ALMA MATER STUDIORUM - UNIVERSITY OF BOLOGNA

ADVANCED AUTOMOTIVE ELECTRONIC ENGINEERING

Master Thesis in *Applied Automatic Controls*

Design and implementation of motorbike lean angle observers

Author:
Mirko Shtylla

Academic Supervisor:
Prof. Nicola Mimmo

Company Supervisor:
Ing. Dario Zinelli

October 7, 2021

MOTORVEHICLE UNIVERISITY OF EMILIA ROMAGNA

Master degree: Advanced Automotive Electronics Engineering

Design and implementation of motorbike lean angle observers

Abstract

This thesis focuses on the investigation and the implementation of different observers for the estimation of the roll angle of a motorbike. The central core of the activity is applying a Model-Based design in order to outline, simulate and implement the filters with the aim of a final comparison of the performances.
This approach is crucially underlined among the chapters that articulate this document: first the design and tuning of an Extended Kalman Filter and a Complementary Filter in a pure simulation environment emphasize the most accurate choice for the particular problem. After this, several steps were performed in order to move from the aforementioned simulation environment to a real hardware application.
In conclusion, several sensor configurations were tested and compared in order to highlight which sensor suite gives the best performances.

Contents

List of Figures

List of Abbreviations

CAN	**C**ontroller **A**rea **N**etwork
CF	**C**omplementary **F**ilter
DBC	**D**ata **B**ase **C**an
EKF	**E**xtended **K**alman **F**ilter
HIL	**H**ardware **I**n the **L**oop
IMU	**I**nertial **M**easurement **U**nit
MCU	**M**icroprocessor **C**ontrol **U**nit
MIL	**M**odel **I**n the **L**oop
PIL	**P**rocessor **I**n the **L**oop
SIL	**S**oftware **I**n the **L**oop

Chapter 1

Introduction

The work reported in this thesis has been accomplished through an internship at Aviorace, a company which deals with the conception and production of motorsport electronic components.
The Research and Development department of the company has designed an Inertial Measurement Unit intended for an advanced automotive market: motorsport or even high-performance road vehicles. Although the measuring part of the sensor was almost ready for market deployment, the company focused on extending the sensor capabilities by implementing an attitude estimation on-board.
Attitude estimation is a highly challenging subject for a significant amount of applications. For this particular purpose, the interest was settled on one specific quantity: the lean angle.
In this chapter will be first of all offered an overview of the reasons that lead to such great interest to this particular topic and how the knowledge of this angle is exploited. A synopsis of how this problem can be faced by implementing state of the art technologies will precede the description of the approach pursued in this work.

1.1 Motivational example

The investigation of the attitude of a motorbike is a non trivial topic due to the severe dynamics to which the vehicle is exposed. Before dealing with possible solutions and algorithms, it is worth introducing the reasons why this information may be so meaningful.
By exploiting the Eulerian angles, it is possible to describe the attitude of a rigid body with three angles: ϕ for roll, θ for pitch and ψ for yaw. As mentioned before, the interest of this work relies on the roll angle, an example is shown in Figure 1.1. This can be defined as the inclination of a vehicle with respect to the vertical direction.

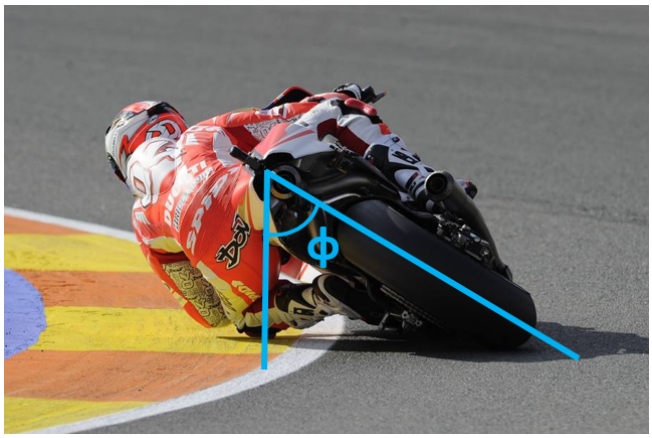


FIGURE 1.1: Lean angle ϕ of a motorbike

This information is properly combined with other data in other to implement different control strategies on the motorbike. A very common example is the integration of the lean angle in the Traction Control System as shown in the table depicted in Figure 1.2 that is an actual running strategy on a MoTeC Electronic Control Unit. The traction control is performed by the ECU in order to avoid the rear wheel slipping which can turn into an unpredictable and dangerous behaviour of the vehicle.
In the most simple configuration of this system, the ECU retrieves the speed of each wheel from two phonic wheels and calculates the slip of the motorbike as the difference between the tangential velocity at the touch point of the front and rear wheel. This speed difference will at the end multiply a gain that determines the percentage of throttle reduction to restore the grip on the rear wheel. For a stability purpose, the maximum allowed difference of speed between the two wheels may be tuned accordingly to the lean angle. This allows a better calibration since the power cut will be more severe as the motorbike is more leaned and lighter when the motorbike is in a vertical position allowing always the maximum power available while staying in a safe condition.

	Lean Angle Absolute [°]								
Gear	0.0	5.0	10.0	15.0	20.0	25.0	30.0	35.0	40.0
Sixth	11.3	11.2	11.2	10.9	10.6	9.9	8.9	7.8	6.5
Fifth	11.3	11.2	11.2	10.9	10.6	9.9	8.9	7.8	6.5
Fourth	11.3	11.2	11.2	10.9	10.6	9.9	8.9	7.8	6.5
Third	11.3	11.2	11.2	10.9	10.6	9.9	8.9	7.8	6.5
Second	11.3	11.2	11.2	10.9	10.6	9.9	8.9	7.8	6.5
First	11.3	11.2	11.2	10.9	10.6	9.9	8.9	7.8	6.5
Neutral	11.3	11.2	11.2	10.9	10.6	9.9	8.9	7.8	6.5
Default	11.3	11.2	11.2	10.9	10.6	9.9	8.9	7.8	6.5

FIGURE 1.2: Maximum slip allowed for each gear and roll angle

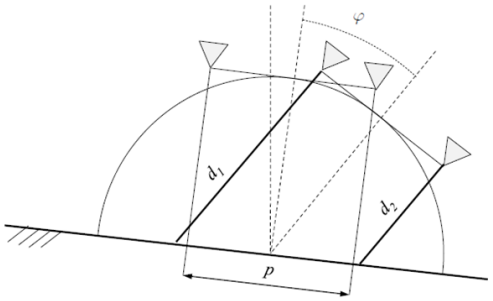
Traction Control is one of the most typical systems that uses this information but all the strategies running on the ECU can be improved and completed by it (e.g. Cornering ABS, Stability Control, Combined Breaking System).
To generalize, several algorithms implemented on the ECU base their actuation on a power cut where the proper actuator is represented by the throttle body. Among all the information, the roll angle enables an additional discrimination for a more precise control.

1.1.1 State of the art of lean angle retrieval

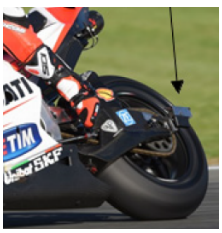
Lean angle estimation is highly discussed in literature since the problem can be faced with different approaches. Since the reference market is motorsport, the main evaluation criteria of any system are the cost, the weight and the performance.
There are two main approaches to the topic:

- Optical systems, based on time of flight sensors or video elaboration;
- MEMS sensors data fusion, algorithms that integrate information coming from different sensors.

The optical systems are preferable where the highest precision is needed and an increase of weight is not a problem. As mentioned before, one possible application is extracting the desired information by elaborating images coming from a camera mounted on-board. In Figure 1.3a instead is shown an example of how to extract the lean angle with two laser sensors based on the time of flight principle. Each sensor is mounted on the side of the rear wheel as shown in Figure 1.3b.



(A) Measure principle



(B) Laser sensor on-board

By knowing the distance p between the sensors and measuring the distances d_1 and d_2 of each sensor from the ground, it is possible to calculate the roll angle ϕ as:

$$\phi = tan^{-1}\left(\frac{d_1 - d_2}{p}\right) \tag{1.1}$$

The drawback of this system is the weight added by the sensors and mainly by the bracket holding them in position. It can be very useful however in testing other algorithms that are less precise but do not add any weight to the vehicle.

The other main approach, which will be deeply investigated in this work, consists in using sensors that may be already mounted on the bike and extrapolating the information needed by applying an appropriate algorithm.
This approach is usually performed with the aim of an Inertial Measurement Unit able to measure accelerations and angular velocities affecting the sensor. This data can be combined together with other sensors in order to improve estimation performances.

1.1.2 Company requirements

Although the different approaches presented for the particular problem, the activity didn't start from scratch but from a predefined hardware. The company in fact had already designed the sensor board and needed to add the estimation of the roll angle as an additional feature.
As described in Chapter 3, there were some minor possible hardware changes but the general approach for the problem was already defined at the beginning of the work: using the already existing IMU for a real-time estimation, eventually adding sensors for an improved esteem.
For this reason, optical sensors have not been considered for the final product but just for a possible track validation of the developed algorithm.
The main benefit of this approach is the reuse of components already existing on the motorbike without adding weight or invasive solutions.

1.2 Solutions exploited

In its most basic configuration, a 6-axis IMU provides three digital readings about the accelerations (one for each axis that constitutes a three-dimensional space) and the angular velocities. This fundamental configuration can be extended with the implementation of other sensors.
In this work, in fact, it is studied in the first place the performance of the basic IMU eventually combined with a GPS sensor that provides the vehicle inertial speed components along its three axis. After this, the IMU has been redesigned in order to extend its capabilities integrating a 3-axis magnetometer.
A backup estimation is however required to handle a failure of the GPS or magnetic field disturbances on the magnetometer. This redundancy is implemented on the simplest configuration, the 6-axis IMU, and the estimation technique here used is the complementary filter. This type of technique is quite simple to

implement since it combines the readings of accelerations and gyroscopes. On the other hand, the drawback is that it can lead to estimation errors when the sensors are subjected to low-frequency accelerations.
In order to increase the robustness of the system, a more advanced procedure is required: the Extended Kalman Filter. Indeed, it requires more quantities to be involved: as mentioned before, during this work this technique will be evaluated using first of all a GPS in order to implement the corrections and then a magnetometer.

1.3 Hardware exploited

In this Section will be presented the main hardware components that were used during the work.

FIGURE 1.4: Aviorace Inertial Measurement Unit

Figure 1.4 shows the central tool of the whole activity: the Inertial Measurement Unit designed and developed by Aviorace. The detailed datasheet of the board is reported in Appendix A whereas the development of the hardware is deepen in Chapter 3.
All the information coming from the IMU and the different configurations are read or sent through a CAN communication. The sensor board in fact is provided with a 4-pin connector (VBat, GND, CANH, CANL) and is interfaced to

a PC with the help of the CAN sniffer depicted in Fig. 1.5.

FIGURE 1.5: Peak CAN sniffer

Lastly, general laboratory instrumentation was mandatory to link and debug the several components:

- **12V DC Power Supply**: to power up the IMU or the used demoboards;
- **Digital Oscilloscope**: to debug the waveform of desired signals;
- **DSPIC33EP512MU810 Demoboard**: to test hardware configurations different from the actual prototype;
- **STM32G491RE Demoboard**: to test a hardware setup with a different MCU;
- **Microchip DSPIC33 Programmer**: to program the on-board MCU;
- **BMI160 Demoboard**: to test the IMU with any demoboard;
- **Magneto7 Demoboard**: to test the magnetometer with any demoboard.

1.4 Software exploited

Two different major phases can be distinguished by listing the software used during the activity: a first modeling phase and a following hardware implementation phase.
For what concerns the modeling part, the number of software used is minimal:

- **MATLAB**;
- **Simulink**;

As pointed out in Chapter 2, the greatest part of the functions was designed as a MATLAB function. Those functions where then linked together in a Simulink environment.
Regarding the implementation on real hardware, several software were exploited:

- **MikroC Pro**: IDE for DSPIC33 programming;
- **PCAN-Explorer 6**: tool for reading packets received by the PCAN device and designing database for CAN communication;
- **MoTeC i2 Pro**: software for elaborating logged data;
- **Fusion360**: tool for the production of 3D CAD used here for prototype design;
- **STM32CubeIDE**: integrated environment for STM32 microcontrollers configuration and firmware design;
- **Github**: hosting service that operates also as a version control system.

At last, the following software were used for documentation production:

- **Overleaf**: online tool for LaTeX production, used for drafting the thesis and periodic reports;
- **Microsoft Word**: used for the production and revision of datasheets;
- **Microsoft Power Point**: used for the preparation of the final presentation.

Chapter 2

Model setup

This chapter aims to detail better the modeling steps introduced in Chapter 1. First the kinematic model of a rigid body is presented. Different sensor setups are introduced and their implementation is verified through theoretical and simulation studies.
Then a theoretical overview of two filters, i.e. the complementary filter and the Extended Kalman filter, is given.
These first simulation steps are performed in a continuous-time environment even if to implement the algorithm on a real MCU it is mandatory to switch to a discrete-time domain.
The main reason for which it was chosen to perform this further step is because the observation properties are preserved also in the discrete-time model but easier to check and retrieve in a continuous-time one.
After all the considerations, every setup has been simulated during the Model in the loop with sinusoidal noisy inputs in order to check the consistency of the designed filters.

2.1 Kinematic model

To study the attitude of a vehicle with six Degree-of-Freedom (DoF), we consider it as a rigid body.
The relationship between the body reference frame and the inertial reference frame is depicted in Figure 2.1.

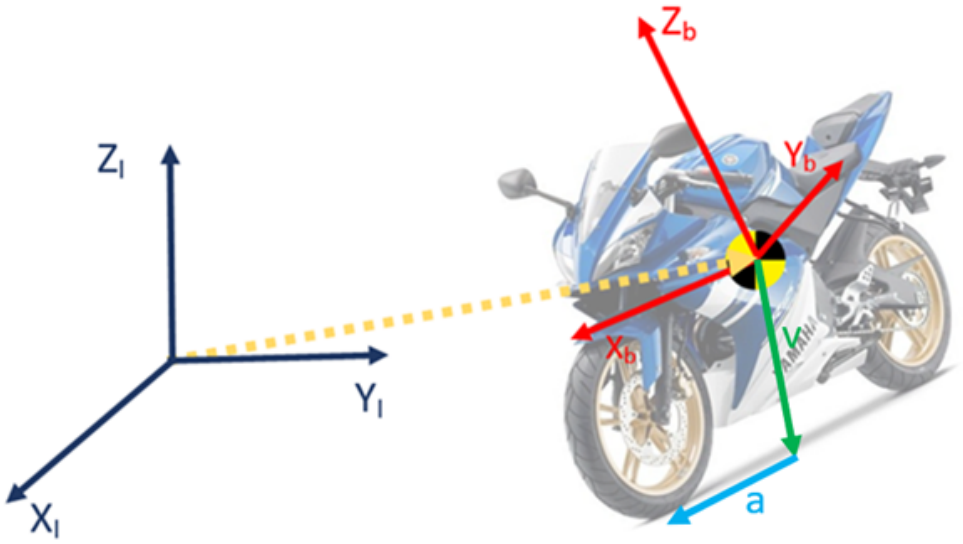


FIGURE 2.1: Quantities involved in the description of the kinematics of a rigid body

Analytically, the attitude of the rigid body with respect to the inertial space is assumed to be described by a rotation matrix with the Euler angles as parameters: ϕ for roll, θ for pitch and ψ for yaw. The kinematics is then given by System 2.1:

$$\begin{cases} \dot{v} = a \\ \dot{\Theta} = M_{\omega}(\Theta)\omega \end{cases} \tag{2.1}$$

Where v is a vector that represents the velocity of the vehicle along its three axis and a the accelerations. Θ instead is a vector that collects the three Euleran angles that describe the rotations of the rigid body in space: ϕ, θ and ψ.
The two inputs of the kinematic model will be given by the readings of the IMU $\boldsymbol{a}$ and $\boldsymbol{\omega}$ whereas the matrix $M_{\omega}(\Theta)$ is defined as:

$$M_{\omega}(\Theta) = \begin{bmatrix} 1 & sin(\phi)tan(\theta) & cos(\phi)tan(\theta) \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi)sec(\theta) & cos(\phi)sec(\theta) \end{bmatrix} \tag{2.2}$$

The system can be also equipped with a sensor suite composed by a GPS and a magnetometer whose readings will be performed as:

$$\begin{cases} y_1 = v + \nu \\ y_2 = R(\Theta)m(p) + \nu \end{cases} \tag{2.3}$$

Even if this sensor suite is here presented in a compact form, it will be investigated during all the stages separately. First exploiting the IMU with only the GPS and then the IMU with only the magnetometer.

Given the rotation matrix from inertial to body:

$$R(\Theta) = \begin{bmatrix} cos\psi cos\theta & sin\psi cos\theta & -sin\theta \\ cos\psi sin\theta sin\phi - cos\phi sin\psi & cos\phi cos\psi + sin\phi sin\psi sin\theta & cos\theta sin\phi \\ sin\phi sin\psi + cos\phi cos\psi sin\theta & cos\phi sin\psi sin\theta - cos\psi sin\phi & cos\phi cos\theta \end{bmatrix} \tag{2.4}$$

it is possible to define the two inputs of the system as:

$$\begin{cases} u_1 = R(\Theta)(a - g) + w_1 \\ u_2 = \omega + w_2 \end{cases} \tag{2.5}$$

The overall non-linear plant can be written down emphasizing the states as:

$$\begin{cases} \dot{v} = R(\Theta)^T(u_1 - w_1) + g \\ \dot{\Theta} = M_\omega(\Theta)(u_2 - w_2) \end{cases} \tag{2.6}$$

The compact form of this general non linear system can be summarized as:

$$\begin{cases} \dot{x} = f(x, u - w) \\ y = h(x) + v \end{cases} \tag{2.7}$$

Where x represents the vector of states composed by the inertial speeds and angles, f the state-transition model from Equation 2.6, u the inputs coming from the IMU, y the outputs of the system, h the sensor model, w and v the noises affecting the process.

2.2 Complementary filter setup

The advantage of implementing a complementary filter relies on its simplicity. This has to be intended on different layers. First of all the set of sensors that is needed for an attitude estimation is minimal: only the accelerations and angular velocities are needed, indeed.

The second advantage is that the algorithm is very simple. This allows to implement this type of filter even in applications that do not have powerful electronics on-board able perform complex calculations.
The basic principle behind this type of filter is quite simpke: by integrating an angular velocity over time, it is possible to retrieve the angular position for every time instant. This however is possible only in an ideal application where the sensor is not subjected to any type of error.
In practice, the readings of the sensor can be quite noisy and, since the working principle implements an integration over time, the error is summed up through the time interval. This summation gives rise to a divergent estimation.
In order to overcome this limitation and try to improve the estimation, acceleration data are used.
The advantage is that no integration has to be performed meaning that this approach those not have a divergent behaviour. The drawback however is that the accelerometers are sensitive to external force acting on the sensor at high frequencies.
Given those considerations, the approach that gives the best results relies on a combination of the two: gyroscopes are very reliable in a short time interval but diverge in a longer one, accelerometers are stable on a long time interval but subject to severe noise at high frequencies.
Analytically the CF has a structure characterized by two filter: a low-pass and a high-pass. As shown in Figure 2.2, there are two signals x_1 and x_2 which are low- and high-frequency noise-corrupted versions of the signal x.

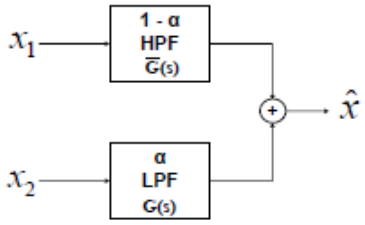


FIGURE 2.2: Low- and high-pass filters of complementary filter

$G(s)$ is the transfer function of the Low-pass filter while $\bar{G}(s)$ is the transfer function of the High pass such that $\bar{G}(s) + G(s) = 1$.
By filtering the signals and filtering them, it is possible to retrieve the output of a general CF as:

$$\hat{x} = x_1 G(s) + x_2 \bar{G}(s) \tag{2.8}$$

For attitude estimation, this general approach can be applied as follows: gyroscope estimates are applied to x_1 while the accelerations to x_2. Applying it to the filter it is possible to write the estimation of $\hat{x}$ as:

$$\hat{x} = \alpha \left(\int \dot{x}_g \, dt \right) + (1 - \alpha) x_a \tag{2.9}$$

Where α represents a weighting coefficient for gyroscope and accelerometer estimates such that $\alpha \in [0\,1]$.
Equation 2.9 can be written in Laplace domain as:

$$\hat{x}(s) = \frac{1}{s} \left(\frac{a}{1 - as} \right) x_g(s) + \left(\frac{1}{1 - as} \right) x_a(s) \tag{2.10}$$

in order to highlight the transfer functions of the HPF and the LPF shown in Figure 2.2.
While x_g and x_a are the estimates of gyroscopes (ϕ_g, θ_g and ψ_g) and accelerations (ϕ_a, θ_a and ψ_a) calculated as:

$$\dot{\phi}_g = p + qsin\phi tan\theta + rcos\phi tan\theta \tag{2.11}$$

$$\dot{\theta}_g = qcos\phi - rsin\theta \tag{2.12}$$

$$\dot{\psi}_g = p + qsin\phi sec\theta + rcos\phi sec\theta \tag{2.13}$$

$$\phi_a = tan^{-1} \left(\frac{a_y}{a_z} \right) \tag{2.14}$$

$$\theta_a = tan^{-1} \left(\frac{-a_x}{a_y sin\phi + a_z cos\phi} \right) \tag{2.15}$$

$$\psi_a = tan^{-1} \left(\frac{a_z sin\phi - a_y cos\phi}{a_x cos\theta + a_y sin\theta sin\phi + a_z sin\theta cos\phi} \right) \tag{2.16}$$

p, q and r represent the angular velocities read on x, y and z axis while a_x, a_y and a_z the accelerations on these axys.

2.3 Extended Kalman Filter setup

EKF is a very powerful observer suitable for the observation of non-linear systems.
In this particular work it has been deeply exploited and tested in order to outline its performances. Opposite to the Complementary filter in fact, the EKF is able to estimate and use as input data the covariance of the measuring noise that affects the sensors. This approach allows to re-adapt automatically instant by instant the weight and "trust" that is given to each reading.
This filter however has two major limitations:

- Because its non-linear nature, its convergence is only locally guaranteed;
- it requires a significantly higher computational power with respect to the complementary filter.

The thesis aims to identify which set of sensors provides the best reduction of the estimation error. The base sensor is obviously the 6-axis IMU for each setup, this has been completed with a GPS in a first instance and after with a magnetometer; the performances are then compared in Chapter 4.
In order to gain a better confidence with this filter, every configuration is firstly tested with a continuous time model. However, since the final goal is a hardware implementation, it becomes mandatory to switch to a discrete time model.
First it is reported a general overview of the steps required to implement a general EKF abstracted from any particular implementation. Then it is reported every specific model injected in these equations to perform the estimation.

2.3.1 Continuous time Extended Kalman Filter

Given a general non linear-system as described in Equation 2.7, an EKF can be implemented as:

$$\begin{cases} \dot{\hat{x}} = f(\hat{x}, u, 0) + L(t)(y - \hat{y}) \\ \hat{y} = h(\hat{x}) \end{cases} \tag{2.17}$$

Where $L(t)$ represents the Kalman Gain and is calculated as:

$$L(t) = PC^T(t)R_d^{-1} \tag{2.18}$$

and P is the solution of a Differential Riccati Equation determined by:

$$\dot{P} = A(t)P + PA^T(t) + B(t)Q_{d11}B^T(t) - PC^T(t)R_d^{-1}C(t)P \tag{2.19}$$

In Equations 2.18 and 2.19 three matrices appear: $A(t)$, $B(t)$ and $C(t)$. Those depend on the particular model that is under investigation but have to be explicitly calculated since the system is non-linear.
The equations to retrieve those quantities are:

$$A(t) := \left.\frac{\partial f}{\partial x}\right|_{\widehat{x},u,0} B(t) := \left.\frac{\partial f}{\partial w}\right|_{\widehat{x},u,0} C(t) := \left.\frac{\partial h}{\partial x}\right|_{\widehat{x}} \tag{2.20}$$

Matrices Q_{d11} and R_d instead are related to the noise affecting the process and the output, those can be calculated as:

$$Q_{d11} = \delta(t-\tau)E\begin{bmatrix} w_1(\tau)w_1^T(t) & \cdots & w_1(\tau)w_n^T(t) \\ \vdots & \ddots & \vdots \\ w_m(\tau)w_1^T(t) & \cdots & w_m(\tau)w_n^T(t) \end{bmatrix} \tag{2.21}$$

$$R_d = \delta(t-\tau)E\begin{bmatrix} \nu_1(\tau)\nu_1^T(t) & \cdots & \nu_1(\tau)\nu_n^T(t) \\ \vdots & \ddots & \vdots \\ \nu_m(\tau)\nu_1^T(t) & \cdots & \nu_m(\tau)\nu_n^T(t) \end{bmatrix} \tag{2.22}$$

Those quantities are retrievable through the datasheets of the particular sensors implemented, they can then be tuned in order to improve the performances of the filter.

2.4 Observability study

A system is observable if its states at a certain time instant can be uniquely determined given a finite sequence of its outputs. Considering the full set of states:

$$x = \begin{bmatrix} v \\ \Theta \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ \phi \\ \theta \\ \psi \end{bmatrix} \tag{2.23}$$

These can be divided in two sets: an observable part and an unobservable part.

$$x = \begin{bmatrix} x_o \\ x_{no} \end{bmatrix} \tag{2.24}$$

For this particular application, since the roll angle ϕ is the quantity of interest, it can be acceptable if some of the other states appear to be unobservable.
The aim of this section is verifying the observability properties of different sensor setups. Even if the continuous-time simulations used for this check can not be implemented on real hardware, they are very useful since these properties hold also when the system is discretized.
The three setups considered for the test are:

- **IMU and GPS**, accelerometers and gyroscopes along GPS velocities projected on the three axis;
- **IMU and magnetometer**, accelerometers and gyroscopes with magnetic field readings on three axis;
- **IMU**, only accelerometers and gyroscopes.

The observability of a Linear Time-Invariant system can be tested by checking the rank of matrix O defined as:

$$O := \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix} \tag{2.25}$$

The system however is time variant, indeed matrices C and A change their values continuously.
For this reason the tests were performed calculating at a continuous time also

the numerical values of these two matrices and building in this way a continuous-time observability matrix with not constant eigenvalues.
A second check was possible for the setups that were observed by means of an Extended Kalman Filter. Exploiting matrix P from Equation 2.19, it is possible to check its eigenvalues. If the eigenvalue with the lowest magnitude does not tend to zero in a given time span the system is "uniformly" observable.
These two tests lead to the following conclusions:

- **IMU and GPS** not fully observable: the yaw angle is unobservable;
- **IMU and magnetometer** not fully observable: the three velocities are unobservable;
- **IMU** not observable.

2.5 Simulink block diagram

The simulation of the model in each configuration has been performed in a Simulink environment. Several block diagrams were designed and as a result of the chosen design it was possible to conduct all the simulations just adapting the **Observer** block according to the sensor suite and the chosen filter (EKF or CF).

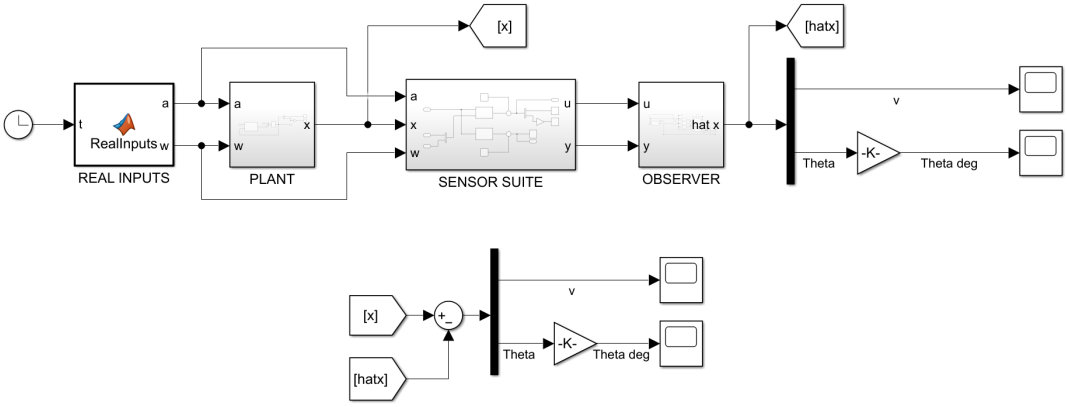


FIGURE 2.3: Continuous time plant

Here is a brief explanation of each block that shows up:

- **Clock**: injects a continuous time to the model;
- **Real Inputs**: generates sinusoidal signals for accelerations and gyroscopes;
- **Plant**: implements Equation 2.1 for calculating the real state;
- **Sensor suite**: implements Equation 2.3 in order to build up GPS or magnetometer readings and to add noise to all the sensors;
- **Observer**: receives noisy sensor readings in order to estimate the states.

For what concerns the implementation of the complementary filter, for simplicity reasons it was directly implemented the "complementaryFilter" object provided by Matlab. This tool has all the properties described in Section 2.2 and allowed a very quick and simple implementation.

2.5.1 Design of the Extended Kalman Filter

The Extended Kalman Filter, enlarged in Figure 2.4, has been designed by expoliting Matlab functions with the proper equations inside and then linked together.

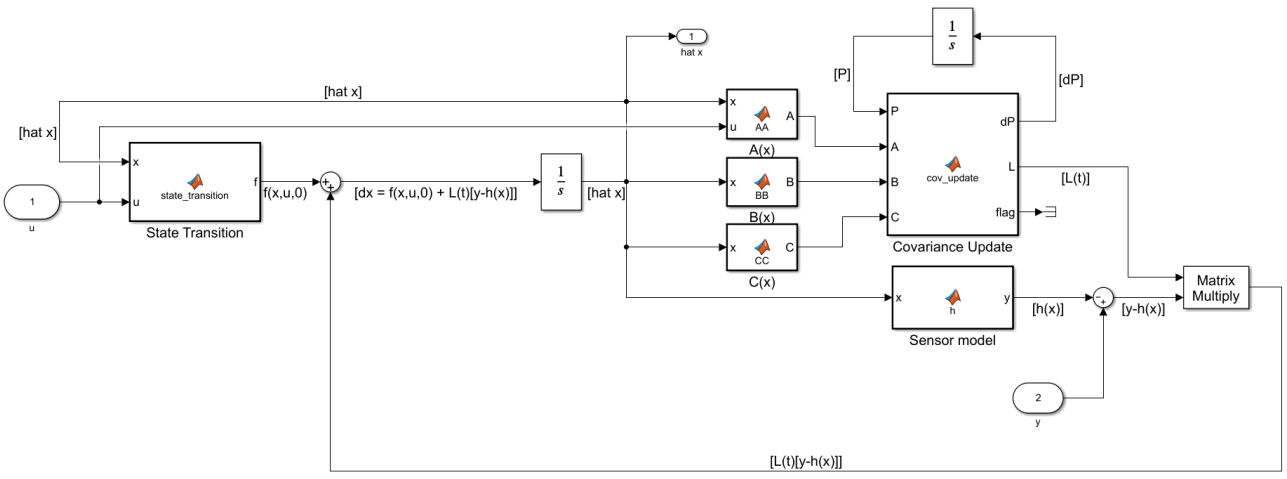


FIGURE 2.4: Continuous time Extended Kalman Filter with GPS

Since this filter has been used both with IMU+GPS and IMU+magnetometer setups, the **Sensor model** and **C(x)** matrices have been properly adapted

based on the current simulation configuration.
Starting from the center of the Figure, it is possible to find three blocks that are calculating matrices A, B and C. The calculation of these matrices was described in an abstracted way by Equation 2.20. For this particular configuration, the linearization of the plant leads to the following quantities:

$$A(t) = \begin{bmatrix} 0 & \frac{\partial R}{\partial \phi} u_1 & \frac{\partial R}{\partial \theta} u_1 & \frac{\partial R}{\partial \psi} u_1 \\ 0 & \frac{\partial M_\omega}{\partial \phi} u_2 & \frac{\partial M_\omega}{\partial \theta} u_2 & 0 \end{bmatrix} \tag{2.26}$$

$$B(t) = - \begin{bmatrix} R^T & 0 \\ 0 & M_\omega \end{bmatrix} \tag{2.27}$$

$$C(t) = \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & \frac{\partial R}{\partial \phi} m & \frac{\partial R}{\partial \theta} m & \frac{\partial R}{\partial \psi} m \end{bmatrix} \tag{2.28}$$

Where R is the rotational matrix described in Equation 2.4 and M in Equation 2.2. Matrix A moreover uses the inputs coming from the IMU. For these reasons the three blocks receive constantly the updated state and input.
Matrices A, B and C, along the current state, feed the covariance update block. This block simply implements Equations 2.18 and 2.19 in order to calculate and output the optimal Kalman Gain $L(t)$.
As indicated by Equation 2.17, this Kalman gain has to be multiplicated by the difference between the real noisy sensor readings and the expected readings. The expected readings are calculated in the "Sensor model" block which simply implements Equation 2.3: knowing the current state of the system and injecting it in the equation it is possible to find out the expected reading of the sensor.
$L(t)$ and $[y - h(x)]$ are multiplied together and sent back to a summation block. The other input of this block is the state transition. This calculation is performed by taking the previous state of the system and the current noisy inputs sent by the IMU. These data are elaborated by means of Equation 2.6 which outputs the first derivative of the state.
The summation block has in this way as inputs the two terms of Equation 2.18: $f(\hat{x}, u, 0)$ and $L(t)(y - h(x))$. By integrating the result it is possible to retrieve the current states of the overall system.

2.6 Continuous-time simulations

In order to check the consistency of the designed filters, some simulations were performed. The aim of this simulation was not to have a quantitative value of the error performed but just a qualitative one.
The simulations were performed with the aforementioned sinusoidal inputs and the results are shown in the Figures below.

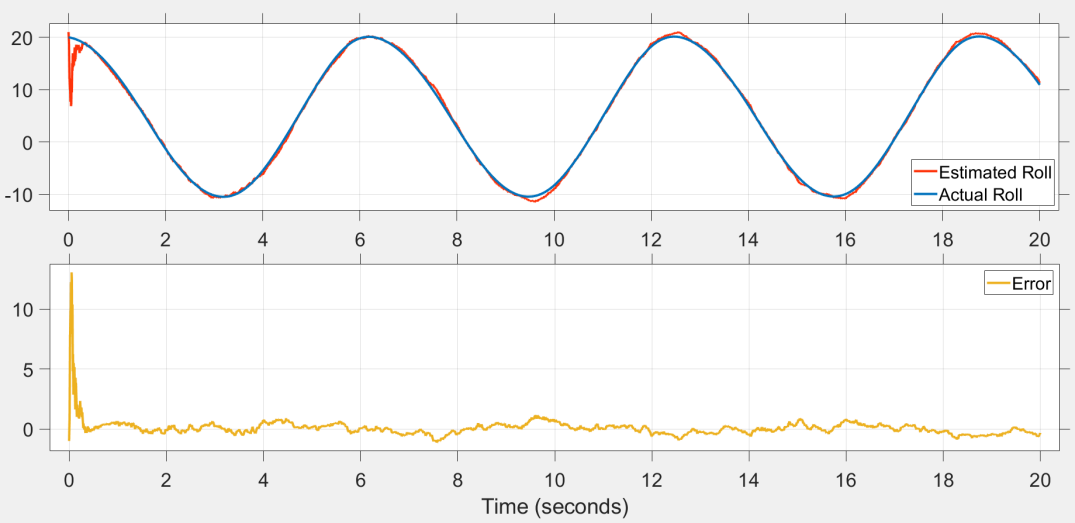


FIGURE 2.5: IMU and GPS fused in continuous time with EKF

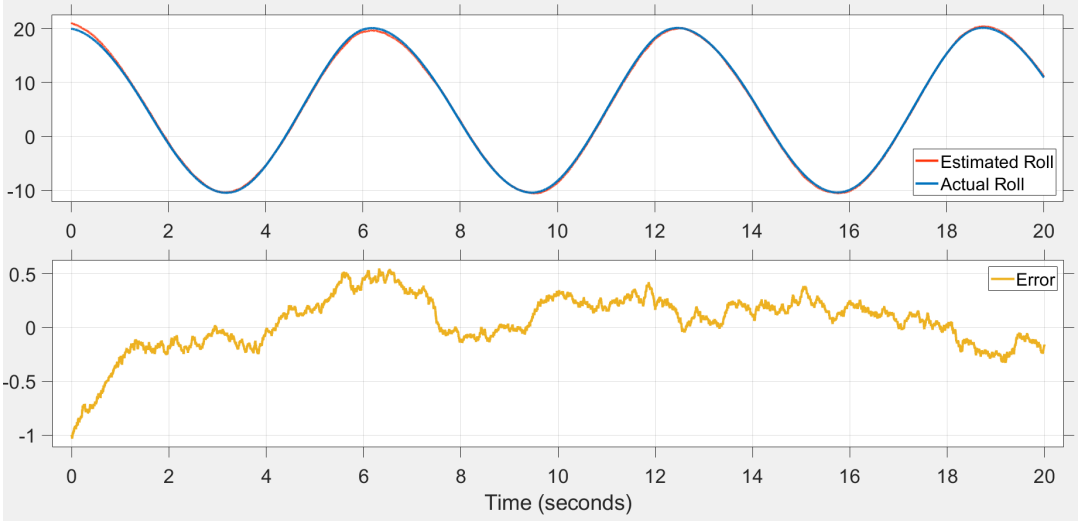


FIGURE 2.6: IMU and magnetometer fused in continuous time with EKF

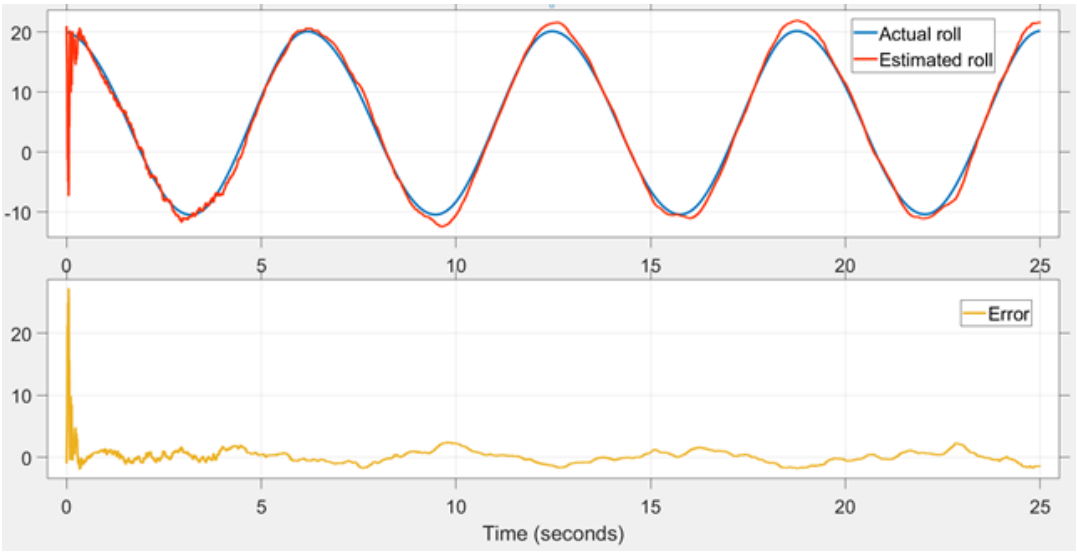


FIGURE 2.7: IMU in continuous time with Complementary Filter

The verification of the filters design is confirmed for every setup, the roll angle in fact is well estimated. The IMU and magnetometer setup is here showing the best performances in terms of error, this parameter however will be verified in Chapter 4 in a discrete-time domain.

Chapter 3

Discrete-time design and hardware implementation

In this chapter are reported the main topics handled in order to move from a continuous-time simulation environment to a hardware implementation. While, in fact, Simulink helps identifying which setup gives the best performances, only a real implementation points out the best choice in a real-world environment.
Discretization is mandatory for hardware application since obviously all MCUs operate in a discrete time domain. First of all are listed the steps that are required in order to move from a continuous-time model to a discrete-time one. Besides the model, also the filters need a readaptation to this simulation environment.
Once the simulation setups are presented, it is possible to move to an overview of the main hardware components that articulate the Inertial Measurement Unit; the detailed datasheet of this product is available in Appendix A.
The second phase is showing the steps that allow to translate Simulink model to C-code once the architecture of the Micro-controller is sufficiently known.
At the end are presented some encountered issues and the solution adopted along some necessary operations carried out to ensure the correct behaviour of the system.

3.1 Discretization of the model

There are many algorithms to solve ordinary difference equations (ODEs) that do not admit an analytical solution. The one implemented in this work is the so-called "forward Euler" and has been chosen for its simplicity with respect to other more advanced techniques like the "Runge-Kutta" methods. This algorithm, in particular, has been exploited in order to discretize the state transition model.

Given a general ODE:

$$\dot{y} = f(t,\, y(t)) \tag{3.1}$$

Its time derivative can be approximated as:

$$\dot{y} \approx \frac{y(t+h) - y(t)}{\Delta t} \tag{3.2}$$

After expliciting $y(t+h)$ and substituting $\dot{y}$ it is possible to get:

$$y(t+h) = y(t) + f(t,\, y(t)) \cdot \Delta t \tag{3.3}$$

By fixing a constant time step h, it is possible to change Equation 3.3 to a sequence:

$$y_{n+1} = y_n + f(t_n, y_n) \cdot \Delta t \tag{3.4}$$

The constant time step has been fixed according to the requirements of the company. The desired refresh rate of the lean angle is in fact at $10Hz$ meaning a Δt of $100ms$.
Beside its simplicity, the drawback of this method is a higher discretization error with respect to other techniques. For this particular application however this error is negligible, indeed it is severely lower than the estimation error performed by the filters.

3.2 Filters in discrete-time

3.2.1 Complementary filter

The discrete time complementary filter is quite straightforward to get: considering the continuous time equation 2.9, the discretized version will simply be:

$$\hat{x}(n) = \alpha[\hat{x}(n-1) + T \cdot x_g(n)] + (1-\alpha)x_a(n) \tag{3.5}$$

As for the previous case, x_a and x_g represent the Eulerian angles estimated respectively with the aim of accelerometers and gyroscopes while α is a weighting coefficient. The most important difference is that the integration is not performed anymore in a continuous time but as a discrete integral.

3.2.2 Extended Kalman Filter

In continuous time the operations of Prediction and Update are fused together in equations 2.18 and 2.19 since it is not necessary to distinguish two separate time instants.
This however is not possible in a discrete domain where the process needs to be predicted and updated at specific intervals given by the sampling period of the sensors.
The first thing to be discretized is obviously the model of the non-linear system:

$$\begin{cases} \hat{x}_k = f_{k-1}(x_{k-1}, u_{k-1}, 0) \\ y_k = h_k(x) \end{cases} \tag{3.6}$$

First step of a discrete EKF is the "Prediction". This step provides the prediction of both the state vector and covariance matrix. The time instant k refers to the current sample while $k-1$ to the previous one. The quantities that have been predicted but not yet updated are marked with a $^-$ sign as apex.

$$\begin{cases} \hat{x}_k^- = f_{k-1}(\hat{x}_{k-1}, u_{k-1}, 0) \\ \hat{x}_k = \hat{x}_k^- + K_k \cdot \tilde{y}_k \\ P_k^- = A_k P_{k-1} A_k^T + Q_k \end{cases} \tag{3.7}$$

After the predictions of the state and covariance matrix are performed, it is possible to move to the update step. First of all is computed the measurement residual as:

$$\tilde{y}_k = y_k - h(\hat{x}_k^-) \tag{3.8}$$

The same step has to be done for the covariance:

$$S_k = C_k P_k^- C_k^T + R_k \tag{3.9}$$

Now is possible to calculate a near-optimal Kalman gain as:

$$K_k = P_k^- C_k^T S_k^{-1} \tag{3.10}$$

And finally the real update step is performed, firstly for the state estimate:

$$\hat{x}_k = \hat{x}_k^- + K_k \cdot \tilde{y}_k \tag{3.11}$$

and for the covariance estimate:

$$P_k = (I - K_k C_k) P_k^- \tag{3.12}$$

Similarly to what was done for the continuous time case, also in this filter it is needed to have a linearization of the discrete time model, this is performed as:

$$A_k := \left.\frac{\partial f}{\partial x}\right|_{\widehat{x}_{k-1}, u_k} \quad C_k := \left.\frac{\partial h}{\partial x}\right|_{\widehat{x}_{k-1}} \tag{3.13}$$

3.3 Simulink model

The *complementaryFilter* object provided by Matlab was again exploited for the CF simulation. This block implements directly equation 3.5 and offers a set of tools for tuning properly the weighting coefficient.
The Extended Kalman Filter was instead designed from the beginning. This filter was again tested with two different setups: IMU with GPS and IMU with magnetometer.
Contrary to the continuous-time case, it is mandatory to consider some constraints on the sampling rate of the sensors. While the IMU and the magnetometer are able to offer a frequency of $100Hz$, this is not possible for the GPS: a typical GPS used in motorsport environment is in fact limited to $10Hz$. Because of this, the EKF applied to the IMU and GPS is a Multi-rate filter. This involves several considerations that will be made in the following sections.
The general setup of the EKF simulations in Simulink environment is the one showed in Figure 3.1.

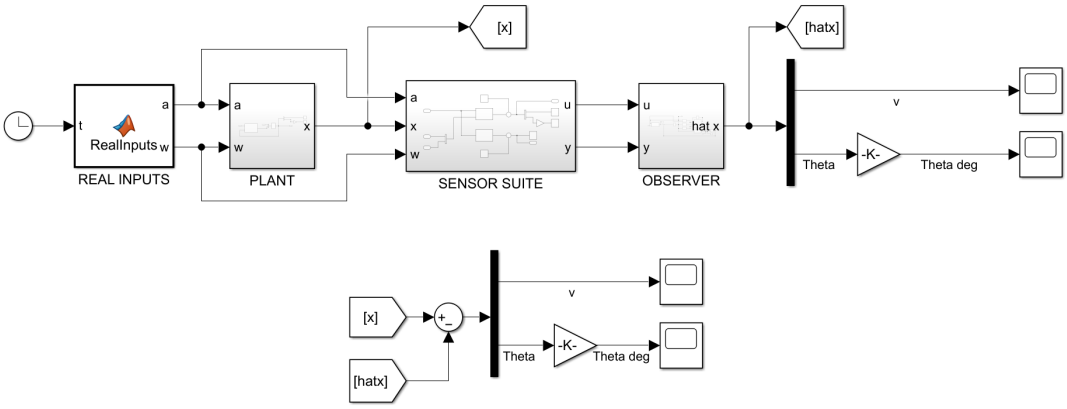


FIGURE 3.1: Discrete time plant

The blocks in this case produce a discrete time signal by exploiting the *samplingTime* property offered by Simulink. Although the structure remains the same among all the setups, the **Observer** block has been properly adapted for each configuration. This block is deepen both for the IMU+GPS and for the IMU+magnetometer cases.

3.3.1 EKF with IMU and GPS

As mentioned before, the complication of the IMU +GPS configuration is that the GPS sensor here considered provides its readings at a 10 Hz frequency against the 100 Hz frequency of the IMU. This is a critical matter since the correction may be performed only when GPS readings are available.
A focus on the Observer designed for this particular setup is shown in Figure 3.2.

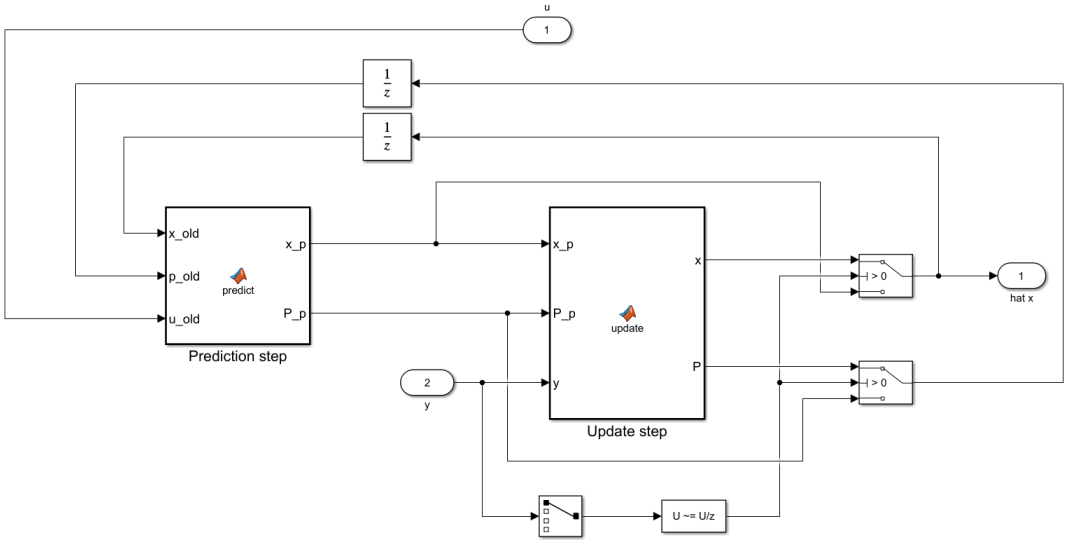


FIGURE 3.2: Discrete time Extended Kalman Filter with GPS

This model exploits only two Matlab functions: the prediction step and the update step. Besides these two blocks, the other part of the model is used to raise a flag whenever GPS data are available in order to enable the correction. Prediction step is used to implement system 3.7. It uses as input the previous state updated, the previous P solution updated and the current IMU measures. The update step instead is used only when GPS data are available and in particular it implements all the stages described by Equations from 3.8 to 3.12.

3.3.2 Discrete time model

The plant for the discrete time model of the magnetometer configuration is simpler with respect to the GPS one. In this case in fact, the magnetometer will be directly implemented on the board and read through an I2C communication instead of having it as an external sensor like a GPS.
This allows to push the magnetometer to the same fetching frequencies of the IMU. The chosen frequency is of 100Hz, in Section 3.6 will be deeply explained the choice of the frequency and some problems faced with it.

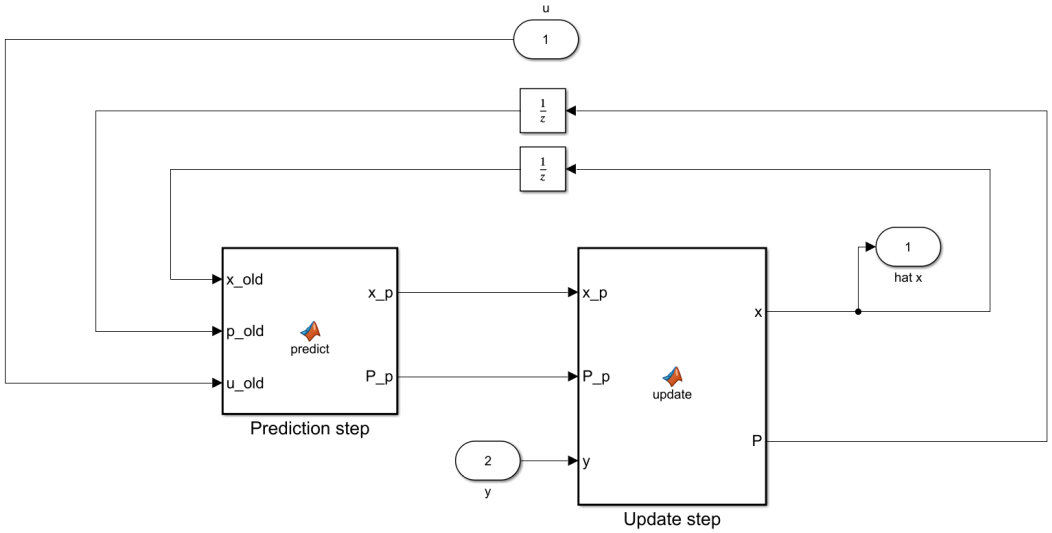


FIGURE 3.3: Discrete time Extended Kalman Filter with magnetometer

As mentioned before, the Simulink setup is simpler as can be noted in Figure 3.3. All the part related to the updating sensor is missing here since IMU and magnetometer are fetched at the same time instant.

3.4 Sensor board overview

The Inertial Measurement Unit designed and produced by Aviorace is a complete board that is composed by the following elements:

- **DSPIC33EP512MU810 Micro-controller**: 16-bit MCU produced by Microchip;
- **BMI160**: 6-axis Inertial Measurement unit produced by Bosch;
- **CAN Transceiver**: converter of data in order to enable CAN communication.

After these three elements, during the work has been added directly over the board a Magneto 7 click that is a demonstration board equipped with a magnetometer with the pins soldered directly on the I2C communication pins of the MCU.
The sensor moreover is equipped with a 4-pin DTM04 motorsport connector with a IP68 certification as the one shown in Figure 3.4.

FIGURE 3.4: DTM04

These 4 pins simply carry a 12V battery power supply, ground, CAN High and Can Low. All the needed information are in this way given by the IMU on the CAN bus. A detailed CAN database has been designed during the activity and for any algorithm provides the following readings:

- Acceleration on each axis with a resolution of 1 mG;
- Angular velocity on each axis with a resolution of 0.1 °/s;
- Roll, pitch and yaw angles with a resolution of 0.1°.

The detailed DBC is supplied to the customer among a software. This software, deeply exploited during the activity, has been developed by Aviorace and allows to read all the incoming data through a CAN sniffer and to send configuration parameters to the IMU as anticipated in Section 1.3.

3.5 Transition from Model to Code

The transition from model to code has been a very challenging activity. Since the model is quiet complex and the time available not so massive, it was exploited the Embedded Coder supplied by Simulink.
This is a very powerful tool that allows to generate code starting from a model or just a function inside of it. It was fundamental since writing by hand a code that can perform operations regarding previous and current states can be very tough.
The main feature is the generation of a reusable C function that can be imported in the code, linked to IMU readings and executed at a certain rate. For example, in the simplest case of IMU and magnetometer configuration, an interrupt was generated by the MCU every 10 ms. This interrupt called a routine were first of all the readings of the IMU and of the magnetometer were fetched. These were then saved and passed to the generated function that implemented the EKF. Once the function-call is completed it returns the attitude values estimated and

all these quantities are then sent via CAN to all the nodes connected to the bus.
The generation of this code, despite is well guided, is not trivial. Through the embedded coder is possible to set many parameters in order to generate code that is suitable for the particular MCU chosen. Once this code is able to run on the MCU, many parameters can be tuned in order to improve the efficiency of the execution.
The choice of these parameters was in part sustained by some tools offered by the embedded coder. Those tools relate to different layers of the Advisor on Simulink. The Advisor is a huge set of testing setups already predefined, by setting some of the targets that the code has to accomplish, this Advisor automatically suggests some tuning of the available parameters. In particular, these are the exploited features:

- **Model Advisor**: to verify if model complies with modeling guidelines;
- **Upgrade Advisor**: to tune modeling parameters;
- **Code Generation Advisor**: to verify the compliance of the code;
- **Performance Advisor**: to tune generation parameters in order to improve the efficiency of the code.

The overall system is created with a Model-based design. This approach is properly underlined in Chapter 4 where the steps are accurately reported.

3.6 Issues and optimization

The IMU has the possibility to be programmed directly via CAN-bus, this is a great advantage considering that once it is assembled it is hermetically sealed. The re-programming of the IMU is possible by linking the CAN pins to the PC by means of a CAN sniffer. By exploiting the same instrument, it is then possible to visualize all incoming data like accelerations, gyroscopes, temperature. An important information provided is the cycle time of the information, that is how frequently a packet on a given CAN address is received.
Thanks to this last feature it was possible to spot some issues. The heaviest problem was the implementation of the generated code on the sensor. Once minor bugs were solved and the IMU was actually estimating the angles, the cycle time of every packet was considerably higher than the desired refreshing frequency. This is because of the architecture of the MCU equipped: the 16-bit and the lack of the floating point unit gives rise to significant effort for difficult calculations.
The IMU has a sequence of three tasks with no operative system managing them:

- Reading acceleration and gyroscopes;
- Calling the EKF function with the read quantities;
- Sending all the information on the CAN bus.

Since those operations are performed sequentially, the delay of just one of those implies an expansion of the overall cycle time.
The issue here encountered regards the execution time of the EKF function. This problem regarded both the IMU and GPS configuration and the IMU and magnetometer one. However, the two problems were solved in a completely different way.
For what concerns the IMU and GPS model, the initial desired refreshing frequency was 100 Hz. The execution time faced with this setup was of 35 ms reducing the frequency to almost 28 Hz. This is not only a matter of how frequently the angles have to be updated but also a matter of how good the filter is performing: the fastest refreshing rate in fact allows to get closer to a "continuous time" simulation.
The trade-off chosen was lowering the refreshing rate to 50 Hz and doing a huge optimization in order to reduce the execution time of the tasks listed above to less than 20 ms. This optimization has been performed on two different layers:

- Exploiting the tools listed in Section 3.5 setting as target the execution time in order to improve code efficiency;
- Redesigning the code already existing on the IMU that dealt with the management of tasks.

For what concerns instead the IMU and magnetometer configuration, the same problem was faced but in a significantly higher way. The execution time with this setup, in fact, was up to 100ms reducing the frequency to 10Hz. This frequency is unacceptable since the filter was behaving really bad.
The optimizations applied to the IMU and GPS configuration were not sufficient to solve this problem because of the considerably higher order of magnitude of the delay.
The solution adopted was completely different. It was considered an MCU produced by STM32 equipped with a 32-bit architecture and a built-in CAN interface.

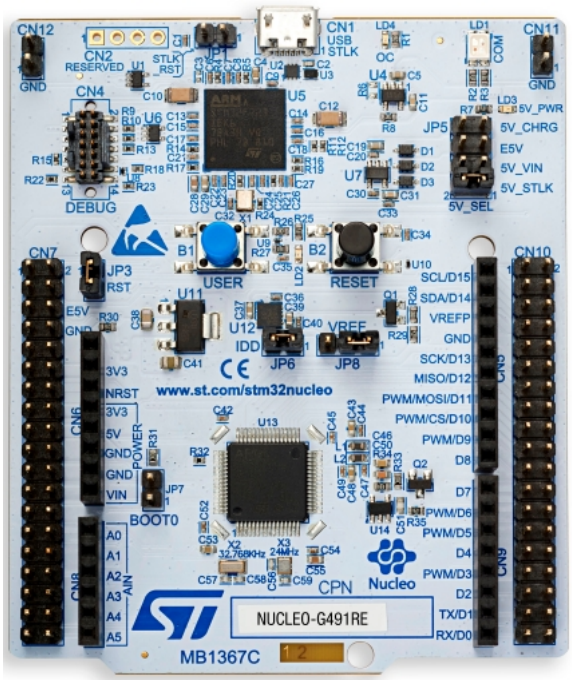


FIGURE 3.5: STM32G491RE Demoboard

The code for the EKF was generated and optimized for this new MCU depicted in Figure 3.5. In order to perform the testing activities, this demoboard was connected to the same CAN network of the IMU and the PC. The procedure is:

- The IMU sensor performs every 10 ms (100 Hz) a read of accelerations, gyroscopes, and magnetic fields and sends them to the CAN-bus;
- The STM32G4 receives these readings and calculates at 100 Hz the angles;
- The STM32G4 sends on the bus the angles which are read also from the PC.

3.7 Temperature compensation

During hardware test of the board, a severe drift was encountered in IMU readings. These tests were performed in a controlled environment: the IMU was cooled until it reached a temperature of 0°C and was then heated up to 80°C. During the whole test the IMU was never moved or stressed outside the thermal domain.

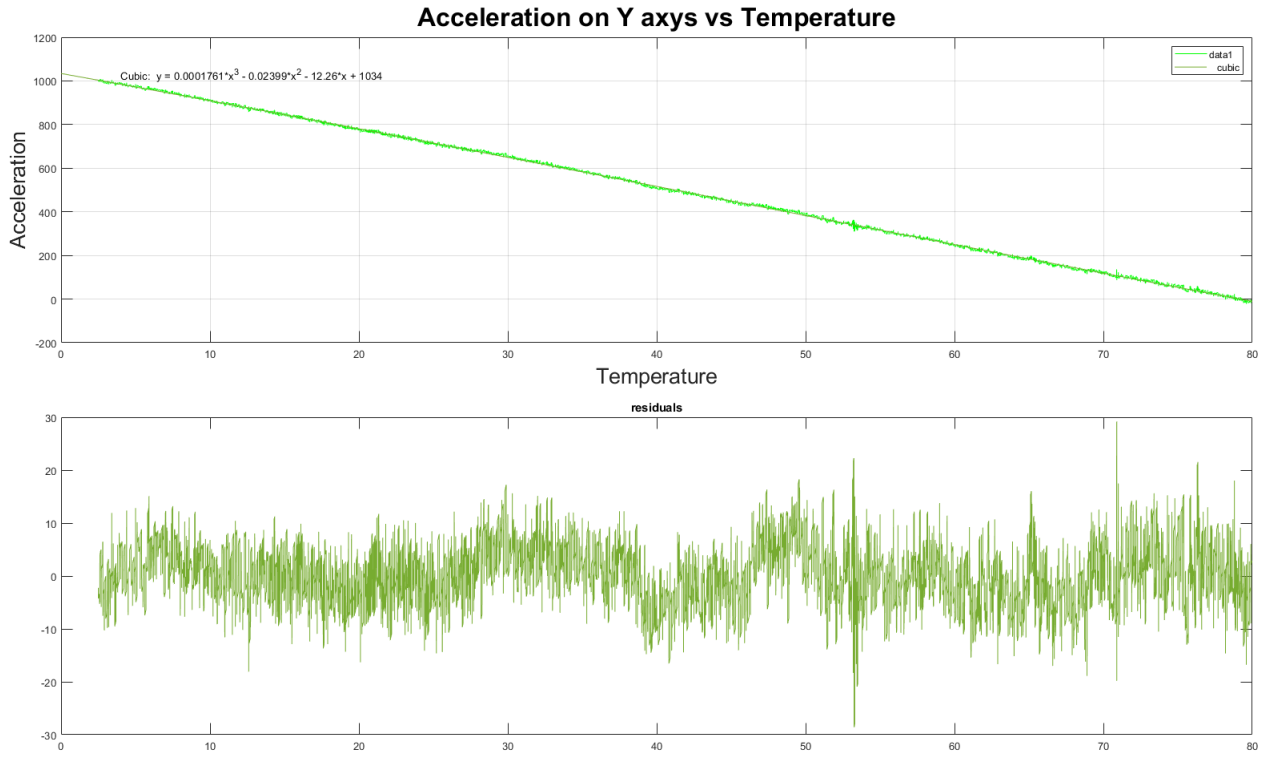


FIGURE 3.6: Drift on acceleration on Y axis

In Figure 3.6 is possible to notice the phenomena described before. While the IMU is in a rest position, it is heated from 0°C to 80°C. The condition of 0 acceleration is achieved only at 80°C and as the temperature lowers it gets more severe.
In order to overcome this problem a compensation has to be applied. The plot shown above representing Acceleration vs Temperature is calculated for each reading. For every set of point it is calculated a third order polynomial where y represents the acceleration and x the particular temperature at which the acceleration value was recorded.
Above this curve it is reported the residual remaining after calculating the curve fitting these points. This curve is then applied as:

$$CompensatedValue(T) = SensorValue - CompensationCurve(T) \quad (3.14)$$

These corrections are implemented directly in the code running on the MCU, the IMU in fact has a temperature sensor on-board that is able to inform the MCU about the temperature at each reading.

These errors depend on production defects, so each IMU needs different compensation curves. Since the company needs to produce and calibrate a considerable amount of boards a fast way is needed. The chosen approach provides a general equation that is:

$$Comp(T) = A * Temperature^3 + B * Temperature^2 + C * Temperature + D \tag{3.15}$$

The coefficients A, B, C and D are saved on the IMU during the calibration phase with an automatic CAN message that sets each of the coefficients for every axis, quantity and range of reading.

3.8 Magnetometer Calibration

Among all MEMS sensors, magnetometers are definitely the ones that need more a calibration. The readings of a magnetometer can be subject to two types of distortions:

- Soft iron;
- Hard iron.

Considering a practical approach, a perfectly compensated magnetometer provides readings concentrated on the surface of a sphere when moved in all the possible directions. This sphere in ideal conditions has to be centered in the origin of the three axis.
Soft iron distortions cause a more oval shape on this sphere while hard iron are responsible for a decentralization of the centre of the sphere from the origin.
Soft iron are tougher to correct since they are not regular on all the parts of the sphere while hard iron are easier since a simple offset solves them. For this reason it was chosen a magnetometer that is quite resilient to soft iron while it severely suffers of hard iron that need to be corrected.

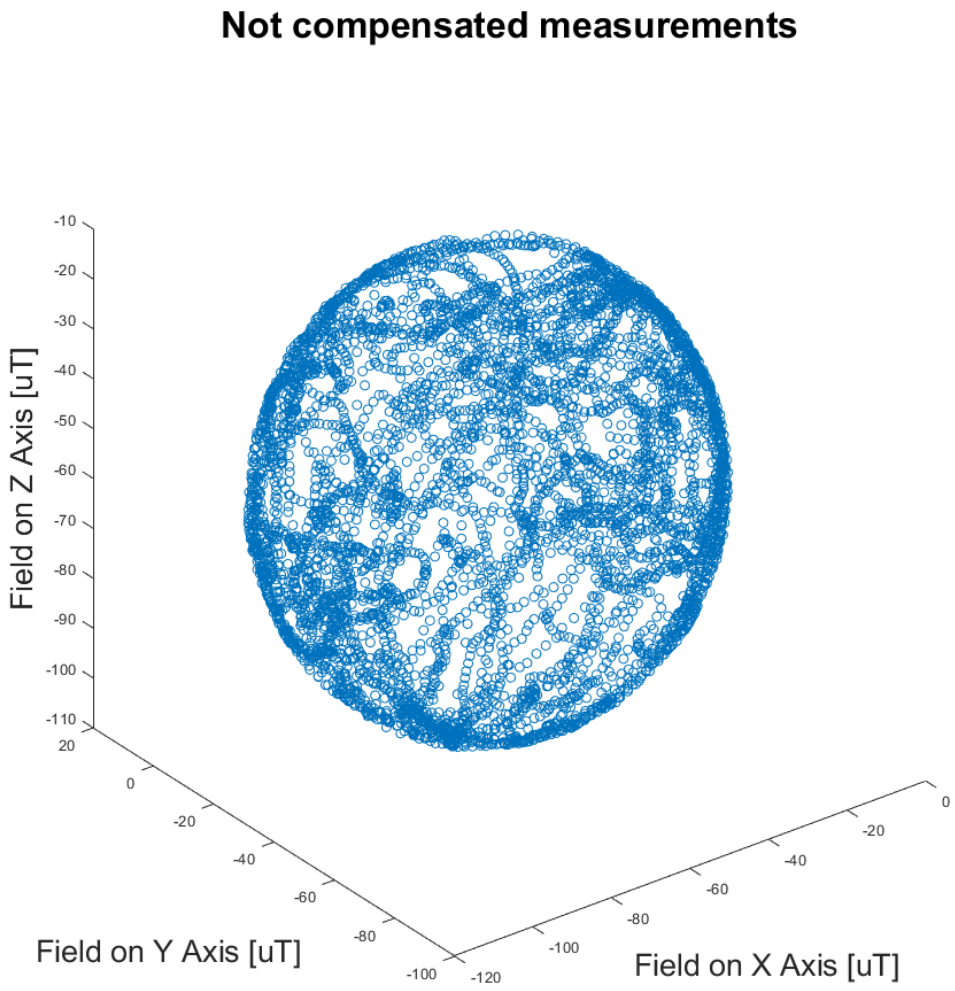


FIGURE 3.7: Magnetometer readings not compensated

In figure 3.7 are shown magnetometer readings not yet compensated. Because of the resistance to soft iron distortions, those readings already look like a sphere making a soft-iron calibration not necessary.
Those readings however are strongly decentralized as the center of the sphere appears to be almost in $[-60, -40, -60]$.
By applying the precise compensation value, it is possible to retrieve the results shown in Figure 3.8. These offsets are applied directly on the MCU in order to provide to the EKF compensated value at each time instant.

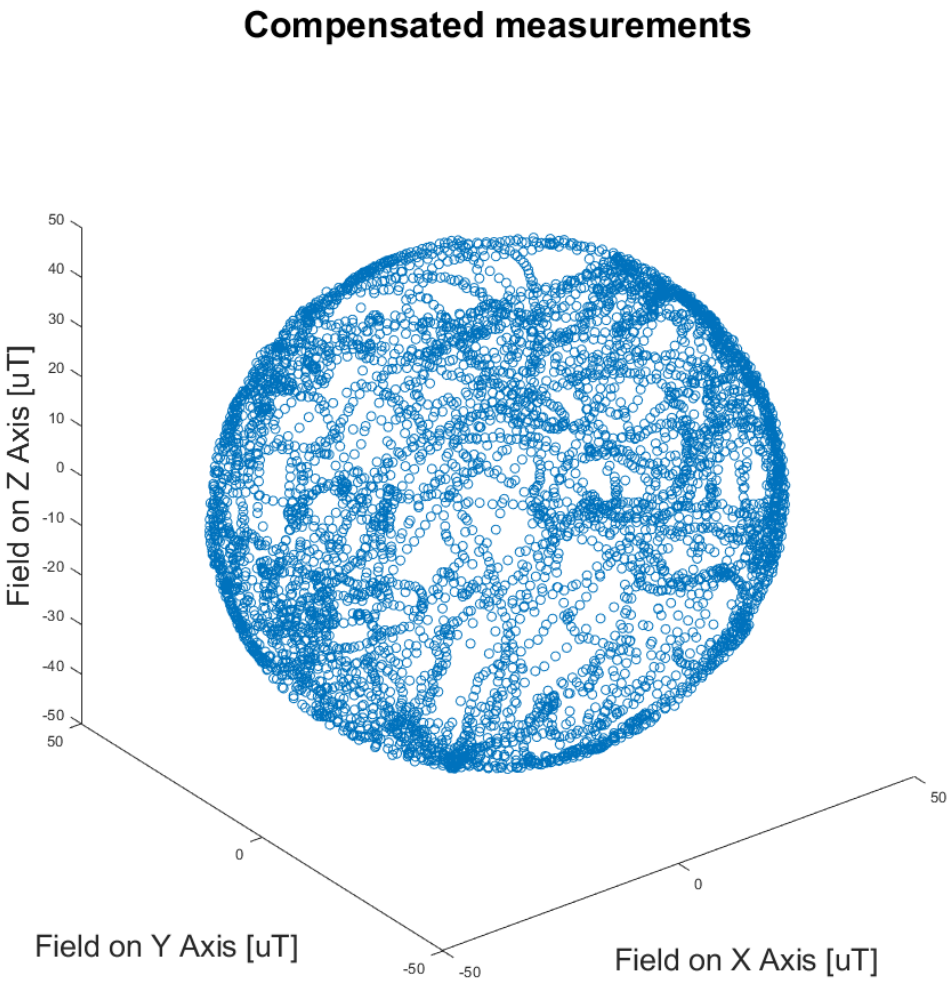


FIGURE 3.8: Magnetometer readings compensated

Chapter 4

Verification and Validation

In this section will be presented all the simulation results, the comparison between them and some considerations that will lead to a final setup choice for the proposed problem.
All the demonstrated models have been first of all simulated with similar inputs for a better comparison. These inputs have been created by exploiting the rigid body equations introduced in Chapter 2. Basically some sinusoidal accelerations and angular velocities are given as input to this model. This block is able to calculate the exact attitude that derives from this quantities in order to know the true states of the system at each time instant.
Those accelerations and angular velocities are also processed by adding noise and producing noisy sensor readings (GPS or magnetometer). All those noisy quantities are then given as input to the actual filter for the true attitude estimation and the results are finally compared to the true state calculated.
A second simulation was possible thanks to different real telemetries made available by the host company. These telemetries are a set of data comprehending among other things accelerations, gyroscopes and GPS. Since magnetometer readings unfortunately were not included, it was possible to use this set of data only for the complementary filter and the IMU and GPS. In this case, the final result was compared with the estimation performed on-board by the E-Lean IMU, an inertial sensor that is considered well performing for this purpose in motorsport applications.
A general picture of the telemetry used is shown in Figure 4.1. The software used, I2 Pro, allows to export the whole set of data in a format that is compatible with MATLAB. The first two streams of data refer to accelerations and gyroscopes while the other are related to GPS data and velocities for comparison purposes.

FIGURE 4.1: Used telemetry

4.1 Complementary filter setup

Some hypotheses were made before implementing the complementary filter. The sensor suite here considered is minimal with respect to the other setups meaning a higher reliability of the input quantities needed for the estimation. This is due to different factors: the GPS is external and has to communicate with the IMU throughout a CAN protocol that can be subject to failure or, more likely, its readings could be noisy when the motorbike is leaned because of a tighter portion of visible satellites. The magnetometer, on the other hand, could be subject to strong magnetic interference because of the presence of near metallic objects. If for any reason a sensor failure happens, meaning a misleading or totally absence read, a backup solution is needed.
Since less sensors implies less possibilities of electronic failure, the complementary filter is a proper candidate for the backup plan. Because of this, its discrete time performances were not directly compared to the EKF simulations of every setup but it was enough checking its consistency on a large time span.
As a consequence, this filter was directly simulated with data coming from the real telemetry and making a comparison between the roll angle estimated by the filter and the roll angle estimated by the E-Lean mounted on-board.

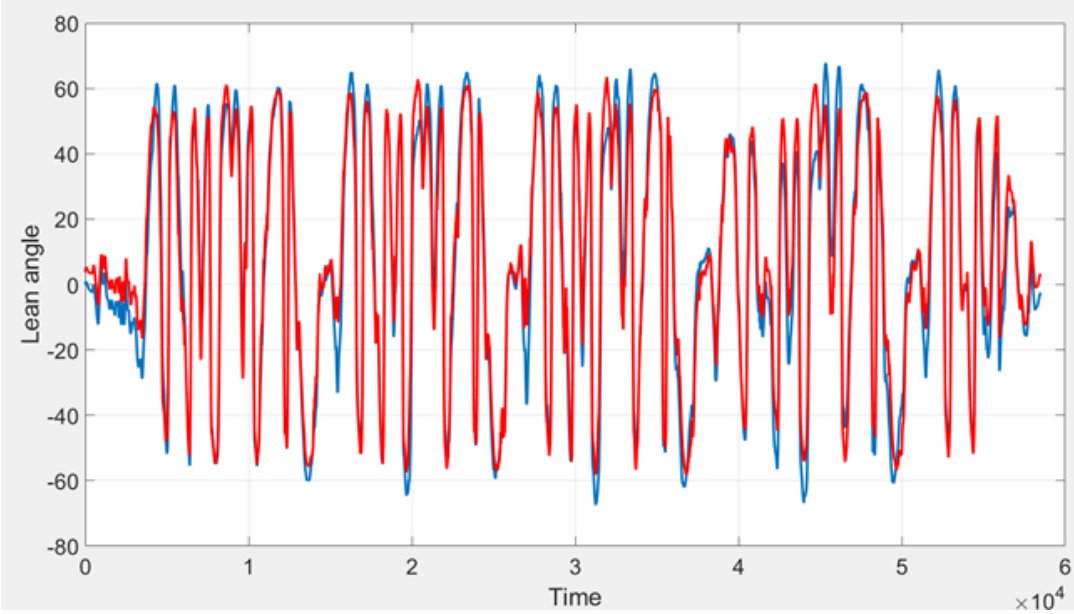


FIGURE 4.2: Complementary filter performances with real telemetry

Figure 4.2 shows the comparison between the lean angle calculated on-board by the E-Lean (blue signal) and the one calculated by the complementary filter given the same accelerations and gyroscopes (red signal). From a qualitative point of view, the red signal is following quite well the reference one. For a more complete view it is shown in Figure 4.3 the error committed calculated as:

$$\epsilon = \tilde{x} - x \tag{4.1}$$

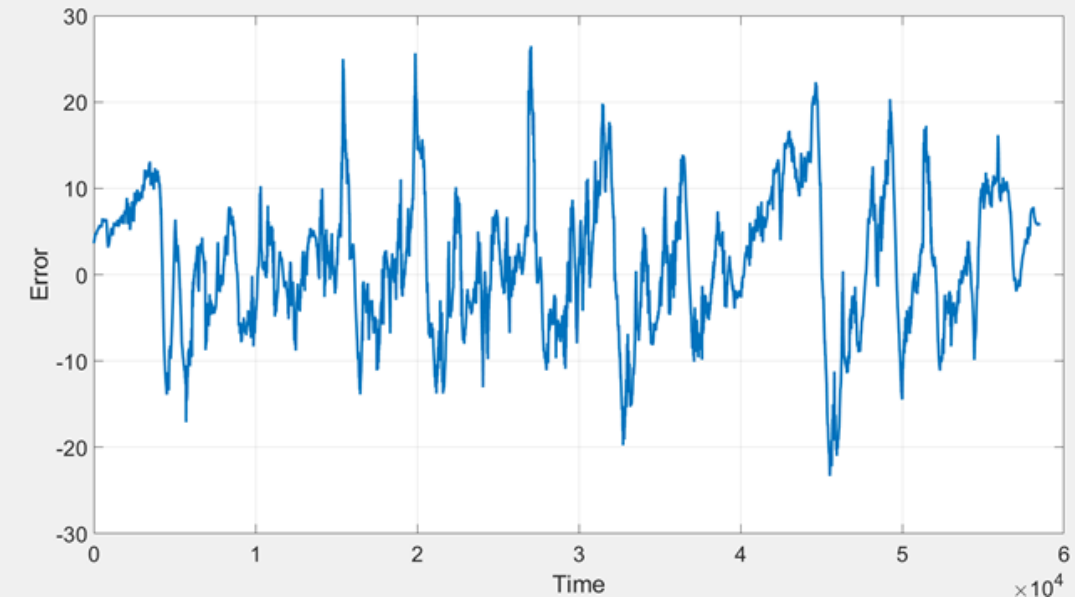


FIGURE 4.3: Complementary filter error with real telemetry

As expected, the magnitude of the error committed by the complementary filter is too high for commercial purposes but acceptable in case of GPS or magnetometer failure. For safety purposes in fact, is better reading the lean angle with an error of 10° than not knowing it at all.

4.2 IMU and GPS setup

Since before the hardware re-design the IMU was not equipped with a magnetometer, the first setup to be designed and tested was the IMU with a GPS sensor.
The design of the discrete time model was critical because of the multi-rate nature of the used sensors. After this, the steps of Software in the loop and Processor in the loop were performed for the smoothest transition possible to a real hardware application.

4.2.1 Model in the loop

As anticipated, the model has been firstly simulated with sinusoidal inputs and then with the real telemetry. The results are here presented in order.

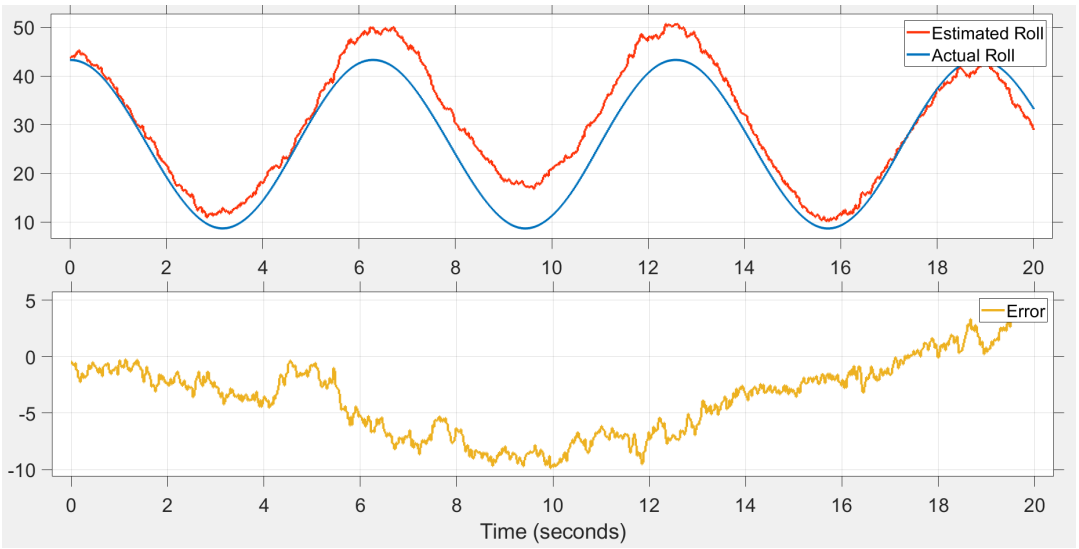


FIGURE 4.4: IMU and GPS fused in discrete time

The error committed is obviously higher then the continuous time simulation shown in Section 2.6. This is because the continuous time has to be interpreted as a discrete time model with an infinite sample time. This "slow" sample time and the multi-rate nature are the responsible of this high increase of the magnitude of the error.

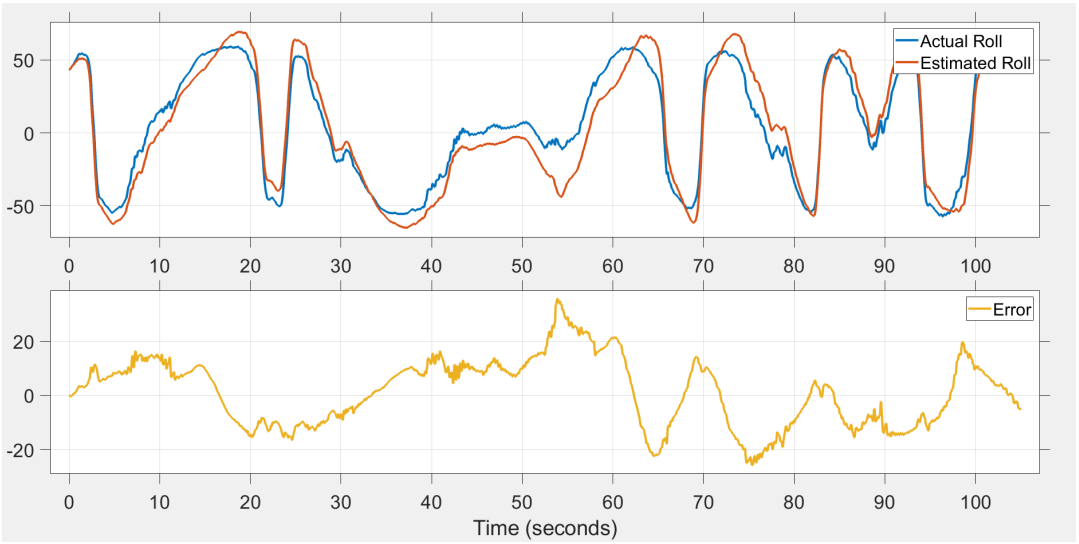


FIGURE 4.5: IMU and GPS performances with real telemetry

In Figure 4.5 is instead proposed the simulation with real data. The magnitude of the error is higher than the simulation with sinusoidal inputs and comparable with the ones of the complementary filter. The reasons behind this where

deeply investigated and the explanation found regards the GPS sensor.
When the motorbike is severely leaned, the constellation seen by the GPS decreases critically making its readings not only useless but also misleading.

4.2.2 Software in the loop

As described in Chapter 3, the code was generated through the Embedded Coder designed by Simulink. This code, however, was not implemented directly on the sensors but some steps were performed. These steps were necessary to check the good behaviour of the code in a different simulation environment.
In order to achieve this consistency, first of all a Software in the loop was performed. This test allows to check if the code is applying the same exact algorithm in C and in Simulink. It basically performs a run with the sinusoidal inputs and then performs the same run with the same inputs but this time simulating the C code on the host PC processor. After this, the identity of the results is checked by applying some safety criteria on the acceptance of the results.

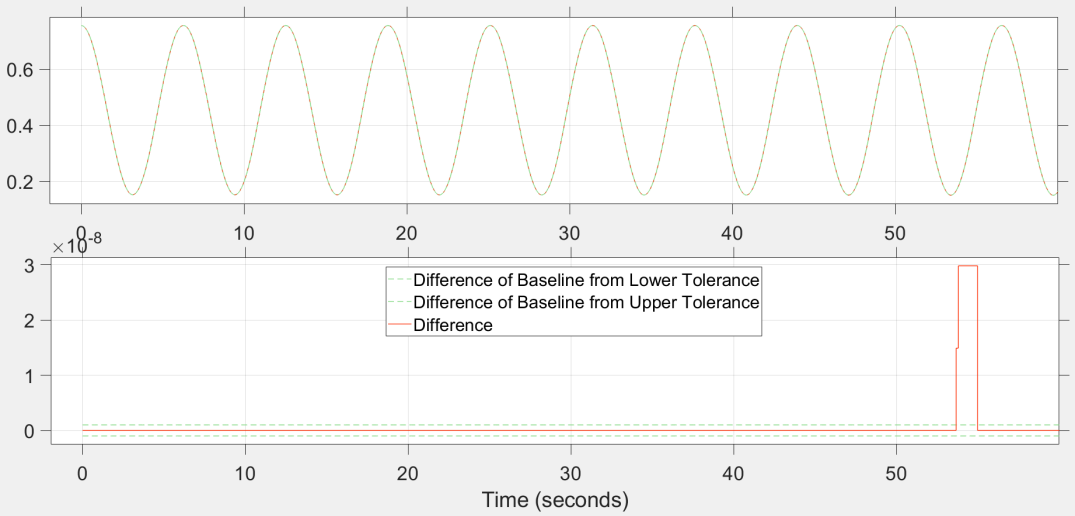


FIGURE 4.6: SIL of IMU with GPS

The test can be considered fully passed since the only deviation has a magnitude of $3 \cdot 10^{-8}$°.

4.2.3 Processor in the loop

A test similar to the SIL is necessary to move from the PC processor to the real sensor micro-controller. This test in fact checks if the micro-controller is executing the C code with acceptable execution time and results.

Unfortunately the DSPIC33 was not directly supported by the PIL tool provided by Simulink. This limitation was overcome by sending in real time the simulation accelerations, gyroscopes and GPS data from Simulink to the IMU via CAN communication. The IMU at this stage did not use readings coming from its sensor but only the received ones. The outputs were finally given by the IMU producing the comparison shown in Figure 4.7.

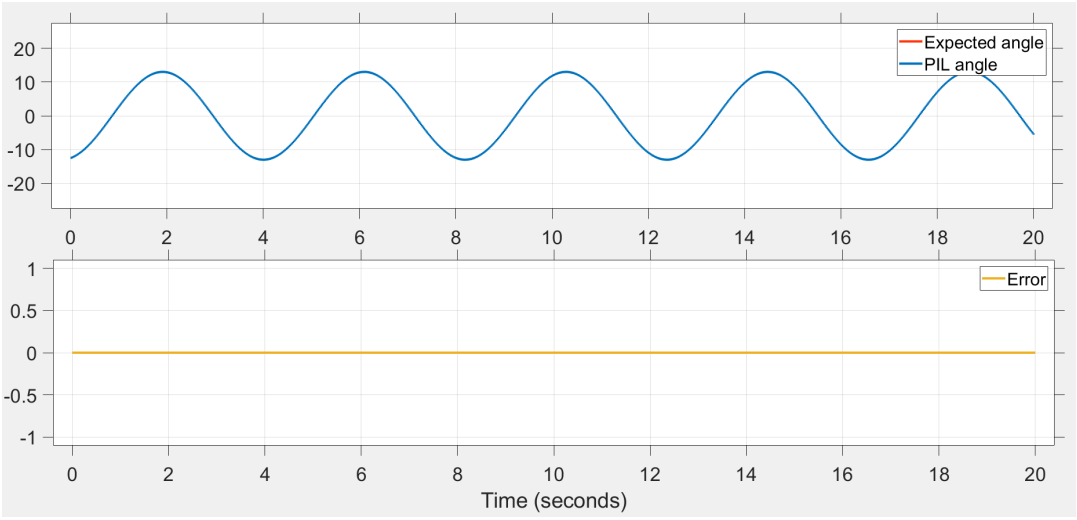


FIGURE 4.7: PIL of IMU with GPS

4.2.4 Hardware in the loop

The Hardware in the loop test was conducted in laboratory through the use of a stepper motor and a 3D printed support connecting the IMU to the shaft of the motor.
A simulation path was setup with the help of an Arduino. The idea is quite simple: knowing the number of steps, the direction of rotation and the angle covered by a single step, it is possible to retrieve at each time instant the angle of the shaft.
Connecting the IMU to the support, it will start estimating the angular position. By getting the estimation from the IMU and the actual angle from the Arduino it is then possible to compare them in order to test the overall performances.

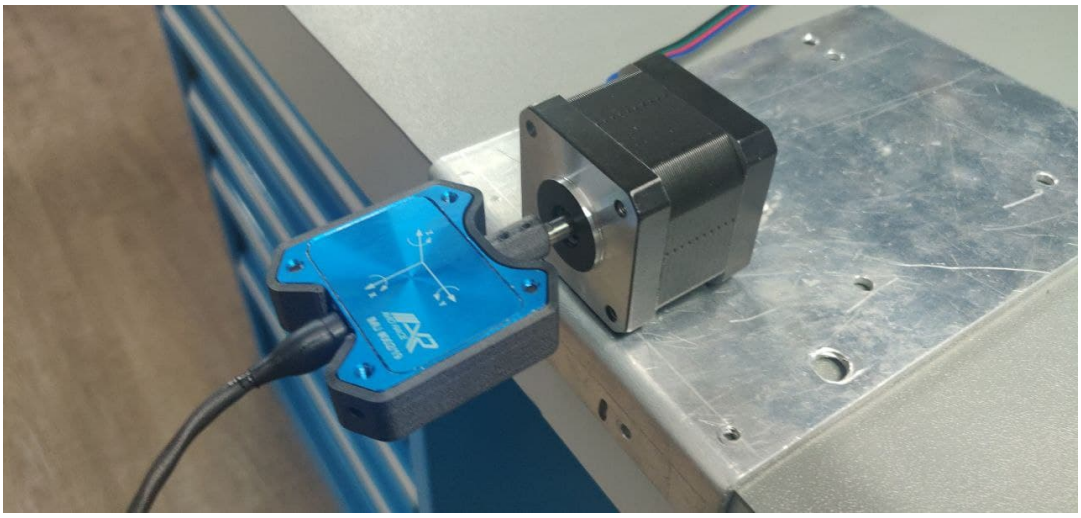


FIGURE 4.8: Testing platform for HIL

Since these tests are performed in a laboratory environment, the GPS readings could not be retrieved with a real sensor reading. The only possible way to conduct this test was by calculating the three velocity components of the IMU and adding an appropriate noise to them.
In Figure 4.9 is possible to see the results of the Hardware in the loop test.

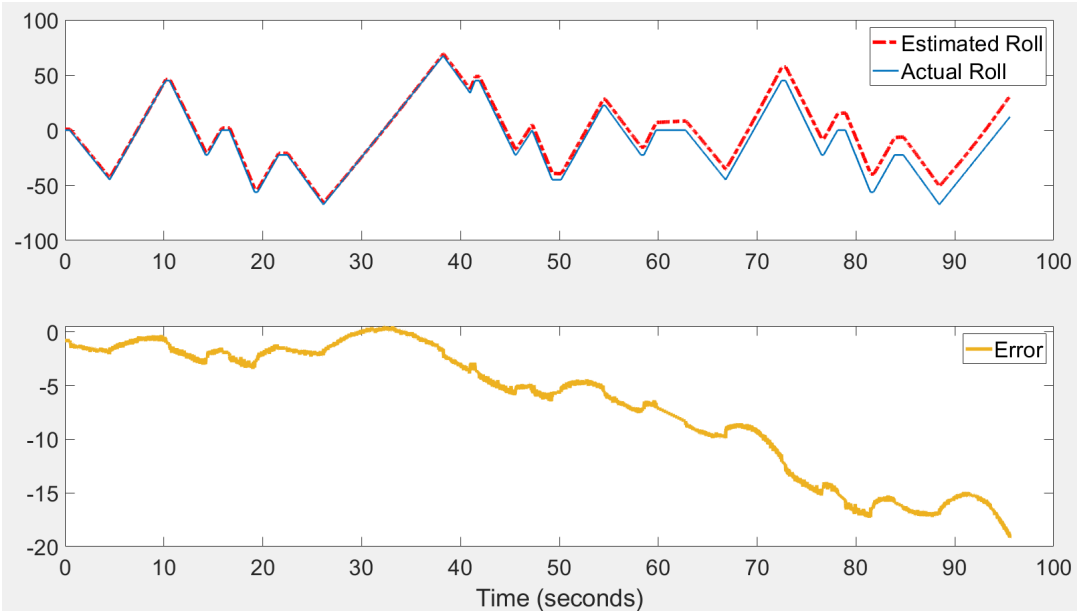


FIGURE 4.9: HIL of IMU with GPS

These results confirm an error that is not compatible with the desired application. At some time instants, as seen also before, this reaches a magnitude of 20°.

4.3 IMU and Magnetometer setup

Because of the error confirmed over the various steps of the IMU and GPS setup, it was decided to move to a corrective sensor that had to be concordant with the IMU for what concerns the mounting and the sampling frequency. This is why it was chosen to implement a magnetometer. The advantage with respect to the previous configuration is represented by the same sampling frequency of the different sensors: both the IMU and the magnetometer will run in the simulations at 100 Hz simplifying a lot the model.

4.3.1 Model in the loop

The performances of this EKF filter with sinusoidal inputs is shown in Figure 4.10.

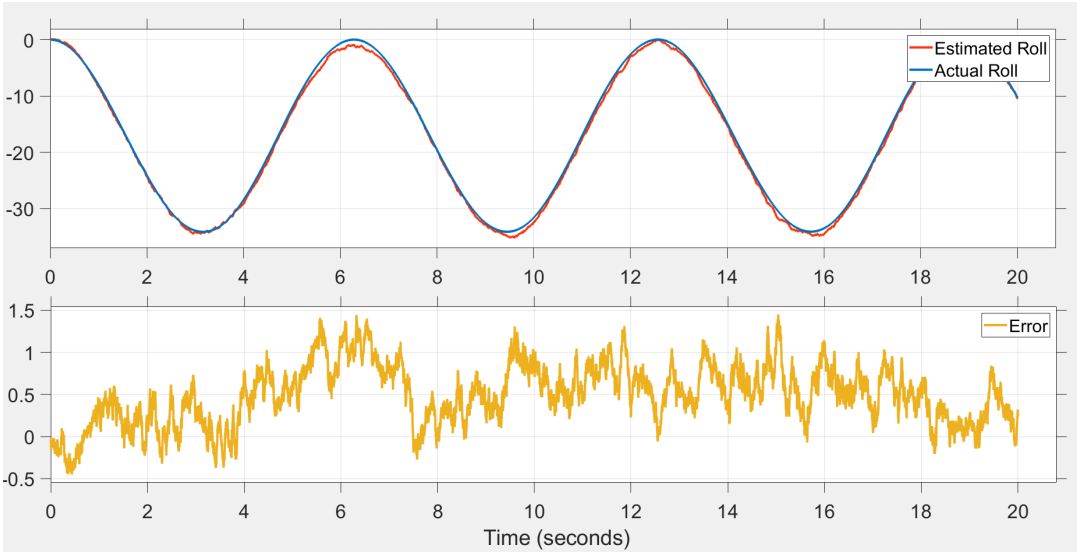


FIGURE 4.10: Discrete time MIL of IMU with Magnetometer

This discretization does not affect too much the magnitude of the error confirming that a frequency of 100 Hz is not severely worst from a continuous time ideal situation and that the same sampling frequency makes the EKF corrections work at their best.

4.3.2 Software in the loop

The same steps for moving from the simulation to the real implementation were exploited also for the magnetometer application. The first of them was the SIL in order to check the consistency of the generated code:

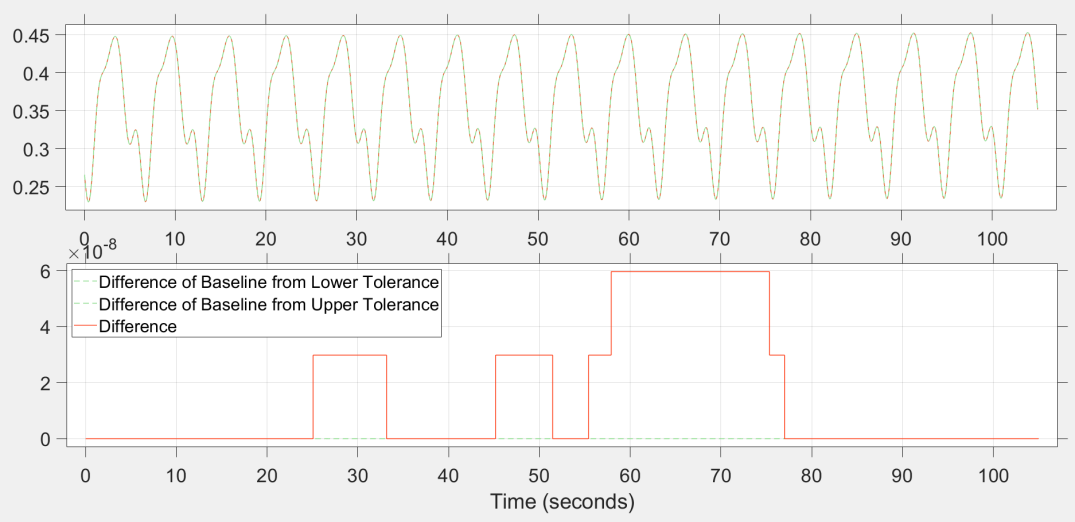


FIGURE 4.11: SIL of IMU with magnetometer

These results are far acceptable considering a maximum difference of $6 \cdot 10^{-8}$°.

4.3.3 Processor in the loop

For the reasons explained in Section 3.6, the MCU used for the implementation of the magnetometer model is different. The approach for the Processor in the loop test however is quite similar.
The STM32G4 demoboard receives via CAN the accelerations, gyroscopes and magnetometers input from the Simulink environment. It sends then back via the same communication interface the states of the system at each time instant. These states are compared with the ones calculated directly in the Simulink model and the results are shown in Figure 4.12.

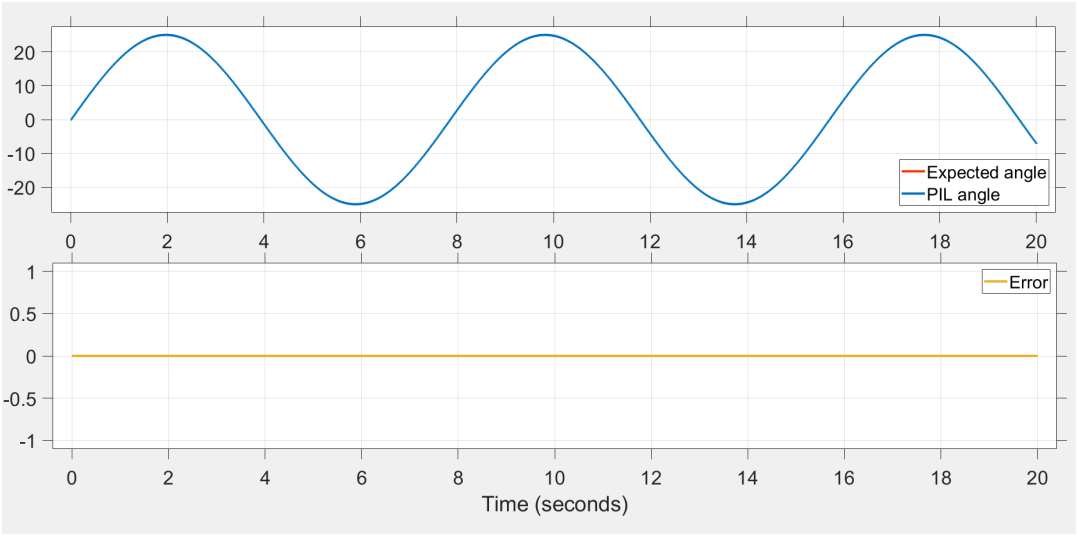


FIGURE 4.12: PIL of IMU with magnetometer

These results again are plenty acceptable since the two simulations are overlapped and identical.

4.3.4 Hardware in the loop

For the hardware in the loop test, it was exploited the same simulating platform used for the same test performed with the GPS setup. For the reasons explained in Section 4.4, the simulation path is identical to the previous one and produces the following results:

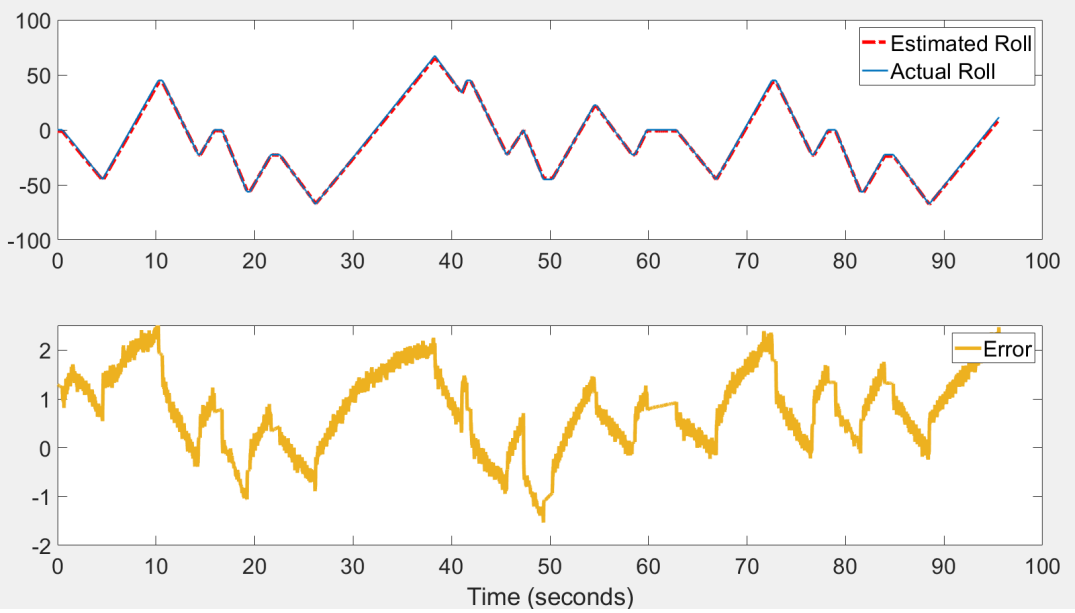


FIGURE 4.13: HIL of IMU with magnetometer

The estimation is far better if compared with the GPS test. As expected from the previous stages, the error remains bounded in just few degrees and is plenty compatible with the desired precision for a motorsport application.

4.4 Results comparison

As anticipated in the previous section, some of the tests have been performed with an identical simulation path in order to compare the performances by layering the different tests.
The first comparison is the one performed over the continuous time models with sinusoidal inputs, this is shown in Figure 4.14.

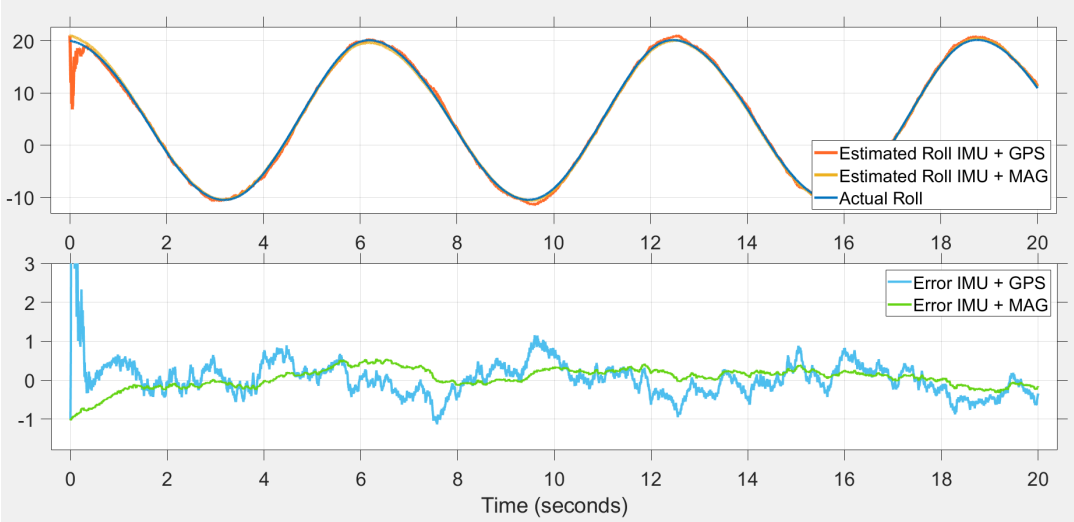


FIGURE 4.14: Comparison of GPS and magnetometer in continuous time

In this simulation can already be seen two interesting points:

- The magnetometer setup is more resilient to stretched initial conditions as it converges quicker to the real state;
- The GPS setup has an higher error magnitude.

The comparison then was focused on the discrete time simulations:

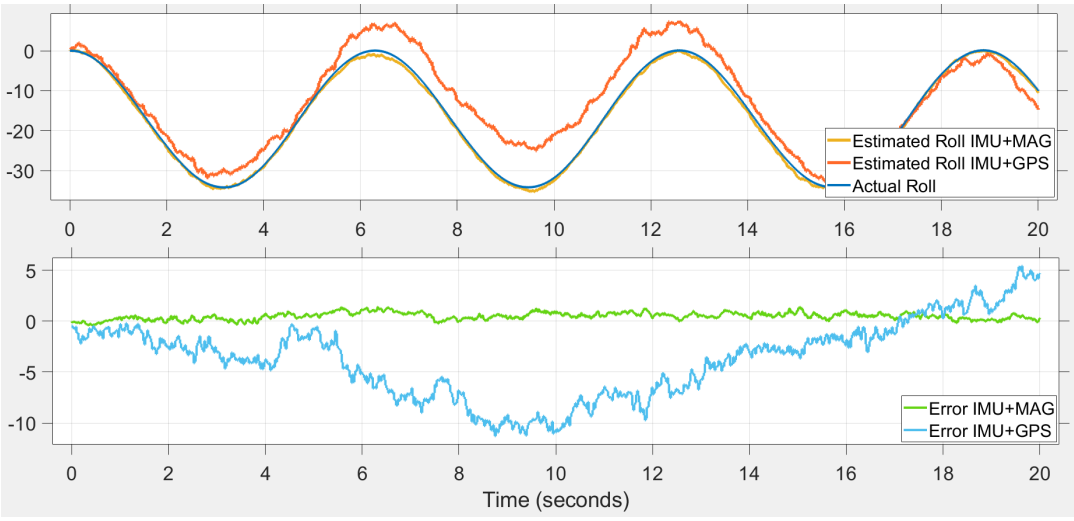


FIGURE 4.15: Comparison of GPS and magnetometer in discrete time

Figure 4.15 confirms the severe suffering of the GPS model when switching to a multi-rate setup. The magnetometer setup instead is correctly keeping its error

constant and bounded in an acceptable interval.

The last results compared relate on the HIL simulation performed with the stepper simulation platform. The comparison is shown in Figure 4.16.

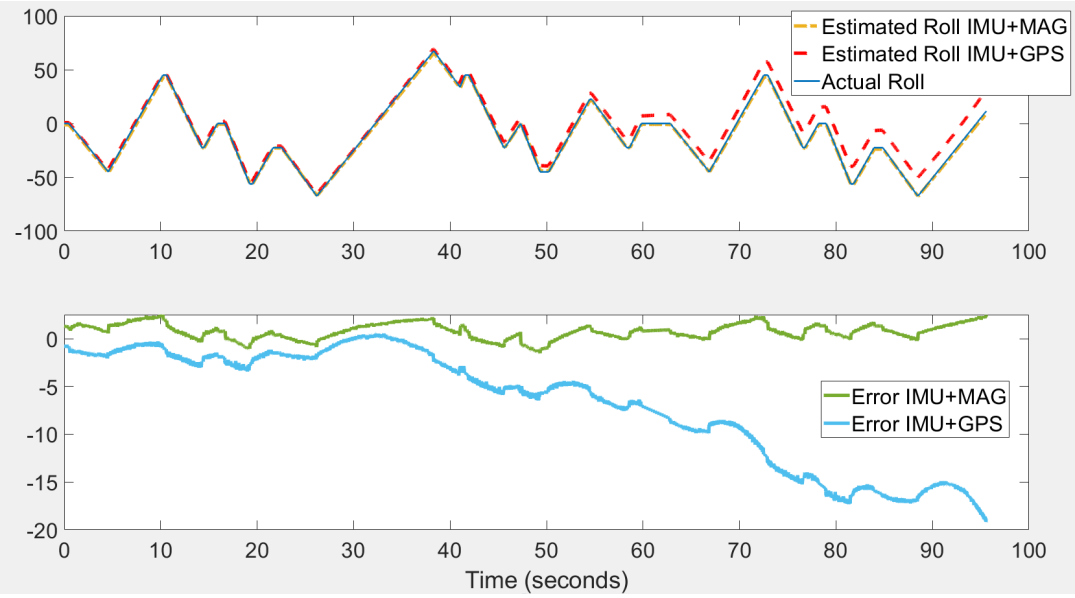


FIGURE 4.16: Comparison of GPS and magnetometer in hardware in the loop test

Once again it is confirmed the better performance of the magnetometer model that remains close to the real state with a magnitude of error lower than the GPS model.

Chapter 5

Conclusions and future developments

5.1 Conclusions

The whole work not only has the intent of identifying the best algorithm for the proposed problem but it also can be considered as a guide for implementing a model based design on a real hardware application.
All the chapters have in fact a double purpose: while they list the peculiarities of each setup, all the steps that make a real possible implementation are sorted.
The conclusions of the work have to take into consideration a variety of factors, some of which are not directly involved with the activity but affect the final balance.
The very first outcome is that the sensor board with the current configuration is not sufficient for gaining the desired performances. Neither the Complementary filter or the Extended Kalman Filter with GPS can guarantee an error sufficiently constant over time and bounded in less than $\pm 2°$. The hardware design of this board was completed before the activity began, meaning that an implementation of an additional sensor, like a magnetometer, may be quite resource consuming for the company.
To summarize, the simulation and comparison at each layer shows how the implementation of a magnetometer improves the estimation. A trade-off, however, has to be found between performances and resources for an hardware review.
The second considerable conclusion regards the model-based design combined with Simulink for its developing. By organising the full schedule of this approach, it is possible with a minimum initial investment in terms of time to reduce significantly the period dedicated to design and development and to minimize issues and errors. Simulink on its side encourages quite well this type of approach by supporting a great number of MCUs where to perform the tests and offers alternative paths when implementing different microcontrollers. Thanks to this and to all the supportive tools offered for optimizing the entire flow of work, it is possible to implement a complex model on real hardware with a lower effort.

5.2 Future developments

Future developments should be concentrated on two fields:

- Improvement of the model;
- Hardware enhancement.

For what concerns the model, the filter can be evolved by extending the equations that describe its dynamics or by adding more information to the filter. These data may come from other sensors already mounted on-board on the motorcycle and connected to the same bus of the IMU. Some competing companies for example, have products that require the speed of the motorbike as input. In this way, a crucial improvement would be implementing a model that links the wheel speed measured by an Hall effect sensor to the real speed of the body. Once this is done, also the three states related to the vehicle velocity can be observed with a direct measure.
The hardware enhancement instead has a more clear path to follow. First of all, in order to implement the magnetometer in an all-in-one solution, an hardware review is needed.
This hardware review allows moreover to introduce several changes:

- For the reasons explained in Section 3.6, the MCU implemented would be changed implementing an STM32G4 microcontroller that can sustain the high load required by the filter;
- The temperature compensations explained in Section 3.7 are quite time-consuming to implement. An inertial sensor with a lower temperature-dependency would be ideal. If the temperature drift in fact is not huge, these compensations should not be needed anymore.

Appendix A

Inertial Measurement Unit Datasheet

AVIORACE SRL
via dell'Artigianato, 348 - 41058 Vignola (Mo) Italy
Tel. 0597700253 - Fax 0597520181
www.aviorace.it

IMU – Inertial Measurement Unit

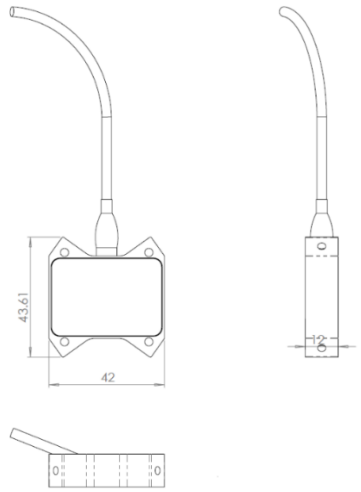
Inertial measurement unit able to output acceleration and angular acceleration on all 3 axis.
Output is over can bus and it is possible to select multiple configurations for can bus parameters and measurement ranges with specific can messages/tool software working with PCAN-USB.

Technical Specification:

Power Supply:	8÷16 Vdc
Current Absorption:	90mA @ 12V
Working Temperature Range:	0÷80 °C
Thermal Effects:	±0.25% FS
Material:	Anodized aluminum
Protection:	IP64
Weight with wires:	≈ 58 g
Wires:	Raychem 55A 26 AWG, length 1000mm
Connection:	RED Power Supply BLACK Gnd WHITE Can-H BLUE Can-L Connector on request

Setting Configuration (default in bold):

- CAN ID: specification 2.0A
- BAUDRATE: **1 Mbps**, 500 Kbits, 250 Kbits
- ACCELEROMETER RANGE: ±16G, **±8G**, ±4G, ±2G
- GYROSCOPE RANGE: ±2000 °/s, ±1000 °/s, ±500 °/s, **±250 °/s**, ±125 °/s
- GYROSCOPE CUTT-OFF FREQ 3dB: 10.7 ; **20.8** ; 39.9 Hz
- ACCELEROMETER CUT-OFF FREQ 3dB: 10.12 ; **20.25** ; 40.50 Hz
- ZERO OFFSET: through can message or input signal to ground



!!! Reserved CAN address used for internal and can settings: 0x0E5 !!!

Default Can Output:

CAN ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x100	Acceleration X		Acceleration Y		Acceleration Z		--	
0x101	Gyroscope X		Gyroscope Y		Gyroscope Z		Internal Temp	

Acceleration Resolution [0.001 G]

Gyroscope Resolution [0.1 °/s]

Internal Temperature Resolution [0.1 °C]

All quotes in mm

User Manual and DBC file on request

Bibliography

[1] L. Romualdi, N. Mancinelli, A. De Felice, and S. Sorrentino, "A new application of the extended kalman filter to the estimation of roll angles of a motorcycle with inertial measurement unit," *FME Transactions*, vol. 48, pp. 255–265, Jan. 2020. DOI: 10.5937/fme2002255R.

[2] I. Boniolo, "Attitude estimation of a motorcycle in a kalman filtering framework," vol. 43, Jul. 2010, pp. 779–784, ISBN: 9783902661722. DOI: 10.3182/20100712-3-DE-2013.00065.

[3] M. Kordestani, M. Dehghani, B. Moshiri, and M. Saif, "A new fusion estimation method for multi-rate multi-sensor systems with missing measurements," *IEEE Access*, vol. 8, pp. 47 522–47 532, 2020. DOI: 10.1109/ACCESS.2020.2979222.

[4] X. Jing, J. Cui, H. He, B. Zhang, D. Ding, and Y. Yang, "Attitude estimation for uav using extended kalman filter," in *2017 29th Chinese Control And Decision Conference (CCDC)*, 2017, pp. 3307–3312. DOI: 10.1109/CCDC.2017.7979077.

[5] P. Narkhede, S. Poddar, R. Walambe, G. Ghinea, and K. Kotecha, "Cascaded complementary filter architecture for sensor fusion in attitude estimation," *Sensors*, vol. 21, no. 6, 2021, ISSN: 1424-8220. DOI: 10.3390/s21061937. [Online]. Available: https://www.mdpi.com/1424-8220/21/6/1937.

[6] M. Euston, P. Coote, R. Mahony, J. Kim, and T. Hamel, "A complementary filter for attitude estimation of a fixed-wing uav," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 340–345. DOI: 10.1109/IROS.2008.4650766.

[7] C. Hajiyev and E. S. Conguroglu, "Integration of algebraic method and ekf for attitude determination of small information satellites," in *2015 7th International Conference on Recent Advances in Space Technologies (RAST)*, 2015, pp. 719–724. DOI: 10.1109/RAST.2015.7208435.