POLITECNICO DI TORINO

Master's Degree in Mechatronic Engineering

Master's Degree Thesis

Wearable Human Upper Limb Motion Tracking System

A Kinematic-Based Filtering Approach

Supervisors

Prof. Alessandro RIZZO

Candidate

Tommaso LOVATO

April 2024

Abstract

This work aims to define a method for estimating the position and the orientation of the human limbs. Accurately tracking the movements of the human body is a crucial operation for what concerns several applications, like, for example, the remote motor rehabilitation or the creation of a persons' virtual model in a video game, to mention just a few. In particular, an inertial measurement unit (IMU) sensor network is exploited to measure the acceleration and the angular velocity of the upper limb parts, that are combined, along with a kinematic model of the arm, in a sensor fusion algorithm to estimate the orientation of the limb. For the purpose, a 7 degrees-of-mobility kinematic chain is used to model a limb, the three links of which represent the clavicle, the upper arm and the forearm. Being the origin of the global reference frame placed in the thorax, the sternoclavicular joint, the shoulder and the elbow are modeled as multiple rotational and spherical joints.

The orientation of the links in space is represented by quaternions in order to overcome the matrix inversion issues proper of minimal attitude representations. The pose of the links is then estimated by means of an Extended Kalman Filter (EKF), in which the accelerations of a forward kinematic model are compared with the data provided by the sensors.

Acknowledgements

I would like to spend a word of thanks for the brain Technologies team, for believing in me and for all the support they have never made me miss. A special thought of gratitude goes to Luca Bussi, for his professionalism but, mainly, for his willingness and kindness.

I would also thank Professor Alessasndro Rizzo, for his availability to be part of this challenge.

To my parents, to all their efforts.

Table of Contents

List of Tables

List of Figures

Acronyms

IMU
Inertial Measurement Unit

KF
Kalman Filter

EKF
Extended Kalman Filter

DoF
Degree of Freedom

AI
Artificial Intelligence

MEMS
Micro-Electro-Mechanical System

MARG
Magnetic, Angular Rate and Gravity

RF
Reference Frame

DoM
Degree of Motion

DCM
Direction Cosine Matrix

RPY

Roll-Pitch-Yaw

AHRS

Attitude and Heading Reference Systems

DMP

Digital Motion Processing

TRIAD

TRI-Axial Attitude Determination

QUEST

QUaternion ESTimation

SO(3)

Special Orthogonal group

RNN

Recurrent Neural Network

GRU

Gated Recurrent Unit

FC

Fully Connected

ReLU

Rectified Linear Unit

ME

Median Error

MAE

Mean Absolute Error

CAST

Calibrated Anatomical System Technique

ISB

International Society of Biomechanics

JCS

Joint coordinate System

DH

Denavit-Hartenberg

PSD

Power Spectral Density

GM

Gauss-Markov

LTI

Linear Time-Invariant

DRE

Difference Riccati Equation

RMSE

Root Mean Square Error

Chapter 1

Introduction

1.1 Applications Overview

The reconstruction of the human body motion is a field of interest that has been widely investigated in the last decades, for applications that could be very different one from each other. Several tracking approaches have been proposed over the years, many technologies have been exploited and a lot of algorithms have been devised for the purpose. In this context, the video games industry has a broad range of examples. Among the most famous, some solutions are the Microsoft Kinect™, which is based on RGB cameras and infrared projectors to map depth through the light; the Nintendo® Wii™ Remote and the Sony PlayStation® Move both integrate inertial sensors for reconstructing the trajectory of the wand controller and a camera to detect the spatial position of the controller itself. In general, an accurate motion tracking method, looking at the most widespread, is based on computer vision techniques, that can be either silhouette-based or marker-based. The two technologies differs in the way that the vision system retrieve the motion of the subject; the markers are wearable devices that the camera recognizes as fixed points of the body, allowing the reconstruction of the movement by exploiting kinematic models of the human body. Markerless capture systems, also referred to as silhouette-based, do not rely on wearable equipment, but only refers to captured images to virtualize the moving body instead. The latter application is sometimes based on AI-powered techniques rather than built on physical models. Examples of brands that has developed their products exploiting this technologies are Vicon, Simi, Codamotion.

As a matter of fact, vision-based tracking systems are often taken as ground truth for the calibration procedure of inertial sensor-based systems, or simply to compare the performances of such solutions, as done for example in [1, 2]. The main issue about the vision systems is the limitations in the working conditions. In

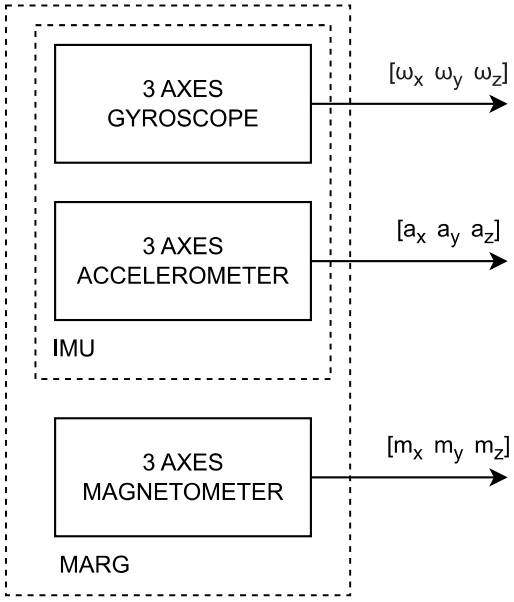

Figure 1.1: Scheme of the IMU and MARG sensors.

fact, as long as those systems are tested in laboratory environments, they work in quasi-ideal condition but, if these conditions are missing, the camera-based systems are no longer trustworthy. The limiting working conditions are for examples the portability of such systems, related to the size and the weight of the devices, the environmental variables, such as the brightness conditions, and the cost of the systems itself.

The class of the inertial sensor considered for this work is somehow complementary to the vision systems introduced above. The most common format, the *inertial measurement unit* (IMU) is composed of a tri-axial accelerometer and a tri-axial gyroscope and provides the measurements of local acceleration along the three orthogonal axes of the sensor and the angular rate about those axes. For applications that do not require high precision instrumentations, the most widespread type of inertial sensors is based on the *micro-electro-mechanical systems* (MEMS), which are microscopic sensors directly realized in the silicon chip. The MEMS integrate both the electronic and the mechanical sensing elements, and possibly the optical ones, providing a variable electric capacitance or resistance value related to the physical quantity to be measured. As a result of the miniaturization of these movement-detection sensors, their application in motion tracking as wearable devices is become a very popular solution, being moreover a quite cheap technology by now. However, the inertial sensors do not provide a position measurement value, but a variation quantity instead, that needs to be time-integrated often more than once to obtain the position, both linear or angular (also referred to as orientation,

see section 2.1). The main issue with this kind of approach is the integration of the non-zero-mean noise that unavoidably affects the measurements, which leads to a time-increasing error that makes the estimated position unreliable after few moments. As a consequence of this, IMUs are hardly used on their own for dead reckoning tracking purposes.

The bias affection in gyroscope measurements is so particularly known that sometimes a different type of sensors is considered instead of IMUs for position estimation: the *magnetic, angular rate and gravity* (MARG) sensors, indeed, are composed of a magnetometer, a part from an accelerometer and a gyroscope. Figure 1.1 helps to visualize the differences between the two classes. The magnetometer measures the external magnetic field in the local coordinates of the sensor, allowing a direct evaluation of the orientation in the Earth's reference frame (global RF). This is particularly effective when a rotational motion happens about the axis defined by gravity vector, a very specific case in which the accelerometer does not provide any relevant information [3].

1.2 Project Description

The stimulus for developing a process to estimate the position of a human arm comes from the necessity to implement an effective remote motor rehabilitation procedure. Motor rehabilitation is an important need for millions of people worldwide; it is in fact a crucial aspect for the well-being of people affected by motor dysfunctions or for cardiovascular diseases recovery, such as stroke [4]. Since it is very difficult and expensive to guarantee a professional physiotherapist for each patient, the development of a system that allows to monitor the correct execution of the exercises prescribed by experts and save the data for a remote evaluation by such professionals would have a dramatic impact on the life quality of victims, as suggested by Balbinot et al. [5].

This thesis proposes an approach, among all the solutions thoroughly investigated in the literature, for tracking the motion of the upper limbs besides evaluating the pose at steady state. An open kinematic chain is adopted for modeling the arm; 7 rotational joints are considered to cause the relative motion between the limb parts. Three 6-axes IMU sensors (3 axes for the accelerometer, 3 axes for the gyroscope) are used for the identification, one per each limb segment: the clavicle, the upper arm and the forearm. A forward kinematic formulation of the manipulator structure defined above is derived, so that the linear acceleration and the angular velocity are calculated for the points of the manipulator where sensors are supposed to be placed. This is the key idea of the work: the two different sources of information relative to the same point (for each point of interest) are combined together, or *filtered*, in a probabilistic framework. In fact, since the measurements are affected

by uncertainties, i.e. noise, and the kinematic model could be satisfyingly accurate only over a certain tolerance interval in the parameters, an estimation process that accounts for the statistical distribution of such variables is needed. For expressing the attitude of the limb segments the quaternion notation is preferred to the minimal representation tools, for the sake of the accuracy of the results and calculation efficiency. In this context, an Extended Kalman filter algorithm is developed, since the dynamic equations describing the attitude evolution are non-linear.

The whole system is simulated in Matlab®; the measurement data are generated by an implementation of the kinematic model described in section 3.2, and then combined in the filter structure discussed in chapter 4. Detailed simulation setup discussion is provided in chapter 5, while achievements and obtained results are reported in chapter 6.

1.3 Thesis Outline

The focus of this work is to define a method for representing the orientation of the limb parts that is based on the fusion among the information from the model of an open kinematic chain and the sensor data related to angular rate and acceleration of the limb. The kinematic model, as well as the Extended Kalman Filter (EKF) algorithm, are implemented in Matlab for evaluating the performance of the method. The thesis organization in based on the following scheme:

- Chapter 1 presents an overview of the application fields of the system under study, comparing some of the different technological solutions that have been developed over the years. The idea behind the project is also introduced, and the specific solutions for the application are explained.
- Chapter 2 introduces the adopted method to represent the attitude of rigid bodies. Then, a comparison among the state-of-the-art literature for what concerns the inertia-based and quaternion-based tracking algorithm is carried out.
- Chapter 3 describes both the kinematic model that is intended to represent the arm and the dynamic system which models the attitude evolution consequent to the motion of the arm joints. The measurement model presents a particular innovation with respect to the literature, primarily developed for navigation purposes.
- Chapter 4 is the most relevant part of the thesis. The data provided by the sensors are combined to the kinematic model of the arm in a Kalman filter algorithm. Being the model non-linear, the application of Extended Kalman filter (EKF) is deepened. Eventually, the determination of the covariance matrix is discussed, since it is a crucial point for the filter performance.

- Chapter 5 presents the simulation conditions and the criteria adopted to generate the dataset used for the performance evaluations of the filter implemented in chapter 4.
- Chapter 6 contains the results of the work. They are reported on the basis of the relevance with respect to the application field of the work.
- Chapter 7 is a summary of the target set and the achievements. It contains a discussion over the choices made and presents some suggestions on possible future developments of the work.

Chapter 2

Background

For investigating the motion of a rigid body in the three-dimensional space, the time evolution of two states of the body must be taken into account: the *position* and the *orientation*. If both are considered, the term *pose* is used. In a cartesian coordinate system, a *reference frame* is defined as a set of three orthonormal vectors; in order to fully describe the body it is convenient identifying 2 reference frames:

- $\mathcal{F}_m = \{O_m, i, j, k\}$ referred to as *mobile*, or *local*, is attached to the body and moves with it. The associated rigid body has constant coordinates in it.

- $\mathcal{F}_0 = \{O, I, J, K\}$ referred to as *fixed*, or *global*, is defined in the space and is usually independent of the moving bodies.

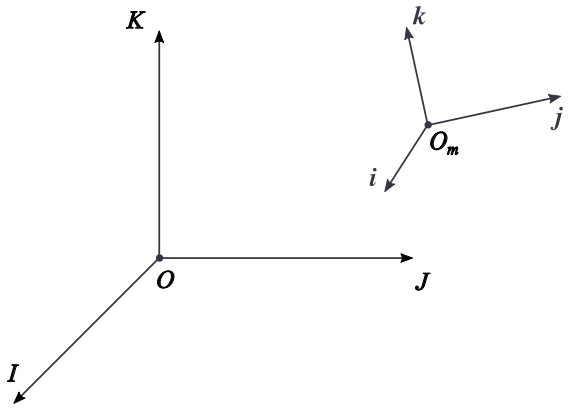

Figure 2.1: Fixed and mobile reference frames.

The two reference frames are shown in figure 2.1. Once the origin and the unit

vectors are set, a generic vector $\overline{v}$ can be represented as:

$$\overline{v} = x\hat{i} + y\hat{j} + z\hat{k}, \text{ in local frame,}$$
$$\overline{v} = X\hat{I} + Y\hat{J} + Z\hat{K}, \text{ in global frame.}$$

Position and orientation, also called *attitude*, of a rigid body with respect to a fixed frame can be characterized by a homogeneous rototranslation matrix ${}^{0}_{m}[T]$, whose structure is:

$${}^{0}_{m}\mathcal{T} = \begin{bmatrix} {}^{0}_{m}\mathcal{R} & \overline{OOm} \\ 0_{1\times 3} & 1 \end{bmatrix}. \tag{2.1}$$

An efficient method for the derivation of such an operator is detailed in section 3.2.1.

2.1 Attitude Representation

The determination of the attitude of a rigid body with respect to a fixed reference frame can be investigated by establishing a relation between two general reference frames. Since various notations to represent the rotation have been developed, let's introduce some of them.

2.1.1 Direction Cosine Matrix

The direction cosine matrix (DCM) is a 3×3 rotation matrix $\mathcal{R}$ whose elements are defined as the dot product between the axes of two generic frames. Let's take as an example the two frames listed in 2:

$$\mathcal{R} \doteq \begin{bmatrix} I \cdot i & I \cdot j & I \cdot k \\ J \cdot i & J \cdot j & J \cdot k \\ K \cdot i & K \cdot j & K \cdot k \end{bmatrix}$$

The columns of the matrix are the projection of the axes of frame $\mathcal{F}_m$ on the axes of frame $\mathcal{F}_0$. Let ${}^{m}\overline{v}$ be a 3×1 vector expressed in mobile frame $\mathcal{F}_m$. The representation of such a vector in frame $\mathcal{F}_0$ can be expressed as

$${}^{0}\overline{v} = \mathcal{R}\ {}^{m}\overline{v}.$$

The rotation matrix is denoted as ${}^{0}_{m}\mathcal{R}$ and is characterized by 9 terms.

Rotation matrix is actually the fundamental tool to derive a formulation for a coordinate change, from a reference frame to another, or to calculate the effect of a rotation applied to a body. However, for denoting a rotation, a less cumbersome notation may be used; out of the nine elements that constitute a rotation matrix,

indeed, only three are independent, since six of them are related by orthogonality constraints due to the fundamental relation:

$$\mathcal{R}^T\mathcal{R} = \mathcal{I}_3$$

where $\mathcal{I}_3$ is the 3×3 identity matrix [6].

2.1.2 Euler Angles

The three parameters defined in the previous section to be the least possible to identify a rotation in space can be thought as independent elementary rotations about three non-parallel consecutive axes. The set of the three parameters is usually referred to as *Euler angles*, and denoted by

$$\Phi = [\phi \quad \theta \quad \psi]^T.$$

Once the axes about which rotations takes place are defined, the corresponding rotation matrices can be derived. In particular, the three elementary rotations about the axes of the body reference frame are represented by the following expressions:

$$\mathcal{R}_x(\phi) \doteq \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{2.2}$$

is the matrix representing a rotation of an angle ϕ about the x axis of the frame;

$$\mathcal{R}_y(\theta) \doteq \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.3}$$

is the matrix representing a rotation of an angle θ about the y axis of the frame;

$$\mathcal{R}_z(\psi) \doteq \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

is the matrix representing a rotation of an angle ψ about the z axis of the frame.

Sometimes a different choice in terms of order in the rotation axes selection may be convenient. Out of the 12 possible configurations of rotations defined about non consecutive axes, the most common are:

- *proper Euler*, are the composition of the rotation about the first axis, e.g. z according to figure 2.2, followed by a rotation about the rotated second axis, e.g. y', and a rotation about the rotated first axis, z''. The complete matrix is obtained as $\mathcal{R}(\Phi) = \mathcal{R}_z(\phi)\mathcal{R}_{y'}(\theta)\mathcal{R}_{z''}(\psi)$ This rotation is also referred to as *Euler ZYZ* or *Euler 323*. Also *Euler ZXZ* (313) representation is commonly used.

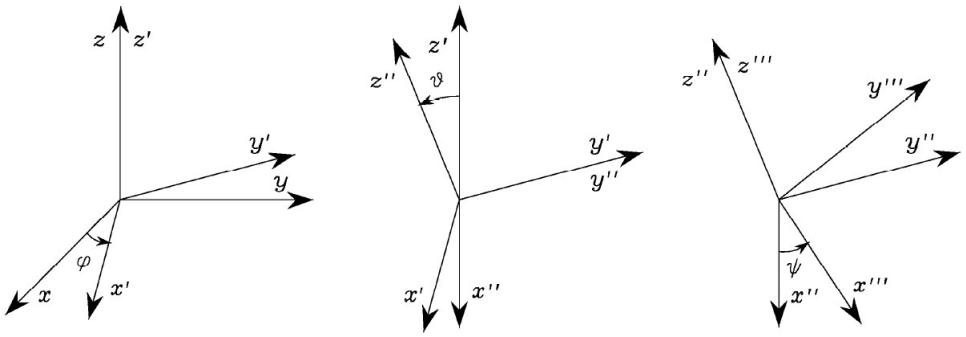

Figure 2.2: Orientation of reference frame with Euler Angles ZYZ. Image originally published in [6].

- *Tait-Bryan*, also known as *Roll-Pitch-Yaw* (RPY) or *Cardan Angles* define three rotations about all the three axes. Taking the *Tait-Bryan 321* for reference, the rotation happens sequentially about axes X, Y, Z of the fixed frame. This combination is expressed by the composition matrix $\mathcal{R}(\Phi) = \mathcal{R}_z(\psi)\mathcal{R}_y(\theta)\mathcal{R}_x(\phi)$ by premultiplying the three elementary matrices in (2.2), (2.3), (2.4). *Tait-Bryan 123* representation is also used.

Considering the rotation matrix that corresponds to a Tait-Bryan 321 representation,

$$\mathcal{R}_{321}(\Phi) = \mathcal{R}_z(\psi)\mathcal{R}_y(\theta)\mathcal{R}_x(\phi) = \begin{bmatrix} c\theta c\phi & -c\phi s\psi + s\phi s\theta c\psi & s\phi s\psi + c\phi s\theta c\psi \\ c\theta s\phi & c\psi + s\phi s\theta s\psi & -s\phi c\psi + c\phi s\theta s\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \quad (2.5)$$

where, for compactness of notation, $s\Phi, c\Phi$ indicate $\sin\Phi, \cos\Phi$, the three angles

ϕ, θ, ψ are calculated as:

$$\begin{aligned}
\phi &= \arctan2\Big(\mathcal{R}_{3,2}, \mathcal{R}_{3,3}\Big) \\
\theta &= \arctan2\Big(-\mathcal{R}_{3,1}, (s\phi\mathcal{R}_{3,2} + c\phi\mathcal{R}_{3,3})\Big) \\
\psi &= \arctan2\Big((-c\phi\mathcal{R}_{1,2} + s\phi\mathcal{R}_{1,3}), (c\phi\mathcal{R}_{2,2} - s\phi\mathcal{R}_{2,3})\Big)
\end{aligned}$$

It can be verified, however, that for $\theta = \pm\pi/2$ a *singularity* occurs. In fact, the rotation matrix in (2.5) becomes:

$$\mathcal{R}_{(\phi,\pm\pi/2,\psi)} = \begin{bmatrix} 0 & \sin(\phi \pm \psi) & \cos(\phi \pm \psi) \\ 0 & \cos(\phi \pm \psi) & -\sin(\phi \pm \psi) \\ -1 & 0 & 0 \end{bmatrix} \tag{2.6}$$

and the three angles are no more independent. In fact, only the sum of ϕ and ψ can be determined, as shown in relation (2.6); in practice, the two axes of rotations ϕ and ψ are aligned (parallel). This occurrence is an important aspect to be taken into account when using a *minimal representation* of rotating bodies, known as *gimbal lock*.

2.1.3 Angle Axis

From the Euler's rotation theorem, it is known that every movement of a rigid body having a point with constant coordinates in a fixed reference frame can be represented by a rotation about an axis passing through that point. The idea behind the proof of such theorem is based on the fact that a rotation matrix has one eigenvalue equal to 1, the eigenvector associated to which is the axis of the rotation,

$$\mathcal{R}\overline{u} = \overline{u}.$$

A rotation is therefore defined by four parameters, the three components of the unit vector of the rotation axis with respect to the local reference frame $\overline{u} = [u_x, u_y, u_z]$ and the value of the angle, say θ, considered positive for counter-clockwise rotations. The angle-axis $\mathcal{R}(\theta, \overline{u})$ is a non-minimal representation of rotations.

2.1.4 Quaternions

A unit quaternion $q = (q_0, q_1, q_2, q_3)$ is a rotation representation based on the Euler's rotation theorem; four terms, known as *Euler parameters*, are defined:

$$\begin{aligned} q_0 &\doteq \cos(\theta/2) \\ q_1 &\doteq u_1 \sin(\theta/2) \\ q_2 &\doteq u_2 \sin(\theta/2) \\ q_3 &\doteq u_3 \sin(\theta/2) \end{aligned}$$

such that the norm is unitary, $\sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1$. This property implies that only three out of the four parameters are independent.

Quaternions are operators defined in the basis $\{1, i, j, k\}$ of a four-dimensional space as an extension of complex numbers, introduced by the Irish mathematician W. R. Hamilton in 1843. The multiplication among the basis vectors is summarized in table 2.1.

×	**1**	**i**	**j**	**k**
1	1	i	j	k
i	i	-1	k	-j
j	j	-k	-1	i
k	k	j	-i	-1

Table 2.1: Multiplication rule of quaternion basis vectors. The first column lists the premultiplication terms, while the first row lists the postmultiplication terms.

Alternative representations of quaternions are:

$$\begin{aligned} q &= q_0 + \overline{q} \\ &= q_0 + q_1 i + q_2 j + q_3 k \\ &= \cos\theta/2 + \overline{u}\sin\theta/2 \\ &= [q_0 \quad \overline{q}]^T \end{aligned}$$

where the term $\overline{q}$ is also denoted as *vector part* of the quaternion. For what concerns basic quaternion algebra, the most common operators are introduced. Being q, p two distinct unit quaternions:

- sum: $q + p = q_0 + p_0 + \overline{q} + \overline{p}$;
- dot product: $\overline{q} \cdot \overline{p} = \sum_{i=1}^{3} q_i p_i$;
- cross product: $\overline{q} \times \overline{p} = \begin{bmatrix} q_2 p_3 - q_3 p_2 \\ q_3 p_1 - q_1 p_3 \\ q_1 p_2 - q2 p_1 \end{bmatrix}$;

- Hamilton product: $q \otimes p = (q_0 + \overline{q}) \otimes (p_0 + \overline{p}) = (q_0 p_0 - \overline{q} \cdot \overline{p}) + (q_0 \overline{p} + p_0 \overline{q} + \overline{q} \times \overline{p})$

Let $\overline{v}_i$ be a vector with coordinates expressed in the reference frame where rotation is defined; the vector that results from a rotation by θ angle about $\overline{u}$ axis, say $\overline{v}_f = \mathcal{R}(\theta, \overline{u})\overline{v}_i$ can be represented by the following quaternion manipulation:

$$(0, \overline{v}_2) = q \otimes (0, \overline{v}_1) \otimes q^*$$

where q is the unit quaternion

$$q \doteq (\cos(\theta/2), u_1 \sin(\theta/2), u_2 \sin(\theta/2), u_3 \sin(\theta/2))$$

and q^* is the complex conjugate relative quaternion, $q^* \doteq [q_0 \quad -\overline{q}]^T$. The quaternion representation of a vector, e.g. $(0, \overline{v}_1)$, is also dentoted as $\overline{v}_1^q$.

For representing successive rotations, defined by the rotation matrices product $\mathcal{R} = \mathcal{R}_1 \mathcal{R}_2 \dots \mathcal{R}_n$, the quaternion rotation operator is the composition of the quaternions of single rotations:

$$q = q_1 \otimes q_2 \otimes \cdots \otimes q_n$$

2.2 State Of The Art

In this section the most relevant approaches for the IMU-based tracking methods are discussed and compared. The development of the research on the orientation tracking received a strong boost with the growth of the aircraft and satellite navigation systems in the 60's, with the implementation of attitude and heading reference systems (AHRS).

Some sensors are provided with an integrated algorithm for the real-time computation of the attitude, also known as Digital Motion Processing™ (DMP) engine.

2.2.1 MEMS Technology

Micro-Electro-Mechanical Systems (MEMS) encompasses the technology of designing and manufacturing miniature integrated devices or systems, that combine electrical and mechanical components on a small chip or substrate. These devices typically range in size from a few micrometers to a few millimeters.

Key aspects of MEMS technology include its extreme miniaturization, enabling compact and portable devices, multifunctionality with multiple sensors or functions on a single chip, fabrication using semiconductor techniques for mass production with high precision, and its wide range of applications across various fields such as consumer electronics, automotive, healthcare, aerospace, and industrial automation. MEMS devices can serve as sensors, actuators, or both, with sensors including

accelerometers, gyroscopes, magnetometers, and pressure sensors, while actuators may consist of micromirrors, microvalves, or microfluidic pumps [7]. Advantages of MEMS technology include low cost, low power consumption, high sensitivity, and robustness, making them suitable for various applications in different environments.

MEMS are nowadays a key technology for the realization of IMU sensors. Accelerometers use a spring-mass system where a proof mass is suspended on stiff supports. When the device experiences acceleration along an axis, the proof mass movement changes the capacitance between the mass and fixed capacitive plates, which is sensed and related to the external acceleration. Gyroscopes, on the other hand, exploit the Coriolis effect. They consist of a vibrating proof mass suspended by flexible beams. When the gyroscope undergoes angular rotation around an axis, the Coriolis force causes the mass to deflect orthogonal to its trajectory. This deflection interferes with capacitive plates or piezoelectric elements, resulting in a change in capacitance or voltage. The output signal represents the rate of angular rotation around one or more axes, depending on the characteristics of the device. In figure 2.3 is shown the structure of a MEMS microsensor and microactuator.

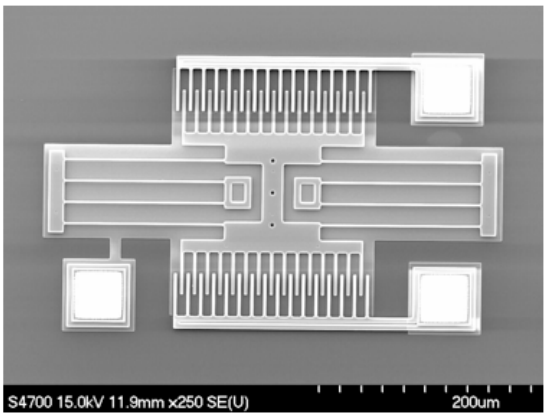

Figure 2.3: MEMS microsensor and microactuator.

2.2.2 TRIAD Algorithm

Tri-Axial Attitude Determination (TRIAD) is a deterministic algorithm which aims at determining the orthogonal matrix A that simultaneously satisfies the relations

$$AV_1 = W_1 \qquad AV_2 = W_2$$

where V_1, V_2 are non parallel reference unit vectors and W_1, W_2 are the corresponding observations unit vectors. Since such a matrix would be overdetermined by the above constraints, the two coordinate systems are defined by the orthogonal

vectors

$$\overline{r}_1 = V_1 \quad \overline{r}_2 = \frac{V_1 \times V_2}{|V_1 \times V_2|} \quad \overline{r}_3 = \frac{V_1 \times (V_1 \times V_2)}{|V_1 \times V_2|}$$

$$\overline{s}_1 = W_1 \quad \overline{s}_2 = \frac{W_1 \times W_2}{|W_1 \times W_2|} \quad \overline{s}_3 = \frac{W_1 \times (W_1 \times W_2)}{|W_1 \times W_2|}$$

which compose the 3×3 matrices

$$M_{ref} = [\overline{r}_1 \ \overline{r}_2 \ \overline{r}_3] \qquad M_{obs} = [\overline{s}_1 \ \overline{s}_2 \ \overline{s}_3].$$

The unique orthogonal matrix that satisfies the above conditions [8] is given by

$$A = M_{obs} M_{ref}^T$$

Being deterministic, the TRIAD algorithm provides a non-optimal solution; moreover, it can accommodate only two observations, and even part of them is discarded, losing in accuracy.

2.2.3 QUEST Algorithm

The QUaternion ESTimation (QUEST) is an optimal algorithm introduced by Shuster et al. [9]. It allows to determine the attitude that achieve the best weighted overlaps of an arbitrary number of reference and observation vectors. A matrix A_{opt} is looked for such that it minimizes the loss function

$$L(A) = \frac{1}{2} \sum_{i=1}^{n} a_i |W_i - AV_i|^2$$

where the coefficients a_i are non-negative weights and

$$\sum_{i=1}^{n} a_i = 1.$$

Vectors V_i, W_i are defined as for TRIAD algorithm in subsection 2.2.2.

Defining the gain function $g(A)$ as $1 - L(A)$, yields

$$g(A) = \sum_{i=1}^{n} a_i W_i^T A V_i \tag{2.7}$$

which is maximized by the optimal value of A, A_{opt}. It is worthwhile to be noticed that when $g(a)$ is maximized, the loss function $L(A)$ is minimized. Exploiting the trace rule, the gain function in (2.7) can be expressed as

$$g(A) = \sum_{i=1}^{n} a_i \operatorname{tr}[W_i^T A V_i] = \operatorname{tr}[AB^T]$$

where the attitude profile matrix B is defined

$$B = \sum_{i=1}^{n} a_i W_i V_i^T$$

Since the matrix A has nine elements subjected to six constraints, the maximization of the gain $g(A)$ would be less complicated by considering the quaternion related to A. The quaternion $\overline{q}$ representing the rotation, is denoted as

$$\overline{q} = [Q \quad q]^T = [\hat{X} sin(\theta/2) \quad cos(\theta/2)]^T$$

where $\hat{X} sin(\theta/2)$ is the vector part defined in section 2.1.4; $\hat{X}$ is the axis about which a rotation of an angle θ occurs.

The attitude matrix A is expressed as a function of the quaternion $\overline{q}$:

$$A(\overline{q}) = (q^2 - Q\dot{Q})I + 2QQ^T + 2q[Q]_x$$

where $[Q]_x$ is defined as the antisymmetric matrix

$$[Q]_x = \begin{bmatrix} 0 & Q_3 & -Q_2 \\ -Q_3 & 0 & Q_1 \\ Q_2 & -Q_1 & 0 \end{bmatrix}.$$

Taking into account the norm constraint of the quaternions expressed in section 2.1.4, the gain function can be formulated as

$$g(\overline{q}) = (q^2 - Q\dot{Q}) \operatorname{tr} B^T + 2 \operatorname{tr}[QQ^T B^T] + 2q \operatorname{tr}[[Q]_x B^T]$$

Some quantities are defined,

$$\sigma = \operatorname{tr} B = B = \sum_{i=1}^{n} a_i W_i V_i$$
$$S = B + B^T = \sum_{i=1}^{n} a_i (W_i V_i^T + V_i W_i^T)$$
$$Z = \sum_{i=1}^{n} a_i (W_i \times V_i)$$

after which the matrix K is defined as

$$K = \begin{bmatrix} S - \sigma I & Z \\ Z^T & \sigma \end{bmatrix}.$$

The combination of the parameters yet defined leads to the bilinear form

$$g(\overline{q}) = \overline{q}^T K \overline{q}$$

which can be redefined by considering the constraint on the quaternion norm in the Lagrange multipliers framework:

$$g'(\overline{q}) = \overline{q}^T K \overline{q} - \lambda \overline{q}^T \overline{q}$$

that yields $K\overline{q} = \lambda\overline{q}$. The gain $g(\overline{q})$ is at maximum if $\overline{q}_{opt}$ is taken as the the eigenvector of K corresponding to its largest eigenvalue, i.e.

$$K\overline{q}_{opt} = \lambda_{max}\overline{q}_{opt}$$

By exploiting the Cayley-Hamilton theorem, a convenient expression for the characteristic equation is derived:

$$\lambda^4 - (a+b)\lambda^2 - c\lambda + (ab + c\sigma - d) = 0$$

where the equation parameters are defined as

$$\begin{array}{rcl} \kappa & = & \mathrm{tr}\left(\mathrm{adj}(S)\right) \\ \Delta & = & \det(S) \\ a & = & \sigma^2 - \kappa \\ b & = & \sigma^2 + Z^T Z \\ c & = & \Delta + Z^T S Z \\ d & = & Z^T S^2 Z \\ \sigma & = & \frac{1}{2}\,\mathrm{tr}\, S \end{array} .$$

By specifying further terms,

$$\begin{array}{rcl} \alpha & = & \lambda^2 - \sigma^2 + \kappa \\ \beta & = & \lambda - \sigma \\ \gamma & = & (\lambda + \sigma)\alpha - \Delta \\ X & = & (\alpha I + \beta S + S^2)Z \end{array}$$

the expression for the optimal quaternion is eventually derived:

$$\overline{q}_{opt} = \frac{1}{\sqrt{\gamma^2 + |X|^2}} \begin{bmatrix} X \\ \gamma \end{bmatrix}$$

The QUEST algorithm is exploited in many applications in combination with an Extended Kalman filter, like in [10, 11]; the accelerometer and magnetometer measurements are combined, and the quaternion estimate provided by the algorithm is fed in the measurement equations of such filter, allowing the measurement model to be linear in quaternion elements.

2.2.4 Mahony's Algorithm

The method proposed by R. Mahony et al. in [12] addresses the problem of retrieving good estimates of the attitude out of systems characterized by high noise levels and time-varying additive biases like low-cost IMU. By formulating deterministic observer kinematics directly on the special orthogonal group SO(3) and employing reconstructed attitude and angular velocity measurements, this approach ensure nearly global stability of the observer error through Lyapunov analysis.

Two observers are introduced: the *direct complementary filter* and the *passive complementary filter*. The former is designed to map gyroscope measurements into the inertial frame and reconstructed attitude directly, while the latter effectively decouples gyro measurements from reconstructed attitude within the observer inputs. Both filters can be extended to estimate gyro bias online, with the passive filter further refined to provide a formulation based on measurement error, avoiding algebraic reconstruction of attitude. The culmination of these advancements is the *explicit complementary filter*, operating only on accelerometer and gyro outputs, suitable for embedded hardware implementation. This filter provides accurate attitude estimates and facilitates online estimation of gyro biases.

Indicating with b the constant bias and with μ the additive noise, the expression for the gyroscope, the accelerometer, and the magnetometer measurements are given:

$$\Omega^y = \Omega + b + \mu$$

$$a = R^T(v \dot{-} g_0) + b_a + \mu_a$$

$$m = R^h + B_m + \mu_b.$$

Common practice is to consider the system in quasi-static conditions, i.e. $\dot{v} \approx 0$, leading to an expression of the accelerometer measurements that only depends on the gravity,

$$v_a = \frac{a}{|a|} \approx -R^T e_3,$$

with $e_3 = \frac{g_0}{g_0} = [0 \quad 0 \quad 1]^T$, being g_0 the gravity acceleration. For what concerns the magnetometer measurements, only the direction of the Earth's magnetic field is relevant,

$$v_m = \frac{m}{|m|}.$$

The vectors of v_a, v_m measurements are combined to retrieve an instantaneous algebraic meeasurement of the rotation from frame B to frame A, ${}^A_B R$:

$$R_y = \underset{R \in SO(3)}{\arg\min}(\lambda_1 |e_3 - R v_a|^2 + \lambda_2 |v_m^* - R v_m|^2)$$

where v_m^* indicates the inertial direction of the magnetic field in the data acquisition area, and λ_1, λ_2 depend on the confidence in the sensor outputs. However, due to the computation complexity of solving an optimization problem of this kind, the error properties of the attitude matrix R_y may be difficult to characterize.

An attitude error criteria is established considering the orthogonal matrix $\hat{R}$ an estimate of the body-fixed rotation matrix R:

$$\tilde{R} \doteq \hat{R}^T R.$$

The cost function definition is based on the Lyapunov stability analysis,

$$E_{tr} = \frac{1}{2}\,\mathrm{tr}(I_3 - R)$$

with the aim of estimating $\hat{R}$ such that the error matrix converges to the identity, $\tilde{R} \to I_3$, yielding $\hat{R} \to R$. Being $\tilde{q} = (\tilde{q}_w \quad \tilde{q}_v)$ the quaternion expressing $\tilde{R}$, the cost function becomes

$$E_{tr} = 2|\tilde{q}_v|^2 = 2(1 - \tilde{q}_w^2)$$

Unfortunately, the drawback of passive and complementary filters require the reconstruction of an estimate of the attitude R_y, also used to map the velocity into the inertial frame. To overcome this issues, the explicit complementary filter is introduced. Let v_i be the body-fixed-frame observations of the fixed inertial directions:

$$v_i = R^T v_{0i} + \mu_i, \quad v_i \in B$$

whose estimate is defined as

$$\hat{v}_i = \hat{R}^T v_{0i}.$$

In the case IMU measurements are collected, the observations become

$$v_a = R^T \frac{a_0}{|a_0|}$$

$$v_m = R^T \frac{m_0}{|m_0|}$$

with a cost function expressed as

$$E_{mes} = k_1(1 - \langle \hat{v}_a, v_a \rangle) + k_2(1 - \langle \hat{v}_m, v_m \rangle)$$

with k_1, k_2 relative sensitivity coefficients. Defining:

$$\begin{aligned} \omega_{mes} &= \textstyle\sum_{i=1}^{n} k_i v_i \times \hat{v}_i \\ \dot{\hat{b}} &= -k_I \omega_{mes} \\ \dot{\hat{q}} &= \tfrac{1}{2}\hat{q}p\left(\Omega^y - \hat{b} + k_P \omega_{mes}\right) \end{aligned}$$

where k_I, k_P are tuning parameters, the estimated attitude in Δt sample time is expressed by

$$q_t = q_{t-1} + \dot{\hat{q}}_t \Delta t$$

This algorithm is used for limb attitude estimation in [1]

2.2.5 Recurrent Neural Network

Wei et al. presented in [13] a novel approach to the real-time estimation of the 3D arm motion. Instead of a 9-axis IMU, that includes an accelerometer, a gyroscope and a magnetometer, the work intends to use a 6-axis IMU, i.e. without the magnetometer, and combine the sensor data in a *recurrent neural network* (RNN) to estimate the position of both the wrist and the elbow. The conceptual scheme is shown in figure 2.4. The system is based on the fundamental assumption that the

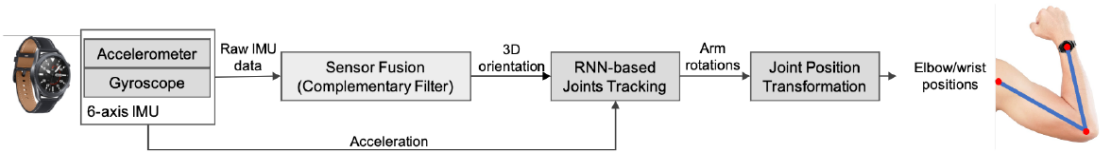

Figure 2.4: Diagram of the proposed arm tracking system.

torso/shoulder is still, otherwise the problem cannot be solved in presence of the ambiguity of the motion caused by the moving body with still arm or still body with moving arm. Figure 2.5 represent the architecture details of the neural network; the accelerometer and gyroscope raw data are combined in a complementary filter to derive an orientation estimate, which is sent, along with the raw accelerometer data, to a Gated Recurrent Unit (GRU) layer, and thereafter to a Fully Connected (FC) layer, a Rectified Linear Unit (ReLU) layer, and a further FC layer.

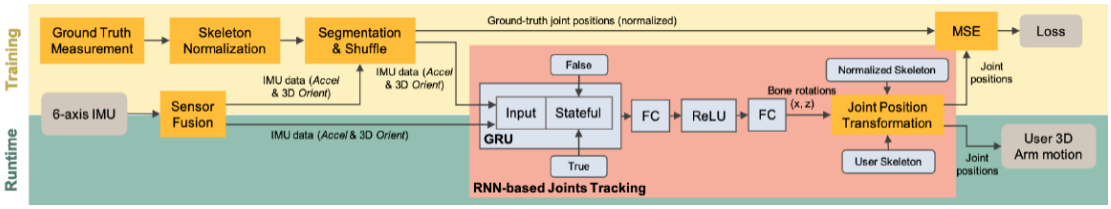

Figure 2.5: Architecture of the RNN-based joints tracking model.

Compared to methods based on 9-axis IMU, the RNN method presented is showing similar performance in terms of median error (ME) and mean absolute error (MAE), but the limiting measurement conditions restrict the application of such a system.

2.3 Proposed Approach

The methods and algorithms shown in the previous section have been developed with the aim of estimating the position and orientation of vehicles or spacecrafts; as a consequence, they focus on a single-body model that moves unconstrained in

the space. In this work, a different perspective is adopted: a kinematic model is considered indeed, in which the estimated motion and attitude of a body depends on the estimated state of the element to which it is jointed to.

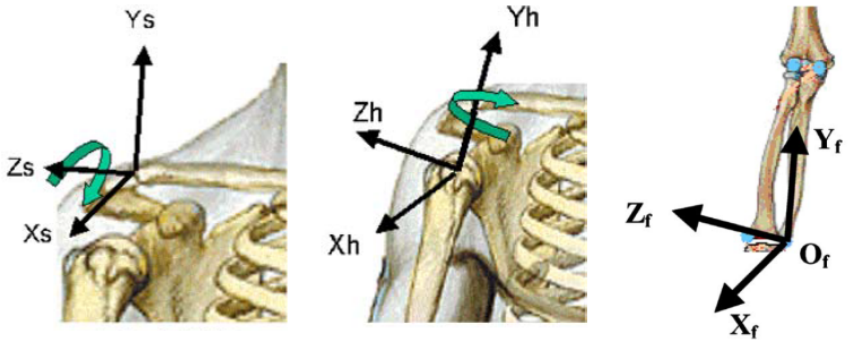

Figure 2.6: Joint coordinate system as defined by ISB [14].

A local reference frame is defined for each part of the arm, according to the procedure suggested by the International Society of Biomechanics (ISB) and summarized in [14]. In figure 2.6 the *joint coordinate system* (JCS) convention is shown; when the arm is in relaxed position, with the palm of the hand pointing forward, the y-axes of the three reference frame are aligned, and so all the x_i and the z_i, with the former pointing forward.

2.3.1 Quaternions Attitude Representation

The attitude of the segment that represents the limb part $\overline{r}_i$ is expressed by means of the quaternion $q = (q_0, q_1, q_2, q_3)$, which denotes the coordinates rotation operator from local frame ${}^{i}\mathcal{F}$ to frame ${}^{i-1}\mathcal{F}$,

$$ {}^{i-1}\overline{r}_i^q = q \otimes {}^{i}\overline{r}_i^q \otimes q^*, \tag{2.8} $$

where the notation $(0, \overline{r}_i) = \overline{r}_i^q$ has been introduced in subsection 2.1.4. The frame ${}^{i-1}\mathcal{F}$ corresponds to ${}^{i}\mathcal{F}$ before the rotation . An equivalent operator for describing the attitude is the rotation matrix $\mathcal{R}_q$ defined after the quaternion q:

$$ \mathcal{R}_q \doteq \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{2.9} $$

Chapter 3

The Model

In order to reconstruct the motion of an arm, two essential elements must be developed: a sensor network to acquire some relevant information and a kinematic model of the limb to compare the measurements to. Since IMU sensors are used, the kinematic analysis is focused on the derivation of an expression for the acceleration in such points where the sensors are supposed to be placed. For the sake of calculation simplification, the aforementioned accelerations are evaluated in correspondence of the arm joints, i.e. three sensors are virtually attached to the scapula, to the elbow and to the wrist and their reference systems are aligned with the JCS. Nonetheless, M.C. Bisi et al. propose a method for calibrating anatomical landmark position in the wearable sensor reference frame based on the *calibrated anatomical system technique* (CAST) protocol in [15].

3.1 Basic Concepts of Functional Anatomy

The human arm motion is made possible by different collaborating systems, organized in a very complicated scheme. The model that is derived for this work, indeed, is only a simplified, yet effective, representation of the very well-designed machine that our body is. The skeleton is composed of the *bones* are the structural part of the system; they give the major contribution to the stiffness of the body and they characterize the length of the limb parts. These rigid elements are linked together by *ligaments* and actuated by *muscles*, to which they are attached by means of *tendons*. The relative motion is guaranteed by the *articulations*, that lower the friction between the bones thanks to a cushion of liquid enclosed by a capsule, called *synovial fluid*.

The articulations responsible for the arm mobility are essentially located in three points of the limb: the thorax, the shoulder and the elbow. The wrist would be the fourth junction to account for, but, as far as the the posture of the arm is

under study, such an articulation is not considered in the overall analysis.

- The *sternoclavicular joint* is a saddle type joint that links the clavicle to the manubrium of the sternum, i.e. the breastbone. It allows the protraction/retraction and depression/elevation movements of the clavicle, motions happening parallel to the transversal plane and to the coronal plane, respectively. The anatomy planes of reference are defined in figure 3.1.

- The *glenohumeral joint* is a ball-and-socket type joint connecting the humerus to the scapula allowing a wide range of movements. Along with the *acromioclavicular* joint, that links the clavicle to the scapula, the glenohumeral joint form the shoulder articulation, that is responsible for the flexion/extension of the upper arm (humerus bone), in the sagittal plane, and the abduction/adduction movement, in the coronal plane.

- The elbow articulation is composed of three joints sharing the same capsule; the *humeroulnar* joint, the *humeroradial* joint and the *proximal radioulnar* joint together form a hinge type joint. The latter is responsible for the pronation/supination of the forearm and the hand (ulna and radius bones), namely the rotation about its longitudinal axis; the wrist itself, in fact, does not allow the dorsal/volar rotation of the hand with respect to the forearm. The humeroulnar and the humeroradial joints cause the flexion/extension movements of the forearm with respect to the upper arm [14, 16].

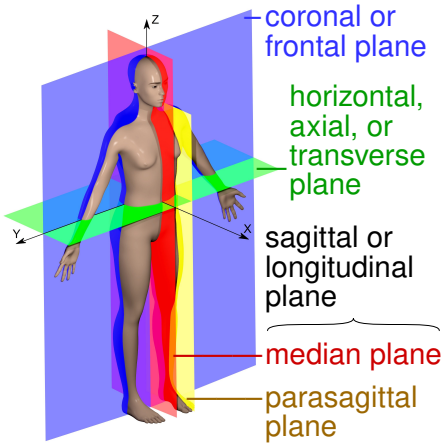

Figure 3.1: Human anatomy planes definition. By David Richfield and Mikael Häggström, M.D. and cmglee - Human anatomy planes, labeled.jpg, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=91212408

3.2 Kinematic Model

The arm, for the purpose of this thesis, is considered to behave like a robotic manipulator having multiple revolute joints, characterized by 7 degrees of mobility. Such a structure is often referred to as 7 DoF in literature [6, p. 58],[17], as each joint acts on a single degree of freedom, called *joint variable*.

The three articulations described in section 3.1 characterize the joints between the three links of the manipulator;

- the sternoclavicular joint, that allows two DoFs between the clavicle, having length l_c, and the thorax, is modeled with two orthogonal revolute joints;
- the shoulder, that allows three DoFs between the clavicle and the upper arm, whose length is l_u, is modeled with three orthogonal revolute joints (behaving like a spherical joint);
- the elbow, that allows two DoFs between the upper arm and the forearm, having length l_f, is modeled by means of two orthogonal revolute joints.

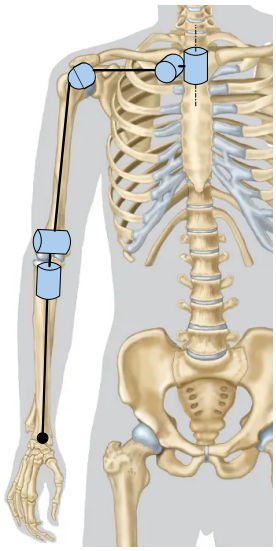

Figure 3.2: Model of the kinematic joints in the arm.

The global reference frame ${}^0\mathcal{F}$ is placed in the thorax, as well as the reference IMU sensor, as will be explained later in this chapter. The collocation of the previously listed joints is visible in figure 3.2. When dealing with kinematic chain analysis, a convenient procedure is to attribute to each link of the chain, r_i, a local reference frame, ${}^i\mathcal{F}$. In manipulator analysis, the main task is carried out by the

last element r_n, called *end-effector*, whose attitude with respect to the *base frame*, i.e. the global reference frame, is calculated as

$$ {}^0_n\mathcal{R} = {}^0_1\mathcal{R}(q_1)\ {}^1_2\mathcal{R}(q_2)\ \ldots\ {}^{n-1}_n\mathcal{R}(q_n), \tag{3.1} $$

where the parameters $[q_1, \ldots, q_n]$ are the joint variables that actually determine the *posture* of the manipulator.

3.2.1 The Denavit-Hartenberg Convention

An effective method to assign a local reference frame to each link of a manipulator has been set up by Jacques Denavit and Richard S. Hartenberg in 1955; when this approach is adopted, the forward kinematic analysis become a simple multiplication of homogeneous transformation matrices related to a single degree of freedom, ordered according to (3.1), and systematically determined.

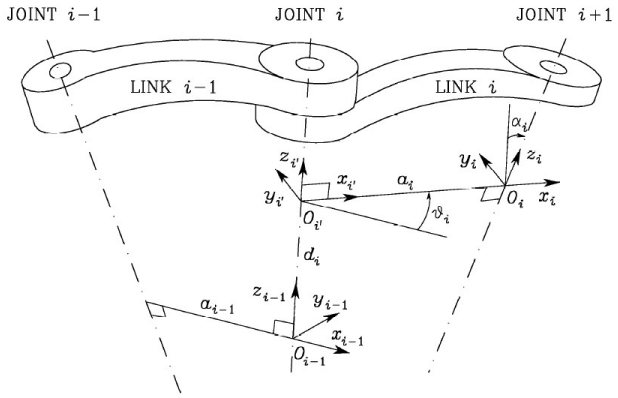

Figure 3.3: Kinematic parameters according to the Denavit-Hartenberg convention. Image originally published in [6].

With reference to the scheme depicted in figure 3.3, the crucial part of the Denavit- Hartenberg (DH) method is the definition of the local reference frame ${}^i\mathcal{F}$ on each joint of the chain:

- axis z_i is chosen in correspondence to the axis of joint J_{i+1};
- axis x_i is chosen along the common normal, i.e. minimum distance vector, to axes z_{i-1} and z_i;
- origin O_i of the local frame is placed at the intersection of axes x_i and z_i;
- axis y_i is chosen in order to complete the right-handed frame ${}^i\mathcal{F}$.

In all such cases when the direction of an axis is indefinite, that is, when the choice is non unique, an arbitrary solution can be adopted for simplifying the calculation.

Once a local frame is attributed to each joint, the whole kinematic chain is defined by specifying four parameters that characterize the joint, usually known as *DH parameters*:

a_i is the distance between the axes of the two consecutive joints J_i and J_{i+1} (the normal);

d_i is the distance between axes x_{i-1} and x_i. It corresponds to the axial length of i^{th} link;

θ_i is the angle between axes x_{i-1} and x_i, about axis z_{i-1};

α_i is the angle between axes z_{i-1} and z_i, about axis x_i.

Depending on the type of the joint, one out the four parameters is the joint variable q_i: θ_i if joint J_i is revolute, d_i if the joint is prismatic. The two remaining parameters are always constant and determine, along with the third excluded by the joint type, the geometry of the manipulator. The general expression for a homogeneous transformation matrix between the coordinate of consecutive joints ${}^{i-1}\mathcal{F}$ and ${}^{i}\mathcal{F}$ is eventually derived:

$$ {}^{i-1}_{i}\mathcal{A}(q_i) = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.2} $$

The matrix A, due to its structure defined in (2.1) has a particular meaning. the 3×3 upper-left submatrix, indeed, is the rotation part and the columns of which are the axes of frame ${}^{i-1}\mathcal{F}$ expressed in frame ${}^{i}\mathcal{F}$; the 3×1 upper-right column vector is the position of the origin of frame ${}^{i-1}\mathcal{F}$ axes, expressed in frame ${}^{i}\mathcal{F}$.

3.2.2 7 DoFs Model

This subsection is devoted to the development of the model built upon the system under study. The Denavit-Hartenberg procedure described in subsection 3.2.1 is adopted to perform the forward kinematic analysis of the manipulator; in figure 3.4 the 7 DoFs kinematic chain represented according to the DH convention is depicted.

As introduced at the very beginning of this section, the articulations allowing multiple degrees of freedom are modeled as orthogonal consecutive rotational joint, connected by dimensionless links. The DH parameters of the manipulator are presented in table 3.1. The expression for the homogeneous transformation matrix

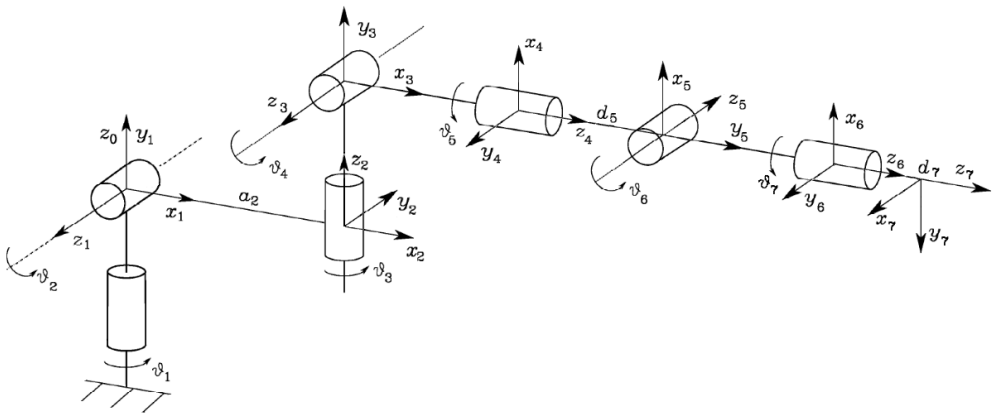

Figure 3.4: DH local frames definition on the kinematic chain.

Joint	a_i	α_i	d_i	θ_i
1	0	$\pi/2$	0	q_1
2	l_c	$-\pi/2$	0	q_2
3	0	$\pi/2$	0	q_3
4	0	$\pi/2$	0	$\pi/2 + q_4$
5	0	$\pi/2$	l_u	q_5
6	0	$-\pi/2$	0	q_6
7	0	0	l_f	$\pi/2 + q_7$

Table 3.1: Denavit-Hartenberg parameters of the manipulator.

$$ {}^{0}_{7}\mathcal{T}(q) = {}^{0}_{1}\mathcal{A}\,{}^{1}_{2}\mathcal{A}\,{}^{2}_{3}\mathcal{A}\,{}^{3}_{4}\mathcal{A}\,{}^{4}_{5}\mathcal{A}\,{}^{5}_{6}\mathcal{A}\,{}^{6}_{7}\mathcal{A} \tag{3.3} $$

of the manipulator defined after (3.2) is reported in appendix A.

Angular Velocity and Acceleration

As introduced at the beginning of the chapter, the purpose of the kinematic analysis is to derive a formulation for the angular velocity and the acceleration expressed in the local sensor frames, that is function of the joint variables q_i. Clearly, sensor

placement is once again a crucial procedure to be carried out; if the i^{th} IMU, s_i, is not aligned with the local frame, the misalignment can be accounted for by introducing a constant transformation matrix ${}^{i}_{s}T$, such that

$$ {}^{0}_{s}\mathcal{T} = {}^{0}_{i}\mathcal{T}(q)\ {}^{i}_{s}\mathcal{T}. \tag{3.4} $$

The angular rate evaluation is based on the observation of the kinematic chain; in fact, according to the DH model, each revolute joint gives a contribution only about the joint axis z_i, according to the expression:

$$ {}^{i-1}\omega_i = [0 \quad 0 \quad \dot{q}_i]^T, \tag{3.5} $$

while the total angular velocity experienced by link r_i is the sum of all the contribution of the joint velocities $\dot{q}_1, \ldots, \dot{q}_i$. It can be iteratively computed as

$$ {}^{i}\omega = {}^{i}_{i-1}\mathcal{R}\ ({}^{i-1}\omega + \dot{q}_i), \tag{3.6} $$

where ${}^{i}_{i-1}\mathcal{R}$ is the inverse of ${}^{i-1}_{i}\mathcal{R}$, that coincides with its transpose; it also corresponds with the upper-left 3×3 submatrix of the DH transformation matrix ${}^{i}_{i-1}A$. Figure 3.4 can be used as a reference to better visualize the concept.

The acceleration sensed in local IMU frames is derived in joint frames for notation simplicity; if the two frames do not correspond, technique shown in (3.4) can be exploited. Each link r_i experiences the acceleration due to the actuation of joints $J_1 \ldots J_i$, that can be found differentiating the fundamental formula of rigid bodies [18]:

$$ \begin{aligned} {}^{i}a &= \frac{d^2\overline{O_i}}{dt^2} \\ &= \frac{d}{dt}(\dot{\overline{O}}_{i-1} + {}^{i}\omega \times \overline{r}_i) \\ &= \ddot{\overline{O}}_{i-1} + {}^{i}\dot{\omega} \times \overline{r}_i + {}^{i}\omega \times {}^{i}\omega \times \overline{r}_i \end{aligned} \tag{3.7} $$

where ${}^{i}\omega$ is the angular velocity from (3.6) and $\ddot{\overline{O}}_{i-1}$ is the acceleration of frame ${}^{i-1}\mathcal{F}$ origin, that can be calculated as

$$ \ddot{\overline{O}}_{i-1} = {}^{i}_{i-1}\mathcal{R}\ {}^{i-1}a \tag{3.8} $$

3.3 Dynamic System Representation

In order to describe the attitude evolution in time of the three limb parts, i.e. clavicle, upper arm, and forearm, a quaternion-based notation is adopted: a non-minimal representation yields more accurate results and avoids the angle ambiguities due to singularity of rotation matrix derived from minimal representations.

The time variation of the quaternion $q(t)$ that represents the attitude of the body subjected to the angular velocity ${}^{i}\omega = [\omega_x \quad \omega_y \quad \omega_z]^T$ is derived considering the rotation $\Delta q(t)$ that occurs in the time interval Δt, according to the quaternion composition:

$$q(t + \Delta t) = q(t) \otimes \Delta q(t),$$

which, thanks to a linearization due to the small time interval Δt, leads to the kinematic relation [19]

$$\dot{q} = \frac{1}{2}\, q \otimes \omega^q. \tag{3.9}$$

The quaternion derivative expression (3.9) results more handy if in the matrix form

$$\dot{q} = \frac{1}{2}\, Q\, \omega, \tag{3.10}$$

where Q is a matrix representation of the quaternion q:

$$Q \doteq \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix} \tag{3.11}$$

3.3.1 The Process Model

From the attitude differential time law, it is clear that equation 3.9 describes a non-linear dynamic system; an effective representation is thus adopted, the continuous-time *state space* description.

The state vector is composed of seven state variables: the three elements of the angular rate ω and the four terms of the quaternion q:

$$[x] = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \\ q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}. \tag{3.12}$$

The state transition function of a continuous-time autonomous dynamic system, i.e. not forced,

$$\dot{x}(t) = f\left(x(t)\right), \tag{3.13}$$

formalized by Yun et al. in [11], is a non-linear combination of the state variables defined in (3.12), due to the dynamics described in (3.10). The dynamic evolution

of $[x_1 \quad x_2 \quad x_3]^T = [\omega_x \quad \omega_y \quad \omega_z]^T$ is characterized by a low predictability, since the free motion of the limbs is arbitrary; Kim et al. in [20] propose a relation between $\dot{x}$ and x which is based on the limits in magnitude and bandwidth that a power spectral density (PSD) function has. This is modeled through a first order system having a time constant τ related to the bandwidth of the system:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = -\frac{1}{\tau} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{3.14}$$

where τ is also referred to as the decorrelation time constant of the Gauss-Markov (GM) model [21]. The remaining part of the state equation is the explicit form of 3.10,

$$\begin{bmatrix} \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{3.15}$$

$$= \frac{1}{2} \begin{bmatrix} -x_1x_5 - x_2x_6 - x_3x_7 \\ x_1x_4 - x_2x_7 + x_3x_6 \\ x_1x_7 + x_2x_4 - x_3x_5 \\ -x_1x_6 + x_2x_5 + x_3x_4 \end{bmatrix}. \tag{3.16}$$

The whole system is depicted in figure 3.5 as presented by Marins in [10]. The quaternion normalization step is fundamental, but can be implemented in simulation framework so as not to increase further the complexity of the state equations.

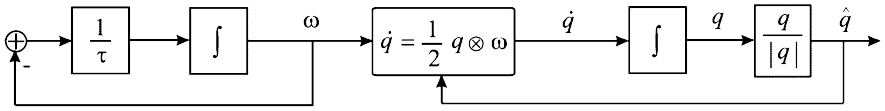

Figure 3.5: Process model block scheme.

3.3.2 The Measurement Model

The measurement model establishes a relation between the measurement data and the state variables. Since IMU sensors provide the local values of angular velocity and linear acceleration, the model to be implemented must account for an expression of the aforementioned physical quantities that is a combination of the state (3.12).

The general, non-linear function of a continuous-time autonomous dynamic system is expressed as

$$z(t) = h\left(x(t)\right),$$

and the output variables, for the i^{th} link are set to be

$$[z] = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \\ z_7 \end{bmatrix} = \begin{bmatrix} {}^i\omega_x \\ {}^i\omega_y \\ {}^i\omega_z \\ {}^ia_x + {}^i_0\mathcal{R}_{1,3}\ g \\ {}^ia_y + {}^i_0\mathcal{R}_{2,3}\ g \\ {}^ia_z + {}^i_0\mathcal{R}_{3,3}\ g \\ \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \end{bmatrix}. \tag{3.17}$$

Equations $[z_1 \quad z_2 \quad z_3]^T$ are referred to the angular velocity expression in (3.6). Equations $[z_4 \quad z_5 \quad z_6]^T$ represent the three components of the acceleration derived in (3.7) plus a contribution due to the projection of the gravity acceleration onto the local frame ${}^i\mathcal{F}$ to be discussed in the next section. The kinematic acceleration measured in local frame is the sum of three components: the tangential acceleration $a_t = {}^i\dot{\omega} \times \overline{r}_i$, the centripetal acceleration $a_c = {}^i\omega \times {}^i\omega \times \overline{r}_i$, illustrated in figure 3.6, and the acceleration of the frame origin, $\ddot{O}_{i-1}$. The angular velocity operator $\omega\times$ can be also expressed in matrix form by means of the skew-symmetric matrix $S(\omega)$, defined as

$$S(\omega) = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix};$$

the same notation can be applied to $\dot{\omega}\times$, $S(\dot{\omega})$. The rotation matrix $\mathcal{R}$ appearing in the output equations is calculated after 2.9 and is function of the state variables $[x_4 \quad x_5 \quad x_6 \quad x_7]^T$.

The output equation z_7 is a constraint on the norm of the quaternion that constitutes the state vector part $[x_4 \quad x_5 \quad x_6 \quad x_7]^T$, introduced both for improving the quaternion convergence and for augmenting the output vector dimension in order to avoid lack of observability issues. If a MARG sensor is used instead of an

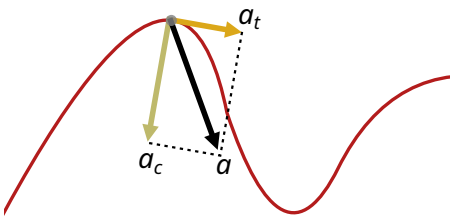

Figure 3.6: Components of the acceleration.

IMU, i.e. also the magnetic field components $\overline{m} = [m_x \quad m_y \quad m_z]^T$ are measured, the three further output equations are defined as

$${}^im = {}^i_0\mathcal{R}\ {}^0m.$$

3.3.3 Gravity Compensation

As previously introduced, the acceleration measurement model (3.17) must take into account the gravity acceleration components in the local frame, that are, by the way, the most consistent part in acceleration magnitude. The gravity vector, expressed in global base frame ${}^0\mathcal{F}$ (figure 3.4) has component only along z_0 direction, $\overline{g} = [0 \quad 0 \quad -g]^T$, where g is the gravity acceleration value that, although the Earth's gravitational field is not constant around the globe, can be considered equal to $g = 9{,}81\, m/s^2$. The gravity contribution on the acceleration measured in frame ${}^i\mathcal{F}$ can thus be expressed by

$$ {}^i a_g = {}^i_0\mathcal{R}\ \overline{g}. \tag{3.18}$$

Chapter 4

The Filter

Often, when dealing with dynamic systems, the interest of the study involves the state variables monitoring; some examples could concern the control of the system or the evaluation of the time evolution of the state as the most relevant source of information of the system. However, in general, only the output variables are the physical quantities that can be measured, thus only an estimate of the state could be available. If this is the case, a *state observer* is an effective tool to be implemented. The system, however, must satisfy *observability* conditions, related somehow to the output capability to affect the state variables; the observability of a dynamic system, linear in general, is assessed by evaluating the rank of the *observability matrix*, defined as:

$$M_o = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix} \tag{4.1}$$

where A and C are the state matrix and the output matrix of a linear time-invariant (LTI) dynamic system

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) \end{cases} .$$

The system is said to be *observable* if and only if the rank of M_o is equal to the system order [22], i.e. the number of states:

$$\text{rank}(M_o) = n.$$

When properly designed, a state observer can provide an estimate of the state $\hat{x}$ that converges to the real state; such a dynamic system is referred to as *asymptotic* state observer, that leads to:

$$\lim_{t \to \infty} |\hat{x}(t) - x(t)| = 0.$$

An insight on the observability conditions for non-linear systems is addressed later in the chapter.

4.1 The Kalman Filter Framework

An asymptotic state observer guarantees its performance as long as the state $x(t)$, the input $u(t)$, and the output $y(t)$ are deterministic, i.e not affected by any disturbance in general. But, since every real measurement data collection is unavoidably characterized noise corruption, the state estimate ought to be assessed within a *probabilistic* estimator framework. Let's consider a discrete-time, LTI system:

$$\begin{cases} x(k+1) &= Ax(k) + Bu(k) + v_p(k) \\ z(k) &= Cx(k) + v_m(k) \end{cases} \qquad k = 1,2,\dots \tag{4.2}$$

The terms v_p and v_m represent the zero-mean additive white noise:

- *process noise* v_p follows a normal distribution with zero-mean value

  $$E[v_p(k)] = 0, \quad \forall k$$

  and variance matrix Q,

  $$E[v_p(k_1)v_p(k_2)^T] \neq 0 \Leftrightarrow k_1 \neq k_2$$

- *measurement noise* v_m follows a normal distribution with zero-mean value

  $$E[v_m(k)] = 0, \quad \forall k$$

  and variance matrix R,

  $$E[v_m(k_1)v_m(k_2)^T] \neq 0 \Leftrightarrow k_1 \neq k_2.$$

The random variables $v_p(k), v_m(k)$ are uncorrelated, $E[v_p(k_1)v_m(k_2)^T] = [0], \forall k_1, k_2$. The introduction of random variables in the state and output equations leads to define that $x(k)$ and $y(k)$ are random variables too.

Under the assumptions outlined above, a recursive filtering technique can be defined:

$$\mathcal{KF} : \begin{cases} K(k) &= P(k)C^T[CP(k)C^T + R]^{-1} \\ P_0(k) &= [I_n - K(k)C]P(k) \\ \hat{z}(k|k-1) &= C\hat{x}(k|k-1) \\ e(k) &= z(k) - \hat{z}(k|k-1) \\ \hat{x}(k|k) &= \hat{x}(k|k-1) + K(k)e(k) \\ P(k+1) &= AP_0(k)A^T + Q \\ \hat{x}(k+1|k) &= A\hat{x}(k|k) + Bu(k) \end{cases} . \tag{4.3}$$

Such a procedure, known as *Kalman Filtering* is named after Rudolf E. Kálmán, who described it in 1960. With reference to (4.3), the scheme in figure 4.1 is defined; the two fundamental parts of the algorithm are the *prediction* and the *correction*. The state transition equations address the *a-priori* estimate, only based on the dynamic model of the system, while the measurement equations, instead, are in charge of updating the estimate, *a-posteriori*. This innovation due to measurements is magnified by the Kalman gain factor $K(k)$, calculated as of the state covariance matrix $P(k)$, which is recursively computed with a *difference Riccati equation* (DRE). Under the assumptions of Gaussian distribution of the noise and linearity in the system modeling, the Kalman filter provides an optimal estimate of the state [23].

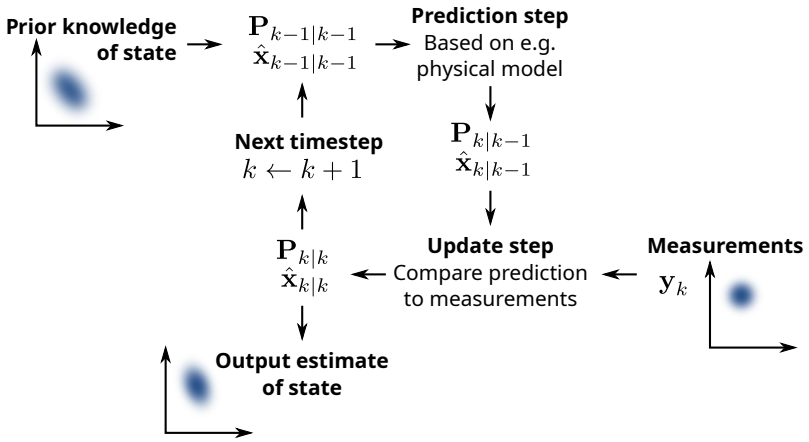

Figure 4.1: Scheme of the estimation steps of a Kalman filter.

Before starting the recursive procedure, an estimate of the initial state $\hat{x}(k=1)$ and its covariance matrix $P(1) = \text{var}[\hat{x}(1)]$ are determined, and their accuracy remarkably affects the convergence time of the algorithm [24].

4.2 The Extended Kalman Filter

The Kalman filter is a powerful tool for estimating the state of a dynamic linear system in presence of measurements or modeling uncertainties, in addition to providing an optimal linear estimate. However, LTI systems represent a limited range of system models; in fact, a large variety of estimation applications concerns non-linear systems, modeled as:

$$\begin{cases} x(k+1) &= f(x(k), u(k)) + v_p(k) \\ z(k) &= h(x(k)) + v_m(k) \end{cases} \qquad k = 1,2,\dots \tag{4.4}$$

in which the state and the measurement equations are defined as functions that combine the state variables in a general, non-linear, relation instead of the linear state-space matrix representation. In all such cases, an extension of the Kalman filter can be formulated, by considering as state and measurement matrices the *jacobian* of the state and measurement functions $f\left(x(k), u(k)\right)$, $h\left(x(k), u(k)\right)$ with respect to variables, evaluated in the predicted state;

$$\hat{F}(k|k-1) = \left.\frac{\partial f(x,u,k)}{\partial x}\right|_{x=\hat{x}(k|k-1)} \tag{4.5}$$

$$\hat{H}(k|k-1) = \left.\frac{\partial h(x,u,k)}{\partial x}\right|_{x=\hat{x}(k|k-1)} \tag{4.6}$$

this formulation derives from the approximation method based on the Taylor series expansion [25]. The filter algorithm becomes:

$$\mathcal{EKF}: \begin{cases} K(k) &= P(k)\hat{H}(k)^T[\hat{H}(k)P(k)\hat{H}(k)^T + R]^{-1} \\ P_0(k) &= [I_n - K(k)\hat{H}(k)]P(k) \\ \hat{z}(k|k-1) &= h\left(\hat{x}(k|k-1)\right) \\ e(k) &= z(k) - \hat{z}(k|k-1) \\ \hat{x}(k|k) &= \hat{x}(k|k-1) + K(k)e(k) \\ P(k+1) &= \hat{F}(k)P_0(k)\hat{F}(k)^T + Q \\ \hat{x}(k+1|k) &= f\left(\hat{x}(k|k)\right) \end{cases} . \tag{4.7}$$

Matrices $\hat{F}$, also called Φ in literature [11], and $\hat{H}$ have to be computed at each time step, leading the *extended Kalman Filter* (EKF) to be much more consuming than the linear version. In the EKF algorithm the Kalman gain $K(k)$ and the state covariance matrix $P(k)$ are calculated on the base of the linearized matrices $\hat{F}, \hat{H}$, while the state and the measurement updates are determined by means of the non-linear equations describing the system (4.4).

4.2.1 Model Linearization

The equations that describe the continuous-time non-linear dynamic system presented in section 3.3 are linearized differentiating (3.14),(3.16),(3.17) with respect to the state variables $x = [x_1 \quad \dots \quad x_7]^T$, yielding to a symbolic definition of matrices $\hat{F}, \hat{H}$, which are evaluated in the current predicted state $\hat{x}(k|k-1)$ at each iteration step. Those matrices are used in the filter algorithm (4.3) in place of A, C to compute the Kalman gain $K(k)$ and the state covariance matrix update $P(k+1)$. The filtered state $\hat{x}(k|k)$ is then updated on the basis of the linearized dynamics of the system.

It may be the case that, albeit the system dynamics is expressed by nonlinear equations, the measurement model is linear in the state variables. If this occurrence verifies, the measurement matrix F is directly employed in the filter. In [10], in order to simplify the filter implementation, an external quaternion convergence algorithm is exploited to obtain an estimate of the quaternion, leading the measurement matrix F to coincide with the identity matrix I; accelerometer and magnetometer measurements are combined in an error function Q that depends on the rotation matrix R expressed in terms of rotation quaternion, which is minimized by means of the Gauss-Newton iterative algorithm. In particular,

$$^{E}Q = \varepsilon^{T}\varepsilon = (^{E}y_1 - M\,^{B}y_0)^{T}(^{E}y_1 - M\,^{B}y_0)$$

where $^{E}y_1$ is the 6×1 vector with the gravity and magnetic field values in the Earth's reference frame, $^{B}y_0$ is the 6×1 vector measurements performed in body frame; M is a rotation matrix defined as

$$M = \begin{bmatrix} R & 0 \\ 0 & R \end{bmatrix}$$

4.2.2 System Discretization Method

The filter algorithm, in order to be implemented, needs a description of dynamic system in discrete time, as pointed out by the formulation in (4.4). This representation requirement essentially derives from the fact that measurement data are acquired with a time interval T_s called *sample time* and because the algorithm itself is iteratively computed.

The relation that links the continuous time state matrix A and the state matrix of the discrete time representation of a dynamic system is expressed as

$$A_d = e^{T_s A}$$

where the term $e^{T_s A}$ is a *matrix exponential* defined as

$$e^{M} = \sum_{k=0}^{\infty} \frac{1}{k!} M^k = I + M + \frac{M^2}{2} + \frac{M^3}{6} + \dots \tag{4.8}$$

Finding a method to approximate the exponential matrix is an active field of research yet, but among the most widespread, the Padé approximants are the most used [26]. Although the simple truncation of the Taylor series in (4.8), $A_d \approx I + T_s A$ is itself an effective approximation, in this work the Tustin approximation approach is considered, since it guarantees more accuracy:

$$F_d = e^{T_s \hat{F}} \approx \left(I + \frac{1}{2} T_s \hat{F}\right)\left(I - \frac{1}{2} T_s \hat{F}\right)^{-1}$$

where $\hat{F}$ is the state linearized matrix in (4.5).

The non-linear dynamics of the system, described in subsection 3.3.1, is discretized exploiting the *forward Euler* method; the derivative of the state vector is approximated by means of the incremental ratio:

$$\dot{x}(t) = \lim_{\Delta t \to \infty} \frac{x(t+\Delta t) - x(t)}{\Delta t} \approx \frac{x(t+\Delta t) - x(t)}{\Delta t}$$

where, since $\tau = T_s$ and the discretized time $t = k\Delta t$, the state can be written as $x(t) = x(k)$, $x(t+\Delta t) = x(k+1)$ for notation simplicity. The discrete-time description of the system dynamics then results expressed by the finite-difference equations:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} (k+1) = \begin{bmatrix} (1 - T_s/\tau)x_1(k) \\ (1 - T_s/\tau)x_2(k) \\ (1 - T_s/\tau)x_3(k) \\ x_4(k) + T_s/2\,(-x_1(k)x_5(k) - x_2(k)x_6(k) - x_3(k)x_7(k)) \\ x_5(k) + T_s/2\,(x_1(k)x_4(k) - x_2(k)x_7(k) + x_3(k)x_6(k)) \\ x_6(k) + T_s/2\,(x_1(k)x_7(k) + x_2(k)x_4(k) - x_3(k)x_5(k)) \\ x_7(k) + T_s/2\,(-x_1(k)x_6(k) + x_2(k)x_5(k) + x_3(k)x_4(k)) \end{bmatrix} \tag{4.9}$$

4.3 Modeling The Noise

As one of the theoretical assumptions made to derive the filter algorithm, the noise entering the system is considered to follow a *gaussian* distribution, with zero mean and known variance; the process and the measurement ought to be uncorrelated in different time instants.

$$p(v_p) \sim WN(0, Q)$$
$$p(v_m) \sim WN(0, R)$$

Since the noise is all but known, only guesses can be made on it. Starting from its gaussian nature.

4.3.1 Process noise

The uncertainties on the process model are represented by the additive noise term $v_p(k)$; it also accounts for the unmodeled dynamics of the system. Sometimes it may be convenient to implement a simplified model of the system in order to reduce the complexity of the calculations, by taking into account the uncertainty by proper selecting the variance matrix Q.

Since the differential relation relating the quaternion evolution with the angular velocity is a deterministic kinematic function, no noise occurs in the quaternion state variables, i.e.

$$Q = \begin{bmatrix} q_{11} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & q_{22} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & q_{33} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.10}$$

Although in [11, 27] methods to obtain the process noise variance matrix from calculations are investigated, the values of the matrix (4.10) have to be manually adjusted in order to obtain the best performances of the algorithm, especially for a system whose states are characterized by low predictability.

4.3.2 Measurement Noise

When dealing with data acquisition systems, the main source of uncertainties in the measurements comes from the sensor; in this regard the datasheets of such sensors provide a reasonable value for the measurements variance. The variance matrix R can be non-diagonal, in case for example that the sensor axes are not uncoupled, but this information is usually available in the instrument documentation.

$$R = \begin{bmatrix} r_{11} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & r_{22} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & r_{33} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & r_{44} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & r_{55} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & r_{66} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & r_{77} \end{bmatrix} \tag{4.11}$$

The measurement variance matrix R considered for this work is diagonal; the elements r_{11}, r_{22}, r_{33} are referred to the gyroscope readings w_x, w_y, w_z, while elements r_{44}, r_{55}, r_{66} represents the variance of accelerometer measurements, a_x, a_y, a_z. The element q_{77} is instead a term representing the fictitious variance relative to the norm of the quaternion to be estimated $\hat{q}$: it would be virtually zero, so a very tiny quantity is considered.

4.3.3 Calibration Setup Routine

A critical point in the estimation procedure of the position is the definition of the initial state $\hat{x}(0)$. An effective method for the determination of the initial attitude

is to consider the system at still; in fact, in static conditions the only acceleration sensed by the IMU is due to the gravity. Since the bare measurements of the accelerometer are not sufficient to fully characterize the three angles which define the attitude in space [3], a widely used approach is to refer to as initial pose the body position assumed in figure4.2, also referred to as *T-pose* [5, 28, 29].

Figure 4.2: Calibration position.

In case a magnetometer sensor is available besides the IMU, the attitude in space can be fully determined by gravity and Earth's magnetic field readings. If a MARG sensor is considered instead of an IMU, the calibration procedure consist in performing circular trajectories in space in order to sense the direction of the Earth's magnetic field. Such a procedure is however reported on sensor datasheets.

4.4 Convergence Conditions

The extended Kalman filter is a robust algorithm, but it may be very sensitive to the occurrence of some conditions.

4.4.1 Linearity

The optimal solution provided by the Kalman filter is based on the assumption of linearity of the model relative to the system under test. The extended Kalman filter, however, operates in a local region of the domain where the non-linear state equation have been linearized; in the case that the model is highly nonlinear in the area around the linearization point, i.e. the state estimate, the result may be poor and the algorithm may not converge at all.

4.4.2 State Initialization

A significant impact on the convergence time, and, sometimes, on the convergence itself, is made by the determination of the initial predicted state, $\hat{x}(1|0)$. By choosing an initial guess as close as possible to the real, unknown, state, would dramatically decrease the convergence time; a great care indeed must be taken to the initial estimate, e.g. by exploiting different sensors and methods, like the one suggested in subsection 4.3.3. Along with the state estimate, the initialization of its covariance matrix $P(1)$ is also crucial; common practice is to define the state covariance matrix equal to the process noise variance.

4.4.3 Observability

As stated among the KF implementation conditions, the system is requested to be observable, i.e. the rank of the observability matrix must be equal to the order of the system, i.e. $\text{rank}(M_o) = n$, otherwise convergence is not guaranteed. Since for non-linear systems the state and, possibly, the measurement equations do not have a matrix representations, the observability of such a class of systems must be investigated exploiting different tools. As pointed out in [30] and thoroughly discussed in [31], the observability matrix of non-linear systems is addressed by evaluating the gradients of the Lie derivatives of the measurement function $h(x(k))$:

$$O(x) = \{\nabla \mathcal{L}^s_{f_i \dots f_j} h(x(k)) | i, j = 0, \dots, l; k = 1, \dots, m; s \in \mathbb{N}\}$$

where m is the dimension of the output vector, l is the dimension of the input vector, s is the order of the derivative.

4.4.4 Noise variance

The most immediate tool to adjust the filter performance are the variance matrices of the noise, relative to both the process model and the noise model. Sometimes, finding the best combination of parameters in terms of convergence and stability of the filtered state is a matter of variance matrices fine tuning.

It may be the case that the implemented measurement model leads to a rank deficiency of the observability matrix. This is in principle due to the fact that only six measurement equations are exploited to estimate seven parameters, although only six of them are linearly independent due to the constraint on the quaternion norm. To overcome this issue, a tri-axial magnetometer can be used in addition to the accelerometer and the gyroscope, adding in this way three new independent equations relating the attitude of the arm with the quaternion state variables. The

drawback would be an increment on the measurement linearized matrix $\hat{H}$, that needs to be inverted in the algorithm loop.

Chapter 5

Simulation Setup

The system has been implemented in Matlab 2020b for the simulation; the script is reported in Appendix B.

A symbolic approach is pursued in order to the derive the analytical expression for the process model and the measurement model. Starting from these formulations, the linearized state and measurement matrices to be implemented in the filter are obtained by differentiation with respect to the state vector, and then evaluated in the current state estimate. The procedure, detailed in section 4.2, is performed by substituting the predicted state at time instant k in the analytical expressions previously discussed.

5.1 Data Generation

The kinematic analysis of the limb, handled in subsection 3.2.2, is performed considering a DH model of the kinematic chain; the angular rate and acceleration data are indeed derived in the local reference frame defined in such model and illustrated in figure 3.4. The JCS defined by the ISB, however, does not coincide with the DH local frames, as can be seen in figure 2.6. In order to guarantee the compatibility of both the coordinate systems, three transformation matrices are introduced, such that the simulated acceleration and angular velocities are rotated into the frames related to the JCS:

$$ {}^{JCS}_{DH}\mathcal{R}_2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} $$

$$ {}^{JCS}_{DH}\mathcal{R}_5 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix} $$

$$ {}^{JCS}_{DH}\mathcal{R}_7 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix} $$

The coordinate transformations expressed by matrices ${}^{JCS}_{DH}\mathcal{R}_2$, ${}^{JCS}_{DH}\mathcal{R}_5$, ${}^{JCS}_{DH}\mathcal{R}_7$ are referred to the T-pose calibration position described in subsection 4.3.3.

5.1.1 Parameters Definition

The simulation is performed collected $N = 7000$ samples with a sampling frequency $f_s = 100\,Hz$, leading to a sampling period $T_s = 0.01\,s$ and thus a total duration of the experiment $T_f = 70\,s$. The time constant of the model τ should be tuned on the base of real measurements. However, since only simulation data are available, it is set to a reasonable but arbitrary value, i.e. $\tau = 0.2\,s$.

5.1.2 Limb Dimensions

The three parameters that characterize the arm are the length of the clavicle from the sternoclavicular joint up to the scapula l_c, the length of the upper arm from the shoulder up to the elbow l_u, and the length of the forearm from the elbow up to the wrist l_f. In table 5.1 are reported the assumed value, compatible with the dimensions of a human arm.

	length [m]
l_c	0.2
l_u	0.3
l_f	0.3

Table 5.1: Caption

5.1.3 Joint Velocities

The angular velocity ω_i, defined for each joint, is characterized by an impulse-shape so that it causes a smooth trapezoidal profile in the angle generated θ_i, and, at the same time, gives rise to a smooth angular acceleration α_i shape, according to the function

$$ \omega(t) = \theta \frac{2}{\sqrt{2\pi}} e^{-2(t-t_1)^2} - \theta \frac{2}{\sqrt{2\pi}} e^{-2(t-t_2)^2} $$

whose time law is shown in figure 5.1.

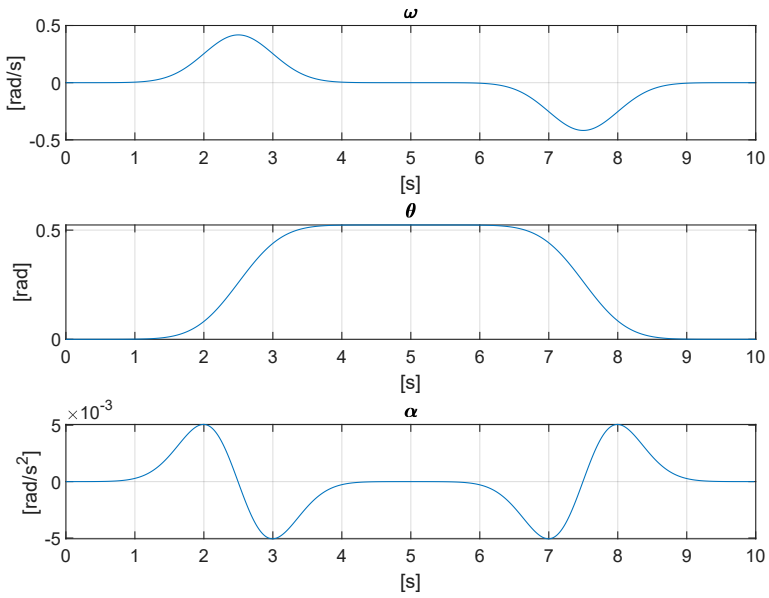

Figure 5.1: Time profile of the angular velocity, angle and acceleration.

Angular Acceleration

In the formulation of the measurement model, the tangential acceleration contribution depends on the angular acceleration $\dot{\omega}$; though, deriving an estimate of $\dot{\omega}$ from the estimate of the angular rate $\hat{\omega}$ would lead to a remarkable performance degradation of the algorithm. In order to accurately estimate the angular acceleration, the state of the system should be augmented, including the components of $\dot{\omega}$ in local frame. For this reason, in the simulation the angular velocity is considered to be zero, introducing a dramatic but necessary approximation.

$$\begin{bmatrix} \hat{\dot{\omega}}_x \\ \hat{\dot{\omega}}_y \\ \hat{\dot{\omega}}_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

5.1.4 Initial Conditions

The initial posture is determined on the base of the T-pose reference, i.e. the starting position of the DH model. The initial state estimate $\hat{x}(1)$ is however corrupted by random noise from the normal distribution having variance R.

The initial state covariance matrix $P(1)$ is assumed to be equal to the process noise variance Q.

5.1.5 Variance Matrices

The variance matrices of process noise and measurement noise are a fundamental tool to settle the performance of the filter. However, in many situations a compromise between the tracking and convergence achievement is difficult to be obtained, so for each task the aforementioned parameters are specified. This behaviour may depend on the lack of observability conditions on the system; although an observability test based of the linearized model is implemented at each iteration step, a deeper analysis can be addressed by assessing the non-linear observability as defined in subsection 4.4.

5.1.6 Simulation Data

In the following, figures 5.2, 5.3, 5.4, 5.5, 5.6, 5.7 represent the measurement datasets that are used for estimating the attitude of the limb by means of the extended Kalman filter algorithm. In particular, the depicted case correspond to the protraction/retraction of the clavicle by an angle $q_2 = \pi/6$.

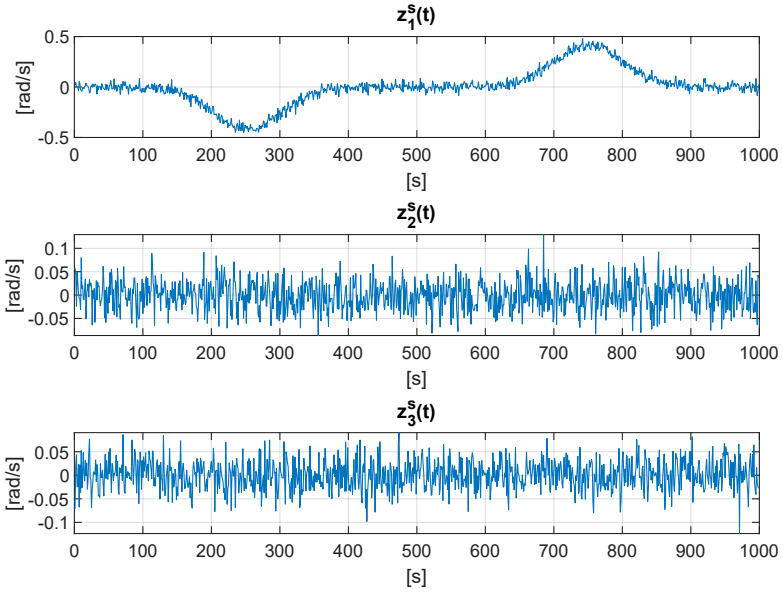

Figure 5.2: Gyroscope measurement data relative to the shoulder.

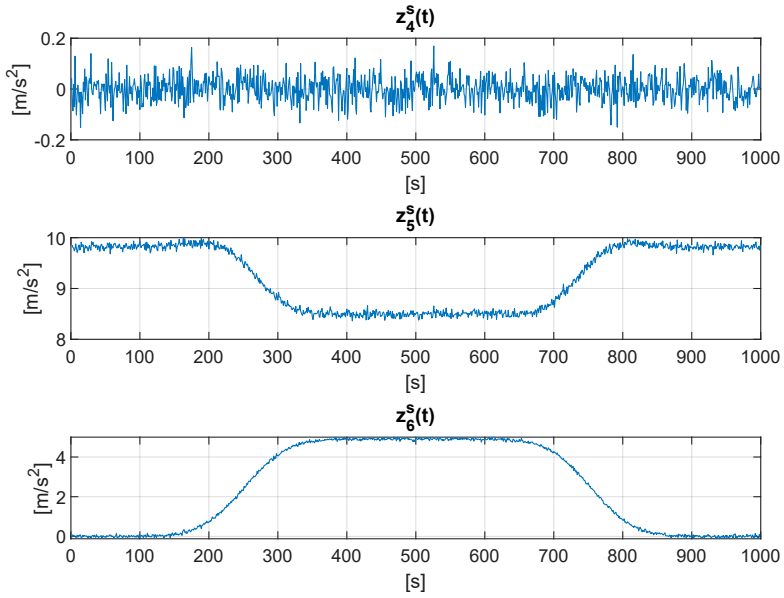

Figure 5.3: Accelerometer measurement data relative to the shoulder.

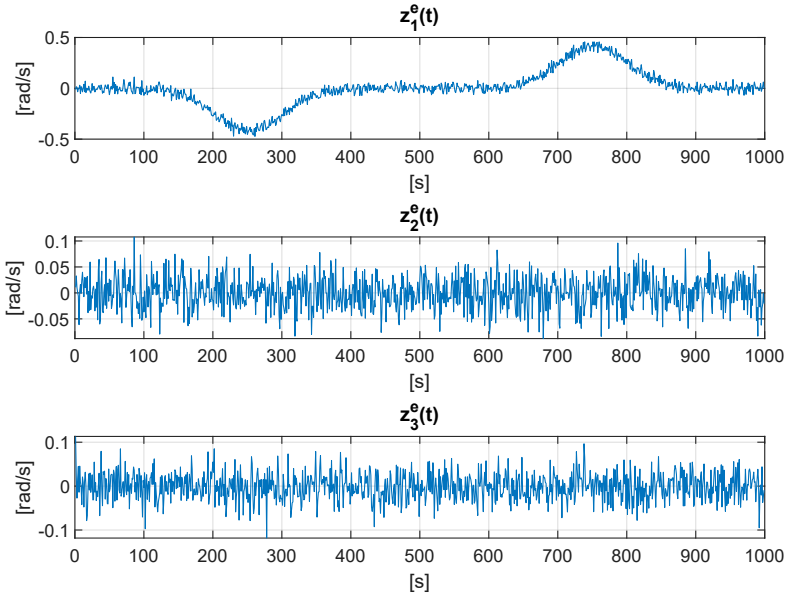

Figure 5.4: Gyroscope measurement data relative to the elbow.

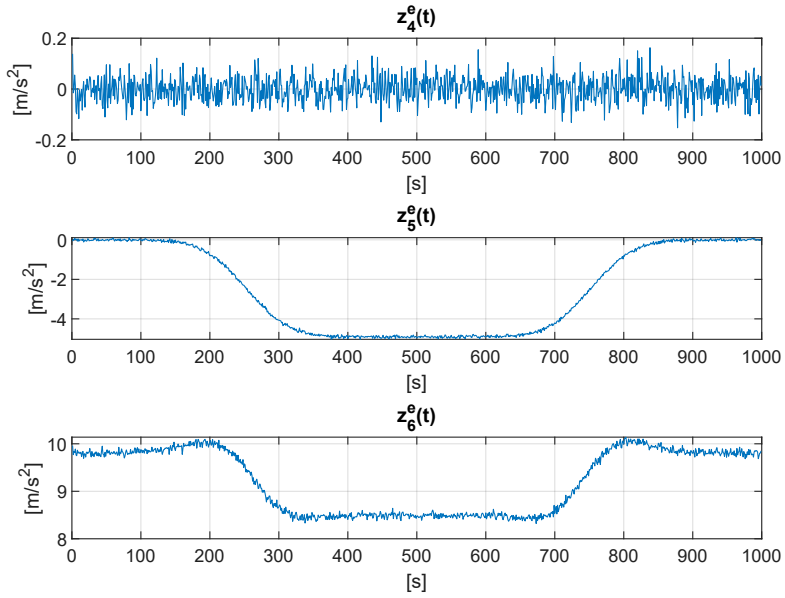

Figure 5.5: Accelerometer measurement data relative to the elbow.

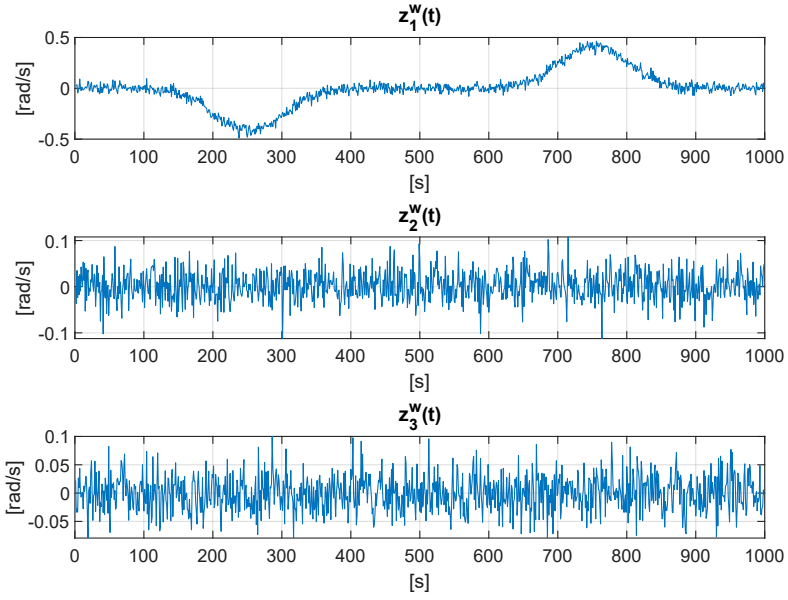

Figure 5.6: Gyroscope measurement data relative to the wrist.

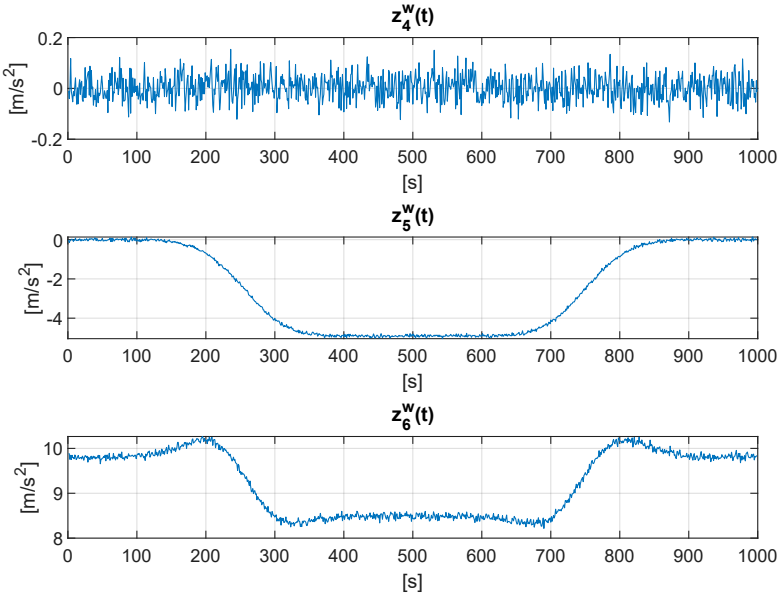

Figure 5.7: Accelerometer measurement data relative to the wrist.

Chapter 6

Results

This chapter collects the results of the tests performed on the filter. As stated in subsection 5.1.5, the conditions and the particular choice of parameters associated to the presented achievements is reported. In order to figure out the quality of the estimate, root mean square errors (RMSE) values relative to the output z of each test evaluation are reported.

6.1 Static Results

In this section the static performance is evaluated; the measurement dataset refers to a still T-pose posture, as depicted in figure 4.2.

	link 1	link 2	link 3
ω_x	0.0318	0.0323	0.0324
ω_y	0.0317	0.0323	0.0319
ω_z	0.0322	0.0320	0.0320
a_x	0.5733	0.0529	0.0543
a_y	0.0533	0.0517	0.0516
a_z	0.0525	0.0497	0.0503

Table 6.1: RMSE of the output in static conditions.

6.2 Dynamic Results

In this section, the joint variables motions are testes separately. A description of the arm degrees of freedom is detailed in section 3.1.

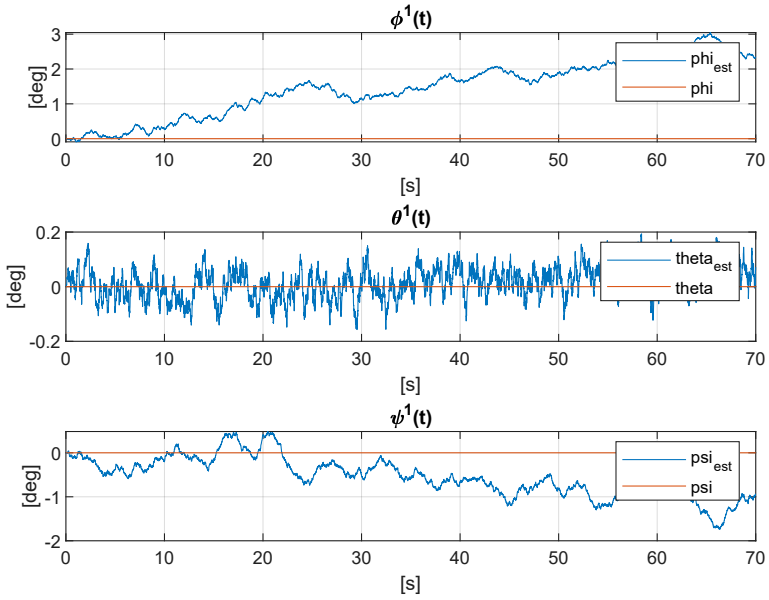

Figure 6.1: Attitude of link 1 in terms of RPY angles in static conditions.

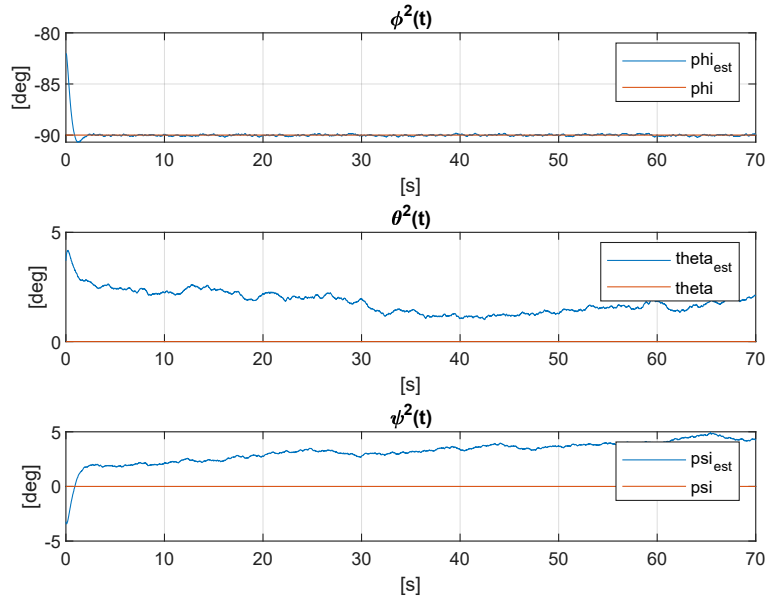

Figure 6.2: Attitude of link 2 in terms of RPY angles in static conditions.

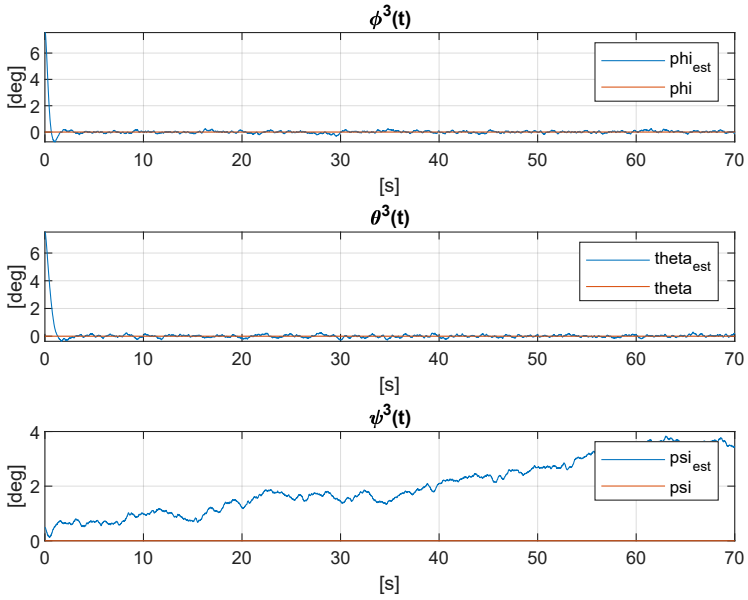

Figure 6.3: Attitude of link 3 in terms of RPY angles in static conditions.

	Variance $[rad^2/s^2]$
q_{11}	$1 \cdot 10^{-5}$
q_{22}	$1 \cdot 10^{-5}$
q_{33}	$1 \cdot 10^{-5}$

Table 6.2: Variance values of the process noise in static conditions.

6.2.1 Forearm Pronation/Supination

A $q_7 = \pi/6$ rotation with a smooth trapezoidal profile is imposed to the joint 7 of the kinematic chain, correspondent to the forearm pronation and successive supination. The resulting angle is shown in the local joint coordinate system.

The measurement variance matrix R for this experiment is the same as assumed in static case, section 6.1.

6.2.2 Forearm Flexion/Extension

A $q_6 = \pi/6$ rotation with a smooth trapezoidal profile is imposed to the joint 6 of the kinematic chain, correspondent to the forearm flexion and successive extension. The resulting angle is shown in the local joint coordinate system.

	Variance $[rad^2/s^2]$	Variance $[m^2/s^4]$	Variance [ ]
r_{11}	$1 \cdot 10^{-3}$		
r_{22}	$1 \cdot 10^{-3}$		
r_{33}	$1 \cdot 10^{-3}$		
r_{44}		$2.5 \cdot 10^{-3}$	
r_{55}		$2.5 \cdot 10^{-3}$	
r_{66}		$2.5 \cdot 10^{-3}$	
r_{77}			$1 \cdot 10^{-12}$

Table 6.3: Variance values of the measurement noise in static conditions.

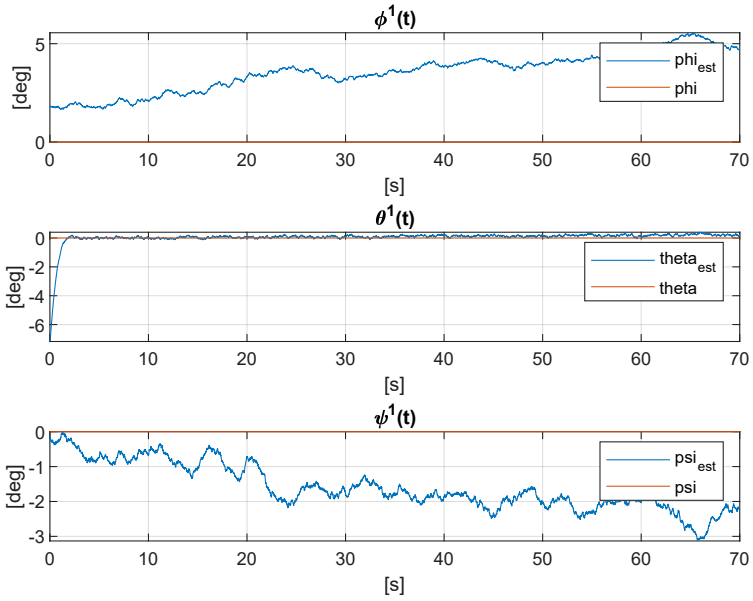

Figure 6.4: Attitude of link 1 in terms of RPY angles during forearm pronation/supination.

The measurement variance matrix R for this experiment is the same as assumed in static case, section 6.1, while the process variance matrix Q is the same as in the case of forearm pronation/supination.

6.2.3 Upper Arm Rotation

A $q_5 = \pi/6$ rotation with a smooth trapezoidal profile is imposed to the joint 5 of the kinematic chain, correspondent to the upper arm rotation. The resulting angle is shown in the local joint coordinate system.

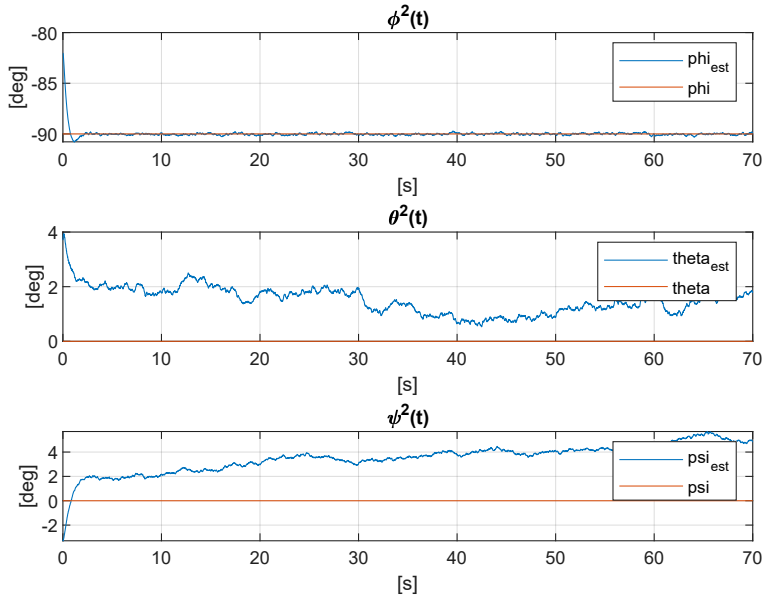

Figure 6.5: Attitude of link 2 in terms of RPY angles during forearm pronation/supination.

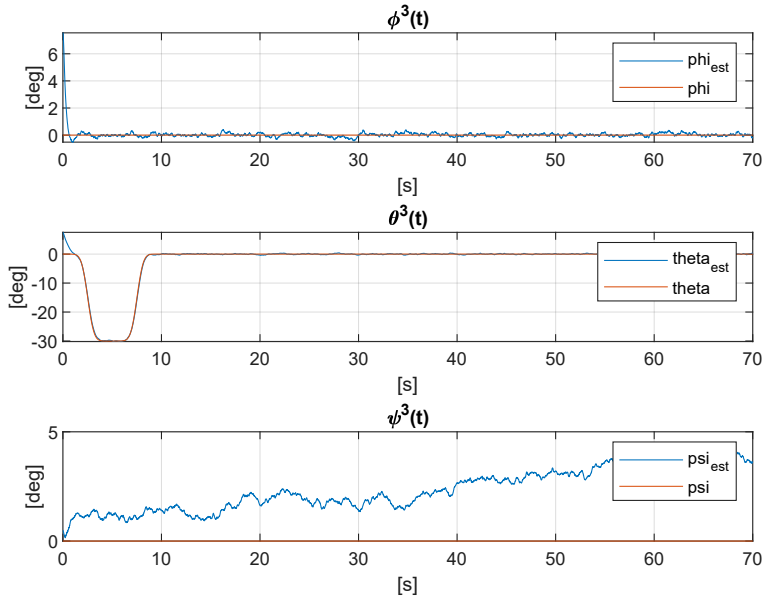

Figure 6.6: Attitude of link 3 in terms of RPY angles during forearm pronation/supination.

	link 1	link 2	link 3
ω_x	0.0332	0.0347	0.0350
ω_y	0.0331	0.0348	0.0364
ω_z	0.0336	0.0346	0.0346
a_x	0.6361	0.0539	0.0583
a_y	0.0551	0.0527	0.0525
a_z	0.0527	0.0497	0.0506

Table 6.4: RMSE of the output relative to forearm pronation/supination.

	Variance $[rad^2/s^2]$
q_{11}	$1 \cdot 10^{-4}$
q_{22}	$1 \cdot 10^{-4}$
q_{33}	$1 \cdot 10^{-4}$

Table 6.5: Variance values of the process noise relative to forearm pronation/supination.

The measurement variance matrix R for this experiment is the same as assumed in static case, section 6.1, while the process variance matrix Q is the same as in the case of forearm pronation/supination.

6.2.4 Upper Arm Abduction/Adduction

A $q_4 = \pi/6$ rotation with a smooth trapezoidal profile is imposed to the joint 4 of the kinematic chain, correspondent to the upper arm abduction and successive adduction. The resulting angle is shown in the local joint coordinate system.

The measurement variance matrix R for this experiment is the same as assumed in static case, section 6.1, while the process variance matrix Q is the same as in the case of forearm pronation/supination.

6.2.5 Upper Arm Flexion/Extension

A $q_3 = \pi/6$ rotation with a smooth trapezoidal profile is imposed to the joint 3 of the kinematic chain, correspondent to the upper arm flexion and successive extension. The resulting angle is shown in the local joint coordinate system.

The measurement variance matrix R for this experiment is the same as assumed in static case, section 6.1, while the process variance matrix Q is the same as in the case of forearm pronation/supination.

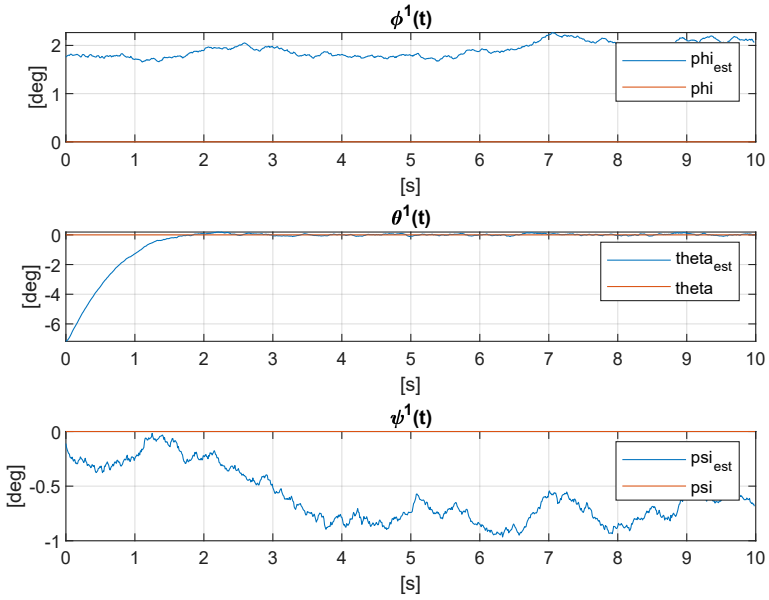

Figure 6.7: Attitude of link 1 in terms of RPY angles during forearm flexion/extension.

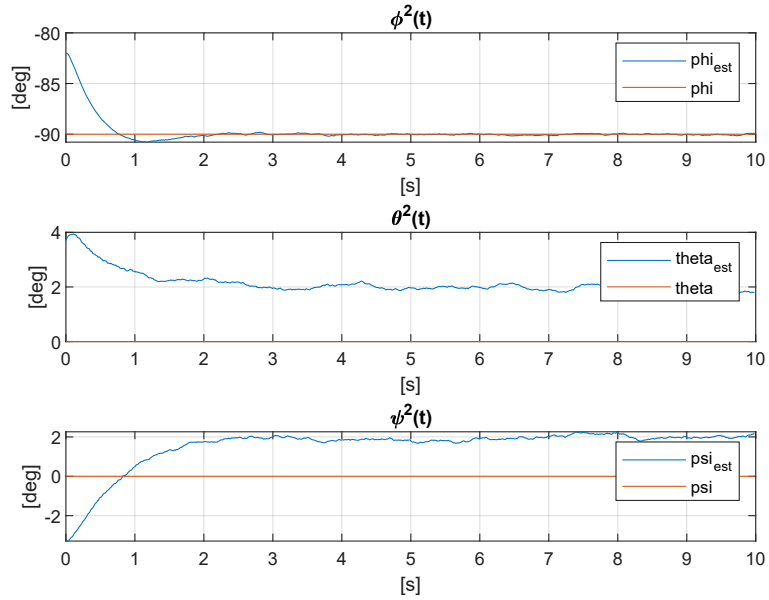

Figure 6.8: Attitude of link 2 in terms of RPY angles during forearm flexion/extension.

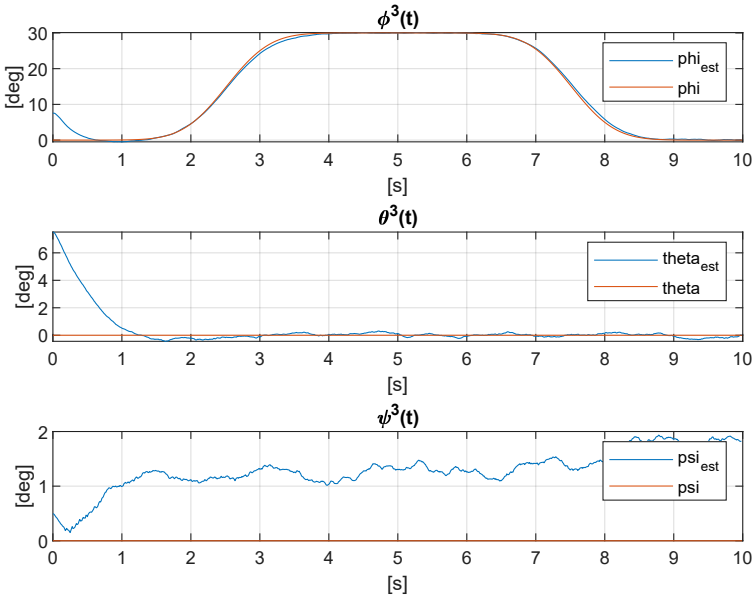

Figure 6.9: Attitude of link 3 in terms of RPY angles during forearm flexion/extension.

	link 1	link 2	link 3
ω_x	0.0336	0.0347	0.0487
ω_y	0.0342	0.0335	0.0345
ω_z	0.0330	0.0340	0.0338
a_x	0.3334	0.0608	0.0626
a_y	0.0506	0.0537	0.0887
a_z	0.0587	0.0494	0.0882

Table 6.6: RMSE of the output relative to forearm flexion/extension.

6.2.6 Clavicle Elevation/Depression

A $q_2 = \pi/6$ rotation with a smooth trapezoidal profile is imposed to the joint 2 of the kinematic chain, correspondent to the clavicle elevation and successive depression. The resulting angle is shown in the local joint coordinate system.

The measurement variance matrix R for this experiment is the same as assumed in static case, section 6.1.

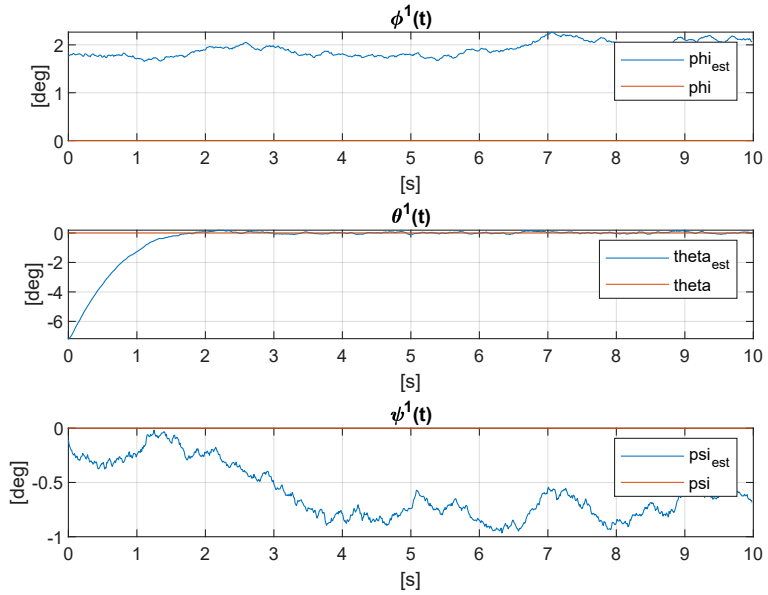

Figure 6.10: Attitude of link 1 in terms of RPY angles during upper arm rotation.

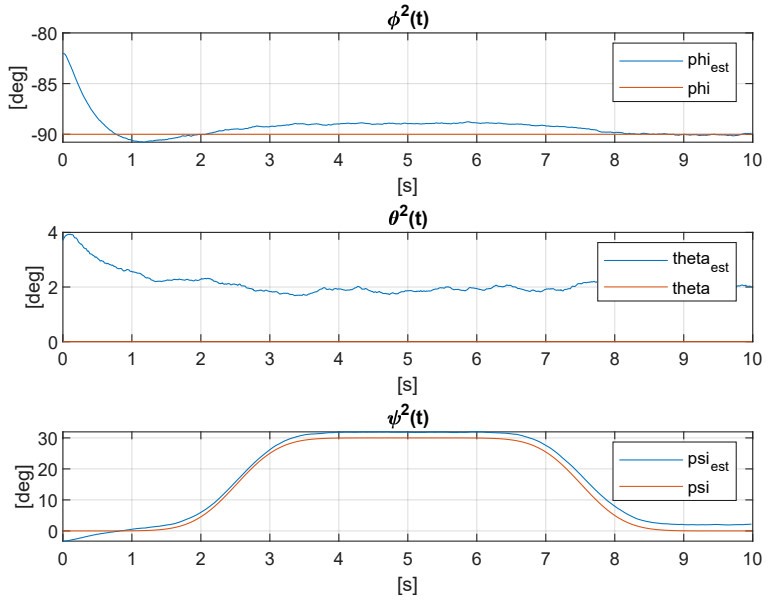

Figure 6.11: Attitude of link 2 in terms of RPY angles during upper arm rotation.

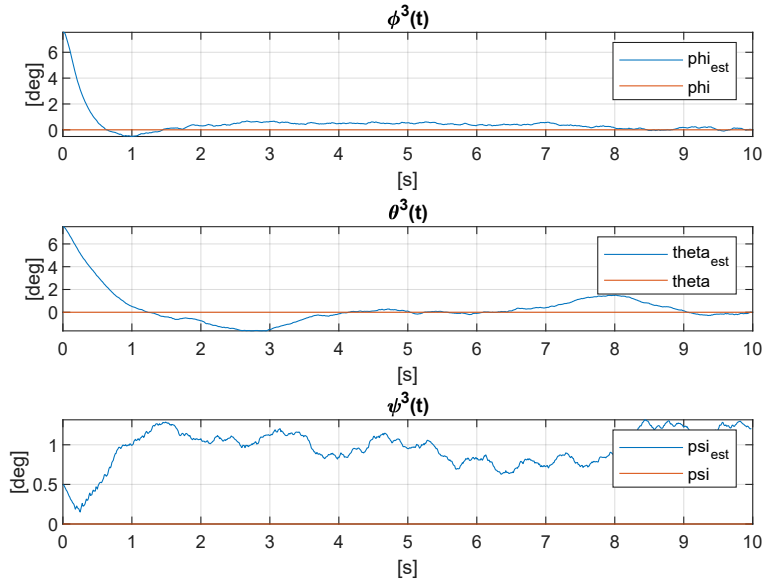

Figure 6.12: Attitude of link 3 in terms of RPY angles during upper arm rotation.

	link 1	link 2	link 3
ω_x	0.0336	0.0347	0.0344
ω_y	0.0342	0.0467	0.0353
ω_z	0.0330	0.0339	0.0337
a_x	0.3334	0.0907	0.0806
a_y	0.0506	0.0538	0.0537
a_z	0.0587	0.0525	0.0540

Table 6.7: RMSE of the output relative to upper arm rotation.

6.2.7 Clavicle Protraction/Retraction

A $q_1 = \pi/6$ rotation with a smooth trapezoidal profile is imposed to the joint 1 of the kinematic chain, correspondent to the clavicle protraction and successive retraction. The resulting angle is shown in the local joint coordinate system.

The measurement variance matrix R for this experiment is the same as assumed in static case, section 6.1.

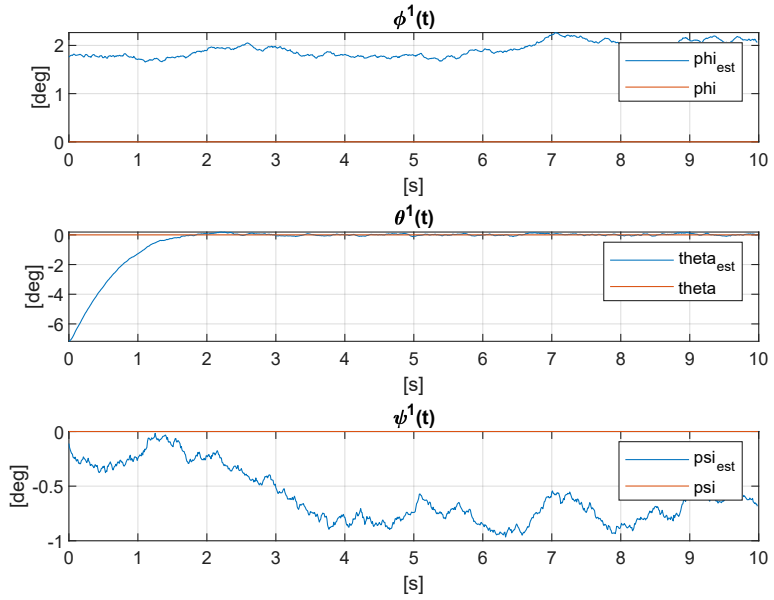

Figure 6.13: Attitude of link 1 in terms of RPY angles during upper arm abduction/adduction.

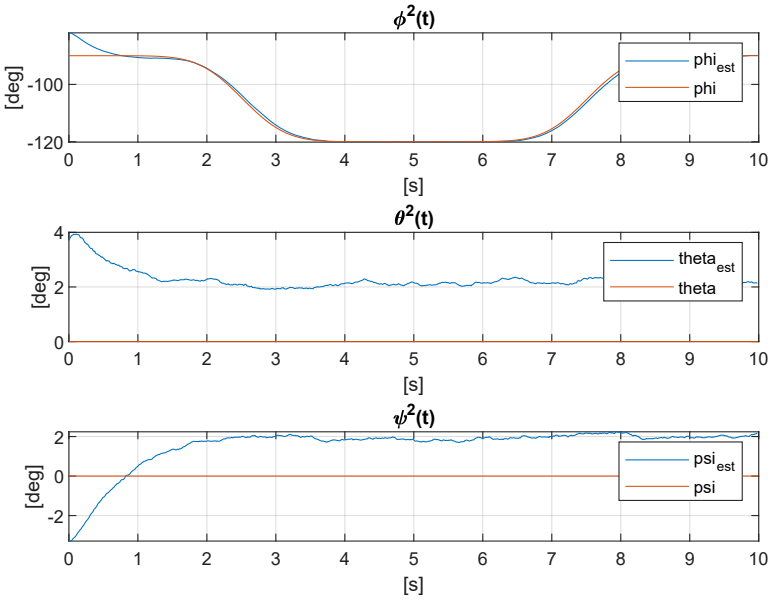

Figure 6.14: Attitude of link 2 in terms of RPY angles during upper arm abduction/adduction.

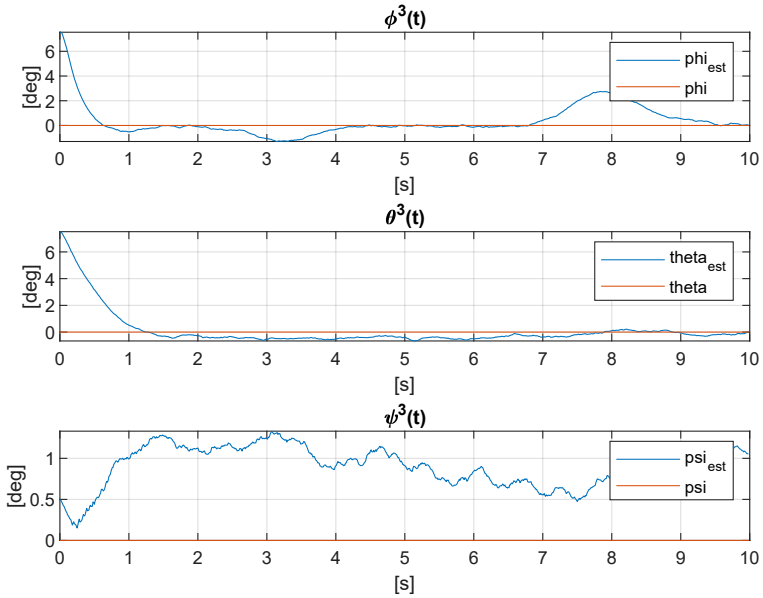

Figure 6.15: Attitude of link 3 in terms of RPY angles during upper arm abduction/adduction.

	link 1	link 2	link 3
ω_x	0.0336	0.0492	0.0356
ω_y	0.0342	0.0336	0.0345
ω_z	0.0330	0.0338	0.0337
a_x	0.3334	0.0607	0.0620
a_y	0.0506	0.1073	0.1018
a_z	0.0587	0.0986	0.1568

Table 6.8: RMSE of the output relative to upper arm abduction/adduction.

	link 1	link 2	link 3
ω_x	0.0376	0.0431	0.0436
ω_y	0.0382	0.0411	0.0444
ω_z	0.0370	0.0456	0.0450
a_x	0.3274	0.0838	0.1066
a_y	0.0506	0.0615	0.0618
a_z		0.0506	0.0615
0.0618			

Table 6.9: RMSE of the output relative to upper arm flexion/extension.

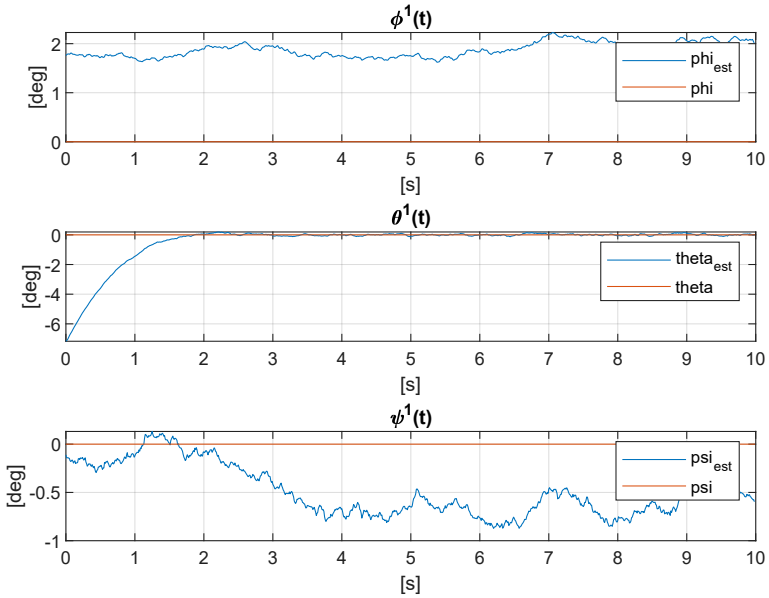

Figure 6.16: Attitude of link 1 in terms of RPY angles during upper arm flexion/extension.

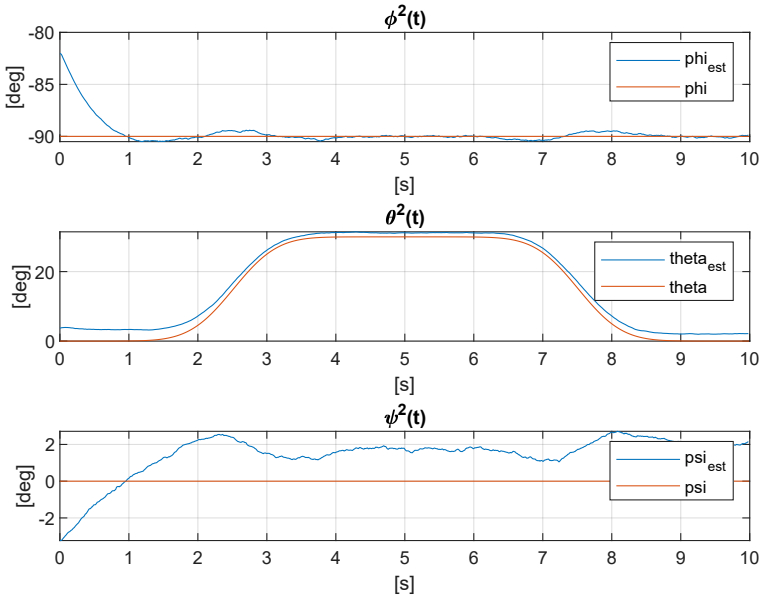

Figure 6.17: Attitude of link 2 in terms of RPY angles during upper arm flexion/extension.

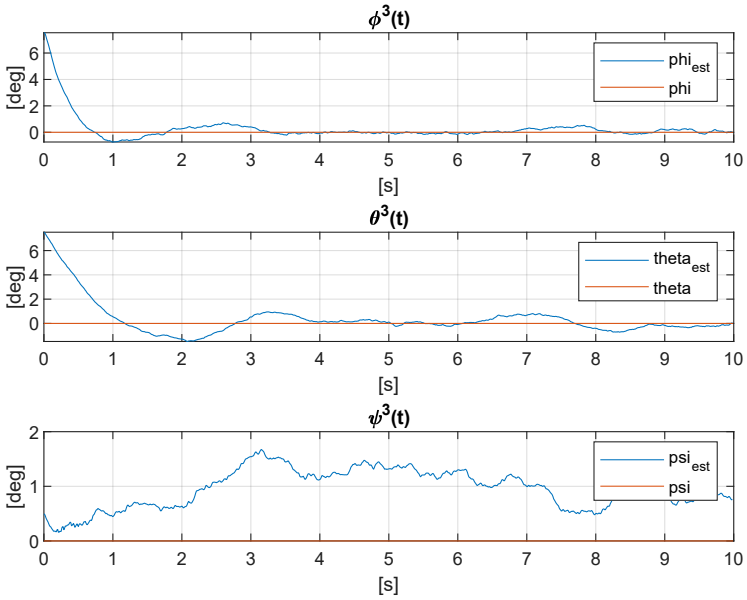

Figure 6.18: Attitude of link 3 in terms of RPY angles during upper arm flexion/extension.

	Variance $[rad^2/s^2]$
q_{11}	$1 \cdot 10^{-3}$
q_{22}	$1 \cdot 10^{-3}$
q_{33}	$1 \cdot 10^{-3}$

Table 6.10: Variance values of the process noise relative to upper arm flexion/extension.

	link 1	link 2	link 3
ω_x	0.1736	0.1390	0.1289
ω_y	0.0571	0.0357	0.0351
ω_z	0.0343	0.0531	0.0313
a_x	0.2717	0.2329	0.3390
a_y	0.6558	1.4712	0.9958
a_z	2.1379	0.5091	0.4432

Table 6.11: RMSE of the output relative to clavicle elevation/depression.

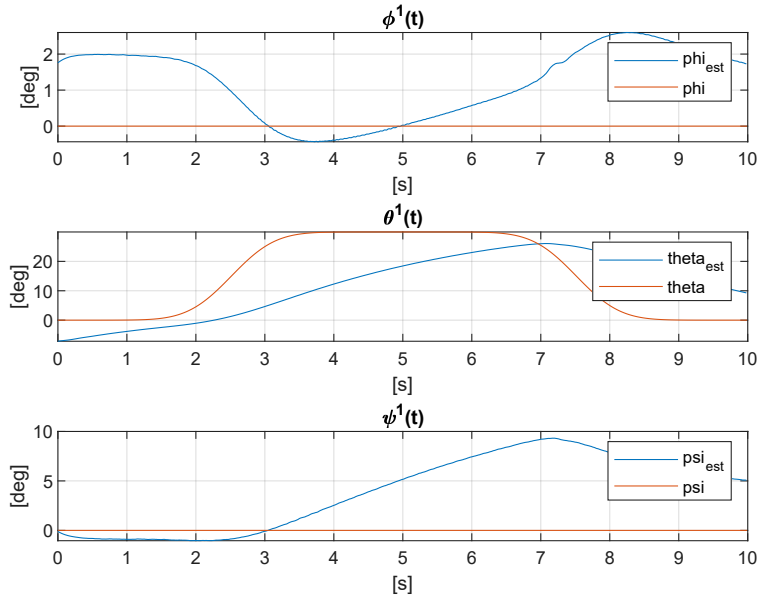

Figure 6.19: Attitude of link 1 in terms of RPY angles during clavicle elevation/depression.

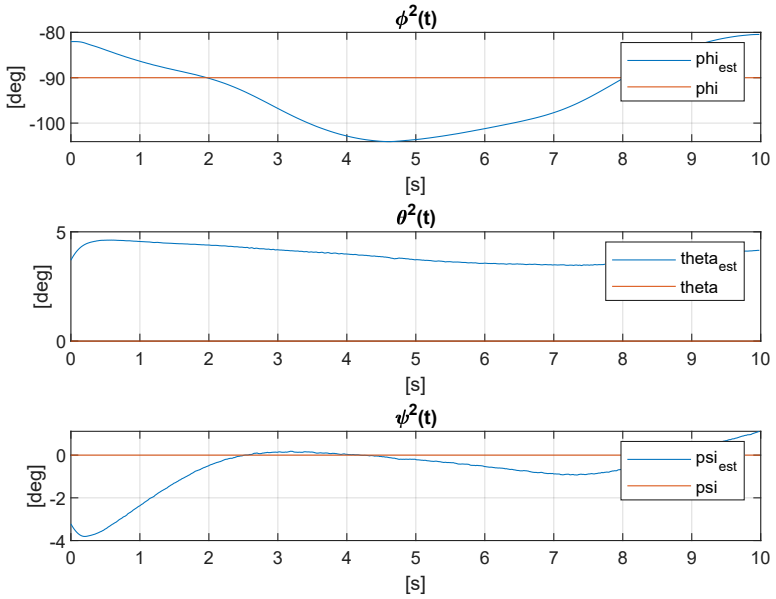

Figure 6.20: Attitude of link 2 in terms of RPY angles during clavicle elevation/depression.

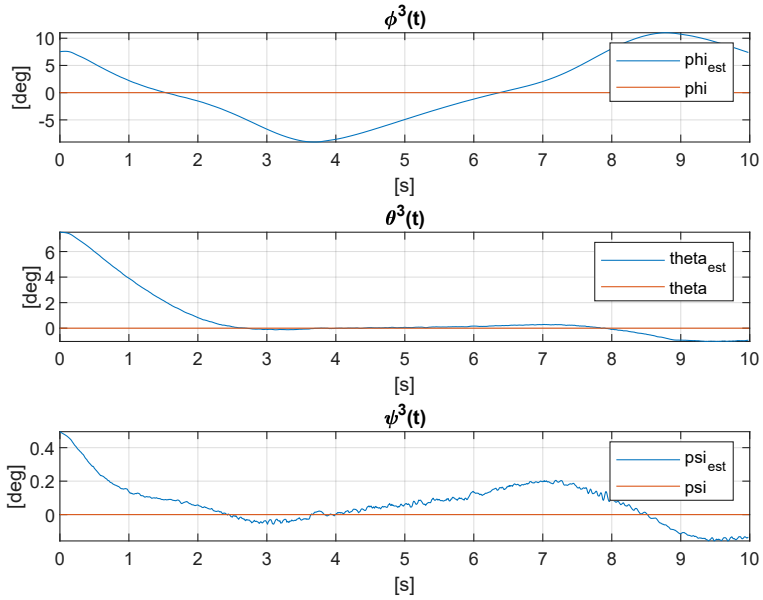

Figure 6.21: Attitude of link 3 in terms of RPY angles during clavicle elevation/depression.

	Variance $[rad^2/s^2]$
q_{11}	$5 \cdot 10^{-7}$
q_{22}	$5 \cdot 10^{-7}$
q_{33}	$5 \cdot 10^{-7}$

Table 6.12: Variance values of the process noise relative to clavicle elevation/depression.

	link 1	link 2	link 3
ω_x	0.0319	0.0335	0.0378
ω_y	0.1690	0.0324	0.0380
ω_z	0.0312	0.1442	0.1419
a_x	0.3935	0.1978	0.2892
a_y	0.0515	0.2076	0.2035
a_z	0.2252	0.0509	0.0535

Table 6.13: RMSE of the output relative to clavicle protraction/retraction.

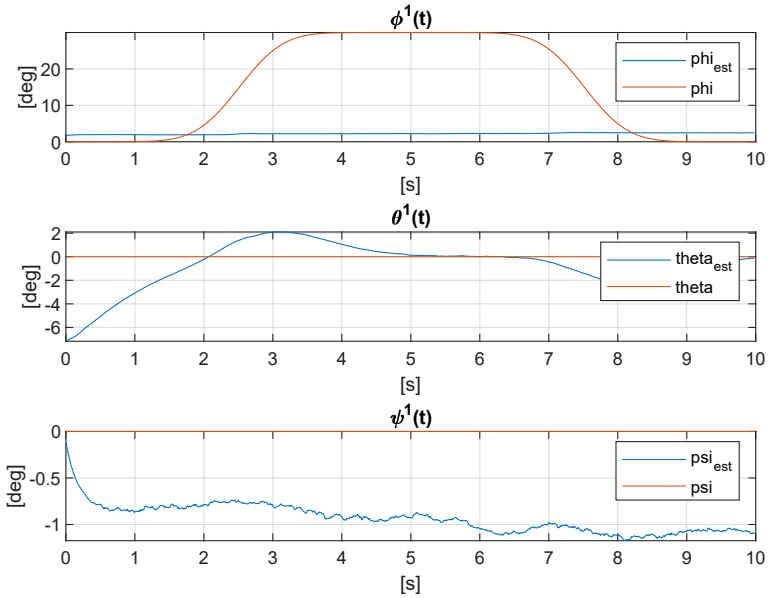

Figure 6.22: Attitude of link 1 in terms of RPY angles during clavicle protraction/retraction.

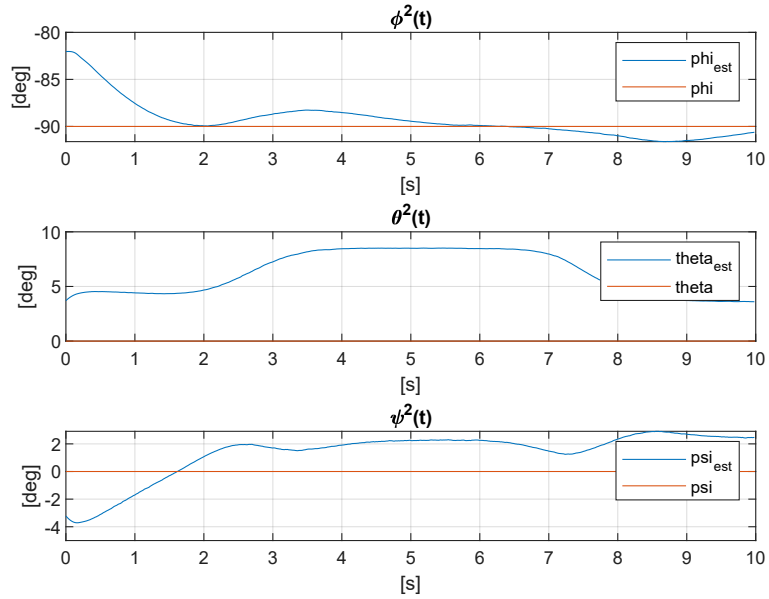

Figure 6.23: Attitude of link 2 in terms of RPY angles during clavicle protraction/retraction.

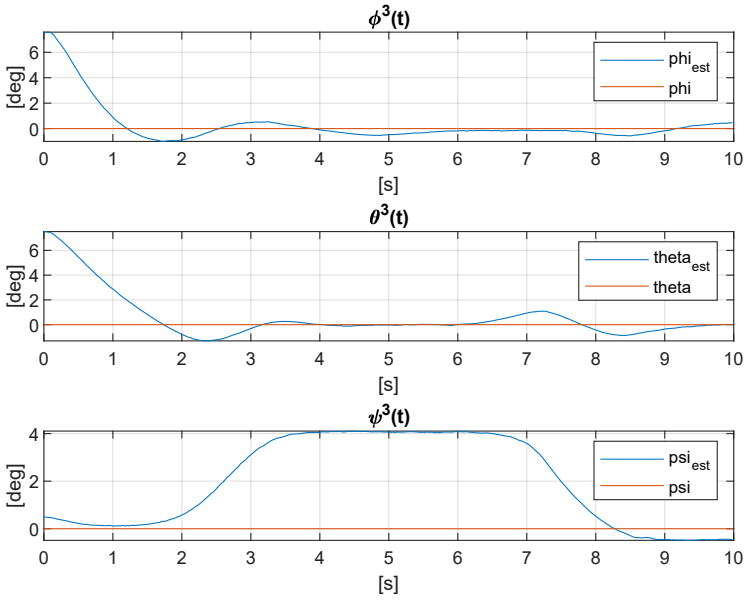

Figure 6.24: Attitude of link 3 in terms of RPY angles during clavicle protraction/retraction.

	Variance $[rad^2/s^2]$
q_{11}	$1 \cdot 10^{-6}$
q_{22}	$1 \cdot 10^{-6}$
q_{33}	$1 \cdot 10^{-6}$

Table 6.14: Variance values of the process noise relative to clavicle protraction/retraction.

Chapter 7

Conclusions

7.1 Analysis of Results

With respect to the estimate of the orientation shown in chapter 6, the static reconstruction turns out to be quite reliable in terms of angle error; in a $70\,s$ simulation the error is always confined within $5°$ (figures 6.1, 6.2, 6.3). For what concerns the tracking error, it can be observed that the performance decays from the wrist up to the shoulder; in fact, while the pronation/supination and the flexion/extension movements of the forearm are estimated with an error always lower than $5°$, figures 6.4, 6.5, 6.6, 6.7, 6.8, 6.9, the elevation/depression and the protraction/retraction of the clavicle estimate procedure leads to very poor result, due to an increasing complexity of the kinematic model. Moreover, considering the estimate of the angular acceleration equal to zero as discussed in subsection 5.1.3, the acceleration measurements do not provide relevant information for the reconstruction of that kind of motion.

The choice of comparing only the RMSE relative to the output is due to the fact that the states of the kinematic model used to generate the simulation data and the states of the model implemented in the filter do not coincide; they are indeed expressed in different reference frames. Moreover, the kinematic model used for the data generation is composed of seven joints, each having a single degree of freedom out of the three axis, so only one state active at a time (quaternion state are excluded from the discussion). Conversely, the model used for estimation is composed of three links, each having two or three relative degrees of freedom, so the full state is active, or near.

7.2 Future Developments

The limits of the proposed model mainly concern the choice of the state variables representing the system. In fact, the tangential component of the acceleration, i.e. the angular acceleration $\dot{\omega}$, has been neglected from the model implemented in the filter in the absence of a reliable estimate. An improvement in the model is the implementation of the angular acceleration components as state variables, in order to retrieve an accurate estimate of such quantities; Peppoloni et al. implemented a similar model in [17]. Another enrichment of the model may be represented by the consideration in the state vector also the bias of the gyroscope, that for real applications is a overwhelming side effect deriving from the application of such sensors, as done for example by Sabatini in [21]. The enhancement of the state leads however to a remarkable increment on the complexity of the algorithm, since it involves the enlargement of the system matrices, that need to be inverted at each sample time for the calculation of the Kalman gain matrix $K(k)$.

The constant bias in the gyroscope measurements can be rejected by using a magnetometer sensor in addition to the already employed accelerometer and gyroscope, as pointed out by Truppa et al. in [32].

Appendix A

Forward Kinematics

Hereunder is reported the homogeneous transformation matrix from frame ${}^{7}\mathcal{F}$ to frame ${}^{0}\mathcal{F}$ derived with the Denavit-Hartenberg convention. Note that matrix ${}^{0}_{7}\mathcal{A}$ is expressed as a function of the seven joint variables q_i.

$$
{}^{0}_{7}\mathcal{A}(q_1, q_2, q_3, q_4, q_5, q_6, q_7) = \begin{bmatrix} R_{1,1} & R_{1,2} & R_{1,3} & d_1 \\ R_{2,1} & R_{2,2} & R_{2,3} & d_2 \\ R_{3,1} & R_{3,2} & R_{3,3} & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$R_{1,1} = sq_6(sq_5(cq_4(sq_1sq_3 - cq_1cq_2cq_3) + cq_1sq_2sq_4) + cq_5(cq_3sq_1 + cq_1cq_2sq_3)) +$$
$$- cq_6(sq_6(sq_4(sq_1sq_3 - cq_1cq_2cq_3) - cq_1cq_4sq_2) + cq_6(cq_5(cq_4(sq_1sq_3 +$$
$$- cq_1cq_2cq_3) + cq_1sq_2sq_4) - sq_5(cq_3sq_1 + cq_1cq_2sq_3)))$$
$$R_{1,2} = cq_6(sq_5(cq_4(sq_1sq_3 - cq_1cq_2cq_3) + cq_1sq_2sq_4) + cq_5(cq_3sq_1 + cq_1cq_2sq_3)) +$$
$$+ sq_6(sq_6(sq_4(sq_1sq_3 - cq_1cq_2cq_3) - cq_1cq_4sq_2) + cq_6(cq_5(cq_4(sq_1sq_3 +$$
$$- cq_1cq_2cq_3) + cq_1sq_2sq_4) - sq_5(cq_3sq_1 + cq_1cq_2sq_3)))$$
$$R_{1,3} = sq_6(cq_5(cq_4(sq_1sq_3 - cq_1cq_2cq_3) + cq_1sq_2sq_4) - sq_5(cq_3sq_1 + cq_1cq_2sq_3)) +$$
$$- cq_6(sq_4(sq_1sq_3 - cq_1cq_2cq_3) - cq_1cq_4sq_2)$$
$$d_1 = l_ccq_1cq_2 - l_u(sq_4(sq_1sq_3 - cq_1cq_2cq_3) - cq_1cq_4sq_2) - l_f(cq_6(sq_4(sq_1sq_3 - cq_1cq_2cq_3) +$$
$$- cq_1cq_4sq_2) - sq_6(cq_5(cq_4(sq_1sq_3 - cq_1cq_2cq_3) + cq_1sq_2sq_4) +$$
$$- sq_5(cq_3sq_1 + cq_1cq_2sq_3)))$$
$$R_{2,1} = cq_6(sq_6(sq_4(cq_1sq_3 + cq_2cq_3sq_1) + cq_4sq_1sq_2) + cq_6(cq_5(cq_4(cq_1sq_3 + cq_2cq_3sq_1) +$$
$$- sq_1sq_2sq_4) - sq_5(cq_1cq_3 - cq_2sq_1sq_3))) - sq_6(sq_5(cq_4(cq_1sq_3 +$$
$$+ cq_2cq_3sq_1) - sq_1sq_2sq_4) + cq_5(cq_1cq_3 - cq_2sq_1sq_3))$$
$$R_{2,2} = -cq_6(sq_5(cq_4(cq_1sq_3 + cq_2cq_3sq_1) - sq_1sq_2sq_4) + cq_5(cq_1cq_3 - cq_2sq_1sq_3)) +$$
$$- sq_6(sq_6(sq_4(cq_1sq_3 + cq_2cq_3sq_1) + cq_4sq_1sq_2) + cq_6(cq_5(cq_4(cq_1sq_3 +$$
$$+ cq_2cq_3sq_1) - sq_1sq_2sq_4) - sq_5(cq_1cq_3 - cq_2sq_1sq_3)))$$
$$R_{2,3} = cq_6(sq_4(cq_1sq_3 + cq_2cq_3sq_1) + cq_4sq_1sq_2) - sq_6(cq_5(cq_4(cq_1sq_3 + cq_2cq_3sq_1) +$$
$$- sq_1sq_2sq_4) - sq_5(cq_1cq_3 - cq_2sq_1sq_3))$$
$$d_2 = l_f(cq_6(sq_4(cq_1sq_3 + cq_2cq_3sq_1) + cq_4sq_1sq_2) - sq_6(cq_5(cq_4(cq_1sq_3 + cq_2cq_3sq_1) +$$
$$- sq_1sq_2sq_4) - sq_5(cq_1cq_3 - cq_2sq_1sq_3))) + l_u(sq_4(cq_1sq_3 + cq_2cq_3sq_1) +$$
$$+ cq_4sq_1sq_2) + l_ccq_2sq_1$$
$$R_{3,1} = cq_6(cq_6(cq_5(cq_2sq_4 + cq_3cq_4sq_2) + sq_2sq_3sq_5) - sq_6(cq_2cq_4 - cq_3sq_2sq_4)) +$$
$$- sq_6(sq_5(cq_2sq_4 + cq_3cq_4sq_2) - cq_5sq_2sq_3)$$
$$R_{3,2} = -cq_6(sq_5(cq_2sq_4 + cq_3cq_4sq_2) - cq_5sq_2sq_3) - sq_6(cq_6(cq_5(cq_2sq_4 + cq_3cq_4sq_2) +$$
$$+ sq_2sq_3sq_5) - sq_6(cq_2cq_4 - cq_3sq_2sq_4))$$
$$R_{3,3} = -sq_6(cq_5(cq_2sq_4 + cq_3cq_4sq_2) + sq_2sq_3sq_5) - cq_6(cq_2cq_4 - cq_3sq_2sq_4)$$
$$d_3 = l_csq_2 - l_f(sq_6(cq_5(cq_2sq_4 + cq_3cq_4sq_2) + sq_2sq_3sq_5) + cq_6(cq_2cq_4 - cq_3sq_2sq_4)) +$$
$$- l_u(cq_2cq_4 - cq_3sq_2sq_4)$$

Jacobian of the state function

$$\hat{F} = \begin{bmatrix} -1/\tau & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1/\tau & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1/\tau & 0 & 0 & 0 & 0 \\ -q_1/2 & -q_2/2 & -q_3/2 & 0 & -\omega_x/2 & -\omega_y/2 & -\omega_z/2 \\ q_0/2 & -q_3/2 & q_2/2 & \omega_x/2 & 0 & \omega_z/2 & -\omega_y/2 \\ q_3/2 & q_0/2 & -q_1/2 & \omega_y/2 & -\omega_z/2 & 0 & \omega_x/2 \\ -q_2/2 & q_1/2 & q_0/2 & \omega_z/2 & \omega_y/2 & -\omega_x/2 & 0 \end{bmatrix}$$

Jacobian of the measurement function relative to link r_1

$$\hat{H} = \left[\begin{array}{ccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ r_y\omega_y + r_z\omega_z & r_y\omega_x - 2r_x\omega_y & r_z\omega_x - 2r_x\omega_z & -2gq_2 & 2gq_3 & -2gq_0 & 2gq_1 \\ r_x\omega_y - 2r_y\omega_x & r_x\omega_x + r_z\omega_z & r_z\omega_y - 2r_y\omega_z & 2gq_1 & 2gq_0 & 2gq_3 & 2gq_2 \\ r_x\omega_z - 2r_z\omega_x & r_y\omega_z - 2r_z\omega_y & r_x\omega_x + r_y\omega_y & 2gq_0 & -2gq_1 & -2gq_2 & 2gq_3 \\ 0 & 0 & 0 & q_0/n(q) & q_1/n(q) & q_2/n(q) & q_3/n(q) \end{array}\right]$$

where $n(q) = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}$

Appendix B

Matlab code

In this appendix is reported the code relative to the system implementation, divided into sections, each referred to a specific functionality.

B.1 Symbolic Definition Of The Model

In the following, the symbolic procedure to obtain an analytical expression of the system is listed. The function *quaternion product* is defined afterwards.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Tommaso Lovato     28/03/2024
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Symbolic procedure for the definition of the analytical formulation
% of the state function and its jacobian, and measurement function
% with the relative jacobian
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc
clear
close all

syms q0 q1 q2 q3 lx ly lz wx wy wz wxd wyd wzd tau g real
vel=sym('v',[3,1],'real'); acc=sym('v',[3,1],'real');

% the quaternion q=(q0,q1,q2,q3) represents the coordinate change
% from RF{m} to RF{0}, [so the vector rotation from {p}^m to {p'}^m.]
% The limb vector in mobile frame is expressed with
% quaternion p_m=(0,lx,ly,lz) up to sensor position.
% 0^p=q@p_m@q*

% intermediate quaternion
[pq0,pq1,pq2,pq3]=quaternion_product([0,lx,ly,lz],[q0,-q1,-q2,-q3]);
```

```
23
24 % position quaternion (position in base frame 0) p=
25 [~,P1,P2,P3]=quaternion_product([q0,q1,q2,q3],[pq0,pq1,pq2,pq3]);
26
27 % velocity expressed in mobile frame m, {v}={w}x{p}=S(w)*{p}
28 vel=[0,-wz,wy
29       wz,0,-wx
30       -wy,wx,0]*[lx,ly,lz]';
31
32 % acceleration expressed mobile frame m, {a}={w'}x{p}+{w}x({w}x{p})
33 acc=[0,-wzd,wyd
34       wzd,0,-wxd
35       -wyd,wxd,0]*[lx,ly,lz]'+[0,-wz,wy
36                                wz,0,-wx
37                                -wy,wx,0]*[vel(1),vel(2),vel(3)]';
38
39 % DCM rotation matrix associated with q
40 R=[q0^2+q1^2-q2^2-q3^2  2*(q1*q2-q0*q3)      2*(q1*q3+q0*q2)
41    2*(q1*q2+q0*q3)      q0^2-q1^2+q2^2-q3^2  2*(q2*q3-q0*q1)
42    2*(q1*q3-q0*q2)      2*(q2*q3+q0*q1)      q0^2-q1^2-q2^2+q3^2];
43
44 % gravity vector
45 g_vec=[0;0;g];
46
47 % process model x'=f(x,t)
48 f=[-wx/tau
49    -wy/tau
50    -wz/tau
51    (-q1*wx-q2*wy-q3*wz)/2
52    (q0*wx-q3*wy+q2*wz)/2
53    (q3*wx+q0*wy-q1*wz)/2
54    (q1*wy-q2*wx+q0*wz)/2];
55
56 % linearized process model x'=Fx
57 F=simplify(jacobian(f,[wx,wy,wz,q0,q1,q2,q3]));
58
59 % measurement model z=h(x,t)
60 h=[wx
61    wy
62    wz
63    acc+R'*g_vec
64    (q0^2+q1^2+q2^2+q3^2)^(1/2)];
65 h=simplify(h);
66
67 % linearized measurement model z=H(x)x
68 H=simplify(jacobian(h,[wx,wy,wz,q0,q1,q2,q3]));
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Tommaso Lovato     28/03/2024
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Definition of a quaternion product function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [t0,t1,t2,t3]=quaternion_product(q,r)

q0=q(1); q1=q(2); q2=q(3); q3=q(4);
r0=r(1); r1=r(2); r2=r(3); r3=r(4);

t0=(r0*q0-r1*q1-r2*q2-r3*q3);
t1=(r0*q1+r1*q0-r2*q3+r3*q2);
t2=(r0*q2+r1*q3+r2*q0-r3*q1);
t3=(r0*q3-r1*q2+r2*q1+r3*q0);

%t=[t0;t1;t2;t3];

end
```

B.2 Kinematic Model For Simulation Data Generation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Tommaso Lovato     28/03/2024
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 7 DoFs Kinematic Model for the generation of linear acceleration and
% angular velocity measurement data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc
clear
close all

rng default

Ts=1/100; %[s]
T_fin=10; %[s]
N=T_fin/Ts;
```

```
16 tvec=0:Ts:T_fin-Ts;
17
18 g=9.81; %[m/s^2]
19 tau=[0.2;0.2;0.2];
20 % gravity vector
21 g_vec=[0;0;g];
22
23 J=7; % number of joints
24 L=3; % number of links
25 n=7; % number of states
26
27 Rc=diag([1e-3,1e-3,1e-3,2.5e-3,2.5e-3,2.5e-3]); % measurement noise
       variance, gyro, acc
28
29 % Initial conditions and vectors allocation
30 w_joint=NaN(J,N);
31 x=NaN(n,N+1); s=x;
32 z=NaN(n,size(x,2)-1);
33 v_sens=NaN(size(z));
34 phi=NaN(J,N); theta=phi; psi=phi;
35 a_j=pt; a_t=pt; w_dot=pt;
36 Pos=cell(1,J);
37 At=Pos;
38 Z=cell(1,J); X=cell(1,J);
39 T=cell(J,N); A=T;
40 A_tot=cell(1,N); Ttot=cell(1,N);
41
42 for k=1:N % ground-truth motion. {p_m}^m
43     w_joint(1,k)=0;%(pi/6)*2/sqrt(2*pi)*exp(-2*(tvec(k)-2.5).^2)-(pi
       /6)*2/sqrt(2*pi)*exp(-2*(tvec(k)-7.5).^2);
44     w_joint(2,k)=0;%(pi/6)*2/sqrt(2*pi)*exp(-2*(tvec(k)-12.5).^2)-(pi
       /6)*2/sqrt(2*pi)*exp(-2*(tvec(k)-17.5).^2);
45     w_joint(3,k)=0;%(pi/6)*2/sqrt(2*pi)*exp(-2*(tvec(k)-22.5).^2)-(pi
       /6)*2/sqrt(2*pi)*exp(-2*(tvec(k)-27.5).^2);
46     w_joint(4,k)=0;%(pi/6)*2/sqrt(2*pi)*exp(-2*(tvec(k)-32.5).^2)-(pi
       /6)*2/sqrt(2*pi)*exp(-2*(tvec(k)-37.5).^2);
47     w_joint(5,k)=0;%(pi/6)*2/sqrt(2*pi)*exp(-2*(tvec(k)-42.5).^2)-(pi
       /6)*2/sqrt(2*pi)*exp(-2*(tvec(k)-47.5).^2);
48     w_joint(6,k)=0;%(pi/6)*2/sqrt(2*pi)*exp(-2*(tvec(k)-52.5).^2)-(pi
       /6)*2/sqrt(2*pi)*exp(-2*(tvec(k)-57.5).^2);
49     w_joint(7,k)=0;%(pi/6)*2/sqrt(2*pi)*exp(-2*(tvec(k)-62.5).^2)-(pi
       /6)*2/sqrt(2*pi)*exp(-2*(tvec(k)-67.5).^2);
50 end
51
52 % length of the links
53 lc=0.2; lu=0.3; lf=0.3;
54
55 % list of the DH paramenters
56 a_vec=[0,lc,0,0,0,0,0]';
```

```
alpha_vec=[pi/2,-pi/2,pi/2,pi/2,pi/2,-pi/2,0]';

% offset on theta DH parameter: TO BE ADDED THE INITIAL ATTITUDE
psi_off=[0,0,0,pi/2,0,0,pi/2]';
d_vec=[0,0,0,0,lu,0,lf]';

% link coordinates in {RF}^m
p_m=[a_vec,zeros(7,1),d_vec];
p_l=[p_m(2,:);p_m(5,:);p_m(7,:)];


for jj=1:J % for each joint
    A{jj,N+1}=eye(4); % DH matrix
    if jj==1
        q_0=[1,0,0,0]';
        x(:,1)=[0;0;0;q_0]; % initial state
        % rotation mtx from quaternions: coordinate change from RF{i} to RF{i-1}
        T{jj,1}=[x(4,1)^2+x(5,1)^2-x(6,1)^2-x(7,1)^2  2*(x(5,1)*x(6,1)-x(4,1)*x(7,1))  2*(x(5,1)*x(7,1)+x(4,1)*x(6,1))
                 2*(x(5,1)*x(6,1)+x(4,1)*x(7,1))  x(4,1)^2-x(5,1)^2+x(6,1)^2-x(7,1)^2  2*(x(5,1)*x(7,1)-x(4,1)*x(5,1))
                 2*(x(5,1)*x(7,1)-x(4,1)*x(6,1))  2*(x(6,1)*x(7,1)+x(4,1)*x(5,1))  x(4,1)^2-x(5,1)^2-x(6,1)^2+x(7,1)^2];
        % RPY or Tait-Bryan 321
        phi(jj,1)=atan2(T{jj,1}(3,2),T{jj,1}(3,3));
        theta(jj,1)=atan2(-T{jj,1}(3,1),sin(phi(jj,1))*T{jj,1}(3,2)+cos(phi(jj,1))*T{jj,1}(3,3));
        psi(jj,1)=atan2(-cos(phi(jj,1))*T{jj,1}(1,2)+sin(phi(jj,1))*T{jj,1}(1,3),cos(phi(jj,1))*T{jj,1}(2,2)-sin(phi(jj,1))*T{jj,1}(2,3));
        % D-H matrix
        A{jj,1}=[cos(psi_off(jj)+psi(jj,1)),-sin(psi_off(jj)+psi(jj,1))*cos(alpha_vec(jj)),sin(psi_off(jj)+psi(jj,1))*sin(alpha_vec(jj)),a_vec(jj)*cos(psi_off(jj)+psi(jj,1))
            sin(psi_off(jj)+psi(jj,1)),cos(psi_off(jj)+psi(jj,1))*cos(alpha_vec(jj)),-cos(psi_off(jj)+psi(jj,1))*sin(alpha_vec(jj)),a_vec(jj)*sin(psi_off(jj)+psi(jj,1))
            0,sin(alpha_vec(jj)),cos(alpha_vec(jj)),d_vec(jj)
            0,0,0,1];
        % Total DH mtx
        A_tot{1}=A{jj,1};
        for k=1:N % for each sample
            % state transition process
            x(1:3,k+1)=[0,0,w_joint(jj,k)]'; % rotation is always about z-axis
            x(4:7,k+1)=x(4:7,k)+1/2*Ts*[-x(5,k),-x(6,k),-x(7,k)
                                        x(4,k),-x(7,k),x(6,k)
                                        x(7,k),x(4,k),-x(5,k)
```

```
94                                                  -x(6,k),x(5,k),x(4,k)]*[x(1,k
    ),x(2,k),x(3,k)]';
95              x(4:7,k+1)=x(4:7,k+1)/norm(x(4:7,k+1));
96              % rotation mtx from quaternions: coordinate change from
    RF{i} to RF{i-1}
97              T{jj,k+1}=[x(4,k+1)^2+x(5,k+1)^2-x(6,k+1)^2-x(7,k+1)^2
    2*(x(5,k+1)*x(6,k+1)-x(4,k+1)*x(7,k+1))       2*(x(5,k+1)*x(7,k+1)+x
    (4,k+1)*x(6,k+1))
98                          2*(x(5,k+1)*x(6,k+1)+x(4,k+1)*x(7,k+1))       x(4,
    k+1)^2-x(5,k+1)^2+x(6,k+1)^2-x(7,k+1)^2  2*(x(5,k+1)*x(7,k+1)-x(4,k
    +1)*x(5,k+1))
99                          2*(x(5,k+1)*x(7,k+1)-x(4,k+1)*x(6,k+1))       2*(x
    (6,k+1)*x(7,k+1)+x(4,k+1)*x(5,k+1))       x(4,k+1)^2-x(5,k+1)^2-x(6,
    k+1)^2+x(7,k+1)^2];
100             % RPY or Tait-Bryan 321
101             phi(jj,k+1)=atan2(T{jj,k+1}(3,2),T{jj,k+1}(3,3));
102             theta(jj,k+1)=atan2(-T{jj,k+1}(3,1),sin(phi(jj,k+1))*T{jj
    ,k+1}(3,2)+cos(phi(jj,k+1))*T{jj,k+1}(3,3));
103             psi(jj,k+1)=atan2(-cos(phi(jj,k+1))*T{jj,k+1}(1,2)+sin(
    phi(jj,k+1))*T{jj,k+1}(1,3),cos(phi(jj,k+1))*T{jj,k+1}(2,2)-sin(
    phi(jj,k+1))*T{jj,k+1}(2,3));
104             % D-H matrix
105             A{jj,k+1}=[cos(psi_off(jj)+psi(jj,k+1)),-sin(psi_off(jj)+
    psi(jj,k+1))*cos(alpha_vec(jj)),sin(psi_off(jj)+psi(jj,k+1))*sin(
    alpha_vec(jj)),a_vec(jj)*cos(psi_off(jj)+psi(jj,k+1))
106                 sin(psi_off(jj)+psi(jj,k+1)),cos(psi_off(jj)+psi(jj,k
    +1))*cos(alpha_vec(jj)),-cos(psi_off(jj)+psi(jj,k+1))*sin(
    alpha_vec(jj)),a_vec(jj)*sin(psi_off(jj)+psi(jj,k+1))
107                 0,sin(alpha_vec(jj)),cos(alpha_vec(jj)),d_vec(jj)
108                 0,0,0,1];
109             % Total DH mtx
110             A_tot{k+1}=A{jj,k+1};
111             % angular acceleration derived from angular velocity w
112             w_dot(:,k)=(x(1:3,k+1)-x(1:3,k))/Ts;
113             % position
114             Pos{jj}(:,k)=A{jj,k}(1:3,4);
115             % acceleration in i^{RF}
116             a_j(1,k)=p_m(jj,3)*w_dot(2,k)-p_m(jj,2)*w_dot(3,k)-x(2,k)
    *(p_m(jj,1)*x(2,k)-p_m(jj,2)*x(1,k))-x(3,k)*(p_m(jj,1)*x(3,k)-p_m(
    jj,3)*x(1,k));
117             a_j(2,k)=p_m(jj,1)*w_dot(3,k)-p_m(jj,3)*w_dot(1,k)+x(1,k)
    *(p_m(jj,1)*x(2,k)-p_m(jj,2)*x(1,k))-x(3,k)*(p_m(jj,2)*x(3,k)-p_m(
    jj,3)*x(2,k));
118             a_j(3,k)=p_m(jj,2)*w_dot(1,k)-p_m(jj,1)*w_dot(2,k)+x(1,k)
    *(p_m(jj,1)*x(3,k)-p_m(jj,3)*x(1,k))+x(2,k)*(p_m(jj,2)*x(3,k)-p_m(
    jj,3)*x(2,k));
119             a_t=a_j;
120             % output equations
121             z(:,k)=[
```

```
122                 A{jj,k}(1:3,1:3)'*x(1:3,k)
123                 A_tot{k}(1:3,1:3)'*(a_t(:,k)+g_vec)
124                 1
125                 ];
126             s(1:3,k)=A{jj,k}(1:3,1:3)'*x(1:3,k);
127             s(4:7,k)=x(4:7,k);
128         end
129     else
130         q_0=[1,0,0,0]';
131         x(:,1)=[0;0;0;q_0]; % initial state; A{jj-1,k+1}(1:3,1:3)'*X{
    jj-1}(1:3,1)+
132         % rotation mtx from quaternions: coordinate change from RF{i}
     to RF{i-1}
133         T{jj,1}=[x(4,1)^2+x(5,1)^2-x(6,1)^2-x(7,1)^2  2*(x(5,1)*x(6,1)
    -x(4,1)*x(7,1))       2*(x(5,1)*x(7,1)+x(4,1)*x(6,1))
134                   2*(x(5,1)*x(6,1)+x(4,1)*x(7,1))       x(4,1)^2-x(5,1)
    ^2+x(6,1)^2-x(7,1)^2  2*(x(5,1)*x(7,1)-x(4,1)*x(5,1))
135                   2*(x(5,1)*x(7,1)-x(4,1)*x(6,1))       2*(x(6,1)*x(7,1)
    +x(4,1)*x(5,1))       x(4,1)^2-x(5,1)^2-x(6,1)^2+x(7,1)^2];
136         Ttot{1}=T{jj-1,1}*T{jj,1};
137         % RPY or Tait-Bryan 321
138         phi(jj,1)=atan2(T{jj,1}(3,2),T{jj,1}(3,3));
139         theta(jj,1)=atan2(-T{jj,1}(3,1),sin(phi(jj,1))*T{jj,1}(3,2)+
    cos(phi(jj,1))*T{jj,1}(3,3));
140         psi(jj,1)=atan2(-cos(phi(jj,1))*T{jj,1}(1,2)+sin(phi(jj,1))*T
    {jj,1}(1,3),cos(phi(jj,1))*T{jj,1}(2,2)-sin(phi(jj,1))*T{jj
    ,1}(2,3));
141         % D-H matrix
142         A{jj,1}=[cos(psi_off(jj)+psi(jj,1)),-sin(psi_off(jj)+psi(jj
    ,1))*cos(alpha_vec(jj)),sin(psi_off(jj)+psi(jj,1))*sin(alpha_vec(
    jj)),a_vec(jj)*cos(psi_off(jj)+psi(jj,1))
143             sin(psi_off(jj)+psi(jj,1)),cos(psi_off(jj)+psi(jj,1))*cos(
    alpha_vec(jj)),-cos(psi_off(jj)+psi(jj,1))*sin(alpha_vec(jj)),
    a_vec(jj)*sin(psi_off(jj)+psi(jj,1))
144             0,sin(alpha_vec(jj)),cos(alpha_vec(jj)),d_vec(jj)
145             0,0,0,1];
146         % Total DH mtx
147         A_tot{1}=A_tot{1}*A{jj,1};
148         p_m(jj,:)=[A{jj,1}(1:3,1:3)'*p_m(jj,:)']';
149         for k=1:N % for each sample
150             % state transition, process
151             x(1:3,k+1)=[0,0,w_joint(jj,k)]'; % rotation is always
    about z-axis;
152             x(4:7,k+1)=x(4:7,k)+1/2*Ts*[-x(5,k),-x(6,k),-x(7,k)
153                                         x(4,k),-x(7,k),x(6,k)
154                                         x(7,k),x(4,k),-x(5,k)
155                                        -x(6,k),x(5,k),x(4,k)]*[x(1,k
    ),x(2,k),x(3,k)]';
156             x(4:7,k+1)=x(4:7,k+1)/norm(x(4:7,k+1));
```

```
157             % rotation mtx from quaternions: coordinate change from
158             % RF{k}(body) to RF{k=1}(local)
159             T{jj,k+1}=[x(4,k+1)^2+x(5,k+1)^2-x(6,k+1)^2-x(7,k+1)^2
    2*(x(5,k+1)*x(6,k+1)-x(4,k+1)*x(7,k+1))       2*(x(5,k+1)*x(7,k+1)+x
    (4,k+1)*x(6,k+1))
160                         2*(x(5,k+1)*x(6,k+1)+x(4,k+1)*x(7,k+1))       x
    (4,k+1)^2-x(5,k+1)^2+x(6,k+1)^2-x(7,k+1)^2  2*(x(5,k+1)*x(7,k+1)-x
    (4,k+1)*x(5,k+1))
161                         2*(x(5,k+1)*x(7,k+1)-x(4,k+1)*x(6,k+1))
    2*(x(6,k+1)*x(7,k+1)+x(4,k+1)*x(5,k+1))       x(4,k+1)^2-x(5,k+1)^2-
    x(6,k+1)^2+x(7,k+1)^2];
162             Ttot{k+1}=T{jj-1,k+1}*T{jj,k+1};
163             % RPY or Tait-Bryan 321
164             phi(jj,k+1)=atan2(T{jj,k+1}(3,2),T{jj,k+1}(3,3));
165             theta(jj,k+1)=atan2(-T{jj,k+1}(3,1),sin(phi(jj,k+1))*T{jj
    ,k+1}(3,2)+cos(phi(jj,k+1))*T{jj,k+1}(3,3));
166             psi(jj,k+1)=atan2(-cos(phi(jj,k+1))*T{jj,k+1}(1,2)+sin(
    phi(jj,k+1))*T{jj,k+1}(1,3),cos(phi(jj,k+1))*T{jj,k+1}(2,2)-sin(
    phi(jj,k+1))*T{jj,k+1}(2,3));
167             % D-H matrix
168             A{jj,k+1}=[cos(psi_off(jj)+psi(jj,k+1)),-sin(psi_off(jj)+
    psi(jj,k+1))*cos(alpha_vec(jj)),sin(psi_off(jj)+psi(jj,k+1))*sin(
    alpha_vec(jj)),a_vec(jj)*cos(psi_off(jj)+psi(jj,k+1))
169                 sin(psi_off(jj)+psi(jj,k+1)),cos(psi_off(jj)+psi(jj,k
    +1))*cos(alpha_vec(jj)),-cos(psi_off(jj)+psi(jj,k+1))*sin(
    alpha_vec(jj)),a_vec(jj)*sin(psi_off(jj)+psi(jj,k+1))
170                 0,sin(alpha_vec(jj)),cos(alpha_vec(jj)),d_vec(jj)
171                 0,0,0,1];
172             % Total DH mtx
173             A_tot{k+1}=A_tot{k+1}*A{jj,k+1};
174             % total angular velocity:
175             s(1:3,k)=A{jj,k}(1:3,1:3)'*(X{jj-1}(1:3,k)+x(1:3,k)); %
    total angular rate
176             s(4:7,k)=x(4:7,k);
177             % angular acceleration derived from angular velocity w:
178             w_dot(:,k)=(A{jj,k+1}(1:3,1:3)'*(X{jj-1}(1:3,k+1)+x(1:3,k
    +1))-s(1:3,k))/Ts; %(s(1:3,k+1)-s(1:3,k))/Ts;
179             % position
180             Pos{jj}(:,k)=A_tot{k}(1:3,4);
181             % acceleration
182             % a=i^(i-1)_[R]*(i-1)^a_(i-1)+{w'}x{p}+{w}x({w}x{p}) in i
    ^{RF}
183             % acceleration of only link jj in jj-1^{RF}, a={w'}x{p}+{
    w}x({w}x{p})
184             a_j(1,k)=p_m(jj,3)*w_dot(2,k)-p_m(jj,2)*w_dot(3,k)-s(2,k)
    *(p_m(jj,1)*s(2,k)-p_m(jj,2)*s(1,k))-s(3,k)*(p_m(jj,1)*s(3,k)-p_m(
    jj,3)*s(1,k));
```

```
185             a_j(2,k)=p_m(jj,1)*w_dot(3,k)-p_m(jj,3)*w_dot(1,k)+s(1,k)
    *(p_m(jj,1)*s(2,k)-p_m(jj,2)*s(1,k))-s(3,k)*(p_m(jj,2)*s(3,k)-p_m(
    jj,3)*s(2,k));
186             a_j(3,k)=p_m(jj,2)*w_dot(1,k)-p_m(jj,1)*w_dot(2,k)+s(1,k)
    *(p_m(jj,1)*s(3,k)-p_m(jj,3)*s(1,k))+s(2,k)*(p_m(jj,2)*s(3,k)-p_m(
    jj,3)*s(2,k));
187             % total acceleration of link jj in i^{RF}: [A_i]'*i-1^
    acc_i
188             a_t(:,k)=A{jj,k}(1:3,1:3)'*At{jj-1}(:,k)+a_j(:,k); %
    parentheses added %T{jj,k}'*At{jj-1}(:,k)+a_j(:,k); OR s OR a_t
    !!! s...
189             % output equations
190             % A{jj,k}(1:3,1:3)'*Z{jj-1}(1:3,k) is w_(jj-1) in jj
    frame
191             z(:,k)=[
192                 s(1:3,k) % YES! % A{jj,k}(1:3,1:3)'*(Z{jj-1}(1:3,k)+x
    (1:3,k))
193                 A_tot{k}(1:3,1:3)'*g_vec+a_t(:,k) % acc&grav in i^th
    frame
194                 1
195                 ];
196         end
197     end
198 X{jj}(1:3,:)=s(1:3,:);
199 X{jj}(4:7,:)=x(4:7,:);
200 At{jj}=a_t(:,:);
201 Z{jj}=z(:,:);
202
203 clear x z
204 x=NaN(n,N+1);
205 z=NaN(7,size(x,2)-1);
206 end
207
208 save output.mat Z
209
210 % rotation into ISB std representation frames
211 R2=[0,1,0
212     0,0,1
213     1,0,0];
214 R5=[0,0,1
215     0,-1,0
216     1,0,0];
217 R7=[-1,0,0
218     0,0,-1
219     0,-1,0];
220 Zo=cell(1,J);
221 % rotation into JCS frame + noise corruption
222 for k=1:N
```

```
223     Zo{1}(:,k)=[R2,zeros(3);zeros(3),R2]*Z{2}(1:6,k)+mvnrnd(zeros
        (1,6),Rc)';
224     Zo{2}(:,k)=[R5,zeros(3);zeros(3),R5]*Z{5}(1:6,k)+mvnrnd(zeros
        (1,6),Rc)';
225     Zo{3}(:,k)=[R7,zeros(3);zeros(3),R7]*Z{7}(1:6,k)+mvnrnd(zeros
        (1,6),Rc)';
226 end
227 Zo{1}(7,:)=ones(1,N);
228 Zo{2}(7,:)=ones(1,N);
229 Zo{3}(7,:)=ones(1,N);
230
231 save out.mat Zo
232 save phi.mat phi
233 save theta.mat theta
234 save psi.mat psi
```

```
1  %% figures
2
3  for jj=1:J
4  figure
5      subplot(311), plot(tvec,X{jj}(1,1:N)), grid on, title(['x_1^',
       num2str(jj),'(t)'])
6      subplot(312), plot(tvec,X{jj}(2,1:N)), grid on, title(['x_2^',
       num2str(jj),'(t)'])
7      subplot(313), plot(tvec,X{jj}(3,1:N)), grid on, title(['x_3^',
       num2str(jj),'(t)'])
8  end
9
10 for jj=1:J
11 figure
12     subplot(411), plot(tvec,X{jj}(4,1:N)), grid on, title(['x_4^',
       num2str(jj),'(t)'])
13     subplot(412), plot(tvec,X{jj}(5,1:N)), grid on, title(['x_5^',
       num2str(jj),'(t)'])
14     subplot(413), plot(tvec,X{jj}(6,1:N)), grid on, title(['x_6^',
       num2str(jj),'(t)'])
15     subplot(414), plot(tvec,X{jj}(7,1:N)), grid on, title(['x_7^',
       num2str(jj),'(t)'])
16 end
17 for jj=1:J
18 figure
19     subplot(311), plot(tvec,Z{jj}(1,1:N)), grid on, title(['z_1^',
       num2str(jj),'(t)'])
20     subplot(312), plot(tvec,Z{jj}(2,1:N)), grid on, title(['z_2^',
       num2str(jj),'(t)'])
21     subplot(313), plot(tvec,Z{jj}(3,1:N)), grid on, title(['z_3^',
       num2str(jj),'(t)'])
```

```
22 end
23 for jj=1:J
24 figure
25     subplot(311), plot(tvec,Z{jj}(4,1:N)), grid on, title(['z_4^',
       num2str(jj),'(t)'])
26     subplot(312), plot(tvec,Z{jj}(5,1:N)), grid on, title(['z_5^',
       num2str(jj),'(t)'])
27     subplot(313), plot(tvec,Z{jj}(6,1:N)), grid on, title(['z_6^',
       num2str(jj),'(t)'])
28 end
29
30 for jj=1:J
31 figure
32     subplot(311), plot(tvec,Pos{jj}(1,1:N)), grid on, title(['P_x^',
       num2str(jj),'(t)'])
33     subplot(312), plot(tvec,Pos{jj}(2,1:N)), grid on, title(['P_y^',
       num2str(jj),'(t)'])
34     subplot(313), plot(tvec,Pos{jj}(3,1:N)), grid on, title(['P_z^',
       num2str(jj),'(t)'])
35 end
36 for jj=1:J
37 figure
38     subplot(311), plot(tvec,At{jj}(1,1:N)), grid on, title(['a_x^',
       num2str(jj),'(t)'])
39     subplot(312), plot(tvec,At{jj}(2,1:N)), grid on, title(['a_y^',
       num2str(jj),'(t)'])
40     subplot(313), plot(tvec,At{jj}(3,1:N)), grid on, title(['a_z^',
       num2str(jj),'(t)'])
41 end
42 for jj=1:J%[2,5,7]
43 figure
44     subplot(311), plot(phi(jj,:)*180/pi), grid on, title(['\phi^',
       num2str(jj),'(t)'])
45     subplot(312), plot(theta(jj,:)*180/pi), grid on, title(['\theta^'
       ,num2str(jj),'(t)'])
46     subplot(313), plot(psi(jj,:)*180/pi), grid on, title(['\psi^',
       num2str(jj),'(t)'])
47 end
```

B.3 Extended Kalman Filter Algorithm Implementation

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Tommaso Lovato    28/03/2024
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Extended Kalman Filter implementation for estimating the attitude of a
% 3 link kinematic chain with revolute joints
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc
clear
close all

load out.mat
load phi.mat
load theta.mat
load psi.mat

figure
    subplot(311), plot(Zo{1}(1,:)), grid on, title('z_1^s(t)'), xlabel('[s]'), ylabel('[rad/s]')
    subplot(312), plot(Zo{1}(2,:)), grid on, title('z_2^s(t)'), xlabel('[s]'), ylabel('[rad/s]')
    subplot(313), plot(Zo{1}(3,:)), grid on, title('z_3^s(t)'), xlabel('[s]'), ylabel('[rad/s]')
figure
    subplot(311), plot(Zo{1}(4,:)), grid on, title('z_4^s(t)'), xlabel('[s]'), ylabel('[m/s^2]')
    subplot(312), plot(Zo{1}(5,:)), grid on, title('z_5^s(t)'), xlabel('[s]'), ylabel('[m/s^2]')
    subplot(313), plot(Zo{1}(6,:)), grid on, title('z_6^s(t)'), xlabel('[s]'), ylabel('[m/s^2]')

figure
    subplot(311), plot(Zo{2}(1,:)), grid on, title('z_1^e(t)'), xlabel('[s]'), ylabel('[rad/s]')
    subplot(312), plot(Zo{2}(2,:)), grid on, title('z_2^e(t)'), xlabel('[s]'), ylabel('[rad/s]')
    subplot(313), plot(Zo{2}(3,:)), grid on, title('z_3^e(t)'), xlabel('[s]'), ylabel('[rad/s]')
figure
    subplot(311), plot(Zo{2}(4,:)), grid on, title('z_4^e(t)'), xlabel('[s]'), ylabel('[m/s^2]')
    subplot(312), plot(Zo{2}(5,:)), grid on, title('z_5^e(t)'), xlabel('[s]'), ylabel('[m/s^2]')
```

```
32      subplot(313), plot(Zo{2}(6,:)), grid on, title('z_6^e(t)'),
        xlabel('[s]'), ylabel('[m/s^2]')
33
34  figure
35      subplot(311), plot(Zo{3}(1,:)), grid on, title('z_1^w(t)'),
        xlabel('[s]'), ylabel('[rad/s]')
36      subplot(312), plot(Zo{3}(2,:)), grid on, title('z_2^w(t)'),
        xlabel('[s]'), ylabel('[rad/s]')
37      subplot(313), plot(Zo{3}(3,:)), grid on, title('z_3^w(t)'),
        xlabel('[s]'), ylabel('[rad/s]')
38  figure
39      subplot(311), plot(Zo{3}(4,:)), grid on, title('z_4^w(t)'),
        xlabel('[s]'), ylabel('[m/s^2]')
40      subplot(312), plot(Zo{3}(5,:)), grid on, title('z_5^w(t)'),
        xlabel('[s]'), ylabel('[m/s^2]')
41      subplot(313), plot(Zo{3}(6,:)), grid on, title('z_6^w(t)'),
        xlabel('[s]'), ylabel('[m/s^2]')
42
43  pause(5)
44
45  %%
46  close all
47
48  rng default
49
50  clear  x_p e x_f P z_p phi_est theta_est psi_et index_not_obsv
51  Ts=1/100; %[s]
52  T_fin=10; %[s]
53  N=T_fin/Ts;
54  tvec=0:Ts:T_fin-Ts;
55
56  g=9.81; %[m/s^2]
57  tau=0.2;
58  % gravity vector
59  g_vec=[0;0;g];
60
61  J=7; % number of joints
62  L=3; % number of links
63  n=7; % number of states
64
65  Qc=1e2*diag([1,1,1,0,0,0,0]); % process noise variance, only on w
        states
66  Rc=diag([1e-3,1e-3,1e-3,2.5e-3,2.5e-3,2.5e-3,1e-12]); % measurement
        noise variance: gyro, acc
67
68  % length of the links
69  lc=0.2; lu=0.3; lf=0.3;
70  p_l=[0,0,lc
71       0,lu,0
```

```
72      0,lf,0];
73
74 T_tot=cell(1,N); T_tot(:)={eye(3)}; T_f=T_tot;
75 x_p=NaN(n,N+1,L);
76 z_p=cell(1,L);
77 e=NaN(7,N); v_sens=zeros(size(e));
78 x_f=NaN(n,N,L);
79 w_dot=NaN(3,N,L); w_dot(:,1,1)=0; w_dot(:,1,2)=0; w_dot(:,1,3)=0;
80 P=cell(L,length(x_p));
81 P{1,1}=Qc; P{2,1}=Qc; P{3,1}=Qc;
82 phi_est=NaN(J,N); theta_est=NaN(J,N); psi_est=NaN(J,N);
83
84 % rotation from DH to ISB of frame2
85 R2=[0,1,0
86      0,0,1
87      1,0,0];
88
89 % initial states
90 x_p(:,1,1)=[0,0,0,1,0,0,0]'+mvnrnd(zeros(1,7),Rc)';
91 x_p(:,1,2)=[0,0,0,cos(-pi/4),sin(-pi/4),0,0]'+mvnrnd(zeros(1,7),Rc)';
92 x_p(:,1,3)=[0,0,0,1,0,0,0]'+mvnrnd(zeros(1,7),Rc)';
93
94 % observability test
95 index_not_obsv=[]; % vector containing the samples in which the
       system is not observable
96
97 for k=1:N-1 % for each sample
98     for jj=1:L % for each link
99         % Linearized process matrix
100        F_hat=[          -1/tau,             0,             0,             0,
                0,             0,             0
101                          0,          -1/tau,             0,             0,
                0,             0,             0
102                          0,             0,          -1/tau,             0,
                0,             0,             0
103                -x_p(5,k,jj)/2, -x_p(6,k,jj)/2, -x_p(7,k,jj)/2,
       0, -x_p(1,k,jj)/2, -x_p(2,k,jj)/2, -x_p(3,k,jj)/2
104                 x_p(4,k,jj)/2, -x_p(7,k,jj)/2,  x_p(6,k,jj)/2, x_p(1,
    k,jj)/2,             0,  x_p(3,k,jj)/2, -x_p(2,k,jj)/2
105                 x_p(7,k,jj)/2,  x_p(4,k,jj)/2, -x_p(5,k,jj)/2, x_p(2,
    k,jj)/2, -x_p(3,k,jj)/2,             0,  x_p(1,k,jj)/2
106                -x_p(6,k,jj)/2,  x_p(5,k,jj)/2,  x_p(4,k,jj)/2, x_p(3,
    k,jj)/2,  x_p(2,k,jj)/2, -x_p(1,k,jj)/2,             0
107                ];
108        % Linearized output matrix
109        if jj==1
110            H_hat=[
111                1,0,0,0,0,0,0
112                0,1,0,0,0,0,0
```

```
113                 0,0,1,0,0,0,0
114                 p_l(jj,2)*x_p(2,k,jj)+p_l(jj,3)*x_p(3,k,jj),p_l(jj,2)
    *x_p(1,k,jj)-2*p_l(jj,1)*x_p(2,k,jj),p_l(jj,3)*x_p(1,k,jj)-2*p_l(
    jj,1)*x_p(3,k,jj),(981*x_p(4,k,jj))/50,(981*x_p(5,k,jj))/50,-(981*
    x_p(6,k,jj))/50,-(981*x_p(7,k,jj))/50
115                 p_l(jj,1)*x_p(2,k,jj)-2*p_l(jj,2)*x_p(1,k,jj),p_l(jj
    ,1)*x_p(1,k,jj)+p_l(jj,3)*x_p(3,k,jj),p_l(jj,3)*x_p(2,k,jj)-2*p_l(
    jj,2)*x_p(3,k,jj),-(981*x_p(7,k,jj))/50,(981*x_p(6,k,jj))/50,(981*
    x_p(5,k,jj))/50,-(981*x_p(4,k,jj))/50
116                 p_l(jj,1)*x_p(3,k,jj)-2*p_l(jj,3)*x_p(1,k,jj),p_l(jj
    ,2)*x_p(3,k,jj)-2*p_l(jj,3)*x_p(2,k,jj),p_l(jj,1)*x_p(1,k,jj)+p_l(
    jj,2)*x_p(2,k,jj),(981*x_p(6,k,jj))/50,(981*x_p(7,k,jj))/50,(981*
    x_p(4,k,jj))/50,(981*x_p(5,k,jj))/50
117                 0,0,0,x_p(4,k,jj)/(x_p(4,k,jj)^2+x_p(5,k,jj)^2+x_p(6,
    k,jj)^2+x_p(7,k,jj)^2)^(1/2),x_p(5,k,jj)/(x_p(4,k,jj)^2+x_p(5,k,jj
    )^2+x_p(6,k,jj)^2+x_p(7,k,jj)^2)^(1/2),x_p(6,k,jj)/(x_p(4,k,jj)^2+
    x_p(5,k,jj)^2+x_p(6,k,jj)^2+x_p(7,k,jj)^2)^(1/2),x_p(7,k,jj)/(x_p
    (4,k,jj)^2+x_p(5,k,jj)^2+x_p(6,k,jj)^2+x_p(7,k,jj)^2)^(1/2)
118                 ];
119         else
120             H_hat=[
121                 1,0,0,2*x_p(4,k,jj)*x_f(1,k,jj-1)+2*x_p(7,k,jj)*x_f
    (2,k,jj-1)-2*x_p(6,k,jj)*x_f(3,k,jj-1),2*x_p(5,k,jj)*x_f(1,k,jj-1)
    +2*x_p(6,k,jj)*x_f(2,k,jj-1)+2*x_p(7,k,jj)*x_f(3,k,jj-1),2*x_p(5,k
    ,jj)*x_f(2,k,jj-1)-2*x_p(6,k,jj)*x_f(1,k,jj-1)-2*x_p(4,k,jj)*x_f
    (3,k,jj-1),2*x_p(4,k,jj)*x_f(2,k,jj-1)-2*x_p(7,k,jj)*x_f(1,k,jj-1)
    +2*x_p(5,k,jj)*x_f(3,k,jj-1)
122                 0,1,0,2*x_p(4,k,jj)*x_f(2,k,jj-1)-2*x_p(7,k,jj)*x_f
    (1,k,jj-1)+2*x_p(5,k,jj)*x_f(3,k,jj-1),2*x_p(6,k,jj)*x_f(1,k,jj-1)
    -2*x_p(5,k,jj)*x_f(2,k,jj-1)+2*x_p(4,k,jj)*x_f(3,k,jj-1),2*x_p(5,k
    ,jj)*x_f(1,k,jj-1)+2*x_p(6,k,jj)*x_f(2,k,jj-1)+2*x_p(7,k,jj)*x_f
    (3,k,jj-1),2*x_p(6,k,jj)*x_f(3,k,jj-1)-2*x_p(7,k,jj)*x_f(2,k,jj-1)
    -2*x_p(4,k,jj)*x_f(1,k,jj-1)
123                 0,0,1,2*x_p(6,k,jj)*x_f(1,k,jj-1)-2*x_p(5,k,jj)*x_f
    (2,k,jj-1)+2*x_p(4,k,jj)*x_f(3,k,jj-1),2*x_p(7,k,jj)*x_f(1,k,jj-1)
    -2*x_p(4,k,jj)*x_f(2,k,jj-1)-2*x_p(5,k,jj)*x_f(3,k,jj-1),2*x_p(4,k
    ,jj)*x_f(1,k,jj-1)+2*x_p(7,k,jj)*x_f(2,k,jj-1)-2*x_p(6,k,jj)*x_f
    (3,k,jj-1),2*x_p(5,k,jj)*x_f(1,k,jj-1)+2*x_p(6,k,jj)*x_f(2,k,jj-1)
    +2*x_p(7,k,jj)*x_f(3,k,jj-1)
124                 p_l(jj,2)*(x_p(2,k,jj)+x_f(2,k,jj-1)*(x_p(4,k,jj)^2-
    x_p(5,k,jj)^2+x_p(6,k,jj)^2-x_p(7,k,jj)^2)-x_f(1,k,jj-1)*(2*x_p(4,
    k,jj)*x_p(7,k,jj)-2*x_p(5,k,jj)*x_p(6,k,jj))+x_f(3,k,jj-1)*(2*x_p
    (4,k,jj)*x_p(5,k,jj)+2*x_p(6,k,jj)*x_p(7,k,jj)))+p_l(jj,3)*(x_p(3,
    k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-x_p(6,k,jj)^2+x_p
    (7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,jj)+2*x_p(5,k,jj)
    *x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)-2*x_p(6,k,
    jj)*x_p(7,k,jj))) ,...
```

```
125                 p_l(jj,2)*(x_p(1,k,jj)+x_f(1,k,jj-1)*(x_p(4,k,jj)^2+
    x_p(5,k,jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)^2)+x_f(2,k,jj-1)*(2*x_p(4,
    k,jj)*x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k,jj))-x_f(3,k,jj-1)*(2*x_p
    (4,k,jj)*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p(7,k,jj)))-2*p_l(jj,1)*(x_p
    (2,k,jj)+x_f(2,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+x_p(6,k,jj)^2-
    x_p(7,k,jj)^2)-x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)-2*x_p(5,k,
    jj)*x_p(6,k,jj))+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)+2*x_p(6,
    k,jj)*x_p(7,k,jj))),...
126                 p_l(jj,3)*(x_p(1,k,jj)+x_f(1,k,jj-1)*(x_p(4,k,jj)^2+
    x_p(5,k,jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)^2)+x_f(2,k,jj-1)*(2*x_p(4,
    k,jj)*x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k,jj))-x_f(3,k,jj-1)*(2*x_p
    (4,k,jj)*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p(7,k,jj)))-2*p_l(jj,1)*(x_p
    (3,k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-x_p(6,k,jj)^2+
    x_p(7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,jj)+2*x_p(5,k,
    jj)*x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)-2*x_p(6,
    k,jj)*x_p(7,k,jj))),...
```

```
127                 (981*T_tot{k}(3,1)*x_p(4,k,jj))/50+(981*T_tot{k}(3,2)
    *x_p(7,k,jj))/50-(981*T_tot{k}(3,3)*x_p(6,k,jj))/50-(p_l(jj,1)*(2*
    x_p(4,k,jj)*x_f(2,k,jj-1)-2*x_p(7,k,jj)*x_f(1,k,jj-1)+2*x_p(5,k,jj
    )*x_f(3,k,jj-1))-p_l(jj,2)*(2*x_p(4,k,jj)*x_f(1,k,jj-1)+2*x_p(7,k,
    jj)*x_f(2,k,jj-1)-2*x_p(6,k,jj)*x_f(3,k,jj-1)))*(x_p(2,k,jj)+x_f
    (2,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+x_p(6,k,jj)^2-x_p(7,k,jj)
    ^2)-x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)-2*x_p(5,k,jj)*x_p(6,k
    ,jj))+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)+2*x_p(6,k,jj)*x_p
    (7,k,jj)))-(p_l(jj,1)*(2*x_p(6,k,jj)*x_f(1,k,jj-1)-2*x_p(5,k,jj)*
    x_f(2,k,jj-1)+2*x_p(4,k,jj)*x_f(3,k,jj-1))-p_l(jj,3)*(2*x_p(4,k,jj
    )*x_f(1,k,jj-1)+2*x_p(7,k,jj)*x_f(2,k,jj-1)-2*x_p(6,k,jj)*x_f(3,k,
    jj-1)))*(x_p(3,k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-
    x_p(6,k,jj)^2+x_p(7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,
    jj)+2*x_p(5,k,jj)*x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,
    k,jj)-2*x_p(6,k,jj)*x_p(7,k,jj)))-(p_l(jj,1)*(x_p(2,k,jj)+x_f(2,k,
    jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+x_p(6,k,jj)^2-x_p(7,k,jj)^2)-
    x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)-2*x_p(5,k,jj)*x_p(6,k,jj)
    )+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)+2*x_p(6,k,jj)*x_p(7,k,
    jj)))-p_l(jj,2)*(x_p(1,k,jj)+x_f(1,k,jj-1)*(x_p(4,k,jj)^2+x_p(5,k,
    jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)^2)+x_f(2,k,jj-1)*(2*x_p(4,k,jj)*
    x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k,jj))-x_f(3,k,jj-1)*(2*x_p(4,k,jj
    )*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p(7,k,jj))))*(2*x_p(4,k,jj)*x_f(2,k,
    jj-1)-2*x_p(7,k,jj)*x_f(1,k,jj-1)+2*x_p(5,k,jj)*x_f(3,k,jj-1))-(
    p_l(jj,1)*(x_p(3,k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-
    x_p(6,k,jj)^2+x_p(7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,
    jj)+2*x_p(5,k,jj)*x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,
    k,jj)-2*x_p(6,k,jj)*x_p(7,k,jj)))-p_l(jj,3)*(x_p(1,k,jj)+x_f(1,k,
    jj-1)*(x_p(4,k,jj)^2+x_p(5,k,jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)^2)+
    x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k,jj)
    )-x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p(7,k,
    jj))))*(2*x_p(6,k,jj)*x_f(1,k,jj-1)-2*x_p(5,k,jj)*x_f(2,k,jj-1)+2*
    x_p(4,k,jj)*x_f(3,k,jj-1))-2*x_p(4,k,jj)*((981*T_tot{k}(3,1))/100-
    z_p{jj-1}(4,k))-2*x_p(7,k,jj)*((981*T_tot{k}(3,2))/100-z_p{jj
    -1}(5,k))+2*x_p(6,k,jj)*((981*T_tot{k}(3,3))/100-z_p{jj-1}(6,k))
    ,...
```

```
128                 (981*T_tot{k}(3,1)*x_p(5,k,jj))/50+(981*T_tot{k}(3,2)
    *x_p(6,k,jj))/50+(981*T_tot{k}(3,3)*x_p(7,k,jj))/50-(p_l(jj,1)*(2*
    x_p(6,k,jj)*x_f(1,k,jj-1)-2*x_p(5,k,jj)*x_f(2,k,jj-1)+2*x_p(4,k,jj
    )*x_f(3,k,jj-1))-p_l(jj,2)*(2*x_p(5,k,jj)*x_f(1,k,jj-1)+2*x_p(6,k,
    jj)*x_f(2,k,jj-1)+2*x_p(7,k,jj)*x_f(3,k,jj-1)))*(x_p(2,k,jj)+x_f
    (2,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+x_p(6,k,jj)^2-x_p(7,k,jj)
    ^2)-x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)-2*x_p(5,k,jj)*x_p(6,k
    ,jj))+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)+2*x_p(6,k,jj)*x_p
    (7,k,jj)))+(p_l(jj,1)*(2*x_p(4,k,jj)*x_f(2,k,jj-1)-2*x_p(7,k,jj)*
    x_f(1,k,jj-1)+2*x_p(5,k,jj)*x_f(3,k,jj-1))+p_l(jj,3)*(2*x_p(5,k,jj
    )*x_f(1,k,jj-1)+2*x_p(6,k,jj)*x_f(2,k,jj-1)+2*x_p(7,k,jj)*x_f(3,k,
    jj-1)))*(x_p(3,k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-
    x_p(6,k,jj)^2+x_p(7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,
    jj)+2*x_p(5,k,jj)*x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,
    k,jj)-2*x_p(6,k,jj)*x_p(7,k,jj)))-(p_l(jj,1)*(x_p(2,k,jj)+x_f(2,k,
    jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+x_p(6,k,jj)^2-x_p(7,k,jj)^2)-
    x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)-2*x_p(5,k,jj)*x_p(6,k,jj)
    )+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)+2*x_p(6,k,jj)*x_p(7,k,
    jj)))-p_l(jj,2)*(x_p(1,k,jj)+x_f(1,k,jj-1)*(x_p(4,k,jj)^2+x_p(5,k,
    jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)^2)+x_f(2,k,jj-1)*(2*x_p(4,k,jj)*
    x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k,jj))-x_f(3,k,jj-1)*(2*x_p(4,k,jj
    )*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p(7,k,jj))))*(2*x_p(6,k,jj)*x_f(1,k,
    jj-1)-2*x_p(5,k,jj)*x_f(2,k,jj-1)+2*x_p(4,k,jj)*x_f(3,k,jj-1))+(
    p_l(jj,1)*(x_p(3,k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-
    x_p(6,k,jj)^2+x_p(7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,
    jj)+2*x_p(5,k,jj)*x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,
    k,jj)-2*x_p(6,k,jj)*x_p(7,k,jj)))-p_l(jj,3)*(x_p(1,k,jj)+x_f(1,k,
    jj-1)*(x_p(4,k,jj)^2+x_p(5,k,jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)^2)+
    x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k,jj)
    )-x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p(7,k,
    jj))))*(2*x_p(4,k,jj)*x_f(2,k,jj-1)-2*x_p(7,k,jj)*x_f(1,k,jj-1)+2*
    x_p(5,k,jj)*x_f(3,k,jj-1))-2*x_p(5,k,jj)*((981*T_tot{k}(3,1))/100-
    z_p{jj-1}(4,k))-2*x_p(6,k,jj)*((981*T_tot{k}(3,2))/100-z_p{jj
    -1}(5,k))-2*x_p(7,k,jj)*((981*T_tot{k}(3,3))/100-z_p{jj-1}(6,k))
    ,...
```

```
129                 (981*T_tot{k}(3,2)*x_p(5,k,jj))/50-(981*T_tot{k}(3,1)
    *x_p(6,k,jj))/50-(981*T_tot{k}(3,3)*x_p(4,k,jj))/50-(p_l(jj,1)*(2*
    x_p(5,k,jj)*x_f(1,k,jj-1)+2*x_p(6,k,jj)*x_f(2,k,jj-1)+2*x_p(7,k,jj
    )*x_f(3,k,jj-1))+p_l(jj,2)*(2*x_p(6,k,jj)*x_f(1,k,jj-1)-2*x_p(5,k,
    jj)*x_f(2,k,jj-1)+2*x_p(4,k,jj)*x_f(3,k,jj-1)))*(x_p(2,k,jj)+x_f
    (2,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+x_p(6,k,jj)^2-x_p(7,k,jj)
    ^2)-x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)-2*x_p(5,k,jj)*x_p(6,k
    ,jj))+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)+2*x_p(6,k,jj)*x_p
    (7,k,jj)))-(p_l(jj,1)*(2*x_p(4,k,jj)*x_f(1,k,jj-1)+2*x_p(7,k,jj)*
    x_f(2,k,jj-1)-2*x_p(6,k,jj)*x_f(3,k,jj-1))+p_l(jj,3)*(2*x_p(6,k,jj
    )*x_f(1,k,jj-1)-2*x_p(5,k,jj)*x_f(2,k,jj-1)+2*x_p(4,k,jj)*x_f(3,k,
    jj-1)))*(x_p(3,k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-
    x_p(6,k,jj)^2+x_p(7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,
    jj)+2*x_p(5,k,jj)*x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,
    k,jj)-2*x_p(6,k,jj)*x_p(7,k,jj)))-(p_l(jj,1)*(x_p(2,k,jj)+x_f(2,k,
    jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+x_p(6,k,jj)^2-x_p(7,k,jj)^2)-
    x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)-2*x_p(5,k,jj)*x_p(6,k,jj)
    )+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)+2*x_p(6,k,jj)*x_p(7,k,
    jj)))-p_l(jj,2)*(x_p(1,k,jj)+x_f(1,k,jj-1)*(x_p(4,k,jj)^2+x_p(5,k,
    jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)^2)+x_f(2,k,jj-1)*(2*x_p(4,k,jj)*
    x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k,jj))-x_f(3,k,jj-1)*(2*x_p(4,k,jj
    )*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p(7,k,jj))))*(2*x_p(5,k,jj)*x_f(1,k,
    jj-1)+2*x_p(6,k,jj)*x_f(2,k,jj-1)+2*x_p(7,k,jj)*x_f(3,k,jj-1))-(
    p_l(jj,1)*(x_p(3,k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-
    x_p(6,k,jj)^2+x_p(7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,
    jj)+2*x_p(5,k,jj)*x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,
    k,jj)-2*x_p(6,k,jj)*x_p(7,k,jj)))-p_l(jj,3)*(x_p(1,k,jj)+x_f(1,k,
    jj-1)*(x_p(4,k,jj)^2+x_p(5,k,jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)^2)+
    x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k,jj)
    )-x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p(7,k,
    jj))))*(2*x_p(4,k,jj)*x_f(1,k,jj-1)+2*x_p(7,k,jj)*x_f(2,k,jj-1)-2*
    x_p(6,k,jj)*x_f(3,k,jj-1))+2*x_p(6,k,jj)*((981*T_tot{k}(3,1))/100-
    z_p{jj-1}(4,k))-2*x_p(5,k,jj)*((981*T_tot{k}(3,2))/100-z_p{jj
    -1}(5,k))+2*x_p(4,k,jj)*((981*T_tot{k}(3,3))/100-z_p{jj-1}(6,k))
    ,...
```

```
130                 (981*T_tot{k}(3,2)*x_p(4,k,jj))/50-(981*T_tot{k}(3,1)
    *x_p(7,k,jj))/50+(981*T_tot{k}(3,3)*x_p(5,k,jj))/50+(p_l(jj,1)*(2*
    x_p(4,k,jj)*x_f(1,k,jj-1)+2*x_p(7,k,jj)*x_f(2,k,jj-1)-2*x_p(6,k,jj
    )*x_f(3,k,jj-1))+p_l(jj,2)*(2*x_p(4,k,jj)*x_f(2,k,jj-1)-2*x_p(7,k,
    jj)*x_f(1,k,jj-1)+2*x_p(5,k,jj)*x_f(3,k,jj-1)))*(x_p(2,k,jj)+x_f
    (2,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+x_p(6,k,jj)^2-x_p(7,k,jj)
    ^2)-x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)-2*x_p(5,k,jj)*x_p(6,k
    ,jj))+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)+2*x_p(6,k,jj)*x_p
    (7,k,jj)))-(p_l(jj,1)*(2*x_p(5,k,jj)*x_f(1,k,jj-1)+2*x_p(6,k,jj)*
    x_f(2,k,jj-1)+2*x_p(7,k,jj)*x_f(3,k,jj-1))-p_l(jj,3)*(2*x_p(4,k,jj
    )*x_f(2,k,jj-1)-2*x_p(7,k,jj)*x_f(1,k,jj-1)+2*x_p(5,k,jj)*x_f(3,k,
    jj-1)))*(x_p(3,k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-
    x_p(6,k,jj)^2+x_p(7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,
    jj)+2*x_p(5,k,jj)*x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,
    k,jj)-2*x_p(6,k,jj)*x_p(7,k,jj)))+(p_l(jj,1)*(x_p(2,k,jj)+x_f(2,k,
    jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+x_p(6,k,jj)^2-x_p(7,k,jj)^2)-
    x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)-2*x_p(5,k,jj)*x_p(6,k,jj)
    )+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)+2*x_p(6,k,jj)*x_p(7,k,
    jj)))-p_l(jj,2)*(x_p(1,k,jj)+x_f(1,k,jj-1)*(x_p(4,k,jj)^2+x_p(5,k,
    jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)^2)+x_f(2,k,jj-1)*(2*x_p(4,k,jj)*
    x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k,jj))-x_f(3,k,jj-1)*(2*x_p(4,k,jj
    )*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p(7,k,jj))))*(2*x_p(4,k,jj)*x_f(1,k,
    jj-1)+2*x_p(7,k,jj)*x_f(2,k,jj-1)-2*x_p(6,k,jj)*x_f(3,k,jj-1))-(
    p_l(jj,1)*(x_p(3,k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-
    x_p(6,k,jj)^2+x_p(7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,
    jj)+2*x_p(5,k,jj)*x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,
    k,jj)-2*x_p(6,k,jj)*x_p(7,k,jj)))-p_l(jj,3)*(x_p(1,k,jj)+x_f(1,k,
    jj-1)*(x_p(4,k,jj)^2+x_p(5,k,jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)^2)+
    x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k,jj)
    )-x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p(7,k,
    jj))))*(2*x_p(5,k,jj)*x_f(1,k,jj-1)+2*x_p(6,k,jj)*x_f(2,k,jj-1)+2*
    x_p(7,k,jj)*x_f(3,k,jj-1))-2*x_p(4,k,jj)*((981*T_tot{k}(3,2))/100-
    z_p{jj-1}(5,k))+2*x_p(7,k,jj)*((981*T_tot{k}(3,1))/100-z_p{jj
    -1}(4,k))-2*x_p(5,k,jj)*((981*T_tot{k}(3,3))/100-z_p{jj-1}(6,k))
```

```
131                 p_l(jj,1)*(x_p(2,k,jj)+x_f(2,k,jj-1)*(x_p(4,k,jj)^2-
    x_p(5,k,jj)^2+x_p(6,k,jj)^2-x_p(7,k,jj)^2)-x_f(1,k,jj-1)*(2*x_p(4,
    k,jj)*x_p(7,k,jj)-2*x_p(5,k,jj)*x_p(6,k,jj))+x_f(3,k,jj-1)*(2*x_p
    (4,k,jj)*x_p(5,k,jj)+2*x_p(6,k,jj)*x_p(7,k,jj)))-2*p_l(jj,2)*(x_p
    (1,k,jj)+x_f(1,k,jj-1)*(x_p(4,k,jj)^2+x_p(5,k,jj)^2-x_p(6,k,jj)^2-
    x_p(7,k,jj)^2)+x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)+2*x_p(5,k,
    jj)*x_p(6,k,jj))-x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,jj)-2*x_p(5,
    k,jj)*x_p(7,k,jj))),p_l(jj,1)*(x_p(1,k,jj)+x_f(1,k,jj-1)*(x_p(4,k,
    jj)^2+x_p(5,k,jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)^2)+x_f(2,k,jj-1)*(2*
    x_p(4,k,jj)*x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k,jj))-x_f(3,k,jj-1)
    *(2*x_p(4,k,jj)*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p(7,k,jj)))+p_l(jj,3)
    *(x_p(3,k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-x_p(6,k,
    jj)^2+x_p(7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,jj)+2*
    x_p(5,k,jj)*x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)
    -2*x_p(6,k,jj)*x_p(7,k,jj))),p_l(jj,3)*(x_p(2,k,jj)+x_f(2,k,jj-1)
    *(x_p(4,k,jj)^2-x_p(5,k,jj)^2+x_p(6,k,jj)^2-x_p(7,k,jj)^2)-x_f(1,k
    ,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)-2*x_p(5,k,jj)*x_p(6,k,jj))+x_f
    (3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)+2*x_p(6,k,jj)*x_p(7,k,jj)))
    -2*p_l(jj,2)*(x_p(3,k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)
    ^2-x_p(6,k,jj)^2+x_p(7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p
    (6,k,jj)+2*x_p(5,k,jj)*x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*
    x_p(5,k,jj)-2*x_p(6,k,jj)*x_p(7,k,jj))) ,...
```

```
132                 (981*T_tot{k}(3,2)*x_p(4,k,jj))/50-(981*T_tot{k}(3,1)
    *x_p(7,k,jj))/50+(981*T_tot{k}(3,3)*x_p(5,k,jj))/50+(p_l(jj,1)*(2*
    x_p(4,k,jj)*x_f(2,k,jj-1)-2*x_p(7,k,jj)*x_f(1,k,jj-1)+2*x_p(5,k,jj
    )*x_f(3,k,jj-1))-p_l(jj,2)*(2*x_p(4,k,jj)*x_f(1,k,jj-1)+2*x_p(7,k,
    jj)*x_f(2,k,jj-1)-2*x_p(6,k,jj)*x_f(3,k,jj-1)))*(x_p(1,k,jj)+x_f
    (1,k,jj-1)*(x_p(4,k,jj)^2+x_p(5,k,jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)
    ^2)+x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k
    ,jj))-x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p
    (7,k,jj)))-(p_l(jj,2)*(2*x_p(6,k,jj)*x_f(1,k,jj-1)-2*x_p(5,k,jj)*
    x_f(2,k,jj-1)+2*x_p(4,k,jj)*x_f(3,k,jj-1))-p_l(jj,3)*(2*x_p(4,k,jj
    )*x_f(2,k,jj-1)-2*x_p(7,k,jj)*x_f(1,k,jj-1)+2*x_p(5,k,jj)*x_f(3,k,
    jj-1)))*(x_p(3,k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-
    x_p(6,k,jj)^2+x_p(7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,
    jj)+2*x_p(5,k,jj)*x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,
    k,jj)-2*x_p(6,k,jj)*x_p(7,k,jj)))+(p_l(jj,1)*(x_p(2,k,jj)+x_f(2,k,
    jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+x_p(6,k,jj)^2-x_p(7,k,jj)^2)-
    x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)-2*x_p(5,k,jj)*x_p(6,k,jj)
    )+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)+2*x_p(6,k,jj)*x_p(7,k,
    jj)))-p_l(jj,2)*(x_p(1,k,jj)+x_f(1,k,jj-1)*(x_p(4,k,jj)^2+x_p(5,k,
    jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)^2)+x_f(2,k,jj-1)*(2*x_p(4,k,jj)*
    x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k,jj))-x_f(3,k,jj-1)*(2*x_p(4,k,jj
    )*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p(7,k,jj))))*(2*x_p(4,k,jj)*x_f(1,k,
    jj-1)+2*x_p(7,k,jj)*x_f(2,k,jj-1)-2*x_p(6,k,jj)*x_f(3,k,jj-1))-(
    p_l(jj,2)*(x_p(3,k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-
    x_p(6,k,jj)^2+x_p(7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,
    jj)+2*x_p(5,k,jj)*x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,
    k,jj)-2*x_p(6,k,jj)*x_p(7,k,jj)))-p_l(jj,3)*(x_p(2,k,jj)+x_f(2,k,
    jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+x_p(6,k,jj)^2-x_p(7,k,jj)^2)-
    x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)-2*x_p(5,k,jj)*x_p(6,k,jj)
    )+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)+2*x_p(6,k,jj)*x_p(7,k,
    jj))))*(2*x_p(6,k,jj)*x_f(1,k,jj-1)-2*x_p(5,k,jj)*x_f(2,k,jj-1)+2*
    x_p(4,k,jj)*x_f(3,k,jj-1))-2*x_p(4,k,jj)*((981*T_tot{k}(3,2))/100-
    z_p{jj-1}(5,k))+2*x_p(7,k,jj)*((981*T_tot{k}(3,1))/100-z_p{jj
    -1}(4,k))-2*x_p(5,k,jj)*((981*T_tot{k}(3,3))/100-z_p{jj-1}(6,k))
    ,...
```

```
133                 (981*T_tot{k}(3,1)*x_p(6,k,jj))/50-(981*T_tot{k}(3,2)
    *x_p(5,k,jj))/50+(981*T_tot{k}(3,3)*x_p(4,k,jj))/50+(p_l(jj,1)*(2*
    x_p(6,k,jj)*x_f(1,k,jj-1)-2*x_p(5,k,jj)*x_f(2,k,jj-1)+2*x_p(4,k,jj
    )*x_f(3,k,jj-1))-p_l(jj,2)*(2*x_p(5,k,jj)*x_f(1,k,jj-1)+2*x_p(6,k,
    jj)*x_f(2,k,jj-1)+2*x_p(7,k,jj)*x_f(3,k,jj-1)))*(x_p(1,k,jj)+x_f
    (1,k,jj-1)*(x_p(4,k,jj)^2+x_p(5,k,jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)
    ^2)+x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k
    ,jj))-x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p
    (7,k,jj)))+(p_l(jj,2)*(2*x_p(4,k,jj)*x_f(2,k,jj-1)-2*x_p(7,k,jj)*
    x_f(1,k,jj-1)+2*x_p(5,k,jj)*x_f(3,k,jj-1))+p_l(jj,3)*(2*x_p(6,k,jj
    )*x_f(1,k,jj-1)-2*x_p(5,k,jj)*x_f(2,k,jj-1)+2*x_p(4,k,jj)*x_f(3,k,
    jj-1)))*(x_p(3,k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-
    x_p(6,k,jj)^2+x_p(7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,
    jj)+2*x_p(5,k,jj)*x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,
    k,jj)-2*x_p(6,k,jj)*x_p(7,k,jj)))+(p_l(jj,1)*(x_p(2,k,jj)+x_f(2,k,
    jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+x_p(6,k,jj)^2-x_p(7,k,jj)^2)-
    x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)-2*x_p(5,k,jj)*x_p(6,k,jj)
    )+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)+2*x_p(6,k,jj)*x_p(7,k,
    jj)))-p_l(jj,2)*(x_p(1,k,jj)+x_f(1,k,jj-1)*(x_p(4,k,jj)^2+x_p(5,k,
    jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)^2)+x_f(2,k,jj-1)*(2*x_p(4,k,jj)*
    x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k,jj))-x_f(3,k,jj-1)*(2*x_p(4,k,jj
    )*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p(7,k,jj))))*(2*x_p(5,k,jj)*x_f(1,k,
    jj-1)+2*x_p(6,k,jj)*x_f(2,k,jj-1)+2*x_p(7,k,jj)*x_f(3,k,jj-1))+(
    p_l(jj,2)*(x_p(3,k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-
    x_p(6,k,jj)^2+x_p(7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,
    jj)+2*x_p(5,k,jj)*x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,
    k,jj)-2*x_p(6,k,jj)*x_p(7,k,jj)))-p_l(jj,3)*(x_p(2,k,jj)+x_f(2,k,
    jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+x_p(6,k,jj)^2-x_p(7,k,jj)^2)-
    x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)-2*x_p(5,k,jj)*x_p(6,k,jj)
    )+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)+2*x_p(6,k,jj)*x_p(7,k,
    jj))))*(2*x_p(4,k,jj)*x_f(2,k,jj-1)-2*x_p(7,k,jj)*x_f(1,k,jj-1)+2*
    x_p(5,k,jj)*x_f(3,k,jj-1))-2*x_p(6,k,jj)*((981*T_tot{k}(3,1))/100-
    z_p{jj-1}(4,k))+2*x_p(5,k,jj)*((981*T_tot{k}(3,2))/100-z_p{jj
    -1}(5,k))-2*x_p(4,k,jj)*((981*T_tot{k}(3,3))/100-z_p{jj-1}(6,k))
    ,...
```

```
134                 (981*T_tot{k}(3,1)*x_p(5,k,jj))/50+(981*T_tot{k}(3,2)
    *x_p(6,k,jj))/50+(981*T_tot{k}(3,3)*x_p(7,k,jj))/50+(p_l(jj,1)*(2*
    x_p(5,k,jj)*x_f(1,k,jj-1)+2*x_p(6,k,jj)*x_f(2,k,jj-1)+2*x_p(7,k,jj
    )*x_f(3,k,jj-1))+p_l(jj,2)*(2*x_p(6,k,jj)*x_f(1,k,jj-1)-2*x_p(5,k,
    jj)*x_f(2,k,jj-1)+2*x_p(4,k,jj)*x_f(3,k,jj-1)))*(x_p(1,k,jj)+x_f
    (1,k,jj-1)*(x_p(4,k,jj)^2+x_p(5,k,jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)
    ^2)+x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k
    ,jj))-x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p
    (7,k,jj)))-(p_l(jj,2)*(2*x_p(4,k,jj)*x_f(1,k,jj-1)+2*x_p(7,k,jj)*
    x_f(2,k,jj-1)-2*x_p(6,k,jj)*x_f(3,k,jj-1))-p_l(jj,3)*(2*x_p(5,k,jj
    )*x_f(1,k,jj-1)+2*x_p(6,k,jj)*x_f(2,k,jj-1)+2*x_p(7,k,jj)*x_f(3,k,
    jj-1)))*(x_p(3,k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-
    x_p(6,k,jj)^2+x_p(7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,
    jj)+2*x_p(5,k,jj)*x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,
    k,jj)-2*x_p(6,k,jj)*x_p(7,k,jj)))-(p_l(jj,1)*(x_p(2,k,jj)+x_f(2,k,
    jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+x_p(6,k,jj)^2-x_p(7,k,jj)^2)-
    x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)-2*x_p(5,k,jj)*x_p(6,k,jj)
    )+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)+2*x_p(6,k,jj)*x_p(7,k,
    jj)))-p_l(jj,2)*(x_p(1,k,jj)+x_f(1,k,jj-1)*(x_p(4,k,jj)^2+x_p(5,k,
    jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)^2)+x_f(2,k,jj-1)*(2*x_p(4,k,jj)*
    x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k,jj))-x_f(3,k,jj-1)*(2*x_p(4,k,jj
    )*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p(7,k,jj))))*(2*x_p(6,k,jj)*x_f(1,k,
    jj-1)-2*x_p(5,k,jj)*x_f(2,k,jj-1)+2*x_p(4,k,jj)*x_f(3,k,jj-1))-(
    p_l(jj,2)*(x_p(3,k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-
    x_p(6,k,jj)^2+x_p(7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,
    jj)+2*x_p(5,k,jj)*x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,
    k,jj)-2*x_p(6,k,jj)*x_p(7,k,jj)))-p_l(jj,3)*(x_p(2,k,jj)+x_f(2,k,
    jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+x_p(6,k,jj)^2-x_p(7,k,jj)^2)-
    x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)-2*x_p(5,k,jj)*x_p(6,k,jj)
    )+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)+2*x_p(6,k,jj)*x_p(7,k,
    jj))))*(2*x_p(4,k,jj)*x_f(1,k,jj-1)+2*x_p(7,k,jj)*x_f(2,k,jj-1)-2*
    x_p(6,k,jj)*x_f(3,k,jj-1))-2*x_p(5,k,jj)*((981*T_tot{k}(3,1))/100-
    z_p{jj-1}(4,k))-2*x_p(6,k,jj)*((981*T_tot{k}(3,2))/100-z_p{jj
    -1}(5,k))-2*x_p(7,k,jj)*((981*T_tot{k}(3,3))/100-z_p{jj-1}(6,k))
    ,...
```

```
135                 (981*T_tot{k}(3,3)*x_p(6,k,jj))/50-(981*T_tot{k}(3,2)
    *x_p(7,k,jj))/50-(981*T_tot{k}(3,1)*x_p(4,k,jj))/50-(p_l(jj,1)*(2*
    x_p(4,k,jj)*x_f(1,k,jj-1)+2*x_p(7,k,jj)*x_f(2,k,jj-1)-2*x_p(6,k,jj
    )*x_f(3,k,jj-1))+p_l(jj,2)*(2*x_p(4,k,jj)*x_f(2,k,jj-1)-2*x_p(7,k,
    jj)*x_f(1,k,jj-1)+2*x_p(5,k,jj)*x_f(3,k,jj-1)))*(x_p(1,k,jj)+x_f
    (1,k,jj-1)*(x_p(4,k,jj)^2+x_p(5,k,jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)
    ^2)+x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k
    ,jj))-x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p
    (7,k,jj)))-(p_l(jj,2)*(2*x_p(5,k,jj)*x_f(1,k,jj-1)+2*x_p(6,k,jj)*
    x_f(2,k,jj-1)+2*x_p(7,k,jj)*x_f(3,k,jj-1))+p_l(jj,3)*(2*x_p(4,k,jj
    )*x_f(1,k,jj-1)+2*x_p(7,k,jj)*x_f(2,k,jj-1)-2*x_p(6,k,jj)*x_f(3,k,
    jj-1)))*(x_p(3,k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-
    x_p(6,k,jj)^2+x_p(7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,
    jj)+2*x_p(5,k,jj)*x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,
    k,jj)-2*x_p(6,k,jj)*x_p(7,k,jj)))+(p_l(jj,1)*(x_p(2,k,jj)+x_f(2,k,
    jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+x_p(6,k,jj)^2-x_p(7,k,jj)^2)-
    x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)-2*x_p(5,k,jj)*x_p(6,k,jj)
    )+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)+2*x_p(6,k,jj)*x_p(7,k,
    jj)))-p_l(jj,2)*(x_p(1,k,jj)+x_f(1,k,jj-1)*(x_p(4,k,jj)^2+x_p(5,k,
    jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)^2)+x_f(2,k,jj-1)*(2*x_p(4,k,jj)*
    x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k,jj))-x_f(3,k,jj-1)*(2*x_p(4,k,jj
    )*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p(7,k,jj))))*(2*x_p(4,k,jj)*x_f(2,k,
    jj-1)-2*x_p(7,k,jj)*x_f(1,k,jj-1)+2*x_p(5,k,jj)*x_f(3,k,jj-1))-(
    p_l(jj,2)*(x_p(3,k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-
    x_p(6,k,jj)^2+x_p(7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,
    jj)+2*x_p(5,k,jj)*x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,
    k,jj)-2*x_p(6,k,jj)*x_p(7,k,jj)))-p_l(jj,3)*(x_p(2,k,jj)+x_f(2,k,
    jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+x_p(6,k,jj)^2-x_p(7,k,jj)^2)-
    x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)-2*x_p(5,k,jj)*x_p(6,k,jj)
    )+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)+2*x_p(6,k,jj)*x_p(7,k,
    jj))))*(2*x_p(5,k,jj)*x_f(1,k,jj-1)+2*x_p(6,k,jj)*x_f(2,k,jj-1)+2*
    x_p(7,k,jj)*x_f(3,k,jj-1))+2*x_p(4,k,jj)*((981*T_tot{k}(3,1))/100-
    z_p{jj-1}(4,k))+2*x_p(7,k,jj)*((981*T_tot{k}(3,2))/100-z_p{jj
    -1}(5,k))-2*x_p(6,k,jj)*((981*T_tot{k}(3,3))/100-z_p{jj-1}(6,k))
136                 p_l(jj,1)*(x_p(3,k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-
    x_p(5,k,jj)^2-x_p(6,k,jj)^2+x_p(7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,
    k,jj)*x_p(6,k,jj)+2*x_p(5,k,jj)*x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p
    (4,k,jj)*x_p(5,k,jj)-2*x_p(6,k,jj)*x_p(7,k,jj)))-2*p_l(jj,3)*(x_p
    (1,k,jj)+x_f(1,k,jj-1)*(x_p(4,k,jj)^2+x_p(5,k,jj)^2-x_p(6,k,jj)^2-
    x_p(7,k,jj)^2)+x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)+2*x_p(5,k,
    jj)*x_p(6,k,jj))-x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,jj)-2*x_p(5,
    k,jj)*x_p(7,k,jj))) ,...
```

```
137                 p_l(jj,2)*(x_p(3,k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-
    x_p(5,k,jj)^2-x_p(6,k,jj)^2+x_p(7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,
    k,jj)*x_p(6,k,jj)+2*x_p(5,k,jj)*x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p
    (4,k,jj)*x_p(5,k,jj)-2*x_p(6,k,jj)*x_p(7,k,jj)))-2*p_l(jj,3)*(x_p
    (2,k,jj)+x_f(2,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+x_p(6,k,jj)^2-
    x_p(7,k,jj)^2)-x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)-2*x_p(5,k,
    jj)*x_p(6,k,jj))+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)+2*x_p(6,
    k,jj)*x_p(7,k,jj))),...
138                 p_l(jj,1)*(x_p(1,k,jj)+x_f(1,k,jj-1)*(x_p(4,k,jj)^2+
    x_p(5,k,jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)^2)+x_f(2,k,jj-1)*(2*x_p(4,
    k,jj)*x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k,jj))-x_f(3,k,jj-1)*(2*x_p
    (4,k,jj)*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p(7,k,jj)))+p_l(jj,2)*(x_p(2,
    k,jj)+x_f(2,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+x_p(6,k,jj)^2-x_p
    (7,k,jj)^2)-x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)-2*x_p(5,k,jj)
    *x_p(6,k,jj))+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)+2*x_p(6,k,
    jj)*x_p(7,k,jj))),...
```

```
139                 (981*T_tot{k}(3,1)*x_p(6,k,jj))/50-(981*T_tot{k}(3,2)
    *x_p(5,k,jj))/50+(981*T_tot{k}(3,3)*x_p(4,k,jj))/50+(p_l(jj,1)*(2*
    x_p(6,k,jj)*x_f(1,k,jj-1)-2*x_p(5,k,jj)*x_f(2,k,jj-1)+2*x_p(4,k,jj
    )*x_f(3,k,jj-1))-p_l(jj,3)*(2*x_p(4,k,jj)*x_f(1,k,jj-1)+2*x_p(7,k,
    jj)*x_f(2,k,jj-1)-2*x_p(6,k,jj)*x_f(3,k,jj-1)))*(x_p(1,k,jj)+x_f
    (1,k,jj-1)*(x_p(4,k,jj)^2+x_p(5,k,jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)
    ^2)+x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k
    ,jj))-x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p
    (7,k,jj)))+(p_l(jj,2)*(2*x_p(6,k,jj)*x_f(1,k,jj-1)-2*x_p(5,k,jj)*
    x_f(2,k,jj-1)+2*x_p(4,k,jj)*x_f(3,k,jj-1))-p_l(jj,3)*(2*x_p(4,k,jj
    )*x_f(2,k,jj-1)-2*x_p(7,k,jj)*x_f(1,k,jj-1)+2*x_p(5,k,jj)*x_f(3,k,
    jj-1)))*(x_p(2,k,jj)+x_f(2,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+
    x_p(6,k,jj)^2-x_p(7,k,jj)^2)-x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,
    jj)-2*x_p(5,k,jj)*x_p(6,k,jj))+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,
    k,jj)+2*x_p(6,k,jj)*x_p(7,k,jj)))+(p_l(jj,1)*(x_p(3,k,jj)+x_f(3,k,
    jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-x_p(6,k,jj)^2+x_p(7,k,jj)^2)+
    x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,jj)+2*x_p(5,k,jj)*x_p(7,k,jj)
    )-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)-2*x_p(6,k,jj)*x_p(7,k,
    jj)))-p_l(jj,3)*(x_p(1,k,jj)+x_f(1,k,jj-1)*(x_p(4,k,jj)^2+x_p(5,k,
    jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)^2)+x_f(2,k,jj-1)*(2*x_p(4,k,jj)*
    x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k,jj))-x_f(3,k,jj-1)*(2*x_p(4,k,jj
    )*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p(7,k,jj))))*(2*x_p(4,k,jj)*x_f(1,k,
    jj-1)+2*x_p(7,k,jj)*x_f(2,k,jj-1)-2*x_p(6,k,jj)*x_f(3,k,jj-1))+(
    p_l(jj,2)*(x_p(3,k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-
    x_p(6,k,jj)^2+x_p(7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,
    jj)+2*x_p(5,k,jj)*x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,
    k,jj)-2*x_p(6,k,jj)*x_p(7,k,jj)))-p_l(jj,3)*(x_p(2,k,jj)+x_f(2,k,
    jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+x_p(6,k,jj)^2-x_p(7,k,jj)^2)-
    x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)-2*x_p(5,k,jj)*x_p(6,k,jj)
    )+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)+2*x_p(6,k,jj)*x_p(7,k,
    jj))))*(2*x_p(4,k,jj)*x_f(2,k,jj-1)-2*x_p(7,k,jj)*x_f(1,k,jj-1)+2*
    x_p(5,k,jj)*x_f(3,k,jj-1))-2*x_p(6,k,jj)*((981*T_tot{k}(3,1))/100-
    z_p{jj-1}(4,k))+2*x_p(5,k,jj)*((981*T_tot{k}(3,2))/100-z_p{jj
    -1}(5,k))-2*x_p(4,k,jj)*((981*T_tot{k}(3,3))/100-z_p{jj-1}(6,k))
    ,...
```

```
140                 (981*T_tot{k}(3,1)*x_p(7,k,jj))/50-(981*T_tot{k}(3,2)
    *x_p(4,k,jj))/50-(981*T_tot{k}(3,3)*x_p(5,k,jj))/50-(p_l(jj,1)*(2*
    x_p(4,k,jj)*x_f(2,k,jj-1)-2*x_p(7,k,jj)*x_f(1,k,jj-1)+2*x_p(5,k,jj
    )*x_f(3,k,jj-1))+p_l(jj,3)*(2*x_p(5,k,jj)*x_f(1,k,jj-1)+2*x_p(6,k,
    jj)*x_f(2,k,jj-1)+2*x_p(7,k,jj)*x_f(3,k,jj-1)))*(x_p(1,k,jj)+x_f
    (1,k,jj-1)*(x_p(4,k,jj)^2+x_p(5,k,jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)
    ^2)+x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k
    ,jj))-x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p
    (7,k,jj)))-(p_l(jj,2)*(2*x_p(4,k,jj)*x_f(2,k,jj-1)-2*x_p(7,k,jj)*
    x_f(1,k,jj-1)+2*x_p(5,k,jj)*x_f(3,k,jj-1))+p_l(jj,3)*(2*x_p(6,k,jj
    )*x_f(1,k,jj-1)-2*x_p(5,k,jj)*x_f(2,k,jj-1)+2*x_p(4,k,jj)*x_f(3,k,
    jj-1)))*(x_p(2,k,jj)+x_f(2,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+
    x_p(6,k,jj)^2-x_p(7,k,jj)^2)-x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,
    jj)-2*x_p(5,k,jj)*x_p(6,k,jj))+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,
    k,jj)+2*x_p(6,k,jj)*x_p(7,k,jj)))+(p_l(jj,1)*(x_p(3,k,jj)+x_f(3,k,
    jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-x_p(6,k,jj)^2+x_p(7,k,jj)^2)+
    x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,jj)+2*x_p(5,k,jj)*x_p(7,k,jj)
    )-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)-2*x_p(6,k,jj)*x_p(7,k,
    jj)))-p_l(jj,3)*(x_p(1,k,jj)+x_f(1,k,jj-1)*(x_p(4,k,jj)^2+x_p(5,k,
    jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)^2)+x_f(2,k,jj-1)*(2*x_p(4,k,jj)*
    x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k,jj))-x_f(3,k,jj-1)*(2*x_p(4,k,jj
    )*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p(7,k,jj))))*(2*x_p(5,k,jj)*x_f(1,k,
    jj-1)+2*x_p(6,k,jj)*x_f(2,k,jj-1)+2*x_p(7,k,jj)*x_f(3,k,jj-1))+(
    p_l(jj,2)*(x_p(3,k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-
    x_p(6,k,jj)^2+x_p(7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,
    jj)+2*x_p(5,k,jj)*x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,
    k,jj)-2*x_p(6,k,jj)*x_p(7,k,jj)))-p_l(jj,3)*(x_p(2,k,jj)+x_f(2,k,
    jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+x_p(6,k,jj)^2-x_p(7,k,jj)^2)-
    x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)-2*x_p(5,k,jj)*x_p(6,k,jj)
    )+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)+2*x_p(6,k,jj)*x_p(7,k,
    jj))))*(2*x_p(6,k,jj)*x_f(1,k,jj-1)-2*x_p(5,k,jj)*x_f(2,k,jj-1)+2*
    x_p(4,k,jj)*x_f(3,k,jj-1))+2*x_p(4,k,jj)*((981*T_tot{k}(3,2))/100-
    z_p{jj-1}(5,k))-2*x_p(7,k,jj)*((981*T_tot{k}(3,1))/100-z_p{jj
    -1}(4,k))+2*x_p(5,k,jj)*((981*T_tot{k}(3,3))/100-z_p{jj-1}(6,k))
    ,...
```

```
141                 (981*T_tot{k}(3,1)*x_p(4,k,jj))/50+(981*T_tot{k}(3,2)
    *x_p(7,k,jj))/50-(981*T_tot{k}(3,3)*x_p(6,k,jj))/50+(p_l(jj,1)*(2*
    x_p(4,k,jj)*x_f(1,k,jj-1)+2*x_p(7,k,jj)*x_f(2,k,jj-1)-2*x_p(6,k,jj
    )*x_f(3,k,jj-1))+p_l(jj,3)*(2*x_p(6,k,jj)*x_f(1,k,jj-1)-2*x_p(5,k,
    jj)*x_f(2,k,jj-1)+2*x_p(4,k,jj)*x_f(3,k,jj-1)))*(x_p(1,k,jj)+x_f
    (1,k,jj-1)*(x_p(4,k,jj)^2+x_p(5,k,jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)
    ^2)+x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k
    ,jj))-x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p
    (7,k,jj)))+(p_l(jj,2)*(2*x_p(4,k,jj)*x_f(1,k,jj-1)+2*x_p(7,k,jj)*
    x_f(2,k,jj-1)-2*x_p(6,k,jj)*x_f(3,k,jj-1))-p_l(jj,3)*(2*x_p(5,k,jj
    )*x_f(1,k,jj-1)+2*x_p(6,k,jj)*x_f(2,k,jj-1)+2*x_p(7,k,jj)*x_f(3,k,
    jj-1)))*(x_p(2,k,jj)+x_f(2,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+
    x_p(6,k,jj)^2-x_p(7,k,jj)^2)-x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,
    jj)-2*x_p(5,k,jj)*x_p(6,k,jj))+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,
    k,jj)+2*x_p(6,k,jj)*x_p(7,k,jj)))-(p_l(jj,1)*(x_p(3,k,jj)+x_f(3,k,
    jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-x_p(6,k,jj)^2+x_p(7,k,jj)^2)+
    x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,jj)+2*x_p(5,k,jj)*x_p(7,k,jj)
    )-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)-2*x_p(6,k,jj)*x_p(7,k,
    jj)))-p_l(jj,3)*(x_p(1,k,jj)+x_f(1,k,jj-1)*(x_p(4,k,jj)^2+x_p(5,k,
    jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)^2)+x_f(2,k,jj-1)*(2*x_p(4,k,jj)*
    x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k,jj))-x_f(3,k,jj-1)*(2*x_p(4,k,jj
    )*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p(7,k,jj))))*(2*x_p(6,k,jj)*x_f(1,k,
    jj-1)-2*x_p(5,k,jj)*x_f(2,k,jj-1)+2*x_p(4,k,jj)*x_f(3,k,jj-1))+(
    p_l(jj,2)*(x_p(3,k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-
    x_p(6,k,jj)^2+x_p(7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,
    jj)+2*x_p(5,k,jj)*x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,
    k,jj)-2*x_p(6,k,jj)*x_p(7,k,jj)))-p_l(jj,3)*(x_p(2,k,jj)+x_f(2,k,
    jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+x_p(6,k,jj)^2-x_p(7,k,jj)^2)-
    x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)-2*x_p(5,k,jj)*x_p(6,k,jj)
    )+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)+2*x_p(6,k,jj)*x_p(7,k,
    jj))))*(2*x_p(5,k,jj)*x_f(1,k,jj-1)+2*x_p(6,k,jj)*x_f(2,k,jj-1)+2*
    x_p(7,k,jj)*x_f(3,k,jj-1))-2*x_p(4,k,jj)*((981*T_tot{k}(3,1))/100-
    z_p{jj-1}(4,k))-2*x_p(7,k,jj)*((981*T_tot{k}(3,2))/100-z_p{jj
    -1}(5,k))+2*x_p(6,k,jj)*((981*T_tot{k}(3,3))/100-z_p{jj-1}(6,k))
    ,...
```

```
142                 (981*T_tot{k}(3,1)*x_p(5,k,jj))/50+(981*T_tot{k}(3,2)
    *x_p(6,k,jj))/50+(981*T_tot{k}(3,3)*x_p(7,k,jj))/50+(p_l(jj,1)*(2*
    x_p(5,k,jj)*x_f(1,k,jj-1)+2*x_p(6,k,jj)*x_f(2,k,jj-1)+2*x_p(7,k,jj
    )*x_f(3,k,jj-1))-p_l(jj,3)*(2*x_p(4,k,jj)*x_f(2,k,jj-1)-2*x_p(7,k,
    jj)*x_f(1,k,jj-1)+2*x_p(5,k,jj)*x_f(3,k,jj-1)))*(x_p(1,k,jj)+x_f
    (1,k,jj-1)*(x_p(4,k,jj)^2+x_p(5,k,jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)
    ^2)+x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k
    ,jj))-x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p
    (7,k,jj)))+(p_l(jj,2)*(2*x_p(5,k,jj)*x_f(1,k,jj-1)+2*x_p(6,k,jj)*
    x_f(2,k,jj-1)+2*x_p(7,k,jj)*x_f(3,k,jj-1))+p_l(jj,3)*(2*x_p(4,k,jj
    )*x_f(1,k,jj-1)+2*x_p(7,k,jj)*x_f(2,k,jj-1)-2*x_p(6,k,jj)*x_f(3,k,
    jj-1)))*(x_p(2,k,jj)+x_f(2,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+
    x_p(6,k,jj)^2-x_p(7,k,jj)^2)-x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,
    jj)-2*x_p(5,k,jj)*x_p(6,k,jj))+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,
    k,jj)+2*x_p(6,k,jj)*x_p(7,k,jj)))+(p_l(jj,1)*(x_p(3,k,jj)+x_f(3,k,
    jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-x_p(6,k,jj)^2+x_p(7,k,jj)^2)+
    x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,jj)+2*x_p(5,k,jj)*x_p(7,k,jj)
    )-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)-2*x_p(6,k,jj)*x_p(7,k,
    jj)))-p_l(jj,3)*(x_p(1,k,jj)+x_f(1,k,jj-1)*(x_p(4,k,jj)^2+x_p(5,k,
    jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)^2)+x_f(2,k,jj-1)*(2*x_p(4,k,jj)*
    x_p(7,k,jj)+2*x_p(5,k,jj)*x_p(6,k,jj))-x_f(3,k,jj-1)*(2*x_p(4,k,jj
    )*x_p(6,k,jj)-2*x_p(5,k,jj)*x_p(7,k,jj))))*(2*x_p(4,k,jj)*x_f(2,k,
    jj-1)-2*x_p(7,k,jj)*x_f(1,k,jj-1)+2*x_p(5,k,jj)*x_f(3,k,jj-1))-(
    p_l(jj,2)*(x_p(3,k,jj)+x_f(3,k,jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2-
    x_p(6,k,jj)^2+x_p(7,k,jj)^2)+x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(6,k,
    jj)+2*x_p(5,k,jj)*x_p(7,k,jj))-x_f(2,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,
    k,jj)-2*x_p(6,k,jj)*x_p(7,k,jj)))-p_l(jj,3)*(x_p(2,k,jj)+x_f(2,k,
    jj-1)*(x_p(4,k,jj)^2-x_p(5,k,jj)^2+x_p(6,k,jj)^2-x_p(7,k,jj)^2)-
    x_f(1,k,jj-1)*(2*x_p(4,k,jj)*x_p(7,k,jj)-2*x_p(5,k,jj)*x_p(6,k,jj)
    )+x_f(3,k,jj-1)*(2*x_p(4,k,jj)*x_p(5,k,jj)+2*x_p(6,k,jj)*x_p(7,k,
    jj))))*(2*x_p(4,k,jj)*x_f(1,k,jj-1)+2*x_p(7,k,jj)*x_f(2,k,jj-1)-2*
    x_p(6,k,jj)*x_f(3,k,jj-1))-2*x_p(5,k,jj)*((981*T_tot{k}(3,1))/100-
    z_p{jj-1}(4,k))-2*x_p(6,k,jj)*((981*T_tot{k}(3,2))/100-z_p{jj
    -1}(5,k))-2*x_p(7,k,jj)*((981*T_tot{k}(3,3))/100-z_p{jj-1}(6,k))
143                 0,0,0,x_p(4,k,jj)/(x_p(4,k,jj)^2+x_p(5,k,jj)^2+x_p(6,
    k,jj)^2+x_p(7,k,jj)^2)^(1/2),x_p(5,k,jj)/(x_p(4,k,jj)^2+x_p(5,k,jj
    )^2+x_p(6,k,jj)^2+x_p(7,k,jj)^2)^(1/2),x_p(6,k,jj)/(x_p(4,k,jj)^2+
    x_p(5,k,jj)^2+x_p(6,k,jj)^2+x_p(7,k,jj)^2)^(1/2),x_p(7,k,jj)/(x_p
    (4,k,jj)^2+x_p(5,k,jj)^2+x_p(6,k,jj)^2+x_p(7,k,jj)^2)^(1/2)
144                 ];
145         end
146         % observation test
147         M1=obsv(F_hat,H_hat); rM1=rank(M1);
148         if rM1~=7
149             index_not_obsv(length(index_not_obsv)+1)=k;
150             % more accurate method in "Nonlinear controllability and
    observability"
151         end
152         % discretization
```

```
153         Fd=(eye(7)+1/2*F_hat*Ts)*inv(eye(7)-1/2*F_hat*Ts);
154         % Kalman gain
155         K0=P{jj,k}*H_hat'*inv(H_hat*P{jj,k}*H_hat'+Rc);
156         P0=(eye(n)-K0*H_hat)*P{jj,k};
157         % T_predicted:
158         T_p=[x_p(4,k,jj)^2+x_p(5,k,jj)^2-x_p(6,k,jj)^2-x_p(7,k,jj)^2
    2*(x_p(5,k,jj)*x_p(6,k,jj)-x_p(4,k,jj)*x_p(7,k,jj))        2*(x_p(5,k
    ,jj)*x_p(7,k,jj)+x_p(4,k,jj)*x_p(6,k,jj))
159              2*(x_p(5,k,jj)*x_p(6,k,jj)+x_p(4,k,jj)*x_p(7,k,jj))
    x_p(4,k,jj)^2-x_p(5,k,jj)^2+x_p(6,k,jj)^2-x_p(7,k,jj)^2  2*(x_p(6,k
    ,jj)*x_p(7,k,jj)-x_p(4,k,jj)*x_p(5,k,jj))
160              2*(x_p(5,k,jj)*x_p(7,k,jj)-x_p(4,k,jj)*x_p(6,k,jj))
    2*(x_p(6,k,jj)*x_p(7,k,jj)+x_p(4,k,jj)*x_p(5,k,jj))        x_p(4,k,jj
    )^2-x_p(5,k,jj)^2-x_p(6,k,jj)^2+x_p(7,k,jj)^2];
161         % output equation
162         if jj==1
163             z_p{jj}(:,k)=[
164                 x_p(1:3,k,jj)
165                 [0,-w_dot(3,k,jj),w_dot(2,k,jj)
166                     w_dot(3,k,jj),0,-w_dot(1,k,jj)
167                     -w_dot(2,k,jj),w_dot(1,k,jj),0]*p_l(jj,:)'+[0,-
    x_p(3,k,jj),x_p(2,k,jj)
168                         x_p(3,k,jj),0,-x_p(1,k,jj)
169                         -x_p(2,k,jj),x_p(1,k,jj),0]*[[0,-x_p(3,k,jj),
    x_p(2,k,jj)
170                             x_p(3,k,jj),0,-x_p(1,k,jj)
171                             -x_p(2,k,jj),x_p(1,k,jj),0]*p_l(jj,:)']+(
    R2*T_p)*g_vec
172                 (x_p(4,k,jj)^2+x_p(5,k,jj)^2+x_p(6,k,jj)^2+x_p(7,k,jj
    )^2)^(1/2)
173                 ];
174
175         else
176             % being the RFs aligned the posture is fully described by
     T_tot
177             % Az(:,k)=R{jj,k}'*(z_p{k,jj-1}(4:6)-T_tot{k}'*{g})+acc
    (:,k)+(T_tot{k}*R)'*{g}, T_tot=T_tot_old if not updated yet
178             % z_predicted vector
179             z_pv(1:3)=[% w=i^(i-1)_[R]*(W{jj-1}(1:3,k)+w(1:3,k))
180                 T_p'*x_f(1:3,k,jj-1)+x_p(1:3,k,jj)
181                 ];
182             % a=i^(i-1)_[R]*(i-1)^a_(i-1)+{w'}x{p}+{w}x({w}x{p}) in i
    ^{RF}
183             % T_tot is not updated yet, so it is referred to jj-1
184             z_pv(4:7)=[
185                 T_p'*(z_p{jj-1}(4:6,k)-T_tot{k}'*g_vec)+[0,-w_dot(3,k
    ,jj),w_dot(2,k,jj)
186                     w_dot(3,k,jj),0,-w_dot(1,k,jj)
```

```
187                         -w_dot(2,k,jj),w_dot(1,k,jj),0]*p_l(jj,:)'+[0,-
    z_pv(3),z_pv(2)
188                             z_pv(3),0,-z_pv(1)
189                             -z_pv(2),z_pv(1),0]*[[0,-z_pv(3),z_pv(2)
190                                 z_pv(3),0,-z_pv(1)
191                                 -z_pv(2),z_pv(1),0]*p_l(jj,:)']+(T_tot{k
    }*T_p)'*g_vec
192                     (x_p(4,k,jj)^2+x_p(5,k,jj)^2+x_p(6,k,jj)^2+x_p(7,k,jj
    )^2)^(1/2)
193                     ];
194                 z_p{jj}(:,k)=z_pv(:);
195             end
196             % measurement error
197             e(:,k)=Zo{jj}(:,k)-z_p{jj}(:,k)+v_sens(:,k);
198             % state update
199             x_f(:,k,jj)=x_p(:,k,jj)+K0*e(:,k);
200             % quaternion normalization
201             x_f(4:7,k,jj)=x_f(4:7,k,jj)/(x_f(4:7,k,jj)'*x_f(4:7,k,jj))
    ^(1/2);
202             % state covariance matrix update
203             P{jj,k+1}=Fd*P0*Fd'+Qc;
204             % state prediction
205             x_p(1:3,k+1,jj)=x_f(1:3,k,jj)+Ts*[-1/tau(1)*x_f(1,k,jj)
206                                               -1/tau(2)*x_f(2,k,jj)
207                                               -1/tau(3)*x_f(3,k,jj)];
208             x_p(4:7,k+1,jj)=x_f(4:7,k,jj)+Ts/2*[-x_f(5,k,jj),-x_f(6,k,jj)
    ,-x_f(7,k,jj)
209                                                 x_f(4,k,jj),-x_f(7,k,jj),x_f
    (6,k,jj)
210                                                 x_f(7,k,jj),x_f(4,k,jj),-x_f
    (5,k,jj)
211                                                -x_f(6,k,jj),x_f(5,k,jj),x_f(4,
    k,jj)]*[x_f(1,k,jj),x_f(2,k,jj),x_f(3,k,jj)]';%/(x_f(4,k,jj)^2+x_f
    (5,k,jj)^2+x_f(6,k,jj)^2+x_f(7,k,jj)^2)^(1/2);
212             % quaternion normalization
213             x_p(4:7,k+1,jj)=x_p(4:7,k+1,jj)/(x_p(4:7,k+1,jj)'*x_p(4:7,k
    +1,jj))^(1/2);%/norm(x_p(4:7,k+1));
214             % angular acceleration estimate
215             w_dot(:,k+1,jj)=[0,0,0]';%(x_p(1:3,k+1)-x_p(1:3,k))/Ts;
216             % Euler angles conversion. T_filtered:
217             T_f{jj}=[x_f(4,k,jj)^2+x_f(5,k,jj)^2-x_f(6,k,jj)^2-x_f(7,k,jj
    )^2  2*(x_f(5,k,jj)*x_f(6,k,jj)-x_f(4,k,jj)*x_f(7,k,jj))      2*(x_f
    (5,k,jj)*x_f(7,k,jj)+x_f(4,k,jj)*x_f(6,k,jj))
218                 2*(x_f(5,k,jj)*x_f(6,k,jj)+x_f(4,k,jj)*x_f(7,k,jj))
      x_f(4,k,jj)^2-x_f(5,k,jj)^2+x_f(6,k,jj)^2-x_f(7,k,jj)^2  2*(x_f
    (6,k,jj)*x_f(7,k,jj)-x_f(4,k,jj)*x_f(5,k,jj))
219                 2*(x_f(5,k,jj)*x_f(7,k,jj)-x_f(4,k,jj)*x_f(6,k,jj))
      2*(x_f(6,k,jj)*x_f(7,k,jj)+x_f(4,k,jj)*x_f(5,k,jj))      x_f(4,k,
    jj)^2-x_f(5,k,jj)^2-x_f(6,k,jj)^2+x_f(7,k,jj)^2];
```

```
220         % Total rotation matrix update
221         T_tot{k}=T_tot{k}*T_f{jj};
222         % TB 321 angle estimates
223         %
224         phi_est(jj,k)=atan2(T_f{jj}(3,2),T_f{jj}(3,3));
225         theta_est(jj,k)=atan2(-T_f{jj}(3,1),sin(phi_est(jj,k))*T_f{jj}(3,2)+cos(phi_est(jj,k))*T_f{jj}(3,3));
226         psi_est(jj,k)=atan2(-cos(phi_est(jj,k))*T_f{jj}(1,2)+sin(phi_est(jj,k))*T_f{jj}(1,3),cos(phi_est(jj,k))*T_f{jj}(2,2)-sin(phi_est(jj,k))*T_f{jj}(2,3));
227         % TB 312
228         %{
229         theta_est(jj,k)=atan2(T_f{jj}(3,1),T_f{jj}(3,3));%atan2(-T_f{jj}(3,1),sqrt(T_f{jj}(1,1)^2+T_f{jj}(2,1)^2));
230         psi_est(jj,k)=atan2(T_f{jj}(1,2),T_f{jj}(2,2));%atan2(T_f{jj}(2,1),T_f{jj}(1,1));
231         phi_est(jj,k)=asin(-T_f{jj}(3,2));%atan2(T_f{jj}(3,2),T_f{jj}(3,3));
232         %
233         % TB 132
234         phi_est(jj,k)=atan2(-T_f{jj}(1,3),T_f{jj}(1,1));
235         theta_est(jj,k)=asin(T_f{jj}(1,2));
236         psi_est(jj,k)=atan2(-T_f{jj}(3,2),T_f{jj}(2,2));
237         %}
238     end
239 end
240 % performance evaluation
241 k0=100; % discard the first k0 samples in the performance evaluations
242 for jj=1:L
243     for ii=1:n
244         RMSE_z(ii,jj)=norm(Zo{jj}(ii,k0+1:N-1)-z_p{jj}(ii,k0+1:N-1))/sqrt(N-1-k0);
245     end
246 end
247
248
249 for jj=1:L
250 figure
251     subplot(311), plot(tvec,phi_est(jj,:)*180/pi), grid on, title(['\phi^',num2str(jj),'(t)']), xlabel('[s]'), ylabel('[deg]')
252     subplot(312), plot(tvec,theta_est(jj,:)*180/pi), grid on, title(['\theta^',num2str(jj),'(t)']), xlabel('[s]'), ylabel('[deg]')
253     subplot(313), plot(tvec,psi_est(jj,:)*180/pi), grid on, title(['\psi^',num2str(jj),'(t)']), xlabel('[s]'), ylabel('[deg]')
254 end
255
256 %%
257 for jj=1:L
258 figure
```

```
259     subplot(311), plot(tvec,x_f(1,1:N,jj)), grid on, title(['x_1^',
        num2str(jj),'(t)'])
260     subplot(312), plot(tvec,x_f(2,1:N,jj)), grid on, title(['x_2^',
        num2str(jj),'(t)'])
261     subplot(313), plot(tvec,x_f(3,1:N,jj)), grid on, title(['x_3^',
        num2str(jj),'(t)'])
262 end
263
264 for jj=1:L
265 figure
266     subplot(411), plot(tvec,x_f(4,1:N,jj)), grid on, title(['x_4^',
        num2str(jj),'(t)'])
267     subplot(412), plot(tvec,x_f(5,1:N,jj)), grid on, title(['x_5^',
        num2str(jj),'(t)'])
268     subplot(413), plot(tvec,x_f(6,1:N,jj)), grid on, title(['x_6^',
        num2str(jj),'(t)'])
269     subplot(414), plot(tvec,x_f(7,1:N,jj)), grid on, title(['x_7^',
        num2str(jj),'(t)'])
270 end
```

B.4 Figures Generation

```
1  %figures
2  %% still
3  close all
4  for jj=1:L
5  figure
6      subplot(311), plot(tvec,phi_est(jj,:)*180/pi), grid on, title(['\
       phi^',num2str(jj),'(t)']), xlabel('[s]'), ylabel('[deg]'), hold on
7      subplot(312), plot(tvec,theta_est(jj,:)*180/pi), grid on, title([
       '\theta^',num2str(jj),'(t)']), xlabel('[s]'), ylabel('[deg]'),hold
        on
8      subplot(313), plot(tvec,psi_est(jj,:)*180/pi), grid on, title(['\
       psi^',num2str(jj),'(t)']), xlabel('[s]'), ylabel('[deg]'),hold on
9  end
10 figure(1)
11     subplot(311),plot(tvec,phi(2,1:N)*180/pi), legend('phi_{est}','
       phi')
12     subplot(312),plot(tvec,theta(2,1:N)*180/pi), legend('theta_{est}'
       ,'theta')
13     subplot(313),plot(tvec,psi(2,1:N)*180/pi), legend('psi_{est}','
       psi')
14
15 figure(2)
```

```
16      subplot(311),plot(tvec,-90*ones(size(tvec))), legend('phi_{est}',
        'phi')
17      subplot(312),plot(tvec,theta(5,1:N)*180/pi), legend('theta_{est}'
        ,'theta')
18      subplot(313),plot(tvec,psi(5,1:N)*180/pi), legend('psi_{est}','
        psi')
19
20  figure(3)
21      subplot(311),plot(tvec,phi(5,1:N)*180/pi), legend('phi_{est}','
        phi')
22      subplot(312),plot(tvec,theta(7,1:N)*180/pi), legend('theta_{est}'
        ,'theta')
23      subplot(313),plot(tvec,psi(7,1:N)*180/pi), legend('psi_{est}','
        psi')
24
25  %% pronation/supination of the forearm
26  close all
27
28  for jj=1:L
29  figure
30      subplot(311), plot(tvec,phi_est(jj,:)*180/pi), grid on, title(['\
        phi^',num2str(jj),'(t)']), xlabel('[s]'), ylabel('[deg]'), hold on
31      subplot(312), plot(tvec,theta_est(jj,:)*180/pi), grid on, title([
        '\theta^',num2str(jj),'(t)']), xlabel('[s]'), ylabel('[deg]'),hold
         on
32      subplot(313), plot(tvec,psi_est(jj,:)*180/pi), grid on, title(['\
        psi^',num2str(jj),'(t)']), xlabel('[s]'), ylabel('[deg]'),hold on
33  end
34  figure(1)
35      subplot(311),plot(tvec,phi(2,1:N)*180/pi), legend('phi_{est}','
        phi')
36      subplot(312),plot(tvec,theta(2,1:N)*180/pi), legend('theta_{est}'
        ,'theta')
37      subplot(313),plot(tvec,psi(2,1:N)*180/pi), legend('psi_{est}','
        psi')
38
39  figure(2)
40      subplot(311),plot(tvec,-90*ones(size(tvec))), legend('phi_{est}',
        'phi')
41      subplot(312),plot(tvec,theta(5,1:N)*180/pi), legend('theta_{est}'
        ,'theta')
42      subplot(313),plot(tvec,psi(5,1:N)*180/pi), legend('psi_{est}','
        psi')
43
44  figure(3)
45      subplot(311),plot(tvec,phi(5,1:N)*180/pi), legend('phi_{est}','
        phi')
46      subplot(312),plot(tvec,-psi(7,1:N)*180/pi), legend('theta_{est}',
        'theta')
```

```
47      subplot(313),plot(tvec,theta(7,1:N)*180/pi), legend('psi_{est}','
        psi')
48
49  %% flexion/extension of the forearm
50  close all
51
52  for jj=1:L
53  figure
54      subplot(311), plot(tvec,phi_est(jj,:)*180/pi), grid on, title(['\
        phi^',num2str(jj),'(t)']), xlabel('[s]'), ylabel('[deg]'), hold on
55      subplot(312), plot(tvec,theta_est(jj,:)*180/pi), grid on, title([
        '\theta^',num2str(jj),'(t)']), xlabel('[s]'), ylabel('[deg]'),hold
         on
56      subplot(313), plot(tvec,psi_est(jj,:)*180/pi), grid on, title(['\
        psi^',num2str(jj),'(t)']), xlabel('[s]'), ylabel('[deg]'),hold on
57  end
58  figure(1)
59      subplot(311),plot(tvec,phi(2,1:N)*180/pi), legend('phi_{est}','
        phi')
60      subplot(312),plot(tvec,theta(2,1:N)*180/pi), legend('theta_{est}'
        ,'theta')
61      subplot(313),plot(tvec,psi(2,1:N)*180/pi), legend('psi_{est}','
        psi')
62
63  figure(2)
64      subplot(311),plot(tvec,-90*ones(size(tvec))), legend('phi_{est}',
        'phi')
65      subplot(312),plot(tvec,theta(5,1:N)*180/pi), legend('theta_{est}'
        ,'theta')
66      subplot(313),plot(tvec,psi(5,1:N)*180/pi), legend('psi_{est}','
        psi')
67
68  figure(3)
69      subplot(311),plot(tvec,psi(5,1:N)*180/pi), legend('phi_{est}','
        phi')
70      subplot(312),plot(tvec,theta(5,1:N)*180/pi), legend('theta_{est}'
        ,'theta')
71      subplot(313),plot(tvec,phi(5,1:N)*180/pi), legend('psi_{est}','
        psi')
72
73  %% rotation of the upper arm
74
75  close all
76
77  for jj=1:L
78  figure
79      subplot(311), plot(tvec,phi_est(jj,:)*180/pi), grid on, title(['\
        phi^',num2str(jj),'(t)']), xlabel('[s]'), ylabel('[deg]'), hold on
```

```
80      subplot(312), plot(tvec,theta_est(jj,:)*180/pi), grid on, title(['\theta^',num2str(jj),'(t)']), xlabel('[s]'), ylabel('[deg]'),hold on
81      subplot(313), plot(tvec,psi_est(jj,:)*180/pi), grid on, title(['\psi^',num2str(jj),'(t)']), xlabel('[s]'), ylabel('[deg]'),hold on
82  end
83  figure(1)
84      subplot(311),plot(tvec,phi(2,1:N)*180/pi), legend('phi_{est}','phi')
85      subplot(312),plot(tvec,theta(2,1:N)*180/pi), legend('theta_{est}','theta')
86      subplot(313),plot(tvec,psi(2,1:N)*180/pi), legend('psi_{est}','psi')
87
88  figure(2)
89      subplot(311),plot(tvec,-90*ones(size(tvec))), legend('phi_{est}','phi')
90      subplot(312),plot(tvec,theta(4,1:N)*180/pi), legend('theta_{est}','theta')
91      subplot(313),plot(tvec,psi(4,1:N)*180/pi), legend('psi_{est}','psi')
92
93  figure(3)
94      subplot(311),plot(tvec,phi(4,1:N)*180/pi), legend('phi_{est}','phi')
95      subplot(312),plot(tvec,theta(4,1:N)*180/pi), legend('theta_{est}','theta')
96      subplot(313),plot(tvec,psi(4,1:N)*180/pi), legend('psi_{est}','psi')
97
98  %% flexion/extension of the upper arm
99
100 close all
101
102 for jj=1:L
103 figure
104     subplot(311), plot(tvec,phi_est(jj,:)*180/pi), grid on, title(['\phi^',num2str(jj),'(t)']), xlabel('[s]'), ylabel('[deg]'), hold on
105     subplot(312), plot(tvec,theta_est(jj,:)*180/pi), grid on, title(['\theta^',num2str(jj),'(t)']), xlabel('[s]'), ylabel('[deg]'),hold on
106     subplot(313), plot(tvec,psi_est(jj,:)*180/pi), grid on, title(['\psi^',num2str(jj),'(t)']), xlabel('[s]'), ylabel('[deg]'),hold on
107 end
108 figure(1)
109     subplot(311),plot(tvec,phi(2,1:N)*180/pi), legend('phi_{est}','phi')
110     subplot(312),plot(tvec,theta(2,1:N)*180/pi), legend('theta_{est}','theta')
```

```
111     subplot(313),plot(tvec,psi(2,1:N)*180/pi), legend('psi_{est}','
        psi')
112
113 figure(2)
114     subplot(311),plot(tvec,-90*ones(size(tvec))), legend('phi_{est}',
        'phi')
115     subplot(312),plot(tvec,psi(3,1:N)*180/pi), legend('theta_{est}','
        theta')
116     subplot(313),plot(tvec,theta(3,1:N)*180/pi), legend('psi_{est}','
        psi')
117
118 figure(3)
119     subplot(311),plot(tvec,phi(7,1:N)*180/pi), legend('phi_{est}','
        phi')
120     subplot(312),plot(tvec,theta(7,1:N)*180/pi), legend('theta_{est}'
        ,'theta')
121     subplot(313),plot(tvec,psi(7,1:N)*180/pi), legend('psi_{est}','
        psi')
122
123 %% elevation/depression of the clavicle
124
125 close all
126
127 for jj=1:L
128 figure
129     subplot(311), plot(tvec,phi_est(jj,:)*180/pi), grid on, title(['\
        phi^',num2str(jj),'(t)']), xlabel('[s]'), ylabel('[deg]'), hold on
130     subplot(312), plot(tvec,theta_est(jj,:)*180/pi), grid on, title([
        '\theta^',num2str(jj),'(t)']), xlabel('[s]'), ylabel('[deg]'),hold
         on
131     subplot(313), plot(tvec,psi_est(jj,:)*180/pi), grid on, title(['\
        psi^',num2str(jj),'(t)']), xlabel('[s]'), ylabel('[deg]'),hold on
132 end
133 figure(1)
134     subplot(311),plot(tvec,phi(2,1:N)*180/pi), legend('phi_{est}','
        phi')
135     subplot(312),plot(tvec,psi(2,1:N)*180/pi), legend('theta_{est}','
        theta')
136     subplot(313),plot(tvec,theta(2,1:N)*180/pi), legend('psi_{est}','
        psi')
137
138 figure(2)
139     subplot(311),plot(tvec,-90*ones(size(tvec))), legend('phi_{est}',
        'phi')
140     subplot(312),plot(tvec,theta(5,1:N)*180/pi), legend('theta_{est}'
        ,'theta')
141     subplot(313),plot(tvec,psi(5,1:N)*180/pi), legend('psi_{est}','
        psi')
142
```

```
143 figure(3)
144     subplot(311),plot(tvec,phi(7,1:N)*180/pi), legend('phi_{est}','
    phi')
145     subplot(312),plot(tvec,theta(7,1:N)*180/pi), legend('theta_{est}'
    ,'theta')
146     subplot(313),plot(tvec,psi(7,1:N)*180/pi), legend('psi_{est}','
    psi')
147
148 %% protraction/retraction of the clavicle
149
150 close all
151
152 for jj=1:L
153 figure
154     subplot(311), plot(tvec,phi_est(jj,:)*180/pi), grid on, title(['\
    phi^',num2str(jj),'(t)']), xlabel('[s]'), ylabel('[deg]'), hold on
155     subplot(312), plot(tvec,theta_est(jj,:)*180/pi), grid on, title([
    '\theta^',num2str(jj),'(t)']), xlabel('[s]'), ylabel('[deg]'),hold
     on
156     subplot(313), plot(tvec,psi_est(jj,:)*180/pi), grid on, title(['\
    psi^',num2str(jj),'(t)']), xlabel('[s]'), ylabel('[deg]'),hold on
157 end
158 figure(1)
159     subplot(311),plot(tvec,psi(1,1:N)*180/pi), legend('phi_{est}','
    phi')
160     subplot(312),plot(tvec,theta(1,1:N)*180/pi), legend('theta_{est}'
    ,'theta')
161     subplot(313),plot(tvec,phi(1,1:N)*180/pi), legend('psi_{est}','
    psi')
162
163 figure(2)
164     subplot(311),plot(tvec,-90*ones(size(tvec))), legend('phi_{est}',
    'phi')
165     subplot(312),plot(tvec,theta(5,1:N)*180/pi), legend('theta_{est}'
    ,'theta')
166     subplot(313),plot(tvec,psi(5,1:N)*180/pi), legend('psi_{est}','
    psi')
167
168 figure(3)
169     subplot(311),plot(tvec,phi(7,1:N)*180/pi), legend('phi_{est}','
    phi')
170     subplot(312),plot(tvec,theta(7,1:N)*180/pi), legend('theta_{est}'
    ,'theta')
171     subplot(313),plot(tvec,psi(7,1:N)*180/pi), legend('psi_{est}','
    psi')
```

Bibliography

[1] Lin Meng, Mengjuan Chen, Baihan Li, Feng He, Rui Xu, and Dong Ming. «An Inertial-Based Upper-Limb Motion Assessment Model: Performance Validation Across Various Motion Tasks». In: *IEEE Sensors Journal* 23.7 (2023), pp. 7168–7177 (cit. on pp. 1, 18).

[2] Huiyu Zhou and Huosheng Hu. «Inertial sensors for motion detection of human upper limbs». In: *Sensor Review* 27.2 (2007), pp. 151–158 (cit. on p. 1).

[3] Simone Sabatelli, Marco Galgani, Luca Fanucci, and Alessandro Rocchi. «A double stage Kalman filter for sensor fusion and orientation tracking in 9D IMU». In: *2012 IEEE Sensors Applications Symposium Proceedings*. IEEE. 2012, pp. 1–5 (cit. on pp. 3, 39).

[4] Gert Kwakkel et al. «Motor rehabilitation after stroke: European Stroke Organisation (ESO) consensus-based definition and guiding framework». In: *European Stroke Journal* 8.4 (2023), pp. 880–894 (cit. on p. 3).

[5] Alexandre Balbinot, Jonas Crauss Rodrigues de Freitas, and Daniel Santos Côrrea. «Use of inertial sensors as devices for upper limb motor monitoring exercises for motor rehabilitation». In: *Health and Technology* 5 (2015), pp. 91–102 (cit. on pp. 3, 39).

[6] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: Modelling, Planning and Control*. Springer, 2009 (cit. on pp. 8, 9, 23, 24).

[7] Farah Afiqa Mohd Ghazali, Md Nazibul Hasan, Tariq Rehman, Marwan Nafea, Mohamed Sultan Mohamed Ali, and Kenichi Takahata. «MEMS actuators for biomedical applications: a review». In: *Journal of Micromechanics and Microengineering* 30.7 (2020), p. 073001 (cit. on p. 13).

[8] Gerald M Lerner. «Three-axis attitude determination». In: *Spacecraft attitude determination and control* 73 (1978), pp. 420–428 (cit. on p. 14).

[9] Malcolm David Shuster and S D_ Oh. «Three-axis attitude determination from vector observations». In: *Journal of guidance and Control* 4.1 (1981), pp. 70–77 (cit. on p. 14).

[10] Joao L Marins. «An extended Kalman filter for quaternion-based attitude estimation». PhD thesis. Monterey, California. Naval Postgraduate School, 2000 (cit. on pp. 16, 29, 36).

[11] Xiaoping Yun and Eric R Bachmann. «Design, implementation, and experimental results of a quaternion-based Kalman filter for human body motion tracking». In: *IEEE transactions on Robotics* 22.6 (2006), pp. 1216–1227 (cit. on pp. 16, 28, 35, 38).

[12] Robert Mahony, Tarek Hamel, and Jean-Michel Pflimlin. «Nonlinear complementary filters on the special orthogonal group». In: *IEEE Transactions on automatic control* 53.5 (2008), pp. 1203–1218 (cit. on p. 17).

[13] Wenchuan Wei, Keiko Kurita, Jilong Kuang, and Alex Gao. «Real-time 3D arm motion tracking using the 6-axis IMU sensor of a smartwatch». In: *2021 IEEE 17th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*. IEEE. 2021, pp. 1–4 (cit. on p. 19).

[14] Ge Wu et al. «ISB recommendation on definitions of joint coordinate systems of various joints for the reporting of human joint motion—Part II: shoulder, elbow, wrist and hand». In: *Journal of biomechanics* 38.5 (2005), pp. 981–992 (cit. on pp. 20, 22).

[15] Maria Cristina Bisi, Rita Stagni, Alessio Caroselli, and Angelo Cappello. «Anatomical calibration for wearable motion capture systems: Video calibrated anatomical system technique». In: *Medical engineering & physics* 37.8 (2015), pp. 813–819 (cit. on p. 21).

[16] Gustav Höglund, Helena Grip, and Fredrik Öhberg. «The importance of inertial measurement unit placement in assessing upper limb motion». In: *Medical Engineering & Physics* 92 (2021), pp. 1–9 (cit. on p. 22).

[17] Lorenzo Peppoloni, Alessandro Filippeschi, Emanuele Ruffaldi, and Carlo Alberto Avizzano. «A novel 7 degrees of freedom model for upper limb kinematic reconstruction based on wearable sensors». In: *2013 IEEE 11th international symposium on intelligent systems and informatics (SISY)*. IEEE. 2013, pp. 105–110 (cit. on pp. 23, 68).

[18] Vittore Cossalter, Mauro Da Lio, Alberto Doria, et al. «Meccanica applicata alle macchine». In: (1994) (cit. on p. 27).

[19] Jack B Kuipers. *Quaternions and rotation sequences: a primer with applications to orbits, aerospace, and virtual reality*. Princeton university press, 1999 (cit. on p. 28).

[20] Anthony Kim and MF Golnaraghi. «A quaternion-based orientation estimation algorithm using an inertial measurement unit». In: *PLANS 2004. Position Location and Navigation Symposium (IEEE Cat. No. 04CH37556)*. IEEE. 2004, pp. 268–272 (cit. on p. 29).

[21] Angelo Maria Sabatini. «Estimating three-dimensional orientation of human body parts by inertial/magnetic sensing». In: *Sensors* 11.2 (2011), pp. 1489–1525 (cit. on pp. 29, 68).

[22] Chi-Tsong Chen. *Linear system theory and design*. Saunders college publishing, 1984 (cit. on p. 32).

[23] Youngjoo Kim, Hyochoong Bang, et al. «Introduction to Kalman filter and its applications». In: *Introduction and Implementations of the Kalman Filter* 1 (2018), pp. 1–16 (cit. on p. 34).

[24] Hugh Durrant-Whyte et al. «Introduction to estimation and the Kalman filter». In: *Australian Centre for Field Robotics* 28.3 (2001), pp. 65–94 (cit. on p. 34).

[25] Greg Welch, Gary Bishop, et al. «An introduction to the Kalman filter». In: (1995) (cit. on p. 35).

[26] Awad H Al-Mohy and Nicholas J Higham. «A new scaling and squaring algorithm for the matrix exponential». In: *SIAM Journal on Matrix Analysis and Applications* 31.3 (2010), pp. 970–989 (cit. on p. 36).

[27] Conor Lynch, Michael J OMahony, and Ted Scully. «Simplified method to derive the Kalman Filter covariance matrices to predict wind speeds from a NWP model». In: *Energy procedia* 62 (2014), pp. 676–685 (cit. on p. 38).

[28] Javier González-Alonso, David Oviedo-Pastor, Héctor J Aguado, Francisco J Dıaz-Pernas, David González-Ortega, and Mario Martınez-Zarzuela. «Custom IMU-based wearable system for robust 2.4 GHz wireless human body parts orientation tracking and 3D movement visualization on an avatar». In: *Sensors* 21.19 (2021), p. 6642 (cit. on p. 39).

[29] Almas Shintemirov, Tasbolat Taunyazov, Bukeikhan Omarali, Aigerim Nurbayeva, Anton Kim, Askhat Bukeyev, and Matteo Rubagotti. «An open-source 7-DOF wireless human arm motion-tracking system for use in robotics research». In: *Sensors* 20.11 (2020), p. 3082 (cit. on p. 39).

[30] Angelo Maria Sabatini. «Kalman-filter-based orientation determination using inertial/magnetic sensors: Observability analysis and performance evaluation». In: *Sensors* 11.10 (2011), pp. 9182–9206 (cit. on p. 40).

[31] Robert Hermann and Arthur Krener. «Nonlinear controllability and observability». In: *IEEE Transactions on automatic control* 22.5 (1977), pp. 728–740 (cit. on p. 40).

[32] Luigi Truppa, Elena Bergamini, Pietro Garofalo, Marta Costantini, Carmelo Fiorentino, Giuseppe Vannozzi, Angelo Maria Sabatini, and Andrea Mannini. «An innovative sensor fusion algorithm for motion tracking with on-line bias compensation: Application to joint angles estimation in yoga». In: *IEEE Sensors Journal* 21.19 (2021), pp. 21285–21294 (cit. on p. 68).