Warsaw University of Technology

FACULTY OF
POWER AND AERONAUTICAL ENGINEERING

Institute of Aeronautics and Applied Mechanics

Master's diploma thesis

in the field of study Automatic Control and Robotics
and specialisation Robotics

Adaptation of inertial navigation systems in robot's positioning systems

Wojciech Gajda
student record book number 304494

thesis supervisor
Marek Wojtyra, D.Sc., Ph.D., M.Eng.

Warsaw, 2024

Abstract

Adaptation of inertial navigation systems in robot's positioning systems

To fill up, after polish version approvement

Keywords: keyword1, keyword2, ...

Streszczenie

Zastosowanie nawigacji intercjalnej w systemach pozycjonowania robotów

W pracy opisano realizację systemu przeznaczonego do estymacji pozycji i orientacji robota, bazując w głownej mierze na pomiarach czujników intercjalych. System pozwala na prowadzenie estymacji w czasie rzeczywistym w oparciu o zróżnicowane źródła danych. Ponadto w systemie uwzględniona została informacja o konstrukcji mechanicznej i jej ograniczeniach. Wynikiem działania systemu jest wizualizacja w czasie rzeczywistym oraz zbiór zarejestrowanych danych, pozwalajacych na późniejszą analizę.

W ramach pracy zgromadzono i uporządkowano wiedzę na temat czujników pomiarowych, metod filtracji ich pomiarów i metod fuzji pomiarów. Bazując na przygotowanych informacjach opracowany został model Filtru Kalmana, pozwalającego na efektywną estymacje pozycji i orientacji robota. Rozważania teoretyczne znalazły odzwiercielenie w rzeczywistej implementacji, której działanie przeanalizowano na wybranym przypadku badawczym.

Słowa kluczowe: estymacja, czujniki, filtracja, fuzja czujników, więzy, Rozszerzony Filtr Kalmana

Contents

1. Introduction

1.1. Motivation

The motivation behind this thesis stems from a research project dedicated to creating hybrid methods for analyzing overconstrained multi-body systems. Research conducted in this project is planned to be validated in an experimental study. The test bench prepared for this purpose is being successively upgraded to meet various requirements. Specifically, the experimental setup needs to be equipped with position and orientation measuring sensors. Fulfilling this demand poses a significant challenge, particularly when attempting to measure forces in dynamic scenarios. The need to correlate position and orientation data with force measurements led to considerations of how measurements can be made in case, the dedicated sensor were not included in system.

The selected variant involves the usage of an inertial sensor and its fusion with sensors commonly used in robotics. The interdisciplinary nature of the project, combining elements from aviation concepts and robotic systems, fueled my enthusiasm to explore and innovate in this unique intersection.

1.2. The aim of thesis

The purpose of this project is to design a universal positioning system based on inertial navigation and knowledge of the multi-body design. The system is an adaptation of the Extended Kalman Filter to be used in robotic. To achieve that many sensors are considered and involved in calculation. The measurements will be filtered and blended to improve estimation results. The algorithm will be optimized to run online in real time. The architecture of system is planned to be very transparent and open to modification in order to increase reusability. It is worth to highlight that the usage scope of system is not limited to presented case. The presented usage has a teaching value due to its limited description. However, there are many applications of complete logging system especially in industry and mobile robots.

The thesis starts with a review of the state-of-the-art methods of estimation position and orientation both in aviation and robotic studies. There are many solutions that have already been checked and can be partially adopted in the project. Based on the gathered information, a novel fusion method will be proposed and checked in the simulation and in real deployment.

The practical outcome of thesis is a design of prototype able to estimate position and orientation with various sensors sets connected. The thesis will include estimation results and error calculation. Tuning and improving scores is also part of the project. Finally, the project ends with experiments, that are designed to confirm the accuracy and precision of the developed system, and to determine whether it is suitable for use in further scientific work.

2. Review

The position and orientation estimation is a problem known in aviation over the years. With the advent of automated control systems, the need to determine exact location and orientation has steadily increased. As a result, mechanical instruments were quickly suppressed by electronic sensors. Based on their measurements, orientation calculation algorithms were developed and implemented. The well-known and used are the Complementary filter [1], the Direct-Cosine-Matrix algorithm [2] and the Madgwick Orientation Filter [3] [4]. All of these methods provide tolerable results depending on the set of sensors used, but without a clear mechanism to control particular sensor's involvement. Also, the position estimation was address by improving path integration methods and inertial sensor measurements integration [5]. Next, the position was corrected by introducing radio beacon and satellite systems. For a long time, position and orientation were estimated separately.

Huge impact on navigation system were caused by adopting the Kalman Filter in estimation. The mathematical concept of a filter based on the equation of state and statistics was first presented in 1960 [6]. Since then, the Kalman Filter, especially in its non-linear form (known as the Extended Kalman Filter) became a standard in state estimation and sensor fusion. Its usability was tested in many different scenarios, leading to an extensive database of articles [7] [8] [9]. The aviation is not the only application area of the Kalman Filter. Thanks to its versatility, the filter can be used wherever it is possible to arrange the appropriate equations of state.

The result of widespread familiarity with the filter is also the development of many modifications and improvements that allow it to be used in specific applications. One of the most useful in presented thesis is possibility to develop the system with an additional correction to meet the constraints of the system. The concept and implementation of correction in article [10]. Correction leads to exact solution in case of linear equality and inequalities, and gives an estimation in the case of non-linear constraints.

The navigation methods outlined also appear in robotics, but the dominant part is mobile robotics [11]. In stationary mechanisms, higher accuracy and repeatability are required. For this reason, a common approach is to use encoder readout and position estimation through a forward kinematics

task [12]. The significant advantage of this solution is that received results comply with mechanism constraints. In the absence of an encoder or similar built-in sensor (e.g. in biorobotics), computer vision algorithms [13] [14] and triangulation methods [15] are used as a substitute. Inertial sensors are also mounted in industrial robots, but the measurements are usually used in quality and health checks like vibration measures [16].

Every algorithm, no matter how sophisticated, bases its results on the data provided. Poor-quality data leads to errors and high uncertainty of results. Garbage in, garbage out. To prevent this, additional steps should be taken at the system preparation stage. By nature, sensors have finite resolution and sampling time. Each sensor measurement is subject to errors, but many factors contribute to this, like bias or high frequency noise. Same of these factor are inevitable, while others can be strongly reduced, like mounting bias. To improve results, all sensors should be checked and calibrated before usage [17] [18] [19]. It is also a good practice to pre-filter raw measurements with frequency filtration [20] to minimize noise and characteristic disturbances.

The speed of calculation is an important factor as well. Sensors have various measurement periods and modes. Some of them, especially inertial sensors, are incredibly fast, leaving only a couple of milliseconds for calculation. To achieve good-quality results, as stated in Nyquist–Shannon sampling theorem [21], the frequency of estimation should be at least twice as high as the maximal frequency observed in the mechanism's movement. Achieving high performance on an embedded system requires suitable implementation solutions. Since most of the calculations are conducted on floating precision numbers, hardware acceleration is required [22] [23].

At the time of writing this review, no publication could be found on the application of inertial navigation systems in industrial robotics. This allows us to conclude that the concept defined in the project objective has not yet been studied and is an open research problem.

3. Sensors

Sensors are the fundamental components of any measurement system, enabling it to perceive the world. There is a vast array of sensor types, each designed to measure specific physical quantities. Even within a one type of measured value, sensors can vary significantly in terms of precision and cost.

Sensors can generally be classified into two main categories: those that operate independently and those that depend on external infrastructure. Independent sensors operate based on the principles of physics. For example, at its simplest, a temperature sensor exploits the fact that electrical resistance changes with temperature variations. On the other hand, dependent sensors require external resources to function. An example of such a sensor is a GNSS receiver, which estimates its position only when a sufficient number of satellites are visible.

Accurately estimating position and orientation typically involves a combination of sensors that measure not only position, velocity, and acceleration but also have an indirect relationship with the state being estimated. The most precise results are achievable when every aspect of the state is directly measured, which is often impractical. The absence of distance sensors and encoders means that state estimation must rely on alternative methods. This thesis will explore dead reckoning, a method that estimates position and orientation by integrating measurements from inertial sensors. At a minimum, this method requires two types of inertial sensors: an accelerometer to measure acceleration and a gyroscope to measure angular velocity. However, results from this minimal setup tend to be suboptimal. In real-world applications, systems are enhanced through redundancy and the incorporation of partial information sources.

3.1. Sensor's types

Sensors that measuring a specific type of value, even within the same family, can differ significantly in the methodologies they employ to obtain that a value. The development of new sensor types is a constant effort, primarily aimed at enhancing accuracy. However, this pursuit of higher accuracy often

results in increased sensor costs. It is worthwhile to briefly outline how these methods have evolved over time.

For inertial sensors, this evolution can be traced from mechanical sensors to Micro-Electro-Mechanical Systems (MEMS) sensors, and onto today's laboratory-grade sensors. Each step on this path not only represents a leap in the technology used but also reflects a balance between achieving greater precision and managing production and operational costs. Mechanical sensors, which were among the first to be developed, rely on mechanical components and physical principles for their operation. Many moving parts often cause that mechanical sensor were unreliable. MEMS sensors, a significant advancement, integrate mechanical components and electronics into a single system, offering improved accuracy, smaller size, and lower power consumption. Today's laboratory-grade sensors, which represent the peak of current technology, provide unparalleled accuracy but at a significantly higher cost. This progression underscores the relentless pursuit of precision in the field of sensor technology, driven by the demands of increasingly sophisticated applications.

However, when considering the balance between quality and cost, MEMS sensors are favored in most applications. They guarantee minimal enclosure size and offer quality that is commensurate with their dimensions. Furthermore, the compactness and cost-effectiveness of MEMS sensors make it feasible to deploy arrays of multiple sensors. Integrating their measurements can significantly enhance accuracy. This approach of blending data from multiple MEMS sensors not only improves the precision of measurements but also leverages the strengths of each sensor to compensate for individual limitations, making it a highly efficient strategy in various applications.

3.2. Sensor's model

The description of sensors so far has mainly concerned their features and types. However, to use them in an estimation, this description should be enriched with a formalized mathematical model. The sensor's model defines the relation between physical quantities and sensor output, but also includes an error. A well-prepared model allows the development of better filtration methods and ultimately leads to better results.

3.2.1. Accelerometer

An accelerometer measures all accelerations that affect the sensor. It means that if the sensor was mount in a non-inertial reference frame, beside the linear acceleration of frame, it will also measure

centripetal, angular and Coriolis acceleration. What's more, masses on Earth experience constant gravitational acceleration, and that feature is especially useful in orientation's estimation. An accelerometer measurement is a resultant of those accelerations. Equation (3.1) represents measured acceleration as a sum of components. Because of its complexity it is often assume that accelerometer is located in frame center and $\boldsymbol{r^W} = \mathbf{0}$. It leads to simplification given in equation (3.2).

$$\boldsymbol{a}_{res}^{B} = \boldsymbol{a}^{B} + \boldsymbol{R}_{B}^{W}\left(\boldsymbol{\omega}^{W} \times \left(\boldsymbol{\omega}^{W} \times \boldsymbol{r}^{W}\right) + \boldsymbol{\epsilon}^{W} \times \boldsymbol{r}^{W} - 2\left(\boldsymbol{\omega}^{W} \times \boldsymbol{v}^{W}\right) - \boldsymbol{g}\right) \tag{3.1}$$

$$\boldsymbol{a}_{res}^{B} = \boldsymbol{a}^{B} - \boldsymbol{R}_{B}^{W}\left(\boldsymbol{g}\right) \tag{3.2}$$

where:

- $\boldsymbol{a}_{res}^{B}$ – resultant acceleration given in moving frame (linked with body),
- $\boldsymbol{a}^{B}$ – acceleration of moving frame given in this frame,
- $\boldsymbol{R}_{B}^{W}$ – rotation matrix form world frame to moving frame,
- $\boldsymbol{\omega}^{W}$ – moving frame's angular velocity given in world frame,
- $\boldsymbol{\epsilon}^{W}$ – moving frame's angular acceleration given in world frame,
- $\boldsymbol{r}^{W}$ – sensor position in moving frame given in world frame,
- $\boldsymbol{v}^{W}$ – moving frame's linear velocity given in world frame,
- $\boldsymbol{g}$ – gravitation acceleration in world frame $\left(\boldsymbol{g} = \begin{bmatrix} 0 & 0 & g \end{bmatrix}^{T}\right)$.

The second part of model is an inclusion of error. Accelerometers' readings are subject to scale error, bias, assembly's inaccuracies and noise. The scale error means that the readings are linearly dependent of acceleration, but the slope factor is not equal 1 precisely. The bias means that the linear function describing transform is not homogeneous and constant coefficient is non-zero. The assembly inaccuracy is due to the fact that the sensor axes are not collinear with the reference frame. However, the assembly inaccuracy is described as rotation that does not affect the measure's norm. Finally, the noise is a random component added to measure. Fortunately, in case of accelerometer, the noise is negligibly small. Equation (3.3) describe the correlation between acceleration and output from sensor.

$$\hat{\boldsymbol{a}}_{res}^{B} = \boldsymbol{R}_{M}\boldsymbol{S}\left(\boldsymbol{a}_{res}^{B} + \boldsymbol{b}_{0}\right) + \boldsymbol{n} \tag{3.3}$$

where:

- $\hat{\boldsymbol{a}}_{res}^{B}$ – accelerometer's output,
- $\boldsymbol{R}_{M}$ – rotation matrix due to assembly inaccuracy,

- $\boldsymbol{S}$ – diagonal scale matrix,
- $\boldsymbol{b}_0$ – sensor constant bias,
- $\boldsymbol{n}$ – random noise.

3.2.2. Gyroscope

A gyroscope is a sensor designed to measure angular velocities. In a non-inertial frame, angular velocity remains constant at every point, making gyroscope readings independent of its position within the moving reference frame.

The primary advantage of a gyroscope lies in its ability to measure raw angular velocities. However, it's important to note that this measurement is not direct; for example, in MEMS sensors, the measurement is based on the transfer of energy between two vibration modes caused by the acceleration of Coriolis [24].

Despite its advantages, a gyroscope's measurements are susceptible to various errors. Two significant sources of error are bias and drift. Bias refers to a constant offset in the measured values, which is inherent to the sensor and remains relatively stable over time. Drift, on the other hand, is the gradual change in bias over time, leading to a continuous deviation from the true value. In addition to bias and drift, gyroscopes are also affected by high-frequency noise, which can obscure the true signal. An equation (3.4) presents a mathematical model of gyroscope.

$$\hat{\boldsymbol{\omega}}^B = \boldsymbol{\omega}^B + \boldsymbol{b}(t) + \boldsymbol{b}_0 + \boldsymbol{h} \tag{3.4}$$

where:

- $\hat{\boldsymbol{\omega}}^B$ – gyroscope's output,
- $\boldsymbol{\omega}^B$ – moving frame angular velocity given in moving reference frame,
- $\boldsymbol{b}(t)$ – bias that change over time,
- $\boldsymbol{h}$ – high-frequency random noise.

While bias presence is inevitable in gyroscopes, it is crucial to monitor and account for drift, as it can significantly affect the accuracy of long-term measurements. Understanding and mitigating these errors are essential for ensuring the reliable performance of gyroscopes in various applications.

Finally, it is worth mentioning why the assembly inaccuracy are not included in sensor's model. Obviously, a gyroscope can be mounted improperly, but the effects of this flaw can be included in bias.

What will transpire later, the gyroscope is not used in absolute orientation estimation, and its error is compensated.

3.2.3. External position and orientation provider

External provides supply additional data that can appear to be useful in the further estimation. There are many sources providing various measurements. Some of them deliver only partial information, while the other can estimate a full description of the system. However, the measurements may be low-precise or too slowly updated to be used alone as an estimation.

An example considered in this project is the use of the robot's tip position calculated by its control system. The industrial robot acquires tip position by calculating forward kinematics based on encoder readings and knowledge about robot's structure. The orientation of tip is also calculated and can be used in estimation if the connection between robot's arm and mechanism is rigid.

The model of external providers as a sensor is not as straightforward as model presented so far. Many factors are involved in it, including, position and type of connection between tip and mechanism, relative positions of them, and the accuracy of robot's calculation. Moreover, the accuracy is not always directly specified, due to the fact that for industrial robots repeatability is much important that it's precision [25]. It is always reasonable to assume that reading include additional noise.

Assume that the tip of the robot is connected with the origin of moving coordinate frame with a rigid drawbar. The drawbar is double-ended with spherical joints so the orientation of the tip does not affect system. In this case the position provided by the robot's control system is a composition of the origin's position and translation inserted by the drawbar. An equation (3.5) presents the relation between an origin's position and a tip's position when the length and orientation of the drawbar are known.

$$\begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \boldsymbol{R}_z\left(az\right)\boldsymbol{R}_y\left(el\right)\begin{bmatrix} d \\ 0 \\ 0 \end{bmatrix} \tag{3.5}$$

where:

- x_t, y_t, z_t – 3 components of the tip's position,
- x, y, z – 3 components of the origin's position,
- d – a length of the drawbar,

- az, el – a description of the drawbar's orientation. The el is an elevation, the angle between the drawbar's translation vector and OXY world plane. The az is an azimuth, the angle between the drawbar's translation vector's projection on OXY plane and OX axis.

3.3. Sensor's calibration and filtration

3.3.1. Accelerometer

The purpose of accelerometer's calibration is to remove a bias and compensate effects of an assembly inaccuracy and a scale error. The accelerometer's noise compensation is usually moved to sensor fusion. Before proposing a calibration method, model given in equation (3.3) can be transformed to equation (3.6). The noise is ignored, while $\boldsymbol{R}_M\ \boldsymbol{S}$ forms new $\boldsymbol{\Phi}^{-1}$ coefficient matrix. The proof that the product of these matrices is the inverse of another 3 x 3 matrix is that the rotation matrix is an orthogonal matrix, and the scale matrix is a diagonal matrix. Both of these matrices are non-singular and so their product can also be inverted.

$$\hat{\boldsymbol{a}}_{res}^{B} \approx \boldsymbol{\Phi}^{-1}\left(\boldsymbol{a}_{res}^{B} + \boldsymbol{b}_0\right) \tag{3.6}$$

That being said, calibration model is proposed in equation (3.7).

$$\tilde{\boldsymbol{a}}_{res}^{B} = \boldsymbol{\Phi}\hat{\boldsymbol{a}}_{res}^{B} - \boldsymbol{b}_0 = \boldsymbol{\Phi}\hat{\boldsymbol{a}}_{res}^{B} + \boldsymbol{\Gamma} \tag{3.7}$$

where:

- $\hat{\boldsymbol{a}}_{res}^{B}$ – calibrated accelerometer's output,
- $\boldsymbol{\Phi}$ – 3 x 3 calibration's coefficient matrix,
- $\boldsymbol{\Gamma}$ – 3 x 1 calibration's coefficient vector.

In result, the accelerometer's calibration comes down to finding values of $3 \cdot 3 + 3 = 12$ coefficients. To achieve this, readings are collected when the sensor is in positions for which the expected values are known. An accelerometer, even if not moving, measures gravitational acceleration. Assume, that sensor is closed in box, that edges are parallel to its coordinates system. When the box rest on each of its face on flat table the accelerometer is exprected to measure $\begin{bmatrix} \pm g & 0 & 0 \end{bmatrix}^T$, $\begin{bmatrix} 0 & \pm g & 0 \end{bmatrix}^T$ or $\begin{bmatrix} 0 & 0 \pm g \end{bmatrix}^T$. Therefore, collecting measures from every face results in $6 \cdot 3 = 18$ equation that can be used to determinate values of calibration's coefficient. As there is more equations that unknowns, linear regression method is used. In the case of this work, it was decided to utilize Ordinary Least Squares method (OLS).

Before method applying, all the equations have to be transformed to a linear form given in equation (3.8).

$$\boldsymbol{X}\boldsymbol{\Theta} = \boldsymbol{Y} \tag{3.8}$$

where:

- $\boldsymbol{X}$ – n x m known matrix,
- $\boldsymbol{\Theta}$ – m x 1 vector, that value is estimated,
- $\boldsymbol{Y}$ – n x 1 known matrix.

The transformation to linear form is given in equations (3.9 - 3.11)

$$\boldsymbol{\Theta} = \begin{bmatrix} \Phi_{11} & \Phi_{12} & \Phi_{13} & \Gamma_1 & \Phi_{21} & \Phi_{22} & \Phi_{23} & \Gamma_2 & \Phi_{31} & \Phi_{32} & \Phi_{33} & \Gamma_3 \end{bmatrix}^T \tag{3.9}$$

$$\boldsymbol{Y} = \begin{bmatrix} 0 & 0 & g & \vdots & 0 & 0 & -g & \vdots & g & 0 & 0 & \vdots & -g & 0 & 0 & \vdots & 0 & g & 0 & \vdots & 0 & -g & 0 \end{bmatrix}^T \tag{3.10}$$

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{a}_T^T & 1 & \boldsymbol{0}_{3x1} & 0 & \boldsymbol{0}_{3x1} & 0 \\ \boldsymbol{0}_{3x1} & 0 & \boldsymbol{a}_T^T & 1 & \boldsymbol{0}_{3x1} & 0 \\ \boldsymbol{0}_{3x1} & 0 & \boldsymbol{0}_{3x1} & 0 & \boldsymbol{a}_T^T & 1 \\ \boldsymbol{a}_D^T & 1 & \boldsymbol{0}_{3x1} & 0 & \boldsymbol{0}_{3x1} & 0 \\ \boldsymbol{0}_{3x1} & 0 & \boldsymbol{a}_D^T & 1 & \boldsymbol{0}_{3x1} & 0 \\ \boldsymbol{0}_{3x1} & 0 & \boldsymbol{0}_{3x1} & 0 & \boldsymbol{a}_D^T & 1 \\ \boldsymbol{a}_F^T & 1 & \boldsymbol{0}_{3x1} & 0 & \boldsymbol{0}_{3x1} & 0 \\ \boldsymbol{0}_{3x1} & 0 & \boldsymbol{a}_F^T & 1 & \boldsymbol{0}_{3x1} & 0 \\ \boldsymbol{0}_{3x1} & 0 & \boldsymbol{0}_{3x1} & 0 & \boldsymbol{a}_F^T & 1 \\ \boldsymbol{a}_B^T & 1 & \boldsymbol{0}_{3x1} & 0 & \boldsymbol{0}_{3x1} & 0 \\ \boldsymbol{0}_{3x1} & 0 & \boldsymbol{a}_B^T & 1 & \boldsymbol{0}_{3x1} & 0 \\ \boldsymbol{0}_{3x1} & 0 & \boldsymbol{0}_{3x1} & 0 & \boldsymbol{a}_B^T & 1 \\ \boldsymbol{a}_R^T & 1 & \boldsymbol{0}_{3x1} & 0 & \boldsymbol{0}_{3x1} & 0 \\ \boldsymbol{0}_{3x1} & 0 & \boldsymbol{a}_R^T & 1 & \boldsymbol{0}_{3x1} & 0 \\ \boldsymbol{0}_{3x1} & 0 & \boldsymbol{0}_{3x1} & 0 & \boldsymbol{a}_R^T & 1 \\ \boldsymbol{a}_L^T & 1 & \boldsymbol{0}_{3x1} & 0 & \boldsymbol{0}_{3x1} & 0 \\ \boldsymbol{0}_{3x1} & 0 & \boldsymbol{a}_L^T & 1 & \boldsymbol{0}_{3x1} & 0 \\ \boldsymbol{0}_{3x1} & 0 & \boldsymbol{0}_{3x1} & 0 & \boldsymbol{a}_L^T & 1 \end{bmatrix} \tag{3.11}$$

where:

- $\boldsymbol{a}_T$ – measurement when box is top face up,

- $\boldsymbol{a}_D$ – measurement when box is down face up,
- $\boldsymbol{a}_F$ – measurement when box is front face up,
- $\boldsymbol{a}_B$ – measurement when box is back face up,
- $\boldsymbol{a}_R$ – measurement when box is right face up,
- $\boldsymbol{a}_L$ – measurement when box is left face up.

According to OLS method, estimation $\hat{\boldsymbol{\Theta}}$ is given in equation (3.12). Ultimately, the result is decomposed into $\boldsymbol{\Phi}$ and $\boldsymbol{\Gamma}$

$$\hat{\boldsymbol{\Theta}} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{Y} \tag{3.12}$$

Additionally, an intuitive procedure of collecting calibration measures is also part of the system. Usually accelerometer's calibration involves placing sensor on calibration table and approving measure (for example, by clicking a button). Moreover, in certain system, an order of sides that being measured is predetermined. This approach has a fundamental advantage, because it recognizes sensors mounted in any orientation. Nevertheless, if the sensor is roughly set properly (axes are directed in accordance, but not perfectly aligned), the calibration's process can be improved, by recognizing on what face sensor was plated. In this purpose sensor measurements are collected into statistics that calculates mean value and standard variance. If standard variance is below predefined threshold, it means that sensor is not moving. Next mean value is normalized and compared with every of 6 axes' unit vectors. If angle between normalized mean and unit vector is less that specified (that can be quickly checked using dot product), the face correlated with unit vector is recognized.

3.3.2. Gyroscope

The gyroscope's readings contain bias and high-frequancy noise. The constant component of bias can be calculated and removed in calibration. After sensor's start, gyroscope's readings are collected for specified period and create a statistic. Next, mean and standard variance is calculated. If standard variance is small enough it is assumed that the sensor is not broken, and it was not moved during the calibration. In the opposite case, the calibration is repeated. The mean is a calculated bias that will be subjected from reading. Unfortunately, there is no method that can remove the drift, before the system run. However, the drift tracking and correction can be a part of sensor fusion that will be presented further.

Moving on, the noise is removed by passing reading through filters. In order to remove high-frequancy noise, low pass filters are used. They are characterized by a cutoff frequency and an order. The cutoff frequency is the limit beyond which the higher frequency signal is dampened. The filter's order defines

the slope factor of this dampening. First order low-pass filter is given by equations (3.13 - 3.14). If readings are collected with a constant sampling rate, α coefficient is calculated only once.

$$\alpha = e^{-\omega_{cf}\Delta t} \tag{3.13}$$

$$\tilde{\boldsymbol{\omega}}^B(t) = \alpha\hat{\boldsymbol{\omega}}^B(t) + (1-\alpha)\,\hat{\boldsymbol{\omega}}^B(t-\Delta t) \tag{3.14}$$

where:

- $\tilde{\boldsymbol{\omega}}^B$ – filtered angular velocity,
- ω_{cf} – cut-off frequency,
- Δt – sampling interval.

Proper frequency selection requires involving multiple factors. If noise bandwidth was not specified by sensor's producer, the frequency can be selected by analyzing logs. Assume that gyroscope was collecting data (without filtration) when the mechanism was moving along the determined trajectory, and for this trajectory the exprected velocities are known. Next, spectral transform is conducted for registered data and expectation. Noise spectrum can be obtained by subtraction spectrums from last step. Ultimately, cut-off frequency should be placed above the highest frequency in expectation's spectrum and below the lowest frequency in noise spectrum.

It should be mentioned, that in specific cases it is rewarding to use a bandpass filters instead low-pass filter. The advantage of this approach is that the bandpass filter removes high-frequency noise, as well as bias. However, the measured signal often overlap the bias bandwidth and using such a filter leads to insensitivity to low velocities. Summarize this solution should be implemented only if an active drift correction is not possible or only high velocities are important for researches.

4. Sensor fusion

A sensor fusion is a method of blending multiple sensors' reading to obtain a desired estimation. The methods involve merging information from a variety of sensors, cross-checking as well as tracking and removing floating error. They utilize partial and redundant information to estimate complete estimation.

There are many methods of sensor fusion in the application of determining position and orientation [26]. The classic orientation's estimation methods, that are based on gyroscope, accelerometer, are Complementary filter method [1], Direction Cosine Matrix method [2] and Madgwick filter method [3].

Complementary filter method involves estimating orientation two ways: incrementally and absolutely. An incremental estimation means integrating gyroscope readings to rotate from a previous moment. The disadvantage of incremental estimation is that bias is also integrated. Besides, accelerometer and magnetometer readings are used to estimate an absolute orientation, assuming that accelerometer reading vector points to down and gyroscope reading vector points to north. It is not perfect assumption as the accelerometer measures all other accelerations too. Also, the magnetometer readings require declination and inclination correction. In result, the absolute estimation is very noisy. The main idea of the method is to filter the incremental esimation with high pass filter that remove bias, and filter the absolute estimation with low pass filter that remove noise. The filter are selected to be complementary, so the sum of estimations covers the full bandwidth.

Direction Cosine Matrix (DCM) method is also based on incremental model with drift correction. To correct drift DCM involves using PI controller. The method is strongly linked with rotation matrix's properties. Moreover, if velocity source is present, the method can explicitly compensate acceleration that is not gravitational.

Madgwick filter method is an algorithm developed by Sebastian Madgwick. The method uses a quaternion representation and structural definition, easy to implement in low-level languages. Unlike traditional methods like the Kalman filter, the Madgwick filter is highly efficient computationally, making it suitable for real-time applications even on resource-constrained platforms. However, it has

closed structure, that makes it less readable. Quoting the description of the Python filter implementation [27]: *The filter employs a quaternion representation of orientation to describe the nature of orientations in three-dimensions and is not subject to the singularities associated with an Euler angle representation, allowing accelerometer and magnetometer data to be used in an analytically derived and optimised gradient-descent algorithm to compute the direction of the gyroscope measurement error as a quaternion derivative.*

All of presented method so far enable orientation estimation only and are hardly modifiable. This results in the need to find more sophisticated methods. For many years the most popular and universal method has been the use of the Kalman filter [7].

4.1. An Kalman Filter

This section is directly taken and translated from my bachelor thesis. Should it be marked somehow?

The Kalman filter is a mathematical estimation method for efficiently tracking the dynamic state of a system and filtering out measurement data. It is based on modeling the system with state equations and measurement equations. The equations of state describe the evolution of the system state over time, while the measurement equations represent the relationship between the system state and the measurement data. Mathematically, this can be represented by equations (4.1 - 4.2).

$$\boldsymbol{x}_k = A \cdot \boldsymbol{x}_{k-1} + B \cdot \boldsymbol{u}_k + \boldsymbol{w}_k \tag{4.1}$$

$$\boldsymbol{z}_k = H \cdot \boldsymbol{x}_k + \boldsymbol{v}_k \tag{4.2}$$

where:

- $\boldsymbol{x}_k$ – state vector at the time of k,
- A – state matrix,
- B – input matrix,
- $\boldsymbol{u}_k$ – control input vector,
- $\boldsymbol{w}_k$ – process noise vector,
- $\boldsymbol{z}_k$ – measurement vector at the time of k,
- H – measurement matrix,
- $\boldsymbol{v}_k$ – measurement noise vector.

The equations presented above are similar to the equation of state of a system with white noise added. The idea of the filter is based on finding the state, which from a statistical point of view is the most probable, for the recorded measurements. The disadvantage of the above notation is the requirement of linear state equation. Fortunately, the solution to this problem is provided by the use of Extended Kalman Filter (EKF), which state is presented in equations (4.3 - 4.4).

$$\boldsymbol{x}_k = f\left(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k\right) + \boldsymbol{w}_k \tag{4.3}$$

$$\boldsymbol{z}_k = h\left(\boldsymbol{x}_k\right) + \boldsymbol{v}_k \tag{4.4}$$

where:

- $f\left(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k\right)$ – state function,
- $h\left(\boldsymbol{x}_k\right)$ – measurement function.

The Kalman filter performs two main steps: prediction and correction (figure 4.1). In the prediction step, the new state of the system and its covariance is predicted. In the correction step, based on the new measurements, an adjustment is made to the prediction, taking into account measurement errors and improving the estimation of the system state. The filter, especially in its extended version, is a major tool in the field of control and state's estimation, enabling systems to maintain the precision of measurements in dynamic conditions and improve the quality of system parameter estimation.

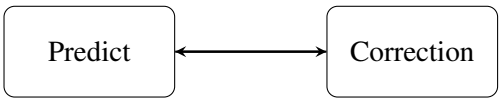

FIGURE 4.1 – An Extended Kalman Filter

Prediction phase:

$$\hat{x}_k^- = f(\hat{x}_{k-1}, \boldsymbol{u}_k) \quad \text{(predicted state)},$$

$$P_k^- = A_k P_{k-1} A_k^T + Q_k \quad \text{(predicted covariance)},$$

where: $A_k = \frac{\partial f}{\partial x}\Big|_{\hat{x}_{k-1}, \boldsymbol{u}_k}$, and Q_k is process noise's coveriance matrix.

Correction phase:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \quad \text{(Kalman's gain)},$$

$$\hat{x}_k = \hat{x}_k^- + K_k(\boldsymbol{z}_k - h(\hat{x}_k^-)) \quad \text{(corrected state)},$$

$$P_k = (I - K_k H_k) P_k^- \quad \text{(corrected covariance)},$$

where: $H_k = \frac{\partial h}{\partial x}\Big|_{\hat{x}_k^-}$, and R_k is measurement noise's covariance matrix.

4.2. An Extended Kalman Filter as position and orientation estimator

Application an Extended Kalman Filter to estimate position and orientation involves determination of state vector and state equation, specification measurement sources and their linkage with state. In prediction phase high sample rate sensors and sensor that measurements must be integrated should be used, like inertial sensor. The correction phase that can be processed rarely should include rest of sensors. In this section the filter's model will be presented.

In preparation, available sensors assigned into particular phase. The accelerometer and gyroscope's readings are used in prediction phase, as the reading are integrated. In correction phase external position source is used to correct position and accelerometer's readings are used to correct orientation. Thus, the equation (4.5) presents the proposed control input vector and equation (4.6) presents measurement vector. Next, it is necessary to define the state vector. The proposed state vector is presented in equation (4.7).

$$\boldsymbol{u} = \begin{bmatrix} g_x & g_y & g_z & a_x & a_y & a_z \end{bmatrix}^T \tag{4.5}$$

$$\boldsymbol{z} = \begin{bmatrix} a_x & a_y & a_z & p_x & p_y & p_z \end{bmatrix}^T \tag{4.6}$$

where:

- g_x, g_y, g_z – 3 components of gyroscope's reading,
- v_x, v_y, v_z – 3 components of accelerometer's reading,
- p_x, p_y, p_z – 3 components of position from external provider.

$$\boldsymbol{x} = \begin{bmatrix} x & y & z & v_x & v_y & v_z & q_0 & q_x & q_y & q_z & b_x & b_y & b_z & az & el \end{bmatrix}^T \tag{4.7}$$

where:

- x, y, z – 3 components of position,
- v_x, v_y, v_z – 3 components of velocity,
- q_0, q_x, q_y, q_z – quaternion of orientation,
- b_x, b_y, b_z – 3 components of gyroscope's bias,
- az and el – azimuth and elevation of drawbar.

The bias and orientation of draw bar require an additional comment. The gyroscope's bias in state vector enable drift tracking and its compensation. Thanks to an Kalman Filter's properties the value of

actual bias will be almost automatically calculated when the filter runs. The azimuth and elevation of drawbar enables utilizing external position sources even if they are not directly connected to system. In this implementation the position is given for point that is connected to system via link with fixed length. To simplify it is assumed that drawbar is connected by spherical joint to origin of moving coordinates system. For the vectors so selected, the equation of state was determined. To increase readability, the state equation was split into equations (4.8 - 4.15). The Jacobi matrix of the state function is calculated in equations (4.16 - 4.17).

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{k+1} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{k} + \Delta t \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}_{k} + \frac{1}{2}(\Delta t)^2 \boldsymbol{a}^W \tag{4.8}$$

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}_{k+1} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}_{k} + \Delta t \boldsymbol{a}^W \tag{4.9}$$

$$\begin{bmatrix} q_0 \\ q_x \\ q_y \\ q_z \end{bmatrix}_{k+1} = \begin{bmatrix} q_0 \\ q_x \\ q_y \\ q_z \end{bmatrix}_{k} + \frac{1}{2}\Delta t \boldsymbol{S}\left(\boldsymbol{q}_k\right) \left(\begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix}_{k} - \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}_{k} \right) \tag{4.10}$$

$$\begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}_{k+1} = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}_{k} \tag{4.11}$$

$$\begin{bmatrix} az \\ el \end{bmatrix}_{k+1} = \begin{bmatrix} az \\ el \end{bmatrix}_{k} \tag{4.12}$$

$$\boldsymbol{DCM}\left(\boldsymbol{q}_k\right) = \begin{bmatrix} q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_0 q_z) & 2(q_x q_z + q_0 q_y) \\ 2(q_x q_y + q_0 q_z) & q_0^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_0 q_x) \\ 2(q_x q_z - q_0 q_y) & 2(q_y q_z + q_0 q_x) & q_0^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix} \tag{4.13}$$

$$\boldsymbol{S}\left(\boldsymbol{q}_k\right) = \begin{bmatrix} -q_x & -q_y & -q_z \\ q_0 & -q_z & q_y \\ q_z & q_0 & -q_x \\ -q_y & q_x & q_0 \end{bmatrix} \tag{4.14}$$

$$\boldsymbol{a}^W = \boldsymbol{DCM}\left(\boldsymbol{q}_k\right)\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{4.15}$$

$$\boldsymbol{D}\left(\boldsymbol{q},\boldsymbol{a}\right) = 2\begin{bmatrix} a_x q_0 + a_y q_z - a_z q_y & a_y q_0 - a_x q_z + a_z q_x & a_z q_0 + a_x q_y - a_y q_x \\ a_x q_x + a_y q_y + a_z q_z & a_z q_0 + a_x q_y - a_y q_x & a_x q_z - a_x q_z - a_z q_x \\ a_y q_x - a_x q_y - a_z q_0 & a_x q_x + a_y q_y + a_z q_z & a_x q_0 - a_y q_z + a_z q_y \\ a_y q_0 - a_x q_z + a_z q_x & a_z q_y - a_y q_z - a_x q_0 & a_x q_x + a_y q_y + a_z q_z \end{bmatrix}^T \tag{4.16}$$

$$\boldsymbol{A}\left(\boldsymbol{x},\boldsymbol{u}\right) = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \left[\begin{array}{c|c|c|c|c} \boldsymbol{I}_{3x3} & \Delta t\boldsymbol{I}_{3x3} & \frac{\Delta t^2}{2}\,\boldsymbol{D}\left(\boldsymbol{q},\boldsymbol{a}\right) & \boldsymbol{0}_{3x3} & \boldsymbol{0}_{3x2} \\ \hline \boldsymbol{0}_{3x3} & \boldsymbol{I}_{3x3} & \Delta t\,\boldsymbol{D}\left(\boldsymbol{q},\boldsymbol{a}\right) & \boldsymbol{0}_{3x3} & \boldsymbol{0}_{3x2} \\ \hline \boldsymbol{0}_{4x3} & \boldsymbol{0}_{4x3} & \boldsymbol{I}_{4x4} & -\frac{\Delta t}{2}\,\boldsymbol{S}\left(\boldsymbol{q}_k\right) & \boldsymbol{0}_{4x2} \\ \hline \boldsymbol{0}_{3x3} & \boldsymbol{0}_{3x3} & \boldsymbol{0}_{3x4} & \boldsymbol{I}_{3x3} & \boldsymbol{0}_{3x2} \\ \hline \boldsymbol{0}_{2x3} & \boldsymbol{0}_{2x3} & \boldsymbol{0}_{2x4} & \boldsymbol{0}_{2x3} & \boldsymbol{I}_{2x2} \end{array}\right] \tag{4.17}$$

Next, a measurement function was determined. The accelerometer reading is used to correct orientation. The external position provider is linked with a position and drawbar orientation. The length of the drawbar is constant and equal d. The measurement function is given in equation (4.18). The Jacobi matrix of the measurement function is calculated in equations (4.19 - 4.21).

$$\boldsymbol{h}\left(\boldsymbol{x}\right) = \begin{bmatrix} \boldsymbol{DCM}\left(\boldsymbol{q}\right)\begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \\ \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \boldsymbol{R}_z\left(az\right)\boldsymbol{R}_y\left(el\right)\begin{bmatrix} d \\ 0 \\ 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 2g(q_x q_z + q_0 q_y) \\ 2g(q_y q_z - q_0 q_x) \\ g(q_0^2 - q_x^2 - q_y^2 + q_z^2) \\ x + d\,cos(az)\,cos(el) \\ y + d\,sin(az)\,cos(el) \\ z - sin(el) \end{bmatrix} \tag{4.18}$$

$$\boldsymbol{Ca}\left(\boldsymbol{q}\right) = \begin{bmatrix} 2q_y & 2q_z & q_0 & 2q_x \\ -2q_x & -2q_0 & 2q_z & 2q_y \\ 2q_0 & -2q_x & -2q_y & 2q_z \end{bmatrix} \tag{4.19}$$

$$\boldsymbol{Cp}\left(az,el\right) = \begin{bmatrix} -sin(az)\,cos(el) & -cos(az)\,sin(el) \\ cos(az)\,cos(el) & -sin(az)\,sin(el) \\ 0 & -cos(el) \end{bmatrix} \tag{4.20}$$

$$\boldsymbol{H}(\boldsymbol{x},\boldsymbol{u}) = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} = \left[\begin{array}{c|c|c|c|c} \mathbf{0}_{3x3} & \mathbf{0}_{3x3} & g\,\boldsymbol{Ca}(\boldsymbol{q}) & \mathbf{0}_{3x3} & \mathbf{0}_{3x2} \\ \hline \boldsymbol{I}_{3x3} & \mathbf{0}_{3x3} & \mathbf{0}_{3x4} & \mathbf{0}_{3x3} & d\,\boldsymbol{Cp}(az,el) \end{array}\right] \tag{4.21}$$

4.3. Time synchronization problem

Sensor's data flow to the server from multiple sensor's instance with various delays. The reading are used in prediction and correction phases. In correction phase, reading time is no so important, as this phase can be split into multiple correction. Formally, this approach is present in Multiplicative and Sequential Kalman Filters. Unfortunately, in predict phase all involved readings should be synchronized. If time of reading is known, the nearest readings are selected, or if calculation are conducted with fixed step, readings can be interpolated.

The time of reading is also not a trivial term, especially in decentralized system. There are many time determination method known, which differ in the terms and conditions ([28] [29]). Consider the case that assume precise clock in server and low-precision clock in every sensor node. An example method of solving the synchronization problem presents itself as follows. Every fixed period server broadcasts time synchronize message that contains only unique sequence number. Sending time T_1 (according to server's clock) is stored. Every node that receives synchronize message replies immediately with sequence number and node's clock time T_n. Received message is saved with timestamp T_2. Based on this reply server calculates transmission delay and clocks' offset. First transmission delay (commonly called *ping*) is calculated as a half of a period between broadcasting message and receiving reply (equation (4.22)). Next, node's clock offset in respect to server's clock is calculated and stored in array (equation (4.23)). Nodes send readings signed with timestamp, that is based on node's clock. Finally, true timestamp T_r in respect to server's clock is calculated by adding relevant offset (equation (4.24)).

$$ping = \frac{T_2 - T_1}{2} \tag{4.22}$$

$$offset = T_2 - ping - T_n \tag{4.23}$$

$$T_r = T_n + offset \tag{4.24}$$

5. Constraints

Constraints are the limitation placed on system's state. They define an allowed setup, that the system can reach. Due to the origin, many types of constraints are distinguished. They can result from system's structure, known trajectory or be directly linked with limitation of mathematical instrument used in system description.

A wide group of constraints is made of design constraints, i.e. the constraints that stem directly from mechanism's structure. They determine the directions in which the mechanism can move so as not to breach the constraints imposed by the existence of bonds.

5.1. Selected constrains

This section presents selected constraints that usually appear in robotics. Every constraint is briefly described and formulated as a function of system's state $\boldsymbol{c}(\boldsymbol{x})$ (constraint is satisfied when the function is equal to a vector of zeros). In view of its later use a derivative of constraint function by state $\frac{\partial \boldsymbol{c}(\boldsymbol{x})}{\partial \boldsymbol{x}}$ is also given.

5.1.1. Quaternion norm constraint

A quaternion is defined as four real numbers, and every coefficient is independent. However, when the quaternion is used as a description of orientation, its norm must be equal 1. Due to computer's precision repetitive quaternion rotating applied in EKF leads to error accumulation and to shrinking or stretching this norm. Moreover, the formula used in EKF is designed to keep constant quaternion norm only if the initial quaternion's norm was equal to 1.

In mechanical simulation this problem is solved by adding an extra pure-synthetic term to differential equation describing system [30]. Unfortunately, this method can not be easily adapted in case of Kalman Filter. The solution is to treat the quaternion norm equal to 1 as a constraint imposed on the system. Equation (5.1) defines the constraint function and equation (5.2) defines its derivative. Naturally, the

constraint function and its derivative is quaternion's dependent only, and it is scalar function.

$$c\left(\boldsymbol{q}\right) = \boldsymbol{q}^T\boldsymbol{q} - 1 \tag{5.1}$$

$$\frac{\partial c}{\partial \boldsymbol{q}}\left(\boldsymbol{q}\right) = 2\boldsymbol{q}^T \tag{5.2}$$

5.1.2. Position and orientation constraints

Position and orientation constraints are the constraints that are known before the system moves. They may affect only selected state variable and apply only at certain times. For example, the position of a robot may be precisely identified as long as the robot is close to sensor. The general formula that describes this is given by equation (5.3). The derivative of this function is a mostly-zero matrix with one only on position correlated with limited state variable.

$$c\left(\boldsymbol{x}\right) = \begin{bmatrix} x_i - f\left(t\right) \\ x_j - g\left(t\right) \\ \vdots \end{bmatrix} \tag{5.3}$$

The constraints which are given by an entangled function of multiple state variables are usually harder to formulate and require an individual analysis. Example of this type is given in next section.

Another example is a fixation of same parameter that can be calculated from system's state. Assume that the designed mechanism should keep a yaw angle of orientation equal to zero. If the state of system contains orientation given by quaternion, a yaw angle can be calculated from the following formula:

$$\psi = \text{atan2}\left(2(q_0q_z + q_xq_y), 1 - 2(q_y^2 + q_z^2)\right) \tag{5.4}$$

However, the value of *atan2* function is equal to zero then and only then the first argument is equal to zero. With that being said, the simple form of yaw constraint is given by equation (5.5). Equation (5.6) defines its derivative. The disadvantage of this approach is that the yaw angle equals to π also fulfills the constraint.

$$c\left(\boldsymbol{q}\right) = q_0q_z + q_xq_y \tag{5.5}$$

$$\frac{\partial c}{\partial \boldsymbol{q}}\left(\boldsymbol{q}\right) = \begin{bmatrix} q_z & q_y & q_x & q_0 \end{bmatrix} \tag{5.6}$$

5.1.3. Distance constraint

A distance constraint is defined as constraint that keeps a constant distance between a fixed point in space and a featured point on the moving element. Equation (5.7) formalize the so defined constraint.

$$\left| \boldsymbol{r}^{(0)} + R(\boldsymbol{q})\boldsymbol{s}^{(1)} - \boldsymbol{r}_p^{(0)} \right| - d = 0 \tag{5.7}$$

where:

- $\boldsymbol{r}^{(0)}$ – position of moving element, given in global coordinates frame,
- $R(\boldsymbol{q})$ – rotation matrix between world frame and element frame,
- $\boldsymbol{s}^{(1)}$ – position of featured point on the moving element, given in element's coordinate system,
- $\boldsymbol{r}_p^{(0)}$ – position of fixed point, given in global coordinates frame,
- d – distance between constraint's points.

However, a norm of vector requires calculation of squared root that value and derivative are more difficult to calculate. Without any loss of generality, constraint can be transformed to equation (5.8).

$$c(\boldsymbol{r}, \boldsymbol{q}) = \left(\boldsymbol{r}^{(0)} + R(\boldsymbol{q})\boldsymbol{s}^{(1)} - \boldsymbol{r}_p^{(0)} \right)^T \left(\boldsymbol{r}^{(0)} + R(\boldsymbol{q})\boldsymbol{s}^{(1)} - \boldsymbol{r}_p^{(0)} \right) - d^2 \tag{5.8}$$

In order to calculate derivative of distance constraint MATLAB Symbolic Toolbox was used (*distance_constraint_derivative.m*). Equation (5.9 – 5.15) presents calculated partial derivatives with respect to $\boldsymbol{r}$ and $\boldsymbol{q}$ elements.

$$\frac{\partial c(\boldsymbol{r}, \boldsymbol{q})}{r_x} = 2(x - r_{bx} + r_{ax}(2q_0^2 + 2q_x^2 - 1) + r_{ay}(2q_0q_z - 2q_xq_y) + r_{az}(2q_0q_y - 2q_xq_z))^2 \tag{5.9}$$

$$\frac{\partial c(\boldsymbol{r}, \boldsymbol{q})}{r_y} = 2(y - r_{by} + r_{ay}(2q_0^2 + 2q_y^2 - 1) + r_{ax}(2q_0q_z + 2q_xq_y) + r_{az}(2q_0q_x + 2q_yq_z))^2 \tag{5.10}$$

$$\frac{\partial c(\boldsymbol{r}, \boldsymbol{q})}{r_z} = 2(z - r_{bz} + r_{az}(2q_0^2 + 2q_z^2 - 1) + r_{ax}(2q_0q_y + 2q_xq_z) - r_{ay}(2q_0q_x - 2q_yq_z))^2 \tag{5.11}$$

$$\begin{aligned} \frac{\partial c(\boldsymbol{r}, \boldsymbol{q})}{q_0} &= 2(2q_yr_{az} - 4q_0r_{ax} + 2q_zr_{ay})(r_{bx} - x - r_{ax}(2q_0^2 + 2q_x^2 - 1) \\ &+ r_{ay}(2q_0q_z - 2q_xq_y) + r_{az}(2q_0q_y - 2q_xq_z))^2 \\ &+ 2(4q_0r_{ay} + 2q_xr_{az} + 2q_zr_{ax}) \cdot (y - r_{by} + r_{ay}(2q_0^2 + 2q_y^2 - 1) \\ &+ r_{ax}(2q_0q_z + 2q_xq_y) + r_{az}(2q_0q_x + 2q_yq_z))^2 + 2(4q_0r_{az} - 2q_xr_{ay} + 2q_yr_{ax}) \\ &\cdot (z - r_{bz} + r_{az}(2q_0^2 + 2q_z^2 - 1) + r_{ax}(2q_0q_y + 2q_xq_z) - r_{ay}(2q_0q_x - 2q_yq_z))^2 \end{aligned} \tag{5.12}$$

$$\begin{aligned}\frac{\partial c(\boldsymbol{r},\boldsymbol{q})}{q_x} &= 2(2q_0r_{az}+2q_yr_{ax})(y-r_{by}+r_{ay}(2q_0^2+2q_y^2-1)\\ &+r_{ax}(2q_0q_z+2q_xq_y)+r_{az}(2q_0q_x+2q_yq_z))^2\\ &-2(4q_xr_{ax}+2q_yr_{ay}+2q_zr_{az})(r_{bx}-x-r_{ax}(2q_0^2+2q_x^2-1)\\ &+r_{ay}(2q_0q_z-2q_xq_y)+r_{az}(2q_0q_y-2q_xq_z))^2-2(2q_0r_{ay}-2q_zr_{ax})\\ &\cdot(z-r_{bz}+r_{az}(2q_0^2+2q_z^2-1)+r_{ax}(2q_0q_y+2q_xq_z)-r_{ay}(2q_0q_x-2q_yq_z))^2\end{aligned} \tag{5.13}$$

$$\begin{aligned}\frac{\partial c(\boldsymbol{r},\boldsymbol{q})}{q_y} &= 2(2q_xr_{ax}+4q_yr_{ay}+2q_zr_{az})(y-r_{by}+r_{ay}(2q_0^2+2q_y^2-1)\\ &+r_{ax}(2q_0q_z+2q_xq_y)+r_{az}(2q_0q_x+2q_yq_z))^2\\ &+2(2q_0r_{az}-2q_xr_{ay})(r_{bx}-x-r_{ax}(2q_0^2+2q_x^2-1)\\ &+r_{ay}(2q_0q_z-2q_xq_y)+r_{az}(2q_0q_y-2q_xq_z))^2+2(2q_0r_{ax}+2q_zr_{ay})\\ &\cdot(z-r_{bz}+r_{az}(2q_0^2+2q_z^2-1)+r_{ax}(2q_0q_y+2q_xq_z)-r_{ay}(2q_0q_x-2q_yq_z))^2\end{aligned} \tag{5.14}$$

$$\begin{aligned}\frac{\partial c(\boldsymbol{r},\boldsymbol{q})}{q_z} &= 2(2q_xr_{ax}+2q_yr_{ay}+4q_zr_{az})(z-r_{bz}+r_{az}(2q_0^2+2q_z^2-1)\\ &+r_{ax}(2q_0q_y+2q_xq_z)-r_{ay}(2q_0q_x-2q_yq_z))^2\\ &+2(2q_0r_{ay}+2q_xr_{ax})(r_{bx}-x-r_{ax}(2q_0^2+2q_x^2-1)\\ &+r_{ay}(2q_0q_z-2q_xq_y)+r_{az}(2q_0q_y-2q_xq_z))^2-2(2q_0r_{ax}-2q_yr_{ay})\\ &\cdot(y-r_{by}+r_{ay}(2q_0^2+2q_y^2-1)+r_{ax}(2q_0q_z+2q_xq_y)+r_{az}(2q_0q_x+2q_yq_z))^2\end{aligned} \tag{5.15}$$

It is worth to mention that, if velocities are known, the equation function (5.8) can be differentiated with respect to time to get extra constraint $c(\boldsymbol{r},\boldsymbol{q},\frac{d\boldsymbol{r}}{dt},\frac{d\boldsymbol{q}}{dt})$. Nevertheless the position and velocity are strictly connected in EKF state function. In result the correction of position leads to correction of velocity.

5.2. Inclusion of constraints in estimation

Assume that a novel system for locating metro wagons is planned to be developed. The first idea that comes to mind is to use a satellite navigation system, as is usually the case in location-based tasks. However, this solution suffers from a significant flaw: these systems perform much worse underground. The question is whether the quality of the position estimate can be easily improved, and the short answer is yes. A simple assumption that can be made is that metro wagons can only be on the rails. Using

the well-known track, it is possible to stick the calculated location to the nearest rails thus improving its accuracy. This illustrative example implicitly demonstrates the principle of constraint's correction.

The inclusion of constraint in state estimation is not a popular solution. Usually transform to other state variables that fulfill constraints is suggested. However, it provokes the redesign of the state estimator every time, when constraints change. The unique approach is by Dan Simon in articles [10], [31] and [32]. It is proposed to add the third phase to conventional Extended Kalman Filter, as it is shown on figure (5.1).

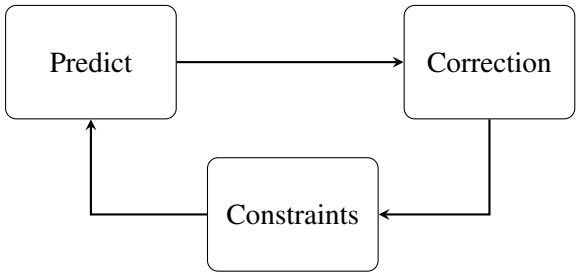

FIGURE 5.1 – An Extended Kalman Filter with constraint correction

In constraints correction phase only state of filter changes and covariance matrix remains unchanged. For linear constraints ($\boldsymbol{Dx} = \boldsymbol{d}$) equation (5.16) presents constraints correction phase.

$$\tilde{\boldsymbol{x}} = \hat{\boldsymbol{x}} - \boldsymbol{W}^{-1}\boldsymbol{D}^T\left(\boldsymbol{D}\boldsymbol{W}^{-1}\boldsymbol{D}^T\right)^{-1}(\boldsymbol{D}\hat{\boldsymbol{x}} - \boldsymbol{d}) \tag{5.16}$$

where:

- $\boldsymbol{D}$, $\boldsymbol{d}$ – matrix and vector defining linear constraints,
- $\boldsymbol{W}$ – square weight matrix,
- $\hat{\boldsymbol{x}}$ – filter state before correction,
- $\tilde{\boldsymbol{x}}$ – filter state after correction.

It is proposed to use inverse of covariance matrix as weight matrix $\boldsymbol{W} = \boldsymbol{P}^{-1}$. In case of linear constraints it leads to the maximum probability estimate of the state subject to state constraints.

Proof of correctness of this method is well known in literature. So as not to duplicate this, another proof will be present, by showing the similarity to Newton-Raphson nonlinear equation's solving method. Assume that $\boldsymbol{D}$ is squared matrix with full rank. In order to find solution of constraint equation, iterative method is used to solve equation (5.17). Newton-Rapshon method requires Jacobi matrix, that in this case is given in equation (5.18)

$$\boldsymbol{f}(\boldsymbol{x}) = (\boldsymbol{D}\boldsymbol{x} - \boldsymbol{d}) \tag{5.17}$$

$$\frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}} = \boldsymbol{D} \tag{5.18}$$

According to method every algorithm iteration is given in equation (5.19)

$$\tilde{\boldsymbol{x}} = \hat{\boldsymbol{x}} - \left(\frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right)^{-1} \boldsymbol{f}(\boldsymbol{x}) = \hat{\boldsymbol{x}} - \boldsymbol{D}^{-1}(\boldsymbol{D}\hat{\boldsymbol{x}} - \boldsymbol{d}) \tag{5.19}$$

In general $\boldsymbol{D}$ matrix is not full rank or is not squared, so it can not be inverse. To address this problem right pseudo-inverse can be used (equation (5.20)).

$$\boldsymbol{D}^{\#} = \boldsymbol{D}^T \left(\boldsymbol{D}\boldsymbol{D}^T\right)^{-1} \tag{5.20}$$

Furthermore, if the equation is underdeterminated, pseudo-inverse form a space of propel inverses, scaled by weight matrix $\boldsymbol{W}$ (equation (5.21)).

$$\boldsymbol{D}^{\#} = \boldsymbol{W}^{-1}\boldsymbol{D}^T \left(\boldsymbol{D}\boldsymbol{W}^{-1}\boldsymbol{D}^T\right)^{-1} \tag{5.21}$$

Inserting equation (5.21) to (5.19) leads to constraint correction equation (5.16).

The reasoning so far is based on the assumption that constraints are linear. However, constraints in mechanic are mostly nonlinear. This problem is addressed by local linearization method. Equations (5.22 – 5.23) define substitution resulting from linearization.

$$\boldsymbol{D} = \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{x}}(\hat{\boldsymbol{x}}) \tag{5.22}$$

$$\boldsymbol{d} = -\boldsymbol{c}(\hat{\boldsymbol{x}}) + \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{x}}(\hat{\boldsymbol{x}})\hat{\boldsymbol{x}} \tag{5.23}$$

In result, constraint correction method can be used for nonlinear equation.

As a part of this thesis, constraint correction method was developed. Based on its similarity to Newton-Raphson method it is proposed to use additional coefficient that scale correction α, and k-times repeats in one phase. The final correction is given in equation (5.24).

$$\hat{\boldsymbol{x}}_{i+1} = \hat{\boldsymbol{x}}_i - \alpha\boldsymbol{P}\boldsymbol{D}^T \left(\boldsymbol{D}\boldsymbol{P}\boldsymbol{D}^T\right)^{-1} (\boldsymbol{D}\hat{\boldsymbol{x}} - \boldsymbol{d}) \quad \text{for i = 1,2,..,k} \tag{5.24}$$

Parameters selection is part of tuning.

6. Designed system

6.1. Specification

Based on the knowledge gathered, the functional requirements for the system were formulated and described using use case methodology, where system is an actor.

System:

- uses inerial sensor the measure accelerations and angular velocities,
- retrieves data from external sensor and data providers,
- filters raw data,
- conduct sensor fusion to estimate position and orientation,
- store readings and estimations in files,
- displays system's state for operator.

Beside above requirement, also non-functional requirements were formed. Table (6.1) presents requirements using FURPS metodology.

6.2. System architecture

The system is organized into modules. The individual modules perform the following tasks:

server – central module processing and storing data. Performs raw data filtration, sensor fusion and saving logs. Communicate with sensors and provides system's time. Issues the API used by the front-end application.

node_udp – sensor node's source code. Performs raw data collecting and efficient forwarding to server. Due to abstraction layer software is highly independent of node's hardware.

client – front-end application to visualize system's status and control it. Ultimately constitutes the main human-machine interface.

constraints – library providing constraints' equations. Separates mechanical structure from server source code.

bridge – sensor node's mock, intended as rapid prototyping tool. Connects to server via same protocol as node_udp.

docker – Docker container's configuration. Container wraps the server and makes it independent of host machine. Prepared image can be run on every Docker-compatible machine.

Figure (6.1) shows system architecture's diagram with communication paths between modules.

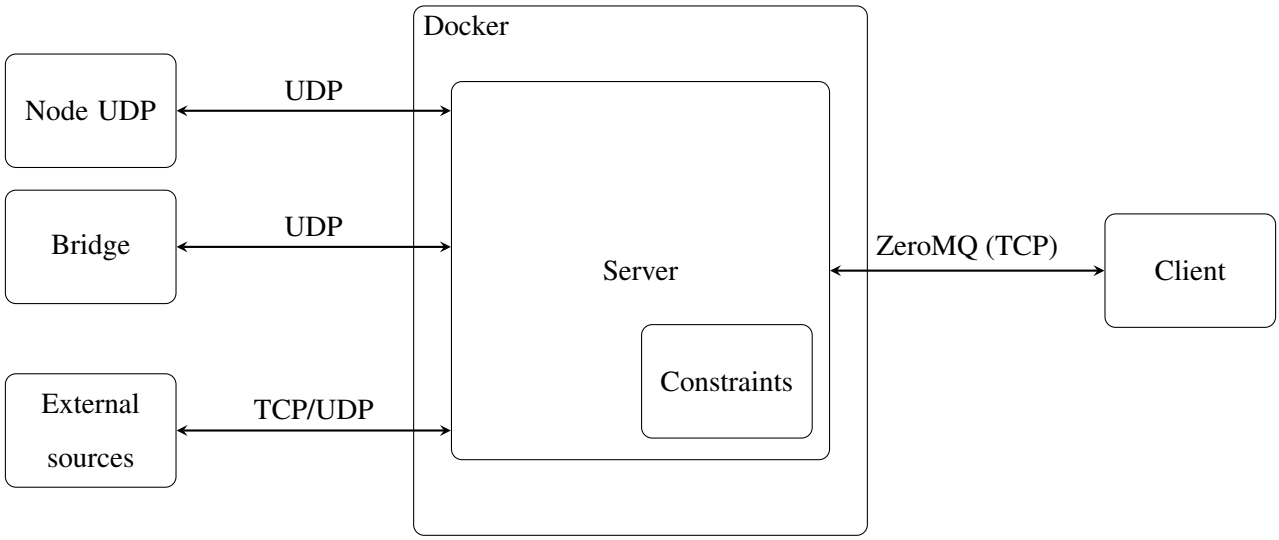

FIGURE 6.1 – System architecture's diagram

6.3. Technologies

The implementation of the project used various software technologies and libraries. The following summary explains the choice of specific technologies.

server, constraints – due to the large computational effort required to run online calculations, the C++ language was chosen. Because of its performance and flexibility, it is a natural choice for efficient computer simulations. In addition, the following libraries were used:

- Eigen3 – a library containing elements of linear algebra: matrices, vectors and related algorithms.

Eigen3 prioritizes efficiency by using SIMD and maintains a clear syntax.

- ZeroMQ – libzmq library binding for C++. Libzmq is a base implementation of ZeroMQ queues written in C,
- yaml-cpp – a library providing YAML serializer and deserializer,
- Boost – multi-purpose C++ library, in case of this project Boost provides asynchronous TCP and UDP client.
- libmodbus – a library providing support for MODBUS protocol

node_udp – embedded source code is also written in C++, due to the fact that microcontroller's producers usually provides a base chip's API using C or C++. C++ is currently the first choice language for Atmega, STM and ESP chips.

client – having in mind the provision of cross-platform, the Qt library was used to implement the front-end application. Further, to avoid multiple compilation Python Qt library was used.

bridge – the implementation also uses Python, but due to its other features. Bridge is supposed to be rapid developed, and its code should be understood for as many developers as possible,

docker – as the name suggests, Docker was used to containerize system.

6.4. Prototype

Project realization includes the preparation of software and hardware. The hardware part is the sensor node, that perform low-lever sensor data collecting and transferring data to server. The selected hardware should provide adequate performance and support network communication. For this purpose, microcontrollers work well. Wemos D1 mini Pro development board was selected. The ESP8266EX chip used on it provides high performance and native support for network functions including Wi-Fi. Board is connected to two shields: power supply shield and custom shield with IMU. Power supply shield enables chip powering from Li-ion battery. Custom shield has installed Pololu MinIMU-9 that contains LSM303 accelerometer and L3GD20 gyroscope. Thus created device is very compact and together with battery fits into a 50x40x35mm 3D-printed enclosure (figure (6.2)).

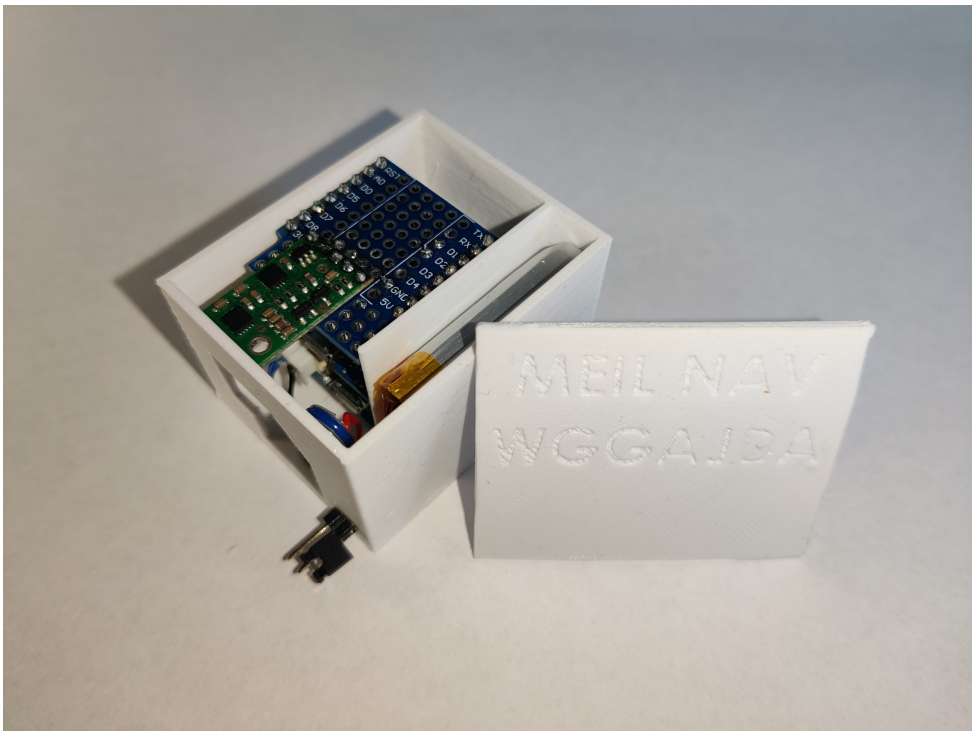

FIGURE 6.2 – Sensor node

Requirement		No.	Description
Usability	Usability	1	Server runs natively on UNIX or in Docker containter. Front-end application is multi-platform.
	Ergonomy	2	User interface should be transparent and intuitive.
Reliability	Precision	3	System should maximize estimation precision for collected readings.
	Verifiability	4	Estimation results should match predictions and be verifiable.
Perf.	Performance	5	System performance should scale relative to number of sensors.
Supportability	Maintenance	6	System implementation should be transparent and easy to develop.
		7	System should be divided into modules, that can be modified separately.
	Installability	8	Installation process should be easy. It is recommanded for front-end software not to be installed.
	Configuration	9	Software should be magic-number-free and all parameters should be configurable.

TABLE 6.1 – Non-functional requirements – FURPS

6.5. Graphical user interface

Figures (6.3 - 6.6) present graphical user interface of client module.

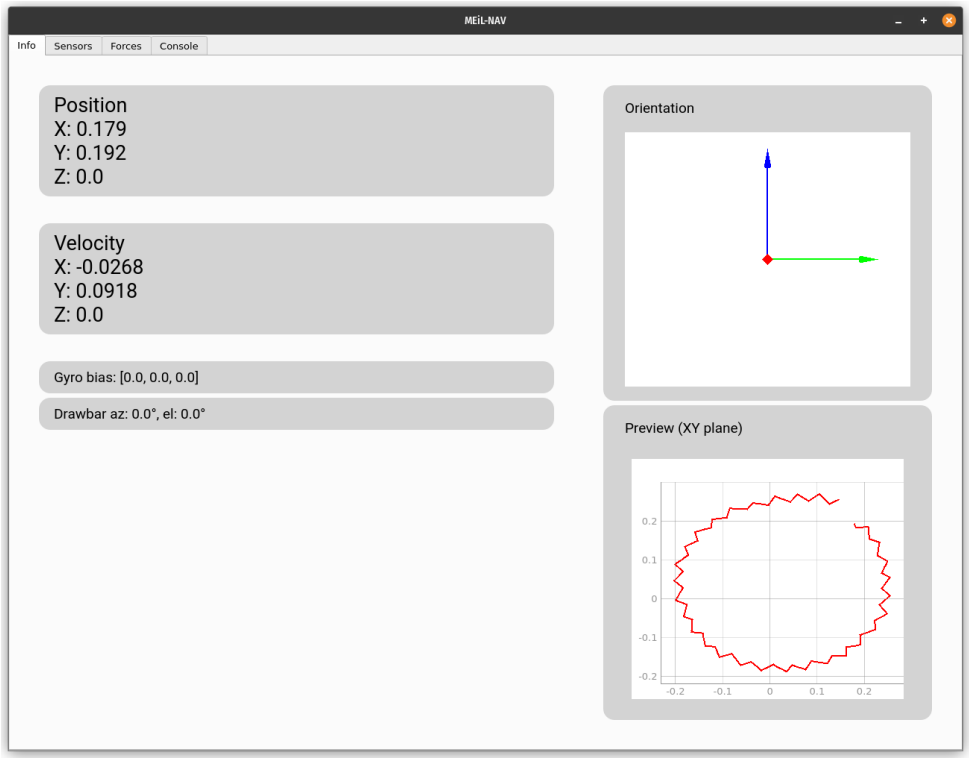

FIGURE 6.3 – GUI: main dashboard

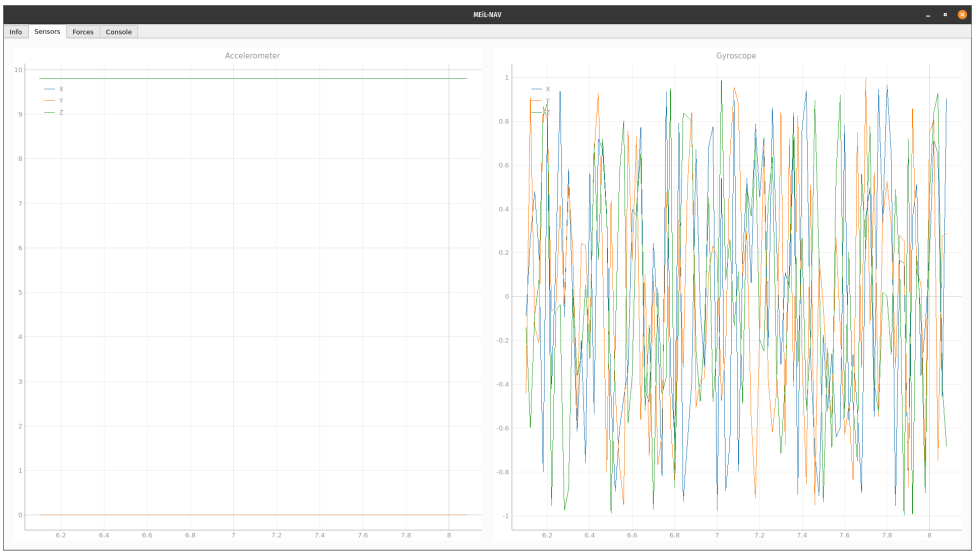

FIGURE 6.4 – GUI: sensor tab

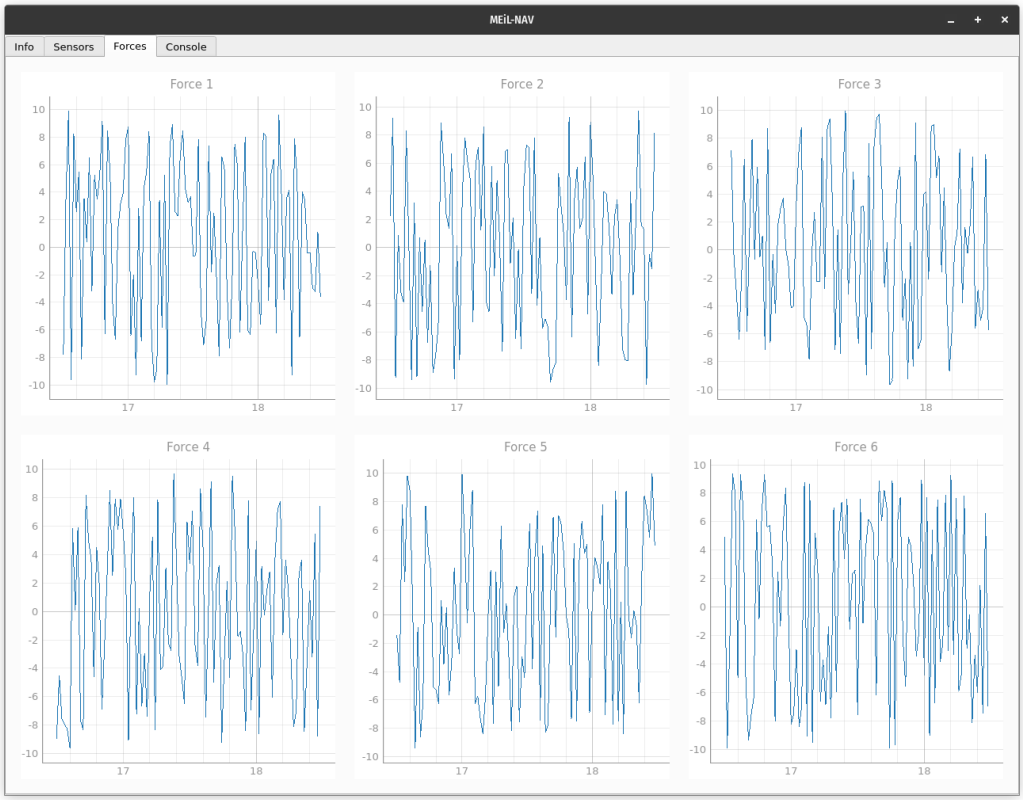

FIGURE 6.5 – GUI: force tab

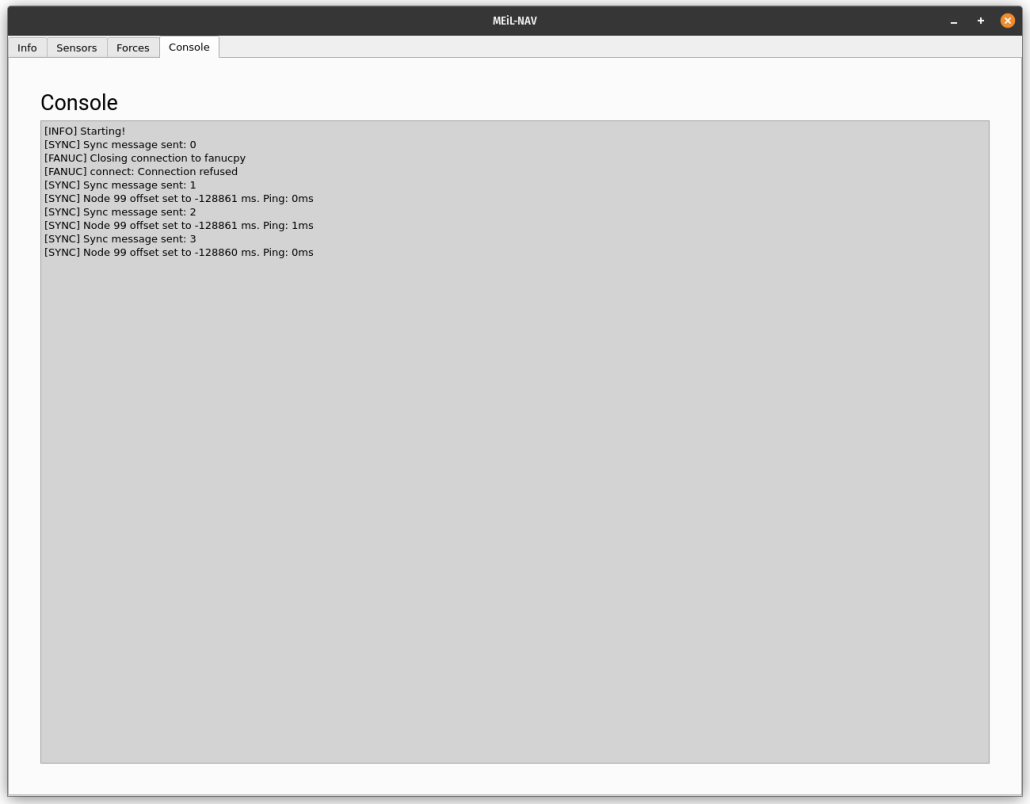

FIGURE 6.6 – GUI: console tab

6.6. Plugins

The developed software creates an advanced multitask system. During its preparation there was a lot of attention paid to make software flexible and easily updated. This allows you to enrich the system with features needed for specific applications. Those addons, that are not strictly connected with system's main task are called plugin.

An example of plugin, that was added to system is force logger. The main task of this plugin is collecting data from tension transducer connected via MODBUS TCP. For this purpose force logger utilizes already implemented timestamp system and save logs in project's specific format. Stored data are also synchronized with other estimations.

6.7. Tuning

Tuning is a process selection of a set of parameters provides the best results in terms of the chosen criterion. For the solution being developed, the greatest number of unknowns are the parameters of sensor filtration and fusion. Based on similar projects and applications, the initial value of parameters can be selected, but the tuning is an unmissable step on the way to precise estimation.

In measurement systems, the final result is affected by almost every element of the system: sensor errors, installation errors, delays and more. Due to that, there is no universal method of tuning. The process largely depends on the operator's experience and ability to recognize system responses. Nevertheless, it is prolific to plan the experiments that highlight certain regions of interest and, as a result, lead to shorter tuning times.

The project took care to include all the parameters in a configuration file that is freely editable. Parameters were divided into categories and a few of them can be modified on-line, when the system runs. The source code is devoid of modifiable numerical constants (so-called *magic numbers*). Additionally, the configuration file contains software variables that define network topology, addressing and other parameters that are configured during deployment.

7. Case study

Case study describes a solution's deployment on the mechanism that was a motivation for this project. The mechanism shown on figures (7.1 - 7.2) is an overconstrainted multibody system ([33]), that involves moving platform connected to ground by 6 rigid rods. The lower and upper rods are parallel to each other and have equal length. Although constraints describing the mechanism form a six-dimensional system of equations, they are linear dependent what is observable as a movement of the platform. The mechanism has one degree of freedom, except for two points where a bifurcation occurs. Due to this anomaly it's hard to simplify mechanism model to minimal set of coordinates.

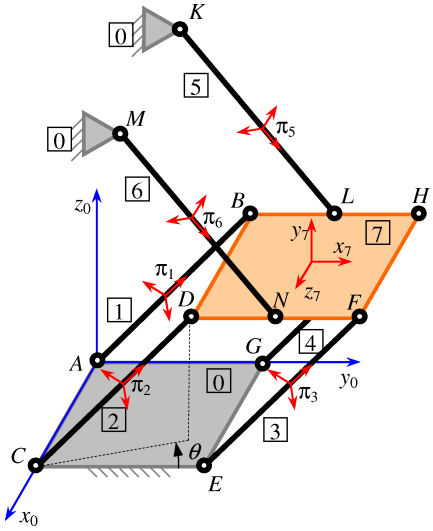

FIGURE 7.1 – Multi-body system with 1 DOF

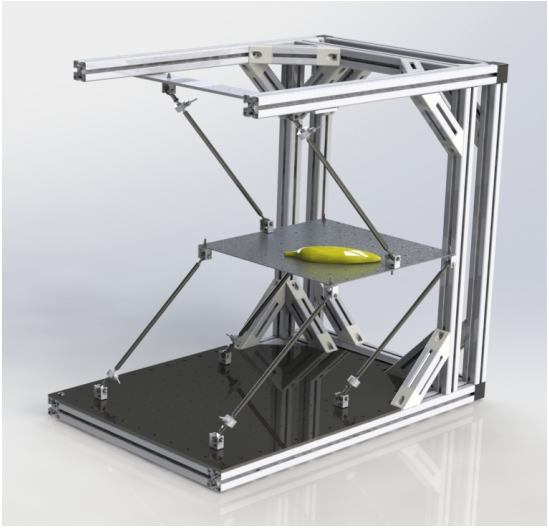

FIGURE 7.2 – Render of MBS

The scientific significance of this mechanism is revealed in the determination of the reaction forces that occur in rods. Mechanism is statically indeterminate which makes it even more difficult to analyze and highlights its research value. The further researches require collecting data from test stand, when the platform moves. State description should contain platform's position and orientation as well as readings from force sensor installed in rods, what is covered by developed system.

The mechanism is driven by FANUC M-10iA serial industrial robots. The robot's control system provides arm's tip position. To avoid problems resulting from inaccurate manufacturing the robot is connected to platform center via a drawbar, so the tip's orientation is unusable. Moreover, the position's

refresh rate is relatively slow, but can be successfully used in sensor fusion.

The mechanism thus presented is a representative example of the application of the developed system. Due to its properties, convetional methods are hard to apply. The proposed solution is challenging as well, but the procedure that lead to working system are structured and split into steps.

7.1. Computer simulation

Before real-life experiments, the fundamental aspects of projected were tested in computer simulation. The simulation involves preparing kinematic simulation in ADAMS and connecting it with simplified estimation system in MATLAB Simulink. A finished model was used, created earlier as part of a research project. The ADAMS's model was extended with new cylinder-shaped body that represents the robot's tip. The added part is connected with the moving platform's center and will be used as a motion source. A position of the simulated robot's tip is set as a control variable, that input to the program. The kinematic's simulation's outputs are postion and orientation of the moving platform, and its linear acceleration and angular velocity expressed in local coordinates system. Those variables will be used to simulate sensors' readings. A figure (7.3) presents multi-body system modeled in ADAMS software. Thus prepared model was converted to a Simulink block.

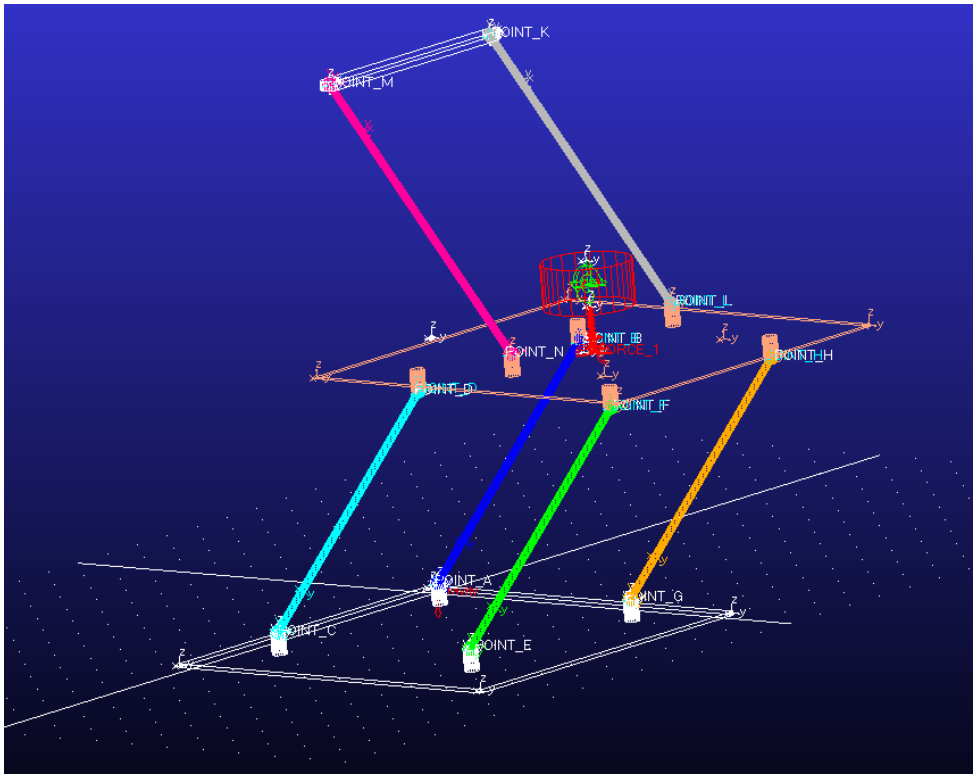

FIGURE 7.3 – The mechanism modeled in ADAMS software

Next, to move the mechanism trajectory generator was prepared. One of the allowed trajectory is a circular motion in the horizontal plane. In order to check position tracking the tangential velocity was constantly increased. Equations (7.1 - 7.3) presents the 3 components of the robot's tip as a function of time.

$$x_{TCP} = x_0 + R\,cos(t^2) \tag{7.1}$$

$$y_{TCP} = y_0 + R\,sin(t^2) \tag{7.2}$$

$$z_{TCP} = z_0 \tag{7.3}$$

With the platform that already moves, the following stage is to simulate inertial sensors: an accelerometer and a gyroscope. The sensors are simulated based on ADAMS block outputs. The gyroscope returns an angular velocity, when the accelerometer returns linear velocity with gravitation acceleration added. A figure (7.4) presents Simulink blocks that represents an ideal accelerometer.

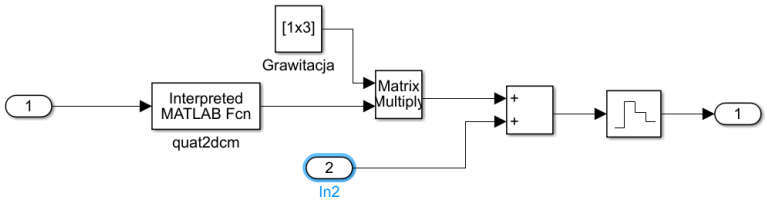

FIGURE 7.4 – The simulation of an accelerometer

The outputs of ideal sensors' simulation are further passed to blocks that represents sensor's error. In this simplified simulation the sensor's error was limited to adding a pink noise and bias. A figure (7.5) presents Simulink blocks that represents a sensor's error.

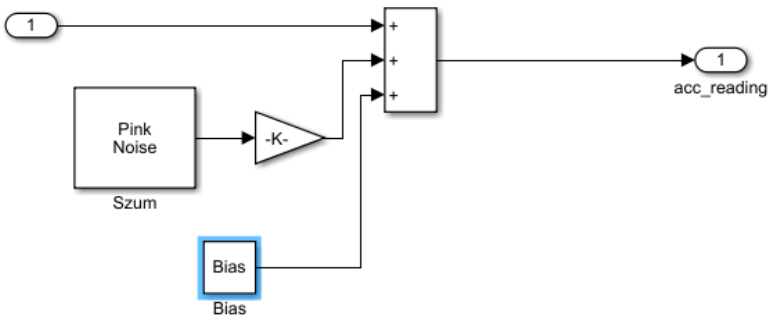

FIGURE 7.5 – The simulation of a sensor's error

The final step was to implement and tune the Kalman Filter with constraints' correction. Once again, the MATLAB implementation is a simplified version of the filter described in section (4.2). An implemented constraints are given in equation (7.5). A figure (7.6) presents the realization of filter in Simulink MATLAB.

$$x^2 + y^2 - r^2 = 0 \tag{7.4}$$

$$z = 0 \tag{7.5}$$

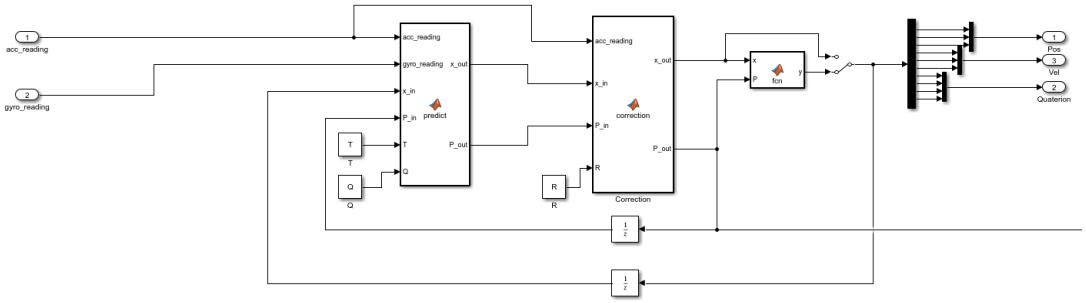

FIGURE 7.6 – The Kalman Filter with constraints' correction

Thus prepared simulation was run multiple time to select an optimal set of parameters. The presented results are the best achieved after tuning process. First the filter work was examinated without the constraints' correction. A figure (7.7) presents the coordinates change in function of time. The plot includes the robot's tip position, the moving platform's center's position and its estimation. The estimation (magenta, yellow and blue colors on plot) drifts heavily over simulation's time.

A figure (7.8) presents the run of same simulation, but with constraints' correction turned on. The estimation match the position for most of the simulation. During the second revolution the system lost tracked value but the match was quickly restored.

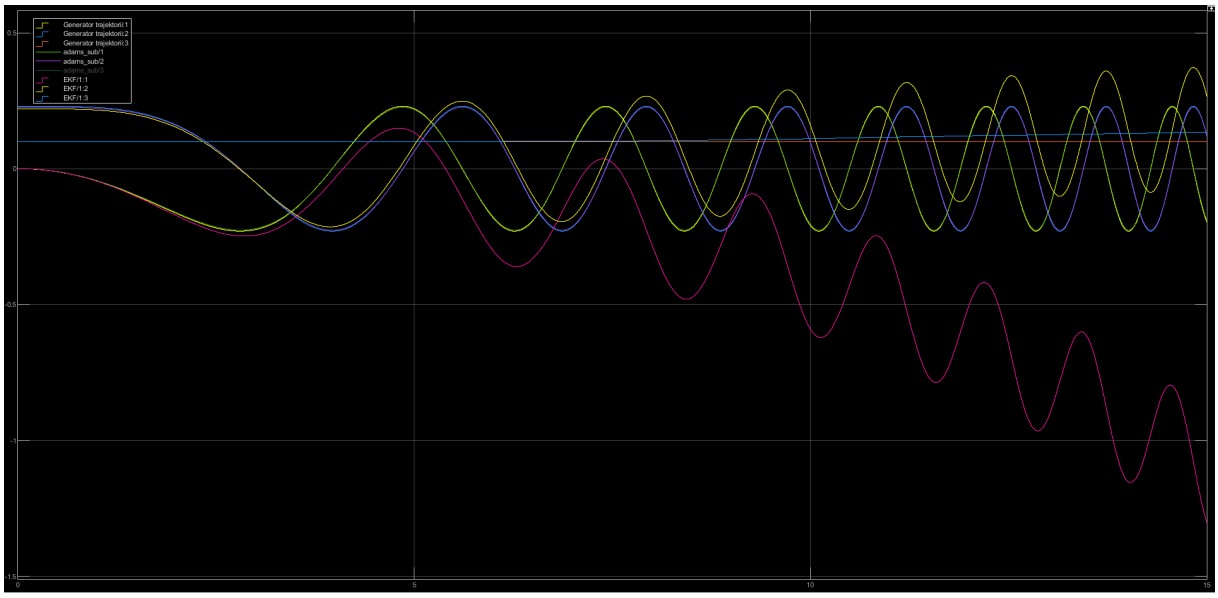

FIGURE 7.7 – The result of simulation (without correction)

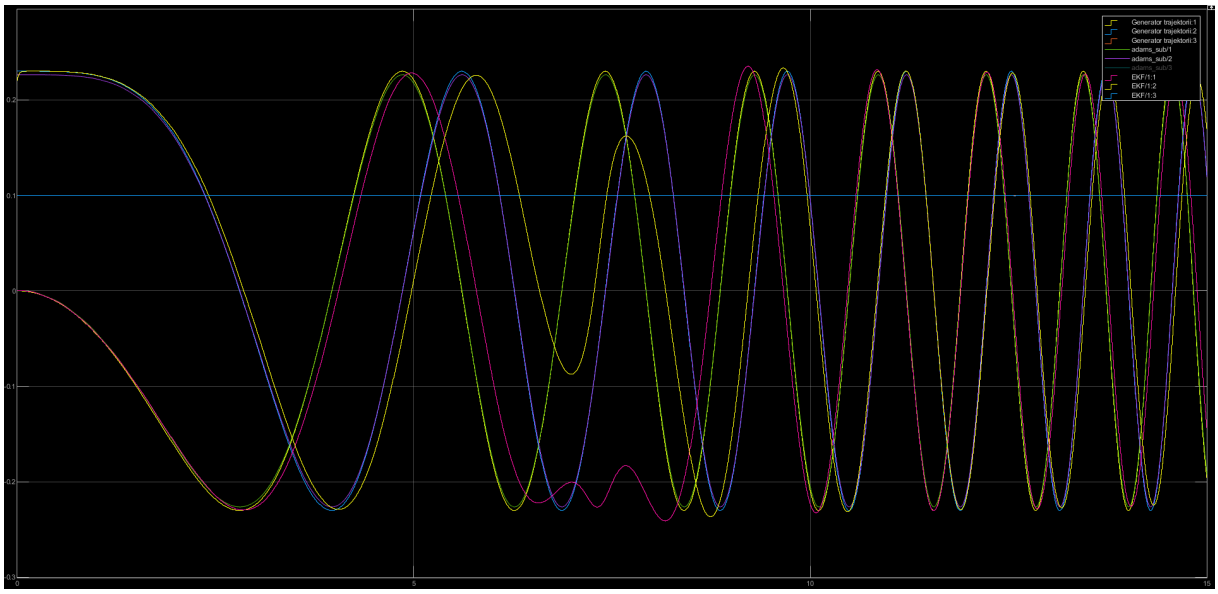

FIGURE 7.8 – The result of simulation (with correction)

Summarize, the simulation shows how positive is the impact of using the constraints' correction. Chronologically, the computer simulation was part of preparations for the realization of this thesis that energized the further work and complete system's implementation.

8. Experiments

The aim of experiments is a validation of system's functionality. According to plan, the system was firstly calibrated and tuned. Sensors' calibration was conducted on a flat surface, before mounting. Next fusion's parameters were tuned when sensor mounted on moves along previously prepared trajectories.

First, the orientation estimation was examined. The node was rigidly linked to the robot's tip. A figure (8.1) presents sensor mounted in the robot's gripper. The sponges were used to mitigate vibrations. Then, the robot executed a predetermined motion in which the position remained constant while only the orientation changed, among five learned configurations. The orientation calculated by the robot's control system was compared with the orientation estimated by the inertial sensors. Due to lack of magnetometer the yaw angle could not be properly estimated so its analyzes was skipped. However, the yaw angle can be fixed by using appropriate constraint (equation 5.5). The experiment was repeated without and with constraint's correction and with various speeds. A listing (8.1) presents the KAREL program that realizes this trajectory.

```
To fill up...
```

LISTING 8.1 – The KAREL program realizing an orientation changes

Next, the system was examined under the default operating conditions – estimates the position and the orientation. The node was mounted on plate that starts circular movement with different frequency and direction, according to mechanism constraints. The estimation was conducted in multiple configuration, that included:

- case A – position provider + gyroscope (accelerometer was used only to correct orientation),
- case B – position provider + gyroscope + accelerometer + constraint's correction,
- case C – position provider + constraint's correction,
- case D – position provider + gyroscope + accelerometer,
- case E – position provider + gyroscope + accelerometer + constraint's correction
- case F – position provider + gyroscope + accelerometer + simplified constraint's correction

The configuration without position provider was also examined, but the results obtained were much worse in terms of quality. During experiments it is assumed that orientation does not change and the

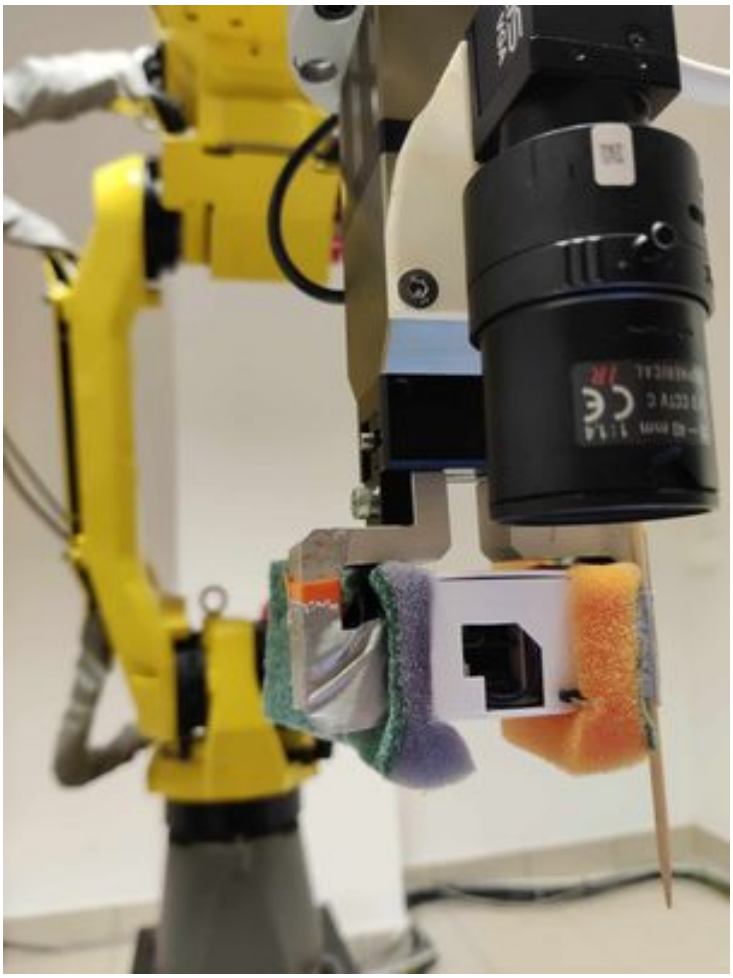

FIGURE 8.1 – Sensor mounted in the robot's gripper

position provided sample rate was limited to 2 Hz. The trajectory generated by the industrial robot is similar to that given in the equations (7.1 - 7.3), but due to the robot's control system limitation the speed is discreetly increased twice per revolution. A listing (8.2) presents the KAREL program that realizes this trajectory.

```
To fill up...
```

LISTING 8.2 – The KAREL program realizing a circular motion

8.1. Results

The gathered logs were post-processed and compared using MATLAB scripts. Figures (8.2 - 8.4) present the orientation's estimation from the first experiment. The estimated orientation angles are compared with the orientation calculated by the robot's positioning system.

Figures (8.5 - (8.10) present the position and orientation estimations obtained for the proposed configuration. It is expected that x and y coordinates estimate circular motion, while the z and orientation angles remain zeros.

The profit from an application of the constraints' correction is eminently visible when corrections with various "strength" are compared. A figure (8.11) presents an estimation of position in circular motion for different parameters (see equation (5.24)). The sample rate of external position provider is constant, so there is increasingly less update for every rotation. If correction is not active, the estimation tends to drift in the peaks. However, the "stronger" correction is active, the less error in those moments.

8.2. Results discussion

The first experiment showed that the system is able to provide adequate results based on inertial sensors' readings only. The estimation is reliable for slow and fast motions, where the fast motion is defined as a limit of used industrial robot. However, in minimal configuration only roll and pitch angles can be estimated. In order achieve full orientation estimation, an additional system information is needed.

The second experiment reflected a behavior of the system under normal operating conditions. It turned out that in case B and E the estimated position was noise. The reason of that phenomenon is specificity of the constraints. For examined mechanism, the orientation other that $\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$ is strictly linked with specified point in space. Moreover, the implemented correction is more likely to change position rather that the orientation. In result, when the orientation drifts, the correction tries to find the position that will fulfill the constraints' equations. To prove that hypothesis, case F was added to an experiment. It shows that if the orientation and position are not linked in constraints the presence of inertial sensor and their drifts do not lead to noisy estimation. Next, the results cast doubt on the profitability of using an accelerometer in the prediction phase. Its presence leads to disturbance in z coordinate. However, this does not exclude accelerometer use in correcting orientation. Furthermore, the influence on constraints' correction in especially visible in estimation amplitude.

Despite the described issues, it is possible to choose a configuration without major defects. The research conducted allows to conclude that the developed system can correctly estimate the position and orientation.

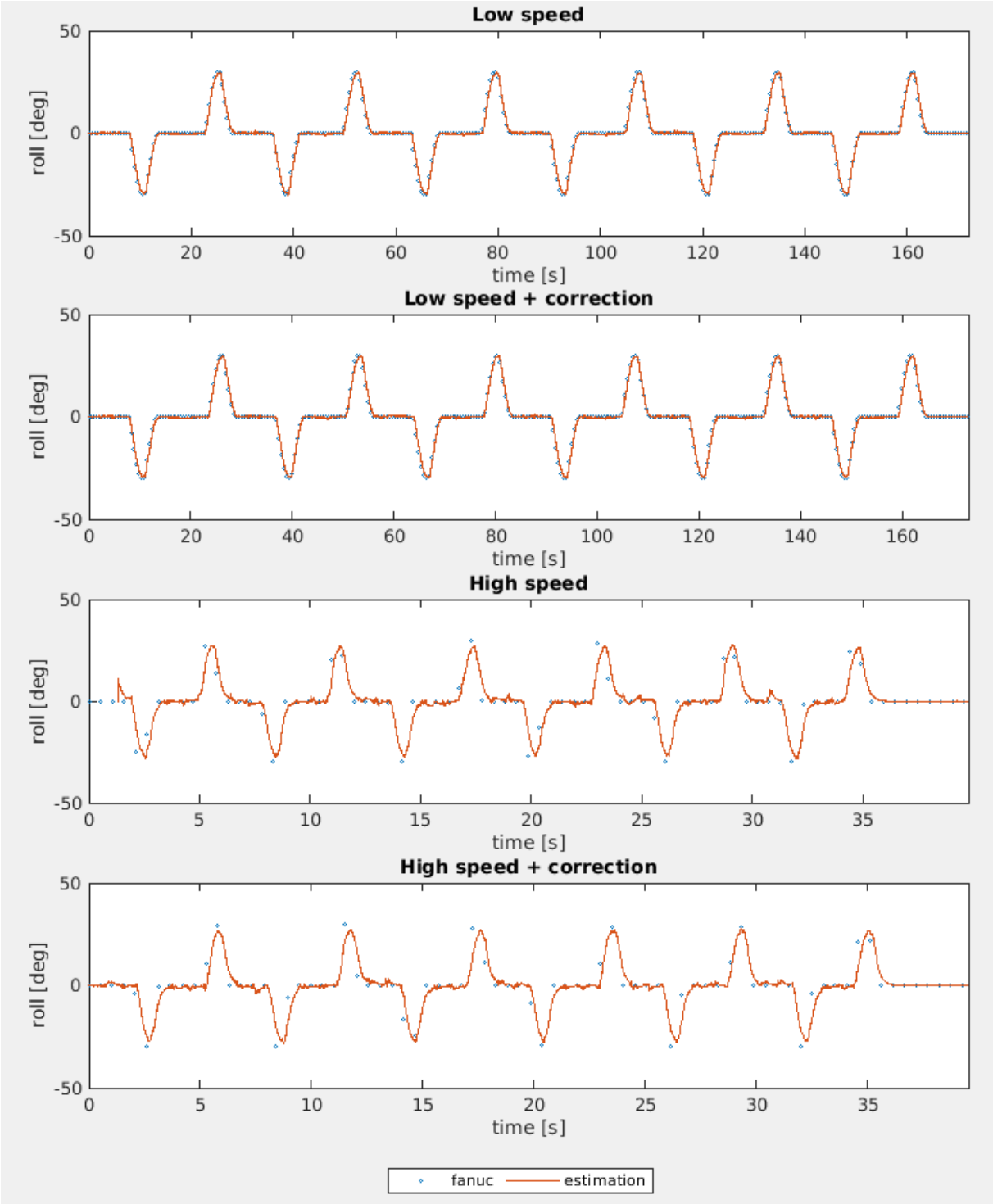

FIGURE 8.2 – Experiment 1: orientation estimation – roll

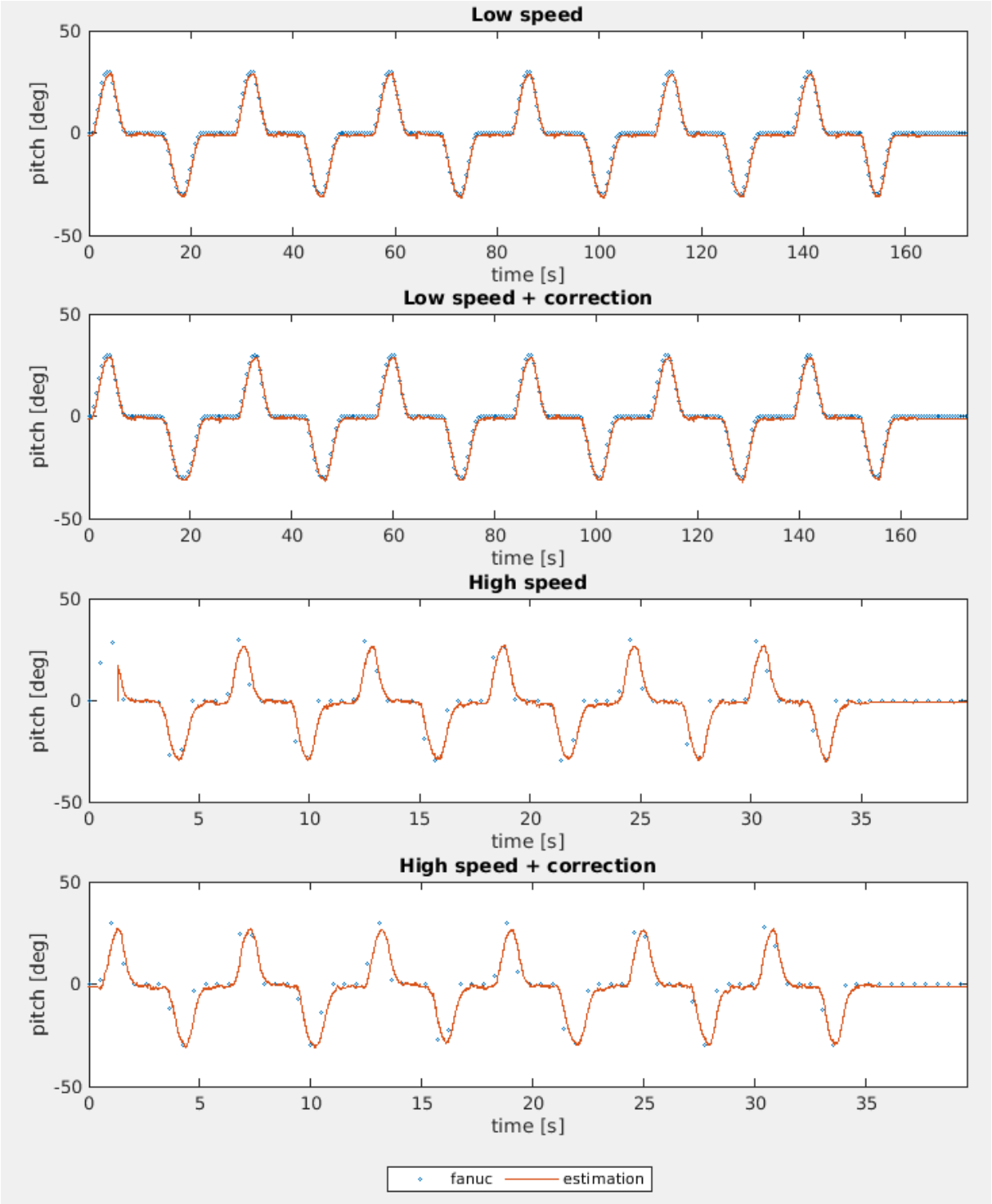

FIGURE 8.3 – Experiment 1: orientation estimation – pitch

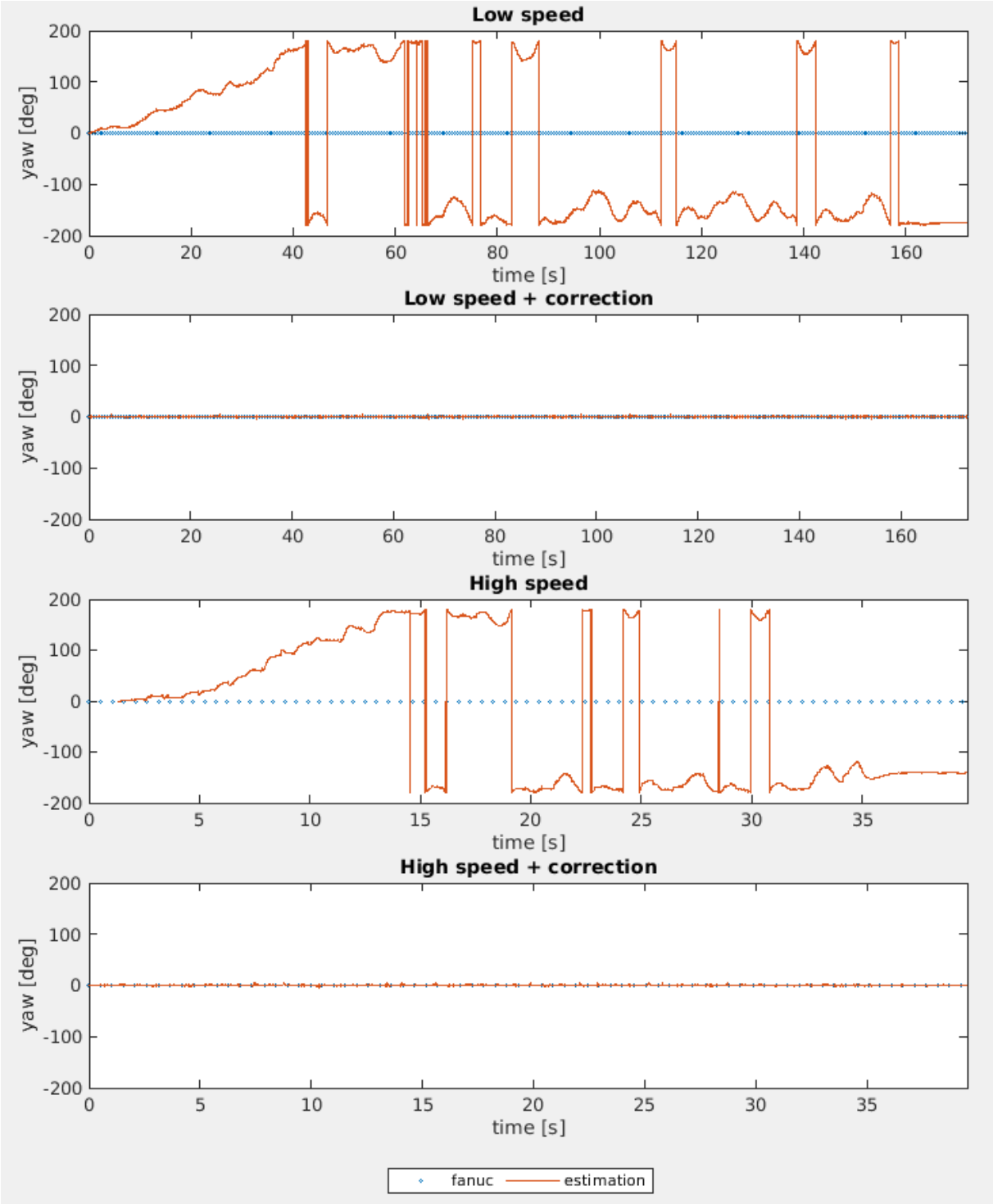

FIGURE 8.4 – Experiment 1: orientation estimation – yaw

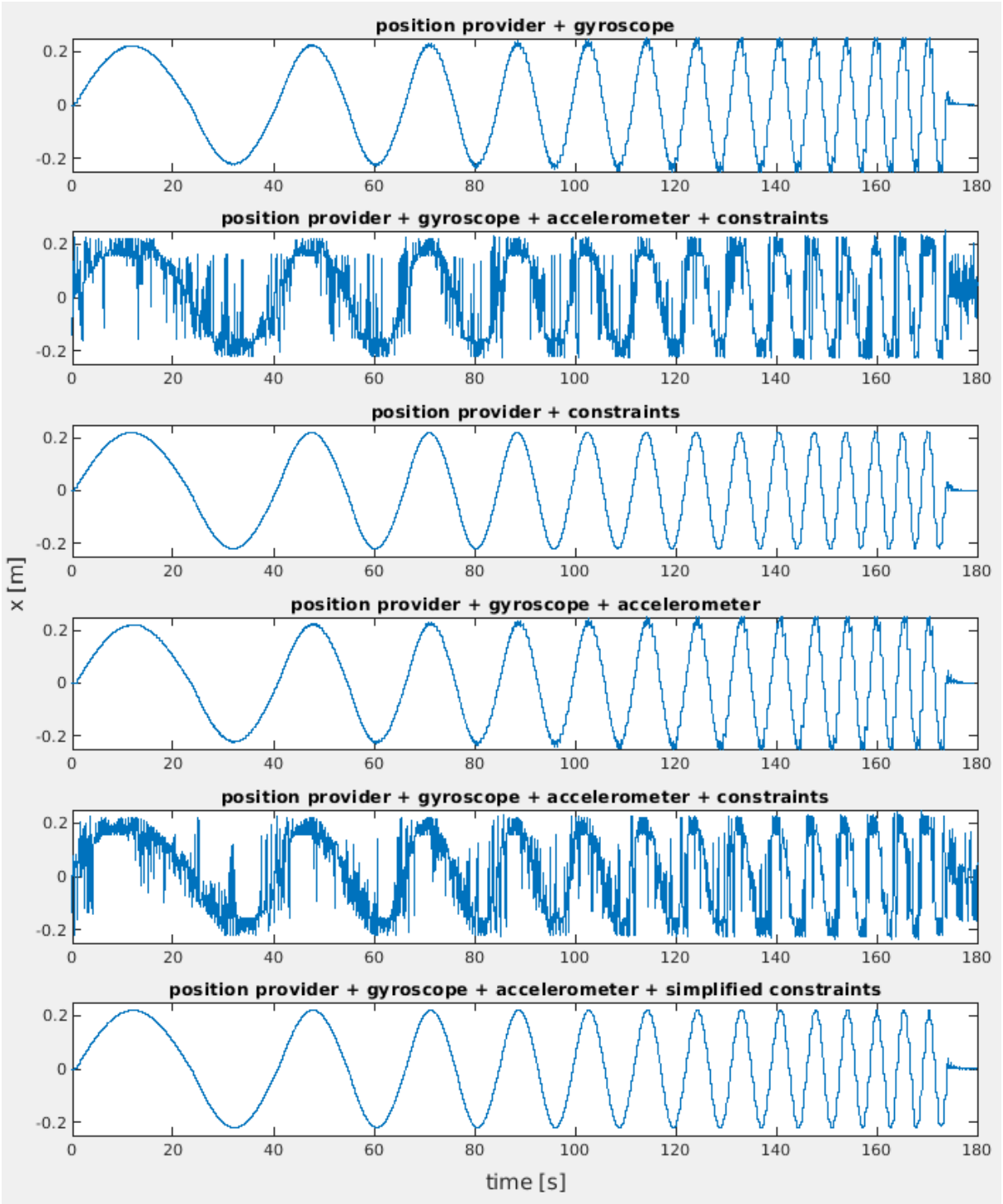

FIGURE 8.5 – Experiment 2: position estimation – x

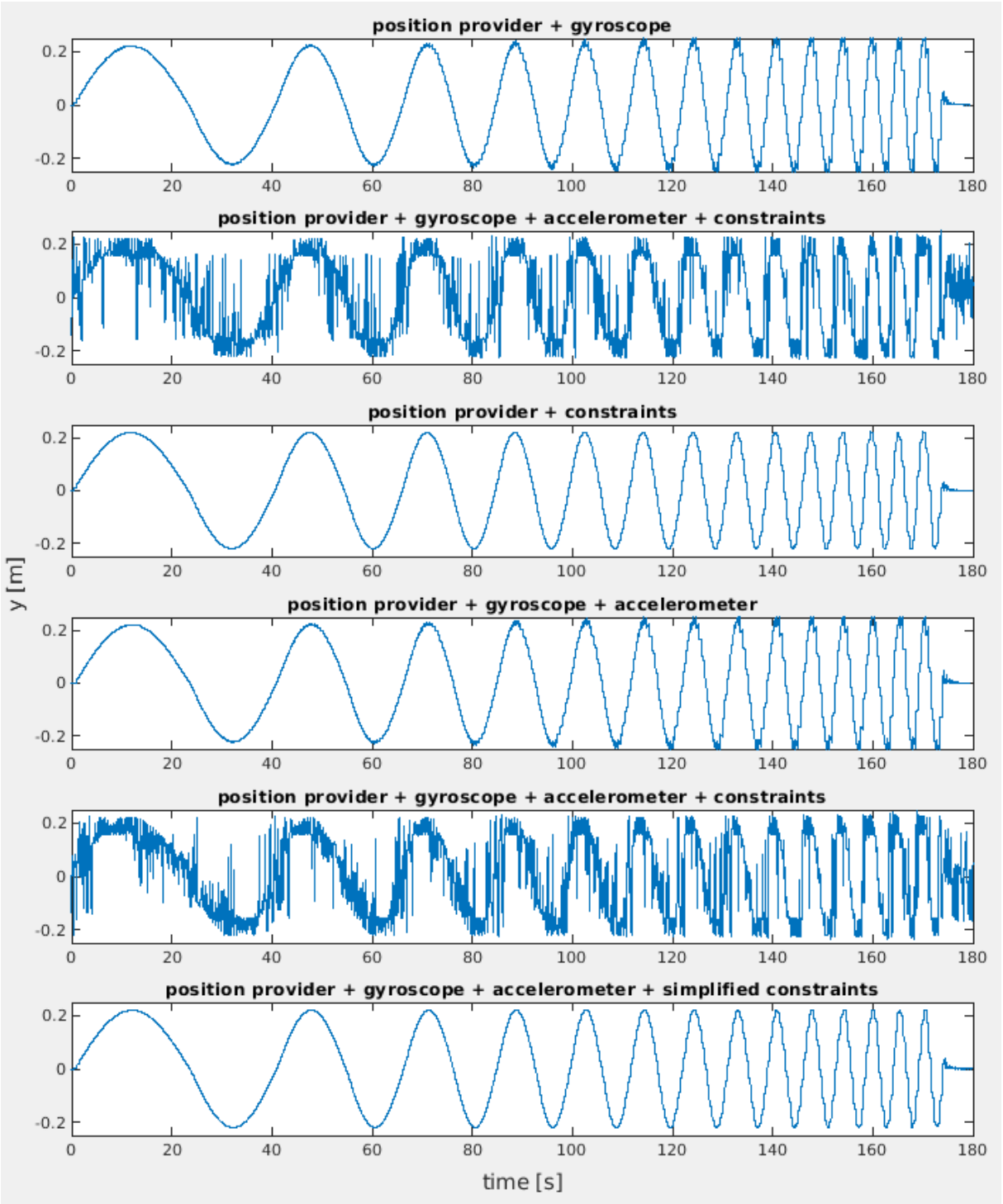

FIGURE 8.6 – Experiment 2: position estimation – y

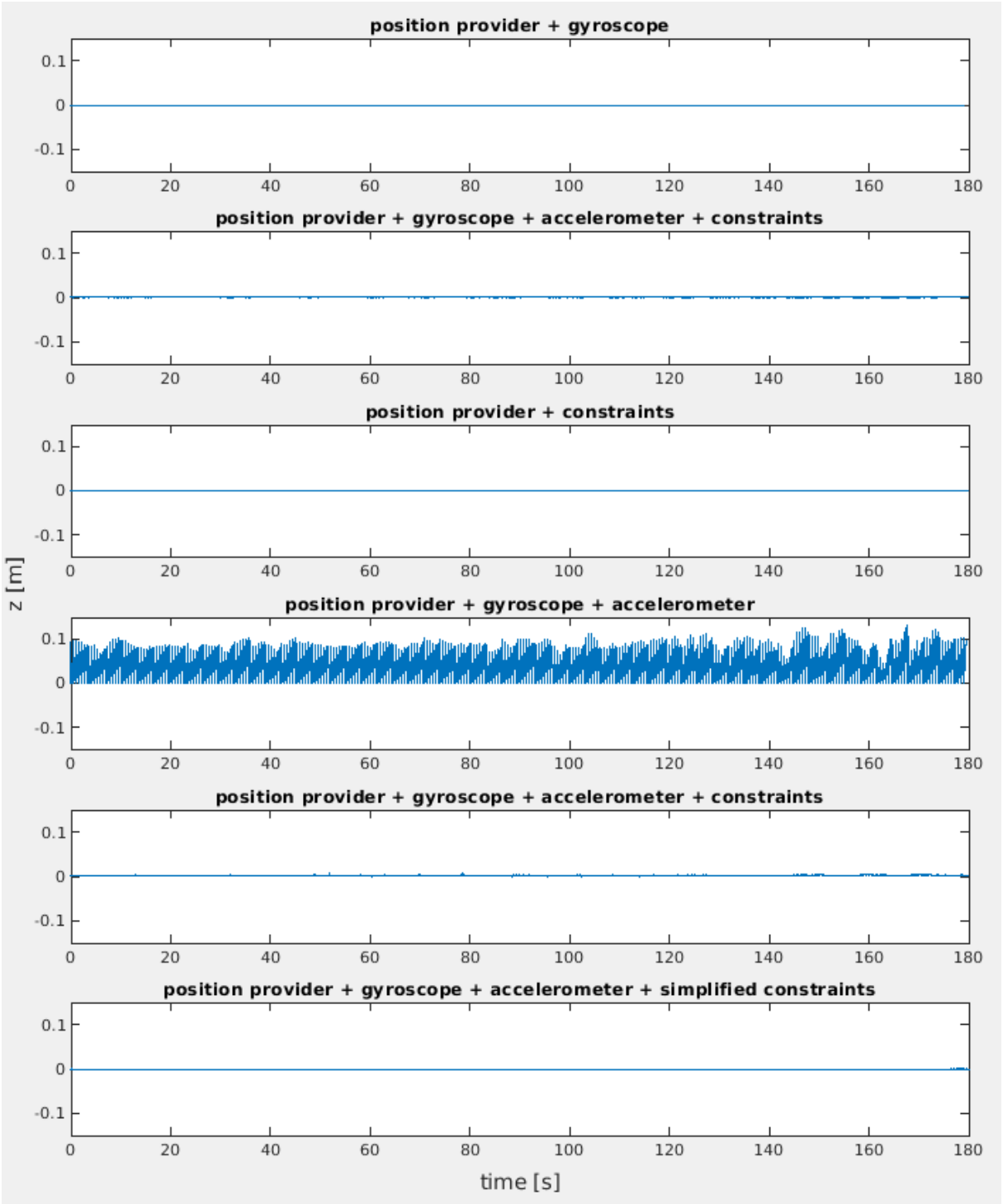

FIGURE 8.7 – Experiment 2: position estimation – z

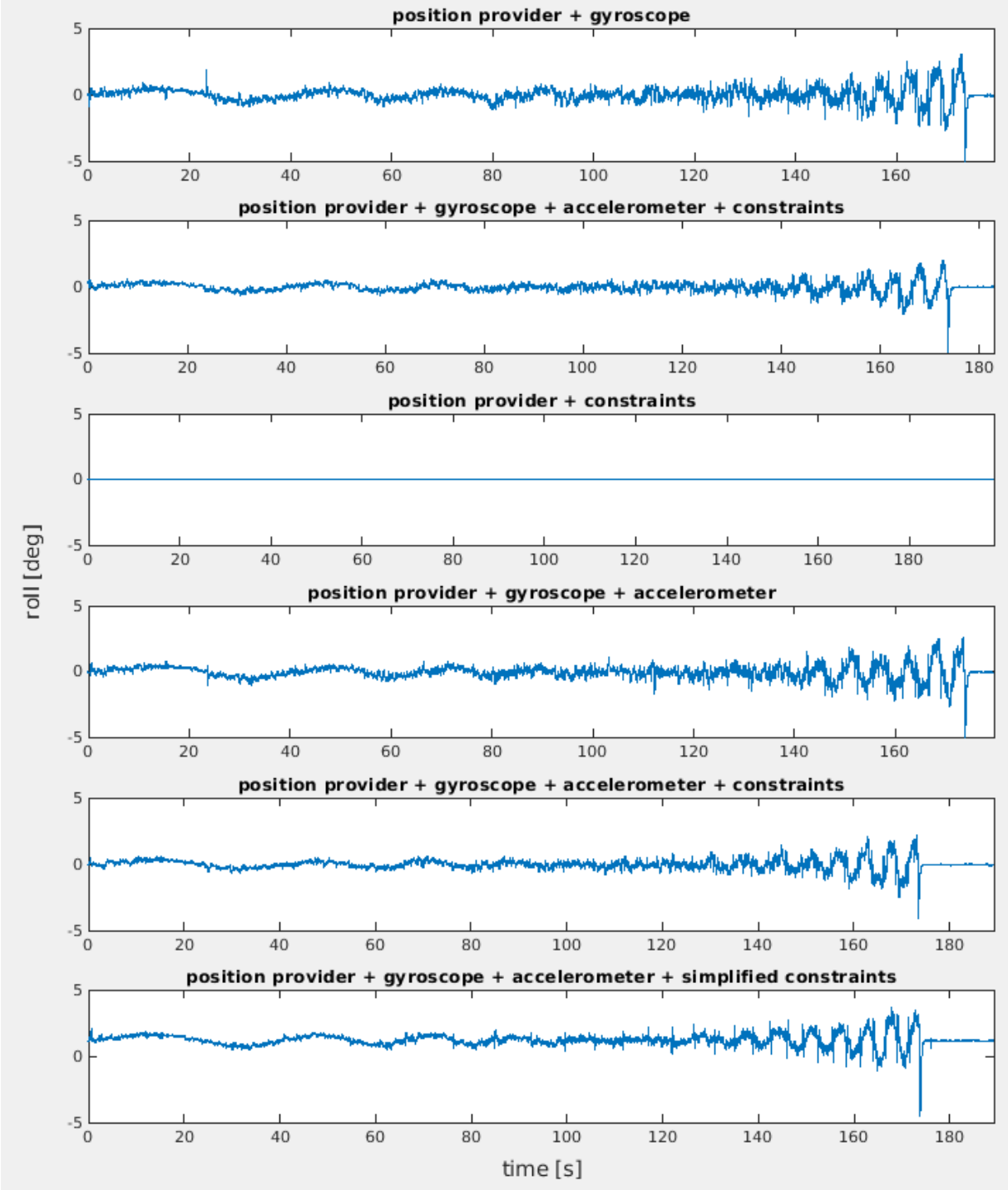

FIGURE 8.8 – Experiment 2: orientation estimation – roll

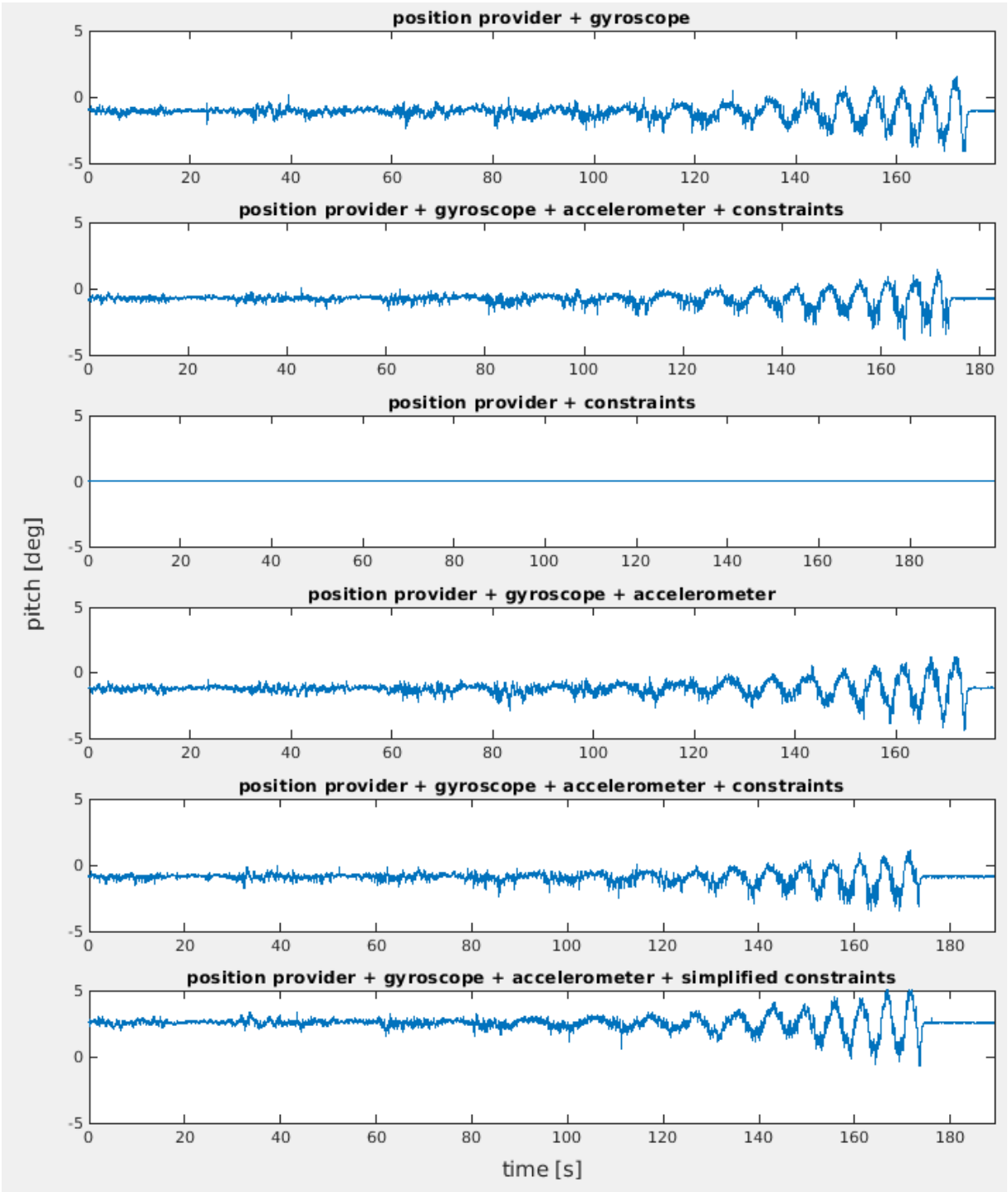

FIGURE 8.9 – Experiment 2: orientation estimation – pitch

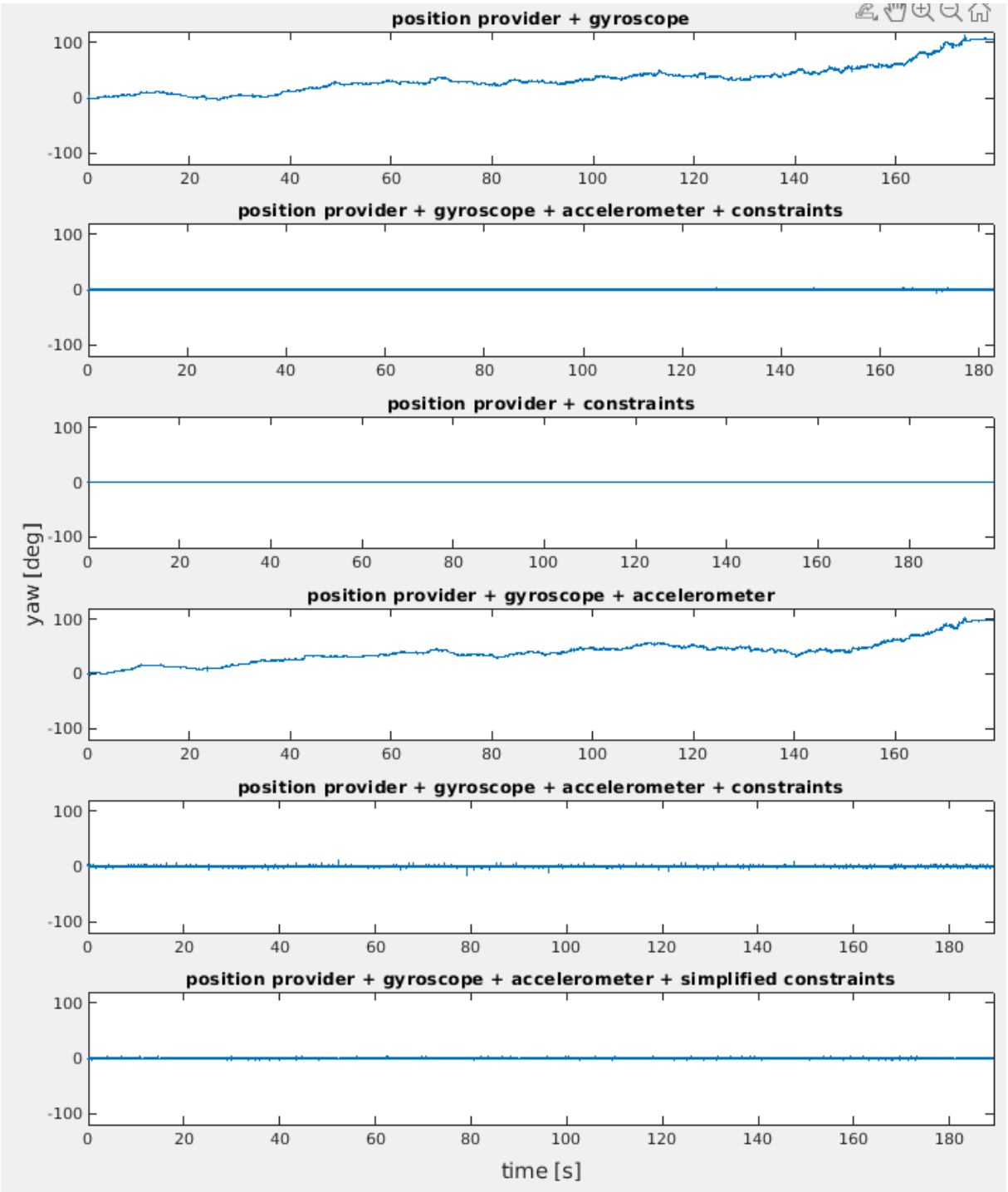

FIGURE 8.10 – Experiment 2: orientation estimation – yaw

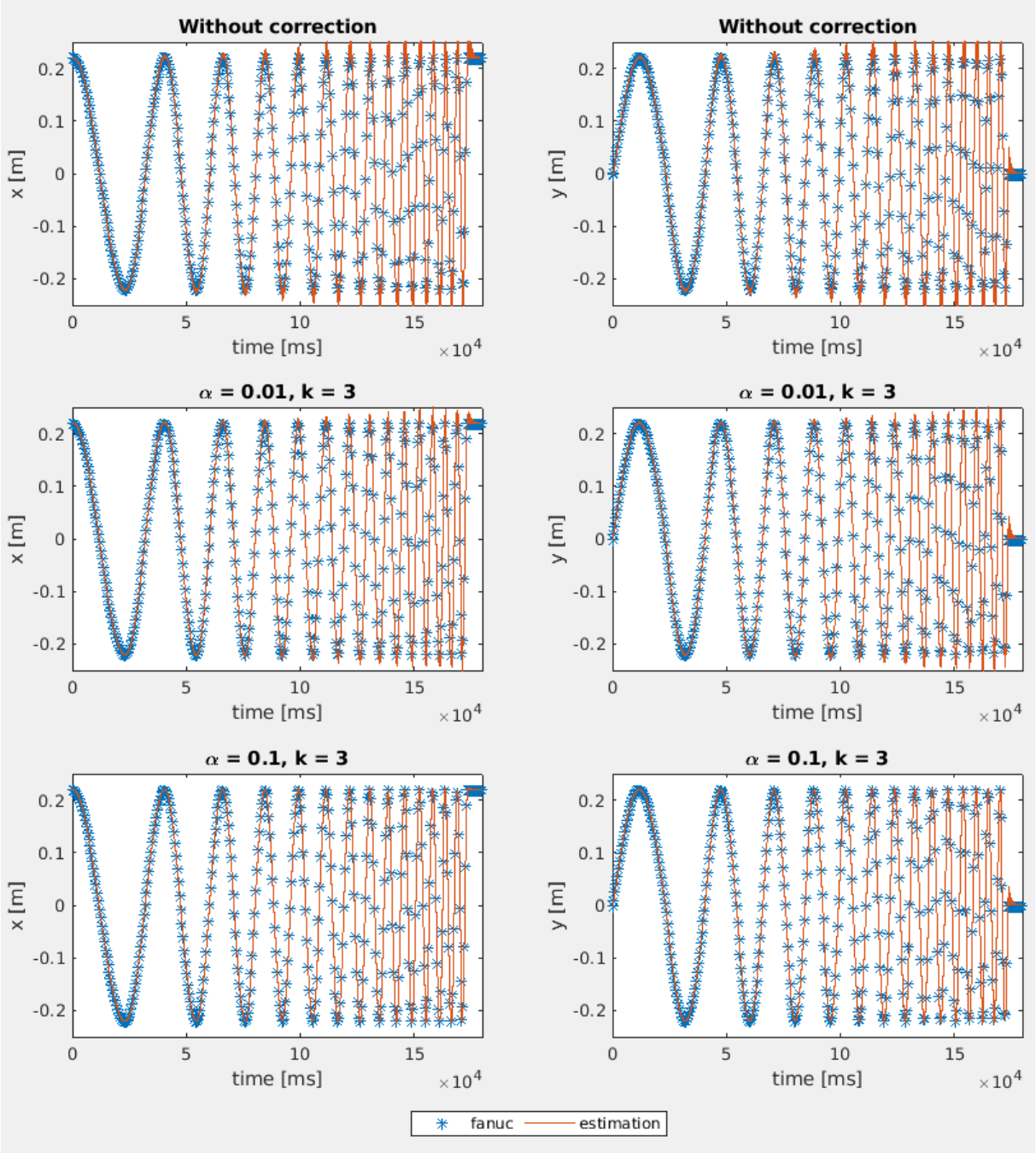

FIGURE 8.11 – Tuning: the influence of constraints' correction

9. Summary

This thesis represents a comprehensive exploration into position and orientation estimation methods. Within the scope of this work it was possible to gather extensive information about sensor fusion, data filtration and signal processing. The thesis introduces a novel approach to utilize a knowledge of the system structure.

Based on the knowledge gathered, a working system was successfully designed and deployed. The developed system is an application software that will be used for further research. For this reason, its operation has been further checked and its correctness has been validated.

The present work opens the topic of using solutions taken from aviation and other technical sciences in the field of robot positioning. In particular, it is important to highlight the pioneering use of constraints' correction in this field of science.

10. Bibliography

Literature

[1] Narkhede, P., Poddar, S., Walambe, R., Ghinea, G., and Kotecha, K., "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.

[2] Premerlani, W. and Bizard, P., "Direction cosine matrix imu: Theory", *DIY DRONE: USA*, Jan. 2009.

[3] Madgwick, S., "An efficient orientation filter for inertial and inertial/magnetic sensor arrays",

[4] Hasan, M. A. and Rahman, M. H., "Smart phone based sensor fusion by using madgwick filter for 3d indoor navigation", *Wireless Personal Communications*, vol. 113, no. 4, pp. 2499–2517, Jan. 2020, ISSN: 1572-834X. DOI: 10.1007/s11277-020-07338-7. [Online]. Available: https://doi.org/10.1007/s11277-020-07338-7.

[6] Kalman, R. E., "A new approach to linear filtering and prediction problems", *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[8] Zhang, S., Yu, S., Liu, C., Yuan, X., and Liu, S., "A dual-linear kalman filter for real-time orientation determination system using low-cost mems sensors", *Sensors*, vol. 16, no. 2, 2016, ISSN: 1424-8220. DOI: 10.3390/s16020264. [Online]. Available: https://www.mdpi.com/1424-8220/16/2/264.

[9] De Marina, H. G., Espinosa, F., and Santos, C., "Adaptive uav attitude estimation employing unscented kalman filter, foam and low-cost mems sensors", *Sensors*, vol. 12, no. 7, pp. 9566–9585, 2012, ISSN: 1424-8220. DOI: 10.3390/s120709566. [Online]. Available: https://www.mdpi.com/1424-8220/12/7/9566.

[10] Simon, D. and Chia, T. L., "Kalman filtering with state equality constraints", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 1, pp. 128–136, 2002. DOI: 10.1109/7.993234.

[11] Liu, H. and Pang, G., "Accelerometer for mobile robot positioning", *Industry Applications, IEEE Transactions on*, vol. 37, pp. 812–819, Jun. 2001. DOI: 10.1109/28.924763.

[12] A. Mahmoodi, A. S. and Menhaj, M. B., "Solution of forward kinematics in stewart platform using six rotary sensors on joints of three legs", *Advanced Robotics*, vol. 28, no. 1, pp. 27–37, 2014. DOI: 10.1080/01691864.2013.854455.

[13] Morar, A., Moldoveanu, A., Mocanu, I., *et al.*, "A comprehensive survey of indoor localization methods based on computer vision", *Sensors*, vol. 20, no. 9, p. 2641, 2020.

[14] Cooper, A. and Hegde, P., "An indoor positioning system facilitated by computer vision", pp. 1–5, 2016.

[15] Depenthal, C. and Schwendemann, J., "Igps - a new system for static and kinematic measurements", *9th Conference on Optical 3D Measurement Techniques*, Jan. 2009.

[16] Dogrusoz, H., Wszołek, G., Czop, P., and Słoniewski, J., "Vibration monitoring of CNC machinery using MEMS sensors", *Journal of Vibroengineering*, vol. 22, no. 3, pp. 735–750, May 2020. DOI: 10.21595/jve.2019.20788. [Online]. Available: https://doi.org/10.21595/jve.2019.20788.

[17] Ru, X., Gu, N., Shang, H., and Zhang, H., "Mems inertial sensor calibration technology: Current status and future trends", *Micromachines*, vol. 13, no. 6, 2022, ISSN: 2072-666X. DOI: 10.3390/mi13060879. [Online]. Available: https://www.mdpi.com/2072-666X/13/6/879.

[19] Valles, A. E., Alva, V. R., and Belokonov, I., "Calibration method of mems gyroscopes using a robot manipulator", *IEEE Aerospace and Electronic Systems Magazine*, vol. 38, no. 3, pp. 20–27, 2023. DOI: 10.1109/MAES.2022.3232728.

[20] "A typical filter design to improve the measured signals from mems accelerometer", *Measurement*, vol. 43, no. 10, pp. 1425–1430, 2010, ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2010.08.011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263224110001855.

[21] Lüke, H.-D., "The origins of the sampling theorem", *IEEE Commun. Mag.*, vol. 37, pp. 106–108, 1999. [Online]. Available: https://api.semanticscholar.org/CorpusID:15418427.

[23] Ionescu, V. M. and Enescu, F. M., "Investigating the performance of micropython and c on esp32 and stm32 microcontrollers", pp. 234–237, 2020. DOI: 10.1109/SIITME50350.2020.9292199.

[24] Passaro, V. M., Cuccovillo, A., Vaiani, L., De Carlo, M., and Campanella, C. E., "Gyroscope technology and applications: A review in the industrial perspective", *Sensors*, vol. 17, no. 10, p. 2284, 2017.

[25] Shiakolas, P., Conrad, K., and Yih, T., "On the accuracy, repeatability, and degree of influence of kinematics parameters for industrial robots", *International journal of modelling and simulation*, vol. 22, no. 4, pp. 245–254, 2002.

[26] Gajda, W. and Faliszewski, I., "Opracowanie wirtualnego środowiska do symulacji dynamiki lotu bezzałogowych statków powietrznych, bachelor thesis", 2023.

[28] Thakur, A. and Satish, M., "Clock synchronization in distributed systems", vol. 9, pp. 1929–1934, Mar. 2022.

[30] Głębocki, R., Żugaj, M., and Jacewicz, M., "Validation of the energy consumption model for a quadrotor using monte-carlo simulation", *Archive of Mechanical Engineering*, vol. vol. 70, no. No 1, pp. 151–178, 2023. DOI: 10.24425/ame.2022.144075. [Online]. Available: http://journals.pan.pl/Content/125283/PDF/AME_2023_144075.pdf.

[31] Simon, D., "Kalman filtering with state constraints: A survey of linear and nonlinear algorithms", *IET Control Theory & Applications*, vol. 4, no. 8, pp. 1303–1318, 2010.

[32] Simon, D. and Simon, D. L., "Kalman filtering with inequality constraints for turbofan engine health estimation", *IEE Proceedings-Control Theory and Applications*, vol. 153, no. 3, pp. 371–378, 2006.

[33] Wojtyra, M. and Frączek, J., "Joint reactions in rigid or flexible body mechanisms with redundant constraints", *Bull. Pol. Acad. Sci.-Tech. Sci.*, vol. 60, no. 3, pp. 617–626, 2012. DOI: 10.2478/v10175-012-0073-y.

Online

[7] "Extended kalman filter implementation", ThePoorEnginner. (), [Online]. Available: https://thepoorengineer.com/en/ekf-impl/ (visited on 10/16/2023).

[18] Hol, J. D. "Sensor fusion and calibration of inertial sensors, vision, ultra-wideband and gps". (2011), [Online]. Available: https://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-66184.

[22] “Floating point unit demonstration on stm32 microcontrollers”, STMElectronics. (), [Online]. Available: https://www.st.com/resource/en/application_note/an4044-floating-point-unit-demonstration-on-stm32-microcontrollers-stmicroelectronics.pdf (visited on 01/16/2024).

[27] “Ahrs: Attitude and heading reference systems”, Mario Garcia. (), [Online]. Available: https://ahrs.readthedocs.io/en/latest/filters/madgwick.html (visited on 04/07/2024).

[29] “Time synchronization in distributed systems”, Medium. (), [Online]. Available: https://medium.com/distributed-knowledge/time-synchronization-in-distributed-systems-a21808928bc8 (visited on 03/31/2024).

List of appendices

1. *server.zip* – source code of server,
2. *node_udp.zip* – source code of node_udp,
3. *client.zip* – source code of client application,
4. *constraints.zip* – source code of constraints library,
5. *bridge.zip* – source code of bridge,
6. *docker.zip* – docker containter's configuration files,
7. *scripts.zip* – MATLAB Scripts.

Acknowledgment

This work has been supported by the National Science Centre, Poland under grant no. 2022/45/B/ST8/00661.