GAIT ANALYSIS USING INERTIAL MEASUREMENT UNITS AS SENSORS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

SELİN KİRDİŞ GEMİCİ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
BIOMEDICAL ENGINEERING

SEPTEMBER 2022

Approval of the thesis:

GAIT ANALYSIS USING INERTIAL MEASUREMENT UNITS AS SENSORS

submitted by **SELİN KİRDİŞ GEMİCİ** in partial fulfillment of the requirements for the degree of **Master of Science** i**n Biomedical Engineering, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Vilda Purutçuoğlu
Head of the Department, **Biomedical Engineering** _____

Assoc. Prof. Dr. Ergin Tönük
Supervisor, **Biomedical Engineering, METU** _____

Prof. Dr. Melek Güneş Yavuzer
Co-Supervisor, **Institute of Health Sciences, Haliç University** _____

Examining Committee Members:

Assoc. Prof. Dr. M. Bülent Özer
Mechanical Engineering, METU _____

Assoc. Prof. Dr. Ergin Tönük
Biomedical Engineering, METU _____

Prof. Dr. İlhami Kuru
Orthopedics and Traumatology, Başkent University _____

Asst. Prof. Dr. Altuğ Özçelikkale
Mechanical Engineering, METU _____

Asst. Prof. Dr. Kutluk Bilge Arıkan
Mechanical Engineering, TED University _____

Date: 16.09.2022

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name Last name : Selin Kirdiş Gemici

Signature :

ABSTRACT

GAIT ANALYSIS USING INERTIAL MEASUREMENT UNITS AS SENSORS

Kirdiş Gemici, Selin
Master of Science, Biomedical Engineering
Supervisor: Assoc. Prof. Dr. Ergin Tönük
Co-Supervisor: Prof. Dr. Melek Güneş Yavuzer

September 2022, 132 pages

Examination of joint kinematics and kinetics after diseases that affect mobility provides information about the movement capacity of the person. The most widely used systems in this field are opto-electronic systems. However, these systems are expensive and cannot be used outside the laboratory. For this reason, wearable technologies (e.g. systems containing inertial sensors), which are relatively new systems, have started to take the place of these systems. Wearable technologies are more accessible to users and allow for long-term movement monitoring of the person outside the laboratory.

Over the years, many researchers have made efforts to establish and improve the gait analysis system at METU Biomechanics Laboratory. After the opto-electronic gait analysis, the KISS system, a new Inertial Measurement Units based gait analysis system was started to be developed.

In the first phase of this study, the kinematic and kinetic gait analysis systems were developed using data acquired utilizing Inertial Measurement Units (IMUs) and force plates. The software is based on an open source software, OpenSim.

After adopting the software for available IMU sensors for lower and upper extremities, the developed systems were examined and validated qualitatively by testing single and multiple sensors with static test setups and additional experiments with a human subject. Data was collected, processed and interpreted with the help of a lower extremity test setup using seven IMUs. Moreover, gait and upper extremity data were collected from a human subject.

The results of the kinematic gait analysis indicate that drift has been observed because of the unwanted motion of the pelvis sensor (which is the main sensor) during movement. The drift problem of the results was solved by using the slope correction code. Overall, analysis results showed that patterns of the kinematic data are consistent with the literature. The differences between some of the final results and the values in the literature were thought to be due to sensor sensitivity and the subject's unique gait pattern.

Although the patterns of the kinetic data are also similar to the patterns of the data in the literature, two main problems were observed. First, it was noticed that some of the curves were in the reversed order of those in the literature. It is thought that this is caused by a difference between the coordinate systems. Secondly, the experimental results were found to be larger than those in the literature. However, after analyzing the data of different subjects with different weights, it was seen that the data was consistent in itself.

Keywords: Gait Analysis, Motion Analysis, Inertial Measurement Unit, OpenSim, Force Plate

ÖZ

EYLEMSİZLİK ÖLÇER KULLANARAK YÜRÜYÜŞ ANALİZİ YAPILMASI

Kirdiş Gemici, Selin
Yüksek Lisans, Biyomedikal Mühendisliği
Tez Yöneticisi: Doç. Dr. Ergin Tönük
Ortak Tez Yöneticisi: Prof. Dr. Melek Güneş Yavuzer

Eylül 2022, 132 sayfa

Kişinin hareket fonksiyonunu etkileyen hastalıklar sonrası eklem kinematik ve kinetiklerinin incelenmesi, kişinin hareket kapasitesi hakkında bilgi vermektedir. Bu konuda en yaygın kullanılan sistemler optoelektronik sistemlerdir. Ancak bu sistemler pahalıdır ve laboratuvar dışında kullanım imkanı bulunmamaktadır. Bu nedenle göreceli olarak yeni sistemler olan giyilebilir teknolojiler(örneğin eylemsizlik sensörü içeren sistemler) bu sistemlerin yerini almaya başlamıştır. Giyilebilir teknolojiler kullanıcılar tarafından daha ulaşılabilirdir ve laboratuvar dışında kişinin uzun süreli hareket takibine imkan sağlayan teknolojilerdir.

Yıllar içinde bir çok araştırmacı METU Biyomekanik Laboratuvarında bulunan yürüyüş analizi sistemini kurmak ve geliştirmek için çaba göstermiştir. İlk yürüyüş analizi sistemi olan KİSS sisteminin ardından, eylemsizlik ölçer tabanlı yeni yürüyüş analiz sistemi çalışılmaya başlanmıştır.

Bu çalışmanın ilk bölümünde, kinematik ve kinetik yürüyüş analiz sistemleri, eylemsizlik ölçer sensörleri ve kuvvet platformları kullanılarak toplanan veriler

kullanılarak geliştirilmiştir. Yazılım açık kaynak kodlu bir yazılım olan OpenSim'e dayanmaktadır.

Alt ve üst ekstremiteler için mevcut eylemsizlik ölçer sensörleri kullanılarak yazılım uyarlandıktan sonra geliştirilen sistemler, tek ve çoklu sensörlerin statik deney düzenekleri ile denenmesi ve tek denekle gerçekleştirilen insanlı deneyler aracılığıyla incelenmiş ve kavramsal doğrulama gerçekleştirilmiştir. Yedi eylemsizlik ölçer sensörü kullanılarak bir alt ekstremite test düzeneği yardımıyla veriler toplanmış, işlenmiş ve yorumlanmıştır. Ayrıca, bir insan denekten yürüyüş ve üst ekstremite hareket verileri toplanmıştır.

Kinematik yürüyüş analizinin sonuçları, hareket sırasında pelvis sensörünün (ana sensör) istenmeyen hareketi nedeniyle kayma gözlemlendiğini göstermektedir. Sonuçlardaki kayma sorunu eğim düzeltici kod kullanılarak çözülmüştür. Genel olarak, analiz sonuçları kinematik veri desenlerinin literatürle tutarlı olduğunu göstermiştir. Bazı sonuçlar ile literatürdeki değerler arasındaki farkların sensör hassasiyeti ve deneğin kendine özgü yürüyüş biçiminden kaynaklandığı düşünülmüştür.

Kinetik verilerin paternleri de literatürdeki verilerin paternleriyle benzer olsa da temel olarak iki sorun gözlemlenmiştir. İlk sorun, bazı eğrilerin literatürdekinin tersi olduğunun fark edilmiş olmasıdır. Bu durumun koordinat sistemi farkından olduğu düşünülmektedir. İkinci olarak, deneysel sonuçların literatürdekilerden daha yüksek değerler olduğu görülmüştür. Ancak farklı ağırlıklara sahip farklı deneklerin verilerinin incelenmesi sonucunda, verilerin kendi içinde tutarlı olduğu görülmüştür.

Anahtar Kelimeler: Yürüyüş Analizi, Hareket Analizi, Eylemsizlik Ölçer, OpenSim, Kuvvet Platformu

To my family

ACKNOWLEDGMENTS

Firstly, I wish to express my deepest gratitude to my supervisor Assoc. Prof. Dr. Ergin Tönük for his support and guidance. His advices led me to learn many valuable things throughout the research.

I would like to thank Prof. Dr. Melek Güneş Yavuzer for her suggestions and comments. The IMU hardware was developed by Adasoft Danışmanlık ve Yazılım Hizmetleri (METU-Technopolis) through a TÜBİTAK TEYDEB grant, assistance of Mr. Aktan Orhan, Oğuz Şenbaklavacı, Yılmaz Özçalışkan is gratefully acknowledged. Also, I wish to thank the staff of METU Mechanical Engineering Machine Shop for their support in the manufacturing process carried out for the thesis.

Finally, I would like to express my gratitude to my family for their encouragement and understanding.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Human voluntary movement is one of the most examined multidisciplinary science subjects. Since muscles, bones, joints, nerves, spinal cord and brain are involved in this process, it is the topic of three main scientific disciplines; anatomy, physiology, and biomechanics [1]. Anatomy examines these structures and the relationships between them, physiology studies the functions of these structures, and biomechanics investigates these biological structures and their functions using classical mechanics perspective.

Locomotion is a branch of voluntary movement. It is one of the essential functions of the body, and it is significant for all humans to perform daily activities. It is an ability or task that a person moves herself/himself from one place to another independently [2]. It includes lower limb movements like walking, running, and jumping. According to Cech and Martin [2], the primary requirements of locomotion are adequate dynamic balance to keep human posture stable, control and strength to continue locomotion and overcome the forces (e.g. gravity force).

According to Andriacchi and Alexander [3], the intention of human locomotion studies changed over time. These studies are motivated by survival questions in the Paleolithic Era, wishing to understand harmony in the universe for Greek philosophers, define diseases and find treatments for our century [3]. Additionally, this analysis is not only used in clinical studies but also used in sports, robotics research and training [4]. These intentions lead scientists to develop analysis methods and devices and define parameters to study human locomotion, specifically human gait.

Gait analysis, a subbranch of biomechanics, examines human gait with the help of the science of classical mechanics. It consists of successive steps. The first step is that kinematic and kinetic data are obtained from a human during locomotion. In the following stages, measured data is applied to the biomechanical model, and computed parameters like joint angles, joint forces, and moments are used for clinical, robotic or sports research.

During early times, gait analysis methods were semi-subjective and depended on the observation of clinicians. With the help of modern technology, more objective and quantitative devices and methods are available. Modern devices used in gait analysis can be classified into three groups: non-wearable sensor systems (NWS), wearable sensor systems (WS) and hybrid systems [5]. According to Muro-de-la-Herran et al. [5], NWS is located on the subject's arranged walkway, WS is located on the patient's body, and the hybrid systems use both NWS and WS. NWS systems are floor sensors and image processing-based sensors [5]. WS systems consist of diverse groups of sensors that include Inertial Measurement Units (IMUs), force sensors, goniometers, etc. The main difference between NWS and WS is that WS can be used outside the laboratory to track patients' daily-life routines. These sensory systems are examined in detail in Chapter 2.

Normal human gait is characterized by gait parameters such as step length, joint angles, ground reaction forces, gait phases etc. Depending on the field of study, various parameters are selected to analyze gait. These parameters and the selection of the parameters are investigated in Chapter 2.

1.1 Motivation and Scope of the Research

Starting from Güler [6], many scientists have worked to improve the gait analysis system (KISS) at METU biomechanics laboratory, which is the first gait analysis system in Turkey that is constructed using of the shelf components. Güler constructed a biomechanical model, and he built marker and force plate setups for

kinematic and kinetic gait analysis, respectively [6]. In later studies, laboratory setup was advanced, used and compared with commercial systems by Shafiq [7], Karpat [8], Afşar [9], Söylemez [10], Civek [11], Kafalı [12], Erer [13] and Biçer [14]. Biçer also established a new kinematic gait analysis system which is the IMU-based system replacing the opto-electronic kinematic motion capture system [14]. This system consists of five IMU sensors and worked with Matlab, C++ and OpenSim programs. Additionally, OpenSim has been used not only for the kinematics but also for the kinetics part of the gait analysis. Despite the proof of the concept of the system, there are several issues that need to be improved in the gait analysis system. This thesis intends to investigate the following parts:

In the first part of the study, OpenSim environment capabilities and software improvements are investigated. With the rapid development of IMUs, they are used more often in clinical studies. Gait analysis software, like OpenSim, have been updated to include IMU data input. OpenSim 3.3 was used for Biçer's thesis [14]. This version does not include the IMU plugin, but OpenSim 4.3 has it. It yields faster and easier to calculate results for gait analysis. Additionally, the accuracy and precision of kinetic data are directly related to kinematic data. So kinetic analysis is more reliable with the improvements in software. Moreover, OpenSim 4.3 allows animating the movement as a result of a more accurate analysis with IMUs. It is thought that animation will help researchers and clinicians to observe movements from different perspectives and identify diseases more precisely.

The second part of the study examines system enhancements. IMU system in METU biomechanics laboratory consists of five IMU sensors; two placed at the proximal part of legs, two at the distal part of legs, and one at the pelvis. This setup was not receiving information from feet. Most of the clinical studies for lower extremities and gait analyses that worked with IMUs include seven IMUs placed at previously explained locations in addition to two feet, and analysis is more reliable when a system gets information from feet.

Thirdly, the kinetic analysis system with force plates was improved. Ground reaction force data were calibrated, coordinate system transformation was performed and sent as an input to OpenSim Inverse Dynamics Tool with movement data from OpenSim IMU Inverse Kinematics Setup.

Additional to the lower extremity kinematic analysis system, the kinematic analysis system was developed for the human upper extremity. This system consists of seven IMU sensors placed at a subject's hands, proximal part of right and left arms, distal part of right and left arms and spine.

In the last part of the thesis, experiments were conducted to test lower extremity and upper extremity kinematic analysis systems. Also, kinetic analysis of the lower extremity system was tested via experiments. The system was validated qualitatively as a result of experiments.

CHAPTER 2

LITERATURE REVIEW

Human movement analysis is a method that is frequently used in areas such as athletic performance, man-machine interfaces and games [15]. Especially it is widely used in the field of medical diagnostics, where a diagnosis can be made by comparing the kinematic and kinetic parameters of the patient with the data of a healthy individual.

In this thesis, human motion analysis system is studied under two categories. These categories are kinematic and kinetic analyses. In order to understand kinematic and kinetic analyses, it is important and necessary to examine the movements that human limbs can perform and the forces and moments that cause or arise due to these movements.

The movements that can be performed by the lower limb are presented in Figure 2.1 [17]. The human pelvis can carry out tilt motion in the sagittal plane, oblique motion in the frontal plane and rotation in the transverse plane. The hip joint can perform abduction-adduction, flexion-extension and internal-external rotation. The knee joint can perform flexion-extension, varus-valgus and internal-external rotation movements. As for the ankle joint, it can perform plantarflexion-dorsiflexion, varus-valgus and internal-external rotation movements.

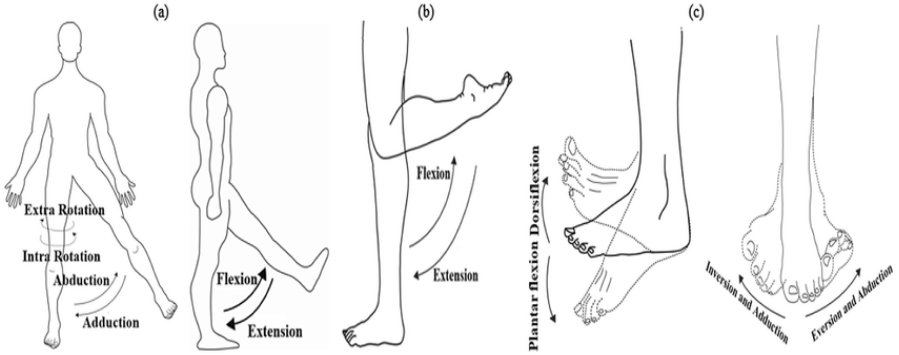


Figure 2.1. (a) Hip joint, (b)knee joint, (c) ankle joint movements [17]

Upper limb movements are presented in Figure 2.2 [18]. Flexion-extension in the sagittal plane, abduction-adduction in the frontal plane, medial-lateral rotation and horizontal flexion-extension in the transverse plane can be performed by the shoulder joint of a human. The elbow joint can carry out flexion-extension movements in the sagittal plane and pronation-supination movements in the transverse plane. Additionally, flexion-extension and abduction-adduction can be performed by the wrist joint of a human.

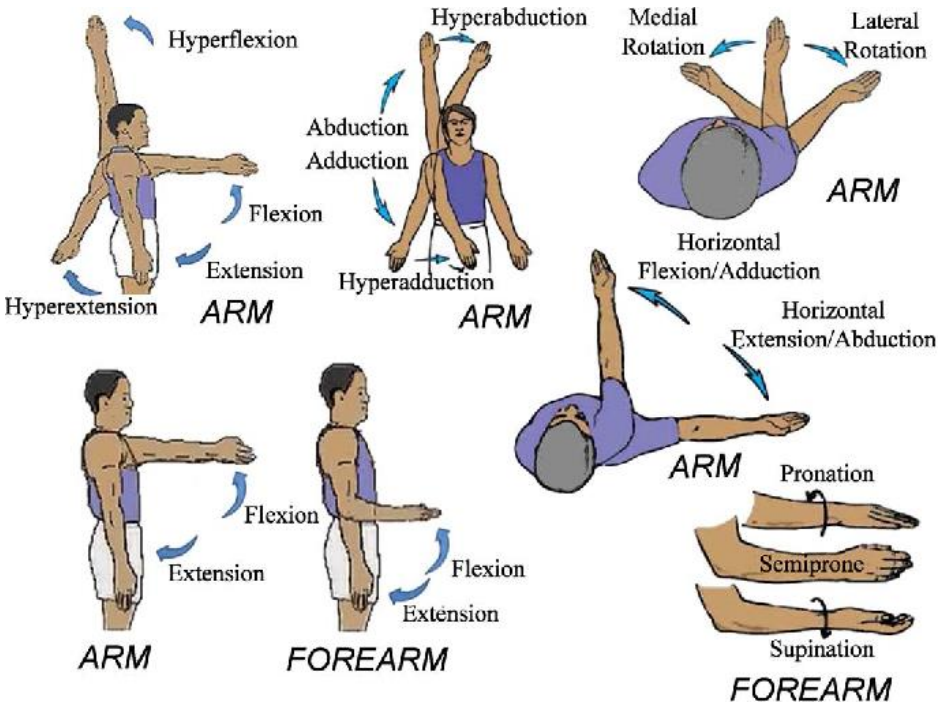


Figure 2.2. Shoulder and elbow joints' movements [18]

Upper and lower extremity movements are important for kinematic analysis. In kinematic analysis, analysis of motion is performed by examining measurements such as angle, angular velocity and acceleration of the related joints. In this thesis, kinematic analysis is computed for the upper and lower extremities separately. Although kinematic analysis of the lower extremity in the literature generally focuses on gait analysis, analysis of many different motions can be performed, especially with the development of out-of-laboratory systems [16].

Kinetic analysis is the concept of analyzing the joints that perform motion in terms of force and moment. Ground reaction force (GRF) is a significant parameter for kinetic analysis, especially in terms of kinetic gait analysis. It can be used to calculate the moments of related joints. Also, GRF is used to interpret the state of irregularity of pathological gait and phases of normal gait [49]. The vertical ground reaction force graph in relation to gait phases for one gait cycle is presented in Figure 2.3 [19].

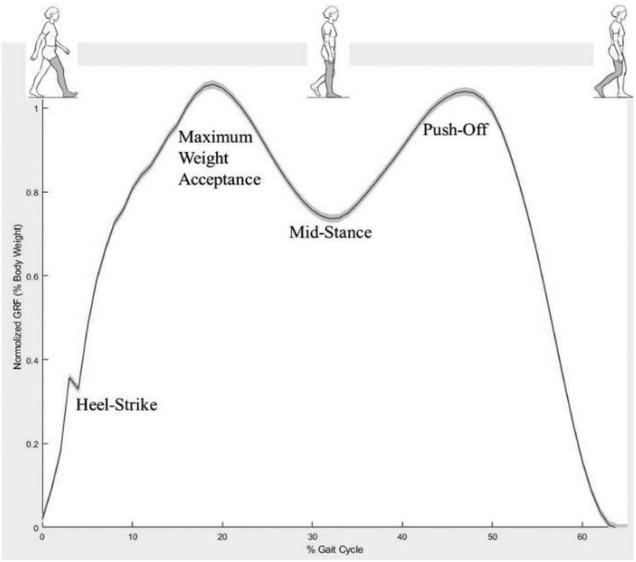


Figure 2.3. Normalized vertical GRF [19]

A typical gait cycle and gait phases are shared in Figure 2.4 [20]. In the stance phase, the foot is in contact with the ground. In the swing phase, the foot examined

during the gait cycle has no contact with the ground. Normal human locomotion is performed by repeating the gait cycle for two legs in sequence.

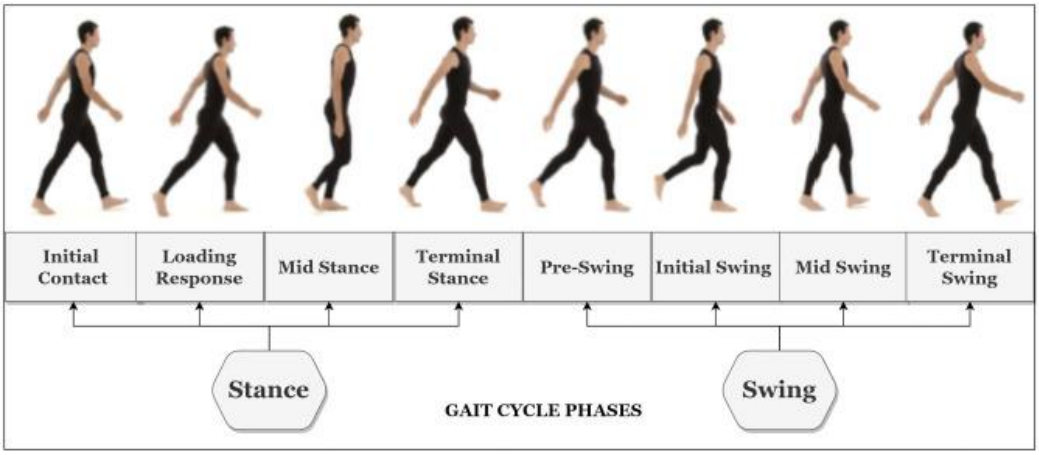


Figure 2.4. Gait cycle phases [20]

2.1 Kinematic Analysis

2.1.1 Systems

There are different systems that can be used to perform kinematic analysis. Among these systems, opto-electronic systems are classified as the gold-standard. It is known that the most widely used brand is VICON [21]. In these systems, active or passive makers are connected to anatomical landmarks on the subject's body, and the movement of the subject is monitored in this way. The drawbacks of opto-electronic systems are that adequate lighting is very crucial, it is not possible to work outside the laboratory, and a high sampling rate is required for fast movements [20].

The second method is Motion Capture Cameras. In these systems, movements are derived from sequences of photographs. The disadvantages of these systems are the same as the opto-electronic systems [20].

The third method is magnetic systems. Such systems do not depend on light, but operation is uncomfortable for the subject and it can be easily interfered by magnetic materials [20].

Alternatively, electro-goniometers can be used for gait analysis. The advantage of this system is that the output can be used directly for computation, but the use is uncomfortable for the subject, and the measurement quality for the lower limb is poor [20].

The last method is inertial systems. For these systems, Inertial Measurement Units are used. The sensors include a gyroscope, an accelerometer and a magnetometer. IMU sensors are cheap, lightweight and can be used out of a laboratory. The fact that these systems can be used out-of-laboratory is an advantage not only for gait analysis but also for tracking many daily life activities (e.g. rowing, stair climbing, cycling, etc.) [22]. The downsides of these systems are the limited battery life, computational complexity, and the possibility that the connected sensors may disturb the user [20]. In addition, it has been reported that drift problems are observed in this sensor type [21]. It has been stated that the drift problem can be observed especially in the horizontal axes of the gyroscope sensor [22].

2.1.2 Parameters

The main aim of kinematic analysis is to obtain kinematic parameters. The obtained parameters are then used to compare subjects (e.g., healthy subjects and patients). The most widely used parameters for gait analysis are spatiotemporal parameters (e.g., step length, step time, stance phase, swing phase, stride length, stride time, cadence, velocity, step duration) and joint angular kinematics [23], [24].

In this thesis, a system that computes joint angular kinematics has been studied, and the data in the literature has been examined in this context. Figure 2.5, from Fukuchi et al. [46], presents joint angular kinematics of twenty-four young adults (age 27.6 ± 4.4 years) during walking. In these graphs, each curve represents a walking speed.

Light blue represents slow speed (30% less than preferred speed), and dark blue indicates fast speed (30% more than preferred speed). The dashed line is the users' preferred walking speed.

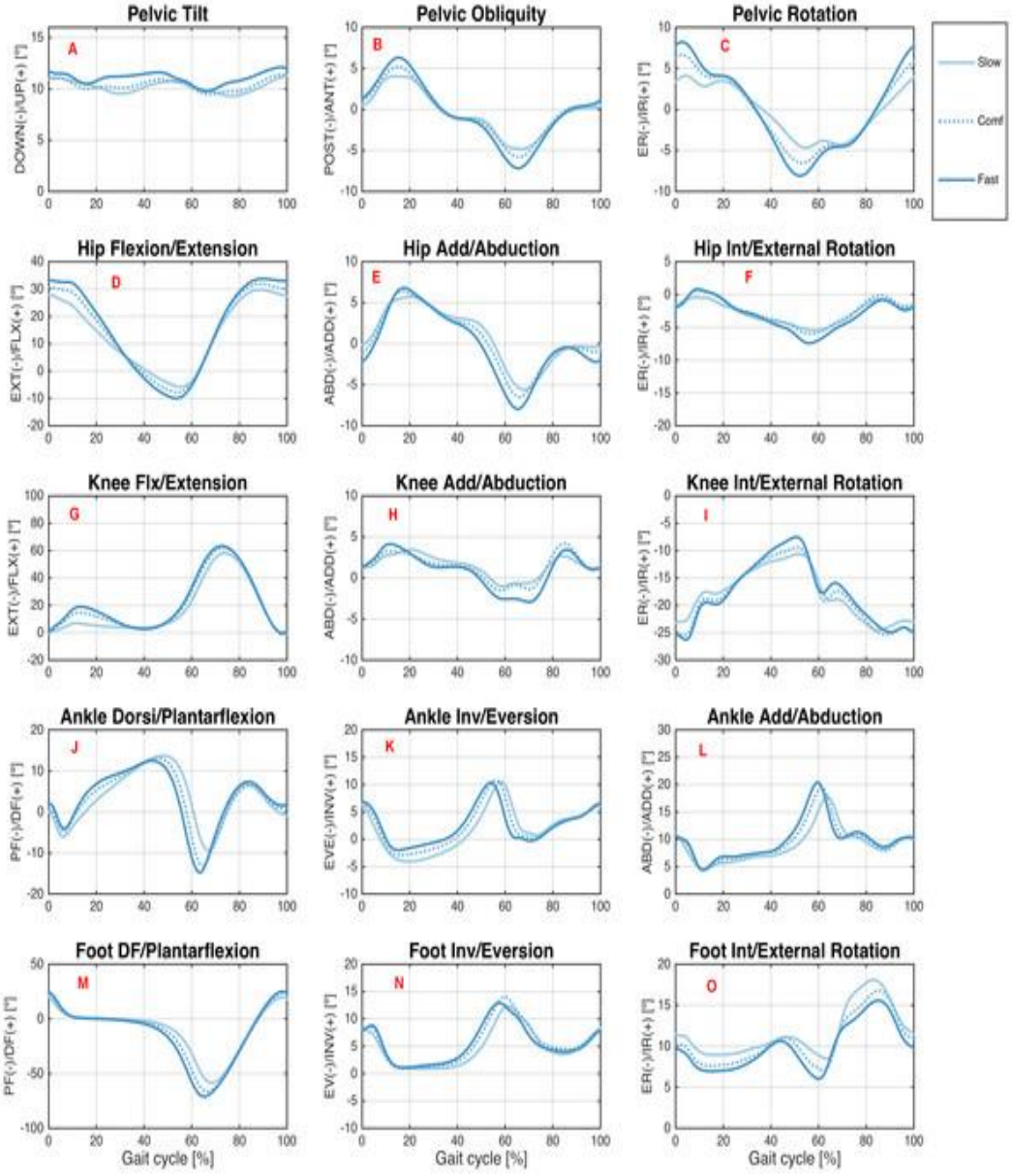


Figure 2.5. Joint angular kinematics of twenty-four healthy young adults [46]

In order to obtain kinematic parameters such as joint angles, it is necessary to find the orientation of the segments connected by the joints. There are various methods to determine the attitude of these rigid bodies, and the most widely used approach is Euler angles.

Euler Angles is a method used to find the rotation of a rigid body in three axes. Angle and the axis of rotation are defined in this method, and sequence is significant. The main disadvantages of this method are that Euler angles suffer from singularities, they depend on the order of rotation and when attitude over time is analyzed, they are less accurate than quaternions [14], [29]. Therefore, quaternions have been studied. Quaternions do not have singularity problems and are a successful method to represent angular velocity. The downside of the quaternions is that the four quaternion parameters have no physical meaning. Therefore, it is difficult to interpret quaternions. Also, it must be in unity norm to represent a rotation [29].

Usually, the analysis does not examine a single instantaneous rigid body attitude but rather monitors the changes of incremental variables over time. This is called real-time attitude estimation of the rigid body. Generally, these systems use data from the previous time point to estimate the pose of the rigid body at the next time point.

The most popular method used in the biomechanics field is the Kalman Filter. There are several different types of Kalman Filter, such as indirect Kalman filter (IKF), extended Kalman filter (EKF), unscented Kalman filter (UKF) [28]. IKF scheme uses the error state for this estimation; EKF linearizes the non-linear system with a truncated Taylor series expansion approach [28]. The UKF uses prior information for this computation [28]. Due to the complexity of these methods, new methods like Madgwick Algorithm [31] and Mahony algorithm [30] have been proposed. The Madgwick Algorithm, that is used in this thesis, is presented in detail in Section 3.4.

2.2 Kinetic Analysis

2.2.1 Systems

Sensors that measure force or pressure are used to perform kinetic analysis. The most widely used type of such systems are force platforms. To analyze gait, force platforms are placed on the subject's walkway, and the ground reaction force of the subject is calculated by measuring force and moment in three axes. Force platforms usually have load cells placed at each corner of two metal plates to measure the forces and moments in each axis [20]. The moments of the subject's joints are then calculated and compared to the joint moments of a healthy individual. The main drawback of such systems is that they require an expensive setup, and it is not possible to use them out of the laboratory [20].

Scientists have developed force shoes as a possible solution to the inability to use force platforms outside the laboratory. In this kind of system, force sensors are attached to the sole of the subject's shoes so that the user's gait can be monitored [20]. The main challenge of force shoes is that the unevenness of the surface leads to a decrease in measurement efficiency [20]. Moreover, the kinematic data of the subject is needed for accurate data processing [20].

Another option is to use pressure mats for kinetic analysis. Pressure mats have sensors positioned in a pad. They are inexpensive and portable. Nevertheless, a proper laboratory setup is required. Additionally, the scan rate decreases as the resolution increases [20].

2.2.2 Parameters

The purpose of kinetic gait analysis is to obtain inter-personally comparable kinetic parameters. Examples of these kinetic parameters are vertical GRF, peak propulsion, peak braking, joint moments and joint powers [25], [26].

Figure 2.6 is presented to illustrate the use of vertical GRF to obtain joint moments [27]. For this case, the force vector is in front of the hip and ankle joints and behind the knee joint. In order to maintain balance, internal moments are required to compensate the external moments, and the internal moments are provided by the muscular system of the subject [27].

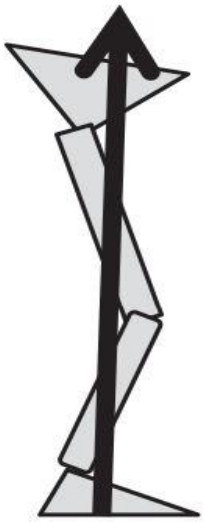


Figure 2.6. Vertical Ground Reaction Force [27]

This thesis studies a kinetic gait analysis system that computes GRF and joint moments of the lower extremity. For this reason, data in the literature were investigated from this perspective. Figure 2.7, from Fukuchi et al., [46] is presented as an example of this data. Fukuchi et al. reported joint moments of twenty-four young adults walking at various speeds. A dashed line represents the speed at which the participants walked comfortably. The slow speed (-30% of comfortable velocity) is represented by a light blue line, and the high speed (+30% of comfortable velocity) is represented by a dark blue line.

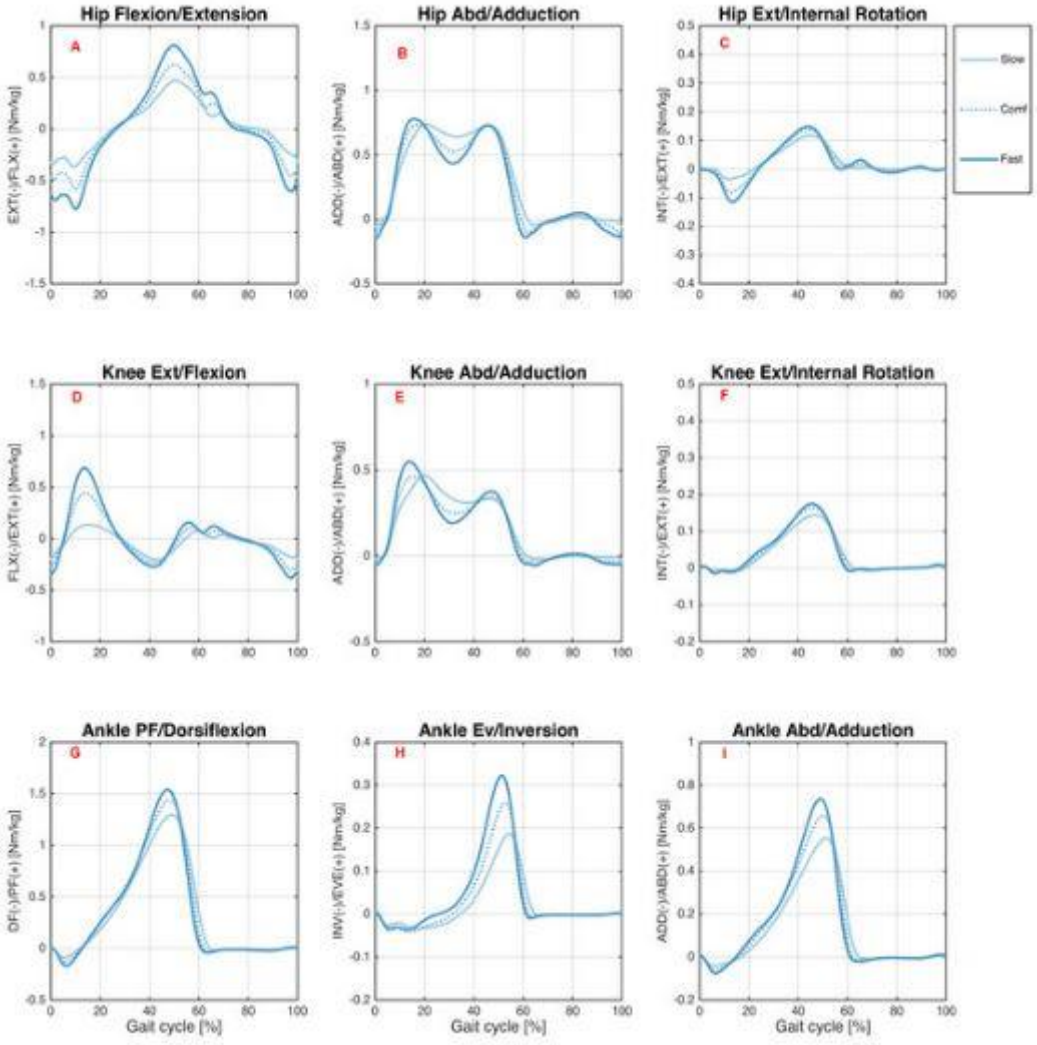


Figure 2.7. Joint angular kinetics of twenty-four healthy young adults [46]

CHAPTER 3

SOFTWARE OF MOTION ANALYSIS SYSTEM

This thesis aims to build a motion analysis system for human upper and lower extremities.

The primary purpose of the lower extremity motion analysis system is kinematic and kinetic gait analysis, whereas for the upper extremity system, it is just kinematic.

The kinetic gait analysis system consists of force plates. By using these sensors, ground reaction forces (GRFs) are measured, coordinate systems of these force plates are transformed to OpenSim coordinate system and analysis is conducted by using OpenSim Inverse Dynamics Setup. As a result, the moments of each joint can be computed using kinematics and GRFs.

The kinematic gait analysis system consists of seven IMU sensors placed in seven segments (pelvis, left and right thighs, shanks and feet) of the lower extremity of a human. By using these sensors, angular velocity, linear acceleration, and magnetic north can be measured. After data is collected, it should be filtered. By using Madgwick Algorithm [31], these filtered sensor data are fused to find the orientation of each sensor. Subsequently, coordinate systems of these sensors should be transformed to OpenSim frame (see Sections 3.1 and 3.5), and combined data from all the sensors are used by OpenSim to estimate angles of human lower limb joints during walking.

On the other hand, the upper extremity motion analysis system includes only kinematic analysis. It utilizes the same procedure as the kinematic gait analysis system. The main difference is that IMU sensors should be attached to segments of the upper extremity of a human (torso, hands and proximal and distal parts of left and right arms). The output of this system is human upper extremity joint angles.

3.1 Coordinate Systems

Commercial LSM9DS1 sensor was used for the kinematic analysis system. The sensors are approximately 60 g and have a size of 7 x 4.5 x 2.5 cm. Accelerometer, gyroscope and magnetometer reference frames are presented in Figure 3.1. Magnetometer and accelerometer reference frames are different, but gyroscope and accelerometer reference frames are the same. Additionally, gyroscope and accelerometer reference frames are left-handed reference frames, but the magnetometer frame is a right-handed coordinate frame.

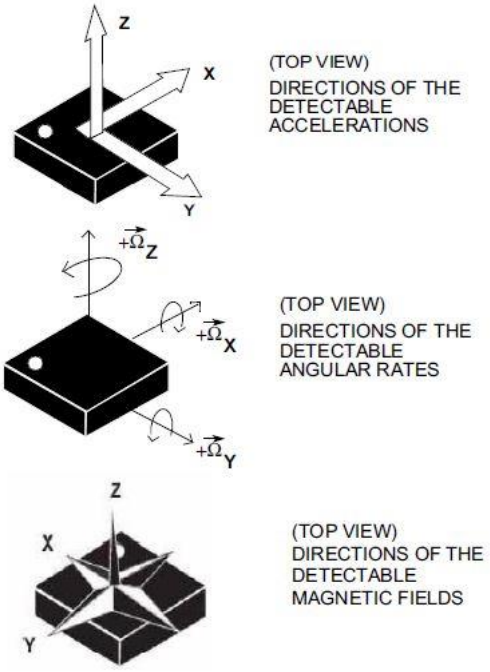


Figure 3.1. Sensor (LSM9DS1) coordinate frames

On the other hand, OpenSim environment, which is used for kinematic analysis for this thesis, uses a coordinate system of X forward (red), Y up (green) and Z right (blue) [33]. This is the standard reference frame system for the OpenSim environment, as shown in Figure 3.2 [33]. Accelerometer frames of the used sensors are added to this figure. The North-East-Down (NED) reference frame is presented in this figure since Madgwick Algorithm uses NED as the earth reference frame.

Moreover, North-West-Up (NWU) reference frame is presented in Figure 3.2. It was found after single sensor experiments (in Chapter 4.2) that Madgwick Algorithm output is in NWU.

Two coordinate frames transformations should be carried out for the kinematic analysis system. To give an example, coordinate frame transformations of the pelvis IMU sensor are presented in Table 3.1.

The first transformation is the transformation of all IMU coordinate frames as NED to use data in the Madgwick Algorithm input. In Table 3.1, the information from the x-axis of the pelvis positioned magnetometer is written as the y-axis. This transformation is needed to give data as Madgwick Algorithm input.

The second transformation is needed to use Madgwick Algorithm output in the OpenSim. In Table 3.1, y axis of the Madgwick Algorithm input is -y axis of the Madgwick Algorithm output. The second transformation is between the Madgwick Algorithm output and OpenSim frame. The OpenSim interface should be used for the second transformation, and it is defined as rotation about the x-axis as -pi/2.

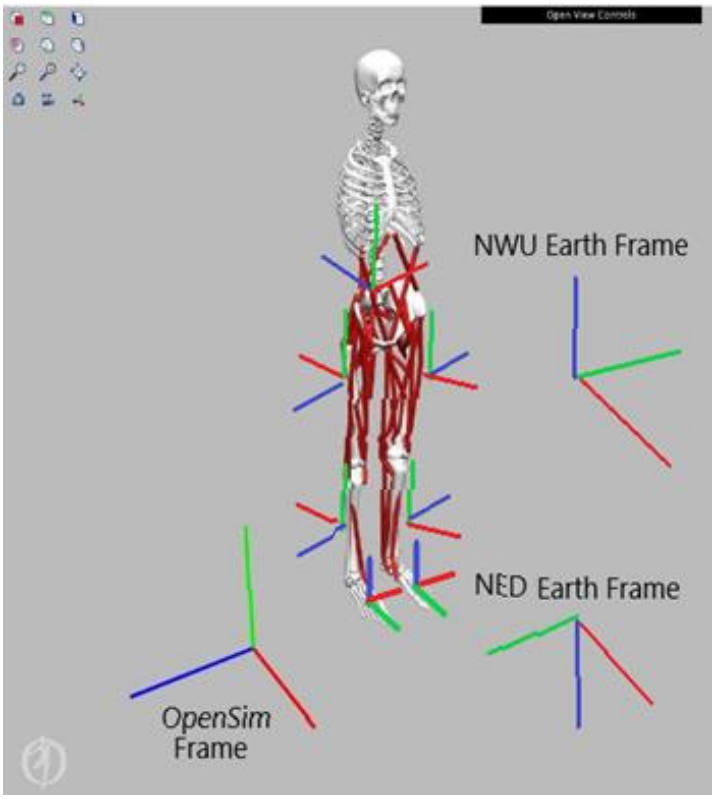


Figure 3.2. Kinematic analysis related coordinate frames (reproduced from [33])

Table 3.1 Kinematic coordinate frame transformation table

NED Earth Frame for Madgwick Algorithm input	Pelvis Gyroscope/Accelerometer Axes	Pelvis Magnetometer Axes	NWU Earth Frame as Madgwick Algorithm output	OpenSim Frame
x	-z	-z	x	x
y	-x	x	-y	-z
z	-y	-y	-z	-y

In addition to the transformations of kinematic analysis system, force plate coordinate systems should be transformed to OpenSim World Frame. Force plate frames are presented in Figure 3.3. According to the walking direction, coordinate system transformations change. Frame transformation table related to direction is presented in Table 3.2.

Table 3.2 Force Plate to OpenSim coordinate frame transformation table

Direction 1			Direction 2		
OpenSim Frame	**Force Plate1 Frame**	**Force Plate2 Frame**	**OpenSim Frame**	**Force Plate1 Frame**	**Force Plate2 Frame**
X	-Y	Y	X	Y	-Y
Y	Z	Z	Y	Z	Z
Z	X	-X	Z	-X	X

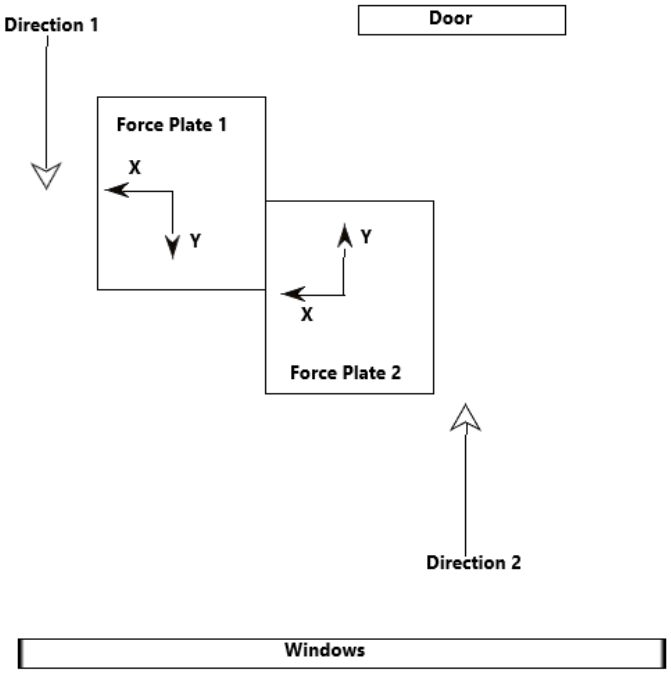


Figure 3.3. Force plate reference frames

3.2 Data from IMU Sensors and Force Plates

The gait analysis system for lower extremity consists of two parts: kinematic and kinetic gait analysis. The kinematic analysis output directly affects the kinetic analysis results, and it is important for the OpenSim input that both data types coincide with the same phases of the walk for kinetic gait analysis. Therefore, it is essential to receive synchronized data from the relevant sensors. The program developed by AdaSoft was used to receive synchronized data simultaneously. The program interface is presented in Figure 3.4.

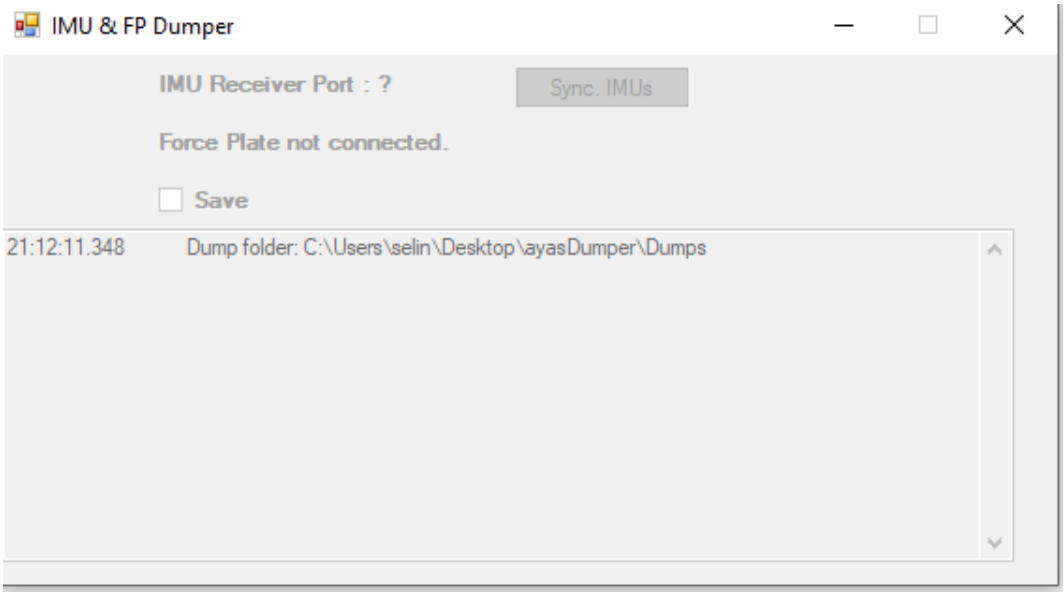


Figure 3.4. IMU and Force Plate Dumper Interface

After connecting the force plate and IMU receivers, the interface lets IMU sensors synchronize. Subsequently, data acquisition can be started by clicking the save button. It is necessary to press the button again to stop recording.

On the other hand, this interface cannot be used for the upper extremity motion analysis system since it does not include only IMU sensors. Therefore, the Tera Term interface can be used to collect data from IMU sensors. The program interface and

settings are presented in Figure 3.5. By pressing button 9 on keyboard, IMU sensors can be synchronized; after then, button 1 can be used to start recording, and 0 is used to stop recording.

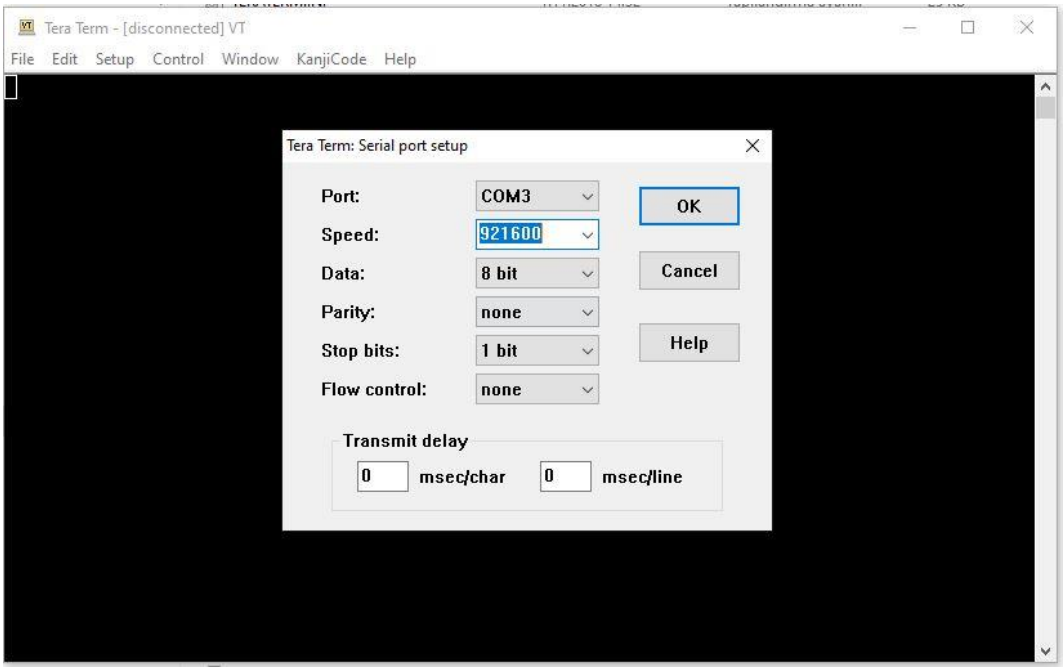


Figure 3.5. Tera Term Interface

3.3 Preprocessing Data

Recorded data from IMU sensors is presented in Figure 3.6. The first row includes the timestamp, sensor number, gyroscope data with related axis (abbreviated as GX, GY, GZ), accelerometer data with related axis (abbreviated as AX, AY, AZ), magnetometer data with related axis (abbreviated as MX, MY, MZ). As seen in Figure 3.6, the sensor data is recorded in a mixed manner. Therefore, the first step is grouping the data using sensor index with the help of a Matlab code. The data count is then equalized so that each sensor's first and last data indices are the same. Finally, the first five thousand data are written as the average of the first one hundred data of gyroscope, accelerometer and magnetometer measurements to find the orientation of

the sensors with respect to the world reference frame. .txt format is used for the files. After the final preprocessing, the data is ready to be used in the Madgwick Algorithm.

The codes written for the pre-processing of the data were originally developed within the scope of this thesis and are presented in Appendix C.

Timestamp	SensorIndex	DataIndex	GX	GY	GZ	AX	AY	AZ	MX	MY	MX
637928783187580000	7	1607	-315	-479	194	-1714	16406	-16	508	-1699	2420
637928783187600000	7	1608	-333	-303	249	-1521	16295	-28	556	-1704	2389
637928783187600000	7	1609	-412	-162	273	-1390	16270	61	556	-1704	2389
637928783187600000	8	1600	-491	-237	58	-5253	-6468	14513	2318	-1857	-3760
637928783187620000	8	1601	-535	-125	98	-5167	-6478	14572	2338	-1832	-3705
637928783187630000	8	1602	-548	-33	118	-5004	-6519	14631	2332	-1828	-3719
637928783187640000	8	1603	-553	-9	120	-4943	-6508	14693	2332	-1828	-3719
637928783187650000	8	1604	-512	-8	92	-4930	-6356	14698	2339	-1841	-3769
637928783187680000	8	1605	-511	-14	76	-4970	-6330	14682	2340	-1834	-3780
637928783187680000	8	1606	-498	-22	68	-4971	-6353	14681	2314	-1832	-3780
637928783187720000	8	1607	-494	-28	38	-4867	-6428	14684	2346	-1805	-3783
637928783187720000	8	1608	-484	-21	11	-4802	-6556	14654	2377	-1824	-3724
637928783187720000	8	1609	-524	-39	-39	-5004	-6499	14569	2377	-1824	-3724
637928783187730000	8	1610	-564	20	-57	-5143	-6470	14510	2340	-1826	-3699
637928783187760000	8	1611	-628	80	-49	-5079	-6308	14632	2329	-1812	-3720
637928783187760000	8	1612	-620	94	-84	-4763	-6343	14719	2282	-1841	-3755
637928783187770000	8	1613	-590	58	-109	-4623	-6456	14748	2328	-1825	-3787
637928783187780000	8	1614	-580	-24	-147	-4800	-6443	14685	2328	-1825	-3787
637928783187800000	8	1615	-580	-24	-147	-4800	-6443	14685	2328	-1825	-3787
637928783187820000	8	1616	-596	-44	-133	-5035	-6329	14683	2297	-1805	-3743
637928783187820000	8	1617	-641	-48	-61	-4936	-6187	14715	2338	-1826	-3713
637928783187850000	8	1618	-627	-62	-47	-4916	-6303	14753	2343	-1814	-3704
637928783187850000	8	1619	-602	-41	-62	-4859	-6548	14698	2345	-1801	-3692
637928783187910000	2	1620	67	202	-30	5130	-5295	14766	1518	-714	-2266
637928783187960000	2	1621	51	261	-55	5169	-5249	14766	1497	-724	-2176
637928783187960000	2	1622	65	304	-42	5208	-5238	14759	1491	-703	-2236
637928783187960000	2	1623	78	344	-44	5323	-5251	14750	1511	-704	-2227
637928783187960000	2	1624	75	350	-76	5330	-5277	14728	1511	-704	-2227

Figure 3.6. IMU recorded data file

3.4 Madgwick Algorithm

According to Madgwick [31], IMU data have a high level of noise; therefore, data from three different types of sensors (accelerometer, magnetometer, gyroscope) should not be used separately [31]. To determine the orientation of the body in the World frame accurately, the data from all sensors should be combined. The combination can be achieved by a fusion algorithm. Several different fusion algorithms can be applied to IMU sensor data, and Madgwick Algorithm is one of them. Sebastian O.H. Madgwick proposed this algorithm in 2010, and he proposed

this method to be worked with the IMU sensors [31]. Normally, Kalman Filter is the most popular method used in biomechanics, but due to the complexity of this method, the Madgwick Algorithm is used. Also, Madgwick Algorithm has better accuracy levels when compared to the Kalman-based algorithm [31].

To understand Madgwick Algortihm, coordinate frames, notation and algorithm steps are explained in this section, respectively.

Firstly, the most common coordinate frames used in this type of algorithm are sensor frame and earth frame. Sensor frame is the frame that moves with the sensor. Typically sensor is attached to the related body, so it is also known as the body frame. The other frame is the earth frame, which is fixed on Earth and does not move.

The second significant part of understanding the algorithm is related to notation. Pre-subscript defines the source coordinate frame, and pre-superscript defines the destination coordinate frame [32]. As an example ${}_{S}^{E}q$ describes the orientation of sensor frame (S) relative to earth frame (E). Also, it is the orientation of the body or sensor in the form of a quaternion [32]. In the case of only pre-superscript being defined, the parameter was measured and represented in the same frame [32].

Madgwick Algorithm has two crucial steps. The first one depends on gyroscope measurements, and the second one is related to accelerometer and magnetometer measurements.

The major equations of the Madgwick Filter related to gyroscope measurements are presented as [32].

$$ {}^{S}w = [0 \quad w_x \quad w_y \quad w_z] \tag{3.1} $$

$$ {}_{E}^{S}\dot{q}_{w,t} = \frac{1}{2} {}_{E}^{S}\hat{q}_{est,t-1} \otimes {}^{S}w_t \tag{3.2} $$

$$ {}_{E}^{S}q_{w,t} = {}_{E}^{S}\hat{q}_{est,t-1} + {}_{E}^{S}\dot{q}_{w,t}\Delta t \tag{3.3} $$

$$ \mu_t = \alpha \left\| {}_{E}^{S}\dot{q}_{w,t} \right\| \Delta t \quad \alpha > 1 \tag{3.4} $$

The general vector for angular velocity measurements can be seen in Equation (3.2). In this equation, angular velocity vector (includes gyroscope measurements on sensor frame at time t) is denoted as ${}^{S}w_t$. ${}^{S}_{E}\dot{q}_{w,t}$, the quaternion derivative, describes the rate of orientation change at time t. ${}^{S}_{E}\hat{q}_{est,t-1}$ is the estimated orientation at the previous time point, Δt is the sampling period, μ_t is the step size at time t and α is the augmentation of μ because of the noise of magnetometer and accelerometer sensors. ${}^{S}_{E}q_{w,t}$ defines the orientation of the earth frame relative to the sensor frame at time t by using previous orientation estimation and gyroscope measurements.

The second part includes equations related to accelerometer and magnetometer measurements. These measurements are substituted into Gradient Descent Algorithm. The final accelerometer (from Equation(3.5) to Equation (3.7)) and magnetometer equations (from Equation(3.8) to Equation (3.11)) were solved separately and combined with each other by using Equations (3.12) and (3.13).

$$ {}^{S}\hat{a} = [0 \quad a_x \quad a_y \quad a_z] \tag{3.5} $$

$$ f_g({}^{S}_{E}\hat{q}, {}^{S}\hat{a}) = \begin{bmatrix} 2(q_2q_4 - q_1q_3) - a_x \\ 2(q_1q_2 - q_3q_4) - a_y \\ 2\left(\frac{1}{2} - q_2^2 - q_3^2\right) - a_z \end{bmatrix} \tag{3.6} $$

$$ J_g({}^{S}_{E}\hat{q}) = \begin{bmatrix} -2q_3 & 2q_4 & -2q_1 & 2q_2 \\ 2q_2 & 2q_1 & 2q_4 & 2q_3 \\ 0 & -4q_2 & -4q_3 & 0 \end{bmatrix} \tag{3.7} $$

$$ {}^{E}\hat{b} = [0 \quad b_x \quad 0 \quad b_z] \tag{3.8} $$

$$ {}^{S}\hat{m} = [0 \quad m_x \quad m_y \quad m_z] \tag{3.9} $$

$$ f_b({}^{S}_{E}\hat{q}, {}^{E}\hat{b}, {}^{S}\hat{m}) = \begin{bmatrix} 2b_x(0.5 - q_3^2 - q_4^2) + 2b_z(q_2q_4 - q_1q_3) - m_x \\ 2b_x(q_2q_3 - q_1q_4) + 2b_z(q_1q_2 - q_3q_4) - m_y \\ 2b_x(q_1q_3 - q_2q_4) + 2b_z(0.5 - q_2^2 - q_3^2) - m_z \end{bmatrix} \tag{3.10} $$

$$J_b\left({}^S_E\hat{q}\,,{}^E\hat{b}\right) =$$

$$\begin{bmatrix} -2b_z q_3 & 2b_z q_4 & -4b_x q_3 - 2b_z q_1 & -4b_x q_4 + 2b_z q_2 \\ -2b_x q_4 + 2b_z q_2 & 2b_x q_3 + 2b_z q_1 & 2b_x q_2 + 2b_z q_4 & -2b_x q_1 + 2b_z q_3 \\ 2b_x q_3 & 2b_x q_4 - 4b_z q_2 & 2b_x q_1 - 4b_z q_3 & 2b_x q_2 \end{bmatrix} \quad (3.11)$$

$$f_{g,b}\left({}^S_E\hat{q}\,,{}^S\hat{a}\,,{}^E\hat{b}\,,{}^S\hat{m}\right) = \begin{bmatrix} f_g({}^S_E\hat{q}\,,{}^S\hat{a}\,) \\ f_b\left({}^S_E\hat{q}\,,{}^E\hat{b}\,,{}^S\hat{m}\right) \end{bmatrix} \quad (3.12)$$

$$J_{g,b}\left({}^S_E\hat{q}\,,{}^E\hat{b}\right) = \begin{bmatrix} J_g^T({}^S_E\hat{q}\,) \\ J_b^T\left({}^S_E\hat{q}\,,{}^E\hat{b}\right) \end{bmatrix} \quad (3.13)$$

Angular velocity (${}^S w_t$) from the gyroscope, linear acceleration (${}^S\hat{a}_t$) from the accelerometer, magnetic direction and strength (${}^S\hat{m}_t$) from magnetometer at time t and Earth's magnetic field in earth frame (${}^E\hat{b}$) are defined in these equations. Superscript s indicates that these vectors are described in the sensor frame. Vector representation of the sensor data can be seen in Equations (3.1), (3.5) and (3.9).

${}^S_E q_{\nabla,t}$ defines the orientation of the earth frame relative to the sensor frame at time t by using previous orientation estimation and magnetometer and accelerometer measurements. ∇ means that the related quaternion is calculated by using the gradient descent algorithm.

$${}^S_E q_{\nabla,t} = {}^S_E\hat{q}_{est,t-1} + \mu_t \frac{\nabla f}{\|\nabla f\|} \quad (3.14)$$

$$\nabla f = \begin{cases} J_g^T\left({}^S_E\hat{q}_{est,t-1}\right) f_g\left({}^S_E\hat{q}_{est,t-1}, {}^S\hat{a}_{\,t}\right) \\ J_{g,b}^T\left({}^S_E\hat{q}_{est,t-1}, {}^E\hat{b}\,\right) f_{g,b}\left({}^S_E\hat{q}_{est,t-1}, {}^S\hat{a}\,,{}^E\hat{b}\,,{}^S\hat{m}\,\right) \end{cases} \quad (3.15)$$

As mentioned earlier, Eqs. (3.3) and (3.14) are the main equations to estimate orientation at time t. Equation (3.3) depends on the angular rate and Eq. (3.14) depends on accelerometer and magnetometer measurements. After fusing these equations, Madgwick calculated "filter fusion algorithm equations" that are presented Eqs. (3.16), (3.17) and (3.18). β is the magnitude of gyroscope measurement error. It can be calculated by multiplying mean zero gyroscope measurement error by $\sqrt{3/4}$.

$$ {}^{S}_{E}q_{est,t} = {}^{S}_{E}\hat{q}_{est,t-1} + {}^{S}_{E}\dot{q}_{est,t}\Delta t \tag{3.16} $$

$$ {}^{S}_{E}\dot{q}_{est,t} = {}^{S}_{E}\dot{q}_{w,t} - \beta\, {}^{S}_{E}\hat{\dot{q}}_{\epsilon,t} \tag{3.17} $$

$$ {}^{S}_{E}\hat{\dot{q}}_{\epsilon,t} = \frac{\nabla f}{\|\nabla f\|} \tag{3.18} $$

The block diagram of Madgwick Algorithm is presented in Figure 3.7. The equations presented do not include magnetic distortion (Group 1) and gyroscope drift compensation (Group 2) in Figure 3.7. These corrections can be used to get more accurate estimations about IMUs' orientation.

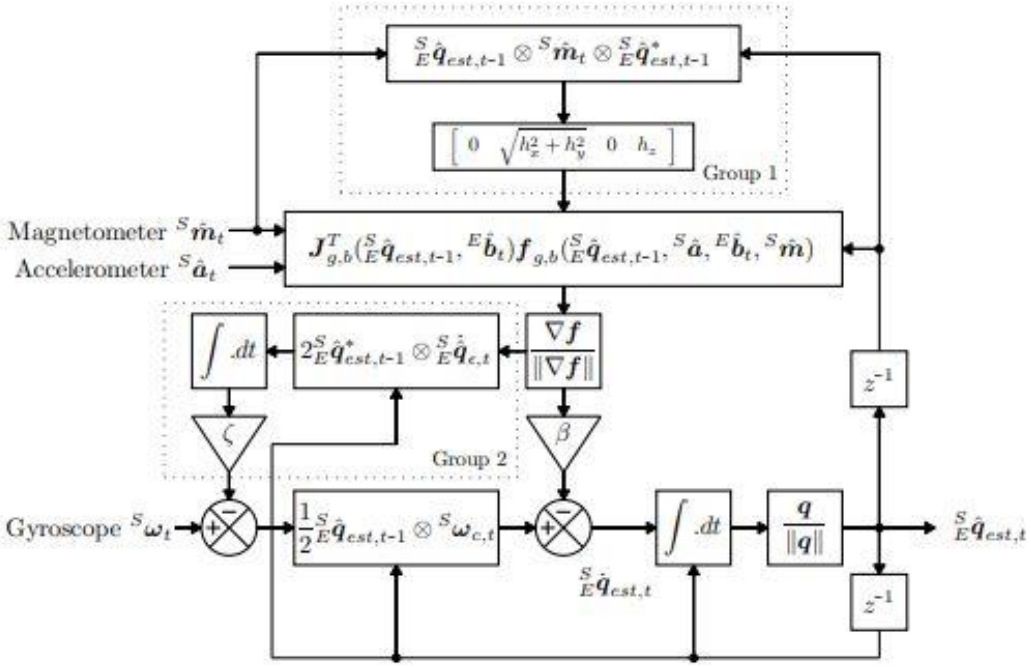


Figure 3.7. Complete block diagram of Madgwick Algorithm [31]

Group 1, in Figure 3.7 represents the magnetic distortion compensation of the magnetometer. Moreover, the need for the reference direction of the Earth can be eliminated by using Group 1 in the algorithm. This compensation can be achieved with the calculation of the reference direction of Earth's magnetic field (${}^{E}\hat{b}_t$) by using the measured direction of Earth's magnetic field with the help of magnetometer (${}^{E}\hat{h}_t$). Eqs. (3.19) and (3.20) can be used for this compensation [31].

$$^{E}\hat{h}_t = [0 \quad h_x \quad h_y \quad h_z] = {}^{S}_{E}\hat{q}_{est,t-1} \otimes {}^{s}\hat{m}_t \otimes {}^{S}_{E}\hat{q}^{*}_{est,t-1} \tag{3.19}$$

$$^{E}\hat{b}_t = \begin{bmatrix} 0 & \sqrt{h_x^2 + h_y^2} & 0 & h_z \end{bmatrix} \tag{3.20}$$

3.5 OpenSim Program

OpenSim is an open source software package that can be used to analyze and simulate movement with the help of models of musculoskeletal structures. It has different modules for kinematic and kinetic analyses. OpenSim 4.3 version was used for this study. After OpenSim 4.1 version OpenSense software was released which was able to use IMU data for kinematic analysis.

The main reasons for using OpenSim are that it provides a fast tool for kinetic and kinematic analysis with the help of manipulable models and IMU plugins and allows animating the motion.

The main window of OpenSim is presented in Fig. 3.8. By using the main window, models and motions can be visualized, and by using OpenSim Tools, analysis can be performed.

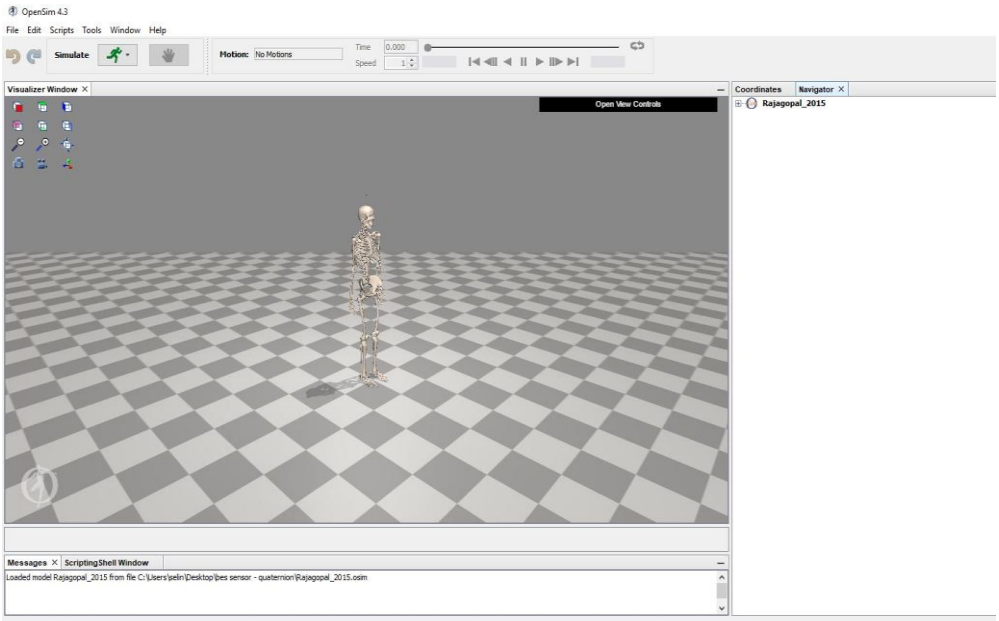


Figure 3.8. OpenSim Main Window

Analysis can be performed after opening a model. There are different musculoskeletal models developed for OpenSim. All released models are shared on OpenSim official website [48]. With the help of the Matlab Scripting Environment, conducting the analysis with Matlab is possible. In the further sections, OpenSim Tools, which are IMU Placer Tool, IMU Inverse Kinematics Tool and Inverse Dynamics Tool, are explained.

3.5.1 IMU Placer Tool

Firstly, an OpenSim model must be loaded in the OpenSim program. The Rajagopal model was used for this study. After the model is loaded, the IMU sensors are positioned on the model; thus, the model is calibrated.

Sensors positioned on the subject should be in approximately the same positions as the sensors positioned on the model, and the sensor axes should be as parallel as possible to the human anatomical planes.

The interface and adjustments of the tool are presented in Figure 3.9.

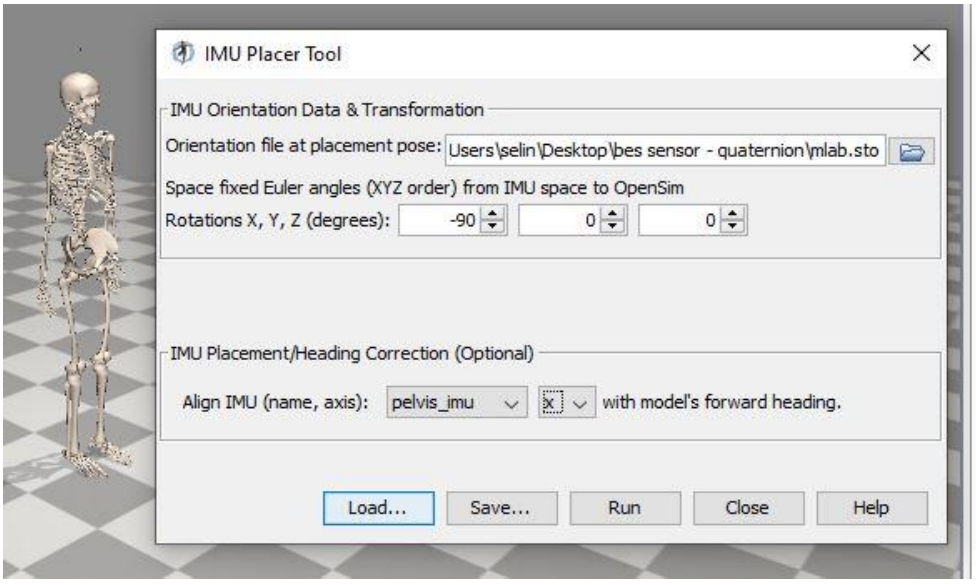


Figure 3.9. OpenSim IMU Placer Tool

This procedure can also be implemented through Matlab. Matlab code is presented in Figure 3.10.

```
%% Set variables to use
modelFileName = 'Rajagopal_2015.osim';
orientationsFileName = 'mlab.sto';
sensor_to_opensim_rotations = Vec3(-pi/2, 0, 0);
baseIMUName = 'pelvis_imu';
baseIMUHeading = 'x';
visulizeCalibration = true;
```

Figure 3.10. Matlab code for OpenSim IMU Placer

3.5.2 IMU Inverse Kinematics Tool

Opensim Inverse Kinematics (IK) Tool is the interface where kinematic analysis is performed. This interface is presented in Figure 3.11.

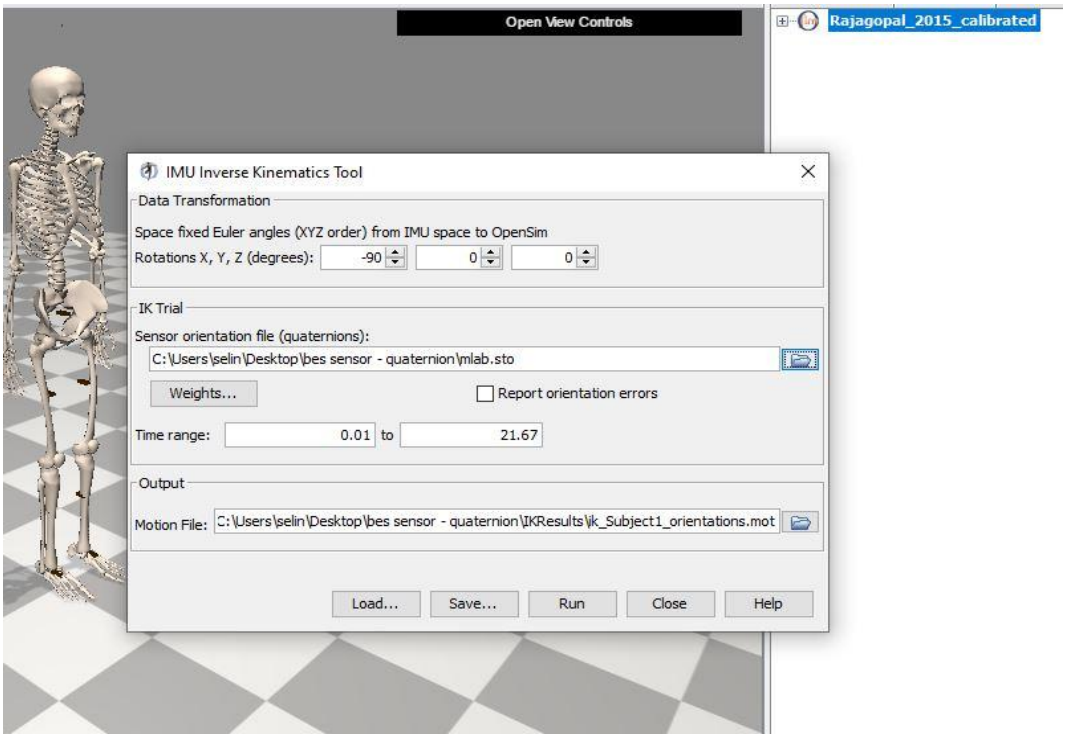
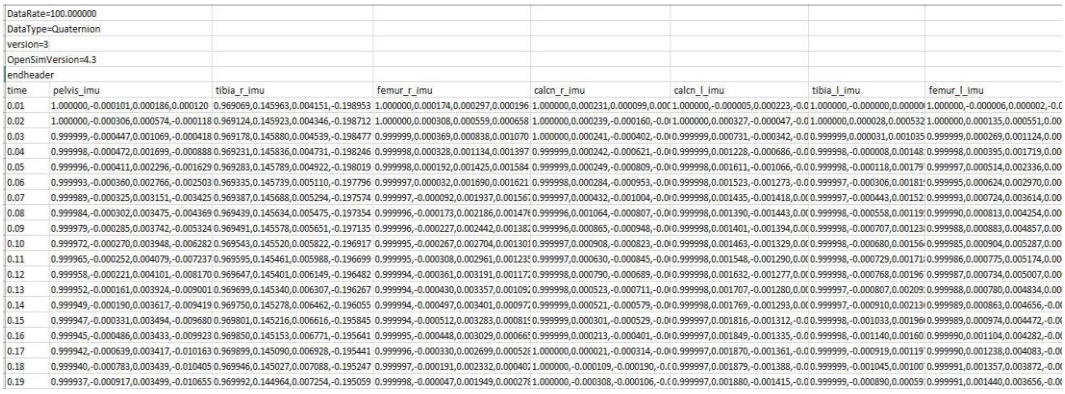


Figure 3.11. OpenSim IK Tool

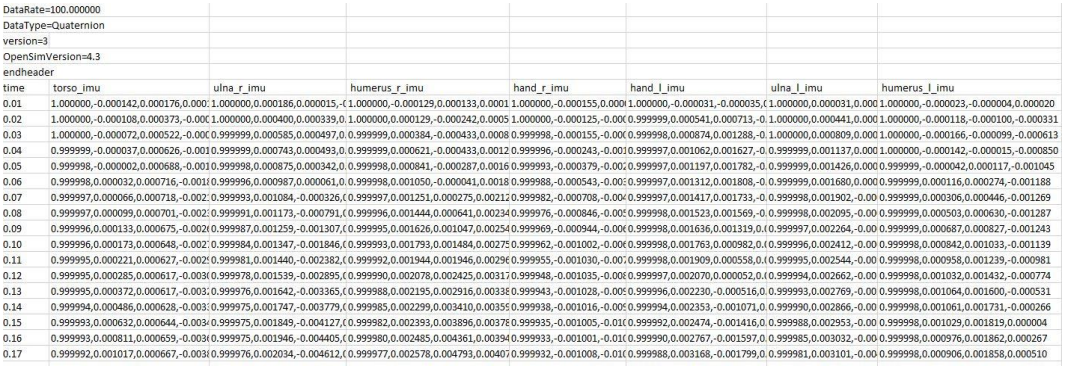
As input to this tool, the sensor data should be provided together with the information of which sensor it belongs to. The input data, "mlab.sto", is presented in Figure 3.12 for the lower extremity and Figure 3.13 for the upper extremity. The first five rows of this document contain the information that must be supplied to OpenSim, and the sixth row includes time and the sensor names defined in OpenSim. The following rows contain sensor orientation data in the form of quaternion. The output of this tool is the subject's joint angles.



DataRate=100.000000
DataType=Quaternion
version=3
OpenSimVersion=4.3
endheader

time	pelvis_imu	tibia_r_imu	femur_r_imu	calcn_r_imu	calcn_l_imu	tibia_l_imu	femur_l_imu
0.01	1.000000,-0.000101,0.000186,0.000120	0.969069,0.145963,0.004151,-0.198953	1.000000,0.000174,0.000297,0.000196	1.000000,0.000231,0.000099,0.000	1.000000,-0.000005,0.000223,-0.0	1.000000,-0.000000,0.000000	1.000000,-0.000006,0.000002,-0.0
0.02	1.000000,-0.000306,0.000574,-0.000118	0.969124,0.145923,0.004346,-0.198712	1.000000,0.000308,0.000559,0.000658	1.000000,0.000239,-0.000160,-0.0	1.000000,0.000327,-0.000047,-0.0	1.000000,0.000028,0.000532	1.000000,0.000135,0.000551,0.00
0.03	0.999999,-0.000447,0.001069,-0.000418	0.969178,0.145880,0.004539,-0.198477	0.999999,0.000369,0.000838,0.001070	1.000000,0.000241,-0.000402,-0.0	0.999999,0.000731,-0.000342,-0.0	0.999999,0.000031,0.001035	0.999999,0.000269,0.001124,0.00
0.04	0.999998,-0.000472,0.001699,-0.000888	0.969231,0.145836,0.004731,-0.198246	0.999998,0.000328,0.001134,0.001397	0.999999,0.000242,-0.000621,-0.0	0.999999,0.001228,-0.000686,-0.0	0.999998,-0.000008,0.00148	0.999998,0.000395,0.001719,0.00
0.05	0.999996,-0.000411,0.002296,-0.001629	0.969283,0.145789,0.004922,-0.198019	0.999998,0.000192,0.001425,0.001584	0.999999,0.000249,-0.000809,-0.0	0.999998,0.001611,-0.001066,-0.0	0.999998,-0.000118,0.00179	0.999997,0.000514,0.002336,0.00
0.06	0.999993,-0.000360,0.002766,-0.002503	0.969335,0.145739,0.005110,-0.197796	0.999997,0.000032,0.001690,0.001621	0.999998,0.000284,-0.000953,-0.0	0.999998,0.001523,-0.001273,-0.0	0.999997,-0.000306,0.00181	0.999995,0.000624,0.002970,0.00
0.07	0.999989,-0.000325,0.003151,-0.003425	0.969387,0.145688,0.005294,-0.197574	0.999997,-0.000092,0.001937,0.001561	0.999997,0.000432,-0.001004,-0.0	0.999998,0.001435,-0.001418,0.00	0.999997,-0.000443,0.00152	0.999993,0.000724,0.003614,0.00
0.08	0.999984,-0.000302,0.003475,-0.004369	0.969439,0.145634,0.005475,-0.197354	0.999996,-0.000173,0.002186,0.001476	0.999996,0.001064,-0.000807,-0.0	0.999998,0.001390,-0.001443,0.00	0.999998,-0.000558,0.00119	0.999990,0.000813,0.004254,0.00
0.09	0.999979,-0.000285,0.003742,-0.005324	0.969491,0.145578,0.005651,-0.197135	0.999996,-0.000227,0.002442,0.001386	0.999996,0.000865,-0.000948,-0.0	0.999998,0.001401,-0.001394,0.00	0.999998,-0.000707,0.00123	0.999988,0.000883,0.004857,0.00
0.10	0.999972,-0.000270,0.003948,-0.006282	0.969543,0.145520,0.005822,-0.196917	0.999995,-0.000267,0.002704,0.001301	0.999997,0.000908,-0.000823,-0.0	0.999998,0.001463,-0.001329,0.00	0.999998,-0.000680,0.00156	0.999985,0.000904,0.005287,0.00
0.11	0.999965,-0.000252,0.004079,-0.007237	0.969595,0.145461,0.005988,-0.196699	0.999995,-0.000308,0.002961,0.001233	0.999997,0.000630,-0.000845,-0.0	0.999998,0.001548,-0.001290,0.00	0.999998,-0.000729,0.00171	0.999986,0.000775,0.005174,0.00
0.12	0.999958,-0.000221,0.004101,-0.008170	0.969647,0.145401,0.006149,-0.196482	0.999994,-0.000361,0.003191,0.001177	0.999998,0.000790,-0.000689,-0.0	0.999998,0.001632,-0.001277,0.00	0.999998,-0.000768,0.00196	0.999987,0.000734,0.005007,0.00
0.13	0.999952,-0.000161,0.003924,-0.009001	0.969689,0.145340,0.006307,-0.196267	0.999994,-0.000430,0.003357,0.001095	0.999998,0.000523,-0.000711,-0.0	0.999998,0.001707,-0.001280,0.00	0.999997,-0.000807,0.00205	0.999988,0.000780,0.004834,0.00
0.14	0.999949,-0.000190,0.003617,-0.009419	0.969750,0.145278,0.006462,-0.196055	0.999994,-0.000497,0.003401,0.000971	0.999999,0.000521,-0.000579,-0.0	0.999998,0.001769,-0.001293,0.00	0.999997,-0.000910,0.00213	0.999989,0.000863,0.004656,-0.00
0.15	0.999947,-0.000331,0.003494,-0.009680	0.969801,0.145216,0.006616,-0.195845	0.999994,-0.000512,0.003283,0.000815	0.999999,0.000301,-0.000529,-0.0	0.999997,0.001816,-0.001312,-0.0	0.999998,-0.001033,0.00196	0.999989,0.000974,0.004472,-0.00
0.16	0.999945,-0.000486,0.003433,-0.009923	0.969850,0.145153,0.006771,-0.195641	0.999995,-0.000448,0.003029,0.000666	0.999999,0.000213,-0.000401,-0.0	0.999997,0.001849,-0.001335,-0.0	0.999998,-0.001140,0.00160	0.999990,0.001104,0.004282,-0.00
0.17	0.999942,-0.000639,0.003417,-0.010163	0.969899,0.145090,0.006928,-0.195441	0.999996,-0.000330,0.002699,0.000526	1.000000,0.000021,-0.000314,-0.0	0.999997,0.001870,-0.001361,-0.0	0.999999,-0.000919,0.00119	0.999990,0.001238,0.004083,-0.00
0.18	0.999940,-0.000783,0.003439,-0.010405	0.969946,0.145027,0.007088,-0.195247	0.999997,-0.000191,0.002332,0.000401	1.000000,-0.000109,-0.000190,-0.0	0.999997,0.001879,-0.001388,-0.0	0.999999,-0.001045,0.00100	0.999991,0.001357,0.003872,-0.00
0.19	0.999937,-0.000917,0.003499,-0.010655	0.969992,0.144964,0.007254,-0.195059	0.999998,-0.000047,0.001949,0.000276	1.000000,-0.000308,-0.000106,-0.0	0.999997,0.001880,-0.001415,-0.0	0.999999,-0.000890,0.00059	0.999991,0.001440,0.003656,-0.00

Figure 3.12. Lower extremity input data for OpenSim IK Tool



DataRate=100.000000
DataType=Quaternion
version=3
OpenSimVersion=4.3
endheader

time	torso_imu	ulna_r_imu	humerus_r_imu	hand_r_imu	hand_l_imu	ulna_l_imu	humerus_l_imu
0.01	1.000000,-0.000142,0.000176,0.000	1.000000,0.000186,0.000015,-(	1.000000,-0.000129,0.000133,0.0001	1.000000,-0.000155,0.000(	1.000000,-0.000031,-0.000035,(	1.000000,0.000031,0.000	1.000000,-0.000023,-0.000004,0.000020
0.02	1.000000,-0.000108,0.000373,-0.000	1.000000,0.000400,0.000339,0.	1.000000,0.000129,-0.000242,0.0005	1.000000,-0.000125,-0.000(	0.999999,0.000541,0.000713,-0.	1.000000,0.000441,0.000	1.000000,-0.000118,-0.000100,-0.000331
0.03	1.000000,-0.000072,0.000522,-0.000	0.999999,0.000585,0.000497,0.	0.999999,0.000384,-0.000433,0.0008	0.999998,-0.000155,-0.000(	0.999998,0.000874,0.001288,-0.	1.000000,0.000809,0.000	1.000000,-0.000166,-0.000099,-0.000613
0.04	0.999999,-0.000037,0.000626,-0.001	0.999999,0.000743,0.000493,0.	0.999999,0.000621,-0.000433,0.0012	0.999996,-0.000243,-0.000(	0.999997,0.001062,0.001627,-0.	0.999999,0.001137,0.000	1.000000,-0.000142,-0.000015,-0.000850
0.05	0.999998,-0.000002,0.000688,-0.001	0.999998,0.000875,0.000342,0.	0.999998,0.000841,-0.000287,0.0016	0.999993,-0.000379,-0.000(	0.999997,0.001197,0.001782,-0.	0.999999,0.001426,0.000	0.999999,-0.000042,0.000117,-0.001045
0.06	0.999998,0.000032,0.000716,-0.001	0.999996,0.000987,0.000061,0.	0.999998,0.001050,-0.000041,0.0018	0.999988,-0.000543,-0.000(	0.999997,0.001312,0.001808,-0.	0.999999,0.001680,0.000	0.999999,0.000116,0.000274,-0.001188
0.07	0.999997,0.000066,0.000718,-0.002	0.999993,0.001084,-0.000326,(	0.999997,0.001251,0.000275,0.0021	0.999982,-0.000708,-0.000(	0.999997,0.001417,0.001733,-0.	0.999998,0.001902,-0.00	0.999999,0.000306,0.000446,-0.001269
0.08	0.999997,0.000099,0.000701,-0.002	0.999991,0.001173,-0.000791,(	0.999996,0.001444,0.000641,0.0023	0.999976,-0.000846,-0.000(	0.999998,0.001523,0.001569,-0.	0.999998,0.002095,-0.00	0.999999,0.000503,0.000630,-0.001287
0.09	0.999996,0.000133,0.000675,-0.002	0.999987,0.001259,-0.001307,(	0.999995,0.001626,0.001047,0.0025	0.999969,-0.000944,-0.000(	0.999998,0.001636,0.001319,0.(	0.999997,0.002264,-0.00	0.999999,0.000687,0.000827,-0.001243
0.10	0.999996,0.000173,0.000648,-0.002	0.999984,0.001347,-0.001846,(	0.999993,0.001793,0.001484,0.0027	0.999962,-0.001002,-0.000(	0.999998,0.001763,0.000982,0.(	0.999996,0.002412,-0.00	0.999998,0.000842,0.001033,-0.001139
0.11	0.999995,0.000221,0.000627,-0.002	0.999981,0.001440,-0.002382,(	0.999992,0.001944,0.001946,0.0029	0.999955,-0.001030,-0.000(	0.999998,0.001909,0.000558,0.(	0.999995,0.002544,-0.00	0.999998,0.000958,0.001239,-0.000981
0.12	0.999995,0.000285,0.000617,-0.003	0.999978,0.001539,-0.002895,(	0.999990,0.002078,0.002425,0.0031	0.999948,-0.001035,-0.000(	0.999997,0.002070,0.000052,0.(	0.999994,0.002662,-0.00	0.999998,0.001032,0.001432,-0.000774
0.13	0.999995,0.000372,0.000617,-0.003	0.999976,0.001642,-0.003365,(	0.999988,0.002195,0.002916,0.0033	0.999943,-0.001028,-0.000(	0.999996,0.002230,-0.000516,0.	0.999993,0.002769,-0.00	0.999998,0.001064,0.001600,-0.000531
0.14	0.999994,0.000486,0.000628,-0.003	0.999975,0.001747,-0.003779,(	0.999985,0.002299,0.003410,0.0035	0.999938,-0.001016,-0.000(	0.999994,0.002353,-0.001071,0.	0.999990,0.002866,-0.00	0.999998,0.001061,0.001731,-0.000266
0.15	0.999993,0.000632,0.000644,-0.003	0.999975,0.001849,-0.004127,(	0.999982,0.002393,0.003896,0.0037	0.999935,-0.001005,-0.001(	0.999992,0.002474,-0.001416,0.	0.999988,0.002953,-0.00	0.999998,0.001029,0.001819,0.000004
0.16	0.999993,0.000811,0.000659,-0.003	0.999975,0.001946,-0.004405,(	0.999980,0.002485,0.004361,0.0039	0.999933,-0.001001,-0.001(	0.999990,0.002767,-0.001597,0.	0.999985,0.003032,-0.00	0.999998,0.000976,0.001862,0.000267
0.17	0.999992,0.001017,0.000667,-0.003	0.999976,0.002034,-0.004612,(	0.999977,0.002578,0.004793,0.0040	0.999932,-0.001008,-0.001(	0.999988,0.003168,-0.001799,0.	0.999981,0.003101,-0.00	0.999998,0.000906,0.001858,0.000510

Figure 3.13. Upper extremity input data for OpenSim IK Tool

This procedure can also be implemented through Matlab. Matlab codes are presented in Figure 3.14.

```
%% Instantiate an InverseKinematicsTool
imuIK = IMUInverseKinematicsTool();
```

Figure 3.14. Matlab code for OpenSim IK Tool

3.5.3 Inverse Dynamics Setup

Opensim Inverse Dynamics (ID) Tool is the interface where kinetic analysis is performed. This interface is presented in Figure 3.15.

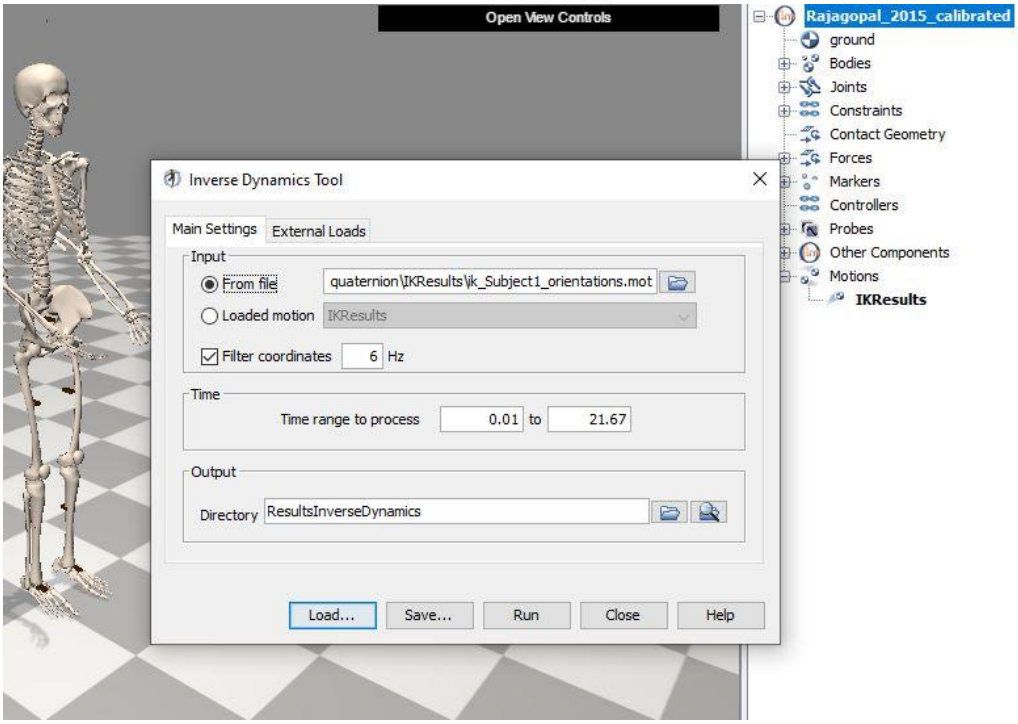


Figure 3.15. OpenSim ID Tool

Joint angles previously computed by the IK tool and Ground Reaction Forces (GRFs) measured by force platforms are input to this tool. Output is the joint moments of the subject.

3.5.4 Animation of the Motion

Kinematic and kinetic data performed by the subject can be animated by using the OpenSim interface or Matlab. As presented in Figure 3.16, GRF can be visualized by associating the GRF data with OpenSim ID Tool.

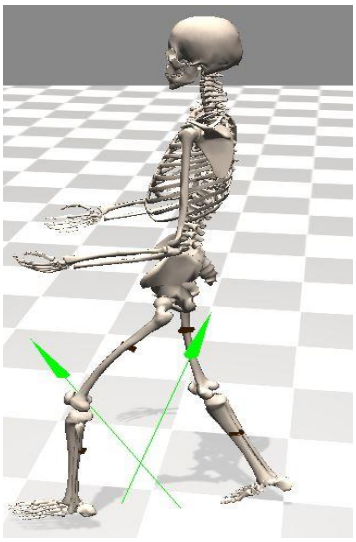


Figure 3.16. Visualizing Ground Reaction Forces

3.6 Interpretation of Code

3.6.1 Code for Kinematic Analysis System

The block diagram of the kinematic analysis system is presented in Figure 3.17. This system consists of two parts; on Matlab and OpenSim. The Matlab Part starts with data acquisition from the IMU sensors and ends with the generation of "mlab.sto" file. All the code used in this part work with the Matlab program (.m file), and the main code is TestCpp. This code ensures that the data from IMU sensors is read; directed to the relevant Codegens, and the resulting data, which is "mlab.sto", is

generated. This file includes information about sensor orientations as quaternions with related segment names.

Matrices code separates the sensor data according to sensor indexes and directs them to the corresponding Calculation code. Calculation1 is for the pelvis sensor, Calculation2 is for right distal and proximal leg sensors, Calculation 3 is for the left distal and proximal leg sensors, and Calculation4 is for the foot sensors. Calculation code align the sensor axes to the NED reference frame. Then data is filtered. After the filtering process, the orientation of the sensor frame relative to the World Reference Frame is computed as a quaternion with the help of the AHRSAlgorithm and the first five thousand data point of motion data. AHRSAlgorithm contains Madgwick Algorithm. By using the measured quaternion as an input to the AHRSalgorithm code, the rest of the motion data is transferred through the AHRSalgorithm code again, and the "mlab.sto" file is generated. This file contains orientation of each sensor with related bodies defined in OpenSim. The file is then used in OpenSim program for IMU Placer, and IMU IK tool, respectively. As a result, "ik_Subject1_orientations.mot" file is generated that contains the subject's joint angles.

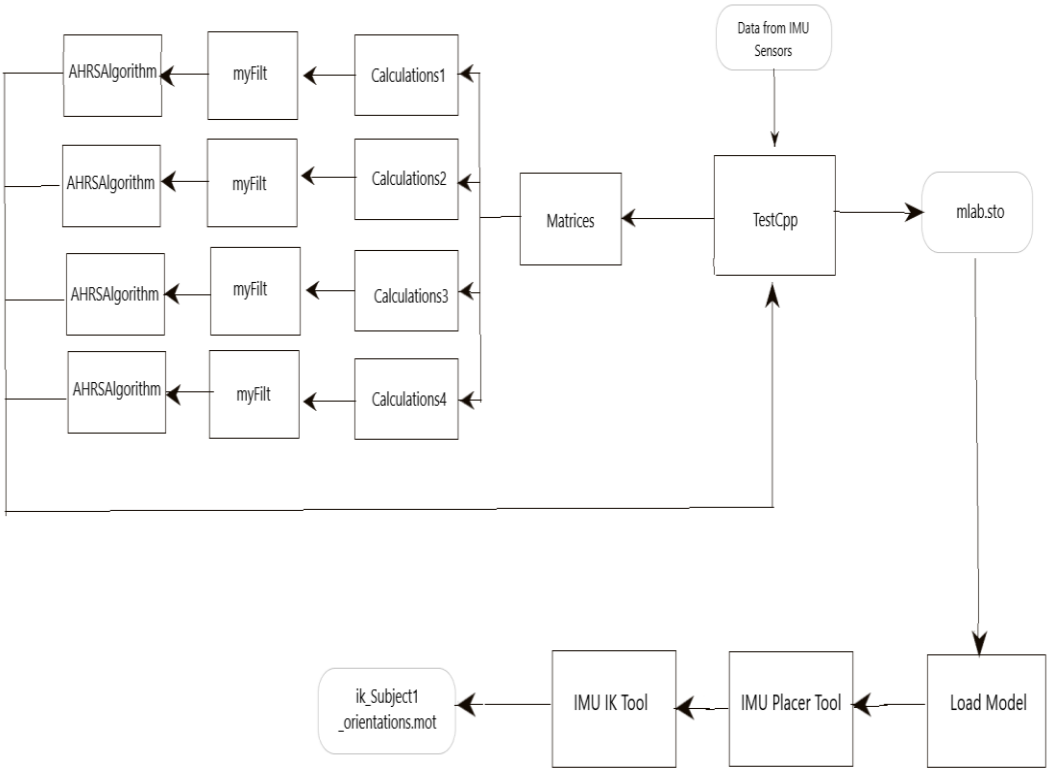


Figure 3.17. Block diagram of the kinematic analysis system

The same procedure is applied for both upper and lower extremity kinematic analysis systems. The main difference is in Calculation codes for the upper extremity motion analysis system, Calculation1 is for the torso sensor, Calculation2 is for right proximal and distal arm sensors, Calculation 3 is for left proximal and distal arm sensors, and Calculation4 is for the hand sensors.

The developed codes are TestCpp, Matrices and Calculations1, Calculations2, Calculations3, and Calculations4. The codes that make calculations based on quaternions (for lower extremity) are presented in Appendix D, the codes that make calculations based on Euler angles (for lower extremity) are presented in Appendix E, and the codes developed for the upper extremity kinematic analysis are presented in Appendix F.

3.6.2 Code for Kinetic Analysis System

The block diagram of the kinetic analysis system is presented in Figure 3.18. The developed codes are grfmot.m codes for two different walking directions. These codes are presented in Appendix G.

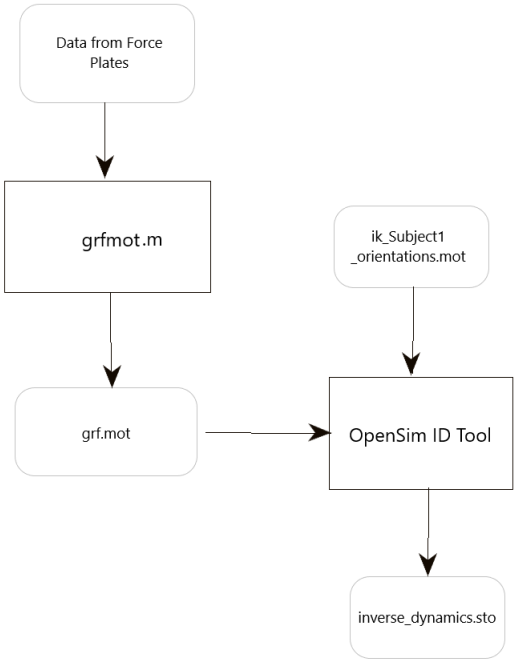


Figure 3.18. Block diagram of the kinetic analysis system

There are two phases in kinetic analysis. In the first phase, the data obtained from the force platform is calibrated to obtain moment and force data from the sensor electrical outputs. Calibration Matrix C1 for Force Plate 1, and Calibration Matrix C2 for Force Plate 2 are used. The mathematical expression of the calibration process is given in Eq. 3.23. In this equation, S is unamplified sensor outputs, C is calibration matrix, F is force in Newtons, and M is moment in Newton.meters. Therefore, amplified force plate outputs should be divided by 10 for the z-axis of the force plate to calculate the unamplified force of the z-axis and should be divided by 20 for all others to compute unamplified outputs.

$$C1 = \begin{bmatrix} -1281.5 & -18.1 & -2.3 & -3.5 & -7.0 & -10.3 \\ -26.6 & 1272.1 & -3.3 & -3.5 & -4.9 & -27.7 \\ 25.5 & 3.6 & 1878.8 & 20.6 & -3.7 & -12.1 \\ 3.8 & -147.0 & 0.3 & 581.8 & 6.8 & 2.9 \\ -146.0 & -0.7 & -0.4 & 2.3 & 402.0 & -2.0 \\ 1.3 & -3.7 & -0.7 & 5.0 & -0.03 & 295.5 \end{bmatrix} \quad (3.21)$$

$$C2 = \begin{bmatrix} 1510 & -29 & 18 & 5 & -6 & -8 \\ 34 & 1519 & 1 & -3 & -1 & -33 \\ -33 & -1 & 3014 & 23 & 4 & -19 \\ -5 & -179 & -1 & 789 & 8 & 3 \\ 176 & 0 & -3 & 6 & 551 & -2 \\ 0 & -5 & -2 & 1 & 2 & 354 \end{bmatrix} \quad (3.22)$$

$$\begin{Bmatrix} F_x \\ F_y \\ F_z \\ M_x \\ M_y \\ M_z \end{Bmatrix} = [C] \begin{Bmatrix} S_{F_x} \\ S_{F_y} \\ S_{F_z} \\ S_{M_x} \\ S_{M_y} \\ S_{M_z} \end{Bmatrix} \quad (3.23)$$

Secondly, the calculation of point of application of force is carried out by using Eqs. 3.24 and 3.25. The application points, relative to the center of the force platform, are x and y, and h is the thickness of the force plate h is accepted as 0,005 m.

$$x = \left(-h.F_x - M_y\right)/F_z \quad (3.24)$$

$$y = \left(-h.F_y + M_x\right)/F_z \quad (3.25)$$

After these processes, force plates' coordinate frames are transformed to the OpenSim reference frame; thus "grf.mot" file is generated. This file is given as input to the ID Tool with the IK results. As a result, "inverse_dynamics.sto" file is generated, which contains the moments of the subject's joints.

CHAPTER 4

EXPERIMENTAL SETUP AND DATA ANALYSIS

4.1 Problems of IMU System

Before evaluating the system together with seven sensors, individual sensors were examined. Three problems were identified which are about unit transformation, magnetometer calibration and sensor drift.

Firstly, as mentioned before, IMU sensors consist of three main sensors. These sensors can be used separately or combined. Fused data from all three sensors were used to get accurate orientation for this system. However, before combining the sensor data, unit transformations should be applied to LSM9DS1 sensor outputs (for all three sensors) to correct the output units.

Sensitivity is used to get the correct units for gyroscope and magnetometer outputs. The sensitivity values of the gyroscope and magnetometer are presented in Table 4.1 [34]. According to this table, gyroscope and magnetometer outputs should be multiplied by 8.75/1000 dps/LSB and 0.29/1000 gauss/LSB, respectively, before these outputs can be used in Madgwick Algorithm.

Table 4.1 Sensitivity Values

Symbol	Parameter	Test conditions	Min.	Typ.(1)	Max.	Unit
LA_So	Linear acceleration sensitivity	Linear acceleration FS = ±2 *g*		0.061		mg/LSB
		Linear acceleration FS = ±4 *g*		0.122		
		Linear acceleration FS = ±8 *g*		0.244		
		Linear acceleration FS = ±16 *g*		0.732		
M_GN	Magnetic sensitivity	Magnetic FS = ±4 gauss		0.14		mgauss/ LSB
		Magnetic FS = ±8 gauss		0.29		
		Magnetic FS = ±12 gauss		0.43		
		Magnetic FS = ±16 gauss		0.58		
G_So	Angular rate sensitivity	Angular rate FS = ±245 dps		8.75		mdps/ LSB
		Angular rate FS = ±500 dps		17.50		
		Angular rate FS = ±2000 dps		70		

Moreover, the sensitivity value of the accelerometer is presented in Table 4.1 According to the data sheet of LSM9DS1, accelerometer outputs should be multiplied by 0.061/1000 g/LSB. This information has been verified by series of experiments.

In these experiments, the calibration constant of the accelerometer is determined by using gravitational acceleration. Sensors were placed at configurations that are presented in Figure 4.1, Figure 4.2 and Figure 4.3, respectively, and data was collected when sensors were standing still. The data from accelerometer axes (that have gravitational effects) of all seven sensors at each configuration were used to find the calibration constant. The mean and standard deviation of the measurements are 16340 LSB/g (0.0612 mg/LSB) and 373 LSB/g, respectively. So, all accelerometer data should be divided to 16340 to get 1 g for Madgwick input.

Figure 4.1. Sensor configuration such that gravitational effect on the z-axis

Figure 4.2. Sensor configuration such that gravitational effect on the x-axis

Figure 4.3. Sensor configuration such that gravitational effect on the y-axis

The second problem is magnetometer calibration. Because of the magnetic interference, erroneous heading estimates from the magnetometer might be obtained [36]. A magnetometer calibration process is challenging since whenever the magnetometer is placed at a new place, it should be recalibrated [37]. Therefore, Madgwick Distortion Compensation was used to overcome this problem which is explained in Section 3.4.

The last problem is gyroscope drift. It is a frequent problem for IMU studies, and the Butterworth filter can be used to remove noise from raw data and correct the gyroscope drift [38], [39]. 6^{th}-order lowpass Butterworth filter with a cut-off frequency of 6 Hz was used for this system for data sampled at 100 Hz [14].

All the problems of the IMU sensors are retested by static experimental setups, namely single sensor experiments, multiple sensor experiments on mechanical test equipment and also on subject experiments.

4.2 Single Sensor Motion Analysis

Before assessing all sensors, one sensor's orientation was tested by conducting a series of experiments. All different orientations of the sensors (foot, right and left legs and pelvis placement) were evaluated separately. The initial orientation of the foot sensors is presented in Figure 4.4. The related sensor is rotated around a single axis (x, y, z) at different pre-defined angles (±30, ±45, ±60, ±90) in these experiments and readings were tested.

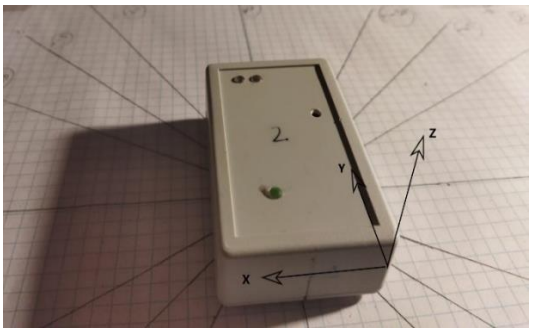


Figure 4.4. Experimental setup for single sensor motion analysis

Before using sensor output in the Madgwick Algorithm, axes of the sensor are arranged in North-East-Down (NED) order. For example, the y-axis for the foot placement orientation shown in Figure 4.4 was changed to x, x changed to -y, and z changed to -z. After using this data as an input of the Madgwick Algorithm, there are two options to reset the initial orientations of the sensors. The resetting process is the alignment of the initial sensor frame to the Earth reference frame. It can be reset by using Euler angles, then converted to quaternion or reset directly as a quaternion. Both options have been evaluated in this thesis.

The first option is using the Madgwick Algorithm output as Euler Angles to reset the initial orientation before changing into a quaternion. The steps of the right leg sensor processes are presented in Figure 4.5 and Figure 4.6. The raw output of the Madgwick Algorithm as Euler Angles is presented in Figure 4.5; it can be seen that the angle switches +180 degrees after it passes to -180 degrees. Hence, the first correction is the correction of the shift, and the result of this process is presented in Figure 4.6. The second step is the resetting of the initial orientation of the sensor by using the first twenty data points, and the final result after operations is presented in Figure 4.7.

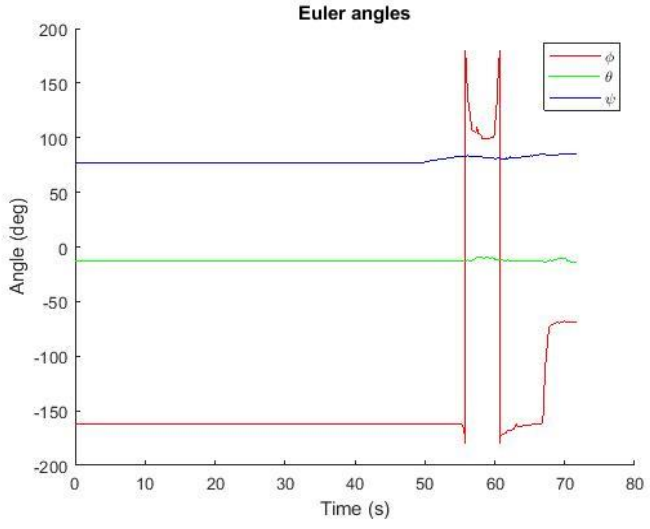


Figure 4.5. The first output of the right leg sensor

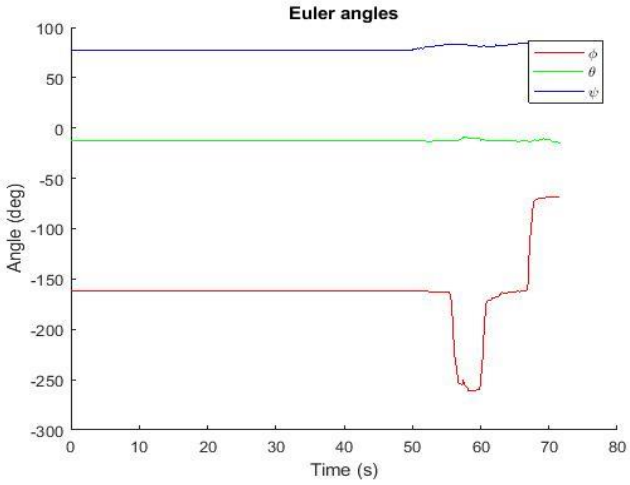


Figure 4.6. Angle correction of the right leg sensor

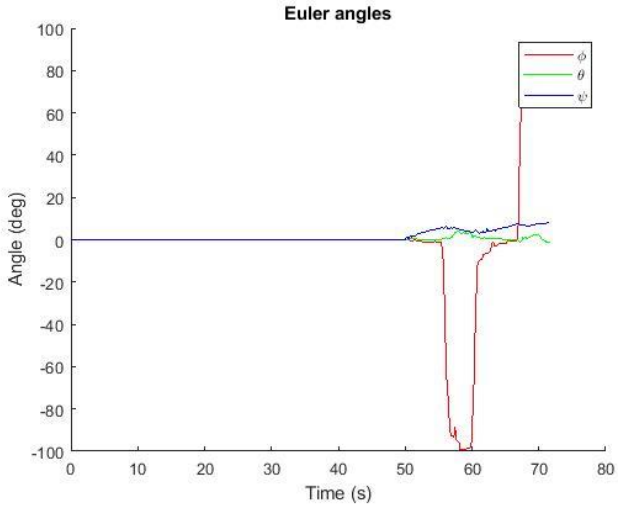


Figure 4.7. Offset correction of the right leg sensor

The same procedures for the Euler Angle option and the result for the left leg sensor are presented in Figure 4.8, Figure 4.9 and Figure 4.10, respectively.

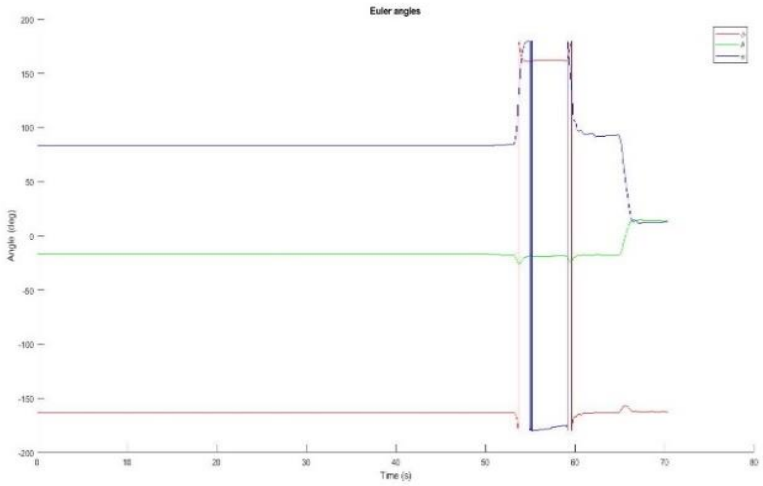


Figure 4.8. The first output of the left leg sensor

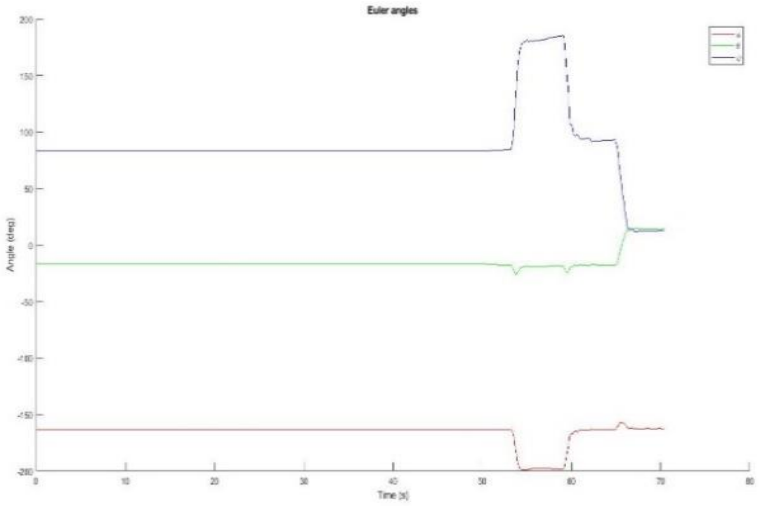


Figure 4.9. Angle correction of the left leg sensor

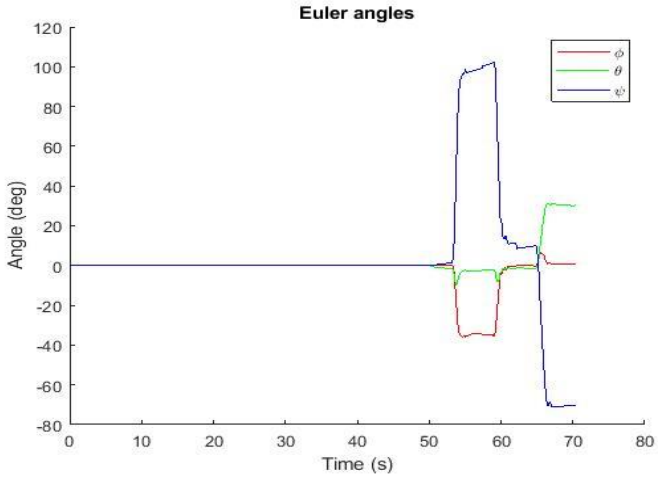


Figure 4.10. Offset correction of the left leg sensor

Results of the foot sensor rotated around the +x axis at +60 degrees, +y axis at +45 degrees, and +z axis at -30 degrees are presented below.

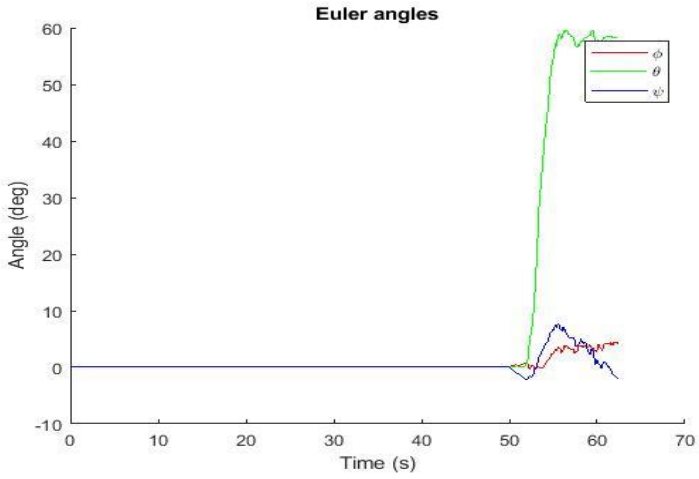


Figure 4.11. Foot sensor rotated around +x axes at +60 degrees

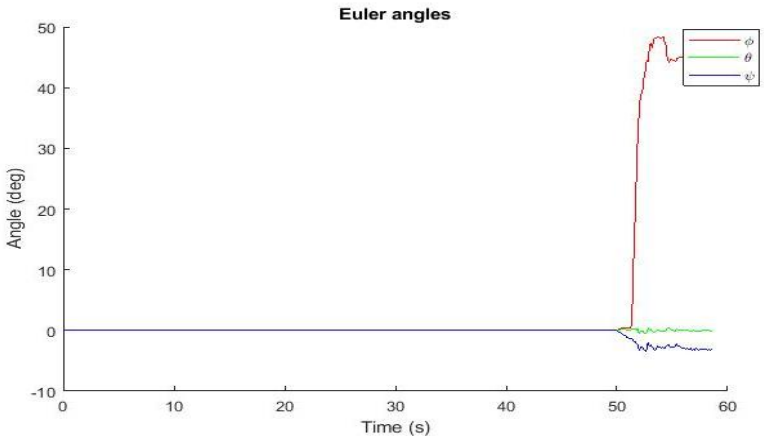


Figure 4.12. Foot sensor rotated around +y axes at +45 degrees

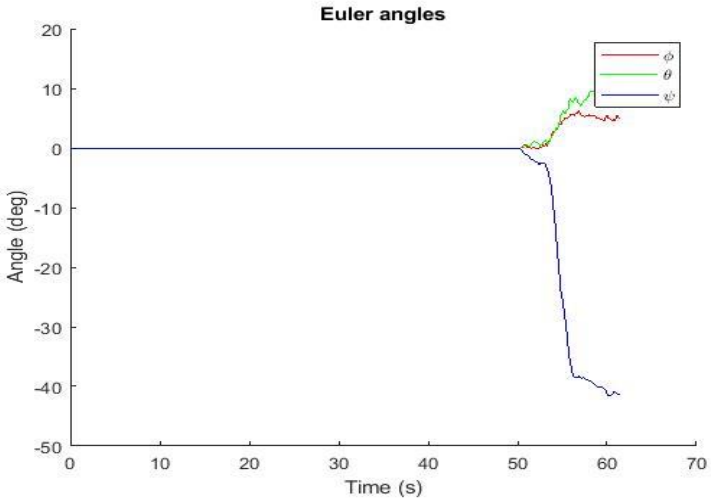


Figure 4.13. Foot sensor rotated around +z axes at -30 degrees

Additionally, the first experimental results with Euler Angles showed that in the case of NED ordered input, Madgwick Algorithm sensor outputs are in North-West-Up (NWU) order. Therefore, before using the data for OpenSim, the rotation of NWU ordered IMU data to the OpenSim world frame is defined as -pi/2 rad around x-axes.

The second option is using the Madgwick Algorithm output directly as quaternion to reset the initial orientation. Equation 4.1 is used to reset the orientation [35]. $\breve{q}_f$ is the main quaternion data that includes examined movement, $\breve{q}_m$ is the quaternion data of the initial orientation of the sensor, and * is the quaternion product.

$$\breve{q} = \breve{q}_f * \breve{q}_m \tag{4.1}$$

After this step, all the sensor placement orientations were evaluated for each axis and frames were arranged in North-West-Up order. The second option was chosen because of Euler Angles' gimbal lock problem.

4.3 Multiple Sensor Motion Analysis with Mechanical Test Equipment

There are seven sensors placed on a human in our analysis setup. Two sensors are placed on the right and left feet, two at distal part of right and left legs, two at proximal part of right and left legs, and one at the pelvis of a human for the lower extremity analysis system. For the upper extremity analysis system, two sensors are placed on the right and left hands, two at distal part of right and left arms, two at proximal part of right and left arms, and one at the torso of a human. When these sensors are placed on human limbs, a couple of problems might affect the signal.

The first thing that may disturb the signal is noise. Butterworth filter is one of the most used filters in the literature for preprocessing the IMU data in human biomechanics [40], [41]. Low pass Butterworth filter with 6^{th} order 6 Hz cut-off frequency was used for the system to remove noise [14].

Secondly, the OpenSim world frame is different from the Madgwick Algorithm World frame. Therefore, it should be transformed by using the sensor to OpenSim rotations. As mentioned in the previous chapters, the sensor frame orientations are transformed in NED order before using in Madgwick Algorithm, and the output of Madgwick Algorithm is defined in NWU order. So, sensor to OpenSim rotation is defined as -pi/2 at x-axes. Section 3.1 explains this transformation process.

Additionally, the initial positions of the IMU sensors are important. OpenSim model must be calibrated before processing the movement data. In the calibration process, initial positions of the IMU frames are found by using OpenSim model body segments and IMU calibration data. The calibration pose of the model was chosen as the default pose, which is all the joints of the model at neutral. Neutral pose is

presented in Figure 4.14. The subject should stand in the same pose as the neutral pose when data acquisition is started.

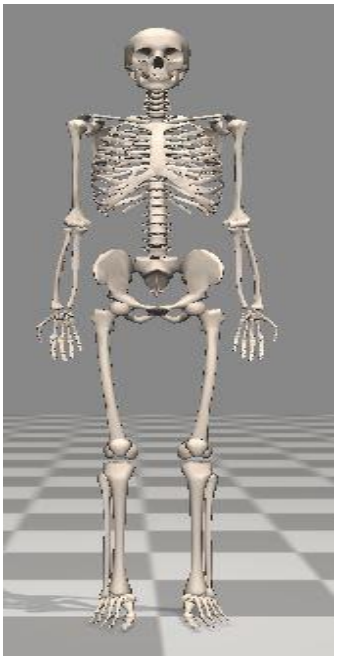


Figure 4.14. Neutral pose of the Rajagopal model

The last effect is the precision of the sensors and human movement. The sensors are LSM9DS1 by SparkFun. This sensor type was chosen because of the low price, but it has slightly worse accuracy (compared to the majority of sensors used in this field) since it is linear acceleration typical zero-g level offset accuracy is ±90 mg, zero-gauss level is ±1 gauss and angular rate typical zero-rate level ±30 dps [34], [42]. Additionally, the electronic circuit and embedded communication systems of the sensors are custom-made. So, the accuracy of the overall system was not known.

In the Section 4.1, the individual sensor problems and solutions are presented, but the overall system tests were needed to assess the related problems above. This need led to design and manufacture passive mechanical system to assess multiple sensors with known angles.

4.3.1 Design of Mechanical System

Before designing the mechanical system, literature was searched to decide the dimensions. Rajagopal_2016 model was used as a biomechanical model [43]. Rajagopal et al. used a 170 cm tall, 75 kg male model's bony geometry and dimensions [43], [44]. The exact dimensions were scaled and used to design a mechanical model. The reason for scaling is to use less material. The scale of the designed equipment to anthropometric data is 0,72. IMU sensor dimensions were considered to decide on the scale.

The designed model has six joints, two at the hip, two at the knee, two at the ankle, and four DOF at each joint. The fourth DOF is caused by the design of the standard parts. Standard parts were designed for joints because it was thought that rather than changing the femur part, changing a relatively smaller standard part is more efficient.

Designed test equipment is presented in Figure 4.15.

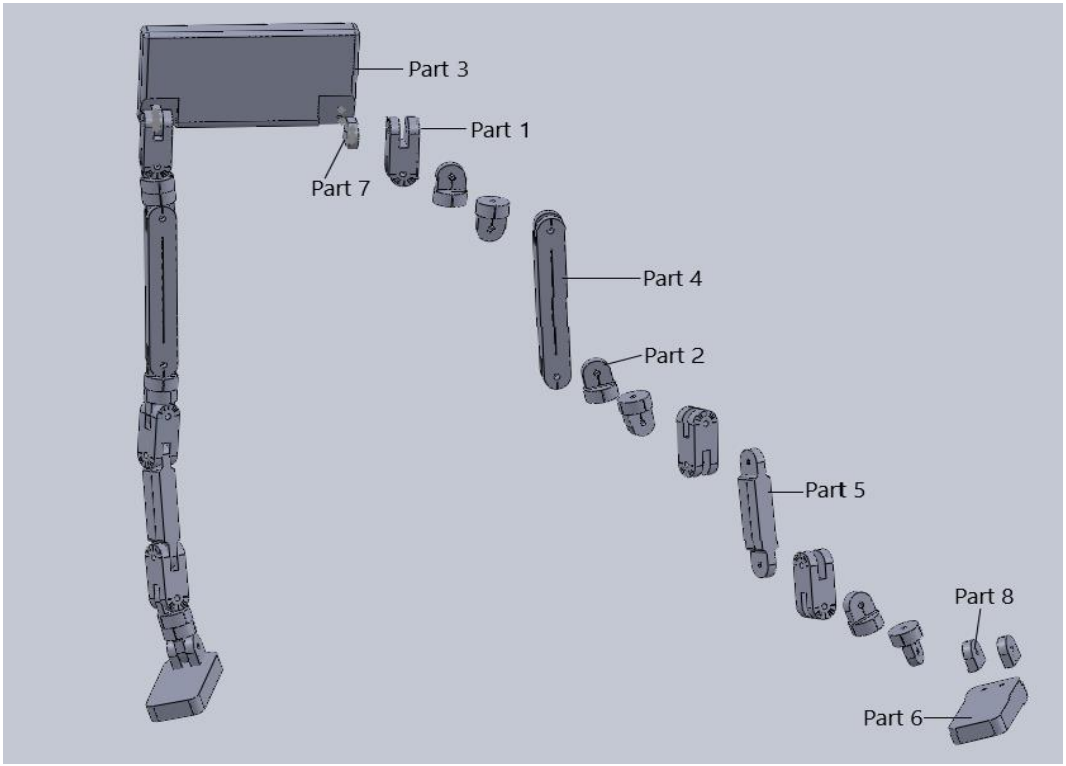


Figure 4.15. Mechanical test equipment

Part 1, Part 2, Part 7 and Part 8 are the standard parts of the system. Joints of the system consist of these parts. Part 3 is a model of the pelvis. The pelvis sensor is attached to Part 3. Part 4, Part 5 and Part 6 are femur, tibia and foot parts, respectively, where the related IMU sensors bind to these parts with the correct orientation. Technical drawings are available at Appendix A.

4.3.2 Manufacturing of Mechanical System

The designed passive human lower extremity model was manufactured at METU Mechanical Engineering Machine Shop. CNC milling machine and lathe were used for the manufacturing.

Polyoxymethylene (trade name Delrin®) was used to manufacture mechanical test equipment. The reason Delrin® is preferred for this system is that it does not have an effect on the magnetometer. Also, it has dimensional stability and high strength when compared to other material options for this system [45].

The manufactured mechanical system is presented in Figure 4.16.

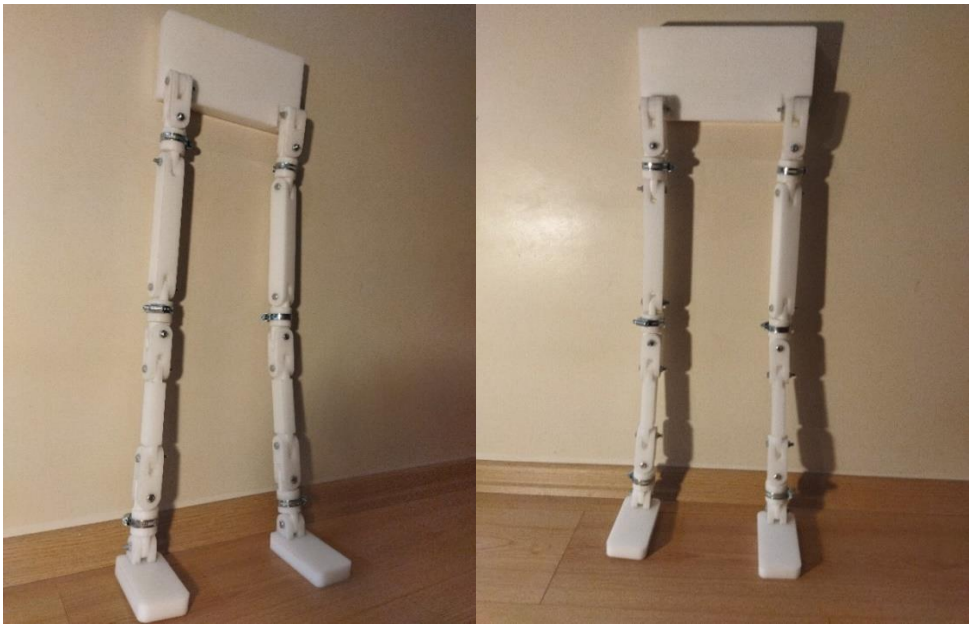


Figure 4.16. Manufactured test equipment

4.3.3 Multi-Sensor Experiments Results for Lower Extremity

Multi-sensor experiments were performed using seven sensors. The sensors are positioned on both feet, two tibias, two femurs, and the pelvis of the mechanical test equipment. Positioning is presented in Figure 4.17.

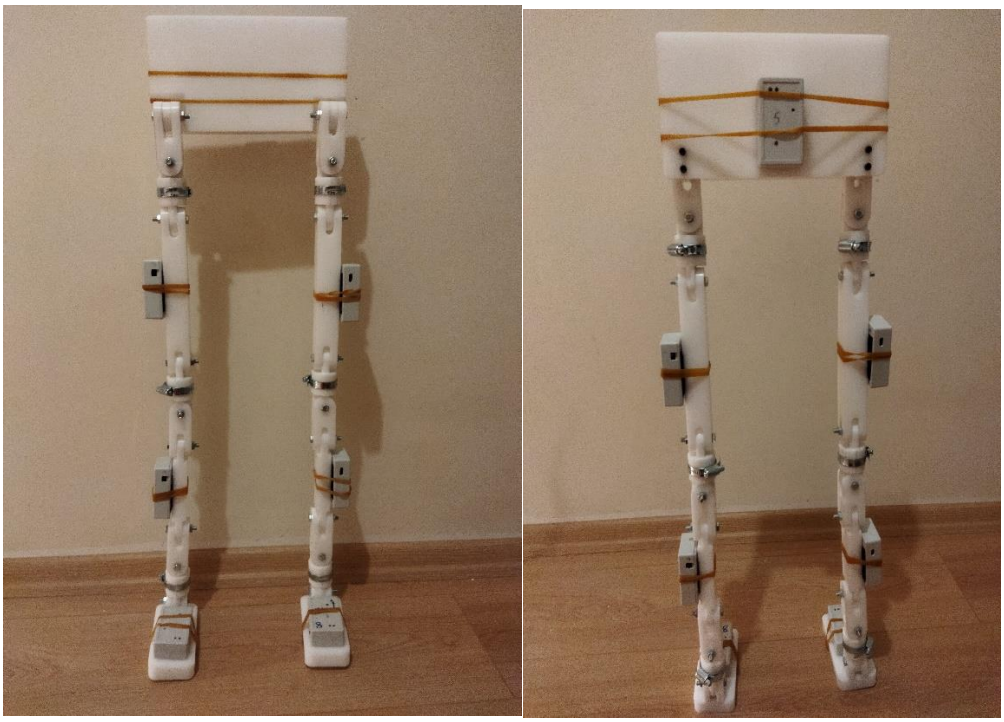


Figure 4.17. Sensor Positions

Data was collected by calibrating the sensors to receive data simultaneously, all the sensor axes were arranged in NWU order, and joint angles were obtained using the Madgwick Algorithm and OpenSim Inverse Kinematics module. Detailed information about Madgwick Algorithm and OpenSim IMU Placer Tool, and OpenSim IMU Inverse Kinematics module can be found in Section 3.5.1 and Section 3.5.2, respectively.

There were a couple of experiments conducted in this part. Firstly, right and left ankle joints move separately in the direction of dorsiflexion-plantar flexion, abduction-adduction, and inversion-eversion at approximately 30, 45, 60, 90

degrees. In the second group experiments, just about 30, 45, 60, and 90 degrees of flexion-extension, abduction-adduction, and internal-external rotation movements were carried out separately at the right and left knee joints. Third group experiments include around 30, 45, 60, and 90 degrees of flexion-extension, abduction-adduction, and internal-external rotation movements at right and left hip joints. Lastly, the overall model was turned right and left. Then, as a result of the analysis, it was examined whether the joint angles performed in the experiment were obtained as a result of the analysis.

As a part of this thesis, two experimental results are examined deeper, which are 30 degrees knee flexion-extension and 30 degrees hip adduction-abduction. The knee flexion-extension experiment is presented in Figure 4.18, and the hip adduction-abduction experiment is presented in Figure 4.19.

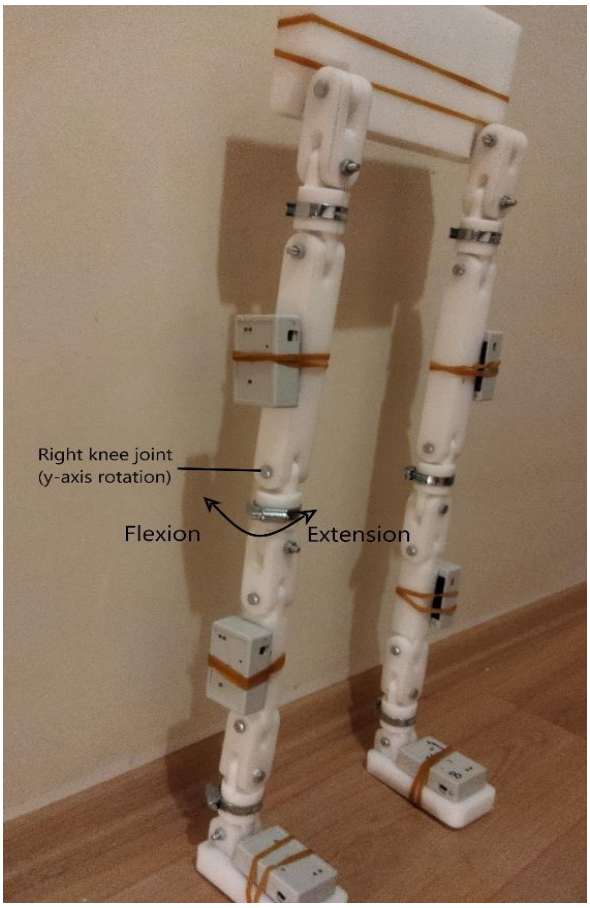


Figure 4.18. Knee flexion-extension experiment

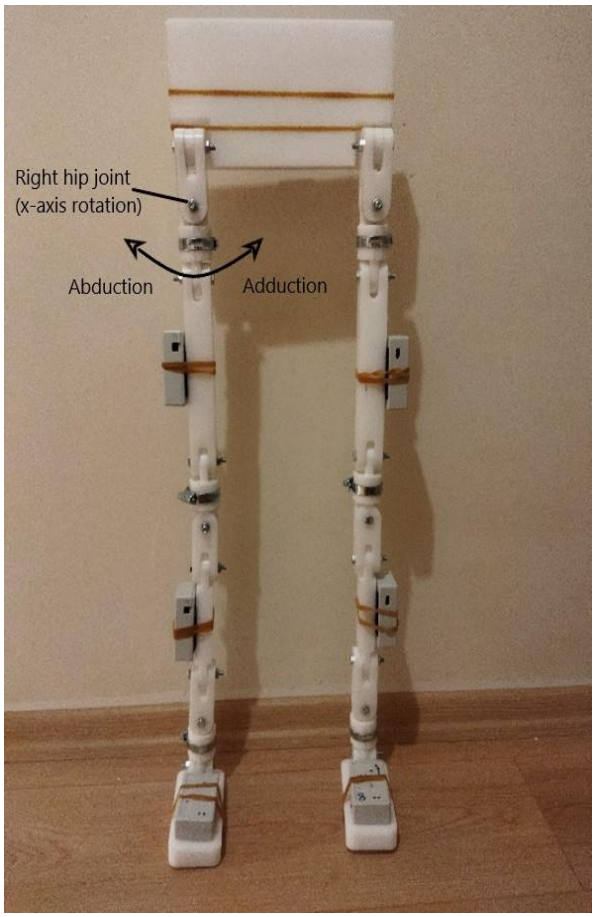


Figure 4.19. Hip adduction-abduction experiment

Rajagopal_2015 model is used in OpenSim. OpenSim IMU Placer Tool and IMU IK Tool adjustments for the right knee 30-degree experiment can be seen in Figures 4.20 and 4.21.

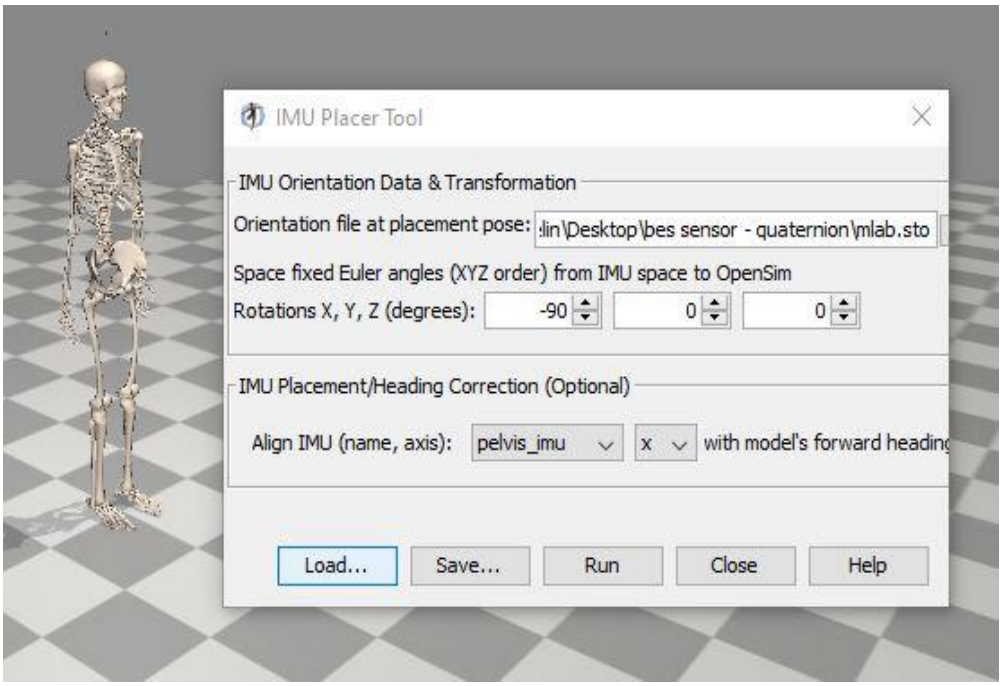


Figure 4.20. OpenSim IMU Placer Tool adjustment

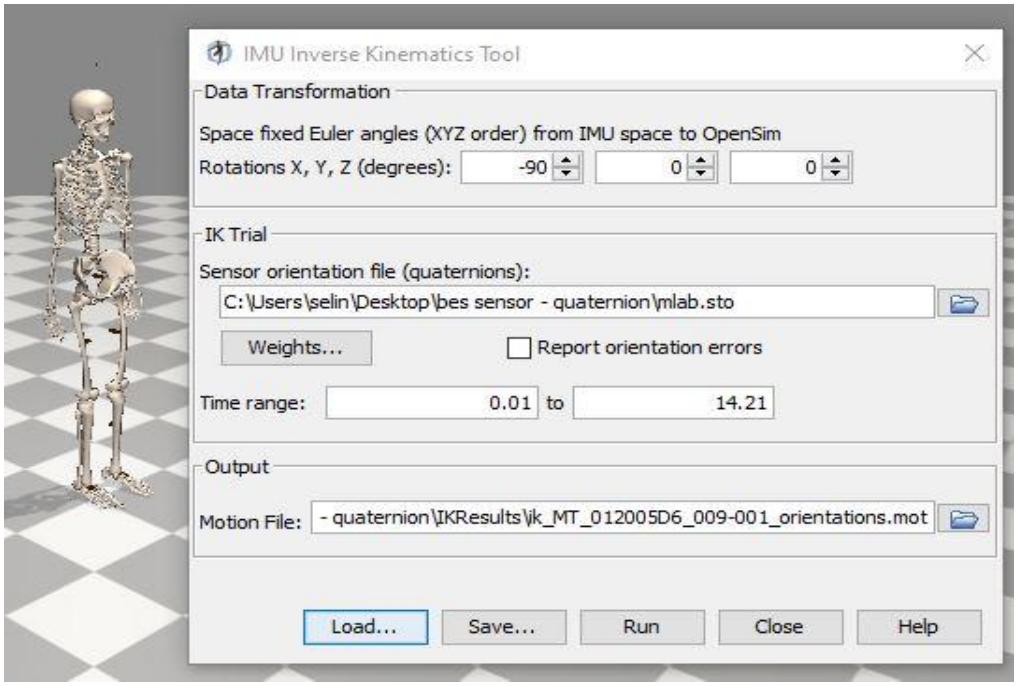


Figure 4.21. OpenSim IMU IK adjustment

The test results are presented below.

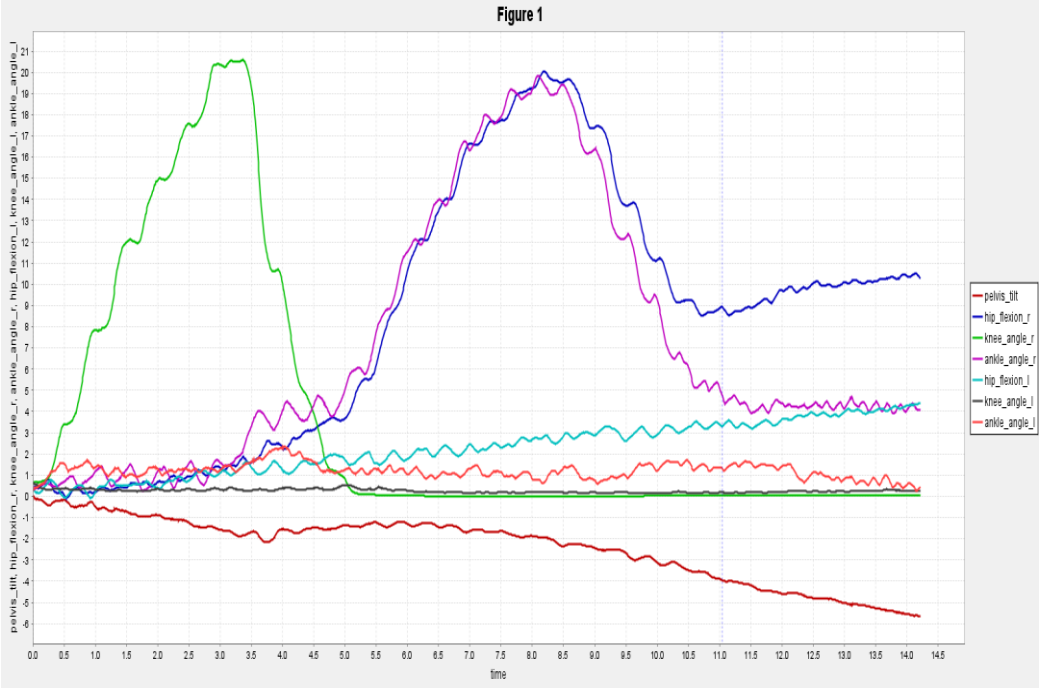


Figure 4.22. Right knee flexion-extension experiment results

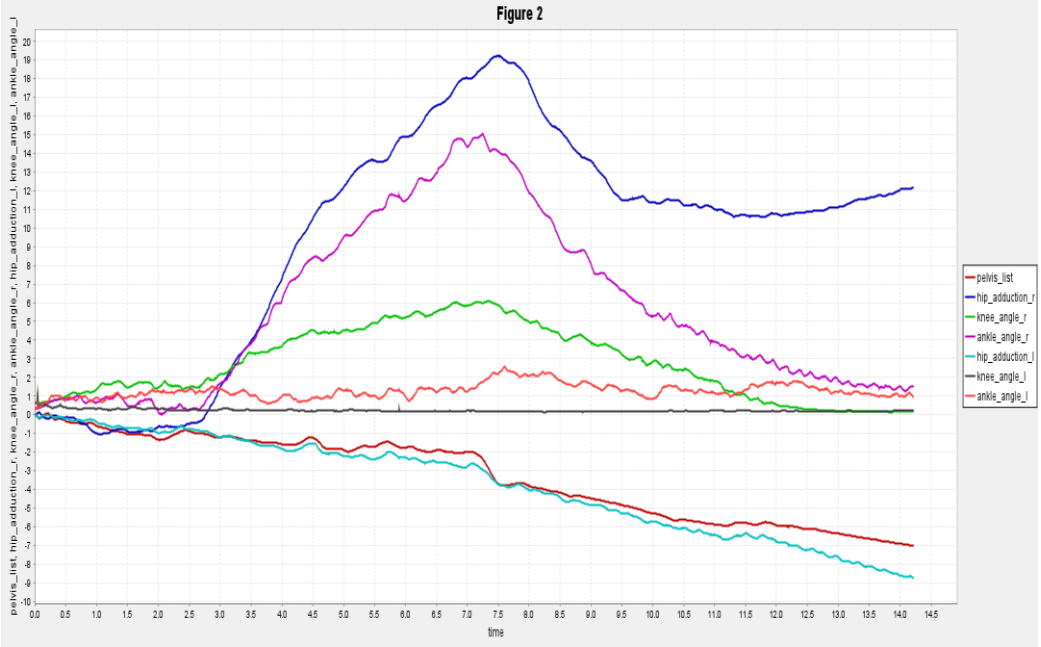


Figure 4.23. Right hip adduction-abduction experiment results

In these experiments, although there was isolated motion in one joint of the mechanical test equipment, the motion was also observed in the other joints of the related limb (in this case right leg). This is because these joints are interconnected with the muscle models in the OpenSim model. Also, Rajagopal model has restrictive motion since it only allows pelvic rotation, pelvic tilt, pelvic list, hip rotation, hip adduction, hip flexion, knee flexion, ankle inversion, ankle dorsiflexion and toe flexion [43]. As an example of the non-isolated motion, in Figure 4.22, flexion was observed in the right hip joint since the Rajagopal biomechanical model does not allow some movements (e.g. hyperextension at knee joint), it performs other allowed movements when there is data from the sensors for this movement.

4.3.4 Multi-Sensor Experiments Results for Upper Extremity

The experiments were performed using seven sensors. Mechanical test equipment was used for these tests. Mechanical testing equipment was designed for lower extremity experiments. However, since the angles and related joint movements are the same, it was used in these experiments by making all the joints of the mechanical test equipment similar to the neutral position of the upper extremity of the Rajagopal model.

The sensors are positioned on both feet parts, two tibia parts, two femur parts, and the pelvis parts of the mechanical test equipment. The foot of the mechanical test equipment is accepted as hand, the proximal part of the leg as the proximal part of the arm, the distal part of the leg as the distal part of the arm and pelvis as the torso.

There were a couple of experiments conducted with this arrangement. Firstly, right and left wrist joints move separately in the three directions. In the second group experiments, right and left elbow joint were performed flexion-extension and pronation-supination movements. Third group experiments include flexion-extension, abduction-adduction, and internal-external rotation movements at right and left shoulder joints.

The right shoulder adduction-abduction experiment result is presented in Figure 4.24. In this experiment, approximately 10-degree right shoulder abduction movement and an approximately 90 degrees right shoulder abduction movement were performed.

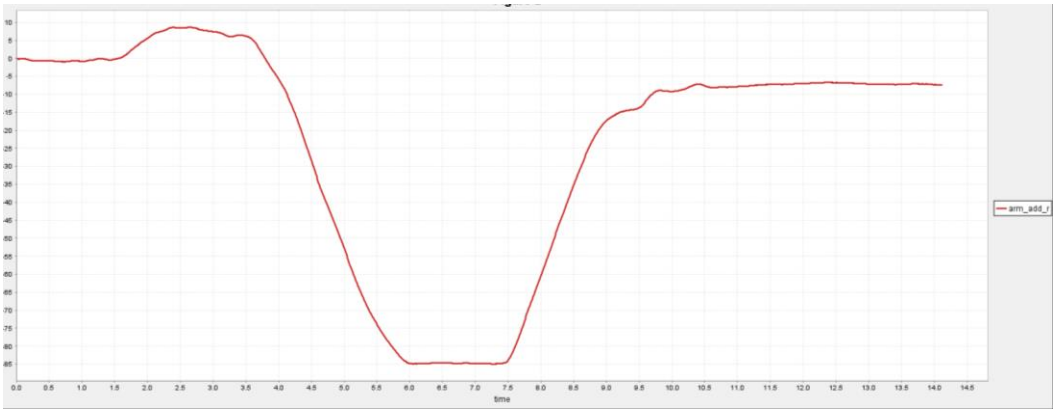


Figure 4.24. Right shoulder adduction-abduction experiment results

The left shoulder adduction-abduction experiment result is presented in Figure 4.25. In this experiment, the subject performed approximately 80-degree left shoulder abduction movement and approximately 10 degrees left shoulder adduction movement.

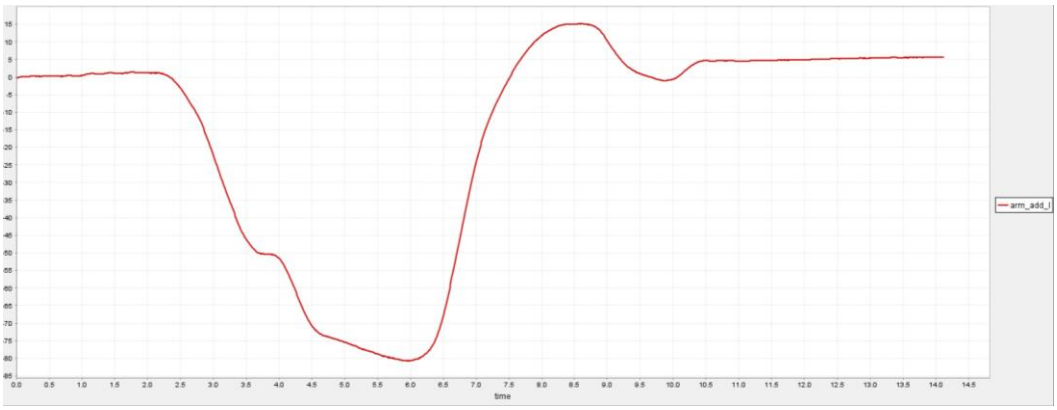


Figure 4.25. Left shoulder adduction-abduction experiment results

The left elbow flexion experiment result is presented in Figure 4.26. In this experiment left elbow flexed to 80-90 degrees.

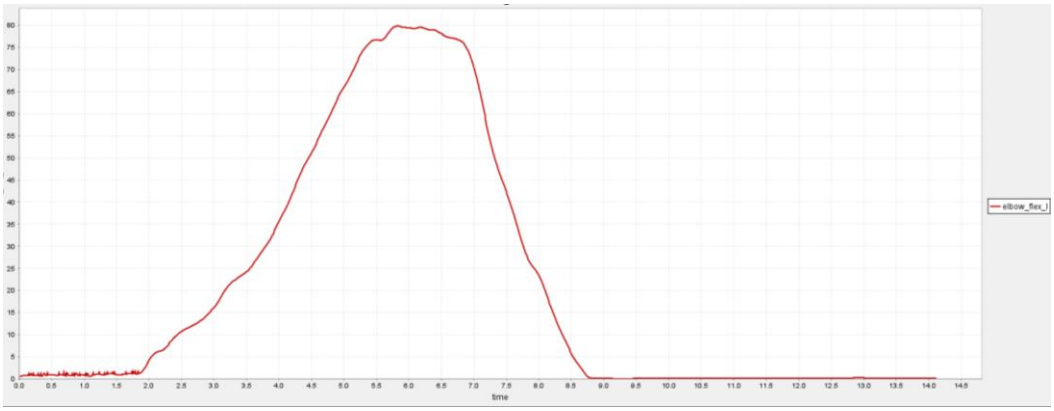


Figure 4.26. Left elbow flexion experiment results

Results and patterns are consistent with the movements. However, as similar to lower extremity tests, the movement of one joint also affects other joints. Especially, this movement is a movement that the Rajagopal model cannot perform (restricted movement of the model joints), it is transferred to other joints.

4.4 Multiple Sensor Kinematic and Kinetic Analysis with Human Movement Data

The last step of the motion analysis test is to examine the system with human gait data. For these tests, recorded kinematic data from IMU system and kinetic data from force plates at METU Biomechanics Laboratory were used.

4.4.1 Kinematic and Kinetic Gait Analysis with Human Subject

Ethical approval for this study was obtained from Middle East Technical University Applied Ethics Research Center İAEK with protocol number 0393-ODTUİAEK-2022, which is presented in Appendix B.

In these tests, IMU data was collected from a female subject with sensors attached to the subject's distal and proximal parts of legs, pelvis and feet. IMU data and force plate data were synchronized to collect data simultaneously. Experimental setup is presented in Figure 4.27.

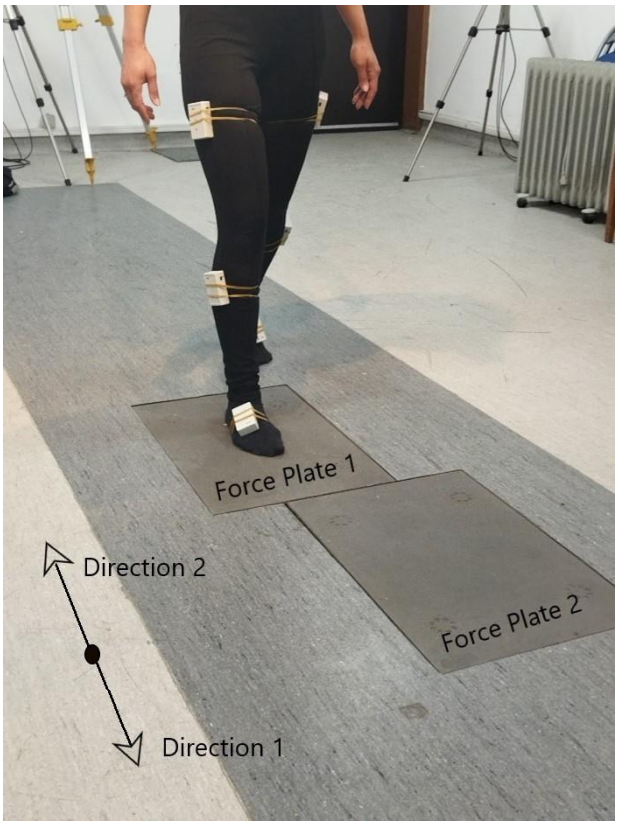


Figure 4.27. Experimental Setup

Forty sets of data from the subject were collected when the subject was walking on the path where force plates were placed. Walking direction is important for force

plates and kinetic analysis. Different codes must be used for different directions. Therefore, the subject walked both in Direction 1 and Direction 2, which is presented in Figure 4.27, to test the code.

The processing of kinematic and kinetic data are described in Chapter 3.

One of the kinematic gait analysis results, when the subject walked to Direction 1, is presented and the results were compared with the literature.

In this experiment, the subject took seven steps, waited 10 seconds before and after movement and third and fourth steps coincided with Force Plate 1 and 2, respectively.

Right hip and left hip flexion results are presented in Figure 4.28. Results are consistent with the experiments of Fukuchi et al. [46]. They found that maximum flexion and extension were 35 and -10 degrees.

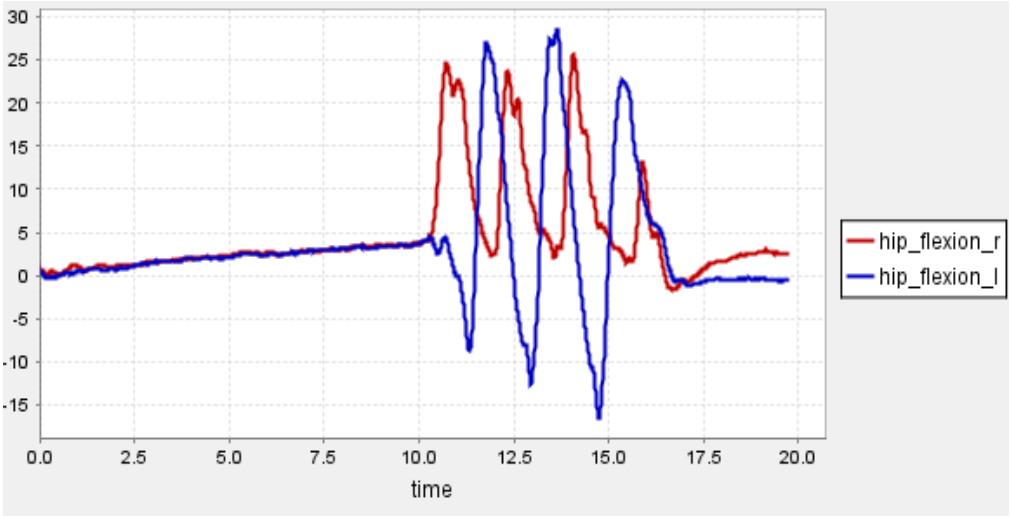


Figure 4.28. Right hip and left hip flexion results

In addition, a comparison of the hip flexion results of the developed system and the study of Bovi et al. is presented in Figure 4.29 [47]. In this figure, it can be seen that the patterns are compatible. The differences are thought to be due to the subject's specific gait pattern.

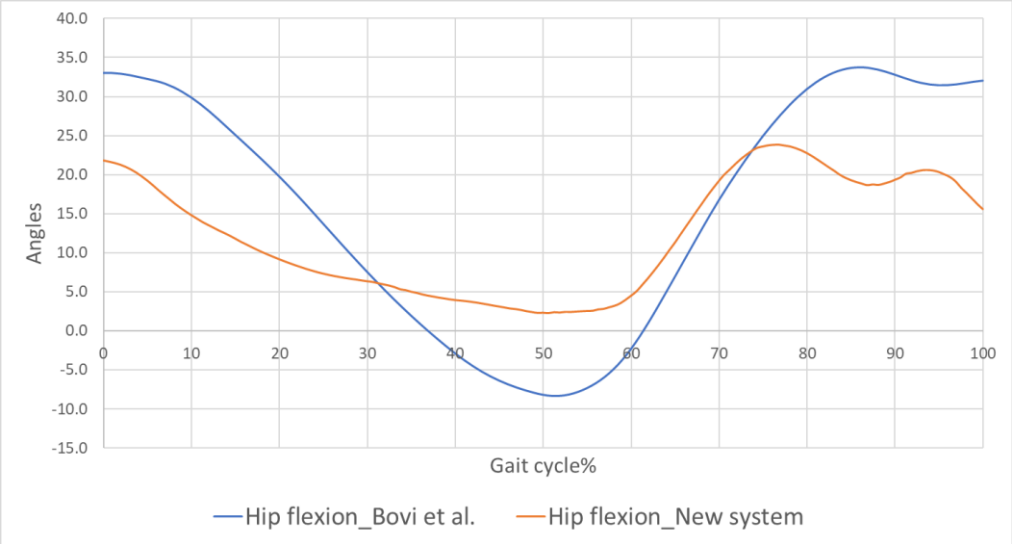


Figure 4.29. Hip flexion comparison for one gait cycle (reproduced from [47])

Right hip, left hip and pelvis rotation results are presented in Figure 4.31. As mentioned before, the subject stands still for 10 seconds before walking. Figure 4.31 shows that there is a drift in the rotation results. Also, drift in the right and left hip adduction, pelvis tilt and pelvis list results are presented in Figure 4.35. This drift problem was caused by the drift observed in the pelvis sensor data. Figure 4.30 shows the pelvis sensor data in the NWU order. Although the subject is standing still in the first 10 seconds, it is seen that the drift slope, especially in the x and z axes, is more noticeable. This drift is consistent with the findings from the pelvis rotation and pelvis list results.

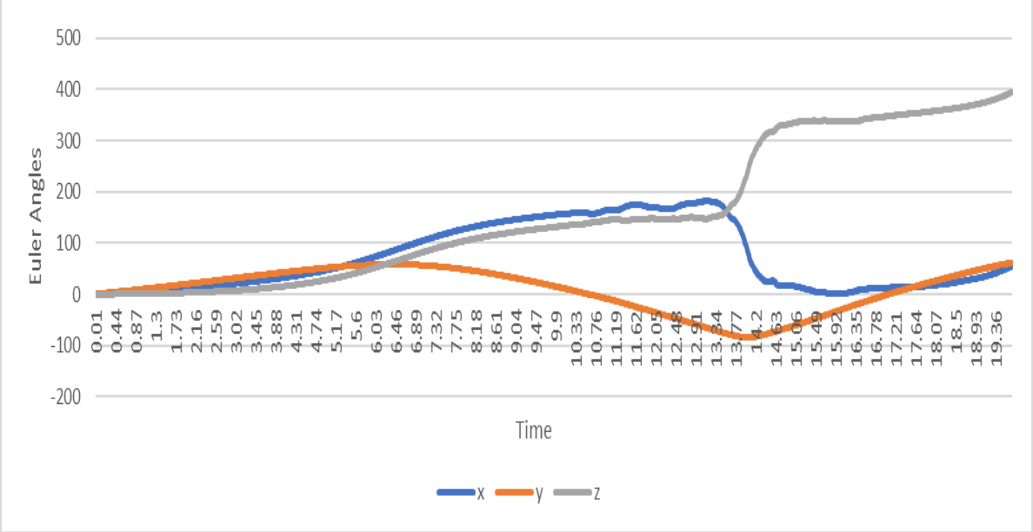


Figure 4.30. Pelvis sensor Euler angles result in NWU order

A slope correction code (presented in Appendix H) was written to solve this problem. In this code, the inclination is measured and corrected by using the first one thousand data points, which coincides with the first 10 seconds of the data.

Right hip, left hip and pelvis rotation results after correction are presented in Figure 4.32.

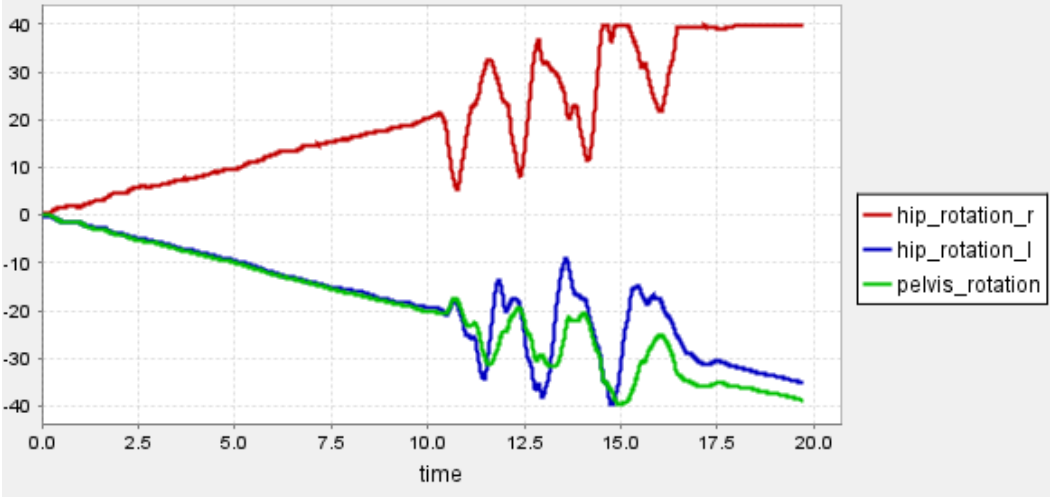


Figure 4.31. Right hip, left hip and pelvis rotation first results

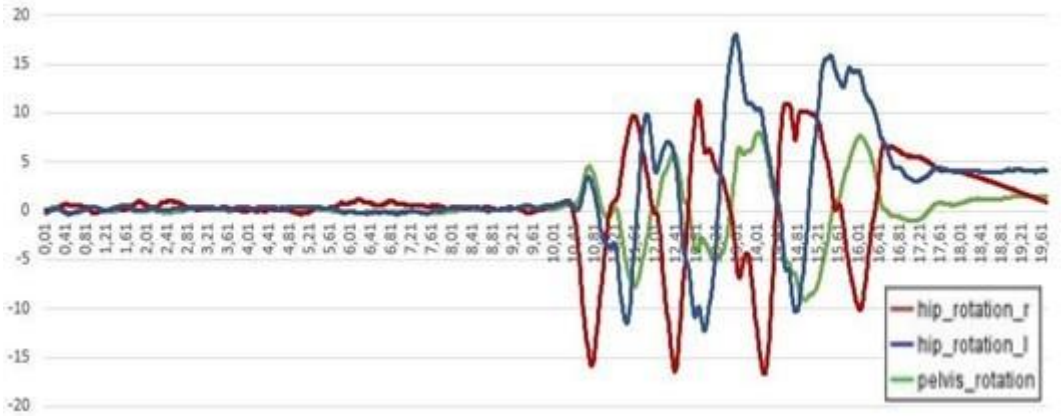


Figure 4.32. Rotation results after slope correction

Different patterns have been found in the literature for hip rotation results. The result calculated by the new system is consistent with the result shared by Fukuchi et al., which is presented in Figure 4.34 [46].

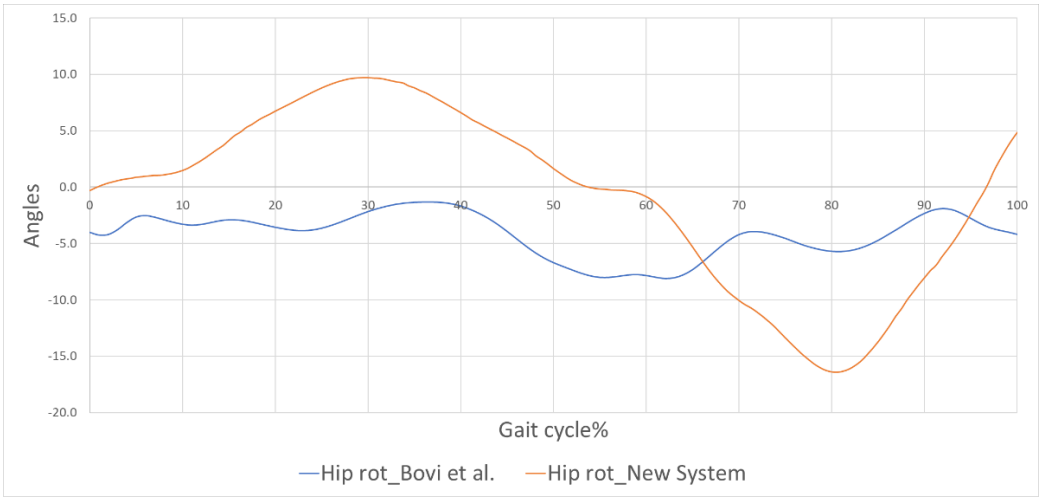


Figure 4.33. Hip rotation comparison for one gait cycle (reproduced from [47])

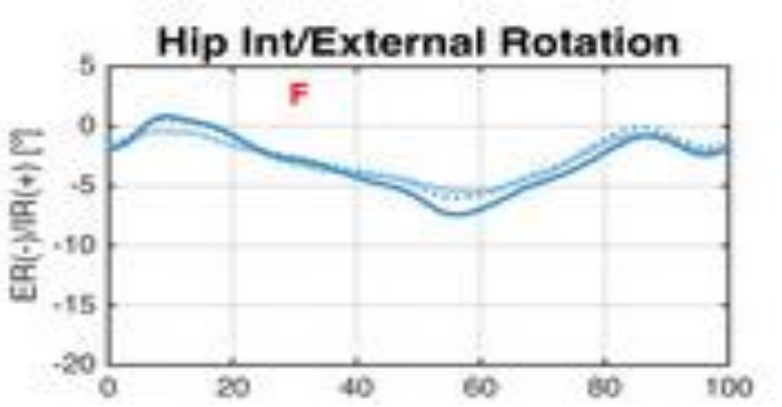


Figure 4.34. Hip rotation for one gait cycle according to Fukuchi et al. [46]

Right and left hip adduction, pelvis tilt and pelvis list results are presented in Figure 4.35. Because of the drift in these results, the same slope correction code was applied to the data. The results after correction are presented in Figure 4.36.

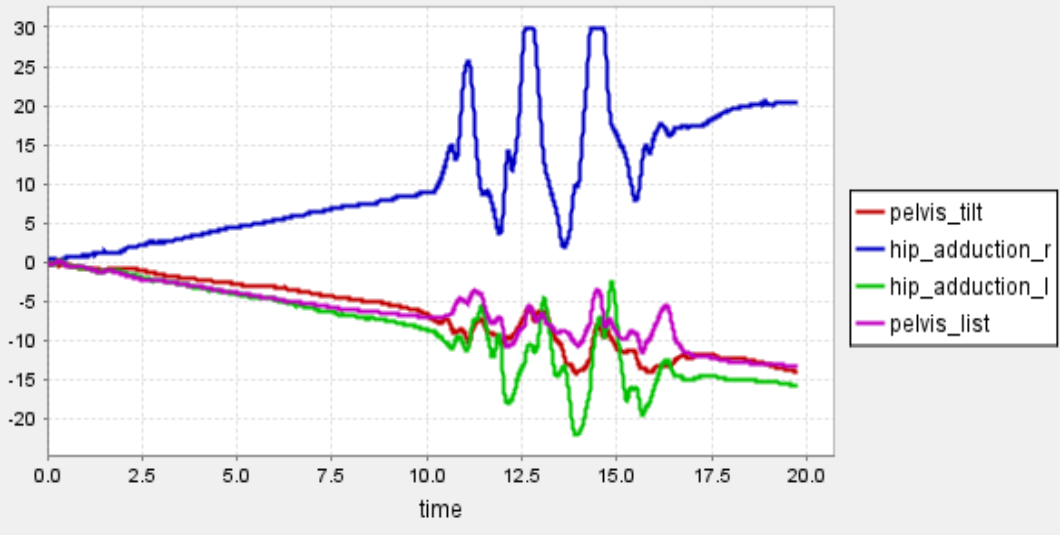


Figure 4.35. Right hip adduction, left hip adduction, pelvis tilt and pelvis list first results

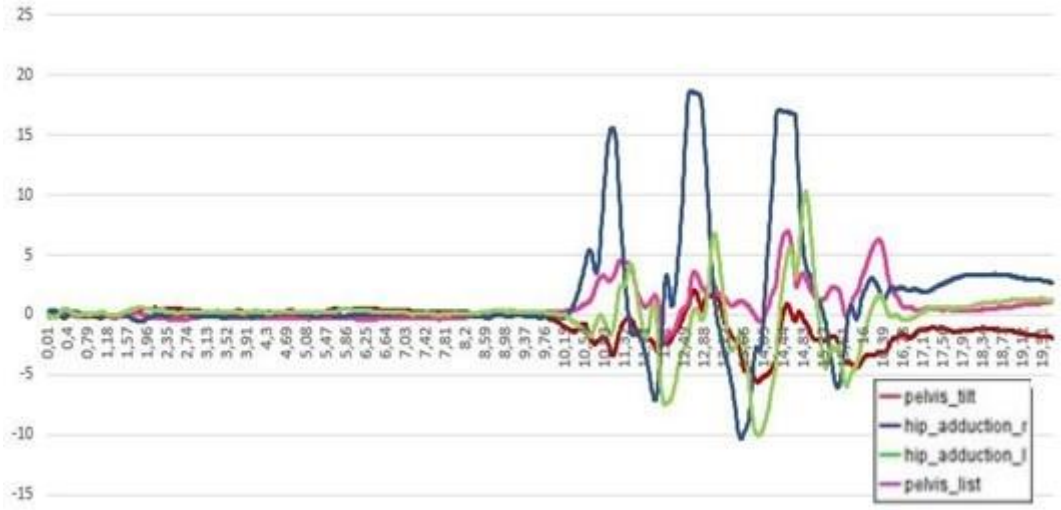


Figure 4.36. Right hip adduction, left hip adduction, pelvis tilt and pelvis list results after slope correction

Different patterns have been found in the literature for pelvis tilt results. The result calculated by the new system is more consistent with the result shared by Fukuchi et al., which is presented in Figure 4.38 [46].

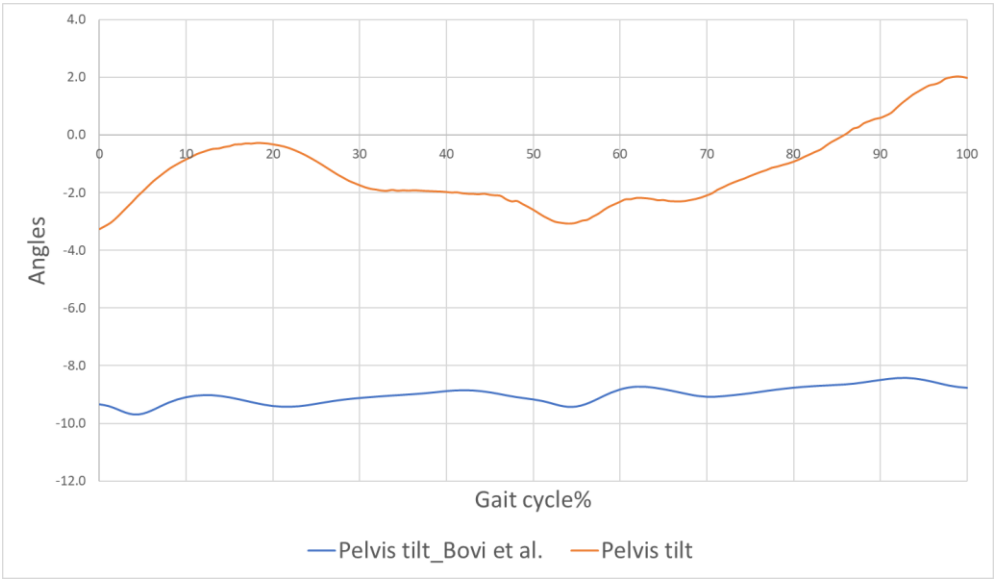


Figure 4.37. Pelvis tilt comparison for one gait cycle (reproduced from [47])

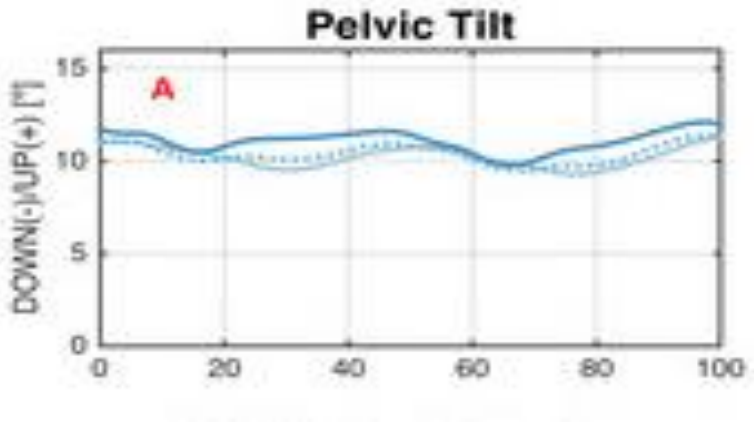


Figure 4.38. Pelvis tilt for one gait cycle according to Fukuchi et al. [46]

It can be seen in the pelvic list comparison results presented in Figure 4.39 that the calculated results are consistent with the literature [47].

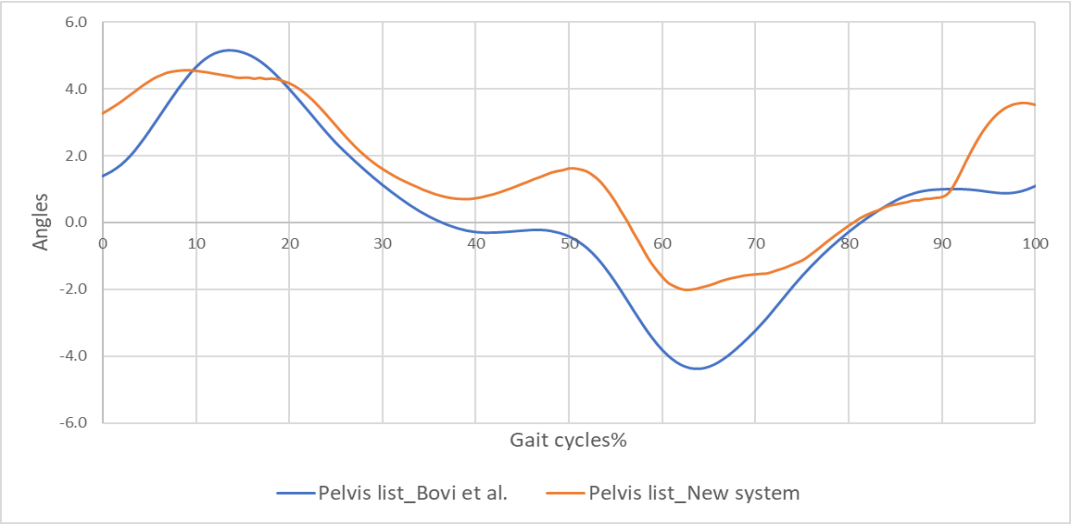


Figure 4.39. Pelvis list comparison for one gait cycle (reproduced from [47])

Right and left knee flexion results are presented in Figure 4.40, and comparison of the results is shared in Figure 4.41. Patterns of the curves are consistent with the experiments of Fukuchi et al. [46] and Bovi et al. [47]. They found that maximum knee flexion was 70 degrees. However, results showed that the subject's left knee

flexion is 30 degrees maximum. It is thought that the subject's unique gait pattern caused this result.

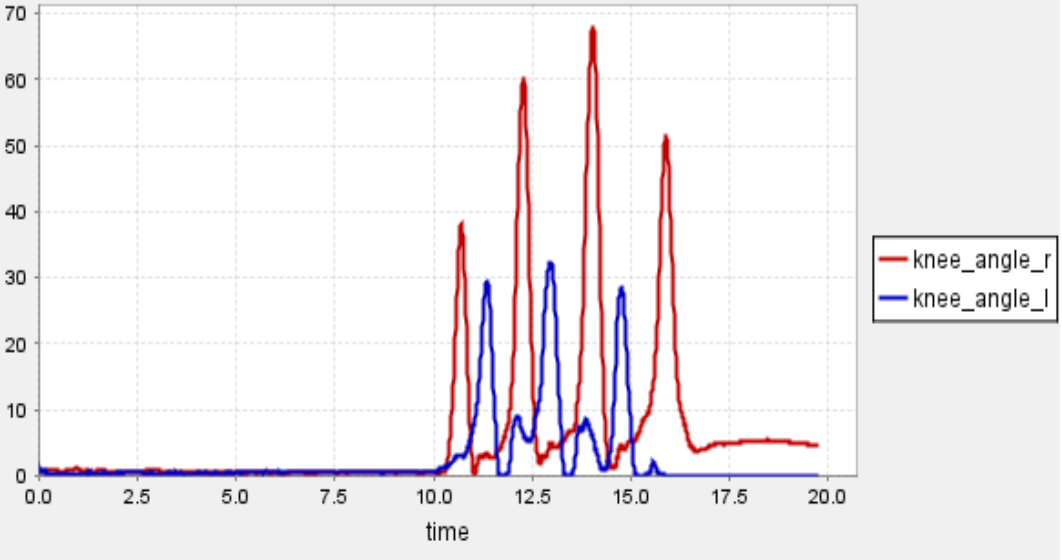


Figure 4.40. Right knee and left knee flexion results

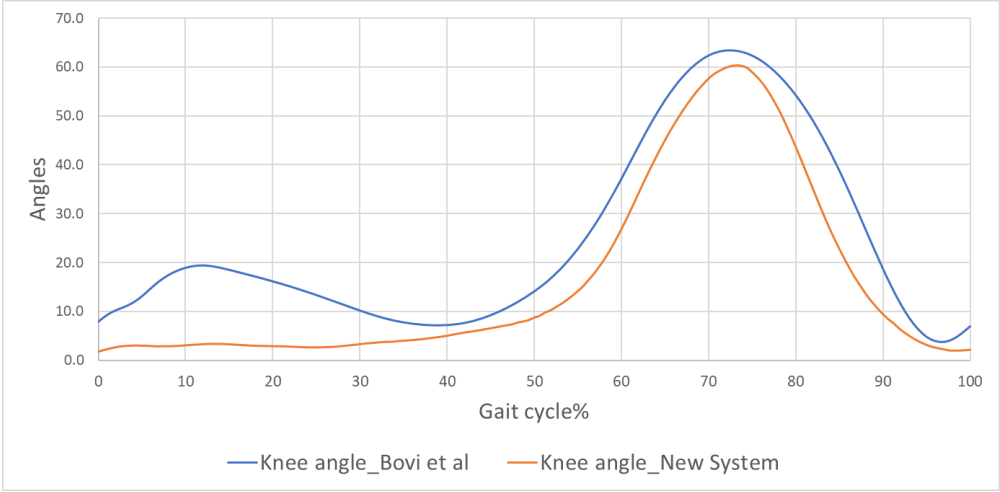


Figure 4.41. Knee flexion comparison for one gait cycle (reproduced from [47])

In several kinematic data (e.g., pelvic list), drift has been detected, and it is thought that the reasons for this are especially the unwanted motion of the pelvis sensor during movement and drift of the pelvis sensor data. The pelvic sensor is the main sensor and therefore is likely to cause some angles to shift. This issue was solved by using the slope correction code. Overall, kinematic analysis results showed that the general pattern is stable.

Before examining the kinematic analysis, GRFs were examined. GRFs from two different kinetic analysis experiments are presented in Figures 4.42 and 4.43. In the first experiment, the subject walked through Direction 1 and in the second experiment subject walked to Direction 2.

Force Plate 1 and 2 coincide with the right and left foot in the first experiment, respectively. In the second experiment, Force Plate 1 coincide with the left step and Force Plate 2 with the right step.

Additionally, GRF from previous kinetic analysis experiments in the same setup with a different subject (male subject walking through Direction 1) is presented in Figure 4.44.

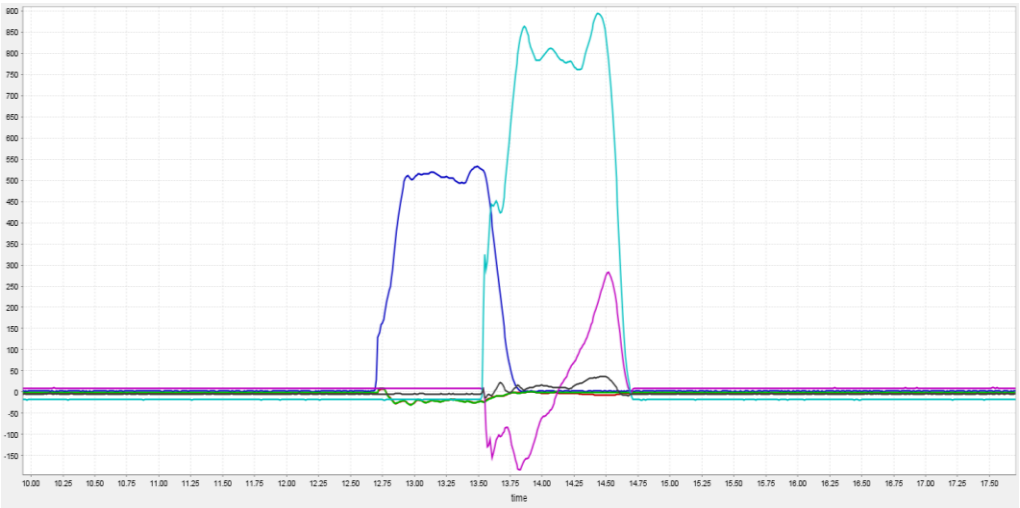


Figure 4.42. GRF in the first kinetic analysis experiment (red line: force plate 1 x axis, dark blue line: force plate 1 y axis, green line: force plate 1 z axis, pink line: force plate 2 x axis, blue line: force plate 2 y axis, grey line: force plate 2 z axis)

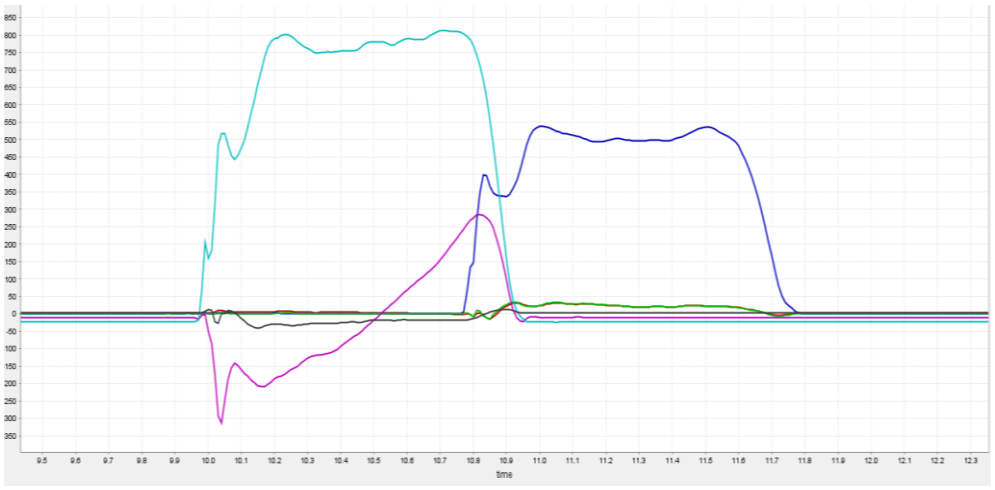


Figure 4.43. GRF in the second kinetic analysis experiment (red line: force plate 1 x axis, dark blue line: force plate 1 y axis, green line: force plate 1 z axis, pink line: force plate 2 x axis, blue line: force plate 2 y axis, grey line: force plate 2 z axis)

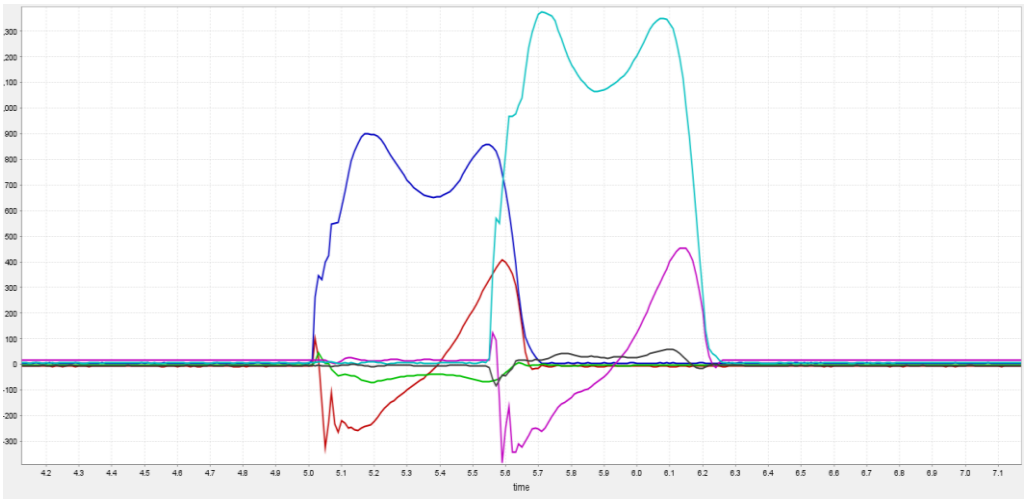


Figure 4.44. GRF example in the previous kinetic analysis experiments (red line: force plate 1 x axis, dark blue line: force plate 1 y axis, green line: force plate 1 z axis, pink line: force plate 2 x axis, blue line: force plate 2 y axis, grey line: force plate 2 z axis)

When comparing the experimental data from previous experiments, it is thought that there is an impairment in the x-axis of Force Plate 1. Therefore, previous experimental data were used to interpret kinetic gait analysis by using data from Force Plate 1. On the other hand, new experimental data were used to interpret kinetic gait analysis by using data from Force Plate 2 (both walking in Direction 1 and Direction 2).

Kinetic gait analysis results from Force Plate 1 are presented in Figure 4.45 for the right leg.

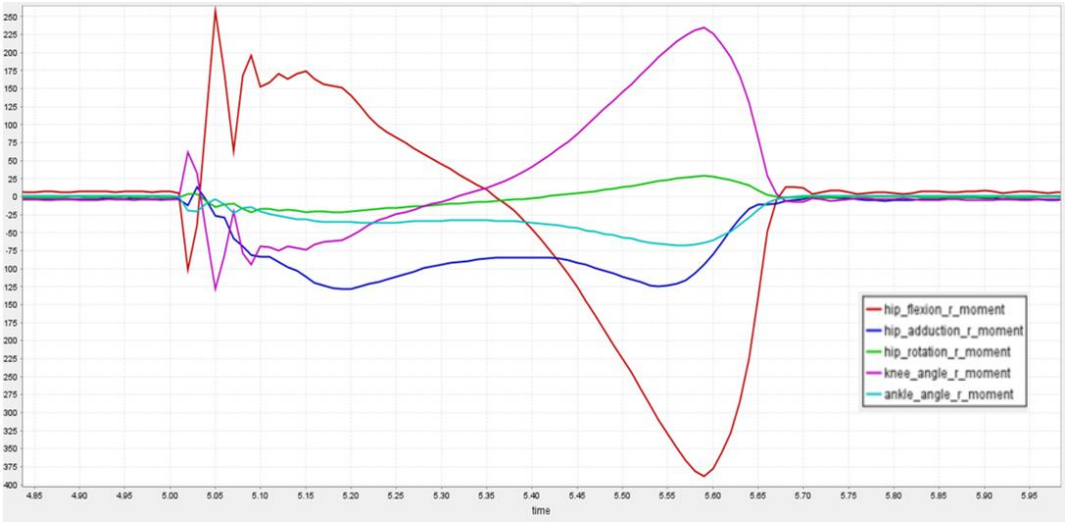


Figure 4.45. Right leg kinetic analysis results

Kinetic gait analysis results from Force Plate 2 (when the subject is walking in Direction 1) is presented in Figure 4.46.

Figure 4.46. Left leg kinetic analysis results when subject walking to Direction 1

Kinetic gait analysis results from Force Plate 2 (when the subject walking in Direction 2) is presented in Figure 4.47.

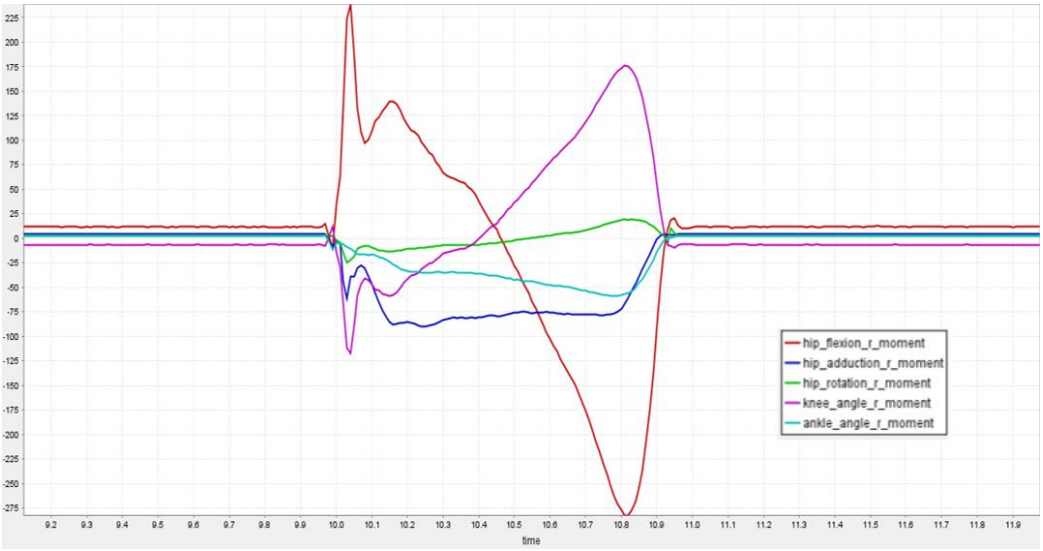


Figure 4.47. Right leg kinetic analysis results when subject is walking in Direction 2

The kinetic analysis results are self-consistent and also consistent with the results in the literature [46], [47]. However, two issues were detected when comparing with the literature results.

The first issue is that it has been noticed that some curves are inverse of the literature. Knee flexion-extension moment is an example of this issue, and it is noticed that this problem is caused by the flexion moment defined to increase progressively in the +y direction in OpenSim. However, it has been seen in the literature that this parameter is defined to decrease in the +y direction.

Secondly, it has been found in the literature that kinetic parameters are shared in a normalized manner. When obtained moments were normalized with the weight of subjects and compared with the literature data, it was observed that experimental results were higher than those in the literature. However, as a result of the comparison of the data of different subjects with different weights, it was observed that the data were consistent within themselves. Additional clinical studies may be required to calibrate the force platforms to another calibrated sensor.

4.4.2 Kinematic Analysis for Human Upper Extremity

Ethical approval for this study was obtained from Middle East Technical University Applied Ethics Research Center İAEK with protocol number 0393-ODTUİAEK-2022, which is presented in Appendix B.

In these tests, IMU sensors were attached to the female subject's upper extremity segments (torso, hands, right and left proximal and distal parts of arms) and 10 set shoulder flexion-extension, shoulder abduction-adduction, elbow flexion-extension, elbow pronation-supination data were collected with the help of IMU sensors. In these experiments, the right and left segments performed the same movement at the same time.

The processing of kinematic data is described in Chapter 3.

The subject flexed her shouders to the highest degree possible in the shoulder flexion experiment. Shoulder flexion-extension analysis results are presented in Figure 4.48.

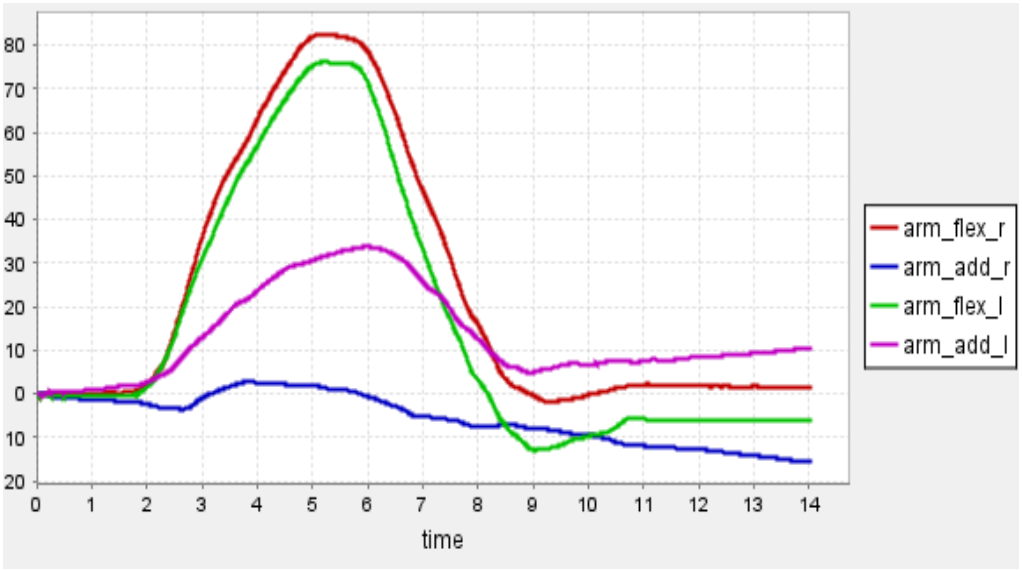


Figure 4.48. Shoulder flexion kinetic analysis results

The subject flexed her right and left elbows to the highest degree possible in the elbow flexion experiment. Elbow flexion analysis results are presented in Figure 4.49.

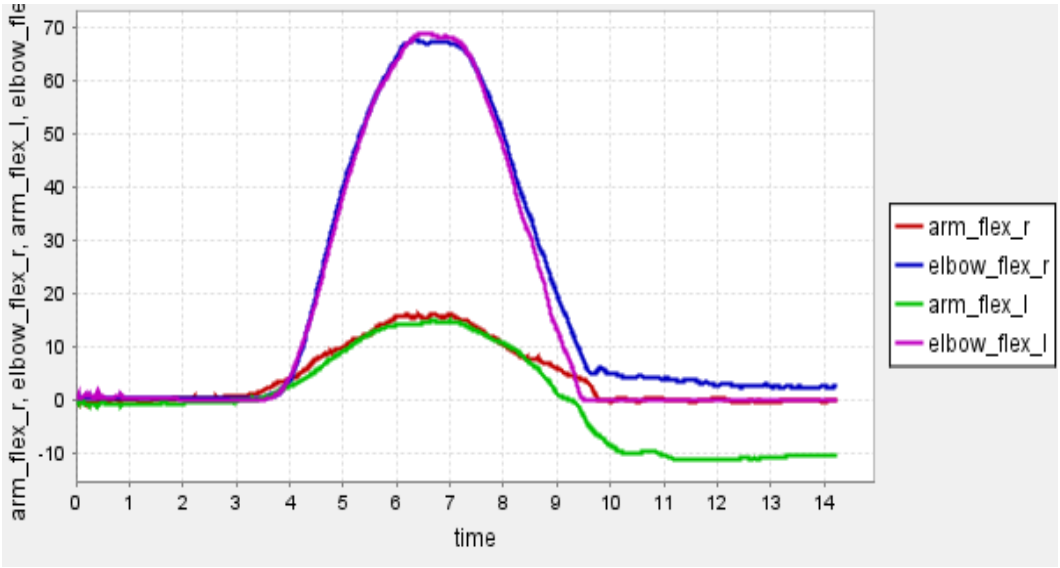


Figure 4.49. Elbow flexion kinetic analysis results

In the elbow pronation-supination experiment, subject pronated and then supinate left and right elbows simultaneously. The analysis results are presented in Figure 4.50.

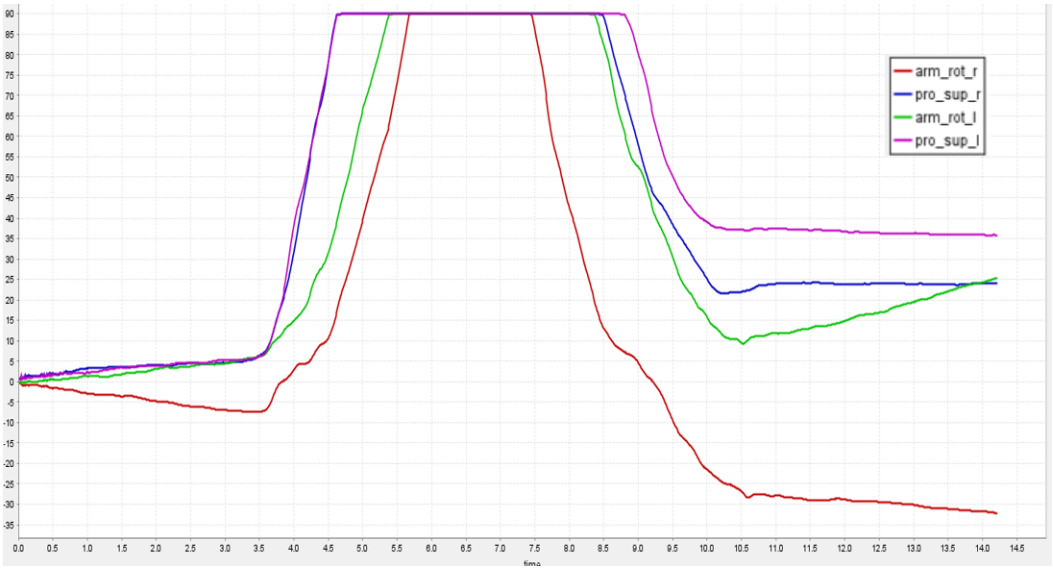


Figure 4.50. Elbow pronation-supination kinetic analysis results

As a result of the experiments, the expected movements in the joints were observed. However, unexpected movements were also observed. This situation can be clearly seen in Figure 4.50. It is thought that there are two main reasons for these unexpected motions.

The first reason is the sensor positions. Due to the subject anatomy, sensor positions can be slightly different from the sensor placements on the Rajagopal model. It is the first reason that unwanted movements to are observed in the joints.

The second reason is that Rajagopal model joints have anatomical limitations. If the joints try to exceed these limitations, the OpenSim program evaluates IMU data from related segments by using other IMU data and transfers this motion to other joints.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

This thesis deals with the development of an IMU based motion analysis system for upper and lower limbs for METU Biomechanics Laboratory. The study is organized into two parts. In the first part, the necessary system developments were carried out, then experiments were conducted to test the system in the second part.

After reviewing the literature, the first step was enhancing pre-written code to make the kinematic analysis system perform accurately with seven IMU sensors. The system is designed to perform both upper and lower limb kinematic analysis. After performing coordinate system transformations, sensor fusion is implemented for each IMU sensor with Madgwick's Algorithm [31]. Later, the resulting data are merged, and joint angles are computed using the Inverse Kinematics Tool of the OpenSim. Moreover, motion animation is also developed to observe the motion.

In the second part of the study, the performance of the system was studied with the help of a series of experiments. Primarily, individual sensors were tested, and it was detected that the problems were generally related to the coordinate system. Therefore the necessary coordinate system transformations were performed again. Furthermore, at this step, it was realized that it is more accurate to work with quaternions instead of Euler angles. The codes were modified according to this new approach.

Next, a mechanical test system was designed and manufactured for multi-sensor experiments. The kinematic analysis system was tested by collecting data from seven sensors simultaneously by using the mechanical test system. It was noticed that due to the interconnected design of the joints in the OpenSim model, the movement of

one joint could also affect other joints. Additionally, drift was detected at some of the kinematic analysis results, and this was corrected with slope correction code.

Thereafter, force plates are used for kinetic analysis. GRFs, moments, the points of application and the data generated through kinematic analysis are given as input to the OpenSim Inverse Dynamics Tool. Thus, the system was developed to calculate the joint reaction moments of the subject.

Finally, proof-of-concept experiments with a human subject were conducted. Ethics committee approval was obtained for these experiments. Kinematic and kinetic data (for the lower limb system) were captured from the subject during walking. The drifts found in the data were removed by using an add-on code. Also, shoulder flexion-extension, shoulder abduction-adduction, elbow flexion-extension, and elbow pronation-supination data were collected from the subject and analyzed kinematically. Overall, results of these analyzes were found to be consistent with the literature.

5.2 Future Work

Firstly, two sensor-specific problems should be considered in future studies. For IMU sensors, experiments should be carried out to verify the drift correction code and also other filters, different from the Butterworth filter, may be applied to solve the drift problem. As for the force platform, distortion was detected in the data from x-axis of the Force Plate 1. It is believed that a hardware-related problem causes this distortion, and therefore it should be investigated.

Secondly, even though experiments to verify the accuracy of the system have been performed, it is important and necessary to perform experiments to verify the sensitivity of the system. The sensitivity of the system is assumed to be low due to the low sensitivity of the LSM9DS1 sensors. Therefore, it might be necessary to upgrade the sensors. Additionally, it is recommended to conduct a controlled clinical

trial to compare the system with a commercially available system in terms of sensitivity.

Furthermore, the discrete operation of the codes complicates the use of the system. In terms of system integrity, it could be important to compile the data collection code, pre-processing code, Madgwick Algorithm and OpenSim codes, which will accelerate the operation of the system.

Finally, during data collection, several issues were encountered. These issues were determined as failure to record sensor data, incomplete recording, missing data, irregular data recording between IMU and force plate, and incorrect recording of sensor numbers if the IMUs are farther away from the receiver. Therefore, the system and codes related to data collection need to be enhanced with more sensitive sensors and a user-friendly interface. Additionally, the visualization of the sensor battery levels and the development of comfortable and practical connection tools for connecting the sensors to the user will increase the performance of the system.

REFERENCES

[1] M. W. Whittle, "Basic sciences," in *Gait Analysis:An Introduction*, 4th ed. U.K.: Butterworth-Heinemann, 2007, ch. 1, pp. 1-45.

[2] D. Cech and S. T. Martin, *Functional Movement Development Across the Life Span*, 3rd ed. Philadelphia, PA, USA: Saunders, 2011.

[3] T. P. Andriacchi and E. J. Alexander, "Studies of human locomotion: Past, present and future," in Journal of Biomechanics, 2000, doi: 10.1016/S0021-9290(00)00061-0.

[4] M. Akhtaruzzaman, A. A. Shafie, and M. R. Khan, "Gait analysis: Systems, technologies, and importance," Journal of Mechanics in Medicine and Biology. 2016, doi: 10.1142/S0219519416300039.

[5] A. Muro-de-la-Herran, B. García-Zapirain, and A. Méndez-Zorrilla, "Gait analysis methods: An overview of wearable and non-wearable systems, highlighting clinical applications," Sensors (Switzerland). 2014, doi: 10.3390/s140203362.

[6] H. Güler, "Biomechanical modeling of lower extremity and simulation of foot during gait," Ph.D. dissertation, Dept. Mech. Eng., Middle East Technical Univ., Ankara, Turkey, 1998.

[7] M. Shafiq, "Motion tracking in gait analysis," M.S. thesis, Dept. Mech. Eng., Middle East Technical Univ., Ankara, Turkey, 1998.

[8] Y. Karpat, "Development and testing of kinematic data acquisition tools for a gait analysis system," M.S. thesis, Dept. Mech. Eng., Middle East Technical Univ., Ankara, Turkey, 2000.

[9] H. Afşar, "Evaluation and compensation of soft tissue movement artefacts for the kiss gait analysis system," M.S. thesis, Dept. Mech. Eng., Middle East Technical Univ., Ankara, Turkey, 2001.

[10] B. Söylemez, "An investigation on the gait analysis protocol of the "kiss" motion analysis system," M.S. thesis, Dept. Mech. Eng., Middle East Technical Univ., Ankara, Turkey, 2002.

[11] E. Civek, "Comparison of kinematic results between metu-kiss & ankarauniversity-vicon gait analysis systems," M.S. thesis, Dept. Mech. Eng., Middle East Technical Univ., Ankara, Turkey, 2006.

[12] P. Kafalı, "Evaluation of sensitivity of metu gait analysis system," M.S. thesis, Dept. Mech. Eng., Middle East Technical Univ., Ankara, Turkey, 2007.

[13] K. Erer, "Verification and matlab implementation of the inverse dynamics model of the metu gait analysis system," M.S. thesis, Dept. Mech. Eng., Middle East Technical Univ., Ankara, Turkey, 2008.

[14] M. Biçer, " On the implementation of Opensim: Applications of marker-based and inertial measurement unit based systems," M.S. thesis, Dept. Mech. Eng., Middle East Technical Univ., Ankara, Turkey, 2019.

[15] J. K. Aggarwal and Q. Cai, “Human Motion Analysis: A Review,” Comput. Vis. Image Underst., 1999, doi: 10.1006/cviu.1998.0744.

[16] M. Rana and V. Mittal, “Wearable Sensors for Real-Time Kinematics Analysis in Sports: A Review,” IEEE Sens. J., 2021, doi: 10.1109/JSEN.2020.3019016.

[17] B. Kalita, J. Narayan, and S. K. Dwivedy, “Development of Active Lower Limb Robotic-Based Orthosis and Exoskeleton Devices: A Systematic Review,” International Journal of Social Robotics. 2021, doi: 10.1007/s12369-020-00662-9.

[18] G. X. Lee and K. S. Low, “A factorized quaternion approach to determine the arm motions using triaxial accelerometers with anatomical and sensor constraints,” IEEE Trans. Instrum. Meas., 2012, doi: 10.1109/TIM.2011.2181884.

[19] R. Haddas and K. L. Ju, “Gait Alteration in Cervical Spondylotic Myelopathy Elucidated by Ground Reaction Forces,” Spine (Phila. Pa. 1976)., 2019, doi: 10.1097/BRS.0000000000002732.

[20] D. Sethi, S. Bharti, and C. Prakash, “A comprehensive survey on gait analysis: History, parameters, approaches, pose estimation, and future work,” Artificial Intelligence in Medicine. 2022, doi: 10.1016/j.artmed.2022.102314.

[21] H. Zhou and H. Hu, “Human motion tracking for rehabilitation-A survey,” Biomedical Signal Processing and Control. 2008, doi: 10.1016/j.bspc.2007.09.001.

[22] D. T. P. Fong and Y. Y. Chan, “The use of wearable inertial motion sensors in human lower limb biomechanics studies: A systematic review,” Sensors (Switzerland). 2010, doi: 10.3390/s101211556.

[23] Y. Fan, Y. Fan, Z. Li, C. Lv, and D. Luo, “Natural gaits of the non-pathological flat foot and high-arched foot,” PLoS One, 2011, doi: 10.1371/journal.pone.0017749.

[24] M. Iosa, P. Picerno, S. Paolucci, and G. Morone, “Wearable inertial sensors for human movement analysis,” Expert Review of Medical Devices. 2016, doi: 10.1080/17434440.2016.1198694.

[25] C. Bonnyaud, D. Pradon, R. Zory, D. Bensmail, N. Vuillerme, and N. Roche, “Does a single gait training session performed either overground or on a treadmill induce specific short-term effects on gait parameters in patients with hemiparesis? a randomized controlled study,” Top. Stroke Rehabil., 2013, doi: 10.1310/tsr2006-509.

[26] G. Vasco et al., “Functional and gait assessment in children and adolescents affected by Friedreich’s ataxia: A one-year longitudinal study,” PLoS One, 2016, doi: 10.1371/journal.pone.0162463.

[27] A. P. Shortland, “Gait and clinical gait analysis,” in Clinical Engineering, 2020.

[28] W. Sun, J. Wu, W. Ding, and S. Duan, “A robust indirect kalman filter based on the gradient descent algorithm for attitude estimation during dynamic conditions,” IEEE Access, 2020, doi: 10.1109/ACCESS.2020.2997250.

[29] J. Diebel, “Representing attitude: Euler angles, unit quaternions, and rotation vectors,” Matrix, 2006.

[30] R. Mahony, T. Hamel, and J. M. Pflimlin, “Nonlinear complementary filters on the special orthogonal group,” IEEE Trans. Automat. Contr., 2008, doi: 10.1109/TAC.2008.923738.

[31] S. O. H. Madgwick, “An efficient orientation filter for inertial and inertial/magnetic sensor arrays,” Rep. x-io Univ. …, 2010.

[32] “Mathematical Model of an IMU,” 2022. [Online]. Available: https://nitinjsanket.github.io/tutorials/attitudeest/madgwick. [Accessed: July 28, 2022].

[33] “Coordinate Systems,” 2017. [Online]. Available: https://simtk-confluence.stanford.edu:8443/display/OpenSim/Coordinate+Systems. [Accessed: July 28, 2022].

[34] ST, “iNEMO inertial module: 3D accelerometer, 3D gyroscope, 3D magnetometer,” LSM9DS1 datasheet, Mar. 2015 [Rev 3].

[35] M. K. Özgören. (2020). Advanced dynamics, Middle East Technical University [PDF Document].

[36] M. Kok, J. D. Hol, T. B. Schön, F. Gustafsson, and H. Luinge, “Calibration of a magnetometer in combination with inertial sensors,” in 15th International Conference on Information Fusion, FUSION 2012, 2012.

[37] M. Kok and T. B. Schon, “Magnetometer calibration using inertial sensors,” IEEE Sens. J., 2016, doi: 10.1109/JSEN.2016.2569160.

[38] R. Takeda, G. Lisco, T. Fujisawa, L. Gastaldi, H. Tohyama, and S. Tadano, “Drift removal for improving the accuracy of gait parameters using wearable sensor systems,” Sensors (Switzerland), 2014, doi: 10.3390/s141223230.

[39] S. Majumder and M. Jamal Deen, “Wearable IMU-Based System for Real-Time Monitoring of Lower-Limb Joints,” IEEE Sens. J., 2021, doi: 10.1109/JSEN.2020.3044800.

[40] T. McGrath, R. Fineman, and L. Stirling, “An auto-calibrating knee flexion-extension axis estimator using principal component analysis with inertial sensors,” Sensors (Switzerland), 2018, doi: 10.3390/s18061882.

[41] M. V. McCabe, D. W. Van Citters, and R. M. Chapman, “Developing a method for quantifying hip joint angles and moments during walking using neural networks and wearables,” Comput. Methods Biomech. Biomed. Engin., 2022, doi: 10.1080/10255842.2022.2044028.

[42] “Adafruit LSM9DS1 Accelerometer + Gyro + Magnetometer 9-DOF Breakout,” 2017. [Online]. Available: https://learn.adafruit.com/adafruit-lsm9ds1-accelerometer-plus-gyro-plus-magnetometer-9-dof-breakout. [Accessed: July 28, 2022].

[43] A. Rajagopal, C. L. Dembia, M. S. DeMers, D. D. Delp, J. L. Hicks, and S. L. Delp, “Full-Body Musculoskeletal Model for Muscle-Driven Simulation of Human Gait,” IEEE Trans. Biomed. Eng., 2016, doi: 10.1109/TBME.2016.2586891.

[44] C. C. Gordon, T. Churchill, C. E. Clauser, J. T. Mcconville, I. Tebbetts, and R. A. Walker, “1988 Anthropometric Survey of U. S. Army Personnel: Methods and Summary Statistics,” Security, 1988.

[45] “The High-Performance Acetal Resin,” 2022. [Online]. Available: https://www.dupont.com/brands/delrin.html#:~:text=Delrin%C2%AE%20acetal%20homopolymer%20(Polyoxymethylene,%C2%B0C)%20and%20good%20colorability. [Accessed: July 28, 2022].

[46] C. A. Fukuchi, R. K. Fukuchi, and M. Duarte, “A public dataset of overground and treadmill walking kinematics and kinetics in healthy individuals,” PeerJ, 2018, doi: 10.7717/peerj.4640.

[47] G. Bovi, M. Rabuffetti, P. Mazzoleni, and M. Ferrarin, “A multiple-task gait analysis approach: Kinematic, kinetic and EMG reference data for healthy young and adult subjects,” Gait Posture, 2011, doi: 10.1016/j.gaitpost.2010.08.009.

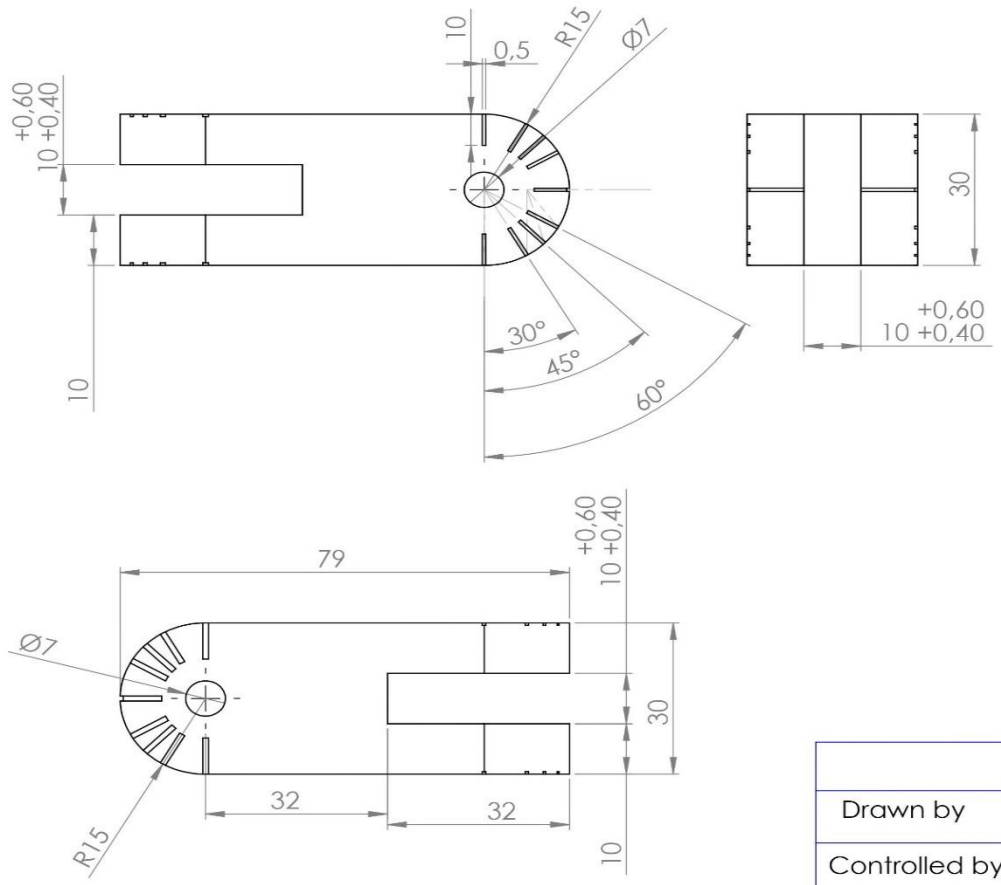
[48] “Musculoskeletal Models,” 2017. [Online]. Available: https://simtk-confluence.stanford.edu:8443/display/OpenSim/Musculoskeletal+Models#:~:text=OpenSim%20Core%20Models,-Models%20included%20with&text=Simulating%20and%20analyzing%20human%20movement,both%20kinematics%20and%20dynamics%20analyses.&text=Primarily%20lower%20extremity%20model%20with%20two%20legs%20and%20a%20lumped%20torso%20segment.,-Includes%2023%20degrees. [Accessed: July 28, 2022].

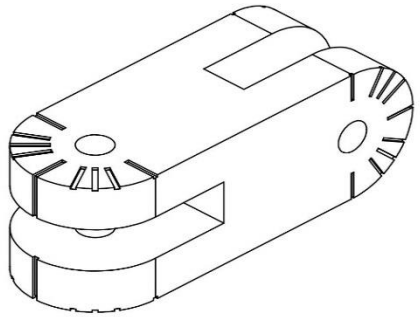
[49] S. Winiarski and A. Rutkowska-Kucharska, “Estimated ground reaction force in normal and pathological gait,” Acta Bioeng. Biomech., 2009.

APPENDICES

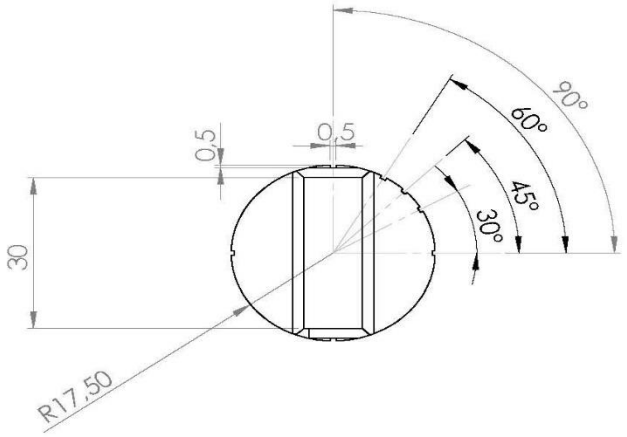
A. Technical Drawings of Mechanical System

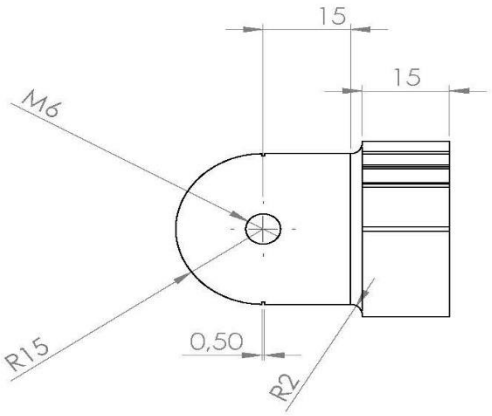
Technical Drawings of the mechanical test equipment are presented as.

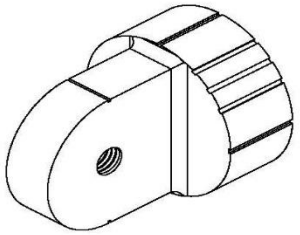


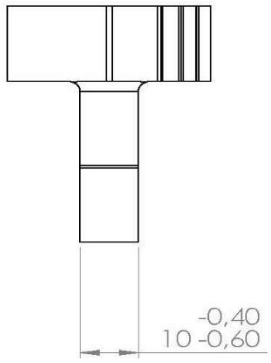


	Name Surname	Date	Part 1	
Drawn by	Selin Kirdiş	18.09.21		
Controlled by	Ergin Tönük		Unit	Piece
Material	Delrin		mm	6

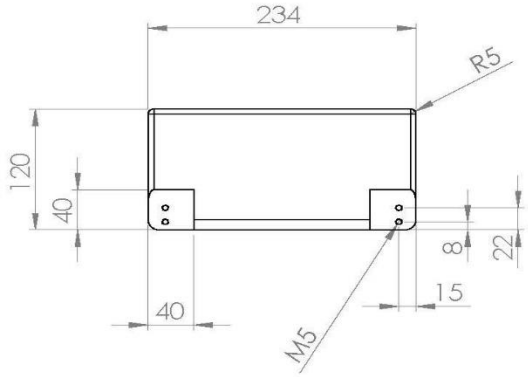


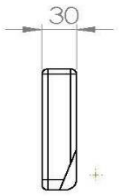


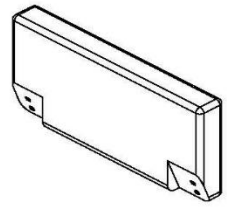


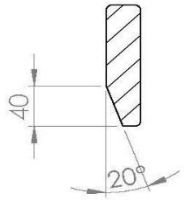


	Name Surname	Date	Part 2	
Drawn by	Selin Kirdiş	18.09.21		
Controlled by	Ergin Tönük		Unit	Piece
Material	Delrin		mm	12

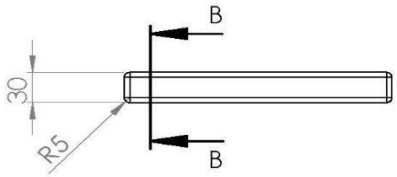




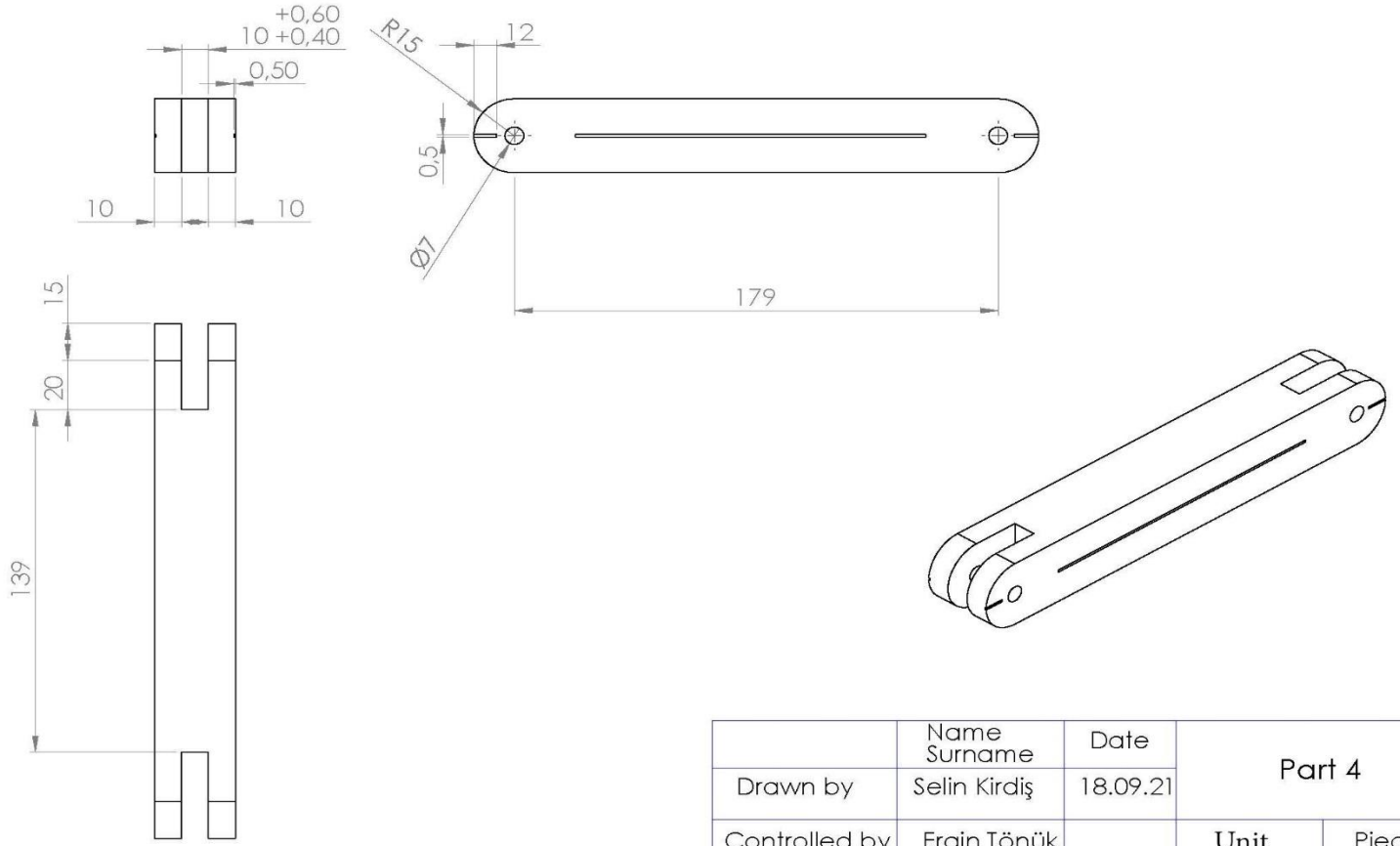




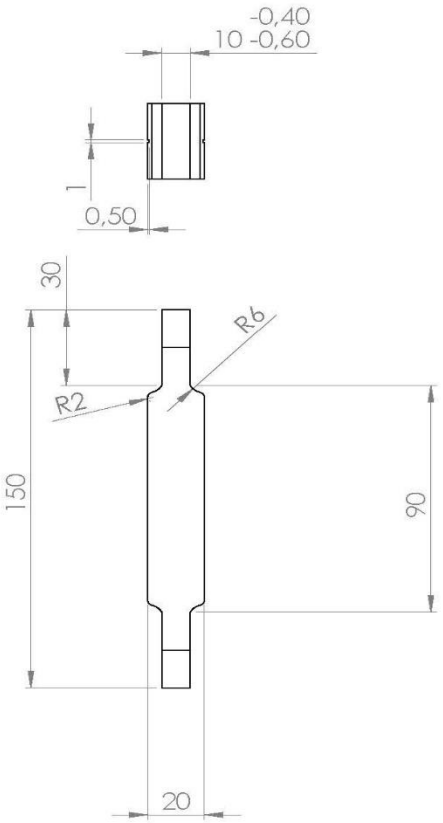
SECTION B-B

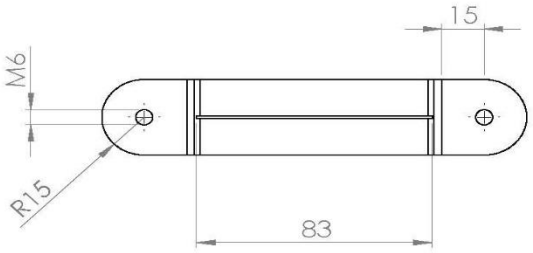


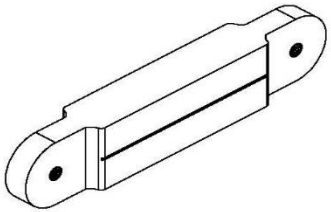
	Name Surname	Date	Part 3	
Drawn by	Selin Kirdiş	18.09.21		
Controlled by	Ergin Tönük		Unit	Piece
Material	Delrin		mm	1



	Name Surname	Date	Part 4	
Drawn by	Selin Kirdiş	18.09.21		
Controlled by	Ergin Tönük		Unit	Piece
Material	Delrin		mm	2







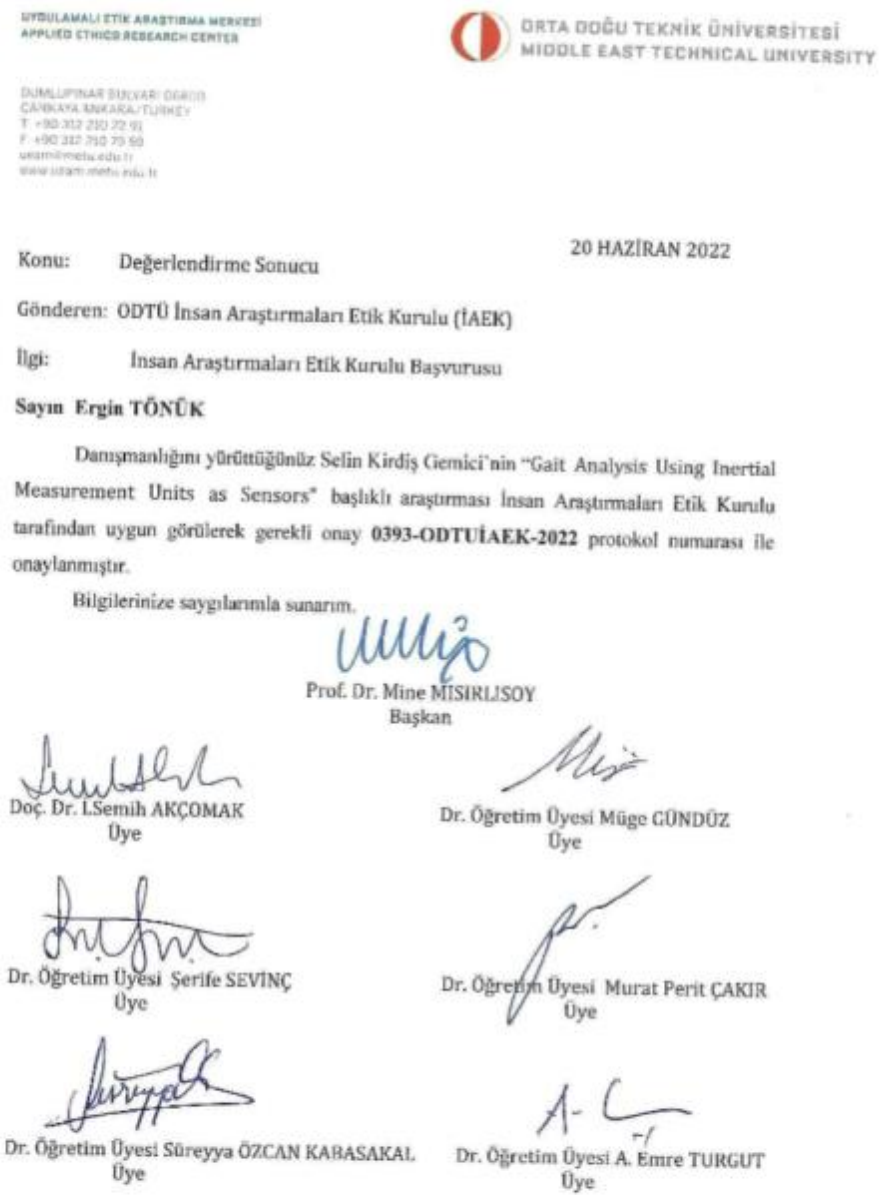
	Name Surname	Date	Part 5	
Drawn by	Selin Kirdiş	18.09.21		
Controlled by	Ergin Tönük		Unit	Piece
Material	Delrin		mm	2

Part 6
Name Surname
Date
Drawn by
Selin Kirdiş
18.09.21
Controlled by
Ergin Tönük
Unit
Piece
Material
Delrin
mm.
2
60
20
120
R10
40
15
20
M5

	Name Surname	Date	Part 7	
Drawn by	Selin Kirdiş	18.09.21		
Controlled by	Ergin Tönük		Unit	Piece
Material	Delrin		mm	2

	Name Surname	Date	Part 8		
Drawn by	Selin Kirdiş	18.09.21		Unit	Piece
Controlled by	Ergin Tönük			mm	2
Material	Delrin				

B. Ethical Approval



UYGULAMALI ETİK ARAŞTIRMA MERKEZİ
APPLIED ETHICS RESEARCH CENTER

ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

DUMLUPINAR BULVARI 06800
ÇANKAYA ANKARA/TURKEY
T +90 312 210 22 91
F +90 312 210 79 59
uygulamalietik@metu.edu.tr
www.uygulamalietik.metu.edu.tr

20 HAZİRAN 2022

Konu: Değerlendirme Sonucu

Gönderen: ODTÜ İnsan Araştırmaları Etik Kurulu (İAEK)

İlgi: İnsan Araştırmaları Etik Kurulu Başvurusu

Sayın Ergin TÖNÜK

Danışmanlığını yürüttüğünüz Selin Kirdiş Gemici'nin "Gait Analysis Using Inertial Measurement Units as Sensors" başlıklı araştırması İnsan Araştırmaları Etik Kurulu tarafından uygun görülerek gerekli onay **0393-ODTUİAEK-2022** protokol numarası ile onaylanmıştır.

Bilgilerinize saygılarımla sunarım.

Prof. Dr. Mine MISIRLISOY
Başkan

Doç. Dr. İ.Semih AKÇOMAK
Üye

Dr. Öğretim Üyesi Müge GÜNDÜZ
Üye

Dr. Öğretim Üyesi Şerife SEVİNÇ
Üye

Dr. Öğretim Üyesi Murat Perit ÇAKIR
Üye

Dr. Öğretim Üyesi Süreyya ÖZCAN KABASAKAL
Üye

Dr. Öğretim Üyesi A. Emre TURGUT
Üye

C. Preprocessing Matlab Codes

- Preprocessing Data Code 1

```
clc;
clear;
dataPath ='C:\Users\selin\Desktop\DENEY\1.txt';
B = dlmread(dataPath);
numbodies = 7;
numdata = length(B);
numframe = numdata/numbodies;

m = 1;
b = 1;
v=1;
y=1;
f=1;
g=1;
e=1;
a=1;
c=1;

b2=zeros(300,12);
b3=zeros(300,12);
b4=zeros(300,12);
b5=zeros(300,12);
b6=zeros(300,12);
b7=zeros(300,12);
b8=zeros(300,12);

%% group data according to sensor numbers

  for i = 1:length(B)
    switch B(i,1)
        case 2
          b2(a,1:end) = B(i,1:end);
          a=a+1;
        case 3
          b3(v,1:end) = B(i,1:end);
          v=v+1;
         case 4
          b4(y,1:end) = B(i,1:end);
          y=y+1;
          case 5
          b5(f,1:end) = B(i,1:end);
          f=f+1;
          case 6
          b6(g,1:end) = B(i,1:end);
          g=g+1;
          case 7
          b7(e,1:end) = B(i,1:end);
          e=e+1;
```

```
          case 8
          b8(c,1:end) = B(i,1:end);
          c=c+1;
    end
  end

X= [b2; b3; b4; b5; b6; b7; b8];

%% write grouped data

fid = fopen('C:\Users\selin\Desktop\DENEY\2.txt','w');
for i=1:numdata
           fprintf(fid,'%.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t
%.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\n ',X(i,1:end));
end
        fclose(fid);
```

- Preprocessing Data Code 2

```
clc;
clear;
dataPath ='C:\Users\selin\Desktop\DENEY\3.txt';
B = dlmread(dataPath);
numbodies = 7;
numdata = length(B);
numframe= numdata/numbodies;

% find mean of accelerometer-magnetometer data by using first 100
data

m = 1;
b = 1;

for z = 1:7

  for a = 6:11

    C(z,(a-5)) = fix(mean(B(m:m+99,a)));

  end
m = b*numframe+1;
b = b+1
end

D(1,:) = [[2 0 0 0 0], C(1,:), 0];
D(2,:) = [[3 0 0 0 0], C(2,:), 0];
D(3,:) = [[4 0 0 0 0], C(3,:), 0];
D(4,:) = [[5 0 0 0 0], C(4,:), 0];
```

```
D(5,:) = [[6 0 0 0 0], C(5,:), 0];
D(6,:) = [[7 0 0 0 0], C(6,:), 0];
D(7,:) = [[8 0 0 0 0], C(7,:), 0];


for i=1:5000
    E(i,:) = D(1,:);
    F(i,:)= D(2,:);
    G(i,:) = D(3,:);
    H(i,:) = D(4,:);
    I(i,:) = D(5,:);
    S(i,:) = D(6,:);
    J(i,:) = D(7,:);
end

%rewrite data with first 5000 data are the mean of first 100 data
point

line_index =100;

fid = fopen('C:\Users\selin\Desktop\DENEY\4.txt','w+','n','UTF-8');

         fseek(fid,0,-1);

for z = 1:5000

fprintf(fid,'%.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t
%.0f\t %.0f\t %.0f\t %.0f\t %.0f\n',E(i,1),
E(i,2),E(i,3),E(i,4),E(i,5),E(i,6),E(i,7),E(i,8),E(i,9),E(i,10),E(i
,11),E(i,12));

end

for k=1: numframe

A = B(k,:);
fprintf(fid,'%.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t
%.0f\t %.0f\t %.0f\t %.0f\t %.0f\n',A);

end

%%

for z = 1:5000

fprintf(fid,'%.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t
%.0f\t %.0f\t %.0f\t %.0f\t
%.0f\n',F(i,1),F(i,2),F(i,3),F(i,4),F(i,5),F(i,6),F(i,7),F(i,8),F(i
,9),F(i,10),F(i,11),F(i,12));

end
```

```
for k=1: numframe

A = B((k+numframe),:);
fprintf(fid,'%.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t
%.0f\t %.0f\t %.0f\t %.0f\t %.0f\n',A);

end

%%

for z = 1:5000

fprintf(fid,'%.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t
%.0f\t %.0f\t %.0f\t %.0f\t
%.0f\n',G(i,1),G(i,2),G(i,3),G(i,4),G(i,5),G(i,6),G(i,7),G(i,8),G(i
,9),G(i,10),G(i,11),G(i,12));

end

for k=1: numframe

A = B((k+(numframe*2)),:);
fprintf(fid,'%.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t
%.0f\t %.0f\t %.0f\t %.0f\t %.0f\n',A);

end

%%

for z = 1:5000

fprintf(fid,'%.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t
%.0f\t %.0f\t %.0f\t %.0f\t
%.0f\n',H(i,1),H(i,2),H(i,3),H(i,4),H(i,5),H(i,6),H(i,7),H(i,8),H(i
,9),H(i,10),H(i,11),H(i,12));

end

for k=1: numframe

A = B((k+(numframe*3)),:);
fprintf(fid,'%.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t
%.0f\t %.0f\t %.0f\t %.0f\t %.0f\n',A);

end

%%

for z = 1:5000

fprintf(fid,'%.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t
%.0f\t %.0f\t %.0f\t %.0f\t
```

```
%.0f\n',I(i,1),I(i,2),I(i,3),I(i,4),I(i,5),I(i,6),I(i,7),I(i,8),I(i
,9),I(i,10),I(i,11),I(i,12));
end

for k=1: numframe

A = B((k+(numframe*4)),:);
fprintf(fid,'%.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t
%.0f\t %.0f\t %.0f\t %.0f\t %.0f\n',A);

end

%%

for z = 1:5000

fprintf(fid,'%.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t
%.0f\t %.0f\t %.0f\t %.0f\t
%.0f\n',S(i,1),S(i,2),S(i,3),S(i,4),S(i,5),S(i,6),S(i,7),S(i,8),S(i
,9),S(i,10),S(i,11),S(i,12));

end

for k=1: numframe

A = B((k+(numframe*5)),:);
fprintf(fid,'%.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t
%.0f\t %.0f\t %.0f\t %.0f\t %.0f\n',A);

end

%%

for z = 1:5000

fprintf(fid,'%.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t
%.0f\t %.0f\t %.0f\t %.0f\t
%.0f\n',J(i,1),J(i,2),J(i,3),J(i,4),J(i,5),J(i,6),J(i,7),J(i,8),J(i
,9),J(i,10),J(i,11),J(i,12));

end

for k=1:numframe

A = B((k+numframe*6),:);
fprintf(fid,'%.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t %.0f\t
%.0f\t %.0f\t %.0f\t %.0f\t %.0f\n',A);

end

fprintf(fid,'\r\n');
fclose(fid);
```

D. Lower Extremity Kinematic Analysis Codes with Quaternions

- TestCpp

```
function C = TestCpp(dataPath)
 %#codegen
fprintf('#App starting..\n\n');
rootpath = '';
%%
if  isdeployed()
    if (nargin<1)
        dataPath='';
        msgbox('Input datapath is empty/null');
    end
else

dataPath ='C:\Users\selin\Desktop\DENEY\1111.txt';
    rootpath = 'C:\Users\selin\Desktop\';
end

if isempty(dataPath)== false
    fprintf('loading file from %s\n',dataPath);
    B = dlmread(dataPath);
    B = B(:,1:end-1);
    numbodies = 7;
    numdata = length(B);

if numdata>=numbodies
        numframe = numdata/numbodies;
        diff = numframe - 5000;
        [C] = Matrices7(B,numbodies,numdata,numframe);
        pelvis  =  (C.pelvis);
        femurR =  (C.femurR);
        femurL =  (C.femurL);
        tibiaR  =  (C.tibiaR);
        footR  =  (C.footR);
        footL  =  (C.footL);
        tibiaL =  (C.tibiaL);


        % Write to file
         fid = fopen('mlab.sto','w+','n','UTF-8');
         fseek(fid,0,-1);

       fprintf(fid,'\r\nDataRate=100.000000');
       fprintf(fid,'\r\nDataType=Quaternion');
       fprintf(fid,'\r\nversion=3');
       fprintf(fid,'\r\nOpenSimVersion=4.3');
       fprintf(fid,'\r\nendheader');

       fprintf(fid,'\r\ntime ');
       fprintf(fid,'\t pelvis_imu');
```

```
        fprintf(fid,'\t tibia_r_imu');
        fprintf(fid,'\t femur_r_imu');
        fprintf(fid,'\t calcn_r_imu');
        fprintf(fid,'\t calcn_l_imu');
        fprintf(fid,'\t tibia_l_imu');
        fprintf(fid,'\t femur_l_imu');
        line_index =1;

         for k=5000: numframe

            fprintf(fid,'\r\n%.2f\t',(line_index/100.0));

            lpelvis1 = pelvis(k,1);
            lpelvis2 = pelvis(k,2);
            lpelvis3 = pelvis(k,3);
            lpelvis4 = pelvis(k,4);
            fprintf(fid,'%.6f,%.6f,%.6f,%.6f\t', lpelvis1, lpelvis2,
lpelvis3, lpelvis4);

            ltibiar1 = tibiaR(k,1);
            ltibiar2 = tibiaR(k,2);
            ltibiar3 = tibiaR(k,3);
            ltibiar4 = tibiaR(k,4);
            fprintf(fid,'%.6f,%.6f,%.6f,%.6f\t', ltibiar1, ltibiar2,
ltibiar3, ltibiar4);

            lfemurr1 = femurR(k,1);
            lfemurr2 = femurR(k,2);
            lfemurr3 = femurR(k,3);
            lfemurr4 = femurR(k,4);
            fprintf(fid,'%.6f,%.6f,%.6f,%.6f\t', lfemurr1, lfemurr2,
lfemurr3, lfemurr4);

            lfootr1 = footR(k,1);
            lfootr2 = footR(k,2);
            lfootr3 = footR(k,3);
            lfootr4 = footR(k,4);
            fprintf(fid,'%.6f,%.6f,%.6f,%.6f\t', lfootr1, lfootr2,
lfootr3, lfootr4);

            lfootl1 = footL(k,1);
            lfootl2 = footL(k,2);
            lfootl3 = footL(k,3);
            lfootl4 = footL(k,4);
            fprintf(fid,'%.6f,%.6f,%.6f,%.6f\t', lfootl1, lfootl2,
lfootl3, lfootl4);

            ltibial1 = tibiaL(k,1);
            ltibial2 = tibiaL(k,2);
            ltibial3 = tibiaL(k,3);
            ltibial4 = tibiaL(k,4);
            fprintf(fid,'%.6f,%.6f,%.6f,%.6f\t', ltibial1, ltibial2,
ltibial3, ltibial4);
```

```
            lfemurl1 = femurL(k,1);
            lfemurl2 = femurL(k,2);
            lfemurl3 = femurL(k,3);
            lfemurl4 = femurL(k,4);
            fprintf(fid,'%.6f,%.6f,%.6f,%.6f', lfemurl1, lfemurl2,
lfemurl3, lfemurl4);

             line_index = line_index+1;
             end
        end

        fprintf(fid,'\r\n');
        fclose(fid);

end

%%
clear all;
close all;
clc;
import org.opensim.modeling.*

%% OpenSim variables
modelFileName = 'Rajagopal_2015.osim';
orientationsFileName = 'mlab.sto';
sensor_to_opensim_rotations = Vec3(-pi/2, 0, 0);
baseIMUName = 'pelvis_imu';
baseIMUHeading = 'x';
visulizeCalibration = true;

%% IMUPlacer tool
imuPlacer = IMUPlacer();

% IMUPlacer properties
imuPlacer.set_model_file(modelFileName);
imuPlacer.set_orientation_file_for_calibration(orientationsFileName
);
imuPlacer.set_sensor_to_opensim_rotations(sensor_to_opensim_rotatio
ns);
imuPlacer.set_base_imu_label(baseIMUName);
imuPlacer.set_base_heading_axis(baseIMUHeading);

imuPlacer.run(visulizeCalibration);

model = imuPlacer.getCalibratedModel();

%% calibrated model
model.print( strrep(modelFileName, '.osim', '_calibrated.osim') );

%% Clear the Workspace variables.
clear all; close all; clc;
import org.opensim.modeling.*
```

```
%% OpenSim variables
modelFileName = 'Rajagopal_2015_calibrated.osim';
orientationsFileName = 'mlab.sto';
sensor_to_opensim_rotation = Vec3(-pi/2, 0, 0);
visualizeTracking = true;
startTime = 0;
endTime = 40;
resultsDirectory = 'IKResults';

%% InverseKinematicsTool
imuIK = IMUInverseKinematicsTool();

%% tracking
imuIK.set_model_file(modelFileName);
imuIK.set_orientations_file(orientationsFileName);
imuIK.set_sensor_to_opensim_rotations(sensor_to_opensim_rotation)

imuIK.set_time_range(0, startTime);
imuIK.set_time_range(1, endTime);

imuIK.set_results_directory(resultsDirectory)

% Run IK

imuIK.run(visualizeTracking);
fprintf('#Finished\n');
end
```

- Matrices

```
function [C] = Matrices7(A,numbodies,numdata,numframe)
rt = 1;rf = 1;p = 1;lf = 1;lt = 1; rfo=1; lfo=1;
B = A(1:numdata,:);
numcolB = size(B,2);
imufield1 = 'val';
imufield2 = 'bodynames';
nul1 = cell(1,numbodies);
nul2 = cell(1,numbodies);

for i = 1:numbodies
    nul1{i} = zeros(numdata,numcolB-1);
    nul2{i} = 'al';
end

imu = struct(imufield1,nul1,imufield2,nul2);
for i = 1:length(B)
    switch B(i,1)
        case 2
            imu(6).val(rfo,:) = B(i,2:end); %right foot
            imu(6).bodynames = 'rfo';
```

```
            rfo = rfo+1;
        case 3
            imu(1).val(rt,:) = B(i,2:end); %right tibia
            imu(1).bodynames = 'rt';
            rt = rt+1;
        case 4
            imu(2).val(rf,:) = B(i,2:end); %right femur
            imu(2).bodynames = 'rf';
            rf = rf+1;
        case 5
            imu(3).val(p,:) = B(i,2:end); %pelvis
            imu(3).bodynames = 'pe';
            p = p+1;
        case 6
            imu(4).val(lf,:) = B(i,2:end); %left femur
            imu(4).bodynames = 'lf';
            lf = lf+1;
        case 7
            imu(5).val(lt,:) = B(i,2:end); %left tibia
            imu(5).bodynames = 'lt';
            lt = lt+1;
        case 8
            imu(7).val(lfo,:) = B(i,2:end); %left foot
            imu(7).bodynames = 'lf0';
            lfo = lfo+1;
    end
end

[C.pelvis] = Calculations(imu(3).val(1:numframe,:));
[C.footR] = Calculations4(imu(6).val(1:numframe,:));
[C.footL] = Calculations4(imu(7).val(1:numframe,:));
[C.femurR] = Calculations2(imu(2).val(1:numframe,:));
[C.femurL] = Calculations3(imu(4).val(1:numframe,:));
[C.tibiaR] = Calculations2(imu(1).val(1:numframe,:));
[C.tibiaL] = Calculations3(imu(5).val(1:numframe,:));

trc_frame_no = size(C.pelvis,4);
num_frame = trc_frame_no-5000;
```

- Calculations

```
function out = Calculations(B)%#codegen
A = B;
frq = 100;
time = (0:1/frq:(length(A(:,1))-1)/frq)';
step = length(time);
a = 5; %in sensor text document acceleration columns are 5,6 and 7
gy = 2; %in sensor text document angular velocity columns are 2,3
and 4
m = 8; %in sensor text document magnetic measurement columns are
8,9 and 10
```

```
%%
acc_mult = 16340;
gyro_mult = 1000/8.75;
magno_mult = 1000/0.29;

A(:,a) =  -B(:,a+2)/acc_mult;%acc z
A(:,a+1) = -B(:,a)/acc_mult;%acc x
A(:,a+2) = -B(:,a+1)/acc_mult;% acc y


A(:,gy) = -B(:,gy+2)/gyro_mult; %gyro z
A(:,gy+1) =  -B(:,gy)/gyro_mult; %gyro x
A(:,gy+2) = -B(:,gy+1)/gyro_mult; %gyro y


A(:,m) = -B(:,m+2)/magno_mult; %magno z
A(:,m+1) =  B(:,m)/magno_mult; %magno x
A(:,m+2) = -B(:,m+1)/magno_mult; %magno y
%%
dur = 1:step;
Accelerometer = [A(dur,a) A(dur,a+1) A(dur,a+2)];
Gyroscope = [A(dur,gy) A(dur,gy+1) A(dur,gy+2)];
Magnetometer = [A(dur,8) A(dur,9) A(dur,10)];
Filtered_GYR_end = myFilt(Gyroscope(5001:step,1:3));
Filtered_ACC_end = myFilt(Accelerometer(5001:step,1:3));
Filtered_MAG_end = myFilt(Magnetometer(5001:step,1:3));
Filtered_GYR = [Gyroscope(1:5000,:);Filtered_GYR_end];
Filtered_ACC = [Accelerometer(1:5000,:);Filtered_ACC_end];
Filtered_MAG = [Magnetometer(1:5000,:);Filtered_MAG_end];
beta = 0.05;
AHRS = AHRSAlgorithm('SamplePeriod', 1/100, 'Beta', beta);
quaternion1 = zeros(length(time), 4);
Cse = zeros(3,3,length(time));
av = 5000;
quaternionst = zeros(av, 4);

for t = 1:av
    AHRS.Update(Filtered_GYR(t,:)*pi/180, Filtered_ACC(t,:),
Filtered_MAG(t,:));    % gyroscope units must be radians
    quaternionst(t, :) = AHRS.Quaternion;
    Cse(:,:,t) = q2mat(quaternionst(t,:));
end

q0 = mean(quaternionst(av-10:av,:));
q1=quaternConj(q0);

AHRS2 = AHRSAlgorithm('SamplePeriod', 1/100, 'Beta',
beta,'Quaternion',q0);

for t = 1:length(time)
    AHRS2.Update(Filtered_GYR(t,:)*pi/180, Filtered_ACC(t,:),
Filtered_MAG(t,:));   % gyroscope units must be radian/s
    quaternion1(t, :) = AHRS2.Quaternion;
```

```
q2(t, :)=quaternConj(quaternion1(t, :));
quaternion2(t, :)=quaternProd(q2(t, :),q0);
quaternion2(t, 4)=-quaternion2(t, 4);

end

out=(quaternion2);
```

- Calculations2

```
function out = Calculations2(B)%#codegen
A = B;
frq = 100;
time = (0:1/frq:(length(A(:,1))-1)/frq)';
step = length(time);
a = 5;
gy = 2;
m = 8;

%%
acc_mult = 16340;
gyro_mult = 1000/8.75;
magno_mult = 1000/0.29;

A(:,a) = -B(:,a)/acc_mult;%acc x
A(:,a+1) = B(:,a+2)/acc_mult;%acc z
A(:,a+2) = -B(:,a+1)/acc_mult; %acc y

A(:,gy) = -B(:,gy)/gyro_mult; %gyro x
A(:,gy+1) = B(:,gy+2)/gyro_mult; %gyro z
A(:,gy+2) = -B(:,gy+1)/gyro_mult; %gyro y

A(:,m) = B(:,m)/magno_mult; %magno x
A(:,m+1) = B(:,m+2)/magno_mult; %magno z
A(:,m+2) = -B(:,m+1)/magno_mult;  %magno y

%%

dur = 1:step;
Accelerometer = [A(dur,a) A(dur,a+1) A(dur,a+2)];
Gyroscope = [A(dur,gy) A(dur,gy+1) A(dur,gy+2)];
Magnetometer = [A(dur,8) A(dur,9) A(dur,10)];
Filtered_GYR_end = myFilt(Gyroscope(5001:step,1:3));
Filtered_ACC_end = myFilt(Accelerometer(5001:step,1:3));
Filtered_MAG_end = myFilt(Magnetometer(5001:step,1:3));
Filtered_GYR = [Gyroscope(1:5000,:);Filtered_GYR_end];
Filtered_ACC = [Accelerometer(1:5000,:);Filtered_ACC_end];
Filtered_MAG = [Magnetometer(1:5000,:);Filtered_MAG_end];
beta = 0.05;
AHRS = AHRSAlgorithm('SamplePeriod', 1/100, 'Beta', beta);
quaternion = zeros(length(time), 4);
```

```
Cse = zeros(3,3,length(time));
av = 5000;
quaternionst = zeros(av, 4);

for t = 1:av
    AHRS.Update(Filtered_GYR(t,:)*pi/180, Filtered_ACC(t,:),
Filtered_MAG(t,:));    % gyroscope units must be radians
    quaternionst(t, :) = AHRS.Quaternion;
    Cse(:,:,t) = q2mat(quaternionst(t,:));
end

q0 = mean(quaternionst(av-10:av,:));
q1=quaternConj(q0);

AHRS2 = AHRSAlgorithm('SamplePeriod', 1/100, 'Beta',
beta,'Quaternion',q0);

for t = 1:length(time)

AHRS2.Update(Filtered_GYR(t,:)*pi/180, Filtered_ACC(t,:),
Filtered_MAG(t,:));   % gyroscope units must be radian/s
quaternion1(t, :) = AHRS2.Quaternion;
q2(t, :)=quaternConj(quaternion1(t, :));
quaternion2(t, :)=quaternProd(q2(t, :),q0);
quaternion2(t, 4)=-quaternion2(t, 4);

end
out=(quaternion2);
```

- Calculations3

```
function out = Calculations3(B)%#codegen
A = B;
frq = 100;
time = (0:1/frq:(length(A(:,1))-1)/frq)';
step = length(time);
a = 5;
gy = 2;
m = 8;

%%
acc_mult = 16340;
gyro_mult = 1000/8.75;
magno_mult = 1000/0.29;

A(:,a) = B(:,a)/acc_mult;%acc x
A(:,a+1) = B(:,a+2)/acc_mult;%acc z
A(:,a+2) = B(:,a+1)/acc_mult;%acc y
```

```
A(:,gy) = B(:,gy)/gyro_mult; %gyro x
A(:,gy+1) = B(:,gy+2)/gyro_mult; %gyro z
A(:,gy+2) = B(:,gy+1)/gyro_mult; %gyro y

A(:,m) = -B(:,m)/magno_mult; %magno x
A(:,m+1) = B(:,m+2)/magno_mult; %magno z
A(:,m+2) = B(:,m+1)/magno_mult; %magno y

%%
dur = 1:step;
Accelerometer = [A(dur,a) A(dur,a+1) A(dur,a+2)];
Gyroscope = [A(dur,gy) A(dur,gy+1) A(dur,gy+2)];
Magnetometer = [A(dur,8) A(dur,9) A(dur,10)];
Filtered_GYR_end = myFilt(Gyroscope(5001:step,1:3));
Filtered_ACC_end = myFilt(Accelerometer(5001:step,1:3));
Filtered_MAG_end = myFilt(Magnetometer(5001:step,1:3));
Filtered_GYR = [Gyroscope(1:5000,:);Filtered_GYR_end];
Filtered_ACC = [Accelerometer(1:5000,:);Filtered_ACC_end];
Filtered_MAG = [Magnetometer(1:5000,:);Filtered_MAG_end];
beta = 0.05;
AHRS = AHRSAlgorithm('SamplePeriod', 1/100, 'Beta', beta);
quaternion1 = zeros(length(time), 4);
Cse = zeros(3,3,length(time));
av = 5000;
quaternionst = zeros(av, 4);

for t = 1:av
    AHRS.Update(Filtered_GYR(t,:)*pi/180, Filtered_ACC(t,:),
Filtered_MAG(t,:));    % gyroscope units must be radians
    quaternionst(t, :) = AHRS.Quaternion;
    Cse(:,:,t) = q2mat(quaternionst(t,:));
end

q0 = mean(quaternionst(av-10:av,:));
```

- Calculations4

```
function out = Calculations4(B)%#codegen
A = B;
frq = 100;
time = (0:1/frq:(length(A(:,1))-1)/frq)';
step = length(time);
a = 5;%in sensor text document acceleration columns are 5,6 and 7
gy = 2;%in sensor text document angular velocity columns are 2,3
and 4
m = 8;%in sensor text document magnetic measurement columns are 8,9
and 10
```

```
%%
acc_mult = 16340;
gyro_mult = 1000/8.75;
magno_mult = 1000/0.29;

A(:,a) =  B(:,a+1)/acc_mult;% acc y
A(:,a+1) = -B(:,a)/acc_mult;%acc x
A(:,a+2) = -B(:,a+2)/acc_mult;%acc z

A(:,gy) = B(:,gy+1)/gyro_mult; %gyro y
A(:,gy+1) =  -B(:,gy)/gyro_mult; %gyro x
A(:,gy+2) = -B(:,gy+2)/gyro_mult; %gyro z

A(:,m) = B(:,m+1)/magno_mult; %magno y
A(:,m+1) =  B(:,m)/magno_mult; %magno x
A(:,m+2) = -B(:,m+2)/magno_mult; %magno z
%%
dur = 1:step;
Accelerometer = [A(dur,a) A(dur,a+1) A(dur,a+2)];
Gyroscope = [A(dur,gy) A(dur,gy+1) A(dur,gy+2)];
Magnetometer = [A(dur,8) A(dur,9) A(dur,10)];
Filtered_GYR_end = myFilt(Gyroscope(5001:step,1:3));
Filtered_ACC_end = myFilt(Accelerometer(5001:step,1:3));
Filtered_MAG_end = myFilt(Magnetometer(5001:step,1:3));
Filtered_GYR = [Gyroscope(1:5000,:);Filtered_GYR_end];
Filtered_ACC = [Accelerometer(1:5000,:);Filtered_ACC_end];
Filtered_MAG = [Magnetometer(1:5000,:);Filtered_MAG_end];
beta = 0.05;
AHRS = AHRSAlgorithm('SamplePeriod', 1/100, 'Beta', beta);
quaternion = zeros(length(time), 4);
Cse = zeros(3,3,length(time));
av = 5000;
quaternionst = zeros(av, 4);

for t = 1:av
    AHRS.Update(Filtered_GYR(t,:)*pi/180, Filtered_ACC(t,:),
Filtered_MAG(t,:));    % gyroscope units must be radians
    quaternionst(t, :) = AHRS.Quaternion;
    Cse(:,:,t) = q2mat(quaternionst(t,:));
end

q0 = mean(quaternionst(av-10:av,:));
q1=quaternConj(q0);

AHRS2 = AHRSAlgorithm('SamplePeriod', 1/100, 'Beta',
beta,'Quaternion',q0);

for t = 1:length(time)
    AHRS2.Update(Filtered_GYR(t,:)*pi/180, Filtered_ACC(t,:),
Filtered_MAG(t,:));   % gyroscope units must be radian/s
    quaternion1(t, :) = AHRS2.Quaternion;
q2(t, :)=quaternConj(quaternion1(t, :));
quaternion2(t, :)=quaternProd(q2(t, :),q0);
```

```
quaternion2(t, 2)=-quaternion2(t, 2);

end
```

E. Lower Extremity Kinematic Analysis Codes with Euler Angles

- Calculations

```
function out = Calculations(B)%#codegen
A = B;
frq = 100;
time = (0:1/frq:(length(A(:,1))-1)/frq)';
step = length(time);
a = 5;%in sensor text document acceleration columns are 5,6 and 7
gy = 2;%in sensor text document angular velocity columns are 2,3
and 4
m = 8;%in sensor text document magnetic measurement columns are 8,9
and 10

%%
acc_mult = 16340;
gyro_mult = 1000/8.75;
magno_mult = 1000/0.29;
A(:,a) =  -B(:,a+2)/acc_mult;%acc z
A(:,a+1) = -B(:,a)/acc_mult;%acc x
A(:,a+2) = -B(:,a+1)/acc_mult;% acc y

A(:,gy) = -B(:,gy+2)/gyro_mult; %gyro z
A(:,gy+1) =  -B(:,gy)/gyro_mult; %gyro x
A(:,gy+2) = -B(:,gy+1)/gyro_mult; %gyro y

A(:,m) = -B(:,m+2)/magno_mult; %magno z
A(:,m+1) =  B(:,m)/magno_mult; %magno x
A(:,m+2) = -B(:,m+1)/magno_mult; %magno y
%%
dur = 1:step;
Accelerometer = [A(dur,a) A(dur,a+1) A(dur,a+2)];
Gyroscope = [A(dur,gy) A(dur,gy+1) A(dur,gy+2)];
Magnetometer = [A(dur,8) A(dur,9) A(dur,10)];
Filtered_GYR_end = myFilt(Gyroscope(5001:step,1:3));
Filtered_ACC_end = myFilt(Accelerometer(5001:step,1:3));
Filtered_MAG_end = myFilt(Magnetometer(5001:step,1:3));
Filtered_GYR = [Gyroscope(1:5000,:);Filtered_GYR_end];
Filtered_ACC = [Accelerometer(1:5000,:);Filtered_ACC_end];
Filtered_MAG = [Magnetometer(1:5000,:);Filtered_MAG_end];
beta = 0.05;
AHRS = AHRSAlgorithm('SamplePeriod', 1/100, 'Beta', beta);
quaternion = zeros(length(time), 4);
Cse = zeros(3,3,length(time));
av = 5000;
quaternionst = zeros(av, 4);
for t = 1:av
    AHRS.Update(Filtered_GYR(t,:)*pi/180, Filtered_ACC(t,:),
Filtered_MAG(t,:));    % gyroscope units must be radians
    quaternionst(t, :) = AHRS.Quaternion;
    Cse(:,:,t) = q2mat(quaternionst(t,:));
end
```

```
q0 = mean(quaternionst(av-10:av,:));
AHRS2 = AHRSAlgorithm('SamplePeriod', 1/100, 'Beta',
beta,'Quaternion',q0);

for t = 1:length(time)
    AHRS2.Update(Filtered_GYR(t,:)*pi/180, Filtered_ACC(t,:),
Filtered_MAG(t,:));   % gyroscope units must be radian/s
    quaternion(t, :) = AHRS2.Quaternion;

end

euler = quatern2euler(quaternConj(quaternion)) * (180/pi);% use
conjugate for sensor frame relative to Earth and convert to
degrees.
euler=eulercorrect(euler,step);
ksimat = euler(:,3)-mean(euler(1:20,3));
thmat = euler(:,2)-mean(euler(1:20,2));
fimat = euler(:,1)-mean(euler(1:20,1));

for i = 1:length(fimat)
 quat(i,:) =
eu2qtr(fimat(i)*pi/180,thmat(i)*pi/180,ksimat(i)*pi/180);
end

out=(quat);
```

- Calculations2

```
function out = Calculations2(B)%#codegen
A = B;
frq = 100;
time = (0:1/frq:(length(A(:,1))-1)/frq)';
step = length(time);
a = 5;
gy = 2;
m = 8;

%%
acc_mult = 16340;
gyro_mult = 1000/8.75;
magno_mult = 1000/0.29;

A(:,a) = -B(:,a)/acc_mult;%acc x
A(:,a+1) = B(:,a+2)/acc_mult;%acc z
A(:,a+2) = -B(:,a+1)/acc_mult; %acc y

A(:,gy) = -B(:,gy)/gyro_mult; %gyro x
A(:,gy+1) = B(:,gy+2)/gyro_mult; %gyro z
A(:,gy+2) = -B(:,gy+1)/gyro_mult; %gyro y

A(:,m) = B(:,m)/magno_mult; %magno x
A(:,m+1) = B(:,m+2)/magno_mult; %magno z
```

```
A(:,m+2) = -B(:,m+1)/magno_mult; %magno y

%%
dur = 1:step;
Accelerometer = [A(dur,a) A(dur,a+1) A(dur,a+2)];
Gyroscope = [A(dur,gy) A(dur,gy+1) A(dur,gy+2)];
Magnetometer = [A(dur,8) A(dur,9) A(dur,10)];
Filtered_GYR_end = myFilt(Gyroscope(5001:step,1:3));
Filtered_ACC_end = myFilt(Accelerometer(5001:step,1:3));
Filtered_MAG_end = myFilt(Magnetometer(5001:step,1:3));
Filtered_GYR = [Gyroscope(1:5000,:);Filtered_GYR_end];
Filtered_ACC = [Accelerometer(1:5000,:);Filtered_ACC_end];
Filtered_MAG = [Magnetometer(1:5000,:);Filtered_MAG_end];
beta = 0.05;
AHRS = AHRSAlgorithm('SamplePeriod', 1/100, 'Beta', beta);
quaternion = zeros(length(time), 4);
Cse = zeros(3,3,length(time));
av = 5000;
quaternionst = zeros(av, 4);

for t = 1:av
    AHRS.Update(Filtered_GYR(t,:)*pi/180, Filtered_ACC(t,:),
Filtered_MAG(t,:));    % gyroscope units must be radians
    quaternionst(t, :) = AHRS.Quaternion;
    Cse(:,:,t) = q2mat(quaternionst(t,:));
end

q0 = mean(quaternionst(av-10:av,:));
AHRS2 = AHRSAlgorithm('SamplePeriod', 1/100, 'Beta',
beta,'Quaternion',q0);

for t = 1:length(time)
    AHRS2.Update(Filtered_GYR(t,:)*pi/180, Filtered_ACC(t,:),
Filtered_MAG(t,:));   % gyroscope units must be radian/s
    quaternion(t, :) = AHRS2.Quaternion;
     quaternion2(t, :) = quaternion(t,:)- quaternConj(q0);
end

euler = quatern2euler(quaternConj(quaternion)) * (180/pi);% use
conjugate for sensor frame relative to Earth and convert to
degrees.
euler=eulercorrect(euler,step);
ksimat = euler(:,3)-mean(euler(1:20,3));
thmat = euler(:,2)-mean(euler(1:20,2));
fimat = euler(:,1)-mean(euler(1:20,1));

for i = 1:length(fimat)
    quat(i,:) =
eu2qtr(fimat(i)*pi/180,thmat(i)*pi/180,ksimat(i)*pi/180);
end

out=(quat);
```

- Calculations3

```
function out = Calculations3(B)%#codegen
A = B;
frq = 100;
time = (0:1/frq:(length(A(:,1))-1)/frq)';
step = length(time);
a = 5;
gy = 2;
m = 8;


%%
acc_mult = 16340;
gyro_mult = 1000/8.75;
magno_mult = 1000/0.29;
A(:,a) = B(:,a)/acc_mult;%acc x
A(:,a+1) = B(:,a+2)/acc_mult;%acc z
A(:,a+2) = B(:,a+1)/acc_mult; %acc y

A(:,gy) = B(:,gy)/gyro_mult; %gyro x
A(:,gy+1) = B(:,gy+2)/gyro_mult; %gyro z
A(:,gy+2) = B(:,gy+1)/gyro_mult; %gyro y

A(:,m) = -B(:,m)/magno_mult; %magno x
A(:,m+1) = B(:,m+2)/magno_mult; %magno z
A(:,m+2) = B(:,m+1)/magno_mult; %magno y
%%
dur = 1:step;
Accelerometer = [A(dur,a) A(dur,a+1) A(dur,a+2)];
Gyroscope = [A(dur,gy) A(dur,gy+1) A(dur,gy+2)];
Magnetometer = [A(dur,8) A(dur,9) A(dur,10)];
Filtered_GYR_end = myFilt(Gyroscope(5001:step,1:3));
Filtered_ACC_end = myFilt(Accelerometer(5001:step,1:3));
Filtered_MAG_end = myFilt(Magnetometer(5001:step,1:3));
Filtered_GYR = [Gyroscope(1:5000,:);Filtered_GYR_end];
Filtered_ACC = [Accelerometer(1:5000,:);Filtered_ACC_end];
Filtered_MAG = [Magnetometer(1:5000,:);Filtered_MAG_end];
beta = 0.05;
AHRS = AHRSAlgorithm('SamplePeriod', 1/100, 'Beta', beta);
quaternion = zeros(length(time), 4);
Cse = zeros(3,3,length(time));
av = 5000;
quaternionst = zeros(av, 4);
for t = 1:av
    AHRS.Update(Filtered_GYR(t,:)*pi/180, Filtered_ACC(t,:),
Filtered_MAG(t,:));    % gyroscope units must be radians
    quaternionst(t, :) = AHRS.Quaternion;
    Cse(:,:,t) = q2mat(quaternionst(t,:));
end

q0 = mean(quaternionst(av-10:av,:));
AHRS2 = AHRSAlgorithm('SamplePeriod', 1/100, 'Beta',
beta,'Quaternion',q0);
```

```
for t = 1:length(time)
    AHRS2.Update(Filtered_GYR(t,:)*pi/180, Filtered_ACC(t,:),
Filtered_MAG(t,:));   % gyroscope units must be radian/s
    quaternion(t, :) = AHRS2.Quaternion;
       quaternion2(t, :) = quaternion(t,:)- quaternConj(q0);
end

euler = quatern2euler(quaternConj(quaternion)) * (180/pi);% use
conjugate for sensor frame relative to Earth and convert to
degrees.
euler=eulercorrect(euler,step);
ksimat = euler(:,3)-mean(euler(1:20,3));
thmat = euler(:,2)-mean(euler(1:20,2));
fimat = euler(:,1)-mean(euler(1:20,1));

for i = 1:length(fimat)
    quat(i,:) =
eu2qtr(fimat(i)*pi/180,thmat(i)*pi/180,ksimat(i)*pi/180);
end

out=(quat);
```

- Calculations4

```
function out = Calculations4(B)%#codegen
A = B;
frq = 100;
time = (0:1/frq:(length(A(:,1))-1)/frq)';
step = length(time);
a = 5;%in sensor text document acceleration columns are 5,6 and 7
gy = 2;%in sensor text document angular velocity columns are 2,3
and 4
m = 8;%in sensor text document magnetic measurement columns are 8,9
and 10

%%
acc_mult = 16340;
gyro_mult = 1000/8.75;
magno_mult = 1000/0.29;

A(:,a) =  B(:,a+1)/acc_mult;% acc y
A(:,a+1) = -B(:,a)/acc_mult;%acc x
A(:,a+2) = -B(:,a+2)/acc_mult;%acc z

A(:,gy) = B(:,gy+1)/gyro_mult; %gyro y
A(:,gy+1) =  -B(:,gy)/gyro_mult; %gyro x
A(:,gy+2) = -B(:,gy+2)/gyro_mult; %gyro z

A(:,m) = B(:,m+1)/magno_mult; %magno y
```

```
A(:,m+1) =  B(:,m)/magno_mult; %magno x
A(:,m+2) = -B(:,m+2)/magno_mult; %magno z
%%
dur = 1:step;
Accelerometer = [A(dur,a) A(dur,a+1) A(dur,a+2)];
Gyroscope = [A(dur,gy) A(dur,gy+1) A(dur,gy+2)];
Magnetometer = [A(dur,8) A(dur,9) A(dur,10)];
Filtered_GYR_end = myFilt(Gyroscope(5001:step,1:3));
Filtered_ACC_end = myFilt(Accelerometer(5001:step,1:3));
Filtered_MAG_end = myFilt(Magnetometer(5001:step,1:3));
Filtered_GYR = [Gyroscope(1:5000,:);Filtered_GYR_end];
Filtered_ACC = [Accelerometer(1:5000,:);Filtered_ACC_end];
Filtered_MAG = [Magnetometer(1:5000,:);Filtered_MAG_end];
beta = 0.05;
AHRS = AHRSAlgorithm('SamplePeriod', 1/100, 'Beta', beta);
quaternion = zeros(length(time), 4);
Cse = zeros(3,3,length(time));
av = 5000;
quaternionst = zeros(av, 4);
for t = 1:av
    AHRS.Update(Filtered_GYR(t,:)*pi/180, Filtered_ACC(t,:),
Filtered_MAG(t,:));    % gyroscope units must be radians
    quaternionst(t, :) = AHRS.Quaternion;
    Cse(:,:,t) = q2mat(quaternionst(t,:));
end

q0 = mean(quaternionst(av-10:av,:));
AHRS2 = AHRSAlgorithm('SamplePeriod', 1/100, 'Beta',
beta,'Quaternion',q0);

for t = 1:length(time)
    AHRS2.Update(Filtered_GYR(t,:)*pi/180, Filtered_ACC(t,:),
Filtered_MAG(t,:));   % gyroscope units must be radian/s
    quaternion(t, :) = AHRS2.Quaternion;
    quaternion2(t, :) = quaternion(t,:)- quaternConj(q0);
end

euler = quatern2euler(quaternConj(quaternion)) * (180/pi);% use
conjugate for sensor frame relative to Earth and convert to
degrees.
euler=eulercorrect(euler,step);
ksimat = euler(:,3)-mean(euler(1:20,3));
thmat = euler(:,2)-mean(euler(1:20,2));
fimat = euler(:,1)-mean(euler(1:20,1));

for i = 1:length(fimat)
    quat(i,:) =
eu2qtr(fimat(i)*pi/180,thmat(i)*pi/180,ksimat(i)*pi/180);
end

out = quat;
```

F. Upper Extremity Kinematic Analysis Codes

- TestCpp

```
function C = TestCpp7(dataPath)
 %#codegen
fprintf('#App starting..\n\n');
rootpath = '';
%%
if  isdeployed()
    if (nargin<1)
        dataPath='';
        msgbox('Input datapath is empty/null');
    end
else
    dataPath ='C:\Users\selin\Desktop\DENEY\1.txt';

    rootpath = 'C:\Users\selin\Desktop\';
end

if isempty(dataPath)== false
    fprintf('loading file from %s\n',dataPath);
    B = dlmread(dataPath);
    B = B(:,1:end-1);
    numbodies = 7;
    numdata = length(B);
    if numdata>=numbodies
        numframe = numdata/numbodies;
        diff = numframe - 5000;
        [C] = Matrices7(B,numbodies,numdata,numframe);
        torso  =  (C.torso);
        humerusR =  (C.humerusR);
        humerusL =  (C.humerusL);
        ulnaR  =  (C.ulnaR);
        handR  =  (C.handR);
        handL  =  (C.handL);
        ulnaL =  (C.ulnaL);

        % Write to file
         fid = fopen('mlab.sto','w+','n','UTF-8');
         fseek(fid,0,-1);

        fprintf(fid,'\r\nDataRate=100.000000');
        fprintf(fid,'\r\nDataType=Quaternion');
        fprintf(fid,'\r\nversion=3');
        fprintf(fid,'\r\nOpenSimVersion=4.3');
        fprintf(fid,'\r\nendheader');



       fprintf(fid,'\r\ntime ');
       fprintf(fid,'\t torso_imu');
```

```
        fprintf(fid,'\t ulna_r_imu');
        fprintf(fid,'\t humerus_r_imu');
        fprintf(fid,'\t hand_r_imu');
        fprintf(fid,'\t hand_l_imu');
        fprintf(fid,'\t ulna_l_imu');
        fprintf(fid,'\t humerus_l_imu');
        line_index =1;

         for k=5000: numframe

             fprintf(fid,'\r\n%.2f\t',(line_index/100.0));

             ltorso1 = torso(k,1);
             ltorso2 = torso(k,2);
             ltorso3 = torso(k,3);
             ltorso4 = torso(k,4);
            fprintf(fid,'%.6f,%.6f,%.6f,%.6f\t', ltorso1, ltorso2,
ltorso3, ltorso4);

             lulnar1 = ulnaR(k,1);
             lulnar2 = ulnaR(k,2);
             lulnar3 = ulnaR(k,3);
             lulnar4 = ulnaR(k,4);
            fprintf(fid,'%.6f,%.6f,%.6f,%.6f\t', lulnar1, lulnar2,
lulnar3, lulnar4);

             lhumerusr1 = humerusR(k,1);
             lhumerusr2 = humerusR(k,2);
             lhumerusr3 = humerusR(k,3);
             lhumerusr4 = humerusR(k,4);
            fprintf(fid,'%.6f,%.6f,%.6f,%.6f\t', lhumerusr1,
lhumerusr2, lhumerusr3, lhumerusr4);

             lhandr1 = handR(k,1);
             lhandr2 = handR(k,2);
             lhandr3 = handR(k,3);
             lhandr4 = handR(k,4);
            fprintf(fid,'%.6f,%.6f,%.6f,%.6f\t', lhandr1, lhandr2,
lhandr3, lhandr4);

             lhandl1 = handL(k,1);
             lhandl2 = handL(k,2);
             lhandl3 = handL(k,3);
             lhandl4 = handL(k,4);
            fprintf(fid,'%.6f,%.6f,%.6f,%.6f\t', lhandl1, lhandl2,
lhandl3, lhandl4);

             lulnal1 = ulnaL(k,1);
             lulnal2 = ulnaL(k,2);
             lulnal3 = ulnaL(k,3);
             lulnal4 = ulnaL(k,4);
            fprintf(fid,'%.6f,%.6f,%.6f,%.6f\t', lulnal1, lulnal2,
lulnal3, lulnal4);
```

```
            lhumerusl1 = humerusL(k,1);
            lhumerusl2 = humerusL(k,2);
            lhumerusl3 = humerusL(k,3);
            lhumerusl4 = humerusL(k,4);
           fprintf(fid,'%.6f,%.6f,%.6f,%.6f', lhumerusl1,
lhumerusl2, lhumerusl3, lhumerusl4);

            line_index = line_index+1;
            end
        end

        fprintf(fid,'\r\n');
        fclose(fid);

end

%%
clear all;
close all;
clc;
import org.opensim.modeling.*

%% OpenSim variables
modelFileName = 'Rajagopal_2015.osim';          % The path to an
input model
orientationsFileName = 'mlab.sto';   % The path to orientation data
for calibration
sensor_to_opensim_rotations = Vec3(pi/2, 0, 0);% The rotation of
IMU data to the OpenSim world frame
baseIMUName = 'torso_imu';                     % The base IMU is
the IMU on the base body of the model that dictates the heading
(forward) direction of the model.
baseIMUHeading = '-x';                          % The Coordinate
Axis of the base IMU that points in the heading direction.
visulizeCalibration = true;                    % Boolean to
Visualize the Output model

%% imuPlacer
imuPlacer = IMUPlacer();

imuPlacer.set_model_file(modelFileName);
imuPlacer.set_orientation_file_for_calibration(orientationsFileName
);
imuPlacer.set_sensor_to_opensim_rotations(sensor_to_opensim_rotatio
ns);
imuPlacer.set_base_imu_label(baseIMUName);
imuPlacer.set_base_heading_axis(baseIMUHeading);

imuPlacer.run(visulizeCalibration);

model = imuPlacer.getCalibratedModel();
```

```
model.print( strrep(modelFileName, '.osim', '_calibrated.osim') );
%%
clear all; close all; clc;
import org.opensim.modeling.*

%% OpenSim Variables
modelFileName = 'Rajagopal_2015_calibrated.osim';                % 
The path to an input model
orientationsFileName = 'mlab.sto';   % The path to orientation data
for calibration
sensor_to_opensim_rotation = Vec3(pi/2, 0, 0); % The rotation of
IMU data to the OpenSim world frame
visualizeTracking = true;  % Boolean to Visualize the tracking
simulation
startTime = 0;          % Start time (in seconds) of the tracking
simulation.
endTime = 40;              % End time (in seconds) of the tracking
simulation.
resultsDirectory = 'IKResults';

%% InverseKinematicsTool
imuIK = IMUInverseKinematicsTool();

%% tracking
imuIK.set_model_file(modelFileName);
imuIK.set_orientations_file(orientationsFileName);
imuIK.set_sensor_to_opensim_rotations(sensor_to_opensim_rotation)

imuIK.set_time_range(0, startTime);
imuIK.set_time_range(1, endTime);

imuIK.set_results_directory(resultsDirectory)

% Run IK
imuIK.run(visualizeTracking);
fprintf('#Finished\n');
end
```

- Matrices

```
function [C] = Matrices7(A,numbodies,numdata,numframe)
rt = 1;rf = 1;p = 1;lf = 1;lt = 1; rfo=1; lfo=1;
B = A(1:numdata,:);
numcolB = size(B,2);
imufield1 = 'val';
imufield2 = 'bodynames';
nul1 = cell(1,numbodies);
nul2 = cell(1,numbodies);
for i = 1:numbodies
    nul1{i} = zeros(numdata,numcolB-1);
    nul2{i} = 'al';
end
```

```
imu = struct(imufield1,nul1,imufield2,nul2);
for i = 1:length(B)
    switch B(i,1)
        case 2
            imu(6).val(rfo,:) = B(i,2:end); %right hand
            imu(6).bodynames = 'rha';
            rfo = rfo+1;
        case 3
            imu(1).val(rt,:) = B(i,2:end); %right ulna
            imu(1).bodynames = 'ru';
            rt = rt+1;
        case 4
            imu(2).val(rf,:) = B(i,2:end); %right humerus
            imu(2).bodynames = 'rh';
            rf = rf+1;
        case 5
            imu(3).val(p,:) = B(i,2:end); %torso
            imu(3).bodynames = 'to';
            p = p+1;
        case 6
            imu(4).val(lf,:) = B(i,2:end); %left humerus
            imu(4).bodynames = 'lh';
            lf = lf+1;
        case 7
            imu(5).val(lt,:) = B(i,2:end); %left ulna
            imu(5).bodynames = 'lu';
            lt = lt+1;
        case 8
            imu(7).val(lfo,:) = B(i,2:end); %left hand
            imu(7).bodynames = 'lha';
            lfo = lfo+1;
    end
end
[C.torso] = Calculations(imu(3).val(1:numframe,:));
[C.handR] = Calculations4(imu(6).val(1:numframe,:));
[C.handL] = Calculations4(imu(7).val(1:numframe,:));
[C.humerusR] = Calculations2(imu(2).val(1:numframe,:));
[C.humerusL] = Calculations3(imu(4).val(1:numframe,:));
[C.ulnaR] = Calculations2(imu(1).val(1:numframe,:));
[C.ulnaL] = Calculations3(imu(5).val(1:numframe,:));

trc_frame_no = size(C.torso,4);
num_frame = trc_frame_no-5000;
```

- Calculations

```
function out = Calculations(B)%#codegen
A = B;
frq = 100;
time = (0:1/frq:(length(A(:,1))-1)/frq)';
step = length(time);
a = 5;%in sensor text document acceleration columns are 5,6 and 7
gy = 2;%in sensor text document angular velocity columns are 2,3
and 4
```

```
m = 8;%in sensor text document magnetic measurement columns are 8,9
and 10

%%
acc_mult = 16340;
gyro_mult = 1000/8.75;
magno_mult = 1000/0.29;

A(:,a) =  -B(:,a+2)/acc_mult;%acc z
A(:,a+1) = -B(:,a)/acc_mult;%acc x
A(:,a+2) = -B(:,a+1)/acc_mult;% acc y

A(:,gy) = -B(:,gy+2)/gyro_mult; %gyro z
A(:,gy+1) =  -B(:,gy)/gyro_mult; %gyro x
A(:,gy+2) = -B(:,gy+1)/gyro_mult; %gyro y

A(:,m) = -B(:,m+2)/magno_mult; %magno z
A(:,m+1) =  B(:,m)/magno_mult; %magno x
A(:,m+2) = -B(:,m+1)/magno_mult; %magno y

%%
dur = 1:step;
Accelerometer = [A(dur,a) A(dur,a+1) A(dur,a+2)];
Gyroscope = [A(dur,gy) A(dur,gy+1) A(dur,gy+2)];
Magnetometer = [A(dur,8) A(dur,9) A(dur,10)];
Filtered_GYR_end = myFilt(Gyroscope(5001:step,1:3));
Filtered_ACC_end = myFilt(Accelerometer(5001:step,1:3));
Filtered_MAG_end = myFilt(Magnetometer(5001:step,1:3));
Filtered_GYR = [Gyroscope(1:5000,:);Filtered_GYR_end];
Filtered_ACC = [Accelerometer(1:5000,:);Filtered_ACC_end];
Filtered_MAG = [Magnetometer(1:5000,:);Filtered_MAG_end];
beta = 0.05;
AHRS = AHRSAlgorithm('SamplePeriod', 1/100, 'Beta', beta);
quaternion1 = zeros(length(time), 4);
Cse = zeros(3,3,length(time));
av = 5000;
quaternionst = zeros(av, 4);
for t = 1:av
    AHRS.Update(Filtered_GYR(t,:)*pi/180, Filtered_ACC(t,:),
Filtered_MAG(t,:));    % gyroscope units must be radians
    quaternionst(t, :) = AHRS.Quaternion;
    Cse(:,:,t) = q2mat(quaternionst(t,:));
end
q0 = mean(quaternionst(av-10:av,:));
q1=quaternConj(q0);

AHRS2 = AHRSAlgorithm('SamplePeriod', 1/100, 'Beta',
beta,'Quaternion',q0);
for t = 1:length(time)
    AHRS2.Update(Filtered_GYR(t,:)*pi/180, Filtered_ACC(t,:),
Filtered_MAG(t,:));   % gyroscope units must be radian/s
    quaternion1(t, :) = AHRS2.Quaternion;
q2(t, :)=quaternConj(quaternion1(t, :));
quaternion2(t, :)=quaternProd(q2(t, :),q0);
```

```
quaternion2(t, 4)=-quaternion2(t, 4);
end

out=(quaternion2);
```

- Calculations2

```
function out = Calculations2(B)%#codegen
A = B;
frq = 100;
time = (0:1/frq:(length(A(:,1))-1)/frq)';
step = length(time);
a = 5;
gy = 2;
m = 8;

%%
acc_mult = 16340;
gyro_mult = 1000/8.75;
magno_mult = 1000/0.29;
A(:,a) = -B(:,a)/acc_mult;%acc x
A(:,a+1) = B(:,a+2)/acc_mult;%acc z
A(:,a+2) = -B(:,a+1)/acc_mult;

A(:,gy) = -B(:,gy)/gyro_mult; %gyro x
A(:,gy+1) = B(:,gy+2)/gyro_mult; %gyro z
A(:,gy+2) = -B(:,gy+1)/gyro_mult; %gyro y

A(:,m) = B(:,m)/magno_mult; %magno x
A(:,m+1) = B(:,m+2)/magno_mult; %magno z
A(:,m+2) = -B(:,m+1)/magno_mult; %magno y
%%
dur = 1:step;
Accelerometer = [A(dur,a) A(dur,a+1) A(dur,a+2)];
Gyroscope = [A(dur,gy) A(dur,gy+1) A(dur,gy+2)];
Magnetometer = [A(dur,8) A(dur,9) A(dur,10)];
Filtered_GYR_end = myFilt(Gyroscope(5001:step,1:3));
Filtered_ACC_end = myFilt(Accelerometer(5001:step,1:3));
Filtered_MAG_end = myFilt(Magnetometer(5001:step,1:3));
Filtered_GYR = [Gyroscope(1:5000,:);Filtered_GYR_end];
Filtered_ACC = [Accelerometer(1:5000,:);Filtered_ACC_end];
Filtered_MAG = [Magnetometer(1:5000,:);Filtered_MAG_end];
beta = 0.05;
AHRS = AHRSAlgorithm('SamplePeriod', 1/100, 'Beta', beta);
quaternion = zeros(length(time), 4);
Cse = zeros(3,3,length(time));
av = 5000;
quaternionst = zeros(av, 4);
for t = 1:av
    AHRS.Update(Filtered_GYR(t,:)*pi/180, Filtered_ACC(t,:),
Filtered_MAG(t,:));    % gyroscope units must be radians
    quaternionst(t, :) = AHRS.Quaternion;
```

```
    Cse(:,:,t) = q2mat(quaternionst(t,:));
end
q0 = mean(quaternionst(av-10:av,:));
q1=quaternConj(q0);
AHRS2 = AHRSAlgorithm('SamplePeriod', 1/100, 'Beta',
beta,'Quaternion',q0);

for t = 1:length(time)
    AHRS2.Update(Filtered_GYR(t,:)*pi/180, Filtered_ACC(t,:),
Filtered_MAG(t,:));   % gyroscope units must be radian/s
    quaternion1(t, :) = AHRS2.Quaternion;
q2(t, :)=quaternConj(quaternion1(t, :));
quaternion2(t, :)=quaternProd(q2(t, :),q0);
quaternion2(t, 4)=-quaternion2(t, 4);
end

out=(quaternion2);
```

- Calculations3

```
function out = Calculations3(B)%#codegen
A = B;
frq = 100;
time = (0:1/frq:(length(A(:,1))-1)/frq)';
step = length(time);
a = 5;
gy = 2;
m = 8;


%%
acc_mult = 16340;
gyro_mult = 1000/8.75;
magno_mult = 1000/0.29;
A(:,a) = B(:,a)/acc_mult;%acc x
A(:,a+1) = B(:,a+2)/acc_mult;%acc z
A(:,a+2) = B(:,a+1)/acc_mult;

A(:,gy) = B(:,gy)/gyro_mult; %gyro x
A(:,gy+1) = B(:,gy+2)/gyro_mult; %gyro z
A(:,gy+2) = B(:,gy+1)/gyro_mult; %gyro y

A(:,m) = -B(:,m)/magno_mult; %magno x
A(:,m+1) = B(:,m+2)/magno_mult; %magno z
A(:,m+2) = B(:,m+1)/magno_mult;
%%
dur = 1:step;
Accelerometer = [A(dur,a) A(dur,a+1) A(dur,a+2)];
Gyroscope = [A(dur,gy) A(dur,gy+1) A(dur,gy+2)];
Magnetometer = [A(dur,8) A(dur,9) A(dur,10)];
Filtered_GYR_end = myFilt(Gyroscope(5001:step,1:3));
Filtered_ACC_end = myFilt(Accelerometer(5001:step,1:3));
Filtered_MAG_end = myFilt(Magnetometer(5001:step,1:3));
```

```
Filtered_GYR = [Gyroscope(1:5000,:);Filtered_GYR_end];
Filtered_ACC = [Accelerometer(1:5000,:);Filtered_ACC_end];
Filtered_MAG = [Magnetometer(1:5000,:);Filtered_MAG_end];
beta = 0.05;
AHRS = AHRSAlgorithm('SamplePeriod', 1/100, 'Beta', beta);
quaternion1 = zeros(length(time), 4);
Cse = zeros(3,3,length(time));
av = 5000;
quaternionst = zeros(av, 4);
for t = 1:av
    AHRS.Update(Filtered_GYR(t,:)*pi/180, Filtered_ACC(t,:),
Filtered_MAG(t,:));    % gyroscope units must be radians
    quaternionst(t, :) = AHRS.Quaternion;
    Cse(:,:,t) = q2mat(quaternionst(t,:));
end
q0 = mean(quaternionst(av-10:av,:));
q1=quaternConj(q0);
AHRS2 = AHRSAlgorithm('SamplePeriod', 1/100, 'Beta',
beta,'Quaternion',q0);
for t = 1:length(time)
    AHRS2.Update(Filtered_GYR(t,:)*pi/180, Filtered_ACC(t,:),
Filtered_MAG(t,:));   % gyroscope units must be radian/s
    quaternion1(t, :) = AHRS2.Quaternion;
q2(t, :)=quaternConj(quaternion1(t, :));
quaternion2(t, :)=quaternProd(q2(t, :),q0);
quaternion2(t, 3)=-quaternion2(t, 3);
end

out=(quaternion2);
```

- Calculations4

```
function out = Calculations4(B)%#codegen
A = B;
frq = 100;
time = (0:1/frq:(length(A(:,1))-1)/frq)';
step = length(time);
a = 5;%in sensor text document acceleration columns are 5,6 and 7
gy = 2;%in sensor text document angular velocity columns are 2,3
and 4
m = 8;%in sensor text document magnetic measurement columns are 8,9
and 10

%%
acc_mult = 16340;
gyro_mult = 1000/8.75;
magno_mult = 1000/0.29;

A(:,a) =  B(:,a+2)/acc_mult;%acc z
A(:,a+1) = B(:,a)/acc_mult;%acc x
A(:,a+2) = -B(:,a+1)/acc_mult;% acc y

A(:,gy) = B(:,gy+2)/gyro_mult; %gyro z
A(:,gy+1) =  B(:,gy)/gyro_mult; %gyro x
```

```
A(:,gy+2) = -B(:,gy+1)/gyro_mult; %gyro y


A(:,m) = B(:,m+2)/magno_mult; %magno z
A(:,m+1) =  -B(:,m)/magno_mult; %magno x
A(:,m+2) = -B(:,m+1)/magno_mult;
%%
dur = 1:step;
Accelerometer = [A(dur,a) A(dur,a+1) A(dur,a+2)];
Gyroscope = [A(dur,gy) A(dur,gy+1) A(dur,gy+2)];
Magnetometer = [A(dur,8) A(dur,9) A(dur,10)];
Filtered_GYR_end = myFilt(Gyroscope(5001:step,1:3));
Filtered_ACC_end = myFilt(Accelerometer(5001:step,1:3));
Filtered_MAG_end = myFilt(Magnetometer(5001:step,1:3));
Filtered_GYR = [Gyroscope(1:5000,:);Filtered_GYR_end];
Filtered_ACC = [Accelerometer(1:5000,:);Filtered_ACC_end];
Filtered_MAG = [Magnetometer(1:5000,:);Filtered_MAG_end];
beta = 0.05;
AHRS = AHRSAlgorithm('SamplePeriod', 1/100, 'Beta', beta);
quaternion = zeros(length(time), 4);
Cse = zeros(3,3,length(time));
av = 5000;
quaternionst = zeros(av, 4);
for t = 1:av
    AHRS.Update(Filtered_GYR(t,:)*pi/180, Filtered_ACC(t,:),
Filtered_MAG(t,:));    % gyroscope units must be radians
    quaternionst(t, :) = AHRS.Quaternion;
    Cse(:,:,t) = q2mat(quaternionst(t,:));
end
q0 = mean(quaternionst(av-10:av,:));
q1=quaternConj(q0);
AHRS2 = AHRSAlgorithm('SamplePeriod', 1/100, 'Beta',
beta,'Quaternion',q0);
for t = 1:length(time)
    AHRS2.Update(Filtered_GYR(t,:)*pi/180, Filtered_ACC(t,:),
Filtered_MAG(t,:));   % gyroscope units must be radian/s
    quaternion1(t, :) = AHRS2.Quaternion;
q2(t, :)=quaternConj(quaternion1(t, :));
quaternion2(t, :)=quaternProd(q2(t, :),q0);
quaternion2(t, 4)=-quaternion2(t, 4);

end

out=(quaternion2);
```

G. Kinetic Analysis Codes

- Grfmot(to Direction 1)

```
% lab to the window(Direction 1)
fprintf('#App starting..\n\n');
rootpath = '';
%%
if  isdeployed()
    if (nargin<1)
        dataPath='';
        msgbox('Input datapath is empty/null');
    end
else
dataPath ='C:\Users\selin\Desktop\bes sensor -
quaternion\yeni\3302_force_plate.txt';
    rootpath = 'C:\Users\selin\Desktop\';
end

if isempty(dataPath)== false
    fprintf('loading file from %s\n',dataPath);
    plate = dlmread(dataPath);
    all_readings = plate(:,1:end-1);
 end
    readings_plate1 = all_readings(:,1:6)/20;
    readings_plate1(:,3) = readings_plate1(:,3)*2;
    readings_plate11=transpose(readings_plate1);

    readings_plate2 = all_readings(:,7:end)/20;
    readings_plate2(:,3) = readings_plate2(:,3)*2;
      readings_plate22=transpose(readings_plate2);


    Cal_C1 = [-1281.5 -18.1 -2.3 -3.5 -7 -10.3;...
              -26.6 1272.1 -3.3 -3.5 -4.9 -27.7;...
              25.5 3.6 1878.8 20.6 -3.7 -12.1;...
              3.8 -147 0.3 581.8 6.8 2.9;...
              -146 -0.7 -0.4 2.3 402 -2.0;...
              1.3 -3.7 -0.7 5 -0.03 295.5];

    Cal_C2 = [1510 -29 18 5 -6 -8;...
              34 1519 1 -3 -1 -33;...
              -33 -1 3014 23 4 -19;...
              -5 -179 -1 789 8 3;...
              176 0 -3 6 551 -2;...
              0 -5 -2 1 2 354];

    FM1 = Cal_C1*readings_plate11;
    FM2 = Cal_C2*readings_plate22;
    FM11=transpose(FM1);
    FM22=transpose(FM2);
   time = size(FM22,1);
```

```
 zz= 0.005*ones(1,time);
 zzz=transpose(zz);

%Point of application of force(force plate1; x axis)

 v1=-transpose(zz)*FM1(1,:);
 x1=(v1(1,:)-FM1(5,:))./FM1(3,:);
 x11=transpose(x1);

%Point of application of force(force plate2; x axis)
 v2=-transpose(zz)*FM2(1,:);
 x2=(v2(1,:)-FM2(5,:))./FM2(3,:);
 x22=transpose(x2);

%Point of application of force(force plate1; y axis)
 q1=-transpose(zz)*FM1(2,:);
 y1=(q1(1,:)+FM1(4,:))./FM1(3,:);
 y11=transpose(y1);

 %Point of application of force(force plate2; y axis)
 q2=-transpose(zz)*FM2(2,:);
 y2=(q2(1,:)+FM2(4,:))./FM2(3,:);
 y22=transpose(y2);

 rows = size(x11,1);
 cols=19;
 fid = fopen('grf.mot','w+','n','UTF-8');
 fseek(fid,0,-1);
 fprintf(fid,'grf_fp.mot');
 fprintf(fid,'\r\nversion=4.3');
 fprintf(fid,'\r\nnRows=%d',rows);
 fprintf(fid,'\r\nnColumns=%d',cols);
 fprintf(fid,'\r\ninDegrees=yes');
 fprintf(fid,'\r\nendheader');

    fprintf(fid,'\r\ntime ');
    fprintf(fid,'\t fp_1x');
    fprintf(fid,'\t fp_1y');
    fprintf(fid,'\t fp_1z');
    fprintf(fid,'\t p_1x');
    fprintf(fid,'\t p_1y');
    fprintf(fid,'\t p_1z');
    fprintf(fid,'\t fp_2x');
    fprintf(fid,'\t fp_2y');
    fprintf(fid,'\t fp_2z');
    fprintf(fid,'\t p_2x');
    fprintf(fid,'\t p_2y');
    fprintf(fid,'\t p_2z');
    fprintf(fid,'\t m_1x');
    fprintf(fid,'\t m_1y');
    fprintf(fid,'\t m_1z');
    fprintf(fid,'\t m_2x');
    fprintf(fid,'\t m_2y');
    fprintf(fid,'\t m_2z');
```

```
       line_index =1;

    for k=0: (rows-1)
            fprintf(fid,'\r\n%.2f',0.01*(k+1));%time
            fprintf(fid,'\t%.3f',-FM11(k+1,2));%f1x
            fprintf(fid,'\t%.3f',FM11(k+1,3));%f1y
            fprintf(fid,'\t%.3f',FM11(k+1,1));%f1z
            fprintf(fid,'\t%.3f',-y11(k+1,1));%point1x
            fprintf(fid,'\t%.3f',zzz(k+1,1));%point1y
            fprintf(fid,'\t%.3f',x11(k+1,1));%point1z
            fprintf(fid,'\t%.3f',FM22(k+1,2));%f2x
            fprintf(fid,'\t%.3f',FM22(k+1,3));%f2y
            fprintf(fid,'\t%.3f',-FM22(k+1,1));%f2z
            fprintf(fid,'\t%.3f',y22(k+1,1));%point2x
            fprintf(fid,'\t%.3f',zzz(k+1,1));%point2y
            fprintf(fid,'\t%.3f',-x22(k+1,1));%point2z
            fprintf(fid,'\t%.3f',-FM11(k+1,5));%t1x
            fprintf(fid,'\t%.3f',FM11(k+1,6));%t1y
            fprintf(fid,'\t%.3f',FM11(k+1,4));%t1z
            fprintf(fid,'\t%.3f',FM22(k+1,5));%t2x
            fprintf(fid,'\t%.3f',FM22(k+1,6));%t2y
            fprintf(fid,'\t%.3f',-FM22(k+1,4));%t2z

        end
    fclose(fid);
```

- Grfmot(to Direction 2)

```
% lab to the door (Direction 2)
fprintf('#App starting..\n\n');
rootpath = '';
%%
if  isdeployed()
    if (nargin<1)
        dataPath='';
        msgbox('Input datapath is empty/null');
    end
else
    dataPath ='C:\Users\selin\Desktop\kd\kd1-fp.txt';

    rootpath = 'C:\Users\selin\Desktop\';
end

if isempty(dataPath)== false
    fprintf('loading file from %s\n',dataPath);
    plate = dlmread(dataPath);
    all_readings = plate(:,1:end-1);
 end
    readings_plate1 = all_readings(:,1:6)/20;
    readings_plate1(:,3) = readings_plate1(:,3)*2;
```

```
readings_plate11=transpose(readings_plate1);

readings_plate2 = all_readings(:,7:end)/20;
readings_plate2(:,3) = readings_plate2(:,3)*2;
  readings_plate22=transpose(readings_plate2);


Cal_C1 = [-1281.5 -18.1 -2.3 -3.5 -7 -10.3;...
          -26.6 1272.1 -3.3 -3.5 -4.9 -27.7;...
          25.5 3.6 1878.8 20.6 -3.7 -12.1;...
          3.8 -147 0.3 581.8 6.8 2.9;...
          -146 -0.7 -0.4 2.3 402 -2.0;...
          1.3 -3.7 -0.7 5 -0.03 295.5];

Cal_C2 = [1510 -29 18 5 -6 -8;...
          34 1519 1 -3 -1 -33;...
          -33 -1 3014 23 4 -19;...
          -5 -179 -1 789 8 3;...
          176 0 -3 6 551 -2;...
          0 -5 -2 1 2 354];

FM1 = Cal_C1*readings_plate11;
FM2 = Cal_C2*readings_plate22;
FM11=transpose(FM1);
FM22=transpose(FM2);
time = size(FM22,1);
zz= 0.005*ones(1,time);
zzz=transpose(zz);

%Point force plate1 x

v1=-transpose(zz)*FM1(1,:);
x1=(v1(1,:)-FM1(5,:))./FM1(3,:);
x11=transpose(x1);

%Point force plate2 x
v2=-transpose(zz)*FM2(1,:);
x2=(v2(1,:)-FM2(5,:))./FM2(3,:);
x22=transpose(x2);

%Point force plate1 y
q1=-transpose(zz)*FM1(2,:);
y1=(q1(1,:)+FM1(4,:))./FM1(3,:);
y11=transpose(y1);

 %Point force plate2 y
q2=-transpose(zz)*FM2(2,:);
y2=(q2(1,:)+FM2(4,:))./FM2(3,:);
y22=transpose(y2);


rows = size(x11,1);
cols=19;
```

```
fid = fopen('grf.mot','w+','n','UTF-8');
fseek(fid,0,-1);
fprintf(fid,'grf.mot');
fprintf(fid,'\r\nversion=4.3');
fprintf(fid,'\r\nnRows=%d',rows);
fprintf(fid,'\r\nnColumns=%d',cols);
fprintf(fid,'\r\ninDegrees=yes');
fprintf(fid,'\r\nendheader');

   fprintf(fid,'\r\ntime ');
   fprintf(fid,'\t fp_1x');
   fprintf(fid,'\t fp_1y');
   fprintf(fid,'\t fp_1z');
   fprintf(fid,'\t p_1x');
   fprintf(fid,'\t p_1y');
   fprintf(fid,'\t p_1z');
   fprintf(fid,'\t fp_2x');
   fprintf(fid,'\t fp_2y');
   fprintf(fid,'\t fp_2z');
   fprintf(fid,'\t p_2x');
   fprintf(fid,'\t p_2y');
   fprintf(fid,'\t p_2z');
   fprintf(fid,'\t m_1x');
   fprintf(fid,'\t m_1y');
   fprintf(fid,'\t m_1z');
   fprintf(fid,'\t m_2x');
   fprintf(fid,'\t m_2y');
   fprintf(fid,'\t m_2z');

   line_index =1;

for k=0: (rows-1)
        fprintf(fid,'\r\n%.2f',0.01*(k+1));%time
        fprintf(fid,'\t%.3f',FM11(k+1,2));%f1x
        fprintf(fid,'\t%.3f',FM11(k+1,3));%f1y
        fprintf(fid,'\t%.3f',-FM11(k+1,1));%f1z
        fprintf(fid,'\t%.3f',y11(k+1,1));%point1x
        fprintf(fid,'\t%.3f',zzz(k+1,1));%point1y
        fprintf(fid,'\t%.3f',-x11(k+1,1));%point1z
        fprintf(fid,'\t%.3f',-FM22(k+1,2));%f2x
        fprintf(fid,'\t%.3f',FM22(k+1,3));%f2y
        fprintf(fid,'\t%.3f',FM22(k+1,1));%f2z
        fprintf(fid,'\t%.3f',-y22(k+1,1));%point2x
        fprintf(fid,'\t%.3f',zzz(k+1,1));%point2y
        fprintf(fid,'\t%.3f',x22(k+1,1));%point2z
        fprintf(fid,'\t%.3f',FM11(k+1,5));%t1x
        fprintf(fid,'\t%.3f',FM11(k+1,6));%t1y
        fprintf(fid,'\t%.3f',-FM11(k+1,4));%t1z
        fprintf(fid,'\t%.3f',-FM22(k+1,5));%t2x
        fprintf(fid,'\t%.3f',FM22(k+1,6));%t2y
        fprintf(fid,'\t%.3f',FM22(k+1,4));%t2z

end
fclose(fid);
```

H. Slope Correction Code

```
clc;
clear;
if  isdeployed()
    if (nargin<1)
        dataPath='';
        msgbox('Input datapath is empty/null');
    end
else
  dataPath ='C:\Users\selin\Desktop\1\add.txt';
    rootpath = 'C:\Users\selin\Desktop\';
end
 rot = dlmread(dataPath);

 coefficients1 = polyfit(rot(1:1000,1), rot(1:1000,2), 1);
 coefficients2 = polyfit(rot(1:1000,1), rot(1:1000,3), 1);
  coefficients3 = polyfit(rot(1:1000,1), rot(1:1000,4), 1);
  coefficients4 = polyfit(rot(1:1000,1), rot(1:1000,5), 1);

% get the slope, which is the first coefficient in the array:
slope1 = coefficients1(1);
slope2 = coefficients2(1);
slope3 = coefficients3(1);
slope4 = coefficients4(1);
new=rot;
time = size(new,1);

for i=1:time
new(i,2)=new(i,2)-slope1*new(i,1);
new(i,3)=new(i,3)-slope2*new(i,1);
new(i,4)=new(i,4)-slope3*new(i,1);
new(i,5)=new(i,5)-slope4*new(i,1);
end

xlswrite('corrected',new);
```