External Measurement System for Robot Dynamics

Yana Sosnovskaya

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington
2017

Committee:
Samuel A. Burden
Blake Hannaford
Sawyer B. Fuller

Program Authorized to Offer Degree:
Electrical Engineering

© Copyright 2017

Yana Sosnovskaya

University of Washington

Abstract

External Measurement System for Robot Dynamics

Yana Sosnovskaya

Chair of the Supervisory Committee:
Assistant Professor Samuel Burden
Department of Electrical Engineering

Robots require testing to verify modeling assumptions, confirm performance characteristics, and quantify their limits. One requirement for formal robot testing is an external measurement system (EMS), that measures the robot's dynamics without relying on the robot's modeling assumptions or using any component of the robot's control system. Most robot dynamics measurement systems in literature are not external: they connect to the robot's instruments, and/or rely on its modeling assumptions.

In this work, I present a new, inertial sensor-based EMS, that consists of a set of inertial sensors, data acquisition hardware, and precisely specified calibration procedures. Static and dynamic calibration algorithms are tested for accelerometers, with static shown to be superior. Monte Carlo-based calibration algorithms are presented, to quantify calibration uncertainties, which can then be propagated to uncertainties in measurements. The EMS is tested on a Hopper robot, demonstrating its usefulness for dynamics measurement, and certain limitations for position measurement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my parents, Natalia and Dmitry Gavrilov, for support and love over all my life. Also, I'd like to thank my parents-in-law, Olga and Ray Hayes, for supporting my husband and me over the hardest immigration time.

I'd like to thank my husband, Gene, who always believes in me, no matter what, and always supports and loves me. Thank you also for discussing with me some technical problems over the course of this project, it helped a lot.

I'd like to thank Dr. Hannaford for his support, and for taking me on board as his Ph.D. student.

I'd like to thank Brenda Larson and Dr. Ostendorf for helping me with a lot of my questions and problems, and supporting me over the Spring 2017 quarter. It means a lot for me!

Lastly, I'd like to thank all of UW BioRobotics Lab for supporting me, and for coming to my defense! You are awesome!

Chapter 1
Introduction

This chapter briefly summarizes the problem addressed in this thesis, and summarizes the thesis's objectives.

1.1 Problem Statement

Robots require testing, among other reasons, to verify modeling assumptions. Formal testing requires two major components: a mechanical system to restrict the robot's degrees of freedom, and an external measurement system (EMS) to measure the robot's dynamics. Vast majority of examples of sensor systems for robots, in literature, are not external measurement systems (rather, they are onboard, which have very different requirements), and the few that are external, have serious issues. A major one is that there does not appear to be any published with a calibration procedure that would quantify uncertainties – which is necessary for a laboratory instrument. So, the problem to solve is this: There is a need for an external measurement system for robot dynamics that is cheap, accurate, has clear calibration protocols, and quantifies its measurement uncertainty.

Such external measurement system is useful for the following purposes:

- To track the dynamics of the robot;
- To calibrate the onboard sensors;
- To verify the modeling assumptions made in robot design and control system development;
- To test and compare the performance of different control systems development;
- To verify end-effector position in the forward kinematics and angles in the inverse kinematics.

1.2 OBJECTIVES

The following objectives are set for this thesis:

- To choose the components and the structure for an external measurement system for robot dynamics;
- To construct such system, and develop associated software;
- To develop and test a calibration procedure for this system; and
- To test the system, together with its calibration, on the jumping robot (Hopper).

1.3 SUMMARY OF CONTRIBUTIONS

The following contributions to the state of the art are delivered with this thesis:

- I designed, built, tested, and quantified the performance of an external measurement system, using inertial sensors. The system consists of inertial sensors, data acquisition hardware, associated software (LabVIEW virtual instruments), and precisely specified calibration procedures for the system's sensors.
- A family of Monte Carlo-based uncertainty quantification protocols, for sensor calibration, were developed.
- Several different calibration procedures for 3D accelerometers were experimentally compared, and the best one selected.
- Test results for the system, on a Hopper robot, are also provided.

1.4 A NOTE TO THE READER

The intended audience of this thesis are robotics engineers who anticipate having to formally, or informally, test mobile robots, on a budget.

Chapter 2 summarizes the state of the art in the field, and provides the necessarily background to understand the contributions made in subsequent chapters. The novel uncertainty-quantifying calibration algorithms are presented in chapter 3. Chapters 4 and 5 provide test results for various calibration procedures, for

accelerometers and gyroscopes. Chapter 6 describes the tests performed on the EMS, and draws conclusions about its performance. Overall conclusions and recommendations for future work are given in chapter 7.

Chapter 2
Background

One of the first objectives for this research was to select the general type of a robot dynamics measurement system. I selected an inertial sensor-based EMS, and this chapter provides the background information for this selection. Section 2.1 provides overall information on the fundamentals and typical applications of inertial sensors. When used in a laboratory setting, calibration and uncertainty quantification become more important than for a less precise use case, like onboard controls, therefore section 2.2 provides some general information on how inertial sensors can be calibrated. Section 2.3 lists different types of dynamics measurement systems, and discusses why inertial sensors were chosen in this project. Lastly, section 2.4 discusses the basics of uncertainty propagation, necessary for testing the EMS on a physical robot.

2.1 Inertial Sensor Fundamentals

Accelerometer is a device that measures proper acceleration. Proper acceleration is not the same as the change of velocity in time; it is instead acceleration relative to freefall [1]. Conceptually, one can think of a single axis of an accelerometer as a mass on a spring. When the accelerometer experiences an acceleration, then by Newton's second law, the mass is displaced to the point that the spring is accelerating the mass at the same rate. The displacement is then measured to compute the acceleration. Modern accelerometers are mostly microelectromechanical systems (MEMS) [2].

Gyroscope is a device that measures angular rate (rate of rotation about an axis) [1]. Gyroscopes are also mostly microelectromechanical systems (MEMS), and their working principle is based on the Foucault pendulum. Specifically: the angular rate induces a Coriolis force on a resonating mass-spring system inside

the accelerometer, this force displaces it, and the displacement is capacitively measured to compute the angular rate [2].

Both accelerometers and gyroscopes can be analog or digital, differing by the type of signal they produce. It is important to note, that fundamentally, both proper acceleration and angular rate are analog quantities, so only the output signal format, and not the internal measurement process, define whether a sensor is analog or digital.

Analog signal is a signal that continuous in time [3]. It is often linearly proportional to the physical quantity being measured. Their main advantages are [4]:

1. Analog sensors do not require programming for data acquisition.
2. It is easier to deal with multi-sensor systems that use analog sensors, because there is no need to synchronize multiple devices like with digital protocols.
3. Analog sensors give more precise readings than digital sensors. This is more of a limitation of digital sensors though, that rises from the fact that digital sensors have an onboard analog-to-digital (ADC) converter, which introduces a discretization error.
4. Analog sensors do not have the same limitations in wire length as digital protocols. Digital sensors' protocols tend to be short-range, typically for communication on the same printed circuit board (PCB). Therefore, if one desires to process data from a sensor elsewhere, for an analog sensor, one can simply pull a wire, while for a digital sensor additional transmission equipment is necessary.

They do have disadvantages as well:

1. Analog sensors do not typically have built-in options, like low-pass filtering to reduce noise, temperature compensation, etc.
2. Analog sensors each require a wire to transmit data, while synchronized digital sensors can transport data over a single synchronized bus.
3. Digital sensors' protocols are significantly more resistant to noise; this is a general advantage of digital technology [3].

Digital signals are discrete in time, and communicate data using one of several short-range communication protocols. The most common ones are I2C, SPI and 1-wire digital protocol. Digital protocols' main advantage is that the signal is generally less noisy, and less subject to noise. Their main disadvantage is that

the data is less precise due to quantization, due to the use of analog to digital converter (ADC) between the measuring device and the output pins.

As discussed in section 2.3, the system proposed in this text uses analog inertial sensors, so the rest of inertial sensor information below is primarily on analog sensors. They were chosen, primarily, due to their superior accuracy, achieved by relying on a benchtop laboratory ADC converter, as opposed to the lower-resolution on-chip that most inertial digital sensors have.

To convert between the signal output by a sensor, and the physical values being measured, a *"transducer function"* is used. The following quote defines the term *"transducer"* [5]:

"The transducer is a device or element (concept of physics) that transforms one type of energy into another, e.g. wind energy into electrical energy or electrical energy into acoustic energy. The transducer acts between both electrical physical quantities and non-electrical physical quantities, such as sound or light, symbolizing a mediator and a converter. In most cases, it consists of converting electrical energy into a mechanical displacement or converting a non-electrical physical quantity such as sound, temperature, pressure, velocity, or light, into electrical quantity."

A transducer function, in context of this work, is therefore simply the mathematical function that relates the output signal (*"electrical quantity"*), and the measured non-electrical physical quantity, of the device. It can also be referred to as a "sensor transfer function" [6]. For 1-axis analog accelerometers and gyroscopes, it typically has the following form:

$$p(v) = s(v - o), \tag{2.1}$$

in which $p(v)$ is the physical value (e.g., acceleration), v is the voltage output by the sensor, o is the voltage offset, and s is the sensor inverse sensitivity.

For 3-axis analog accelerometers and gyroscopes, the transducer function typically has the following form [7]:

$$\vec{\boldsymbol{p}}\left(\vec{\boldsymbol{v}}\right) = \mathbf{A}\left(\vec{\boldsymbol{v}} - \vec{\boldsymbol{o}}\right), \tag{2.2}$$

in which $\vec{\boldsymbol{p}}$, $\vec{\boldsymbol{v}}$ and $\vec{\boldsymbol{o}}$ are the 3-axis vector versions of the corresponding scalar variables from Eq. (2.1), and $\mathbf{A}$ is the sensor calibration matrix.

Analog sensor datasheets generally report the sensor sensitivity k and the offset voltage o, each as a range with a typical value [8]. For multiaxial sensors, the datasheets tend to report a single sensitivity and offset, that are the "typical values" for all 3 axes. k and s are reciprocals of each other:

$$k = \frac{1}{s}. \tag{2.3}$$

Internally, the microelectromechanical axes, also known as *"(internal) sensor axes,"* of the 3-axis MEMS sensor may not be precisely orthogonal to each other, and may also not be exactly aligned to the corresponding *"physical axes."* Physical axes, also known as *"body axes"* [7], are the axes that the sensor is assumed to be measuring along, and are generally marked on the sensor's body. They may also be referred to as the *"mounting axes"* [9]. In Eq. (2.2) above, $\vec{\boldsymbol{p}}$ is assumed to lie on the mutually orthogonal body axes. Lastly, the sensor may not necessarily be equally sensitive on all 3 axes.

The full sensor calibration matrix $\mathbf{A}$ is a 3×3 matrix. It has all 9 possible degrees of freedom, and accounts for the axis sensitivities, their mutual nonorthogonalities, and their misalignment relative to the body axes. The 6 degrees of freedom corresponding to the 3 nonorthogonality and the 3 misalignment errors are, together, known as the "misalignment error angles," or simply as the misalignment angles. For practical sensors, misalignment angles are small (under 4°), and therefore can be linearly approximated. Using this approximation, the calibration matrix can be decomposed as follows [7]:

$$\mathbf{A} = \mathbf{ST} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \begin{bmatrix} 1 & -\alpha_{yz} & \alpha_{zy} \\ \alpha_{xz} & 1 & -\alpha_{zx} \\ -\alpha_{xy} & \alpha_{yx} & 1 \end{bmatrix}, \tag{2.4}$$

in which $\mathbf{S}$ is the inverse sensitivity (or "scaling") matrix, and $\mathbf{T}$ is the misalignment error matrix. The angles in $\mathbf{T}$ are illustrated in Fig. 2.1 below. In it, $\left\{x^B, y^B, z^B\right\}$ are the body axes, that the sensor is treated as measuring, and $\left\{x^S, y^S, z^S\right\}$ are the sensor axes, that the sensor is actually measuring along with inverse sensitivities $\left\{s_x, s_y, s_z\right\}$.

A "datasheet 3-axis sensor" (i.e., one that behaves exactly according to the datasheet stated values) has the following calibration parameters:

$$\mathbf{S} = \begin{bmatrix} k^{-1} & 0 & 0 \\ 0 & k^{-1} & 0 \\ 0 & 0 & k^{-1} \end{bmatrix}, \tag{2.5}$$

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.6}$$

$$\vec{o} = \begin{bmatrix} o \\ o \\ o \end{bmatrix}, \tag{2.7}$$

which effectively means “the misalignments are zero, and the axes are equally sensitive.”

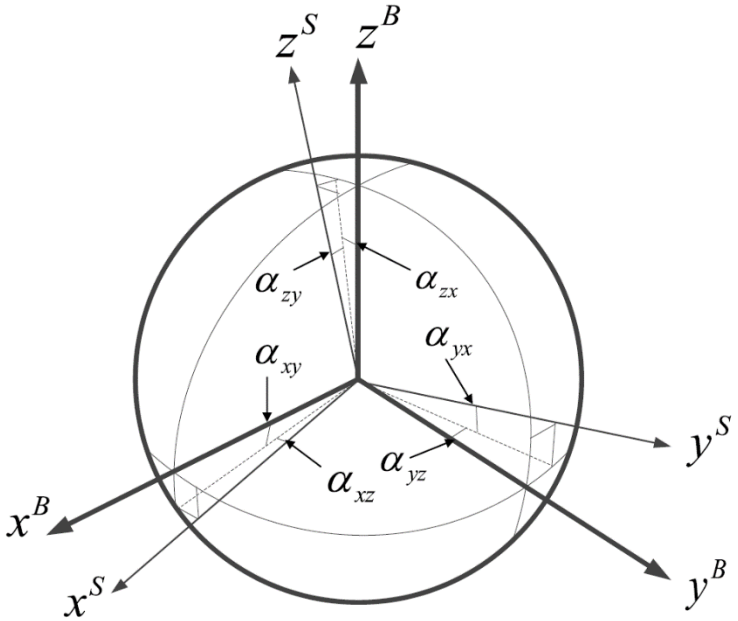

Fig. 2.1. Sensor misalignment angle definitions
(from Ref. [10])

Because of manufacturing variability, and temperature dependence, sensor datasheets normally give ranges for the calibration parameters. Inertial sensors therefore require calibration, which yields the transducer function parameters for the specific sensor. Section 2.2 discusses sensor calibration in more detail.

2.2 Inertial Sensor Calibration

Calibration is the process of comparison between measured sensors’ output values and corresponding standard of known accurate, reference values. Calibration of accelerometers and gyroscopes is necessary to get their offsets, sensitivities and misalignment angles. These parameters are unique for each sensor, due to the tolerances in manufacturing processes of MEMS devices. More generally, calibration parameters are unique to each sensor [6].

The full calibration process consists of two stages: the experimental stage, and the computational stage. The first stage is specific to sensor type, and is discussed in this section. Its purpose is to subject the sensor to a set of well-known, reference physical values, and to measure the signals (i.e., voltages) the sensor outputs for each one. The second stage is the computation of the sensor's transducer function based on the data collected in stage 1. The second stage is discussed in chapter 3. The first stage is sensor type-specific, and is discussed below.

There exist multiple ways to calibrate accelerometers, generally divided into:

1. Centrifugal (or, turntable) calibration.

   A point rotating about an axis with a given angular velocity experiences a constant centripetal acceleration. This fact is used as the basis of turntable calibration: a centrifuge, or a turntable, is used, and the accelerometer is placed on it. It is placed in such a way that the axis being calibrated that calibrated is pointed to the axis of rotation, which then subjects it to a known acceleration. Reference accelerations can be varied by changing radii or changing the velocity of centrifuge. [11]

   Because accelerometers measure proper acceleration, orientation of the axes that do not point along the radius is still important: the vertical component of each axis will experience the gravitational component, same as in one-step static calibration (method 3 below).

   This method can work with both 1-axis and 3-axis accelerometers.

2. Shaking table calibration.

   Shaking table calibration is intended to determine errors in calibration factor due to frequency effects. A shaking table has a pure sinusoidal motion with a minimum harmonic distortion over a wide frequency band. This method is often used in conjunction with others, or for accelerometers later used to identify resonant frequencies of machine parts, which are expected to be exposed to a wide spectrum of mechanical frequencies. [11] [12]

   Depending on the shaking table configuration, this method can work with both 1-axis and 3-axis accelerometers. In practice, accelerometers used to identify resonant frequencies of machine parts are typically multiaxial.

3. One-step static calibration.

   Accelerometers measure proper acceleration, and therefore are able to sense gravitational acceleration g, as $1g$ pointed vertically upward. They sense it as pointed upward because relative to freefall, the accelerometer is being vertically accelerated up, at acceleration $1g$. [7]

The one-step static calibration method consists of orienting the accelerometer in several known, static positions, such that the projection of the gravitational acceleration onto each of the accelerometer's body axes is known. This method does not require knowing the sensor axes' orientation at any point, but does require knowing the body axes' orientation. A proposed set of 12 orientations for the body axes, to use with this method, is given in Ref. [7].

The primary disadvantage of this method is that all reference accelerations the accelerometer experiences have magnitude $1g$. Such restricted reference value ranges can bias the resulting calibration parameters, to make them more accurate near constant velocities, and less accurate when the device is rapidly accelerating. In robotics, accelerations experience by parts exceed $1g$, particularly on impact. Another disadvantage is that precisely orienting the sensor is difficult; however, as will be shown in chapter 4, this disadvantage is largely cancelled out by the fact that orientation errors are independent of each other.

Theoretically, this method can work with both 1-axis and 3-axis accelerometers (as long as the orientations are chosen appropriately), but in practice, it appears to primarily be used for 3-axis accelerometers (e.g., Ref. [7]).

4. Two-step static calibration.

   The two-step static calibration method is proposed by Bonnet, et al. [9]. In the first step, the accelerometer is placed in at least 9 static orientations, that are not necessarily known, other than the fact that all orientations are distinct. The measured voltage vectors are used to evaluate the offset vector $\vec{\boldsymbol{o}}$, and the upper triangular matrix $\mathbf{B}$. $\mathbf{B}$ accounts for 6 of the 9 degrees of freedom in $\mathbf{A}$: the sensitivities and the orthogonalization, but not the alignment from the orthogonalized sensor frame to the body frame. Together, $\mathbf{B}$ and $\vec{\boldsymbol{o}}$ are not yet usable (unless only the acceleration magnitude is of interest): they do not produce the acceleration vector in the body frame. Instead, $\mathbf{B}\left(\vec{\boldsymbol{v}} - \vec{\boldsymbol{o}}\right)$ yields the acceleration vector in the orthogonalized sensor frame.

   In the second step, one of the accelerometer's body axes is fixed (not parallel or antiparallel to gravity), and the sensor is rotated about this fixed axis. During this rotation, it is stopped several times (at least 3 per axis), and the voltage vector measured. This procedure is repeated for every body axis. The resulting data is used to evaluate the rotation matrix $\mathbf{R}^{sb}$, which rotates the measured acceleration vector from the orthogonalized sensor to the body frame. This step relies on the fact that when a body axis is

held fixed, the projection of gravity on this axis is constant, independently of the sensor's orientation about this axis. After the two steps, $\mathbf{A}$ is computed through:

$$\mathbf{A} = \mathbf{R}^{sb}\mathbf{B}. \tag{2.8}$$

The primary advantage of this method is that it does not rely on precisely knowing orientations – it only needs the orientation of a single axis, at a time, fixed, during the second step. It has the same disadvantage as the one-step static calibration method above: only a limited range of accelerations ($\pm 1g$) is covered. Its other disadvantage is that during the second step, the calibrating engineer is required to hold one axis fixed, and, as will be shown in chapter 4, the resulting errors can be significant.

This method only works with 3-axis accelerometers (and also 3-axis magnetometers, which are not used in this work).

Accelerometers are typically calibrated using the method that best resembles their intended application. Gyroscopes do not have such diversity of calibration methods, because they measure angular rates. Effectively, gyroscope calibration is limited to method 1 above: turntable calibration method [13].

It is important to note, that when a reference acceleration is imposed on an accelerometer – whether a controlled one via a turntable, or simply the gravitational acceleration in a static pose – its vector can only be known (within an uncertainty) in the body frame, but not in the sensor frame. This is due to the fact, that without the 3 alignment angles (the 3 degrees of freedom in $\mathbf{R}^{sb}$), one cannot know the orientation of the sensor frame, and therefore cannot evaluate the orientation of a reference acceleration in it.

However, one can know (within an uncertainty) the length and orientation of the reference acceleration vector in the body frame. This is due to the fact, that the body frame – which, again, may be potentially different from the sensor frame – is effectively chosen by the engineer using the sensor, typically to correspond with the edges of the sensor's rectangular parallelepiped body. Therefore, the fact that one does not know the orientation of the reference acceleration vector in the sensor frame does not introduce an unaccountable error into methods 1 and 3 above: in both of those methods, the reference accelerations are those in the body frame, and not the sensor frame.

Method 2 above is not used in this work. Methods 1 and 3 above are used, and each can utilize the one-step calibration computational stage, which relies on having a set of voltage measurements $\{\vec{v}\}$ and corresponding known reference accelerations, with known orientations, $\{\vec{p}\}$. Method 4 above is also used in this work,

and it requires a special calibration computational stage, that would not work for the other methods. The one-step (for methods 1 and 3) and two-step (for method 4) computational methods, in both deterministic and uncertainty-quantifying versions, developed for this work, are described in chapter 3.

There are instances in literature, in which Kalman filter is mentioned as an online or real-time calibration method, such as in Refs. [14] [15]. However, this terminology is incorrect: by definition referenced at the beginning of this section, a calibration procedure is one that involves the exposure of a sensor to a set of known reference physical values in order to optimize its transducer function. Kalman filter, on the other hand, is a recursive filter that helps with state estimation for moving objects [16]. It relies on the system's dynamic model to correct the state estimate obtained from the sensors' measurements – thus effectively increasing the sensors' usefulness, but not their accuracy. The calibration procedure, on the other hand, substantially increases the sensors' accuracy, relative to using datasheet sensitivities and offsets; see, for example, chapters 4 and 5 of this thesis.

2.3 Robot Dynamics Measurement Systems

An objective of this thesis is to develop and test an external robot dynamics measurement system. In the following subsections, the difference between onboard measurement systems and EMSs is explained, multiple types of EMSs are reviewed, and justifications for selecting an inertial sensor-based external robot dynamics measurement system are provided.

2.3.1 *Onboard Sensors*

Onboard measurement system is a sensor system that is permanently mounted on a controllable object, and measures its dynamic parameters in real time. This measured data is then fed to the control system, which processes the measured data, estimates the system's state based on its measurements, and uses this estimated state in its control algorithms.

The most common onboard measurement systems are Inertial Measurement Units (IMUs), which consist of accelerometers, gyroscopes and magnetometers, sometimes also a thermistor is included; motor encoders, onboard cameras, and Global Positioning System (GPS) units.

All the above sensors' measurements can be used for state estimation in the control system of a robot. Typically, Kalman filters are first applied to correct the robot's position and velocity using the robot's dynamics equations. Kalman filter itself has three different types: the Kalman filter, the Extended Kalman filter (EKF) and the Unscented Kalman filter (UKF). The Kalman filter has serious limitations, because it works only with linear robot dynamics equations, but most modern robots have nonlinear dynamics. Extended Kalman filter relies on Taylor expansion-based linearization of the robot's nonlinear dynamics. The accuracy of the linear approximation applied by the EKF depends on two factors: the degree of uncertainty and the degree of local nonlinearity of the functions that are being approximated**.** The Unscented Kalman filter is used for more complicated nonlinear dynamics, where it performs a stochastic linearization using a weighted statistical linear regression process. UKF's main disadvantage is computational time: it often requires pre-processing and post-processing some procedures, and so it can be nearly impossible to run fully UKF in real-time for complicated nonlinear dynamics. [16]

The differences between onboard sensors and external measurement systems are the following:

1. Onboard sensors give measurements to the control system of the robot to correct its states by using any of above described Kalman filters and robot dynamic equations [17]. External measurement system is designed for measuring the robot's dynamics separately from the control system, without relying on the robot's dynamics. It can therefore be used to verify or reject the modeling assumptions that underlie the robot's dynamic model and control system, which onboard sensors' measurements cannot be: they are, as described above, reliant on the robot's model.
2. Onboard sensors can be a part of the control PCB (mainboard, microcontroller, FPGA, processor unit, etc.), or built into another part of the robot. This seriously diminishes the options for calibrating the onboard sensors. In Ref. [13], the entire robot was placed on a Stewart platform to calibrate the robot's IMU. Obviously, such method cannot possibly work for bigger robots, like humanoids, unless massive, expensive calibration platforms are used.

   Another example could be the IMU calibration process on robotic manipulators, to calibrate the positioning of end effector. Such position-based calibration procedure can be used only for manipulators [18].

   On the other hand, much more precise calibration is possible for an external measurement system's sensors: they are not permanently attached to any robot, and therefore can be calibrated separately, and placed back on the robot post-calibration.

3. Because onboard sensors feed their signals directly to the control board (mainboard, microcontroller, FPGA, processor unit, etc.) to control the robot, their computational, power and memory resources are restricted, due to space and weight limitation. External measurement systems do not have such restrictions, because to process the data from external measurement systems, an external computer, external DAQs, power supplies or other powerful, high-memory and high-speed benchtop equipment, can be used.

   This also illustrates why an external measurement system cannot be thought of as a replacement for onboard sensors: onboard sensors are a component of the robot, working "in the field," while an external system is a laboratory instrument, intended for use in a laboratory setting, for robot testing.

2.3.2 *External Measurement Systems*

The following definition is given in literature for external robot dynamics measurement systems [19]:

"External measurement system is designed to evaluate and calibrate the performance of a robot by means of external measuring instruments."

Below, several potential options for robot dynamics EMSs are discussed.

1. High-speed camera or motion tracking camera.

   A high-speed camera is an optical device that track the reflective markers' motion in time, and can either record videos, or dynamically calculate the time-dependent positions of the markers. High-speed camera requires some calibration procedure to calibrate its measurements depending of the location of the camera and the tracked object. To fully track a 3D robot's motion, multiple cameras around the robot are needed, followed by a sophisticated algorithm to fuse the data from all the cameras. Moreover, the high-speed camera suffers from occlusions and false data detection based on reflections from metal (reflective) parts of the robot. [20]
2. Laser-based measurement systems.

   A notable example of a laser-based measurement system is described in Ref. [21]. It consists of a set of infrared LEDs and a camera that is used for calibration of kinematics of the manipulator. The measurement technique is based on a laser interferometry tracker (Leica LT500 Laser Tracker). This system is not able to provide orientation data; the proposed method was implemented for estimating position

errors only. Like a high-speed camera with reflectors, such system also suffers from occlusions and types of materials that can be used in the experiments [21].

3. Inertial sensor-based measurement systems.

   Inertial sensor-based measurement systems mostly are used for placing accelerometers on a human body to track its movement [22]. There appears to be only a limited published experience with inertial sensor-based EMSs in robotics. Two examples include: 1) a system with inertial sensors are used for getting precise measurements of joint angles on pin joints or positions of the links [20] [23], and 2) an accelerometer-based system for tracking manipulator positioning with high precision [24]. These systems are clearly external, but they are not intended for dynamics tracking. Also, despite being laboratory measurement systems, they do not quantify their uncertainties explicitly, or provide a technique to do so. Therefore, these inertial sensor-based EMSs cannot be considered to meet the requirements of the system designed in this work.

   Inertial sensors measurement systems do not suffer from occlusions or material dependencies, and are significantly less expensive than the high-speed cameras. They can also easily track 3D motion, not only translational but also rotational. As is discussed in sections 2.1 and 2.2 above, inertial sensors require calibrations to get the offsets, sensitivities and misalignment angles: datasheet values are listed, but are not accurate enough.

4. Ultrasound measurement system.

   Ultrasound measurement systems can be used for position correction of the manipulator. Such method works in the following way: the ultrasound 3D sensor is placed on the end effector of the manipulator; then the measurements are converting into translations and rotations of the end effector by multiplying a matrix comprised of the offset measurement by a previously prepared conversion matrix; next the coordinates are corrected by applying such rotations and translations. [25]

   Outside of robotics, a position- and orientation-tracking ultrasound measurement system has been used in surgery [26]. In this system, ultrasound markers (i.e., transmitters) were placed on surgical instruments, and ultrasound microphones outside the patient. Knowing the dimensions of the usually rigid instruments, it was possible to calculate the positions and orientations of the instruments inside the patient, by identifying the markers' positions.

   Wireless ultrasound sensors' data suffers from temperature, humidity and air pressure influence. It is also obviously susceptible to acoustic noise. Ultrasound sensors are generally less accurate compared

to optical sensors, however, ultrasound sensors are not affected by object reflectivity. Working terrain-traversing robot's motors can influence ultrasound results as well, as they emit noise over a wide spectrum of frequencies, and tend to be louder than the precise, low-speed motors used in robot manipulators.

There also exist combinations of the systems described above, for example: inertial sensors and force sensor [27], camera and laser-tracking systems, camera and ultrasound systems, camera and inertial sensors [28]. Based on the above comparison of EMSs for robot dynamics, for the purposes of this work, I have selected to develop an inertial sensor-based EMS. The motivation for this is the following:

- While they require an external DAQ, power supply and control computer, these are reasonable requirements for any dedicated laboratory system. The sensors themselves, however, are significantly cheaper than the hardware required for the other options listed above.
- Terrain-traversing robots generally move in 3 dimensions, and the EMS has to work with this. A single high-speed camera cannot effectively track 3-dimensional motion, and multiple cameras are very costly, and difficult to fuse the signals from. Occlusion is another considerable problem for robots with constantly moving parts, particularly if one desires to track limbs, and not the main body of the terrain-traversing robot. This rules out laser- and camera-based systems.
- Ultrasound sensor systems, as described in the above list, are not well-suited to terrain-traversing robot applications, due to the noise that is likely to be produced by the motors. Furthermore, many of these robots are made of metal, which can affect the ultrasound sensors' performance.

A set of inertial sensors does not become a proper external measurement system until it is calibrated, and software and hardware are configured together with it. There appear to be no uncertainty-quantifying calibration algorithms for inertial sensors present in literature; the closest work is Ref. [19], in which Monte Carlo-based uncertainty propagation was used to deduce the Denavit-Hartenberg (DH) parameters for a robot manipulator. Other related works include Ref. [29] in which Monte Carlo is used to propagate sensor mounting errors to the outputs of an inertial navigation system, and Ref. [30], in which Monte Carlo is again used to judge the numeric sensitivity of several different computational algorithms for accelerometer calibration.

None of the three publications above actually quantify the sensor calibration uncertainties. A calibration that quantifies a measurement system's uncertainties is much more preferable to one that does not. Therefore, part of this work's results are the development and testing of new, more advanced calibration algorithms, that quantify transducer functions' uncertainties.

Section 2.2 discussed experimental setups that can be used for inertial sensor calibration, and chapter 3 presents the computational algorithms, developed as part of this work, that provide uncertainty quantification together with best fit calibration parameters for inertial sensors. The uncertainties in calibration parameters can be propagated to the actual physical parameters being measured; the fundamentals of the mathematics for propagating uncertainties are covered in the following subsection.

2.4 UNCERTAINTY PROPAGATION FUNDAMENTALS

The mathematics of uncertainty propagation presented in this section are taken from chapter 3 of Ref. [31]. First, consider an uncertain quantity x with a small uncertainty δx. Here, "small" implies that for a smooth function $f(x)$, linear Taylor approximation may be used to approximate $f(x \pm \delta x)$ as $f(x) \pm \frac{\partial f}{\partial x}\big|_x \delta x$. Here and below, $\frac{\partial f}{\partial x}\big|_x$ is the first derivative of $f(x)$ evaluated at x. All uncertain quantities below are assumed to have small uncertainties. Also, note, that all uncertainty magnitudes (e.g., δx) are, by definition, nonnegative.

The sum q with uncertainty δq of two variables x and y with independent uncertainties is given by:

$$q \pm \delta q = x + y \pm \sqrt{\delta x^2 + \delta y^2}, \tag{2.9}$$

and the difference $q \pm \delta q$ of two variables x and y with independent uncertainties is given by:

$$q \pm \delta q = x - y \pm \sqrt{\delta x^2 + \delta y^2}. \tag{2.10}$$

In other words, for a sum or a difference of uncertain variables with independent uncertainties, uncertainties add in quadrature.

However, if there is no guarantee that the uncertainties are independent, then the sum $q \pm \delta q$ becomes:

$$q \pm \delta q = x + y \pm [\delta x + \delta y], \tag{2.11}$$

and the difference:

$$q \pm \delta q = x - y \pm [\delta x + \delta y], \tag{2.12}$$

In other words, for a sum or a difference of uncertain variables with potentially systematic (i.e., dependent) uncertainties, uncertainties add directly.

The product $q \pm \delta q$ of an uncertain variable $x \pm \delta x$ and an exactly known constant C is given by:

$$q \pm \delta q = Cx \pm |C|\delta x. \tag{2.13}$$

If $q \pm \delta q$ is the value of a smooth function $f(x)$, evaluated at $x \pm \delta x$, using linear approximation yields:

$$q \pm \delta q = f(x) \pm \left|\frac{\partial f}{\partial x}\Big|_x\right| \delta x. \tag{2.14}$$

The product $q \pm \delta q$ of two uncertain variables x and y with independent uncertainties is given by:

$$q \pm \delta q = xy \pm \sqrt{(x\delta y)^2 + (y\delta x)^2}, \tag{2.15}$$

and the product $q \pm \delta q$ of two uncertain variables x and y with potentially systematic (i.e., dependent) uncertainties is:

$$q \pm \delta q = xy \pm [|y|\delta x + |x|\delta y]. \tag{2.16}$$

The algorithms in chapter 3 produce transducer functions with independently uncertain parameters. The 1-axis transducer function (Eq. (2.1)) therefore becomes characterized by inverse sensitivity $s \pm \delta s$ and offset $o \pm \delta o$. To estimate the uncertainty δp in measured physical value p due to calibration uncertainties (i.e., without accounting for uncertainty δv in measured voltage v), we may therefore apply the above rules to obtain:

$$p \pm \delta p = s(v - o) \pm \sqrt{(v\delta s)^2 + (s\delta o)^2 + (o\delta s)^2}. \tag{2.17}$$

The 3-axis transducer function (Eq. (2.2)) is characterized, with uncertainties, by the sensor calibration matrix $\mathbf{A} \pm \delta\mathbf{A}$:

$$\mathbf{A} \pm \delta\mathbf{A} = \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{bmatrix} \pm \begin{bmatrix} \delta A_{1,1} & \delta A_{1,2} & \delta A_{1,3} \\ \delta A_{2,1} & \delta A_{2,2} & \delta A_{2,3} \\ \delta A_{3,1} & \delta A_{3,2} & \delta A_{3,3} \end{bmatrix}, \tag{2.18}$$

and the offset vector $\vec{o} \pm \delta\vec{o}$:

$$\vec{o} \pm \delta\vec{o} = \begin{bmatrix} o_x \\ o_y \\ o_z \end{bmatrix} \pm \begin{bmatrix} \delta o_x \\ \delta o_y \\ \delta o_z \end{bmatrix}. \tag{2.19}$$

As stated above, all 12 of the above parameters (9 elements of $\mathbf{A}$ and 3 of $\vec{o}$) are independently uncertain. Recognizing this, we again apply the above rules to estimate the (independent) uncertainties $\delta\vec{p}$ in the measured physical value vector $\vec{p}$:

$$p_j \pm \delta p_j = \sum_{i=1}^{3} A_{j,i}\left(v_j - o_j\right) \pm \sqrt{\sum_{i=1}^{3}\left(A_{j,i}\delta o_i\right)^2 + \sum_{i=1}^{3}\left(o_i \delta A_{j,i}\right)^2 + \sum_{i=1}^{3}\left(v_i \delta A_{j,i}\right)^2}, \tag{2.20}$$

in which the subscript j, with $j = 1 = x$, $j = 2 = y$, and $j = 3 = z$, was used.

With the above rules, these expressions for uncertain transducer functions, and a set of calibration parameters with quantified uncertainties for a given sensor, one can quantify the uncertainties and best estimate values for any measurement made with this sensor.

In the following chapter, the computational stages for 1- and 3-axis inertial sensor calibration, including calibration uncertainty quantification, are provided.

Chapter 3
Inertial Sensor Calibration Theory

As was discussed in section 2.1, inertial sensors require calibration. The outcome of an inertial sensor calibration process is this sensor's *"transducer function,"* which converts between the voltage(s) that the sensor outputs, and the physical value(s) that the sensor measured. In this text, the *"calibration process"* for an inertial sensor is understood to consist of:

1. Specifications for an experiment, or a series of experiments, that impose a set of *"reference"* (i.e., known) physical values onto the sensor, and measure the corresponding voltages. If the calibration process provides uncertainty quantification for the transducer function, it must also include guidelines for how to quantify the uncertainties in the measured voltages and reference physical values.

   These specifications are the applied part of the calibration process, and generally constitute the bulk of the time necessary to apply it. Several such specifications are described in section 2.2.
2. A calibration algorithm that uses the values obtained in stage 1 to construct the sensor's transducer function. A *"deterministic"* calibration algorithm results in the best fit function, while an *"uncertainty-quantifying"* calibration algorithm results in the best fit function, and some specification for this function's uncertainties (typically error margins for each term).

   The application of the calibration algorithm, once it is properly developed, generally requires simply running a computer program, like a MATLAB script, to process the values from stage 1. Even for a large number of measurements, this stage is generally much faster.

Experiment specifications from stage 1 are specific to the sensor type, and, by definition, to the experiment. Calibration algorithms, however, can be more general: one-step calibration algorithms presented in this

chapter can be applied to any sensor with the algorithm-specific number of axes, and algorithm-specific transducer function shape. The two-step calibration algorithm in this chapter can only be applied to the two-step static calibration experiment data (calibration method 4 in section 2.2), and is therefore only applicable to 3-axis accelerometers (and also to magnetometers, which are not used in this work).

The External Measurement System developed in this work consists of a set of inertial sensors, with associated data acquisition systems, and carefully specified calibration processes for them. The experiment specifications from these calibration processes are given in chapters 4 and 5, and several calibration algorithms are presented below.

3.1 DETERMINISTIC CALIBRATION ALGORITHMS

As was discussed in section 2.1, a 1-axis inertial sensor typically has a linear transducer function, with a (normally) nonzero offset voltage:

$$p(v) = s(v - o), \tag{3.1}$$

in which $p(v)$ is the physical value (e.g., acceleration), v is the voltage output by the sensor, o is the voltage offset, and s is the sensor inverse sensitivity.

A 3-axis inertial sensor also has a linear transducer function:

$$\vec{\boldsymbol{p}}\left(\vec{\boldsymbol{v}}\right) = \mathbf{A}\left(\vec{\boldsymbol{v}} - \vec{\boldsymbol{o}}\right), \tag{3.2}$$

in which $\vec{\boldsymbol{p}}$, $\vec{\boldsymbol{v}}$ and $\vec{\boldsymbol{o}}$ are the 3-axis vector versions of the corresponding scalar variables from Eq. (3.1), and $\mathbf{A}$ is the sensor calibration matrix. 2-axis sensors were not used in this work, but they too would have the transducer function in Eq. (3.2). Recall from section 2.1, that $\vec{\boldsymbol{p}}$ exists in the body frame of the sensor.

The objective of the 1-axis deterministic calibration algorithm is to find the inverse sensitivity s and offset o which best fit, in the least squares sense, the set of N scalar voltage measurements $\{v\} = \{v_1, \ldots, v_N\}$, with corresponding scalar reference physical values $\{p\} = \{p_1, \ldots, p_N\}$. Similarly, the objective of the one-step 3-axis deterministic calibration algorithm is to find $\mathbf{A}$ and $\vec{\boldsymbol{o}}$ which best fit, in the least squares sense, the set of N vector voltage measurements $\left\{\vec{\boldsymbol{v}}\right\} = \left\{\vec{\boldsymbol{v}}_1, \ldots, \vec{\boldsymbol{v}}_N\right\}$, with corresponding vector reference physical values $\left\{\vec{\boldsymbol{p}}\right\} = \left\{\vec{\boldsymbol{p}}_1, \ldots, \vec{\boldsymbol{p}}_N\right\}$. Both algorithms ignore the uncertainties in both the measurements and the reference values; see section 3.2 for the uncertainty-quantifying algorithms that do account for them.

The two-step 3-axis deterministic calibration algorithm also finds $\mathbf{A}$ and $\vec{o}$, but does so based two different specific sets of data, and is only applicable to 3-axis accelerometers (and magnetometers).
The deterministic algorithm for 1-axis sensors is presented in subsection 3.1.1 below. The one-step 3-axis deterministic algorithm is in subsection 3.1.2. The more complicated two-step 3-axis deterministic algorithm is in subsection 3.1.3.

3.1.1 *Deterministic Algorithm for 1-axis Sensors*

Equation (3.1) is a linear equation, which can be expressed as the following more conventional 1st-order polynomial:

$$p(v) = sv + c_0, \tag{3.3}$$

with the y-intercept c_0 related to s and o through:

$$c_0 = -so. \tag{3.4}$$

The task of finding the linear polynomial coefficients c_0 and s that "best fit" a given dataset $(\{v\}, \{p\})$ is a very common one, known as *"(simple) linear regression"* (Ref. [31], chapter 8). Here, the N voltages $\{v\}$ are the independent (x-) values, and the N reference physical values $\{p\}$ are the dependent (y-) values. The following requirement applies:

$$N \geq 2, \tag{3.5}$$

otherwise the system is underdetermined.
Typically, the best fit is defined as one that yields the lowest possible sum of squares of the errors E, expressed as follows (Ref. [31], chapter 8):

$$E(c_0, s) = E\left(\vec{c}\right) = \sum_{i=1}^{N} (p(v_i) - p_i)^2 = \sum_{i=1}^{N} (sv_i + c_0 - p_i)^2. \tag{3.6}$$

The minimization of Eq. (3.6) is a scalar minimization problem for the convex function $E(c_0, s)$, which, for such problems, is known as the *"cost function."* The scalar minimization problem for a general multivariate scalar function can be expressed as follows:

$$\min_{\vec{\mathbf{x}}} f\left(\vec{\mathbf{x}}\right) \text{ for any real scalar function } f\left(\vec{\mathbf{x}}\right), \tag{3.7}$$

where, in case of Eq. (3.6), $E\left(\vec{c}\right)$ plays the role of $f\left(\vec{\mathbf{x}}\right)$, with $\vec{c} = \left[c_0, s\right]^{\mathrm{T}}$.

Many methods exist for solving the problem of Eq. (3.7); below I list three of them:

1. Steepest Descent [32]. This method works by constructing (either numerically, or analytically, if available) the gradient $\nabla f\left(\vec{\mathbf{x}}\right)$, taking a step in the opposite (from the gradient) direction, reevaluating the cost function, and repeating until the steps yield no further improvement.
2. Quasi-Newton algorithm, implemented in the **`fminunc`** function of the MATLAB Optimization Toolbox: it is essentially an accelerated form of the above-mentioned Steepest Descent method. It utilizes a finite difference-based gradient calculator, and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) formula for Hessian ("gradient of gradient") approximation, known to be highly robust [33] [34] [35] [36].
3. The Nelder-Mead simplex search algorithm, implemented in the MATLAB **`fminsearch`** function [37]. This method does not rely on either numeric or analytic estimates of gradients. It is generally slower than the Quasi-Newton algorithm, but for small problems, can be superior, because it is able to take a lot faster (less efficient) iterations, than the gradient-based algorithms.

Note, that all 3 of the above methods are meant for the general multivariate scalar function minimization problem of Eq. (3.7). Our problem, however, is more specific: it is a least squares minimization problem, which refines Eq. (3.7) as follows:

$$\min_{\vec{\mathbf{x}}} f\left(\vec{\mathbf{x}}\right) \text{ for } f\left(\vec{\mathbf{x}}\right) = \sum_{i=1}^{N} \left(g_i\left(\vec{\mathbf{x}}, v_i\right) - p_i\right)^2, \tag{3.8}$$

where, in our case, $p(v)$ from Eq. (3.3) plays the role of $g_i\left(\vec{\mathbf{x}}, v_i\right)$, with $\vec{c}$ instead of $\vec{\mathbf{x}}$.

The classic method for solving this much less general problem is the Levenberg-Marquardt algorithm [38] [39]. It also relies on a gradient, but recognizes that each term of the sum in Eq. (3.8) is a square of an element of the error vector, and takes advantage of this knowledge. In MATLAB Optimization Toolbox, the function **`lsqnonlin`** can use this algorithm, and **`lsqcurvefit`** can use it with the recognition that the error vector is of the form in Eq. (3.8).

Our problem is even more specific than Eq. (3.8) though: $g_i\left(\vec{\mathbf{x}}, v_i\right)$ is not a general, but a linear scalar function. The problem therefore becomes:

$$\min_{\vec{\mathbf{x}}} f\left(\vec{\mathbf{x}}\right) \text{ for } f\left(\vec{\mathbf{x}}\right) = f(c_0, s) = \sum_{i=1}^{N} (s v_i + c_0 - p_i)^2. \tag{3.9}$$

Equation (3.9) turns out to be the statement of an overdetermined linear algebraic problem, solved in the least squares sense. To show this, first consider the following linear algebraic problem:

$$\begin{bmatrix} 1 & v_1 \\ \vdots & \vdots \\ 1 & v_N \end{bmatrix} \begin{bmatrix} c_0 \\ s \end{bmatrix} = \begin{bmatrix} p_1 \\ \vdots \\ p_N \end{bmatrix}. \tag{3.10}$$

If $N = 2$, Eq. (3.10) is solvable exactly, unless $v_1 = v_2$ (in which case the data is useless). For $N > 2$ though, Eq. (3.10) has no exact solution, because it is an overdetermined algebraic linear system. Such systems are discussed in section 2.4, in which it is stated, that they are normally solved "in the least squares sense." In effect, this means finding the solution vector $\vec{c}$ that minimizes the sum of the squares of the differences between the left- and right-hand sides of the system.

To express this sum, I multiply the coefficient matrix by the vector, and get:

$$\begin{aligned} c_0 + v_1 s &= p_1, \\ &\vdots \\ c_0 + v_N s &= p_{N.} \end{aligned} \tag{3.11}$$

The sum of the squares of the differences between the left- and right-hand sides then becomes:

$$(c_0 + v_1 s - p_1)^2 + \cdots + (c_0 + v_N s - p_N)^2 = \sum_{i=1}^{N} (s v_i + c_0 - p_i)^2, \tag{3.12}$$

which happens to be the exact expression in Eqs. (3.6) and (3.9). This confirms, that the search for the linear regression function which the 1-axis deterministic calibration algorithm must find (Eq. (3.1)), can be viewed as simply an overdetermined linear algebraic problem, constructed as follows:

- The coefficient matrix $\mathbf{M}$ is given by:

$$\mathbf{M} = \begin{bmatrix} 1 & v_1 \\ \vdots & \vdots \\ 1 & v_N \end{bmatrix}. \tag{3.13}$$

- The right hand side vector $\vec{\mathbf{b}}$ is given by:

$$\vec{\mathbf{b}} = \begin{bmatrix} p_1 \\ \vdots \\ p_N \end{bmatrix}. \tag{3.14}$$

- The solution vector are the coefficients of best fit, $\vec{c}$.

This problem must then be solved in the least squares sense. This is therefore not, strictly, a "linear" algebraic problem – it is a least squares problem for a linear curve.

The most common method for solving the overdetermined linear algebraic problem in the least squares sense is the orthogonal-triangular decomposition (or "factorization") method, often simply called *"QR factorization."* This is the method that MATLAB's "\" operator, also called by the **`mldivide`** function, uses to solve overdetermined linear algebraic problems in the least squares sense.

Note, that all of the optimization methods above, except QR factorization, are iterative. Therefore, they do not yield an exact solution: they converge to it, but terminate once reaching a specified tolerance (which, for a cost function, can be hard to select, particularly for a multiscale problem). They also require an initial guess for the coefficients. QR factorization is different: it is an exact method, that does not require initial guesses, and yields an exact solution. Therefore, assuming similar or superior speeds, it is generally a better method for our purposes.

Any one of the optimization methods discussed above can potentially be used as part of the deterministic algorithm for 1-axis sensor calibration. In this thesis, I have tested two for this purpose: MATLAB's **`fminsearch`** (Nelder-Mead simplex search, method 3 in the list of general methods above) to minimize Eq. (3.6), and MATLAB's "\" operator, treating the problem as an overdetermined linear algebraic problem, using Eqs. (3.13) and (3.14). The first approach was chosen because it is frequently reported to be used for this problem in literature (e.g., Ref. [40]), and also because for small problems (e.g., fewer than a hundred measurements) it converges faster than the gradient-dependent methods. (Note, that it may take more iterations, but the speed is still higher, because the cost of an iteration is much lower). The second approach was chosen because it is the most narrow-purpose one, and because unlike the other iterative algorithms, it is exact.

The 1-axis deterministic calibration algorithm, using the **`fminsearch`**-based approach, consists of the following steps:

1. Accept the voltage set $\{v\}$ and the reference physical value set $\{p\}$ as inputs. The sets must be of the same size $N \geq 2$.
2. Select an initial guess vector $\vec{c}^{(0)}$, using the inverse sensitivity and offset values specified in the sensor's datasheet, and Eq. (3.4) to compute the coefficient c_0.

3. In MATLAB, construct the cost function $E(\vec{c})$ using Eq. (3.6), and the data from step 1.
4. Use MATLAB **`fminsearch`** function to minimize $E(\vec{c})$, with the initial guess from step 2.
5. Using the resulting $\vec{c}$, extract s and compute o:

$$o = -\frac{c_0}{s}. \tag{3.15}$$

The 1-axis deterministic calibration algorithm, using the overdetermined linear algebraic problem approach, consists of the following steps:

1. Accept the voltage set $\{v\}$ and the reference physical value set $\{p\}$ as inputs. The sets must be of the same size $N \geq 2$.
2. Construct the coefficient matrix $\mathbf{M}$, using Eq. (3.13).
3. Construct the right hand side vector $\vec{\mathbf{b}}$, using Eq. (3.14).
4. Use MATLAB **`mldivide`** function, or the "\" operator, to solve the $\mathbf{M}\vec{c} = \vec{\mathbf{b}}$ overconstrained linear problem in the least squares sense.
5. Using the resulting $\vec{c}$, extract s and compute o (Eq. (3.15)).

The two approaches can be compared: the quality of fit can be evaluated by substituting $\vec{c}$ into Eq. (3.6). In chapter 4, I show that the overdetermined linear algebraic problem approach is clearly superior for 1-axis sensors.

The one-step 3-axis deterministic algorithm, which is an extension of the one presented above, is given in the following subsection.

3.1.2 *Deterministic One-step Algorithm for 3-axis Sensors*

Equation (3.2) is a linear equation, like Eq. (3.1). Rearranging it yields:

$$\vec{p}(\vec{v}) = \mathbf{A}\vec{v} + \vec{\mathbf{c}}_0, \tag{3.16}$$

with the following coefficients:

$$\mathbf{A} = \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{bmatrix}, \tag{3.17}$$

$$\vec{\mathbf{c}}_0 = \begin{bmatrix} c_{0,1} \\ c_{0,2} \\ c_{0,3} \end{bmatrix} = -\mathbf{A}\vec{o}. \tag{3.18}$$

The task of finding the coefficient matrix $\mathbf{A}$ and the coefficient vector $\vec{\mathbf{c}}_0$ that "best fit" a given dataset of vector data pairs $\left(\{\vec{v}\}, \{\vec{p}\}\right)$ is known as *"general linear regression"* [41]. Here, the N voltages $\{\vec{v}\}$ are the independent values, and the N reference physical values $\{\vec{p}\}$ are the dependent values. For N_d-dimensional datasets $\{\vec{v}\}$ and $\{\vec{p}\}$, the following requirements apply:

$$\mathbf{A} \in \mathbb{R}^{N_d \times N_d}, \tag{3.19}$$

$$\vec{\mathbf{c}}_0 \in \mathbb{R}^{N_d}, \tag{3.20}$$

$$N \geq N_d + 1. \tag{3.21}$$

The first two requirements arise directly from Eq. (3.16), and the third is necessary, because otherwise the system would be underdetermined.

As with simple linear regression, the best fit is usually defined as one that yields the lowest possible sum of squares of the Euclidean norms of errors E, now expressed as follows [41]:

$$E\left(\mathbf{A}, \vec{\mathbf{c}}_0\right) = E\left(\vec{c}\right) = \sum_{i=1}^{N} \left\| \vec{p}\left(\vec{v}_i\right) - \vec{p}_i \right\|^2 = \sum_{i=1}^{N} \left\| \mathbf{A}\vec{v}_i + \vec{\mathbf{c}}_0 - \vec{p}_i \right\|^2. \tag{3.22}$$

Here, $\|\vec{v}\|$ denotes the Euclidean norm of vector $\vec{v}$, and the coefficient vector $\vec{c}$ can be written as:

$$\vec{c} = \left[c_{0,1}, c_{0,2}, c_{0,3}, A_{1,1}, A_{1,2}, A_{1,3}, A_{2,1}, A_{2,2}, A_{2,3}, A_{3,1}, A_{3,2}, A_{3,3}\right]^{\mathrm{T}}. \tag{3.23}$$

As was discussed in subsection 3.1.1 above, the minimization of a general multivariate scalar cost function is a known problem. Multiple methods, including the Nelder-Mead simplex search algorithm, on which MATLAB's **`fminsearch`** is based, were presented there. Using Eq. (3.22) instead of Eq. (3.6), the rest of the **`fminsearch`**-based approach holds for the 3-axis algorithm without any additional changes. For 3-axis sensors, most texts in literature appear to use some form of this approach [7] [40].

As with the 1-axis problem, we note, that ours is more specific than Eq. (3.7). This is also a least squares minimization problem, but the squares are those of Euclidean norms of the error vectors, and not of the scalar differences between the reference and predicted values. We can rearrange Eq. (3.22), however, to make the residual function look structurally similar to the one Eq. (3.8).

Expanding a single element of the sum in Eq. (3.22), for a 3-axis problem:

$$\begin{aligned}\left\|\vec{\boldsymbol{p}}\left(\vec{\boldsymbol{v}}_i\right)-\vec{\boldsymbol{p}}_i\right\|^2 &= \left\|\mathbf{A}\vec{\boldsymbol{v}}_i+\vec{\mathbf{c}}_0-\vec{\boldsymbol{p}}_i\right\|^2 = \\ &= \left[\sqrt{\left(\mathbf{A}\vec{\boldsymbol{v}}_i+\vec{\mathbf{c}}_0-\vec{\boldsymbol{p}}_i\right)_x^2+\left(\mathbf{A}\vec{\boldsymbol{v}}_i+\vec{\mathbf{c}}_0-\vec{\boldsymbol{p}}_i\right)_y^2+\left(\mathbf{A}\vec{\boldsymbol{v}}_i+\vec{\mathbf{c}}_0-\vec{\boldsymbol{p}}_i\right)_z^2}\right]^2 = \\ &= \left(A_{1,1}v_{ix}+A_{1,2}v_{iy}+A_{1,3}v_{iz}+c_{0x}-p_{ix}\right)^2+\cdots \\ &\ldots+\left(A_{2,1}v_{ix}+A_{2,2}v_{iy}+A_{2,3}v_{iz}+c_{0y}-p_{iy}\right)^2+\cdots \\ &\ldots+\left(A_{3,1}v_{ix}+A_{3,2}v_{iy}+A_{3,3}v_{iz}+c_{0z}-p_{iz}\right)^2.\end{aligned} \tag{3.24}$$

Using subscript j, with $j=1=x$, $j=2=y$, and $j=3=z$, we can write the sum as follows:

$$\left\|\mathbf{A}\vec{\boldsymbol{v}}_i+\vec{\mathbf{c}}_0-\vec{\boldsymbol{p}}_i\right\|^2=\sum_{j=1}^{3}\left(A_{j,1}v_{ix}+A_{j,2}v_{iy}+A_{j,3}v_{iz}+c_{0j}-p_{ij}\right)^2. \tag{3.25}$$

Substituting Eqs. (3.24) and (3.25) into Eq. (3.22) yields the following form for the residual function:

$$E\left(\vec{\boldsymbol{c}}\right)=\sum_{i=1}^{N}\sum_{j=1}^{3}\left(A_{j,1}v_{ix}+A_{j,2}v_{iy}+A_{j,3}v_{iz}+c_{0j}-p_{ij}\right)^2. \tag{3.26}$$

As we can see, this is a sum of squares! Minimizing this function is therefore, again, a least squares minimization problem, exactly like the one in Eq. (3.8). Like Eq. (3.8), this problem can be solved using the Levenberg-Marquardt algorithm.

Also like in Eq. (3.8), the function in Eq. (3.26) is more specific: each error term is the difference between a linear 3-variable function, and a scalar. It is therefore a multivariate version of Eq. (3.9), which we know to be the statement of an overdetermined linear algebraic problem, solved in the least squares sense.

Before demonstrating this for Eq. (3.26), it will be helpful to demonstrate a small fact. Consider a vector $\vec{\mathbf{y}}$ of length N_y, and a vector $\vec{\mathbf{z}}$ of length N_z. Composing them into a vector $\vec{\mathbf{q}}$:

$$\vec{\mathbf{q}}=\begin{bmatrix}\vec{\mathbf{y}}\\ \vec{\mathbf{z}}\end{bmatrix}=\begin{bmatrix}y_1 & \cdots & y_{N_y} & z_1 & \ldots & z_{N_z}\end{bmatrix}^{\mathrm{T}}. \tag{3.27}$$

Evaluating the square of its Euclidean norm:

$$\|\vec{\mathbf{q}}\|^2 = \sum_{i_y=1}^{N_y} y_{j_y}^2 + \sum_{i_z=1}^{N_z} z_{i_z}^2 = \|\vec{\mathbf{y}}\|^2 + \|\vec{\mathbf{z}}\|^2. \tag{3.28}$$

We can see that the square of the sum of the terms of this vector can be written as the sum of the Euclidean norms of its two constituent vectors, with "constituent" in the sense defined in Eq. (3.27). By trivial deduction, one can see that this will be with any number of constituent vectors.

Next, consider the following matrix:

$$\mathbf{M}_i = \begin{bmatrix} 1 & 0 & 0 & v_{ix} & v_{iy} & v_{iz} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & v_{ix} & v_{iy} & v_{iz} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & v_{ix} & v_{iy} & v_{iz} \end{bmatrix}. \tag{3.29}$$

Multiplying $\mathbf{M}_i$ by $\vec{\boldsymbol{c}}$ from Eq. (3.23):

$$\mathbf{M}_i\vec{\boldsymbol{c}} = \begin{bmatrix} c_{0x} + A_{1,1}v_{ix} + A_{1,2}v_{iy} + A_{1,3}v_{iz} \\ c_{0y} + A_{2,1}v_{ix} + A_{2,2}v_{iy} + A_{2,3}v_{iz} \\ c_{0z} + A_{3,1}v_{ix} + A_{3,2}v_{iy} + A_{3,3}v_{iz} \end{bmatrix}. \tag{3.30}$$

Next, consider the following problem:

$$\mathbf{M}_i\vec{\boldsymbol{c}} = \vec{\boldsymbol{p}}_i. \tag{3.31}$$

By itself, it is underdetermined: at least 4 such problems are needed to make it determined, and more to make it overdetermined. Still, we can express the square of its residual:

$$\begin{aligned} \|\mathbf{M}_i\vec{\boldsymbol{c}} - \vec{\boldsymbol{p}}_i\|^2 = & \\ &= \left(c_{0x} + A_{1,1}v_{ix} + A_{1,2}v_{iy} + A_{1,3}v_{iz} - p_{ix}\right)^2 + \cdots \\ &\ldots + \left(c_{0y} + A_{2,1}v_{ix} + A_{2,2}v_{iy} + A_{3,3}v_{iz} - p_{iy}\right)^2 + \\ &\ldots + \left(c_{0z} + A_{3,1}v_{ix} + A_{3,2}v_{iy} + A_{3,3}v_{iz} - p_{iz}\right)^2 = \\ &= \sum_{j=1}^{3}\left(c_{0j} + A_{j,1}v_{ix} + A_{j,2}v_{iy} + A_{j,3}v_{iz} - p_{ij}\right)^2 \end{aligned} \tag{3.32}$$

Clearly the squared residual in Eq. (3.32) is simply the inner sum of Eq. (3.26).

Next, we construct the following right-hand side vector:

$$\vec{\mathbf{b}} = \begin{bmatrix} \vec{p}_1 \\ \vdots \\ \vec{p}_N \end{bmatrix} = \begin{bmatrix} p_{1x} \\ p_{1y} \\ p_{1z} \\ \vdots \\ p_{Nz} \end{bmatrix}, \tag{3.33}$$

and the following full coefficient matrix:

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_1 \\ \vdots \\ \mathbf{M}_N \end{bmatrix}. \tag{3.34}$$

Lastly, we construct the linear algebraic problem:

$$\mathbf{M}\vec{c} = \vec{\mathbf{b}}. \tag{3.35}$$

Like Eq. (3.10), Eq. (3.35) is a linear algebraic problem, determined if $N = 4$, and overdetermined if $N > 4$. When solved in the least squares sense, and using the fact from Eq. (3.28) for N constituent vectors of 3 elements each, its residual squared can be written as follows:

$$\left\|\mathbf{M}\vec{c} - \vec{\mathbf{b}}\right\|^2 = \sum_{i=1}^{N} \|\mathbf{M}_i\vec{c} - \vec{p}_i\|^2. \tag{3.36}$$

Substituting Eq. (3.32) into Eq. (3.36) finally yields the following double sum:

$$\left\|\mathbf{M}\vec{c} - \vec{\mathbf{b}}\right\|^2 = \sum_{i=1}^{N}\sum_{j=1}^{3}\left(c_{0j} + A_{j,1}v_{ix} + A_{j,2}v_{iy} + A_{j,3}v_{iz} - p_{ij}\right)^2, \tag{3.37}$$

which happens to be the exact expression in Eq. (3.26). This confirms, that the search for the general linear regression, which the 3-axis deterministic calibration algorithm must find (Eq. (3.2)) can, like the 1-axis problem, be viewed as simply an overdetermined linear algebraic problem, constructed as follows:

- The coefficient matrix $\mathbf{M}$ is given by Eq. (3.34), with its components from Eq. (3.29).
- The right-hand side vector $\vec{\mathbf{b}}$ is given by Eq. (3.33).
- The solution vector are the coefficients of best fit, $\vec{c}$, from Eq. (3.23).

This problem can be solved in the least squares sense, using exactly the same methods (i.e., QR factorization through MATLAB "\" operator) as those used for this purpose in subsection 3.1.1.

As in subsection 3.1.1, I can now formulate two approaches to the one-step 3-axis deterministic calibration algorithm. The one-step 3-axis deterministic calibration algorithm, using the **fminsearch**-based approach, consists of the following steps:

1. Accept the voltage set $\{\vec{v}\}$ and the reference physical value set $\{\vec{p}\}$ as inputs. The sets must be of the same size $N \geq 4$, and each element must be a 3-dimensional vector.
2. Select an initial guess vector $\vec{c}^{(0)}$, using the sensitivities (and misalignment angles, if they are specified to be nonzero, which is generally not the case for commercially available sensors) and offset values specified in the sensor's datasheet. Use Eq. (3.18) to calculate the coefficient vector $\vec{c}_0$. Because most sensors are specified with a single sensitivity $k = s^{-1}$ for all axes, and 0 misalignment angles, the following initial guess for the coefficient matrix $\mathbf{A}^{(0)}$ is usually the best available:

$$\mathbf{A}^{(0)} = \begin{bmatrix} \frac{1}{k} & 0 & 0 \\ 0 & \frac{1}{k} & 0 \\ 0 & 0 & \frac{1}{k} \end{bmatrix}. \tag{3.38}$$

3. In MATLAB, construct the cost function $E(\vec{c})$ using Eq. (3.22), and the data from step 1.
4. Use MATLAB **fminsearch** to minimize $E(\vec{c})$, with the initial guess from step 2.
5. Using the resulting $\vec{c}$, construct $\mathbf{A}$ using Eq. (3.23), and extract the coefficient vector $\vec{c}_0$.
6. Compute $\vec{o}$:

$$\vec{o} = -\mathbf{A}^{-1}\vec{c}_0. \tag{3.39}$$

$\mathbf{A}^{-1}$ is always possible, because $\mathbf{A}$ is strictly diagonally dominant, and therefore is nonsingular [42].

The 3-axis deterministic calibration algorithm, using the overdetermined linear problem approach, consists of the following steps:

1. Accept the voltage set $\{\vec{v}\}$ and the reference physical value set $\{\vec{p}\}$ as inputs. The sets must be of the same size $N \geq 4$, and each element must be a 3-dimensional vector.
2. Construct the coefficient matrix $\mathbf{M}$, using Eq. (3.34), with components from Eq. (3.29).
3. Construct the right-hand side vector $\vec{\mathbf{b}}$, using Eq. (3.33).
4. Use MATLAB **mldivide** function, or the "\" operator, to solve Eq. (3.35) in the least squares sense.
5. Using the resulting $\vec{c}$, construct $\mathbf{A}$ using Eq. (3.23), and extract the coefficient vector $\vec{c}_0$.

6. Compute $\vec{o}$ (Eq. 3.39).

As with the 1-axis problem, the two approaches can be compared: the quality of fit can be evaluated by substituting $\vec{c}$ into Eq. (3.22). In chapter 4, I show that the overdetermined linear algebraic problem approach is also clearly superior for 3-axis sensors. This by itself is a useful result, because in literature the use of `fminsearch` is prevalent.

The two-step deterministic algorithm for 3-axis accelerometers, which is reliant on data from method 4 from section 2.2, is described in the following subsection.

3.1.3 *Deterministic Two-step Algorithm for 3-axis Sensors*

Bonnet et al. use the factorization of $\mathbf{A}$ in Eq. (2.8), and separate the finding of the calibration parameters into two steps [9]. The algorithm described below is taken nearly verbatim from Ref. [9], with appropriate notation changes, and several modifications explicitly noted as such. It may be used for calibrating accelerometers and magnetometers.

In step 1, $\mathbf{B}$ and $\vec{o}$ are found. The step 1 of the experiment, associated with this calibration method, produces a set of $N \geq 9$ voltage vectors $\{\vec{v}\}$, each with a fixed magnitude (i.e., Euclidean norm) of m. Their orientations are not used.

Step 1 of this calibration algorithm consists of minimizing the following cost function:

$$F(\mathbf{B}, \vec{o}) = \sum_{i=1}^{N} \left[(\vec{v}_i - \vec{o})^{\mathrm{T}} \mathbf{B}^{\mathrm{T}} \mathbf{B} (\vec{v}_i - \vec{o}) - m^2 \right]^2, \tag{3.40}$$

The minimization of this function is the ellipsoid optimization problem (i.e., the search for best fit, using an ellipsoid as the fitting function), because for an upper triangular $\mathbf{B}$, Eq. (3.41) is the general ellipsoid equation in $\vec{v} \in \mathbb{R}^3$:

$$(\vec{v} - \vec{o})^{\mathrm{T}} \mathbf{B}^{\mathrm{T}} \mathbf{B} (\vec{v} - \vec{o}) = m^2. \tag{3.41}$$

Because gravity is used as the field with constant magnitude m:

$$m = 1g. \tag{3.42}$$

The cost function in Eq. (3.40) is nonlinear, and unlike Eq. (3.6) or Eq. (3.22), cannot be reformulated as a sum of residual squares with a linear fitting function. Its minimization is therefore not solvable via the overdetermined linear problem. Bonnet et al. propose using the Quasi-Newton nonlinear optimizer, a form of which is implemented in the MATLAB Optimization Toolbox as the **`fminunc`** function.

In step 2, $\mathbf{R}^{sb}$ is found. The step 2 of the experiment, associated with this calibration method, produces 3 sets of measurements, of N_x, N_y and N_z voltage vectors. Each of the 3 sets $\left\{\vec{\boldsymbol{v}}\right\}_j = \left\{\vec{\boldsymbol{v}}_1^j, \ldots, \vec{\boldsymbol{v}}_{N_j}^j\right\}$ consists of measurements made when rotating the sensor about body axis j, keeping axis j fixed relative to gravity. Step 2 of this calibration algorithm relies on the fact that when the accelerometer is rotated about a fixed body axis j, the projection of gravity onto this axis is constant:

$$\mathbf{B}\left(\vec{\boldsymbol{v}}_i^j - \vec{\boldsymbol{o}}\right) \cdot \hat{\boldsymbol{m}}^j = 1g\cos\left(\theta_j\right) \;\; \forall\, i = 1 \ldots N_j. \tag{3.43}$$

Here $\hat{\boldsymbol{m}}^j$ is the unit vector for body axis j in the orthogonalized sensor frame, to be used to construct $\mathbf{R}^{sb}$ below, and θ_j is the angle between body axis j and the vertically upward direction. This vector can be found as the solution for the following cost function minimization problem:

$$\min_{\hat{\boldsymbol{m}}^j} \tilde{G}_j\left(\hat{\boldsymbol{m}}^j\right) = \min_{\hat{\boldsymbol{m}}^j} \sum_{i=1}^{N_j} \left(\mathbf{B}\left(\vec{\boldsymbol{v}}_i^j - \vec{\boldsymbol{o}}\right) \cdot \hat{\boldsymbol{m}}^j - 1g\cos\left(\theta_j\right)\right)^2. \tag{3.44}$$

The minimization problem in Eq. (3.44), when solved for each body axis j, yields 3 unit vectors in the orthogonalized sensor frame. This problem is clearly an overdetermined linear algebraic problem:

$$\mathbf{M}^j \vec{\boldsymbol{m}}^j = \vec{\boldsymbol{b}}^j, \tag{3.45}$$

with the following matrix $\mathbf{M}^j$:

$$\mathbf{M}^j = \begin{bmatrix} \left(\mathbf{B}\left(\vec{\boldsymbol{v}}_1^j - \vec{\boldsymbol{o}}\right)\right)^{\mathrm{T}} \\ \vdots \\ \left(\mathbf{B}\left(\vec{\boldsymbol{v}}_{N_j}^j - \vec{\boldsymbol{o}}\right)\right)^{\mathrm{T}} \end{bmatrix}, \tag{3.46}$$

and the following right-hand side vector $\vec{\boldsymbol{b}}^j$:

$$\vec{\boldsymbol{b}}^j = \begin{bmatrix} 1g\cos\left(\theta_j\right) \\ \vdots \\ 1g\cos\left(\theta_j\right) \end{bmatrix} \in \mathbb{R}^{N_j}. \tag{3.47}$$

However, Eq. (3.47) requires knowing θ_j, which can introduce an additional error. Dividing both sides of Eq. (3.45) by $g\cos\left(\theta_j\right)$ yields the following:

$$\mathbf{M}^j\vec{\boldsymbol{h}}^j = \vec{\mathbf{1}} \in \mathbb{R}^{N_j}, \tag{3.48}$$

in which $\vec{\boldsymbol{h}}^j$ are non-unit vectors that are parallel to $\widehat{\boldsymbol{m}}^j$, and from which $\widehat{\boldsymbol{m}}^j$ can therefore be obtained:

$$\widehat{\boldsymbol{m}}^j = \frac{\vec{\boldsymbol{h}}^j}{\left\|\vec{\boldsymbol{h}}^j\right\|}. \tag{3.49}$$

Unfortunately, Eq. (3.48) can only be valid if $\theta_j \neq 90°$, otherwise $\cos\left(\theta_j\right) = 0$, and dividing Eq. (3.45) by $g\cos\left(\theta_j\right)$ would mean dividing by 0. For an accelerometer, this describes a horizontal rotation body axis j, which is likely the most convenient axis to hold fixed (e.g., by mounting the accelerometer on a rectangular parallelepiped block, and rotating the block in 90° increments on a flat, precisely horizontal surface, as in chapter 4). Furthermore, as Bonnet et al. state, $\theta_j = 90°$ provides the most possible variation of the measured quantity in the orthogonal (to $\widehat{\boldsymbol{m}}^j$) plane, and therefore actually provides the best estimate of $\widehat{\boldsymbol{m}}^j$.
To address these challenges, Bonnet et al. point out, that by: (a) computing the means $\overline{M}_x^j$, $\overline{M}_y^j$ and $\overline{M}_z^j$, (b) and subtracting each mean from the corresponding column of $\mathbf{M}^j$, the problem of Eq. (3.45) is transformed. For illustration purposes, this operation is illustrated for a set of simulated measurements, on Fig. 3.1 below.

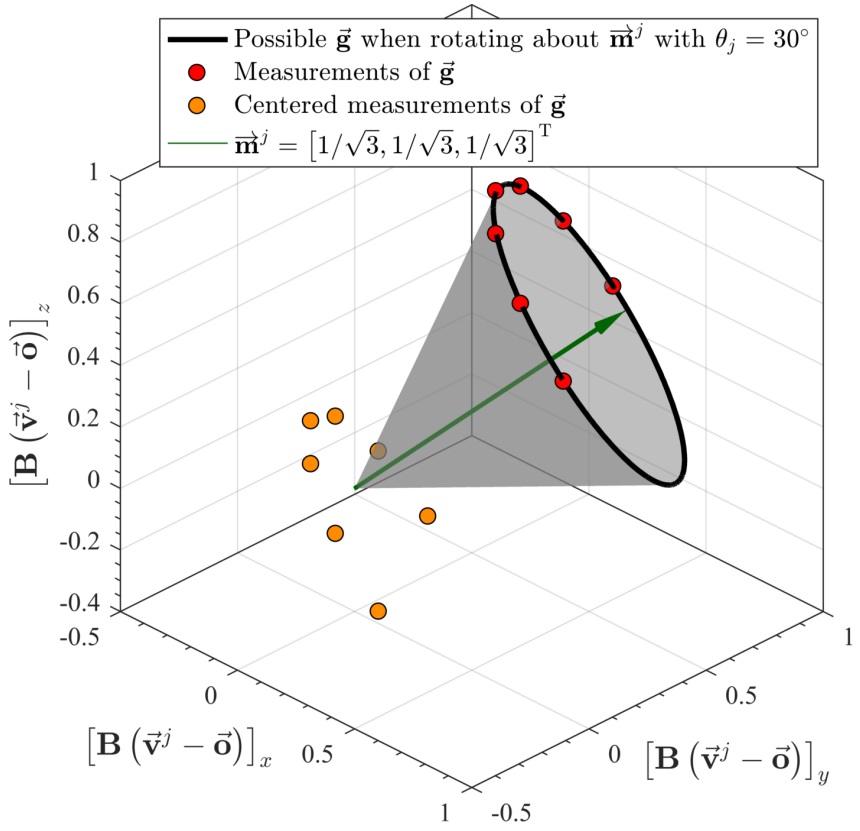

Fig. 3.1. Illustration of Step 2 of Deterministic Two-step Algorithm for 3-axis Sensors

This illustration demonstrates, that this centering (i.e., subtraction of each column's mean from the corresponding column of $\mathbf{M}^j$), in effect, puts all the data, as closely as possible, on a plane that (a) is orthogonal to $\hat{\boldsymbol{m}}^j$, and (b) runs through the origin. And, if $\theta_j = 90°$, data would already be on such plane! (Although not necessarily centered about the origin, until the centering transformation). Therefore, whether or not θ_j is 90°, or some other value, by centering the data, the problem becomes the following:

$$\mathbf{M}_c^j \vec{\boldsymbol{m}}^j = \vec{\mathbf{0}} \in \mathbb{R}^{N_j}, \tag{3.50}$$

in which $\mathbf{M}_c^j$ is the centered version of $\mathbf{M}^j$.

This problem is known as the *"overdetermined homogeneous linear algebraic system."* It obviously has a trivial ($\vec{\mathbf{0}}$) solution, but also has nontrivial solutions, because it is rank-deficient (by construction, it has rank 2, not 3). To find the nontrivial solutions, we utilize the following Singular Value Decomposition (SVD), performed by MATLAB's **`svd`** function on a centered matrix like $\mathbf{M}_c^j$:

$$\mathbf{M}_c^j = \mathbf{P}_j \mathbf{\Sigma}_j \mathbf{Q}_j^{\mathrm{T}}, \tag{3.51}$$

in which $\mathbf{P}_j \in \mathbb{R}^{N_j \times N_j}$, $\mathbf{\Sigma}_j$ is a diagonal 3×3 matrix, and $\mathbf{Q}_j \in \mathbb{R}^{3\times 3}$. $\mathbf{P}_j$ and $\mathbf{Q}_j$ are both unitary (i.e., norm-preserving) matrices, with $\mathbf{Q}_j$'s columns consisting of the 3 right-singular vectors of $\mathbf{M}_c^j$. The 3 nonnegative values in $\mathbf{\Sigma}_j$ are the "singular values" of $\mathbf{M}_c^j$, arranged (along the diagonal) from the highest to the lowest. These values quantify the variance in the data along the right-singular vectors, with a higher singular value corresponding to greater variance.

As is obviously illustrated in Fig. 3.1 above, we are clearly interested in the singular vector with the least possible variance along it, which is aligned with $\overrightarrow{\boldsymbol{m}}^j$. Since the singular values are arranged in decreasing order in $\mathbf{\Sigma}_j$, and the columns of $\mathbf{Q}_j$ correspond to these values, we are clearly interested in the 3$^{\text{rd}}$ column of $\mathbf{Q}_j$, $\vec{\boldsymbol{q}}_3^j$.

The other two vectors in $\mathbf{Q}_j$ will lie in the plane in which the centered data points lie (or the best fit to such a plane, if there is small transverse variance due to noise and uncertainties). Their exact orientations will depend on how asymmetric (i.e., how unevenly distributed over the edge of the cone) the data is; see Fig. 3.1 for an illustration of a fairly asymmetric (180°) dataset.

All of the above details were either implied, or stated directly, by Bonnet et al. However, one point was not stated: as we can clearly see from Fig. 3.1 above, if $\vec{\boldsymbol{q}}_3^j$ is parallel to the vector along which the least variance in the data occurs, $-\vec{\boldsymbol{q}}_3^j$ will be antiparallel, but with exactly the same amount of variance. We therefore cannot automatically state that $\overrightarrow{\boldsymbol{m}}^j = \vec{\boldsymbol{q}}_3^j$: it can be shown, that the SVD, due to the uncertainty in data, can converge to $\vec{\boldsymbol{q}}_3^j = -\overrightarrow{\boldsymbol{m}}^j$. In fact, for example, MATLAB's `svd`, when executed on the (centered) data in Fig. 3.1 above, converges to $\vec{\boldsymbol{q}}_3^j = -\begin{bmatrix}\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}}\end{bmatrix}^{\mathrm{T}}$, which is clearly $-\overrightarrow{\boldsymbol{m}}^j$.

To determine the appropriate sign to apply to $\vec{\boldsymbol{q}}_3^j$, I propose the following consideration: we know, that misalignment angles (i.e., the 3 orientation correction angles that define $\mathbf{R}^{sb}$) between the orthogonalized sensor frame, and the body frame, are, for real commercial sensors, generally relatively small, on the order of a few degrees.

Therefore, we can use the following fact: even for a very poor, and very noisy, dataset, $\overrightarrow{\boldsymbol{m}}^j$ is going to be closer aligned to $\hat{\boldsymbol{e}}_j$, with $\hat{\boldsymbol{e}}_j$ as the unit vector in the direction of orthogonal sensor frame axis j. Therefore, to assess whether $\overrightarrow{\boldsymbol{m}}^j = \vec{\boldsymbol{q}}_3^j$ or $\overrightarrow{\boldsymbol{m}}^j = -\vec{\boldsymbol{q}}_3^j$, I propose the following:

$$\vec{\boldsymbol{m}}^j = \begin{cases} \vec{\boldsymbol{q}}_3^j & \text{if } \vec{\boldsymbol{q}}_3^j \cdot \hat{\boldsymbol{e}}_j \geq -\vec{\boldsymbol{q}}_3^j \cdot \hat{\boldsymbol{e}}_j, \\ -\vec{\boldsymbol{q}}_3^j & \text{otherwise.} \end{cases} \tag{3.52}$$

Beyond this point, the algorithm is once again, as specified by Bonnet, et al.

After $\vec{\boldsymbol{m}}^x$, $\vec{\boldsymbol{m}}^y$ and $\vec{\boldsymbol{m}}^z$ are all found, regardless of how this was accomplished (i.e., by using Eq. (3.48) with known, nonzero θ_j, or by using Eqs. (3.50)–(3.52)), the following matrix is constructed:

$$\mathbf{R} = [\vec{\boldsymbol{m}}^x \quad \vec{\boldsymbol{m}}^y \quad \vec{\boldsymbol{m}}^z]^{\mathrm{T}}, \tag{3.53}$$

and decomposed using SVD:

$$\mathbf{R} = \mathbf{P}\boldsymbol{\Sigma}\mathbf{Q}^{\mathrm{T}}, \tag{3.54}$$

from which $\mathbf{R}^{sb}$ can finally be obtained:

$$\mathbf{R}^{sb} = \mathbf{P}\mathbf{Q}^{\mathrm{T}}. \tag{3.55}$$

Lastly, with $\mathbf{B}$, $\vec{\boldsymbol{o}}$ and $\mathbf{R}^{sb}$ computed, we can finally compute $\mathbf{A}$:

$$\mathbf{A} = \mathbf{R}^{sb}\mathbf{B}. \tag{3.56}$$

In this work, I will use the more general technique for finding $\left\{\vec{\boldsymbol{m}}^x, \vec{\boldsymbol{m}}^y, \vec{\boldsymbol{m}}^z\right\}$, based on Eqs. (3.50)–(3.52). The deterministic two-step calibration algorithm for 3-axis accelerometers, due to Bonnet et al. [9], with my modifications, is therefore summarized as follows:

1. Accept as inputs: the voltage set $\left\{\vec{\boldsymbol{v}}\right\}$ of size N measurements from step 1 of the two-step accelerometer calibration experiment, and the voltage sets $\left\{\vec{\boldsymbol{v}}\right\}^x$, $\left\{\vec{\boldsymbol{v}}\right\}^y$ and $\left\{\vec{\boldsymbol{v}}\right\}^z$ of sizes N_x, N_y and N_z from step 2 of the two-step experiment. The following requirements apply: $N \geq 9$, and $N_j \geq 3$ for all $j = x, y, z$. Each element of each set must be a 3-dimensional voltage vector.
2. In MATLAB, construct the cost function $F\left(\mathbf{B}, \vec{\boldsymbol{o}}\right)$ from Eq. (3.40), using $m = 1g$ and data from step 1.
3. Set the initial guesses $\mathbf{B}^{(0)}$ and $\vec{\boldsymbol{o}}^{(0)}$, using the typical sensitivities k and offset values o specified in the sensor datasheet. Because most sensors are specified with a single sensitivity $k = s^{-1}$ for all axes, and 0 misalignment angles, the following initial guess for $\mathbf{B}^{(0)}$ is usually the best available:

$$\mathbf{B}^{(0)} = \begin{bmatrix} \frac{1}{k} & 0 & 0 \\ 0 & \frac{1}{k} & 0 \\ 0 & 0 & \frac{1}{k} \end{bmatrix}. \tag{3.57}$$

4. Use MATLAB Optimization Toolbox `fminunc` function to minimize $F\left(\mathbf{B}, \vec{o}\right)$, with the initial guess from step 3, yielding $\mathbf{B}$ and $\vec{o}$.
5. For each body axis $j = \{x, y, z\}$:
    a. Construct the coefficient matrix $\mathbf{M}^j$ according to Eq. (3.46), using the data from step 1, and $\mathbf{B}$ and $\vec{o}$ from step 4.
    b. Compute the means $\overline{M}_x^j$, $\overline{M}_y^j$ and $\overline{M}_z^j$ of each column of $\mathbf{M}^j$.
    c. Center the matrix $\mathbf{M}^j$ by subtracting the column means from their corresponding columns.
    d. Using MATLAB `svd` function, perform the SVD decomposition of Eq. (3.51).
    e. Extract $\vec{q}_3^j$ as the 3rd column of $\mathbf{Q}_j$.
    f. Select $\overline{m}^j$ (i.e., $\overline{m}^j = \vec{q}_3^j$ or $\overline{m}^j = -\vec{q}_3^j$) using Eq. (3.52).
6. Assemble $\mathbf{R}$ according to Eq. (3.53).
7. Perform SVD on $\mathbf{R}$ using MATLAB `svd` function, yielding the orthogonal unitary matrices $\mathbf{P}$ and $\mathbf{Q}$ (and diagonal matrix $\mathbf{\Sigma}$, which is not used), as per Eq. (3.54).
8. Compute rotation matrix $\mathbf{R}^{sb}$ using Eq. (3.55).
9. Compute $\mathbf{A}$ using Eq. (3.56).

This concludes the summary of the deterministic calibration algorithms for 1- and 3-axis sensors, including low-level details of their implementation. These algorithms, as was stated at the beginning of this chapter, do not quantify the sensors' uncertainties. In the following section, I propose Monte Carlo-based modifications, which give these algorithms uncertainty quantification capabilities.

3.2 Uncertainty-Quantifying Calibration Algorithms

By definition, an uncertain quantity is one that is represented by a probability density function (PDF), as opposed to a single best estimate [16]. As discussed in the introduction to this chapter, an uncertainty-quantifying calibration algorithm is one that accepts uncertain measured voltages and reference physical

values, and outputs a transducer function that also computes the uncertainties in the measured physical values.

Most generally, uncertainties are quantified via PDFs. However, in engineering and scientific practice, it is very common to assume a shape to the PDF, and quantify the uncertain parameter via two quantities: the PDF's mean (aka, the best estimate for the value), and some measure of the PDF's spread (e.g., the standard deviation, or the width of a confidence interval) [31]. In engineering in particular, the half-width of a 95% confidence interval (CI) is commonly used [6].

The normal distribution, also known as the *"Gaussian distribution,"* is the most commonly used form of PDF for uncertain parameters. The reason for this is the following: by the central limit theorem (CLT), "if a measurement is subject to many small sources of random error, and negligible systematic error, the measured values will be distributed in accordance with a bell-shape curve and this curve will be centered on the true value [of the measurement]" [31]. Here by "bell-shaped curve" Taylor refers to the Gaussian PDF. This means that if the uncertainty in a parameter is caused by several independent, small random effects, the parameter will be quantified by a Gaussian PDF.

In this work, both measured voltages and reference values will be treated as independent uncertain parameters, quantified with independent Gaussian PDFs. This is done for three reasons:

1. The source of the uncertainty in measured voltages is the signal noise, which tends to be modeled either as white noise (flat PDF) or as Gaussian noise (Gaussian PDF) [3]. Signals from sensors tend to have Gaussian noise, which makes this a good assumption.
2. Uncertainties in reference physical values depend on the experimental setup, and how the reference value was calculated. However, regardless of the shapes of the PDFs in the individual quantities that contribute to the true physical value experienced by the sensor, as long as the uncertainties in these individual quantities are small, the PDF for the reference physical value will be a Gaussian, centered on the best estimate. This is by the CLT, as described above. Also, by design of experiments, we can clearly assume that the errors in the individual contributing quantities (e.g., the orientations of the sensors, the noise in the nonzero voltages, etc.) are relatively small.
3. Assuming the uncertainties in measured voltages and in reference physical values to be independent from each other also significantly simplifies the analysis (specifically, the PDFs of the calibration parameters), so partly this assumption is for convenience. For measured voltages, this is also a good assumption simply because the 3 analog channels of the DAQ are completely independent from

each other, and their analog-to-digital conversion (ADC) is the primary source of the voltage uncertainties. For reference physical values, this assumption may not necessarily be a good one, but in the calibration experiments used in this work (chapters 4 and 5), it is.

The Gaussian PDF for an uncertain parameter v is given by [31]:

$$\mathcal{N}(v|\mu_v, \sigma_v) = \frac{1}{\sigma_v\sqrt{2\pi}} e^{-\frac{(v-\mu_v)^2}{2\sigma_v^2}}, \tag{3.58}$$

where μ_v and σ_v are the PDF's mean (also the best estimate for v) and standard deviation. It is a PDF, which means that the probability $P(v_L \leq v \leq v_H)$ of sampling v between v_L and v_H is given by:

$$P(v_L \leq v \leq v_H) = \int_{v_L}^{v_H} dv'\, \mathcal{N}(v'|\mu_v, \sigma_v). \tag{3.59}$$

As is discussed above, the half-width of the 95% confidence interval, in engineering practice, is normally used instead of the standard deviation. The two are linearly proportional to each other. To convert between the two, we use the normal distribution quantile function $\Phi^{-1}(p|\mu_v, \sigma_v)$, which is defined as the inverse of the normal distribution's cumulative distribution function (CDF). This means that for a given probability p, $\Phi^{-1}(p|\mu_v, \sigma_v)$ gives the value v_H such that:

$$P(-\infty \leq v \leq v_H) = p. \tag{3.60}$$

In other words, the fraction p of the area under the curve $\mathcal{N}(v|\mu_v, \sigma_v)$ is between $-\infty$ and $\Phi^{-1}(p|\mu_v, \sigma_v)$. $\Phi^{-1}(p|\mu_v, \sigma_v)$ is given by:

$$\Phi^{-1}(p|\mu_v, \sigma_v) = \mu_v + \sigma_v\sqrt{2}\mathrm{erf}^{-1}(2p-1), \tag{3.61}$$

where $\mathrm{erf}^{-1}(x)$ is the inverse error function. By definition, the 95% confidence interval is the interval $\left[v_L, v_H\right]$ such that $P(v_L \leq v \leq v_H) = 95\%$. Equation (3.58) is symmetric about μ_v, therefore the CI's half-width, δ_v, is enough to quantify the width of the CI, yielding $v_L = \mu_v - \delta_v$ and $v_H = \mu_v + \delta_v$. Again from symmetricity, the corresponding p values for these v_L and v_H values are 2.5% and 97.5%, respectively. This allows us to relate δ_v and σ_v. Substituting $p = 97.5\%$ and $v_H = \mu_v + \delta_v$ into Eq. (3.61) yields:

$$\mu_v + \delta_v = \mu_v + \sigma_v\sqrt{2}\text{erf}^{-1}(2 \cdot 0.975 - 1), \tag{3.62}$$

which yields the ratio:

$$\delta_v \approx 1.9600\sigma_v. \tag{3.63}$$

Below, an uncertain quantity v with a best estimate μ_v and a 95% CI half-width δ_v will be denoted:

$$v = \mu_v \pm \delta_v. \tag{3.64}$$

In summary: in this work, all quantities that serve as inputs to the uncertainty-quantifying calibration algorithms are quantified using independent Gaussian PDFs, with means and 95% confidence interval half-widths. The 95% confidence internal half-widths are referred to as these quantities' *"uncertainties,"* for brevity.

The Monte Carlo-based uncertainty-quantifying modification to the deterministic algorithm for 1-axis sensors is presented in subsection 3.2.1 below. Similar modifications for the one-step and two-step 3-axis deterministic algorithms are in subsections 3.2.2 and 3.2.3, respectively.

3.2.1 *Monte Carlo Algorithm for 1-axis Sensors*

As discussed in subsection 3.1.1, the deterministic calibration algorithm for 1-axis sensors takes the N voltages $\{v\}$ and the N reference physical values $\{p\}$ as inputs. The uncertainty-quantifying version of this algorithm also requires the corresponding sets of uncertainties $\{\delta v\}$ and $\{\delta p\}$, for the voltages and reference physical values, respectively. Both $\{\delta v\}$ and $\{\delta p\}$ are also of size N each.

There exist well-defined algebraic rules for propagating uncertainties in independent random variables through closed-form expressions, summarized in section 2.4. However, the algorithm presented at the end of subsection 3.1.1 (both **`fminsearch`**- and **`mldivide`**-based approaches) is not a closed-form expression. Therefore, uncertainties in $\{v\}$ and $\{p\}$ cannot be easily propagated through it using the above-mentioned uncertainty propagation rules.

The most general method for propagating uncertainties through arbitrary nonlinear functions is Monte Carlo-based uncertainty quantification [43]. The 1-axis deterministic calibration algorithm can be viewed as such nonlinear function. In general, it works as follows:

1. Sample every uncertain quantity that serves as an input to the nonlinear function, according to their PDFs.
2. Evaluate the nonlinear function using the results of the sampling from step 1, and record the function's resulting value.
3. Repeat steps 1 and 2, enough times to get the function's value to approach a limiting distribution. Assuming the uncertainties are small and independent, this distribution will be Gaussian, as discussed at the beginning of section 3.2.

With enough samples, this approach will work on any function and input quantities' PDFs. However, a very large number of samples can be required, which makes this method potentially slow. It is therefore important that:

a) the number of input quantities is sufficiently low,
b) the function is sufficiently smooth, and
c) the function can be quickly evaluated, since it needs to be evaluated once per iteration.

Fortunately, our problem meets these 3 criteria:

a) N will generally not be above a few hundred, as it's prohibitively time-consuming to construct a larger dataset for calibration.
b) Eq. (3.6) is a sum of 2^{nd}-order polynomials, so it is smooth.
c) As will be shown in chapter 4, a single pass-through of the 1-axis deterministic calibration algorithm is sufficiently rapid.

To quantify the uncertainties in 1-axis sensor calibration parameters, I therefore propose the following uncertainty-quantifying Monte Carlo calibration algorithm. Recall, that it uses all of the assumptions discussed in section 3.2:

1. Accept the voltage best estimate set $\{v\}$, the voltage uncertainties set $\{\delta v\}$, the reference physical value best estimate set $\{p\}$, the reference physical value uncertainties set $\{\delta p\}$, and the number of simulations K, as inputs. The sets must be of the same size $N \geq 2$. As shown in chapter 4, taking $K = 100{,}000$ with realistic data tends to be enough for convergence.
2. Seed the random number generator that will be used throughout the program. This step is technically optional, but is recommended, to ensure complete reproducibility of results.
3. Initialize the simulation counter k to 1.

4. For each $i = 1 \dots N$, use Eq. (3.63) to convert from the 95% CI half-widths to corresponding standard deviations.
5. For each $i = 1 \dots N$, sample $v_i \pm \delta v_i$ and $p_i \pm \delta p_i$. To sample a Gaussian PDF with mean 0 and standard deviation 1 (known as *"standard normal distribution"*), a variety of high-performance random number generators exist, such as MATLAB's **`randn`** function. Let such (normalized) sample for the variable v_i, in simulation k, be denoted $\tilde{v}_i^{(k)}$. To convert from a standard normal distribution sample to a Gaussian sample with mean μ_{v_i} and standard deviation σ_{v_i}, the following expression may be used:

$$v_i^{(k)} = \mu_{v_i} + \sigma_{v_i} \tilde{v}_i^{(k)}. \tag{3.65}$$

6. Apply the deterministic calibration algorithm for 1-axis sensor (subsection 3.1.1) to the sampled voltage set $\left\{v^{(k)}\right\} = \left\{v_1^{(k)} \dots v_N^{(k)}\right\}$ and the sampled reference physical value set $\left\{p^{(k)}\right\} = \left\{p_1^{(k)} \dots p_N^{(k)}\right\}$. This will yield the sensor inverse sensitivity $s^{(k)}$ and offset $o^{(k)}$ for simulation k. Record them, and discard $\left\{v^{(k)}\right\}$ and $\left\{p^{(k)}\right\}$.
7. Increment simulation counter k by 1, and go back to step 5 above. Repeat steps 5–7 until running K simulations.
8. With K large enough, and $\{\delta v\}$ and $\{\delta p\}$ small enough, the distributions of $\left\{s^{(k)}\right\}$ and $\left\{o^{(k)}\right\}$ should converge to the limiting Gaussian distributions. At this stage, $s \pm \delta s$ and $o \pm \delta o$, which constitute the outputs of the algorithm, can be computed. The best estimates are computed by calculating the means of the corresponding distributions, and the 95% CI half-widths can be computed as follows (for δs):
    a. Evaluate the 2.5% percentile of $\left\{s^{(k)}\right\}$, denoting it s_L.
    b. Evaluate the 97.5% percentile of $\left\{s^{(k)}\right\}$, denoting it s_H.
    c. Evaluate δs as:

$$\delta s = \frac{s_H - s_L}{2}. \tag{3.66}$$

To compute percentiles, MATLAB Statistics and Machine Learning Toolbox function **`prctile`** can be used.

Note, that step 6 in this algorithm will work regardless of the approach used in the deterministic 1-axis algorithm. However, this step is also run K times, for a large K, and so it is preferable to use the fastest approach available, which, as is discussed in subsection 3.1.1, is the **`mldivide`**-based approach.
The 3-axis version of the above algorithm is presented in the following subsection.

3.2.2 *Monte Carlo One-step Algorithm for 3-axis Sensors*

As discussed in subsection 3.1.2, the one-step deterministic calibration algorithm for 3-axis sensors takes the N voltage vectors $\{\vec{\boldsymbol{v}}\}$ and the N reference physical values $\{\vec{\boldsymbol{p}}\}$ as inputs. The uncertainty-quantifying version of this algorithm also requires the corresponding sets of uncertainties $\{\delta\vec{\boldsymbol{v}}\}$ and $\{\delta\vec{\boldsymbol{p}}\}$, for the voltages and reference physical values, respectively. Both $\{\delta\vec{\boldsymbol{v}}\}$ and $\{\delta\vec{\boldsymbol{p}}\}$ are also of size N each.
By the assumptions described at the beginning of section 3.2, a single voltage measurement vector $\vec{\boldsymbol{v}}_i$, and a single reference physical value vector $\vec{\boldsymbol{p}}_i$, are treated in this work as triplets of independent uncertain variables each. This means that in a Monte Carlo simulation, v_{ix}, v_{iy} and v_{iz} can be sampled independently from each other; same with p_{ix}, p_{iy} and p_{iz}.
Similarly, the outputs, which consist of 12 variables (9 elements of $\mathbf{A}$, 3 of $\vec{\boldsymbol{o}}$), are also treated as independent uncertain variables.
Under these assumptions, to quantify the uncertainties in 3-axis sensor calibration parameters, I therefore propose the following uncertainty-quantifying Monte Carlo calibration algorithm:

1. Accept the voltage best estimate set $\{\vec{\boldsymbol{v}}\}$, the voltage uncertainties set $\{\delta\vec{\boldsymbol{v}}\}$, the reference physical value best estimate set $\{\vec{\boldsymbol{p}}\}$, the reference physical value uncertainties set $\{\delta\vec{\boldsymbol{p}}\}$, and the number of simulations K, as inputs. The sets must be of the same size $N \geq 4$. As shown in chapter 4, taking $K = 100{,}000$ with realistic data tends to be enough for convergence.
2. Seed the random number generator that will be used throughout the program. This step is technically optional, but is recommended, to ensure complete reproducibility of results.
3. Initialize the simulation counter k to 1.
4. For each $i = 1 \ldots N$, use Eq. (3.63) to convert from the 95% CI half-widths to corresponding standard deviations. Note, that each vector has 3 components, and each component has an associated uncertainty (i.e., δv_{ix}, δv_{iy} and δv_{iz} for $\vec{\boldsymbol{v}}_i$, and δp_{ix}, δp_{iy} and δp_{iz} for $\vec{\boldsymbol{p}}_i$), and all of them must be converted.

5. For each $i = 1 \dots N$, and each axis $j = x, y, z$, sample $v_{ij} \pm \delta v_{ij}$ and $p_{ij} \pm \delta p_{ij}$. See step 5 of the algorithm in subsection 3.2.1 for details on how to sample a Gaussian PDF for each of these variables. This will yield the dataset for simulation k: $\left(\left\{\vec{\boldsymbol{v}}^{(k)}\right\}, \left\{\vec{\boldsymbol{p}}^{(k)}\right\}\right)$.
6. Apply the one-step deterministic calibration algorithm for 3-axis sensor (subsection 3.1.2) to the sampled voltage set $\left\{\vec{\boldsymbol{v}}^{(k)}\right\} = \left\{\vec{\boldsymbol{v}}_1^{(k)} \dots \vec{\boldsymbol{v}}_N^{(k)}\right\}$ and the sampled reference physical value set $\left\{\vec{\boldsymbol{p}}^{(k)}\right\} = \left\{\vec{\boldsymbol{p}}_1^{(k)} \dots \vec{\boldsymbol{p}}_N^{(k)}\right\}$. This will yield the sensor coefficient matrix $\mathbf{A}^{(k)}$ and offset vector $\vec{\boldsymbol{o}}^{(k)}$ for simulation k. Record them, and discard $\left\{\vec{\boldsymbol{v}}^{(k)}\right\}$ and $\left\{\vec{\boldsymbol{p}}^{(k)}\right\}$.
7. Increment the simulation counter k by 1, and go back to step 5 above. Repeat steps 5–7 until running K simulations.
8. With K large enough, and $\left\{\delta\vec{\boldsymbol{v}}\right\}$ and $\left\{\delta\vec{\boldsymbol{p}}\right\}$ small enough, the distributions of each element of $\mathbf{A}^{(k)}$ and $\vec{\boldsymbol{o}}^{(k)}$ should converge to the limiting Gaussian distributions. At this stage, $\mathbf{A} \pm \delta\mathbf{A}$ and $\vec{\boldsymbol{o}} \pm \delta\vec{\boldsymbol{o}}$, which constitute the outputs of the algorithm, can be computed. The best estimates are computed by calculating the means of the corresponding distributions, for each element. The 95% CI half-widths can be computed the same way they are in step 8 of the algorithm in subsection 3.2.1: by computing the 2.5% and 97.5% percentiles for each element of the output, and dividing their difference by 2.

As with the 1-axis Monte Carlo uncertainty-quantifying calibration algorithm, step 6 in this algorithm will work regardless of the approach used in the one-step deterministic 3-axis algorithm. This step is also run K times, for a large K, and so it is preferable to use the fastest approach available, which, as is discussed in subsection 3.1.2, is the **`mldivide`**-based approach.

The two-step Monte Carlo algorithm for 3-axis sensors, which is reliant on data from method 4 from section 2.2, is described in the following subsection.

3.2.3 *Monte Carlo Two-step Algorithm for 3-axis Sensors*

As discussed in subsection 3.1.3, the two-step deterministic calibration algorithm for 3-axis sensors takes two sets of inputs: the set of N voltage vectors $\left\{\vec{\boldsymbol{v}}\right\}$ from step 1 of the two-step accelerometer calibration experiment, and the voltage sets $\left\{\vec{\boldsymbol{v}}\right\}^x$, $\left\{\vec{\boldsymbol{v}}\right\}^y$ and $\left\{\vec{\boldsymbol{v}}\right\}^z$ of sizes N_x, N_y and N_z from step 2 of the two-step experiment. The uncertainty-quantifying version of this algorithm also requires the corresponding sets of

uncertainties $\{\delta\vec{\boldsymbol{v}}\}$, $\{\delta\vec{\boldsymbol{v}}\}^x$, $\{\delta\vec{\boldsymbol{v}}\}^y$ and $\{\delta\vec{\boldsymbol{v}}\}^z$, also of sizes N, N_x, N_y and N_z, respectively. Additionally, it requires orientation uncertainty angles $\delta\theta_x$, $\delta\theta_y$ and $\delta\theta_z$, to quantify the potential orientation error in step 2 of the two-step calibration method.

By the assumptions described at the beginning of section 3.2, any single voltage measurement vector $\vec{\boldsymbol{v}}_i$ is treated in this work as a triplet of independent uncertain variables. This means that in a Monte Carlo simulation, v_{ix}, v_{iy} and v_{iz} can be sampled independently from each other. This applies to both the voltages from step 1, and from step 2.

Similarly, the outputs, which consist of 12 variables (9 elements of $\mathbf{A}$, 3 of $\vec{\boldsymbol{o}}$), are also treated as independent uncertain variables.

The uncertainty sets $\{\delta\vec{\boldsymbol{v}}\}$, $\{\delta\vec{\boldsymbol{v}}\}^x$, $\{\delta\vec{\boldsymbol{v}}\}^y$ and $\{\delta\vec{\boldsymbol{v}}\}^z$ account for the potential errors due to noise in the voltage signals. For step 1, there actually is no other possible contribution to uncertainty, assuming that it is known that gravitational acceleration (or ambient magnetic field, for 3-axis magnetometers) stays completely fixed, which generally is the case in a stationary laboratory. However, in step 2, besides the uncertainties due to voltage noises, there is the added uncertainty due to the orientation error of the gravity vector, relative to the axis of rotation. Specifically: while rotating the sensor about body axis j, it is not necessary to know what the angle between the rotation body axis j, and the vertical, is, but it is necessary to keep it constant. The goal of the uncertainty-quantifying Monte Carlo calibration algorithm in this section is therefore to account not only for the effects of voltage noise on the calibration parameters, but also for the orientation errors during the 3 rotation substeps.

When rotating about a body axis j, a number of measurements $\vec{\boldsymbol{v}}_i^j$ are made. The product $\vec{\boldsymbol{p}}_i^{sj} = \mathbf{B}\left(\vec{\boldsymbol{v}}_i^j - \vec{\boldsymbol{o}}\right)$ yields the corresponding acceleration in the orthogonalized sensor frame. These products comprise the columns of $\mathbf{M}^j$, as per Eq. (3.45). An orientation uncertainty associated with this vector simply means that, in a given Monte Carlo simulation, the vector should be kept the same length ($1g$), but be sampled from a cone of possible directions, centered at $\vec{\boldsymbol{p}}_i^{sj}$, and with a half-aperture of $\delta\theta_j$.

To sample a cone like this, we can use spherical coordinates. First, we sample the zenith angle (i.e., the polar angle, the angle between the vector and the z-axis), which here plays the role of the half-aperture angle. By the above assumptions, it is described by a Gaussian PDF with mean 0 and 95% CI half-width $\delta\theta_j$. Next, we sample the azimuth angle (the angle between the x-axis and the projection of the vector onto the xy-plane, in the right-hand sense about the z-axis), which is described by a uniform PDF between 0 and 360°.

Lastly, we have to rotate the resulting sampled vector, such that the z-axis that it was just sampled about, becomes aligned with $\vec{\boldsymbol{p}}_i^{sj}$. The mathematics for this rotation can get somewhat complicated, but they are efficiently handled by MATLAB Simulink 3D Animation functions **`vrrotvec`** and **`vrrotvec2mat`**. **`vrrotvec`** provides an (axis, angle) pair that specify a rotation, that would transform one given vector to another. **`vrrotvec2mat`** converts this (axis, angle) pair to a 3×3 rotation matrix. The two functions robustly handle special cases (i.e., when $\vec{\boldsymbol{p}}_i^{sj} \cdot \hat{\boldsymbol{e}}_z = 0$, meaning $\vec{\boldsymbol{p}}_i^{sj}$ is parallel or antiparallel to z-axis), and they will be used in this work.

In short, after sampling a vector $\vec{\boldsymbol{u}}_i^{sj,(k)}$ about the z-axis, as described above, scale it by the length of $\vec{\boldsymbol{p}}_i^{sj}$ (should be $1g$, but may be slightly different, due to the uncertainties in step 1), construct a rotation matrix from $\hat{\boldsymbol{e}}_z$ to $\vec{\boldsymbol{p}}_i^{sj}$ and apply it to $\vec{\boldsymbol{u}}_i^{sj,(k)}$, to obtain the sampled $\vec{\boldsymbol{p}}_i^{sj,(k)}$ for simulation k. This vector will have a 95% probability to be directed within $\delta\theta_j$ of $\vec{\boldsymbol{p}}_i^{sj} = \mathbf{B}\left(\vec{\boldsymbol{v}}_i^{j} - \vec{\boldsymbol{o}}\right)$, and it will have the same length.

The rest of the procedure is similar to the one in the previous subsections: Monte Carlo sampling is used to perturb each row of $\mathbf{M}^j$ for each j, followed by the application of the deterministic calibration algorithm of the second step (subsection 3.1.3).

Under the above assumptions, and using the above method for accounting for orientation uncertainty in each measurement of step 2, to quantify the uncertainties in 3-axis sensor calibration parameters, I therefore propose the following uncertainty-quantifying Monte Carlo calibration algorithm:

1. Accept as inputs: the voltage best estimate set $\left\{\vec{\boldsymbol{v}}\right\}$, the voltage uncertainties set $\left\{\delta\vec{\boldsymbol{v}}\right\}$, the voltage best estimate sets $\left\{\vec{\boldsymbol{v}}\right\}^x$, $\left\{\vec{\boldsymbol{v}}\right\}^y$ and $\left\{\vec{\boldsymbol{v}}\right\}^z$, the voltage uncertainties sets $\left\{\delta\vec{\boldsymbol{v}}\right\}^x$, $\left\{\delta\vec{\boldsymbol{v}}\right\}^y$ and $\left\{\delta\vec{\boldsymbol{v}}\right\}^z$, the orientation uncertainty triplet $\left\{\delta\theta_x, \delta\theta_y, \delta\theta_z\right\}$ and the number of simulations K. $\left\{\vec{\boldsymbol{v}}\right\}$ and $\left\{\delta\vec{\boldsymbol{v}}\right\}$ come from step 1 of the two-step accelerometer calibration experiment, and are of size N each. $\left(\left\{\vec{\boldsymbol{v}}\right\}^x, \left\{\delta\vec{\boldsymbol{v}}\right\}^x\right)$, $\left(\left\{\vec{\boldsymbol{v}}\right\}^y, \left\{\delta\vec{\boldsymbol{v}}\right\}^y\right)$ and $\left(\left\{\vec{\boldsymbol{v}}\right\}^z, \left\{\delta\vec{\boldsymbol{v}}\right\}^z\right)$ come from step 2 of the two-step accelerometer calibration experiment, and are of sizes N_x, N_y and N_z. $\left\{\delta\theta_x, \delta\theta_y, \delta\theta_z\right\}$ also characterize step 2 of the calibration experiment. The following requirements apply: $N \geq 9$, and $N_j \geq 3$ for all $j = x, y, z$. Each element of each set must be a 3-dimensional voltage vector. Lastly, $K = 100{,}000$ with realistic data tends to be enough for convergence.
2. Seed the random number generator that will be used throughout the program. This step is technically optional, but is recommended, to ensure complete reproducibility of results.
3. Initialize the simulation counter k to 1.

4. For each $i = 1 \dots N$, use Eq. (3.63) to convert from the 95% CI half-widths to corresponding standard deviations. Note, that each vector has 3 components, and each component has an associated uncertainty, and all of them must be converted.
5. For each $i = 1 \dots N$, and each axis $j = x, y, z$, sample $v_{ij} \pm \delta v_{ij}$ from the $\left(\left\{\vec{\boldsymbol{v}}\right\}, \left\{\delta\vec{\boldsymbol{v}}\right\}\right)$ dataset. See step 5 of the algorithm in subsection 3.2.1 for details on how to sample a Gaussian PDF for each of these variables. This will yield the dataset $\left\{\vec{\boldsymbol{v}}^{(k)}\right\}$ for step 1 of simulation k.
6. Apply steps 2–4 of the two-step deterministic calibration algorithm for 3-axis sensor (subsection 3.1.3) to the sampled voltage set $\left\{\vec{\boldsymbol{v}}^{(k)}\right\} = \left\{\vec{\boldsymbol{v}}_1^{(k)} \dots \vec{\boldsymbol{v}}_N^{(k)}\right\}$. This will yield the sensitivity and orthogonalization upper triangular matrix $\mathbf{B}^{(k)}$ and offset vector $\vec{\boldsymbol{o}}^{(k)}$ for simulation k.
7. For each body axis $j = \{x, y, z\}$:
    a. For each $i = 1 \dots N_j$, and each axis $l = x, y, z$, sample $v_{il}^j \pm \delta v_{il}^j$ from the $\left(\left\{\vec{\boldsymbol{v}}\right\}^j, \left\{\delta\vec{\boldsymbol{v}}\right\}^j\right)$ dataset. This will yield the dataset $\left\{\vec{\boldsymbol{v}}\right\}^{j,(k)}$ for body axis j of step 2 of simulation k.
    b. For each $i = 1 \dots N_j$, using $\mathbf{B}^{(k)}$ and $\vec{\boldsymbol{o}}^{(k)}$ from step 6 above, compute $\vec{\boldsymbol{t}}_i^{sj,(k)} = \mathbf{B}^{(k)}\left(\vec{\boldsymbol{v}}_i^{j,(k)} - \vec{\boldsymbol{o}}^{(k)}\right)$.
    c. For each $i = 1 \dots N_j$, sample a Gaussian PDF to obtain the zenith angle set $\left\{\theta_z^{(k)}\right\}^j = \left\{\theta_{z1}^{j,(k)}, \dots, \theta_{zN_j}^{j,(k)}\right\}$. Here $\theta_{zi}^{j,(k)}$ is the angle by which $\vec{\boldsymbol{t}}_i^{sj,(k)}$ will be perturbed, and it is sampled from a Gaussian PDF with mean 0 and 95% CI half-width $\delta\theta_j$.
    d. Similarly, for each $i = 1 \dots N_j$, sample a uniform PDF from 0 to 360° to obtain the azimuth angle set $\left\{\varphi^{(k)}\right\}^j = \left\{\varphi_1^{j,(k)}, \dots, \varphi_{N_j}^{j,(k)}\right\}$. Here $\varphi_i^{j,(k)}$ is the azimuthal angle that decides the direction in which $\vec{\boldsymbol{t}}_i^{sj,(k)}$ will be perturbed. MATLAB function **`rand`** samples uniformly from 0 to 1, so to sample from 0 to 360°, simply multiple the outcome by 360 (or by 2π, if a result in radians is desired).
    e. For each $i = 1 \dots N_j$, construct a vector $\vec{\boldsymbol{u}}_i^{sj,(k)}$:

$$\vec{\boldsymbol{u}}_i^{sj,(k)} = \left\|\vec{\boldsymbol{t}}_i^{sj,(k)}\right\| \begin{bmatrix} \sin\left(\theta_{zi}^{j,(k)}\right)\cos\left(\varphi_i^{j,(k)}\right) \\ \sin\left(\theta_{zi}^{j,(k)}\right)\sin\left(\varphi_i^{j,(k)}\right) \\ \cos\left(\theta_{zi}^{j,(k)}\right) \end{bmatrix}. \tag{3.67}$$

f. For each $i = 1 \dots N_j$, supply $\hat{\boldsymbol{e}}_z$ and $\vec{\boldsymbol{t}}_i^{sj,(k)}$ to MATLAB functions **`vrrotvec`** and **`vrrotvec2mat`**, to obtain the rotation matrix $\mathbf{R}_i^{j,(k)}$ from the z-axis (in orthogonalized sensor frame) to the $\vec{\boldsymbol{t}}_i^{sj,(k)}$ vector.

g. For each $i = 1 \dots N_j$, compute the sampled orthogonalized sensor frame gravity vectors $\vec{\boldsymbol{p}}_i^{sj,(k)}$:

$$\vec{\boldsymbol{p}}_i^{sj,(k)} = \mathbf{R}_i^{j,(k)} \, \vec{\boldsymbol{u}}_i^{sj,(k)}. \tag{3.68}$$

This will yield finally yield the orthogonalized sensor frame gravity vector set $\left\{\vec{\boldsymbol{p}}^{s,(k)}\right\}^j = \left\{\vec{\boldsymbol{p}}_1^{sj,(k)}, \dots, \vec{\boldsymbol{p}}_{N_j}^{sj,(k)}\right\}$.

h. Construct the coefficient matrix $\mathbf{M}^{j,(k)}$, by replacing the rows in Eq. (3.46) with $\left\{\vec{\boldsymbol{p}}^{s,(k)}\right\}^j$:

$$\mathbf{M}^{j,(k)} = \begin{bmatrix} \left(\vec{\boldsymbol{p}}_1^{sj,(k)}\right)^{\mathrm{T}} \\ \vdots \\ \left(\vec{\boldsymbol{p}}_{N_j}^{sj,(k)}\right)^{\mathrm{T}} \end{bmatrix}. \tag{3.69}$$

i. Apply steps 5.b–5.f of the two-step deterministic calibration algorithm for 3-axis sensor (subsection 3.1.3) to $\mathbf{M}^{j,(k)}$, to obtain $\overline{\boldsymbol{m}}^{j,(k)}$.

8. Using $\overline{\boldsymbol{m}}^{x,(k)}$, $\overline{\boldsymbol{m}}^{y,(k)}$ and $\overline{\boldsymbol{m}}^{z,(k)}$ obtained in step 7.i above, and $\mathbf{B}^{(k)}$ from step 6 above, apply steps 6–9 of the two-step deterministic calibration algorithm for 3-axis sensor (subsection 3.1.3) to finally obtain the full calibration matrix $\mathbf{A}^{(k)}$ for simulation k. Record it, and $\vec{\boldsymbol{o}}^{(k)}$.

9. Increment the simulation counter k by 1, and go back to step 5 above. Repeat steps 5–8 until running K simulations.

10. With K large enough, and the uncertainties small enough, the distributions of each element of $\mathbf{A}^{(k)}$ and $\vec{\boldsymbol{o}}^{(k)}$ should converge to the limiting Gaussian distributions. At this stage, $\mathbf{A} \pm \delta\mathbf{A}$ and $\vec{\boldsymbol{o}} \pm \delta\vec{\boldsymbol{o}}$, which constitute the outputs of the algorithm, can be computed. The best estimates are computed by calculating the means of the corresponding distributions, for each element. The 95% CI half-widths can be computed the same way they are in step 8 of the algorithm in subsection 3.2.1: by computing the 2.5% and 97.5% percentiles for each element of the output, and dividing their difference by 2.

This concludes the summary of the uncertainty-quantifying calibration algorithms, developed as part of this work. These algorithms are used in chapters 4 and 5 to process the calibration datasets gathered through several different calibration experiments, for 3-axis accelerometers and 1-axis gyroscopes.

Chapter 4
Accelerometer Calibration via "Box" Experiments

This chapter presents the results of the two "Box" experiments for accelerometer calibration: one-step and two-step methods. Its purpose is two-fold:

1. To study the performance of the calibration algorithms from chapter 3, and
2. To produce calibration parameter sets for a pair of MEMS accelerometers.

Section 4.1 details the equipment used, and describes the experiments. Section 4.2 lists the results from the one-step "Box" experiment, and uses them to analyze the performance of various aspects of the algorithms from chapter 3. The computed calibration parameters for the two accelerometers from the one-step "Box" experiment are also presented. Section 4.3 presents and discusses the results from the two-step "Box" calibration experiment. The conclusions drawn from both "Box" experiments are summarized in section 4.4.

4.1 Experimental Setup and Descriptions of Experiments

4.1.1 *Experimental Setup*

Experimental setup for "Box" experiments includes:

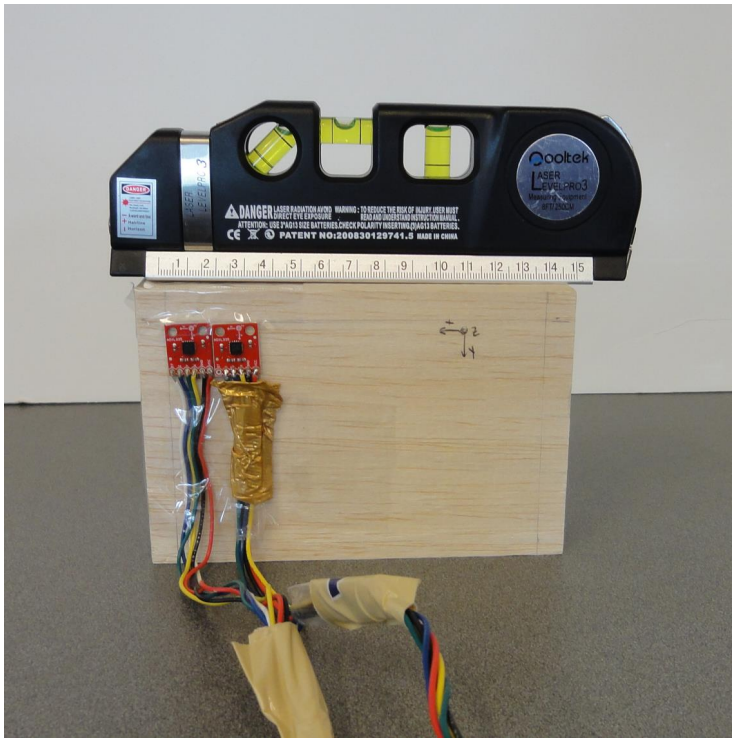

Fig. 4.1. Accelerometers mounted on the rectangular parallelepiped wooden "Box"

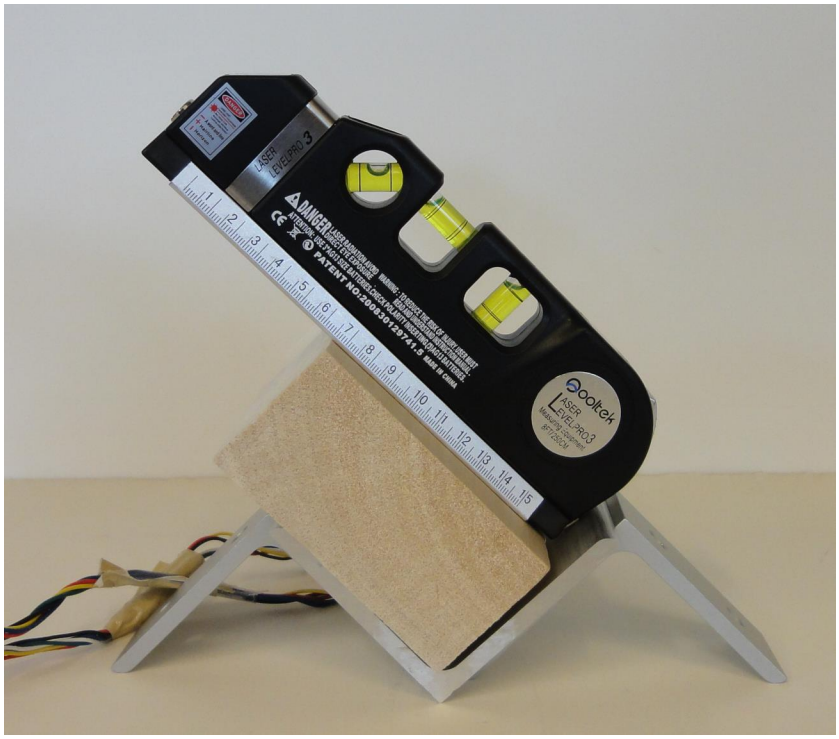

Fig. 4.2. "Box" experiment in a 45° orientation, supported by steel angles

- Rectangular parallelepiped wooden block, aka the "Box" after which this chapter is named.
- Professional bubble level tool, Qooltek LASER LEVELPRO3.
- Benchtop data acquisition device, National Instruments (NI) DAQ 6341.
- Two analog accelerometers, Analog Devices ADXL335.
- A desktop personal computer (PC).

All experimental equipment used in this work, including the equipment listed above, is detailed in Appendix A. During both "Box" experiments, the accelerometers are attached to the wooden block, such that the xy-axes (body frame) are parallel to the sides of the wooden block. Figure 4.1 above illustrates how the accelerometers are mounted on the box.

To read the data from the accelerometers in each orientation, the two accelerometers are plugged into the analog ports of the NI DAQ 6341. The accelerometers are powered by the DAQ via a voltage divider due to lack of voltage regulator on accelerometers' breakout. The DAQ is connected to a desktop PC through a USB 2.0 port.

I developed a special LabVIEW Virtual Instrument (VI) to sample the analog data from the accelerometers via the DAQ, plot it, optionally filter it, and record it in plain text into an `.lvm` file. The data can then be processed in MATLAB, Microsoft Excel, Python, or another application; all data processing in this thesis was performed using MATLAB.

Because the "Box" experiment uses only static positions, for each measurement, the box with the accelerometers was held stationary for 1 minute, with the DAQ and LabVIEW VI continuously recording the data at 1 kHz.

The other details experimental details varied depending on the experiment. The following two subsections present these details.

4.1.2 *One-step "Box" Experiment Description*

In this section, the one-step "Box" experiment, which uses the general principles listed as method 3 ("one-step static calibration") in section 2.2, is described.

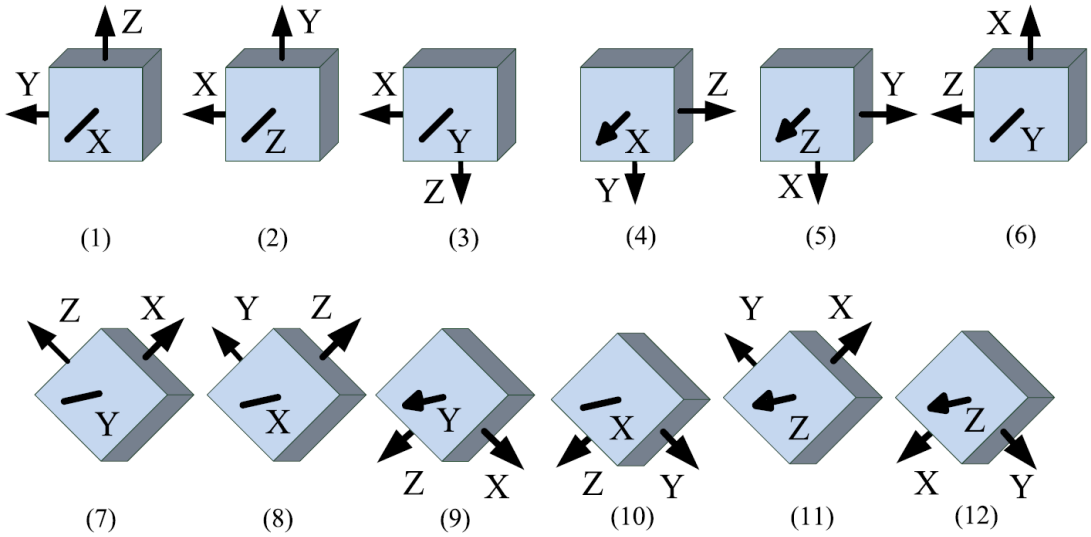

Fig. 4.3. 12 orientations of the accelerometers for one-step "Box" experiment
(from Ref. [7])

In the one-step "Box" experiment, the accelerometers are simply subjected to 12 known (within an uncertainty) orientations, illustrated on Fig. 4.3 above. These orientations were recommended by Zhang, et al. [7].

Unlike for step 1 of the two-step "Box" experiment, in the one-step version, precise orientation is very important, because not only the magnitudes, but also the orientations, are used for calibration, as described in subsections 3.1.2 and 3.2.2. To ensure precise orientation, in every one of the 12 positions, the Qooltek LASER LEVELPRO3 bubble level tool is used to verify that the box is correctly positioned. If needed, the level of the box is corrected by placing index cards under the box. It is important to use a level tool, or some other similarly precise instrument, for the one-step "Box" experiment, because otherwise the uneven surface of the table or the floor can introduce additional errors in gravity orientations.

In the last 6 orientations in Fig. 4.3, the accelerometers (and therefore the box) are placed at 45º angles to the vertical. As is shown in Fig. 4.2 above, steel angles were used for such orientations. Also, as seen in Fig. 4.2, the bubble level tool was equipped to provide 45º incline as well, adjusted, once again, by placing index cards under the steel angles.

In this experiment, the uncertainties in the reference values arise from orientation errors, and the uncertainties in voltage from the Gaussian noise. In the first 6 of the 12 orientations, one of the accelerometer's axes is vertical, and the other two are horizontal. In this situation, any horizontal orientation error does not affect

the reference value, but a tilt from the vertical does. By the specification of the LEVELPRO3 level tool, for these orientations, I estimated the error interval between the accelerometer's and true vertical axis to be $\pm 0.5°$. For these orientations, this results in the uncertainty of $\pm\sin(0.5°)\, g$ along the horizontal axes, and of $\pm[1 - \cos(0.5°)]g$ along the vertical axis.

The last 6 of the 12 orientations are more complicated. In those, one axis is horizontal, and the other two are at 45° to the vertical. It is more difficult to precisely position the wooden block at exactly this orientation: steel angles are used, but I still estimate the orientation errors to be more significant than in the first 6 orientations. The error interval about the horizontal axis is still estimated to be $\pm 0.5°$. The error interval about the non-horizontal axes – which is a function of not only the block's position, but also the orientation of the accelerometer about the axis normal to the surface it is mounted on – is estimated to be wider, at $\pm 2.0°$. There are therefore several contributions to the error on each axis.

Consider the illustration in Fig. 4.4 below. This corresponds to orientation 11 in Fig. 4.3, but the idea here is simply to project, and appropriately combine, the potential errors on each axis due to imprecise orientation; when applying the errors, the appropriate axes should be affected instead. Here, axes x and y are at $\theta_z = 45°$ incline to the horizontal, and axis z is horizontal.

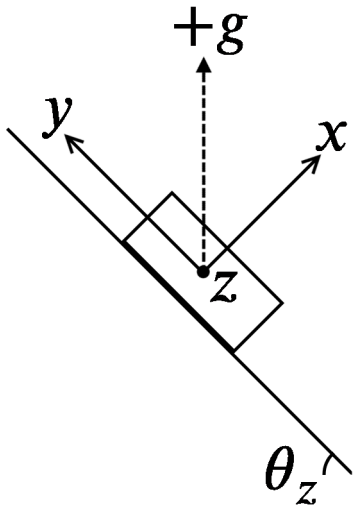

Fig. 4.4. Summary of notation for uncertainty derivations for the 45° orientations of the "Box" experiment

First, we note, that the orientation error about horizontal axis z affects x and y equally. Specifically, if $\theta_z = 45°$ is the best fit, and $\delta\theta_z$ is the error in it, then the difference in projection magnitudes between the best fit value, and the value with error, becomes:

$$\delta p_{xz} = \delta p_{yz} = g\left[\sin(45° + \delta\theta_z) - \sin(45°)\right], \tag{4.1}$$

with δp_{jl} as the error in p_j due to an error in orientation about axis l.
By the above assumptions, $\delta\theta_z = 0.5°$, which yields:

$$\delta p_{xz} = \delta p_{yz} = g\left[\sin(45.5°) - \frac{1}{\sqrt{2}}\right]. \tag{4.2}$$

Next, we recognize that x- and y-axes are symmetric with regards to the gravity vector, and with regards to the z-axis, and each other. The uncertainties in orientations about them are also equal: $\delta\theta_x = \delta\theta_y = 2°$. So, the error on x-axis due to an orientation error about the y-axis would have the same expression as an error on y-axis due to an orientation error about the x-axis:

$$\delta p_{yx} = \delta p_{xy}, \tag{4.3}$$

and

$$\delta p_{zx} = \delta p_{zy}. \tag{4.4}$$

To calculate these quantities, it is enough to express the proper gravitational acceleration vector ($+g$ on Fig. 4.4) as:

$$\vec{\boldsymbol{g}} = g\begin{bmatrix}\sin(\theta_z)\\ \cos(\theta_z)\\ 0\end{bmatrix} = g\begin{bmatrix}1/\sqrt{2}\\ 1/\sqrt{2}\\ 0\end{bmatrix}. \tag{4.5}$$

Next, to express the error in $\vec{\boldsymbol{g}}$ due to an orientation error about x, we build a rotation matrix $\mathbf{R}_x$ that rotates $\vec{\boldsymbol{g}}$ by $\delta\theta_x$ about x-axis, and similarly a rotation matrix $\mathbf{R}_y$ to calculate the change in $\vec{\boldsymbol{g}}$ due to rotation about the y-axis. The errors then simply become:

$$\delta p_{yx} = \left|\left[\mathbf{R}_x\vec{\boldsymbol{g}} - \vec{\boldsymbol{g}}\right]_y\right|, \tag{4.6}$$

and similarly:

$$\delta p_{zx} = \left|\left[\mathbf{R}_x \vec{g} - \vec{g}\right]_z\right|. \tag{4.7}$$

Equivalent expressions δp_{xy} and δp_{zy} can be built using $\mathbf{R}_y$. Building the above matrices and going through the algebra yields (with $\theta_z = 45°$):

$$\delta p_{yx} = \mathrm{g}\frac{1 - \cos(\delta\theta_x)}{\sqrt{2}}, \tag{4.8}$$

which with $\delta\theta_x = \delta\theta_y = 2°$ means:

$$\delta p_{yx} = \delta p_{xy} = \mathrm{g}\frac{1 - \cos(2°)}{\sqrt{2}}. \tag{4.9}$$

Similarly:

$$\delta p_{zx} = \mathrm{g}\frac{\sin(\delta\theta_x)}{\sqrt{2}}, \tag{4.10}$$

which with $\delta\theta_x = \delta\theta_y = 2°$ means:

$$\delta p_{zx} = \delta p_{zy} = \mathrm{g}\frac{\sin(2°)}{\sqrt{2}}, \tag{4.11}$$

Together, Eqs. (4.2), (4.9) and (4.11) quantify the individual contributions to the uncertainties in reference value in this orientation (and, by trivial extension, to all of the last 6 orientations). However, we first need to combine them.

To ensure that the uncertainty is not underestimated, to quantify the uncertainties to reference values throughout this work, I will use the more conservative, potentially-systematic uncertainty propagation methods from section 2.4. For a sum, this simply means that uncertainties add:

$$\delta\vec{p} = \mathrm{g}\begin{bmatrix} \sin(45.5°) - \dfrac{\cos(2°)}{\sqrt{2}} \\ \sin(45.5°) - \dfrac{\cos(2°)}{\sqrt{2}} \\ \sqrt{2}\sin(2°) \end{bmatrix}. \tag{4.12}$$

Again, consult Fig. 4.4 above for reference on how to apply this to the other 5 of the last 6 orientations. This describes how the uncertainties in the 12 reference accelerations were quantified. The uncertainties in voltages were simply taken as the 95% CI half-widths from the voltage readings collected over the 1 minute that the accelerometers were held stationary. As subsection 4.2.1 will show, the voltage noise is approximately Gaussian, therefore this is a reasonable method. Most of this noise presumably comes from minute thermal fluctuations, both in the DAQ and in the accelerometers.

After conducting the experiment, as described above, its data was used to calibrate the two accelerometers, and to test the calibration algorithms. Both sets of results are in the section 4.2.

4.1.3 *Two-step "Box" Experiment Description*

In this section, the two-step "Box" experiment, which uses the general principles listed as method 4 ("two-step static calibration") in section 2.2, is described.

The experimental specification for this method consists of two steps. During the first, the accelerometer must simply be exposed to at least 9 different static orientations, without regard for what these orientations are. I therefore reuse the data from the 12 orientations one-step "Box" experiment, ignoring the reference accelerations.

However, for the second step, a new experiment must be conducted: the box with the accelerometers mounted on it must be rotated about each of the 3 body axes (which, as was described in subsection 4.1.1, are mounted in parallel to the box's edges).

For this step, I have chosen to keep the axis of rotation horizontal, for reasons discussed in subsection 3.1.3: this provides the best possible data. This also, however, enforces me to use the overdetermined homogeneous linear algebraic system-based approach (i.e., based on Eqs. (3.50)–(3.52)), because the angle between the axis of rotation and gravity is 90°.

The measurement procedure is the same as in subsection 4.1.1. The wooden block is placed in such way that the x-axis is horizontal, readings of all 3 axes are recorded for 1 minute using the DAQ and the PC with LabVIEW controlling the DAQ, then the box is rotated about the x-axis, and the results are similarly recorded for 1 minute. This happens 6 times, for 6 different orientations (4 lying flat, and 2 more at 45 degree angles, to ensure the best possible spread of data about the circle in Fig. 3.1). Next, this procedure is repeated for the y-axis, and then the z-axis.

The uncertainties in step 1 arise only from the voltage noises, which are quantified the same way as in subsection 4.1.2. The uncertainties in step 2 arise from voltage noises as well, and also due to the potential orientation errors about the theoretical axis of rotation. Subsection 3.2.3 describes, how this uncertainty is accounted for using orientation uncertainty angles $\delta\theta_j$, one for each axis. These uncertainties are estimated to be $\delta\theta_x = \delta\theta_y = \delta\theta_z = 2°$.

The data from the two-step experiment was used to calibrate the two accelerometers, with the results presented in section 4.3.

4.2 Results and Discussion for One-step “Box” Experiment

4.2.1 *Sensor Noise Analysis*

Prior to running the calibration algorithms, we can take the physical voltage signal from a stationary (therefore, constant proper acceleration) accelerometer, and use it to confirm the assumption made in section 3.2: that the noise of a voltage signal for an inertial sensor is Gaussian. Figure 4.5 below, built from several seconds of a typical voltage signal from one of the accelerometers, clearly confirms this.

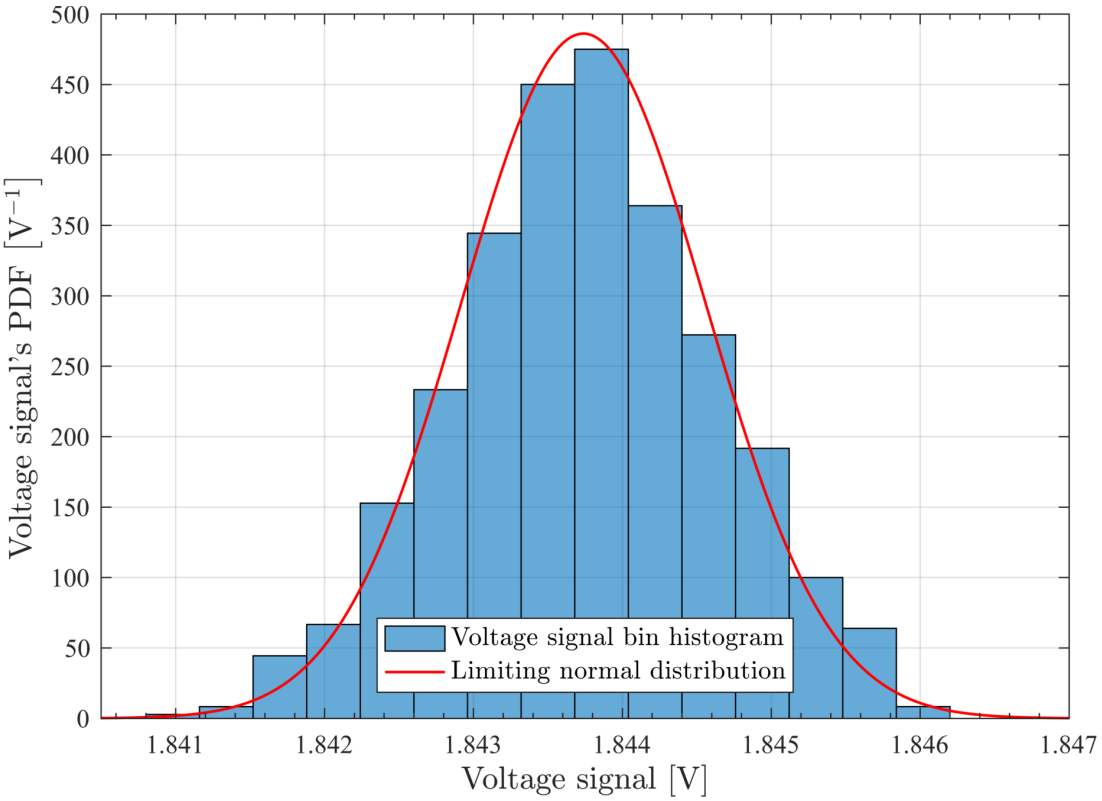

Fig. 4.5. Representative voltage signal noise bin histogram

4.2.2 *Selection of Computational Approach*

All accelerometers that are used in this thesis are 3-axis. Therefore, 3-axis data was collected during both "Box" calibration procedures. However, one of the goals of this experiment was also to study the behavior of the 1-axis algorithms, and confirm that these algorithms are working. To do this, we can split the 3-axis data into 3 datasets of 1-axis data, and test the 1-axis algorithms on those.

1-axis and one-step 3-axis algorithms in chapter 3 all have two versions: MATLAB **`fminsearch`**-based, and MATLAB **`mldivide`**-based. As is described in subsections 3.2.1 and 3.2.2, either can be used for running multiple Monte Carlo simulations on the datasets with uncertainties. Running multiple simulations on the constantly varying data is also beneficial, because it provides a more accurate estimate of the time a single calculation takes, due to repetition. Table 4.1 below presents a comparison between the two approaches to 1-axis calibrations, based on 100,000 Monte Carlo simulations.

Table 4.1. Comparison of 1-axis calibration algorithm approaches

ACCELEROMETER	APPROACH	AXIS	TIME PER SIMULATION [ms]	COST OF BEST ESTIMATE
1	`fminsearch`	x	3.450	0.0325
1	`mldivide`	x	0.052	0.0325
1	`fminsearch`	y	3.318	0.0183
1	`mldivide`	y	0.023	0.0183
1	`fminsearch`	z	3.433	0.0220
1	`mldivide`	z	0.022	0.0220
2	`fminsearch`	x	3.411	0.0054
2	`mldivide`	x	0.024	0.0054
2	`fminsearch`	y	3.403	0.0040
2	`mldivide`	y	0.023	0.0040
2	`fminsearch`	z	3.520	0.0038
2	`mldivide`	z	0.037	0.0038

Table 4.2 below presents the same results, but for the 3-axis calibration algorithms, also with 100,000 simulations.

Table 4.2. Comparison of 3-axis calibration algorithm approaches

ACCELEROMETER	APPROACH	TIME PER SIMULATION [ms]	COST OF BEST ESTIMATE
1	`fminsearch`	89.599	0.0728
1	`mldivide`	0.216	0.0246
2	`fminsearch`	52.119	0.0104
2	`mldivide`	0.195	0.0077

As we can see from both tables, approach 2 (**`mldivide`**) is superior to approach 1 (**`fminsearch`**) in time per simulation for both 1-axis and 3-axis datasets. Results from cost function for 1-axis datasets don't have any difference in the approaches, however results from cost function for 3-axis datasets have significant improvement with approach (**`mldivide`**). This demonstrates, that for 3-axis datasets, **`mldivide`** algorithm gives better fit to compare with **`fminsearch`** algorithm. This is actually expected: **`mldivide`**, being a direct

method, is guaranteed to find the exact solution to the overdetermined algebraic linear problem, while **`fminsearch`** is iterative, and therefore only converges within a certain tolerance.

From this point, all subsequent work for 1-axis and one-step 3-axis sensor calibrations, will make use of the **`mldivide`**-based calibration algorithms.

4.2.3 *Monte Carlo Convergence Study*

It is well-understood that Monte Carlo algorithms must be run with a sufficient number of Monte Carlo simulations, otherwise they do not converge. It is also understood that due to the stochastic nature of the method, convergence may not necessarily be smooth, particularly at low simulation counts. We therefore need to assess a good number of simulations K, that can be considered enough for the Monte Carlo algorithms in section 3.2.

Below, a quantity will be considered "converged" if it is less than 1% different from the same quantity at 10^6 simulations.

First, we study the convergence of the 1-axis inverse sensitivity parameters. Figure 4.6 below shows their convergence for both accelerometers. On it, we can observe: (a) while the accelerometers are of exactly the same model, their parameters are clearly different, and (b) their calibration parameters converge very rapidly.

Applying the 3-axis convergence algorithms to the same datasets yields the results in Figs. 4.7 and 4.8 below. We note, that the 3-axis parameters remain different, and also that the off-diagonal (i.e., misalignment-affecting) elements of $\mathbf{A}$ take slightly longer to converge.

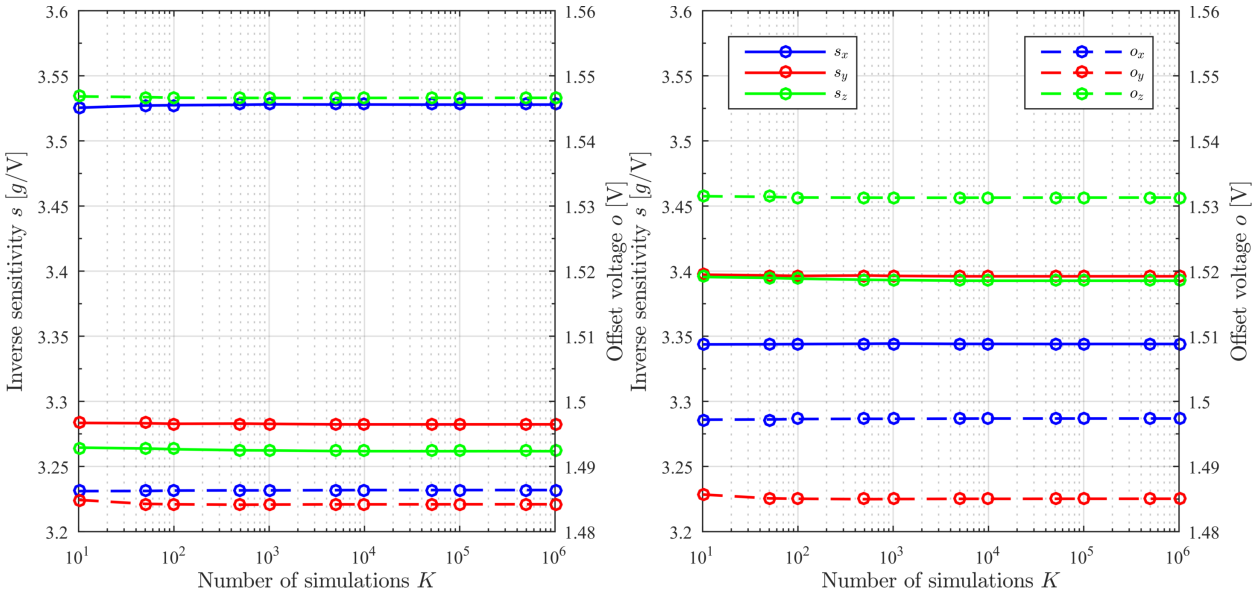

Fig. 4.6. Convergence of all elements of s and o for 1-axis data: accelerometers 1 (left) and 2 (right)

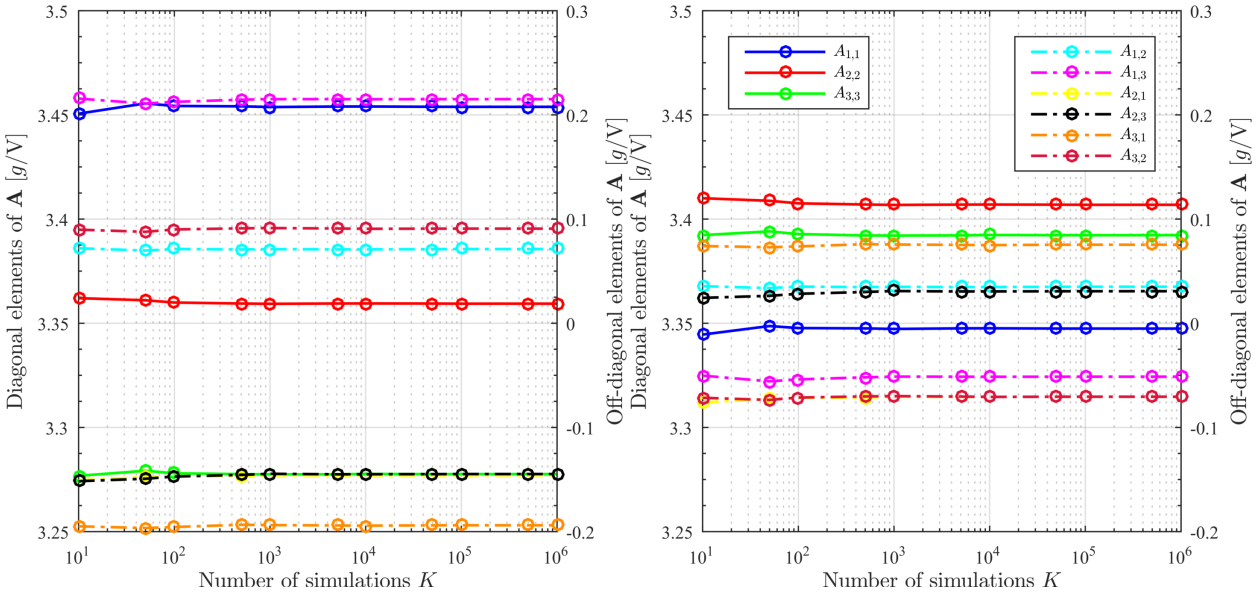

Fig. 4.7. Convergence of all elements of $\mathbf{A}$ for 3-axis data: accelerometers 1 (left) and 2 (right)

The products of an uncertainty-quantifying calibration algorithm are both the best fit sensor calibration quantities, and their uncertainties. Both quantities must be converged to judge a Monte Carlo calibration algorithm converged. Figure 4.9 below demonstrates the convergence of the 1-axis sensitivity and its uncertainty.

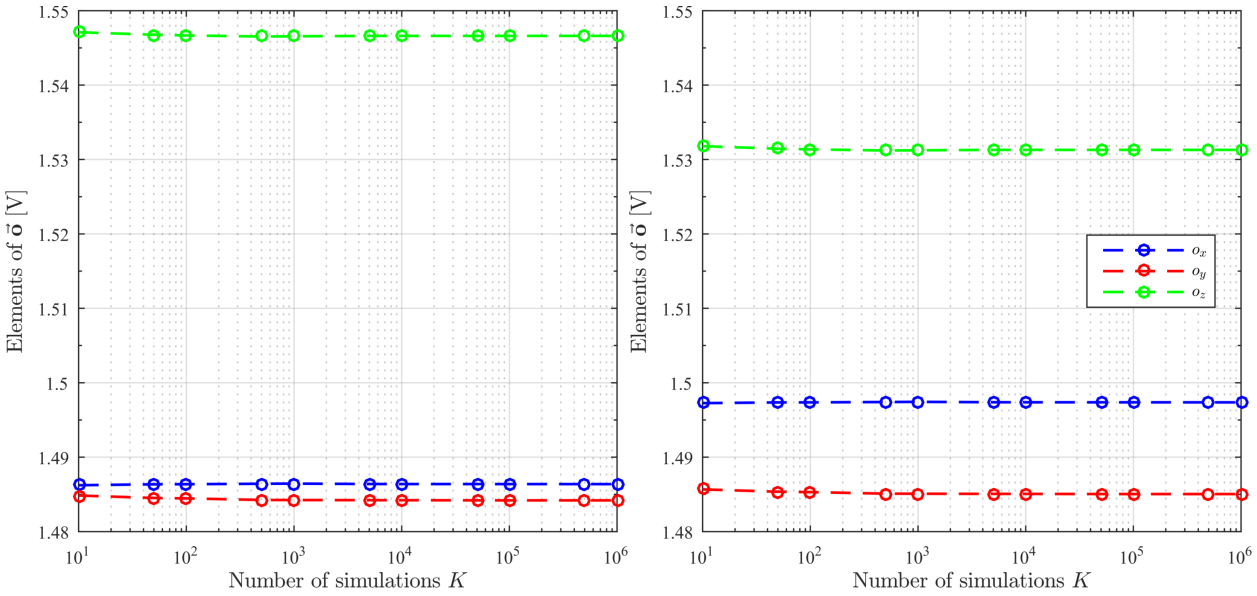

Fig. 4.8. Convergence of the offset vectors: accelerometers 1 (left) and 2 (right)

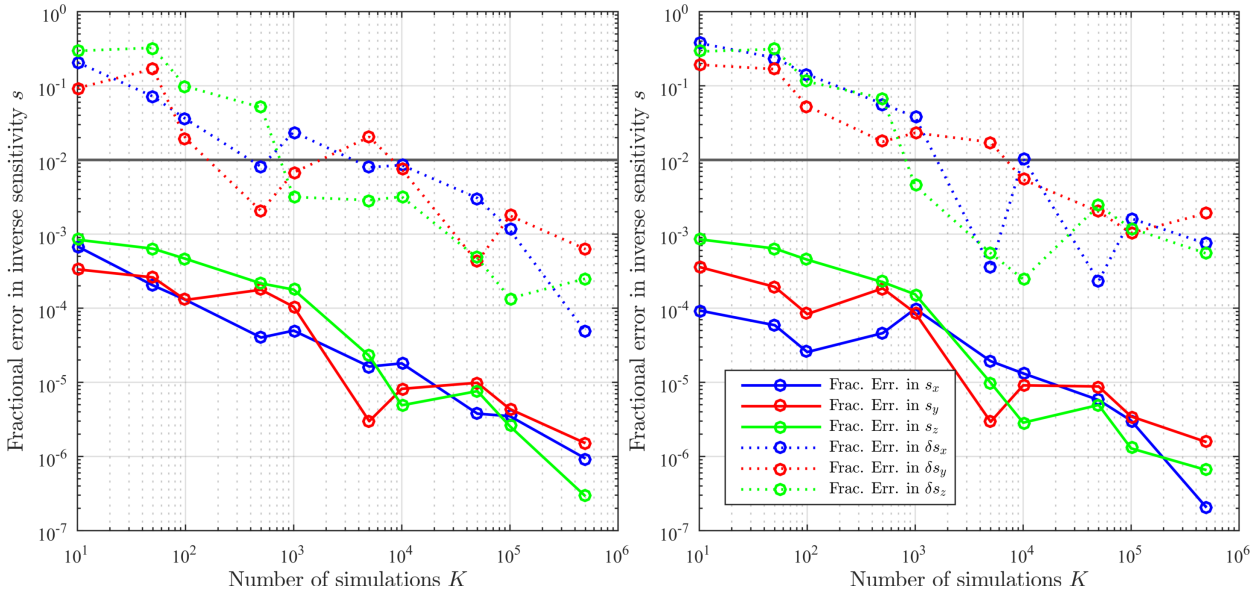

Fig. 4.9. Convergence of fractional errors of s and δs for 1-axis data: accelerometers 1 (left) and 2 (right)

As we can see from Fig. 4.9, it takes 100,000 simulations for the fractional error in uncertainties to drop below 1%. We also note, that uncertainties converge significantly slower than the best fits.

Convergence of fractional errors of elements of the matrix **A** for 3-axis data is in Fig. 4.1 below. We can see, that the fractional error again drops below 1% with 100,000 simulations. These plots also illustrate that off-diagonal elements of **A** converge slightly slower.

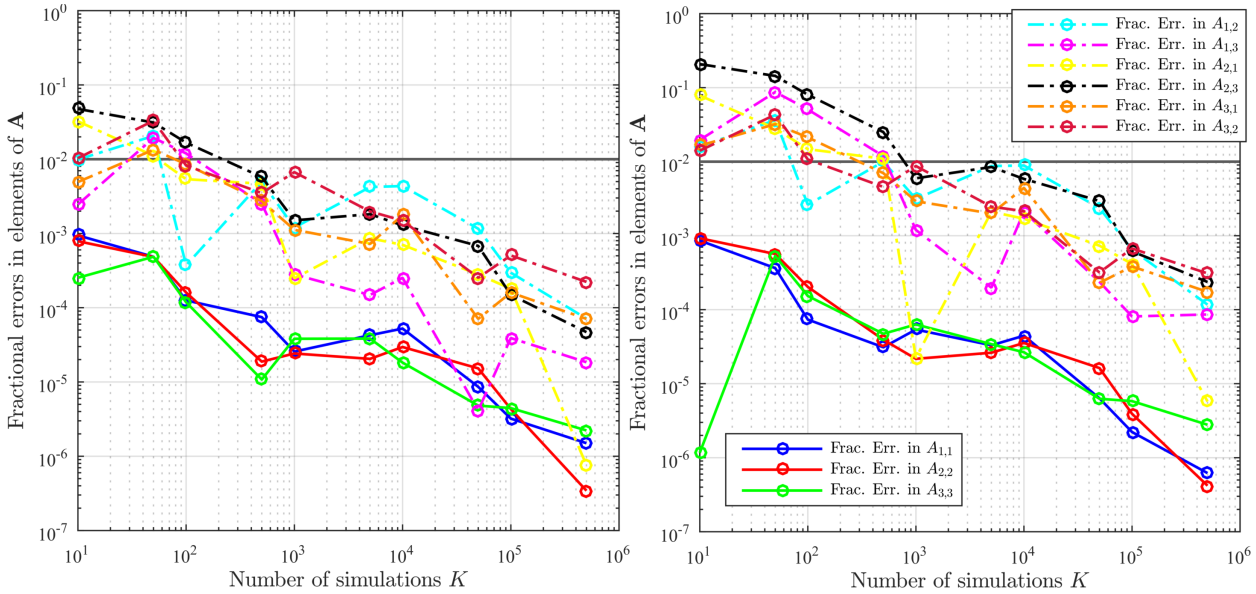

Fig. 4.10. Convergence of fractional errors of **A** for 3-axis data: accelerometers 1 (left) and 2 (right)

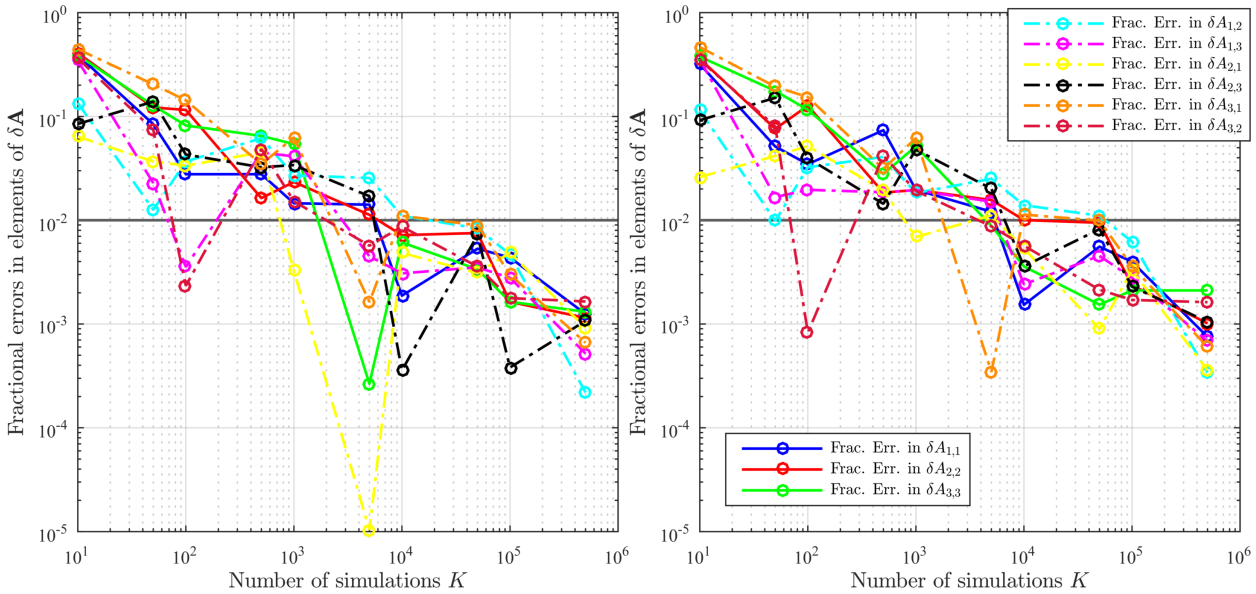

Fig. 4.11. Convergence of fractional errors for δ**A**: accelerometers 1 (left) and 2 (right)

The convergence of fractional errors of δ**A** for 3-axis data is presented in Fig. 4.11 above. It highlights the statistical nature of the method, but once again demonstrates that all errors fall below 1% at 100,000 simulations.

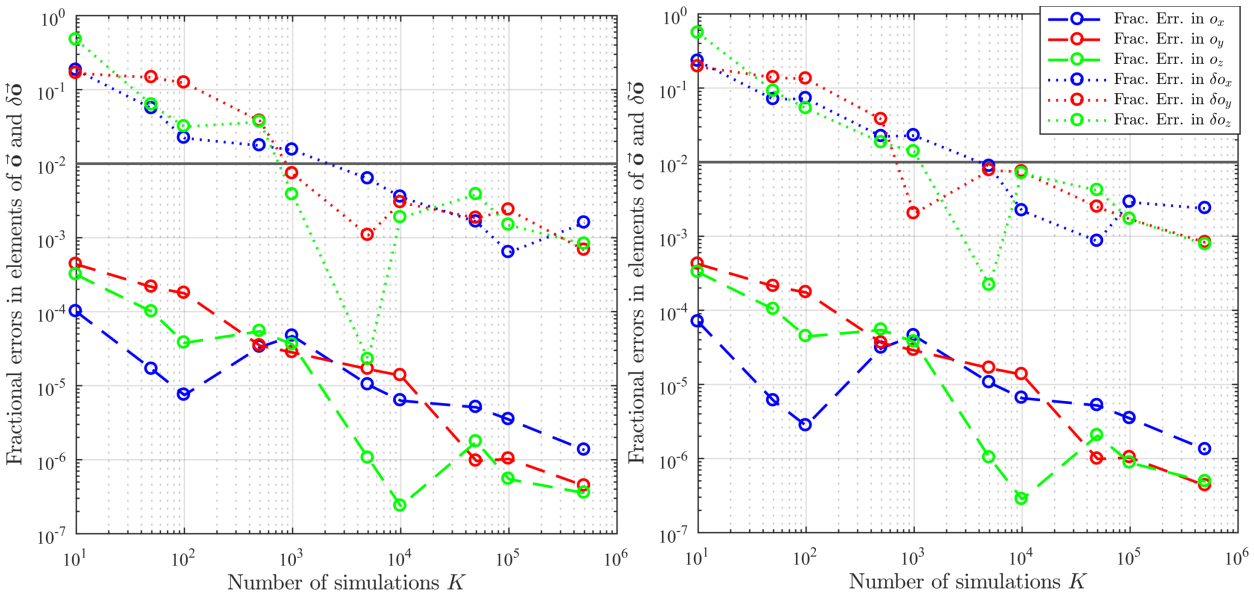

Fig. 4.12. Convergence of fractional errors of $\vec{o}$ and $\delta\vec{o}$ for 3-axis data: accelerometers 1 (left) and 2 (right)

Lastly, the convergence of fractional errors of the offsets and their uncertainties, from 3-axis data, is provided in Fig. 4.12 above. Like all the other fractional error convergence studies, this plot again confirms, that: (a) uncertainties converge slower than best estimates, and (b) 100,000 simulations is sufficient to reduce all fractional errors below 1%.

At this point, we know that **`mldivide`** is the superior approach, and that 100,000 simulations are enough. We can finally use this knowledge, to compute the calibration parameters for both accelerometers (from the one-step "Box" experiment data), and to evaluate the quality of the resulting fit, with uncertainties.

4.2.4 *Evaluation of Quality of Fit*

Applying the uncertainty-quantifying calibration algorithm from subsection 3.2.2, we obtain the 3-axis calibration parameters for the two accelerometers, with uncertainties. The quality of the resulting calibration (i.e., "quality of fit") can be evaluated quantitatively, simply by evaluating the 3-axis cost function (Eq. (3.22)). The issue with this evaluation is that it does not evaluate the quality of the uncertainty quantifications that the Monte Carlo algorithm produces. Another way to evaluate the fit is to simply plot it, with uncertainties, and analyze whether, together with the uncertainties, the fit accurately accounts for the physical measurements it was based on.

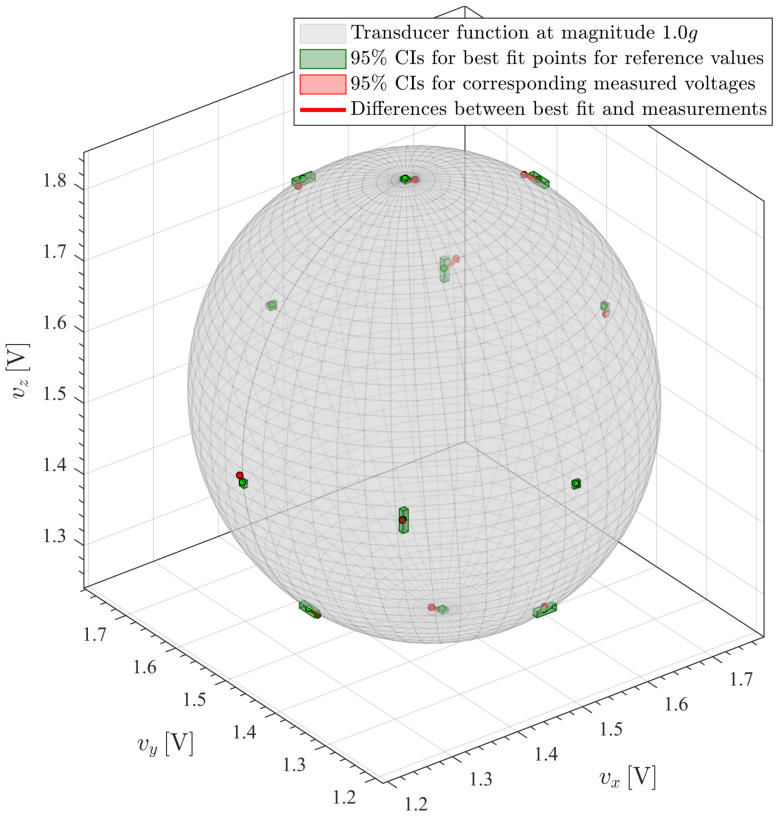

Fig. 4.13. Quality of fit plot for accelerometer 1, using one-step “Box” experiment data

A 3-axis calibration relates 6 different variables – the 3 physical axes, and the 3 voltages. It is therefore not possible to plot them all, as is commonly done when evaluating 2-variable regressions. Instead, we may construct our plots as follows:

1. Select a single reference physical value magnitude, such as $\|\vec{\boldsymbol{p}}\| = 1.0g$. This is obviously the best choice for the “Box” experiments, because all measured values should have this magnitude.
2. By inverting Eq. (3.2), find the voltages that the best fit transducer function would accept to result in the best fit reference physical values. These will be referred to as the “best fit reference voltages.”
3. Using Monte Carlo sampling, and the inverted form of Eq. (3.2), propagate the uncertainties in $\mathbf{A}$, $\vec{\boldsymbol{o}}$ and $\left\{\vec{\boldsymbol{p}}\right\}$ to the best fit reference voltages. Alternatively, propagate the uncertainties deterministically, using the procedures in section 2.4.
4. Plot the ellipsoidal surface, on the $\left(v_x, v_y, v_z\right)$ axes, that the best fit transducer function produces with the above fixed reference physical value magnitude.

5. Plot the best fit reference voltages, with uncertainties, and the corresponding measured voltages, with uncertainties, on the same axes. The uncertainties are shaped as rectangular parallelepipeds.

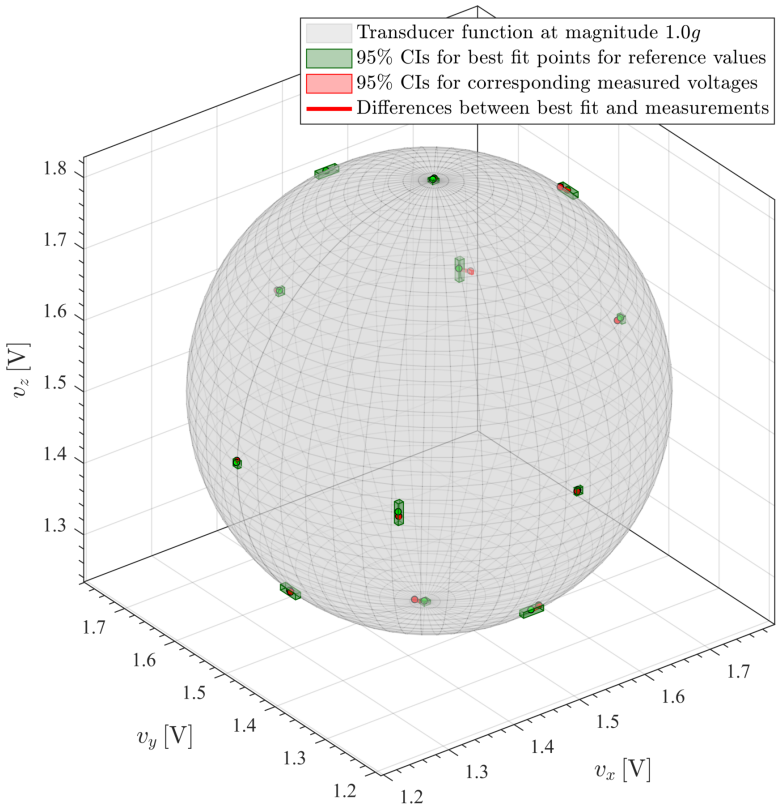

Fig. 4.14. Quality of fit plot for accelerometer 2, using one-step "Box" experiment data

Figures 4.13 and 4.14 above provide these two quality of fit plots, one for each accelerometer.

While there are a couple measurements that fall just out of range of the estimated uncertainties, overall, the plots clearly demonstrate, that the uncertainties are adequate: the best fit uncertainties (green) clearly primarily overlap the measurements' uncertainties (red). This provides a qualitative (i.e., graphical) confirmation, that the one-step 3-axis Monte Carlo calibration algorithm was successful for the 3-axis accelerometers, using one-step "Box" experiment data.

3-axis calibration is generally more complicated than 1-axis. It is therefore worth assessing, whether it is necessary to account for the potential sensor misalignments, considering that they are near zero in modern MEMS sensors, and that a 1-axis calibration experiment would generally be much faster. Figures 4.15 and 4.16 plot the qualities of fit for these simplified, misalignment-ignoring calibrations.

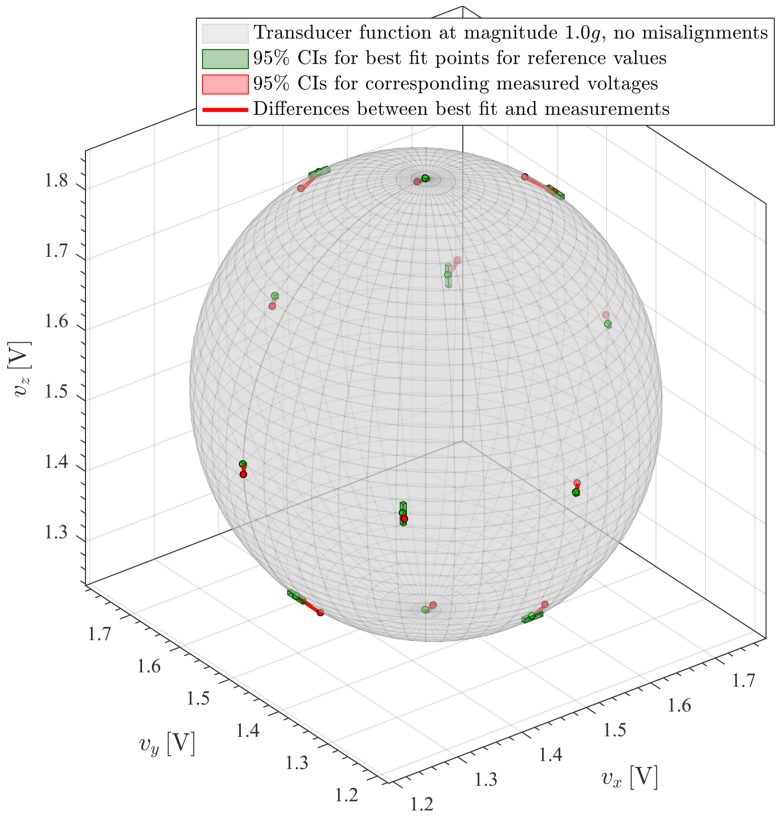

Fig. 4.15. Quality of fit plot for accelerometer 1, using one-step “Box” experiment data, and ignoring misalignments

These plots, particularly when compared to Figs. 4.13 and 4.14 above, clearly illustrate, that despite being small, misalignments may not be ignored: the fits are notably worse. Particularly for the (less well-aligned) accelerometer 1, each of the points falls out of confidence intervals. This indirectly confirms an important fact: that of the necessity of per-sensor calibration. Kalman filter, for example, cannot do anything to correct a misalignment error, because such error results in a bias, and not a loss of precision, and datasheets do not generally report misalignments.

Table 4.3. Quality of fit quantification, using one-step “Box” experiment data

	Cost of best estimate	
Calibration parameters	Accelerometer 1	Accelerometer 2
Without misalignment angles	0.0728	0.013
With misalignment angles	0.0246	0.008
Advantage	2.9579	1.709

As stated above, to quantitatively assess a quality of fit for a dataset, the cost function can be used. Table 4.3 above quantifies qualities of fit for the two accelerometers, with and without misalignments.

This data quantitatively confirms what was demonstrated qualitatively above: that accounting for misalignments produces a notable, useful improvement in the quality of fit.

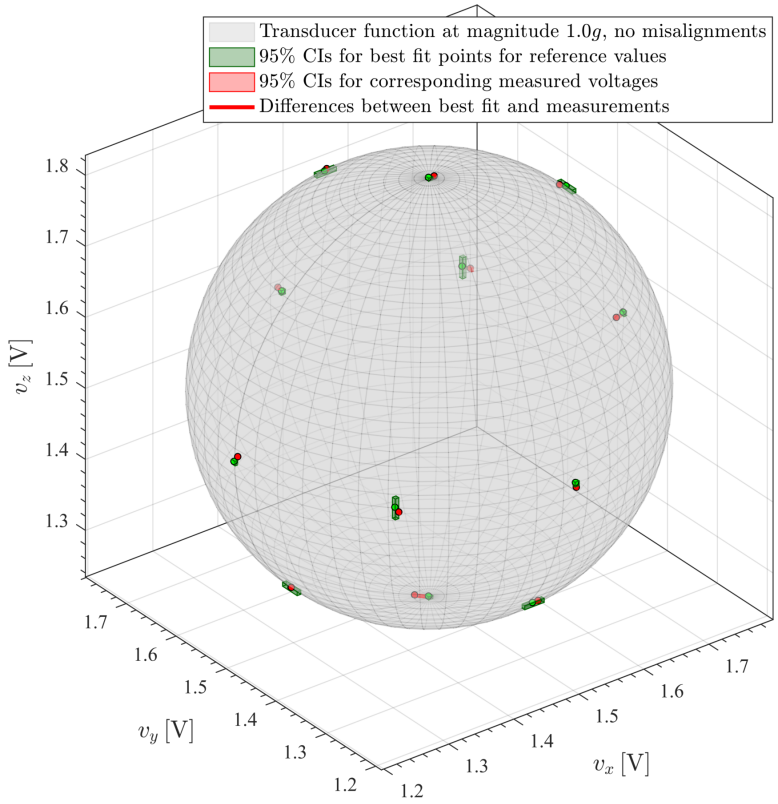

Fig. 4.16. Quality of fit plot for accelerometer 2, using one-step "Box" experiment data, and ignoring misalignments

4.2.5 *Final 3-axis Accelerometer Calibration Parameters from One-step "Box" Experiment*

In this section, calibration parameters, obtained from the one-step "Box" experiment, and processed with the 3-axis uncertainty quantifying algorithm from subsection 3.2.2, are presented.

Accelerometer 1 full sensor calibration matrix $\mathbf{A}$:

$$\mathbf{A} = \begin{bmatrix} 3.454 & 0.07 & 0.22 \\ -0.15 & 3.359 & -0.15 \\ -0.19 & 0.09 & 3.278 \end{bmatrix} \frac{g}{\mathrm{V}} \pm \begin{bmatrix} 0.026 & 0.04 & 0.04 \\ 0.036 & 0.019 & 0.04 \\ 0.04 & 0.04 & 0.020 \end{bmatrix} \frac{g}{\mathrm{V}}. \tag{4.13}$$

Accelerometer 1 offset voltages $\vec{o}$:

$$\vec{o} = \begin{bmatrix} 1.486 \\ 1.484 \\ 1.547 \end{bmatrix} \text{V} \pm \begin{bmatrix} 0.002 \\ 0.002 \\ 0.002 \end{bmatrix} \text{V}. \tag{4.14}$$

Accelerometer 1 (direct, not inverse) sensitivities vector $\vec{k}$, obtained by inverting the diagonal of **A**, from Eqs. (2.3) and (2.4):

$$\vec{k} = \begin{bmatrix} 290 \\ 298 \\ 305 \end{bmatrix} \frac{\text{mV}}{g}. \tag{4.15}$$

Accelerometer 1 misalignment angles, obtained from **A** and Eq. (2.4):

$$\begin{aligned} \alpha_{xz} &= -2.5^\circ \\ \alpha_{xy} &= 3.4^\circ \\ \alpha_{yz} &= -1.2^\circ \\ \alpha_{yx} &= 1.6^\circ \\ \alpha_{zy} &= 3.6^\circ \\ \alpha_{zx} &= 2.5^\circ. \end{aligned} \tag{4.16}$$

Accelerometer 2 full sensor calibration matrix **A**:

$$\mathbf{A} = \begin{bmatrix} 3.347 & 0.04 & 0.05 \\ -0.07 & 3.407 & 0.03 \\ 0.08 & -0.07 & 3.392 \end{bmatrix} \frac{g}{\text{V}} \pm \begin{bmatrix} 0.022 & 0.04 & 0.04 \\ 0.04 & 0.021 & 0.04 \\ 0.04 & 0.04 & 0.022 \end{bmatrix} \frac{g}{\text{V}}. \tag{4.17}$$

Accelerometer 2 offset voltages $\vec{o}$:

$$\vec{o} = \begin{bmatrix} 1.497 \\ 1.485 \\ 1.531 \end{bmatrix} \text{V} \pm \begin{bmatrix} 0.002 \\ 0.002 \\ 0.002 \end{bmatrix} \text{V}. \tag{4.18}$$

Accelerometer 2 (direct, not inverse) sensitivities vector $\vec{k}$:

$$\vec{k} = \begin{bmatrix} 299 \\ 294 \\ 295 \end{bmatrix} \frac{\text{mV}}{g}. \tag{4.19}$$

Accelerometer 2 misalignment angles:

$$\begin{aligned}
\alpha_{xz} &= -1.2° \\
\alpha_{xy} &= -1.3° \\
\alpha_{yz} &= -0.6° \\
\alpha_{yx} &= -1.2° \\
\alpha_{zy} &= -0.9° \\
\alpha_{zx} &= -0.5°.
\end{aligned} \tag{4.20}$$

All of these values fall well within the stated ranges from the accelerometer datasheets, as expected.

4.3 RESULTS AND DISCUSSION FOR TWO-STEP "BOX" EXPERIMENT

The quality of fit plots with calibration parameters obtained from two-step "Box" experiment, with the uncertainty-quantifying Monte Carlo algorithm from subsection 3.2.3, are presented below, on Fig. 4.17 for accelerometer 1 and on Fig. 4.18 for accelerometer 2.

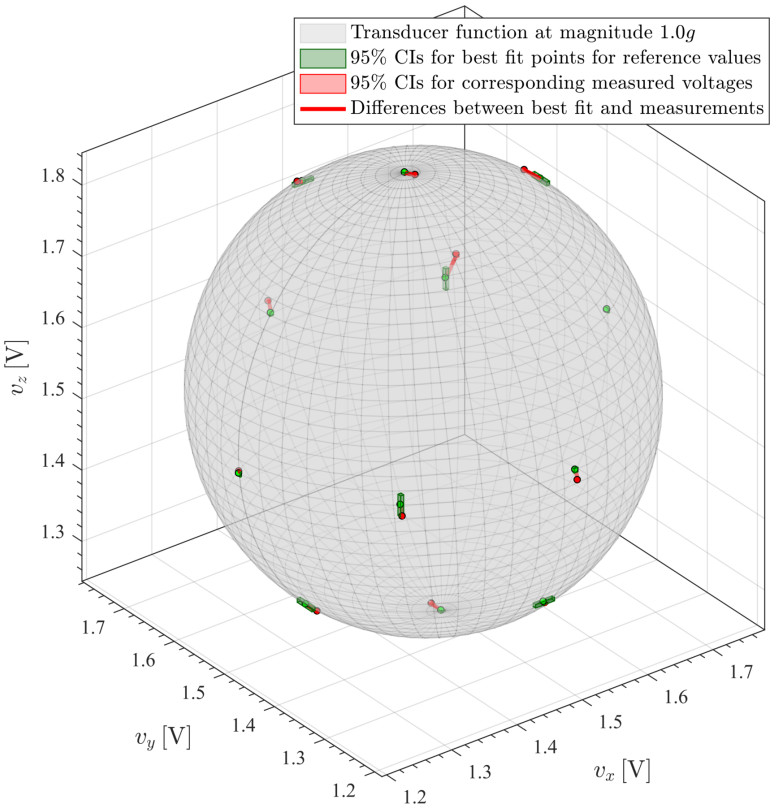

Fig. 4.17. Quality of fit plot for accelerometer 1, using two-step "Box" experiment data

The calibration parameters obtained are also presented below.

Accelerometer 1 full sensor calibration matrix **A**:

$$\mathbf{A} = \begin{bmatrix} 3.420 & 0.063 & 0.085 \\ -0.15 & 3.335 & -0.296 \\ -0.087 & 0.275 & 3.278 \end{bmatrix} \frac{g}{\mathrm{V}} \pm \begin{bmatrix} 0.009 & 0.016 & 0.017 \\ 0.016 & 0.007 & 0.016 \\ 0.017 & 0.015 & 0.013 \end{bmatrix} \frac{g}{\mathrm{V}}. \tag{4.21}$$

Accelerometer 1 offset voltages $\vec{o}$:

$$\vec{o} = \begin{bmatrix} 1.486 \\ 1.485 \\ 1.544 \end{bmatrix} \mathrm{V} \pm \begin{bmatrix} 0.001 \\ 0.001 \\ 0.001 \end{bmatrix} \mathrm{V}. \tag{4.22}$$

Accelerometer 1 sensitivities vector $\vec{k}$:

$$\vec{k} = \begin{bmatrix} 292 \\ 300 \\ 300 \end{bmatrix} \frac{\mathrm{mV}}{g}. \tag{4.23}$$

Accelerometer 1 misalignment angles:

$$\begin{aligned}
\alpha_{xz} &= -2.5^\circ \\
\alpha_{xy} &= 1.5^\circ \\
\alpha_{yz} &= -1.1^\circ \\
\alpha_{yx} &= 4.7^\circ \\
\alpha_{zy} &= 1.4^\circ \\
\alpha_{zx} &= 5.0^\circ.
\end{aligned} \tag{4.24}$$

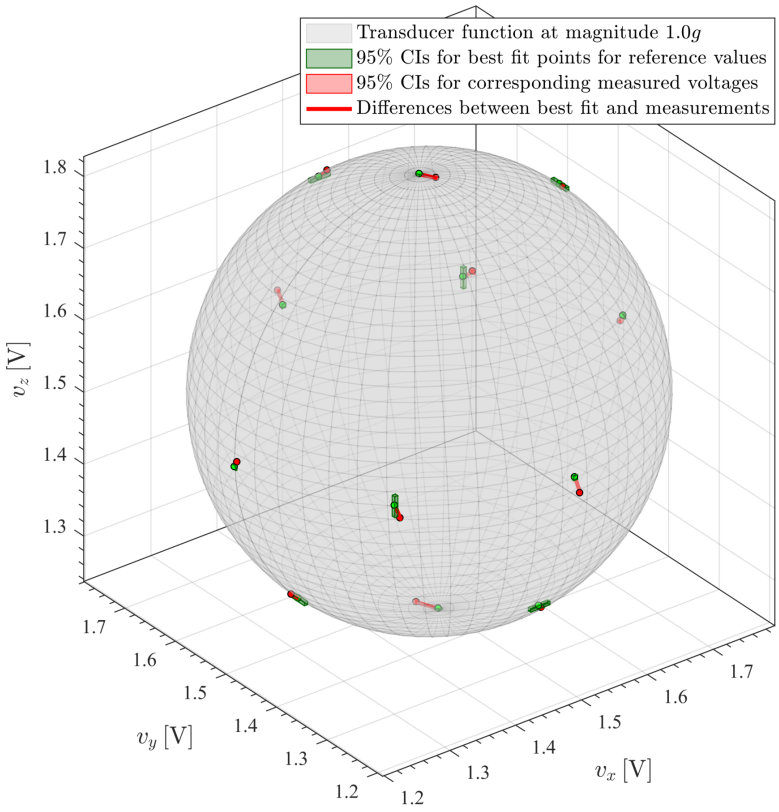

Fig. 4.18. Quality of fit plot for accelerometer 2, using two-step "Box" experiment data

Accelerometer 2 full sensor calibration matrix $\mathbf{A}$:

$$\mathbf{A} = \begin{bmatrix} 3.370 & -0.049 & 0.007 \\ -0.025 & 3.385 & -0.213 \\ -0.001 & 0.193 & 3.392 \end{bmatrix} \frac{g}{\mathrm{V}} \pm \begin{bmatrix} 0.009 & 0.016 & 0.014 \\ 0.015 & 0.009 & 0.016 \\ 0.014 & 0.015 & 0.011 \end{bmatrix} \frac{g}{\mathrm{V}}. \tag{4.25}$$

Accelerometer 2 offset voltages $\vec{o}$:

$$\vec{o} = \begin{bmatrix} 1.496 \\ 1.485 \\ 1.532 \end{bmatrix} \mathrm{V} \pm \begin{bmatrix} 0.001 \\ 0.001 \\ 0.001 \end{bmatrix} \mathrm{V}. \tag{4.26}$$

Accelerometer 2 (direct, not inverse) sensitivities vector $\vec{k}$:

$$\vec{k} = \begin{bmatrix} 297 \\ 295 \\ 295 \end{bmatrix} \frac{\text{mV}}{g}. \tag{4.27}$$

Accelerometer 2 misalignment angles:

$$\begin{aligned} \alpha_{xz} &= -0.4^\circ \\ \alpha_{xy} &= 0.0^\circ \\ \alpha_{yz} &= 0.8^\circ \\ \alpha_{yx} &= 3.3^\circ \\ \alpha_{zy} &= 0.1^\circ \\ \alpha_{zx} &= 3.6^\circ. \end{aligned} \tag{4.28}$$

We can see from both the plots, and the numerical results, that the two-step experiment yielded significantly higher misalignment angles, particularly for accelerometer 1. The estimated uncertainties, however, are lower. Overall, this appears to be a calibration of inferior quality to the one-step method: there are clearly significant mismatches on the quality of fit plots, and the misalignment angles are unexpectedly high.

It is likely that the method suffered from the same caveat that Bonnet, et al. noted, to explain the imperfections in their experiments: the two-step method appears to be very sensitive to orientation imperfections during step 2 [9].

The one-step "Box" experiment clearly provides a better fit for the available data. This may be explained by the following: while the one-step method does rely on knowing orientations more explicitly, it does not require any strict relations between different orientations. Step 2 of the two-step method, clearly, does. This may result in a systematic error that the one-step experiment avoids, thus resulting in a superior fit.

Overall though, even though the two-step method seems to be inferior, it is not a failure: the fit it provided is fairly close to that of the one-step method, so the method did not diverge – it just provided a worse calibration.

4.4 Summary

The one-step and two-step static calibration experiments were executed on a pair of accelerometers, and the Monte Carlo uncertainty-quantifying algorithm developed in chapter 3 was applied to it. The following results were demonstrated:

1. Voltage noise was confirmed to be Gaussian, justifying an assumption that went into the Monte Carlo uncertainty-quantifying calibration algorithms.

2. For one-step calibration methods, both 1-axis and 3-axis, **`mldivide`**-based calibration algorithms performed strictly better than **`fminsearch`**-based algorithms: faster, and more accurate. This is notable, because **fminsearch**-based (and other optimization-based approaches, even for systems that can be viewed as overdetermined linear algebraic systems) approaches appear prevalent in literature [7] [40].
3. 100,000 Monte Carlo simulations were shown to be more than enough to get the fractional error of less than 1% for all calibration parameters, including uncertainties. Therefore, $K = 100{,}000$ is a good "magic number" to use in the future.
4. The need for appropriate misalignment angle estimation for 3-axis accelerometers was confirmed. This is important, because misalignment calibration is more complicated than offset and sensitivity, and because datasheets almost never provide nonzero misalignments. This also implicitly justifies the effort of the first half of this thesis: calibration was demonstrated to be very important to match the sensor measurements to physical results.
5. Both accelerometers used throughout the project were confirmed to be within the technical specification. However, even though the obtained calibration parameters are close, they are still not exact typical values stated in the technical specification, due to the complexity of MEMS manufacturing process, and temperature dependence.
6. Two-step static calibration was shown to be notably slower than one-step calibration (up to a factor of 1000 for some simulations), due to the need for a nonlinear calibration at each Monte Carlo step. It also clearly did not provide superior results: quality of fit plots, and questionable misalignment angle estimates, clearly demonstrated the superiority of one-step calibration. This appears to be explainable by the same justification as given by the method's authors: step 2 of the method is surprisingly sensitive to orientation errors, likely because it relies on a strict geometric relationship between the 3 datasets of step 2. One-step static calibration does not suffer from this: it does rely on knowing orientations, which can introduce errors, but because it does not rely on any geometric relationships between these orientations, occasional orientation errors do not seriously damage the results.

In short, this chapter demonstrated both the need for an uncertainty-quantifying calibration algorithm for 3-axis accelerometers, and the effectiveness of one of proposed ones.

Chapter 5
Accelerometer and Gyroscope Calibration via "Turntable" Experiments

This chapter is similar to chapter 4: it presents the results of the turntable experiment. Two important results should arise: (1) is there a tangible benefit to accelerometer calibration via turntable vs box, and (2) which is better for accelerometer calibration: varying speeds, or varying radii. Neither choice exists for gyroscope calibration.

Section 5.1 details the equipment used, including some built for these experiments. Accelerometer calibration using the turntable is described in section 5.2, and a comparison of all 3 accelerometer calibration methods, tested in this work, is given in subsection 5.2.4. Gyroscope calibration using the turntable is given in section 5.3.

5.1 Experimental Setup

Experimental setup for "Turntable" experiments includes:

- Tiger Motor U8-16 100kv U-Power Professional Motor.
- Rotating laser-cut wooden platform ("turntable" surface), mounted on the motor.
- Benchtop data acquisition device, National Instruments (NI) DAQ 6341.
- Benchtop programmable power supply unit, B&K Precision DC power supply 9115.
- Two analog accelerometers, Analog Devices ADXL335.
- Two analog gyroscopes, Cytron RB-Cyt-141.

- A desktop personal computer (PC).
- "Hopper" robot's mainboard, to control the turntable motor's angular velocity.
- 90° steel angles to mount the sensors perpendicular to the turntable's surface.

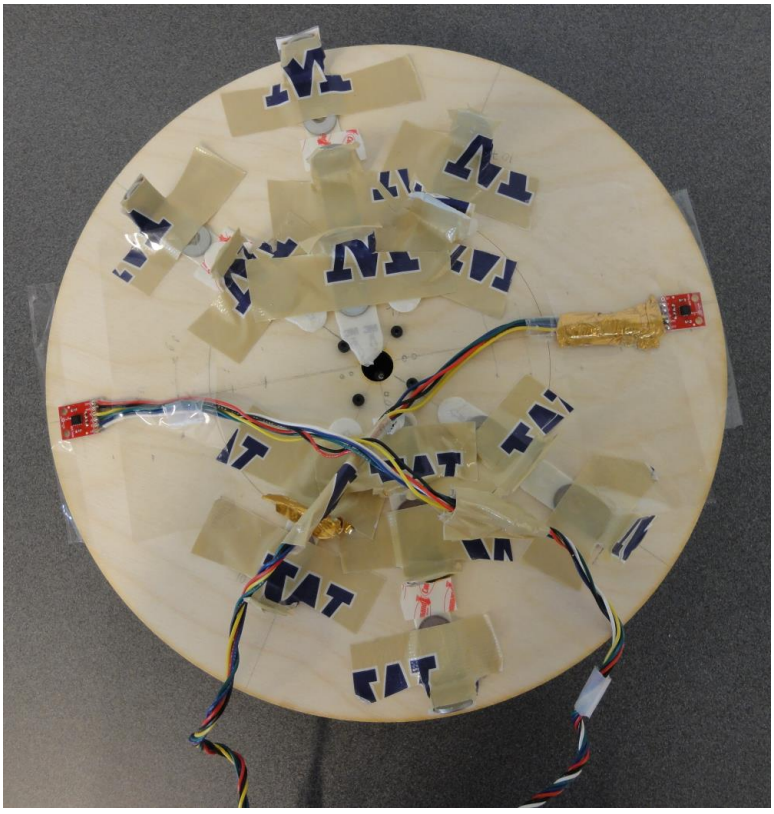

Fig. 5.1. Horizontal sensor placement on the turntable for accelerometer *x*- and *y*-axis calibration

The turntable built for this experiment is show in Figs. 5.1 and 5.2 below. The laser-cut wooden disk is mounted on the motor, which is controlled using the Hopper robot's mainboard [44]. Steel angles are attached to the rotating wooden platform at different radii, in mirrored placement. The angles are present to allow the operator to mount sensors perpendicular to the platform, thus allowing, for example, the calibration of the 1-axis gyroscope (its sensitive axis is parallel to the PCB it is mounted on).

The accelerometers are powered by the DAQ via a voltage divider due to lack of voltage regulator on accelerometers' breakout. The gyroscopes are powered by the DAQ directly, without a voltage divider, because the Cytron RB-Cyt-141 PCB has a voltage regulator on breakout.

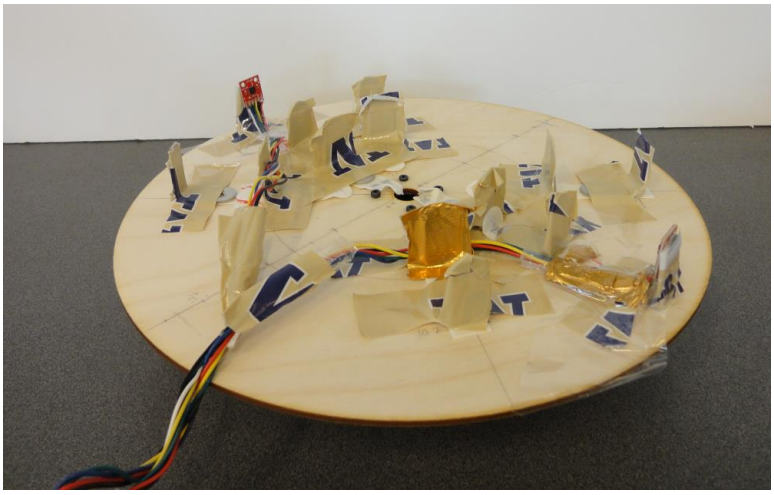

Fig. 5.2. Vertical sensor placement on the turntable for accelerometer z-axis and gyroscope calibration

The motor's onboard microcontroller is able to output time-dependent positioning data from its encoders. The motor is powered by a programmable power supply with 16 V / 10 A.

All of the equipment used in this experiment, and others in this work, is detailed in Appendix A.

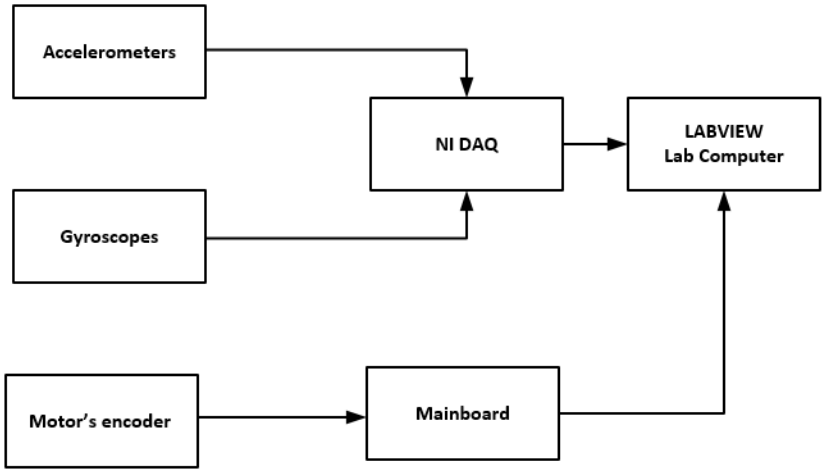

Fig. 5.3. The laboratory setup for data acquisition in "Turntable" experiment

Figure 5.3 above illustrates the general equipment configuration used in the turntable experiment. The motor's encoder is used to estimate, with an uncertainty margin, the turntable's rotating velocity, and to ensure that the encoder-provided velocity is not appreciably different from the theoretical velocity controlled by the mainboard.

As in chapter 4, the sensors' signals were read via DAQ analog channels, plotted (and optionally, smoothed) by a LabVIEW VI, and recorded to a plain text file.

A substantial difficult with the turntable experiment arose from the fact that, because the sensors were wired with multi-wire cables (leading from the DAQ to the sensors), the cables twisted as the turntable rotated. The twist could provide an additional, unaccounted for, resistance, which would be very damaging to an analog sensor. Therefore, the turntable was only launched in short bursts (20-30 seconds), such that the cable did not become appreciably twisted until the very end of the time interval. Still, the unclear resistive effects of the stressed wires may have negatively impacted the experiments.

The rest of the experimental details are experiment-specific, provided in sections 5.2 (accelerometers) and 5.3 (gyroscopes) below.

5.2 3-AXIS ACCELEROMETER CALIBRATION BY VARYING SPEEDS

5.2.1 *Description of Experiment*

One significant benefit of the turntable is that it is able to subject the accelerometer to a range of accelerations, beyond the $\pm 1g$ that static calibration methods can impose. To calibrate 3-axis accelerometers via the turntable, the experiment was set up as follows:

1. Accelerometers are placed on the turntable, precisely (with a ± 2 mm placement uncertainty) at the radius of 13.8 cm, with the $+x$-axis pointing toward the center of rotation.
2. The turntable is rotated with a constant velocity, for about a 20 to 30 second period. If the cable is unexpectedly twisted, and visibly pulls/dislodges the accelerometer, the turntable is stopped, and the trial discarded. (This generally happened ever 4–5 trials).
3. During rotation, the encoder position history, and accelerometer readings, were simultaneously read, plotted and written to plain text file via a LabVIEW VI.
4. At this same velocity, the experiment was repeated 5 times, to attempt to reduce potential errors.
5. Then, another constant velocity was set, and steps 2–4 repeated. This was done for a total of 7 different velocities.
6. After finishing with the entire velocity set, the accelerometers were reoriented, now with the $+y$-axis pointing toward the center of rotation. Steps 2–5 repeated.

7. Then, same procedure was repeated for the $+z$-axis, and all 3 negative axes.

Throughout the experiment, the radius is held constant, except for the $-z$ direction, when the accelerometers had to be mounted on the other sides of the steel angles, which increased the radius by 2 mm.

Consider the following orientation: $+z$ pointing upward, $+x$ pointing horizontally toward the axis of rotation, and $-y$ pointing horizontally, transversely along the direction of travel (i.e., tangent to the accelerometer's trajectory). In this orientation, the reference proper acceleration that the accelerometer experiences, with uncertainties, is given by:

$$\vec{p} = \begin{bmatrix} r\omega^2 \\ -r\alpha \\ g \end{bmatrix} \pm \begin{bmatrix} \omega^2\delta r + 2r|\omega|\delta\omega + g\sin(\delta\theta_z) \\ |\alpha|\delta r + r\delta\alpha + g\sin(\delta\theta_z) \\ 1 - \cos(\delta\theta_z) \end{bmatrix}, \tag{5.1}$$

in which r is the radius at which the accelerometers are placed, δr is the placement error (visually estimated), ω is the angular velocity, $\delta\omega$ is the angular velocity error (estimated from numeric derivative of the encoder position history), α is the angular acceleration (theoretically zero, but still accounted for from the encoder history), $\delta\alpha$ is the angular acceleration uncertainty (same as $\delta\omega$, estimated by analyzing the second numeric derivative of the encoder position history), and $\delta\theta_z$ is the uncertainty in vertical orientation (estimated based on bubble level).

The following geometric uncertainty estimates were used: $\delta\theta_z = 1°$ for $\pm x$ and $\pm y$ orientations (i.e., sensors horizontally mounted) and $\delta\theta_z = 2°$ for $\pm z$ orientations (sensors mounted perpendicular to the turntable). $\delta r = 2$ mm, visually estimated through several placements. $\delta\omega$ and $\delta\alpha$, as described above, are trial-specific, estimated from each individual signal.

After obtaining the data, the time history files were trimmed to get the values for 3-axis accelerometers only when the motor is working at a constant velocity. The encoder data is trimmed too, low-passed, then differentiated to get velocity in radians/sec and compared with theoretical velocity for validity of the trimmed signal. It was then differentiated again, to get α and $\delta\alpha$.

Equation (5.1) clearly illustrates that one-step 3-axis calibration algorithm, with uncertainty quantification, may be used here. The following subsection presents results for the two accelerometers.

5.2.2 *Experimental Results*

The surface plots for transducer function at magnitudes $1.01g$, $1.04g$ and $1.32g$ are presented accordingly on Figs. 5.4, 5.5 and 5.6 below for accelerometer 1 (on the left) and accelerometer 2 (on the right).

The plots clearly show broad uncertainties, that still sometimes fail to capture the measurements. Presumably, the uncertainties are so high because of the potential issues due to the wire twist, as well as sensor misplacement.

Note though, that the fit does improve at higher accelerations. This is expected: the cost function is biased toward providing a better fit at higher accelerations.

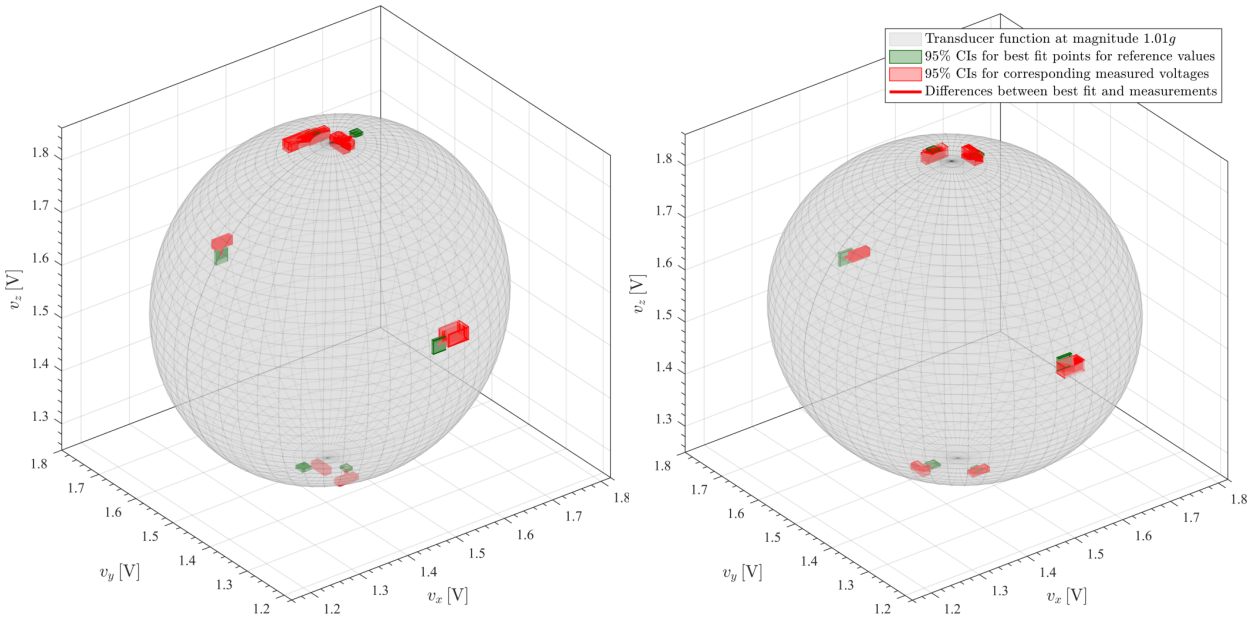

Fig. 5.4. Quality of fit plots for transducer functions at magnitude $1.01g$; accelerometers: 1 (left) and 2 (right)

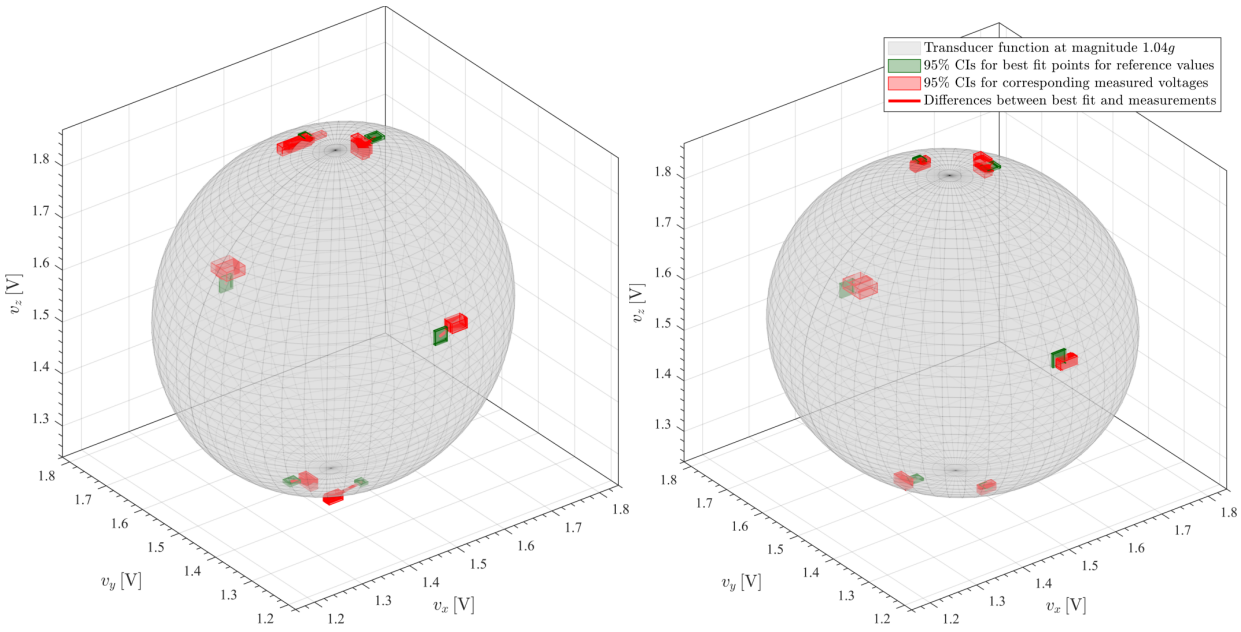

Fig. 5.5. Quality of fit plots for transducer functions at magnitude 1.04g; accelerometers: 1 (left) and 2 (right)

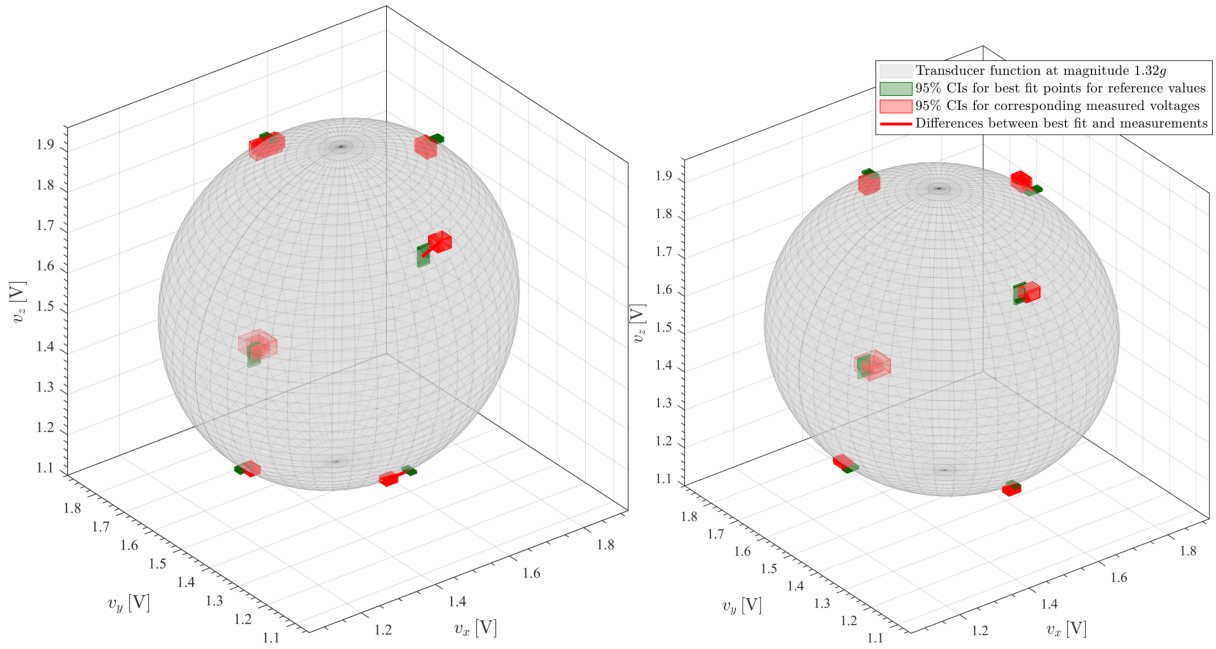

Fig. 5.6. Quality of fit plots for transducer functions at magnitude 1.32g; accelerometers: 1 (left) and 2 (right)

The computed calibration parameters, with uncertainties, are presented below.

Accelerometer 1 full sensor calibration matrix **A**:

$$\mathbf{A} = \begin{bmatrix} 3.53 & -0.060 & -0.169 \\ 0.12 & 3.341 & -0.148 \\ -0.41 & 0.054 & 3.536 \end{bmatrix} \frac{g}{\mathrm{V}} \pm \begin{bmatrix} 0.08 & 0.027 & 0.028 \\ 0.06 & 0.022 & 0.022 \\ 0.05 & 0.022 & 0.023 \end{bmatrix} \frac{g}{\mathrm{V}}. \tag{5.2}$$

Accelerometer 1 offset voltages $\vec{o}$:

$$\vec{o} = \begin{bmatrix} 1.480 \\ 1.493 \\ 1.562 \end{bmatrix} \mathrm{V} \pm \begin{bmatrix} 0.001 \\ 0.001 \\ 0.001 \end{bmatrix} \mathrm{V}. \tag{5.3}$$

Accelerometer 1 (direct, not inverse) sensitivities vector $\vec{k}$, obtained by inverting the diagonal of **A**, from Eqs. (2.3) and (2.4):

$$\vec{k} = \begin{bmatrix} 284 \\ 299 \\ 283 \end{bmatrix} \frac{\mathrm{mV}}{g}. \tag{5.4}$$

Accelerometer 1 misalignment angles, obtained from **A** and Eq. (2.4):

$$\begin{aligned} \alpha_{xz} &= 2.1^\circ \\ \alpha_{xy} &= 6.6^\circ \\ \alpha_{yz} &= 1.0^\circ \\ \alpha_{yx} &= 0.9^\circ \\ \alpha_{zy} &= -2.8^\circ \\ \alpha_{zx} &= 2.5^\circ. \end{aligned} \tag{5.5}$$

Accelerometer 2 full sensor calibration matrix **A**:

$$\mathbf{A} = \begin{bmatrix} 3.16 & 0.019 & 0.102 \\ -0.08 & 3.365 & 0.043 \\ 0.18 & -0.164 & 3.501 \end{bmatrix} \frac{g}{\mathrm{V}} \pm \begin{bmatrix} 0.05 & 0.024 & 0.020 \\ 0.05 & 0.022 & 0.019 \\ 0.04 & 0.022 & 0.018 \end{bmatrix} \frac{g}{\mathrm{V}}. \tag{5.6}$$

Accelerometer 2 offset voltages $\vec{o}$:

$$\vec{o} = \begin{bmatrix} 1.493 \\ 1.499 \\ 1.545 \end{bmatrix} \mathrm{V} \pm \begin{bmatrix} 0.001 \\ 0.001 \\ 0.001 \end{bmatrix} \mathrm{V}. \tag{5.7}$$

Accelerometer 2 (direct, not inverse) sensitivities vector $\vec{k}$:

$$\vec{k} = \begin{bmatrix} 317 \\ 297 \\ 285 \end{bmatrix} \frac{\mathrm{mV}}{g}. \tag{5.8}$$

Accelerometer 2 misalignment angles:

$$\begin{aligned}\alpha_{xz} &= -1.4^\circ \\ \alpha_{xy} &= -3.0^\circ \\ \alpha_{yz} &= -0.3^\circ \\ \alpha_{yx} &= -2.7^\circ \\ \alpha_{zy} &= -1.9^\circ \\ \alpha_{zx} &= -0.7^\circ.\end{aligned} \tag{5.9}$$

5.2.3 *Discussion*

Uncertainties are higher for the "Turntable" calibration method than for both "Box" experiments. This is most likely because of the higher chance of misplacing sensors on the rotational platform, as well as the errors from differentiation of the encoder's readings for calculating reference angular velocity. Overall, the offsets and the inverse sensitivities are close to technical specification, but not exactly the typical values.

5.2.4 *Comparison of "Box" One-step, "Box" Two-step and "Turntable" Calibration Methods*

Sections 4.2 and 4.3, and subsection 5.2.2, contain results for the static one- and two-step methods, and the turntable method, for 3-axis accelerometer calibration. In this subsection, I will now compare the three methods, and draw a conclusion about the best one.

Table 5.1 below contains the main advantages and disadvantages of all three calibration methods. It assumes that all 3 procedures are then processed using the appropriate uncertainty-quantifying algorithms in section 3.2.

Table 5.1. Comparison of the three calibration methods for 3-axis accelerometers

PROCEDURE	ADVANTAGES	DISADVANTAGES
"Box" Experiment One-step	• Simplest algorithm • Guarantees best possible fit if no reference value errors • Accurate • Solvable entirely with `mldivide`	• Limited range (only $\pm 1g$) • Requires precise alignment with gravity
"Box" Experiment Two-step	• Second simplest algorithm • Does not require precise alignment with gravity • Accurate	• Requires nonlinear optimization (measurably slower) • Requires precise rotation about an axis

"Turntable" Experiment	• Able to subject accelerometers to full range of values • Actual dynamic calibration	• Least accurate (most likely orientation errors underestimated) • Far more complex

It should be noted, that none of the 3 methods could be considered a complete failure: they all provided results that are close to the datasheet values, with the most significant variations in the misalignment angle sets. Judging by the quality of fit plots, the one-step "Box" experiment calibrated the misalignment angles the best, the two-step "Box" experiment, second best, and "Turntable," the worst. This could be considered expected: aligning the sensors on the turntable is substantially harder than aligning the wooden rectangular parallelepiped block.

As Table 5.1 suggests, the "Turntable" is, theoretically, the best calibration experiment, because it provides more control over reference velocities, and is also simply closer to the accelerometers' actual use case (i.e., they are used to track a **moving** robot). Unfortunately, the wire twist issues, and the difficulty of consistent placement, coupled with the uncertainties due to the encoder signals, make the turntable uncertainties too high, and likely also overestimate the misalignment angles.

The two-step experiment, surprisingly, also yielded higher misalignment angles, particularly for accelerometer 1. From quality of fit plots, we can also see, that its calibration did not appear to precisely match the reference values (even the ones used in its first step). This method probably suffered from the same caveat that Bonnet, et al. noted: the method is very sensitive to orientation precision during step 2.

Algorithmically, the two-step static method and the "Turntable" experiment are substantially more complex than the one-step static method, but this alone is not a compelling argument against them.

To summarize: I recommend the one-step "Box" experiment, as the best calibration method for 3-axis accelerometers, that I have tested. The turntable, though, is not useless – it is still the best way to calibrate not only gyroscopes, but also combined (i.e., IMU) devices.

5.3 1-AXIS GYROSCOPE CALIBRATION BY VARYING SPEEDS

5.3.1 *Description of Experiment*

Gyroscope calibration is similar to the velocity-based calibration for accelerometers. However, unlike accelerometers, gyroscopes are sensitive to the direction of rotation (accelerometers really only see centripetal acceleration and gravity, which are not dependent on the direction of rotation).

The experiment is essentially the same: gyroscopes are rotated with 7 different velocities, 5 trials per velocity, and the data is acquired by NI DAQ 6341 and processed via LABVIEW VI. The placement radius on the rotational platform is the same for all gyroscopes experiments – 13.8 cm, although this is actually unimportant: the angular rate is not a function of position on the surface.

Since for gyroscopes, the placement on the wood platform doesn't matter, the only source of uncertainty (other than the voltage noise) is the noise in encoder speed, computed via differentiation of encoder signal, as well as the mismatch between encoder average speed and the theoretical speed. Basically, the larger of the two should be the effective uncertainty for the angular speed – potentially different for each measurement. (In practice, however, encoder speed has a substantially higher noise amplitude than the difference between encoder average speed and theoretical speed).

The gyroscopes used in this work are 1-axis sensors, therefore the uncertainty-quantifying algorithm from subsection 3.2.1 should be used to process the data.

5.3.2 *Experimental Results and Discussion*

Running stochastic algorithm described in 3.2.1, we get the results shown on Fig. 5.7, where red bars show measurement uncertainty and green bars show reference uncertainty. The figure clearly shows a pretty much perfect linear fit, thus indicating that the calibration was completely successful.

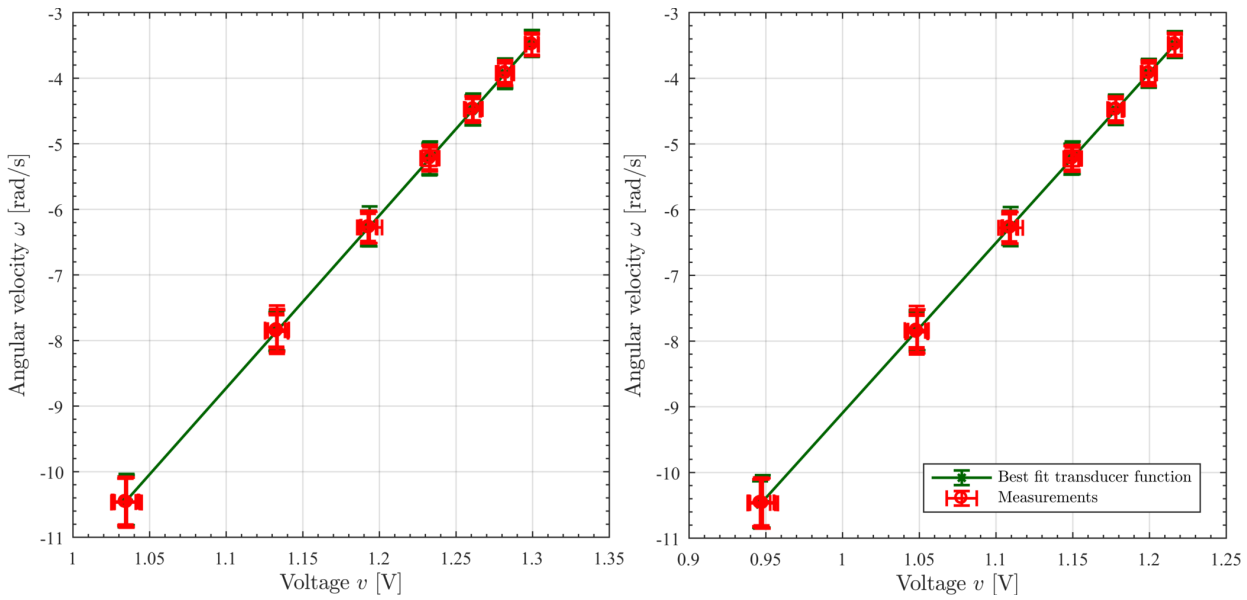

Fig. 5.7. Linear regressions with uncertainty quantifications for gyroscopes 1 (left) 2 (right)

The computed calibration parameters, with uncertainties, are presented below.

Gyroscope 1 inverse sensitivity s:

$$s = 26.3 \frac{\text{rad/s}}{\text{V}} \pm 0.7 \frac{\text{rad/s}}{\text{V}}. \tag{5.10}$$

Gyroscope 1 offset voltage o:

$$o = 1.431 \text{ V} \pm 0.005 \text{ V}. \tag{5.11}$$

Gyroscope 1 calculated (best estimate) sensitivity k:

$$k = 0.663 \frac{\text{mV}}{^\circ\text{/s}}. \tag{5.12}$$

Gyroscope 2 inverse sensitivity s:

$$s = 25.9 \frac{\text{rad/s}}{\text{V}} \pm 0.7 \frac{\text{rad/s}}{\text{V}}. \tag{5.13}$$

Gyroscope 2 offset voltage o:

$$o = 1.351 \text{ V} \pm 0.005 \text{ V}. \tag{5.14}$$

Gyroscope 2 calculated (best estimate) sensitivity k:

$$k = 0.674 \frac{\mathrm{mV}}{^{\circ}/\mathrm{s}}. \tag{5.15}$$

All of the above results are nearly identical to the datasheet values, with the exception of the difference in offsets. This is unexpected but the gyroscopes, otherwise, clearly work, so this should not be a problem.

This calibration also shows, that the turntable performed fine for gyroscope calibration. Gyroscope signals are much cleaner than accelerometers, and the references are not placement-dependent, so it is expected, that the calibration would be more successful, with lower uncertainties (mostly from reference velocities).

Chapter 6
External Measurement System Testing on a "Hopper" Robot

The External Measurement System for robot testing, developed in this work, consists of a selection of sensors, with associated data acquisition systems, and carefully specified uncertainty-quantifying calibration processes for them. The system's hardware is detailed in Appendix A, the best calibration procedures in subsections 4.1.2 and 5.3.1, and the corresponding calibration algorithms in subsections 3.2.2 and 3.2.1. The purpose of this chapter, therefore, is to link it all together, and test the resulting EMS on a real robot. For comparison, besides the inertial sensor-based EMS, a high-speed motion camera, and the robot's onboard encoders, are used for comparison.

6.1 Experimental Setup

The experimental setup for testing the EMS on a Hopper robot is presented on Fig. 6.1 below. The setup consists of:

- 80/20 aluminum frame for support of the harness.
- Linear motion guide (LMG).
- Steel angles for attaching the Hopper to the LMG. The Hopper is designed to be a single leg of a larger robot, and so when testing it individually, translational vertical movement is reasonable restriction.

- Benchtop programmable power supply unit, B&K Precision DC power supply 9115, to power the motors.
- The rest of the EMS (see Appendix A).
- The Hopper robot itself.

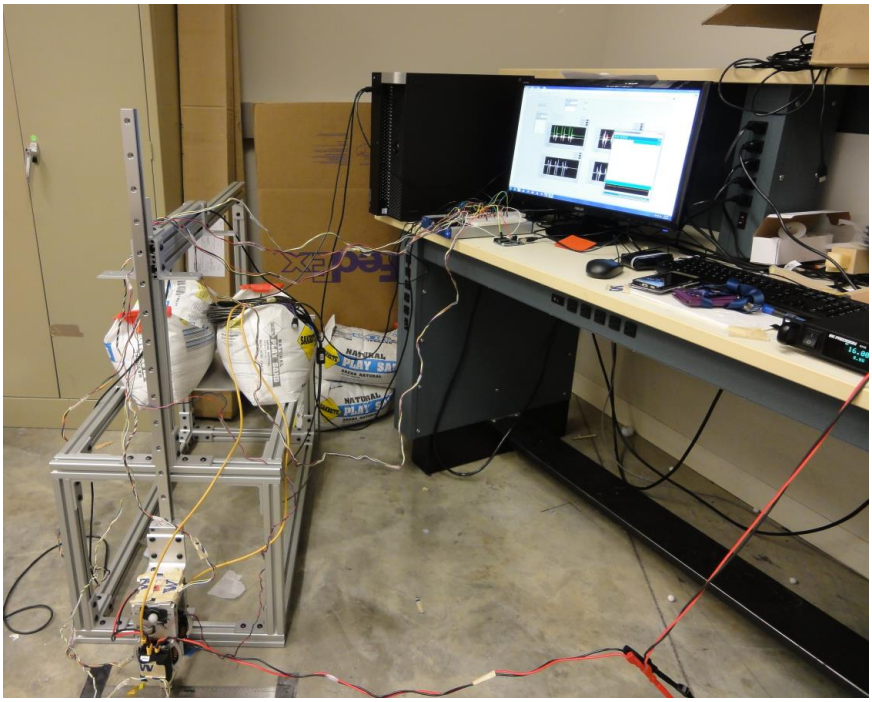

Fig. 6.1. Experimental setup for testing the EMS on the Hopper robot, linked to a vertical slider harness

The Hopper robot is designed by Ghost Robotics LLC, and is described in Ref. [44]. It is a 2-degree of freedom pin-jointed 5-bar linkage, driven by two identical motors (same as the turntable motor in chapter 5). For 1-dimensional motion, the motors rapidly rotate in the opposite directions, causing the robot to jump. Precisely and synchronously tracking the robot's individual links, while also timing its jump, its rest, and its travel distance, is a non-trivial problem. The robot is illustrated on Fig. 6.2 below.

A ball-bearing LMG was chosen to minimize dynamic friction. The harness and the robot attachment plate were designed by me, and the frame of the harness was built by Jacob Baldassini [45].

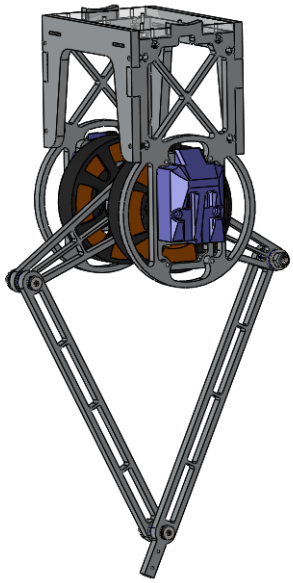

Fig. 6.2. Hopper robot CAD model
(provided by Gavin Kenneally of University of Pennsylvania, PA, USA)

The robot's motors are powered by the B&K Precision DC power supply 9115. Gyroscope 1 is placed directly on one of the motors, to see if the encoders would suffer (i.e., skip) due to the shock the robot experiences during liftoff and when it hits the ground. Gyroscope 2 is placed on the link attached to motor 2, for the same reason.

Accelerometer 1 is placed on the robot's main body, with its z-axis pointing vertically upward. Accelerometer 2 is placed on the robot's foot, with its y-axis pointing vertically upward when the robot is at rest (i.e., the stationary standing position). The purpose of accelerometer 1 placement was to see if it would be possible to accurately track the robot's main body's trajectory as a function of time, using only the accelerometer, and without relying on the robot's dynamic model. The purpose of accelerometer 2 in this setup was primarily to try to study the directions and amplitudes of vibrations that the foot experiences on impact, and also to precisely time the robot's resting period (not a theoretical quantity: just because the motors are moving does not meant that the foot is, it may be still touching the ground).

The accelerometers and the gyroscopes are connected in the same fashion they are in the previous experiments in this work. Due to a limited number of ports on the DAQ, the gyroscopes were instead powered by an Arduino UNO board (the data is still acquired by the DAQ and processed by the LabVIEW VI).

To provide a "reference" position measurement, a Qualisys Oqus high-speed camera was used. The camera works by tracking the positions of reflective markers, which were placed on the robot's main body, and on one of the lower links.

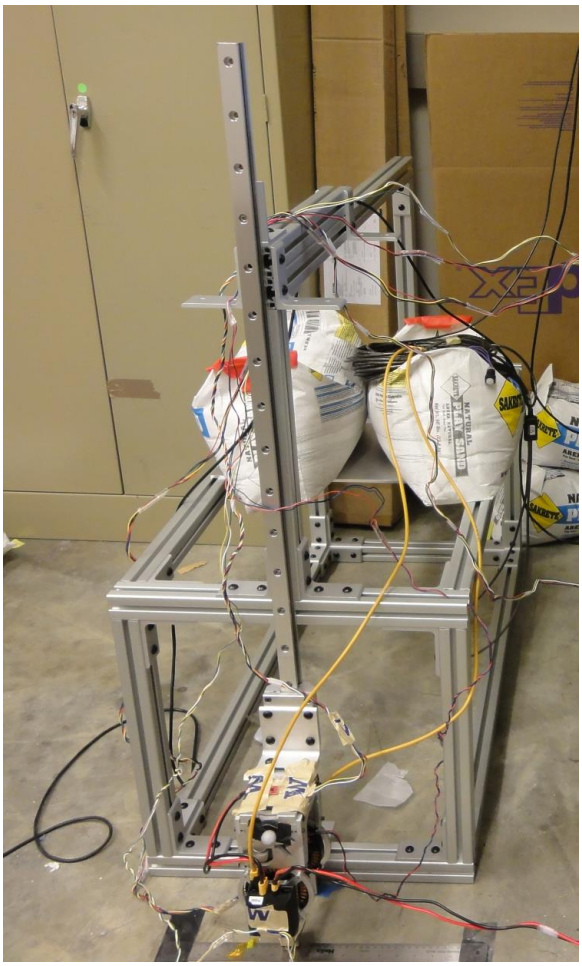

Fig. 6.3. Vertical slider harness for Hopper robot
(harness and attachment plate designed by me; harness built by Jacob Baldassini [45])

The algorithms used to process the data gathered during the tests from the EMS and other sources are discussed in section 6.2.

6.2 Data Processing Algorithms

To get position from an accelerometer signal, a double time integral is needed. Furthermore, initial position and velocity are technically required. The Hopper's motion is periodic, and so as long as our integration period starts from a rest interval, we can effectively treat this starting position as $z = 0$, and stationary.
Numeric integration can be done by different methods: e.g., rectangular (i.e., Riemann sum) integration, trapezoidal rule, Simpson's rule. To integrate the acceleration signals, I choose the trapezoidal rule: it's one of the best, and definitely the fastest, method for integrating discrete signals. MATLAB function for trapezoidal rule integration is **`cumtrapz`**.
Unfortunately, accelerometers have drift related to the DC bias, which leads to the integration errors. Prior to integrating the accelerometer signal at all, it gets filtered by the LabVIEW VI, to apply some smoothing. A simple moving average filter is used. After the first integration, we get a ramped signal because of the DC bias, and after the second integration of the ramped signal, the error grows quadratically in time. To solve the problem of DC bias, high-pass filtering is needed [46].
In this MS Thesis, type I Chebyshev high-pass filter is chosen to filter out DC bias. The algorithm to get position from acceleration, overall, looks as follows (again, recall that it assumes that the starting state is $z = 0$, $\dot{z} = 0$):

1. The raw discrete data from accelerometers processes via smoothing filter in the LabVIEW VI, in real-time, to smooth the high frequency noise that arises due to high sampling frequency.
2. The data from step 1 is processed via type I Chebyshev high-pass filter with low cutoff frequency, to eliminate the DC bias.
3. The data from step 2 is integrated using the trapezoidal method, to get the velocity signal from the acceleration.
4. The data from step 3 processes via the same filter as described in step 2 – type I Chebyshev high-pass filter.
5. The data from step 4 is integrated again, using the trapezoidal method, to obtain the position signal.

Besides type I Chebyshev, I have attempted a number of other high-pass filters:

- Butterworth high-pass filter. This filter seriously damaged the signal amplitude, and required compensation after integration.

- Elliptic high-pass filter. This filter also gave worse results, but was not as damping as Butterworth high-pass.
- Type II Chebyshev high-pass filter. This filter gave essentially the same results as type I Chebyshev.

Similarly to the above transition from acceleration to velocity time signal, the acceleration data from the high-speed camera is also calculated by double differentiating its position signal. The velocity data is calculated by single-differentiating the position signal.

Note also, that like the inertial sensors, the high-speed camera also requires calibration. It is calibrated by placing a ruler with reflective stickers as far from the camera's lens as the robot that it is about to track. The calibration helps the camera convert the pixel data to physical units of length (its outputs are in centimeters). As was discussed in subsection 2.3.2, the camera suffers from occlusions, and also from reflectivity of the robot's metal parts. This necessitates considerable fine-tuning of camera's configuration, like intensity and frame rate.

The robot's encoders were sampled in the same fashion the turntable's motor's encoder was. Again, one of the objectives of the test was to see if the encoders suffered from shock when impacting the ground. To compare the gyroscope and encoder data, the encoder's angular position signal was differentiated to get angular velocity, which is also the physical quantity that the gyroscopes measure.

6.3 EXPERIMENTAL RESULTS

Figures 6.4 and 6.5 below illustrate the 3 different estimates for the position of the upper body, obtained using the 3 different calibration methods' results, on the same signal, and processing it as described in section 6.2.

All 3 plots clearly show that the "Box" experiment results are essentially identical, with the difference of less than 1 mm, which can be attributed purely to noise. The turntable calibration clearly performed worse. We can also see that while the accelerometers accurately track the robot as it rises, during the fall the high-pass filters cause the signal to drop much slower than it should. This is an artifact of high-pass filtering: the position drifts, because the "stationary" component of the signal – i.e., the robot standing essentially still – is filtered out – is removed.

We can also see that the uncertainties do not appear to propagate themselves into the position estimate (the curves essentially overlap the best estimate). This is actually expected: as is shown later in this section (Fig.

6.7), the accelerometer uncertainty margins are nearly constant around the signal. Thus, they can be viewed as near-zero frequency components of the acceleration signal – which the high-pass filters immediately remove.

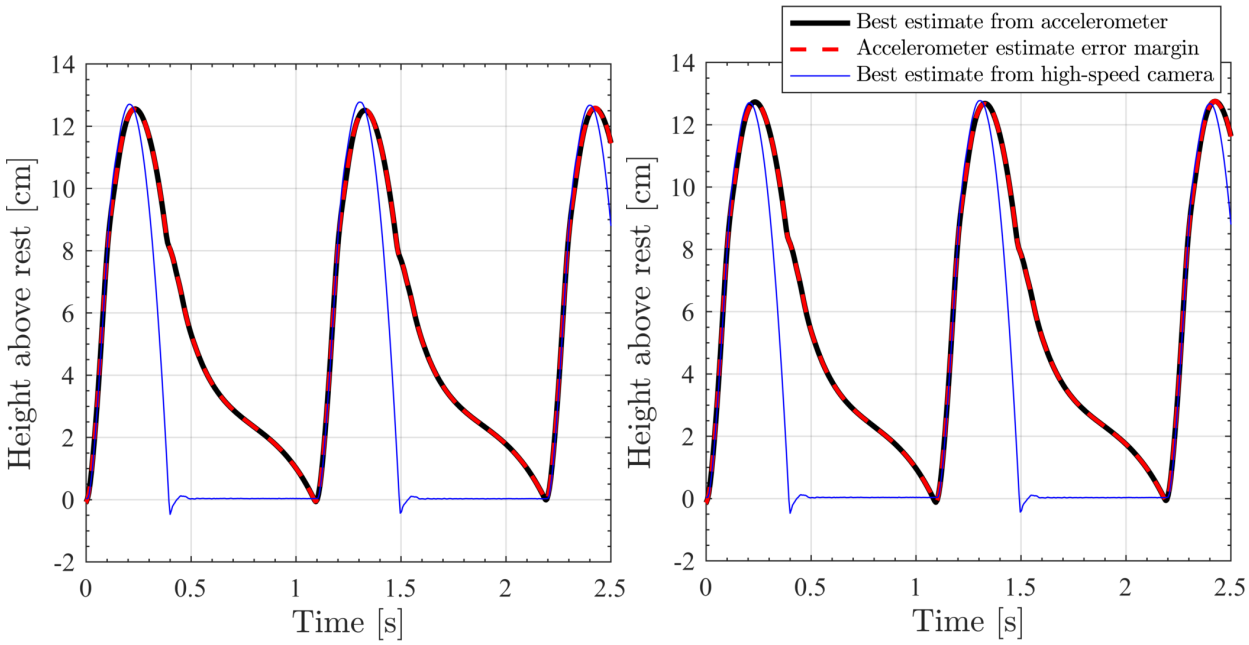

Fig. 6.4. Height of main body above rest, using one-step (left) and two-step (right) “Box” calibration methods

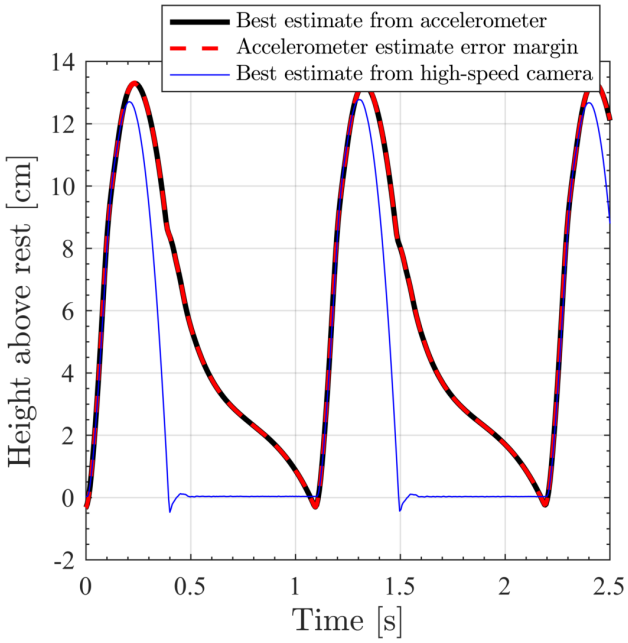

Fig. 6.5. Height of main body above rest, using turntable calibration method

The position measurements resulting from the better ("Box") calibrations are clearly accurate enough in the amplitude, and in timing, but do not track the position accurately throughout the transient. They would certainly be superior for a constant moving part, as the high-pass filter would then not end up removing an important component of the signal. Overall, what Figs. 6.4 and 6.5 above show is not an issue with the accelerometers, or their calibrations, but rather an issue with the filtering algorithms of section 6.2. This is clearly an area with a potential for future improvement.

Figure 6.6 below provides a comparison of the velocity signals provided by the gyroscopes, and by differentiating the motors' encoder signals. We can see, for the gyroscope mounted on the motor (motor 1), the reading is clearly fairly noisy, and the uncertainties in particular reflect this. The gyroscope mounted on the leg itself provides a much cleaner signal, that is actually smoother (and likely more accurate) than the differentiated encoder signal, which suffers from numeric differentiation errors.

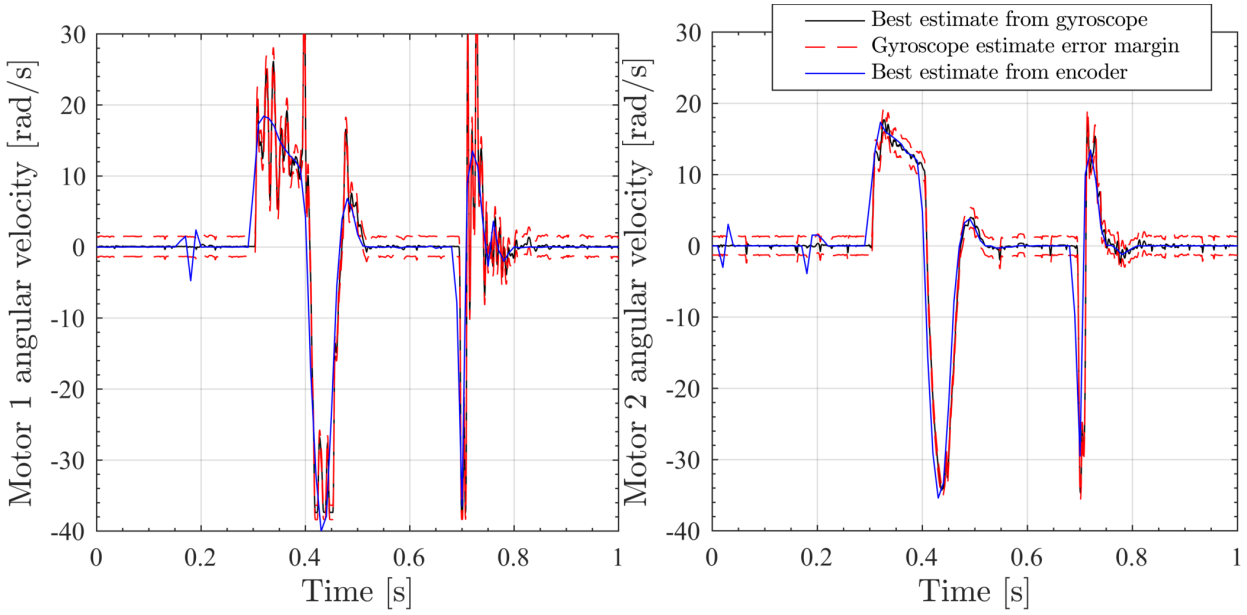

Fig. 6.6. Angular velocities read by gyroscopes and encoders: motors 1 (left) and 2 (right)

The vertical acceleration of the main body, obtained from high-speed camera by double differentiation and acceleration from accelerometer 1 (using one-step "Box" experiment calibration) are presented on Fig. 6.7 below. As we can see from Fig. 6.7, acceleration obtained from high-speed camera is distorted to compare with the raw data from accelerometer, due to double differentiation error.

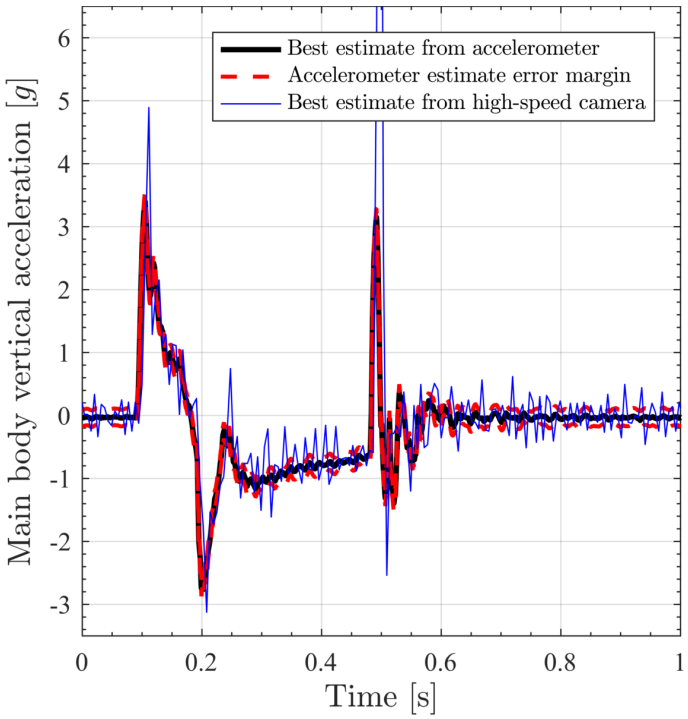

Fig. 6.7. Main body vertical acceleration, reported by accelerometer and high-speed camera

6.4 Discussion

From results presented in section 6.3, we can conclude the following about the inertial sensor-based external measurement system developed in this thesis:

1. The EMS is clearly best for directly measuring the dynamics of the robot. Position-based instruments, both external (high-speed camera) and onboard (motor encoders) are clearly less precise, and far noisier, than the EMS's sensors, assuming proper calibration was performed.
2. The EMS is acceptable for tracking position amplitudes (particularly during fast motion), but with the signal processing algorithms attempted in section 6.2 (in particular, type I Chebyshev high-pass filters), the EMS suffers from not being able to follow zero velocity. This is not a hardware flaw: instead, it is an artifact of the numeric double integration, coupled with a high-pass filter. An improved filtering algorithm is likely possible, which would make the EMS significantly more useful for position tracking.

3. One-step "Box" experiment was again confirmed to be the superior calibration method for accelerometer calibration, although two-step was also very good. Turntable calibration was demonstrated to be insufficient, at least with the way it was performed in chapter 5.
4. The high-speed camera, in particular, turned out to suffer badly from differentiation errors: in one instance, it overestimated the acceleration by over a factor of 7. This clearly shows, that the camera is not, alone, anywhere near sufficient for estimating the stresses a robot experiences, or the robot's dynamics; it was, however, clearly the superior tool for tracking 1-dimensional (and likely, also 2-dimensional) position, at least with the accelerometer signal processing algorithms that were attempted.

The EMS was, in fact, helpful for verifying several modeling assumptions, such as the assumption that encoders do not suffer from impact shock, and do not "skip." This is precisely the purpose of the EMS: it is intended for testing robots in a laboratory setting, and not for robot control.

Chapter 7
Conclusions and Recommendations for Future Work

7.1 Conclusions

One of the main contributions for this MS Thesis is a development a new calibration method that based on existing , and modified, sensors calibration methods and Monte Carlo simulations algorithm for uncertainty quantification. Based on a developed new calibration algorithm for sensors, we can define not only sensors parameters like offsets, sensitivities and misalignment angles but also uncertainties in each of the parameters. Three calibration methods are used for accelerometers: static one- and two-step methods, and dynamic turntable-based method. All of these three methods are used with Monte Carlo simulations algorithm. As results show, the most accurate procedure turned out to be the one-step static method, however this method depends on orientation of gravity vector. The second simplest method is the two-step static method, but it depends on nonlinear optimization which slows down the whole algorithm. It also depends on rotation about a fixed axis, which can be challenging, and can introduce systematic errors. "Turntable" method is the hardest because it requires special equipment like a motor, the rotational platform, and it requires to take into account multiple references: centripetal acceleration, gravity and tangent acceleration. However, "Turntable" equipment could be used also for gyroscopes calibration, and calibration of both accelerometers and gyroscopes could be done simultaneously on the same equipment.

External Measurement System that has two 3-axis accelerometers and two 1-axis gyroscopes is tested on a Hopper robot. Such robot is planar and jumping only vertically due to a rigid harness - linear motion guide. Performance of External Measurement System is compared with high-speed motion camera data and robot's encoders. The results show that External Measurement System has a good potential to be a complimentary system for high-speed motion camera. External Measurement System is superior in direct measurements (acceleration, angular velocity) compared to differentiated signals from high-speed motion camera and encoders. The developed system is good for estimating attitude of a jump of the robot and a period of jump, but cannot accurately predict position trajectory versus time, due to drifting properties of accelerometers, and the signal processing methods used.

7.2 RECOMMENDATIONS FOR FUTURE WORK

To improve the external measurement system, I suggest:

- Improve the signal processing methods for position tracking. This'll make the EMS far more useful.
- Multi-IMU networking instead of the separated accelerometers and gyroscopes.
- Multi-IMU networking could be a good substitution for the separated sensors, however it needs to consider transmission capability of both available digital protocols for such sensors: SPI and I2C.
- Artificial Neural Networking (ANN) for learning robot's dynamics based on sensors' data fusion.
- In Ref. [40], authors tried to use artificial neural networking for calibration IMU on turntable. They got satisfactory value 67%. Authors believe that ANN has a good potential for calibration sensors.
- Sophisticated user interface with real-time data processing and displaying the information.
- LabVIEW VI have a potential to develop the real-time data processing tool, not only for collecting the data, but also for processing the data, including of applying filters to all sensors data, conversion from voltages/digital values to real physical values, implementing calibrating algorithm and outputting to the user the signals' plots in real-time.
- Fully automated calibration process: design and build the experimental setup for calibration procedures.

Fully automatic calibration set up could be build that includes not only as described above LabVIEW VI as user interface program, but also could include fully-automatic platform for calibration of the sensors with automatic data processing. Such calibration platform should have multiple degrees of freedom to calibrate multi-axis sensors.

Appendix A. List of Equipment Used

1. **B&K Precision, Multi-Range Programmable DC Power Supply, Model 9115.**
2. **National Instruments, Data Acquisition Tool, Model NI DAQ USB-6341.**
3. **U-POWER, Tiger Motor U8-16, 100kv.**
4. **2 × Analog Devices, MEMS accelerometer ADXL335.**
5. **2 × Cytron, Single axis Gyroscope SN-ENC03R**.
6. Qualisys, Motion capture camera Oqus.
7. Ghost Robotics, Hopper robot.
8. **Qooltek, Multipurpose Laser Level Measurement Tool, LASER LEVELPRO3.**

The bolded entries are the hardware components of the EMS, necessary either for operation, or for calibration.

References

[1] R. M. Rogers, Applied Mathematics in Integrated Navigation Systems, 3rd ed., Reston, VA, USA: American Institute of Aeronautics and Astronautics, 2007.

[2] M. Elwenspoek and R. Wiegerink, Mechanical Microsensors, Berlin, Germany: Springer-Verlag Berlin Heidelberg, 2001.

[3] A. V. Oppenheim and R. W. Schafer, Discrete-Time Signal Processing, 3rd ed., Upper Saddle River, NJ, USA: Pearson Higher Education, 2010.

[4] F. Perea, Arduino Essentials, Birmingham, UK: Packt Publishing, 2015.

[5] Á. Seiça, "The transducer function: an introduction to a theoretical typology in electronic literature and digital art," *Journal of Science and Technology of the Arts,* vol. 4, no. 1, 2012.

[6] T. G. Beckwith, R. D. Marangoni and J. H. Leinhard V, Mechanical Measurements, 6th ed., Upper Saddle River, NJ, USA: Pearson Prentice Hall, 2007.

[7] R. Zhang, F. Höflinger and L. M. Reindl, "Calibration of an IMU using 3-D rotation platform," *IEEE Sensor Journal,* vol. 14, no. 6, pp. 1778-1787, 2014.

[8] "Sparkfun Electronics ADXL335 datasheet," [Online]. Available: http://www.sparkfun.com/datasheets/Components/SMD/adxl335.pdf. [Accessed 1 May 2017].

[9] S. Bonnet, C. Bassompierre, C. Godin, S. Lesecq and A. Barraud, "Calibration methods for inertial and magnetic sensors," *Sensors and Actuators A: Physical,* vol. 156, no. 2, pp. 302-311, 2009.

[10] C. M. Cheuk, T. K. Lau, K. W. Lin and Y. Liu, "Automatic calibration for inertial measurement unit," in *12th International Conference on Control Automation Robotics & Vision (ICARCV)*, Guangzhou, China, 2013.

[11] S. Levy, A. E. McPherson and E. V. Hobbs, "Calibration of accelerometers," *Journal of Research of the National Bureau of Standards,* vol. 41, pp. 359-369, 1948.

[12] Y. Kamer and S. Ikizoglu, "Effective accelerometer test beds for output enchancement of an inertial navigation system," *Measurement,* vol. 46, no. 5, pp. 1641-1649, 2013.

[13] Ø. Magnussen, M. Ottestad and G. Hovland, "Experimental validation of a quaternion-based attitude estimation with direct input to a quadcopter control system," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, Atlanta, GA, USA, 2013.

[14] M. Glueck, D. Oshinubi and Y. Manoli, "Automatic real-time offset calibration of gyroscopes," *Microsystem Technologies,* vol. 21, no. 2, pp. 429-443, 2015.

[15] S. Weiss, M. W. Achtelik, M. Chli and R. Siegwart, "Versatile distributed pose estimation and sensor self-calibration for an autonomous MAV," in *IEEE International Conference on Robotics and Automation*, St. Paul, MN, USA, 2012.

[16] S. Thrun, W. Burgard and D. Fox, Probabilistic Robotics, Cambridge, MA, USA: The MIT Press, 2005.

[17] M. Bloesch, M. Hutter, M. A. Hoepflinger, S. Leutenegger, C. Gehring, C. D. Remy and R. Siegwart, "State estimation for legged robots - consistent fusion of leg kinematics and IMU," in *Robotics: Science and Systems VIII*, Sydney, Australia, 2012.

[18] B. Olofsson, J. Antonsson, H. G. Kortier, B. Bernhardsson, A. Robertsson and R. Johansson, "Sensor Fusion for Robotic Workspace State Estimation," *IEEE/ASME Transactions on Mechatronics,* vol. 21, no. 5, pp. 2236-2248, 2016.

[19] J. Santolaria, F.-J. Brosed, J. Velazquez and R. Jimenez, "Self-alignment of on-board measurement sensors for robot kinematic calibration," *Precision Engineering,* vol. 37, no. 3, pp. 699-710, 2013.

[20] M. El-Gohary and J. McNames, "Human joint angle estimation with inertial sensors and validation with a robot arm," *IEEE Transactions on Biomedical Engineering,* vol. 62, no. 7, pp. 1759-1767, 2015.

[21] G. Alici and B. Shirinzadeh, "A systematic technique to estimate positioning errors for robot accuracy improvement using laser interferometry based sensing," *Mechanism and Machine Theory,* vol. 40, no. 8, pp. 879-906, 2005.

[22] J. A. Paradiso, "Some novel applications for wireless inertial sensors," in *NSTI Nanotechnology Conference and Trade Show*, Boston, MA, USA, 2006.

[23] P. Cheng and B. Oelmann, "Joint-angle measurement using accelerometers and gyroscopes - a survey," *IEEE Transactions on Instrumentation and Measurement,* vol. 59, no. 2, pp. 404-414, 2010.

[24] M. Quigley, R. D. Brewer, S. P. Soundararaj, V. Pradeep, Q. V. Le and A. Y. Ng, "Low-cost accelerometers for robotic manipulator perception," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, 2010.

[25] Y. Veryha, L. Kourtch, B. Verigo and V. Murashko, "Method and system for robot end effector path correction using 3-D ultrasound sensors," in *3rd World Congress on Intelligent Control and Automation (WCICA)*, Hefei, China, 2000.

[26] F. Tatar, J. Mollinger and A. Bossche, "Ultrasound system for measuring position and orientation of laparoscopic surgery tools," in *IEEE Sensors*, Toronto, Canada, 2003.

[27] J. Li, Q. Huang, W. Zhang, Z. Yu and K. Li, "Real-time foot attitude estimation for a humanoid robot based on inertial sensors and force sensor," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Bangkok, Thailand, 2009.

[28] J. Lobo and J. Dias, "Relative pose calibration between visual and inertial sensors," *The International Journal of Robotics Research,* vol. 26, no. 6, pp. 561-575, 2007.

[29] N. Song, Q. Cai, G. Yang and H. Yin, "Analysis and calibration of the mounting errors between inertial measurement unit and turntable in dual-axis rotational inertial navigation system," *Measurement Science and Technology,* vol. 24, no. 11, 2013.

[30] N. Grip and N. Sabourova, "Simple non-iterative calibration for triaxial accelerometers," *Measurement Science and Technology,* vol. 22, no. 12, 2011.

[31] J. R. Taylor, An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements, 2nd ed., Sausalito, CA, USA: University Science Books, 1997.

[32] A. L. Peressini, F. E. Sullivan and J. J. Uhl, The Mathematics of Nonlinear Programming, New York, NY, USA: Springer-Verlag New York, 1988.

[33] C. G. Broyden, "The convergence of a class of double-rank minimization algorithms," *Journal of the Institute of Mathematics and its Applications,* vol. 6, no. 1, pp. 76-90, 1970.

[34] R. Fletcher, "A new approach to variable metric algorithms," *The Computer Journal,* vol. 13, no. 3, pp. 317-322, 1970.

[35] D. Goldfarb, "A family of variable metric updates derived by variational means," *Mathematics of Computation,* vol. 24, no. 109, pp. 23-26, 1970.

[36] D. F. Shanno, "Conditioning of quasi-Newton methods for function minimization," *Mathematics of Computation,* vol. 24, no. 111, pp. 647-659, 1970.

[37] J. Lagarias, J. A. Reeds, P. E. Wright and P. E. Wright, "Convergence properties of the Nelder-Mead simplex method in low dimensions," *SIAM Journal of Optimization,* vol. 9, no. 1, pp. 112-147, 1998.

[38] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of Applied Mathematics,* vol. 2, no. 2, pp. 164-168, 1944.

[39] D. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *SIAM Journal on Applied Mathematics,* vol. 11, no. 2, pp. 431-441, 1963.

[40] S. İkizoğlu and Y. Kamer, "Acceleration data correction of an inertial navigation unit using turntable test bed," in *World Congress on Electrical Engineering and Computer Systems and Science (EECSS)*, Barcelona, Spain, 2015.

[41] A. C. Rencher and W. F. Christensen, Methods of Multivariate Analysis, 3rd ed., Hoboken, NJ, USA: John Wiley & Sons, 2012.

[42] D. C. Lay, Linear Algebra and Its Applications, 3rd ed., Boston, MA, USA: Pearson Addison-Wesley, 2002.

[43] S. H. Lee and W. Chen, "A comparative study of uncertainty propagation methods for black-box-type problems," *Structural and Multidisciplinary Optimization,* vol. 37, no. 3, 2009.

[44] G. Kenneally and D. E. Koditschek, "Leg design for energy management in an electromechanical robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, 2015.

[45] J. Baldassini, "An examination of the effects of deformable foam contact surfaces on robotic locomotion," University of Washington, Seattle, WA, USA, 2017.

[46] L. D. Slifka, "An Accelerometer Based Approach to Measuring Displacement of a Vehicle Body," University of Machigan-Dearborn, Dearborn, MI, USA, 2004.