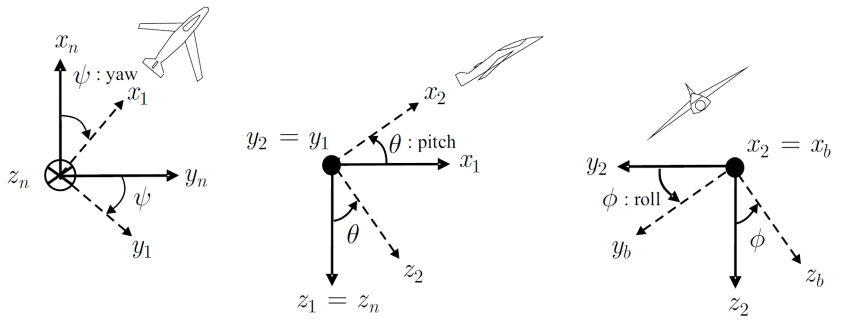


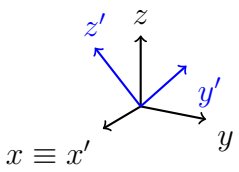
Figure 2.1: Euler Angles [3]

Euler angles are a common way of describing the angle differentiation between two Cartesian reference frames. In this thesis the notation roll pitch yaw will be used. Euler angles gives clear physical interpretation as shown in Figure (2.1).

2.2.1 Simple rotations

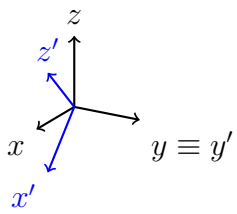
Multiple coordinate frames is essential in the modelling of navigation systems. The most important tool is the direct cosine matrix. A direct cosine matrix can rotate a vector from one frame to another when the angle between the frames is known. There are 3 simple rotations, a simple rotation is a rotation around one axis. Angles are given in radians; ϕ, θ and ψ, where these angles correspond to roll pitch and yaw.

1: Rotation around X (Pitch) Below the rotation matrix that rotates any vector around the x axis is shown.



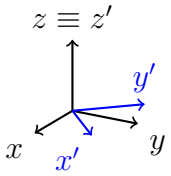
$$R(\phi)_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix}$$

2: Rotation around Y (Roll) Below the rotation matrix that rotates any vector around the y axis is shown.



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

3: Rotation around Z (Yaw) Below the rotation matrix that rotates any vector around the z axis is shown.



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

2.2.2 Rotation matrix derivative

A navigation system is a time variant system. The differential equation describing rotation matrix is given as:

$$\dot{R} = S(\underline{\omega})R \tag{2.8}$$

A direct cosine matrix is an orthogonal matrix:

$$RR^T = I \tag{2.9}$$

The time derivation when using the product rule yields:

$$\dot{R}R^T + R\dot{R}^T = 0 \tag{2.10}$$

$$\dot{R}R^T + (\dot{R}R^T)^T = 0 \tag{2.11}$$

S is then defined as:

$$S = \dot{R}R^T \tag{2.12}$$

S is a skew symmetric matrix because (2.11) is zero:

$$S + S^T = 0; \tag{2.13}$$

This means that there exists a relationship between the derivative and the skew symmetric matrix:

$$R^{-1} = R^T \Leftrightarrow RR^T = I \tag{2.14}$$

$$S = \dot{R}R^T \tag{2.15}$$

$$S = \dot{R}R^{-1} \tag{2.16}$$

$$\dot{R} = SR \tag{2.17}$$

S must then have the form:

$$S(\underline{\omega}) = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{2.18}$$

where ω is rotation velocity. The derivative of a rotation matrix then becomes:

$$\dot{R} = S(\underline{\omega})R \tag{2.19}$$

2.2.3 Euler 321 direct cosine matrix

It is common to reduce the three simple rotation matrices into one direction cosine matrix. There are multiple ways to do this:

$$R_{123} = R_x R_y R_z \tag{2.20}$$
$$R_{132} = R_x R_z R_y \tag{2.21}$$
$$R_{231} = R_y R_z R_x \tag{2.22}$$
$$R_{213} = R_y R_x R_z \tag{2.23}$$
$$R_{312} = R_z R_x R_y \tag{2.24}$$
$$R_{321} = R_z R_y R_x \tag{2.25}$$

There are 6 different orders of multiplication of the simple rotation matrices. Each order of multiplication gives different results in range definitions about the axis. The 321 DCM is the preferred way for an aircraft because the yaw and roll angles are defined from $-180°$ to $180°$ and the pitch angle is defined in the range $-90°$ to $90°$. The aircraft's velocity is defined along the x-axis of the body coordinates system. Since a aircraft seldom has its nose pointing straight up or down is a rational explanation of why just these rage definitions are chosen.

$$R(\psi, \theta, \phi) = R(\psi)R(\theta)R(\phi) \tag{2.26}$$

$$R(\psi, \theta, \phi) = \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\phi + c_\psi s_\theta s_\phi & s_\psi s_\phi + c_\psi c_\phi s_\theta \\ s_\psi c_\theta & c_\psi c_\phi + s_\phi s_\theta s_\psi & -c_\psi s_\phi + s_\theta s_\psi c_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \tag{2.27}$$

where this matrix can be expressed as:

$$R(\psi, \theta, \phi) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.28}$$

2.2.4 Quaternions

In this thesis the properties of quaternion algebra is used to describe attitude between two three dimensional coordinate systems [14]. The advantage of using quaternions is the ability to describe the attitude with four numbers. When comparing this way of describing attitude with euler angles the singularity problem becomes solved. The euler DCM requires trigonometrical functions which the quaternions does not require. This reduces the computational requirements of quaternion DCM compared to the 321 Euler DCM. Special consideration is required as the covariance matrix in a Kalman filter becomes singular when quaternions are implemented as a rotation operator [16]. Quaternions are defined as:

$$\bar{q} = q_4 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} \tag{2.29}$$

Quaternions must satisfy:

$$\mathbf{ii} = \mathbf{jj} = \mathbf{kk} = -1 \tag{2.30}$$
$$-\mathbf{ij} = \mathbf{ji} = \mathbf{k} \tag{2.31}$$
$$-\mathbf{jk} = \mathbf{kj} = \mathbf{i} \tag{2.32}$$
$$-\mathbf{ki} = \mathbf{ik} = \mathbf{j} \tag{2.33}$$

Quaterions can be written as a four dimentional column matrix:

$$\bar{q} = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}^T \tag{2.34}$$

Quaternion multiplication in matrix form is defined as:

$$q \otimes p = \begin{bmatrix} q_4 & q_3 & -q_2 & q_1 \\ -q_3 & q_4 & q_1 & q_2 \\ q_2 & -q_1 & q_4 & q_3 \\ -q_1 & -q_2 & -q_3 & q_4 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} \tag{2.35}$$

The DCM matrix is defined as:

$$C = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 + q_3q_4) & 2(q_1q_3 - q_2q_4) \\ 2(q_1q_2 - q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 + q_1q_4) \\ 2(q_1q_3 + q_2q_4) & 2(q_2q_3 + q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \tag{2.36}$$

2.3 Coordinate Frames

For navigation purposes on earth, multiple reference frames are needed to describe the navigation system. A basic navigation system consists of at least

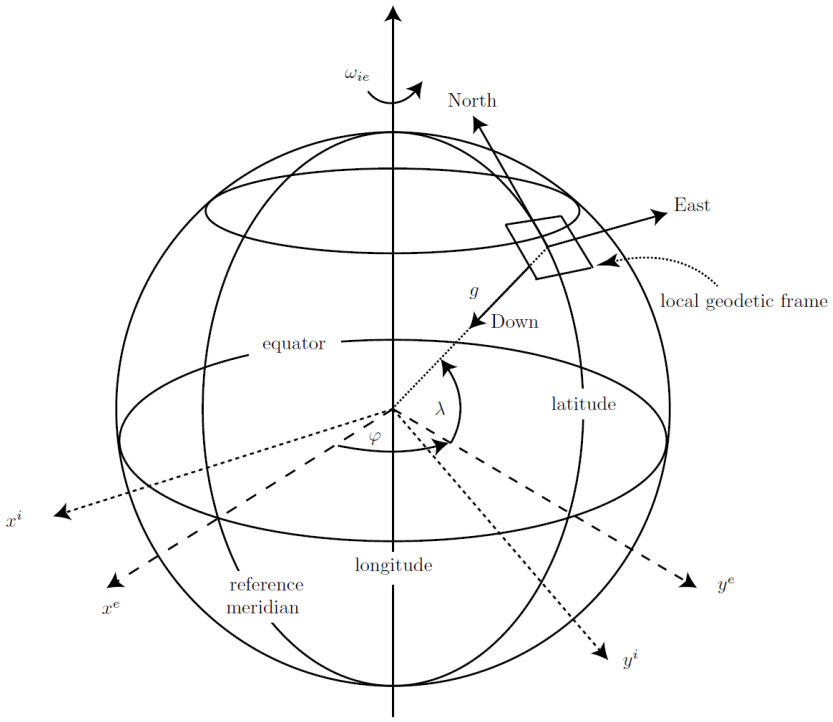


Figure 2.2: Relations between Earth-, NED-, Inertial-, and Body- frame

four frames, earth-, inertial-, navigation-, and body-frame. Accelerometers and gyroscopes measure the acceleration and angular velocity between the inertial- and body-frame. The goal of a navigation system is to describe the position, velocity and attitude of the body frame in respect to the navigation frame. To achieve this the measured accelerations and angular velocities need to be processed in such a way that they describe movement in the navigation frame. In this section reference frames are presented and a final definition of the frames used in this thesis is described.

2.3.1 Earth frame (*e*-frame)

Planet earth is a spherical star in our constellation and to be able to navigate in relation to this star a reference frame attached to earth has to be defined. In [13] this frame is called the Earth-centred earth-fixed frame (ECEF). This coordinate system has its origin at the center of the earth and rotates with the earth. Throughout this thesis e will be the suffix designated to the earth coordinate system. e is placed at the center of earth where the z axis points north and xy axis lays at the equatorial plane. Earth rotates around z axis one time per day, because of this fact x is defined to point toward the

Greenwich median.

2.3.2 Navigation frame (*NED*-frame)

This coordinate system is often defined as the NED or local level frame. *NED* is an abbreviation for **N**orth **E**ast **D**own coordinate system. This reference frame is always tangential to the surface on earth. The D-axis is pointing at the center of e, N-axis is pointing north, E-axis is pointing east. NED corresponds to XYZ when compared to Cartesian suffix. This is the frame which the object of interest is desired to be related. Which means that this frame is the frame one wants to represent velocity and attitude of an object.

2.3.3 Inertial frame (i-frame)

An inertial frame is a coordinate frame in which Newton's laws of motions apply. It is preferable to place this coordinate system's origin in the same origin as the e-frame. The difference between the i- and e-frame then becomes only a simple rotation. This frame should not be confused with the ideal inertial frame since the gravity force applies.

2.3.4 Body frame (b-frame)

The body coordinate system is often defined in the same way as the NED system to make the relation between the two frames simple. The body frame is associated with the platform where the sensors are mounted. The body coordinate system has its origin placed at the center of gravity of the platform it is mounted on. The sensor's internal axis and the platform's axis should be aligned perfectly, if this is not the case a new coordinate system has to be defined to describe the difference between the body and sensor axis.

2.3.5 Simplifications to n-frame

Sensors used in this thesis are not accurate enough to measure the earth's rotation, as a result the earth can be simplified to have a fixed position and attitude in space. More specifically this results in i being fixed in reference to e, thus equal. The scope of the platform developed is to be able to navigate accurately in the minutes time region at non sonic speeds. The curvature of the earth becomes much less than the bias and noise characteristics of the accelerometers. Because of this the the curvature of the earth can be neglected making it flat an non rotating. The reference frames e, *NED* and i can therefore be reduced to one frame. In this thesis the n will be the

navigation frame used as a Cartesian fixed position reference frame. Also this frame is defined as z pointing upwards as opposed to the *NED*, this is also the case for the b-frame. Through out this thesis there will therefore be only two Cartesian frames, namely the body frame and the navigation frame. Gravity will be a positive constant in the n-frame resulting in z pointing upwards.

Chapter 3

Strapdown Inertial Navigations Systems

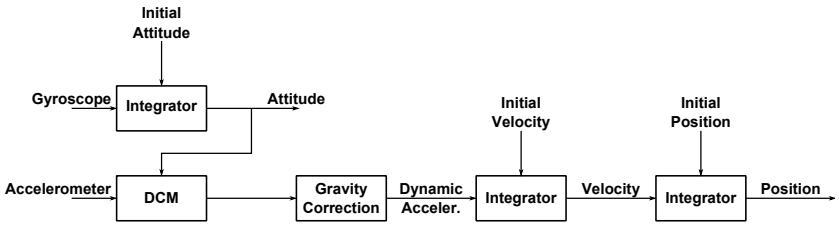


Figure 3.1: Block Diagram of Strapdown Inertial Navigation System

Before strapdown systems were used gimbal mechanical systems were used instead. Where the attitude could be directly read from the position of the internal platform. The internal platform is stabilized with a spinning wheel which in essence uses the gyro forces to stabilise the inner platform.

A strapdown INS system consist of multiple sensors, among them the two most important are the gyroscope and accelerometer. A accelerometers signals correspond to acceleration along the three internal axis of the accelerometer. To translate these accelerations to position a double integration process is all that is needed. This is the case if a representation in the b-frame is the scope. In a INS system the representation on the n-frame is the scope. This requires the knowledge of the Direct Cosine Matrix (DCM) which translates the acceleration vector from the b-frame to the n-frame. The DCM transforming the acceleration vector from the b-frame to the n-frame is found by integrating the gyroscope velocity vector. This results in three differential equations:

$$\dot{\underline{p}}^n = v^n \tag{3.1}$$
$$\dot{\underline{v}}^n = R_b^n \underline{\tilde{f}}^b - g^n \tag{3.2}$$
$$\dot{R}_b^n = R_b^n S(\underline{\tilde{\omega}}_b^{nb}) \tag{3.3}$$

Where $\dot{\underline{p}}^n$ represents position in the n-frame, $\dot{\underline{v}}^n$ represent the velocity in the navigation frame and $\dot{R}_b^n$ represent the transformation matrix between the b-frame and the n-frame. There are multiple ways of calculating this transformation matrix, in this thesis quaternion, 321 euler and 9 element rotation matrix will be reviewed.

3.1 321 Euler angle DCM

The 321 DCM is found by solving the differential equation with respect to euler angles. Euler angles are defined as the angle difference between two Cartesian coordinate systems. Consider two reference frames consisting of the basis vectors n_i and b_i. Roll (ϕ) is defined as the angular difference between n_2 and b_2. Pitch (θ) becomes the angular difference between the basis vectors n_3 and b_3. Yaw (ψ) becomes the difference between the n_1 and b_1. The inertial navigation systems differential equations can be expressed as [5, 6]:

$$\dot{\underline{p}}^n = v^n \tag{3.4}$$
$$\dot{\underline{v}}^n = R_b^n(\Theta_b^n)\tilde{f}^b - g^n \tag{3.5}$$
$$\dot{\underline{\Theta}}_b^n = D(\Theta_b^n)\underline{\tilde{\omega}}_b^{nb} \tag{3.6}$$

where:

$$\Theta_b^n = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T \tag{3.7}$$

Where the 321 rotation matrix is given as [6]:

$$D(\Theta_b^n) = \begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & \frac{s_\phi}{c_\theta} & \frac{c_\phi}{c_\theta} \end{bmatrix} \tag{3.8}$$

The DCM; $R_b^n(\Theta)$ is defined as:

$$R_b^n(\Theta) = \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\phi + c_\psi s_\theta s_\phi & s_\psi s_\phi + c_\psi c_\phi s_\theta \\ s_\psi c_\theta & c_\psi c_\phi + s_\phi s_\theta s_\psi & -c_\psi s_\phi + s_\theta s_\psi c_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \tag{3.9}$$

3.2 Quaternion

Attitude can be expressed in the form of quaternion algebra, this has its advantages as singularities and trigonometrical functions are avoided. The inertial navigation differential equations becomes [14]:

$$\dot{p}^n = v^n \tag{3.10}$$

$$\dot{v}^n = C_b^n \underline{\tilde{f}}^b - g^n \tag{3.11}$$

$$\dot{q}^n = \Omega(\underline{\tilde{\omega}}_b^{nb})q \tag{3.12}$$

Where:

$$\Omega(\underline{\tilde{\omega}}_b^{nb}) = \frac{1}{2}\begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \tag{3.13}$$

The DCM C_b^n is defined as:

$$C_b^n = \begin{bmatrix} (q_1^2 + q_2^2 - q_3^2 - q_4^2) & 2(q_2q_3 - q_1q_4) & 2(q_2q_4 + q_1q_3) \\ 2(q_2q_3 + q_1q_4) & (q_1^2 - q_2^2 + q_3^2 - q_4^2) & 2(q_3q_4 + q_1q_2) \\ 2(q_2q_4 - q_12_2) & 2(q_3q_4 + q_1q_2) & (q_1^2 - q_2^2 - q_3^2 + q_4^2) \end{bmatrix} \tag{3.14}$$

Quaternions has no clear physical interpretation therefore converting them to euler angles are important to be able to analyse the result. This conversion introduces singularities. The conversion is given by:

$$\phi = atan2\left(2(q_0q_1 + q_2q_3, 1 - 2(q_1^2 + q_2^2)\right) \tag{3.15}$$

$$\theta = asin\left(2(q_0q_2 - q_3q_1)\right) \tag{3.16}$$

$$\psi = atan2\left(2(q_0q_3 + q_1q_2), 1 - 2(q_2^2 + q_3^2)\right) \tag{3.17}$$

where these angles are given in radians.

3.3 9 Element matrix

A DCM is in essence a square matrix consisting of 9 elements. As explained, it is possible to translate this linear rotation operator to/from either quaternions or euler angles. In the case of quaternions, multiple multiplications is required. In the euler case, sine and cosine multiplication is required. These

calculation operators reduce the algorithm efficiency. Derivation of the rotation matrix gives an interpretation that can be implemented. As euler and quaternions express rotations they are converted to a DCM. The whole process can be simplified, the inertial navigation differential equations then become:

$$\dot{\underline{p}}^n = v^n \tag{3.18}$$

$$\dot{\underline{v}}^n = R_b^n \underline{\tilde{f}}^b - g^n \tag{3.19}$$

$$\dot{R}_b^n = R_b^n S(\underline{\tilde{\omega}}_b^{nb}) \tag{3.20}$$

where R is a DCM consisting of 9 elements:

$$R_b^n = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{3.21}$$

and skew symmetric form of the angular velocity:

$$S(\underline{\tilde{\omega}}_b^{nb}) = \begin{bmatrix} 0 & -\tilde{\omega}_3 & \tilde{\omega}_2 \\ \tilde{\omega}_3 & 0 & -\tilde{\omega}_1 \\ -\tilde{\omega}_2 & \tilde{\omega}_1 & 0 \end{bmatrix} \tag{3.22}$$

The DCM R_b^n can be converted to euler angles by [6]:

$$\phi = atan2\left(\frac{r_{32}}{cos(\theta)}, \frac{r_{33}}{cos(\theta)}\right) \tag{3.23}$$

$$\theta = asin\left(-r_{31}, \sqrt{(r_{11}^2 + r_{21}^2)}\right) \tag{3.24}$$

$$\psi = atan2\left(\frac{r_{21}}{cos(\theta)}, \frac{r_{q11}}{cos(\theta)}\right) \tag{3.25}$$

3.4 Path generator

To be able to validate the performance of the navigation algorithm, a deterministic solution for a given path is necessary. For this purpose a circular path has been chosen. This is because a circular path is easy to implement and requires high accuracy from the integration routine. If r corresponds to the radius of a circle, the parametric equation is [9]:

$$p_x^n(t) = cos(\omega(t))r \tag{3.26}$$

$$p_y^n(t) = sin(\omega(t))r \tag{3.27}$$

$$p_z^n(t) = 0 \tag{3.28}$$

Velocity equations then becomes a partial derivative of (3.27) with respect to time:

$$v_x^n(t) = -sin(\omega(t))\dot{\omega}(t)r \tag{3.29}$$
$$v_y^n(t) = cos(\omega(t))\dot{\omega}(t)r \tag{3.30}$$
$$v_z^n(t) = 0 \tag{3.31}$$

Acceleration equations are also found by the partial derivatives:

$$a_x^n(t) = -r(\dot{\omega}^2(t)cos(w(t)) + \ddot{\omega}(t)sin(w(t)) \tag{3.32}$$
$$a_y^n(t) = r(-\dot{\omega}^2(t)sin(w(t)) + \ddot{\omega}(t)cos(w(t)) \tag{3.33}$$
$$a_z^n(t) = 0 \tag{3.34}$$

These equations describe the position, velocity and acceleration in the navigation frame and is dependent on the angular position(ω) and the rotation velocity($\dot{\omega}$). Where ω is an time dependent and defines start and stop angular position. If the scope is to simulate one circle iteration of the particle $\omega(t_0)$ would become zero and $\omega(t_{max})$ would become 2π. For one circle iteration the angular position, velocity and acceleration becomes:

$$\omega^n(t) = \frac{2\pi t}{t_{max}} \tag{3.35}$$
$$\dot{\omega}^n(t) = \frac{2\pi}{t_{max}} \tag{3.36}$$
$$\ddot{\omega}^n(t) = 0 \tag{3.37}$$

The input to the INS is acceleration and rotation velocity in the body frame. The path generator described above is represented in the navigation frame. To convert the acceleration to the body frame it is possible to find $R_n^b(t)$ and use this to rotate the a^n.

$$a^b(t) = R_n^b(t) * a^n(t) \tag{3.38}$$

The x-axis of the body frame is defined tangential to the circle which this results in the body y-axis always pointing towards the center of the circle. The dependency of a rotation operator is not wanted because it is subject to review. Therefore the acceleration in the body axis is simplified. As the centripetal force formulas are adequate to describe both the constant force and initial velocity in the body frame.

$$F = ma = \frac{mv^2}{r} = mr\omega^2 \tag{3.39}$$

Rearranging these to acceleration and angular velocity yields:

$$v^b(t_0) = \sqrt{r^2\omega^2} \tag{3.40}$$

$$a^b(t) = \frac{r^2\omega^2}{r} \tag{3.41}$$

The initial conditions then become:

$$p^n(t_0) = \begin{bmatrix} r \\ 0 \\ 0 \end{bmatrix} v^n(t_0) = \begin{bmatrix} 0 \\ \sqrt{r^2\omega^2} \\ 0 \end{bmatrix} R_b^n(t_0) = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.42}$$

The path generator is simplified to:

$$p^n(t) = \begin{bmatrix} cos(\omega(t))r \\ sin(\omega(t))r \\ 0 \end{bmatrix} v^n(t) = \begin{bmatrix} -sin(\omega(t))\dot{\omega}(t)r \\ cos(\omega(t))\dot{\omega}(t)r \\ 0 \end{bmatrix} a^n(t) = \begin{bmatrix} -r\dot{\omega}^2(t)cos(w(t)) \\ -r\dot{\omega}^2(t)sin(w(t)) \\ 0 \end{bmatrix} \tag{3.43}$$

The specific force and rotation velocity in the body frame becomes constant:

$$a^b(t) = \begin{bmatrix} 0 \\ \frac{r^2\omega^2}{r} \\ 0 \end{bmatrix} w_b^{nb}(t) = \begin{bmatrix} 0 \\ 0 \\ \frac{2\pi}{t_{max}} \end{bmatrix} \tag{3.44}$$

3.5 Integration Routines

An integration routine is a tool for solving Ordinary Differential Equations (ODE's). The most basic way of solving such a problem numerically is by using the Euler forward or backward method [2]. Given the ODE:

$$\dot{x} = f(t, x) = x + t \tag{3.45}$$

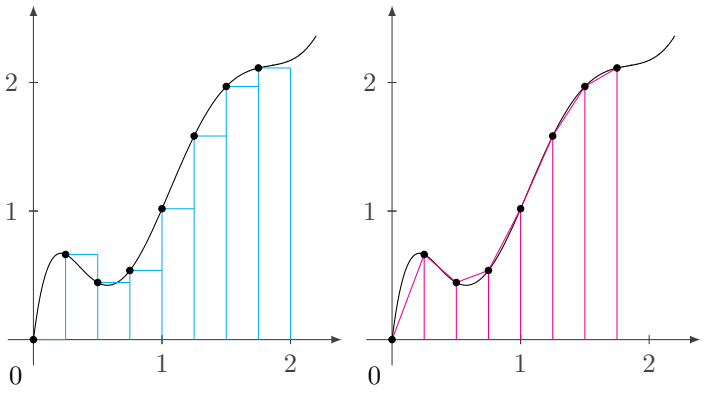
Linearisation of (3.45) is done by the Taylor expansion of the function f, where the time step is given as Ts.

$$x(t_k) = \sum_{i=0}^{\infty} \frac{x}{t!}(x - a)^n \tag{3.46}$$

$$x(t_k) = x(t_{k-1}) + Ts\dot{x}(t_{k-1}) + \frac{1}{2}Ts^2\ddot{x}(t_{k-1}) + ... \tag{3.47}$$

Euler backward method is found by only including the first order approximation in the term (3.47), this yields:

$$x_k = x_{k-1} + Tsf(t_{k-1}, x_{k-1}); \tag{3.48}$$



(a) Euler (b) Heun

Figure 3.2: Euler vs Heun

Euler backwards and forward can then be written as:

$$y_k = y_{k-1} + Tsf(t_{k-1}, y_{k-1}) \tag{3.49}$$

$$y_{k+1} = y_k + Tsf(t_k, y_k) \tag{3.50}$$

Eulers method may also be referred to as the rectangular rule [9] Figure (3.2a) shows a function and the sampling of this function. Euler's method assumes that the function is constant in the time interval. With this assumption the solution is found by the integral of each of the blue boxes. It is clearly illustrated that this results an error, and that decreasing the sampling interval would reduce this error. Another approach is to assume that the original signal can be interpolated with a straight line between the sampling points as shown in Figure (3.2b). This method is called the trapezoid or Heun's method. Which in fact is just an expansion of Euler's method. Heuns [9] method is given by:

$$y'_{k+1} = y_k + Tsf(t, y) \tag{3.51}$$

$$y_{k+1} = y_k + \frac{Ts}{2}(f(t_k, y_k) + f(t_{k+1}, y'_{k+1}) \tag{3.52}$$

In this thesis the Euler and Euler-Heuns integration method will be compared to make a basis on which integration method that is best suited for the INS system. Euler's weakness lies in the fact that the accuracy improves linearly with the step time. This means that a high step time is required to yield acceptable results. Euler-Heun method is a modification to the euler method which result in the accuracy improving quadratically with the step

time compared to Euler. A simple simulation of equation (3.45) shows that Heuns is better compared to its counterpart. Figure (3.3) shows the difference between the analytical solution and the numerical integration routines.

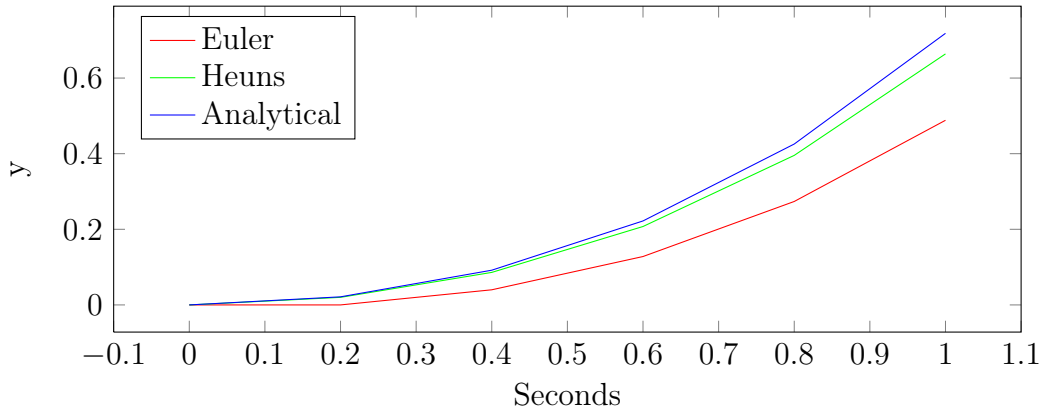


Figure 3.3: Integration Method Comarison Euler, Heun, analytical Solution

3.6 Strapdown Simulation

In this section the analysis of how the different approaches to the strapdown INS equations together with integration routine is reviewed. First Euler's forward is used to solve the strap down differential equations, both for 321 Euler angles, quaternions and 9 element matrix method. After this the Heuns method is used to solve the differential equations. Lastly a comparison between the combinations of methods is reviewed.

In all simulations and plots following in this section 321 euler angles is displayed as red, 9 element matrix is displayed as green, quaternions as blue and analytical as black. All simulations are based on the same angular velocity:

$$\underline{\omega}_b^{nb} = \begin{bmatrix} 0 \\ 0 \\ \frac{\pi}{25} \end{bmatrix} \tag{3.53}$$

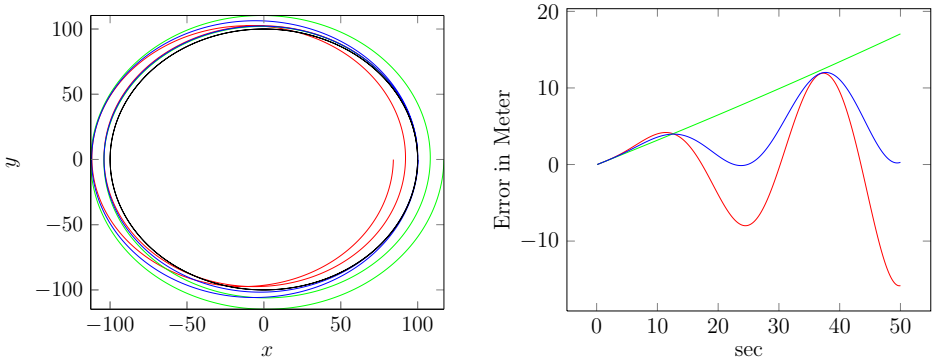
The simulations also have a constant centripital acceleration:

$$\underline{f}^b = \begin{bmatrix} 0 \\ 6.3152 \frac{m}{s^2} \\ 0 \end{bmatrix} \tag{3.54}$$

These constants are result of the path generator previously discussed. The result of these parameters create a perfect circular path. Throughout the

simulations shown in this section the simulation time is held constant at 50 seconds where only the Δ_t (time step) is varied.

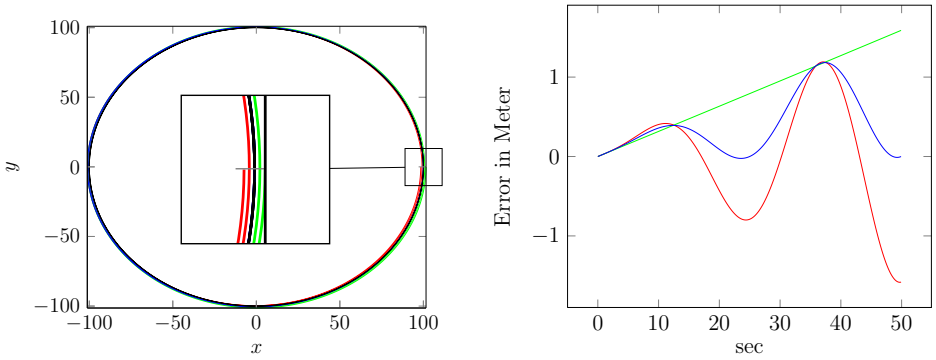
3.6.1 Euler Simulation



(a) Simulation $\Delta_t = 0.1$ (b) Error $\Delta_t = 0.1$

Figure 3.4: Euler Simulation with 10 Hz ($\Delta_t = 0.1$)

In Figure (3.4) two plots are shown, (3.4a) represent the position in the x, y plane. Figure (3.4b) represent the error or deviance from the actual path. All units are in meters and seconds, where the circle radius is 100m. Two sampling intervals are are analysed, namely 10 and 100 Hz. The Euler integration routine is applied to 321-, Quaternion,- and Matrix differential equations in Figure (3.4) with a time step of $\Delta_t = 0.1$. The simulation shows that the euler method has problems finding the correct solution with such a low time step. The error is growing to 10 meters or more in all cases. More precisely the quaternion have a error of 0.04525 meters at 25 seconds while the matrix have 8.177meters, and euler 321 have -7.925 meters. The point mass should at the 25 second mark exactly have one revolution as the angular velocity in the body z axis is $\frac{\pi}{25}$radians. Increasing the sampling rate should increase the accuracy of the simulation. Therefore a new simulation with 100 hz sampling rate is performed. The angular velocity, centripetal acceleration and simulation time is the same as before.

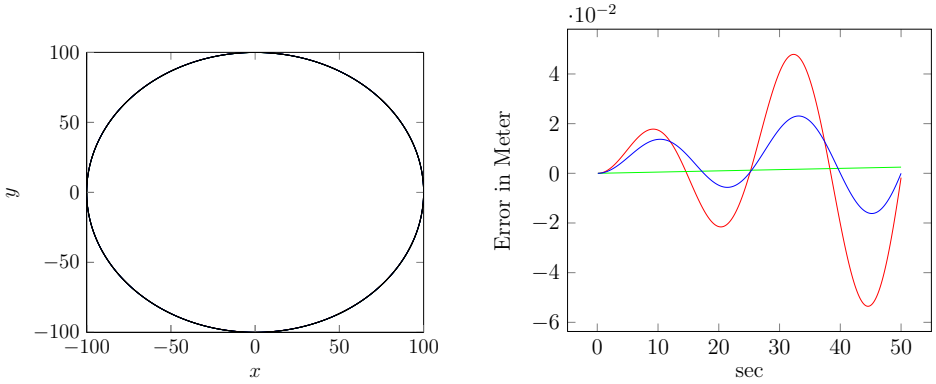


(a) Simulation $\Delta_t = 0.01$ (b) Error $\Delta_t = 0.01$

Figure 3.5: Euler Simulation with 100 Hz ($\Delta_t = 0.01$)

In Figure (3.5) the time step has been increased to 100 Hz. This yields a tenfold improvement in the simulation, which is expected. The error trend is the same as with the 10 Hz simulation. Quaternion is periodically the solution which almost returns the state to the initial position in both of the two revolutions.

3.6.2 Heuns Simaulation

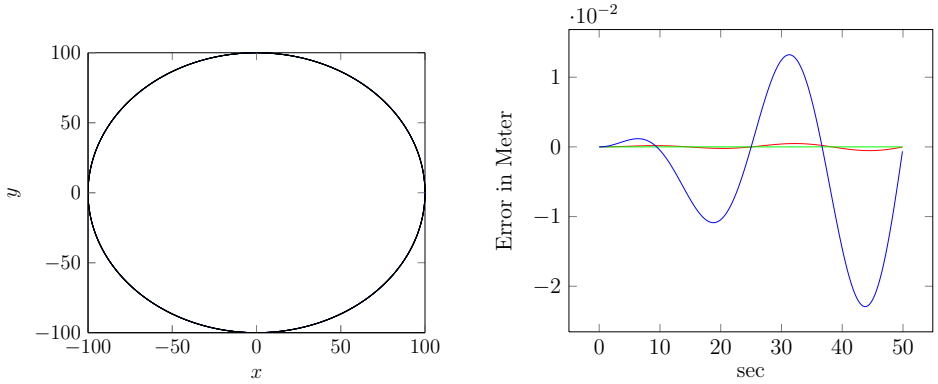


(a) Simulation $\Delta_t = 0.1$ (b) Error $\Delta_t = 0.1$

Figure 3.6: Heuns Simulation with 10 Hz ($\Delta_t = 0.1$)

In Figure (3.6) the changes are dramatic as the simulation proves that Heuns is more precise than the Euler simulation. This is a fact even when the Euler simulation has ten times higher sampling frequency. The error trend

continues to be the same, but is now reduced to the ±4 cm region as opposed to ±10 meters.



(a) Simulation $\Delta_t = 0.01$ (b) Error $\Delta_t = 0.01$

Figure 3.7: Heuns Simulation with 100 Hz ($\Delta_t = 0.01$)

In Figure (3.7) the Heuns integration routine is run at 100Hz. In all the simulations 9-element rotation matrix has had a linear error growth. When Heuns is applied the 9-element method error is the lowest and is seemingly equal to zero. Quaternions is showing odd results as both 321-Euler and 9-element have significanly lower errors in the Heuns 100Hz case.

3.6.3 Methods Analysis

Integration of all the error plots are shown in the table below:

Table 3.1: Performance Summary 10Hz samplings frequency $\Delta_t = 0.1$

	$\|\|e_{321}\|\|$	$\|\|e_{quat}\|\|$	$\|\|e_R\|\|$
Euler	272.113006	205.411400	415.322753
Heun	1.084616	0.474931	0.062191

Table 3.2: Performance Summary 100Hz samplings frequency $\Delta_t = 0.01$

	$\|\|e_{321}\|\|$	$\|\|e_{quat}\|\|$	$\|\|e_R\|\|$
Euler	27.402799	19.739167	39.670262
Heun	0.010847	0.396758	0.000062

In Table (3.2) and (3.1) the error is integrated over the 50 seconds of simulation. This is done by:

$$||e|| = \sum_{i=1}^{n} |e_i| \Delta_t \tag{3.55}$$

Where e is the deviance from the analytical position which is plotted in Figures (3.4b),(3.5b), (3.6b) and (3.7b). The 9-element Rotation Matrix method is has the best overall performance when Heuns at 100Hz is applied. The best numerical methods for solving the system is then the combination of Heuns and 9-Element Rotation matrix. This combination is therefore chosen as the preferred methods in this thesis and will be used in the Linerized Kalman Filter.

Chapter 4

Hardware

In this thesis an Analog Devices ADIS 16407 has been used. This sensor has; accelerometer, gyroscope, magnetometer, barometer and temperature MEMS sensors. Some of these instruments are the basis for the INS system, noise and data acquisition is in focus in this chapter.

4.1 Data Acquisition (DAQ)

DAQ involves measuring signals from a real world physical system and digitizing the measurements for storage, analysis and presentation. In this thesis a Mbed micro-controller (MCU) was used as a digital bridge between a Linux computer and the ADIS 16407 module.

4.1.1 Mbed

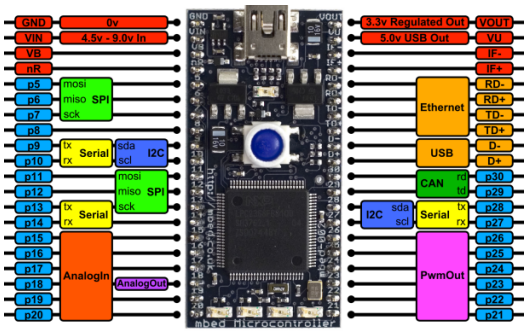


Figure 4.1: Mbed pinout [mbed.org]

Mbed is a micro-controller development board made by Mbed.org. It is a rapid prototyping platform featuring a NXP LPC1768 MCU running at 96

MHz with 512KB FLASH, 32KB RAM. The NXP LPC1768 features a ARM Cortex-M3 core and many peripherals such as; Ethernet, USB Host and Device, CAN, SPI, I2C, ADC, DAC, PWM and other I/O interfaces.

As previously mentioned the Mbed platform acts as a bridge between the computer and ADIS module. The features that is utilised in the Mbed board are the USART and SPI interface. The USART serial bus is tunnelled through a USB interface which provides power to both the ADIS module and the NXP MCU. The SPI interface is utilized to sample data from the ADIS module where as the USART is used to transmit this data to the computer.

4.1.2 ADIS

The ADIS 16407 module embeds multiple micro-electro-mechanical sensors. Because of the mechanical nature of MEMS sensors the actual raw measurements are electrical. This involves the utilization of a analogue to digital converter. Signal conditioning is therefore a necessity and is conveniently embedded into the module. The ADIS module is therefore a complete inertial system capable of measuring everything required in a semi aided INS system.

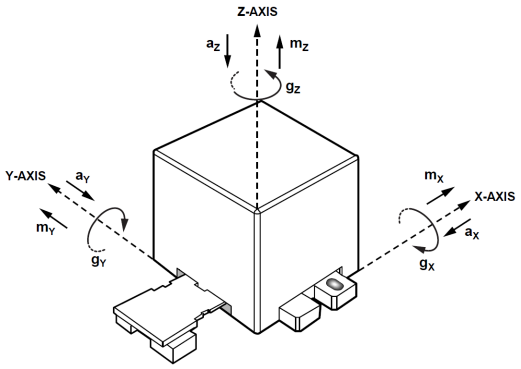


Figure 4.2: ADIS 16407 Sensor axis [1]

Mechanically the ADIS module is a square box with a ribbon cable connected to it. Figure (4.2) illustrates how the MEMS sensors measurements are polarized and the axis systems for each sensor. Mounting the ADIS module requires only two screws and two alignment pins. For precise alignment to a platform the alignment holes provides accurate placement and are essential. The ADIS module is primarily placed in a aluminium box to protect it but also to be able to accurately orientate the sensor in reference to the gravity vector. The aluminium box was accurately milled out on an CNC machine

which drilled out both screw holes and alignment holes. The finished DAQ unit including the alumineum housing is shown in the Figure (4.3).

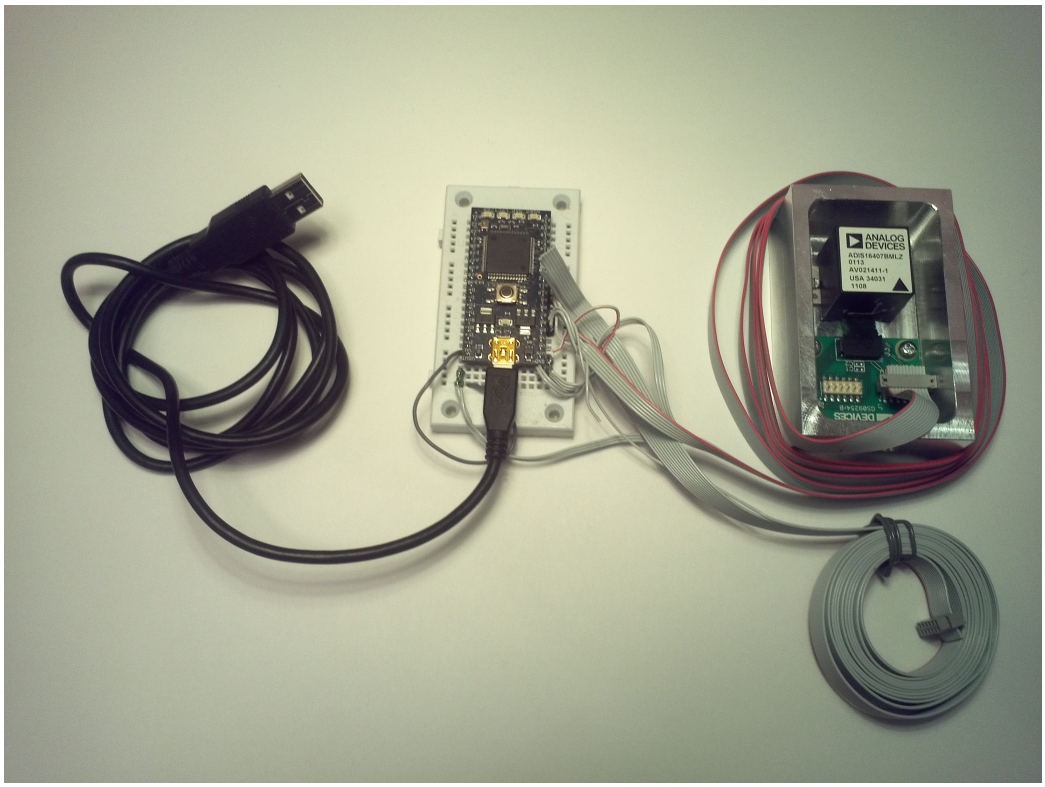


Figure 4.3: ADIS 16407 Sensor and Mbed

4.1.3 Serial Peripheral Interface Bus

SPI is a synchronous serial data link bus used for communication between a peripheral device and a master device. This bus allows multiple slave devices to connect to it by the use of chip select lines. Each slave device on the bus needs one separate IO line which is used by the master to communicate with the selected slave. A description of each pin is given in the Table (4.1).

The c++ library accompanying the Mbed platform is what make it extremely useful in a rapid prototyping process. The code necessary to initialise and use SPI and USART is only 8 lines. Initialisation of the SPI bus is 4 lines:

First MISO MOSI and CLK pins are defined, then the chip select pin is chosen. `format` is a member of the SPI Mbed class is used to define mode and number of bytes per data frame. Mode defines the phase and polarity of the bus clock in reference to the bits on the bus lines. Phase is a boolean

Table 4.1: Abrivations

Abbreviation	Description
MOSI	Master Output, Slave Input
MISO	Master Input, Slave Output
SS	Slave Select
CS	Chip Select
SCLK	Serial Clock
DIN	(Serial)Data in
DOUT	(Serial) Data Out
IRQ	Interrupt request

Listing 4.1 SPI initialization Mbed c++

```
1 /* Init SPI Mbed */
2 SPI ADIS16407(p5, p6, p7); // SPI class, MISO MOSI SCK
3 DigitalOut CS(p8); // Chip select
4 ADIS16407.format(16,3); // 16bit frame Mode 3
5 ADIS16407.frequency(1000000); // SCK freq 1MHz
```

variable defined as `CHPA`, similarly is the polarity variable defined as `CPOL`. A timing diagram of this process is shown in Figure (4.4). The ADIS module data sheet states that the module utilises SPI mode 3 and has a 16 bit data frame.

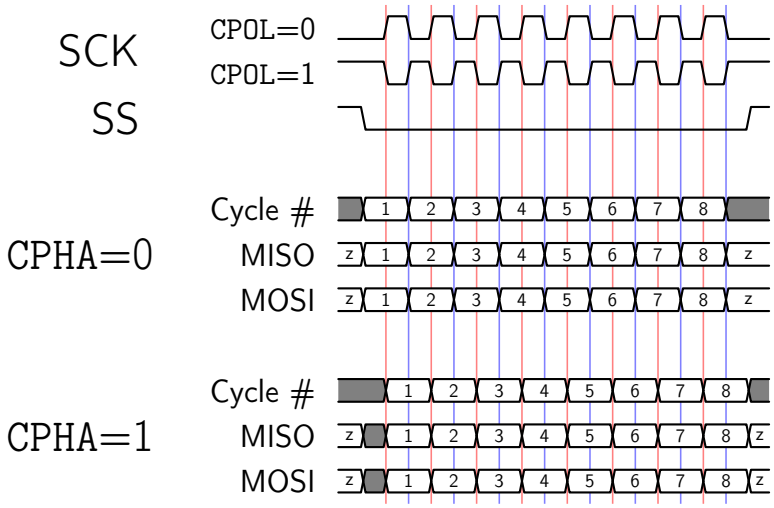


Figure 4.4: SPI Timing Diagram

Table 4.2: Mode selection SPI.format

Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

Table (4.2) illustrates the different modes and the resulting polarities and phases.

The SPI bus is used to connect the MCU and ADIS module together for data Acquisition. Figure (4.5) shows the connection diagram.

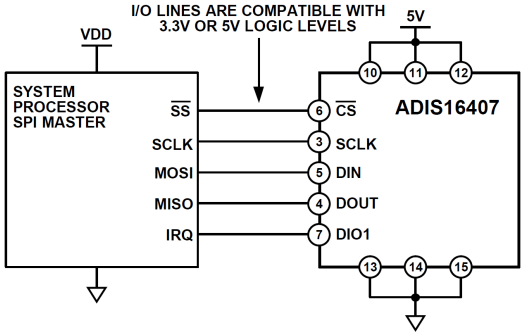


Figure 4.5: SPI Connection Diagram [1]

According to [1] the ADIS module has a maximum serial clock of 1.0MHz when reading data in burst mode. Burst mode is a feature of the ADIS module where only one bit sequence is needed to initiate a transfer of all sensor data, Figure (4.6) shows this progress [1].

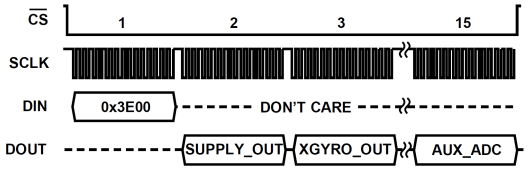


Figure 4.6: Burst read mode [1]

The module also has the option to read a single sensor, i.e. read just the gyroscopes x axis. SPI is a full duplex bus, this means the master and slave always talk simultaneously. I.e. when the master sends a bit sequence out

it simultaneously receives the same amount of bits. If a single axis of the gyroscope sensor is to be read the controller would first request a read of the memory address that contains this value. To receive the actual value a new sequence of bits is sent where the received bits are the gyroscope value. In this thesis the burst read mode has been used. Table (4.3) lists the data received when reading in burst mode.

Table 4.3: Data received burst mode

Number	**Register**	**Address**	**Measurement**
1	SUPPLY_OUT	0x02	Power supply
2	XGYRO_OUT	0x04	Gyroscope, x-axis
3	YGYRO_OUT	0x06	Gyroscope, y-axis
4	ZGYRO_OUT	0x08	Gyroscope, z-axis
5	XACCL_OUT	0x0A	Accelerometer, x-axis
6	YACCL_OUT	0x0C	Accelerometer, y-axis
7	ZACCL_OUT	0x0E	Accelerometer, z-axis
8	XMAGN_OUT	0x10	Magnetometer, x-axis
9	YMAGN_OUT	0x12	Magnetometer, y-axis
10	ZMAGN_OUT	0x14	Magnetometer, z-axis
11	BARO_OUT	0x16	Barometer/pressure, higher
12	BARO_OUTL	0x18	Barometer/pressure, lower
13	TEMP_OUT1	0x1A	Internal temperature
14	AUX_ADC	0x1C	Auxiliary ADC

All the data received are in twos complement from, but the sign bit placement slightly varies depending on which register that is read. Where as the received 16 bit integers have their sign bit misplaced. For the gyroscopes, accelerometers, magnetometer and barometer the two most significant bits are flags that indicate new data and error. Twos complement is the most common method of representing signed integers on computers. A 16 bit integer is able to represent the values in the region: 0 to $2^{16} - 1$ which is 0 to $65,535$, in signed form it becomes: -2^{15} to $2^{15} - 1$ which is $-32,768$ to $32,767$. To be able to represent the actual numerical measurement value the two most significant bits needs to be removed and the 14th bit needs to be moved. This is because the computer treats all 16 bit signed integers in the same way thus making the flags a part of the numerical value which is wrong. Because of this the actual numerical range of the measurements is reduced to 14 bits with yields the new range; $2^{13} - 1$ to -2^{13} which becomes 8191 to -8192. Where the 14th bit represent the plus minus sign.

As mentioned the maximum clock frequency on the SPI bus is 1MHz, this yields:

$$\texttt{SPI}_{\texttt{bps}} = 1\texttt{Mbs} \tag{4.1}$$

$$\texttt{SPI}_{\texttt{KBs}} = \frac{1000000}{8} = 125\texttt{KBs} \tag{4.2}$$

The micro controller is connected to the computer using USB. Through the USB-bus a serial USART bus is tunnelled. The Mbed micro-controller supports a baud rate of 115200bps which yields

$$\texttt{USART}_{\texttt{bps}} = 115,2\texttt{Kbs} \tag{4.3}$$

$$\texttt{USART}_{\texttt{KBs}} = \frac{115200}{8} = 14,4\texttt{KBs} \tag{4.4}$$

This means that 14400 bytes can be transferred per second, taking into account that a single axis measurement from the ADIS module is 16bit, 2bytes, a total of 7200 measurements can be transferred per second. According to Table (4.3) 15 measurements make one data frame which is 30 bytes. The maximum samplings frequency then becomes:

$$f_{max} = \frac{14400}{30} = 480Hz \tag{4.5}$$

This calculation leaves no room for overhead bytes required by a communications protocol with CRC. Implementation has shown that the Mbed USB USART tunnel is robust and a protocol is unnecessary. Raw data transfer of the 30byte data frame is the assumption made in Equation (4.5).

Matlab has no feature to import binary files, but it is capable of importing structured ASCII files. A conversion from binary to ASCII is therefore necessary, this conversion can either be done on the micro-controller or on the computer. An attempt to do the conversion on the micro controller was initially done. This causes problems where ASCII has a much larger bit footprint thus reducing the maximum possible transfer rate. ASCII is a character encoding scheme based in the *American Standard Code for Information Interchange*. ASCII describes digits and characters with 1 byte. In ASCII the number 127 would require 3 bytes, -128 would require 4 bytes, both these numbers can be expressed by a singe raw byte. Transferring ASCII data over the bus would therefore increase the total bit count per second, which again reduces the maximum data transfer rate. Transferring raw binary data through the USB USART tunnel is therefore more efficient. This method of transfer requires a conversion on the computer side, a C++ program were therefore written to convert the binary files to ASCII for Matlab integration.

As previously mentioned methods for signal conditioning is embedded into the ADIS module. This includes a hardware low pass filter with a cutoff at 330Hz. Since the internal clock of the ADC in the ADIS module is 819.2 Hz and the cut off frequency needs to satisfy the Nyquist theorem it has to be lower than:

$$\text{LPF}_{\text{cutoff}} <= \frac{819.2}{2} = 409.6 \tag{4.6}$$

A cut off frequency of 409.6 is the highest possible to be able to reproduce the frequency in the sampled signal and avoid aliasing. 330Hz is then a more suitable cutoff as the signal is adequately sampled to reproduce both shape and frequency.

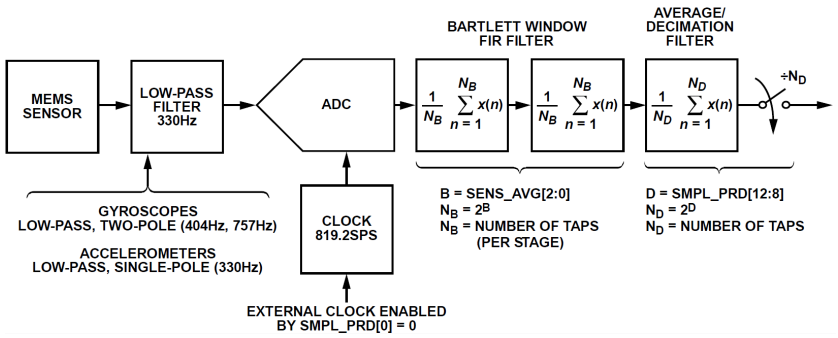


Figure 4.7: Signal conditioning ADIS

Figure (4.7) is a block diagram of the internal signal condition functions embedded into the ADIS module. As previously mentioned the maximum sampling frequency which is possible over the USB USART tunnel is 480Hz. Therefore it is impossible to avoid utilizing the internal average/Bartlett-window functions as aliasing would occur. Physical test have shown that Windows is not capable of a continuous $100Hz * 30Byte$ over a longer period of time. Linux has proven more fruitful as $200Hz * 30Byte$ is possible over a long period of time. As a general rule of thumb problems arise when the theoretical maximum transfer rate is approached. Thus keeping the bus flow at $\frac{200}{480} = 41.6\%$ leaves room for the buffer at the computer side to not overflow. The average decimation filter is therefore utilized. According to the datasheet, the average decimation filter output frequency is devided by a factor of 2 which yields:

$$\text{AVGfreq} = \frac{819,2}{2^x} \tag{4.7}$$

Where x is a user provided integer, this was set to be 2:

$$\text{AVGfreq} = \frac{819,2}{2^2} = 204,8Hz \tag{4.8}$$

With this setting the module can be sampled at 200 Hz by the Mbed controller as the 5Hz difference is neglect-able.

4.1.4 Data files

The data files have the order and unit as displayed in Table(4.4).

Table 4.4: Raw Logg Data Order

Number	Description	Unit
1	Time	Milli seconds
2	Module Power	volt
3	Gyro X	Degrees pr Sec
4	Gyro Y	Degrees pr Sec
5	Gyro Z	Degrees pr Sec
6	Accelerometer X	meter pr sec^2
7	Accelerometer Y	meter pr sec^2
8	Accelerometer Z	meter pr sec^2
9	Magnetometer x	Gauss
10	Magnetometer x	Gauss
11	Magnetometer x	Gauss
12	Barometer High	Bar
13	Barometer Low	Bar
14	Temperature	Celsius
15	ADIS External ADC	Volt

Tree datasets were captured and is listed in Table (4.5)

Table 4.5: Logged datasets

File Name	Duration	Description
DriftTest.mat	87.25 sec	Sensor is placed on the ground and not moved
MagSphere.mat	67.79 sec	Sensor is moved in a spiral upwards
7nov3.mat	20803.95 sec	Sensor is placed on the ground and not moved
AnglesTest.mat	72.865 sec	All axis experience $\pm 9.81 \frac{m}{s^2}$

4.2 Software

In this section the hardware and data acquisition C++ code is in focus. The configuration and initialization of the ADIS module and the timers interrupts is covered here.

4.2.1 Micro-controller

As described in the DAQ section only eight lines of code is necessary for the initialisation and use of both UASART and SPI. This illustrates the rapid prototyping nature of the Mbed library. This is beneficial is because the user is separated from the internal workings of the Cortex-M3 and vendor specific registers which reduces development time. In this section the Mbed firmware code will be described. First the Mbed library is included:

```
1  #include "mbed.h" // Mbed Library Header
```

Then the definition of io pins are done:

```
2  SPI ADIS16407(p5, p6, p7); // MISO MOSI SCK
3  DigitalOut SS(p8); // Slave Select
4  DigitalOut RST(p9); // ADIS Reset pin
5  Serial pc(USBTX, USBRX); // USB USART tunnel
6  DigitalOut L1(LED1); // Debug Led
7  DigitalOut L2(LED1); // Debug Led
```

To be able to sample the sensor in a timed interval a timer interrupt object is created. To be able to confirm the sample rate in Matlab a time-stamp object it also created.

```
8  Ticker timerINT; // Ticker timer interrupt obj
9  Timer timestamp; // Timer timestamp obj
```

Since the USART bus uses 8 bit data frames and the SPI bus uses 16 bit data frames a conversion from `int16` into two `int8` is necessary.

```
10 union ByteSplit{
11     int16_t int16;
12     int8_t int8[2];
13 };
```

Writing to the USART bus is done in a function where the 16 to 8 bit array conversion is done. A `ByteSplit` object called `split` is created. The actual conversion is done when writing to the `union` member `int16`. The 8 bit array is then sent over the USART bus by the `Serial` class member `putc()`.

```
14 void WriteSerialINT(int16_t temp){
15         ByteSplit split; // Create ByteSplit object
16         split.int16 = temp; // Insert temp into int16
17         pc.putc(split.int8[1]); // Write MSBs of int16 to USART
18         pc.putc(split.int8[0]); // Write LSBs of int16 to USART
19 }
```

Sampling data from the SPI bus is done by the function `ReadData`. First slave select (`SS`) is set low to "select" the ADIS module. The boolean flag `Reading` is making sure that multiple read sequences is not executed simultaneously (error detection). `L1` is one of the blue leds on the mbed, this led changes state every time a frame is read. The time stamp is written to USART with ms accuracy. `.write()` is a member of the `SPI` class and is responsible for the full duplex transmission. A for loop cycles 14 times to receive one sample from each sensor on the ADIS device.

```
20 bool Reading=0;
21 void ReadData() {
22     if (!Reading) { // Error detection
23         Reading=1; // Detection Flag
24         SS=0; // Slave Select (Active Low)
25         L1= !L1; // Toggle LED1
26         WriteSerialINT(timestamp.read_ms()); // Timestamp
27         ADIS16407.write(0x4200); // Initiate ADIS Burst Read
28         for (int i=1; i<=14; i++){ // Read ADIS
29             tmp=ADIS16407.write(0x0000);
30             WriteSerialINT(tmp); // Transmit Measurement data
31         }
32         SS=1; // Realease Slave
33         Reading=0; // Ready for new Sample frame
34     }
35 }
```

The ADIS module average decimation filter need to be set. The ADIS module has several registers, `SMPL_PRD` is the one controlling the decimation filter. The base address of this register is `0x3A`, since the object is to write to this address the MSB is set high wich yields: `0xBA`. Writing to the Decimation rate is the bits 8:12 meaning the address is over 2 bytes which yields: `0xBB`. Setting the decimation rate then becomes writing `0xBB` and the value for the register `0x02` to the SPI bus.

```
36 void ConfADIS() {
37     SS=0; // Slave Select (Active Low)
38     wait_us(1); // Waiting for device receive SS
39     ADIS16407.write(0xBB02); // 204.8Hz decimation
40     SS=1; // Release Salve
41     wait_ms(1); // Wait for ADIS process
42 }
```

General initialisation of classes and method is done in the function `Initalize`. A sequence of 15 `0xFF00` is written to the USB USART bus. This marks the start of the file, this is necessary because the computers USB USART buffer may not be empty upon start of the logging process. The program HxD was used to remove unwanted buffers from the binary file.

```
42 void Initalize() {
43     pc.baud(115200); // Set baud rate
44     ADIS16407.format(16,3); // Set SPI bus with and Mode
45     ADIS16407.frequency(1000000); // Set SPI bus freq
46 
47     SS=1; // Initial Slave select (Active Low)
48     RST=1; // Reset (Active Low)
49 
50     // Writing 0xFF00 to PC for start of transmission mark
51     for(int j=0;j<15;j++){
52         WriteSerialINT(0xFF00);
53         }
54     ConfADIS(); // Setup ADIS
55 
56     // Timer interrupt setup
57     timerINT.attach(&ReadData,0.005);    // Ts=0.005->200Hz
58 
59     // Timestamp reset and start
60     timestamp.reset(); // Reset timer
61     timestamp.start(); // Start timer
62 }
```

The `main` function is only responsible for executing the initialisation function and running no-operations while waiting for next timer interrupt.

```
63 int main() {
64     Initalize(); // Run init
65     while (1); // noop on free time =)
66 }
```

4.2.2 Computer

To receive the samples from the USB USART tunnel, logically the USB cable is attached to a computer. In the case of this thesis a Linux operating system was utilized. Linux, regardless of distribution, ships with `cat`. `cat` is a command line function that enables the user to concatenate files, or standard input, to standard output. Where as standard input may be the USB tunnel and standard output may be a binary file. When connected the Mbed USB USART tunnel is identified with the standard input `/dev/ttyACM0`. USART has some configuration options such as; baud rate, stop bit, data bit(frame), hardware/software flow control and parity.

```
stty -F /dev/ttyACM0 cs8 115200 ignbrk -brkint -imaxbel -opost
     -onlcr -isig -icanon -iexten -echo -echoe -echok -echoctl
     -echoke noflsh -ixon -crtscts
```

Using cat to save binary files is done by writing:

```
cat /dev/ttyACM0 > data.bin
```

This results in a binary file saved on the harddrive, this binary file need to be converted to ASCII for Matlab integration. `mingw`, minimal GNU for windows was utilized to compile the program for Windows, similarly GNU can be used to compile the same code on Linux. This is because the use of windows specific headers are avoided:

```
1  #include <iostream> //Standard Input / Output Streams Library
2  #include <fstream> // Input/output file stream class
```

`std` is a namespace and needs to be defined to avoid writing `std::` in front of all standard functions such as cout cin ect. A pointer called `memblock` is created to store the read binary file. The conversion from singed integers to float are defined in a float array.

```
2  using namespace std; // Namespace declaration
3  ifstream::pos_type size; // Type def for file pos & buffer (fstream)
4  char * memblock; // Declare char array pointer
5  float convert[]={0.002418, // Voltage
6                   0.05, 0.05, 0.05, // Gyro
7                   0.0326673, 0.0326673, 0.0326673, // Acc
8                   0.0005, 0.0005, 0.0005, // Mag
9                   0.00008, 0.0000003125, // Baro H+L
10                  0.136, 0.0008059}; //etc
```

`ByteSplit` has the same task here as in the MCU.

```
11 union ByteSplit{
12     int16_t int16;
13     int8_t int8[2];
14 };
```

`Main` function holds the full functionality of the conversion algorithm. First the binary file is read from the disk with the binary specific read functionality embedded in c++. The binary file is placed in memory with the `memblock` pointer.

```
15 int main (int argc, char *argv[]) {
16
17         // Define input, read mode, pointer at end
18         ifstream inFile ("data.bin", ios::in|ios::binary|ios::ate);
19
20         if (inFile.is_open()) // Make shure file is open
21         {
22                 size = inFile.tellg(); // Get size
23                 memblock = new char [size]; // Allocate char array
24                 inFile.seekg (0, ios::beg); // Set pointer pos at beginning
25                 inFile.read (memblock, size); // Bin file -> memory
26                 inFile.close(); // Close file
27         } else {
28                 cout << "Unable to open file";
29                 return 1;
30         }
```

A output file is then created to be able to write the converted binary data to a ASCII text file. C++ has embedded functionality to write in ASCII to file, `fopen fprint` is utilised for this purpose. The imported binary file has a "start of file" frame with 15 `0xFF00`'s, this makes the starting position 30. `tellg()` returns a size with the type `ifstream::pos_type` and needs to be typecasted to integer to be able to use it.

```
40         // Output file
41         FILE * OutFile; // Dype def FILE pointer
42         OutFile = fopen ("data.txt","w"); // Open/create data.txt
43
44         // Conversion algorithem
45         int filesize = (int)size; // Typecast to int
46         int start = 30; // Start of file
47         int l = 0; // Conversion list counter
48         ByteSplit splitter; // Byte split object
49         int16_t temp; // Temporary storage
```

The first `for` loop starts at the binary files 30th bit, the backwards conversion to a signed integer is done with the splitter object. Since the first 2 bits are the power supply there is not necessary to convert to a sing. According to the data sheet the 4 most significant bits are to be neglected which results in the bitwise `AND` with `0x0FFF`. The next `for` loop cycles trough the rest of the 30 bit data frame. Each bulk of 2 bits are converted to singed integers, these integers also have their sign bit misplaced. The bitwise `AND` with `0x3FFF` ensures the two most significant bits are removed. The `if` statement checks if the sing bit is present and moves it if necessary. Additionally the the barometer(k=20) and low barometer(k=22) measurements are defined as unsigned integers, meaning a sing movement is unnecessary. Lastly

the temperature measurement is a singed integer with the 4 most significant bits defined as flags or not used. The `else if` statement handles both the masking and conversion of this.

```
40          for(int j=start; j<filesize; j=j+30){
41                  if(start+j>filesize) break; // Break if no data
42                  splitter.int8[1]=(unsigned char)memblock[(j)];
43                  splitter.int8[0]=(unsigned char)memblock[(j+1)];
44
45                  fprintf(OutFile, "%f\t",(float)(splitter.int16&0x0FFF)); // Print
46                  l = 0; // Reset counter
47                  for(int k=2; k<=28; k=k+2){
48                          splitter.int8[1]=(unsigned char)memblock[(j+k)];
49                          splitter.int8[0]=(unsigned char)memblock[(j+k+1)];
50                          temp=splitter.int16&0x3FFF; // AND mask
51                          if ( ((temp&0x2000)>>13) && (k!=20) && (k!=22) && (k!=24)) {
52                  temp=(temp-16383); // Sing conversion
53              } else if(k==24){
54                      temp=(temp&0x0FFF)-4095; // Sign conversion
55              }
56              fprintf(OutFile, "%f\t", convert[l]*(float)(temp)); //Print
57              l++; // Count
58          }
59          fprintf(OutFile, "\r\n"); // Add carriage return and newline
60          }
61          delete[] memblock; // Cleanup
62          fclose(OutFile); // Release file
63          return 0;
64  }
```

Chapter 5

Sensors

MEMS sensors are relatively low cost but suffer from significant errors, compared to its mechanical brothers. In this chapter the MEMS sensors and their characteristics is covered. Some general terms are introduced and important parameters highlighted.

5.1 Noise

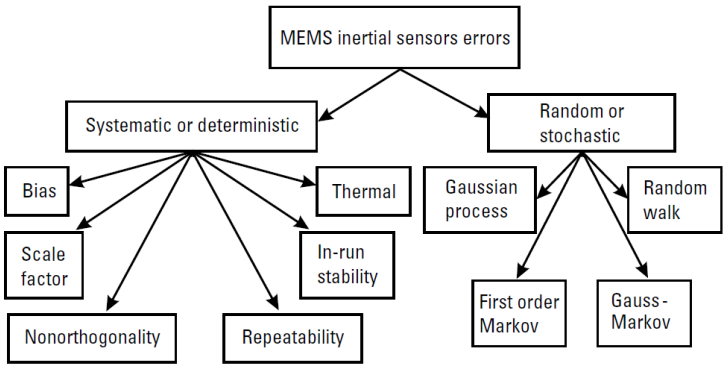


Figure 5.1: Noise Errors Classifications [15]

Noise is a random varying or constant part of a measured signal, it distorts the true value and is the main problem in many engineering tasks. As such, the INS system is vulnerable to noise. This is because the system relies on single and double integration of a noisy measurement. The Gyroscope measurements are integrated once which results in attitude. Accelerometers are rotated by the integrated Gyroscope measurements and then integrated twice to find the position. This process sums up all the noise which often results

in drift (a constant bias results in a constant drift). It is therefore important to find and eliminate as many of these noise components as possible. Figure (5.1) divides MEMS error sources into two groups, namely deterministic ans stochastic. Systematic errors come from defects in the manufacturing process, these types of errors can be calibrated. Calibration of a sensor is based on the specific module in hand, meaning this process is repeated for each module. Stochastic errors are random and may be caused by electrical interference such as magnetic forces.

5.1.1 Aliasing:

Consider a analogue signal consisting of a sine wave with a frequency of 10 Hz. Sampling this signal with a ADC which has a sampling frequency of 10 Hz would result in a constant measurement. This is because every sample would be taken when the continuous signal periodically is equal to itself. Nyquist's theorem states that for a signal to be reconstructed it needs to be sampled with atleast twice the bandwidth of the signal. An analogue signal has no definite sampling frequency so a direct sampling of this signal would cause aliasing. A solution to this problem is low pass filtering a signal before the ADC conversion takes place.

5.1.2 Non-orthogonality

Non-orthogonality is the fault associated with sensors axis misalignment. A perfect orthogonal axis system is difficult to manufacture. Non-orthogonality can be measured using a rate table and can therefore also be compensated for. The factor can be calculated by analysing what the sensor should show and what it actually shown in a rate table test.

5.1.3 Bias

There are multiple types of bias, where the term comes from the field of statistics. A statistic is biased if it is calculated in such a way that it is systematically different from the population parameter of interest. When no external forces are applied to the accelerometer or gyroscope the measurement should be zero, excluding the gravity on the accelerometer. If the average of the gyroscope or accelerometer over time is non zero while no forces are applied is called bias. The ADIS data sheet defines a list of multiple biases; *initial bias*, *In-Run Bias*, *Temperature Bias*, *Voltage Bias*. These biases can be calculated and modelled accurately and thus be removed.

Since INS system relies solely on accelerometers and gyroscopes it is possible to calculate the position error over time with respect to the biases. Disregarding the attitude the position error can be calculated as [17]:

$$p_{\text{error}} = \frac{1}{2} a_\beta t^2 \tag{5.1}$$

The ADIS data sheed listst the initial bias as ±50mg which is $\pm 0.49 \frac{m}{s^2}$, using the Equation (5.1) the drift along one axis can be calculated. For a duration of 10 seconds with a constant bias this yields:

$$p_{\text{error}} \approx \frac{1}{2} \cdot \pm 0.49 \frac{m}{s^2} \cdot 10^2 = \pm 24.5\text{m} \tag{5.2}$$

Over a 1 minute interval the error becomes:

$$p_{\text{error}} \approx \frac{1}{2} \cdot \pm 0.49 \frac{m}{s^2} \cdot 60^2 = \pm 882\text{m} \tag{5.3}$$

An error in attitude will affect the position calculations described above. As a bias in the gyroscope will introduce an angle error in the attitude, this error can be described as:

$$\theta = \omega_\beta t \tag{5.4}$$

Considering this errors impact on the position as it will yield a projection of the acceleration vector in the wrong direction [15]. Where:

$$a_{\text{projection}} = a_\beta sin(\Theta) \approx a_\beta \Theta = a_\beta \omega_\beta \tag{5.5}$$

this yields

$$p_{\text{error}} = \frac{1}{6} \omega_\beta a_\beta t^3 \tag{5.6}$$

The ADIS datasheet lists initial bias error to be $\pm 3 \frac{deg}{sec}$ which yields:

$$p_{\text{error}} = \frac{1}{6} \cdot \pm 3 \frac{deg}{s} \cdot \frac{\pi}{180} \cdot \pm 0.49 \frac{m}{s^2} \cdot 10^3 = \pm 4.27m \tag{5.7}$$

Similarly in the 60 seconds time frame:

$$p_{\text{error}} = \frac{1}{6} \cdot \pm 3 \frac{deg}{s} \cdot \frac{\pi}{180} \cdot \pm 0.49 \frac{m}{s^2} \cdot 60^3 = \pm 923.62m \tag{5.8}$$

The where this error is only the projected part of the error meaning a sumation of the error based on the accelerometer error and the error based on the gyroscope bias must be summarized:

$$10 \text{ sec} : p_{\text{error}} = 24.5 + 4.27 = 28.77m \tag{5.9}$$

$$60 \text{ sec} : p_{\text{error}} = 882 + 923.62 = 1805.62m \tag{5.10}$$

$$\tag{5.11}$$

These calculations indicate that the gyroscope bias introduce cubic errors where as the accelerometer bias introduce quadratic errors. As time progresses the bias from the gyroscope is decisive when compared to the impact of the accelerometer bias.

Initial bias is a bias type that is present from the device is switched on. It changes between each power cycle and stays constant while the unit is powered. This bias is equal to the mean of an axis over a time period of time where there are no external forces applied.

In-Run Bias defines the stability of the bias over time, this is typically a stochastic process.

Temperature Bias is the effect the temperature has on the measurements as the temperature scales the measurements. Experiments with the sensor is confined to a office the temperature bias is not modelled because of the relatively constant temperature.

Voltage Bias is the effect voltage variations has on the measurements. This will not be modelled as the voltage provided by the computer is stable.

5.1.4 Allan Variance

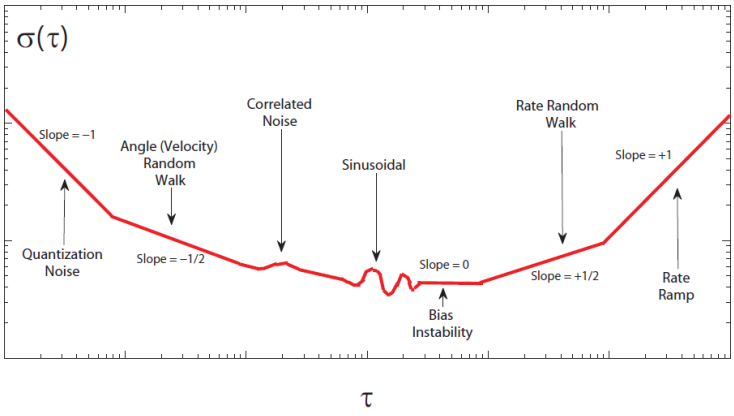


Figure 5.2: Allan Variance Classifications[IEEE Std.952-1997]

Allan Variance is a method that was intended to analyse the stability of an oscillator. The application of AVAR is not just limited to oscillators, it can also be used to analyse the stochastic variables in Gyroscope and

Accelerometer measurements. From the AVAR Quantisation Noise, Random Walk, Bias Instability, Rate Random Walk, Rate Ramp, Correlated Noise and Sinusoidal noise can be found.

Figure (5.2) shows the different variables that is possible to identify with AVAR. According to [15] the process noise matrix Q can be found by AVAR, where Angular Random Walk (ARW) and Velocity Random Walk (VRW) are of interest. In this thesis a matlab allan variance library has been used, the library is based on [12], the library can be found at [8]. From the AVAR plot at $\tau = 1$ both the ARW and VRW can be directly read. The ARW and VRW is used as parameters in the process noise matrix Q in the Kalman filter. Where bias instability is used in the measurement noise matrix R to describe the noise on the bias states.

5.2 Accelerometer

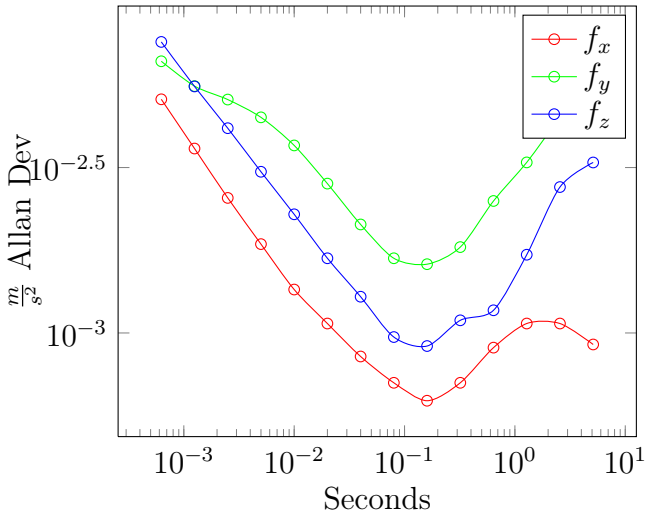


Figure 5.3: Accelerometer AVAR plot

Figure (5.3) shows the allan variance plot for the ADIS accelerometers. The AVAR plot is based on a time series where the sensor is left running in a fixed positon for 6 hours. The sampling frequency was set to 200Hz. The velocity random walk and the bias instability results are shown int the Table (5.1).

Table 5.1: Accelerometer Stochastic variables

	X-axis	Y-axis	Z-axis	Datasheet	Unit
Bias Instab.	0.00062394	0.0016138	0.00091252	0.001962	$\frac{m}{s^2}$
VRW	0.0018564	0.0044847	0.0030709	0.003333	$\frac{m}{s}$

Allan variance confirms that the sampling of the accelerometer yields results that correlate with the datasheet. These parameters are used to construct the process noise matrix in the Kalman filter.

5.3 Gyroscope

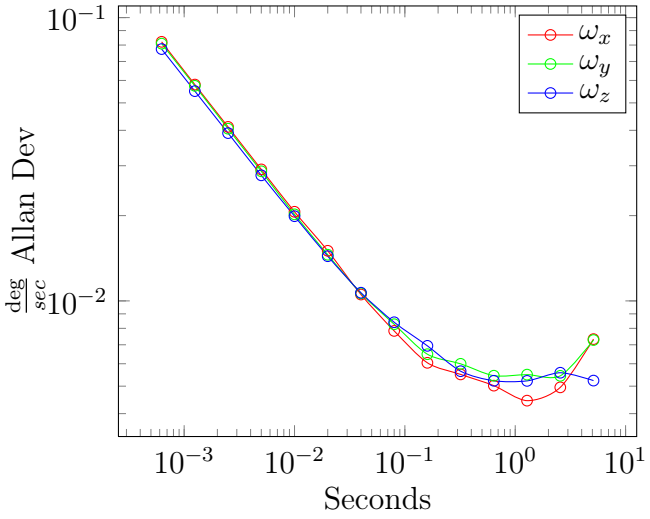


Figure 5.4: Gyroscope AVAR plot

The same data set was used to calculate the gyroscopes AVAR plot as the one previously used for the accelerometers. Figure (5.4) shows the AVAR plot where the results are shown in Table (5.3)

Table 5.2: Gyroscope Stochastic variables

	X-axis	Y-axis	Z-axis	Datasheet	Unit
Bias Instab.	0.0044451	0.0054394	0.0052201	0.007	$\frac{deg}{s}$
ARW	0.029117	0.028674	0.027748	0.316666	$\frac{deg}{s}$

5.3.1 Magnetometer

An apex plot of the magnetometer gives an indication on errors in the measurements. A test where the magnetometer was rotated in many directions was conducted manually. If this rotation is done by a machine the result should be a perfect sphere. Figure (5.5) shows the manual execution of this test.

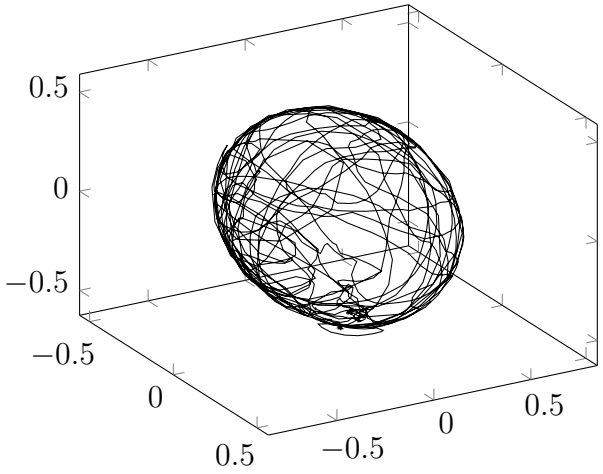


Figure 5.5: Magnetometer Apex

The figure illustrates the apex plotted from the measurement data. The assumption that the electro-magnetic force is constant are made. This assumption result in an expectation of the plot being a sphere. Manual analysis of the plot shows that the magnetometer reading are relatively accurate as the form is spherical. Further analysis can be done where the spheres should be centred at 0, the outer rim on all sides should therefore have the same value.

Table 5.3: Magnetometer Calibration

	max	min	Bias	Unit
X-Axis	0.4925	-0.4830	0.0048	Gauss
Y-Axis	0.4785	-0.4915	-0.0065	Gauss
Z-Axis	0.4895	-0.5245	-0.0175	Gauss

Manual rotation of the device affects this process where the maximum and minimum points may not be reached. This method gives a indication of

big errors and is a simple method that verifies a proper functioning magnetometer. This method is more applicable if a rate table is used.

Chapter 6

Kalman Filter

The Kalman filter was introduced by Rudolf E. Kálmán in the 1960's. The Kalman is a set of mathematical equations that provides the means to estimate the state of a process based on discrete data influenced by white noise [4, 18]. The Kalman filter minimizes the mean of the squared error. It has multiple uses as it can be used to estimate past and present states, prediction of future states is also possible. In other words, the Kalman filter is an optimal state estimator which is based on knowledge of the process and sensors. Knowledge of the measurement and process noise is modelled in the set of ordinary differential equations that describe the process. When Rudolf E. Kálmán introduced the kalman filter it was men to be used on linear systems, the INS equations are non-linear. There are multiple adaptations of the Kalman filter that are meant to be used on an non-linear system. Among them are the Unscented, Extended and Linearised filters. In this thesis the linearised is used.

6.1 Linearised Kalman Filter

As the kalman filter in essence is a linear estimator the non-linear states of the INS system need to be linearised. The LKF linearise around a nominal trajectory. Where the nominal trajectory is the integrated INS system based on accelerometer and gyroscope measurements. The nominal trajectory is assumed to be close to the actual trajectory. [7] lists the continuous and discrete linearised Kalman filter equations:

Algorithm 1 Linearised Kalman Filter Standard Equations

Measurement update:

$\delta \underline{z}_k = \underline{z}_k - \underline{\tilde{z}}_k$

$K_k = \bar{P}_k H_k^T (H_k \bar{P}_k H_k^T)^{-1}$

$\delta \underline{\hat{x}}_k = \delta \underline{\bar{x}}_k + K_k(\delta \underline{z}_k - H \delta \underline{\hat{x}}_k)$

$\underline{\hat{x}}_k = \underline{\tilde{x}}_k + \delta \underline{\hat{x}}_k$

$\hat{P}_k = (I - K_k H_k)\bar{P}$

Time Update:

$\dot{\underline{\tilde{x}}} = \underline{f}(\underline{\tilde{x}}, \underline{u})$

$\delta \dot{\underline{\bar{x}}}(t) = F(t) \delta \underline{\bar{x}}(t)$

$\underline{\bar{x}}(t) = \underline{\tilde{x}} + \delta \dot{\underline{\bar{x}}}(t)$

$\dot{\bar{P}}(t) = F(t)\bar{P}(t) + \bar{P}(t)F^T + G(t)\tilde{Q}^T(t)$

In Algorithm (1) the nominel trajectory is denoted as $\underline{\tilde{x}}$. The difference between the nominal trajectory and the true trajectory results in the error state the Kalman filter is estimating:

$$x_{\text{error}} = x_{\text{true}} - x_{\text{nominal}} \tag{6.1}$$

$$\delta \underline{\bar{x}} = \underline{\bar{x}} - \underline{\tilde{x}} \tag{6.2}$$

Rearranging this then becomes:

$$\underline{\bar{x}} = \delta \underline{x} + \underline{\tilde{x}} \tag{6.3}$$

Tables (6.1), (6.2) and (6.3) sums up all equations that is necessary for the

Error definitions		
$\delta \underline{f} = \underline{f}^b - \underline{\tilde{f}}^b$	$\delta \underline{p}^n = \underline{p}^n - \underline{\tilde{p}}^n$	$R_b^n = \tilde{R}_b^n R(\epsilon)$
$\delta \underline{\omega} = \underline{\omega}_{ib}^b - \underline{\tilde{\omega}}_{ib}^b$	$\delta \underline{v}^n = \underline{v}^n - \underline{\tilde{v}}^n$	$R(\epsilon) = I + S(\epsilon)$

Table 6.1: Error Definitions INS

Physical System	Mechanised System	Error Equations
$\dot{\underline{p}}^n = \underline{v}^n$	$\dot{\underline{\tilde{p}}}^n = \underline{\tilde{v}}^n$	$\delta \dot{\underline{p}}^n = \delta \underline{v}^n$
$\dot{\underline{v}}^n = R_b^n \underline{f}^b - \underline{g}^n$	$\dot{\underline{\tilde{v}}}^n = \tilde{R}_b^n \underline{\tilde{f}}^b - \underline{g}^n$	$\delta \dot{\underline{v}}^n = -\tilde{R}_b^n S(\underline{\tilde{f}}^b)\underline{\epsilon} + \tilde{R}_b^n \delta \underline{f}$
$\dot{R}_b^n = R_b^n S(\underline{\omega}_b^{nb})$	$\dot{\tilde{R}}_b^n = \tilde{R}_b^n S(\underline{\tilde{\omega}}_b^{nb})$	$\dot{\underline{\epsilon}} = -S(\underline{\tilde{\omega}}_b^{nb})\underline{\epsilon} + \delta \underline{\omega}_b^{nb}$

Table 6.2: Physical, Mechanisation and Error equations

development of the Kalman error differential equations [5].

Variable	Description
p	Position
v	Velocity
f	Acceleration / specific force
g^n	Gravitation component represented in n $[9.81\frac{m}{s^2}]$
$\underline{\tilde{\omega}}_b^{nb}$	Measured gyro velocity between n and b represented in b
$\underline{\omega}_b^{nb}$	True rotation velocity
$\delta\underline{\omega}$	Gyroscope error
$\underline{\tilde{f}}^b$	Specific force represented in the body axis
$\underline{f}^b$	True Specific force in the body axis
$\underline{\delta f}$	Accelerometer error
R_b^n	Rotation matrix, body to navigation frame

Table 6.3: Variable Description

6.2 Discretisation

Discretization is a process that transfers continuous time models into discrete models. This process is necessary because of the sampling process of a computer. In Algorithm (1) on line 7 and 9 the error system F and the Kalman covariance is continuous. In this section the process of discretisation is shown of the error error equations is shown. Given the continuous state space model:

$$\dot{\underline{x}}(t) = F(t)\underline{x}(t) + L(t)\underline{u}(t) + G(t)\underline{v}(t) \tag{6.4}$$

the descritisation can be writen as:

$$\underline{x}_{k+1} = \Phi_k\underline{x}_k + \Lambda_k\underline{u}_k + \Gamma_k\underline{v}_k \tag{6.5}$$

which can be written as a integral over a time region where G and L is assumed to be constant:

$$\Phi_k = e^{F((k+1)-k)} \tag{6.6}$$

$$\Lambda_k = \int_k^{k+1} e^{F((k+1)-\tau)} L_k\underline{u}(\tau)d\tau \tag{6.7}$$

$$\Gamma_k = \int_k^{k+1} e^{F((k+1)-\tau)} G_k\underline{v}(\tau)d\tau \tag{6.8}$$

Since the navigation equations does not include $\underline{u}(t)$ the $L(t)/\Lambda_k$ is considered to be zero. The continuous linearised Kalman filter equations which include

F and G are:

$$\delta\dot{\underline{\bar{x}}}(t) = F^*(t)\delta\underline{\bar{x}}(t) \tag{6.9}$$

$$\dot{\bar{P}}(t) = F^*(t)\bar{P}(t) + \bar{P}(t)F^{*T}(t) + G^*(t)\tilde{Q}G^*(t)^T \tag{6.10}$$

Relationship between continous and descrete F then becomes:

$$\Phi_k = e^{F\Delta_t} \tag{6.11}$$

where $\delta\dot{\underline{\bar{x}}}(t)$ can be re written as:

$$\delta\dot{\underline{\bar{x}}}_{k+1} = \Phi_k\delta\underline{\bar{x}}_k \tag{6.12}$$

To solve $\dot{\bar{P}}(t)$,

$$P(t) = X(t)Z^{-1}(t) \tag{6.13}$$

is defined where $P(t_0)$ is given and F, G, $\tilde{Q}$ is assumed to be invariant or constant over the time interval Δ_t. At the time t_0: $X(t_0) = P(t_0)$ and $Z(t_0) = I$ where I is the identity matrix. Rearangig 6.13:

$$X(t) = P(t)Z(t) \tag{6.14}$$

derivate on both sides and insert for $\dot{P}$ yields:

$$\dot{X} = \dot{P}Z + P\dot{Z} \tag{6.15}$$

$$\dot{X} = [FP + PF^T + G\tilde{Q}G^T]Z + P\dot{Z} \tag{6.16}$$

$$\dot{X} - FPZ - G\tilde{Q}G^TZ = PF^TZ + P\dot{Z} \tag{6.17}$$

$$\dot{X}FX - G\tilde{Q}G^TZ = P[F^TZ + \dot{Z}] \tag{6.18}$$

This can be written in matrix from:

$$\begin{bmatrix}\dot{X}\\ \dot{Z}\end{bmatrix} = \underbrace{\begin{bmatrix}F & G\tilde{Q}G^T\\ 0 & -F^T\end{bmatrix}}_{\Xi_{xz}}\begin{bmatrix}X\\ Z\end{bmatrix} \tag{6.19}$$

Linearisation where:

$$\Xi_{xz} = e^{F_{xz}(t,t_0)} \tag{6.20}$$

is equivalent to the Power Series or Tailor expansion to the matrix:

$$e^{F_{xz}} = \sum_{i=0}^{\infty}\frac{1}{i!}[F_{xz}(t,t_0)]^i \tag{6.21}$$

Linerarized matrix from becomes:

$$\begin{bmatrix} X(t) \\ Z(t) \end{bmatrix} = \begin{bmatrix} \Xi_{11}(t,t_0) & \Xi_{12}(t,t_0) \\ 0 & \Xi_{22}(t,t_0) \end{bmatrix} \begin{bmatrix} X(t_0) \\ Z(t_0) \end{bmatrix} \tag{6.22}$$

The equations become:

$$Z(t) = \Xi_{22}(t,t_0) \tag{6.23}$$
$$X(t) = \Xi_{11}(t,t_0)X(t_0) + \Xi_{12}(t,t_0) \tag{6.24}$$

Inserting this into (6.14) yields:

$$P(t) = \Xi_{11}(t,t_0)X(t_0)\Xi_{22}^{-1}(t,t_0) + \underbrace{\Xi_{12}(t,t_0)\Xi_{22}^{-1}(t,t_0)}_{\Gamma\tilde{Q}\Gamma^T} \tag{6.25}$$

Drawing out the descretetisation of $\Gamma\tilde{Q}\Gamma^T$ yeilds:

$$S = \Xi_{12}(t,t_0)\Xi_{22}^{-1}(t,t_0) \tag{6.26}$$

Finally the covariance of the prediction can be written as:

$$\bar{P}_{k+1} = \Phi_k \bar{P}_k \Phi_k^T + S \tag{6.27}$$

The calculation of S has been implemented in the matlab function *k2dS*, pseudo code:

Algorithm 2 Contionous to descrete calculation of $\Gamma\tilde{Q}\Gamma^T$

1: **function** K2DS($F(t), G(t), \tilde{Q}, \Delta_t$)
2: $A = \begin{bmatrix} F & G\tilde{Q}G^T \\ 0 & -F^T \end{bmatrix}$
3: $\Xi = e^A\Delta_t$
4: $k2dS = \frac{\Xi_{11}}{\Xi_{22}}$
5: **end function**

This can be simplified to the a first order approximation:

$$\Gamma_k Q \Gamma_k^T = \int_k^{k+1} \Phi_k G_k \tilde{Q} G_k^T \Phi_k^T \tag{6.28}$$
$$\approx \int_k^{k+1} I G_k \tilde{Q} G_k^T I \tag{6.29}$$
$$\approx G_k \tilde{Q} G_k^T \Delta_t \tag{6.30}$$

which can be a reasonable choice if the object of implementation has low computational power. The task of recursively calculating line 7 and 9 Algorithm (1) becomes:

$$\Phi_k = e^{F\Delta_t} \tag{6.31}$$

$$\bar{P}_{k+1} = \Phi_k \bar{P}_k \Phi_k^T + K2DS(F(t), G(t), \tilde{Q}, \Delta_t) \tag{6.32}$$

The navigation equations on line 6 in Algorithm (1) is continuous and is solved with Heuns integration method.

6.3 Kalman error equations

The kalman filters differential equations can be written as:

$$\dot{\underline{x}} = F\underline{x} + G\underline{v} \tag{6.33}$$

Where F is the process matrix G is the process noise matrix. The unlinear error model F, is described by [5] as:

$$\delta\dot{\underline{p}}^n = \delta\underline{v}^n \tag{6.34}$$

$$\delta\dot{\underline{v}}^n = -S(\tilde{R}_b^n \underline{\tilde{f}}^b)\underline{\epsilon} + \tilde{R}_b^n \delta\underline{f} \tag{6.35}$$

$$\dot{\underline{\epsilon}} = -S(\underline{\tilde{\omega}}_b^{nb})\underline{\epsilon} + \tilde{R}_b^n \delta\underline{\omega}_b^{nb} \tag{6.36}$$

Where the error of the sensors such as bias is incorporated into $\delta\underline{\omega}_b^{nb}$ and $\delta\underline{f}$, these errors is subject to expansion. In this thesis a bias for the accelerometer and a bias for the gyroscope will be modelled. The bias differential equation becomes:

$$\delta\underline{f} \equiv \dot{\underline{\beta}}^{acc} = 0 \tag{6.37}$$

$$\delta\underline{\omega}_b^{nb} \equiv \dot{\underline{\beta}}^{gyr} = 0 \tag{6.38}$$

$$\tag{6.39}$$

The kalman error state vector then becomes:

$$\delta\dot{\underline{\tilde{x}}} = \begin{bmatrix} \dot{\underline{p}}^n & \dot{\underline{v}}^n & \dot{\underline{\epsilon}}^n & \dot{\underline{\beta}}^{acc} & \dot{\underline{\beta}}^{gyr} \end{bmatrix}^T \tag{6.40}$$

The error system F then becomes:

$$F(t) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -S(\tilde{R}_b^n(t)\underline{\tilde{f}}^n(t)) & \tilde{R}_b^n(t) & 0 \\ 0 & 0 & 0 & 0 & \tilde{R}_b^n(t) \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.41}$$

and the process noise matrix becomes:

$$G(t) = \begin{bmatrix} 0 & 0 \\ \tilde{R}_b^n(t) & 0 \\ 0 & \tilde{R}_b^n(t) \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{6.42}$$

6.4 Aiding sensors

A aiding sensor is a sensor which is giving an absolute value of the state and does not rely on the information already in the filter. Examples of this type of aiding sensor can be the magnetometer for the attitude estimation. The magnetometer measures magnetic flux and is sensitive enough to measure the earth's magnetic field. Though this sensor is vulnerable to external magnetic fields from electrical engines, magnets ect. Another example could be GPS for the position, some GPS sensors connected to a triad of antennas are able to measure attitude as well. Barometric pressure is possible to convert to an altitude, this stabilizes the drift in the z axis.

6.4.1 Magnetometer Aid

In this thesis the magnetometer is used to stabilise the attitude. The error definition of the measurements is defined as:

Variable	Description
$\underline{z}$	True Measurement
$\underline{\tilde{z}}$	Simulated measurement
$\delta\underline{z}$	Error in simulated measurement

Table 6.4: Variable Description

The true magnetic measurement is the magnetometer measurement $\underline{\tilde{B}}^b$, where as the simulated measurement is found by the World Magnetic Model (WMM) [11]. Matlab has a function [10] that calculates the local magnetic flux in the NED navigation frame. This vector is the measurement of the flux in the NED frame which tangential to the surface and has its x axis pointing to the north. The input to the WMM is longitude latitude and altitude, for testing purposes was this data found by the use of an android enabled smart phone. Kjeller is located at 59.97487°N, 11.04532°E the altitude was set to

zero. The resulting magnetic vector became:

$$\underline{B}^{n(WMM)} = \begin{bmatrix} 0.151094 & 0.005486 & 0.487058 \end{bmatrix} \tag{6.43}$$

The WMM function in Matlab returns the magnetic field vector in NED form. The ADIS module is placed on the ground with its magnetic z-axis pointing up. To be able to compare the two magnetic vectors both a normalization and rotation is necessary. The WMM magnetic vector is rotated 180° around its x axis as the definition of the n-frame is requiring z to point upwards. The normalisation is done by dividing the WMM magnetic vector by its own magnitude. This magnitude is found by the use of the euclidean norm:

$$\underline{B}^{n} = \frac{\begin{bmatrix} \underline{B}_x^{n(WMM)} & -\underline{B}_y^{n(WMM)} & -\underline{B}_z^{n(WMM)} \end{bmatrix}}{||\underline{B}^{n(WMM)}||} \tag{6.44}$$

This became:

$$\underline{B}^{n} = \begin{bmatrix} 0.2963 & -0.0108 & -0.9550 \end{bmatrix} \tag{6.45}$$

$\underline{B}^n$ it then the flux vector from World Magnetic Model, $\tilde{\underline{B}}^b$ is the measured magnetic filed from the ADIS sensor. $\underline{B}^b$ also needs to be normalized:

$$\underline{B}_k^b = \frac{\underline{B}_k^b}{||\underline{B}^b||} \tag{6.46}$$

The two magnetic field measurements are defined in two different frames therefore $\underline{B}^n$ has to be rotated to the body frame. This is done with attitude found by simulation:

$$\underline{B}^b = (\tilde{R}_b^n)^T \underline{B}^n \tag{6.47}$$

$\delta \underline{z}_k$ for the magnetic measurement then becomes:

$$\delta \underline{z}_k = \underline{z}_k - \tilde{\underline{z}}_k \tag{6.48}$$

$$\delta \underline{B}_k^b = \tilde{\underline{B}}^b - (\tilde{R}_b^n)^T \underline{B}^n \tag{6.49}$$

It is beneficial to let the covariance and bias converge before the platform is exposed to external forces. Therefore two versions of the measurement matrix H has been used. One in the alignment phase and one for free run. As the sensor is kept completely still in the alignment phase both velocity and position is known. The measurement matrix element corresponding to position and velocity becomes the identity matrix. To find the element that corresponds to the magnetic field in the measurement matrix the angulation ϵ

has to be used. The scope is to find the error systems simulated measurement based on the angulation ϵ. The error definitions are:

$$R(\epsilon) = I + S(\epsilon) \tag{6.50}$$
$$R_b^n = R(\epsilon)\tilde{R}_b^n \tag{6.51}$$

the scope is to re write $\tilde{R_b^n}$ to be based on ϵ:

$$\begin{aligned}
\delta\underline{z}_k^b &= \underline{\tilde{B}}^b - (\tilde{R}_b^n)^T\underline{B}^n && (6.52)\\
&= (R_b^n)^T\underline{B} - (\tilde{R}_b^n)^T\underline{B}^n && (6.53)\\
&= [(R_b^n)^T - (\tilde{R}_b^n)^T]\underline{B}^n && (6.54)\\
&= [(I + S(\epsilon^n)\tilde{R}_b^n)^T - (\tilde{R}_b^n)^T]\underline{B}^n && (6.55)\\
&= [I + S(\epsilon^n)^T(\tilde{R}_b^n)^T - (\tilde{R}_b^n)^T]\underline{B}^n && (6.56)\\
&= [(\tilde{R}_b^n)^T + S(\epsilon^n)^T(\tilde{R}_b^n)^T - (\tilde{R}_b^n)^T]\underline{B}^n && (6.57)\\
&= S(\epsilon^n)^T(\tilde{R}_b^n)^T\underline{B}^n && (6.58)\\
&= -(\tilde{R}_b^n)^T S(\epsilon^n)\underline{B}^n && (6.59)\\
\delta\underline{z}_k^b &= (\tilde{R}_b^n)^T S(\underline{B}^n)\epsilon^n + \underline{w}^b && (6.60)\\
& && (6.61)
\end{aligned}$$

Where

$$\begin{aligned}
S^T &= -S && (6.62)\\
a \times b &= -b \times a && (6.63)\\
S(a)b &= -S(b)a && (6.64)\\
R_b^n &= (I + S(\epsilon))\tilde{R}_b^n && (6.65)\\
(R_b^n)^T &= R_b^b && (6.66)
\end{aligned}$$

From [7] the $\delta\underline{\hat{x}}$ is given as:

$$\delta\underline{\hat{x}} = \delta\underline{\bar{x}} + K(\delta\underline{z} - H\delta\underline{\bar{x}}) \tag{6.67}$$

Since epsilon is a part of the state $\delta\underline{x}$ which is multiplied with H it is omitted and the measurement matrix become:

$$H_k = \begin{bmatrix} I & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 \\ 0 & 0 & (\tilde{R}_b^n)^T S(\underline{B}^n) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.68}$$

When the position of the system is unknown the measurement matrix becomes:

$$H_k = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & (\tilde{R}_b^n)^T S(\underline{B}^n) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.69}$$

6.5 Initial Alignment

An initial alignment algorithm has been developed to initialize the navigation attitude. An initial error in the attitude may cause huge errors, the initial alignment is a rough estimate on the attitude of the body frame. This is fine tuned by the alignment phase incorperated in the kalman filter. The algorithem is given as:

Algorithm 3 Initial Alignment

function $\tilde{R}_b^n = \text{INITR}(\underline{B}^n, \underline{\tilde{B}}^b)$
  $R_c^b = \begin{bmatrix} \frac{\underline{\tilde{B}}_m^b}{||\underline{\tilde{B}}_m^b||} & \frac{\underline{\tilde{B}}_m^b \times \underline{\tilde{f}}_m^b}{||\underline{\tilde{B}}_m^b \times \underline{\tilde{f}}_m^b||} & \frac{\underline{\tilde{B}}_m^b}{||\underline{\tilde{B}}_m^b||} \times \frac{\underline{\tilde{B}}_m^b \times \underline{\tilde{f}}_m^b}{||\underline{\tilde{B}}_m^b \times \underline{\tilde{f}}_m^b||} \end{bmatrix}$
  $R_c^n = \begin{bmatrix} \frac{\underline{B}^n}{||\underline{B}^n||} & \frac{\underline{B}^n \times \underline{f}^n}{||\underline{B}^n \times \underline{f}^n||} & \frac{\underline{B}^n}{||\underline{B}^n||} \times \frac{\underline{B}^n \times \underline{f}^n}{||\underline{B}^n \times \underline{f}^n||} \end{bmatrix}$
  $R_b^n = R_c^n (R_c^b)^T$
end function

6.6 LKF Pseudo

The linerised kalman filter starts with the integration of the navigation equations:

Algorithm 4 LKF Pseudo code part I (Navigation Simulation)

for $k = 1 \rightarrow N$ **do**
  $\underline{\tilde{x}}'_{k+1} = \underline{\tilde{x}}_k + \Delta_t f(\underline{\tilde{x}}_k, \underline{u}_k)$
  $\underline{\tilde{x}}_{k+1} = \underline{\tilde{x}}_k + \frac{\Delta_t}{2}(f(\underline{\tilde{x}}_k, \underline{u}_k) + f(\underline{\tilde{x}}'_{k+1}, \underline{u}_{k+1})$
  $\tilde{R}_b^n = \begin{bmatrix} \tilde{x}_7 & \tilde{x}_8 & \tilde{x}_9 \\ \tilde{x}_{10} & \tilde{x}_{11} & \tilde{x}_{12} \\ \tilde{x}_{13} & \tilde{x}_{14} & \tilde{x}_{15} \end{bmatrix}$

The time update is given as:

Algorithm 5 LKF Pseudo code part II (Time Update)

5: $F = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -S(R_b^n \underline{\widetilde{f}}_k^n) & \widetilde{R}_b^n & 0 \\ 0 & 0 & 0 & 0 & \widetilde{R}_b^n \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

6: $G = \begin{bmatrix} 0 & 0 \\ \widetilde{R}_b^n & 0 \\ 0 & \widetilde{R}_b^n \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$

7: $\Phi = expm(F\Delta_t)$

8: $\Gamma = kp2dpGa(F, G, Q, \Delta_t)$

9: $\delta\underline{\bar{x}}_k = \Phi\delta\underline{\bar{x}}_k$

10: $\bar{P}_k = \Phi\hat{P}_k\Phi^T + \Gamma Q\Gamma^T$

11: $\underline{\bar{x}}_k = ([\underline{\tilde{x}}_k]^T \begin{bmatrix} I_{6x6} \\ 0_{9x6} \end{bmatrix})^T + \delta\underline{\bar{x}}_k$

12: $\epsilon = \begin{bmatrix} \delta\bar{x}_{k_1} & \delta\bar{x}_{k_2} & \delta\bar{x}_{k_3} \end{bmatrix}$

13: $\bar{R}_b^n = \tilde{R}_b^n + (I + S(\epsilon))\tilde{R}_b^n$

Algorithm 6 LKF Pseudo code part III (Measurement Update)

if n is odd **then**

$$H_k = \begin{bmatrix} I & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 \\ 0 & 0 & (\tilde{R}_b^n)^T S(\underline{B}^n) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$K_k = \bar{P}_k H_k^* (H_k \bar{P}_k H^{*T} + R_k)^{-1}$

$\hat{P}_k = (I - K_k^* H_k^*) \bar{P}_k$

$\underline{z}_k = \begin{bmatrix} 0 & 0 & \underline{\tilde{B}}^b & 0 & 0 \end{bmatrix}$

$\underline{\tilde{z}}_k = \begin{bmatrix} \underline{\tilde{p}}_k & \underline{\tilde{v}}_k & \tilde{R}_b^n \underline{B}^n & 0 & 0 \end{bmatrix}$

$\delta \underline{z}_k = \underline{z}_k - \underline{\tilde{z}}_k$

$\delta \underline{\hat{x}}_k = \delta \underline{\bar{x}}_k + K_k (\delta \underline{z}_k - H_k^* \delta \underline{\bar{x}}_k)$

$\underline{\hat{x}}_k = \underline{\tilde{x}}_k + \delta \underline{\hat{x}}_k$

else

$$H_k = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & (\tilde{R}_b^n)^T S(\underline{B}^n) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$K_k = \bar{P}_k H_k^* (H_k \bar{P}_k H^{*T} + R_k)^{-1}$

$\hat{P}_k = (I - K_k^* H_k^*) \bar{P}_k$

$\underline{z}_k = \begin{bmatrix} 0 & 0 & \underline{B}^b & 0 & 0 \end{bmatrix}$

$\underline{\tilde{z}}_k = \begin{bmatrix} 0 & 0 & (\tilde{R}_b^n)^T \underline{B}^n & 0 & 0 \end{bmatrix}$

$\delta \underline{z}_k = \underline{z}_k - \underline{\tilde{z}}_k$

$\delta \underline{\hat{x}}_k = \delta \underline{\bar{x}}_k + K_k (\delta \underline{z}_k - H_k^* \delta \underline{\bar{x}}_k)$

$\underline{\hat{x}}_k = \underline{\tilde{x}}_k + \delta \underline{\hat{x}}_k$

end if

end for

Chapter 7

Results

In this chapter the performance of the implemented Kalman filter is shown.

7.1 Drift Results

A series of measurements were take when the sensor was completely still. The length of the data gathered was 8000 samples at 200Hz which is 40 seconds.

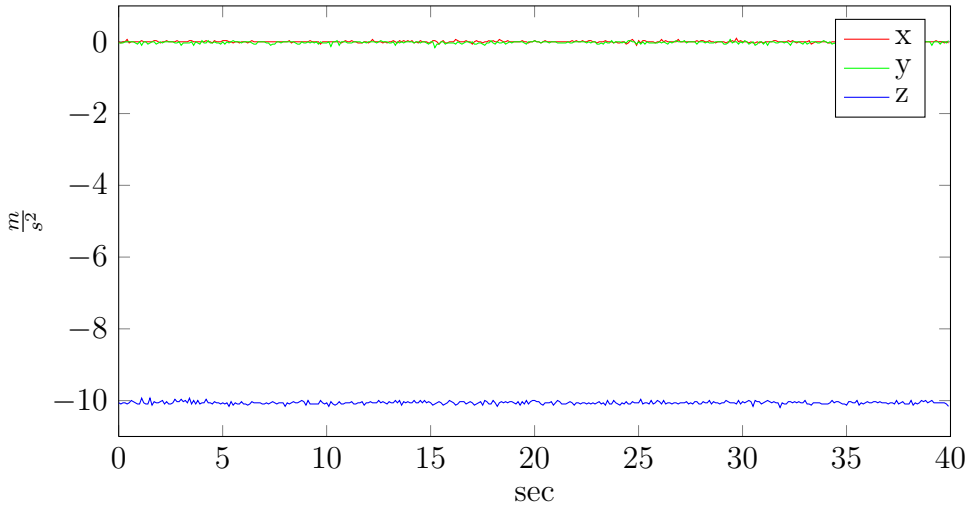


Figure 7.1: Raw Accelerometer measurements

Figure (7.1) displays the raw unaltered measurement of the accelerometer is plotted, the x- and y-axis measurement should be zero and the z-axis measurement should be 9.81. The z-axis is clearly affected by a scale factor or bias, the mean of the z-axis in this plot is -10.0601. The accelerometer

measurements are inverted ($\underline{\tilde{f}}^b \cdot (-1)$) before they are used in the Kalman filter.

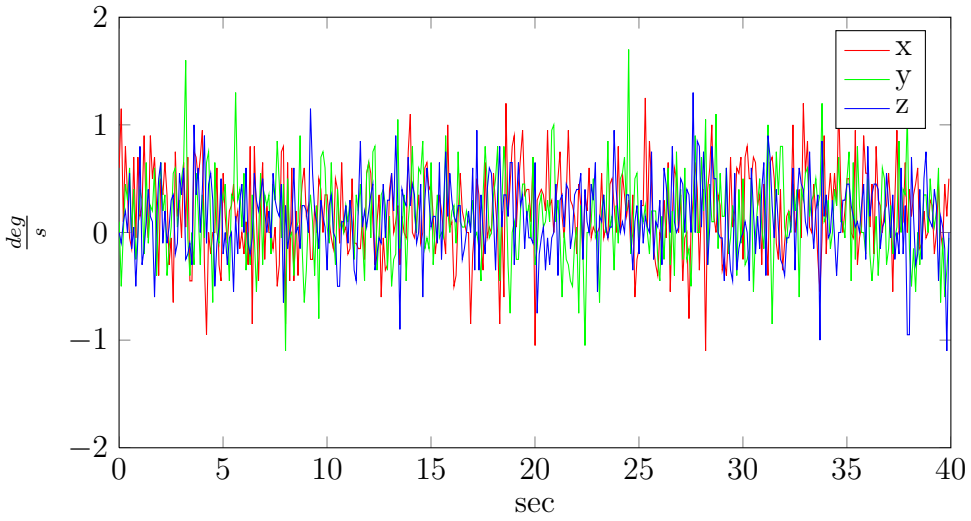


Figure 7.2: Raw Gyroscope measurements

Figure (7.2) displays the raw unaltered measurement of the gyroscope is plotted, there is a constant bias present. Calculating the mean yields the bias:

$$\tilde{\omega}_x^b = \frac{1}{8000}\sum_{i=1}^{8000}[\tilde{\omega}_x^b]_i = 0.1961^\circ \tag{7.1}$$

$$\tilde{\omega}_y^b = \frac{1}{8000}\sum_{i=1}^{8000}[\tilde{\omega}_y^b]_i = 0.1634^\circ \tag{7.2}$$

$$\tilde{\omega}_z^b = \frac{1}{8000}\sum_{i=1}^{8000}[\tilde{\omega}_z^b]_i = 0.1456^\circ \tag{7.3}$$

This bias is removed from the data before the measurements are used in the Kalman filter, then $\underline{\omega}^b$ is converted to radians.

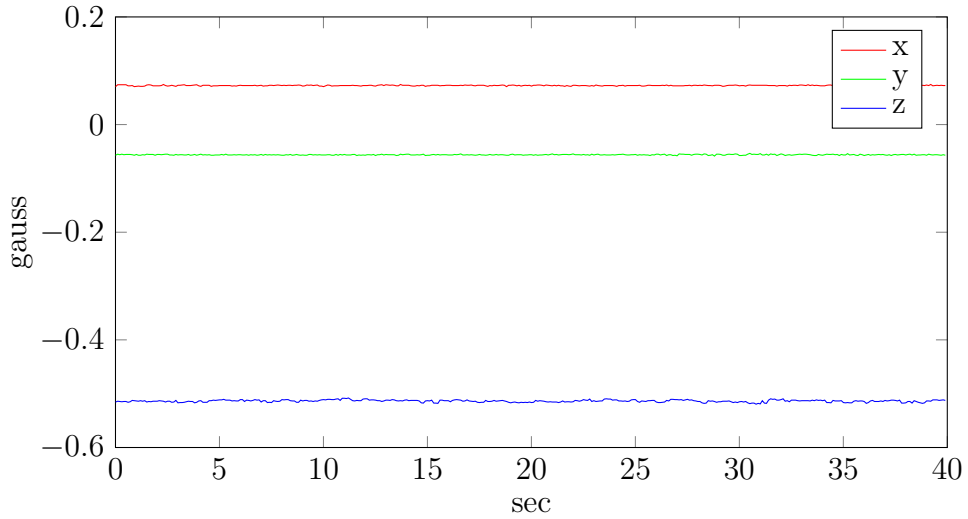


Figure 7.3: Raw Magnetometer measurements

Figure (7.3) displays the raw unaltered measurement of the magnetometer is plotted. This plot does not illustrate bias the apex is plotted to get more information as the apex should be constant. The apex is found by the recursively calculating the euclidean norm as shown in Figure (7.4).

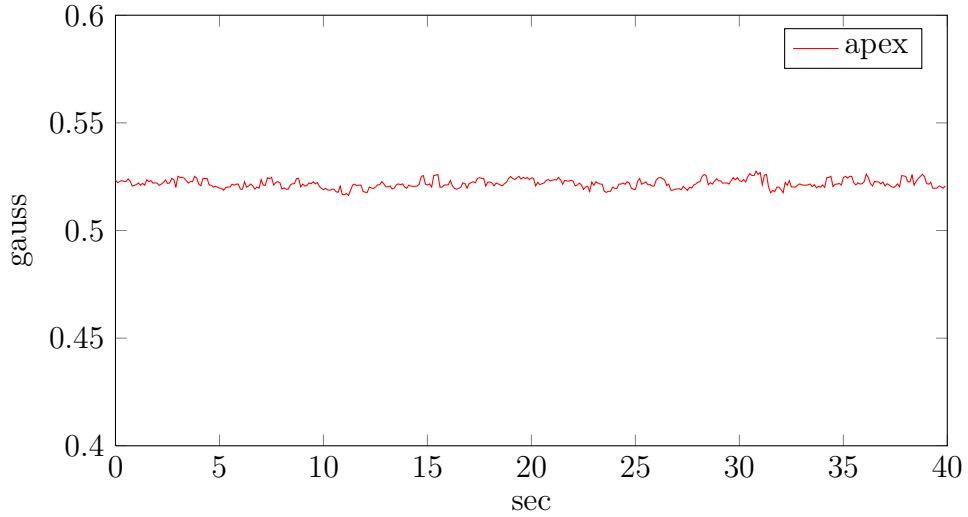


Figure 7.4: Apex magnetometer

In Table (7.1) the max, min and average value of the apex is shown:

Table 7.1: Apex mean max min

min	mean	max	Δmax/min
0.5151	0.5218	0.5280	-0.0128

In the Kalman filter section the $\underline{B}^n$ was normalized, as the $\underline{\tilde{B}}^b$ varies around 0.5218 a normalisation process is necessary also here. This is done by dividing the whole $\underline{\tilde{B}}^b$ by 0.5218. This results in the two magnetic vectors having the same magnitude which is important for the error calculation in the Kalman filter.

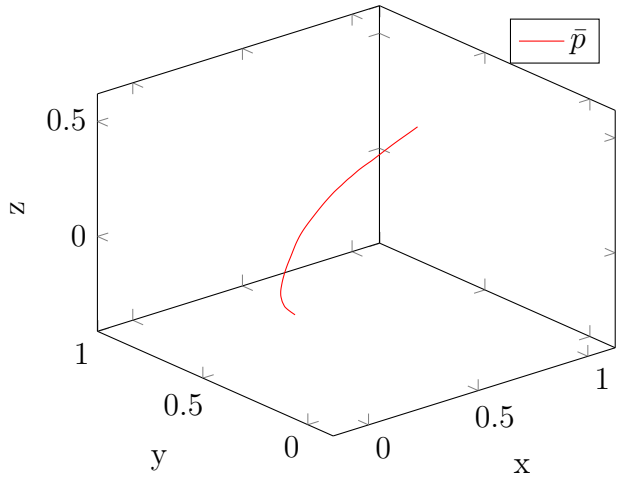


Figure 7.5: 10 seconds Drift result Position

Figure (7.5) shows the drift over 10 seconds of free run time of the estimated position which is 2.0791m. Comparing this to the end position of the nominal trajectory which became 452.6m a vast improvement is seen. Though 2.0791m is still a big drift considering only 10 seconds have passed. This suggest the necessity of a measurement update for the position states. The following Figures (7.6)-(7.14) illustrates the standard deviation based on the diagonal of the covariance P plotted against the estimated value of the corresponding state. From 0 to 30 seconds the Kalman filter is in its alignment phase where the position is known, from 30-40 seconds it runs without any position aid. Which is why the standard deviation is converging to 30 second and diverging after. The simulations are based on the diagonal

process noise matrix Q:

$$Q_{1,1} = 0.0018564 \qquad Q_{2,2} = 0.0044847 \qquad Q_{3,3} = 0.0030709 \tag{7.4}$$
$$Q_{4,4} = 0.029117 \qquad Q_{5,5} = 0.028674 \qquad Q_{6,6} = 0.027748 \tag{7.5}$$

and the diagonal measurement noise matrix R:

$$R_{1,1} = (10.0609^2)/2 \qquad R_{2,2} = (14.7831^2)/2 \qquad R_{3,3} = (24.3630^2)/2 \tag{7.6}$$
$$R_{4,4} = 10.0609 \qquad R_{5,5} = 14.7831 \qquad R_{6,6} = 24.3630 \tag{7.7}$$
$$R_{7,7} = 0.8425 \qquad R_{8,8} = 0.7427 \qquad R_{9,9} = 0.7427 \tag{7.8}$$
$$R_{10,10} = 0.00062394 \qquad R_{11,11} = 0.0016138 \qquad R_{12,12} = 0.00091252 \tag{7.9}$$
$$R_{13,13} = 0.0044451 \qquad R_{14,14} = 0.0054394 \qquad R_{15,15} = 0.0054394 \tag{7.10}$$

The R and Q diagonal elements are found by Allan variance and auto correlation of the corresponding signal with the $E\{x \cdot t\} = 0$. The measurement update frequency is set to be equally frequent as the time update:

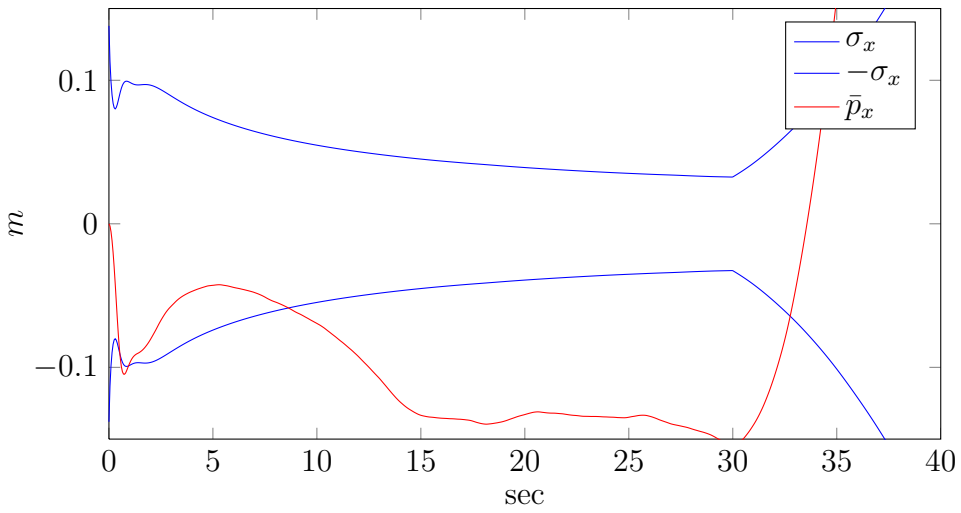


Figure 7.6: The standard deviation of the X Position is converging and the kalman filter is trusting the measurements almost correctly

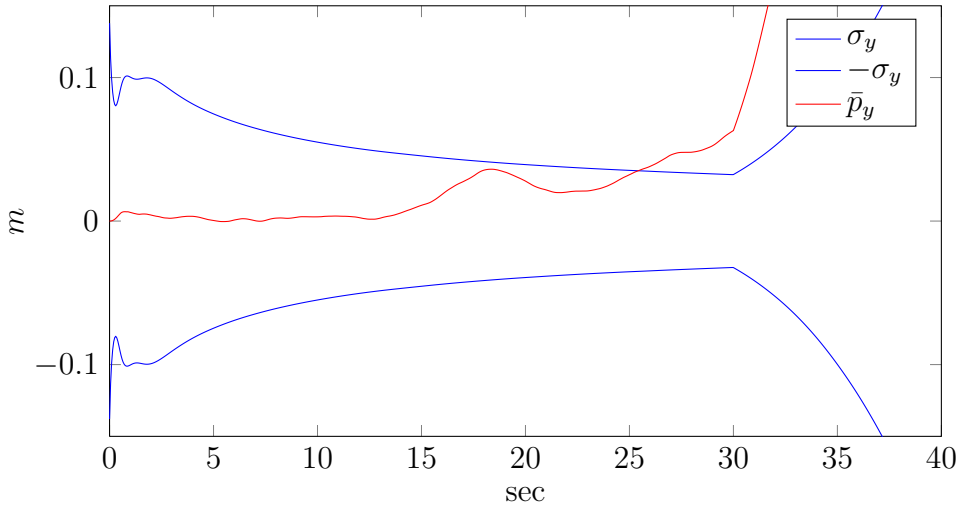


Figure 7.7: In this plot of the Y position the predicted value is outside the standard deviation around 1/3 of the time which is correct

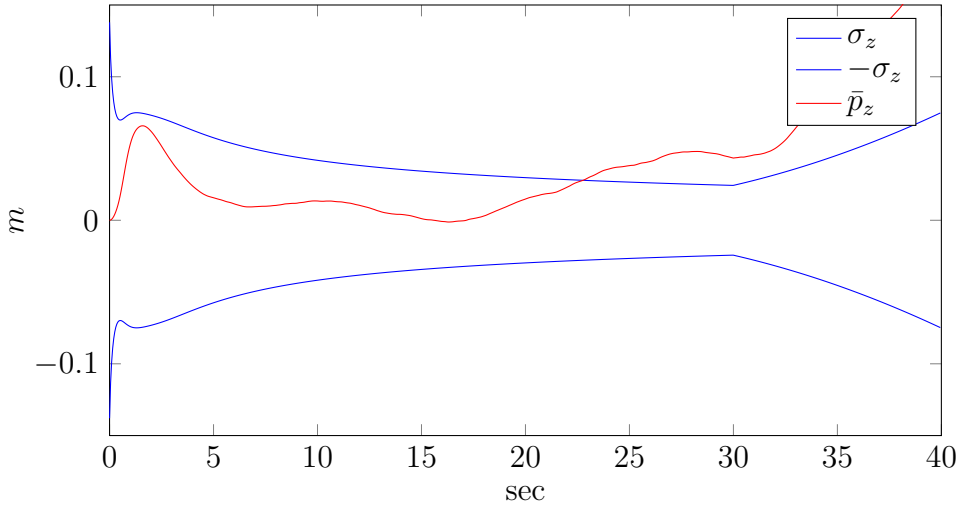


Figure 7.8: The kalman filter is trusting the measurements correctly but the estimate is to stable, this is the case for all the standard deviation plots presented above.

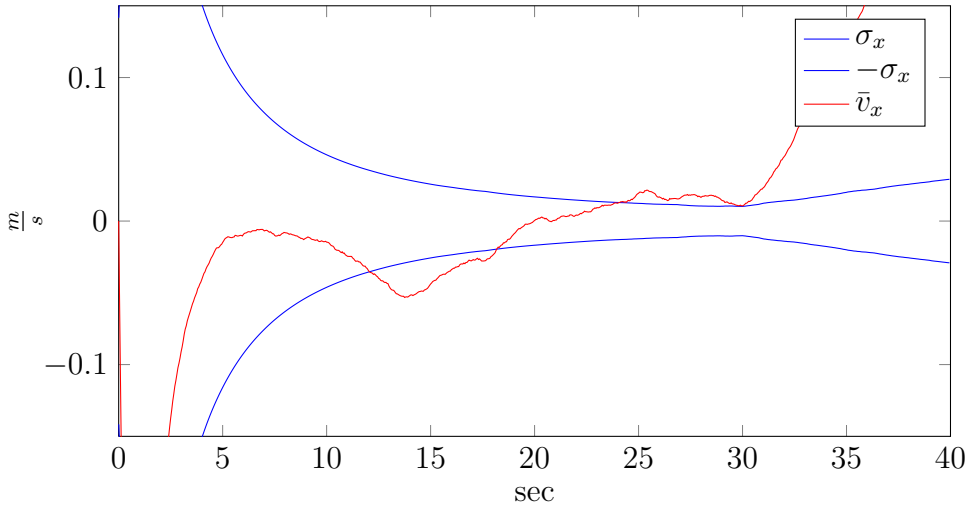


Figure 7.9: The velocity estimate in the x-axis is much better than the position as the estimate is varying

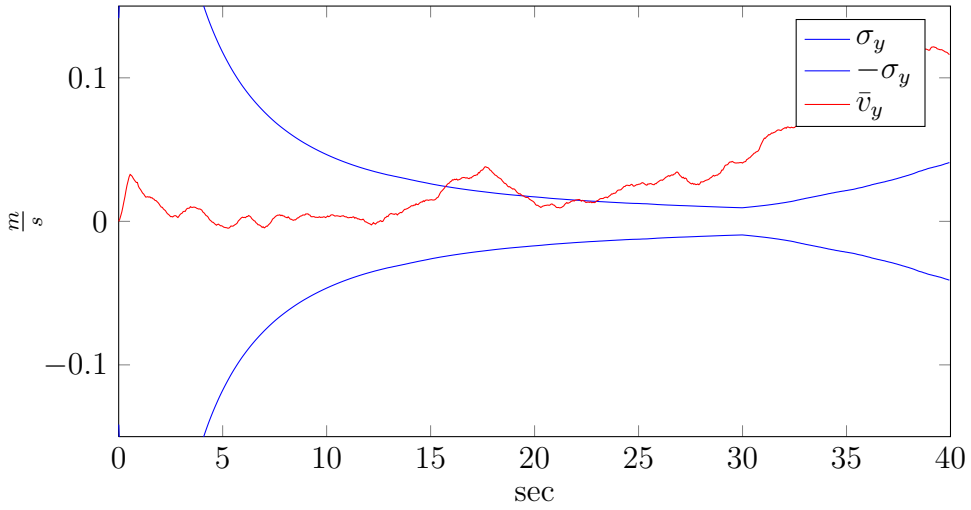


Figure 7.10: Velocity estimate in the y-axis is also converging correctly

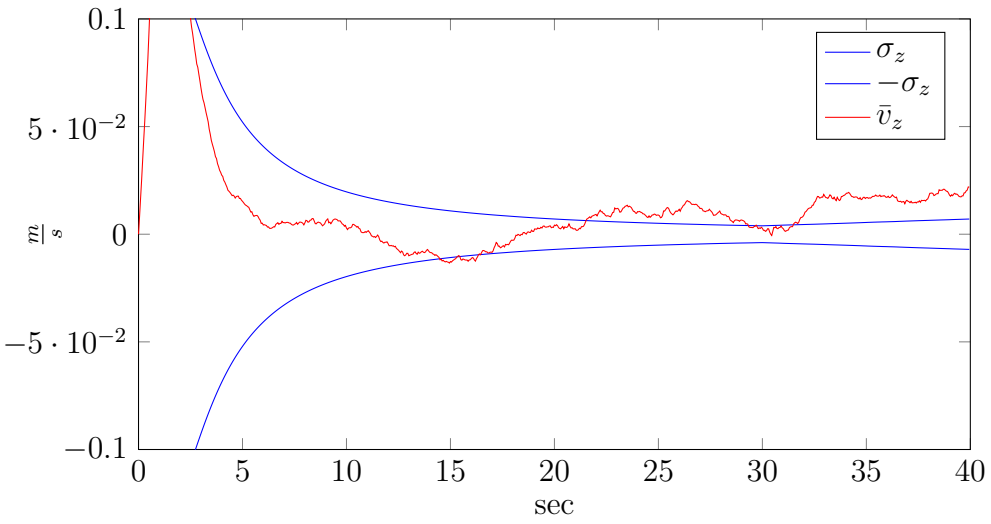


Figure 7.11: The standard deviation of the covariance is also converging for the velocity z-axis the Kalman filter is trusting the measurements sufficiently and correctly as the estimate is outside the $\pm\sigma$ one third of the time

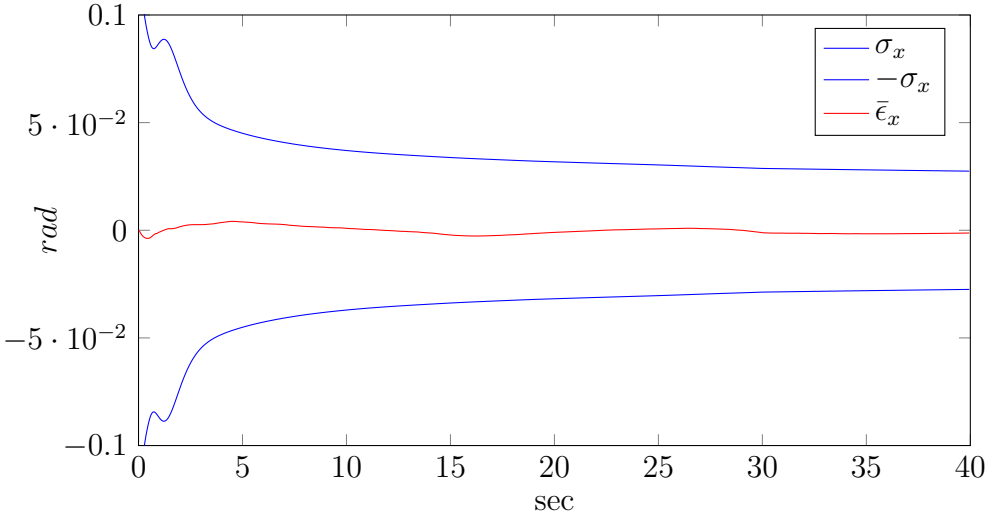


Figure 7.12: $\pm\sigma_x$ converges and does not diverge after 30 seconds as all the previous plots this is correct as the magnetometer measurement is always present

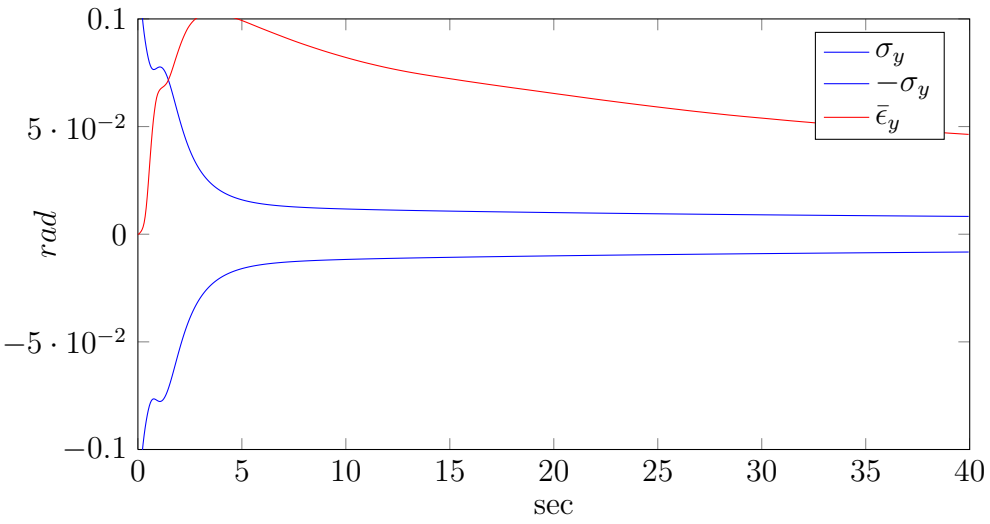


Figure 7.13: $\pm\sigma_y$ converges as in the x case but here the Kalman filter is trusting the measurement to much

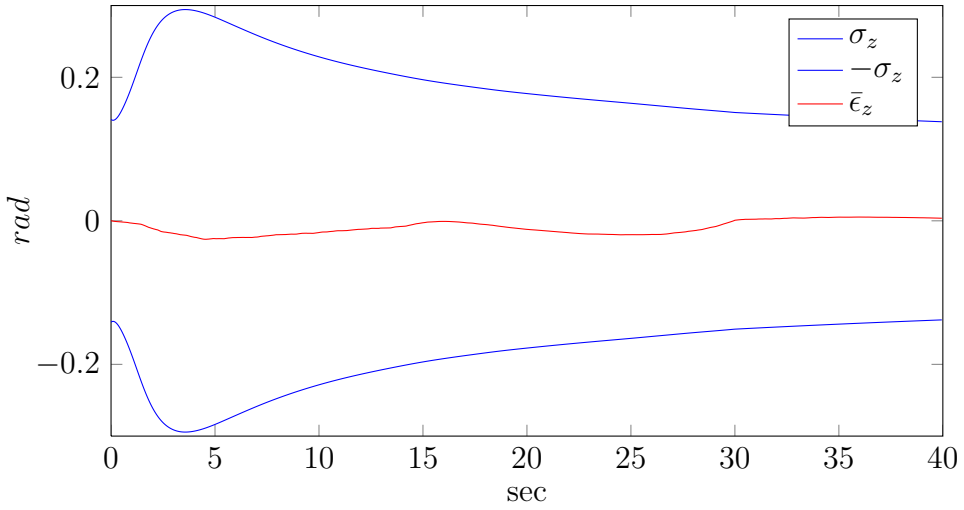


Figure 7.14: $\pm\sigma_z$ converges as in the x and y case but this result is similar to the x case

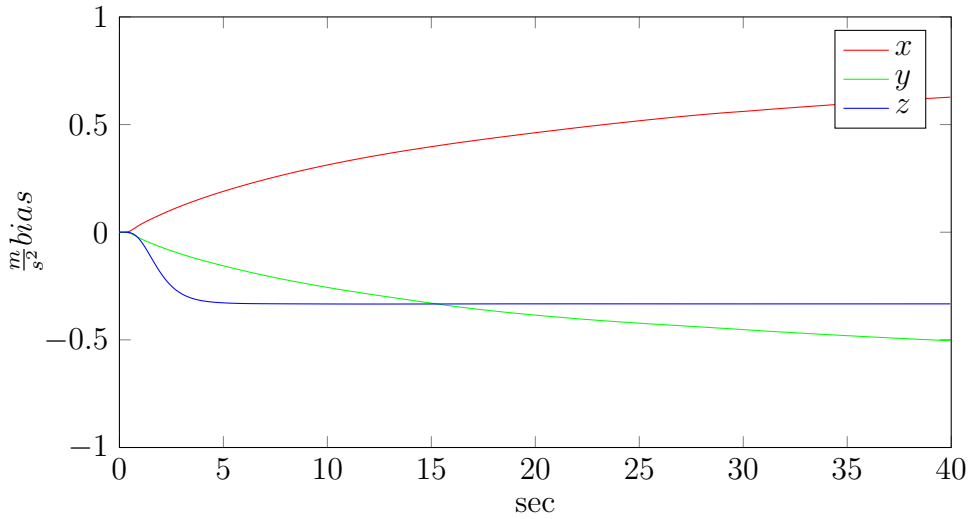


Figure 7.15: Bias converges for the accelerometer

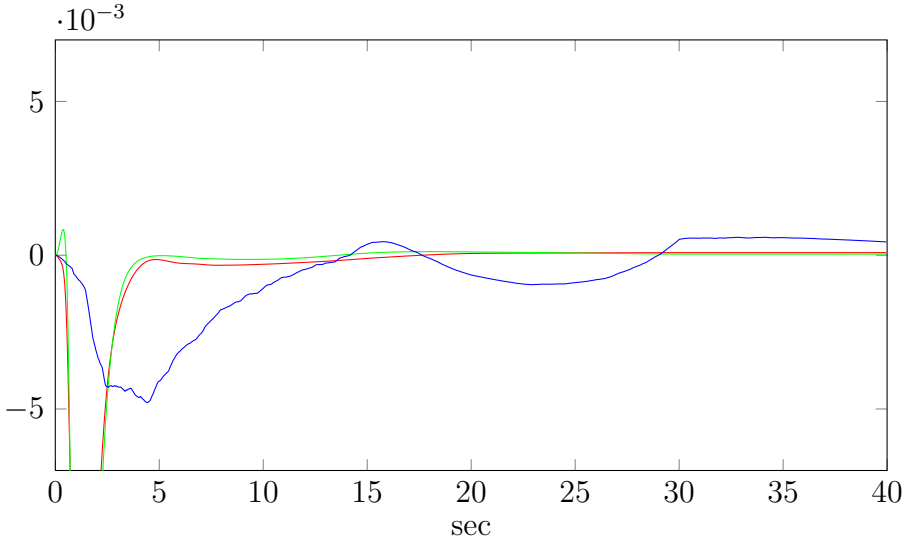


Figure 7.16: Bias for the gyroscopes converges

Common for all the standard deviation plots for the drift test is that the estimate is to stable, multiple attempts has been made adjusting bot R, Q and P_0 without success.

7.2 Spiral

The sensor was held completely still for 11 seconds before the sensor was led in a upwards going spiral. The spiral took 4.5 seconds to complete it was executed manually. A video of the execution is available as a supplement, this clearly illustrates that the implementation of the Kalman filter an measurements are correct though there are some strange behaviour. The R and Q matrices were exactly equal to the ones used on the drift dataset. The square root of the diagonal elements of P is plotted together with the corresponding predicted value in Figures (7.18) - (7.26)

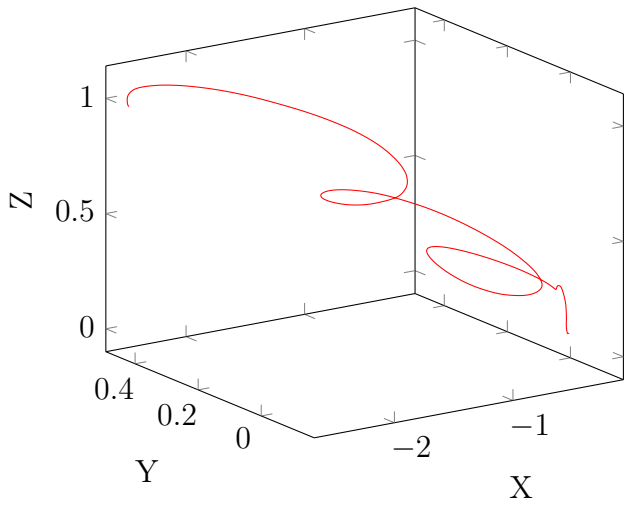


Figure 7.17: Plot of the estimated value of the position in three dimensions. This spiral is plot very similar to the execution done by the author. There is no accurate way of determining how accurate the position is since the true position is unknown. The y axis has seamingly a big bias as the spiral drifts in the y direction

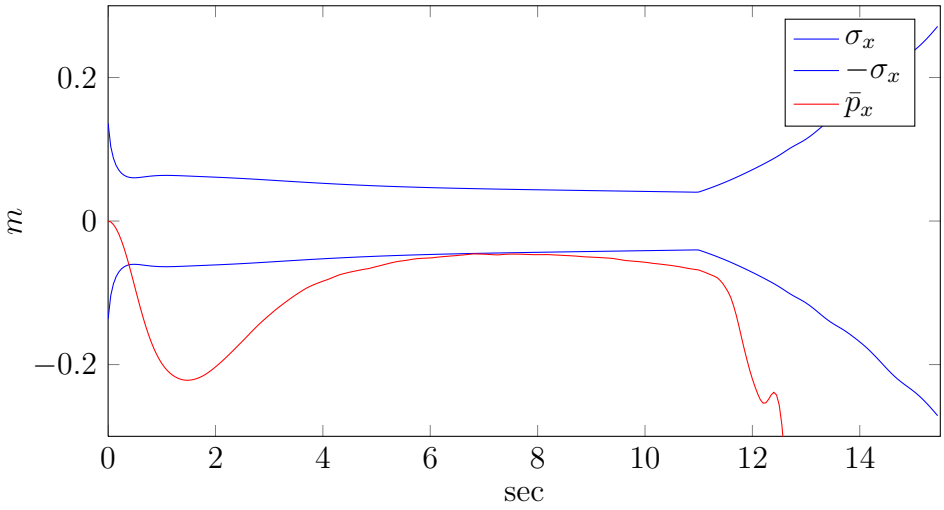


Figure 7.18: Standard Deviation vs X position plot

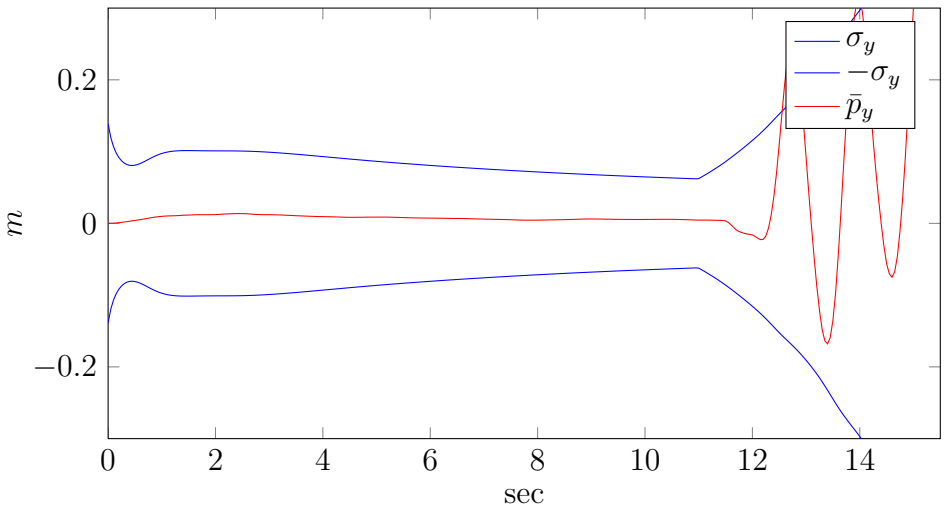


Figure 7.19: Standard Deviation vs Y position plot

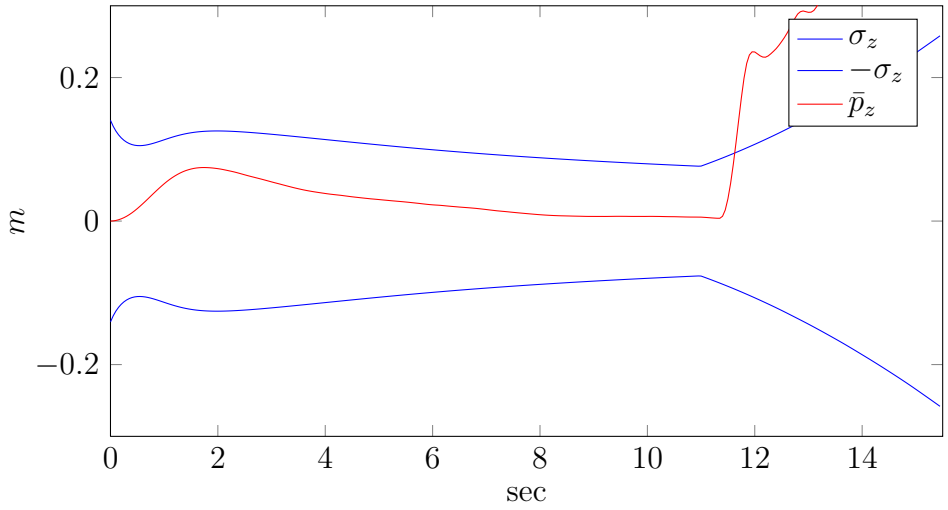


Figure 7.20: Standard Deviation vs Z position plot

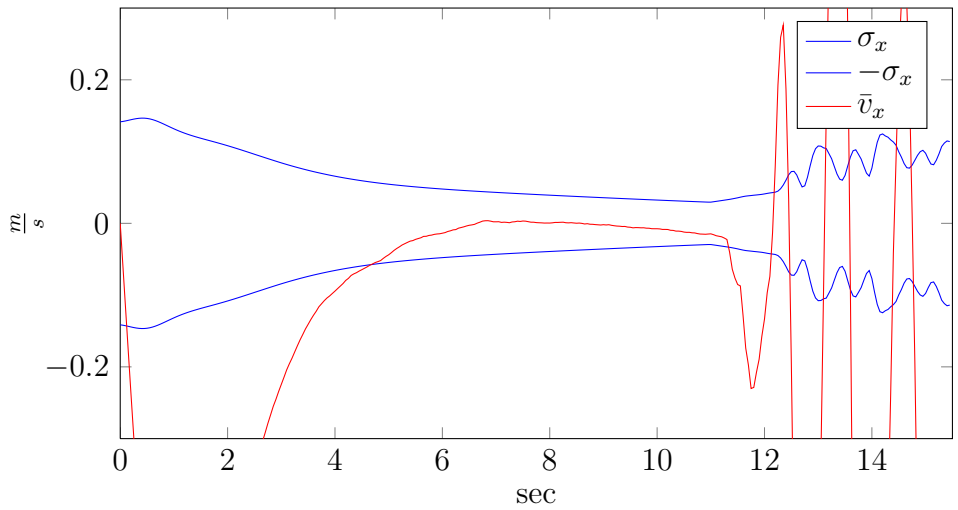


Figure 7.21: Standard Deviation vs X Velocity plot

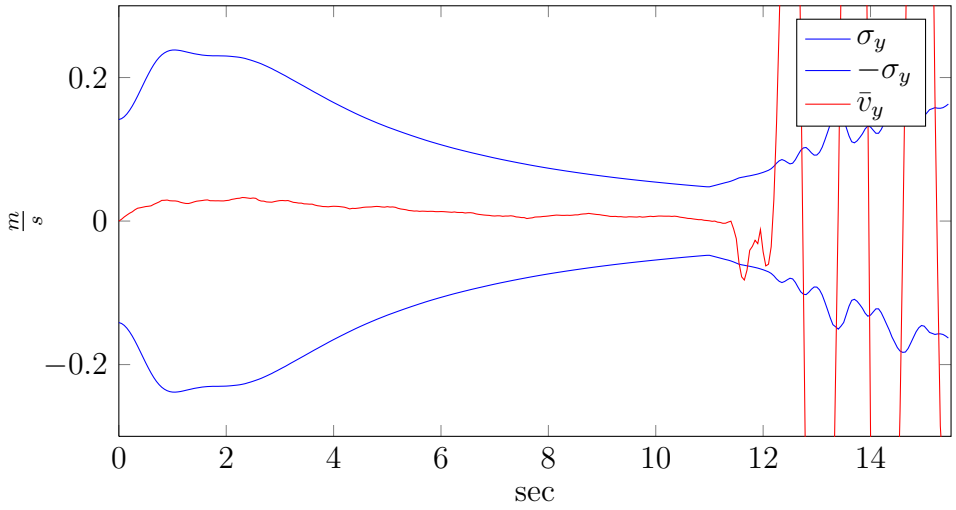


Figure 7.22: Standard Deviation vs Y Velocity plot

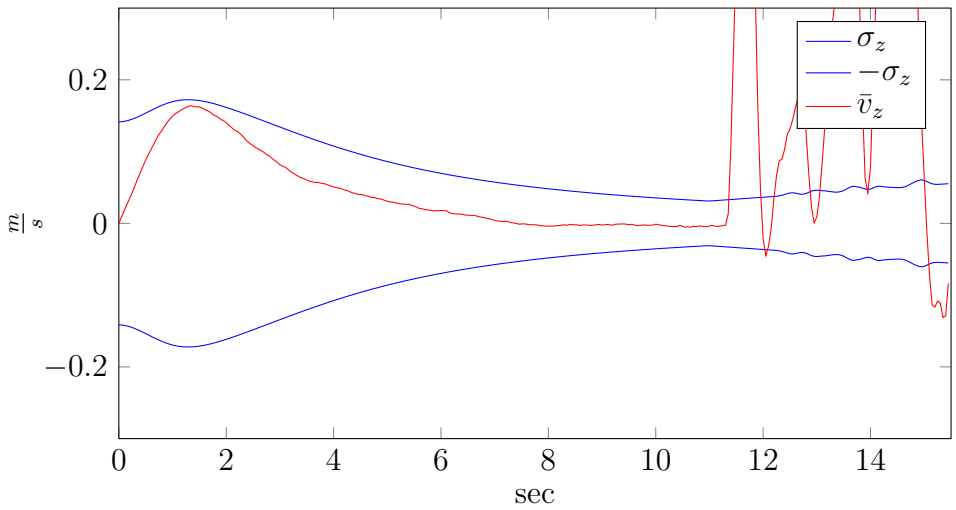


Figure 7.23: Standard Deviation vs Z Velocity plot

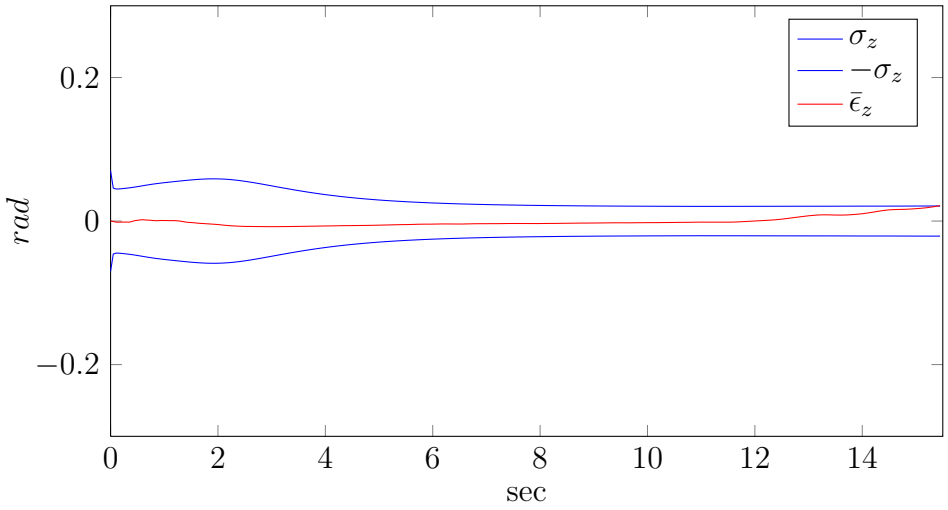


Figure 7.24: Standard Deviation vs ϵ_x Angulation plot

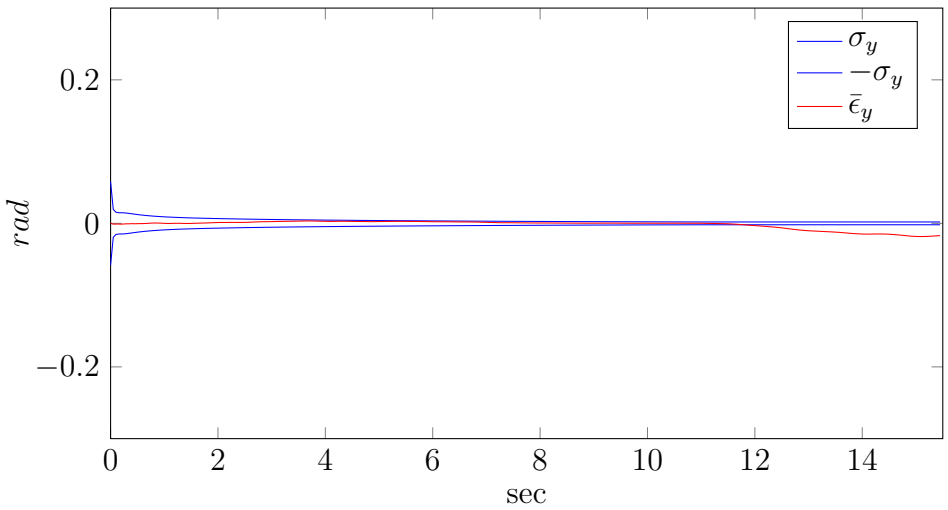


Figure 7.25: Standard Deviation vs ϵ_y Angulation plot

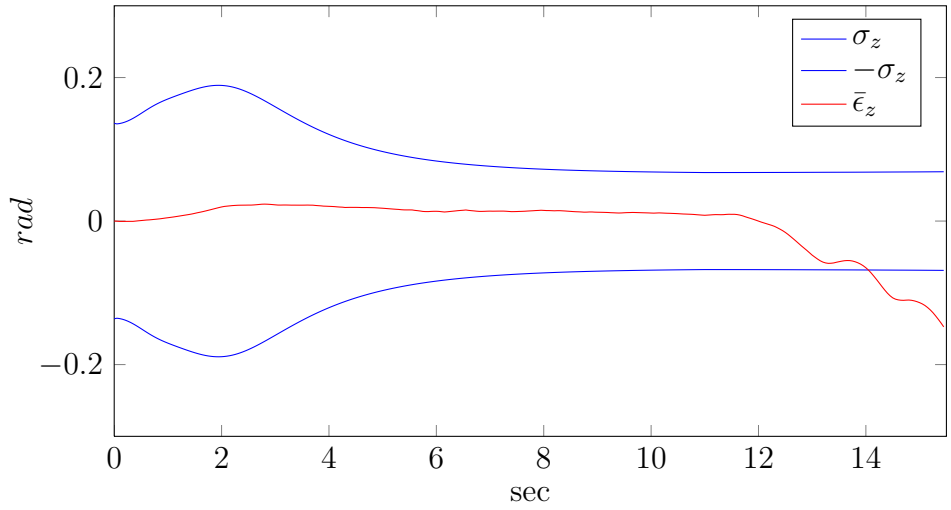


Figure 7.26: Standard Deviation vs ϵ_z Angulation plot

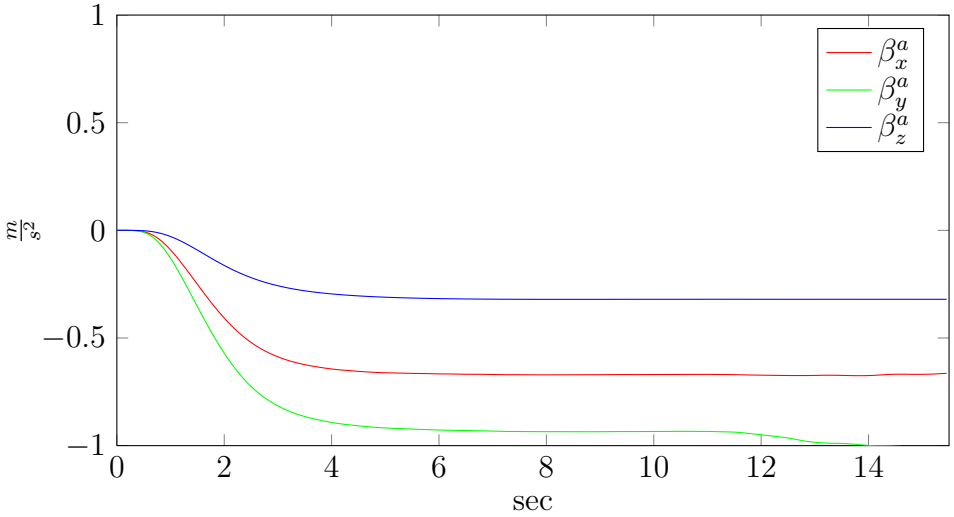


Figure 7.27: Accelerometers Bias

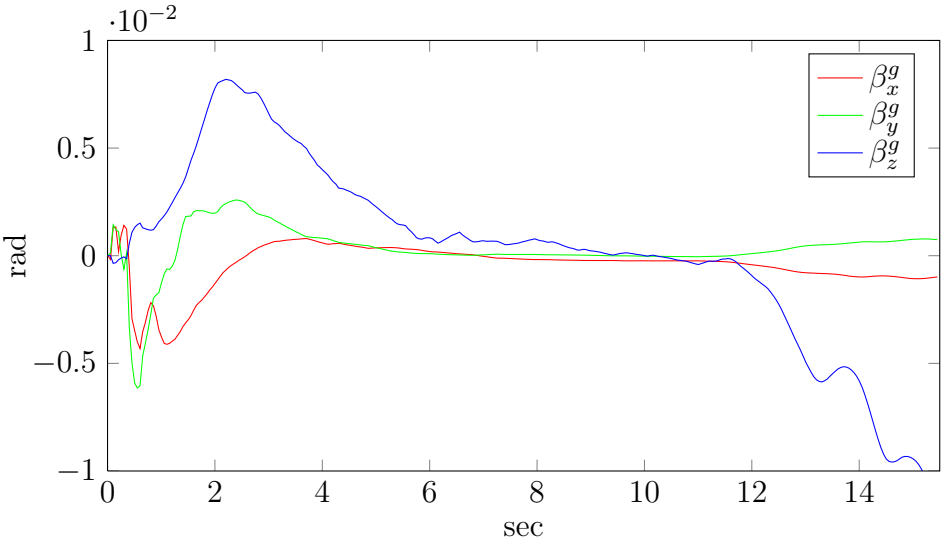


Figure 7.28: Gyroscope Bias

Chapter 8

Conclusion

In this project several approaches to the rotation matrix in the INS algorithm have been reviewed with performance in mind. Numerical integration routines have been applied to the different solutions which is also reviewed. The 9-element direct rotation matrix solved by the skew symmetric angular velocity multiplied with the rotation matrix was found to perform best. This requires the use of Heuns numerical integration routine.

A physical MEMS sensor was used to gather data required by the INS algorithm. A ARM Cortex-M3 microcontroller was used to sample data from the MEMS sensor at a data rate of 200Hz. Some MEMS noise parameters have been found by the use of the Allan variance method. These noise parameters was confirmed correlate with the data sheet. A linearised Kalman filter has been implemented to remove white noise and bias on the measurements. A drift of 2.079m was found over a 10 second run of the Kalman filter while the sensor was stationary. When comparing this to the 452.6m drift of the navigation equations a vast improvement is caused by the implemented Kalman filter.

The standard deviation for the position and angulation ϵ in the stationary measurement drift test are not as expected. The estimated value is too stable and indicate an error being present. The spirals test indicate that the filter is doing its job which is somewhat a contradiction though confirming this fact is somewhat a educated guess.

To summarise, the performance of a MEMS based INS was expected to be poor because of the many error sources associated with MEMS sensors. Though the MEMS sensors are acceptable over a short interval. This indicates that the introduction of aiding sensors for position which are stable over time should yield much better results. Depending on the platform this system is intended to be implemented on, constraints can be introduced in the filter. A example of this is a car, a car does not normally move sideways,

this constraint together with information on the speed and altitude from a barometer may be enough to stabilise the platform.

Chapter 9

Further work

There is indications of a error being present in the implementation of the Kalman filter, further investigation on where the source of error is should be the first priority. Using a rate table to find more noise and scale factors that increase the ability of the filter should be the second priority. Implementation of LKF on and microcrontroller is a demanding task balancing processing power requirements and accuracy which would be a fun extension to the assignment. Expanding the filter with more states which may increase the accuracy would be the next step after successful implementation on a MCU. Further investigation of the position performance should be done, together with a various aiding sensors and constraints dependent on the platform.

Bibliography

[1] Analog Devices. *Ten Degrees of Freedom Inertial Sensor ADIS16407*, bth edition, 7 11.

[2] U. M. Ascher and L. R. Petzold. *Computer Methods for Ordinary Differential Equations And Differential Algebraic Equations*. siam.

[3] G. J. J. Ducard. *Fault-tolerant Flight Control and Guidance Systems*. Springer.

[4] M. S. GREWAL and A. P. ANDREWS. *Kalman Filtering, Theory and Practice Using MATLAB*. Wiley.

[5] O. Hallingstad. Ekempler på tns-modeller. Lecture, 2003.

[6] O. Hallingstad. Forelesnings notater: Matematisk modelereing av dynamiske systemer. UNIK, 2009.

[7] O. Hallingstad. Sammendrag av modeller og likninger for kalmanfilteret anvendt på et ulinjært system. Lecture, 2009.

[8] M. A. Hopcroft. Modified allan. Matlab Central, 10 2010. ttp://www.mathworks.com/matlabcentral/fileexchange/26637-allanmodified.

[9] E. Kreyszig. *Advanced Engineering Mathematics*. Wiley.

[10] Matlab. World magnetic model function. Internet, Published in 2009. `http://www.mathworks.se/help/toolbox/aerotbx/ug/wrldmagm.html`.

[11] NASA. World magnetic model 2010. Internet, 2009.

[12] W. Riley. The calculation of time domain frequency stability. *IEEE*.

[13] R. M. Rogers. *Applied Mathematics in Integrated Navigation Systems*. AIAA.

[14] S. Sakka. Notes on quaternion. Centre of Excellence in Computational Complex Systems Research, 2007.

[15] P. A. Z. Syed and A. N. N. El-Sheimy. *MEMS-Based Integrated Navigation*. Artecch House.

[16] N. Trawny and S. I. Roumeliotis. Indirect kalman filter for 3d attitude estimation. University of Minnesota, 2007.

[17] Young and Freedman. *University Physics*. Pearson.

[18] P. Zarchan. *Fundamentals of Kalman Filtering: A Practical Approach*. American Institute of Aeronautics and Astronautics.

Appendix A

Code

A.1 Navigation Analysis Matlab Software

A.1.1 NavSim.m

```
%% Navigation Simulation
clear all; clc; close all;

%% Global Inital Conditions

Ts=0.01;            % Time step
Samples=5000;       % Number of samples
% Ts=1;             % Time step
% Samples=100;      % Number of samples
Te=Samples*Ts;      % End time
t=(1:Samples)*Ts;   % Time vector

r=100;              % Radius of circle
w=2*pi/(Te*0.5);    % Rotation speed
%w=0.6283;
w=0.2513
pos=[r 0 0]';       % Position in nav frame
vel=[0 sqrt(r^2*w^2) 0]';% Velocity in nav frame
Theta=[0 0 pi/2]'; % Attitude
F_b=[0 (r^2*w^2)/r 0]';% Force in body
w_b_nb=[0 0 w]';    % Rotation in body

%% dermenistic simulation
d.w=w*t;           % Rotation pos
d.w_dot=w;         % Rotation vel
d.w_ddot=0;        % Rotation acc

d.pos(1,:)=cos(d.w);
d.pos(2,:)=sin(d.w);
d.pos(3,:)=0;
d.pos=d.pos*r;
d.vel(1,:)=-sin(d.w)*d.w_dot;
d.vel(2,:)=cos(d.w)*d.w_dot;
d.vel(3,:)=0;
d.vel=d.vel*r;
d.acc(1,:)=(-cos(d.w)*d.w_dot*d.w_dot - sin(d.w)*d.w_ddot);
```

```
d.acc(2,:)=(-sin(d.w)*d.w_dot*d.w_dot - cos(d.w)*d.w_ddot);
d.acc(3,:)=-9.81;
d.acc=d.acc*r;

%% Eulers Method
x=[F_b,w_b_nb];

% 321 Euler angles
eul.y_321(:,1)=[pos;vel;Theta]; % Inital pos & vel
for k=1:Samples;
    eul.y_321(:,k+1) = eul.y_321(:,k) + feul( eul.y_321(:,k),x)*Ts;
end


% Rotation matrix
R=DCM([0 0 pi/2]);mx(1:3,1)=R(1,:); mx(4:6,1)=R(2,:); mx(7:9,1)=R(3,:);
eul.y_matrix(:,1)=[pos;vel;mx];
for k=1:Samples;
    eul.y_matrix(:,k+1) = eul.y_matrix(:,k) + fmatrix( eul.y_matrix(:,k),x)*Ts;
end

% Quaternion
q_init= [0.7071;0;0;0.7071];
eul.y_quat(:,1)=[pos;vel;q_init];
for k=1:Samples;
    eul.y_quat(:,k+1) = eul.y_quat(:,k) + fquat( eul.y_quat(:,k),x)*Ts;
end

%% Heuns Method

% 321 Euler angles
heu.y_321(:,1)=[pos;vel;Theta];
for k=1:Samples;
    heu.z_321(:,k+1) = heu.y_321(:,k) + feul( heu.y_321(:,k),x)*Ts;
    heu.y_321(:,k+1) = heu.y_321(:,k)+( (Ts/2) * (feul(heu.y_321(:,k),x)
                                        +feul(heu.z_321(:,k+1),x)) );
end

% Rotation matrix
R=DCM([0 0 pi/2]);mx(1:3,1)=R(1,:); mx(4:6,1)=R(2,:); mx(7:9,1)=R(3,:);
heu.y_matrix(:,1)=[pos;vel;mx];      % Inital pos & vel
for k=1:Samples;
    heu.z_matrix(:,k+1) = heu.y_matrix(:,k) + fmatrix( heu.y_matrix(:,k),x)*Ts;
    heu.y_matrix(:,k+1) = heu.y_matrix(:,k)+( (Ts/2) * (fmatrix(heu.y_matrix(:,k),x)
                                        +fmatrix(heu.z_matrix(:,k+1),x)) );
end

% Quaternion
q_init= [0.7071;0;0;0.7071];
heu.y_quat(:,1)=[pos;vel;q_init]; % Inital pos & vel
for k=1:Samples;
    heu.z_quat(:,k+1) = heu.y_quat(:,k) + fquat( heu.y_quat(:,k),x)*Ts;
    heu.y_quat(:,k+1) = heu.y_quat(:,k)+( (Ts/2) * (fquat(heu.y_quat(:,k),x)
                                        +fquat(heu.z_quat(:,k+1),x)) );
end

% Converting to euler angles for plot
for k=1:Samples;
    eul.a_321(:,k+1) = eul.y_321(7:end,k)*180/pi;
    eul.a_matrix(:,k+1) = R2e(eul.y_matrix(7:end,k));
    eul.a_quat(:,k+1) = q2e(eul.y_quat(7:end,k));
    heu.a_321(:,k+1) = heu.y_321(7:end,k)*180/pi;
```

```
99      heu.a_matrix(:,k+1) = R2e(heu.y_matrix(7:end,k));
100     heu.a_quat(:,k+1) = q2e(heu.y_quat(7:end,k));
101 end
102
103 % Calulating diff from correct value
104 for k=1:Samples
105     diffeul321(k)=norm(eul.y_321(1:3,k))-norm(d.pos(:,k));
106     diffheu321(k)=norm(heu.y_321(1:3,k))-norm(d.pos(:,k));
107     diffeulmatrix(k)=norm(eul.y_matrix(1:3,k))-norm(d.pos(:,k));
108     diffheumatrix(k)=norm(heu.y_matrix(1:3,k))-norm(d.pos(:,k));
109     diffeulquat(k)=norm(eul.y_quat(1:3,k))-norm(d.pos(:,k));
110     diffheuquat(k)=norm(heu.y_quat(1:3,k))-norm(d.pos(:,k));
111 end
112 %% Plot Results
113 figure(1)
114 clf
115 subplot(2,2,1), hold on, grid on, title('Position');
116     plot3(eul.y_321(1,:),eul.y_321(2,:),eul.y_321(3,:),'r')
117     plot3(eul.y_matrix(1,:),eul.y_matrix(2,:),eul.y_matrix(3,:),'g')
118     plot3(eul.y_quat(1,:),eul.y_quat(2,:),eul.y_quat(3,:),'b')
119     plot3(d.pos(1,1:end),d.pos(2,1:end),d.pos(3,1:end),'--k')
120 axis equal
121 subplot(2,2,2), hold on, grid on, title('Velocity');
122     plot3(eul.y_321(4,:),eul.y_321(5,:),eul.y_321(6,:),'r')
123     plot3(eul.y_matrix(4,:),eul.y_matrix(5,:),eul.y_matrix(6,:),'g')
124     plot3(eul.y_quat(4,:),eul.y_quat(5,:),eul.y_quat(6,:),'b')
125     plot3(d.vel(1,1:end),d.vel(2,1:end),d.vel(3,1:end),'--k')
126
127 subplot(2,2,3), hold on, grid on, title('Euler Angles');
128     plot(eul.a_321(3,:),'r')
129     plot(eul.a_matrix(3,:),'g')
130     plot(eul.a_quat(3,:),'b')
131
132 subplot(2,2,4), hold on, grid on, title('Diff pos');
133     plot(t,diffeul321,'r');
134     plot(t,diffeulmatrix,'g');
135     plot(t,diffeulquat,'b');
136
137 figure(2)
138 clf
139 subplot(2,2,1), hold on, grid on, title('Position');
140     plot3(heu.y_321(1,:),heu.y_321(2,:),heu.y_321(3,:),'--r')
141     plot3(heu.y_matrix(1,:),heu.y_matrix(2,:),heu.y_matrix(3,:),'--g')
142     plot3(heu.y_quat(1,:),heu.y_quat(2,:),heu.y_quat(3,:),'--b')
143     plot3(d.pos(1,1:end),d.pos(2,1:end),d.pos(3,1:end),'k')
144
145 subplot(2,2,2), hold on, grid on, title('Velocity');
146     plot3(heu.y_321(4,:),heu.y_321(5,:),heu.y_321(6,:),'--r')
147     plot3(heu.y_matrix(4,:),heu.y_matrix(5,:),heu.y_matrix(6,:),'--g')
148     plot3(heu.y_quat(4,:),heu.y_quat(5,:),heu.y_quat(6,:),'--b')
149     plot3(d.vel(1,1:end),d.vel(2,1:end),d.vel(3,1:end),'k')
150
151 subplot(2,2,3), hold on, grid on, title('Euler Angles');
152     plot(heu.a_321(3,:),'r')
153     plot(heu.a_matrix(3,:),'g')
154     plot(heu.a_quat(3,:),'b')
155
156 subplot(2,2,4), hold on, grid on, title('Diff pos');
157     plot(t,diffheu321,'r');
158     plot(t,diffheumatrix,'g');
159     plot(t,diffheuquat),'b';
160
```

```
161 %% Error Integration:
162 heu.err321=sum(abs(diffheu321))*Ts;
163 heu.errmatrix=sum(abs(diffheumatrix))*Ts;
164 heu.errquat=sum(abs(diffheuquat))*Ts;
165
166 eul.err321=sum(abs(diffeul321))*Ts;
167 eul.errmatrix=sum(abs(diffeulmatrix))*Ts;
168 eul.errquat=sum(abs(diffeulquat))*Ts;
169
170 fprintf('Euler Integrated absolute error\n')
171 fprintf('321: %f\t matrix: %f\t Quat: %f\n',eul.err321,eul.errmatrix,eul.errquat)
172 fprintf('Heun Integrated absolute error\n')
173 fprintf('321: %f\t matrix: %f\t Quat: %f\n',heu.err321,heu.errmatrix,heu.errquat)
```

A.1.2 fmatrix.m

```
function [ y ] = fmatrix( y_in,x )
% Constants
g=[0,0,0]';

% Input translation
Pos=y_in(1:3);
Vel=y_in(4:6);
R=vec2mat(y_in(7:15),3);
F_b=x(1:3)';
W_b_nb=x(4:6)';

% Position
Pos_dot=Vel;

% Velocity
Vel_dot=(R*F_b)-g;

% Attitude
R_dot=R*skew(W_b_nb);

% Output translation
y(1:3,1)=Pos_dot;
y(4:6,1)=Vel_dot;
y(7:9,1)=R_dot(1,:);
y(10:12,1)=R_dot(2,:);
y(13:15,1)=R_dot(3,:);
end

function [ S ] = skew( w )
%SKEW Takes a vector of 3 parameters and sets them in a skew matrix with 9
% elements (3x3)
S=  [
     0       -w(3)    w(2);
     w(3)     0      -w(1);
    -w(2)     w(1)    0;
    ];

end
```

A.1.3 feul.m

```
function [ y ] = feul( y_in,x )
% Constants
g=[0,0,0]';

% Input translation
Pos=y_in(1:3);
Vel=y_in(4:6);
Theta=y_in(7:9);
F_b=x(1:3)';
W_b_nb=x(4:6)';

% Position
Pos_dot=Vel;

% Velocity
Vel_dot=(DCM(Theta)*F_b)-g;

% Attitude
Theta_dot=D(Theta)*W_b_nb;

% Output translation
y(1:3,1)=Pos_dot;
y(4:6,1)=Vel_dot;
y(7:9,1)=Theta_dot;
end

function [ Matrix ] = D( Theta )
Matrix =[ 1    sin(Theta(1))*tan(Theta(2))    cos(Theta(1))*tan(Theta(2))
          0    cos(Theta(1))                  -sin(Theta(1))
          0    sin(Theta(1))/cos(Theta(2))    cos(Theta(1))/cos(Theta(2)) ];
end

function [ Theta_out ] = DCM(Th)
Theta_out=[
        cos(Th(3))*cos(Th(2)), ...
        cos(Th(3))*sin(Th(2))*sin(Th(1))-sin(Th(3))*cos(Th(1)), ...
        cos(Th(3))*sin(Th(2))*cos(Th(1))+sin(Th(3))*sin(Th(1)); ...
        sin(Th(3))*cos(Th(2)), ...
        sin(Th(3))*sin(Th(2))*sin(Th(1))+cos(Th(3))*cos(Th(1)), ...
        sin(Th(3))*sin(Th(2))*cos(Th(1))-cos(Th(3))*sin(Th(1));
        -sin(Th(2)), ...
        cos(Th(2))*sin(Th(1)), ...
        cos(Th(2))*cos(Th(1))];
end
```

A.1.4 fquat.m

```
function [ y ] = fquat( y_in,x )

% Constants
g=[0,0,0]';

% Input translation
Pos=y_in(1:3);
Vel=y_in(4:6);
q=y_in(7:10);

% Control input signal to system
F_b=x(1:3)';
W_b_ib=x(4:6)';

% Position
Pos_dot=Vel;

% Velocity
Vel_dot=(DCMquat(q)*F_b)-g;

% Attitude
q_dot=0.5*qrotvel(W_b_ib)*q;


% Output translation
y(1:3,1)=Pos_dot;
y(4:6,1)=Vel_dot;
y(7:10,1)=q_dot;


end

function [ Matrix ] = qrotvel( w )
Matrix = [
            0,      -w(1),      -w(2),      -w(3);
            w(1),   0,          w(3),       -w(2);
            w(2),   -w(3),      0,          w(1);
            w(3),   w(2),       -w(1),      0;
         ];
end

function [ C ] = DCMquat(q)

C = [
    (q(1)^2+q(2)^2-q(3)^2-q(4)^2), (2*(q(2)*q(3)-q(1)*q(4))), (2*(q(2)*q(4)+q(1)*q(3)));
    (2*(q(2)*q(3)+q(1)*q(4))), (q(1)^2-q(2)^2+q(3)^2-q(4)^2), (2*(q(3)*q(4)-q(1)*q(2)));
    (2*(q(2)*q(4)-q(1)*q(3))), (2*(q(3)*q(4)+q(1)*q(2))), (q(1)^2-q(2)^2-q(3)^2+q(4)^2);
    ];
end
```

A.1.5 q2e.m

```
function [ e ] = q2e( q )
% converts quaternions to euler angles

r2d=180/pi;
q0=q(1); q1=q(2); q2=q(3); q3=q(4);
e(1)=atan2(2*(q0*q1+q2*q3),1-2*(q1^2+q2^2)) * r2d;
```

```
e(2)=asin(2*(q0*q2-q3*q1))                  * r2d;
e(3)=atan2(2*(q0*q3+q1*q2),1-2*(q2^2+q3^2)) * r2d;
end
```

A.1.6 R2e.m

```
function [ e ] = R2e( R_in )
%R2E Summary of this function goes here
%   Detailed explanation goes here
r2d=180/pi;

R=vec2mat(R_in,3);
    if(R(3,1) ~= 1 || R(3,1) ~= -1 )
        y1 = -asin(R(3,1));
        y2 = pi - y1;
        x1 = atan2( R(3,2)/cos(y1), R(3,3)/cos(y1) );
        x2 = atan2( R(3,2)/cos(y2), R(3,3)/cos(y2) );
        z1 = atan2( R(2,1)/cos(y1), R(1,1)/cos(y1) );
        z2 = atan2( R(2,1)/cos(y2), R(1,1)/cos(y2) );
    else
        z1 = 0;
        if(R(3,1) == -1)
            y1 = pi/2;
            x1 = z1 + atan2(R(1,2),R(1,3));
        else
            y1 = -pi/2;
            x1 = -z1 + atan2(-R(1,2),-R(1,3));
        end
    end
e=[x1 y1 z1]*r2d;
end
```

A.2 Sensor Data Analysis Matlab Software

A.2.1 CalcAlan.m

```
%% Allan variance and auto correlation
clc, clear all, close all;

%% Load data
%load('3nov2h6m28s.mat')
load('7nov3.mat');
t.Ts=0.005;

%% Calculate time
t.time=length(data)*t.Ts;
t.days = round(t.time / 86400);
t.hours = round((t.time / 3600) - (t.days * 24));
t.minutes = round((t.time / 60) - (t.days * 1440) - (t.hours * 60));
t.seconds = round(mod(t.time,60));
disp(['Logg data length: ',num2str(t.hours),':',num2str(t.minutes),':',num2str(t.seconds)]);

%% Calculate allan variance
AllanD.rate=1/t.Ts;
AllanD.tau=[];

AllanD.freq=data(:,3)-mean(data(:,3));
[Gx.retval, Gx.s, Gx.errorb, Gx.tau] = allan(AllanD,AllanD.tau,'GyroX',0);

AllanD.freq=data(:,4)-mean(data(:,4));
[Gy.retval, Gy.s, Gy.errorb, Gy.tau] = allan(AllanD,AllanD.tau,'GyroY',0);

AllanD.freq=data(:,5)-mean(data(:,5));
[Gz.retval, Gz.s, Gz.errorb, Gz.tau] = allan(AllanD,AllanD.tau,'GyroZ',0);

AllanD.freq=data(:,6)-mean(data(:,6));
[Ax.retval, Ax.s, Ax.errorb, Ax.tau] = allan(AllanD,AllanD.tau,'AccX',0);

AllanD.freq=data(:,7)-mean(data(:,7));
[Ay.retval, Ay.s, Ay.errorb, Ay.tau] = allan(AllanD,AllanD.tau,'AccY',0);

AllanD.freq=data(:,8)-mean(data(:,8));
[Az.retval, Az.s, Az.errorb, Az.tau] = allan(AllanD,AllanD.tau,'AccZ',0);

clear AllanD;
%% Calculate ARW & Bias Instabiliti

Gx.BiasInstability = min(Gx.retval); % bias instability grader/sekund
Gy.BiasInstability = min(Gy.retval); % bias instability grader/sekund
Gz.BiasInstability = min(Gz.retval); % bias instability grader/sekund
Ax.BiasInstability = min(Ax.retval); % bias instability grader/sekund
Ay.BiasInstability = min(Ay.retval); % bias instability grader/sekund
Az.BiasInstability = min(Az.retval); % bias instability grader/sekund

Gx.ARW=Gx.retval(Gx.tau==1);
Gy.ARW=Gy.retval(Gy.tau==1);
Gz.ARW=Gz.retval(Gz.tau==1);
Ax.ARW=Ax.retval(Ax.tau==1);
Ay.ARW=Ay.retval(Ay.tau==1);
Az.ARW=Az.retval(Az.tau==1);

disp('Gyro parameters:')
disp(['Bias Instability: ',num2str(Gx.BiasInstability),...
```

```
                    ' ,',num2str(Gy.BiasInstability),...
                    ' ,',num2str(Gz.BiasInstability) ])
disp(['ARW: ',num2str(Gx.ARW),...
        ' ,',num2str(Gy.ARW),...
              ' ,',num2str(Gz.ARW) ])

disp('Acc parameters:')
disp(['Bias Instability: ',num2str(Ax.BiasInstability), ...
                    ' ,',num2str(Ay.BiasInstability), ...
                                        ' ,',num2str(Az.BiasInstability) ])
disp(['VRW: ',num2str(Ax.ARW), ...
        ' ,',num2str(Ay.ARW), ...
              ' ,',num2str(Az.ARW) ])

%% Plot results
figure;
subplot(2,2,1)
    loglog(Gx.tau,(Gx.retval),'.-r','LineWidth',2,'MarkerSize',14);
    hold on, grid on;
    loglog(Gy.tau,(Gy.retval),'.-b','LineWidth',2,'MarkerSize',14);
    loglog(Gz.tau,(Gz.retval),'.-g','LineWidth',2,'MarkerSize',14);
    legend('X-Gyro','Y-Gyro','Z-Gyro')
    alpha 0.1
subplot(2,2,2)
    hold on, grid on;
    t.auto=(-((length(data)-1)):((length(data)-1)))*t.Ts;
    plot(t.auto,xcorr(data(:,3)-mean(data(:,3))),'r')
    plot(t.auto,xcorr(data(:,4)-mean(data(:,4))),'b')
    plot(t.auto,xcorr(data(:,5)-mean(data(:,5))),'g')
    legend('X-Gyro','Y-Gyro','Z-Gyro')
subplot(2,2,3)
    loglog(Ax.tau,(Ax.retval),'.-r','LineWidth',2,'MarkerSize',14);
    hold on, grid on;
    loglog(Ay.tau,(Ay.retval),'.-b','LineWidth',2,'MarkerSize',14);
    loglog(Az.tau,(Az.retval),'.-g','LineWidth',2,'MarkerSize',14);
    legend('X-Acc','Y-Acc','Z-Acc')
subplot(2,2,4)
    hold on, grid on;
    t.auto=(-((length(data)-1)):((length(data)-1)))*t.Ts;
    plot(t.auto,xcorr(data(:,6)-mean(data(:,6))),'r')
    plot(t.auto,xcorr(data(:,7)-mean(data(:,7))),'b')
    plot(t.auto,xcorr(data(:,8)-mean(data(:,8))),'g')
    legend('X-Acc','Y-Acc','Z-Acc')
```

A.3 Kalman Filter

A.3.1 RunLKF.m

```
% Linearized kalman filter
clear all; clc;  close all;

%% Import data
%load('7nov3');
% load('spiral.mat')
% N=3100;
% PauseTime=2200;
load('DriftTest.mat')
N=8000;
PauseTime=6000;
%load('AnglesTest.mat')
acc=-data(:,6:8)';
gyr=data(:,3:5)';
mag=data(:,9:11)';

% Dataset Spesific variables + general
r2d=180/pi;
d2r=pi/180;
SampleFreq=200;
Ts=1/SampleFreq;

%% Normalize, bias removal Raw Data
for i=1:N
    normal(:,i)=norm(mag(:,i));
end
normal2=mean(normal);
for i=1:N
    mag(:,i)=mag(:,i)/normal2;
end
% Bias removal
BiasGyroX=mean(gyr(1,1:PauseTime));
BiasGyroY=mean(gyr(2,1:PauseTime));
BiasGyroZ=mean(gyr(3,1:PauseTime));
gyr(1,:)=gyr(1,:)-BiasGyroX;
gyr(2,:)=gyr(2,:)-BiasGyroY;
gyr(3,:)=gyr(3,:)-BiasGyroZ;


% Rename
fb=acc;
wb=gyr*d2r;
Bb=mag;

%% Plot Raw data
figure(5);
subplot(3,1,1); hold on, grid on;
    plot(gyr(1,:),'r'); plot(gyr(2,:),'g'); plot(gyr(3,:),'b');
    legend('GyrX','GyrY','GyrZ')
subplot(3,1,2); hold on, grid on;
    plot(acc(1,:),'r'); plot(acc(2,:),'g'); plot(acc(3,:),'b');
        legend('AccX','AccY','AccZ')
subplot(3,1,3); hold on, grid on;
    plot(mag(1,:),'r'); plot(mag(2,:),'g'); plot(mag(3,:),'b');
        legend('MagX','MagY','MagZ')

%% Initial Attitude
```

```
58  fn=[0,0,9.81]';
59  Bn=wrldmagm(0, 59.97487, 11.04532, 2011);
60  Bn=[Bn(1), -Bn(2), -Bn(3)]';
61
62  Bn=Bn/norm(Bn);
63
64  for i=1:1;
65  Rcb=[Bb(:,i)/norm(Bb(:,i)), ...
66      ( cross(Bb(:,i),fb(:,i)))/(norm(cross(Bb(:,i),fb(:,i))) ), ...
67      cross( Bb(:,i)/norm(Bb(:,i)) ,( cross(Bb(:,i),fb(:,i)))/(norm(cross(Bb(:,i),fb(:,i))) ) )];
68  Rcn=[Bn(:,1)/norm(Bn(:,1)), ...
69      ( cross(Bn(:,1),fn(:,1)))/(norm(cross(Bn(:,1),fn(:,1))) ), ...
70      cross( Bn(:,1)/norm(Bn(:,1)) ,( cross(Bn(:,1),fn(:,1)))/(norm(cross(Bn(:,1),fn(:,1))) ) )];
71  Rbn=Rcn*Rcb';
72  InitEuler=R2e(Rbn);
73  end
74
75  %% Kalman Init
76  Q=diag([0.0018564,0.0044847,0.0030709,0.029117,0.028674,0.027748]);
77  R=diag([(10.0609)^2/400, (14.7831^2)^2/400, (24.3630^2)^2/400  ...
78          10.0609/400     14.7831/400    24.3630/400 ...
79          0.8425      0.8030      0.7427, ...
80          0.00062394  0.0016138   0.00091252...% Acc Bias instab
81          0.0044451   0.0054394   0.0052201])*Ts; % Gyro bias instab
82  R=diag([(10)^2/400, (10^2)^2/400, (10^2)^2/400  ...
83          10/400     10/400    20/400 ...
84          0.8425      0.8030      0.7427, ...
85          0.00062394  0.0016138   0.00091252...% Acc Bias instab
86          0.0044451   0.0054394   0.0052201])*Ts; % Gyro bias instab
87  Pp=eye(15)*0.05*200;
88  Pe=eye(15)*0.05*200;
89  U=[fb;wb];
90  j=0;
91
92  % Preallocation
93  Xss=zeros(15,N);
94  Xs=zeros(15,N);
95  dXp=zeros(15,N);
96  dXe=zeros(15,N);
97  Xp=zeros(15,N);
98  Xe=zeros(15,N);
99  EulS=zeros(3,N);
100 EulP=zeros(3,N);
101 EulE=zeros(3,N);
102 Psave=zeros(15,15,N);
103
104 % More initialisation
105 Xs(:,1)=[zeros(6,1);Rbn(1,:)';Rbn(2,:)';Rbn(3,:)'];
106
107 %% Kalman Filter
108 for k=1:N-1
109     % Simulation of mecanisation (Heuns)
110     Xss(:,k+1) = Xs(:,k) + fmatrix( Xs(:,k),U(:,k)')*Ts;
111     Xs(:,k+1) = Xs(:,k)+( (Ts/2) * (fmatrix(Xs(:,k),U(:,k)')+fmatrix(Xss(:,k+1),U(:,k+1)')) );
112
113     % Rotation matrix from simulation
114     RbnS=(vec2mat(Xs(7:15,k+1),3));
115
116     % Conversion from rotation matrix to euler angles
117     EulS(:,k+1)=R2e(RbnS);
118
119     % Error model
```

```
120     F=[ zeros(3,3), eye(3), zeros(3,3), zeros(3,3), zeros(3,3);
121         zeros(3,3), zeros(3,3), -skew( RbnS*(fb(:,k)) ), RbnS, zeros(3,3);
122         zeros(3,3), zeros(3,3), zeros(3,3), zeros(3,3), RbnS;
123         zeros(3,3), zeros(3,3), zeros(3,3), zeros(3,3), zeros(3,3);
124         zeros(3,3), zeros(3,3), zeros(3,3), zeros(3,3), zeros(3,3);];
125     G=[ zeros(3,3), zeros(3,3);
126         RbnS,        zeros(3,3);
127         zeros(3,3), RbnS;
128         zeros(3,3), zeros(3,3);
129         zeros(3,3), zeros(3,3)];
130
131     % Discretisation
132     Fi=expm(F*Ts);
133     S=k2dS(F,G,Q,Ts);
134
135     % Error update
136     dXp(:,k+1)=Fi*dXp(:,k); % dXp=dXe if applicable!
137
138     % Correction
139     Xp(:,k+1)= [Xs(1:6,k+1);zeros(9,1)] + dXp(:,k+1);
140     RbnP=( (eye(3) + skew(dXp(7:9,k+1)))*RbnS);
141
142     % Conversion from rotation matrix to euler angles
143     EulP(:,k)=R2e(RbnP);
144
145     % Predicted Covariance
146     Pp=Fi*Pp*Fi'+S; % Pe=Pp if applicable!
147
148     j=j+1;
149      if (ismember(j,[1:N,N-1]))
150         if(k<PauseTime)
151             % Measurement Matrix
152             H=zeros(15,15);
153             H(1:3,1:3)=eye(3);
154             %H(4:6,4:6)=eye(3);
155             H(7:9,7:9)=RbnS'*skew(Bn);
156
157             % Kalman Gain Matrix
158             K=Pp*H'/(H*Pp*H'+R);
159
160             % Estimated Covariance
161             Pe=(eye(15)-K*H)*Pp;
162
163             % Measurements
164             Z=[zeros(1,3), zeros(1,3), Bb(:,k)', zeros(1,3), zeros(1,3)]';
165             %Zs=[Xs(1:3,k)', Xs(4:6,k)', (RbnS'*Bn)', zeros(1,3), zeros(1,3)]';
166             Zs=[Xs(1:3,k)', zeros(1,3), (RbnS'*Bn)', zeros(1,3), zeros(1,3)]';
167             dZ=Z-Zs;
168
169             % Error update
170             dXe(:,k+1)= dXp(:,k+1) + K*(dZ - H*dXp(:,k+1));
171
172             % Correction
173             Xe(:,k+1)= [Xs(1:6,k+1);zeros(9,1)]+dXe(:,k+1);
174             RbnE=( (eye(3) + skew(dXe(7:9,k+1)))*RbnS );
175
176             % Conversion from rotation matrix to euler angles
177             EulE(:,k)=R2e(RbnE);
178
179             % Feedback to prediction
180             Pp=Pe;
181             dXp(:,k+1)=dXe(:,k+1);
```

```
182             if (k+1>PauseTime)
183                 Xs(7:15,k+1)=[RbnE(1,:)';RbnE(2,:)';RbnE(3,:)'];
184                 Xe(7:9,k+1)=zeros(3,1);
185                 Xp(7:9,k+1)=zeros(3,1);
186                 dXe(7:9,k+1)=zeros(3,1);
187                 dXp(7:9,k+1)=zeros(3,1);
188             end
189         else
190
191             % Measurement Matrix
192             H=zeros(15,15);
193             H(7:9,7:9)=RbnS'*skew(Bn);
194
195             % Kalman Gain Matrix
196             K=Pp*H'/(H*Pp*H'+R);
197
198             % Estimated Covariance
199             Pe=(eye(15)-K*H)*Pp;
200
201             % Measurements
202             Z=[zeros(1,3), zeros(1,3), Bb(:,k)', zeros(1,3), zeros(1,3)]';
203             Zs=[zeros(1,3), zeros(1,3), (RbnS'*Bn)', zeros(1,3), zeros(1,3)]';
204             dZ=Z-Zs;
205
206             % Error update
207             dXe(:,k+1)= dXp(:,k+1) + K*(dZ - H*dXp(:,k+1));
208
209             % Correction
210             Xe(:,k+1)= [Xs(1:6,k+1);zeros(9,1)]+dXe(:,k+1);
211             RbnE=( (eye(3) + skew(dXe(7:9,k+1)))*RbnS );
212
213             % Conversion from rotation matrix to euler angles
214             EulE(:,k)=R2e(RbnE);
215
216             % Feedback to prediction
217             Pp=Pe;
218             dXp(:,k+1)=dXe(:,k+1);
219         end
220      end
221     Psave(:,:,k)=Pp;
222     Pdiag(:,k)=diag(Pp);
223 end
224 %% Display results
225 disp(['Gyro Bias Adjustment-----------------------------------------------'])
226 disp(['x = ',num2str(BiasGyroX),' deg/sec'])
227 disp(['y = ',num2str(BiasGyroY),' deg/sec'])
228 disp(['z = ',num2str(BiasGyroZ),' deg/sec'])
229
230 disp(['Timing-------------------------------------------------------------'])
231 disp(['Total Dataset Time = ',num2str(N*Ts),' sec'])
232 disp(['Init time          = ',num2str(PauseTime*Ts),' sec'])
233 disp(['Free Run time      = ',num2str((N-PauseTime)*Ts),' sec']);
234
235 disp(['Initial Attitude:--------------------------------------------------'])
236 disp(['Roll  = ',num2str(InitEuler(1)),' deg'])
237 disp(['Pitch = ',num2str(InitEuler(2)),' deg'])
238 disp(['Yaw   = ',num2str(InitEuler(3)),' deg'])
239
240 disp(['Errors-------------------------------------------------------------'])
241 disp(['Navgation eq Drift = ',num2str(norm(Xs(1:3,N))-norm(Xs(1:3,PauseTime)))])
242 disp(['Total Drift = ',num2str(norm(Xp(1:3,N)))])
243 disp(['Drift Per sec = ',num2str(norm(Xp(1:3,N))/((N-PauseTime)*Ts))])
```

```
244
245 %% Plot Results
246 figure(1) % 3D Position Plot
247     hold on, grid on, view(3);
248     %plot3(Xs(1,PauseTime:N)-Xs(1,PauseTime),Xs(2,PauseTime:N)-Xs(2,PauseTime),Xs(3,PauseTime:N)-Xs(3,PauseTime),'b--');
249     %plot3(Xe(1,PauseTime:N),Xe(2,PauseTime:N),Xs(3,PauseTime:N),'r');
250     plot3(Xp(1,PauseTime:N),Xp(2,PauseTime:N),Xp(3,PauseTime:N),'k--');
251     title('Position'),          xlabel('x'),          ylabel('y'), zlabel('z'), axis equal
252
253 figure(2) % Attitude
254         hold on, grid on;
255         XMIN=0;XMAX=N*Ts;YMIN=-180;YMAX=180;
256         subplot(3,1,1)
257         hold on, grid on; title('Roll')
258                 stairs(((1:N)*Ts),EulS(1,:),'b--');
259                 stairs(((1:N)*Ts),EulE(1,:),'r');
260                 stairs(((1:N)*Ts),EulP(1,:),'g');
261                 legend('Sim','Est','Pred')
262                 %axis([XMIN XMAX YMIN YMAX])
263         subplot(3,1,2)
264         hold on, grid on; title('Pitch')
265                 stairs(((1:N)*Ts),EulS(2,:),'b--');
266                 stairs(((1:N)*Ts),EulE(2,:),'r');
267                 stairs(((1:N)*Ts),EulP(2,:),'g');
268                 legend('Sim','Est','Pred')
269                 %axis([XMIN XMAX YMIN YMAX])
270         subplot(3,1,3)
271         hold on, grid on; title('Yaw')
272                 stairs(((1:N)*Ts),EulS(3,:),'b--');
273                 stairs(((1:N)*Ts),EulE(3,:),'r');
274                 stairs(((1:N)*Ts),EulP(3,:),'g');
275                 legend('Sim','Est','Pred')
276                 %axis([XMIN XMAX YMIN YMAX])
277
278 figure(3) % Standard deviation Plots
279 name=[  'p        ';
280         'p        ';
281         'p        ';
282         'v        ';
283         'v        ';
284         'v        ';
285         '\epsilon ';
286         '\epsilon ';
287         '\epsilon '];
288 for i = 1:9
289         subplot(3,3,i)
290         hold on, grid on,
291                 plot(((0:N-2)*1/200),sqrt(Pdiag(i,:)),'b')
292                 plot(((0:N-2)*1/200),-sqrt(Pdiag(i,:)),'b')
293                 plot(((0:N-1)*1/200),Xp(i,:),'r')
294                 % stairs(t2,Xe(i,2:200:end),'g')
295                 %legend(['P',num2str(i)],['-P',num2str(i)],['Xp',num2str(i)],['Xe',num2str(i)])
296                 title(name(i,:))
297                 hold off
298 end
299 figure(4) % Bias Plots
300         subplot(2,1,1),hold on, grid on;
301                 title('Accelerometer Bias')
302                 plot(((1:N)*Ts),Xp(10,:),((1:N)*Ts),Xp(11,:),((1:N)*Ts),Xp(12,:))
303                 legend('X','Y','Z')
304         subplot(2,1,2),hold on, grid on;
305                 title('Gyro Bias')
```

```
306                 plot(((1:N)*Ts),Xp(13,:),((1:N)*Ts),Xp(14,:),((1:N)*Ts),Xp(15,:))
307                 legend('X','Y','Z')
```

A.3.2 k2dS.m

```
1  function S  = k2dS(F, G, Q_tilde, d)
2      [m,n] = size(F);
3      A = [F          G*Q_tilde*G';
4          zeros(m,n)  -F'         ]*d;
5      B = expm(A);
6      S = B(1:n,n+1:2*n) * inv(B(n+1:2*n,n+1:2*n));
```

A.3.3 skew.m

```
1  function [ S ] = skew( w )
2  %SKEW Takes a vector of 3 parameters and sets them in a skew matrix with 9
3  % elements (3x3)
4  S=  [
5       0       -w(3)    w(2);
6       w(3)     0      -w(1);
7      -w(2)     w(1)    0;
8      ];
9
10 end
```

A.3.4 fmatrix.m

```
1  function [ y ] = fmatrix( y_in,x )
2  % Constants
3  g=[0,0,9.81]';
4
5  % Input translation
6  Pos=y_in(1:3);
7  Vel=y_in(4:6);
8  R=vec2mat(y_in(7:15),3);
9  F_b=x(1:3)';
10 W_b_nb=x(4:6)';
11
12 % Position
13 Pos_dot=Vel;
14
15 % Velocity
16 Vel_dot=(R*F_b)-g;
17
18 % Attitude
19 R_dot=R*skew(W_b_nb);
20
21 % Output translation
22 y(1:3,1)=Pos_dot;
23 y(4:6,1)=Vel_dot;
24 y(7:9,1)=R_dot(1,:);
25 y(10:12,1)=R_dot(2,:);
26 y(13:15,1)=R_dot(3,:);
27 end
```

A.3.5 DCM.m

```
function [ Theta_out ] = DCM(Th)
Theta_out=[
    cos(Th(3))*cos(Th(2)), ...
        cos(Th(3))*sin(Th(2))*sin(Th(1))-sin(Th(3))*cos(Th(1)), ...
        cos(Th(3))*sin(Th(2))*cos(Th(1))+sin(Th(3))*sin(Th(1)); ...
    sin(Th(3))*cos(Th(2)), ...
        sin(Th(3))*sin(Th(2))*sin(Th(1))+cos(Th(3))*cos(Th(1)), ...
        sin(Th(3))*sin(Th(2))*cos(Th(1))-cos(Th(3))*sin(Th(1)); ...
    -sin(Th(2)), ...
        cos(Th(2))*sin(Th(1)), ...
        cos(Th(2))*cos(Th(1)); ...
    ];
end
```

A.4 C++ Code MCU & PC

A.4.1 mbed.cpp

A.4.2 DCM.m

```
#include "mbed.h" // Mbed Library Header
SPI ADIS16407(p5, p6, p7); // MISO MOSI SCK
DigitalOut SS(p8); // Slave Select
DigitalOut RST(p9); // ADIS Reset pin
Serial pc(USBTX, USBRX); // USB USART tunnel
DigitalOut L1(LED1); // Debug Led
DigitalOut L2(LED1); // Debug Led
Ticker timerINT; // Ticker timer interrupt obj
Timer timestamp; // Timer timestamp obj

union ByteSplit {
    int16_t int16;
    int8_t int8[2];
};

void WriteSerialINT(int16_t temp) {
    ByteSplit split; // Create ByteSplit object
    split.int16 = temp; // Insert temp into int16
    pc.putc(split.int8[1]); // Write MSBs of int16 to USART
    pc.putc(split.int8[0]); // Write LSBs of int16 to USART
}

bool Reading=0;
void ReadData() {
    if (!Reading) { // Error detection
        Reading=1; // Detection Flag
        SS=0; // Slave Select (Active Low)
        L1= !L1; // Toggle LED1
        WriteSerialINT(timestamp.read_ms()); // Timestamp
        ADIS16407.write(0x4200); // Initiate ADIS Burst Read
        for (int i=1; i<=14; i++) { // Read ADIS
            WriteSerialINT(ADIS16407.write(0x0000)); // Transmit Measurement data
        }
        SS=1; // Realease Slave
        Reading=0; // Ready for new Sample frame
    }
}

void ConfADIS() {
    SS=0; // Slave Select (Active Low)
    wait_us(1); // Waiting for device receive SS
    ADIS16407.write(0xBB02); // 204.8Hz decimation
    SS=1; // Release Salve
    wait_ms(1); // Wait for ADIS process
}

void Initalize() {
    pc.baud(115200); // Set baud rate
    ADIS16407.format(16,3); // Set SPI bus with and Mode
    ADIS16407.frequency(1000000); // Set SPI bus freq
    SS=1; // Initial Slave select (Active Low)
    RST=1; // Reset (Active Low)
    // Writing 0xFF00 to PC for start of transmission mark
    for (int j=0; j<15; j++) {
```

```
55          WriteSerialINT(0xFF00);
56      }
57      ConfADIS(); // Setup ADIS
58      // Timer interrupt setup
59      timerINT.attach(&ReadData,0.005); // Ts=0.005->200Hz
60      // Timestamp reset and start
61      timestamp.reset();
62      timestamp.start();
63  }
64  int main() {
65      Initalize(); // Run init
66      while (1); // noop on free time =)
67  }
```

A.4.3 Decode.cpp

```
1   #include <iostream>
2   #include <fstream>
3
4   using namespace std;
5   ifstream::pos_type size;
6   char * memblock;
7   float convert[]={       0.002418,
8                                       0.05, 0.05, 0.05,
9                                       0.0326673, 0.0326673, 0.0326673,
10                                      0.0005, 0.0005, 0.0005,
11                                      0.00008, 0.0000003125,
12                                      0.136, 0.0008059};
13
14  union ByteSplit
15  {
16      int16_t int16;
17      int8_t int8[2];
18  };
19
20  int main () {
21
22          // Input file
23          ifstream inFile ("data.bin", ios::in|ios::binary|ios::ate);// Define input & read mode
24
25          if (inFile.is_open())// Make shure file is open
26          {
27                  size = inFile.tellg(); // Get size
28                  memblock = new char [size]; // Allocate char array
29                  inFile.seekg (0, ios::beg); // Set pointer pos at beginning
30                  inFile.read (memblock, size); // Read Whole content into memory
31                  inFile.close(); // Close file
32          } else {
33                  cout << "Unable to open file";
34                  return 1;
35          }
36
37          // Output file
38          FILE * OutFile;
39          OutFile = fopen ("data.txt","w");
40
41          // Conversion algorithem
42          int filesize = (int)size; // Typecast file size to integer (files bigger than 2GB may cause problems)
43
44          int start = 30;
```

```
45          int l=0;
46          ByteSplit splitter;
47          int16_t temp;
48          int16_t stamp;
49
50          for(int j=start; j<filesize; j=j+30){
51                  if(start+j>filesize) break;
52                  splitter.int8[1]=(unsigned char)memblock[(j)];
53                  splitter.int8[0]=(unsigned char)memblock[(j+1)];
54                  stamp=splitter.int16;
55
56                  fprintf(OutFile, "%f\t",(float)stamp);
57                  l = 0;
58                  for(int k=2; k<=28; k=k+2){
59                          splitter.int8[1]=(unsigned char)memblock[(j+k)];
60                          splitter.int8[0]=(unsigned char)memblock[(j+k+1)];
61                          temp=splitter.int16&0x3FFF;
62                          if ( ((temp&0x2000)>>13) && (k!=20) && (k!=22) && (k!=24)) {
63                  temp=(temp-16383);
64              } else if(k==24){
65                       temp=(temp&0x0FFF)-4095;
66              }
67                          fprintf(OutFile, "%f\t", convert[l]*(float)(temp));
68                          l++;
69
70          }
71                  fprintf(OutFile, "\r\n");
72          }
73
74      delete[] memblock;
75          fclose (OutFile);
76          printf("\r\n");
77    return 0;
78  }
```

Appendix B

Oddvar Hallingstad Lecture Notes

Eksempler på TNS-modeller

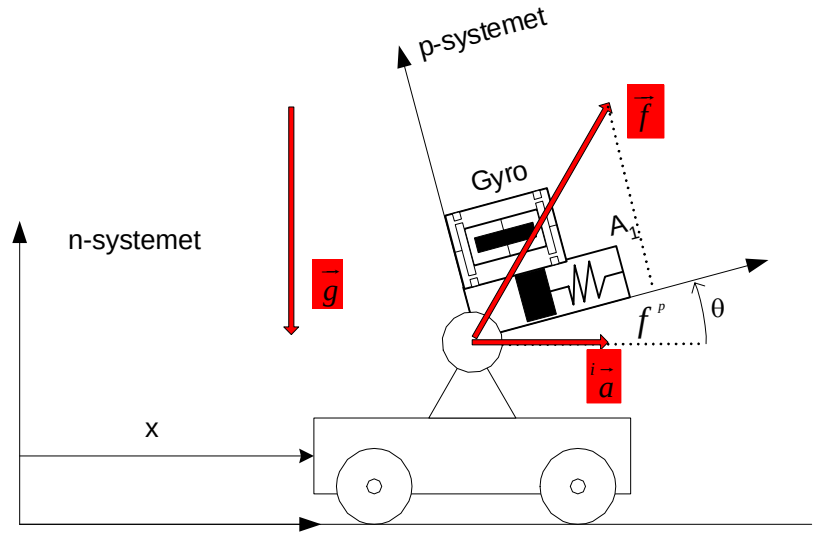
Oddvar Hallingstad

3. mars 2004

---

1 En-akset plattform

Figuren nedenfor viser en en-akset plattform (den måler akselerasjonen bare langs en akse) som kan dreie seg om en akse vinkelrett på papirplanet. Denne dreiningen måles av en gyro.



Figur 1: Enakset plattform

Feildefinisjoner		
$\delta f = f - \tilde{f}$	$\delta x = x - \tilde{x}$	$\delta\theta = \theta - \tilde{\theta}$
$\delta\omega = \omega - \tilde{\omega}$	$\delta v = v - \tilde{v}$	

Legg merke til definisjonene av feil. De er valgt på denne måten fordi dette gir en standard systemmodell for et eventuelt Kalmanfilter. Målingene $\tilde{f}$ og $\tilde{\omega}$ kan ses på som pådrag. Dersom vi måler posisjonen for bruk i et Kalmanfilter som måling må feilmodellen være $y = \tilde{x} = x + w$ hvor w er hvit støy.

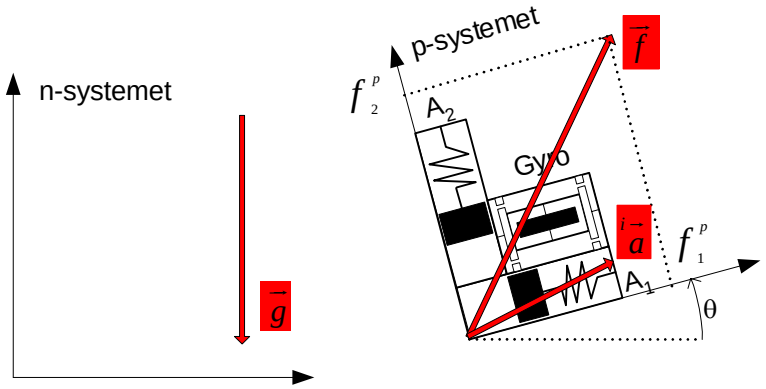
Fysisk system	Mekanisert system	Feillikninger
$\dot{x} = v$	$\dot{\tilde{x}} = \tilde{v}$	$\delta\dot{x} = \delta v$
$\dot{v} = \frac{f - g\sin\theta}{\cos\theta}$	$\dot{\tilde{v}} = \frac{\tilde{f} - g\sin\tilde{\theta}}{\cos\tilde{\theta}}$	$\delta\dot{v} = \frac{1}{\cos^2\tilde{\theta}}\left(\tilde{f}\sin\tilde{\theta} - g\right)\delta\theta + \frac{1}{\cos\tilde{\theta}}\delta f$
$\dot{\theta} = \omega$	$\dot{\tilde{\theta}} = \tilde{\omega}$	$\delta\dot{\theta} = \delta\omega$

Bare likningen for $\delta\dot{v}$ krever en egen utledning:

$$\delta\dot{v} = \frac{f - g\sin\theta}{\cos\theta} - \frac{\tilde{f} - g\sin\tilde{\theta}}{\cos\tilde{\theta}} = \frac{\tilde{f} + \delta f - g\sin\left(\tilde{\theta} + \delta\theta\right)}{\cos\left(\tilde{\theta} + \delta\theta\right)} - \frac{\tilde{f} - g\sin\tilde{\theta}}{\cos\tilde{\theta}} = \frac{\delta f}{\cos\tilde{\theta}} + \left(\frac{\tilde{f}\sin\tilde{\theta}}{\cos^2\tilde{\theta}} - g\left(1 + \tan^2\theta\right)\right)\delta\theta + O\left(\delta\theta^2, \delta f^2\right)$$

$$= \frac{1}{\cos^2\tilde{\theta}}\left(\tilde{f}\sin\tilde{\theta} - g\right)\delta\theta + \frac{1}{\cos\tilde{\theta}}\delta f + O\left(\delta\theta^2, \delta f^2\right)\ \delta\dot{v} = \frac{f - g\sin\theta}{\cos\theta} - \frac{\tilde{f} - g\sin\tilde{\theta}}{\cos\tilde{\theta}}$$

2 To-akset plattform



Figur 2: To-akset plattform

Feildefinisjoner		
$\delta \underline{f} = \underline{f}^p - \underline{\tilde{f}}^p$	$\delta \underline{x} = \underline{x}^n - \underline{\tilde{x}}^n$	$\delta\theta = \theta - \tilde{\theta}$
$\delta\omega = \omega - \tilde{\omega}$	$\delta \underline{v} = \underline{v}^n - \underline{\tilde{v}}^n$	$R_p^n = R\left(\tilde{\theta}\right)\left(I + \delta\theta \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\right)$

Den elementære rotasjonsmatrisa R er gitt ved

$$R_p^n = R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Feillikningene skrevet på vektorform:

Fysisk system	Mekanisert system	Feillikninger
$\dot{\underline{x}}^n = \underline{v}^n$	$\dot{\underline{\tilde{x}}}^n = \underline{\tilde{v}}^n$	$\delta\dot{\underline{x}} = \delta \underline{v}$
$\dot{\underline{v}}^n = R_p^n \underline{f}^p - \underline{g}^n$	$\dot{\underline{\tilde{v}}}^n = \tilde{R}_p^n \underline{\tilde{f}}^p - \underline{g}^n$	$\delta\dot{\underline{v}}^n = R_3\left(\tilde{\theta}\right)\begin{bmatrix} -\tilde{f}_y \\ \tilde{f}_x \end{bmatrix}\delta\theta + R_3\left(\tilde{\theta}\right)\delta \underline{f}$
$\dot{\theta} = \omega$	$\dot{\tilde{\theta}} = \tilde{\omega}$	$\delta\dot{\theta} = \delta\omega$

Feilikningene skrevet på komponentform:

Fysisk system	Mekanisert system	Feillikninger
$\dot{\underline{x}} = \underline{v}$	$\dot{\underline{\tilde{x}}} = \underline{\tilde{v}}$	$\delta\dot{\underline{x}} = \delta \underline{v}$
$\dot{v}_1 = f_x\cos\theta - f_y\sin\theta$	$\dot{\tilde{v}}_1 = \tilde{f}_x\cos\tilde{\theta} - \tilde{f}_y\sin\tilde{\theta}$	$\delta\dot{v}_1 = -\left(\tilde{f}_x\sin\tilde{\theta} + \tilde{f}_y\cos\tilde{\theta}\right)\delta\theta$ $+\delta f_x\cos\tilde{\theta} - \delta f_y\sin\tilde{\theta}$
$\dot{v}_2 = f_x\sin\theta + f_y\cos\theta - g$	$\dot{\tilde{v}}_2 = \tilde{f}_x\sin\tilde{\theta} + \tilde{f}_y\cos\tilde{\theta} - g$	$\delta\dot{v}_2 = \left(\tilde{f}_x\cos\tilde{\theta} - \tilde{f}_y\sin\tilde{\theta}\right)\delta\theta$ $+\delta f_x\sin\tilde{\theta} + \delta f_y\cos\tilde{\theta}$
$\dot{\theta} = \omega$	$\dot{\tilde{\theta}} = \tilde{\omega}$	$\delta\dot{\theta} = \delta\omega$

Utledning av likningen for hastighetsfeil:

$$\delta\dot{\underline{v}}^n = R_p^n \underline{f}^p - \tilde{R}_p^n \underline{\tilde{f}}^p = R\left(\tilde{\theta}\right)\left(I + \delta\theta\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\right)\left(\underline{\tilde{f}}^p + \delta \underline{f}\right) - R\left(\tilde{\theta}\right)\underline{\tilde{f}}^p == R\left(\tilde{\theta}\right)\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\underline{\tilde{f}}^p\delta\theta +$$
$$R\left(\tilde{\theta}\right)\delta \underline{f} = R\left(\tilde{\theta}\right)\begin{bmatrix} -\tilde{f}_y \\ \tilde{f}_x \end{bmatrix}\delta\theta + R\left(\tilde{\theta}\right)\delta \underline{f}$$

3 Tre-akset plattform, flat ikke-roterende jord

For en tre-akset plattform kan feilen defineres på flere måter.

3.1 Transformasjonsfeilen referert eulervinklene

Feildefinisjoner		
$\delta \underline{f} = \underline{f}^p - \underline{\tilde{f}}^p$	$\delta \underline{x}^n = \underline{x}^n - \underline{\tilde{x}}^n$	$\delta \underline{\theta} = \underline{\theta} - \underline{\tilde{\theta}}$
$\delta \underline{\omega} = \underline{\omega}_{np}^p - \underline{\tilde{\omega}}_{np}^p$	$\delta \underline{v}^n = \underline{v}^n - \underline{\tilde{v}}^n$	

Jeg har et eget notat for utledningen her.

3.2 Tranformasjonsfeilen referert n-systemet

Vi skal her se på feillikningene når feilen i beregningen av rotasjonsmatrisa R_p^n refereres til n-systemet.

Feildefinisjoner		
$\delta \underline{f} = \underline{f}^p - \underline{\tilde{f}}^p$	$\delta \underline{x}^n = \underline{x}^n - \underline{\tilde{x}}^n$	$R_p^n = R\left(\underline{\varepsilon}\right) \tilde{R}_p^n$
$\delta \underline{\omega} = \underline{\omega}_{np}^p - \underline{\tilde{\omega}}_{np}^p$	$\delta \underline{v}^n = \underline{v}^n - \underline{\tilde{v}}^n$	$R\left(\underline{\varepsilon}\right) = I + S\left(\underline{\varepsilon}\right)$

Fysisk system	Mekanisert system	Feillikninger
$\dot{\underline{x}}^n = \underline{v}^n$	$\dot{\underline{\tilde{x}}}^n = \underline{\tilde{v}}^n$	$\delta \dot{\underline{x}} = \delta \underline{v}$
$\dot{\underline{v}}^n = R_p^n \underline{f}^p - \underline{g}^n$	$\dot{\underline{\tilde{v}}}^n = \tilde{R}_p^n \underline{\tilde{f}}^p - \underline{g}^n$	$\delta \dot{\underline{v}}^n = -S\left(\tilde{R}_p^n \underline{\tilde{f}}^p\right) \underline{\varepsilon} + \tilde{R}_p^n \delta \underline{f}$
$\dot{R}_p^n = R_p^n S\left(\underline{\omega}_{np}^p\right)$	$\dot{\tilde{R}}_p^n = \tilde{R}_p^n S\left(\underline{\tilde{\omega}}_{np}^p\right)$	$\dot{\underline{\varepsilon}} = \tilde{R}_p^n \delta \underline{\omega}$

Bevis av d.l. for $\delta \underline{v}^n$:

$\delta \dot{\underline{v}}^n = R_p^n \underline{f}^p - \underline{g}^n - \left(\tilde{R}_p^n \underline{\tilde{f}}^p - \underline{g}^n\right) = \left(I + S\left(\underline{\varepsilon}\right)\right) \tilde{R}_p^n \left(\underline{\tilde{f}}^p + \delta \underline{f}\right) - \tilde{R}_p^n \underline{\tilde{f}}^p = S\left(\underline{\varepsilon}\right) \tilde{R}_p^n \underline{\tilde{f}}^p + \tilde{R}_p^n \delta \underline{f}$

$=$

Bevis av d.l. for $\underline{\varepsilon}$:

V. siden: $\dot{R}_p^n - \dot{\tilde{R}}_p^n = S\left(\dot{\underline{\varepsilon}}\right) \tilde{R}_p^n + \left(I + S\left(\underline{\varepsilon}\right)\right) \dot{\tilde{R}}_p^n - \dot{\tilde{R}}_p^n = S\left(\dot{\underline{\varepsilon}}\right) \tilde{R}_p^n + S\left(\underline{\varepsilon}\right) \dot{\tilde{R}}_p^n = S\left(\dot{\underline{\varepsilon}}\right) \tilde{R}_p^n + S\left(\underline{\varepsilon}\right) \tilde{R}_p^n S\left(\underline{\tilde{\omega}}_{np}^p\right)$

H. siden: $R_p^n S\left(\underline{\omega}_{np}^p\right) - \tilde{R}_p^n S\left(\underline{\tilde{\omega}}_{np}^p\right) = \left(I + S\left(\underline{\varepsilon}\right)\right) \tilde{R}_p^n \left(S\left(\underline{\tilde{\omega}}_{np}^p\right) + S\left(\delta \underline{\omega}\right)\right) - \tilde{R}_p^n S\left(\underline{\tilde{\omega}}_{np}^p\right) = S\left(\underline{\varepsilon}\right) \tilde{R}_p^n S\left(\underline{\tilde{\omega}}_{np}^p\right) + \tilde{R}_p^n S\left(\delta \underline{\omega}\right)$

Satt sammen:

$S\left(\dot{\underline{\varepsilon}}\right) \tilde{R}_p^n + S\left(\underline{\varepsilon}\right) \tilde{R}_p^n S\left(\underline{\tilde{\omega}}_{np}^p\right) = S\left(\underline{\varepsilon}\right) \tilde{R}_p^n S\left(\underline{\tilde{\omega}}_{np}^p\right) + \tilde{R}_p^n S\left(\delta \underline{\omega}\right)$

$S\left(\dot{\underline{\varepsilon}}\right) \tilde{R}_p^n = \tilde{R}_p^n S\left(\delta \underline{\omega}\right)$

$S\left(\dot{\underline{\varepsilon}}\right) = \tilde{R}_p^n S\left(\delta \underline{\omega}\right) \tilde{R}_n^p \Leftrightarrow \dot{\underline{\varepsilon}} = \tilde{R}_p^n \delta \underline{\omega}$

3.3 Transformasjonsfeilen referert p-systemet

Vi skal her se på feillikningene når feilen i beregningen av rotasjonsmatrisa R_p^n refereres til p-systemet.

Feildefinisjoner		
$\delta \underline{f} = \underline{f}^p - \underline{\tilde{f}}^p$	$\delta \underline{x}^n = \underline{x}^n - \underline{\tilde{x}}^n$	$R_p^n = \tilde{R}_p^n R\left(\underline{\varepsilon}\right)$
$\delta \underline{\omega} = \underline{\omega}_{np}^p - \underline{\tilde{\omega}}_{np}^p$	$\delta \underline{v}^n = \underline{v}^n - \underline{\tilde{v}}^n$	$R\left(\underline{\varepsilon}\right) = I + S\left(\underline{\varepsilon}\right)$

Fysisk system	Mekanisert system	Feillikninger
$\underline{\dot{x}}^n = \underline{v}^n$	$\dot{\underline{\tilde{x}}}^n = \underline{\tilde{v}}^n$	$\delta\underline{\dot{x}} = \delta\underline{v}$
$\underline{\dot{v}}^n = R_p^n \underline{f}^p - \underline{g}^n$	$\dot{\underline{\tilde{v}}}^n = \tilde{R}_p^n \underline{\tilde{f}}^p - \underline{g}^n$	$\delta\underline{\dot{v}}^n = -\tilde{R}_p^n S\left(\underline{\tilde{f}}^p\right)\underline{\varepsilon} + \tilde{R}_p^n \delta\underline{f}$
$\dot{R}_p^n = R_p^n S\left(\underline{\omega}_{np}^p\right)$	$\dot{\tilde{R}}_p^n = \tilde{R}_p^n S\left(\underline{\tilde{\omega}}_{np}^p\right)$	$\underline{\dot{\varepsilon}} = -S\left(\underline{\tilde{\omega}}\right)\underline{\varepsilon} + \delta\underline{\omega}$

Bevis av d.l. for $\delta\underline{v}^n$:

$\delta\underline{\dot{v}}^n = R_p^n \underline{f}^p - \underline{g}^n - \left(\tilde{R}_p^n \underline{\tilde{f}}^p - \underline{g}^n\right) = \tilde{R}_p^n \left(I + S\left(\underline{\varepsilon}\right)\right)\left(\underline{\tilde{f}}^p + \delta\underline{f}\right) - \tilde{R}_p^n \underline{\tilde{f}}^p = \tilde{R}_p^n S\left(\underline{\varepsilon}\right)\underline{\tilde{f}}^p + \tilde{R}_p^n \delta\underline{f}$

$= -\tilde{R}_p^n S\left(\underline{\tilde{f}}^p\right)\underline{\varepsilon} + \tilde{R}_p^n \delta\underline{f}$

Bevis av d.l. for $S\left(\underline{\dot{\varepsilon}}\right)$:

V. siden: $\dot{R}_p^n - \dot{\tilde{R}}_p^n = \dot{\tilde{R}}_p \left(I + S\left(\underline{\varepsilon}\right)\right) + \tilde{R}_p^n S\left(\underline{\dot{\varepsilon}}\right) - \dot{\tilde{R}}_p^n = \dot{\tilde{R}}_p S\left(\underline{\varepsilon}\right) + \tilde{R}_p^n S\left(\underline{\dot{\varepsilon}}\right) = \tilde{R}_p^n S\left(\underline{\tilde{\omega}}_{np}^p\right) S\left(\underline{\varepsilon}\right) + \tilde{R}_p^n S\left(\underline{\dot{\varepsilon}}\right)$

H. siden: $R_p^n S\left(\underline{\omega}_{np}^p\right) - \tilde{R}_p^n S\left(\underline{\tilde{\omega}}_{np}^p\right) = \tilde{R}_p^n \left(I + S\left(\underline{\varepsilon}\right)\right)\left(S\left(\underline{\tilde{\omega}}_{np}^p\right) + S\left(\delta\underline{\omega}\right)\right) - \tilde{R}_p^n S\left(\underline{\tilde{\omega}}_{np}^p\right)$

$= \tilde{R}_p^n S\left(\underline{\varepsilon}\right) S\left(\underline{\tilde{\omega}}_{np}^p\right) + \tilde{R}_p^n S\left(\delta\underline{\omega}\right)$

Satt sammen:

$\tilde{R}_p^n S\left(\underline{\tilde{\omega}}_{np}^p\right) S\left(\underline{\varepsilon}\right) + \tilde{R}_p^n S\left(\underline{\dot{\varepsilon}}\right) = \tilde{R}_p^n S\left(\underline{\varepsilon}\right) S\left(\underline{\tilde{\omega}}_{np}^p\right) + \tilde{R}_p^n S\left(\delta\underline{\omega}\right)$

$S\left(\underline{\dot{\varepsilon}}\right) = S\left(\underline{\varepsilon}\right) S\left(\underline{\tilde{\omega}}_{np}^p\right) - S\left(\underline{\tilde{\omega}}_{np}^p\right) S\left(\underline{\varepsilon}\right) + S\left(\delta\underline{\omega}\right)$

$$S_\delta = \begin{bmatrix} 0 & -\delta_z & \delta_y \\ \delta_z & 0 & -\delta_x \\ -\delta_y & \delta_x & 0 \end{bmatrix}, \quad S_\omega = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}, \quad S_\varepsilon = \begin{bmatrix} 0 & -\varepsilon_z & \varepsilon_y \\ \varepsilon_z & 0 & -\varepsilon_x \\ -\varepsilon_y & \varepsilon_x & 0 \end{bmatrix}$$

$$S_\varepsilon S_\omega - S_\omega S_\varepsilon + S_\delta = \begin{bmatrix} 0 & \varepsilon_y\omega_x - \varepsilon_x\omega_y - \delta_z & \varepsilon_z\omega_x - \varepsilon_x\omega_z + \delta_y \\ \varepsilon_x\omega_y - \varepsilon_y\omega_x + \delta_z & 0 & \varepsilon_z\omega_y - \varepsilon_y\omega_z - \delta_x \\ \varepsilon_x\omega_z - \varepsilon_z\omega_x - \delta_y & \varepsilon_y\omega_z - \varepsilon_z\omega_y + \delta_x & 0 \end{bmatrix}$$

D.l. for $\underline{\varepsilon}$ skrevet ut:

$$\begin{aligned} \dot{\varepsilon}_x &= \varepsilon_y\tilde{\omega}_z - \varepsilon_z\tilde{\omega}_y + \delta\tilde{\omega}_x \\ \dot{\varepsilon}_y &= \varepsilon_z\tilde{\omega}_x - \varepsilon_x\tilde{\omega}_z + \delta\tilde{\omega}_y \\ \dot{\varepsilon}_z &= \varepsilon_x\tilde{\omega}_y - \varepsilon_y\tilde{\omega}_x + \delta\tilde{\omega}_z \end{aligned}$$

Dette kan også skrives

$$\underline{\dot{\varepsilon}} = -S\left(\underline{\tilde{\omega}}\right)\underline{\varepsilon} + \delta\underline{\omega}$$

	SAMMENDRAG AV MODELLER OG LIKNINGER FOR KALMANFILTERET ANVENT PÅ ET ULINEÆRT SYSTEM			
$\mathcal{S}^{US}$	**Sann ulineær systemmodell**	$\dot{\underline{x}} = \underline{f}(\underline{x},\underline{u}) + G\underline{v}$ $\underline{z}_k = \underline{h}_k(\underline{x}_k) + \underline{w}_k$	$\underline{x}_0 \sim \mathcal{N}(\hat{\underline{x}}_0, \hat{P}_0),\ \underline{v}_k \sim N(\underline{0}, \tilde{Q}\delta(t-\tau))$ $\underline{w}_k \sim \mathcal{N}(\underline{0}, R\delta_{kl})$	$E\left\{\underline{x}_0\underline{w}_k^T\right\} = 0,\ E\left\{\underline{x}_0\underline{v}^T(t)\right\} = 0$ $E\left\{\underline{v}(t)\,\underline{w}_k^T\right\} = 0$
$\mathcal{F}^{UD}$	**Ulineær det. filtermodell**	$\dot{\tilde{\underline{x}}} = \underline{f}^*(\tilde{\underline{x}},\underline{u})$ $\tilde{\underline{z}}_k = \underline{h}_k^*(\tilde{\underline{x}}_k)$	$\tilde{\underline{x}}_0 = T\hat{\underline{x}}_0$ $T = \left[\begin{array}{cc} I & 0 \end{array}\right]$	
$\mathcal{S}^{LS}$	**Sann lineær feilmodell**	$\delta\dot{\underline{x}} = F(\tilde{\underline{x}},t)\,\delta\underline{x} + G\underline{v}$ $\delta\underline{z}_k = H_k(\tilde{\underline{x}}) + \underline{w}_k$	$\delta\underline{x}_0 \sim \mathcal{N}(\hat{\underline{x}}_0 - T^T\tilde{\underline{x}}_0, \hat{P}_0),\ \underline{v} \sim \mathcal{N}(\underline{0}, \tilde{Q}\delta(t-\tau))$ $\underline{w}_k \sim \mathcal{N}(\underline{0}, R\delta_{kl})$	$E\left\{\delta\underline{x}_0.\underline{w}_k^T\right\} = 0,\ E\left\{\delta\underline{x}_0.\underline{v}^T(t)\right\} = 0$ $E\left\{\underline{v}(t).\underline{w}_k^T\right\} = 0$
$\mathcal{F}^{LS}$	**Lineær filtermodel**	$\delta\dot{\underline{x}}^* = F^*(\tilde{\underline{x}},t) + G^*\underline{v}^*$ $\delta\underline{z}_k = H^*(\tilde{\underline{x}}) + \underline{w}^*$	$\delta\underline{x}_0^* \sim \mathcal{N}(\hat{\underline{x}}_0^*, \hat{P}_0^*),\ \underline{v}^* \sim N(\underline{0}, \tilde{Q}^*\delta(t-\tau))$ $\underline{w}_k^* \sim \mathcal{N}(\underline{0}, R^*\delta_{kl})$	$E\left\{\underline{x}_0^*.\underline{w}_k^{*T}\right\} = 0,\ E\left\{\underline{x}_0^*.\underline{v}^{*T}(t)\right\} = 0$ $E\left\{\underline{v}^*(t).\underline{w}_k^{*T}\right\} = 0$
	Dimensjoner	$\dim\underline{x} = \dim\delta\underline{x} = n_x \geq \dim\tilde{\underline{x}} =$	$\dim\delta\underline{x}^* = n_{x^*}$	
	Filter:	**TKF-Tilbakekoblet Kalmanfilter**	**LKF-Linearisert Kalmanfilter**	**UKF-Utvidet Kalmanfilter**
IV	**Initialv.** $t = \hat{t}_0^+$	$\tilde{\underline{x}}\left(\hat{t}_0^+\right) = \tilde{\underline{x}}_0, \delta\bar{\underline{x}}_0 = \underline{0},\ \bar{P}_0$: gitt	$\tilde{\underline{x}}(\hat{t}_0^+) = \tilde{\underline{x}}_0, \delta\bar{\underline{x}}_0 = 0,\ \bar{P}_0$: gitt	$\bar{\underline{x}}_0 = \tilde{\underline{x}}(\hat{t}_0^+) = \tilde{\underline{x}}_0,\ \bar{P}_0$: gitt
MO	**Måle-oppdatering**	$\delta\underline{z}_k = \underline{z}_k - \tilde{\underline{z}}_k$ $\delta\hat{\underline{x}}_k = \delta\bar{\underline{x}}_k + K_k^*(\delta\underline{z}_k - H_k^*\delta\bar{\underline{x}}_k)$	$\delta\underline{z}_k = \underline{z}_k - \tilde{\underline{z}}_k$ $\delta\hat{\underline{x}}_k = \delta\bar{\underline{x}}_k + K_k^*(\delta\underline{z}_k - H_k^*\delta\bar{\underline{x}}_k)$ $\hat{\underline{x}}_k = \tilde{\underline{x}}_k + \delta\hat{\underline{x}}_k$	$\hat{\underline{x}}_k = \bar{\underline{x}}_k + K_k^*(\underline{z}_k - \underline{h}_k^*(\bar{\underline{x}}_k))$
		$K_k^* = \bar{P}_k H_k^{*T}(H_k\bar{P}_k H_k^{*T} + R_k^*)^{-1}$	$\hat{P}_k = (I - K_k^*H_k^*)\bar{P}_k$	
TK	**Tilbakekobl. ved** $t = \hat{t}_k^+$	$\hat{\underline{x}}_k^+ = \tilde{\underline{x}}_k + S^*\delta\hat{\underline{x}}_k$ $\delta\hat{\underline{x}}_k^+ = (I - S^*)\,\delta\hat{\underline{x}}_k$ $\hat{P}_k^+ = \hat{P}_k$		
TO	**Tids-oppdatering**	$\dot{\tilde{\underline{x}}} = \underline{f}^*(\tilde{\underline{x}},\underline{u}),\ \tilde{\underline{x}}(\hat{t}_k^+) = \hat{\underline{x}}_k^+,\ \ t \in \left[\hat{t}_k^+, \bar{t}_{k+1}\right]$ $\delta\dot{\bar{\underline{x}}}(t) = F^*(t)\cdot\delta\bar{\underline{x}}(t);\ \delta\bar{\underline{x}}(\hat{t}_k^+) = \delta\hat{\underline{x}}_k^+$ $\bar{\underline{x}}(t) = \tilde{\underline{x}}(t) + \delta\bar{\underline{x}}(t)$	$\dot{\tilde{\underline{x}}} = \underline{f}^*(\tilde{\underline{x}},\underline{u}),\ \tilde{\underline{x}}(\hat{t}_k) = \tilde{\underline{x}}\left(\bar{t}_k\right),\ \ t \in \left[\hat{t}_k, \bar{t}_{k+1}\right]$ $\delta\dot{\bar{\underline{x}}}(t) = F^*(t)\cdot\delta\bar{\underline{x}}(t);\ \delta\bar{\underline{x}}(\hat{t}_k) = \delta\hat{\underline{x}}_k$ $\bar{\underline{x}}(t) = \tilde{\underline{x}}(t) + \delta\bar{\underline{x}}(t)$	$\dot{\bar{\underline{x}}} = \underline{f}^*(\bar{\underline{x}},\underline{u}),\ \bar{\underline{x}}(\hat{t}_k) = \hat{\underline{x}}_k,\ \ t \in \left[\hat{t}_k, \bar{t}_{k+1}\right]$
		$\dot{\bar{P}}(t) = F^*(t)\bar{P}(t) + \bar{P}(t)F^{*T}(t) + \ G^*(t)$	$\tilde{Q}^*(t)G^{*T}(t);\quad \bar{P}\left(\hat{t}_k\right) = \hat{P}_k$	
		LKF ($S^* = 0$) og UKF ($S^* = I$) er	spesialtilfeller av TKF	

Ten Degrees of Freedom Inertial Sensor

ADIS16407

FEATURES

Triaxial digital gyroscope with digital range scaling
±75°/sec, ±150°/sec, ±300°/sec settings
Axis-to-axis alignment, <0.05°
Triaxial digital accelerometer, ±18 *g* minimum
Triaxial digital magnetometer, ±2.5 gauss minimum
Digital barometer, 10 mbar to 1200 mbar
Calibrated pressure range: 300 mbar to 1100 mbar
Autonomous operation and data collection
No external configuration commands required
210 ms start-up time, 4 ms sleep mode recovery time
Factory calibrated sensitivity, bias, and axial alignment
Calibration temperature range: −40°C to +85°C
SPI-compatible serial interface
Embedded temperature sensor
Programmable operation and control
Automatic and manual bias correction controls
Bartlett window FIR length, number of taps
Digital I/O: data ready, alarm indicator, general-purpose
Alarms for condition monitoring
Sleep mode for power management
DAC output voltage
Enable external sample clock input up to 1.1 kHz
Single command self test
Single-supply operation: 4.75 V to 5.25 V
2000 *g* shock survivability
Operating temperature range: −40°C to +105°C

APPLICATIONS

Platform stabilization and control
Navigation
Robotics

GENERAL DESCRIPTION

The ADIS16407 *i*Sensor® device is a complete inertial system that includes a triaxial gyroscope, a triaxial accelerometer, a triaxial magnetometer, and pressure sensors. Each sensor in the ADIS16407 combines industry-leading *i*MEMS® technology with signal conditioning that optimizes dynamic performance. The factory calibration characterizes each sensor for sensitivity, bias, alignment, and linear acceleration (gyro bias). As a result, each sensor has its own dynamic compensation formulas that provide accurate sensor measurements.

The ADIS16407 provides a simple, cost-effective method for integrating accurate, multiaxis inertial sensing into industrial systems, especially when compared with the complexity and investment associated with discrete designs. All necessary motion testing and calibration are part of the production process at the factory, greatly reducing system integration time. Tight orthogonal alignment simplifies inertial frame alignment in navigation systems. The SPI and register structure provide a simple interface for data collection and configuration control.

The ADIS16407 has a compatible pinout for systems that currently use ADIS1635x, ADIS1636x, and ADIS1640x IMU products. The ADIS16407 is packaged in a module that is approximately 23 mm × 23 mm × 23 mm and has a standard connector interface.

FUNCTIONAL BLOCK DIAGRAM

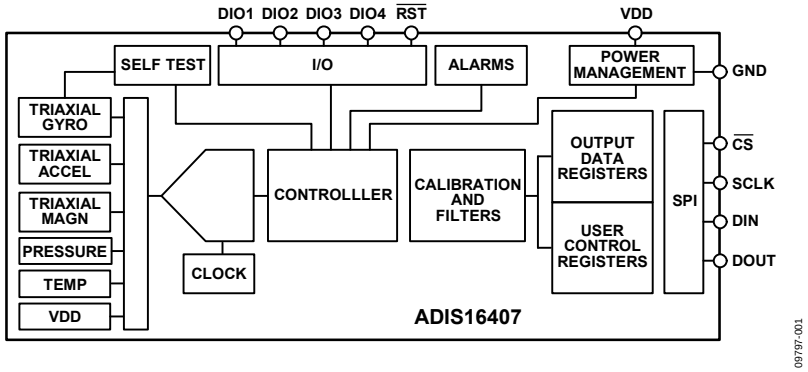


Figure 1.

Rev. B

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700 **www.analog.com**
Fax: 781.461.3113

TABLE OF CONTENTS

REVISION HISTORY

7/11—Rev. A to Rev. B

Change to Table 1, Barometer, Sensitivity Parameter ... 4
Added Barometer Section; Changes to Table 40 ... 16
Changes to Table 55, Table 56, Table 57 ... 20
Changes to Table 58, Table 59, Table 60 ... 21

6/11—Rev. 0 to Rev. A

Changes to Device Configuration Section and Figure 16 ... 13
Changes to Figure 19 ... 18
Changes to Figure 25 Caption ... 24
Changes to Ordering Guide ... 24

4/11—Revision 0: Initial Version

SPECIFICATIONS

T_A = 25°C, VDD = 5 V, angular rate = 0°/sec, dynamic range = ±300°/sec ± 1 *g*, unless otherwise noted.

Table 1.

Parameter	Test Conditions/Comments	Min	Typ	Max	Unit
GYROSCOPES					
Dynamic Range		±300	±350		°/sec
Initial Sensitivity	Dynamic range = ±300°/sec	0.0495	0.05	0.0505	°/sec/LSB
	Dynamic range = ±150°/sec		0.025		°/sec/LSB
	Dynamic range = ±75°/sec		0.0125		°/sec/LSB
Sensitivity Temperature Coefficient	−40°C ≤ T_A ≤ +85°C		±40		ppm/°C
Misalignment	Axis to axis		±0.05		Degrees
	Axis to frame (package)		±0.5		Degrees
Nonlinearity	Best fit straight line		±0.1		% of FS
Initial Bias Error	±1 σ		±3		°/sec
In-Run Bias Stability	1 σ, SMPL_PRD = 0x0001		0.007		°/sec
Angular Random Walk	1 σ, SMPL_PRD = 0x0001		1.9		°/√hr
Bias Temperature Coefficient	−40°C ≤ T_A ≤ +85°C		±0.01		°/sec/°C
Linear Acceleration Effect on Bias	Any axis, 1 σ (MSC_CTRL[7] = 1)		0.05		°/sec/*g*
Bias Voltage Sensitivity	VDD = 4.75 V to 5.25 V		±0.3		°/sec/V
Output Noise	±300°/sec range, no filtering		0.8		°/sec rms
Rate Noise Density	f = 25 Hz, ±300°/sec range, no filtering		0.044		°/sec/√Hz rms
3 dB Bandwidth			330		Hz
Sensor Resonant Frequency			14.5		kHz
ACCELEROMETERS	Each axis				
Dynamic Range		±18			*g*
Initial Sensitivity		3.285	3.33	3.38	m*g*/LSB
Sensitivity Temperature Coefficient	−40°C ≤ T_A ≤ +85°C		±50		ppm/°C
Misalignment	Axis to axis		0.2		Degrees
	Axis to frame (package)		±0.5		Degrees
Nonlinearity	Best fit straight line		0.1		% of FS
Initial Bias Error	±1 σ		±50		m*g*
In-Run Bias Stability	1 σ, SMPL_PRD = 0x0001		0.2		m*g*
Velocity Random Walk	1 σ, SMPL_PRD = 0x0001		0.2		m/sec/√hr
Bias Temperature Coefficient	−40°C ≤ T_A ≤ +85°C		±0.3		m*g*/°C
Bias Voltage Sensitivity	VDD = 4.75 V to 5.25 V		2.5		m*g*/V
Output Noise	No filtering		9		m*g* rms
Noise Density	No filtering		0.5		m*g*/√Hz rms
3 dB Bandwidth			330		Hz
Sensor Resonant Frequency			5.5		kHz
MAGNETOMETER					
Dynamic Range		±2.5	±3.5		gauss
Initial Sensitivity	25°C	0.49	0.5	0.51	mgauss/LSB
Sensitivity Temperature Coefficient	25°C, 1 σ		600		ppm/°C
Misalignment	Axis to axis		0.25		Degrees
	Axis to frame (package)		0.5		Degrees
Nonlinearity	Best fit straight line		0.5		% of FS
Initial Bias Error	25°C, 0 gauss stimulus		±4		mgauss
Bias Temperature Coefficient	−40°C ≤ T_A ≤ +85°C		0.5		mgauss/°C
Output Noise	25°C, no filtering, rms		1.15		mgauss
Noise Density	25°C, no filtering, rms		0.06		mgauss/√Hz
Bandwidth	−3 dB		1540		Hz

Parameter	Test Conditions/Comments	Min	Typ	Max	Unit
BAROMETER					
Pressure Range					
Operating		300		1100	mbar
Extended[1]		10		1200	mbar
Sensitivity			0.3125		µbar/LSB
Total Error	25°C, 300 mbar to 1100 mbar		1.5		mbar
Relative Error[2]	−40°C to +85°C, 300 mbar to 1100 mbar		2.5		mbar
Linearity[3]	25°C, 300 mbar to 1100 mbar		0.1		% of FS
	−40°C to +85°C, 300 mbar to 1100 mbar		0.15		% of FS
Noise			0.027		mbar rms
TEMPERATURE SENSOR					
Scale Factor	25°C, output = 0x0000		0.14		°C/LSB
ADC INPUT					
Resolution			12		Bits
Integral Nonlinearity			±2		LSB
Differential Nonlinearity			±1		LSB
Offset Error			±4		LSB
Gain Error			±2		LSB
Input Range		0		3.3	V
Input Capacitance	During acquisition		20		pF
DAC OUTPUT	5 kΩ/100 pF to GND				
Resolution			12		Bits
Relative Accuracy	101 LSB ≤ input code ≤ 4095 LSB		±4		LSB
Differential Nonlinearity			±1		LSB
Offset Error			±5		mV
Gain Error			±0.5		%
Output Range		0		3.3	V
Output Impedance			2		Ω
Output Settling Time			10		µs
LOGIC INPUTS[4]					
Input High Voltage, V_{IH}		2.0			V
Input Low Voltage, V_{IL}				0.8	V
	$\overline{CS}$ signal to wake up from sleep mode			0.55	V
$\overline{CS}$ Wake-Up Pulse Width		20			µs
Logic 1 Input Current, I_{IH}	V_{IH} = 3.3 V		±0.2	±10	µA
Logic 0 Input Current, I_{IL}	V_{IL} = 0 V				
All Pins Except $\overline{RST}$			40	60	µA
$\overline{RST}$ Pin			1		mA
Input Capacitance, C_{IN}			10		pF
DIGITAL OUTPUTS[4]					
Output High Voltage, V_{OH}	I_{SOURCE} = 1.6 mA	2.4			V
Output Low Voltage, V_{OL}	I_{SINK} = 1.6 mA			0.4	V
FLASH MEMORY	Endurance[5]	10,000			Cycles
Data Retention[6]	T_J = 85°C	20			Years
FUNCTIONAL TIMES[7]	Time until new data is available				
Power-On Start-Up Time			220		ms
Reset Recovery Time			105		ms
Sleep Mode Recovery Time			7		ms
Flash Memory Update Time			75		ms
Flash Memory Test Time			30		ms
Automatic Self Test Time	SMPL_PRD = 0x0001		52		ms

Parameter	Test Conditions/Comments	Min	Typ	Max	Unit
CONVERSION RATE					SPS
xGYRO_OUT, xACCL_OUT, xMAGN_OUT	SMPL_PRD = 0x0001		819.2		SPS
BAR_OUT, BARO_OUTL[8]	SMPL_PRD = 0x0001		51.2		SPS
Clock Accuracy				±3	%
Sync Input Clock[9]		0.8		1.1	kHz
POWER SUPPLY	Operating voltage range, VDD	4.75	5.0	5.25	V
Power Supply Current			70		mA
	Sleep mode		1.4		mA

[1] The extended pressure range is guaranteed by design.
[2] The relative error assumes that the initial error, at +25°C, is corrected in the end application.
[3] Linearity errors assume a full scale (FS) of 1000 mbar.
[4] The digital I/O signals are driven by an internal 3.3 V supply, and the inputs are 5 V tolerant.
[5] Endurance is qualified as per JEDEC Standard 22, Method A117, and measured at −40°C, +25°C, +85°C, and +125°C.
[6] The data retention lifetime equivalent is at a junction temperature (T_J) of 85°C as per JEDEC Standard 22, Method A117. Data retention lifetime decreases with junction temperature.
[7] These times do not include thermal settling and internal filter response times (330 Hz bandwidth), which may affect overall accuracy.
[8] The BARO_OUT and BARO_OUTL registers sample at a rate that is 1/16th that of the other output registers.
[9] The sync input clock functions below the specified minimum value, but at reduced performance levels.

TIMING SPECIFICATIONS

T_A = 25°C, VDD = 5 V, unless otherwise noted.

Table 2.

		Normal Mode			Burst Read			
Parameter	**Description**	**Min[1]**	**Typ**	**Max**	**Min[1]**	**Typ**	**Max**	**Unit**
f_{SCLK}	Serial clock	0.01		2.0	0.01		1.0	MHz
t_{STALL}	Stall period between data	9			$1/f_{SCLK}$			µs
$t_{READRATE}$	Read rate	40						µs
$t_{\overline{CS}}$	Chip select to SCLK edge	48.8			48.8			ns
t_{DAV}	DOUT valid after SCLK edge			100			100	ns
t_{DSU}	DIN setup time before SCLK rising edge	24.4			24.4			ns
t_{DHD}	DIN hold time after SCLK rising edge	48.8			48.8			ns
t_{SCLKR}, t_{SCLKF}	SCLK rise/fall times, not shown in Timing Diagrams		5	12.5		5	12.5	ns
t_{DR}, t_{DF}	DOUT rise/fall times, not shown in Timing Diagrams		5	12.5		5	12.5	ns
t_{SFS}	$\overline{CS}$ high after SCLK edge	5			5			ns
t_1	Input sync positive pulse width	5			5			µs
t_x	Input sync low time	100			100			µs
t_2	Input sync to data ready output		600			600		µs
t_3	Input sync period	910			910			µs

[1] Guaranteed by design and characterization, but not tested in production.

Timing Diagrams

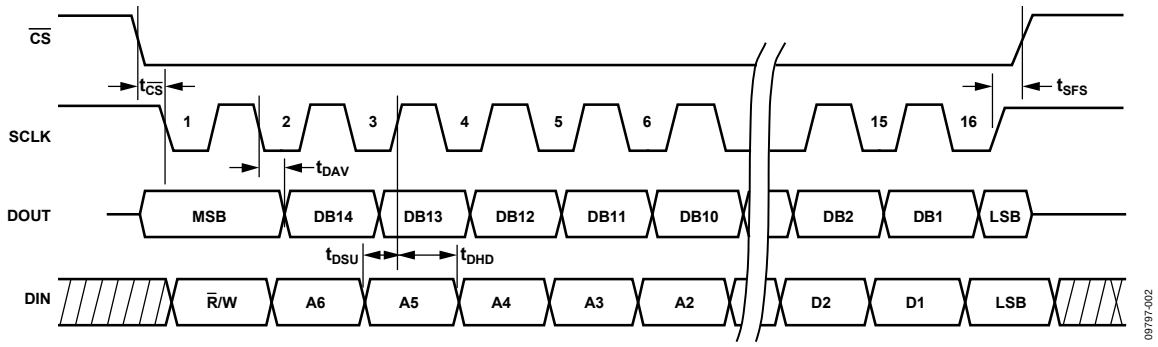


Figure 2. SPI Timing and Sequence

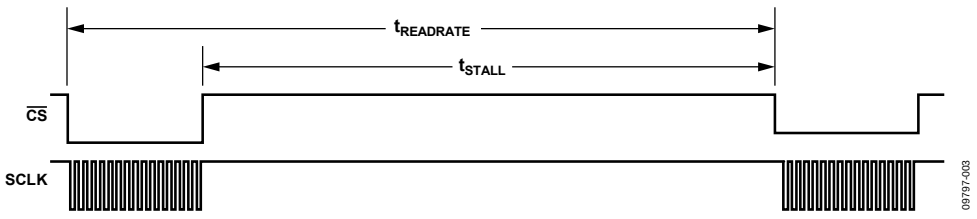


Figure 3. Stall Time and Data Rate

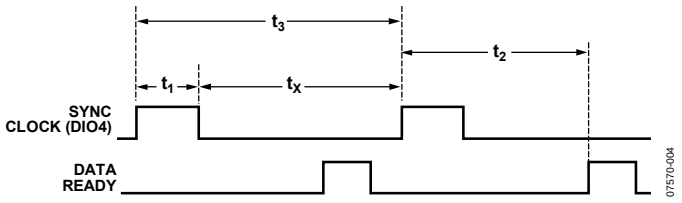


Figure 4. Input Clock Timing Diagram

ABSOLUTE MAXIMUM RATINGS

Table 3.

Parameter	Rating
Acceleration	
Any Axis, Unpowered	2000 *g*
Any Axis, Powered	2000 *g*
VDD to GND	−0.3 V to +6.0 V
Digital Input Voltage to GND	−0.3 V to +5.3 V
Digital Output Voltage to GND	−0.3 V to +3.6 V
Analog Input to GND	−0.3 V to +3.6 V
Temperature	
Operating Range	−40°C to +105°C
Storage Range	−65°C to +125°C[1, 2]
Pressure	6 bar

[1] Extended exposure to temperatures outside the specified temperature range of −40°C to +105°C can adversely affect the accuracy of the factory calibration. For best accuracy, store the parts within the specified operating range of −40°C to +105°C.
[2] Although the device is capable of withstanding short-term exposure to 150°C, long-term exposure threatens internal mechanical integrity.

Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; functional operation of the device at these or any other conditions above those indicated in the operational section of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Table 4. Package Characteristics

Package Type	θ_{JA}	θ_{JC}	Device Weight
24-Lead Module (ML-24-2)	39.8°C/W	14.2°C/W	16 grams

ESD CAUTION

ESD (electrostatic discharge) sensitive device. Charged devices and circuit boards can discharge without detection. Although this product features patented or proprietary protection circuitry, damage may occur on devices subjected to high energy ESD. Therefore, proper ESD precautions should be taken to avoid performance degradation or loss of functionality.

PIN CONFIGURATION AND FUNCTION DESCRIPTIONS

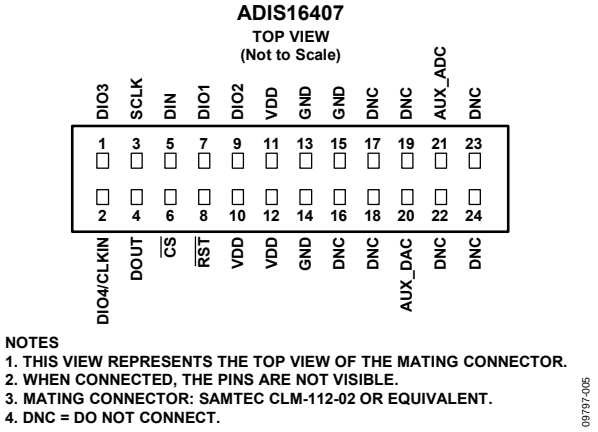


Figure 5. Pin Configuration

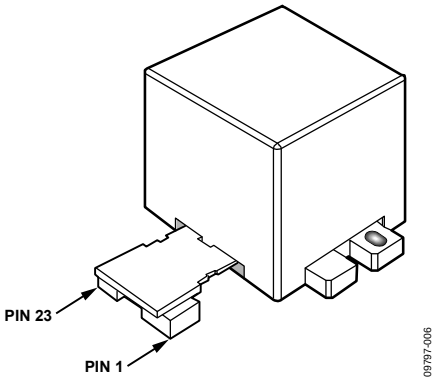


Figure 6. Axial Orientation

Table 5. Pin Function Descriptions

Pin No.	Mnemonic	Type[1]	Description
1	DIO3	I/O	Configurable Digital Input/Output.
2	DIO4/CLKIN	I/O	Configurable Digital Input/Output or Sync Clock Input.
3	SCLK	I	SPI Serial Clock.
4	DOUT	O	SPI Data Output. Clocks the output on the SCLK falling edge.
5	DIN	I	SPI Data Input. Clocks the input on the SCLK rising edge.
6	$\overline{CS}$	I	SPI Chip Select.
7	DIO1	I/O	Configurable Digital Input/Output.
8	$\overline{RST}$	I	Reset.
9	DIO2	I/O	Configurable Digital Input/Output.
10, 11, 12	VDD	S	Power Supply.
13, 14, 15	GND	S	Power Ground.
16, 17, 18, 19, 22, 23, 24	DNC	N/A	Do Not Connect. Do not connect to these pins.
20	AUX_DAC	O	Auxiliary, 12-Bit DAC Output.
21	AUX_ADC	I	Auxiliary, 12-Bit ADC Input.

[1] S is supply, O is output, I is input, N/A is not applicable.

TYPICAL PERFORMANCE CHARACTERISTICS

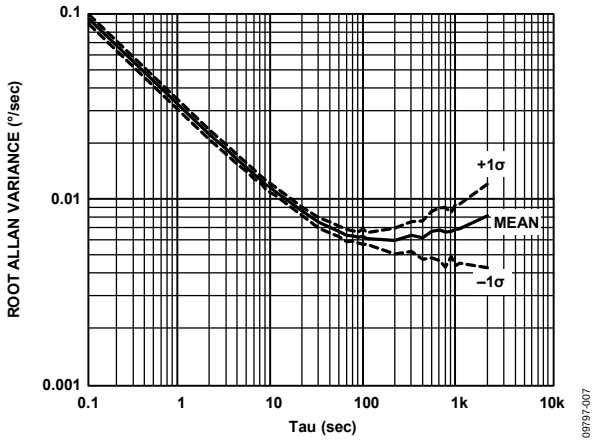


Figure 7. Gyroscope Root Allan Variance

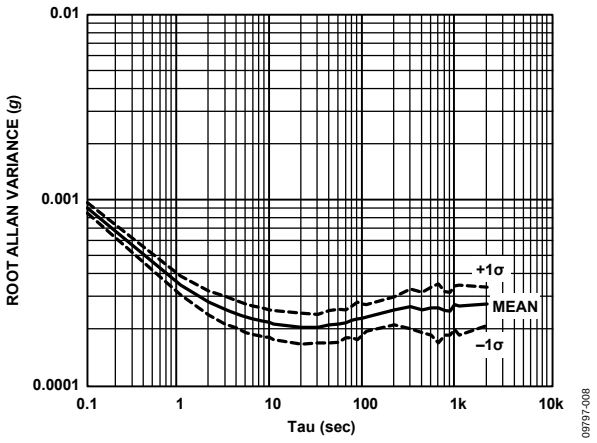


Figure 8. Accelerometer Root Allan Variance

BASIC OPERATION

The ADIS16407 is an autonomous system that requires no user initialization. When it has a valid power supply, it initializes itself and starts sampling, processing, and loading sensor data into the output registers at a sample rate of 819.2 SPS. DIO1 pulses high after each sample cycle concludes. The SPI interface enables simple integration with many embedded processor platforms, as shown in Figure 9 (electrical connection) and Table 6 (pin functions).

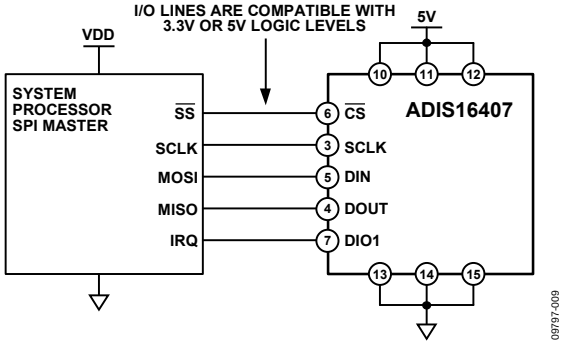


Figure 9. Electrical Connection Diagram

Table 6. Generic Master Processor Pin Names and Functions

Pin Name	Function
$\overline{SS}$	Slave select
SCLK	Serial clock
MOSI	Master output, slave input
MISO	Master input, slave output
IRQ	Interrupt request

The ADIS16407 SPI interface supports full duplex serial communication (simultaneous transmit and receive) and uses the bit sequence shown in Figure 13. Table 7 provides a list of the most common settings that require attention to initialize the serial port of a processor for the ADIS16407 SPI interface.

Table 7. Generic Master Processor SPI Settings

Processor Setting	Description
Master	The ADIS16407 operates as a slave
SCLK Rate ≤ 2 MHz[1]	Maximum serial clock rate
SPI Mode 3	CPOL = 1 (polarity), CPHA = 1 (phase)
MSB-First Mode	Bit sequence
16-Bit Mode	Shift register/data length

[1] For burst read, SCLK rate ≤ 1 MHz.

READING SENSOR DATA

The ADIS16407 provides two different options for acquiring sensor data: single register and burst register. A single register read requires two 16-bit SPI cycles. The first cycle requests the contents of a register using the bit assignments in Figure 13. Bit DC7 to Bit DC0 are don't care for a read, and then the output register contents follow on DOUT during the second sequence. Figure 10 includes three single register reads in succession. In this example, the process starts with DIN = 0x0400 to request the contents of XGYRO_OUT, then follows with 0x0600 to request YGYRO_OUT and 0x0800 to request ZGYRO_OUT. Full duplex operation enables processors to use the same 16-bit SPI cycle to read data from DOUT while requesting the next set of data on DIN. Figure 11 provides an example of the four SPI signals when reading XGYRO_OUT in a repeating pattern.

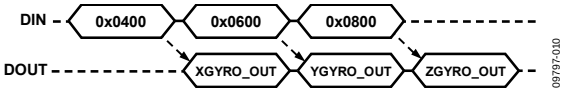


Figure 10. SPI Read Example

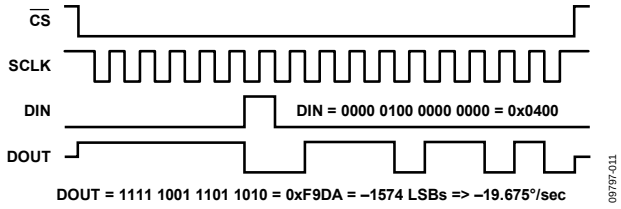


Figure 11. Example SPI Read, Second 16-Bit Sequence

Burst Read Function

The burst read function enables the user to read all output registers using one command on the DIN line and shortens the stall time between each 16-bit segment to one SCLK cycle (see Table 2). Figure 12 provides the burst read sequence of data on each SPI signal. The sequence starts with writing 0x3E00 to DIN, followed by each output register clocking out on DOUT, in the order in which they appear in Table 8.

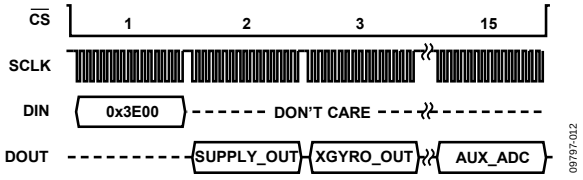


Figure 12. Burst Read Sequence

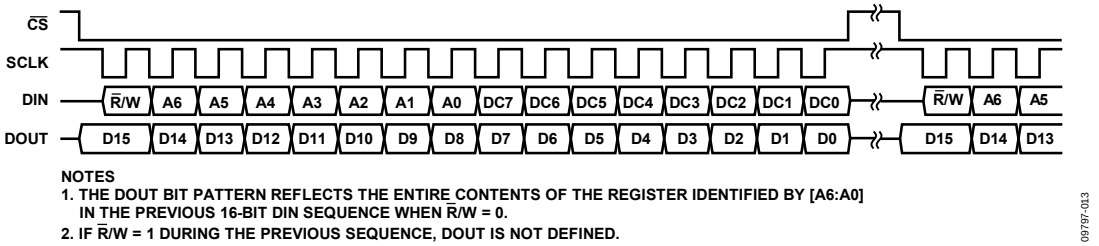


Figure 13. SPI Communication Bit Sequence

OUTPUT DATA REGISTERS

The output registers in Table 8 provide the most recent sensor data produced by the ADIS16407. Each output register has flags for new data indication and error/alarm conditions, which reduces the need to monitor DIAG_STAT.

Table 8. Output Data Register Formats

Register	Address	Measurement
SUPPLY_OUT	0x02	Power supply
XGYRO_OUT	0x04	Gyroscope, x-axis
YGYRO_OUT	0x06	Gyroscope, y-axis
ZGYRO_OUT	0x08	Gyroscope, z-axis
XACCL_OUT	0x0A	Accelerometer, x-axis
YACCL_OUT	0x0C	Accelerometer, y-axis
ZACCL_OUT	0x0E	Accelerometer, z-axis
XMAGN_OUT	0x10	Magnetometer, x-axis
YMAGN_OUT	0x12	Magnetometer, y-axis
ZMAGN_OUT	0x14	Magnetometer, z-axis
BARO_OUT	0x16	Barometer/pressure, higher
BARO_OUTL	0x18	Barometer/pressure, lower
TEMP_OUT[1]	0x1A	Internal temperature
AUX_ADC	0x1C	Auxiliary ADC

[1] This is most useful for monitoring relative changes in the temperature.

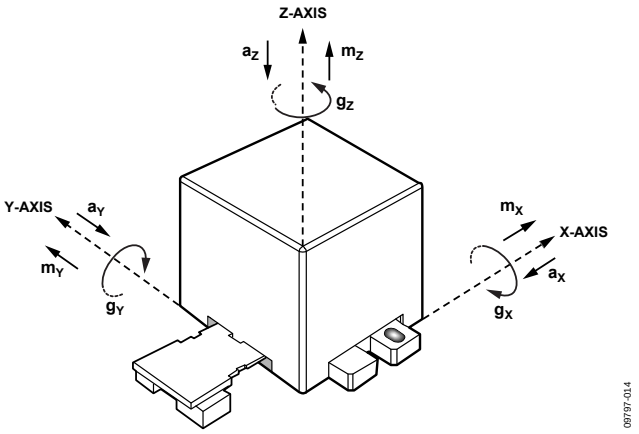


Figure 14. Inertial Sensor Direction Reference

Gyroscopes

Figure 14 provides arrows (g_X, g_Y, g_Z) that indicate the direction of rotation, which produces a positive response in the gyroscope output registers: XGYRO_OUT (x-axis, Table 9), YGYRO_OUT (y-axis, Table 10), and ZGYRO_OUT (z-axis, Table 11). Table 12 illustrates the gyroscope data format.

Table 9. XGYRO_OUT (Base Address = 0x04), Read Only

Bits	Description
[15]	New data indicator (ND), 1 = new data in register
[14]	Error/alarm, 1 = active, see DIAG_STAT for error flags
[13:0]	X-axis gyroscope data, twos complement format, 0.05°/sec per LSB, when SENS_AVG[15:8] = 0x04

Table 10. YGYRO_OUT (Base Address = 0x06), Read Only

Bits	Description
[15]	New data indicator (ND), 1 = new data in register
[14]	Error/alarm, 1 = active, see DIAG_STAT for error flags
[13:0]	Y-axis gyroscope data, twos complement format, 0.05°/sec per LSB, when SENS_AVG[15:8] = 0x04

Table 11. ZGYRO_OUT (Base Address = 0x08), Read Only

Bits	Description
[15]	New data indicator (ND), 1 = new data in register
[14]	Error/alarm, 1 = active, see DIAG_STAT for error flags
[13:0]	Z-axis gyroscope data, twos complement format, 0.05°/sec per LSB, when SENS_AVG[15:8] = 0x04

Table 12. Rotation Rate, Twos Complement Format

Rotation Rate	Decimal	Hex	Binary
+300°/sec	+6000	0x1770	xx01 0111 0111 0000
+0.1°/sec	+2	0x0002	xx00 0000 0000 0010
+0.05°/sec	+1	0x0001	xx00 0000 0000 0001
0°/sec	0	0x0000	xx00 0000 0000 0000
−0.05°/sec	−1	0x3FFF	xx11 1111 1111 1111
−0.1°/sec	−2	0x3FFE	xx11 1111 1111 1110
−300°/sec	−6000	0x2890	xx10 1000 1001 0000

Accelerometers

Figure 14 provides arrows (a_X, a_Y, a_Z) that indicate the direction of acceleration, which produces a positive response in the gyroscope output registers: XACCL_OUT (x-axis, Table 13), YACCL_OUT (y-axis, Table 14), and ZACCL_OUT (z-axis, Table 15). Table 16 illustrates the accelerometer data format.

Table 13. XACCL_OUT (Base Address = 0x0A), Read Only

Bits	Description
[15]	New data indicator (ND), 1 = new data in register
[14]	Error/alarm, 1 = active, see DIAG_STAT for error flags
[13:0]	X-axis acceleration data, twos complement format, 0.25 m*g* per LSB

Table 14. YACCL_OUT (Base Address = 0x0C), Read Only

Bits	Description
[15]	New data indicator (ND), 1 = new data in register
[14]	Error/alarm, 1 = active, see DIAG_STAT for error flags
[13:0]	Y-axis acceleration data, twos complement format, 0.25 m*g* per LSB

Table 15. ZACCL_OUT (Base Address = 0x0E), Read Only

Bits	Description
[15]	New data indicator (ND), 1 = new data in register
[14]	Error/alarm, 1 = active, see DIAG_STAT for error flags
[13:0]	Z-axis acceleration data, twos complement format, 0.25 m*g* per LSB

Table 16. Acceleration, Twos Complement Format

Acceleration	Decimal	Hex	Binary
+18 *g*	+5401	0x1519	xx01 0101 0001 1001
+6.667 m*g*	+2	0x0002	xx00 0000 0000 0010
+3.333 m*g*	+1	0x0001	xx00 0000 0000 0001
0 *g*	0	0x0000	xx00 0000 0000 0000
−3.333 m*g*	−1	0x3FFF	xx11 1111 1111 1111
−6.667 m*g*	−2	0x3FFE	xx11 1111 1111 1110
−18 *g*	−5401	0x2AE7	xx10 1010 1110 0111

Magnetometers

Figure 14 provides arrows (m_X, m_Y, m_Z) that indicate the direction of the magnetic field, which produces a positive response in the gyroscope output registers: XMAGN_OUT (x-axis, Table 17), YMAGN_OUT (y-axis, Table 18), and ZAMAGN_OUT (z-axis, Table 19). Table 20 illustrates the magnetic field intensity data format.

Table 17. XMAGN_OUT (Base Address = 0x10), Read Only

Bits	Description
[15]	New data indicator (ND), 1 = new data in register
[14]	Error/alarm, 1 = active, see DIAG_STAT for error flags
[13:0]	X-axis magnetic field intensity data, twos complement format, 0.5 mgauss per LSB

Table 18. YMAGN_OUT (Base Address = 0x12), Read Only

Bits	Description
[15]	New data indicator (ND), 1 = new data in register
[14]	Error/alarm, 1 = active, see DIAG_STAT for error flags
[13:0]	Y-axis magnetic field intensity data, twos complement format, 0.5 mgauss per LSB

Table 19. ZMAGN_OUT (Base Address = 0x14), Read Only

Bits	Description
[15]	New data indicator (ND), 1 = new data in register
[14]	Error/alarm, 1 = active, see DIAG_STAT for error flags
[13:0]	Z-axis magnetic field intensity data, twos complement format, 0.5 mgauss per LSB

Table 20. Magnetometer, Twos Complement Format

Magnetic Field	Decimal	Hex	Binary
+2.5 gauss	+5000	0x1388	xx01 0011 1000 1000
+0.001 gauss	+2	0x0002	xx00 0000 0000 0010
+0.0005 gauss	+1	0x0001	xx00 0000 0000 0001
0 gauss	0	0x0000	xx00 0000 0000 0000
−0.0005 gauss	−1	0x3FFF	xx11 1111 1111 1111
−0.0005 gauss	−2	0x3FFE	xx11 1111 1111 1110
−2.5 gauss	−5000	0x2C78	xx10 1100 0111 1000

Barometric Pressure

The barometric pressure measurements are contained in two registers, BARO_OUT (Table 21) and BARO_OUTL (Table 22) registers. Table 23 provides several numerical format examples for BARO_OUT, which is sufficient for most applications.

Use BAR_OUTL and the following steps to increase the numerical resolution by 8-bits for best performance:

1. Read BAR_OUT and multiply by 256 (shift 8 bits)
2. Read BAR_OUTL and max off upper 8 bits
3. Add results together for a 24-bit result, where 1 LSB = 0.0003125 and 0x00000 = 0 mbar

Table 21. BARO_OUT (Base Address = 0x16), Read Only

Bits	Description
[15]	New data indicator (ND), 1 = new data in register
[14]	Error/alarm, 1 = active, see DIAG_STAT for error flags
[13:0]	Barometric pressure data, binary data format, 0.08 mbar per LSB, 0x0000 = 0 mbar

Table 22. BARO_OUTL (Base Address = 0x18), Read Only

Bits	Description
[15:8]	Not used
[7:0]	Barometric pressure data, binary data format, 0.0003125 mbar per LSB, 0x0000 = 0 mbar

Table 23. Pressure, Binary, BARO_OUT Only

Pressure (mbar)	Decimal	Hex	Binary
1200	15,000	0x3A98	xx11 1010 1001 1000
1100	13,750	0x35B6	xx11 0101 1011 0110
1000	12,500	0x30D4	xx11 0000 1101 0100
0.16	2	0x0002	xx00 0000 0000 0010
0.08	1	0x0001	xx00 0000 0000 0001
0	0	0x0000	xx00 0000 0000 0000

Internal Temperature

The internal temperature measurement data loads into the TEMP_OUT (Table 24) register. Table 25 illustrates the temperature data format.

Table 24. TEMP_OUT (Base Address = 0x1A), Read Only

Bits	Description
[15]	New data indicator (ND), 1 = new data in register
[14]	Error/alarm, 1 = active, see DIAG_STAT for error flags
[13:12]	Not used
[11:0]	Internal temperature data, twos complement, 0.136°C/LSB, 25°C = 0x000

Table 25. Temperature, Twos Complement Format

Temperature	Decimal	Hex	Binary
+105°C	+588 LSB	0x24C	xxxx 0010 0100 1100
+85°C	+441 LSB	0x1B9	xxxx 0001 1011 1001
+25.272°C	+2 LSB	0x002	xxxx 0000 0000 0010
+25.136°C	+1 LSB	0x001	xxxx 0000 0000 0001
+25°C	0 LSB	0x000	xxxx 0000 0000 0000
+24.864°C	−1 LSB	0xFFF	xxxx 1111 1111 1111
+24.728°C	−2 LSB	0xFFE	xxxx 1111 1111 1110
−40°C	−478 LSB	0xE22	xxxx 1110 0010 0010

Power Supply

The SUPPLY_OUT register (Table 26) provides a measurement of the voltage that is on the VDD pins of the device. Table 27 illustrates the power supply data format.

Table 26. SUPPLY_OUT (Base Address = 0x02), Read Only

Bits	Description
[15]	New data indicator (ND), 1 = new data in register
[14]	Error/alarm, 1 = active, see DIAG_STAT for error flags
[13:12]	Not used
[11:0]	Power supply measurement data, binary format, 2.418 mV/LSB, 0 V = 0x000

Table 27. Power Supply Data, Binary Format

Voltage	Decimal	Hex	Binary
+5.25 V	2171	0x87B	xxxx 1000 0111 1011
+5.0 V	2068	0x814	xxxx 1000 0001 0100
+4.75 V	1964	0x7AC	xxxx 0111 1010 1100
1 V	414	0x19E	xxxx 0001 1001 1110
4.836 mV	2	0x002	xxxx 0000 0000 0010
2.418 mV	1	0x001	xxxx 0000 0000 0001
0 V	0	0x000	xxxx 0000 0000 0000

INPUT ADC CHANNEL

The AUX_ADC register provides access to the auxiliary ADC input channel. The ADC is a 12-bit successive approximation converter that has an input circuit equivalent to the one shown in Figure 15. The maximum input is 3.3 V. The ESD protection diodes can handle 10 mA without causing irreversible damage. The on resistance (R1) of the switch has a typical value of 100 Ω. The sampling capacitor, C2, has a typical value of 16 pF.

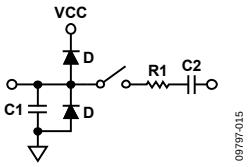


Figure 15. Equivalent Analog Input Circuit (Conversion Phase: Switch Open, Track Phase: Switch Closed)

Table 28. AUX_ADC (Base Address = 0x1C), Read Only

Bits	Description
[15]	New data indicator (ND), 1 = new data in register
[14]	Error/alarm, 1 = active, see DIAG_STAT for error flags
[13:12]	Not used
[11:0]	Analog input channel data, binary format, 0.8059 mV/LSB, 0 V = 0x000

Table 29. Analog Input, Offset Binary Format

Input Voltage	Decimal	Hex	Binary
3.3 V	4095	0xFFF	xxxx 1111 1111 1111
1 V	1241	0x4D9	xxxx 0100 1101 1001
1.6118 mV	2	0x002	xxxx 0000 0000 0010
805.9 µV	1	0x001	xxxx 0000 0000 0001
0 V	0	0x000	xxxx 0000 0000 0000

DEVICE CONFIGURATION

The control registers in Table 30 provide users with a variety of configuration options. The SPI provides access to these registers, one byte at a time, using the bit assignments in Figure 13. Each register has 16 bits, where Bits[7:0] represent the lower address, and Bits[15:8] represent the upper address. Figure 16 provides an example of writing 0x03 to Address 0x3B (SMPL_PRD[15:8]), using DIN = 0xBB03. This example reduces the sample rate by a factor of eight (see Table 46).

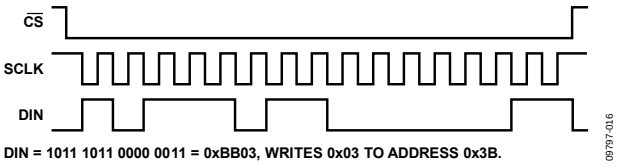


Figure 16. Example SPI Write Sequence

Dual Memory Structure

Writing configuration data to a control register updates its SRAM contents, which are volatile. After optimizing each relevant control register setting in a system, set GLOB_CMD[3] = 1 (DIN = 0xBE08) to backup these settings in nonvolatile flash memory. The flash backup process requires a valid power supply level for the entire 75 ms process time. Table 30 provides a user register memory map that includes a flash backup column. A "yes" in this column indicates that a register has a mirror location in flash and, when backed up properly, it automatically restores itself during startup or after a reset. Figure 17 provides a diagram of the dual memory structure used to manage operation and store critical user settings.

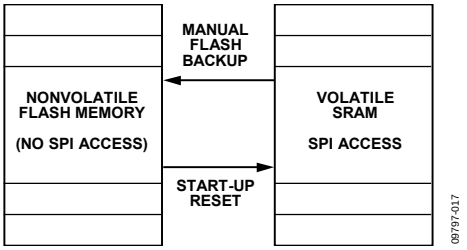


Figure 17. SRAM and Flash Memory Diagram

USER REGISTERS

Table 30. User Register Memory Map[1]

Name	R/W	Flash Backup	Address[2]	Default	Function	Bit Assignments
FLASH_CNT	R	Yes	0x00	N/A	Flash memory write count	See Table 38
SUPPLY_OUT	R	No	0x02	N/A	Power supply measurement	See Table 26
XGYRO_OUT	R	No	0x04	N/A	X-axis gyroscope output	See Table 9
YGYRO_OUT	R	No	0x06	N/A	Y-axis gyroscope output	See Table 10
ZGYRO_OUT	R	No	0x08	N/A	Z-axis gyroscope output	See Table 11
XACCL_OUT	R	No	0x0A	N/A	X-axis accelerometer output	See Table 13
YACCL_OUT	R	No	0x0C	N/A	Y-axis accelerometer output	See Table 14
ZACCL_OUT	R	No	0x0E	N/A	Z-axis accelerometer output	See Table 15
XMAGN_OUT	R	No	0x10	N/A	X-axis magnetometer measurement	See Table 17
YMAGN_OUT	R	No	0x12	N/A	Y-axis magnetometer measurement	See Table 18
ZMAGN_OUT	R	No	0x14	N/A	Z-axis magnetometer measurement	See Table 19
BARO_OUT	R	No	0x16	N/A	Barometer pressure measurement, high word	See Table 21
BARO_OUTL	R	No	0x18	N/A	Barometer pressure measurement, low word	See Table 22
TEMP_OUT	R	No	0x1A	N/A	Temperature output	See Table 24
AUX_ADC	R	No	0x1C	N/A	Auxiliary ADC measurement	See Table 28
XGYRO_OFF	R/W	Yes	0x1E	0x0000	X-axis gyroscope bias offset factor	See Table 49
YGYRO_OFF	R/W	Yes	0x20	0x0000	Y-axis gyroscope bias offset factor	See Table 50
ZGYRO_OFF	R/W	Yes	0x22	0x0000	Z-axis gyroscope bias offset factor	See Table 51
XACCL_OFF	R/W	Yes	0x24	0x0000	X-axis acceleration bias offset factor	See Table 52
YACCL_OFF	R/W	Yes	0x26	0x0000	Y-axis acceleration bias offset factor	See Table 53
ZACCL_OFF	R/W	Yes	0x28	0x0000	Z-axis acceleration bias offset factor	See Table 54
XMAGN_HIC	R/W	Yes	0x2A	0x0000	X-axis magnetometer, hard iron factor	See Table 55
YMAGN_HIC	R/W	Yes	0x2C	0x0000	Y-axis magnetometer, hard iron factor	See Table 56
ZMAGN_HIC	R/W	Yes	0x2E	0x0000	Z-axis magnetometer, hard iron factor	See Table 57
XMAGN_SIC	R/W	Yes	0x30	0x0800	X-axis magnetometer, soft iron factor	See Table 58
YMAGN_SIC	R/W	Yes	0x32	0x0800	Y-axis magnetometer, soft iron factor	See Table 59
ZMAGN_SIC	R/W	Yes	0x34	0x0800	Z-axis magnetometer, soft iron factor	See Table 60
GPIO_CTRL	R/W	No	0x36	0x0000	Auxiliary digital input/output control	See Table 42
MSC_CTRL	R/W	Yes	0x38	0x0006	Miscellaneous control	See Table 39
SMPL_PRD	R/W	Yes	0x3A	0x0001	Internal sample period (rate) control	See Table 46
SENS_AVG	R/W	Yes	0x3C	0x0402	Dynamic range and digital filter control	See Table 47
SLP_CTRL	W	No	0x3E	N/A	Sleep mode control	See Table 33
DIAG_STAT	R	No	0x40	0x0000	System status	See Table 40
GLOB_CMD	W	N/A	0x42	0x0000	System command	See Table 32
ALM_MAG1	R/W	Yes	0x44	0x0000	Alarm 1 amplitude threshold	See Table 62
ALM_MAG2	R/W	Yes	0x46	0x0000	Alarm 2 amplitude threshold	See Table 63
ALM_SMPL1	R/W	Yes	0x48	0x0000	Alarm 1 sample size	See Table 64
ALM_SMPL2	R/W	Yes	0x4A	0x0000	Alarm 2 sample size	See Table 65
ALM_CTRL	R/W	Yes	0x4C	0x0000	Alarm control	See Table 66
AUX_DAC	R/W	No	0x4E	0x0000	Auxiliary DAC data	See Table 43
Reserved	N/A	N/A	0x50	N/A	Reserved	
LOT_ID1	R	Yes	0x52	N/A	Lot identification number	See Table 34
LOT_ID2	R	Yes	0x54	N/A	Lot identification number	See Table 35
PROD_ID	R	Yes	0x56	0x4107	Product identifier	See Table 36
SERIAL_NUM	R	Yes	0x58	N/A		See Table 37

[1] N/A means not applicable.
[2] Each register contains two bytes. The address of the lower byte is displayed. The address of the upper byte is equal to the address of the lower byte plus 1.

SYSTEM FUNCTIONS

The ADIS16407 provides a number of system level controls for managing its operation, using the registers in Table 31.

Table 31. System Tool Registers

Register Name	Address	Description
MSC_CTRL	0x38	Self test, calibration, data ready
SLP_CTRL	0x3E	Sleep mode control
DIAG_STAT	0x40	Error flags
GLOB_CMD	0x42	Single command functions
LOT_ID1	0x52	Lot Identification Code 1
LOT_ID2	0x54	Lot Identification Code 2
PROD_ID	0x56	Product identifier
SERIAL_NUM	0x58	Serial number

GLOBAL COMMANDS

The GLOB_CMD register in Table 32 provides trigger bits for software reset, flash memory management, DAC control, and calibration control. Start each of these functions by writing a 1 to the assigned bit in GLOB_CMD. After completing the task, the bit automatically returns to 0. For example, set GLOB_CMD[7] = 1 (DIN = 0xC280) to initiate a software reset, which stops the sensor operation and runs the device through its start-up sequence. Set GLOB_CMD[3] = 1 (DIN = 0xC208) to back up the user register contents in nonvolatile flash. This sequence includes loading the control registers with the data in their respective flash memory locations prior to producing new data.

Table 32. GLOB_CMD (Base Address = 0x42), Write Only

Bits	Description (Default = 0x0000)
[15:8]	Not used
[7]	Software reset
[6:4]	Not used
[3]	Flash update
[2]	Auxiliary DAC data latch
[1]	Factory calibration restore
[0]	Gyroscope bias correction

POWER MANAGEMENT

The SLP_CTRL register (see Table 33) provides two sleep modes for system level management: normal and timed. Set SLP_CTRL[8] = 1 (DIN = 0xBF01) to start normal sleep mode. When the device is in sleep mode, the following events can cause it to wake up: asserting $\overline{CS}$ from high to low, asserting $\overline{RST}$ from high to low, or cycling the power. Use SLP_CTRL[7:0] to put the device into sleep mode for a specified period. For example, SLP_CNT[7:0] = 0x64 (DIN = 0xBE64) puts the ADIS16407 to sleep for 50 seconds.

Table 33. SLP_CTRL (Base Address = 0x3E), Write Only

Bits	Description
[15:9]	Not used
[8]	Normal sleep mode (1 = start sleep mode)
[7:0]	Timed sleep mode (write 0x01 to 0xFF to start) Sleep mode duration, binary, 0.5 sec/LSB

PRODUCT IDENTIFICATION

The PROD_ID register in Table 36 contains the binary equivalent of 16,407. It provides a product specific variable for systems that need to track this in their system software. The LOT_ID1 and LOT_ID2 registers in Table 34 and Table 35 combine to provide a unique, 32-bit lot identification code. The SERIAL_NUM register in Table 37 contains a binary number that represents the serial number on the device label. The assigned serial numbers in SERIAL_NUM are lot specific.

Table 34. LOT_ID1 (Base Address = 0x52), Read Only

Bits	Description
[15:0]	Lot identification, binary code

Table 35. LOT_ID2 (Base Address = 0x54), Read Only

Bits	Description
[15:0]	Lot identification, binary code

Table 36. PROD_ID Bit (Base Address = 0x56), Read Only

Bits	Description (Default = 0x4017)
[15:0]	Product identification = 0x4017

Table 37. SERIAL_NUM (Base Address = 0x58), Read Only

Bits	Description
[15:12]	Reserved
[11:0]	Serial number, 1 to 4094 (0xFFE)

MEMORY MANAGEMENT

The FLASH_CNT register in Table 38 provides a 16-bit counter that helps track the number of write cycles to the nonvolatile flash memory. The flash updates every time a manual flash update occurs. A manual flash update is initiated by the GLOB_CMD[3] bit and is also performed at the completion of the GLOB_CMD[1:0] functions (see Table 32).

Table 38. FLASH_CNT (Base Address = 0x00), Read Only

Bits	Description
[15:0]	Binary counter

Checksum Test

Set MSC_CTRL[11] = 1 (DIN = 0xB908) to perform a check-sum test of the internal program memory. This function takes a summation of the internal program memory and compares it with the original summation value for the same locations (from factory configuration). Check the results in the DIAG_STAT register, which is in Table 40. DIAG_STAT[6] equals 0 if the sum matches the correct value, and 1 if it does not. Make sure that the power supply is within specification for the entire 20 ms that this function takes to complete.

SELF TEST FUNCTION

Gyroscopes/Accelerometers

The MSC_CTRL register in Table 39 provides a self test function for the gyroscopes and accelerometers. This function allows the user to verify the mechanical integrity of each MEMS sensor. When enabled, the self test applies an electrostatic force to each internal sensor element, which causes them to move. The movement in each element simulates its response to actual rotation/acceleration and generates a predictable electrical response in the sensor outputs. The ADIS16407 exercises this function and compares the response to an expected range of responses and reports a pass/fail response to DIAG_STAT[5]. If this is high, the DIAG_STAT[15:10] provide pass/fail flags for each inertial sensor.

Table 39. MSC_CTRL (Base Address = 0x38), Read/Write

Bits	Description (Default = 0x0006)
[15:12]	Not used
[11]	Checksum memory test (cleared upon completion)[1] 1 = enabled, 0 = disabled
[10]	Internal self test (cleared upon completion)[1] 1 = enabled, 0 = disabled
[9:8]	Do not use, always set to 00
[7]	Linear acceleration bias compensation for gyroscopes 1 = enabled, 0 = disabled
[6]	Point of percussion, see Figure 6 1 = enabled, 0 = disabled
[5:3]	Not used
[2]	Data ready enable 1 = enabled, 0 = disabled
[1]	Data ready polarity 1 = active high, 0 = active low
[0]	Data ready line select 1 = DIO2, 0 = DIO1

[1] The bit is automatically reset to 0 after finishing the test.

Barometer

The barometer self test function is part of the power-on and reset initialization processes. DIAG_STAT[7] (see Table 40) contains the result of this test after the device completes normal operation. If DIAG_STAT[7] = 1, initiate a software reset by setting GLOB_CMD[7] = 1 (DIN = 0xC280). If DIAG_STAT[7] = 0 after the reset process completes, then the barometer is functional. A persistent fail result in DIAG_STAT[7] indicates a potential problem with the barometer.

STATUS/ERROR FLAGS

The DIAG_STAT register in Table 40 provides error flags for a number of functions. Each flag uses 1 to indicate an error condition and 0 to indicate a normal condition. Reading this register provides access to the status of each flag and resets all of the bits to 0 for monitoring future operation. If the error condition remains, the error flag returns to 1 at the conclusion of the next sample cycle. DIAG_STAT[0] does not require a read of this register to return to 0. If the power supply voltage goes back into range, this flag clears automatically. The SPI communication error flag in DIAG_STAT[3] indicates that the number of SCLKs in a SPI sequence did not equal a multiple of 16 SCLKs.

Table 40. DIAG_STAT (Base Address = 0x40), Read Only

Bits	Description (Default = 0x0000)
[15]	Z-axis accelerometer self test result 1 = fail, 0 = pass
[14]	Y-axis accelerometer self test result 1 = fail, 0 = pass
[13]	X-axis accelerometer self test result 1 = fail, 0 = pass
[12]	Z-axis gyroscope self test result 0 = pass
[11]	Y-axis gyroscope self test result 1 = fail, 0 = pass
[10]	X-axis gyroscope self test result 1 = fail, 0 = pass
[9]	Alarm 2 status 1 = active, 0 = inactive
[8]	Alarm 1 status 1 = active, 0 = inactive
[7]	Barometer self test 1 = fail (issue with sensor function), 0 = pass (no issue)
[6]	Flash test (checksum) result 1 = fail, 0 = pass
[5]	Self test diagnostic result 1 = fail, 0 = pass
[4]	Sensor overrange condition 1 = overrange, 0 = normal
[3]	SPI communication 1 = fail (number of SCLKs not equal to a multiple of 16) 0 = pass (number of SCLKs is equal to a multiple of 16)
[2]	Flash update verification 1 = fail (flash update was not successful) 0 = pass (flash update was successful)
[1]	Power supply high 1 = VDD > 5.25 V 0 = VDD ≤ 5.25 V
[0]	Power supply low 1 = VDD < 4.75 V 0 = VDD ≥ 4.75 V

INPUT/OUTPUT CONFIGURATION

Table 41 provides a summary of registers that provide input/output configuration and control.

Table 41. Input/Output Registers

Register Name	Address	Description
GPIO_CTRL	0x36	General-purpose I/O control
MSC_CTRL	0x38	Self test, calibration, data ready
AUX_DAC	0x4E	Output voltage control, AUX_DAC

DATA READY INDICATOR

The factory default setting of MSC_CTRL[2:0] = 110 establishes DIO1 as a positive polarity data ready signal. See Table 39 for additional data ready configuration options. For example, set MSC_CTRL[2:0] = 100 (DIN = 0xB804) to change the polarity of the data ready signal on DIO1 for interrupt inputs that require negative logic inputs for activation. The pulse width is typically between 60 µs and 150 µs, including jitter (±30 µs).

GENERAL-PURPOSE INPUT/OUTPUT

DIO1, DIO2, DIO3, and DIO4 are configurable, general-purpose input/output lines that serve multiple purposes. The data ready controls in MSC_CTRL[2:0] have the highest priority for configuring DIO1 and DIO2. The alarm indicator controls in ALM_CTRL[2:0] have the second highest priority for configuring DIO1 and DIO2. The external clock control associated with SMPL_PRD[0] has the highest priority for DIO4 configuration (see Table 46). GPIO_CTRL in Table 42 has the lowest priority for configuring DIO1, DIO2, and DIO4, and has absolute control over DIO3.

Table 42. GPIO_CTRL (Base Address = 0x36), Read/Write

Bits	Description (Default = 0x0000)
[15:12]	Not used
[11]	General-Purpose I/O Line 4 (DIO4) data level
[10]	General-Purpose I/O Line 3 (DIO3) data level
[9]	General-Purpose I/O Line 2 (DIO2) data level
[8]	General-Purpose I/O Line 1 (DIO1) data level
[7:4]	Not used
[3]	General-Purpose I/O Line 4 (DIO4) direction control 1 = output, 0 = input
[2]	General-Purpose I/O Line 3 (DIO3) direction control 1 = output, 0 = input
[1]	General-Purpose I/O Line 2 (DIO2) direction control 1 = output, 0 = input
[0]	General-Purpose I/O Line 1 (DIO1) direction control 1 = output, 0 = input

Example Input/Output Configuration

For example, set GPIO_CTRL[3:0] = 0100 (DIN = 0xB604) to set DIO3 as an output signal pin and DIO1, DIO2, and DIO4 as input signal pins. Set the output on DIO3 to 1 by setting GPIO_CTRL[10] = 1 (DIN = 0xB704). Then, read GPIO_CTRL[7:0] (DIN = 0x3600) and mask off GPIO_CTRL[9:8] and GPIO_CTRL[11] to monitor the digital signal levels on DIO4, DIO2, and DIO1.

AUXILIARY DAC

The AUX_DAC register in Table 43 provides user controls for setting the output voltage on the AUX_DAC pin. The 12-bit AUX_DAC line can drive its output to within 5 mV of the ground reference when it is not sinking current. As the output approaches 0 V, the linearity begins to degrade (~100 LSB starting point). As the sink current increases, the nonlinear range increases. The DAC latch command in GLOB_CMD[2] (see Table 32) moves the values of the AUX_DAC register into the DAC input register, enabling both bytes to take effect at the same time. This prevents undesirable output levels, which reflect single byte changes of the AUX_DAC register.

Table 43. AUX_DAC (Base Address = 0x4E), Read/Write

Bits	Description (Default = 0x0000)
[15:12]	Not used
[11:0]	Data bits, scale factor = 0.8059 mV/LSB, offset binary format, 0 V = 0 LSB

Table 44. Setting AUX_DAC = 1 V

DIN	Description
0xCED9	AUX_DAC[7:0] = 0xD9 (217 LSB)
0xCF04	AUX_DAC[15:8] = 0x04 (1024 LSB)
0xC204	GLOB_CMD[2] = 1; move values into the DAC input register, resulting in a 1 V output level

DIGITAL PROCESSING CONFIGURATION

Table 45. Digital Processing Registers

Register Name	Address	Description
SMPL_PRD	0x3A	Sample rate control
SENS_AVG	0x3C	Digital filtering and range control

SAMPLE RATE

The internal sampling system produces new data in the output data registers at a rate of 819.2 SPS. The SMPL_PRD register in Table 46 provides two functional controls that affect sampling and register update rates. SMPL_PRD[12:8] provides a control for reducing the update rate, using an averaging filter with a decimated output. These bits provide a binomial control that divides the data rate by a factor of 2 every time this number increases by 1. For example, set SMPL_PRD[15:8] = 0x04 (DIN = 0xBB04) to set the decimation factor to 16. This reduces the update rate to 51 SPS and the bandwidth to 25 Hz.

Table 46. SMPL_PRD (Base Address = 0x3A), Read/Write

Bits	Description (Default = 0x0001)
[15:13]	Not used
[12:8]	D, decimation rate setting, binomial, see Figure 19
[7:1]	Not used
[0]	Clock 1 = internal 819.2 SPS 0 = external

INPUT CLOCK CONFIGURATION

SMPL_PRD[0] provides a control for synchronizing the internal sampling to an external clock source. Set SMPL_PRD[0] = 0 (DIN = 0xBA00) and GPIO_CTRL[3] = 0 (DIN = 0xB600) to enable the external clock. See Table 2 and Figure 4 for timing information.

DIGITAL FILTERING

The SENS_AVG register in Table 47 provides user controls for the low-pass filter. This filter contains two cascaded averaging filters that provide a Bartlett window, FIR filter response (see Figure 19). For example, set SENS_AVG[2:0] = 100 (DIN = 0xBC04) to set each stage to 16 taps. When used with the default sample rate of 819.2 SPS and zero decimation (SMPL_PRD[15:8] = 0x00), this value reduces the sensor bandwidth to approximately 16 Hz.

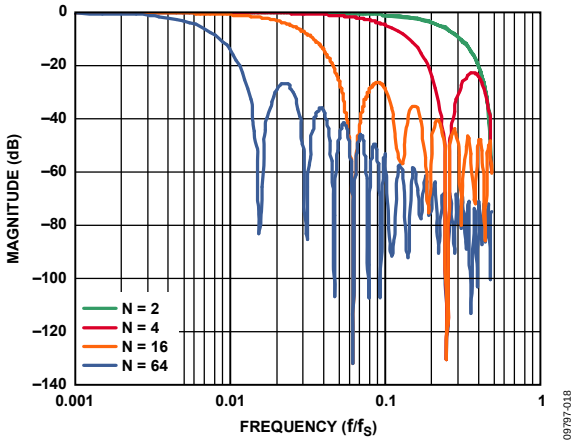


Figure 18. Bartlett Window, FIR Filter Frequency Response (Phase Delay = N Samples)

DYNAMIC RANGE

The SENS_AVG[10:8] bits provide three dynamic range settings for this gyroscope. The lower dynamic range settings (±75°/sec and ±150°/sec) limit the minimum filter tap sizes to maintain resolution. For example, set SENS_AVG[10:8] = 010 (DIN = 0xBD02) for a measurement range of ±150°/sec. Because this setting can influence the filter settings, program SENS_AVG[10:8] before programming SENS_AVG[2:0] if more filtering is required.

Table 47. SENS_AVG (Base Address = 0x3C), Read/Write

Bits	Description (Default = 0x0402)
[15:11]	Not used
[10:8]	Measurement range (sensitivity) selection 100 = ±300°/sec (default condition) 010 = ±150°/sec, filter taps ≥ 4 (Bits[2:0] ≥ 0x02) 001 = ±75°/sec, filter taps ≥ 16 (Bits[2:0] ≥ 0x04)
[7:3]	Not used
[2:0]	Filter Size Variable B Number of taps in each stage; $N_B = 2^B$ See Figure 18 for filter response

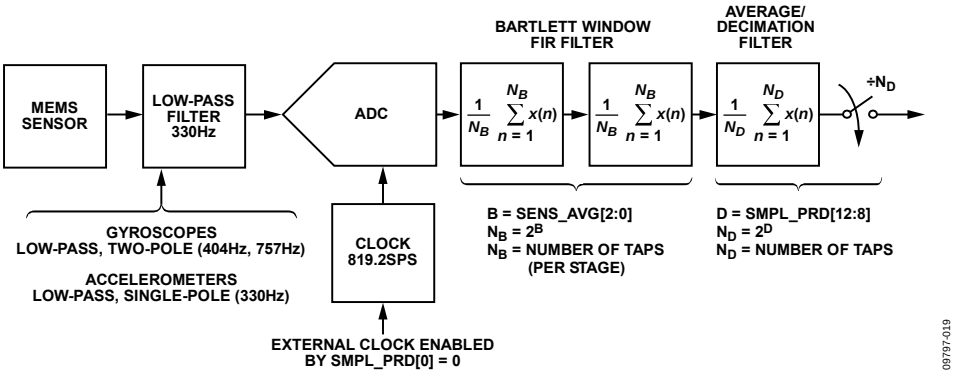


Figure 19. Sampling and Frequency Response Block Diagram

CALIBRATION

The mechanical structure and assembly process of the ADIS16407 provide excellent position and alignment stability for each sensor, even after subjected to temperature cycles, shock, vibration, and other environmental conditions. The factory calibration includes a dynamic characterization of each gyroscope and accelerometer over temperature and generates sensor specific correction formulas. Table 48 provides a list of registers that can help optimize system performance after installation. Figure 20 illustrates the summing function for the offset correction register of each sensor.

Table 48. Registers for User Calibration

Register	Address	Description
XGYRO_OFF	0x1E	Gyroscope bias, x-axis
YGYRO_OFF	0x20	Gyroscope bias, y-axis
ZGYRO_OFF	0x22	Gyroscope bias, z-axis
XACCL_OFF	0x24	Accelerometer bias, x-axis
YACCL_OFF	0x26	Accelerometer bias, y-axis
ZACCL_OFF	0x28	Accelerometer bias, z-axis
XMAGN_HIC	0x2A	Hard iron correction, x-axis
YMAGN_HIC	0x2C	Hard iron correction, y-axis
ZMAGN_HIC	0x2E	Hard iron correction, z-axis
XMAGN_SIC	0x30	Soft iron correction, x-axis
YMAGN_SIC	0x32	Soft iron correction, y-axis
ZMAGN_SIC	0x34	Soft iron correction, z-axis
MSC_CTRL	0x38	Miscellaneous calibration
GLOB_CMD	0x42	Automatic calibration

GYROSCOPES

The XGYRO_OFF (Table 49), YGYRO_ OFF (Table 50), and ZGYRO_ OFF (Table 51) registers provide user-programmable bias adjustment function for the x-, y-, and z-axis gyroscopes, respectively. Figure 20 illustrates that they contain bias correction factors that adjust to the sensor data immediately before it loads into the output register.

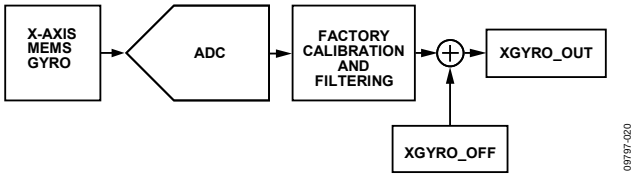


Figure 20. User Calibration, XGYRO_OFF Example

Gyroscope Bias Error Estimation

Any system level calibration function must start with an estimate of the bias errors, which typically comes from a sample of gyroscope output data, when the device is not in motion. The sample size of data depends on the accuracy goals. Figure 7 provides a trade-off relationship between averaging time and the expected accuracy of a bias measurement. Vibration, thermal gradients, and power supply instability can influence the accuracy of this process.

Table 49. XGYRO_OFF (Base Address = 0x1E), Read/Write

Bits	Description (Default = 0x0000)
[15:14]	Not used
[13:0]	X-axis, gyroscope offset correction factor, twos complement, 0.0125°/sec per LSB

Table 50. YGYRO_OFF (Base Address = 0x20), Read/Write

Bits	Description (Default = 0x0000)
[15:14]	Not used
[13:0]	Y-axis, gyroscope offset correction factor, twos complement, 0.0125°/sec per LSB

Table 51. ZGYRO_OFF (Base Address = 0x22), Read/Write

Bits	Description (Default = 0x0000)
[15:14]	Not used
[13:0]	Z-axis, gyroscope offset correction factor, twos complement, 0.0125°/sec per LSB

Gyroscope Bias Correction Factors

When the bias estimate is complete, multiply the estimate by −1 to change its polarity, convert it into digital format for the offset correction registers (Table 49), and write the correction factors to the correction registers. For example, lower the x-axis bias by 10 LSB (0.125°/sec) by setting XGYRO_OFF = 0x1FF6 (DIN = 0x9F1F, 0x9EF6).

Single Command Bias Correction

GLOB_CMD[0] (Table 32) loads the xGYRO_OFF registers with the values that are the opposite of the values that are in xGYRO_OUT, at the time of initiation. Use this command, together with the decimation filter (SMPL_PRD[12:8], Table 46), to automatically average the gyroscope data and improve the accuracy of this function, as follows:

1. Set SENS_AVG[10:8] = 001 (DIN = 0xBD01) to optimize the xGYRO_OUT sensitivity to 0.0125°/sec/LSB.
2. Set SMPL_PRD[12:8] = 0x10 (DIN = 0xBB10) to set the decimation rate to 65,536 (2^{16}), which provides an averaging time of 80 seconds (65,536 ÷ 819.2 SPS).
3. Wait for 80 seconds while keeping the device motionless.
4. Set GLOB_CMD[0] = 1 (DIN = 0xC201) and wait for the time it takes to perform the flash memory backup (~75 ms).

ACCELEROMETERS

The XACCL_ OFF (Table 52), YACCL_ OFF (Table 53), and ZACCL_ OFF (Table 54) registers provide user programmable bias adjustment function for the x-, y-, and z-axis accelerometers, respectively. These registers adjust the accelerometer data in the same manner as XGYRO_OFF functions in Figure 20.

Table 52. XACCL_OFF (Base Address = 0x24), Read/Write

Bits	Description (Default = 0x0000)
[15:14]	Not used
[13:0]	X-axis, accelerometer offset correction factor, twos complement, 0.25 m*g*/LSB

Table 53. YACCL_OFF (Base Address = 0x26), Read/Write

Bits	Description (Default = 0x0000)
[15:14]	Not used
[13:0]	Y-axis, accelerometer offset correction factor, twos complement, 0.25 m*g*/LSB

Table 54. ZACCL_OFF (Base Address = 0x28), Read/Write

Bits	Description (Default = 0x0000)
[15:14]	Not used
[13:0]	Z-axis, accelerometer offset correction factor, twos complement, 0.25 m*g*/LSB

Accelerometer Bias Error Estimation

Under static conditions, orient each accelerometer in positions where the response to gravity is predictable. A common approach to this is to measure the response of each accelerometer when they are oriented in peak response positions, that is, where ±1 *g* is the ideal measurement position. Next, average the +1 *g* and −1 *g* accelerometer measurements together to estimate the residual bias error. Using more points in the rotation can improve the accuracy of the response.

Accelerometer Bias Correction Factors

When the bias estimate is complete, multiply the estimate by −1 to change its polarity, convert it to the digital format for the offset correction registers (Table 52), and write the correction factors to the correction registers. For example, lower the x-axis bias by 10 LSB (33.3 mg) by setting XACCL_OFF = 0x1FF6 (DIN = 0xA51F, 0xA4F6).

Point of Percussion Alignment

Set MSC_CTRL[6] = 1 (DIN = 0xB846) to enable this feature and maintain the factory default settings for DIO1. This feature performs a point of percussion translation to the point identified in Figure 21. See Table 39 for more information on MSC_CTRL.

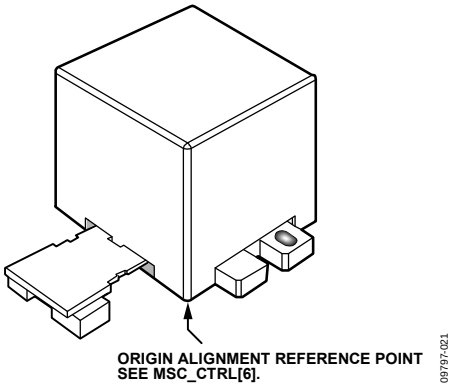


Figure 21. Point of Percussion Physical Reference

MAGNETOMETER CALIBRATION

The ADIS16407 provides registers that contribute to both hard iron and soft iron correction factors, as shown in Figure 22

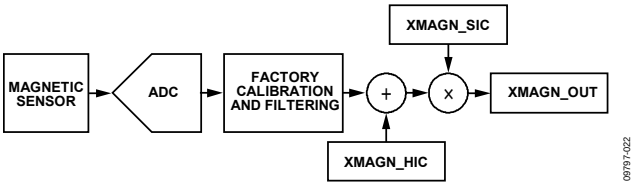


Figure 22. Hard Iron and Soft Iron Factor Correction

Hard Iron Correction

The XMAGN_HIC (Table 55), YMAGN_HIC (Table 56), and ZMAGN_HIC (Table 57) registers provide the user programmable bias adjustment function for the x-, y-, and z-axis magnetometers, respectively. Hard iron effects result in an offset of the magnetometer response.

Table 55. XMAGN_HIC (Base Address = 0x2A), Read/Write

Bits	Description (Default = 0x0800)
[15:14]	Not used
[13:0]	X-axis hard iron correction factor, twos complement, 0.5 mgauss/LSB, 0x0000 = 0

Table 56. YMAGN_HIC (Base Address = 0x2C), Read/Write

Bits	Description (Default = 0x0800)
[15:14]	Not used
[13:0]	Y-axis hard iron correction factor, twos complement, 0.5 mgauss/LSB, 0x0000 = 0

Table 57. ZMAGN_HIC (Base Address = 0x2E), Read/Write

Bits	Description (Default = 0x0800)
[15:14]	Not used
[13:0]	Z-axis hard iron correction factor, twos complement, 0.5 mgauss/LSB, 0x0000 = 0

Hard Iron Factors

When the hard iron error estimation is complete, take the following steps:

1. Multiply the estimate by −1 to change its polarity.
2. Convert it into digital format for the hard iron correction registers (Table 55).
3. Write the correction factors to the correction registers. For example, lower the x-axis bias by 10 LSB (5 mgauss) by setting XMAGN_HIC = 0x1FF6 (DIN = 0xAB1F, 0xAAF6).

Soft Iron Effects

The XMAGN_SIC (Table 58), YMAGN_SIC (Table 59), and ZMAGN_SIC (Table 60) registers provide an adjustment variable for the magnetometer sensitivity adjustment in each magnetometer response to simplify the process of performing a system level soft iron correction.

Table 58. XMAGN_SIC (Base Address = 0x30), Read/Write

Bits	Description (Default = 0x0800)
[15:12]	Not used
[11:0]	X-axis soft iron correction factor, binary format, Scale factor = 100%/2048LSB, 0x000 = 0 Example: 0x800 = 100% (factory scale unchanged) Maximum = 0xFFF = 200% − 100%/2048

Table 59. YMAGN_SIC (Base Address = 0x32), Read/Write

Bits	Description (Default = 0x0800)
[15:12]	Not used
[11:0]	Y-axis soft iron correction factor, binary format, Scale factor = 100%/2048LSB, 0x000 = 0 Example: 0x800 = 100% (factory scale unchanged) Maximum = 0xFFF = 200% − 100%/2048

Table 60. ZMAGN_SIC (Base Address = 0x34), Read/Write

Bits	Description (Default = 0x0800)
[15:12]	Not used
[11:0]	Z-axis soft iron correction factor, binary format, Scale factor = 100%/2048LSB, 0x000 = 0 Example: 0x800 = 100% (factory scale unchanged) Maximum = 0xFFF = 200% − 100%/2048

Soft Iron Factors

When the soft iron error estimation is complete, convert the sensitivity into the digital format for the soft iron correction registers (Table 58) and write the correction factors to the correction registers. A simple method for converting the correction factor is to divide it by 2 and multiply it by 4095. For example, increasing the default soft iron factor to approximately 1.15 uses a binary code for 2355, or 0x933. Increase the soft iron correction factor for the y-axis to approximately 1.15 by setting YMAGN_SIC = 0x0933 (DIN = 0xB309, 0xB233).

FLASH UPDATES

When using the user calibration registers to optimize system level accuracy, keep in mind that the register values are volatile until their contents are saved in the nonvolatile flash memory. After writing all of the correction factors into the user correction registers, set GLOB_CMD[3] = 1 (DIN = 0xC204) to save these settings in nonvolatile flash memory. Be sure to consider the endurance rating of the flash memory when determining how often to update the user correction factors in the flash memory.

RESTORING FACTORY CALIBRATION

Set GLOB_CMD[1] = 1 (DIN = 0xC202) to execute the factory calibration restore function. This is a single command function, which resets the gyroscope and accelerometer offset registers to 0x0000 and all sensor data to 0. Then, it automatically updates the flash memory within 75 ms and restarts sampling and processing data. See Table 32 for more information on GLOB_CMD.

ALARMS

Alarm 1 and Alarm 2 provide two independent alarms. Table 61 lists the alarm control registers, including ALM_CTRL (Table 66), which provides control bits for data source selection, static/dynamic comparison, filtering, and alarm indicator.

Table 61. Registers for Alarm Configuration

Register	Address	Description
ALM_MAG1	0x44	Alarm 1 trigger setting
ALM_MAG2	0x46	Alarm 2 trigger setting
ALM_SMPL1	0x48	Alarm 1 sample period
ALM_SMPL2	0x4A	Alarm 2 sample period
ALM_CTRL	0x4C	Alarm configuration

STATIC ALARM USE

The static alarms setting compares the data source selection (ALM_CTRL[15:8]) with the values in the ALM_MAGx registers listed in Table 62 and Table 63, using ALM_MAGx[15] to determine the trigger polarity. The data format in these registers matches the format of the data selection in ALM_CTRL[15:8]. See Table 67, Alarm 1, for a static alarm configuration example.

Table 62. ALM_MAG1 (Base Address = 0x44), Read/Write

Bits	Description (Default = 0x0000)
[15]	Trigger polarity 1 = greater than, 0 = less than
[14]	Not used
[13:0]	Threshold setting; matches for format of ALM_CTRL[11:8] output register selection

Table 63. ALM_MAG2 (Base Address = 0x46), Read/Write

Bits	Description (Default = 0x0000)
[15]	Trigger polarity 1 = greater than, 0 = less than
[14]	Not used
[13:0]	Threshold setting; matches for format of ALM_CTRL[15:12] output register selection

DYNAMIC ALARM USE

The dynamic alarm setting monitors the data selection for a rate-of-change comparison. The rate-of-change comparison is represented by the magnitude in the ALM_MAGx registers over the time represented by the number-of-samples setting in the ALM_SMPLx registers, located in Table 64. See Table 67, Alarm 2, for a dynamic alarm configuration example.

Table 64. ALM_SMPL1 (Base Address = 0x48), Read/Write

Bits	Description (Default = 0x0000)
[15:8]	Not used
[7:0]	Binary, number of samples (both 0x00 and 0x01 = 1)

Table 65. ALM_SMPL2 (Base Address = 0x4A), Read/Write

Bits	Description (Default = 0x0000)
[15:8]	Not used
[7:0]	Binary, number of samples (both 0x00 and 0x01 = 1)

ALARM REPORTING

The DIAG_STAT[9:8] bits provide error flags that indicate an alarm condition. The ALM_CTRL[2:0] bits provide controls for a hardware indicator using DIO1 or DIO2.

Table 66. ALM_CTRL (Base Address = 0x4C), Read/Write

Bits	Description (Default = 0x0000)
[15:12]	Alarm 2 data source selection 0000 = disable 0001 = SUPPLY_OUT 0010 = XGYRO_OUT 0011 = YGYRO_OUT 0100 = ZGYRO_OUT 0101 = XACCL_OUT 0110 = YACCL_OUT 0111 = ZACCL_OUT 1001 =XMAGN_OUT 1010 = YMAGN_OUT 1011 = ZMAGN_OUT 1100 = AUX_ADC
[11:8]	Alarm 1 data source selection (same as Alarm 2)
[7]	Alarm 2, dynamic/static (1 = dynamic, 0 = static)
[6]	Alarm 1, dynamic/static (1 = dynamic, 0 = static)
[5]	Not used
[4]	Data source filtering (1 = filtered, 0 = unfiltered)
[3]	Not used
[2]	Alarm indicator (1 = enabled, 0 = disabled)
[1]	Alarm indicator active polarity (1 = high, 0 = low)
[0]	Alarm output line select (1 = DIO2, 0 = DIO1)

Alarm Example

Table 67 offers an example that configures Alarm 1 to trigger when filtered ZACCL_OUT data drops below 0.7 *g*, and Alarm 2 to trigger when filtered ZGYRO_OUT data changes by more than 50°/sec over a 100 ms period, or 500°/sec^2. The filter setting helps reduce false triggers from noise and refine the accuracy of the trigger points. The ALM_SMPL2 setting of 82 samples provides a comparison period that is approximately equal to 100 ms for an internal sample rate of 819.2 SPS.

Table 67. Alarm Configuration Example 1

DIN	Description
0xCD47, 0xCC97	ALM_CTRL = 0x4797 Alarm 2: dynamic, Δ-ZGYRO_OUT (Δ-time, ALM_SMPL2) > ALM_MAG2 Alarm 1: static, ZACCL_OUT < ALM_MAG1, filtered data DIO2 output indicator, positive polarity
0xC703, 0xC6E8	ALM_MAG2 = 0x03E8 = 1,000 LSB = 50°/sec
0xC500, 0xC4D2	ALM_MAG1 = 0x00D2 = 210 LSB = +0.7 *g*
0xC866	ALM_SMPL2[7:0] = 0x52 = 82 samples 82 samples ÷ 819.2 SPS = ~100 ms

APPLICATIONS INFORMATION

INSTALLATION/HANDLING

For ADIS16407 installation, use the following two step process:

1. Secure the base plate using machine screws.
2. Press the connector into its mate.

For removal

1. Gently pry the connector from its mate using a small slot screwdriver.
2. Remove the screws and lift up the device.

Never attempt to unplug the connector by pulling on the plastic case or base plate. Although the flexible connector is very reliable in normal operation, it can break when subjected to unreasonable handling. When broken, the flexible connector cannot be repaired. The AN-1045 Application Note, *iSensor® IMU Mounting Tips*, provides more information about developing an appropriate mechanical interface design.

GYROSCOPE BIAS OPTIMIZATION

The factory calibration corrects for initial and temperature dependent bias errors in the gyroscopes. Use the autonull command (GLOB_CMD[0]) and decimation filter (SMPL_PRD[12:8]) to address rate random walk (RRW) behaviors. Control physical, power supply, and temperature stability during the averaging times to help ensure optimal accuracy during this process. Refer to the AN-1041 Application Note, *iSensor® IMU Quick Start Guide and Bias Optimization Tips*, for more information about optimizing performance.

INTERFACE PRINTED CIRCUIT BOARD (PCB)

The ADIS16407/PCBZ includes one ADIS16407BMLZ and one interface PCB. The interface PCB simplifies the process of integrating these products into an existing processor system.

J1 and J2 are dual row, 2 mm (pitch) connectors that work with a number of ribbon cable systems, including 3M Part 152212-0100-GB (ribbon crimp connector) and 3M Part 3625/12 (ribbon cable). Figure 23 provides a hole pattern design for installing the ADIS16407BMLZ and the interface PCB onto the same surface. Figure 24 provides the pin assignments for each connector, which match the pin descriptions for the ADIS16407BMLZ. The ADIS16407does not require any external capacitors for normal operation; therefore, the interface PCB does not use the C1/C2 pads (not shown in Figure 23).

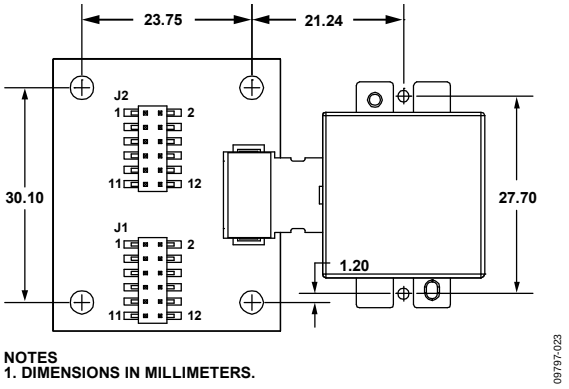


Figure 23. Physical Diagram for the ADIS16407/PCBZ

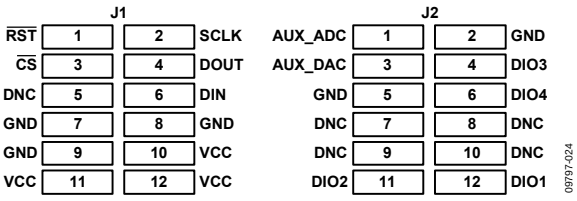


Figure 24. J1/J2 Pin Assignments

OUTLINE DIMENSIONS

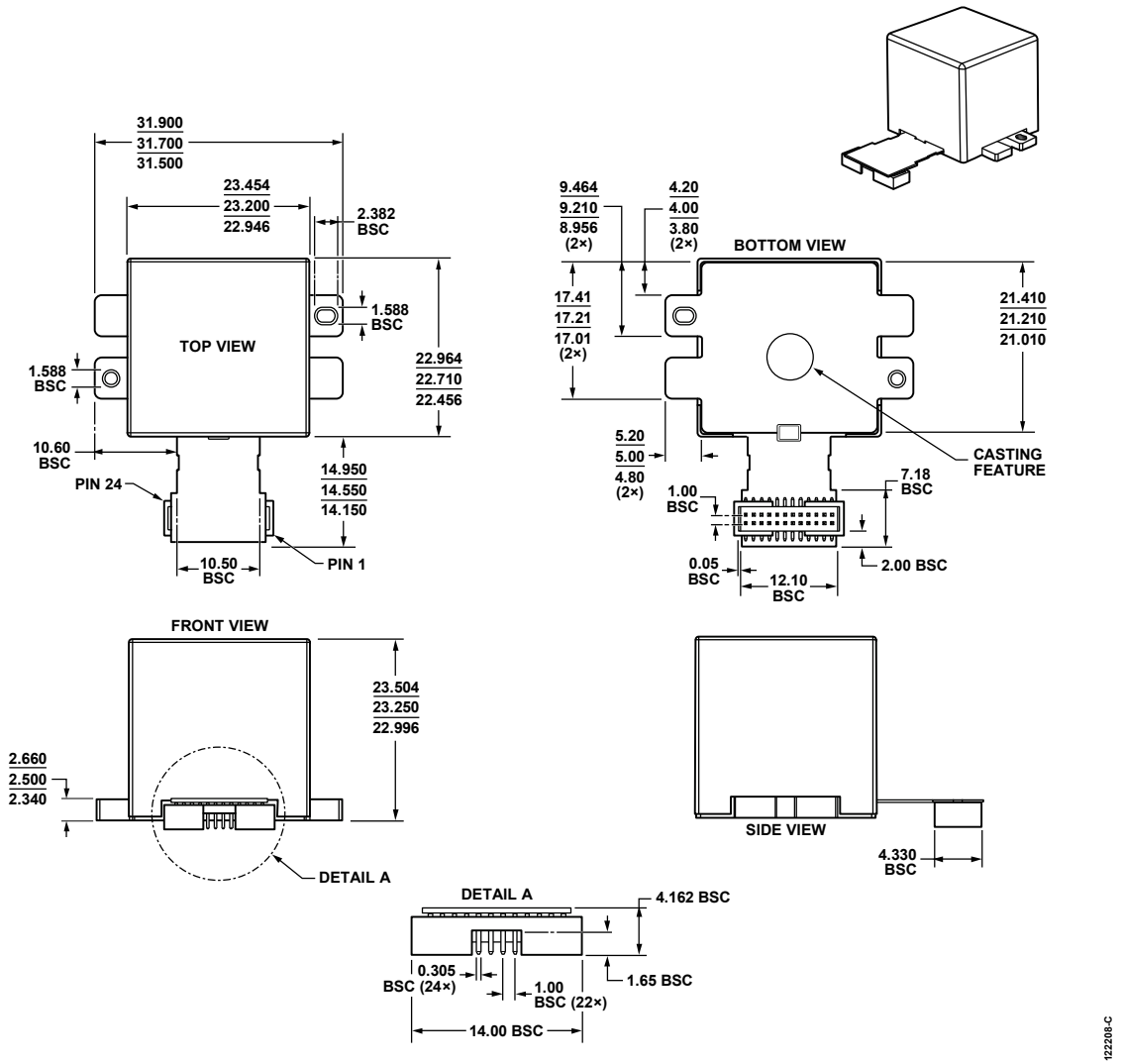


Figure 25. 24-Lead Module with Connector Interface
(ML-24-2)
Dimensions shown in millimeters

ORDERING GUIDE

Model[1]	Temperature Range	Package Description	Package Option
ADIS16407BMLZ	−40°C to +105°C	24-Lead Module with Connector Interface	ML-24-2
ADIS16407/PCBZ		Interface PCB	

[1] Z = RoHS Compliant Part.