Sensor fusion for estimating joint kinematics and kinetics of biomechanical systems

Validation using a robotic manipulator

J. Boelens

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Sensor fusion for estimating joint kinematics and kinetics of biomechanical systems

Validation using a robotic manipulator

Master of Science Thesis

For the degree of Master of Science in Systems and Control and Mechanical Engineering at Delft University of Technology

J. Boelens

May 27, 2022

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

The study of human motion consists of the analysis of kinematics, dealing with joint angles, velocities and accelerations, and kinetics, which deals with joint torques and interaction forces. Traditionally, kinematics and kinetics are estimated from optical marker data and sometimes measured interaction forces. Inertial measurement units (IMUs) were introduced as a low cost alternative that makes measurements outside the laboratory possible. Measurements are often used in a loosely coupled combination of marker-based inverse kinematics (IK) or orientation-based inverse kinematics (OBIK) and inverse dynamics (ID). The orientations for OBIK are first estimated from IMU data. Recently, tightly coupled estimation methods based on nonlinear state space models have been introduced for both sensor modalities to improve accuracy. Sensor fusion of marker and IMU data is a relatively unexplored area of research. Markers provide a measurement on the position level whereas IMUs measure on the velocity and acceleration levels. Fusing the two sensor modalities is hypothesized to result in the most accurate estimation of kinematics and kinetics. The objective of this thesis is to investigate the effects of the sensor modality and the estimation method on the accuracy of estimated kinematics and kinetics. An iterated extended Kalman filter (IEKF) was developed which estimates joint angles, velocities and torques using any combination of marker data, IMU data and measured interaction forces. To validate the system with respect to ground truth, two experiments (fast motion and interaction force) were carried out on a robot equipped with all three sensor types. For comparison, IK, OBIK and a marker/IMU-based kinematic extended Kalman filter (EKF) were used together with ID to estimate kinematics and kinetics as well. The IEKFs estimated joint angles with a root mean square error (RMSE) between 0.33° (markers and interaction forces only) and 1.58° (IMUs only, fast motion). The marker-based IEKF and the IMU-based IEKF outperformed IK and OBIK respectively in both experiments. The IMU-based IEKF improved joint velocity RMSE from 4.13 deg/s to 1.10 deg/s in the fast motion experiment. Sharp peaks in the joint torque signal were only estimated accurately using IMU data directly. The marker/IMU IEKF was the only method that estimated joint angles, velocities and torques with RMSEs below 1°, 1.5 deg/s and 1.5 Nm respectively in both experiments. It is concluded that using IMU data in a tightly coupled system improves joint velocity and torque estimation. Assuming accurate sensor registration and no soft tissue artifacts, a tightly coupled IMU-only method can be sufficiently accurate in practice.

Table of Contents

List of Figures

List of Tables

Preface

This thesis project came to be after I met dr. Manon Kok, department of Systems and Control, and dr. ir. Ajay Seth, department of BioMechanical Engineering. During my time at TU Delft, I have gained an appreciation for the modeling of complex systems. Finding a mathematical description that is as simple as possible, while still capturing the complexities of the real world, is as much an art as it is a science. When I started my master's degree, I was excited to apply this knowledge in robotics. As I learned more about the field of biomechanics, I also became fascinated with the human body as a mechanical system. As engineers, we can only dream to one day make a machine as complex and fine-tuned as our own body. When it became time to write my master's thesis, I was immediately captivated by this project. The fact that we can use our knowledge of robotics to improve our understanding of the human body is a great example of how these two fields can learn from each other. Furthermore, this project has given me the opportunity to learn new skills in modeling, experimental design and data processing.

Acknowledgements

First and foremost, I owe my gratitude to my two supervisors, dr. Manon Kok and dr. ir. Ajay Seth. They provided the opportunity to discuss my progress and ask for help frequently, despite their busy schedules. They gave me the opportunity to define my own project and to set my own goals, they provided guidance when I needed it and they challenged me to get the most out of this project. With their help, I was able to make this thesis into something I am proud of.

My thanks also go to dr. J. Micah Prendergast, who kindly provided me with much of his time to help me design, plan and execute the experiments you will read about in this thesis. Without his knowledge and expertise I would not have been able to get many of the results that made this project into a success for me.

Furthermore, I would like to thank all my fellow students with whom I had many interesting discussions over the course of my thesis project. I am thankful to the sensor fusion group at the DCSC department and the computational biomechanics group at the BME department in particular. They gave me new insights, writing advice and sometimes just a much wanted distraction from the many hours spent writing this thesis. My friends and colleagues Thijs and Dirk-Jan deserve my gratitude for their patience in proofreading many versions of this document and correcting my mistakes.

Finally, I want to thank my family, my roommates and my friends for their patience and never-ending support during this sometimes difficult year. They helped me put things in perspective and motivated me to keep moving forward.

Delft, University of Technology
May 27, 2022

J. Boelens

“I tore myself away from the safe comfort of certainties through my love for truth - and truth rewarded me.”

— *Simone de Beauvoir*

Chapter 1

Introduction

The motion of humans and animals has fascinated people for decades. In fact, one of the first examples of high-speed photography was invented to study the gallop of horses [1]. Human motion analysis in particular is a broad field of research that covers a variety of applications and methods. Its applications range from computer animation to sports research and diagnostics. Each application focuses on different aspects of human motion and therefore has different requirements. In the most general sense, the analysis of human motion can be divided into the study of *kinematics* and *kinetics*. The study of kinematics is only concerned with the motion as such, that is the positions, orientations, velocities and accelerations of the body segments. The study of kinetics considers quantities such as forces, moments and power. This allows biomechanicists to study the functioning of muscles or muscle groups. In sports research, the kinetics of high-intensity motion in particular have been tied to the risk of injury [2]. In a clinical setting, the kinetics of motion may be used to inform the clinician's decision with regard to treatment [3]. Kinetic quantities may be derived from a dynamic model of the body and the kinematics of the motion. They may also be measured directly using force plates, force/torque sensors et cetera [4].

Modeling of the kinematics and dynamics of the human body is the primary purpose of OpenSim. OpenSim is an open-source software system to create and analyze dynamic simulations of movement [5,6]. In OpenSim, the human body is modeled as a chain or tree of rigid bodies, connected by joints [7,8]. The motion of soft tissue attached to the rigid bodies, like muscles and skin, is usually not accounted for in this model. This can introduce a mismatch between modeled kinematics and observed motion, referred to as soft tissue artifacts (STA). In such a *kinematic tree*, one body segment forms the base. Each other body segment is connected to another through one or more joints. Multiple types of joints, such as pin joints, hinge joints and more complex types are available. The degrees of freedom (DoFs) of these joints describe how the body segments can move through space.

When the model is described as a kinematic tree, its configuration can be expressed with a set of *generalized coordinates*. Typically, the generalized coordinates are related to the DoFs of the joints. When analyzing the kinetics of motion, the kinematic tree formulation is advantageous because it allows transforming muscle forces and external forces like gravity to

(a) A reflective marker for motion capture.

(b) An infrared camera for motion capture from the OptiTrack motion capture system that was used in this thesis.

Figure 1-1: The hardware required for marker-based motion capture.

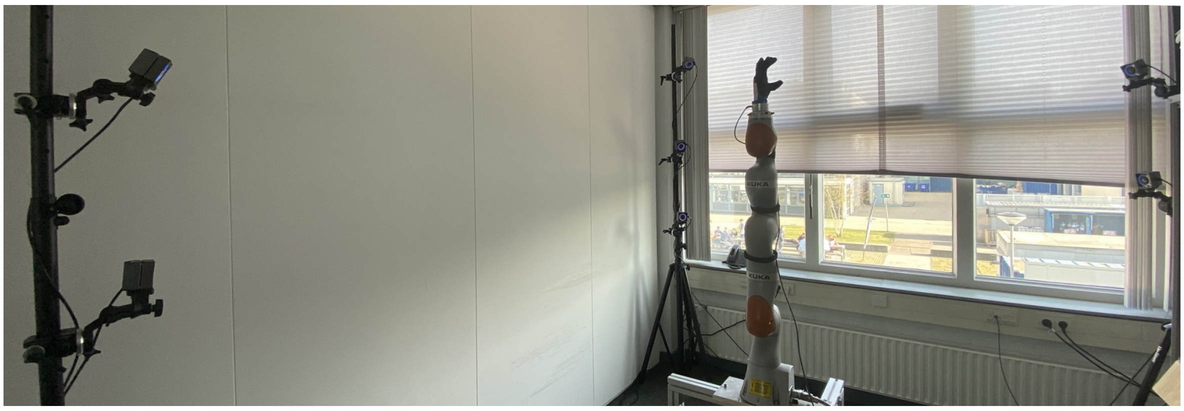


Figure 1-2: A motion capture setup. This photo shows seven of the nine OptiTrack cameras, with a robotic arm in the center of the capture volume.

a set of *generalized forces*, which are linked to the DoFs of the joints as well. By considering the generalized coordinates and forces at the joints, the kinematics and kinetics of motion are coupled.

1-1 Sensing

The input to a motion analysis in OpenSim is typically a collection of sensor data collected while a human subject performed the motion of interest. The dataset must contain measurements that can be used to estimate the kinematics of the kinematic tree. These may then be combined with the dynamic model and data from the aforementioned force sensors, if available, to estimate the kinetics. The most popular method of acquiring this data is by using optical markers. This method involves attaching a number of reflective markers to selected points on the subject's skin. The minimum number of markers depends on the kinematic

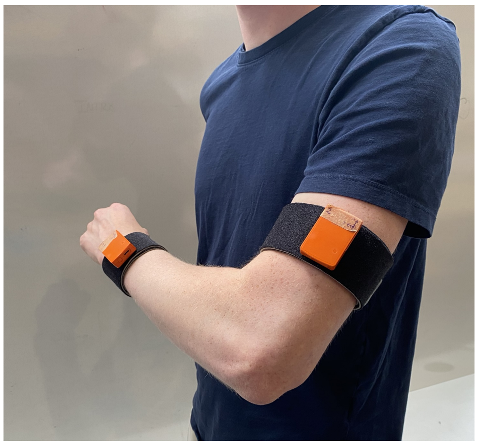


Figure 1-3: Two wearable IMUs from the Xsens wireless system used in this thesis, attached to a human subject's arm.

model and on the method with which the measurements are processed, but typically at least three markers are attached to each body segment. More markers can be used to reduce the effect of STA [9]. Several cameras record and track the positions of these markers. Typically, infrared cameras are used for this purpose. Figure 1-1 shows such a reflective marker and infrared camera. Figure 1-2 shows a typical motion capture setup. Optical motion capture systems are often praised for their accuracy in measuring the marker positions, but they restrict the analysis to motions that can be performed within the capture volume of the system [10]. A recent innovation in human motion analysis is the use of wearable inertial measurement units (IMUs). IMUs are gaining popularity because they make it possible to take measurements outside the capture volume of a motion capture system, for example on a sports field or at home. They are also typically lower in cost than a complete motion capture setup. IMUs are devices that contain a number of different sensors. An accelerometer and a gyroscope are always included. Sometimes, a magnetometer is also included. In a typical commercially available IMU for human motion analysis, three instances of each type of sensor are arranged in a so-called *triaxial* configuration [11]. The triaxial accelerometer measures *specific force* along three perpendicular axes, which is the sum of the linear acceleration of the sensor and the gravitational acceleration. The triaxial gyroscope measures angular velocity along those same axes. When a triaxial magnetometer is present, it also measures the magnetic field along these axes. The magnetic field measurement provides information about the orientation of the sensor relative to the earth's magnetic field. Like in the case of marker measurements, the number of sensors depends on the estimation method, but usually one sensor per body segment is used [10]. An example of IMUs for use in a human motion experiment is shown in Figure 1-3. The quantities measured by IMUs (specific force and angular velocity) are fundamentally different from that measured by a marker-based motion capture system (position). Therefore, the way that the data obtained from these systems is processed to estimate kinematics and kinetics is also different.

1-2 Estimation methods

The way that the collected sensor data is processed depends not only on the quantities measured by the sensors but also on the chosen estimation method. The estimation methods found in literature can be classified on a scale ranging from more *loosely coupled* methods to more *tightly coupled* methods. By loosely coupled, we mean that sensor data is processed in multiple subsequent, independent stages to yield the final estimated kinematics and kinetics. One advantage of a loosely coupled method is that one stage of the processing workflow can be changed, for example to deal with a different type of sensor data, without affecting the remaining stages. On the other end of the scale, the most tightly coupled methods take raw sensor data as input and produce a complete estimate of the kinematics and kinetics as output. The advantage of this approach is a reduced workload for the user and the fact that there is a direct coupling between the sensor data and the estimate. This provides insight into how measurement error affects the estimate and reduces the chances of an error propagating across multiple stages. There is not necessarily a relation between the type of sensor used and the degree to which the estimation is tightly coupled; both loosely and tightly coupled methods exist for marker-based and IMU-based estimation.

1-3 Sensor fusion

Sensor fusion is a key concept in any motion estimation method that uses IMU data. The purpose of sensor fusion is to combine data originating from different types of sensors in a single estimation method in an optimal way. This can have multiple advantages, such as improved accuracy, redundancy or increased robustness. Since IMUs provide measurements from both gyroscopes and accelerometers, some degree of sensor fusion is always needed to combine these different types of data. However, sensor fusion is not limited to IMU data in general. As discussed above, force data is frequently combined with marker data or IMU data to obtain estimates of the kinetics of motion. These can be considered examples of sensor fusion as well. The fusion of marker and IMU data has received limited attention in literature. A loosely coupled approach was proposed for 3D motion of the lower extremities [12]. A tightly coupled approach has also been proposed but was only tested on a two-dimensional model [13]. Neither study validated the results with respect to ground truth. However, precisely the fusion of marker data and IMU data could be a promising innovation in the field of human motion analysis. The idea is that the combination of optical markers, gyroscopes and accelerometers allows a direct measurement on the levels of position, velocity and acceleration. In essence, each type of sensor data contains different information about the motion of the system. Since all three of these quantities are required to describe the system dynamics, the fusion of the different types of sensor data is hypothesized to improve the accuracy of estimated kinematics and kinetics.

1-4 Research goals

The overarching goal of this research project is to improve the accuracy and practical applicability of human motion analysis. More specifically, the aim is to understand the influence of the sensor type and estimation method on the accuracy of estimated kinematics and kinetics for a given dynamic model. The differences between the use of marker data and IMU data and between loosely coupled and tightly coupled estimation methods and the effect of fusing marker data with IMU data are the main points of interest. This understanding can inform the decision making of biomechanicists when they design their measurement setup and data processing workflow. The accuracy of estimated kinetics in particular has received limited attention in literature so far. This can partly be attributed to the difficulty in validating estimated kinetics, since in a typical human subject experiment no ground truth data is available. To circumvent this issue, the analyses in this thesis focus only on joint angles, velocities and torques, for which a robotic arm can be used to obtain ground truth data. This means that the estimation of individual muscle forces will be left out of the scope of this thesis. A robotic arm also removes the errors introduced by STA. The aforementioned kinematic tree formulation will be adopted to estimate the kinematics and kinetics, because it produces these estimates in the form of joint angles, velocities and torques which can be validated. In the analyses in this thesis, every segment of the kinematic tree is equipped with an IMU and markers, in line with a typical human subject experiment. The minimum number of required sensors and markers to estimate kinematics and kinetics is therefore also left out of the scope of this thesis. Given the stated goal of this research project and the properties of the sensor types and estimation methods, three hypotheses are formed which will be evaluated in the remainder this thesis.

1. ***Given a kinematic model, the use of optical marker data will allow the most accurate estimation of its joint angles, compared to IMU-only methods.***
2. ***When IMU data is used in a tightly coupled estimation method, direct angular velocity and specific force measurements will improve joint velocity and torque estimates for a given dynamic model, compared to loosely coupled methods and marker-only methods.***
3. ***Given hypotheses one and two, the fusion of marker and IMU data in a tightly coupled estimation method will yield the most accurate joint kinematics and kinetics.***

1-5 Contributions

To test these hypotheses, four main contributions are presented in this thesis. First, a novel framework that enables the fusion of optical marker data, IMU data and force data for kinematics and kinetics estimation in a tightly coupled fashion is proposed. To enable wider use with a variety of biomechanical models, the OpenSim simulation environment is used to develop this framework. Secondly, an accurate dynamic model of the robotic arm, compatible with OpenSim, is required to compute accurate kinetics. The identification of the parameters of this dynamic model from experimental data is an important part of the validation method. Chapter 4 explains the estimation framework and the system identification method from an algorithmic point of view. Thirdly, in order to test the viability of this framework and to investigate the influence of each sensor type on the estimation, an experimental dataset containing optical marker data, IMU data, force data, and ground truth kinematics and kinetics for various types of motion is created using the robotic arm. The experimental setup used to collect all data, both for the system identification and for the validation dataset is described in Chapter 5. Finally, both the proposed estimation framework and a selection of state-of-the-art estimation methods were assessed on their ability to estimate kinematics and kinetics using the validation dataset and model. The results of this assessment and the system identification are shown in Chapter 6. These results will then be used to test the previously posed hypotheses and draw conclusions about the advantages and limitations of each method in Chapter 7.

Before describing each of these contributions in detail, the related work in kinematics and kinetics estimation of human motion will be discussed in Chapter 2. This is where the current state-of-the-art in estimation methods is described more thoroughly. The concept of loosely and tightly coupled methods is also explored further. Subsequently, Chapter 3 will introduce the required notation, the dynamics equations used in the OpenSim simulation environment and the state-space model adopted for the estimation framework.

Chapter 2

Related work

Before introducing the proposed new estimation framework, it is important to know the current state-of-the-art in the field of human kinematics and kinetics estimation. It is important to note that there are almost as many unique estimation methods as there are biomechanics studies, since there are so many approaches to achieve in essence the same goal. Each approach also features its own set of tuning parameters, making the possibility for customization practically endless. That is why this chapter aims to provide a general overview of the most popular methods. Additionally, some innovative methods will be highlighted that possess interesting properties in the context of this thesis. Special attention will be paid to the concept of loosely coupled and tightly coupled estimation, in an attempt to place the methods discussed here on that spectrum. The possibility of sensor fusion will also be taken into account. The discussion will be roughly ordered from the most loosely coupled to the most tightly coupled methods.

2-1 Marker-based inverse kinematics

The most popular method for estimating kinematics is marker-based inverse kinematics (IK). Many different formulations of the inverse kinematics problem exist, but the goal is always to minimize the distance between measured marker positions and modeled marker positions [14]. Since this thesis focuses on the modeling framework from OpenSim, its kinematic tree formulation will be assumed to explain the concept of IK. As explained in Chapter 1, the kinematic tree formulation relies on a set of generalized coordinates. Typically, there is one generalized coordinate per degree of freedom (DoF) in the system and the joint angles are used as generalized coordinates. This means that the modeled marker positions at a single time instance are fully determined by these generalized coordinates. The IK problem can thus be formulated as a weighted nonlinear least-squares problem [15]. Closed loops in the kinematic tree can be implemented as nonlinear constraints and the problem can be solved with a general purpose nonlinear optimization tool. An important property to note is that IK only yields an estimate of the generalized coordinates at each time instance, so it does not provide any information about the velocity or acceleration of the system. To obtain velocity and acceleration estimates, some form of numerical differentiation is required (e.g. a finite difference method or fitting a spline and taking its analytical derivatives). However, small errors in the estimated generalized coordinates, caused by measurement noise or limited numerical accuracy, will be amplified in this stage. That is why the IK solution is usually low-pass filtered first. It is typically argued that human movement is band-limited, so the true signal can be recovered by selecting an appropriate cut-off frequency [16]. However, the bandwidth of the true signal is not known a priori and may even change over time, so low-pass filtering introduces the risk of losing some high-frequency content of the true signal. Depending on the type of low-pass filter used, the filtering of kinematic data can also make the estimation method non-causal. A non-causal method can distort the order in which events are perceived to occur [17] and cannot be applied in real-time, should that be desired. Since these subsequent processing steps are required to estimate kinetics, IK can be classified as a more loosely coupled approach.

2-2 Orientation-based inverse kinematics

The concept of orientation-based inverse kinematics (OBIK) was designed as an extension to marker-based IK which could be used outside a dedicated motion capture laboratory [18]. It is based on the fact that IMU-measured specific forces, angular velocities and magnetic fields can be pre-processed to obtain an estimate of the 3D orientation of the IMU [19]. This is usually done through some variety of the extended Kalman filter (EKF) and many manufacturers of commercially available IMUs supply this orientation estimation as part of the measurement system [11]. Like modeled marker positions, the kinematic tree formulation can be used to determine modeled IMU orientations uniquely. The difference between measured and predicted orientations can then be represented by an angular magnitude. The estimated orientations can therefore similarly be used in a weighted nonlinear least-squares problem to estimate generalized coordinates. This fact also makes it possible to fuse IMU data with marker data, as was shown by Koning et al. [12]. By including both the terms for marker distance and orientation deviation angle in the cost function, a relative weighting can be assigned to each sensor type. Again, a filtering and differentiation step is applied after the estimation to obtain generalized velocities and accelerations. Because the raw measurements coming from the IMUs are filtered in the orientation estimation stage, this method can be viewed as even more loosely coupled than marker-based IK. OBIK is available in OpenSim under the name OpenSense [20].

2-3 Marker-based kinematic nonlinear Kalman filters and smoothers

Nonlinear Kalman filters have been proposed in various studies to estimate kinematics from marker data, beginning with the extended Kalman filter (EKF). The kinematic EKF was presented as an alternative to the low-pass filtering and numerical differentiation described in Section 2-1 [21]. It also aims to make marker-based estimation more robust against occluded or missing markers. The EKF is an estimation method that found its origins in control systems. It is a two-stage state space algorithm, where a prediction step based on a dynamic model is followed by a correction step, in which sensor data is used to modify the predicted state. It relies on linearizing the dynamic and measurement model at every time step to enable state estimation for nonlinear systems. The EKF is a statistical method, where the relative weightings of the prediction and the measurement are based on the relative level of uncertainty specified by the user. The advantage of this is that large deviations from the predicted state are discouraged by the algorithm, so it has an inherent smoothing effect on the estimates. Furthermore, it can be made robust against missing data by relying purely on the prediction at time instances where no measurements are available. The state space model used in kinematic EKFs in literature usually defines the state as a combination of the generalized coordinates in the form of joint angles and a number of their time derivatives, most commonly up to acceleration. In this case, the acceleration would be described by a random walk model. The dynamic model is then linear and the nonlinearity is only introduced by the measurement function. The unscented Kalman filter (UKF), a different nonlinear Kalman filter, has been used to estimate kinematics from marker data as well [22]. The difference between the nonlinear Kalman filters lies not in the model, but in the way they approximate the uncertainty of the prediction and measurement. As was mentioned above, the EKF linearizes the system at every time step to propagate the uncertainty of the estimates. The UKF propagates the uncertainty by sampling the predicted state and measurement. The state space models used for kinematics estimation can be applied in either method. Since the nonlinear Kalman filters eliminate the need for low-pass filtering and numerical differentiation, these can be considered more tightly coupled when compared to IK.

An extension of nonlinear Kalman filtering is the so-called Kalman smoother. Where Kalman filtering uses the data up to and including the current time instance to estimate the state of the system, in Kalman smoothing the entire measurement history is used for each time instance. Compared to a nonlinear Kalman filter, a nonlinear Kalman smoother with the same state space model can estimate generalized coordinates without introducing a delay [23]. However, Kalman smoothing eliminates the causal nature of the Kalman filtering methods. Nonlinear Kalman smoothers can be used with the same state space models as nonlinear Kalman filters and both extended and unscented variants exist.

2-4 IMU-based kinematic nonlinear Kalman filters and smoothers

A variation on the marker-based kinematic nonlinear Kalman filter method described above was proposed for IMU data [24]. The dynamic model is the same as in the case of marker data, however the measurement model needs to be adapted to be able to predict specific forces and angular velocities. This is possible as long as the generalized accelerations are included in the state. Because this method uses a similar state space model as the nonlinear Kalman filters for marker data, it is also possible to combine the two estimation methods. This was tested for a 2D model by Matthew et al. [13]. In this study, it was suggested that this form of sensor fusion could improve estimation for high acceleration motions.

Different state-space approaches to estimating kinematics from IMU data, which do not rely on the generalized coordinate formulation of the system, have also been proposed. The definition of the state in these methods is more varied. What they have in common is that they estimate individual IMU orientations in a nonlinear Kalman filter or smoother, similar to the approach mentioned in Section 2-2. However, in the estimation algorithm, the orientation estimates of multiple IMUs are constrained such that they comply with the definition of the kinematic tree model. The joint angles, velocities and accelerations are not estimated directly, but can be derived from the estimated orientations. Since these estimates are consistent with the kinematic model to begin with, the OBIK step is eliminated. Methods were proposed that used an EKF or a Kalman smoother [25] or a UKF [26]. Like before, the state space model and the type of nonlinear filter or smoother can be viewed separately.

The main difference of all these methods, compared to OBIK, is that they use the IMU data directly without estimating sensor orientations first. The estimated velocities and accelerations are therefore coupled directly to the measurements, making it more tightly coupled. Furthermore, it provides the same statistical weighting of data and robustness against missing measurements as the marker-based EKF described previously.

2-5 Inverse Dynamics and muscle force estimation

When an estimate of the complete kinematics, including velocities and accelerations, is available, this can be applied to a dynamic model of the human body to obtain an estimate of the kinetics. It is at this stage that any measured interaction forces and torques are applied to the system. The method of inverse dynamics (ID) in essence consists of applying estimated kinematics and measured forces to the nonlinear equations of motion (EoM) of the dynamic model [27]. In the context of a kinematic tree model, this yields the generalized force associated with each generalized coordinate.

Since most joints in the human body are actuated by multiple muscles and muscles often actuate multiple joints, a set of generalized forces does not generally correspond with a unique set of muscle forces. When individual muscle forces are of interest, this problem can be solved by constrained optimization [28]. The assumption is that the human body works to minimize some quantity, like energy expenditure. The constraints make sure the estimated generalized forces are balanced. They also enforce physically consistent behavior, such as the fact that muscles can only pull and not push. It is possible to estimate not only the forces but also the neuromuscular activation signals in this way [29]. Since the estimation of kinematics, generalized forces and muscle forces all happen in independent stages, this method of kinetics estimation can be considered the most loosely coupled approach. An example of a complete motion analysis performed this way is given by Karatsidis et al. [30]. In this study, muscle forces were estimated from IMU data. From start to finish, five subsequent data processing stages can be identified in their method when including the IMU orientation estimation.

2-6 Optimal control methods for kinetics estimation

As an alternative to ID and muscle force optimization, optimal control methods have been proposed [31]. The main difference is that these methods do not attempt to match the estimated kinematics exactly, as is the case with ID. Instead, they attempt to find a trajectory of forces, torques or activation signals that can be applied to the dynamic model to track the estimated generalized coordinates ‘as close as possible’. In practice, that means it minimizes a weighted sum of the ‘effort’, measured by e.g. total energy expenditure, and the distance to the estimated coordinates. A nice property of this type of method is that it always yields an estimate of the kinetics that can be applied in a forward simulation of the motion. Because the trajectory is usually estimated from all kinematic data at once, optimal control methods as described here are non-causal. Compared to ID and muscle force optimization, optimal control methods can be considered more tightly coupled. They eliminate the need for ID, as well as the pre-processing of estimated kinematics (filtering, differentiating). It should be noted that this comes at a greater computational cost. In OpenSim, optimal control methods are available under the name MocoTrack [32].

Instead of tracking estimated generalized coordinates, Dorschky et al. [33] used an optimal control method to directly track the specific force and angular velocity of a set of IMUs. The novelty of this approach is that the modeled muscle forces are directly coupled to the measured specific force and angular velocity. The intermediate orientation estimation and OBIK steps required for other optimal control methods introduce some assumptions about the nature of the motion, such as smoothness and limited bandwidth, which can be avoided by tracking sensor data directly. In other words, the proposed method can be considered very tightly coupled. In the referenced study, the method was only applied to a 2D model.

2-7 Nonlinear Kalman filtering for kinetics estimation

Inspired by the work of Dorschky et al., De Kanter [34] also attempted to directly estimate kinetics from IMU data. However, this was not done through the optimal control framework but rather as an extension of the Kinematic EKF for IMU data. In this work, the iterated extended Kalman filter (IEKF) was used to estimate joint angles and joint torques of a robotic arm moving in three dimensions directly from IMU data. The use of a robot enabled validation with respect to ground truth. The validation focused mostly on the kinematics, which showed excellent agreement. The IEKF relies on linearization of the state space model like the EKF, but the linearization point is iterated multiple times for each time instance. The optimal control method by Dorschky et al. and the IEKF for joint torque estimation by De Kanter are quite closely related, but there are some notable differences. The IEKF only estimates the kinematics and kinetics of one time instance at a time, whereas optimal control methods can optimize over the whole time history. Because of this property, the IEKF is a causal method. The relative weighting of different sensors and the predicted state in the IEKF is based on statistical properties of the signals, whereas the weights in an optimal control problem are set by the user in a more heuristic sense. Of course, the IEKF also requires some tuning, but it can be insightful if the parameters have a physical interpretation. Inherent to the functioning of the IEKF is that it estimates the uncertainty of the state along with the state itself [35]. This can help gain insight into the statistical significance of the results.

In an effort to show the relations and interdependence between the estimation methods discussed above, the diagram in Figure 2-1 was constructed. This figure is not intended to be a complete description of existing work, but rather an overview that highlights the similarities and differences between estimation methods. Combinations of methods that are theoretically possible, but not discussed above are not shown in this figure for the sake of compactness. This diagram also shows where the framework proposed in this thesis should fit in. The work by De Kanter was used as a starting point. Its IEKF algorithm is the basis for this new framework. The goal was to generalize it to a variety of OpenSim models, to include more sensor modalities and to validate both the estimated kinematics and kinetics. In the following chapter, the mathematical foundation of the kinematics and kinetics as they are modeled in

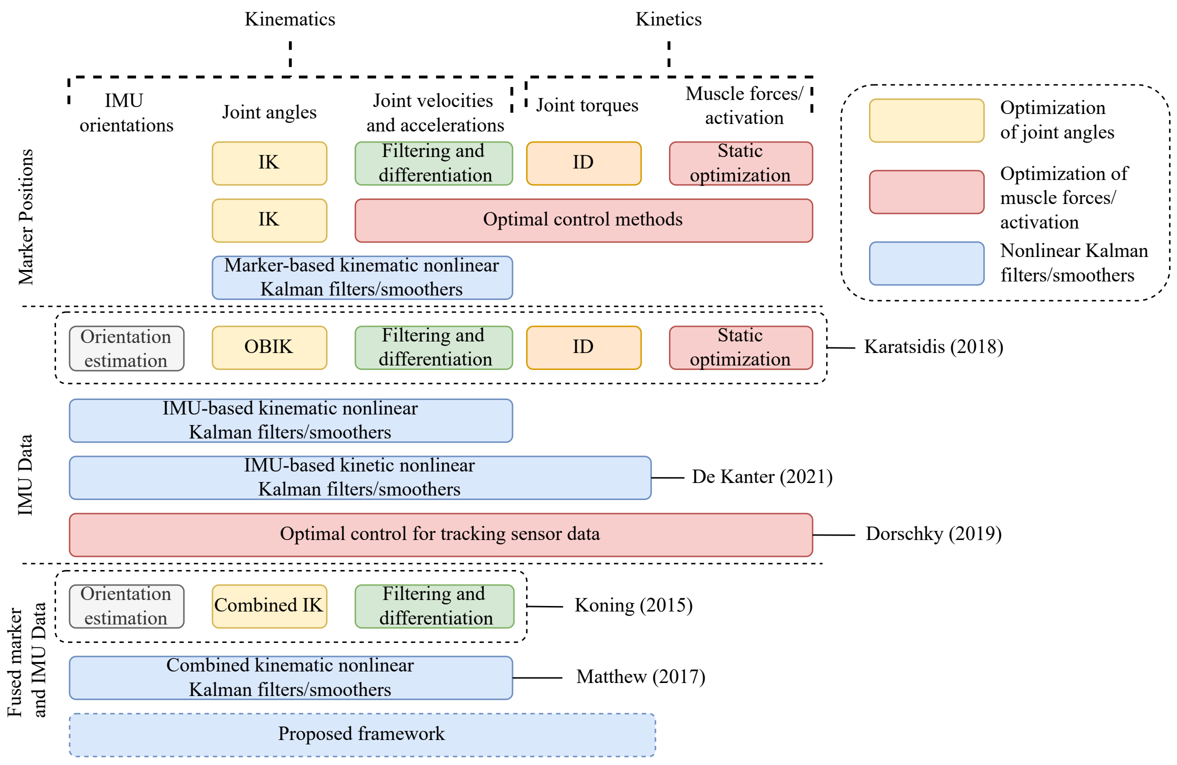

Kinematics
Kinetics
IMU orientations
Joint angles
Joint velocities and accelerations
Joint torques
Muscle forces/ activation
Marker Positions
IK
Filtering and differentiation
ID
Static optimization
IK
Optimal control methods
Marker-based kinematic nonlinear Kalman filters/smoothers
Optimization of joint angles
Optimization of muscle forces/ activation
Nonlinear Kalman filters/smoothers
Orientation estimation
OBIK
Filtering and differentiation
ID
Static optimization
Karatsidis (2018)
IMU Data
IMU-based kinematic nonlinear Kalman filters/smoothers
IMU-based kinetic nonlinear Kalman filters/smoothers
De Kanter (2021)
Optimal control for tracking sensor data
Dorschky (2019)
Fused marker and IMU Data
Orientation estimation
Combined IK
Filtering and differentiation
Koning (2015)
Combined kinematic nonlinear Kalman filters/smoothers
Matthew (2017)
Proposed framework

Figure 2-1: A map of kinematics and kinetics estimation methods from literature. On the horizontal axis, the variables estimated by the methods are listed. On the vertical axis, the sensor data used by the methods are listed. The names listed refer to the publications discussed explicitly in this chapter. Not shown in this figure is the inclusion of force measurements. The blue block with a dashed line shows the space that the framework proposed in this thesis aims to fill.

OpenSim will be detailed. This will form the basis for the development of the algorithm and the validation model, which will be described in Chapter 4.

Chapter 3

Modeling

At the heart of the estimation framework proposed in this thesis is the dynamic model of the system for which the kinematics and kinetics are estimated. The way in which this system is modeled is essential, since it determines to an extent the results of the estimation itself. When describing a physical system mathematically, it is always necessary to make some assumptions and simplifications. These assumptions and simplification not only make it possible to describe the system, they also determine the set of parameters that can be estimated given the simplified representation of reality. For example, when a model is constructed under the assumption that a body is completely rigid, it will be impossible to use that model to infer anything about the elastic deformation of that body. The goal of this chapter is to describe the modeling method used in OpenSim and the proposed estimation framework, to introduce the necessary notation and to clarify the assumptions and simplifications introduced by this modeling method. As mentioned before, OpenSim uses a generalized coordinate formulation of a kinematic tree structure to model the neuromusculoskeletal system [7, 8]. The generalized coordinate formulation of multibody dynamics will be detailed in the Section 3-2. The derivations, in so far as they are given, are based on the work by Vallery and Schwab [36].

3-1 Notation

Before the multibody dynamics equations are described in detail, it is useful to introduce the notation of variables that will be used in this section and throughout the thesis. In general, lowercase letters (m) will be used to refer to scalar quantities, lowercase bold letters for vectors ($\boldsymbol{q}$) and uppercase letters for matrices (A). Left superscripts are used to indicate that a vector is expressed in a certain reference frame, whereas right subscripts indicate a description of a variable, an index or a combination of the two. For example, ${}^{\mathcal{G}}\boldsymbol{p}_{CoM,i}$ would refer to the position $\boldsymbol{p}$ of the center of mass (CoM) of body i, expressed in the ground reference frame $\mathcal{G}$. Where $\boldsymbol{p}$ is used to denote position, $\dot{\boldsymbol{p}}$ and $\ddot{\boldsymbol{p}}$ denote velocity and acceleration respectively. For angular velocity and angular acceleration, $\boldsymbol{\omega}$ and $\dot{\boldsymbol{\omega}}$ are used respectively.

Throughout this thesis, different reference frames will be used when describing quantities as vectors. The ground reference frame $\mathcal{G}$ was already introduced above. The ground reference frame is an *inertial frame*. In OpenSim, each rigid body also has its own reference frame, which translates and rotates with the body and is therefore non-inertial. Additionally, sensor-fixed reference frames are often used when discussing measurement models, denoted by $\mathcal{S}_i$. Since sensors are typically modeled as being rigidly attached to body segments, the sensor-fixed reference frames are connected to the corresponding body frames by a constant transformation and are also non-inertial.

To convert a vector expressed in one reference frame to another reference frame, transformations are used. For rotations, these are expressed as rotation matrices, where ${}^{\mathcal{B}}R_{\mathcal{A}}$ describes a rotation from frame $\mathcal{A}$ to frame $\mathcal{B}$. For example,

$$ {}^{\mathcal{B}}\boldsymbol{\omega} = {}^{\mathcal{B}}R_{\mathcal{A}}\,{}^{\mathcal{A}}\boldsymbol{\omega}. $$

The rotation from frame $\mathcal{B}$ to frame $\mathcal{A}$ is given by the inverse of this rotation matrix. Conveniently, the inverse is equal to the transpose of the matrix.

$$ {}^{\mathcal{A}}R_{\mathcal{B}} = ({}^{\mathcal{B}}R_{\mathcal{A}})^{-1} = ({}^{\mathcal{B}}R_{\mathcal{A}})^{T} \tag{3-1} $$

When positions are considered, the location of the origin of reference frame $\mathcal{A}$, $\mathcal{O}_{\mathcal{A}}$, has to be taken into account.

$$ {}^{\mathcal{B}}\boldsymbol{p} = {}^{\mathcal{B}}R_{\mathcal{A}}\,{}^{\mathcal{A}}\boldsymbol{p} + {}^{\mathcal{B}}\mathcal{O}_{\mathcal{A}} $$

The homogeneous transformation ${}^{\mathcal{B}}H_{\mathcal{A}}$ is used to achieve this. It is defined such that

$$ \begin{bmatrix} {}^{\mathcal{B}}\boldsymbol{p} \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{\mathcal{B}}R_{\mathcal{A}} & {}^{\mathcal{B}}\mathcal{O}_{\mathcal{A}} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} {}^{\mathcal{A}}\boldsymbol{p} \\ 1 \end{bmatrix} = {}^{\mathcal{B}}H_{\mathcal{A}} \begin{bmatrix} {}^{\mathcal{A}}\boldsymbol{p} \\ 1 \end{bmatrix}. \tag{3-2} $$

Combining this definition with (3-1), the inverse is also easily computed.

$$ {}^{\mathcal{A}}H_{\mathcal{B}} = {}^{\mathcal{B}}H_{\mathcal{A}}{}^{-1} = \begin{bmatrix} ({}^{\mathcal{B}}R_{\mathcal{A}})^{T} & -({}^{\mathcal{B}}R_{\mathcal{A}})^{T}\,{}^{\mathcal{B}}\mathcal{O}_{\mathcal{A}} \\ \mathbf{0} & 1 \end{bmatrix} \tag{3-3} $$

3-2 Multibody Dynamics

Given the notation from Section 3-1, the Newton-Euler equations for a system of rigid bodies may be written as

$$ \boldsymbol{f}_i = m_i\,\ddot{\boldsymbol{p}}_{CoM,i} - m_i\,\boldsymbol{g}, \tag{3-4a} $$
$$ \boldsymbol{\tau}_i = I_i\,\dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times (I_i\boldsymbol{\omega}_i), \tag{3-4b} $$

where m_i and I_i denote the mass and the moment of inertia tensor about the CoM, of the rigid body with index i. The force $\boldsymbol{f}_i$ and moment $\boldsymbol{\tau}_i$ are the sums of all contact forces and moments acting on body i. The gravitational force, $m_i\,\boldsymbol{g}$, is considered separately. The contact forces and moments include the externally applied forces as well as the interaction forces of the bodies in the system.

$$\boldsymbol{f}_i = \boldsymbol{f}_{ext,i} + \boldsymbol{f}_{int,i}, \tag{3-5a}$$

$$\boldsymbol{\tau}_i = \boldsymbol{\tau}_{ext,i} + \boldsymbol{\tau}_{int,i}. \tag{3-5b}$$

Note that no reference frame is given in these equations since they apply generally. When applying the equations in practice, the moment of inertia tensor I_i should be expressed in the same reference frame as the angular acceleration $\dot{\boldsymbol{\omega}}_i$ and the moment $\boldsymbol{\tau}_i$. This is usually the body frame $\mathcal{B}_i$, since the moment of inertia tensor can then be stored as a constant matrix. Additionally, when the force $\boldsymbol{f}_i$ is expressed in the ground frame $\mathcal{G}$, the gravitational acceleration $\boldsymbol{g}$ can be stored as a constant vector. Combining the Newton-Euler equations for a system of N rigid bodies yields the matrix-vector equation

$$\underbrace{\begin{bmatrix}\boldsymbol{f}_1\\ \boldsymbol{\tau}_1\\ \vdots\\ \boldsymbol{f}_N\\ \boldsymbol{\tau}_N\end{bmatrix}}_{\boldsymbol{f}} = \underbrace{\begin{bmatrix}m_1\mathcal{I}_{3\times3} & & & & \\ & I_1 & & & \\ & & \ddots & & \\ & & & m_N\mathcal{I}_{3\times3} & \\ & & & & I_N\end{bmatrix}}_{M} \underbrace{\begin{bmatrix}\ddot{\boldsymbol{p}}_{CoM,1}\\ \dot{\boldsymbol{\omega}}_1\\ \vdots\\ \ddot{\boldsymbol{p}}_{CoM,N}\\ \dot{\boldsymbol{\omega}}_N\end{bmatrix}}_{\ddot{\boldsymbol{p}}} + \underbrace{\begin{bmatrix}-m_1\,\boldsymbol{g}\\ \boldsymbol{\omega}_1\times(I_1\boldsymbol{\omega}_1)\\ \vdots\\ -m_1\,\boldsymbol{g}\\ \boldsymbol{\omega}_N\times(I_N\boldsymbol{\omega}_N)\end{bmatrix}}_{\boldsymbol{f}_{inertia}}, \tag{3-6}$$

where $\mathcal{I}_{3\times3}$ denotes the 3×3 identity matrix. The mass matrix for all bodies combined is denoted by the symbol M. The vector $\ddot{\boldsymbol{p}}$ is used to describe all body accelerations and the vector $\boldsymbol{f}_{inertia}$ contains both the gravitational and gyroscopic forces for all bodies. The concept of generalized coordinates is introduced at this stage to reduce the dimension of this system of equations and to describe how the motion of the individual bodies is constrained. A set of N_q generalized coordinates $\boldsymbol{q}$ must be chosen such that it uniquely determines the position and orientation of each body in the system. When the number of generalized coordinates is equal to the number of degrees of freedom (DoFs) of the system, this description is a *minimal* description of the configuration of the system and therefore eliminates the need for any constraints. Often, this is the case when the angles of rotation or displacements at the joints connecting the bodies are used as generalized coordinates. In any case, the position of the center of mass for any body can be written as a function of the generalized coordinates $\boldsymbol{p}_{CoM,i}(\boldsymbol{q})$. By the chain rule, the velocity $\dot{\boldsymbol{p}}_{CoM,i}$ is then a linear function of the *generalized velocities* $\dot{\boldsymbol{q}}$. A similar argument can be made for the angular velocity $\boldsymbol{\omega}_i$. Combining these functions gives (3-7), where $J_{CoM,i}$ is the *kinematic Jacobian* of body i at the CoM.

$$\begin{bmatrix}\dot{\boldsymbol{p}}_{CoM,i}\\ \boldsymbol{\omega}_i\end{bmatrix} = J_{CoM,i}(\boldsymbol{q})\,\dot{\boldsymbol{q}} \tag{3-7}$$

By again applying the chain rule and the product rule, the accelerations can also be determined:

$$\begin{bmatrix} \ddot{\boldsymbol{p}}_i \\ \dot{\boldsymbol{\omega}}_i \end{bmatrix} = J_{CoM,i}(\boldsymbol{q})\, \ddot{\boldsymbol{q}} + \dot{J}_{CoM,i}(\boldsymbol{q}, \dot{\boldsymbol{q}})\, \dot{\boldsymbol{q}} \tag{3-8}$$

Combining the kinematic Jacobians for all bodies results in the *system Jacobian* $J(\boldsymbol{q})$.

$$\ddot{\boldsymbol{p}} = J(\boldsymbol{q})\, \ddot{\boldsymbol{q}} + \dot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}})\, \dot{\boldsymbol{q}} \tag{3-9}$$

The motion of the system can now be completely described using the generalized coordinates, velocities and accelerations. This description is referred to as the *kinematics* of the model and the variables $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ are referred to as *kinematic variables.* The next step is to link the kinematic variables to the *kinetic* variables, describing the forces and moments in the system. Substituting (3-9) into (3-6) gives

$$\boldsymbol{f}_{int} + \boldsymbol{f}_{ext} = M\, J(\boldsymbol{q})\, \ddot{\boldsymbol{q}} + M\, \dot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}})\, \dot{\boldsymbol{q}} + \boldsymbol{f}_{inertia}. \tag{3-10}$$

The system Jacobian can be interpreted as spanning the space of velocities that are permitted by the defined generalized coordinates in a given configuration. Conveniently, the interaction forces in non-permissible directions, that is, the forces that constrain the system, are in the nullspace of the transpose of the system Jacobian. These forces do not produce any work, since no motion is associated with them. When the interaction forces $\boldsymbol{f}_{int}$ are projected onto the transpose of the system Jacobian, the remainder are the *generalized forces* $\boldsymbol{\tau}$ that do produce work. The generalized forces are related to the DoFs defined by the generalized coordinates. For example, when a generalized coordinate defines the angle of rotation about a hinge joint, the corresponding generalized force describes the torque about that joint.

$$\boldsymbol{\tau} = J^T(\boldsymbol{q})\, \boldsymbol{f}_{int} \tag{3-11}$$

The same projection can be applied to the externally applied forces $\boldsymbol{f}_{ext}$ using the kinematic Jacobian at their application points, $J_f(\boldsymbol{q})$. Combining (3-10) and (3-11) finally gives a compact description of the complete system dynamics.

$$\boldsymbol{\tau} + J_f^T(\boldsymbol{q})\boldsymbol{f}_{ext} = J^T(\boldsymbol{q}) M J(\boldsymbol{q})\, \ddot{\boldsymbol{q}} + J^T(\boldsymbol{q})\, M\, \dot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}})\, \dot{\boldsymbol{q}} + J^T(\boldsymbol{q})\boldsymbol{f}_{inertia} \tag{3-12}$$

By introducing the *generalized mass matrix* $\bar{M}(\boldsymbol{q}) = J^T(\boldsymbol{q}) M J(\boldsymbol{q})$, grouping all velocity-dependent terms into the function $\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and grouping all gravity terms into the function $\boldsymbol{g}(\boldsymbol{q})$, the familiar form of the equations of motion (EoM) is obtained. The function $\boldsymbol{g}(\boldsymbol{q})$ can also be used to absorb any other position-dependent conservative forces, such as spring forces.

$$\boldsymbol{\tau} + J_f^T(\boldsymbol{q})\boldsymbol{f}_{ext} = \bar{M}(\boldsymbol{q})\, \ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) \tag{3-13}$$

In order to clarify the concepts discussed above, Figure 3-1 shows a simple multibody system with two rigid bodies, two generalized coordinates ($\boldsymbol{q}_1$ and $\boldsymbol{q}_2$), an external force applied to body 1 and a sensor attached to body 2. As this figure shows, the orientations and positions of the body frames with respect to the ground frame and each other can vary. These relative orientations and positions are completely determined by the generalized coordinates, because the chosen generalized coordinates are a minimal representation of the system.

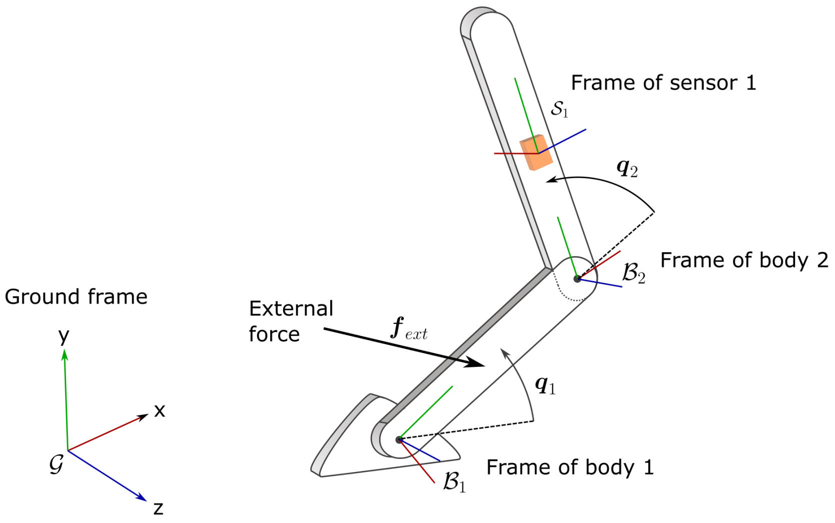


Figure 3-1: Two-DoF multibody system with two rigid bodies, one external force and one sensor.

3-3 Sensor models

The EoM can be used to describe the system of rigid bodies in terms of generalized coordinates, velocities, accelerations and forces. The goal of this thesis however is to infer these quantities from measurement data. For this, the model should describe how the measurements coming from sensors attached to the body segments, are related to the kinematic and kinetic quantities in the multibody model. Three types of sensors, measuring different quantities, will be considered.

3-3-1 Force/torque sensor

The most straightforward sensor model is that of the force/torque sensor. In the model, a 6 degree of freedom (DoF) force/torque sensor is assumed that is connected rigidly to a known point on one of the body segments. It directly measures any force that is applied to the body segment through the sensor. Any force applied to the system that is not transferred through the sensor cannot be measured. The measured force is expressed in a reference frame that is fixed to the sensor, denoted by $\mathcal{S}_f$. The sensor-fixed reference frame is attached to one of the body segments with a fixed and known transformation. It is assumed that the measurement is affected by Gaussian white noise with covariance Q_f. Since the measured external force is modeled as an input to the system, it is denoted by the symbol $\boldsymbol{u}$, as is conventional in control systems literature.

$$\boldsymbol{u} = {}^{\mathcal{S}_f}\boldsymbol{f}_{ext} + \boldsymbol{v}_f, \quad \boldsymbol{v}_f \sim \mathcal{N}(0, Q_f) \tag{3-14}$$

3-3-2 Marker positions

In addition to the force/torque sensor, the use of a 3-dimensional marker-based optical motion capture system will be modeled. It is assumed that this system outputs an estimation of the

3D position of each marker, expressed in the motion capture reference frame $\mathcal{M}$. Since the cameras of the motion capture system do not move with respect to the ground frame, the motion capture reference frame is connected to the ground reference frame by a fixed transformation ${}^{\mathcal{G}}H_{\mathcal{M}}$. The output of the motion capture system is a vector containing the positions of all markers, numbered 1 through N_m. Since the markers are assumed to be attached rigidly to the body segments, their position is uniquely determined by the generalized coordinates as well. It is assumed that this measurement is also corrupted by Gaussian white noise, with covariance R_m.

$$\boldsymbol{y}_m = \begin{bmatrix} {}^{\mathcal{M}}\boldsymbol{p}_{m,1}(\boldsymbol{q}) \\ \vdots \\ {}^{\mathcal{M}}\boldsymbol{p}_{m,N_m}(\boldsymbol{q}) \end{bmatrix} + \boldsymbol{w}_m = \boldsymbol{p}_m(\boldsymbol{q}) + \boldsymbol{w}_m, \quad \boldsymbol{w}_m \sim \mathcal{N}(0, R_m) \tag{3-15}$$

3-3-3 IMU measurements

Finally, the use of inertial measurement units (IMUs) is also modeled. For the algorithms developed in this thesis, only the accelerometer and gyroscope of the IMU are modeled. The accelerometer measures *specific force* along three perpendicular axes. The gyroscope measures angular velocity along those same axes. The three sensor axes of IMU i together span the sensor frame $\mathcal{S}_i$.

Specific force

The specific force measured by an accelerometer consists of the difference between the acceleration of the sensor with respect to the ground frame and the gravitational acceleration. The measurement is expressed in the sensor frame. It is assumed that the position and orientation of the sensor relative to the body segment to which it is attached are known. The acceleration can then be modeled using the kinematic Jacobian, similar to (3-8). In this case, only the linear acceleration of the sensor $\ddot{\boldsymbol{p}}_{s,i}$ is considered and the kinematic Jacobian is computed at the origin of the sensor frame. The part of the kinematic Jacobian relating to linear acceleration, computed at the origin of the frame of sensor i, is denoted by $J_{p,s,i}$.

$${}^{\mathcal{S}_i}\ddot{\boldsymbol{p}}_{s,i}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = {}^{\mathcal{S}_i}J_{p,s,i}(\boldsymbol{q})\, \ddot{\boldsymbol{q}} + {}^{\mathcal{S}_i}\dot{J}_{p,s,i}(\boldsymbol{q}, \dot{\boldsymbol{q}})\, \dot{\boldsymbol{q}} \tag{3-16}$$

The gravitational acceleration is constant in the ground reference frame. However, the orientation of the sensor frame with respect to the ground frame is not. The gravitational component of the specific force measurement therefore depends on the generalized coordinates as

$${}^{\mathcal{S}_i}\boldsymbol{g}(\boldsymbol{q}) = {}^{\mathcal{S}_i}R_{\mathcal{G}}(\boldsymbol{q})\, {}^{\mathcal{G}}\boldsymbol{g} \tag{3-17}$$

Combining (3-16) and (3-17) gives the overall specific force measured by the IMU, denoted by $\boldsymbol{a}_i$:

$${}^{\mathcal{S}_i}\boldsymbol{a}_{s,i}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = {}^{\mathcal{S}_i}\ddot{\boldsymbol{p}}_{s,i}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) - {}^{\mathcal{S}_i}\boldsymbol{g}(\boldsymbol{q}) \tag{3-18}$$

Angular velocity

When computing the angular velocity, the kinematic Jacobian is used again. This time, the part of the kinematic Jacobian relating to angular velocity is used, denoted by $J_{\omega,s,i}$.

$$^{\mathcal{S}_i}\boldsymbol{\omega}_{s,i}(\boldsymbol{q},\dot{\boldsymbol{q}}) = {}^{\mathcal{S}_i}J_{\omega,s,i}(\boldsymbol{q})\,\dot{\boldsymbol{q}} \tag{3-19}$$

The overall IMU measurement for sensor i consists of the specific force, the angular velocity and Gaussian white noise with covariance $R_{s,i}$.

$$\boldsymbol{y}_{s,i}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}) = \begin{bmatrix} ^{\mathcal{S}_i}\boldsymbol{a}_{s,i}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}) \\ ^{\mathcal{S}_i}\boldsymbol{\omega}_{s,i}(\boldsymbol{q},\dot{\boldsymbol{q}}) \end{bmatrix} + \begin{bmatrix} \boldsymbol{w}_{a,s,i} \\ \boldsymbol{w}_{\omega,s,i} \end{bmatrix} = \begin{bmatrix} ^{\mathcal{S}_i}\boldsymbol{a}_{s,i} \\ ^{\mathcal{S}_i}\boldsymbol{\omega}_{s,i} \end{bmatrix} + \boldsymbol{w}_{s,i}, \quad \boldsymbol{w}_{s,i} \sim \mathcal{N}(0, R_{s,i}) \tag{3-20}$$

The measurements from N_{IMU} different IMUs can be combined to yield the overall IMU measurement vector $\boldsymbol{y}_{IMU}$

$$\boldsymbol{y}_{IMU} = \begin{bmatrix} \boldsymbol{y}_{s,1} \\ \vdots \\ \boldsymbol{y}_{s,N_{IMU}} \end{bmatrix} \tag{3-21}$$

3-4 State space model

The dynamic model from Section 3-2 and the measurement models from Section 3-3 can be used to construct a model in the *state space* formulation. The state space formulation enables the use of general purpose state estimation techniques from literature, like the iterated extended Kalman filter (IEKF). To obtain a state space model, some choices ought to be made about the definition of input, output and state variables. In general, the unknown variables that are to be estimated should be part of the state vector. The input and output vectors are reserved for the known variables, i.e. the measured sensor output. Since the goal is to estimate the kinematic and kinetic variables of the system, there are four unknown variables in the system, namely the generalized coordinates, velocities, accelerations and forces ($\boldsymbol{q}$,$\dot{\boldsymbol{q}}$,$\ddot{\boldsymbol{q}}$ and $\boldsymbol{\tau}$). However, because the system always obeys (3-13), these variables are not independent. To obtain a smaller description of the system, a subset of variables can be included in the state vector. Similarly to earlier work [34], it was chosen to directly include the generalized forces in the state vector. This makes it possible to include the system dynamics in the state space formulation, which could benefit the accuracy of the estimation. Additionally, this eliminates the use of inverse dynamics (ID) as a post-processing step. The state vector $\boldsymbol{x}$ is therefore defined as

$$\boldsymbol{x}(t) = \begin{bmatrix} \boldsymbol{q}(t) \\ \dot{\boldsymbol{q}}(t) \\ \boldsymbol{\tau}(t) \end{bmatrix} \tag{3-22}$$

The dependence on time of the state variables is made explicit here to highlight the fact that the system is now considered in continuous time. Any state space model is made up of a dynamic model and a measurement model. This means that for each state variable, its time derivative must be computed. For the generalized coordinates and velocities, these are simply given by the generalized velocities and accelerations respectively. Combining (3-13) and (3-14) and solving for $\ddot{\boldsymbol{q}}$ results in a description of the generalized accelerations in terms of the state variables, the measured external forces $\boldsymbol{u}$ and the force sensor noise $\boldsymbol{v}_f$.

$$\begin{aligned}\ddot{\boldsymbol{q}} &= \bar{M}^{-1}(\boldsymbol{q})(\boldsymbol{\tau} + {}^{\mathcal{S}_f}J_f^T(\boldsymbol{q})\boldsymbol{f}_{ext} - \boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}}) - \boldsymbol{g}(\boldsymbol{q}))\\ &= \bar{M}^{-1}(\boldsymbol{q})(\boldsymbol{\tau} + {}^{\mathcal{S}_f}J_f^T(\boldsymbol{q})\boldsymbol{u} - \boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}}) - \boldsymbol{g}(\boldsymbol{q}) - {}^{\mathcal{S}_f}J_f^T(\boldsymbol{q})\boldsymbol{v}_f)\end{aligned}$$

Since the noise term $\boldsymbol{v}_f$ is zero-mean, its sign may be flipped without affecting its properties:

$$\ddot{\boldsymbol{q}} = \bar{M}^{-1}(\boldsymbol{q})(\boldsymbol{\tau} + {}^{\mathcal{S}_f}J_f^T(\boldsymbol{q})\boldsymbol{u} - \boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}}) - \boldsymbol{g}(\boldsymbol{q}) + {}^{\mathcal{S}_f}J_f^T(\boldsymbol{q})\bar{\boldsymbol{v}}_f, \quad \bar{\boldsymbol{v}}_f = -\boldsymbol{v}_f \sim \mathcal{N}(0, Q_f). \tag{3-23}$$

For the time derivative of the generalized forces, an assumption is introduced at this stage. It is assumed that the generalized forces evolve as a random walk process. This means that the time derivative of the generalized forces is given by a white noise signal, $\boldsymbol{v}_\tau$, with covariance Q_τ. This random walk assumption is common in the case where there is little knowledge about the true signal [23, 37, 38]. Combining this assumption with the definition of the state and (3-23) gives the continuous time process model:

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix}\dot{\boldsymbol{q}}\\ \ddot{\boldsymbol{q}}\\ \dot{\boldsymbol{\tau}}\end{bmatrix} = \begin{bmatrix}\dot{\boldsymbol{q}}\\ \bar{M}^{-1}(\boldsymbol{q})(\boldsymbol{\tau} + {}^{\mathcal{S}_f}J_f^T(\boldsymbol{q})\boldsymbol{u} - \boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}}) - \boldsymbol{g}(\boldsymbol{q}) + {}^{\mathcal{S}_f}J_f^T(\boldsymbol{q})\bar{\boldsymbol{v}}_f\\ \boldsymbol{v}_\tau\end{bmatrix} = \bar{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}(t)), \tag{3-24}$$

where

$$\boldsymbol{v} \sim \mathcal{N}(0, Q), \quad \boldsymbol{v} = \begin{bmatrix}\boldsymbol{v}_\tau\\ \bar{\boldsymbol{v}}_f\end{bmatrix}, \quad Q = \begin{bmatrix}Q_\tau & \mathbf{0}\\ \mathbf{0} & Q_f\end{bmatrix}.$$

A visual summary of the continuous-time process model and the sensor models is given as a block scheme in Figure 3-2.

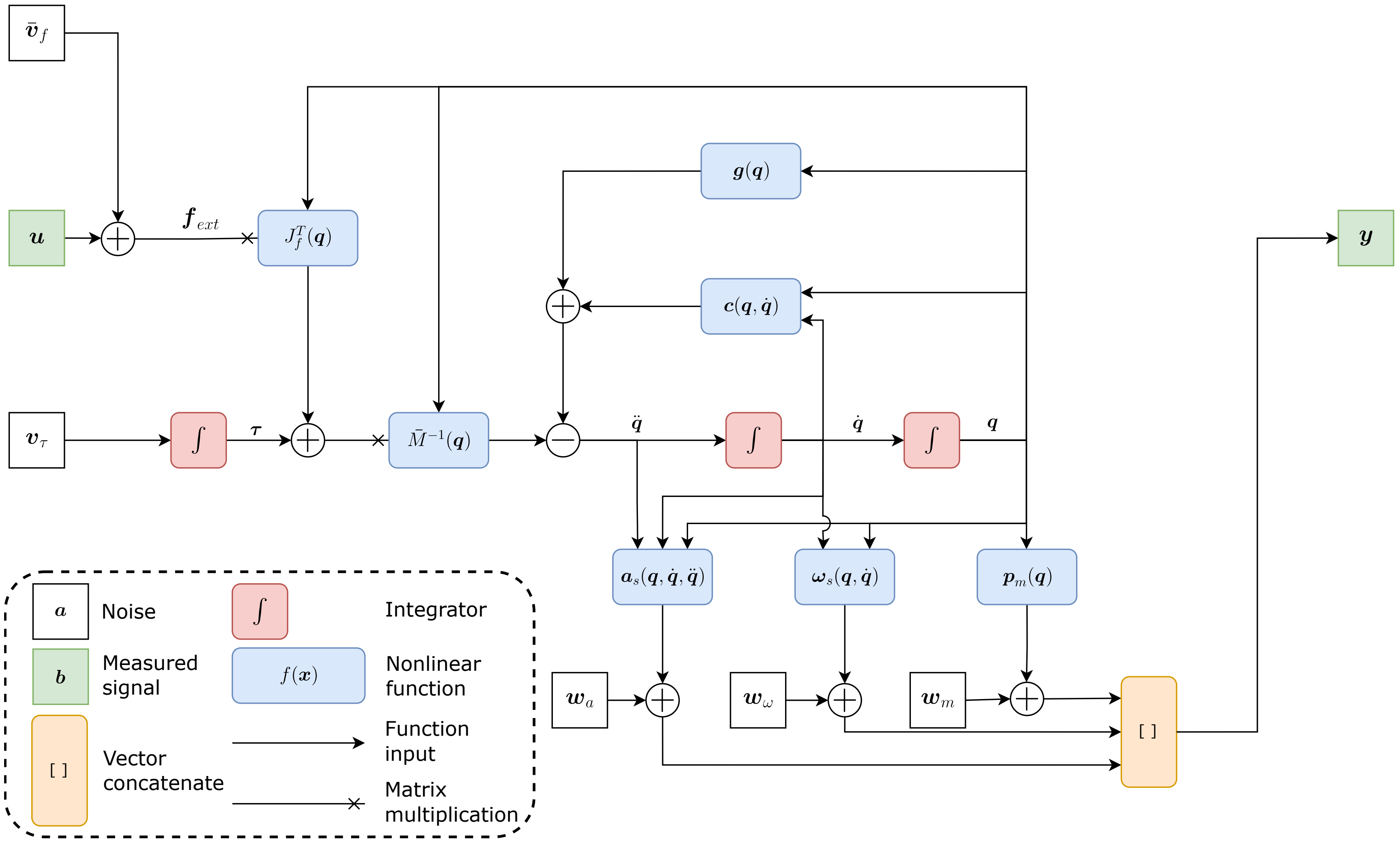


Figure 3-2: Block scheme of the continuous-time process model and the sensor models.

3-4-1 Discretization

Because of the sample-based nature of the available measurements, a discrete-time model is required. This means that the continuous time state dynamics $\bar{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}(t))$ from (3-24) should be discretized. To that end, a constant sampling time T is assumed. Additionally, it is assumed that the process noise $\boldsymbol{v}(t)$ and the input signal $\boldsymbol{u}(t)$ are constant on the interval $t \in [t_k, t_k + T]$, where k is the index of the current time instance. This corresponds to the zero-order hold (ZOH) assumption and allows us to make an approximation of the system dynamics on this interval [39]. Throughout this thesis, the notation $\boldsymbol{x}_k$ will be used as shorthand for $\boldsymbol{x}(t_k)$. The same shorthand is applied to other time-dependent variables such as $\boldsymbol{u}(t_k)$, $\boldsymbol{y}(t_k)$ et cetera. To find an expression for the state $\boldsymbol{x}_{k+1}$ given $\boldsymbol{x}_k$, $\boldsymbol{u}_k$ and $\boldsymbol{v}_k$ the following integral must be computed:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \int_{t_k}^{t_k+T} \bar{f}(\boldsymbol{x}(t), \boldsymbol{u}_k, \boldsymbol{v}_k) dt.$$

Because of the nonlinearities present in the dynamic model, it is generally not possible to compute this integral exactly. Instead, it is approximated by first linearizing the function $\bar{f}(\boldsymbol{x})$ around the current state $\boldsymbol{x}_k$. This linearization requires the computation of the Jacobians of $\bar{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v})$ with respect to $\boldsymbol{x}$, $\boldsymbol{u}$ and $\boldsymbol{v}$. While the model in (3-24) is highly nonlinear, the Jacobians possess some structure that can be exploited.

$$A(\boldsymbol{x}_k) = \left.\frac{\partial \bar{f}}{\partial \boldsymbol{x}}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_k \\ \boldsymbol{u}=\boldsymbol{u}_k \\ \boldsymbol{v}=0}} = \begin{bmatrix} \mathbf{0} & \mathcal{I}_{N_q} & \mathbf{0} \\ \frac{\partial \text{EoM}}{\partial \boldsymbol{q}_k} & \bar{M}^{-1}(\boldsymbol{q}_k)\frac{\partial \boldsymbol{c}}{\partial \dot{\boldsymbol{q}}_k} & \bar{M}^{-1}(\boldsymbol{q}_k) \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix},$$

$$B_u(\boldsymbol{x}_k) = \left.\frac{\partial \bar{f}}{\partial \boldsymbol{u}}\right|_{\boldsymbol{x}=\boldsymbol{x}_k} = \begin{bmatrix} \mathbf{0} \\ \bar{M}^{-1}(\boldsymbol{q}_k)\,{}^{\mathcal{S}_f}J_f^T(\boldsymbol{q}_k) \\ \mathbf{0} \end{bmatrix},$$

$$B_v(\boldsymbol{x}_k) = \left.\frac{\partial \bar{f}}{\partial \boldsymbol{v}}\right|_{\boldsymbol{x}=\boldsymbol{x}_k} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \bar{M}^{-1}(\boldsymbol{q}_k)\,{}^{\mathcal{S}_f}J_f^T(\boldsymbol{q}_k) \\ \mathcal{I}_{N_q} & \mathbf{0} \end{bmatrix}. \tag{3-25}$$

Where $\frac{\partial \text{EoM}}{\partial \boldsymbol{q}_k}$ is used as a shorthand for

$$\frac{\partial \text{EoM}}{\partial \boldsymbol{q}_k} = \frac{\partial}{\partial \boldsymbol{q}_k}(\bar{M}^{-1}(\boldsymbol{q}_k)(\boldsymbol{\tau}_k + {}^{\mathcal{S}_f}J_f^T(\boldsymbol{q}_k)\boldsymbol{u}_k - \boldsymbol{c}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k) - \boldsymbol{g}(\boldsymbol{q}_k) + {}^{\mathcal{S}_f}J_f^T(\boldsymbol{q}_k)\bar{\boldsymbol{v}}_{f,k}).$$

The parts of this Jacobian denoted by $\frac{\partial \text{EoM}}{\partial \boldsymbol{q}_k}$ and $\frac{\partial \boldsymbol{c}}{\partial \dot{\boldsymbol{q}}_k}$ are shown in this implicit form because an analytic expression requires the derivative of the kinematic and system Jacobians, which is generally not available for arbitrary systems in simulation software such as OpenSim. The structure present in the Jacobian matrices $A(\boldsymbol{x}_k)$, $B_u(\boldsymbol{x}_k)$ and $B_v(\boldsymbol{x}_k)$ makes it possible to compute the remaining parts of them using built-in OpenSim functionality. This allows for an easy and efficient implementation of the proposed algorithm for a variety of models. The continuous time process model can be approximated using the Jacobian matrices from (3-25):

$$\dot{\boldsymbol{x}} = \bar{f}(\boldsymbol{x}) \approx \bar{f}(\boldsymbol{x}_k) + A(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k) + B_u(\boldsymbol{x}_k)\boldsymbol{u}_k + B_v(\boldsymbol{x}_k)\boldsymbol{v}_k.$$

This first-order linear differential equation does have an exact solution, given by

$$\boldsymbol{x}(t) = \exp(A(\boldsymbol{x}_k)(t-t_k))\boldsymbol{x}_k + \int_{t_k}^{t} \exp(A(\boldsymbol{x}_k)(t'-t_k))dt'(B_u(\boldsymbol{x}_k)\boldsymbol{u}_k + B_v(\boldsymbol{x}_k)\boldsymbol{v}_k + \bar{f}(\boldsymbol{x}_k) - A(\boldsymbol{x}_k)\boldsymbol{x}_k). \tag{3-26}$$

Introducing the matrices F, L_u and L_v and the vector $\bar{f}_k$ as

$$F(\boldsymbol{x}_k) = \exp(A(\boldsymbol{x}_k)T), \quad L_u(\boldsymbol{x}_k) = \int_0^T \exp(A(\boldsymbol{x}_k)t')dt'\, B_u(\boldsymbol{x}_k),$$
$$L_v(\boldsymbol{x}_k) = \int_0^T \exp(A(\boldsymbol{x}_k)t')dt'\, B_v(\boldsymbol{x}_k), \quad \bar{f}_k = \int_0^T \exp(A(\boldsymbol{x}_k)t')dt'\, (\bar{f}(\boldsymbol{x}_k) - A(\boldsymbol{x}_k)\boldsymbol{x}_k), \tag{3-27}$$

(3-26) can be used to compute $\boldsymbol{x}_{k+1}$ as

$$\boldsymbol{x}_{k+1} \approx F(\boldsymbol{x}_k)\,\boldsymbol{x}_k + L_k(\boldsymbol{x}_k)\boldsymbol{u}_k + L_v(\boldsymbol{x}_k)\boldsymbol{v}_k + \bar{f}_k = f(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k). \tag{3-28}$$

The matrices F, L_u and L_v and the vector $\bar{f}_k$ can be easily computed by applying the matrix exponential to a larger block matrix:

$$\begin{bmatrix} F(\boldsymbol{x}_k) & L_u(\boldsymbol{x}_k) & L_v(\boldsymbol{x}_k) & \bar{f}_k \\ \boldsymbol{0} & & \mathcal{I}_{(N_u+N_v+1)} & \end{bmatrix} = \exp\left(\begin{bmatrix} A(\boldsymbol{x}_k) & B_u(\boldsymbol{x}_k) & B_v(\boldsymbol{x}_k) & (\bar{f}(\boldsymbol{x}_k) - A(\boldsymbol{x}_k)\boldsymbol{x}_k) \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}\right). \tag{3-29}$$

3-4-2 Measurement model

The second component of the state space model is the measurement model. The purpose of this model is to capture the sensor models from Section 3-3 as a function of the state vector, the measured input signal and the noise signals. For the marker positions, this is trivial since the generalized coordinates are part of the state vector so (3-15) can be applied directly. In the case of IMU measurements, (3-23) has to be used to substitute for $\ddot{\boldsymbol{q}}$ in (3-20). For notational convenience, the intermediate variable $\boldsymbol{z}_k$ is defined here as

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{q}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \\ \bar{M}^{-1}(\boldsymbol{q})(\boldsymbol{\tau} + {}^{\mathcal{S}_f}J_f^T(\boldsymbol{q})\boldsymbol{u} - \boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}}) - \boldsymbol{g}(\boldsymbol{q}) + ({}^{\mathcal{S}_f}J_f)^T(\boldsymbol{q})\bar{\boldsymbol{v}}_f \end{bmatrix} = \xi(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}). \tag{3-30}$$

Combining (3-15) and (3-16) through (3-21) results in the overall measurement model

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_m \\ \boldsymbol{y}_{IMU} \end{bmatrix} = \begin{bmatrix} {}^{\mathcal{M}}\boldsymbol{p}_{m,1}(\boldsymbol{q}) + \boldsymbol{w}_{m,1} \\ \vdots \\ {}^{\mathcal{S}_1}\boldsymbol{a}_{s,1}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) + \boldsymbol{w}_{a,s,1} \\ {}^{\mathcal{S}_1}\boldsymbol{\omega}_{s,1}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{w}_{\omega,s,1} \\ \vdots \end{bmatrix}$$

$$= \begin{bmatrix} {}^{\mathcal{M}}\boldsymbol{p}_{m,1}(\boldsymbol{q}) + \boldsymbol{w}_{m,1} \\ \vdots \\ {}^{\mathcal{S}_1}J_{p,s,1}(\boldsymbol{q})\ddot{\boldsymbol{q}} + {}^{\mathcal{S}_i}\dot{J}_{p,s,1}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - {}^{\mathcal{S}_i}R_{\mathcal{G}}(\boldsymbol{q})\,{}^{\mathcal{G}}\boldsymbol{g} + \boldsymbol{w}_{a,s,1} \\ {}^{\mathcal{S}_i}J_{\omega,s,1}(\boldsymbol{q})\,\dot{\boldsymbol{q}} + \boldsymbol{w}_{\omega,s,1} \\ \vdots \end{bmatrix} = g(\boldsymbol{z}, \boldsymbol{w}). \tag{3-31}$$

The measurement function, as a function of the state and the noise variables, is obtained by composing (3-31) with (3-30). The time index k is used here to indicate the sample-based nature of the measurements $\boldsymbol{y}$.

$$\boldsymbol{y}_k = h(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k, \boldsymbol{w}_k) = g(\xi(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k), \boldsymbol{w}_k) \tag{3-32}$$

The introduction of the intermediate variable $\boldsymbol{z}$ and the function $g(\boldsymbol{z}, \boldsymbol{w})$ has another advantage for the purposes of this thesis. Note that, when the variable $\boldsymbol{z}_k$ is used as the state instead and the function $g(\boldsymbol{z}_k, \boldsymbol{w}_k)$ is used as the measurement function, a state space model for kinematics estimation as described in Section 2-3 is obtained. Of course, this also requires an adaptation of the process model but this will not be discussed here. By separating the measurement model in these two functions, the implementation of a kinematic nonlinear Kalman filter becomes almost trivial, which makes it easier to compare it with the proposed framework. To conclude this Chapter, Table 3-1 provides an overview of the most important variables and functions that were introduced.

Symbol	Meaning
$\boldsymbol{q}$	Generalized coordinates
$\dot{\boldsymbol{q}}$	Generalized velocities
$\ddot{\boldsymbol{q}}$	Generalized accelerations
$\boldsymbol{\tau}$	Generalized forces
M	Mass matrix
$J(\boldsymbol{q})$	System Jacobian
$\bar{M}(\boldsymbol{q})$	Generalized mass matrix
$\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}})$	Velocity-dependent forces
$\boldsymbol{g}(\boldsymbol{q})$	Gravity forces
${}^{\mathcal{M}}\boldsymbol{p}_m(\boldsymbol{q})$	Marker positions in the motion capture frame
${}^{\mathcal{M}}J_{p,m,i}(\boldsymbol{q})$	Kinematic Jacobian of the marker positions
${}^{\mathcal{S}_i}\boldsymbol{a}_{s,i}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$	Specific force in the sensor frame
${}^{\mathcal{S}_i}J_{p,s,i}(\boldsymbol{q})$	Kinematic Jacobian of the sensor position
${}^{\mathcal{S}_i}\dot{J}_{p,s,i}(\boldsymbol{q})$	Time derivative of the kinematic Jacobian of the sensor position
${}^{\mathcal{S}_i}\boldsymbol{\omega}_{s,i}(\boldsymbol{q}, \dot{\boldsymbol{q}})$	Angular velocity in the sensor frame
${}^{\mathcal{S}_i}J_{\omega,s,i}(\boldsymbol{q})$	Kinematic Jacobian of the sensor's angular velocity
$\boldsymbol{x}$	System state
$\boldsymbol{u}$	Measured external force input
$\boldsymbol{v}$	Process noise
$\boldsymbol{y}$	Measured sensor output
$\boldsymbol{w}$	Measurement noise
$f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v})$	State update function
$h(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w})$	Measurement function
$F(\boldsymbol{x})$	State update Jacobian
$L_u(\boldsymbol{x})$	Input Jacobian
$L_v(\boldsymbol{x})$	Process noise Jacobian
$H(\boldsymbol{x})$	Measurement Jacobian
$D_u(\boldsymbol{x})$	Input direct feedthrough matrix
$D_v(\boldsymbol{x})$	Process noise direct feedthrough matrix
M	Measurement noise Jacobian

Table 3-1: Important variables and functions used in the proposed estimation algorithm.

Chapter 4

Method

In this chapter, the concepts from Chapter 3 will be applied to derive two algorithms. First, the algorithm that was used to identify the dynamic model of the robotic manipulator, used in the validation experiments, will be described. Secondly, the iterated extended Kalman filter (IEKF) estimation framework, based on the state-space formulation from Section 3-4, will be explained. In this section, the focus will be on the implementation in the OpenSim simulation environment.

4-1 System Identification

In section 3-4, the state-space model used in the estimation procedure is described. In this state-space description, the multibody model describing the system dynamics is assumed to be known and constant. However, for the specific robotic manipulator that was used for the experimental validation portion of this thesis, the KUKA LBR iiwa 7 R800, no such model is available. The model for this robot most commonly used in the robotics community is amalgamation of different sources [40, 41] and is freely available in the robotics toolbox in MATLAB. While developing the experimental setup, it was found that this model could not provide sufficient accuracy to properly validate the estimated kinetics. The robot is equipped with a joint angle encoder and torque sensor in each of its seven revolute joints. The robot reads these sensors and outputs the measured joint angles, velocities and torques at a sampling rate of 200 Hz. In order to verify the accuracy of the model, the joint angle encoder measurements were used as the input to an inverse dynamics (ID) procedure. The result is shown in Figure 4-1 for joints five and six. The resulting joint torques did not match the measured joint torques.

For the joints shown in Figure 4-1, the root mean square error (RMSE) of the ID solution is even larger than the standard deviation (SD) of the measured joint torques. The other joints also showed substantial errors. This indicates an error in the underlying multibody model. This motivates the need to create a more accurate model using the sensor data made available by the robot. The procedure by which this model was created is further detailed in the remainder of this section.

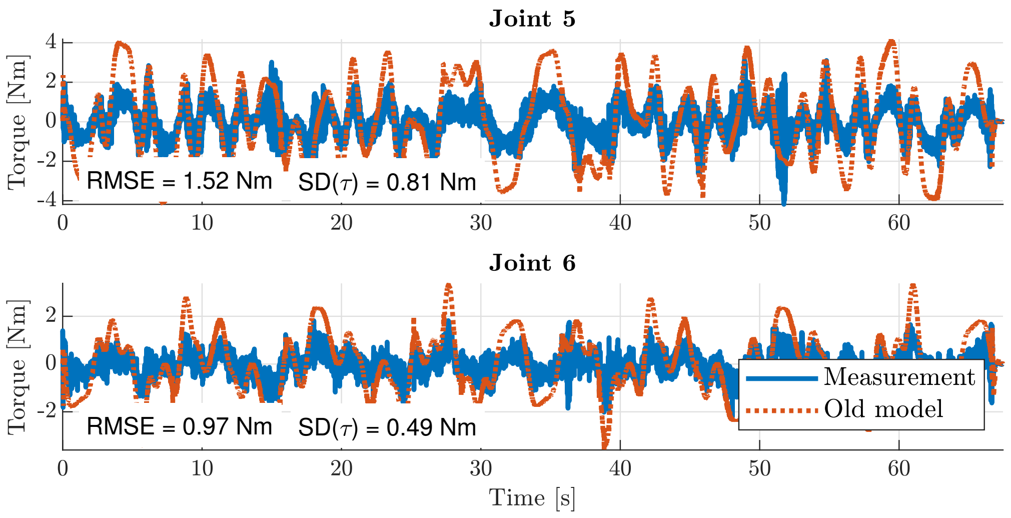


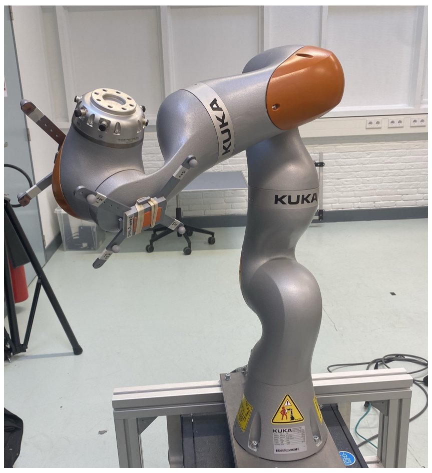
Figure 4-1: ID result of the old dynamic model, compared to the measured joint torques of the robot, for joints five and six.

4-1-1 Robot model

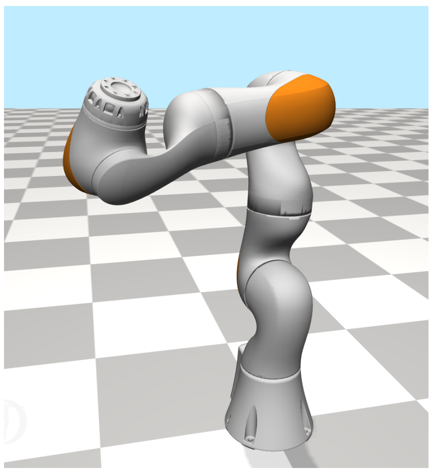
The model from the robotics toolbox was adapted for OpenSim by De Kanter [34]. It consists of eight rigid bodies. It is shown together with the real robot in Figure 4-2. The body closest to the ground, the base of the robot, is completely fixed to the ground. In OpenSim, this was modeled with a `WeldJoint`. The base is labeled as body zero. Moving up from the base, there are bodies one through seven. Each body is connected to the previous one by a `PinJoint`. The body's local coordinate frame is defined such that the origin is located at the attachment point to the previous body, the z-axis passes through the rotation axis of the pin joint and the y-axis is parallel with the rotation axis of the next pin joint. In total, this gives the robot seven unconstrained degrees of freedom (DoFs). When the joint angles of the seven pin joints are used as generalized coordinates, a minimal and unique description of the configuration of the robot is achieved.

$$\boldsymbol{q} = \begin{bmatrix} \theta_{0,1} \\ \theta_{1,2} \\ \vdots \\ \theta_{6,7} \end{bmatrix} \tag{4-1}$$

The aforementioned multibody model from the robotics toolbox provides not only the kinematics of the robot, i.e. the relative location of all the revolute joints. It also provides a dynamic model of the robot, consisting of the location of the center of mass, the mass and the 3×3 symmetric moment of inertia tensor for each of the rigid bodies. Both the kinematics and the dynamics are needed to describe the system kinetics. As mentioned, the model could not explain the measured joint torques in the ID analysis. The source of this error could be in the kinematic model, the dynamic model or both. In order to identify the source of the error, the model's kinematics were assessed using a motion capture system.



(a) The KUKA LBR iiwa 7 R800.



(b) The OpenSim model of the iiwa 7.

Figure 4-2: Side by side comparison of the real iiwa 7 robot and the OpenSim model.

This was done as follows:

1. Four markers were attached to each of the lower six of the robot's seven segments.
2. The markers' positions were measured using the motion capture system while the robot was not moving.
3. The measured joint angles in this static pose were prescribed to the model and the markers were registered onto the model in this pose.
4. The robot performed a motion and the markers' positions and the robot's joint angles were measured during this motion.
5. The measured joint angles were prescribed to the model and the model was then used to predict the marker positions.
6. The predicted marker positions were compared to the measured ones.

The RMSE of the marker positions predicted by the kinematic model, compared to the measured marker positions, was between one and six millimeters. This slight mismatch could not explain the large errors observed in the predicted torques, so the kinematics of the model were ruled out as the source of the error. In the remainder of this chapter, the kinematics provided by the model are therefore assumed to be correct.

This leaves the dynamic model as the source of the error. As will become clear in section 4-1-2, the identification of a new dynamic model is greatly simplified by introducing some more assumptions. Eliminating the location of the center of mass as a variable to identify gives the problem some nice properties that will make it feasible to find an optimal solution. Therefore, the location of the center of mass for each body was prescribed using the CAD geometry available in the robot model. The model provides an STL file for each of the robot's segments. The mean position of the vertices in this file was used as the location of the center of mass. Since all bodies look symmetric about the z,y-plane from the outside, the x-coordinate

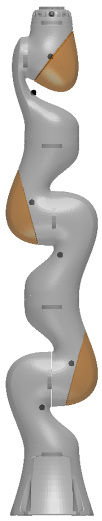


Figure 4-3: Prescribed locations of the robot bodies' centers of mass.

of the center of mass for each body was manually set to zero. The resulting locations of the bodies' centers of mass are shown in Figure 4-3 and summarized in Table 4-1. This leaves the mass and moment of inertia tensor for each body as optimization variables in the system identification problem.

	Center of mass location [m]		
	x	y	z
Body 1	0	-0.025	0.113
Body 2	0	0.040	0.049
Body 3	0	0.021	0.125
Body 4	0	0.039	0.049
Body 5	0	0.045	0.127
Body 6	0	0.002	0.057
Body 7	0	-0.000	0.027

Table 4-1: Locations of the centers of mass of each body, expressed in that body's own coordinate frame.

4-1-2 Optimization

The vector $\boldsymbol{x}$ of optimization variables is defined as

$$\boldsymbol{x}_i = \begin{bmatrix} m_i \\ I_{xx,i} \\ I_{xy,i} \\ I_{xz,i} \\ I_{yy,i} \\ I_{yz,i} \\ I_{zz,i} \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_7 \end{bmatrix}, \tag{4-2}$$

where m_i refers to the mass of segment i and $I_{xx,i}$ through $I_{zz,i}$ are the six unique elements of the symmetric 3×3 mass moment of inertia tensor of body i as introduced in Section 3-2. The joint torques can be written as a linear combination of the unknown optimization variables and a *data matrix* constructed from measured kinematics and known center of mass (CoM) coordinates:

$$\boldsymbol{\tau} = Y(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\, \boldsymbol{x}. \tag{4-3}$$

For the derivation of this data matrix $Y(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$, the reader is referred to Appendix B. This linear form will be employed to find values for the optimization variables that best match measured data from the robot. As mentioned, the robot outputs its measured joint angles, velocities and torques. In (4-3), the data matrix Y also depends on the joint accelerations. These were computed by a straightforward second order finite difference approximation of the joint velocities. Since there are more optimization variables in x than measured joint torques, (4-3) does not have a unique solution when only the measurements from one time instance are used. Furthermore, the measured joint torques are affected by sensor noise. By combining a series of measurements at different time instances, an overdetermined system of equations is formed and the effect of measurement noise is minimized. Let $\boldsymbol{q}_k$, $\dot{\boldsymbol{q}}_k$ and $\boldsymbol{\tau}_k$ be the measured joint angles, velocities and torques at time instance k. The joint acceleration $\ddot{\boldsymbol{q}}_k$ is then approximated from $\dot{\boldsymbol{q}}_{k-1}$ and $\dot{\boldsymbol{q}}_{k+1}$. The data matrix Y_k is defined as $Y_k = Y(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k)$. By introducing two new variables $\boldsymbol{y}$ and Γ, we can combine measurements from K different time instances.

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{\tau}_1 \\ \vdots \\ \boldsymbol{\tau}_K \end{bmatrix}, \quad \Gamma = \begin{bmatrix} Y_1 \\ \vdots \\ Y_K \end{bmatrix}$$

The goal is now to find the values of the optimization variables that minimize the difference between the modeled and measured joint torques for the given joint kinematics. This is formalized as a weighted least-squares optimization problem. The weighting matrix W is introduced to compensate for the fact that much larger torques are applied to the joints close to the base of the robot, compared to the joints closer to the end effector. Without any weighting, the cost function for the optimization problem would be dominated by these large torques. The weights were therefore chosen as follows: for each element i of the torque vector $\boldsymbol{\tau}$, the time series $\boldsymbol{\tau}_{1,i}, \ldots, \boldsymbol{\tau}_{K,i}$ was extracted from the measurement data. The terms in the error function $\boldsymbol{y} - \Gamma \boldsymbol{x}$ corresponding to that torque were then weighted by the inverse of the variance of the measured torque signal:

$$W = \begin{bmatrix} \frac{1}{\sigma^2_{\boldsymbol{\tau},1}} & & & & & & \\ & \ddots & & & & & \\ & & \frac{1}{\sigma^2_{\boldsymbol{\tau},7}} & & & & \\ & & & \ddots & & & \\ & & & & \frac{1}{\sigma^2_{\boldsymbol{\tau},1}} & & \\ & & & & & \ddots & \\ & & & & & & \frac{1}{\sigma^2_{\boldsymbol{\tau},7}} \end{bmatrix}, \quad \sigma^2_{\boldsymbol{\tau},i} = \operatorname{var}(\boldsymbol{\tau}_{1,i}, \ldots, \boldsymbol{\tau}_{K,i}).$$

With weighting, the least-squares cost function becomes

$$V(\boldsymbol{x}) = \|\boldsymbol{y} - \Gamma \boldsymbol{x}\|_W^2 = (\boldsymbol{y}^T - \boldsymbol{x}^T \Gamma^T)\, W\, (\boldsymbol{y} - \Gamma \boldsymbol{x})$$

The main advantage of the structure of this cost function is that the matrix Γ only has to be computed once in order to solve the problem. Computing the center of mass Jacobian for each body and each time instance can be time consuming, so formulating the cost function this way greatly improves the efficiency of the system identification procedure. The unconstrained weighted linear least-squares problem has an analytical solution in the form of

$$\underset{x}{\operatorname{argmin}} \|\boldsymbol{y} - \Gamma \boldsymbol{x}\|_W^2 = (\Gamma^T W \Gamma)^{-1} \Gamma^T W \boldsymbol{y}.$$

However as will be made clear in Subsection 4-1-2, the problem at hand has some nonlinear constraints owing to the physical meaning of the optimization variables. Because the joint angle, velocity and torque measurements are noisy, the solution to the unconstrained problem is not guaranteed to satisfy these constraints. This means that an iterative solver is needed to find a feasible solution. To maximize the efficiency of the solver, it is beneficial to rewrite the cost function in terms of the unconstrained solution, which allows us to eliminate the large matrix-vector product $\Gamma \boldsymbol{x}$. This will speed up the process of finding a feasible solution. Let $\boldsymbol{x}_0$ be the unconstrained solution

$$\boldsymbol{x}_0 = (\Gamma^T W \Gamma)^{-1} \Gamma^T W \boldsymbol{y}. \tag{4-4}$$

Notice that the cost function $V(x)$ can now equivalently be written as

$$V(\boldsymbol{x}) = (\boldsymbol{x}^T - \boldsymbol{x}_0^T)\, (\Gamma^T\, W\, \Gamma)\, (\boldsymbol{x} - \boldsymbol{x}_0) + \boldsymbol{y}^T\, (\mathcal{I} - W^T \Gamma (\Gamma^T W \Gamma)^{-1} \Gamma^T W)\, \boldsymbol{y}$$

By eliminating the constant term $\boldsymbol{y}^T\, (\mathcal{I} - W^T \Gamma (\Gamma^T W \Gamma)^{-1} \Gamma^T W)\, \boldsymbol{y}$, which does not affect the location of the minimum, an augmented cost function is defined.

$$\bar{V}(\boldsymbol{x}) = (\boldsymbol{x}^T - \boldsymbol{x}_0^T)\, (\Gamma^T\, W\, \Gamma)\, (\boldsymbol{x} - \boldsymbol{x}_0)$$

Evaluating this cost function only requires multiplication with the matrix $\Gamma^T\, W\, \Gamma$, which can be pre-computed and the size of which does not depend on the number of data points. This augmented cost function was used in the final implementation of the system identification procedure. In practice, the data matrix Γ is not necessarily full-rank. Consider for example the first segment of the robot. Since it can only rotate about its own z-axis, the optimization variable $I_{xx,1}$ does not affect the measured joint torques, reducing the rank of Γ. This means the inverse in (4-4) does not actually exist. In the implementation of the system identification procedure, the MATLAB function `lsqminnorm` was used to compute the unconstrained solution instead. This function minimizes the cost function $V(x)$ and when the minimizing solution is not unique, it minimizes the norm of $\boldsymbol{x}_0$ as well. This means that, in the case where multiple solutions result in the same cost, the solver prefers the solution with lower masses and inertias.

Constraints

Because of the definition of the optimization variables in (4-2), some constraints must be imposed on the solution of the optimization problem posed in Section 4-1-2. Most obviously, the mass of each body should be positive.

$$m_i > 0 \quad \forall i \tag{4-5}$$

Additionally, the sum of the individual masses should be equal to the total mass of the robot as specified by the manufacturer, $m_{total} = 23.9$kg [42]. However, the specified mass of the robot includes the base. The mass and moment of inertia tensor of the base are not included in the optimization variables because the base cannot move. The mass of the base was therefore set to 5 kg, based on the model from [40]. Because this is probably not the exact mass of the base, some slack was allowed in this constraint, turning it into an inequality.

$$0.75\,(m_{total} - m_{base}) \le \sum_{i=1}^{N} m_i \le 1.25\,(m_{total} - m_{base}) \tag{4-6}$$

From the definition of the moment of inertia tensor [36], it follows that

$$I_{xx,i} \le m_i d_{yz,i}^2, \tag{4-7}$$

where $d_{yz,i}$ is the maximum distance, in the y, z-plane of body i, that a point mass can be from the center of mass while still being within the geometry of the object. This is a very conservative upper bound on the size of the moment of inertia, since it is impossible for all of the segment's mass to be located so far from the center of mass. Similar constraints can be derived for $I_{yy,i}$ and $I_{zz,i}$. It was assumed that all the mass for a segment of the robot is contained within the bounding box provided by the STL file for that segment. The maximum distance from the center of mass to the edges of the bounding box was then used to define the constraints on $I_{xx,i}$, $I_{yy,i}$ and $I_{zz,i}$. These distances are illustrated for one of the segments in Figure 4-4.

Furthermore, to produce a physically permissible model, the eigenvalues of the moment of inertia tensor for each body must be positive. The necessity of this constraint can be made clear by noticing that a negative eigenvalue of the moment of inertia tensor would permit the

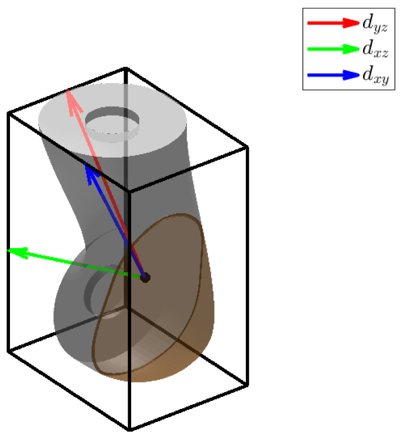


Figure 4-4: Bounding box of the geometry of segment two and the associated distances $d_{yz,2}$, $d_{xz,2}$ and $d_{xy,2}$.

body to have negative kinetic energy. Whereas constraints (4-5) to (4-7) are linear constraints, this constraint on the eigenvalues is nonlinear. This necessitates a nonlinear solver.

$$\text{eig}\left(\begin{bmatrix} I_{xx,i} & I_{xy,i} & I_{xz,i} \\ I_{xy,i} & I_{yy,i} & I_{yz,i} \\ I_{xz,i} & I_{yz,i} & I_{zz,i} \end{bmatrix}\right) > 0 \quad \forall i \tag{4-8}$$

The aforementioned constraints are necessary to produce a physically permissible model. However, as mentioned above, the data matrix Γ is not necessarily full-rank. This means that there might be multiple physically permissible solutions with the same value for the cost function $V(\boldsymbol{x})$. While implementing the optimization routine, this was observed in the tendency for the solver to concentrate almost all mass of the robot into one or two of its segments. While physically permissible, this is clearly not realistic. Therefore, additional constraints were specified based on basic observations of the robot. As can be seen in Figure 4-3, the first and third segments of the robot (counting from the base upwards) have nearly identical shapes and sizes. It was therefore assumed that the masses of these two segments do not differ by more than 1 kg. The same assumption was made for segments two and four. On top of that, it was assumed that the difference in mass between segments one and two does not exceed 2 kg. The MATLAB function `fmincon` was used to solve the nonlinear least-squares problem.

The solution of the optimization problem largely depends on the matrix Γ. The value of this matrix is determined completely by the trajectory that the robot is made to follow during the measurement. It is therefore critical to design the trajectory in such a way that the matrix Γ is as well-conditioned as possible. This was achieved by making sure the robot passes through many unique configurations during the measurement. The accuracy of the measurement is another factor of concern. As is the case with any measurement system, the joint angle encoders and torque sensors in the joints of the robot are affected by noise. The trajectory was designed to maximize the signal-to-noise ratio of these measurements by operating the robot close to its velocity limit. In the designed trajectory, each of the seven joint angles follows a sine wave, where the frequency of the sine wave is unique for each joint. The first joint, counting upwards from the base, moves at base frequency ω_0. The second joint moves at a frequency of $\frac{8}{7}\omega_0$, the third at $\frac{9}{7}\omega_0$ and so on up to $\frac{13}{7}\omega_0$. Because of this offset in frequency, this pattern only repeats itself after seven periods of the base frequency, ensuring a large number of unique configurations. The amplitude of each coordinate's sine wave was tuned manually to ensure that the robot would not collide with anything in its environment during the motion. The trajectory can be summarized mathematically as

$$\boldsymbol{q}(t) = \begin{bmatrix} \boldsymbol{q}_{amp,1} \sin(\omega_0 t) \\ \boldsymbol{q}_{amp,2} \sin(\frac{8}{7}\omega_0 t) \\ \boldsymbol{q}_{amp,3} \sin(\frac{9}{7}\omega_0 t) \\ \boldsymbol{q}_{amp,4} \sin(\frac{10}{7}\omega_0 t) \\ \boldsymbol{q}_{amp,5} \sin(\frac{11}{7}\omega_0 t) \\ \boldsymbol{q}_{amp,6} \sin(\frac{12}{7}\omega_0 t) \\ \boldsymbol{q}_{amp,7} \sin(\frac{13}{7}\omega_0 t) \end{bmatrix}, \quad \boldsymbol{q}_{amp} = \begin{bmatrix} 135^\circ \\ 60^\circ \\ 60^\circ \\ 90^\circ \\ 135^\circ \\ 90^\circ \\ 135^\circ \end{bmatrix} \tag{4-9}$$

Because all coordinates are determined by sine waves, the velocities can be easily found by differentiating the coordinate functions. Finally, the base frequency was tuned such that the largest velocity applied to each joint is lower than the maximum velocity specified by the manufacturer [42]. This was achieved by setting the base frequency to $\omega_0 = 0.66$ rad/s. The trajectory is completed after seven periods of the base frequency.

4-2 Framework for kinematics and kinetics estimation in OpenSim

Where the previous section dealt with the identification of the parameters of a dynamic model, in this section of the thesis we will assume that the dynamic model is both known and constant. Instead, we focus on the estimation of the unknown system state $\boldsymbol{x}_k$ of the state space model from Section 3-4, consisting of the generalized coordinates, velocities and forces. To find the unknown state, the IEKF state estimation algorithm is used. Besides making it possible to build on the work done by De Kanter [34], this structure has some other favorable properties. First of all, the IEKF estimates the state based on statistical properties of the input data and also returns an estimate of the uncertainty of the state. Secondly, the IEKF is a causal estimation method and has the possibility of extension to a real-time system. Finally and most importantly, since the extended Kalman filter (EKF) is, mathematically speaking, a special case of the IEKF, other methods from literature like the marker-based kinematic EKF discussed in Section 2-3 can be embedded in the new framework. This makes it easy to compare and identify the added benefit of novel additions like iteration, sensor fusion and kinetics estimation. Elements of the algorithm may be taken away selectively to identify their contribution to the end result. With this goal in mind, a number of requirements for the proposed framework was defined.

4-2-1 Requirements for the proposed framework

The framework should minimally be capable of estimating kinematics and kinetics using IMUs, optical markers or a combination of the two. Since these measurement systems are currently the most popular in biomechanics research, it would be enlightening to investigate their individual contributions to the estimation result. Enabling the use of both types of sensors also increases the ease with which the proposed framework could be adopted by the research community using existing datasets or measurement setups. In many practical applications the subject is interacting with the environment during the execution of a motor task. Often, the kinetics of this motion can not be accurately described without taking into account the interaction force between the subject and the environment. The framework should therefore be able to deal with measured interaction forces, to enable its use in a wider range of real life settings.

The goal of this thesis is to validate the proposed framework on a single, well-defined system. However, the ability to apply it to different systems or by using different models of the same system is an important feature. In the first place, this makes adoption of the framework in different settings much easier. Secondly, when an estimation method functions reliably on a variety of models, it can be used to study the influence of modeling choices on the estimation result. This is particularly interesting for biomechanics research since, when it comes to the human body, modeling choices are always subject to debate. To facilitate the use of different models, the framework should be set up such that the IEKF algorithm runs independently from the OpenSim model. This means properties like the state and output dimensions should scale based on the given model. It is however not the goal of this thesis to design a method that is guaranteed to work on all OpenSim models. For example, models with closed-loop constraints in the kinematic tree are not taken into account in this thesis. Also, the focus will be on the estimation of joint kinematics and kinetics, since these can be validated using the robotic arm. The muscle redundancy problem is therefore also not in the scope of this thesis.

4-2-2 The IEKF algorithm for kinematics and kinetics estimation

The IEKF as implemented in this thesis mostly corresponds to the 'textbook version' [35]. This algorithm can be interpreted as solving a maximum a posteriori (MAP) nonlinear estimation problem. One notable difference to the textbook implementation was introduced in this thesis: in the state space model derived in Section 3-4, the process noise that corrupts the measured force, $\bar{\boldsymbol{v}}_f$, directly affects the measurements of the accelerometers through the measurement function $h(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w})$. This in turn affects the estimated covariance of the measurement, which needs to be taken into account when computing the MAP estimate. The IEKF prediction and correction steps, as implemented in the estimation framework, are given below. The prediction step remains the same as for the textbook version. Both the state one time step ahead and its error covariance P are estimated in this step. To propagate the error covariance, the Jacobians of the discrete time state update function from Section 3-4-1 are used. The process noise covariance matrix Q, consisting of the covariance matrices of the generalized forces' random walk model and the noise of the force sensor, is applied in this step.

$$\hat{\boldsymbol{x}}_{k|k-1} = f(\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{0}) \tag{4-10a}$$
$$\hat{P}_{k|k-1} = F(\hat{\boldsymbol{x}}_{k-1|k-1})\, \hat{P}_{k-1|k-1}\, F(\hat{\boldsymbol{x}}_{k-1|k-1})^T + L_v(\hat{\boldsymbol{x}}_{k-1|k-1})\, Q\, L_v(\hat{\boldsymbol{x}}_{k-1|k-1})^T. \tag{4-10b}$$

The goal of the algorithm is to find a state estimate that minimizes both the deviation from the predicted state and from the observed measurement $\boldsymbol{y}_k$, according to the measurement model. These two error terms are weighted by the inverse of their respective estimated covariances. The covariances in turn are estimated by linearizing the system at the current best estimate, so a degree of approximation is introduced here. The linearization of the measurement model requires the computation of the Jacobians of the measurement fuction $h(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w})$ with respect to $\boldsymbol{x}$, $\boldsymbol{v}$ and $\boldsymbol{w}$. The Jacobian with respect to $\boldsymbol{u}$ is not needed, since $\boldsymbol{u}$ is not a stochastic variable. If we recall the definition of the intermediate variable $\boldsymbol{z}$ and the intermediate functions $\xi(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v})$ and $g(\boldsymbol{z}, \boldsymbol{w})$ from Section 3-4-2, the Jacobians of h can be derived using the chain rule.

$$
\begin{aligned}
\Xi_x(\boldsymbol{x}_k) &= \left.\frac{\partial \xi}{\partial \boldsymbol{x}}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_k \\ \boldsymbol{u}=\boldsymbol{u}_k \\ \boldsymbol{v}=0}}, \quad \Xi_v(\boldsymbol{x}_k) = \left.\frac{\partial \xi}{\partial \boldsymbol{v}}\right|_{\boldsymbol{x}=\boldsymbol{x}_k}, \\
G(\boldsymbol{z}_k) &= \left.\frac{\partial g}{\partial \boldsymbol{z}}\right|_{\substack{\boldsymbol{z}=\boldsymbol{z}_k \\ \boldsymbol{w}=0}}, \quad M = \frac{\partial g}{\partial \boldsymbol{w}_k} = \mathcal{I}_{N_y}, \\
H(\boldsymbol{x}_k) &= \left.\frac{\partial h}{\partial \boldsymbol{x}}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_k \\ \boldsymbol{u}=\boldsymbol{u}_k \\ \boldsymbol{v}=0 \\ \boldsymbol{w}=0}} = G(\xi(\boldsymbol{x}_k, \boldsymbol{u}_k, 0))\, \Xi_x(\boldsymbol{x}_k), \\
D_v(\boldsymbol{x}_k) &= \left.\frac{\partial h}{\partial \boldsymbol{v}}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_k \\ \boldsymbol{u}=\boldsymbol{u}_k \\ \boldsymbol{v}=0 \\ \boldsymbol{w}=0}} = G(\xi(\boldsymbol{x}_k, \boldsymbol{u}_k, 0))\, \Xi_v(\boldsymbol{x}_k).
\end{aligned} \tag{4-11}
$$

The calculation of the Jacobians $F(\boldsymbol{x}_k)$, $L_v(\boldsymbol{x}_k)$, $\Xi_x(\boldsymbol{x}_k)$, $\Xi_v(\boldsymbol{x}_k)$ and $G(\boldsymbol{z}_k)$ in OpenSim will be detailed in Section 4-2-3. Because the predicted measurement is affected by the process noise according to (3-32), its covariance matrix features a term containing D_v and Q. This direct feedthrough is the only difference between the IEKF used in this thesis and the textbook IEKF. The minimization problem to be solved by the IEKF can be expressed in terms of the covariance matrices as

$$
\hat{\boldsymbol{x}}_{k|k} = \underset{\hat{\boldsymbol{x}}_{k|k}}{\operatorname{argmin}} (\hat{\boldsymbol{x}}_{k|k} - \hat{\boldsymbol{x}}_{k|k-1})^T\, \hat{P}_{k|k-1}^{-1}\, (\hat{\boldsymbol{x}}_{k|k} - \hat{\boldsymbol{x}}_{k|k-1}) + (\boldsymbol{y}_k - h(\hat{\boldsymbol{x}}_{k|k}))^T\, \bar{R}_k^{-1}\, (\boldsymbol{y}_k - \hat{\boldsymbol{x}}_{k|k}), \tag{4-12}
$$

where $\bar{R}_k = M\,R\,M^T + D_v(\hat{\boldsymbol{x}}_{k|k})\,Q\,D_v(\hat{\boldsymbol{x}}_{k|k})^T$. In this step, we assume that $\boldsymbol{v}_k$ and $\boldsymbol{w}_k$ are independent. The update step of the IEKF is equivalent to the Gauss-Newton method for optimization, applied to this MAP optimization problem. It is computed differently, but it can be shown that the end result is the same [35]. We set the current state estimate equal to the predicted state, $\hat{\boldsymbol{x}}_{k,i} = \hat{\boldsymbol{x}}_{k|k-1}$. We then proceed by calculating the predicted measurement $\hat{\boldsymbol{y}}_{k,i}$ and the matrix $S_{k,i}$.

$$
\hat{\boldsymbol{y}}_{k,i} = h(\hat{\boldsymbol{x}}_{k,i}, \boldsymbol{u}_k, \boldsymbol{0}, \boldsymbol{0}), \tag{4-13a}
$$

$$
S_{k,i} = H(\hat{\boldsymbol{x}}_{k,i})\, \hat{P}_{k|k-1}\, H(\hat{\boldsymbol{x}}_{k,i})^T + M\,R\,M^T + D_v(\hat{\boldsymbol{x}}_{k,i})\,Q\,D_v(\hat{\boldsymbol{x}}_{k,i})^T. \tag{4-13b}
$$

Next, the Kalman gain $K_{k,i}$ may be computed as

$$K_{k,i} = \hat{P}_{k|k-1}\, H(\hat{\boldsymbol{x}}_{k,i})^T\, S_{k,i}^{-1}. \tag{4-14}$$

To find the MAP estimate, the current state estimate is updated iteratively. The next iteration of the state estimate is computed by comparing the predicted measurement to the real one:

$$\hat{\boldsymbol{x}}_{k,i+1} = \hat{\boldsymbol{x}}_{k|k-1} + K_{k,i}\,(\boldsymbol{y}_k - \hat{\boldsymbol{y}}_{k,i} - H(\hat{\boldsymbol{x}}_{k,i})\,(\hat{\boldsymbol{x}}_{k|k-1} - \hat{\boldsymbol{x}}_{k,i})). \tag{4-15}$$

After the next iteration has been computed, the procedure starts again from (4-13). The state estimate $\hat{\boldsymbol{x}}_{k,i}$ is updated until it converges or until a maximum number of iterations is reached. The limit on the number of iterations is necessary, since the Gauss-Newton method is not guaranteed to converge [43]. Throughout this thesis, a maximum number of iterations of $N_{iter} = 5$ was used. Convergence was determined by comparing the norm of the step size to the norm of the estimated state. When the norm of the step was less than 10^{-5} times the size of the norm of the estimated state, the iterations were stopped. Once either condition is satisfied, the final iteration is taken as the new state estimate and the state covariance matrix is updated:

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k,i}, \tag{4-16a}$$

$$\hat{P}_{k|k} = (I - K_{k,i}\, H(\hat{\boldsymbol{x}}_{k,i}))\, \hat{P}_{k|k-1}. \tag{4-16b}$$

From this point, the state estimate $\hat{\boldsymbol{x}}_{k|k}$ is stored and the procedure is repeated from (4-10) for the next time instance. To initialize the algorithm at the first time instance $k = 0$, an estimate of the initial state $\hat{x}_0$ and its error covariance $\hat{P}_0$ should be specified by the user. This initial state estimate is then used in (4-13).

4-2-3 Implementation of the state space model in OpenSim

The functions f and h and the Jacobians F, L_v, H and D_v need to be recomputed at every time step. In fact, h , H and D_v need to be computed for each iteration within a time step. To keep the algorithm computationally tractable, it is important to make sure that the functions and their Jacobians are computed in an efficient way. The IEKF framework runs in MATLAB (R2021a) [44] and the required OpenSim functionality is accessed through the OpenSim application programming interface (API) [45]. The goal is to compute the functions f and h and their Jacobians using built-in OpenSim functions wherever possible. MATLAB functions are used for operations that have no OpenSim implementation. The matrices F and L_v are computed from A and B_v in MATLAB using (3-29). A and B_v are in turn computed using OpenSim functions. Likewise, f is computed from $\bar{f}$ using (3-28) in MATLAB and $\bar{f}$ is computed in OpenSim. (4-17) shows the definition of the matrices A and B_v from (3-25) again. The definitions of the matrices Ξ_x, Ξ_v and G that were introduced in (4-11) are also given here. From Ξ_x, Ξ_v and G, the required Jacobians H and D_v can be calculated. In this equation, elements of the matrix are colored green if they can be computed using built-in OpenSim functions. They are colored red if they were approximated numerically using a finite difference method. The parts of the matrices that are not colored were computed directly in

MATLAB. A detailed description of the OpenSim API functions needed to compute the state update function, the measurement function and their Jacobians can be found in Appendix A.

$$A(\boldsymbol{x}_k) = \begin{bmatrix} \mathbf{0} & \mathcal{I}_{N_q} & \mathbf{0} \\ \frac{\partial \text{EoM}}{\partial \boldsymbol{q}_k} & \bar{M}^{-1}(\boldsymbol{q}_k)\frac{\partial \boldsymbol{c}}{\partial \dot{\boldsymbol{q}}_k} & \bar{M}^{-1}(\boldsymbol{q}_k) \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad B_v(\boldsymbol{x}_k) = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \bar{M}^{-1}(\boldsymbol{q}_k)\,{}^{\mathcal{S}_f}J_f^T(\boldsymbol{q}_k) \\ \mathcal{I}_{N_q} & \mathbf{0} \end{bmatrix},$$

$$\Xi_x(\boldsymbol{x}_k) = \begin{bmatrix} \mathcal{I}_{N_q} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathcal{I}_{N_q} & \mathbf{0} \\ \frac{\partial \text{EoM}}{\partial \boldsymbol{q}_k} & \bar{M}^{-1}(\boldsymbol{q}_k)\frac{\partial \boldsymbol{c}}{\partial \dot{\boldsymbol{q}}_k} & \bar{M}^{-1}(\boldsymbol{q}_k) \end{bmatrix}, \quad \Xi_v(\boldsymbol{x}_k) = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \bar{M}^{-1}(\boldsymbol{q}_k)({}^{\mathcal{S}_f}J_f)^T(\boldsymbol{q}_k) \\ \mathbf{0} & \mathbf{0} \end{bmatrix},$$

$$G(\boldsymbol{z}_k) = \begin{bmatrix} {}^{\mathcal{M}}J_{p,m,1}(\boldsymbol{q}_k) & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots \\ \frac{\partial}{\partial \boldsymbol{q}_k}\,{}^{\mathcal{S}_1}\boldsymbol{a}_{s,1}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}) & \frac{\partial}{\partial \dot{\boldsymbol{q}}_k}\,{}^{\mathcal{S}_1}\boldsymbol{a}_{s,1}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}) & {}^{\mathcal{S}_1}J_{p,s,1}(\boldsymbol{q}) \\ \frac{\partial}{\partial \boldsymbol{q}_k}\,{}^{\mathcal{S}_1}\boldsymbol{\omega}_{s,1}(\boldsymbol{q},\dot{\boldsymbol{q}}) & {}^{\mathcal{S}_1}J_{\omega,s,1}(\boldsymbol{q}) & \mathbf{0} \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{4-17}$$

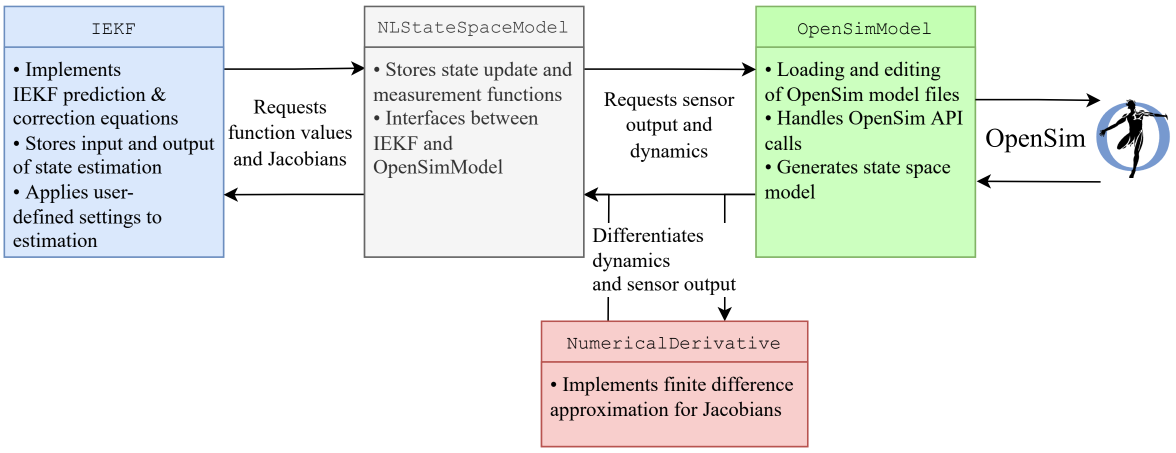

IEKF
• Implements IEKF prediction & correction equations
• Stores input and output of state estimation
• Applies user-defined settings to estimation
Requests function values and Jacobians
NLStateSpaceModel
• Stores state update and measurement functions
• Interfaces between IEKF and OpenSimModel
Requests sensor output and dynamics
OpenSimModel
• Loading and editing of OpenSim model files
• Handles OpenSim API calls
• Generates state space model
OpenSim
Differentiates dynamics and sensor output
NumericalDerivative
• Implements finite difference approximation for Jacobians

Figure 4-5: An overview of the four main classes responsible for the required computations in the IEKF estimation framework.

The algorithm was developed using MATLAB's built-in object oriented programming (OOP) tools. This made it easier to separate the algorithm into individual components. Using this approach, it is possible to swap out parts of the algorithm for different components to compare their performance. For example, a state space model could be defined which uses $\boldsymbol{z}_k$ as its state and $g(\boldsymbol{z}_k, \boldsymbol{w}_k)$ as its measurement function. This way, an implementation of the kinematic nonlinear Kalman filter described in Section 2-3 is obtained. The main body of the algorithm consists of four classes that are responsible for all computations. The `IEKF` class takes care of the estimation loop described in (4-10) to (4-16). It requests the value of the functions f and h and their Jacobians from the `NLStateSpaceModel` class, which is a general-purpose implementation of a nonlinear state space model. This intermediate class does not implement any computations itself, instead it simply delegates the request by the `IEKF` class to the `OpenSimModel` class, which contains all the information about the dynamic model, the sensors and the force inputs. The `OpenSimModel` class handles all communication with the OpenSim API. It also provides methods to place sensors on the model and perform ID using OpenSim's built-in `InverseDynamicsSolver`. For the elements of the Jacobians that cannot be computed analytically, the `OpenSimModel` class calls on the `NumericalDerivative` class to approximate them. This class implements a straightforward first order finite differencing scheme, which perturbs each element of the input of a function by a small amount ϵ and records the change in output. For all approximated Jacobians in this thesis, $\epsilon = 10^{-8}$ was used as the perturbation size. An overview of the four main classes and their interactions is given in Figure 4-5. With both the estimation framework and the OpenSim model in place, it is possible to process measured sensor data to estimate the kinematics and kinetics of the robot. The next chapter will deal with the practical aspect of gathering this data, so that the estimation framework may be validated.

Chapter 5

Experimental validation

In this thesis project, a number of experiments were conducted with a robotic manipulator. The experiments served two purposes: first, to provide the data required to identify an accurate dynamic model of the robot using the algorithm from Section 4-1. Secondly, to gather real-world data to validate the iterated extended Kalman filter (IEKF) algorithm described in Section 4-2 and to test its accuracy in realistic scenarios. What is meant by a 'realistic scenario' is a system with multiple degrees of freedom (DoFs) performing a motion in all three dimensions, with a range of motion between $40°$ and $230°$ in each joint. For these ranges of motion, corresponding to those found in human joints [46], the effects of the system's nonlinearity are significant and the ability of the IEKF algorithm to deal with these nonlinearities can be tested. In this chapter, the experimental setup that was used to gather all this data will be discussed. Then, each experiment will be described in detail, providing insight into the experimental procedure and the measures that were taken to ensure the accuracy and usability of the data.

5-1 Sensor system

Before discussing the individual experiments, a general overview of the sensor system is given. The IEKF algorithm introduced in Section 4-2 takes measured force/torque data, marker positions and IMU data as its inputs. It compares these measurements to the predicted sensor output computed in OpenSim. This means a virtual counterpart for each sensor is needed in OpenSim. The positions and orientations of the virtual markers, inertial measurement units (IMUs) and force/torque sensor in the OpenSim model should correspond to the real measurement setup. This problem is referred to as *sensor registration*. It is not possible to feed raw sensor data into the IEKF algorithm, for a number of reasons. First, the IEKF assumes that the measurements it receives were taken simultaneously and at a fixed rate. Secondly, real sensors suffer from imperfections such as bias, scaling errors and noise. The IEKF assumes the data is corrupted only by Gaussian white noise, so biases and scaling errors should be removed whenever possible. The sensor system has to solve three problems to make the data compatible with the IEKF algorithm:

- All virtual sensors should be registered onto the OpenSim model at the position and orientation of their real-world counterparts.
- The sensors should be calibrated to remove biases and scaling errors.
- The data coming from different sensors should be synchronized.

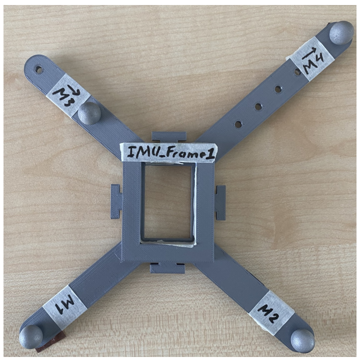
The sensor system is also responsible for collecting the ground truth data from the robot. The data collection, sensor registration and sensor calibration is discussed for each sensor type below. The synchronization of the different sensors is explained in Sections 5-3-1 and 5-3-2.

5-1-1 Robot sensing and control

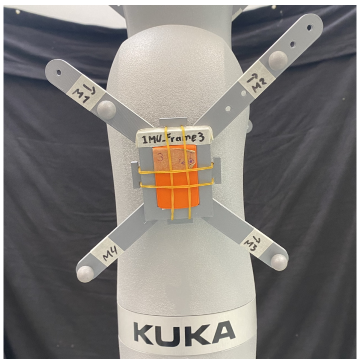
As mentioned in Chapter 4, the robot (KUKA LBR iiwa 7 R800) is equipped with joint angle encoders as well as joint torque sensors. The measurements coming from these sensors will serve as the ground truth against which the estimated kinematics and kinetics will be compared. The robot outputs these measurements via KUKA's Fast Research Interface (FRI) at a rate of 200 Hz. During the experiments, the FRI was not accessed directly. Instead, it was accessed through the Robot Operating System (ROS) [47]. ROS is an open-source software project for building robot applications. ROS was used because it makes it possible to collect the readings from all sensors in one place. There are many open-source libraries that can be used in conjunction with ROS to collect and store all the required sensor data. The ROS library that was used to read the joint sensors was also used to send commands to the robot via the FRI [48]. In all experiments discussed in this thesis, the robot is operated in position control mode, where the desired joint angles of a predefined trajectory are sent to the robot. The robot then attempts to steer the joints to the desired angles by increasing or decreasing the motor torque on each joint. A virtual stiffness parameter can be sent to the FRI, which determines the amount of torque the robot will use to compensate errors between the desired and measured joint angles. Different values for this virtual stiffness were used in different experiments.

5-1-2 Optical markers

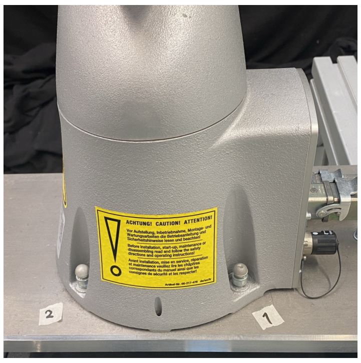
Passive optical markers were attached to the base and the lower six of the robot's seven segments. An OptiTrack motion capture system with nine Prime^x 13 cameras and the OptiTrack Motive software [49] was used to track the position of these markers at a rate of 100 Hz. The OptiTrack system was calibrated with the included calibration wand before each experiment. Four markers were attached to each segment. This was done to ensure that the position and orientation of each segment can be completely determined even when one of the markers is occluded. The markers on the base were only used for sensor registration of the other markers in OpenSim. The markers attached to segments one through six were used for the actual estimation. These markers were not placed on the robot directly, but on 3D printed frames, as shown in Figure 5-1a. The 3D printed frames have holes in them, so that the markers, which are made with threaded inserts, can simply be bolted on. Since the markers are not directly on the surface of the robot, they are less likely to be occluded by the robot itself.



(a) 3D printed frame with four markers.



(b) 3D printed frame with four markers and an IMU, attached to the robot.



(c) Two of the four markers attached to known locations on the base of the robot

Figure 5-1: Placement of the different sensors on the robot.

More importantly, when four markers are bolted onto a frame, they are always in the same location relative to each other. This makes it possible to define a *Rigid Body* in the Motive software. In Motive, a rigid body is a cluster of markers with a fixed shape. Now, whenever the software recognizes the shape of a cluster of four markers, it will label them according to which frame they were attached to. This way, even if a marker is temporarily out of view of the cameras, it will still be correctly labeled when it enters the view again. As can be seen in Figure 5-1a, the frames have a number of possible marker placements so that a unique cluster of four markers can be created for each segment of the robot. Additionally, the frames are made with a cutout that exactly fits the IMUs. The measured marker positions can then be used for the sensor registration of the IMUs, which will be explained in Section 5-1-3. The frame and IMU together form a rigid unit, which was attached to the robot using velcro tape as shown in Figure 5-1b.

Data collection

During the experiments, all data was recorded in Motive on a dedicated PC. Motive allows recording both the position of individual markers as well as the position and orientation of rigid bodies (marker clusters). The IEKF estimation framework relies on individual marker positions. However, the ROS library that was used together with Motive only allows to stream the rigid body poses [50]. Therefore, the following approach was taken: both the individual marker data as well as the rigid body poses were recorded in Motive. The rigid body poses were also streamed to a different PC running ROS. This made it possible to synchronize the Motive recording with the ROS recording after the experiment was finished.

Sensor registration

Since the base of the robot is fixed to the ground, the markers on the base of the robot were used to map the motion capture frame $\mathcal{M}$ to the OpenSim ground frame $\mathcal{G}$. These markers are labeled 1 through 4. These four markers were placed on fixed locations for which the coordinates in the OpenSim ground frame were known, shown in Figure 5-1c. The coordinates in OpenSim were determined from the available .STL file of the robot base. Before running the experiments, the robot was placed in a known reference pose. The positions of the base markers during this stationary trial were used to find ${}^{\mathcal{G}}H_{\mathcal{M}}$. Let ${}^{\mathcal{G}}\boldsymbol{p}_i$ be the known coordinates for $i = 1, 2, 3, 4$ and ${}^{\mathcal{M}}\bar{p}_{i,k}$ the measured positions at time instance k, for $k = 1, 2, ..., N_s$, with N_s the length of the stationary trial. The transformation is then determined by solving the quadratically constrained optimization problem

$$\begin{aligned} \{{}^{\mathcal{G}}R_{\mathcal{M}}, {}^{\mathcal{G}}\mathcal{O}_{\mathcal{M}}\} = &\underset{{}^{\mathcal{G}}R_{\mathcal{M}}, {}^{\mathcal{G}}\mathcal{O}_{\mathcal{M}}}{\operatorname{argmin}} \sum_{i=1}^{4} \sum_{k=1}^{N_s} \left\| {}^{\mathcal{G}}\boldsymbol{p}_i - {}^{\mathcal{G}}R_{\mathcal{M}}\, {}^{\mathcal{M}}\bar{p}_{i,k} - {}^{\mathcal{G}}\mathcal{O}_{\mathcal{M}} \right\|_2^2 \\ &\text{s.t.} \\ ({}^{\mathcal{G}}R_{\mathcal{M}})^T\, {}^{\mathcal{G}}R_{\mathcal{M}} &= \mathcal{I} \\ \det({}^{\mathcal{G}}R_{\mathcal{M}}) &= 1 \end{aligned} \tag{5-1}$$

and applying (3-2). The remaining 24 markers, labeled 5 through 28, still need to be registered to the model. The positions of these markers during the stationary trial were transformed to the OpenSim ground frame using ${}^{\mathcal{G}}H_{\mathcal{M}}$. By putting the OpenSim model in the same reference pose, the virtual markers could now be placed on their respective bodies in the OpenSim model at the measured position. During the actual experiments, the positions of markers 5 through 28 were also recorded. These are denoted by ${}^{\mathcal{M}}\bar{\boldsymbol{p}}_{i,k}$, for $i = 5, 6, ..., 28$ and $k = 1, 2, ..., N_k$ with N_k the length of the experiment. These positions were then transformed to the OpenSim ground frame to give the marker measurement vector $\boldsymbol{y}_{m,k}$:

$$\boldsymbol{y}_{m,k} = \begin{bmatrix} {}^{\mathcal{G}}R_{\mathcal{M}}\ {}^{\mathcal{M}}\bar{\boldsymbol{p}}_{5,k} + {}^{\mathcal{G}}\mathcal{O}_{\mathcal{M}} \\ \vdots \\ {}^{\mathcal{G}}R_{\mathcal{M}}\ {}^{\mathcal{M}}\bar{\boldsymbol{p}}_{28,k} + {}^{\mathcal{G}}\mathcal{O}_{\mathcal{M}} \end{bmatrix} \tag{5-2}$$

5-1-3 IMUs

In each of the six 3D printed frames, one wireless Xsens MTw Awinda Motion Tracker was placed. These devices function as IMUs, but also contain magnetometers, batteries and wireless communication hardware. They communicate with an Xsens Awinda Station via radio. The Awinda Station in turn is connected to a PC via USB. Using the software development kit (SDK) supplied by Xsens [51] and an open-source ROS library [52], the IMU measurements can be streamed directly to ROS at a rate of 100 Hz.

Calibration

Whereas Motive outputs data that is already calibrated, the IMUs stream their raw sensor readings to ROS. These sensor readings are affected by bias, noise and scaling errors. To remove as many sources of error as possible, the raw data was modified according to the same procedure used by De Kanter [34]. In short, this procedure consists of subtracting a bias from all gyroscope and accelerometer measurements and applying a linear transformation to the accelerometer measurements. This calibration procedure requires that two calibration trials are done for each sensor before each experiment:

1. One trial in which the sensor is completely stationary, which is used to estimate the gyroscope bias
2. One trial in which the sensor is slowly rotated to cover as many orientations as possible. By introducing the assumption that the accelerometer only measures the gravitational acceleration during this trial, the accelerometer bias, scaling error and non-orthogonality can be estimated.

The gyroscope bias for sensor i is denoted by $\boldsymbol{b}_{\omega,i}$ and the accelerometer bias by $\boldsymbol{b}_{a,i}$. The scaling and non-orthogonality of the accelerometer are grouped into the linear transformation D_i. The calibrated IMU measurement vector $\boldsymbol{y}_{IMU,k}$ can be constructed from the measured specific forces $\bar{\boldsymbol{a}}_{i,k}$ and angular velocities $\bar{\boldsymbol{\omega}}_{i,k}$:

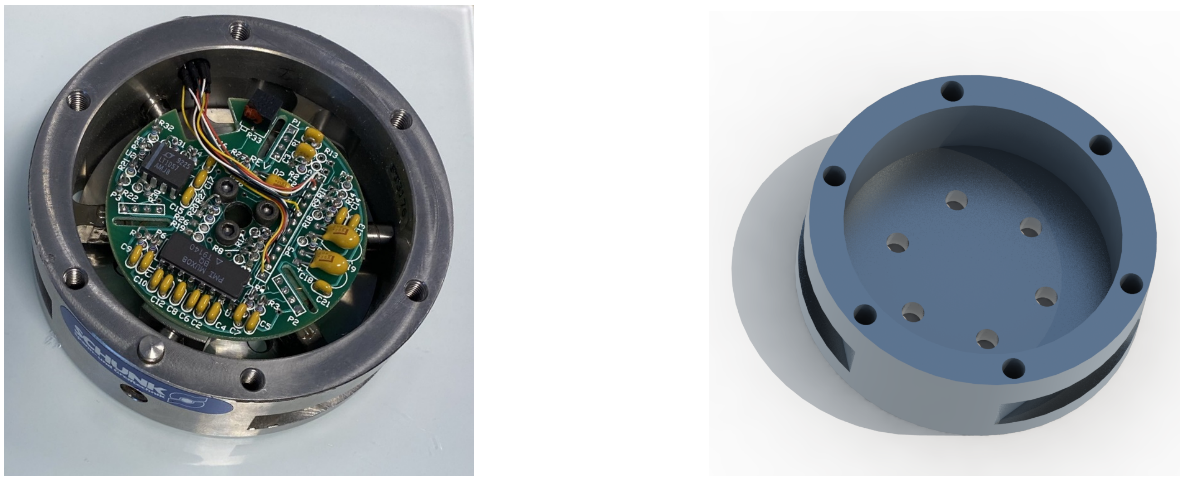
$$\boldsymbol{y}_{IMU,k} = \begin{bmatrix} D_1\left(\bar{\boldsymbol{a}}_{1,k} - \boldsymbol{b}_{a,1}\right) \\ \bar{\boldsymbol{\omega}}_{1,k} - \boldsymbol{b}_{\omega,1} \\ \vdots \\ D_6\left(\bar{\boldsymbol{a}}_{6,k} - \boldsymbol{b}_{a,6}\right) \\ \bar{\boldsymbol{\omega}}_{6,k} - \boldsymbol{b}_{\omega,6} \end{bmatrix} \tag{5-3}$$

Sensor registration

It is important that the virtual IMUs in OpenSim, which are essentially coordinate frames, have their origins at the same positions as the physical accelerometers and gyroscopes in the real IMUs. The axes of the coordinate frames should also be aligned with the axes of the sensors. The position and orientation of the sensors within the MTw Awinda Motion Trackers is given relative to the outer casing in the user manual [53]. In the experimental setup, each IMU was fixated in a 3D printed frame with optical markers. From the design of the frame, it is known where the markers are located relative to the IMU. The position of the markers can then be expressed in the IMU's sensor-fixed reference frame $\mathcal{S}_i$ as ${}^{\mathcal{S}_i}\boldsymbol{p}_j$. Given the measured positions of these markers, the transformation ${}^{\mathcal{S}_i}H_{\mathcal{M}}$ can be found by solving a problem similar to (5-1), where ${}^{\mathcal{S}_i}R_{\mathcal{M}}$ and ${}^{\mathcal{S}_i}\mathcal{O}_{\mathcal{M}}$ are the optimization variables instead of ${}^{\mathcal{G}}R_{\mathcal{M}}$ and ${}^{\mathcal{G}}\mathcal{O}_{\mathcal{M}}$ and the known positions are given by ${}^{\mathcal{S}_i}\boldsymbol{p}_j$. ${}^{\mathcal{G}}H_{\mathcal{S}_i}$ is then easily found by taking the inverse of ${}^{\mathcal{S}_i}H_{\mathcal{M}}$ and pre-multiplying it by ${}^{\mathcal{G}}H_{\mathcal{M}}$. For this procedure, the position data of markers 5 through 28 during the stationary trial was used. Markers 5 through 8 were attached to IMU 1, markers 9 through 12 to IMU 2 et cetera. From ${}^{\mathcal{G}}H_{\mathcal{S}_i}$, the position and orientation of the IMU can be obtained. This position and orientation were then prescribed to the virtual IMU in OpenSim, with the model again placed in the reference pose.

5-1-4 Force/torque sensor

During the experiment where an interaction force was applied to the robot, a 6-axis SCHUNK FTD-Delta SI-330-30 force/torque sensor was attached to the end effector via a 3D printed coupler. The sensor was connected to a data acquisition system by a cable, which somewhat limited the range of motion of the robot during this experiment. The data acquisition system was in turn connected to the PC, running a custom ROS library to stream the data at 100 Hz.



(a) The real force/torque sensor. **(b)** The virtual force/torque sensor.

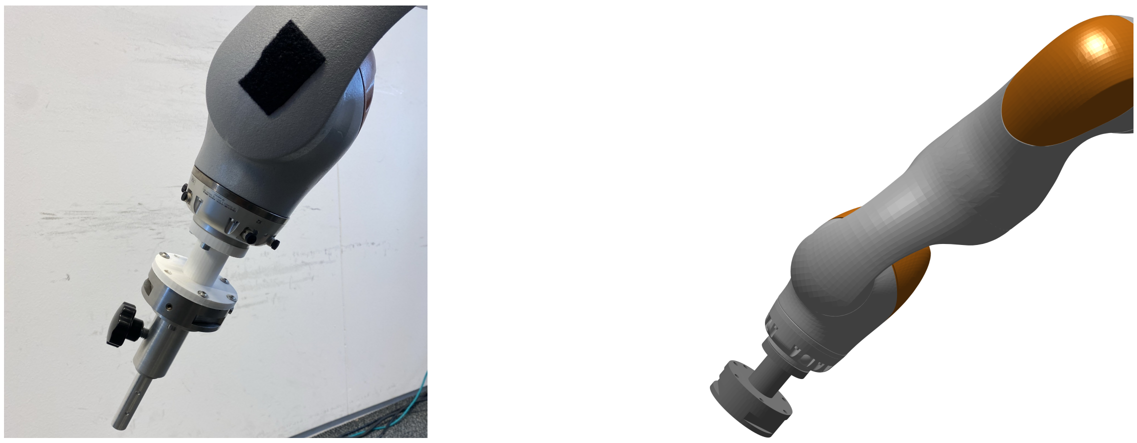
Figure 5-2: The real and virtual force/torque sensors side-by-side.

Sensor registration

In order to make sure that the measured interaction force is applied to the OpenSim model at the correct position and in the correct direction, a virtual copy of both the force/torque sensor and the 3D printed coupler were made. The OpenSim model of the robot was modified to include these components. The dimensions of the sensor were taken from the manufacturer's specifications [54]. Any dimensions not given in the specifications were measured by hand. A side-by-side comparison of the real sensor and its virtual counterpart is shown in Figure 5-2. In the experimental setup, the force/torque sensor was bolted rigidly to the coupler, which was bolted to the end effector of the robot. In OpenSim, this was modeled by adding a new body for the sensor and coupler and coupling it to the end effector body with a `WeldJoint`. The mass, center of mass and mass moment of inertia of the sensor should also be included in the OpenSim model if the kinetics of the robot are to be estimated accurately. The mass was determined by weighing the sensor and coupler. The mass of the sensor was 0.805 kg and the mass of the coupler was 0.147 kg. The center of mass and mass moment of inertia were approximated by changing the density of the 3D models in SolidWorks so that the mass matched the measured mass. Then, using the 'mass properties' tool, the center of mass and the 3×3 mass moment of inertia tensor were computed by SolidWorks. In the real setup, an additional handle was attached to the sensor so that the robot could be manipulated by moving the handle. The handle was not modeled in OpenSim, since its weight is already included in the measured interaction force. The real setup and the modified OpenSim model are shown in Figure 5-3. The position at and directions in which the sensor measures the forces and torques were also taken from the manufacturer's specifications.

Calibration

Like the IMUs' accelerometer data, the force/torque data is corrupted by sensor bias, scaling error and noise. In order to remove the bias and scaling error, the sensor was calibrated using the torque sensors in the robot. In a calibration trial, the robot was manipulated by moving


(a) The force/torque sensor and the handle as they were attached to the robot.

(b) The virtual force/torque sensor in the OpenSim model

Figure 5-3: Comparison of the force/torque sensor in the experimental setup and in OpenSim.

the handle on the force/torque sensor. The interaction force and the joint angles, velocities and torques were recorded during this trial. The joint accelerations were approximated using a second-order finite difference scheme. By applying (3-13) at each time instance of the calibration trial, the following set of equations is created:

$$\boldsymbol{\tau}_k + ({}^{\mathcal{S}_f}J_f)^T(\boldsymbol{q}_k)\,{}^{\mathcal{S}_f}\boldsymbol{f}_{ext,k} = \bar{M}(\boldsymbol{q}_k)\,\ddot{\boldsymbol{q}}_k + \boldsymbol{c}(\boldsymbol{q}_k,\dot{\boldsymbol{q}}_k) + \boldsymbol{g}(\boldsymbol{q}_k), \quad k = 1,2,...,N_c, \tag{5-4}$$

with N_c the length of the calibration trial. The right hand side of this equation can be computed by taking the kinematics measured by the robot and feeding them to OpenSim's built-in `InverseDynamicsSolver` tool, without any interaction force applied to the model. On the left hand side, $\boldsymbol{\tau}_k$ are the measured joint torques and ${}^{\mathcal{S}_f}J_f(\boldsymbol{q}_k)$ can be computed using OpenSim's `calcFrameJacobian` and `expressVectorInAnotherFrame` API functions. In this computation, the noise acting on the joint angles, velocities and torques is neglected. This leaves only the true interaction forces and torques ${}^{\mathcal{S}_f}\boldsymbol{f}_{ext,k}$ as an unknown variable. By now introducing the assumption that the measured interaction forces and torques ${}^{\mathcal{S}_f}\bar{\boldsymbol{f}}_{ext,k}$ are only affected by constant bias and scaling errors, we find

$$ {}^{\mathcal{S}_f}\boldsymbol{f}_{ext,k} = \begin{bmatrix} s_{f,x} & 0 & 0 \\ 0 & s_{f,y} & 0 \\ 0 & 0 & s_{f,z} \end{bmatrix} {}^{\mathcal{S}_f}\bar{\boldsymbol{f}}_{ext,k} - \boldsymbol{b}_f. \tag{5-5}$$

Substituting this into (5-4), the unknown scaling and bias parameters and the measured torques are related linearly.

$$\underbrace{\bar{M}(\boldsymbol{q}_k)\,\ddot{\boldsymbol{q}}_k + \boldsymbol{c}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k) + \boldsymbol{g}(\boldsymbol{q}_k) - \boldsymbol{\tau}_k}_{\boldsymbol{\tau}_{ext,k}} = \underbrace{\left[({}^{S_f}J_f)^T(\boldsymbol{q}_k)\,\mathrm{diag}({}^{S_f}\bar{\boldsymbol{f}}_{ext,k}) \quad -({}^{S_f}J_f)^T(\boldsymbol{q}_k)\right]}_{Z_k} \underbrace{\begin{bmatrix} s_{f,x} \\ s_{f,y} \\ s_{f,z} \\ \boldsymbol{b}_f \end{bmatrix}}_{\boldsymbol{z}_f} \tag{5-6}$$

In reality, the measured force data is also affected by noise, which is ignored in (5-5). Under the assumption that this noise is Gaussian white noise, $\boldsymbol{\tau}_{ext,k}$ and Z_k are computed for each time instance in the calibration trial to form an overdetermined system of linear equations, which can be solved in a least-squares sense to find the vector $\boldsymbol{z}_f$ containing the bias and scaling parameters for the force sensor. The external force input for the IEKF algorithm is computed by applying (5-5) to the data measured during the actual experiments.

$$\boldsymbol{u}_k = \begin{bmatrix} s_{f,x} & 0 & 0 \\ 0 & s_{f,y} & 0 \\ 0 & 0 & s_{f,z} \end{bmatrix} {}^{S_f}\bar{\boldsymbol{f}}_{ext,k} - \boldsymbol{b}_f. \tag{5-7}$$

5-2 Data collection for system identification

To carry out the system identification procedure described in Section 4-1, a dataset with known joint angles, velocities and torques is required. To obtain this data, the coordinate trajectory given in (4-9) was prescribed to the robot controller in ROS. To ensure the robot tracks the desired trajectory closely, a high virtual stiffness value of 500 (in normalized units) was given to each joint in the controller settings. During the experiment, no external force was applied to the robot. For the system identification algorithm, only the robot's internal sensor measurements are required. Therefore, no other sensors were attached to the robot during the trial. The robot's internal sensor measurements were recorded in ROS at 200 Hz. These measurements were used directly in the system identification procedure explained in Section 4-1. The results of the identification are detailed in Section 6-1.

5-3 Data collection for validation of the IEKF

To validate the IEKF algorithm and to determine its accuracy, two experiments were carried out. In these experiments, the motion of the robot was recorded by the internal joint sensors, the motion capture system and the IMUs described in the previous section. During both experiments, the seventh segment of the robot, the end effector, was not equipped with any sensors because of a lack of available markers and IMUs. Because this renders the seventh joint angle and torque unobservable, this angle was fixed to $0°$ in both the OpenSim model and the robot controller, effectively making segments six and seven one rigid body. This means six joint angles, velocities and torques were estimated, resulting in a state space model with 18 states. The accuracy of the estimated kinematics and kinetics was determined by comparing the measured and estimated values of only the first six joints. The characteristics

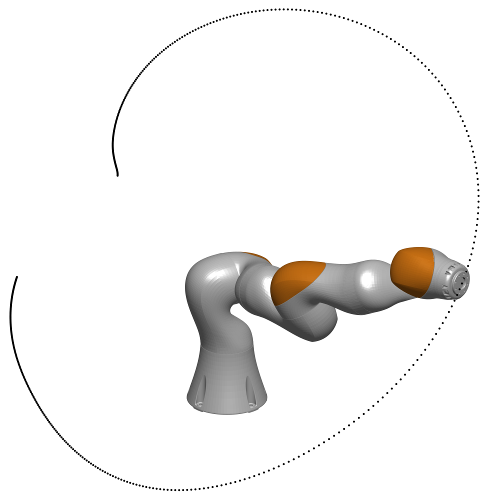


Figure 5-4: The robot in the base pose of the pre-defined trajectory, where the velocity of the end effector is maximized. Also shown is the trajectory of the end effector during the sweeping motion.

of the motions performed by the robot during the two experiments were different. During the first experiment, the robot followed a pre-defined trajectory close to its velocity limits. The trajectory generation and the limits of the robot are detailed in Section 5-3-1. The aim of this experiment was to determine the accuracy of the algorithm when estimating highly dynamic motion. This is interesting because the assumption that the specific force measured by the IMUs is dominated by the gravitational acceleration is violated. Where some estimation methods from literature rely on this assumption, the IEKF does not. One of the goals of this experiment is to determine how this assumption affects the estimation accuracy. During this experiment, the force/torque sensor was not used and no interaction force was applied to the robot. In the second experiment, the influence of measured interaction forces on the estimated kinematics and kinetics was investigated. In this experiment, the force/torque sensor was attached to the robot. No trajectory was defined. Instead, the robot was manipulated by hand by moving the handle of the force sensor.

5-3-1 Fast motion experiment

The trajectory that was prescribed to the robot during the fast motion experiment was designed to apply a high linear and angular velocity and high linear acceleration to the markers and IMUs attached to the robot. This was done by first finding a configuration of the robot that maximizes the velocity of the end effector, when all joints are moving at their maximum speed. In this configuration, the robot arm is extended in a horizontal orientation. Then, a number of control points were defined around this base pose in configuration space to allow the robot to smoothly increase and decrease its joint velocities. These control points were chosen such that the robot did not exceed any joint limits and also did not collide with itself or its surroundings. The trajectory was then defined by a piecewise cubic spline that passes through each control point, resulting in a sweeping motion. The base pose and the position of the end effector during the sweep are shown in Figure 5-4. In the simulation of this motion, the end effector reaches a linear velocity of 3.4 m/s and an angular velocity of 447.9 deg/s.

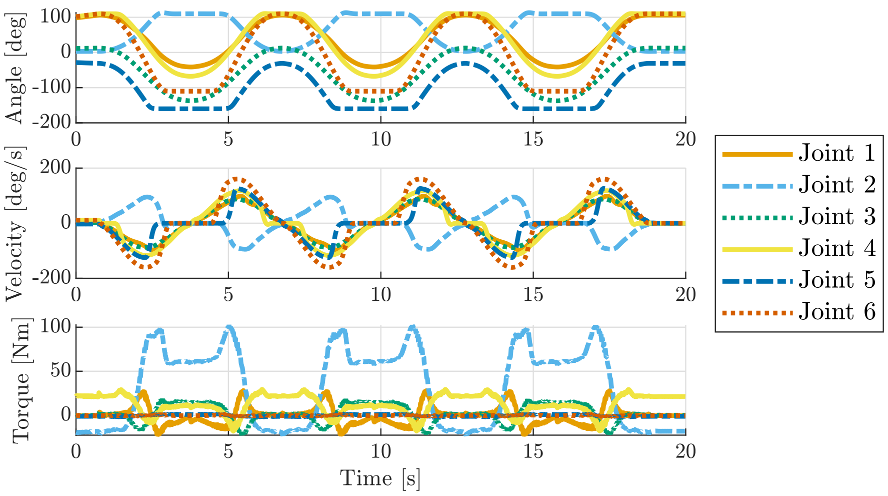


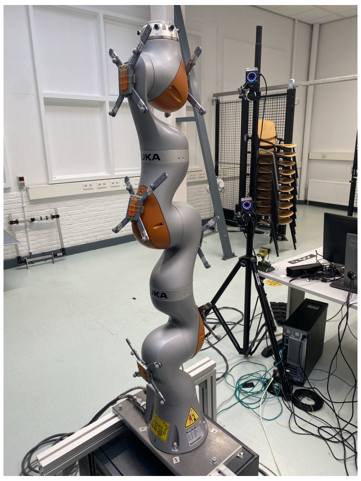
Figure 5-5: The trajectory of the measured joint angles, velocities and torques during the fast motion experiment.

A maximum linear acceleration of 16.9 m/s^2 is applied. The sweeping motion was performed by the real robot in both directions and then repeated three times. During this experiment, the same virtual stiffness parameter (500) was used to ensure accurate tracking. Altogether, this trial took 20 seconds to complete. The measured joint angles, velocities and torques are shown in Figure 5-5. The velocity limits per joint and the maximum observed velocity are compared in Table 5-1. For joint one, the observed velocity actually exceeds the limit, likely because the reported velocity limits include some safety margin. Table 5-1 also shows the range of motion for each joint, which is comparable to that of human joints [46].

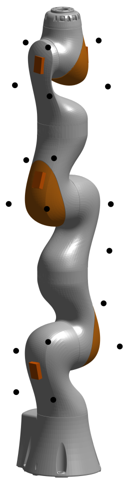
	Reported velocity limit [deg/s]	Maximum observed velocity [deg/s]	Range of measured motion
Joint 1	98	98.3	148°
Joint 2	98	95.1	110°
Joint 3	100	88.3	150°
Joint 4	130	118	179°
Joint 5	140	126	131°
Joint 6	180	160	220°
Joint 7	180	0	0°

Table 5-1: The velocity limits of each joint of the KUKA iiwa 7, as reported by the manual [42], the maximum velocity as measured by the joint encoders and the range of the measured motion during the fast motion experiment.

Before performing the sweeping motion, the robot was placed in a known reference pose, with all joint angles set to 0°, to perform the sensor registration of the markers and the IMUs as described in the previous section. The result of the sensor registration is visualized in Figure 5-6. After the stationary trial, the first joint was briefly moved backwards and forwards. This motion excited all sensors and could therefore be used to synchronize the sensors with each other.



(a) The experimental setup for the fast motion experiment.

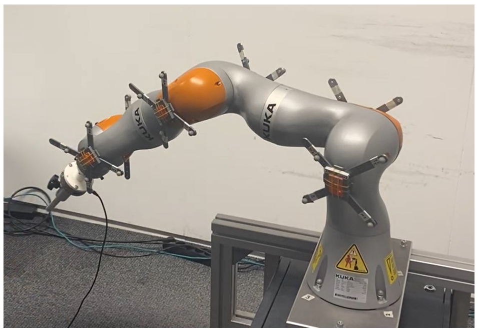


(b) The OpenSim model with its virtual sensors.

Figure 5-6: Comparison between the experimental setup and the OpenSim model during the fast motion experiment.

Synchronization

All data was streamed to ROS and recorded in real-time. In ROS, every incoming message was timestamped. However, this does not guarantee that the sensor readings occur simultaneously. Since the sensors sample at a rate of 100 Hz, this could introduce a mismatch of up to 0.005 seconds in the data when not accounted for. Furthermore, the data collection PC timestamps each message when it is received, not when it is sent. This means that any delay in the communication between the Motive PC and the ROS PC or between the wireless IMUs and the ROS PC is unaccounted for. To tackle this issue, the ground truth data collected by the robot was used to synchronize all sensors with each other. Only the data collected during the synchronization motion was used for this. Because the delay might be smaller than a single sampling interval, this data was upsampled in MATLAB to a rate of 10 kHz. The data recorded in Motive was synchronized to ROS by comparing the rigid body poses. The distance traveled by the first rigid body was calculated from both recordings. The cross-correlation between these two signals was computed. By finding the delay at which the cross-correlation is maximized, the delay in the measurement system can be estimated. Then, the distance traveled by the first marker was compared to the angle of the first joint measured by the robot. Again, the cross-correlation was used to estimate the delay between the robot data and the marker data. For the IMUs, the cross-correlation of the angular velocity of the IMU attached to the first segment and the velocity of the first joint of the robot was used to estimate the delay. It was assumed that the level of delay was the same for each IMU. By subtracting the estimated delays from the timestamps given to the data in ROS, the delays were removed from the measured data. To make sure that the measurements that are used in the IEKF are also simultaneous, the robot sensor data was first downsampled to 100 Hz. Then, the motion capture data and IMU data were interpolated linearly to match the timestamps of the robot data.


(a) The experimental setup for the interaction force experiment.

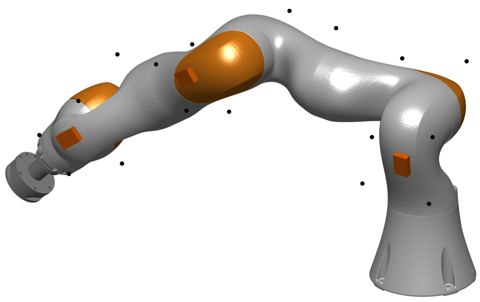

(b) The OpenSim model with its virtual sensors.

Figure 5-7: Comparison between the experimental setup and the OpenSim model during the interaction force experiment.

5-3-2 Interaction force experiment

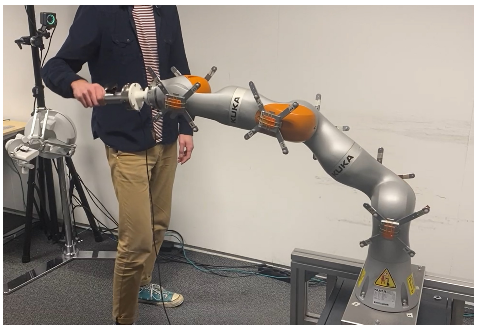


Figure 5-8: The author moving the robot by hand during the interaction force experiment.

The second experiment was performed with all three sensor systems (markers, IMUs, force/-torque sensor) attached to the robot. No trajectory was pre-defined for this experiment. The motion was generated by moving the robot around by hand. This was done by steering it to the configuration where all joint angles are 0°, but setting the virtual stiffness to a very low value of 10. Only the virtual stiffness of the end effector was kept at 500. With this low stiffness, the robot does not supply enough torque to support its own weight so it must be carefully moved to a position where it is in static equilibrium. Once in this position, the stationary trial for the sensor registration was recorded. The result of the sensor registration is shown in Figure 5-7. From the equilibrium position, it could then easily be moved by hand. During the experiment, the robot was moved for a total of 30 seconds. The experiment is shown in Figure 5-8. When inspecting the data, it was found that two markers were mislabeled in four frames of the Motive recording. This data was edited manually to correct the marker labels. The recorded joint angles, velocities and torques are shown in Figure 5-9. For comparison, the maximum joint velocity and the joint range of motion are given for this experiment as well in Table 5-2. As expected, the ranges of motion in this experiment are smaller than those of the fast motion experiment. The maximum velocities are still close to the reported limits and sometimes even larger. Despite its large virtual stiffness value, the end effector did move somewhat under the influence of the applied forces and torques.

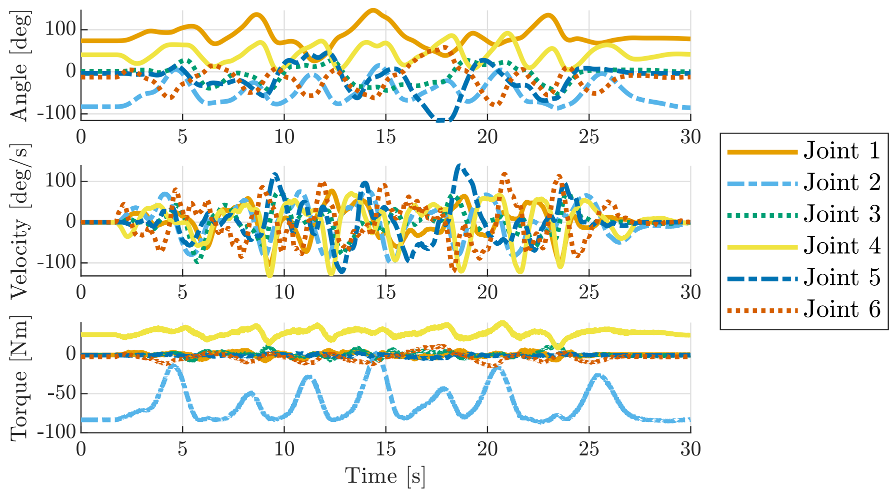


Figure 5-9: The trajectory of the measured joint angles, velocities and torques during the interaction force experiment.

	Reported velocity limit [deg/s]	Maximum observed velocity [deg/s]	Range of measured motion
Joint 1	98	104	120°
Joint 2	98	98.8	102°
Joint 3	100	97.4	82°
Joint 4	130	132	88°
Joint 5	140	138	161°
Joint 6	180	120	138°
Joint 7	180	4.09	1°

Table 5-2: The velocity limits of each joint of the KUKA iiwa 7, the maximum velocity as measured by the joint encoders and the range of the measured motion during the interaction force experiment.

Synchronization

The synchronization of the motion capture data and the IMU data was done in the same way as for the fast motion experiment. Instead of instructing the robot to do a synchronization motion, the robot was moved by hand through the force sensor. This made it possible to also synchronize the force sensor with the internal sensors of the robot. This synchronization was again done by computing the cross-correlation of two signals. The torque measured by the force/torque sensor around its z-axis was cross-correlated to the torque measured in the seventh joint of the robot. Since the two sensors are aligned, these signals should be very similar.

5-3-3 Tuning the IEKF

The two experiments described above provide all the necessary data to validate the working of the IEKF algorithm. However, before any results are obtained, the parameters of the filter should still be determined. There are six parameters that are required to run the estimation.

First, there are four noise covariance matrices: the covariance matrix of the process noise associated with the joint torques, Q_τ, the covariance matrix of the process noise associated with the force input, Q_f, the marker measurement noise covariance matrix R_m and the IMU measurement noise covariance matrix R_{IMU}. The cross-covariance between the two sources of process noise and between the markers and the IMUs was neglected. Additionally, an estimate of the initial state $\hat{\boldsymbol{x}}_0$ and the error covariance of this estimate $\hat{P}_0$ are required. Q_τ was tuned by hand for each experiment. The diagonals of the matrices were set roughly based on the magnitudes of the ground truth torques and then the whole matrix was scaled. For the fast motion experiment, the value $Q_\tau = \mathrm{diag}(\begin{bmatrix}10^5 & 10^5 & 10^5 & 10^5 & 10^4 & 10^4\end{bmatrix})\ \mathrm{Nm}^2/\mathrm{s}^2$ was used. For the interaction force experiment, the value $Q_\tau = 5.2\,\mathrm{diag}(\begin{bmatrix}1000 & 3500 & 500 & 1000 & 70 & 150\end{bmatrix})$ $\mathrm{Nm}^2/\mathrm{s}^2$ was used. The covariance matrix of the input force noise, Q_f, was computed from the force/torque data recorded during the stationary trial. This approach was also taken initially for the markers and IMUs, but this turned out not to work very well. For example, in the case of the marker data, the covariance of the measured positions during a stationary trial is very low, because the motion capture system is very consistent. However, during the motion trials, the marker data is affected by other sources of error as well. Examples include inaccuracies in the kinematic model or marker motion relative to the robot. Effectively, this means that the assumption that the measurement is affected by only Gaussian white noise is invalid. Using only the stationary noise covariance matrix would give the algorithm false confidence in these measurements, thereby decreasing the estimation quality. The following approach was taken to circumvent this issue: the measurement error during the motion trials was computed by applying the ground truth kinematics, measured by the robot, to the OpenSim model. The model's predicted measurements from its virtual sensors were compared to the real measurements and the covariance of this error signal was calculated. The resulting covariance matrices will be referred to as $\bar{R}_m$ and $\bar{R}_{IMU}$. Unfortunately, this means that some ground truth information is encoded in the parameters of the filter. In a human subject experiment, no ground truth data is available so this method of tuning would not be applicable. In marker-based inverse kinematics (IK), measurements are sometimes weighted according to the residual error of an unweighted estimation [15]. Perhaps a similar method can be applied to the IEKF, but this was not tested in this thesis. To test if the algorithm is overly sensitive to the values of the noise covariance matrices, multiple estimation runs were carried out, where the ratios of R_m and R_{IMU} relative to Q_τ were modified by scaling $\bar{R}_m$ and $\bar{R}_{IMU}$ over multiple orders of magnitude. From (4-13) through (4-16), it can be observed that only the ratio between these matrices is relevant to the estimated state. When all covariance matrices are scaled by the same amount, the resulting estimate is identical. The matrices R_m and R_{IMU} were calculated as

$$R_{m,r} = 10^r \,\frac{Q(1,1)}{\bar{R}_m(1,1)}\,\bar{R}_m, \quad R_{IMU,s} = 10^s \,\frac{Q(1,1)}{\bar{R}_{IMU}(1,1)}\,\bar{R}_{IMU}, \tag{5-8}$$

where $Q(1,1)$ denotes the first element of the matrix Q. The values for r and s were found through trial and error. In total, 89 estimation runs were carried out for the fast motion experiment. In six runs, only marker data was used and r was varied from -11 to -6. In eleven runs, only IMU data was used and s was varied from -13 to -3. In 72 runs, different combinations of r and s were used in the estimation, with r between -11 and -6 and s between -10 and -2. For the interaction force experiment, 79 estimation runs were carried out: six with only marker data (r between -11 and -7), nine with only IMU data (s between -11 and -3) and 64 with the combined data (r between -11 and -4 and s between -10 and -3). In each run, the initial state was the same. The initial joint angles were estimated by applying IK to the first sample of the marker data. The initial joint velocities were all set to zero and the initial joint torques were set to the values needed to achieve static equilibrium in this pose, calculated using inverse dynamics (ID). The initial covariance $\hat{P}_0$ was set to a value much larger than what the marker/IMU IEKF converged to during initial testing. This was done to make sure that the algorithm is not overconfident in the initial state. $\hat{P}_0 = \text{diag}(\begin{bmatrix}10^{-3} & \dots & 10^{-3} & 10^{2} & \dots & 10^{2} & 10^{4} & \dots & 10^{4} & 10^{3} & 10^{3}\end{bmatrix})$ was used for the fast motion experiment. The last two joint torques were given a smaller value than the other four. $\hat{P}_0 = \text{diag}(\begin{bmatrix}10^{-3} & \dots & 10^{-3} & 1 & \dots & 1 & 10^{2} & \dots & 10^{2}\end{bmatrix})$ was used for the interaction force experiment. To find out if the algorithm is overly sensitive to the accuracy of the initial state, one extra estimation run was performed per experiment where the initial state was offset from the true value. The offset was determined for each of the 18 states by finding the minimum and maximum value of the ground truth state, and giving an offset of $\pm$ 20% of the range between them.

5-3-4 Comparison with state-of-the-art methods

For the sake of comparison, the joint angles, velocities, torques and powers were estimated with a number of more conventional methods from literature as well. The methods that were chosen to compare the results to were as follows:

- A marker-based IK/ID method
- An IMU-based OBIK/ID method
- A kinematic EKF, estimating joint angles, velocities and accelerations using both marker and IMU data
- An EKF with the same state space model used in the IEKF. The goal of this comparison was to determine the effect of iterating the estimated state.

A more detailed description of how these methods work is provided in Chapter 2. All methods used the same OpenSim model. Of course, the results of these methods depend on a number of parameters just like the IEKF. The tuning of each method will be described briefly now.

Marker-based IK/ID

For the IK step, OpenSim's built-in `InverseKinematicsTool` was used with the same marker data given to the IEKF. Each marker was given an equal weighting in the optimization and the resulting kinematics were low-pass filtered with a fourth-order Butterworth filter. Two variations of this filtering step were applied. First, a causal low-pass filter was applied to the kinematics, using MATLAB function `filter`. This means that the filtered data only depends on data up to the current time instance. Since the IEKF algorithm only uses data up to the current time instance as well, this provides a fair comparison. However, when pre-recorded data is analyzed, non-causal filters are frequently used, since they do not introduce delays and therefore typically provide better results. A non-causal filter was therefore also applied to the IK result using MATLAB function `filtfilt`. For both filters, the estimation accuracy depended heavily on the cut-off frequency of the Butterworth filter. In all subsequent comparisons, the cut-off frequency which provided the lowest root mean square error (RMSE) on the estimated joint torques was used. The exact value of the cut-off frequency was different for the causal and non-causal filters and for the two experiments. For the fast motion experiment, a cut-off frequency of 8.5 Hz and 3.5 Hz was used for the causal and non-causal filters respectively. For the interaction force experiment, cut-off frequencies of 9.25 Hz and 3.75 Hz were used. The causal filters had a higher cut-off frequency because this introduces less delay, but it makes them more sensitive to noise. The non-causal filters do not introduce any delay, so a lower cut-off frequency could be used. The filtered kinematics were compared to ground truth in the validation. The filtered kinematics were then used in OpenSim's `InverseDynamicsSolver` tool to compute the joint torques.

IMU-based OBIK/ID

In order to apply orientation-based inverse kinematics (OBIK), the orientation of each individual IMU should be estimated from measurement data. Conventional algorithms rely on magnetometer data for this estimation to avoid drift in the yaw angle [19]. During the experiment, the Xsens MTw Awinda Motion Trackers recorded the magnetic field. However, they were attached to a robot which uses strong electric motors to move its joints. The induced magnetic field caused by the motors of the robot was found to have a disastrous impact on the orientation estimation of the IMUs. To circumvent this issue so that a fair comparison may still be made, inspiration was taken from the work of De Kanter [34]. The ground truth kinematics were used in the OpenSim model to create 'perfect' magnetometer data for each IMU, which was subsequently used together with the gyroscope and accelerometer data to estimate its orientation. For the orientation estimation, a so-called Madgwick filter was used [55]. This filter relies on the assumption that the gravitational acceleration dominates the measured specific force, unlike the IEKF. The Madgwick filter requires the tuning of one parameter, β. By computing the orientation of the virtual IMUs from the ground truth kinematics in OpenSim, the value of β which minimized the error compared to these orientations could be selected. $\beta = 0.04$ was used for the fast motion experiment and $\beta = 0.02$ was used for the interaction force experiment. Then, OBIK was performed using OpenSim's built-in `IMUInverseKinematicsTool`. The same causal and non-causal filtering steps were applied to the estimated kinematics as for marker-based IK, with cut-off frequencies of 8.25 Hz and 2.75 for the fast motion experiment and 9 Hz and 2.75 Hz for the interaction force experiment. The filtered kinematics were again used in ID to compute the joint torques.

Kinematic EKF

A kinematic extended Kalman filter (EKF) in the generalized coordinate formulation was also tested. The state update model was defined according to a constant acceleration model, meaning that the state vector contained the joint angles, velocities and accelerations. Process noise affected only the accelerations. The resulting discrete-time state update equation reads

$$\boldsymbol{x}_{k+1} = \begin{bmatrix} \boldsymbol{q}_{k+1} \\ \dot{\boldsymbol{q}}_{k+1} \\ \ddot{\boldsymbol{q}}_{k+1} \end{bmatrix} = \begin{bmatrix} \mathcal{I} & \Delta t\, \mathcal{I} & \frac{1}{2}\Delta t^2\, \mathcal{I} \\ \mathbf{0} & \mathcal{I} & \Delta t\, \mathcal{I} \\ \mathbf{0} & \mathbf{0} & \mathcal{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{q}_k \\ \dot{\boldsymbol{q}}_k \\ \ddot{\boldsymbol{q}}_k \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\Delta t^2\, \mathcal{I} \\ \Delta t\, \mathcal{I} \\ \mathcal{I} \end{bmatrix} \boldsymbol{v}_k, \quad \boldsymbol{v} \sim \mathcal{N}(0, Q). \tag{5-9}$$

The measurement model for this EKF included both the marker and IMU measurements and is given by the function $g(\boldsymbol{x}, \boldsymbol{w})$ defined in (3-31). The kinematic EKF was built using the same tools as the IEKF algorithm, described in Section 4-2. This means both the OpenSim model and sensor models are identical. The only difference is the definition of the state variables and their dynamics. This EKF does not process measured interaction forces, so the Q matrix only contains the covariance of the generalized acceleration random walk model. The values for R_m and R_{IMU} were taken to be the same as in the best performing IEKF. By 'best performing IEKF', the IEKF with the lowest overall joint torque RMSE is meant. The value for Q was then tuned by hand to achieve optimal performance. The joint angles, velocities and accelerations estimated by the EKF were used in ID to estimate the torques.

Marker/IMU EKF

The state space model used in the marker/IMU IEKF was used in an EKF as well, to determine the effect of iterating the state estimate. The noise covariance matrices of the best performing IEKF were also used for the EKF, only the maximum number of iterations was changed from five to one. In the next chapter, the results of all previously described estimation runs will be shown and discussed.

Chapter 6

Results

The results from each experiment described in Chapter 5 will be presented one by one. The presentation of the validation results is further subdivided into results that provide insight about the workings of the iterated extended Kalman filter (IEKF) algorithm and a comparison with several conventional methods from literature. The results of the system identification algorithm will be presented first, since the accuracy of the identified model has some implications for the accuracy of the IEKF as well.

6-1 System Identification of the KUKA iiwa 7

	Mass [kg]	Mass moment of inertia [10^{-3} kg $\cdot$ m^2]					
	m	I_{xx}	I_{xy}	I_{xz}	I_{yy}	I_{yz}	I_{zz}
Segment 1	4.60	1.54	0.00	0.00	1.53	0.00	0.12
Segment 2	3.50	54.59	43.31	3.35	35.05	-4.19	88.87
Segment 3	5.54	25.16	-2.64	-10.33	28.12	-6.03	6.15
Segment 4	2.51	0.11	-0.08	-0.01	0.42	0.01	0.44
Segment 5	1.43	0.48	0.05	-0.07	0.19	0.01	0.46
Segment 6	3.23	2.49	-0.04	-0.25	2.45	-0.36	0.10
Segment 7	0.82	1.51	-0.02	0.25	1.54	0.12	0.05

Table 6-1: The optimized values for the mass and moments of inertia for each segment of the robot.

The results of the optimization procedure from Section 4-1 are summarized in Table 6-1. The new values for the masses and moments of inertia were prescribed to the OpenSim model and the new model was used to perform inverse dynamics (ID) using the measured robot kinematics. This resulted in a time series of computed torques, which were compared to the torques measured by the robot. The same procedure was done for the old model from [40].

This comparison is shown in Figure 6-1 for joint three. For a plot of the torques in the other joints, the reader is referred to Appendix C. This figure shows that the optimized model results in a lower root mean square error (RMSE) for the ID torques than the old model. The RMSE for the other joints is given in table 6-2. The standard deviation (SD) of the measured torque signal is also shown to give a sense of the size of the error relative to the magnitude of the signal. The optimized model showed a decrease in RMSE for each joint, however the effect was more pronounced for some joints than for others. The decrease in RMSE, as a percentage of the RMSE of the unoptimized model, is also given in table 6-2. For joint seven, the decrease in RMSE was minimal. This can be explained by the fact that the mass and mass moment of inertia of the end effector are very small compared to the other segments. The signal-to-noise ratio of the torque sensor in the seventh joint is therefore much lower.

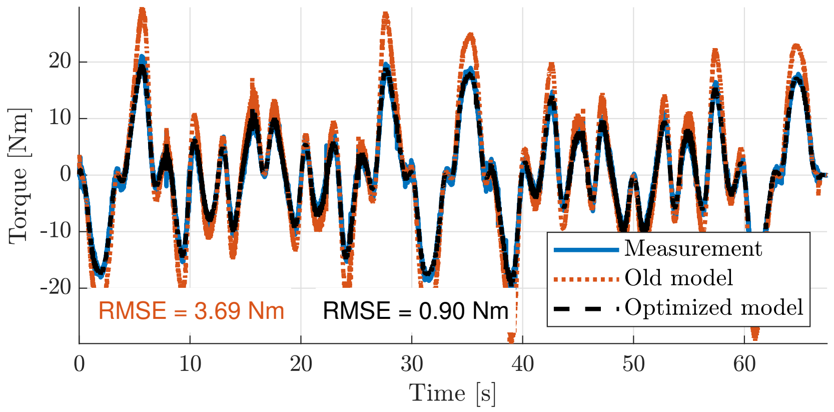


Figure 6-1: ID results of the old robot model and the optimized model for joint three, compared to ground truth torque of the optimization dataset.

Joint	1	2	3	4	5	6	7
SD of ground truth torque [Nm]	3.16	46.70	8.68	16.24	0.81	0.49	0.11
RMSE (old model) [Nm]	1.90	10.35	3.69	6.60	1.52	0.97	0.11
RMSE (optimized model) [Nm]	1.32	2.37	0.90	1.13	0.46	0.31	0.11
Percentage improvement	30%	77%	76%	83%	70%	69%	3%

Table 6-2: RMSE of the ID torques for the old model and the optimized model. The SD of the torque for each joint is shown for reference.

To make sure that the new model was not overfitted to the dataset used in the optimization, ID was performed for both the old and the new models on the dataset from the fast motion experiment described in Section 5-3-1 as well. The results for this validation test are shown in Table 6-3 and Figure 6-2. Figure 6-2 again shows the torque measured in joint three. A similar level of improvement compared to the old model was observed for the validation dataset, indicating that overfitting is unlikely. Again, a minimal improvement was observed in joint seven. As was explained in Section 5-3, in the fast motion experiment the seventh joint was locked at 0° so the signal-to-noise ratio of this joint was even lower in this trial.

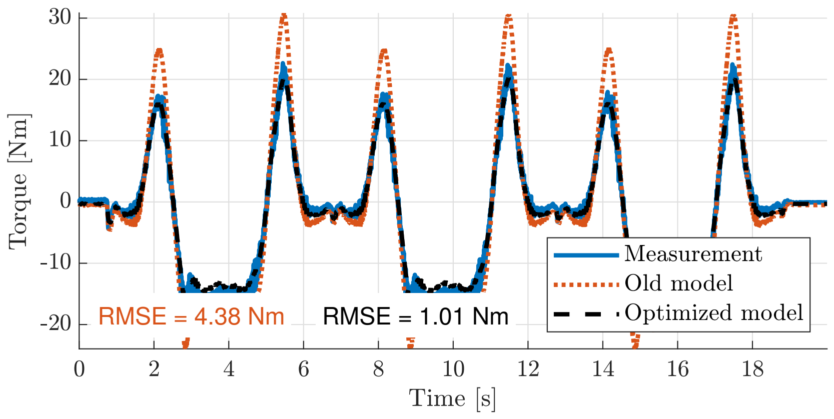


Figure 6-2: ID results of the old robot model and the optimized model for joint three, compared to ground truth torque of the validation dataset.

Joint	1	2	3	4	5	6	7
SD of ground truth torque [Nm]	10.14	43.84	10.29	10.80	1.06	0.50	0.03
RMSE (old model) [Nm]	4.08	10.79	4.38	6.40	1.98	0.99	0.03
RMSE (optimized model) [Nm]	1.71	3.37	1.01	1.26	0.33	0.34	0.03
Percentage improvement	58%	69%	77%	80%	83%	66%	1%

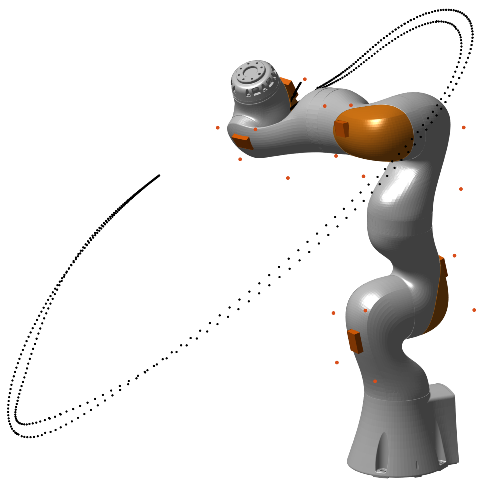
Table 6-3: RMSE of the ID torques for the old model and the optimized model on the validation dataset. The SD of the torque for each joint is shown for reference.

6-2 Experimental validation of the IEKF

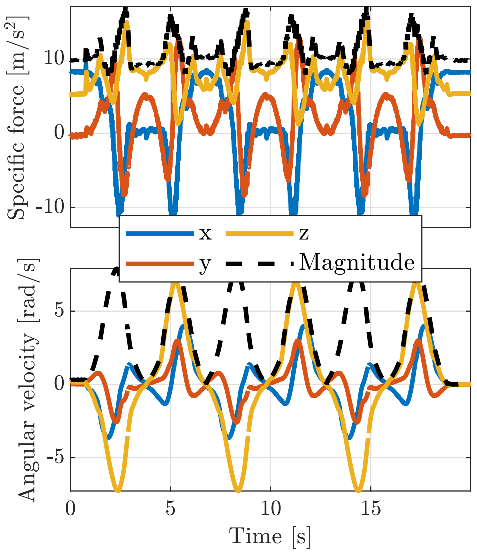
The identified OpenSim model was used in the IEKF algorithm to process the data gathered during the fast motion experiment and the interaction force experiment. For the interaction force experiment, the model was modified with the virtual force sensor described in Section 5-1-4. The accuracy of all estimated quantities (joint angles, velocities and torques) was determined in comparison to the ground truth data. Additionally, the estimated joint power was compared to ground truth. The joint power was calculated simply as the product of the joint torque and joint velocity, so it is not actually an independent estimated variable. However, since joint power is a quantity of interest in many biomechanics studies, it was included in the results. The previous section has shown that the identified dynamic model, while it improves ID results considerably, does not perfectly describe the measured joint torques even when supplied with the ground truth kinematics. This indicates that there is a limit in the estimation accuracy that can be achieved using this dynamic model. The joint torques and powers were also calculated by applying ID to the measured kinematics. Whenever joint torque or power is compared to the ground truth data, this result is also shown to indicate the lower bound determined by the model.

6-2-1 Fast motion experiment

As an example of the data on which the estimations were based, the measured 3D trajectory of one of the markers and the data measured by one of the IMUs is shown in Figure 6-3. From



(a) The measured positions of one of the markers attached to segment six of the robot.



(b) The data measured by the IMU attached to segment six of the robot.

Figure 6-3: Some of the sensor data measured in the fast motion experiment.

Figure 6-3b we can observe that the motion performed by the robot induced a specific force significantly different from the gravitational acceleration of 9.81 m/s^2. The assumption that the measured specific force is dominated by gravity, on which the tested orientation-based inverse kinematics (OBIK) method relies, is violated. As will become clear in this section, this affects the estimation accuracy of this method.

Sensitivity to tuning

The 89 estimation runs described in Section 5-3-3 were evaluated on their accuracy in estimating joint angles, velocities, torques and powers. The RMSE was computed for each quantity and for each joint separately, as well as for all joints combined. The combined RMSE of the joint angles for all runs is shown in Figure 6-4. The RMSE is plotted against the ratio of R_{IMU} to Q_τ and R_m to Q_τ, which was varied according to (5-8). This ratio was only computed from the first element of each matrix. For example, the ratio of R_m to Q_τ is computed from $R_m(1,1)$, the variance of the noise acting on the x-coordinate of the first marker and $Q_\tau(1,1)$, the joint torque process noise of joint one. The absolute value of this ratio does not provide much insight to the physical system. In relation to other instances of the IEKF, it is informative because it shows how much the IEKF relies on either sensor modality. When viewing this figure, it should be kept in mind that a *larger* value of R_m indicates that the IEKF has *lower* confidence in the marker measurements. When R is large compared to Q, more confidence will be placed in prediction from the state update function. Since the values for R_m, R_{IMU} and the RMSE level are in a range of more than one order of magnitude, all axes are scaled logarithmically. Since marker-based inverse kinematics (IK) is currently considered the gold standard in literature, the level of accuracy achieved with the two variations of marker-based IK described in Section 5-3-4 are also shown for reference.

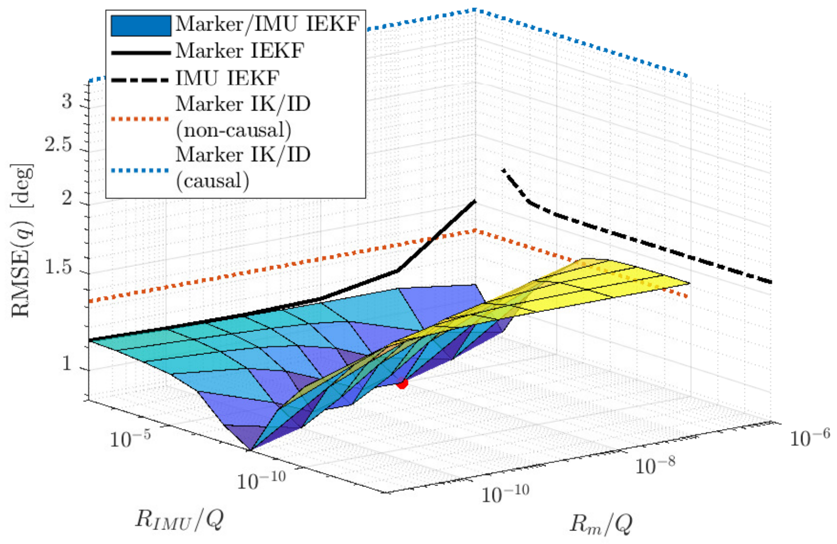


Figure 6-4: The joint angle RMSE of all IEKF estimation runs for the fast motion experiment. The RMSE is plotted against the ratios of the noise matrices of the filter. The filters that had access to only marker data or only IMU data are shown as solid and dashed lines respectively. The filters that had access to both types of data are shown as a surface. Since the filter with only marker data has no R_{IMU} matrix, the solid line is placed arbitrarily to be aligned with the surface plot and vice versa for the filter with only IMU data. Blue and red dotted lines indicate the level of accuracy achieved by causally and non-causally filtered IK data respectively. The red dot indicates the location of the lowest RMSE value.

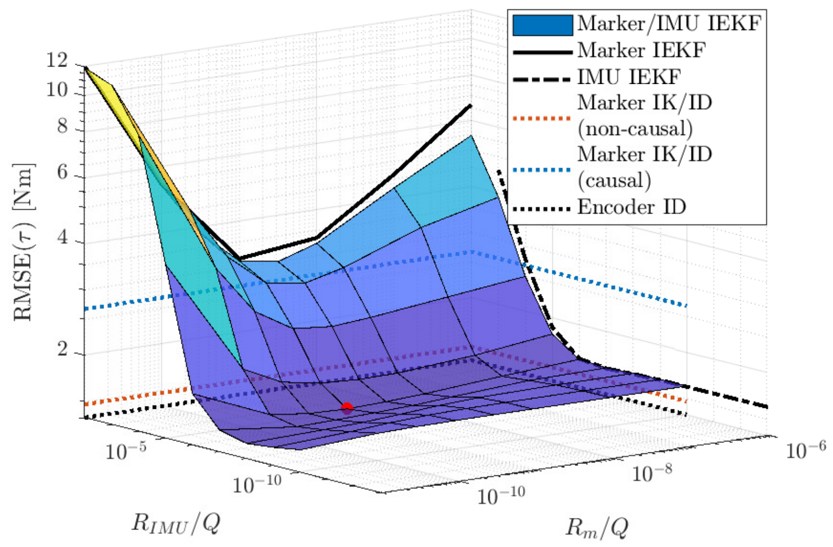


Figure 6-5: The joint torque RMSE of all IEKF estimation runs for the fast motion experiment. The RMSE is plotted against the ratios of the noise matrices of the filter. Blue and red dotted lines indicate the level of accuracy achieved by causally and non-causally filtered IK/ID data respectively. The accuracy of ID performed with ground truth kinematics is indicated with a black dotted line. The red dot indicates the location of the lowest RMSE value.

Four main observations can be made by looking at Figure 6-4. First, all IEKF methods outperform the causally filtered IK result. Using only IMU data, nearly the same level of accuracy as the non-causally filtered IK result can be obtained. Using only marker data in the IEKF, accurate kinematics can be estimated surpassing the level of accuracy of all IK methods. Neither IEKF is very sensitive to the tuning of its measurement noise covariance matrix. Both IEKFs perform best when the value of R_m or R_{IMU} is relatively low. This indicates that the prediction step is not really needed to estimate the kinematics and that the measurements contain all necessary information to estimate the pose of the robot. Finally, the best result is obtained when a particular combination of IMU data and marker data is used, bringing the RMSE below the level of $1°$. This indicates that the IMU data contains some additional information the IEKF can use to estimate the pose of the robot.

A similar plot was created showing the RMSE of the estimated joint torques for all estimation runs. This plot is shown in Figure 6-5. In this figure, the result obtained from ID with ground truth kinematics is also shown to indicate the lower bound imposed by the dynamic model. From Figure 6-5 the following observations were made: using only marker data, the IEKF was not able to estimate joint torques more accurately than either of the IK/ID methods. Like the causal IK method, the marker-based IEKF introduces a delay which negatively affects the estimation. Using only IMU data, the joint torques were estimated much more accurately than in the marker-based IEKF. The accuracy of the IMU IEKF approached the level of accuracy obtained with non-causal marker-based IK/ID. While the IEKF is a causal method, this was possible because the specific force and angular velocity measurements can be propagated to the estimated joint torques without delay. The very best performance was again achieved by combining marker data with IMU data. In the combined IEKF, the accurate kinematics provided by the marker data are fused with the direct specific force and angular velocity measurements from the IMU data for an optimal and delay-free result. When looking at the sensitivity of the IEKF, a different trend from Figure 6-4 is observed. The marker IEKF is quite sensitive to the tuning parameters, whereas the IMU IEKF is able to estimate accurate joint torques for a range of R_{IMU} values. A minimal RMSE is achieved at $R_{IMU}/Q = 10^{-7}$, but decreasing the ratio from this optimal point does not decrease the estimation quality dramatically. This indicates that the IMU measurements contain enough information to estimate torques without relying much on the process model. A surface plot of the other estimated quantities (joint velocity and power) was also generated, but not shown here for the sake of compactness. These plots can be found in Appendix C. In the remainder of this section, the discussion will focus on the results obtained only from the IEKFs with the lowest joint torque RMSE.

Comparison with state-of-the-art methods

From these IEKFs, the joint angle RMSE and joint torque RMSE for each individual joint, as well as for the overall system, are compared to all methods described in Section 5-3-4 in Tables 6-4 and 6-5 respectively. The comparisons for velocity and power are again omitted for brevity, but these can be found in Appendix C. In Table 6-4, it is quite clear which method has the lowest accuracy in estimating joint angles: the causal OBIK method is consistently outperformed by the other methods. It is likely negatively affected by its inaccurate assumption that the linear acceleration of the inertial measurement units (IMUs) is negligible compared to the gravitational acceleration. Out of all IK and OBIK methods, the non-causal marker-based IK method is the only one that is comparable in accuracy to the Kalman filter-based methods, even though these methods are causal and therefore have to work with less information. It is less clear which method is most preferable though. The marker/IMU extended Kalman filter (EKF) and IEKF perform essentially the same and have the lowest overall RMSE, but the differences with the kinematic EKF are small. The results are close especially when comparing the torque estimates in Table 6-5. To summarize Tables 6-4 and 6-5, the overall RMSE of each method is shown in comparison to the IEKF results in Figure 6-6. In this figure, all estimated variables (i.e. joint angles, velocities, torques and powers) were evaluated.

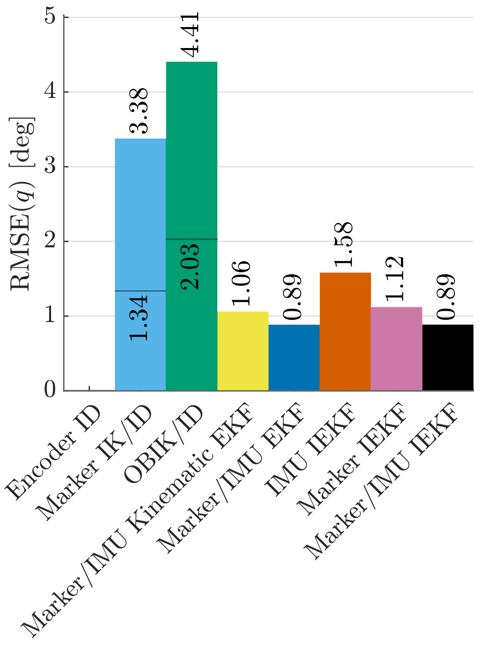
Joint angle RMSE [deg]							
Estimation method	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	Overall
Marker IK/ID (non-causal)	1.00	0.82	2.35	1.23	1.14	0.86	1.34
Marker IK/ID (causal)	2.80	2.40	3.55	3.60	3.05	4.48	3.38
OBIK/ID (non-causal)	0.46	1.88	2.31	2.72	2.16	1.88	2.03
OBIK/ID (causal)	2.61	3.95	3.70	5.10	4.52	5.82	4.41
Kinematic EKF	0.26	0.50	0.64	1.26	0.75	1.97	1.06
Marker/IMU EKF	0.28	0.59	0.66	1.14	0.41	1.54	0.89
IMU IEKF	1.81	1.11	1.59	1.35	1.50	1.98	1.58
Marker IEKF	0.40	0.68	0.93	1.07	1.10	1.92	1.12
Marker/IMU IEKF	0.28	0.58	0.66	1.14	0.41	1.55	0.89

Table 6-4: The RMSE of the estimated joint angles in the fast motion experiment, for each estimation method. The highest and lowest RMSE values are indicated in red and green respectively.

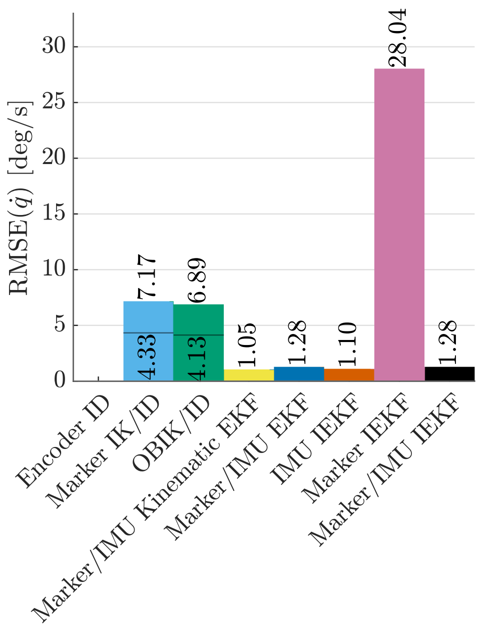
Joint torque RMSE [Nm]							
Estimation method	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	Overall
Encoder ID	1.49	2.51	0.97	1.17	0.32	0.33	1.36
Marker IK/ID (non-causal)	1.39	2.90	0.99	1.24	0.32	0.32	1.48
Marker IK/ID (causal)	3.21	4.99	1.78	1.99	0.35	0.34	2.66
OBIK/ID (non-causal)	1.83	2.88	1.31	1.45	0.33	0.32	1.61
OBIK/ID (causal)	4.85	6.60	2.30	3.21	0.36	0.34	3.72
Kinematic EKF	1.38	2.84	1.06	1.20	0.33	0.33	1.46
Marker/IMU EKF	1.37	2.89	1.01	1.23	0.32	0.33	1.47
IMU IEKF	1.48	3.11	1.20	1.32	0.32	0.33	1.59
Marker IEKF	3.20	6.16	2.07	2.59	0.62	0.53	3.16
Marker/IMU IEKF	1.37	2.90	1.01	1.23	0.32	0.33	1.47

Table 6-5: The RMSE of the estimated joint torques in the fast motion experiment, for each estimation method. The highest and lowest RMSE values are indicated in red and green respectively.

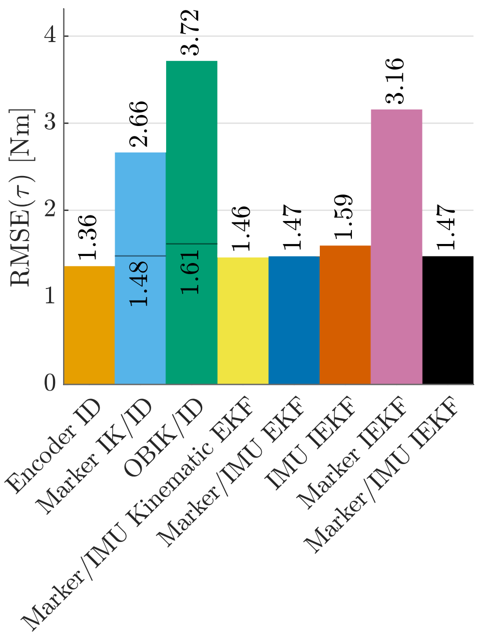
The differences between the various methods are much more pronounced when comparing the



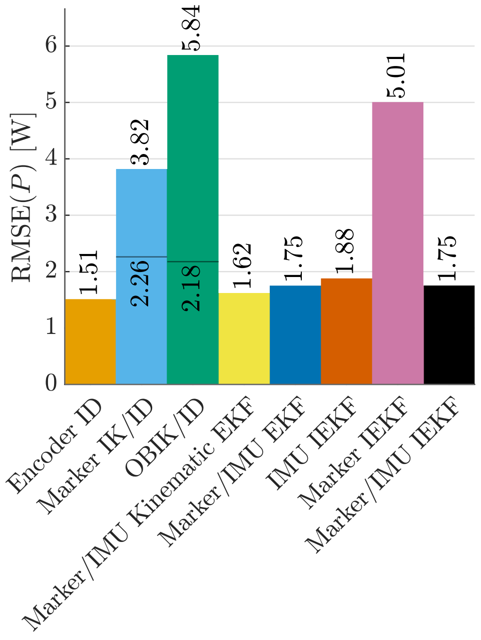
(a) Joint angle RMSE for all estimation methods. Encoder ID has zero error since this is the ground truth signal.



(b) Joint velocity RMSE for all estimation methods. Encoder ID has zero error since this is the ground truth signal.



(c) Joint torque RMSE for all estimation methods.



(d) Joint power RMSE for all estimation methods.

Figure 6-6: Comparison of the accuracy of all estimation methods in the fast motion experiment. Marker IK/ID and OBIK/ID show two values to indicate the difference between causal and non-causal filtering.

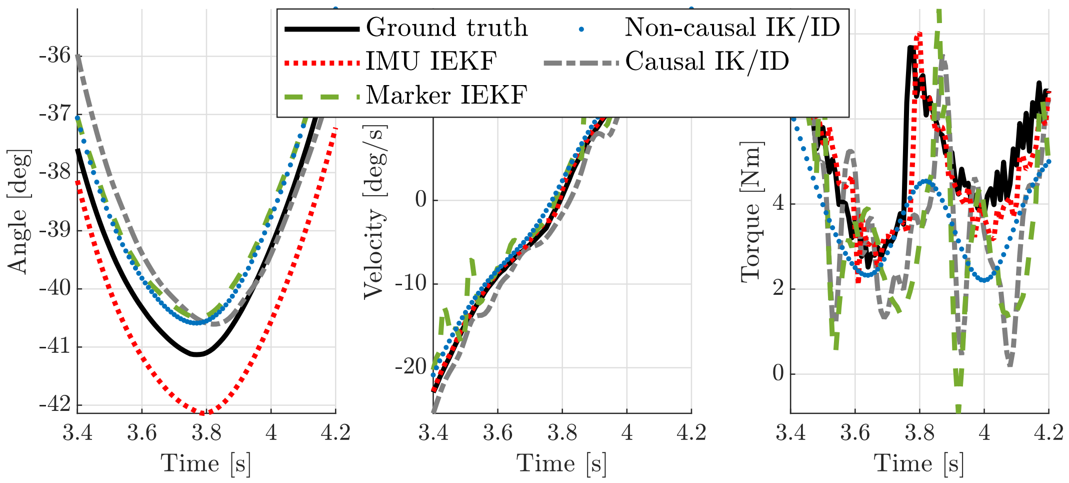


Figure 6-7: The estimated joint angle, velocity and torque for joint one during a section of the experiment where the robot changes direction. Only the IMU IEKF can capture the resulting peak in torque accurately.

velocities. All methods that use IMU-measured angular velocity directly in the estimation, like the kinematic EKF or the IMU IEKF, show accurate velocity estimation in the range of 1 deg/s RMSE. All methods that rely only on marker data or pre-estimated IMU orientations show much larger error values. Clearly, having a velocity level measurement of the kinematics is important to the accuracy of the estimation. While one might expect this trend to be even more pronounced in the joint torque RMSE levels, the results are actually much closer together on the torque level. All methods perform reasonably well, except the causally filtered IK/ID and OBIK/ID methods and the marker-based IEKF. The inaccuracy of these methods can be largely contributed to the fact that they introduce a delay in the estimation and that the torque estimates are very sensitive to noise. While a non-causal implementation of the proposed framework in the form of a nonlinear Kalman smoother was not tested in this thesis, this could help to solve some of these problems. The non-causally filtered ID methods do not introduce any delay and the IMU-based IEKF methods have access to direct acceleration measurements to correct for this. It should also be noted that an accurate estimation of joint angles does not guarantee accurate joint torque estimation: the marker-based IEKF performed quite well when estimating the joint angles, but its accuracy in estimating kinetics is among the worst. The reason that the large differences between methods observed in the velocity estimation is not observed here could be the dynamic model. The accuracy of the dynamic model limits the accuracy of all estimation methods. Many estimation methods, like the marker/IMU IEKF, but also the non-causal marker IK/ID approach the lower bound given by the encoder ID. Despite the level of torque RMSE being similar across these methods, there are notable differences in the estimated torque signals. These differences have implications for the practical applicability of these methods. One example is shown in Figure 6-7.

Figure 6-7 shows the ground truth joint angle, velocity and torque signals for the first joint of the robot. Also shown are the estimated signals from marker-based IK/ID, the marker-based IEKF and the IMU IEKF. A portion of the experiment is shown just when the velocity of the joint passes through zero, i.e. when it changes direction. Right as this happens, a sharp peak is observed in the torque signal which slowly decreases again. It is difficult to say what exactly

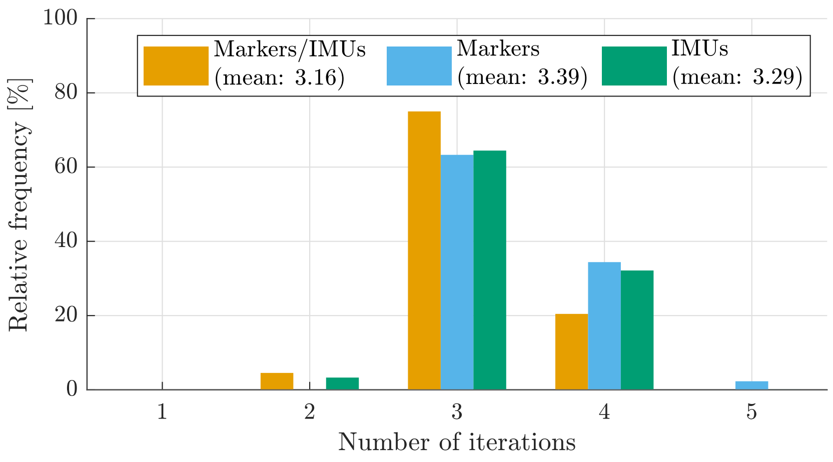
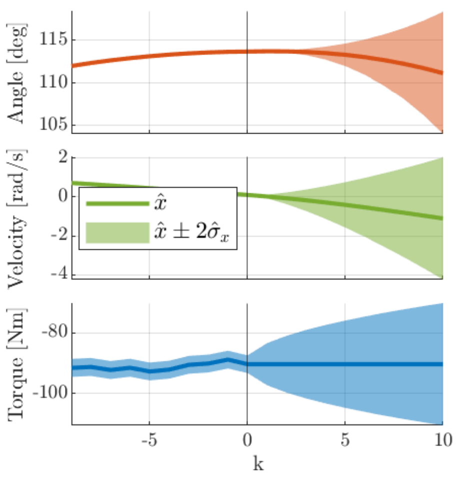


Figure 6-8: Relative frequency of the number of iterations required for convergence of the IEKF. The maximum number of iterations was set to five.

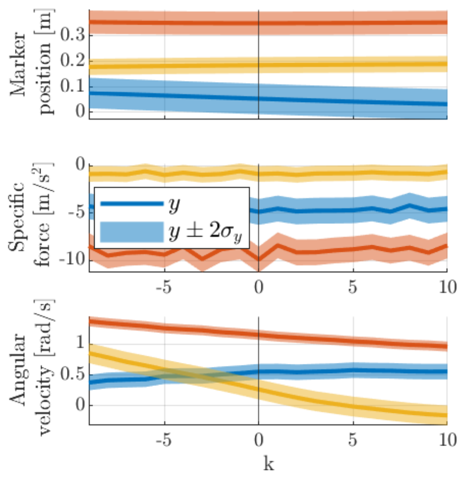
causes this peak, since it was not part of the planned trajectory. One hypothesis is that there is some play in the internal mechanism in the robot, which means two parts of the mechanism collide when changing direction. Whatever the exact cause of the peak may be, the IMU IEKF is the only estimation method that is able to capture it accurately. The non-causally filtered ID result shows a small peak, but its amplitude is underestimated and the signal is overly smooth. Furthermore, the torque signal starts rising before the measured peak occurs. This is an inherent downside of non-causal filtering, which has implications for its applicability. When the causality of an observed event is of interest in a human subject experiment, care should be taken when applying these types of low-pass filters since the non-causality of the filter can distort the order in which events occur. The causal low-pass filter does not distort this causality, however the ID result it produces is so corrupted by delay and filtering artifacts that the measured torque peak is hardly recognizable. The same applies to the marker-based IEKF. The result produced by the IMU IEKF clearly shows the sharp rise in torque and the slow descent following it. The rise in the estimated torque can be directly traced back to a peak in the specific force measured by the IMUs. This is a good demonstration of the value of specific force measurements for estimating joint torques. Together with the improvement in velocity estimation, the added benefit of IMU measurements is clearly demonstrated in these results.

Properties of the IEKF

Another interesting observation from Figure 6-6 is that the marker/IMU EKF and IEKF perform essentially the same. This indicates that the iterations of the IEKF do not significantly affect the estimation. To investigate this further, the number of iterations needed for convergence is shown in Figure 6-8. This figure shows how often the various variations of IEKF use one, two, three, four or five iterations, as a fraction of the number of time instances in the experiment. The IEKFs almost always converge within three or four iterations. This indicates that the predicted state is often quite close to the final estimated state. This could explain why the difference with the EKF is very small. The interpretation of the IEKF as a Gauss-Newton iteration process provides some more ways to gain insight into its functioning.



(a) The level of uncertainty ascribed by the IEKF to the estimated and predicted joint angle, velocity and torque for joint 2.



(b) The level of uncertainty ascribed by the IEKF to the measured position of marker 5 and the measured specific force and angular velocity of IMU 2.

Figure 6-9: Relative levels of uncertainty of the predicted state and the measurements.

One of these ways is by looking at the cost function it minimizes. This cost function, given in (4-12), features two terms. One term penalizes the deviation from the predicted state, the other penalizes the deviation from the observed measurement. To get a better understanding of the relative importance of these two terms, Figure 6-9 was created.

In Figure 6-9a, the estimated state up to the current time instance $k = 0$ is plotted for the marker/IMU IEKF. Only the joint angle, velocity and torque of a single joint are shown in this figure. Around the state estimate, a band of width $\pm 2\hat{\sigma}_x$ is shown, approximately corresponding to the 95% confidence interval. The standard deviation $\hat{\sigma}_x$ was computed by taking the square root of the diagonal elements of $\hat{P}_{k|k}$ at time instances $k = -10, -9, ...0$. The state space model was also used to make predictions of the states at $k = 1, 2, ..., 10$ by calculating the state dynamics under the constant torque assumption. The uncertainty was propagated according to (4-10). The uncertainty of the estimated torque is relatively large and increases at an approximately constant rate, in correspondence with the random walk model. The uncertainties of the estimated angle and velocity are comparatively low, but increase at a more rapid rate. Of course, only the prediction for $k = 1$ is used in the estimation, after which the measurement update is used to reduce the uncertainty of the state estimate. The measurements and corresponding uncertainties of the sensors are shown in Figure 6-9b. It should be noted that only the relative level of uncertainty is important to the estimation. The error covariance estimated by the IEKF underrepresented the actual level of error present in the estimated state. This is likely due to sources of error not accounted for in the state space model, like errors in the dynamic model. Since the IEKF can not estimate the level of uncertainty introduced by these errors, care should be taken when interpreting the covariance matrices that it estimates. Looking at Figure 6-9, it becomes apparent that the IEKF does not have a lot of confidence in the predicted joint torque, whereas the predicted joint angle is assumed to be quite accurate. This means the cost function is more sensitive to the joint angle states.

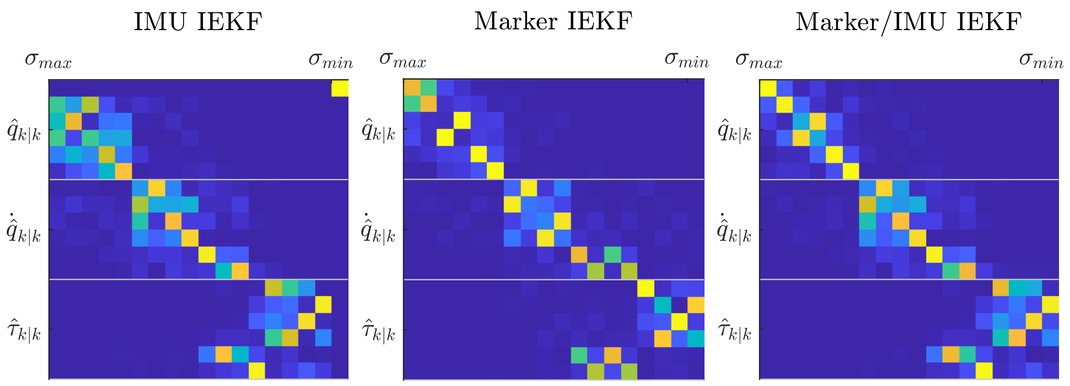


Figure 6-10: Singular vectors of the Hessian of the IEKF's cost function, ordered by singular value, for the three varieties of IEKF.

The sensitivity of the cost function is another way to compare the three varieties of IEKF developed in this thesis. This was done as follows: the Hessian matrix of the cost function from (4-12) was approximated for one time instance of the estimation at the local minimum that the IEKF converged to. The singular values of this matrix determine the amount with which the cost function increases when the state is perturbed in the direction of the associated singular vector. The singular vectors are shown, ordered by the magnitude of their singular value, in Figure 6-10. The figure is colored according to the magnitude of each element in the singular vectors. The most obvious takeaway from Figure 6-10 is that the IMU IEKF's cost function is least sensitive to the first joint angle. This makes intuitive sense: since the magnetometer measurements of the IMUs were not used in the estimation, the absolute heading of the system cannot reliably be estimated from only IMU data. Because the first joint of the robot is oriented in the horizontal plane, it controls only the heading and can therefore not be uniquely determined by the IEKF. Its velocity can be observed, so the motion relative to the initial pose can be estimated by integrating the joint velocity. This introduces some drift to the estimation. On top of $0.9°$ error in the initial state, over the 20 second duration of the experiment, an error of $0.7°$ accumulated. Figure 6-10 also shows that adding markers to the estimation fixes this issue. Another observation is that the cost function for all three IEKFs is generally the most sensitive to the joint angles. The distinction between joint velocities and torques is less clear. Finally, the sensitivity of the marker/IMU IEKF shows a resemblance to both the IMU IEKF and the marker IEKF. The singular vectors relating to joint angles look more like those found in the marker IEKF, whereas the ones relating to joint velocities and torques bear more resemblance to the ones found in the IMU IEKF. This could be an indication that the combined IEKF uses the different types of measurements to estimate these different quantities.

Practical applicability of the IEKF

In Section 5-3-3, it was explained how the IEKF was tuned and how the initial state was set. Figure 6-4 and Figure 6-5 show that, when the tuning is optimal, a significant improvement with respect to the state-of-the-art can be achieved. Of course, in a practical application, the ground truth is not known. This raises two questions:

- How can one know what the optimal tuning of the IEKF is without knowing the true state?
- What happens when the initial state given to the IEKF is incorrect?

With regards to the first question, the following hypotheses were posed: the optimal filter, in terms of joint angle and joint torque RMSE, can better predict the measured sensor signals. Additionally, when the optimal tuning for the IMU-based IEKF is combined with the optimal tuning for a marker-based IEKF, the optimal combined estimation is achieved. These hypotheses were tested by feeding the estimated kinematics and kinetics from the marker-based and IMU-based IEKFs to the OpenSim model and using the model to predict sensor measurements. The difference between the predicted and measured sensor signals is the *reprojection error*. The reprojected and real measurements were compared for marker position, IMU specific force and IMU angular velocity separately. For each filter that was tested, the joint angle RMSE and joint torque RMSE is plotted against the reprojection error in Figure 6-11 and Figure 6-12. A least-squares linear fit to this data and the Pearson correlation coefficient ρ are also shown. Figure 6-11 shows the reprojection errors for the marker-based IEKFs. From this figure, we may interpret that the IMU reprojection errors are both strongly correlated to the joint torque error. The marker reprojection error also shows a very strong correlation with the joint angle error, but this could also be due to one outlier affecting the calculation. The other correlations in Figure 6-11 are quite weak, which supports the argument made previously that accurate joint angle estimation does not guarantee a good reconstruction of the joint torques. Figure 6-12 gives the data for the IMU-based IEKFs. Similar to Figure 6-11, there are strong correlations between the IMU measurement reprojection errors and the joint torque RMSE, whereas the marker reprojection error correlates strongly with the joint angle RMSE. However, Figure 6-12 is also quite sensitive to outliers. Because the accuracy of the estimation was quite constant throughout most trials, the two trials with lower accuracy disproportionally affect the calculation. One notable observation is that, while the correlation coefficients shown in these two figures are quite similar, the scale of the reprojection errors is very different. The marker-based IEKFs all show overall marker reprojection errors in the range of 1.3 to 1.5 centimeters, whereas the IMU-based IEKFs all achieve a reprojection error of around 2.5 centimeters. Conversely, the IMU reprojection errors are in the range of 2 to 12 m/s^2 and 0.4 to 1.2 rad/s for the marker-based filters, compared to maximum errors of 1.3 m/s^2 and 0.13 rad/s respectively for the IMU-based filters. The two methods have different strengths and weaknesses, which is why the combined IEKFs show the best performance in Figure 6-6.

In the hypothetical case where the filters were tuned to achieve a minimal marker reprojection error, the filters with $R_m/Q = 10^{-9}$ and $R_{IMU}/Q = 10^{-8}$ would be selected as optimal. Using these values in the combined IEKF would have resulted in a joint angle RMSE of 1.24° and a joint torque RMSE of 1.55 Nm. Tuning based on specific force reprojection would result in $R_m/Q = 10^{-8}$ and $R_{IMU}/Q = 10^{-10}$, giving a joint angle RMSE of 1.42° and joint torque RMSE of 1.62 Nm. In reality, the optimal combined IEKF has $R_m/Q = 10^{-9}$ and $R_{IMU}/Q = 10^{-6}$ and achieves 0.89° and 1.47 Nm RMSE levels. It can be concluded that tuning the IEKF this way does not necessarily result in optimal performance. It is more reasonable to say that this method could be used to find a good starting point, from which the IEKF should be tuned further. Tuning based on sensor reprojection also carries with it the risk of overfitting the estimation to the sensor data. To circumvent this, the IMU-based

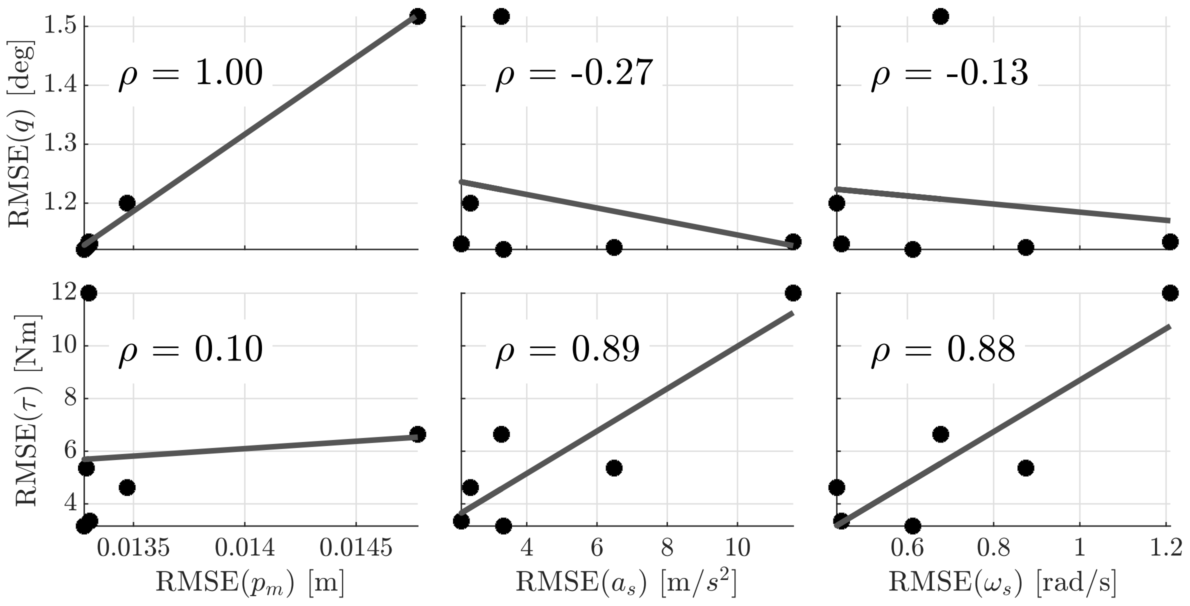


Figure 6-11: Reprojection errors of the marker-based IEKFs. The RMSE of the joint angles and torques is plotted against the reprojection error of the marker position, IMU specific force and IMU angular velocity. Each plot also shows the least-squares linear fit and the Pearson correlation coefficient.

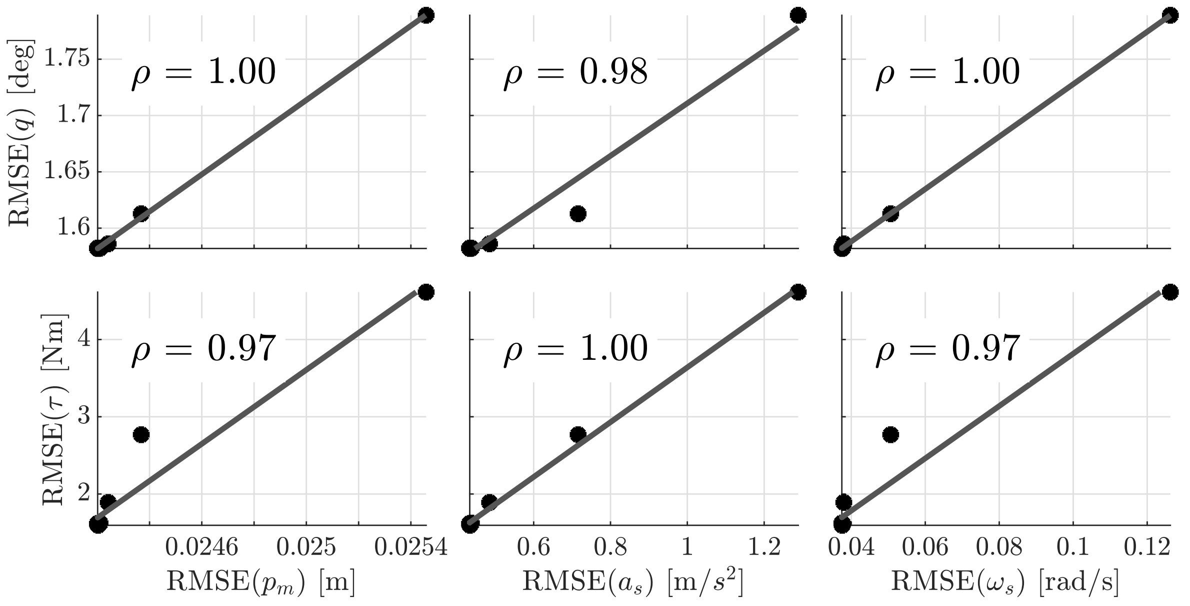


Figure 6-12: Reprojection errors of the IMU-based IEKFs. The RMSE of the joint angles and torques is plotted against the reprojection error of the marker position, IMU specific force and IMU angular velocity. Each plot also shows the least-squares linear fit and the Pearson correlation coefficient.

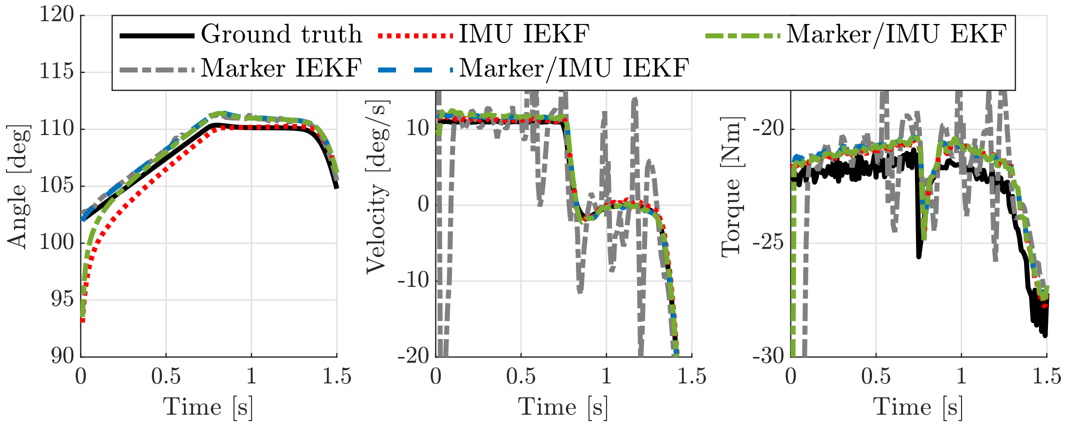
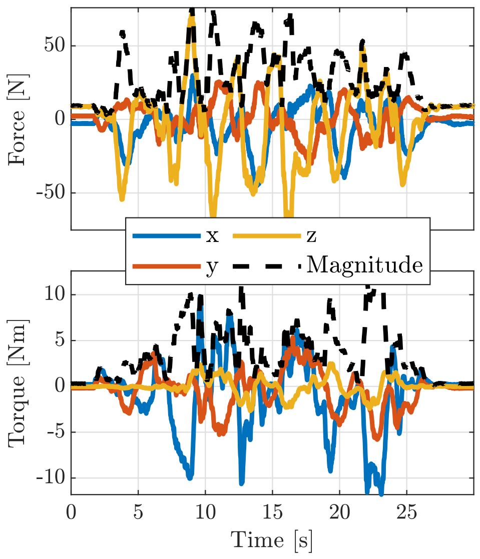


Figure 6-13: The estimated joint angle, velocity and torque for joint two during the first three seconds of the estimation runs with an inaccurate initial state.

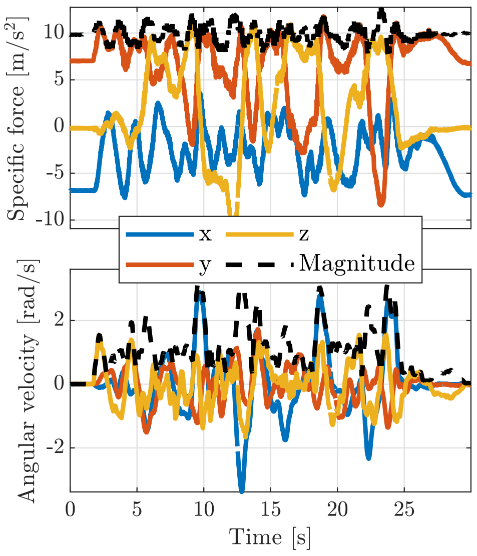
filter could be tuned to minimize marker reprojection error. The marker filter could then be tuned according to the IMU data. Another option, which does not require two measurement systems, would be to include an extra marker or IMU that is not used for the estimation. This validation sensor can then be used to tune the IEKF.

In order to answer the second question, an estimation run was performed for each type of IEKF and for the EKF where each element of the initial state was offset by $\pm$ 20% of the range of motion of that element throughout the experiment. The results are shown for the first 1.5 seconds and for the fourth joint in Figure 6-13. After the first 1.5 seconds, all filters had converged to near the ground truth state and followed roughly the same trajectory as in the case of an accurate initial state. The joint angles of the marker-based IEKF converge very rapidly. The correction of the angles then propagates to the velocities and torques, which overshoot when the filter attempts to balance the system dynamics. The filter that had access to both IMU and marker data used the IMU data to correct the inaccurate velocities and torques in the first time step, so it does not show an overshoot. The EKF, which has the same state-space model and filter parameters, takes longer to converge. This can be explained by the fact that the EKF is only allowed to make one Gauss-Newton iteration towards the true state at each time step. While the estimation accuracy of the IEKF and EKF are virtually the same for the trials with an accurate initial state, the IEKF has an advantage here because it can move to the true state multiple times at each time step. Interestingly, the IMU-based IEKF takes longest to converge, about one second. Because the IMUs' magnetometer data is not used, it could be the case that there is not enough information to uniquely determine the pose of the robot at the beginning of the experiment. The filter can only use the gravity component of the specific force to estimate the pose, but the specific force measurement is also affected by the joint torques. It seems that the IEKF has some trouble separating these effects. As was shown in Figure 6-10, the IMU-based filter cannot reliably estimate the heading of the robot, so the estimate for first joint angle does not converge to the true state at all. Instead, it remains at an approximately constant offset from the ground truth for the duration of the experiment. This has implications for the use of IMUs in human subject experiments when no motion capture data is available. It might be necessary to ask the subject to perform some calibration motions to make sure that the IEKF converges to the true pose and that the accuracy during the motion of interest is as high as possible.

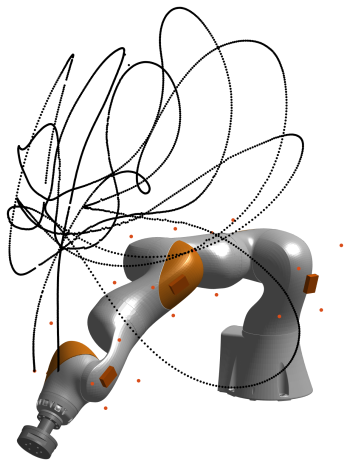
6-2-2 Interaction force experiment



(a) The data measured by the force sensor attached to the end effector of the robot.



(b) The data measured by the IMU attached to segment six of the robot.



(c) The measured positions of one of the markers attached to segment six of the robot.

Figure 6-14: Some of the sensor data measured in the interaction force experiment.

The sensitivity to the tuning parameters and the comparison to state-of-the-art methods was also investigated for the data from the interaction force experiment. Some of the data gathered in this experiment is visualized in Figure 6-14. From Figure 6-14b, it is obvious that the motion in this experiment was slower than that of the fast motion experiment. The magnitude of the measured specific force stays much closer to its static value of 9.81 and the magnitude of the angular velocity is also lower compared to Figure 6-3b. Figure 6-14c shows that the robot moved through many unique poses during the experiment.

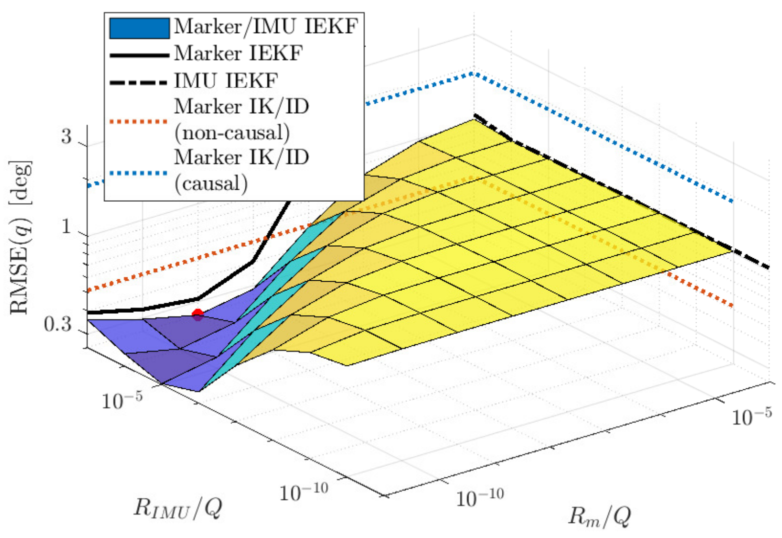


Figure 6-15: The joint angle RMSE of all IEKF estimation runs for the interaction force experiment. The RMSE is plotted against the ratios of the noise matrices of the filter. Blue and red dotted lines indicate the level of accuracy achieved by causally and non-causally filtered IK data respectively. The red dot indicates the location of the lowest RMSE value.

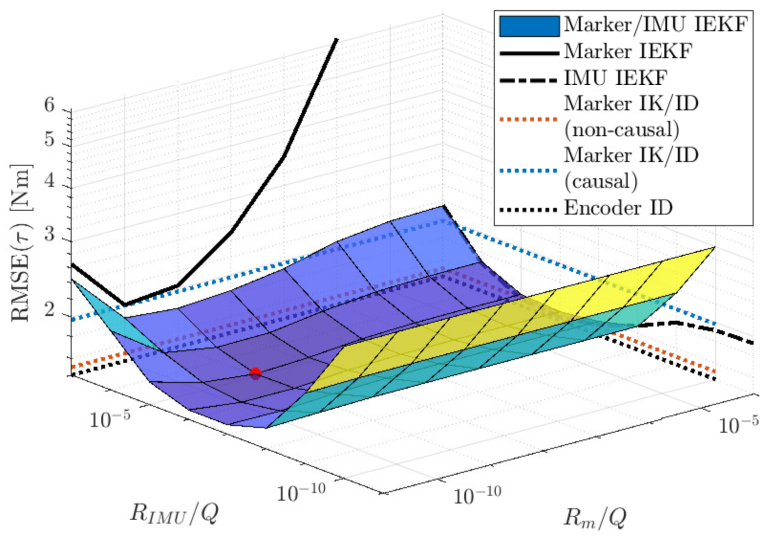


Figure 6-16: The joint torque RMSE of all IEKF estimation runs for the interaction force experiment. The RMSE is plotted against the ratios of the noise matrices of the filter. Blue and red dotted lines indicate the level of accuracy achieved by causally and non-causally filtered IK/ID data respectively. The accuracy of ID performed with ground truth kinematics is indicated with a black dotted line. The red dot indicates the location of the lowest RMSE value. Not shown in this figure is the relative noise covariance of the force sensor.

Sensitivity to tuning

The joint angle and joint torque RMSE values for all 79 estimation runs are shown in Figure 6-15 and Figure 6-16 respectively. Similar plots for joint velocity and joint power can be found in Appendix C. For the joint angles, the results seem similar to those obtained in the fast motion experiment. The IMU-based filters achieve an accuracy somewhere between the causal and non-causal marker IK methods. Adding marker data improves these results. A plateau where marker data dominates the estimation, like in Figure 6-4, is not visible in this figure. One could imagine that it exists for higher levels of R_{IMU}/Q, but this was not tested. The quality of the joint torque estimates of the IMU IEKF shows a much clearer optimum than in the fast motion experiment. When comparing the joint torque estimates of the IMU-based IEKFs to the combined estimation, it might look as if the quality sometimes decreases when marker data is added. This is in fact caused by a third factor, which is not visualized in this plot. In the analysis of the IMU-based IEKFs, the noise covariance of the force sensor Q_f was kept in a constant ratio with respect to R_{IMU}. In the analysis of the combined filter, the ratio of Q_f to Q_τ was kept constant instead. Clearly, this greatly influences the quality of the estimation. Unfortunately, due to the computation time required for these analyses, this factor was not explored further. Another difference, compared to Figure 6-4 and Figure 6-5, is that there is not one set of parameters which is optimal for both joint angle and joint torque estimation. In a practical sense, this means that the tuning of the filter will be a compromise between these two quantities. For further analysis, the set of parameters resulting in the lowest joint torque RMSE was selected. This should be kept in mind when viewing Tables 6-6 and 6-7 and Figure 6-17.

Comparison with state-of-the-art methods

Table 6-6 shows that the causal OBIK/ID method again has the lowest joint angle accuracy. The marker-based IEKF performs the best in this regard. It should be noted that the combined IEKF can improve on this result, as is visible in Figure 6-4, but to provide a consistent and fair comparison only the previously selected combined IEKF was used for this comparison. In this IEKF, the joint angle accuracy of was compromised for better joint torque accuracy. Table 6-7 shows that the joint angle accuracy of the marker-based IEKF does not translate to good joint torque estimation. The combined marker/IMU IEKF performs best overall, but again several methods are very close together here. The results are summarized in Figure 6-17, also for the joint velocity and power.

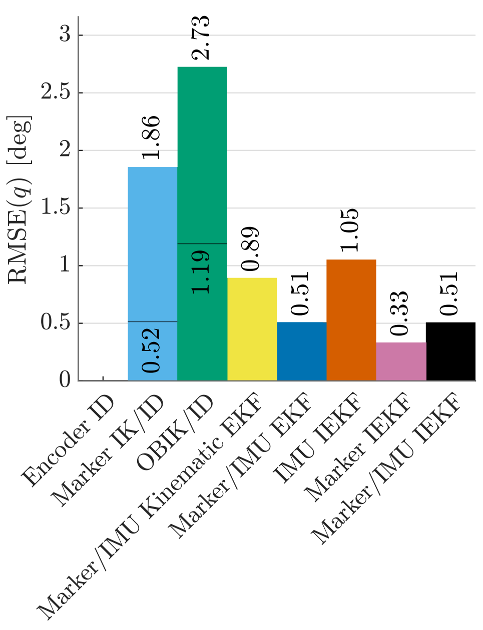
As was the case in the fast motion experiment, only the methods that use IMU data directly are able to accurately estimate the joint velocities. These are also the methods that result in the most accurate joint torque estimation, although the differences are less significant there. In this experiment, the difference between the kinematic EKF and the IEKF, which does include system dynamics, is larger. Interestingly, this difference is mostly observed at the level of joint angles, while the joint torque RMSEs for the two methods are quite similar. Nonetheless, when we focus on a section of the data, the estimated joint torques do show some differences. A section of the results for the kinematic EKF and the marker/IMU IEKF is shown together with the ground truth data in Figure 6-18. The estimated kinematics in this section are very similar, however the torque estimates differ somewhat. The difference is minor, but the IEKF produces a smoother joint torque estimate here. The reason for this

Joint angle RMSE [deg]							
Estimation method	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	Overall
Marker IK/ID (non-causal)	0.59	0.39	0.74	0.47	0.42	0.39	0.52
Marker IK/ID (causal)	1.53	1.88	1.26	2.03	2.06	2.20	1.86
OBIK/ID (non-causal)	0.59	2.03	0.92	0.85	1.26	0.96	1.19
OBIK/ID (causal)	2.03	3.39	2.07	2.74	2.98	2.88	2.73
Kinematic EKF	0.64	0.91	0.73	1.04	1.03	0.96	0.89
Marker/IMU EKF	0.28	0.49	0.28	0.58	0.58	0.70	0.51
IMU IEKF	0.80	1.14	0.96	1.14	1.24	0.98	1.05
Marker IEKF	0.05	0.09	0.11	0.14	0.33	0.72	0.33
Marker/IMU IEKF	0.28	0.49	0.28	0.58	0.58	0.70	0.51

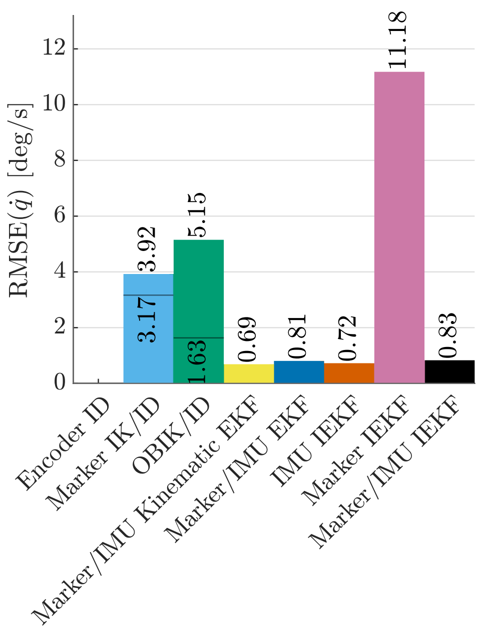
Table 6-6: The RMSE of the estimated joint angles in the interaction force experiment, for each estimation method. The highest and lowest RMSE values are indicated in red and green respectively.

Joint torque RMSE [Nm]							
Estimation method	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	Overall
Encoder ID	2.09	1.98	1.16	1.59	0.49	0.70	1.46
Marker IK/ID (non-causal)	2.15	2.14	1.16	1.65	0.48	0.71	1.53
Marker IK/ID (causal)	2.52	3.30	1.43	1.81	0.50	0.73	1.97
OBIK/ID (non-causal)	2.18	2.80	1.24	1.60	0.50	0.70	1.71
OBIK/ID (causal)	2.89	5.23	1.56	1.89	0.53	0.70	2.66
Kinematic EKF	1.87	2.34	1.18	1.46	0.49	0.69	1.49
Marker/IMU EKF	2.02	2.12	1.20	1.55	0.51	0.72	1.48
IMU IEKF	1.90	2.45	1.20	1.55	0.49	0.71	1.54
Marker IEKF	2.33	3.34	1.37	2.00	0.58	0.85	1.98
Marker/IMU IEKF	2.02	2.12	1.20	1.55	0.51	0.72	1.48

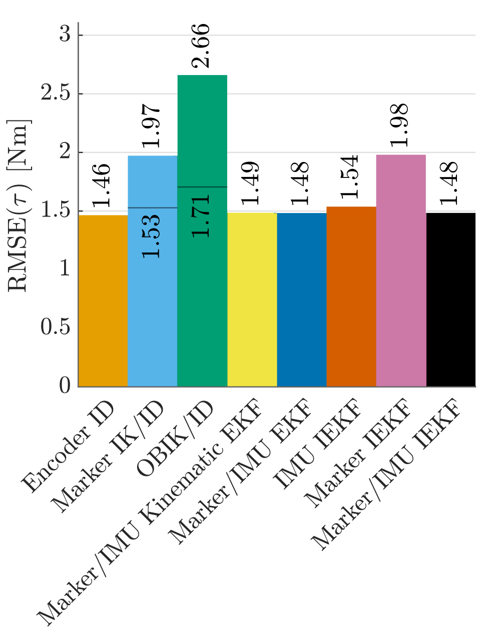
Table 6-7: The RMSE of the estimated joint torques in the interaction force experiment, for each estimation method. The highest and lowest RMSE values are indicated in red and green respectively.



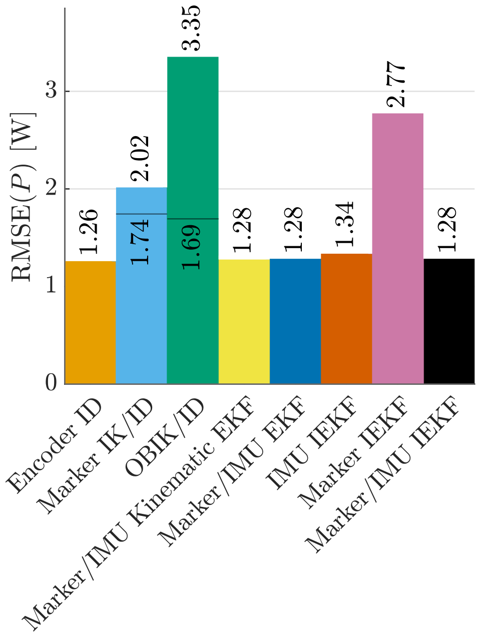
(a) Joint angle RMSE for all estimation methods. Encoder ID has zero error since this is the ground truth signal.



(b) Joint velocity RMSE for all estimation methods. Encoder ID has zero error since this is the ground truth signal.



(c) Joint torque RMSE for all estimation methods.



(d) Joint power RMSE for all estimation methods.

Figure 6-17: Comparison of the accuracy of all estimation methods in the interaction force experiment. Marker IK/ID and OBIK/ID show two values to indicate the difference between causal and non-causal filtering.

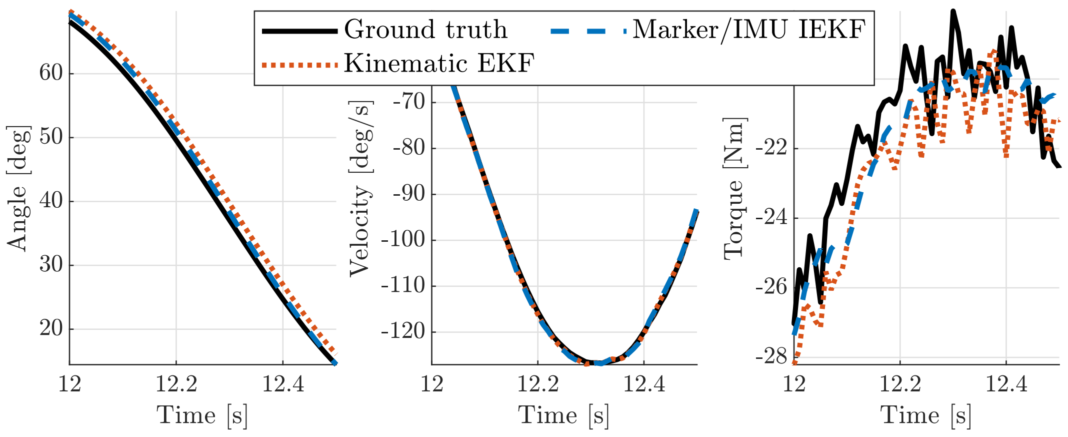


Figure 6-18: The estimated joint angle, velocity and torque for joint four during a section of the interaction force experiment.

effect is that the joint torque estimate for the kinematic EKF comes from an ID calculation. In ID, the measured interaction forces are applied as-is, while in the IEKF they are treated as a noisy input. This means the IEKF does not always balance the measured forces exactly. Instead, it bases its torque estimate on the random walk model and the sensor data, weighted by their relative uncertainties. Since the torque sensor in the robot is also affected by some sensor noise, it is difficult to tell which of these two methods paints the most realistic picture. Again, it appears as if the lower bound for the joint torque RMSE is determined not by the accuracy of the sensors but rather that of the dynamic model.

Throughout this chapter, the IEKF estimation framework presented in this thesis has been evaluated both on its properties as an algorithm, as well as on how it compares to several state-of-the-art methods. Wherever possible, the differences between various results were linked to the properties of the algorithms that produced them. What remains for the next chapter, is to connect conclusions and recommendations to these results.

Chapter 7

Conclusion

Based on the results presented in Chapter 6, a number of conclusions can be drawn about the efficacy of the estimation methods evaluated in this thesis. Before doing that, however, it is a good idea to return to the goals of this thesis as they were laid out in Chapter 1. The aim of the thesis was to design not only an estimation framework, but also the necessary tools to test and validate it. The purpose of this endeavor was to determine how kinematics and kinetics are estimated most effectively in biomechanical systems. The conclusions presented in this chapter can be used to inform biomechanicists designing a human motion experiment. The practical application of this framework in a real human subject experiment will therefore be kept in mind when we evaluate the three hypotheses posed in Chapter 1.

Given a kinematic model, the use of optical marker data will allow the most accurate estimation of its joint angles, compared to IMU-only methods.

When we consider the IEKFs that use only marker data or only IMU data, the difference in estimation results is clear: the marker-based filter results in a better estimate of the joint angles in both experiments. This confirms expectations: the marker data gives a direct position-level measurement of the kinematics. With four markers per segment, a redundant description of the position and orientation of each segment is given. The IMU-based filter has to separate the position-dependent and acceleration-dependent parts of the measured specific force, which makes it more difficult to estimate joint angles. Furthermore, the IMU-only IEKF relies on integration of the angular velocity to estimate the robot's heading, which introduces some drift.

When we compare the marker-based IEKF with the marker-based and IMU-based methods from literature, namely marker-based IK and OBIK, we find that only the non-causally filtered methods can to come close in accuracy to the IEKF, even though the IEKF is a causal estimation method. This suggests that the accuracy of the proposed framework is possibly also improved by a non-causal implementation in the form of a nonlinear Kalman smoother. The OBIK method estimates the least accurate joint angles, even though it was given 'perfect' magnetometer measurements. This can likely be attributed to the erroneous assumption in the Madgwick orientation estimation that gravity dominates the measured specific force.

This effect was observed in both experiments, even though the induced acceleration in the second experiment was lower. Comparing the marker-based methods among themselves, the tightly coupled IEKF provides more accurate joint angle estimates than the traditional IK/ID methods. The quality of the estimates degrades when comparing joint velocities, torques and powers though. This is in part explained by the delay introduced by the IEKF. However, the IEKF also performs worse than the causally filtered IK/ID method, which also suffers from delay. It is therefore not the case that a tightly coupled system always outperforms a loosely coupled one.

Finally, while it is true that marker-based methods can provide better joint angle estimates than the IMU IEKF, the differences are in the range of 0.20° to 0.75°. When designing a biomechanics experiment, it should be questioned whether these differences warrant the use of a motion capture system, especially since such a system is often an order of magnitude more expensive than a set of high-quality IMUs.

When IMU data is used in a tightly coupled estimation method, direct angular velocity and specific force measurements will improve joint velocity and torque estimates for a given dynamic model, compared to loosely coupled methods and marker-only methods.

Out of all single sensor type methods (IK/ID, OBIK/ID, marker-based IEKF, IMU-based IEKF), the IMU-based IEKF performs considerably better than the rest when estimating joint velocities. This can be attributed to the direct use of angular velocity measurements in the estimation. It also performs among the best when estimating joint torques, where it is only outperformed by the non-causal IK/ID method. The accurate estimation of joint velocities is also visible in the joint power estimates, for which the IMU IEKF is the most accurate. The direct acceleration level measurements in particular make it possible to estimate sudden changes in torque, as shown in Figure 6-7. Sudden changes in joint torques might occur in motions that involve contact events. Examples include floor contact during gait or ball contact in sports like field hockey, football or tennis. When studying these types of motion, including IMU data could improve the accuracy of the results.

The fact that the IMU IEKF is a causal method does not limit its accuracy. Since it has access to direct velocity and acceleration level measurements, it does not introduce any delay like the causal marker-based methods and the causal OBIK/ID method. A downside of the non-causal methods is that they can distort the order of events in the estimated signals. When the goal of a human subject experiment is to infer information about the causality of observed events, a non-causal filter is therefore discouraged. Since the causally filtered methods do not provide the same level of accuracy, the use of IMUs in a tightly coupled fashion can improve results in this type of experiment. This conclusion also applies in a scenario where a real-time system is required, like a coach giving instantaneous feedback to an athlete. The added benefit of angular velocity and specific force measurements provided by the IMUs only applies when these measurements are integrated in a tightly coupled way. The information they provide about the instantaneous velocity and acceleration of the system is discarded when the IMU data is only used indirectly, like in OBIK. This is visible in the poor accuracy of the OBIK/ID-derived joint velocities.

Given hypotheses one and two, the fusion of marker and IMU data in a tightly coupled estimation method will yield the most accurate joint kinematics and kinetics.

The estimation methods that had access to both types of data consistently outperformed all methods that used only one sensor type. The combination of drift-free and accurate position data with angular velocity and specific force measurements made it possible to simultaneously estimate joint angles, velocities and torques with RMSEs below $1°$, 1.5 deg/s and 1.5 Nm respectively in both experiments. No method that uses only one type of sensor was able to provide this level of accuracy on all estimated quantities. This proves the potential of sensor fusion in motion analysis: it allows the algorithm to take advantage of the strengths of one sensor type, while compensating for its weaknesses with another sensor type. From the methods that were tested (kinematic EKF, marker/IMU EKF and marker/IMU IEKF), there is not one that is clearly preferable to the others. The methods that included system dynamics estimated joint angles more accurately, but they introduce extra complexity as well as more computation time. The methods that include system dynamics also filter some of the high-frequency oscillations introduced to the torque estimates by the force sensor. This filtering effect does not result in a lower RMSE however, so it is not clear whether the added complexity introduced by the system dynamics is really needed for accurate estimation. Between the EKF and IEKF with system dynamics, no significant difference in estimation accuracy existed, but the IEKF recovered from an inaccurate initial state faster. There are other situations, not explored in this thesis, in which the IEKF might prove to be useful. This will be discussed in more detail in Section 7-2.

Whether the high accuracy from the marker/IMU IEKF would translate to a real human subject experiment depends on many additional factors. Since this estimation method requires two measurement systems, experiment designers should ask themselves what level of accuracy is needed for their particular application. Consider also the fact that the parameters of the dynamic model used in this thesis were identified from ground truth measurements of the joint torques. Even still, the inaccuracy of this model explains a significant part of the error observed in the joint torque estimates. Biomechanical models of humans cannot be validated with respect to any ground truth, so they can contain significant errors as well. The errors in the dynamic models are likely a greater contribution to the estimation errors than the particular type of sensor used for the data collection.

7-1 Limitations

This study did not take into account many of the complexities that occur in a human subject experiment. For example, the sensors in this study were attached to the rigid segments of the robot. When measuring human motion, the sensors would be attached to the skin and can therefore move relative to the rigid bones, introducing soft tissue artifacts (STA). This could be problematic in the case of IMUs more so than for markers, since markers are more easily attached to bony landmarks on the body segments. The robot used in the experiments was also fixed to the ground, so there was no need to estimate the position of its base. Position estimation is something IMU-based methods often struggle with. Furthermore, the sensor registration of the IMUs was done with the aid of the motion capture system. This is of course not possible in the case of an IMU-only experiment, so a different way to reliably align the sensors with the model is needed there. De Kanter showed that IMU-only estimation is most sensitive to errors in the orientation of the IMUs with respect to the body segments, rather than the position [34]. This should be kept in mind when designing a human subject experiment. Finally, some of the parameters of the IEKFs in this study were based on the ground truth data, which is also not available in a human subject experiment. While the IEKF is not overly sensitive to these parameters, its optimal functioning relies on realistic values for the noise covariance matrices. The tuning of these matrices without access to ground truth data was investigated in Section 6-2-1, but the suggested method does not guarantee optimal performance. Also, only the scaling of the matrices was investigated. The sensitivity to the base values for $\bar{R}_m$ and $\bar{R}_{IMU}$ was not investigated due to the large number of parameters involved. Perhaps an estimate for $\bar{R}_m$ and $\bar{R}_{IMU}$ could also be derived from the reprojection error.

Another limitation of the present study is the large computation time required for some of the estimation runs. The computation time varied for the different types of IEKFs that were tested. Calculations that were found to be computationally expensive were the generalized accelerations and their Jacobian and the IMU measurements and their Jacobian. These are the calculations that require a numerical approximation of the Jacobian and had to be evaluated many times. The filters that did not need some of these calculations, like the kinematic EKF or the marker-based IEKF could be evaluated much faster, but the IEKF with all sensors included (markers, IMUs and force/torque sensor) could take up to 40 minutes to process the data recorded in the 30 second experiment. Due to this large computation time, some factors were not investigated, like the influence of the relative level of noise ascribed to the force sensor or the influence of sensor registration errors. These factors could influence the result, as was seen in Figure 6-16.

The accuracy of the identified model can also be considered a limitation of this thesis, since it was found that it limits the degree to which the accuracy of the joint torque estimates could be determined. When all estimation methods make use of the same faulty dynamic model, their individual differences in estimation are less visible in the result. However, this limitation is not unique to the present study. Any biomechanical model is bound to have some inaccuracies. What this thesis also shows is that the estimation error contributed by the model inaccuracy is at least as significant as that from the sensor type and the estimation method.

7-2 Future work

From the aforementioned conclusions and limitations, some recommendations for future work can be made. A major upgrade to the estimation framework would be to improve its efficiency. Currently, the computation times involved are limiting further exploration of the system's capabilities. When evaluating the code's performance, it was found that a lot of overhead is created in the communication between MATLAB and the OpenSim application programming interface (API). This overhead could be avoided by rewriting the system in OpenSim's native C++. This would also make it possible to run some computations in parallel. The IEKF algorithm itself is not very suitable for parallelization, since the estimation at one time instance has to be completed before starting the next. However, the function evaluations for the numerical approximation of Jacobians could in principle be carried out in parallel. These are currently responsible for a large portion of the computation time. The implementation in MATLAB unfortunately does not allow for this. If it is possible to improve the efficiency of the algorithm by a sufficient amount, a real-time implementation could be the next step. This would enable clinicians or coaches to give instantaneous feedback to patients or athletes about the safety or performance of their movement.

Another recommendation stems from the observations made about Figure 6-7. The improvement observed in the estimation of sudden changes in joint torque when IMU measurements are used is something that could be investigated further in an experiment that involves contact events by design. Furthermore, the applicability of the IEKF to different systems should also be evaluated. The robot arm used in this thesis is a great tool for validation of the algorithm, since it measures the quantities of interest. While it is a system with many degrees of freedom performing complex 3D motion, it lacks some of the properties that make the analysis of real biomechanical systems more difficult. One example of this is the aforementioned STA, which could be simulated by not mounting the sensors to the robot rigidly but instead through a soft material representing the muscle and skin of the human body. Another property of biomechanical systems is that they sometimes involve closed-loop kinematic chains, which impose constraints on the generalized coordinates. The estimated generalized coordinates should not violate these constraints if the estimation is to be realistic. Currently, the IEKF framework cannot estimate the motion of constrained systems. This would be a valuable extension of the algorithm. The interpretation of the IEKF as an optimization problem could be helpful here. One might take inspiration from the field of constrained optimization to extend the algorithm and include constraints. In the current study, the performance of the IEKF and the EKF was nearly identical. In the case where constraints are present, the iterative nature of the IEKF could have a larger effect on the outcome of the estimation. Taking further inspiration from the field of optimization, an implementation of the IEKF which features some form of line search with a variable step size could help to make sure that the IEKF converges more quickly. Based on the large differences between the causal and non-causal IK/ID and OBIK/ID methods, an extension of the framework to a nonlinear Kalman smoother could also be an improvement. The largest difference would likely be observed in the marker-based estimation, since no direct velocity or acceleration measurements can be used there. Of course, care should be taken when applying a non-causal estimation method in a real experiment.

To make the algorithm more useful for real experiments, one approach would be to include muscle dynamics in the state space model so they may be estimated by the IEKF as well, similar to the method proposed by Dorschky et al. [33]. Currently, the inclusion of the system dynamics in the IEKF did not make a large difference in the estimated kinetics. In a human subject experiment, this might be different when a dynamic model is prescribed to the muscle forces. In reality, the change in muscle force from one time instance to the next is limited, a fact that is not taken into account in the static optimization of muscle forces explained in Section 2-5. The estimation of muscle forces brings with it two main challenges that should be tackled: the muscle redundancy problem and the physical constraints on muscle force. Since multiple muscles can actuate one joint, the muscle forces for a given joint torque are generally not unique. The IEKF should have some way of deciding the most likely set of muscle forces. Furthermore, muscle forces are inherently constrained. For example, muscles can only pull, not push. To solve this, the constrained estimation proposed earlier could provide a solution.

The most important recommendation to be made is to apply the proposed estimation framework to a real human subject experiment. While some suggestions were made throughout this thesis on how the proposed framework might be applied in practice, a real experiment always exposes new challenges and insights. Modeling, sensor registration, tuning and validation are only some of the challenges when dealing with the complexity of a human subject experiment. All these challenges require a solution before the framework can be successfully applied in a clinical setting or in sports research.

Final remarks

The results presented in this thesis highlight potential benefits that IMUs and sensor fusion can bring to the study of biomechanical systems. This study is a stepping stone to bring the field of human motion analysis out of the lab and to improve our understanding of the dynamics of our bodies. The main takeaway for the reader is that we have much to gain if we use the information we have at our disposal in the right way. By understanding the systems we use and knowing their strengths and weaknesses, we can not only improve our results but also gain insight into their reliability. We still have much to learn about our bodies and knowing the limitations of our knowledge is the first step to improving it. With this research, I hope to have contributed a small amount to this improvement.

Appendix A

Implementation of the state space model in the OpenSim API

In Section 4-2, the iterated extended Kalman filter (IEKF) algorithm for kinematics and kinetics estimation in OpenSim was introduced. In this appendix, the computations in OpenSim was required will be discussed in more detail. For the evaluations of the continuous time process model, the measurement function and the Jacobians of these functions, introduced in Section 3-4, the algorithm uses the OpenSim application programming interface (API). The definitions of these functions and their Jacobians are repeated in this appendix for convenience. The continuous time process model and its Jacobians were defined in (3-24) and (3-25) respectively as

$$\dot{\boldsymbol{x}} = \bar{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}(t)) = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \bar{M}^{-1}(\boldsymbol{q})(\boldsymbol{\tau} + {}^{\mathcal{S}_f}J_f^T(\boldsymbol{q})\boldsymbol{u} - \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \boldsymbol{g}(\boldsymbol{q}) + {}^{\mathcal{S}_f}J_f^T(\boldsymbol{q})\bar{\boldsymbol{v}}_f \\ \boldsymbol{v}_\tau \end{bmatrix}$$

and

$$A(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{0} & \mathcal{I}_{N_q} & \boldsymbol{0} \\ \frac{\partial \text{EoM}}{\partial \boldsymbol{q}} & \bar{M}^{-1}(\boldsymbol{q})\frac{\partial \boldsymbol{c}}{\partial \dot{\boldsymbol{q}}} & \bar{M}^{-1}(\boldsymbol{q}) \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}, \quad B_v(\boldsymbol{x}_k) = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \bar{M}^{-1}(\boldsymbol{q})\,{}^{\mathcal{S}_f}J_f^T(\boldsymbol{q}) \\ \mathcal{I}_{N_q} & \boldsymbol{0}. \end{bmatrix}$$

The equation $\bar{M}^{-1}(\boldsymbol{q})(\boldsymbol{\tau} + {}^{\mathcal{S}_f}J_f^T(\boldsymbol{q})\boldsymbol{u} - \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \boldsymbol{g}(\boldsymbol{q}) + {}^{\mathcal{S}_f}J_f^T(\boldsymbol{q})\bar{\boldsymbol{v}}_f$ was solved using the `calcAcceleration` method of OpenSim's `SimbodyMatterSubsystem` class. This method was also used in the numerical approximation of the Jacobians $\frac{\partial \text{EoM}}{\partial \boldsymbol{q}_k}$ and $\bar{M}^{-1}(\boldsymbol{q}_k)\frac{\partial \boldsymbol{c}}{\partial \dot{\boldsymbol{q}}_k}$. The matrices $\bar{M}^{-1}(\boldsymbol{q}_k)$ and ${}^{\mathcal{S}_f}J_f^T(\boldsymbol{q}_k)$ could be evaluated explicitly. For $\bar{M}^{-1}(\boldsymbol{q}_k)$, the `calcMInv` method was used. ${}^{\mathcal{S}_f}J_f^T(\boldsymbol{q}_k)$, the kinematic Jacobian of the force/torque sensor expressed in the sensor-fixed reference frame, can be computed with the `calcFrameJacobian` and `expressVectorInAnotherFrame` methods. These three methods are all implemented in the

`SimbobyMatterSubsystem` class as well. We now focus our attention on the measurement model, defined in (3-32):

$$\boldsymbol{y}_k = h(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k, \boldsymbol{w}_k) = g(\xi(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k), \boldsymbol{w}_k).$$

To compute $h(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k, \boldsymbol{w}_k)$, the intermediate function $\xi(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v})$ and its Jacobians $\Xi_x(\boldsymbol{x}_k)$ and $\Xi_v(\boldsymbol{x}_k)$, introduced in (3-30) and (4-17) respectively, are needed. The elements of this function and its Jacobians are clearly the same as for the continuous time process model and were therefore computed identically.

$$\boldsymbol{z} = \xi(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}) = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \\ \bar{M}^{-1}(\boldsymbol{q})(\boldsymbol{\tau} + {}^{\mathcal{S}_f}J_f^T(\boldsymbol{q})\boldsymbol{u} - \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \boldsymbol{g}(\boldsymbol{q}) + ({}^{\mathcal{S}_f}J_f)^T(\boldsymbol{q})\bar{\boldsymbol{v}}_f \end{bmatrix}$$

$$\Xi_x(\boldsymbol{x}) = \begin{bmatrix} \mathcal{I}_{N_q} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathcal{I}_{N_q} & \mathbf{0} \\ \frac{\partial \text{EoM}}{\partial \boldsymbol{q}} & \bar{M}^{-1}(\boldsymbol{q})\frac{\partial \boldsymbol{c}}{\partial \dot{\boldsymbol{q}}} & \bar{M}^{-1}(\boldsymbol{q}) \end{bmatrix}, \quad \Xi_v(\boldsymbol{x}_k) = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \bar{M}^{-1}(\boldsymbol{q})({}^{\mathcal{S}_f}J_f)^T(\boldsymbol{q}) \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

Furthermore, to compute the measurement function we need the intermediate function $g(\boldsymbol{z}, \boldsymbol{w})$ and its Jacobian $G(\boldsymbol{z})$, introduces in (3-31) and (4-17) respectively.

$$\boldsymbol{y} = g(\boldsymbol{z}, \boldsymbol{w}) = \begin{bmatrix} {}^{\mathcal{M}}\boldsymbol{p}_{m,1}(\boldsymbol{q}) + \boldsymbol{w}_{m,1} \\ \vdots \\ {}^{\mathcal{S}_1}J_{p,s,1}(\boldsymbol{q})\ddot{\boldsymbol{q}} + {}^{\mathcal{S}_i}\dot{J}_{p,s,1}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - {}^{\mathcal{S}_i}R_{\mathcal{G}}(\boldsymbol{q})\,{}^{\mathcal{G}}\boldsymbol{g} + \boldsymbol{w}_{a,s,1} \\ {}^{\mathcal{S}_i}J_{\omega,s,1}(\boldsymbol{q})\,\dot{\boldsymbol{q}} + \boldsymbol{w}_{\omega,s,1} \\ \vdots \end{bmatrix}$$

$$G(\boldsymbol{z}) = \begin{bmatrix} {}^{\mathcal{M}}J_{p,m,1}(\boldsymbol{q}) & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots \\ \frac{\partial}{\partial \boldsymbol{q}}{}^{\mathcal{S}_1}\boldsymbol{a}_{s,1}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) & \frac{\partial}{\partial \dot{\boldsymbol{q}}}{}^{\mathcal{S}_1}\boldsymbol{a}_{s,1}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) & {}^{\mathcal{S}_1}J_{p,s,1}(\boldsymbol{q}) \\ \frac{\partial}{\partial \boldsymbol{q}_k}{}^{\mathcal{S}_1}\boldsymbol{\omega}_{s,1}(\boldsymbol{q}, \dot{\boldsymbol{q}}) & {}^{\mathcal{S}_1}J_{\omega,s,1}(\boldsymbol{q}) & \mathbf{0} \\ \vdots & \vdots & \vdots \end{bmatrix}$$

The sensor models derived in Section 3-3 can be used to compute this function and its Jacobian in OpenSim. The method by which this was achieved and the required API functions are explained per sensor type below.

Force/torque sensor

Modeling the force/torque sensor in OpenSim brought with it two issues. First of all, the measured force needs to somehow be applied to the model, so that the OpenSim dynamics engine can take these forces into account when computing joint torques and accelerations. Secondly, the physical sensor itself should also be modeled. The dimensions of the sensor determine the application point of the measured force on the model's body segment. This determines the moment arm and therefore significantly affects the dynamics. The sensor also adds mass and rotational inertia to the system, which should be modeled as well. To tackle the first issue, a `PrescribedForce` object was added to the OpenSimModel. The `forceIsGlobal` and `pointIsGlobal` properties of this object should be set to false, since the force/torque sensor measures everything in a local coordinate frame. The measured force data was prescribed to the `PrescribedForce` object in the form of a time-dependent signal. This way, the force data is stored in the model itself which decreases the amount of data transferred between MATLAB and OpenSim during the estimation, which was a significant source of overhead. The second issue was resolved by carefully measuring the dimensions and weight of the force/torque sensor and its coupler and creating a virtual force/torque sensor in OpenSim. This was discussed in detail in Section 5-1-4.

Optical markers

OpenSim provides built-in functionality to handle optical marker measurements. For each experimental marker, a `Marker` object was added to the model in OpenSim. In the estimation, it is assumed that all marker measurements are expressed in the OpenSim ground frame $\mathcal{G}$. However, the motion capture system supplies data in its own reference frame $\mathcal{M}$. To make the measured data compatible with OpenSim, a fixed transformation ${}^{\mathcal{G}}H_{\mathcal{M}}$ was applied to all measured data before estimating the state. The procedure for finding this transformation is detailed in Section 5-1-2. With all markers fixed to the body segments in OpenSim, their predicted measurements and kinematic Jacobians can easily be obtained with the `getLocationInGround` method of the `Marker` class and the `calcStationJacobian` method of the `SimbobyMatterSubsystem` class respectively.

IMUs

In OpenSim, the inertial measurement units (IMUs) were modeled as `PhysicalOffsetFrame` objects. From (3-16) and (3-18) we know that the specific force measured by an IMU is given by

$$^{\mathcal{S}_i}\boldsymbol{a}_{s,i}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}) = {}^{\mathcal{S}_i}J_{p,s,i}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + {}^{\mathcal{S}_i}\dot{J}_{p,s,i}(\boldsymbol{q},\dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}} - {}^{\mathcal{S}_i}\boldsymbol{g}(\boldsymbol{q}).$$

The `SimbodyMatterSubsystem` methods `calcFrameJacobian`, `multiplyByFrameJacobian` and `calcBiasForFrameJacobian` are used to compute the acceleration and the kinematic Jacobian of an IMU in the ground frame. The model's `Gravity` property contains the gravitational acceleration in the ground frame. However, IMUs provide all measurements in a sensor-fixed coordinate frame. Therefore, the `expressVectorInAnotherFrame` method was applied to both the modeled specific force and the kinematic Jacobian to arrive at the final result. Because the angular velocity is simply given by

$$^{\mathcal{S}_i}\boldsymbol{\omega}_{s,i}(\boldsymbol{q},\dot{\boldsymbol{q}}) = {}^{\mathcal{S}_i}J_{\omega,s,i}(\boldsymbol{q})\,\dot{\boldsymbol{q}},$$

only the `calcFrameJacobian`, `multiplyByFrameJacobian` and `expressVectorInAnotherFrame` methods were needed to compute this quantity. The elements of the functions $\bar{f}$ and h and the

Jacobians A, B_v, H and D_v are shown in the leftmost column of Table A-1. The rightmost column shows the OpenSim API functions used to compute them.

Variable	Description	OpenSim API functions
$\ddot{\boldsymbol{q}}$	Joint accelerations	`calcAcceleration`
${}^{\mathcal{G}}\boldsymbol{p}_{m,i}(\boldsymbol{q})$	Marker positions	`getLocationInGround`
${}^{\mathcal{S}_i}\boldsymbol{a}_{s,i}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})$	IMU specific force	`multiplyByFrameJacobian` `calcBiasForFrameJacobian` `expressVectorInAnotherFrame`
${}^{\mathcal{S}_i}\boldsymbol{\omega}_{s,i}(\boldsymbol{q},\dot{\boldsymbol{q}})$	IMU angular velocity	`multiplyByFrameJacobian` `expressVectorInAnotherFrame`
${}^{\mathcal{G}}J_{p,m,i}(\boldsymbol{q})$	Marker Jacobian	`calcStationJacobian`
${}^{\mathcal{S}_f}J_f(\boldsymbol{q})$ ${}^{\mathcal{S}_i}J_{p,s,i}(\boldsymbol{q})$ ${}^{\mathcal{S}_i}J_{\omega,s,i}(\boldsymbol{q})$	Sensor Jacobians	`calcFrameJacobian`, `expressVectorInAnotherFrame`
$\bar{M}^{-1}(\boldsymbol{q})$	Inverted mass matrix	`calcMInv`

Table A-1: The OpenSim API functions used to compute the constituent parts of the state space functions and Jacobians.

Appendix B

Derivation of the system identification data matrix

In Section 3-2, the equations of motion for a general multibody system were derived. For convenience, they are repeated here:

$$\boldsymbol{\tau} + J_f^T(\boldsymbol{q})\boldsymbol{f}_{ext} = J^T(\boldsymbol{q})MJ(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + J^T(\boldsymbol{q})\,M\,\dot{J}(\boldsymbol{q},\dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}} + J^T(\boldsymbol{q})\boldsymbol{f}_{inertia}$$

Also recall the definition of the mass matrix M,

$$M = \begin{bmatrix} m_1\mathcal{I}_{3\times3} & & & & \\ & I_1 & & & \\ & & \ddots & & \\ & & & m_N\mathcal{I}_{3\times3} & \\ & & & & I_N \end{bmatrix},$$

Where J is the kinematic Jacobian of a local frame attached to the center of mass of each body and N is the number of bodies in the system. While the main goal of this thesis is to employ these equations to estimate the generalized coordinates, velocities and forces for a given model, they can also be used in the opposite direction. When the generalized coordinates, velocities, accelerations and forces are all known, as well as the model's kinematic Jacobians, the equations of motion can be solved for the model's masses and moment of inertia tensors $m_1, I_1, \ldots, m_N, I_N$. To achieve this, the equations have to be rewritten to a form that isolates these optimization variables. This is done by first grouping $J\ddot{\boldsymbol{q}}$ and $\dot{J}\dot{\boldsymbol{q}}$ into the new variable $\boldsymbol{a}$.

$$J^T\,M\,J\ddot{\boldsymbol{q}} + J^T\,M\dot{J}\dot{\boldsymbol{q}} + = J^T\,M\,\underbrace{(J\ddot{\boldsymbol{q}} + \dot{J}\dot{\boldsymbol{q}})}_{\boldsymbol{a}}$$

Now, writing this equation in terms of the rows of the mass matrix, $M_1, \ldots, M_{6N}$, and employing the fact that the mass matrix is symmetric, the mass matrix can be vectorized and isolated.

$$J^T\, M\, \underbrace{(J\ddot{\boldsymbol{q}} + \dot{J}\dot{\boldsymbol{q}})}_{\boldsymbol{a}} = J^T \begin{bmatrix} M_1\boldsymbol{a} \\ \vdots \\ M_{6N}\boldsymbol{a} \end{bmatrix} = J^T \begin{bmatrix} \boldsymbol{a}^T & \ldots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \ldots & \boldsymbol{a}^T \end{bmatrix} \begin{bmatrix} M_1^T \\ \vdots \\ M_{6N}^T \end{bmatrix}$$

Using the Kronecker product, denoted by $\otimes$, this equation may be written as

$$J^T M\boldsymbol{a} = J^T \left(\mathcal{I}_{6N} \otimes \boldsymbol{a}^T\right) \operatorname{vec}(M)$$

The vectorized mass matrix can in turn be written as a linear combination of the optimization variables for body i, $\boldsymbol{x}_i$, which is illustrated here for the first row of the matrix:

$$M_1^T = \begin{bmatrix} m_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 \end{bmatrix}}_{S_{1,1}} \underbrace{\begin{bmatrix} m_1 \\ I_{xx,1} \\ I_{xy,1} \\ I_{xz,1} \\ I_{yy,1} \\ I_{yz,1} \\ I_{zz,1} \end{bmatrix}}_{\boldsymbol{x}_1}. \tag{B-1}$$

Constructing a similar matrix $S_{i,j}$ for the ith row of the mass matrix using the optimization variables for body j, the vectorized mass matrix can be written in terms of all optimization variables $\boldsymbol{x}$ as

$$\operatorname{vec}(M) = \begin{bmatrix} S_{1,1} & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ S_{6,1} & 0 & \ldots & 0 \\ 0 & S_{7,2} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & S_{12,2} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & S_{6N-5,N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & S_{6N,N} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_N \end{bmatrix} = S\boldsymbol{x}$$

Recall from (3-6) that the inertia forces also depend on the optimization variables:

$$\boldsymbol{f}_{inertia} = \begin{bmatrix} m_1\,\boldsymbol{g} \\ \boldsymbol{\omega}_1 \times (I_1\boldsymbol{\omega}_1) \\ \vdots \\ m_1\,\boldsymbol{g} \\ \boldsymbol{\omega}_N \times (I_N\boldsymbol{\omega}_N) \end{bmatrix}$$

The optimization variables must be isolated from this equation in a similar manner. This will be illustrated for a single body i in the system:

$$\boldsymbol{f}_{inertia,i} = \begin{bmatrix} m_i\,\boldsymbol{g} \\ \boldsymbol{\omega}_i \times (I_i\boldsymbol{\omega}_i) \end{bmatrix}$$

Since the kinematics are known, the value for $\boldsymbol{\omega}_i$ for each body can be calculated using (3-7). Notice that the cross product $\boldsymbol{\omega}_i \times (I_i\boldsymbol{\omega}_i)$ can be equivalently written as

$$\boldsymbol{\omega}_i \times (I_i\boldsymbol{\omega}_i) = \underbrace{\begin{bmatrix} 0 & -\boldsymbol{\omega}_{i,x}\boldsymbol{\omega}_{i,z} & \boldsymbol{\omega}_{i,x}\boldsymbol{\omega}_{i,y} & -\boldsymbol{\omega}_{i,y}\boldsymbol{\omega}_{i,z} & \boldsymbol{\omega}_{i,y}^2 - \boldsymbol{\omega}_{i,z}^2 & \boldsymbol{\omega}_{i,y}\boldsymbol{\omega}_{i,z} \\ \boldsymbol{\omega}_{i,x}\boldsymbol{\omega}_{i,z} & \boldsymbol{\omega}_{i,y}\boldsymbol{\omega}_{i,z} & -\boldsymbol{\omega}_{i,x}^2 + \boldsymbol{\omega}_{i,z}^2 & 0 & -\boldsymbol{\omega}_{i,x}\boldsymbol{\omega}_{i,y} & -\boldsymbol{\omega}_{i,x}\boldsymbol{\omega}_{i,z} \\ -\boldsymbol{\omega}_{i,x}\boldsymbol{\omega}_{i,y} & \boldsymbol{\omega}_{i,x}^2 - \boldsymbol{\omega}_{i,y}^2 & -\boldsymbol{\omega}_{i,y}\boldsymbol{\omega}_{i,z} & \boldsymbol{\omega}_{i,x}\boldsymbol{\omega}_{i,y} & \boldsymbol{\omega}_{i,x}\boldsymbol{\omega}_{i,z} & 0 \end{bmatrix}}_{\Omega_i} \begin{bmatrix} I_{xx,i} \\ I_{xy,i} \\ I_{xz,i} \\ I_{yy,i} \\ I_{yz,i} \\ I_{zz,i} \end{bmatrix}.$$

This observation allows us to write the internal forces as a linear function of the optimization variables $\boldsymbol{x}_i$ as introduced in (B-1):

$$\boldsymbol{f}_{inertia,i} = \begin{bmatrix} \boldsymbol{g} & \\ & \Omega_i \end{bmatrix} \boldsymbol{x}_i = G_i\boldsymbol{x}_i$$

Combining these equations for all bodies yields an equation for all inertia forces

$$\boldsymbol{f}_{inertia} = \begin{bmatrix} G_1 & & \\ & \ddots & \\ & & G_N \end{bmatrix} \boldsymbol{x} = G\boldsymbol{x}$$

Finally, we can write the generalized forces as a linear function of the optimization variables

$$\boldsymbol{\tau} + J_f^T(\boldsymbol{q})\boldsymbol{f}_{ext} = J^T(\,(\mathcal{I}_{6N} \otimes \boldsymbol{a}^T)S + G)\,\boldsymbol{x} = Y(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\boldsymbol{x} \tag{B-2}$$

This concludes the derivation of the data matrix $Y(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$. Notice that, in order to arrive at this linear form the center of mass Jacobian J and the applied forces $\boldsymbol{f}_{ext}$ have to be known. This is the reason why the location of the center of mass was prescribed in Section 4-1. The fact that the known generalized forces are linear in the optimization variables allows an efficient solution to the system identification problem. In Chapter 4, the term containing the externally applied forces is omitted since no such forces were present during the experiments used for the optimization and validation of the model.

Appendix C

Additional results

C-1 System identification of the KUKA iiwa 7

The figures below present the measured joint torques and the result from inverse dynamics (ID) for the unoptimized and optimized dynamic models of the KUKA iiwa 7 robot. Figure C-1 shows the results for the optimization dataset, Figure C-2 shows the validation dataset.

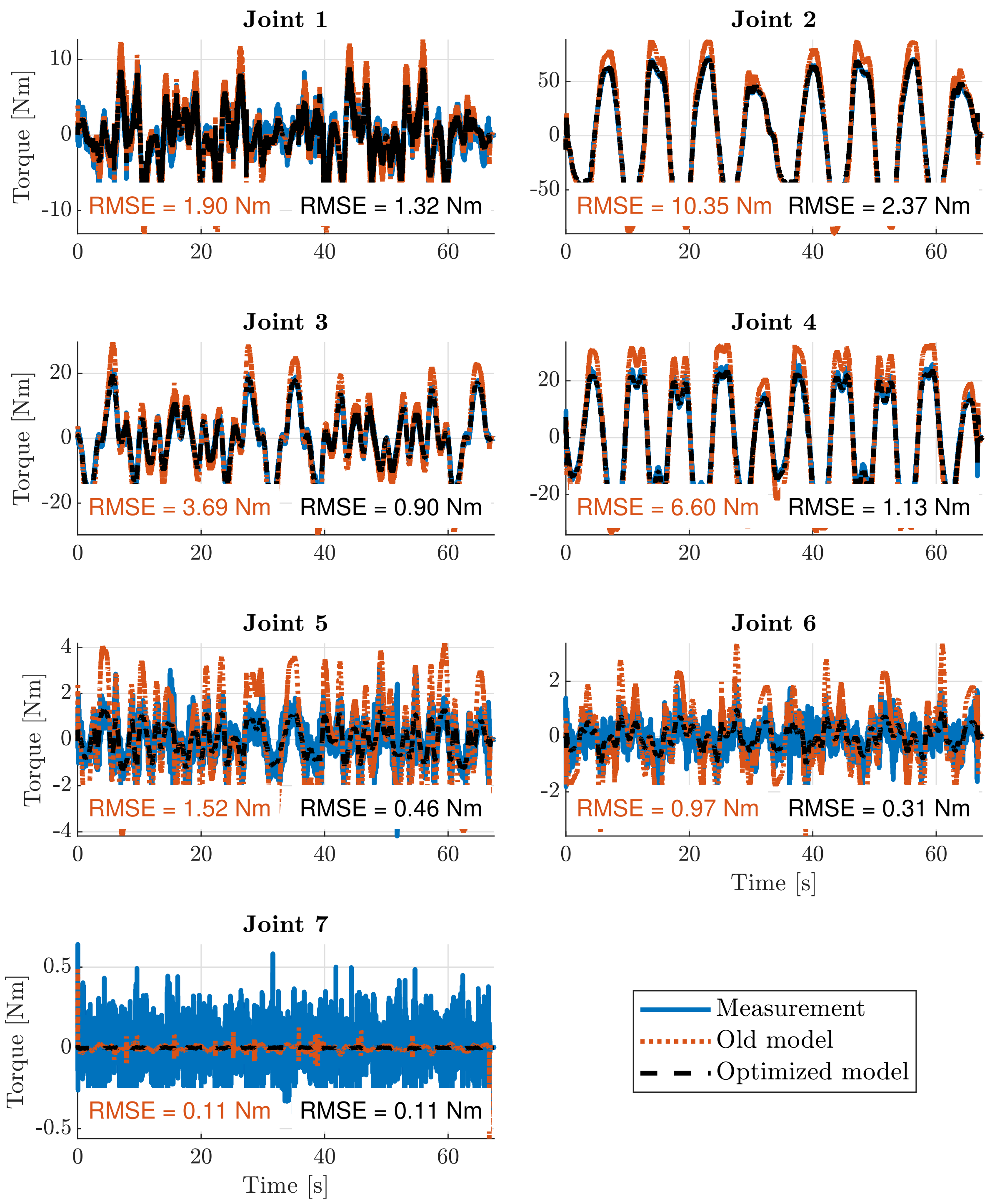


Figure C-1: ID results of the old robot model and the optimized model for all joints, compared to ground truth torque of the optimization dataset.

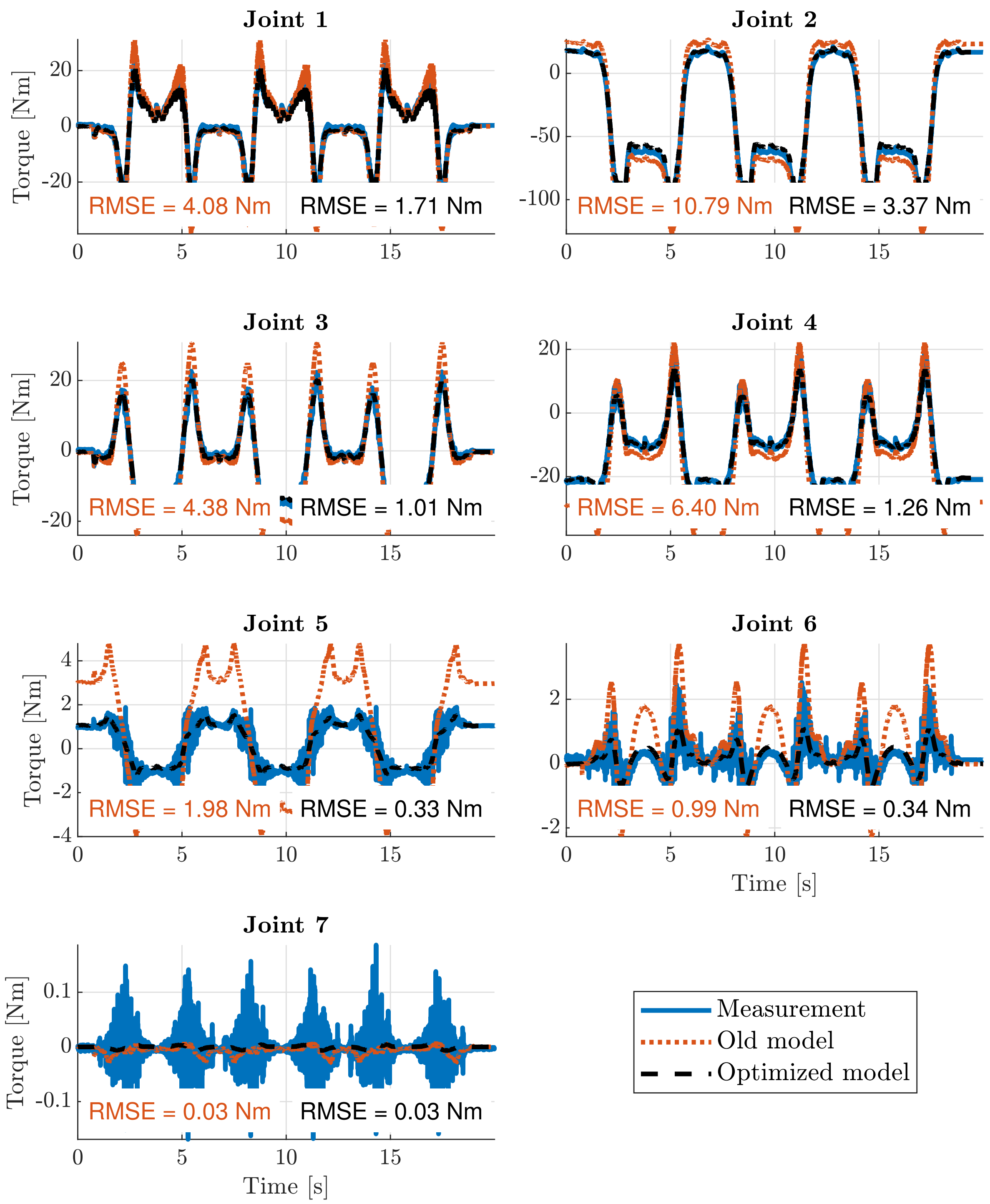


Figure C-2: ID results of the old robot model and the optimized model for all joints, compared to ground truth torque of the validation dataset.

C-2 Experimental validation of the IEKF

For the validation experiments, similar plots as in Figure 6-4, Figure 6-5, Figure 6-15 and Figure 6-16 were made for the joint velocity and power estimated in each run of the iterated extended Kalman filter (IEKF) algorithm. The comparison of the root mean square error (RMSE) for the joint velocity and joint power data for each individual joint is also given in Tables C-1 through C-4.

Joint velocity RMSE [deg/s]							
Estimation method	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	Overall
Marker IK/ID (non-causal)	2.80	2.83	6.14	3.09	5.60	4.25	4.33
Marker IK/ID (causal)	3.85	2.53	10.77	5.93	10.34	5.38	7.17
OBIK/ID (non-causal)	1.91	2.32	3.40	4.59	5.34	5.68	4.13
OBIK/ID (causal)	2.70	5.25	3.91	7.59	9.83	8.97	6.89
Kinematic EKF	0.33	0.71	0.60	0.72	1.11	1.97	1.05
Marker/IMU EKF	0.41	1.12	0.67	0.87	1.34	2.32	1.28
IMU IEKF	0.37	0.57	0.75	0.67	1.04	2.18	1.10
Marker IEKF	4.39	4.76	18.42	9.74	41.99	49.78	28.04
Marker/IMU IEKF	0.41	1.12	0.67	0.87	1.34	2.32	1.28

Table C-1: The RMSE of the estimated joint velocities in the fast motion experiment, for each estimation method. The highest and lowest RMSE values are indicated in red and green respectively.

Joint power RMSE [W]							
Estimation method	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	Overall
Encoder ID	1.69	2.34	1.01	1.87	0.51	0.75	1.51
Marker IK/ID (non-causal)	1.69	4.53	1.37	2.15	0.52	0.76	2.26
Marker IK/ID (causal)	4.84	6.64	2.74	3.40	0.55	0.79	3.82
OBIK/ID (non-causal)	2.28	3.51	1.86	2.57	0.51	0.76	2.18
OBIK/ID (causal)	7.34	10.56	3.07	5.37	0.54	0.81	5.84
Kinematic EKF	1.74	2.70	1.17	1.81	0.51	0.75	1.62
Marker/IMU EKF	1.76	3.12	1.12	1.87	0.51	0.76	1.75
IMU IEKF	1.98	3.16	1.34	2.16	0.51	0.76	1.88
Marker IEKF	4.31	9.34	3.99	4.72	1.70	1.87	5.01
Marker/IMU IEKF	1.76	3.13	1.12	1.88	0.51	0.76	1.75

Table C-2: The RMSE of the estimated joint powers in the fast motion experiment, for each estimation method. The highest and lowest RMSE values are indicated in red and green respectively.

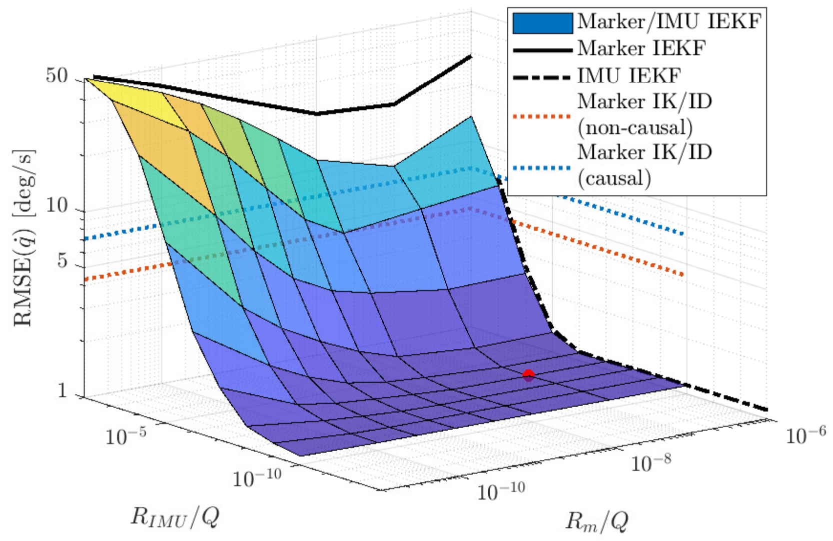


Figure C-3: The joint velocity RMSE of all IEKF estimation runs for the fast motion experiment. The RMSE is plotted against the ratios of the noise matrices of the filter. Blue and red dotted lines indicate the level of accuracy achieved by causally and non-causally filtered IK data respectively. The red dot indicates the location of the lowest RMSE value.

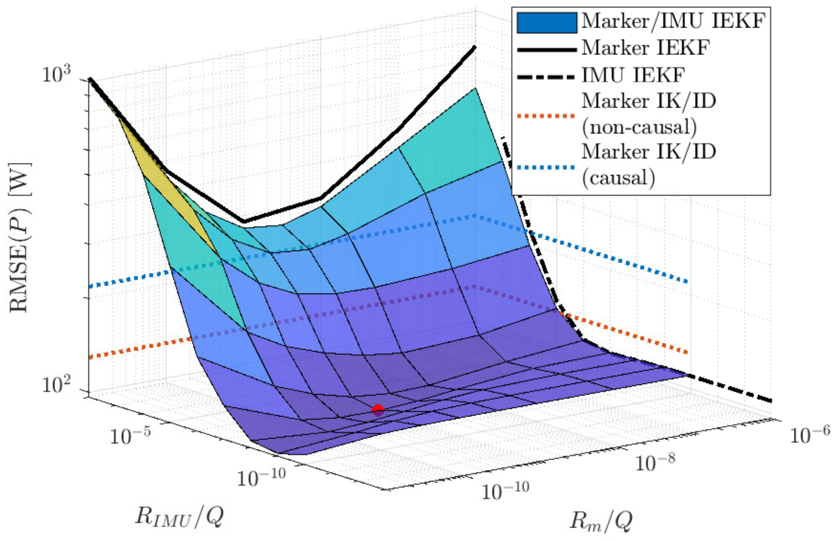


Figure C-4: The joint power RMSE of all IEKF estimation runs for the fast motion experiment. The RMSE is plotted against the ratios of the noise matrices of the filter. Blue and red dotted lines indicate the level of accuracy achieved by causally and non-causally filtered IK/ID data respectively. The accuracy of ID performed with ground truth kinematics is indicated with a black dotted line. The red dot indicates the location of the lowest RMSE value.

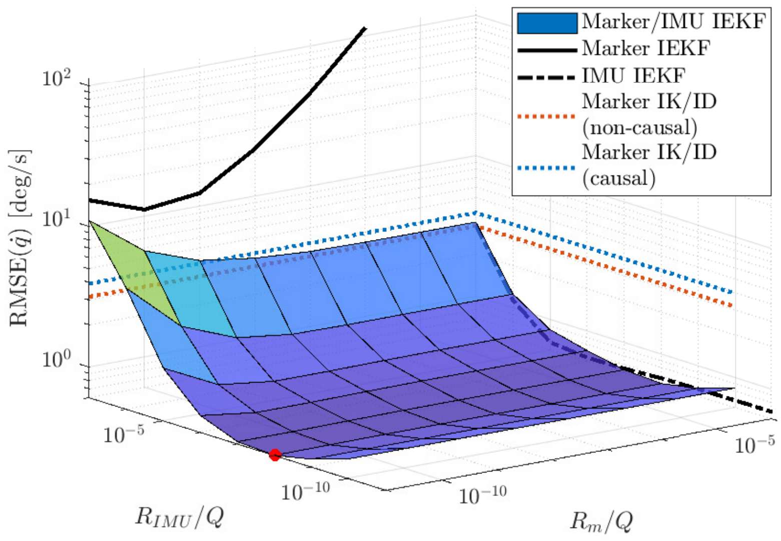


Figure C-5: The joint velocity RMSE of all IEKF estimation runs for the interaction force experiment. The RMSE is plotted against the ratios of the noise matrices of the filter. Blue and red dotted lines indicate the level of accuracy achieved by causally and non-causally filtered IK data respectively. The red dot indicates the location of the lowest RMSE value.

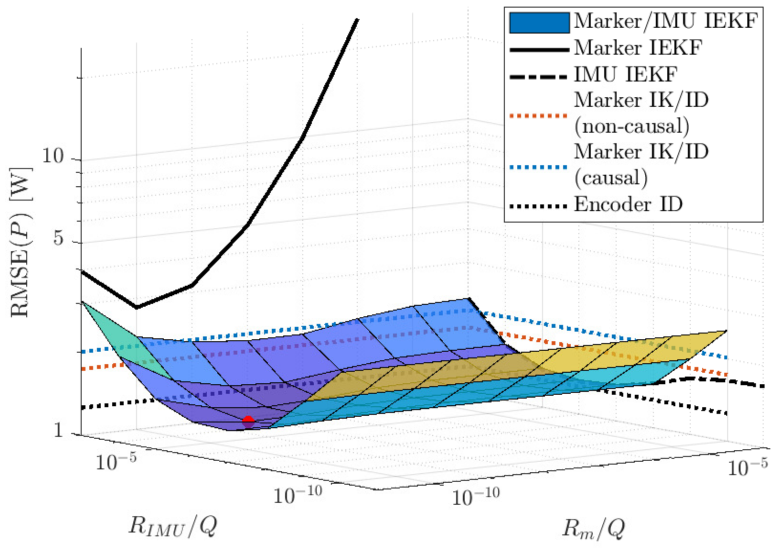


Figure C-6: The joint power RMSE of all IEKF estimation runs for the interaction force experiment. The RMSE is plotted against the ratios of the noise matrices of the filter. Blue and red dotted lines indicate the level of accuracy achieved by causally and non-causally filtered IK/ID data respectively. The accuracy of ID performed with ground truth kinematics is indicated with a black dotted line. The red dot indicates the location of the lowest RMSE value.

Joint velocity RMSE [deg/s]							
Estimation method	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	Overall
Marker IK/ID (non-causal)	3.09	2.28	3.48	3.13	2.96	3.83	3.17
Marker IK/ID (causal)	2.90	2.13	3.08	3.35	3.76	6.67	3.92
OBIK/ID (non-causal)	1.02	1.31	1.49	1.64	1.70	2.34	1.63
OBIK/ID (causal)	3.40	4.88	4.03	5.52	5.05	7.18	5.15
Kinematic EKF	0.32	0.27	0.51	0.66	0.62	1.26	0.69
Marker/IMU EKF	0.39	0.35	0.58	0.67	1.04	1.32	0.81
IMU IEKF	0.37	0.28	0.64	0.67	0.97	1.07	0.72
Marker IEKF	3.14	3.47	5.05	6.06	14.33	21.45	11.18
Marker/IMU IEKF	0.39	0.35	0.58	0.67	1.04	1.32	0.81

Table C-3: The RMSE of the estimated joint velocities in the interaction force experiment, for each estimation method. The highest and lowest RMSE values are indicated in red and green respectively.

Joint power RMSE [W]							
Estimation method	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	Overall
Encoder ID	1.86	1.52	0.61	1.65	0.40	0.70	1.26
Marker IK/ID (non-causal)	1.72	3.00	0.67	2.24	0.40	0.82	1.74
Marker IK/ID (causal)	2.01	3.67	0.83	2.20	0.42	1.07	2.02
OBIK/ID (non-causal)	1.79	3.04	0.69	1.88	0.43	0.74	1.69
OBIK/ID (causal)	2.65	6.97	0.98	3.09	0.47	1.06	3.35
Kinematic EKF	1.53	2.01	0.64	1.52	0.41	0.71	1.28
Marker/IMU EKF	1.62	1.82	0.68	1.66	0.44	0.73	1.28
IMU IEKF	1.53	2.15	0.65	1.62	0.42	0.72	1.34
Marker IEKF	1.69	4.82	0.89	3.64	0.69	2.35	2.77
Marker/IMU IEKF	1.62	1.82	0.68	1.66	0.44	0.73	1.28

Table C-4: The RMSE of the estimated joint powers in the interaction force experiment, for each estimation method. The highest and lowest RMSE values are indicated in red and green respectively.

Bibliography

[1] J. D. B. Stillman and E. Muybridge, *The horse in motion as shown by instantaneous photography, with a study on animal mechanics founded on anatomy and the revelations of the camera, in which is demonstrated the theory of quadrupedal locomotion.* Boston: James R. Osgood and Company, 1882.

[2] D. Fortenbaugh, G. S. Fleisig, and J. R. Andrews, "Baseball pitching biomechanics in relation to injury risk and performance.," *Sports health*, vol. 1, pp. 314–20, jul 2009.

[3] S. Õunpuu, R. B. Davis, and P. A. DeLuca, "Joint kinetics: methods, interpretation and treatment decision-making in children with cerebral palsy and myelomeningocele," *Gait & Posture*, vol. 4, pp. 62–78, jan 1996.

[4] D. G. E. Robertson, G. E. Caldwell, J. Hamill, G. Kamen, and S. N. Whittlesey, *Research methods in biomechanics.* Human Kinetics, 2 ed., 2014.

[5] S. L. Delp, F. C. Anderson, A. S. Arnold, P. Loan, A. Habib, C. T. John, E. Guendelman, and D. G. Thelen, "OpenSim: Open-source software to create and analyze dynamic simulations of movement," *IEEE Transactions on Biomedical Engineering*, vol. 54, pp. 1940–1950, nov 2007.

[6] A. Seth, J. L. Hicks, T. K. Uchida, A. Habib, C. L. Dembia, J. J. Dunne, C. F. Ong, M. S. DeMers, A. Rajagopal, M. Millard, S. R. Hamner, E. M. Arnold, J. R. Yong, S. K. Lakshmikanth, M. A. Sherman, J. P. Ku, and S. L. Delp, "OpenSim: Simulating musculoskeletal dynamics and neuromuscular control to study human and animal movement," *PLOS Computational Biology*, vol. 14, p. e1006223, jul 2018.

[7] A. Seth, M. Sherman, P. Eastman, and S. Delp, "Minimal formulation of joint motion for biomechanisms," *Nonlinear Dynamics 2010 62:1*, vol. 62, pp. 291–303, may 2010.

[8] M. A. Sherman, A. Seth, and S. L. Delp, "Simbody: multibody dynamics for biomedical research," *Procedia IUTAM*, vol. 2, pp. 241–261, jan 2011.

[9] M. Begon, P. B. Wieber, and M. R. Yeadon, "Kinematics estimation of straddled movements on high bar from a limited number of skin markers using a chain model," *Journal of Biomechanics*, vol. 41, no. 3, pp. 581–586, 2008.

[10] L. Sy, M. Raitor, M. D. Rosario, H. Khamis, L. Kark, N. H. Lovell, and S. J. Redmond, "Estimating Lower Limb Kinematics Using a Reduced Wearable Sensor Count," *IEEE Transactions on Biomedical Engineering*, vol. 68, pp. 1293–1304, apr 2021.

[11] M. Iosa, P. Picerno, S. Paolucci, and G. Morone, "Wearable inertial sensors for human movement analysis," *Expert review of medical devices*, vol. 13, pp. 641–659, jul 2016.

[12] B. H. Koning, M. M. van der Krogt, C. T. Baten, and B. F. Koopman, "Driving a musculoskeletal model with inertial and magnetic measurement units," *Computer Methods in Biomechanics and Biomedical Engineering*, 2015.

[13] R. P. Matthew, S. Seko, and R. Bajcsy, "Fusing motion-capture and inertial measurements for improved joint state recovery: An application for sit-to-stand actions," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, pp. 1893–1896, Institute of Electrical and Electronics Engineers Inc., sep 2017.

[14] M. Begon, M. S. Andersen, and R. Dumas, "Multibody Kinematics Optimization for the Estimation of Upper and Lower Limb Human Joint Kinematics: A Systematized Methodological Review," *Journal of Biomechanical Engineering*, vol. 140, no. 3, 2018.

[15] T. W. Lu and J. J. O'Connor, "Bone position estimation from skin marker co-ordinates using global optimisation with joint constraints," *Journal of Biomechanics*, vol. 32, pp. 129–134, feb 1999.

[16] L. Chiari, U. Della Croce, A. Leardini, and A. Cappozzo, "Human movement analysis using stereophotogrammetry. Part 2: Instrumental errors," 2005.

[17] A. Widmann and E. Schröger, "Filter effects and filter artifacts in the analysis of electrophysiological data," *Frontiers in Psychology*, vol. 3, no. JUL, p. 233, 2012.

[18] M. Brodie, A. Walmsley, and W. Page, "Fusion motion capture: A prototype system using inertial measurement units and GPS for the biomechanical analysis of ski racing," *Sports Technology*, vol. 1, no. 1, pp. 17–28, 2008.

[19] M. Kok, J. D. Hol, and T. B. Schön, "Using inertial sensors for position and orientation estimation," *Foundations and Trends in Signal Processing*, vol. 11, no. 1-2, pp. 1–153, 2017.

[20] M. Al Borno, J. O'Day, V. Ibarra, J. Dunne, A. Seth, A. Habib, C. Ong, J. Hicks, S. Uhlrich, and S. Delp, "OpenSense: An open-source toolbox for inertial-measurement-unit-based measurement of lower extremity kinematics over long durations," *Journal of NeuroEngineering and Rehabilitation*, vol. 19, pp. 1–11, dec 2022.

[21] P. Cerveri, A. Pedotti, and G. Ferrigno, "Robust recovery of human motion from video using Kalman filters and virtual humans," *Human Movement Science*, vol. 22, pp. 377–404, aug 2003.

[22] P. Vartiainen, T. Bragge, J. P. Arokoski, and P. A. Karjalainen, “Nonlinear State-Space Modeling of Human Motion Using 2-D Marker Observations,” *IEEE Transactions on Biomedical Engineering*, vol. 61, no. 7, pp. 2167–2178, 2014.

[23] F. De Groote, T. De Laet, I. Jonkers, and J. De Schutter, “Kalman smoothing improves the estimation of joint kinematics and kinetics in marker-based human gait analysis,” *Journal of Biomechanics*, vol. 41, pp. 3390–3398, dec 2008.

[24] M. El-Gohary and J. McNames, “Shoulder and elbow joint angle tracking with inertial sensors,” *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 9, pp. 2635–2641, 2012.

[25] I. Weygers, M. Kok, H. De Vroey, T. Verbeerst, M. Versteyhe, H. Hallez, and K. Claeys, “Drift-Free Inertial Sensor-Based Joint Kinematics for Long-Term Arbitrary Movements,” *IEEE Sensors Journal*, vol. 20, no. 14, pp. 7969–7979, 2020.

[26] G. Tao, Z. Huang, Y. Sun, S. Yao, and J. Wu, “Biomechanical model-based multi-sensor motion estimation,” in *2013 IEEE Sensors Applications Symposium, SAS 2013 - Proceedings*, pp. 156–161, 2013.

[27] A. D. Kuo, “A least-squares estimation approach to improving the precision of inverse dynamics computations,” *Journal of Biomechanical Engineering*, vol. 120, no. 1, pp. 148–159, 1998.

[28] A. Seireg and R. J. Arvikar, “The prediction of muscular load sharing and joint forces in the lower extremities during walking,” *Journal of Biomechanics*, vol. 8, pp. 89–102, mar 1975.

[29] K. R. Kaufman, K. N. Au, W. J. Litchy, and E. Y. Chao, “Physiological prediction of muscle forces—II. Application to isokinetic exercise,” *Neuroscience*, vol. 40, pp. 793–804, jan 1991.

[30] A. Karatsidis, M. Jung, H. M. Schepers, G. Bellusci, M. de Zee, P. H. Veltink, and M. S. Andersen, “Predicting kinetics using musculoskeletal modeling and inertial motion capture,” *arXiv*, jan 2018.

[31] C. K. Chow and D. H. Jacobson, “Studies of Human Locomotion via Optimal Programming,” *Mathematical Biosciences*, vol. 10, pp. 239–306, 1971.

[32] C. L. Dembia, N. A. Bianco, A. Falisse, J. L. Hicks, and S. L. Delp, “OpenSim Moco: Musculoskeletal optimal control,” *PLOS Computational Biology*, vol. 16, p. e1008493, dec 2020.

[33] E. Dorschky, M. Nitschke, A. K. Seifer, A. J. van den Bogert, and B. M. Eskofier, “Estimation of gait kinematics and kinetics from inertial sensor data using optimal control of musculoskeletal models,” *Journal of Biomechanics*, vol. 95, oct 2019.

[34] P. de Kanter, “An IMU tracking algorithm for 3D motion reconstruction using OpenSim dynamical models,” tech. rep., TU Delft, Delft, 2021.

[35] T. D. Barfoot, *State estimation for robotics.* Cambridge University Press, jan 2017.

[36] H. Vallery and A. L. Schwab, *Advanced Dynamics.* TU Delft, 0.2 ed., 2018.

[37] P. Cerveri, M. Rabuffetti, A. Pedotti, and G. Ferrigno, "Real-time human motion estimation using biomechanical models and non-linear state-space filters," *Med. Biol. Eng. Comput*, vol. 41, pp. 109–123, 2003.

[38] V. Fohanno, M. Begon, P. Lacouture, and F. Colloud, "Estimating joint kinematics of a whole body chain model with closed-loop constraints," *Multibody System Dynamics*, vol. 31, no. 4, pp. 433–449, 2014.

[39] T. Nguyen-Van and N. Hori, "New class of discrete-time models for non-linear systems through discretisation of integration gains," *IET Control Theory & Applications*, vol. 7, pp. 80–89, jan 2013.

[40] S. Virga and M. Esposito, "iiwa_stack: ROS integration for the KUKA LBR IIWA R800/R820 (7/14 Kg).," 2018.

[41] A. A. Hayat, V. Abhishek, and S. K. Saha, "Dynamic Identification of Manipulator: Comparison between CAD and Actual Parameters," in *2nd international and 17th National Conference on Machines and Mechanisms*, 2015.

[42] KUKA Roboter GmbH, "LBR iiwa Specification," 2015.

[43] R. Fletcher, *Practical Methods of Optimization.* John Wiley & Sons, Ltd, 2 ed., may 2000.

[44] The Mathworks Inc., "MATLAB," 2021.

[45] OpenSim, "OpenSim API 4.3," 2021.

[46] K. Luttgens and N. N. P. Hamilton, *Kinesiology : scientific basis of human motion.* McGraw-Hill, 12 ed., 1997.

[47] Stanford Artificial Intelligence Laboratory et al., "Robot Operating System," 2018.

[48] K. Chatzilygeroudis, B. Fichera, and W. Amanhoud, "iiwa_ros: ROS Stack for KUKA's IIWA robots," 2021.

[49] OptiTrack, "Motion Capture Systems."

[50] P. Bovbel, "VRPN ROS Client," 2021.

[51] XSens, "MTw Awinda."

[52] M. Guidolin and L. Tagliapietra, "Hi-ROS Xsens MTw Wrapper," 2021.

[53] Xsens Technologies B.V, "MTw Awinda User Manual," 2018.

[54] SCHUNK GmbH & Co. KG, "FTD-Delta SI-330-30," 2022.

[55] S. O. Madgwick, A. J. Harrison, and R. Vaidyanathan, "Estimation of IMU and MARG orientation using a gradient descent algorithm," *IEEE International Conference on Rehabilitation Robotics*, 2011.

Glossary

List of Acronyms

DCSC	Delft Center for Systems and Control
BME	BioMechanical Engineering
CoM	center of mass
EoM	equations of motion
IMUs	inertial measurement units
DoF	degree of freedom
DoFs	degrees of freedom
RMSE	root mean square error
SD	standard deviation
STA	soft tissue artifacts
IK	inverse kinematics
OBIK	orientation-based inverse kinematics
ID	inverse dynamics
EKF	extended Kalman filter
IEKF	iterated extended Kalman filter
UKF	unscented Kalman filter
MAP	maximum a posteriori
ZOH	zero-order hold
OOP	object oriented programming
API	application programming interface
ROS	Robot Operating System
FRI	Fast Research Interface
SDK	software development kit

List of Symbols

$\boldsymbol{\omega}$	Angular velocity
$\bar{M}(\boldsymbol{q})$	Generalized mass matrix
$\boldsymbol{g}$	Gravitational acceleration
${}^{\mathcal{B}}H_{\mathcal{A}}$	Homogeneous transform from frame $\mathcal{A}$ to frame $\mathcal{B}$
${}^{\mathcal{B}}R_{\mathcal{A}}$	Rotation matrix from frame $\mathcal{A}$ to frame $\mathcal{B}$
$\boldsymbol{p}$	Position vector
$\boldsymbol{q}$	Vector of generalized coordinates
$\boldsymbol{u}$	External force input
$\boldsymbol{x}$	System state
$\boldsymbol{y}$	Measurement output
I_i	Moment of inertia tensor of body i
$J(\boldsymbol{q})$	System Jacobian
M	Mass matrix
m_i	Mass of body i
$_{CoM}$	Center of Mass
$_{m,i}$	Marker i
$_{s,i}$	Sensor i
$^{\mathcal{B}_i}$	In the frame of body i
$^{\mathcal{G}}$	In the ground frame
$^{\mathcal{M}}$	In the motion capture frame
$^{\mathcal{S}_i}$	In the frame of sensor i