Estimation of lower extremity joint moments in Clinical Gait Analysis by using Artificial Neural Networks

PHATCHARAPA OSATEERAKUN

A thesis submitted in partial fulfilment of the requirements of Liverpool John Moores University

for the degree of Doctor of Philosophy

July 2021

Acknowledgements

This PhD project would not have been successful without the supervisory team, especially Professor Gabor Barton, the primary supervisor who has been directing the project for me from day zero when I only had a little knowledge about gait analysis. Thanks for allowing me to attend as a visiting scholar in January 2016 and supervising me all the way through this PhD in the later years, professionally and personally. You have inspired me that one can well be a biomechanist and an MD at the same time. I am greatly appreciated for all the supports you have given me. My appreciations are also expressed here to the second supervisor, Dr Richard Foster and the third supervisor, Professor Simon Bennett for their supports during the PhD project. Also, thanks to the volunteers of this project.

This project would not have been smoothly accomplished without the continued support from Bangkok by the Thai Red Cross Society, especially my mentor, Professor Noppachart Limpaphayom, the head of Paediatric Orthopaedics who has always supported me. The appreciations are extended to Professor Somsak Kuptniratsaikul the head of the Orthopaedics Department, Professor Pibul Ittirawiwong, Associate Professor Yongsak Wangroongsap, Associate Professor Vajara Wilairatana, Associate Professor Pravit Kittidumrongsook and Dr Jirun Apinun, all from Faculty of Medicine, Chulalongkorn University, Bangkok.

I would like to show my appreciation to Associate Professor Pairatch Prasongchin from Faculty of Medicine, Chulalongkorn University, Bangkok, my first gait analysis teacher who taught me to use observational gait analysis to help in Orthopaedic treatments and Mr. Andrew Roberts from the Robert Jones and Agnes Hunt Orthopaedic Hospital, UK, my second gait analysis teacher who showed me the modern day three dimensional gait analysis for orthopaedic treatments. Special thanks to Ms. Aree Chimklin, who has been a messenger, an administrator and a great friend.

Also, thanks to my friends Eirian, Yaya, Cindy, Tasha, Keiko, Raihana, Hannah, Elena and everybody at the lab as well as Paul's mum, Sandra Hatton and her partner Dave Howarth who have been great supporters and painted the colourful life in the UK.

Nothing is bigger than the support from my family to get me through this highest educational level. There are no words that can express all my feelings to my dad, Anan, my aunty, Warapohn and my grandmother, Sue, who had passed before I finished this project. More importantly, my mum, Somkhuan, who is the most supportive and strongest person in my world. Also, thank you very very much my sister, Anny, and my brother, BM who have always been my cushion and make sure that I would never fall down on a stony road.

Paul Hatton, my husband, I could not be more grateful, and no words can be expressed for your understanding, patience and love I have received from you during these years. You have shown me something I could never have imagined how much a man can sacrifice for his wife. Please know this PhD is yours too.

Abstract

Gait analysis is typically conducted using an optoelectronic system which is known as the standard method for motion analysis. Despite advance development of instruments related to the optoelectronic approach, there are still a few limitations of the traditional gait analysis which limit the accessibility for individuals who would benefit from the investigation. A newly developed three-dimension motion capture system, known as Inertial Measurement Units (IMU) was introduced as an option for gait analysis. The IMU system is a transportable camera-free motion capture system. This also motivated the principle of out-of-the lab gait analysis. To broaden the use of the new system, this PhD project was conducted to examine whether the system should be used confidently for clinical gait analysis.

The main purpose of this PhD project was to examine the feasibility of incorporating a machine learning method to estimate the kinetics of gait using the kinematics data obtained from an IMU system. Firstly, as pilot studies, an artificial neural network (ANN) was trained using gait data derived from the potential input signals which were signals of marker coordinates and joint angles obtained from an IMU system (Xsens) to predict joint moments of lower extremities. Promising findings were found as the ANN could reasonably predict the target joint moments. The results also showed the generalisation ability of the ANN to estimate the joint moment that it has not seen before, for instance, the ANN could fairly predict joint moments of the contralateral limb.

The Xsens system was validated against the standard motion capture system before the main estimation study of the joint moment in gait began. The results revealed that joint angles obtained from the Xsens were comparable with the optoelectronic system in the sagittal plane and less comparable in the frontal plane according to the coefficient of multiple correlation and the linear fit methods. The results from the transverse plane were non-real numbers.

The ANN was then trained using the joint angles derived from the Xsens system of three different walking speeds to predict the knee abduction moment (KAM). Gait data of 15 healthy volunteers were used to train the network. The ANN performed well, shown by small values of average normalised root mean square errors. Several methods were used to enhance the ANN performance. Due to the limited number of gait data used to train the network the randomisation of the input-target output data was performed. The results showed a remarkable improvement of the ANN performance. The best KAM estimation was found when the data of marker coordinates were used to train the ANN instead of joint angles. As few as three marker coordinates could provide sufficient information for the ANN to be trained and predict the KAM accurately. Principal component analysis was also used as input data manipulation and provided a reasonable KAM prediction.

Overall, the kinematic gait data obtained from the Xsens could be used to train the ANN to predict the KAM in healthy gait. There is a possibility to combine machine learning methods with IMU data to produce a clinical gait analysis without the restriction of the traditional motion laboratory.

Declaration

No portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Table of Contents

List of tables

List of figures

List of Abbreviations

2DGA: Two-dimensional gait analysis

3DGA: Three-dimensional gait analysis

ANNs: Artificial neural networks

ANOVA: Analysis of variance

CMC: Coefficient of multiple correlation

CP: Cerebral Palsy

EMG: Electromyography

FFANN: Feedforward neural network

GOAT: Gait offline analysis tool

GRF: Ground reaction force

HH: Helen Hayes model

HBM: Human Body Model

IMUs: Inertial measurement units

IMMUs: Inertial magnetic measurement units

KAM: Knee abduction moment

KOA: Knee osteoarthritis

LFM: Linear fit method

LJMU: Liverpool John Moores University

LSTM: Long short-term memory neural network

MFRL: Movement Function Research Laboratory

MLP: Multilayer Perceptron neural network

NRMSE: Normalised root mean square error

PCA: Principal component analysis

PIG: Plug-in Gait model

RMSE: Root mean square error

SDI: Strap down integration

SMAPE: Symmetric mean absolute percentage error

WNN: Wavelet neural network

Research outputs relating to the thesis

1. Osateerakun P, Barton GJ, Foster R, Bennett S, Lakshminarayan R (2018) Prediction of moments from movements without force platforms using artificial neural networks: A pilot test. *A poster presentation*, 27th ESMAC Meeting and Conference, Prague, Czech Republic, 27-29 Sep 2018. Gait & Posture, 65 (1). pp. 299-300. https://doi.org/10.1016/j.gaitpost.2018.06.194
2. Osateerakun P, Barton GJ, Foster R, Bennett S, Lakshminarayan R (2018) Artificial Neural Network (ANN) a potential strategy for predicting Knee Abduction Moment (KAM). *An oral presentation,* PhD Symposium 2019, John Lennon Art and Design Building, Liverpool, UK 25-26 July 2019.
3. Osateerakun P, Barton GJ, Foster R, Bennett S, Lakshminarayan R (2019) The ability of an artificial neural network to predict knee abduction moment in clinical gait analysis. *A poster presentation*, 28th ESMAC Meeting and Conference, Amsterdam, The Netherlands 23-28 Sep 2019. Gait & Posture, 73 (1). pp. 553-554. https://doi.org/10.1016/j.gaitpost.2019.07.316

Chapter 1: Introduction

For almost 40 years gait analysis has played an important role supporting, amongst others (Baker, 2007), clinicians in providing appropriate treatments for gait problems to individuals suffering from, potentially, several conditions for instance, cerebral palsy, stroke, and Parkinsonism (Wren et al., 2013; Wren et al., 2020). Three-dimensional gait analysis (3DGA) objectively quantifies gait, the most basic but effective form of human locomotion, and provides a more accurate quantification of human gait than traditional methods such as observational gait analysis or two-dimensional gait analysis.

The instruments for 3DGA have been invented and developed to help understand the fundamentals of human locomotion and give information that determines normal or abnormal walking. Baker et al. (2016) modified from Brand and Crowninshield (1981) stated four main reasons of clinically applied use of gait analysis: 1. For differential diagnosis between diseases 2. To assess the severity, extent or nature of a disease or injury 3. To monitor progression of a condition without or following intervention 4. To predict the outcome of the treatment or intervention.

In general, gait analysis typically comprises two mandatory components to indicate individual gait abnormality. First, kinematics, the study of joint movement which includes the translation and acceleration of body segments (joint angles) regardless of the forces that cause the movement. Secondly, kinetics, the study of forces (joint loading) that create the movements (Winter, 1990) which include ground reaction forces (GRFs), joint moments and joint power. Deviations of the joint movement and joint reaction force from the norm can direct gait analysts to uncover the underlying gait problems (King, Barton and Ranganath, 2017). Some joint movements are more readily recognised when one is observing abnormal gait pathology, without a need

for exact measurements (e.g. crouch gait in children with cerebral palsy). The joint loadings which cause the abnormal movement, on the other hand, are difficult to identify due to the requirement of the calculation of the resultant force between agonist and antagonist muscles acting about the particular joint (Winkelstein, 2013).

In most gait laboratories, at present, the standard GA is typically performed as 3DGA by using an optoelectronic motion capture system. After anthropometric measurements are taken, retroreflective markers are attached at particular anatomical landmarks on a subject according to a biomechanical gait model (Ferrari et al., 2008) for example the Helen Hayes (HH), six degrees of freedom (6DOF) or Human Body Model (HBM). Several charge-coupled-device (CCD) cameras arranged around the walkway capture reflections from the markers inside a calibrated three-dimensional volume, transformed to electrical signals which are processed by a software to reconstruct a model that represents the human body and are used to produce gait kinematics using anthropometric data. At the same time, the GRFs are collected via force transducers by a force plate embedded in the walkway then correspondingly transferred to the software, where joint reaction forces and joint moments are calculated. The joint moments are generally indirectly derived by inverse dynamics based on Newton's laws of motion using kinematics data, the GRFs obtained from the force transducers and the anthropometrics of the participant.

However, the standard 3DGA requires costly instruments, high levels of expertise and is typically limited to be conducted in a gait or movement centre (Cappozzo et al., 2005; Chiari et al., 2005; Leardini et al., 2005). As a result, there are some patients who cannot access such a specialist facility and miss the opportunity to receive such investigation, subsequently being deprived of the appropriate treatments for their

gait problems. Additionally, being observed during the investigation could affect spatiotemporal and kinematics of individual's gait known as observational awareness or the "Hawthorne effect"(Parsons, 1974). Therefore, the gait analysis performed under such unfamiliar conditions may not represent the genuine walk of the individual (Ardestani and Hornby, 2020).

In the past decades, alternative motion capture systems have been developed thanks to rapid advances of technology in several forms. Inertial measurement units (IMUs) have emerged as newly invented motion capture systems (Picerno, 2017). To be able to measure the movement, IMUs employ accelerometers, gyroscopes and magnetometers. They are regularly built as wearable, small size sensors which can record motion by calculating the displacement of segment orientation without requiring cameras thus providing a great potential to conduct the out-of-lab gait analysis (Washabaugh et al., 2017). Subsequently, such an unrestricted system would benefit those patients who have mobility difficulties getting to a motion laboratory. Moreover, the genuine walk can be recorded and analysed by this alternative system with no limitation of the laboratory environment.

As mentioned herein, both kinematics (joint angles) and kinetics of gait (joint moments and power) are crucial for interpreting abnormal gait patterns. While the IMU system instantly provides kinematics of gait by recording the orientation of body segments, obtaining the kinetics data outside the gait laboratory has been an issue of interest. Commercial companies and many researchers are still continuously working on how to acquire a complete gait analysis including both kinematics and kinetics when using IMUs. Forces could be collected by several methods in combination with the IMU system in order to complete gait data, for instance, by

using portable foot pressure sensors, wearable load cells or by estimating the GRFs by means of machine learning (Ancillao et al., 2018). Despite reasonable results were reported from the stated methods, none of these could be practically applied as an effective kinetic quantification for gait analysis. An estimation of the kinetics data using a machine learning algorithm, artificial neural network for instance, dominates the others due to its potential to produce the kinetic data without a need of another force recording instrument or further laboratory set up.

Amongst off-the-shelf IMU systems, Xsens Awinda has been developed as a wireless full body motion capture system that is composed of a 3D accelerometer, a 3D gyroscope and a 3D magnetometer contained within a small casing of 47 mm x 30 mm x 13 mm and only 16 g mass (Xsens, 2020). The motion trackers, therefore, fit well with the concept of out-of-lab motion analysis. The IMU system has recently been used more regularly in gait analysis (Loose and Orlowski, 2015; Al-Amri et al., 2018; Pauli et al., 2019) since the shortcomings, for example, the data drifting phenomenon of the angular motion recorded by the gyroscope which affects the accuracy of joint angle calculation and the disturbance from ferromagnetic objects around the motion tracking area affects accuracy were improved (Roetenberg, Baten and Veltink, 2007).

Artificial neural networks (ANNs) are computer algorithms that can mathematically model a complex relationship between a set of inputs and the expected outcomes (target outputs). They were created to imitate biological neural systems in order to estimate or forecast a result (Zurada, 1992). In general, an ANN is trained by the presented input to predict the target output. Backpropagation feed forward neural networks are commonly used for predicting gait data. Such systems have effectively

been used to predict GRFs form various inputs, for example, electromyography signal (EMG) (Oh, Choi and Mun, 2013; Johnson et al., 2018). There is a relationship in gait data as kinetics and kinematics are mathematically related by means of inverse dynamic equation and are biophysically connected as the kinetics are the causes to produce the movements. Theoretically, kinetics particularly joint moments should be able to be estimated by an ANN trained by component of kinematics data.

There were successful studies that predicted joint moments from an ANN trained by a variety of gait parameters (Hahn and O'Keefe, 2008; Favre et al., 2012; Mundt et al., 2019), however, to current knowledge, the prediction of joint moment in gait by using an ANN trained by joint angles obtained directly from an IMU system has not been explored.

The aim of this PhD project was to extend the usefulness of IMU systems to be practically used for out-of-lab gait analysis by complementing the kinematics from IMU system with predictive kinetics obtained by an ANN that was trained by the joint angles derived by a set of Xsens sensors.

1. The first objective of the study was to examine the feasibility of using an ANN trained by a variety of the input variables including marker coordinates obtained from a typical optoelectronic system and joint angles obtained from an IMU system (Xsens Awinda) to predict joint moments of gait.
2. The second objective was to predict knee abduction moment using the ANN trained by the joint angles obtained from the IMU system.
3. The third objective was to affirm the capability of using the inertial motion capture system in practice by validating the IMU system against the standard motion capture system.

4. The fourth objective was to determine which input variables based on the IMU data and which data pre-processing methods could provide the most effective KAM prediction.

Thesis outline

The PhD project was conducted as a series of seven studies. Chapter 1 is the introduction to overall concepts of the PhD project. A literature review follows in chapter 2 that brings up the motivation of the out-of-the lab gait analysis, the development of the inertial motion capture system including its evolution and the addition of an extended Kalman filter in order to improve the accuracy of orientation and position. The pilot studies are then reported in chapter 3 representing the very first step attempts to train the ANN to predict joint moment of gait as described in the objective one. The ability of the feedforward ANN is shown and the promising findings for this chapter were then further investigated in the later chapters. Chapter 4 is the beginning of the main study where the Xsens system was validated against an optoelectronics system to examine the accuracy of the inertial motion capture when being used in gait analysis stated as objective three which was an important part to accomplish the next study in chapter 5. The results from this study were also compared with the similarly conducted previous studies. In chapter 5, following the objective two, the joint angles obtained from the Xsens system were used as input to train the ANN. Gait data from 15 healthy volunteers were recorded and used to train the neural network. Generalisation ability of the ANN in order to predict unseen knee abduction moment was also examined. Chapter 6,7,8 and 9 contained the positive results of using different strategies to improve the ANN performance including a variety of inputs used, data pre-processing by principal component analysis and

discovering appropriate hidden neuron ratios. These chapters achieved the objective four of the study. The results from chapter 6,7,8, and 9 were integrated and discussed for the potential improvement of the generalisation ability of the FFANN model in chapter 10. Lastly, in chapter 11, general discussion and conclusion highlighted the research findings that could be, in the future, adopted to conduct out-of-the lab gait analysis which will clinically and economically be beneficial to people in need.

Chapter 2: Literature review

2.1 Clinical Gait Analysis

To be able to understand how humans walk has been acknowledged as a topic of interest since Ancient Greek era (Baker, 2007). About two millennia later, during the last decade, the number of published studies involving gait analysis has increased for almost ten times (Wren et al., 2020). Being part of human movement study, the majority of the interests involve two groups of people, firstly physicians and physiotherapists whose work is to identify abnormal movement, for example, neuromuscular disorders such as Cerebral Palsy (CP) or stroke while the other group focuses on maximising individual performance of athletes (Davis, 1988). Nowadays, it has been accepted that three-dimensional gait analysis is the gold standard to measure the fundamental way of transport and to produce quantitative data to document an individual's walking performance (Vastola et al., 2016)(Jacquelin Perry, 2010). Typically, two sets of gait data are collected and recorded: 1. Kinematics, the study that describes the body segment motion, joint angles in particular and 2. Kinetics, to describe ground reaction forces, joint reaction forces, joint moments and powers (Mayich et al., 2014). Clinically, both types of data play important roles to diagnose gait abnormality, to assess the severity of a particular gait problem, to monitor the progression of a disease and, finally, to predict prognosis of the outcome of a treatment method (Baker, 2006). Kinematics data are vital parameters for clinicians to decide appropriate treatment modality (Wren et al., 2011; Khouri and Desailly, 2017; Davids et al., 2019), similarly, kinetics data also provide crucial piece of information of how a pathology is created (Amin et al., 2004; Favre and Jolles, 2016), recommending treatment options (Barrios, Crossley and Davis, 2010;

Shepherd et al., 2020) and monitoring of the treatment outcomes (Foucher and Wimmer, 2012; Whatling et al., 2020).

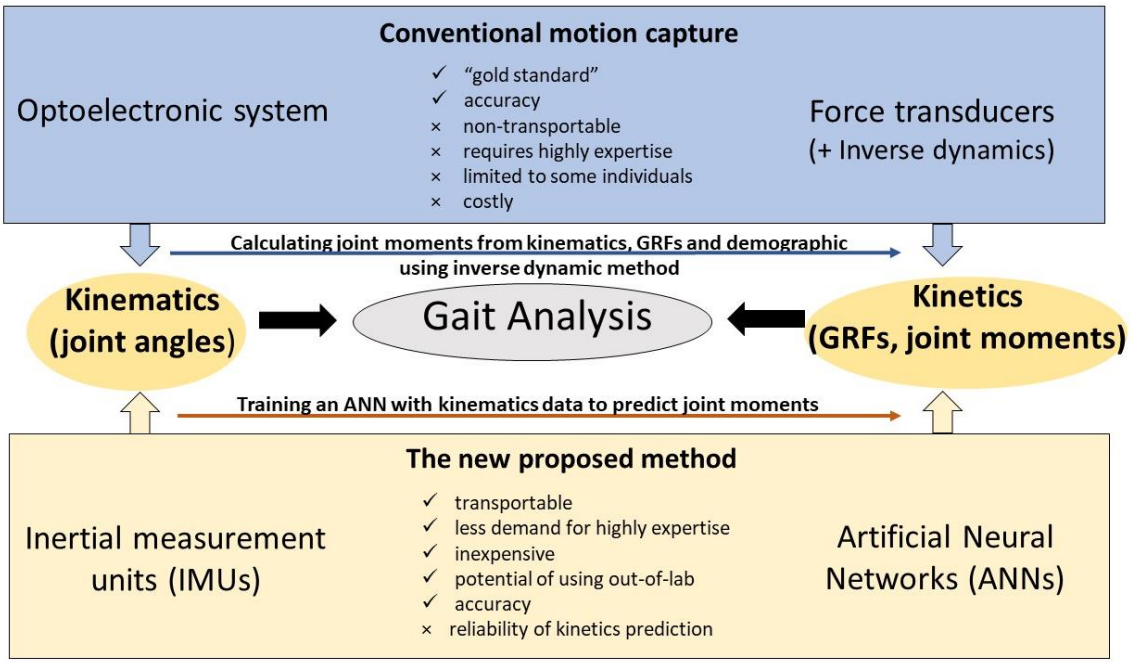

Figure 2.1 The diagram illustrates the conventional method for gait analysis compared with the method used for this PhD project. Normally, the two main components of gait (kinematics and kinetics) are obtained from the optoelectronic system incorporate with force transducers and processed with inverse dynamics. The kinematics data, however, can be derived from an IMU system and the data will be used to train an ANN to predict the joint moments.

2.2 The contribution of gait kinetics in clinical practice

During the stance phase of gait, the body weight is lowered and being transferred across the support foot thus generating GRFs in all vertical, horizontal and mediolateral directions (Jacquelin Perry, 2010). These forces can be quantified via a

floor mounted force platform before further calculated, based on Newton's laws of motion, in combination with the segment mass and its centre of gravity. As a result, joint torques (joint moments) can be computed with the method known as inverse dynamics to end up with turning moments acting about joint axes, ultimately causing rotational joint movements. Information of the joint moments in gait has been useful in clinical contexts especially the health issues related with a weight bearing joint. Abnormal knee abduction moment is considered as an etiology related to knee osteoarthritis (KOA) (Vincent et al., 2012), one of the most common degenerative diseases that affects more than 250 million people around the world (Vos et al., 2012) and costed over $330 billion in 2003 in the USA only (Yelin et al., 2007). Amin et al. (2004) reported that new chronic knee pain is related to a higher baseline KAM in the elderly. Moreover, as the medial tibiofemoral joint is the most commonly affected anatomical site in this disease, a factor related to an alteration of the KAM in KOA is the varus malalignment of the knee (van Tunen et al., 2018). The deformity creates a larger distance between the vertical GRF and the joint centre thus increasing the KAM. An excessive loading that occurs in the medial compartment of the knee could alter the biological environment and biomechanical properties of the articular cartilage, meniscus and subchondral bone leading to joint destruction, narrowing of the joint space and creating more varus deformity (Chen et al., 2020), these eventually turn into a vicious cycle. Many people who suffer from KOA, consequently modify their gait to help walking more comfortably with less pain by means of naturally reducing the KAM at stance phase. Several gait modification techniques were reported including: trunk sway, medialising the knee, walking with a wider base of gait, out toeing, and reducing the walking speed (Vincent et al., 2012). These findings have encouraged gait researchers to apply gait retraining in order to

maintain the KAM to its normal range and prevent gait deterioration of individuals (Barrios, Crossley and Davis, 2010).

In an established motion laboratory, three-dimensional gait analysis is typically conducted using an optoelectronic system. The movements are recorded from reflective markers attached to anatomical landmarks and detected by infrared cameras, and GRFs registered simultaneously from floor mounted force plates. Kinematics and kinetics of gait will then be processed by a highly trained specialist.

The limitations of the method such as providing access in only movement analysis centres, costly maintenance and the Hawthorne effect have recently inspired the concept of out-of-the lab gait analysis. The attention is currently focused on the feasibility of using a portable motion capture system to measure gait outside the traditional motion laboratory.

2.3 Inertial measurement units

Alternative methods for movement analysis, inertial measurement units, two-dimensional markerless motion capture system or motion sensing input devices such as Microsoft Kinect, have been developed in the past decade to overcome the limitations of traditional optical motion capture (Sandau et al., 2014; Müller et al., 2017). Inertial motion capture has become an outstanding system to offer a transportable motion analysis laboratory. The system comprises of a combination of multiple Inertial measurement units (IMUs) that can provide reliable joint angles computed from data obtained from its main components: 3D accelerometer, 3D gyroscope and 3D magnetometer (Poitras et al., 2019). As a result, no cameras nor

dedicated laboratory space are required for setting up such an alternative motion capture system. Such motion capture systems were firstly reported for 2D human motion analysis in the early 1990s when the sagittal knee joint angle was calculated from gait data obtained from eight uniaxial accelerometers that were attached on a PVC bracket comparing with angles obtained from a flexible goniometer that was also attached to the knee brace. Based on Newton's laws of motion and, as a 2D motion, the assumption that the distances between the joint and the sensors were constant, the knee angle in sagittal plane was calculated with standard deviation ranging between 0.04-0.09 radians compared to the angle measured by the goniometer (Willemsen, van Alsté and Boom, 1990).

The term "inertial" refers to the function of the sensor units that measure the sensor's movement or the movement of a rigid body that the sensor is attached to when the movement is accelerated by external translation forces (accelerometer) or rotational forces (gyroscope) (Picerno, 2017). Initially, IMU devices comprised of a 3D accelerometer and a 3D gyroscope (Luinge and Veltink, 2005). After the sensor-to-segment calibration, data signals from the two sensors, are integrated to compute a body segment orientation compared with the proximal body segment when an IMU is placed on each segment (Luinge and Veltink, 2005). Although the body segment orientation could be estimated from data of a gyroscope, it was proved that there was integration drift when the data were recorded for a long period of time (Luinge, Veltink and Baten, 1999). When the gravitational vector component from an accelerometer were combined to the angular velocity obtained from the gyroscope and a Kalman filter, the resultant estimated orientation showed a remarkable improvement (Luinge, Veltink and Baten, 1999). However, tracking a movement by

using data from these two sensors still inherited some integration drift by the nature of integrating data from the gyroscope (Roetenberg, Slycke and Veltink, 2007). To this extent, many developers have incorporated a 3D magnetometer in the IMU in order to help directing the sensor by following the magnetic north as the heading direction (de Vries et al., 2009). Generally, in an IMU used for motion capture, the magnetometer works as a compass needle for the system and the accelerometer works similarly to a spirit level following the gravitational acceleration. If there is no movement the linear acceleration equals to zero, and the acceleration is then equal to the gravitational acceleration. The gravitational vector is therefore used to identify pitch and roll of the body segments (Paulich et al., 2018). The system that includes a magnetometer, therefore, can be called as inertial magnetic measurement units (IMMUs). Nowadays, the inertial sensors are more likely to include all three components rather than containing only gyroscope and accelerometer (Picerno, 2017).

The combination of cumulative drifting of the data from the integration drift by the gyroscope and the disturbance of the magnetometer signals due to the effects of ferromagnetic materials has been highlighted and investigated in depth as these factors have lessened the use of the IMMUs when the accuracy and precision of the measurement is expected, for instance, in clinical practice (Picerno, 2017). Considering that ferromagnetic materials possibly exist, for example, construction iron in a building, hardware instruments and electrical appliances, the homogeneity of the magnetic field can be disturbed. In a typical indoor environment, there could be a constant magnetic interference that could cause local magnetic field deflection ranged from 12.6° (a heater/ a computer monitor) to 16.1° (a large metal shelf).

Keeping at least about 60 cm (two feet) from the sources of magnetic interference was recommended (Bachmann, Xiaoping and Peterson, 2004). De Vries et al. (2009) showed the strongest disturbance of the magnetic field at five cm above a motion laboratory floor but decreasing at the 100 cm height. However, the study showed that a Kalman filter could reduce the magnetic field distortion and the motion capture with and IMMU system was recommended at least 40 cm above the laboratory floor (de Vries et al., 2009).

Mathematical algorithms have also been adopted to overcome these two error factors in order to improve the performances of the IMMUs. The strap down integration (SDI) algorithm has been included in the data processing to solve the integration drift and improve the accuracy of the accelerometer and the gyroscope (Paulich et al., 2018). The Kalman filter is a fusion algorithm that works recursively to estimate an outcome from time series measurements that has been commonly used in navigation technologies (Sabatini, 2011). For human motion tracking, a complementary Kalman filter (figure 2.2) is adopted to estimate body segments' orientation by means of IMMUs (Foxlin, 1996; Roetenberg et al., 2005). The filter estimates the body segment orientation based on a model of errors, the estimation process is computed through four components of the sensor including the '*a priori*' model prediction of the state where the signals of the three components are integrated (the orientation changes are calculated from the angular velocity via the SDI algorithm, the error model, the Kalman filter and the state correction yielding the *a posteriori* state estimate (Roetenberg et al., 2005)).

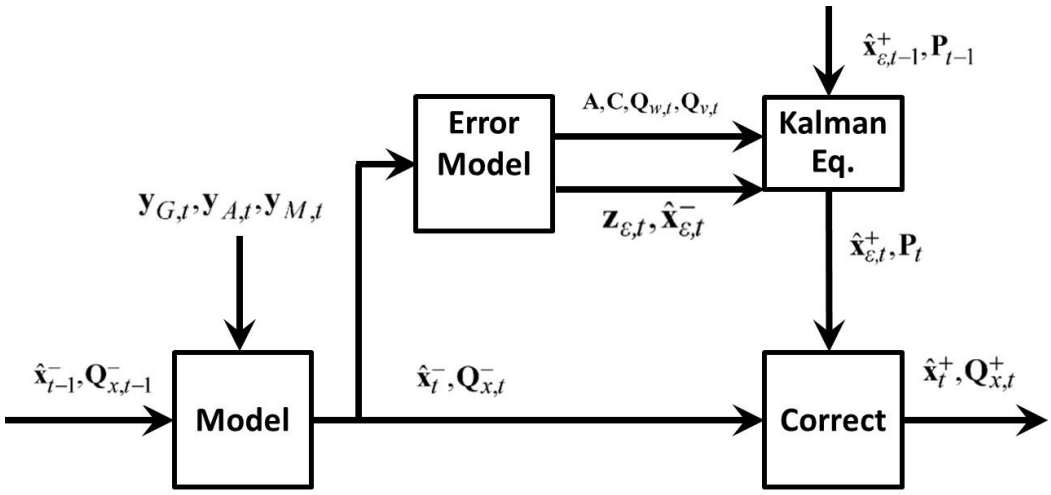

Figure 2.2 An example of a Kalman filter structure used for IMMUs (the figure was adapted from Roetenberg et al. (2005)), the first part of the filter is the state model where the models of the signal of the gyroscope, accelerometer ($y_{G,t}$, $y_{A,t}$, y_{Mt}, respectively) at a specific time are integrated with the estimated orientation state model (x) from before. The error model is then created and presented to the filter an error state vector (x_{ε}), the correlations between the previous and the current error state (A), the correlation between the error state and the real measurement, the covariances of the system ($Q_{w,t}$) and the measurement noise ($Q_{v,t}$), the error inputs from the signal generation system: the inclination and the magnetic ($Z_{\varepsilon t}$). The filter then estimates the orientation with the Kalman filter covariance (P). A hat on the top indicates estimated vector, a minus superscript is the *a priori* state and a plus superscript is the estimation made after being corrected by the filter.

The Kalman filter showed the ability to estimate the IMMU signals when it was placed close to a 3.75 kg iron cylinder at root means square of 1.4° for static condition and 2.6° for dynamic condition when compared with an optical reference system (Roetenberg et al., 2005).

2.4 Validity and reliability of inertial measurement units in gait analysis

The validity and reliability of using IMUs in gait analysis are now in the focus of interest for motion analysts following the improvement of data estimation from the three main sensors of the IMUs by incorporating the SDI algorithm and the complementary Kalman filter. Gait data obtained from the IMUs system have been

reviewed against the data obtained from a standard optoelectronic system in two main categories: spatiotemporal parameters and 3D joint kinematics. A recent systematic review and meta-analysis of studies reported the validity and reliability of using the IMUs system in order to quantify gait from healthy participants. Eighty-two studies were included for the analysis, and the researchers found that only a few were categorised as high-quality studies. Subsequently, there were insufficient number of the included studies to be pooled and analysed to conclude about the validity of the 3D joint kinematics of the IMUs. Good to excellent correlation between the two systems were found with the range of the spatiotemporal parameters (Kobsar et al., 2020).

Zhang et al. (2013) validated 3D joint kinematics calculated from a commercial IMUs system (Xsens) in three daily activities: level walking, stair ascent and stair descent in 10 healthy subjects. Focusing on level walking, the sagittal plane joint angles of the lower limbs including hip, knee and ankle joint presented the best correlation when compared with gait data collected from an optoelectronic system and derived in a standard gait model. They reported a range of the grand mean joint angle estimation errors of angles derived from the IMUs system at 1.38°-6.69°. On average, amongst all joint angles, the coefficient of multiple correlation (CMC) was 0.96 or higher for flexion and extension. In contrast, the CMC values of the frontal and transverse plane of motion ranged from 0.50-0.85. The authors yet suggested that the differences may be mainly due to the difference in the anatomical reference frame description of the compared motion capture systems which affects more the frontal and transverse planes. They concluded that caution should be taken when 3D joint kinematics, specifically in the frontal and transverse planes, are used with the Xsens system

Frontal plane data, however, are essential information for clinical gait analysis application.

Similarly, Al-Amri et al. (2018) studied the concurrent validity of 3D joint kinematics from Xsens sensors compared with those of an optoelectronic system. Gait data were collected from 27 healthy participants in three activities of level walking, squatting and vertical jumping. In terms of kinematics in level walking, the results showed excellent CMC at > 0.9 in the sagittal plane from all hip, knee and ankle joints and also excellent for the frontal plane of the hip joint. However, the results were not shown in the other planes of motion of the rest because they were not real numbers. The correlations according to Linear fit method (LFM) illustrated the relevant outcomes with the CMC. Average R^2 of the sagittal plane motion in all joints was > 0.8, while fair to good similarity were found for transverse and frontal plane with R^2 at 0.4 – 0.8. Poor similarity or the two systems were found in the frontal plane and transverse plane joint angles and in the transverse plane of hip joints. Excellent correlations were found in the sagittal plane of movement which generally has a wider range of motion. The poor relationships were, by contrast, shown in joints that have a small range of motion suggesting that the motions were difficult to be accurately computed, most likely from marker misplacement. Moreover, the difference of biomechanical model of the two motion capture systems could also contribute to the poor kinematics waveform similarity.

The experiment was carried out by two researchers who never used an IMU system: an experienced musculoskeletal physiotherapist and an experienced clinical movement scientist in order to examine the reliability of joint kinematics using an IMU motion capture system. Classified by interclass correlation (ICC), the reliability

of overall joint kinematics in walking analysis by the different researchers were acceptable at ICC > 0.6° with standard error of measurement at < 5°. They summarised that the IMUs system can be conducted independently without the requirement of a high expertise technician. The system provides reliable 3D joint kinematics to experimenters at any level of research experience. As can be seen, the IMUs can provide a reasonable 3D joint kinematics for gait analysis. The lack of kinetics is still an area of interest. Kinetics can be produced using wearable force devices such as force sensor insoles or pressure mats (van den Noort et al., 2013; Shahabpoor and Pavic, 2017). Estimating gait kinetics by means of machine learning has also become an appealing method recently (Ancillao et al., 2018).

2.5 Artificial neural network

The artificial neural network, a machine learning method, was inspired by biological neural system by its learning process which memorises and recognises patterns and relationships of data from previous knowledge and congregates from the experiences to predict a particular incident (Agatonovic-Kustrin and Beresford, 2000). Physically, biological neurons get input information or stimuli from the surrounding environment via sensory receptors delivered by sensory neurons through an interconnected network which eventually transfer the signal to the central nervous system to process and analyse before sending the response reaction back via output or efferent neurons to react with those input stimuli (Agatonovic-Kustrin and Beresford, 2000). A common and yet complex example is when temperature drops and muscles around the body start shivering. A higher level function of ANNs is that they are able to learn from (be trained by) a set of input information, as a

mathematical model, that is aiming to create a set of output information. Such ANN models have been utilised in many branches of sciences including engineering, medicine, pharmacy, and biomechanics, where typically the ANNs are used for pattern recognition, forecasting of events or data classification (figure 2.3) (Pedersen, Jorgensen and Pedersen, 1996; Agatonovic-Kustrin and Beresford, 2000; Favre et al., 2012).

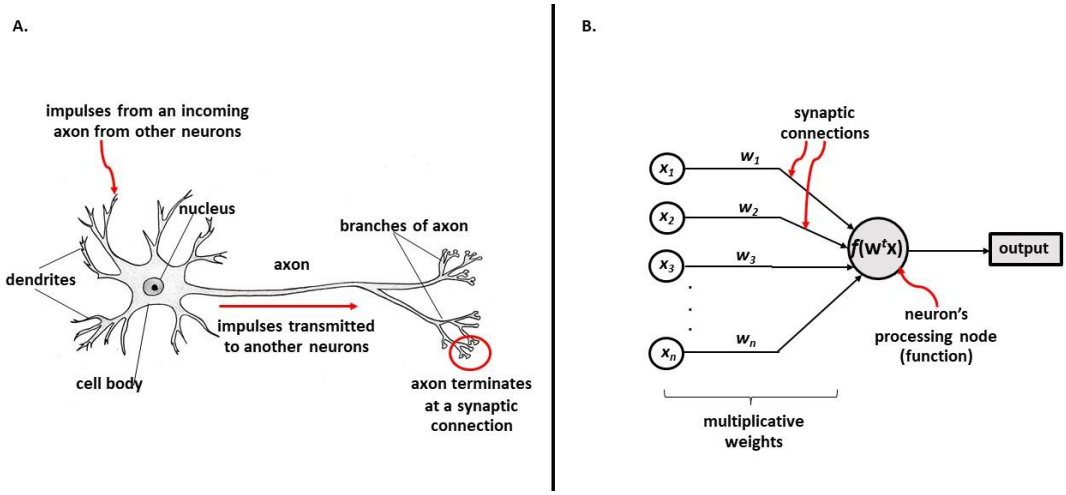

Figure 2.3 The inspiration of biological nervous system to the artificial neural network. A. a biological neuron receives some stimuli from another neuron (input) through its dendrites and transmits the impulses to the next neuron via axons at the synaptic junction (output). B. an ANN node receives some inputs (x) and process by a non-linear function ($f(w^t x)$) using weights (bias) to express and output (the figure was adapted from Zurada. (1992)).

There are two main approaches for training an ANN, supervised and unsupervised. In supervised training, an ANN aims for predicting target output form input variables by using various algorithms depending on the example input-output pair that are introduced to the ANN. This type of ANN then has to be constructed with fully interconnected neurons and comprises of at least three layers: input layer, hidden

layer and output layer (Svozil, Kvasnicka and Pospichal, 1997). The feedforward neural network (FFANN), is one of the most efficient ANN algorithms for time series data prediction (Favre et al., 2012). The difference between the input and weights, known as an error is propagated back to change the weights in order to reduce the error during the prediction to determine the most accurate predicted output (Farizawani et al., 2020). Levenberg-Marquardt algorithm is the most efficient batch mode back propagation learning. It has shown the most accurate prediction due to its ability to obtain lower mean square error than any other algorithms and also subsequently requires lower amount of computation (Zayani, Bouallegue and Roviras, 2008; Sapna, Tamilarasi and Kumar, 2012). On the other hand, the unsupervised ANN employs a different training method since the ANN requires only input variables to be presented to the network (Svozil, Kvasnicka and Pospichal, 1997). The system then organises similar patterns of the inputs and groups them together before making the decision of what specific feature can be extracted from the input variables. This type of ANN, therefore, is known to be good at data clustering and visualisation. The example of the unsupervised ANN is the Self Organising Map (SOM) (Du, 2010).

The backpropagation learning can be conducted in two different modes, pattern mode and batch mode. The former is suitable for the pattern classification as the error adjustment is accomplished after each pattern is presented to the network while the latter, the average weight is calculated only when the entire set of data are presented to the network, this mode suits nonlinear regression equation and requires less weight update which is then resulting in a faster training (Rafiq, Bugmann and Easterbrook, 2001).

Given that IMUs cannot provide gait kinetics such as GRFs or joint moments, ANNs have been one preferred method for gait analysts to be able to achieve kinetics data for a gait analysis obtained by IMUs which will benefit for the outside laboratory motion capture system. To highlight the capacity of an ANN to estimate GRFs during walking, Oh, Choi and Mun (2013) utilised an FFANN trained from 14 variables for estimating GRFs and moments in all three axes from normal gait data collected from 48 healthy participants by using an optoelectronic motion capture system. The 14 input variables were extracted by using an SOM to determine the most independent, less correlated and less similar parameters. Their FFANN architecture had three layers: one input layer with 14 input variables, one hidden layer with three hidden neurons and one output layer with six output variables.

High correlation between the predicted GRFs and the actual GRFs were shown with correlation coefficients of 0.918, 0.985 and 0.991 for the medial-lateral, anterior-posterior and vertical axis respectively. Correspondingly, the R values for ground moments were at 0.987, 0.841 and 0.898 for the sagittal, frontal and transverse plane respectively. The authors concluded that the proposed FFANN algorithm may be used instead of raw GRF data and the more complicated inverse dynamics method for calculating joint dynamics in gait analysis. Also, using an unsupervised ANN such as the SOM to reduce the dimensionality of input data showed the advantage in terms of reducing the redundant unnecessary input data which could consequently cause longer calculation time and potentially create more error (Oh, Choi and Mun, 2013).

Aljaaf et al. (2016) reported the feasibility of an ANN to predict joint moments in gait analysis directly after being trained by related input variables. They evaluated the capacity of four types of machine learning algorithms: Decision tree, Random forest,

Linear Regression and Multilayer Perceptron neural network (MLP) in order to predict knee abduction moment from gait data obtained from a group of 31 alkaptonuria patients. The gait analysis was conducted using an optoelectronic motion capture system. Joint angles of pelvis, hip, knee and ankle were normalised to 100 values in a gait cycle and used as the input variables. Initially, 12 input variables: ankle X Y Z, knee X Y Z, hip X Y Z and pelvis X Y Z, therefore from 31 participants there were 3,131 instances to be presented to the machine learning algorithms. The number of input variables were reduced in sub-experiments based on correlation analysis between inputs and the target outputs. The Decision tree and Linear regression showed higher performances compared to the others while the MLP showed the lowest performance with R^2 at 0.00009 and only 0.54 of area under the recall curve. In the second experiment, the correlation coefficients between the input and the target were calculated to determine the most correlated input-output variation. Five input variables comprised of X ankle, X knee, Z knee, X hip and Y hip showed the best correlation and were presented to the four algorithms for training again. The performance of the MLP significantly improved in the second experiment with R^2 at 0.8616 and the area under the recall curve at 0.874. The authors concluded that MLP was a reasonably good algorithm to predict the knee abduction moment in gait analysis.

A wavelet neural network (WNN), a three-layer FFANN where neurons are activated by wavelets activation functions was used by Ardestani et al. (2014) in order to predict lower extremity joint moments in post total knee arthroplasty gait. Gait data were collected from four patients with three different gait patterns: normal gait, medial thrust and walk with a walking pole, under a standard motion capture system.

The WNN was claimed to potentially have higher performance because it overcomes the disadvantage of the original FFANN such as the slowness of the system due to randomly adjusting of the initial weight at the beginning of the training algorithm. Both WNN and FFANN were trained using eight EMG signals and two GRF components as inputs to predict the following six joint moments: hip abduction-adduction, hip flexion-extension, hip external rotation, knee flexion-extension, ankle plantar flexion and subtalar eversion moment. The FFAAN could predict joint moments reasonably accurately with average normalised root mean square error (NRMSE) of only 7.70%, 8.67% and 8.25% for normal gait, medial thrust and walking pole respectively. The similarity of the prediction ability of both algorithms was shown by cross correlation values, ranged from $\rho = 0.86 – 0.98$. As expected, the WNN showed a slightly better performance than the FFANN with average NRMSE at 5.00%, 5.10% and 5.98% for normal gait, medial thrust and walking pole respectively and average cross correlation values was at 0.96.

Recently, the idea of predicting joint moments from an array of IMUs data is one step closer. Mundt et al. (2020) reported a good result of the lower limb joint angles and moments prediction from a long short-term memory neural network (LSTM) using simulated IMUs data compared with an FFANN. The results from the FFANN were more accurate than the LSTM in all planes of motion. Interestingly, adding some anthropometric data augmentation did not improve performances of the FFANNs. From this study, the FFANN outperformed the LSTM neural network, however both machine learning algorithms still showed that they were able to predict joint moments in gait with a reasonable outcome compared to the standard motion capture system. This highlights the potential of conducting gait analysis outside the

motion laboratory by using the combination of kinematics data obtained from (simulated) inertial sensors and kinetics computed from an ANN, particularly the simple FFANN.

Chapter 3: Pilot studies for feasibility of predicting knee abduction moment by an artificial neural network

Background

Standard procedures exist for calculating joint angles and joint moments of gait from positional information collected by an optoelectronic system and GRFs from force platforms. Gait analysis, therefore, is typically accomplished in a movement laboratory. An alternative system, for instance, IMMUs can also be used to quantify joint angles anywhere, however, the alternative techniques to measure GRFs or joint moments outside a laboratory are not well established.

An advantage of using IMMUs for gait analysis over the optoelectronic system is that it allows researchers to collect gait data outside a gait laboratory, providing a better opportunity for some groups of people to access such investigations (Picerno, Cereatti and Cappozzo, 2008). Concurrently, to incorporate a force measurement system for examining the causes of the movements obtained by the IMMUs is a challenge. Several methods were proposed in order to estimate GRFs thus leading to the calculation of joint moments. At first, instrumented force shoes were used to record the forces during walking (Veltink et al., 2005). The instruments combined with an IMMU system could be used for the estimation of external knee adduction moment in osteoarthritis patients (van den Noort et al., 2013). However, wearing those shoes was impractical and could interfere with natural gait due to the cumbersome electromechanical equipment (Shahabpoor and Pavic, 2017) and the shoe height (3.2 cm) and weight (1.1 kg mass each) (Koch et al., 2016). Another proposed method is using an insole pressure measurement device which is wearable therefore, serves the out of the lab gait analysis principle, however the device is less accurate compared to the former method (Abdul Razak et al., 2012).

Artificial neural networks have been introduced to predict GRFs and joint moments in both normal and pathological gait with reasonable outcomes (Hahn and O'Keefe, 2008; Favre et al., 2012; Aljaaf et al., 2016). The machine learning algorithm models a non-linear relationship between input-output pairs to predict the measured output (target) thereafter (Oh, Choi and Mun, 2013). Generally, an ANN is trained with a set of input-output pairs with parallel validation and lastly the ANN prediction ability is evaluated at the testing part by being presented with a small set of previously not seen inputs. The prediction from this testing part represents how well the ANN can predict outputs from the new input data. Several kinds of input data, for instance, marker trajectory, acceleration, and electromyographic signals were used to train an ANN to predict GRFs and joint moments in gait as well as a number of ANN architectures such as a backpropagation ANN, a wavelet ANN and a long short-term memory ANN (Oh, Choi and Mun, 2013; Ardestani et al., 2014; Mundt et al., 2019; Aljaaf et al., 2016).

3.1 Pilot study 1. Prediction of joint moment in gait using marker trajectories as input

Materials and methods

The study took place in the Movement Function Research Laboratory (MFRL) at Tom Reilly Building, Liverpool John Moors University (LJMU).

Participant preparation

A female participant, 31 years of age (body height and body mass were 1.56 metres and 50 kg) was informed with the experimental protocol and a verbal consent was given. The volunteer was contacted and given the detail of the pilot study protocol to make her decision to participate the study one week before the study was conducted. The department risk assessment protocol (appendix 1.) was strictly followed throughout the study. The participant was asked to wear a tightfitting t-shirt, a pair of tight shorts and bare feet. The knee and ankle width were measured to be used with the Plug-in Gait model (PIG) (Davis et al., 1991). Thereafter, 16 reflective markers were placed on bony landmarks including both sides of the anterior superior iliac spines (ASI), posterior superior iliac spines (PSI), knee lateral epicondyles (KNE), lateral malleoli of ankles (ANK), lateral side of both thighs (THI), lateral sides of both shanks (TIB), the dorsum of the foot between second and third metatarsal heads (TOE) and both heels (HEE) according to the PiG model, using medical grade double sided adhesive tape (figure 3.1).

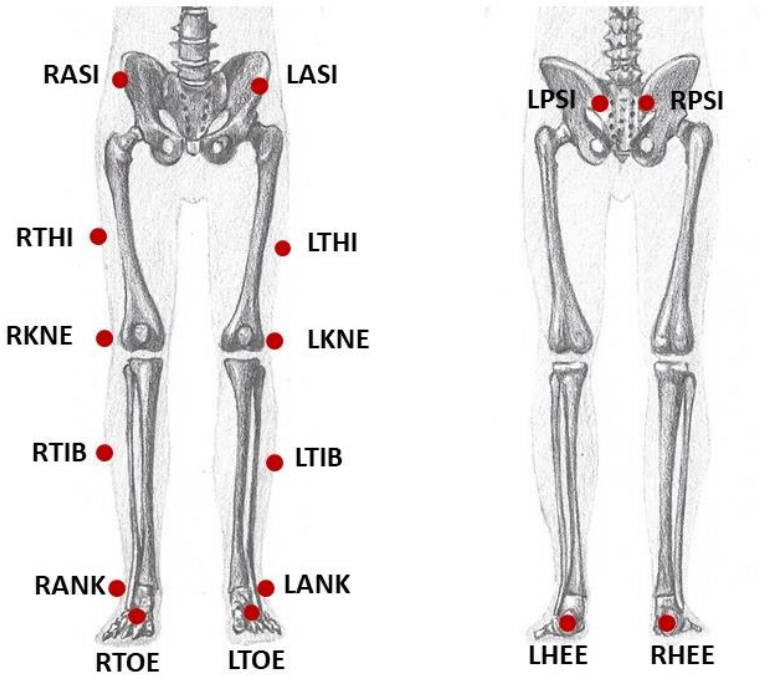

Figure 3.1 Plug-in Gait (PIG) model used in the study

Data collection, processing and analysis

The participant was then asked to walk at self-selected comforatable speed over an eight-meter walkway with two 0.6 x 0.4 m floor mounted force plates (Kistler Instrument Ltd, Winterthur, Switzerland). Five gait trials with a good contact with the centre of the force plate were recorded. Eight cameras were used for an optoelectronic motion capture system (Vicon MX T160, Oxford Metrics Ltd, Oxford, UK). The gait data were then processed with Vicon Nexus 2.5 software using the Plug-in Gait model, kinematics and kinetics data were derived and normalised to 100% gait cycle by Visual3D software (C-Motion, Inc. Maryland, USA). Input data were extracted from y coordinates of the following seven markers from the left side: LASI, LKNE, LANK, LTHI, LTIB, LTOE and LHEE creating seven input variables. Measured outputs

were sagittal plane hip, knee, and ankle moments (three output variables) from the same leg. A custom Matlab script (appendix 2.) was written for training a three layers FFANN with five hidden neurons using the Levenberg-Marquardt algorithm (Matlab2017, The MathWorks Inc, Massachusetts, USA). Data splitting was applied in the ANN training in order to evaluate the capability of the ANN model to predict the joint moments and to prevent overfitting. The recommended percentage of training data is maximally 70% of the whole data while the test set ranged from 10% to 50% (Korjus, Hebart and Vicente, 2016). In this pilot study the data were divided into 70% for training the network, 15% for validation and 15% for testing the performance of the FFANN. Thereafter, the FFANN was trained again using the same input variables and the opposite side hip, knee and ankle sagittal plane joint moments as targets (or expected outputs). Root mean square error (RMSE) was calculated to compare the measured and the predicted joint moments.

Results

The FFANN trained by data from the marker coordinates of the left side could predict sagittal plane hip, knee and ankle joint moments of the same side, with an accuracy shown by RMSE of 0.034, 0.064 and 0.046 Nm/Kg respectively (figure 3.2). In addition, the FFANN trained by the left side marker coordinates also showed the capability of predicting sagittal plane joint moments of the right hip, knee and ankle with RMSEs (calculated at the training, validation and testing part) of 0.121, 0.073 and 0.076 Nm/Kg respectively.

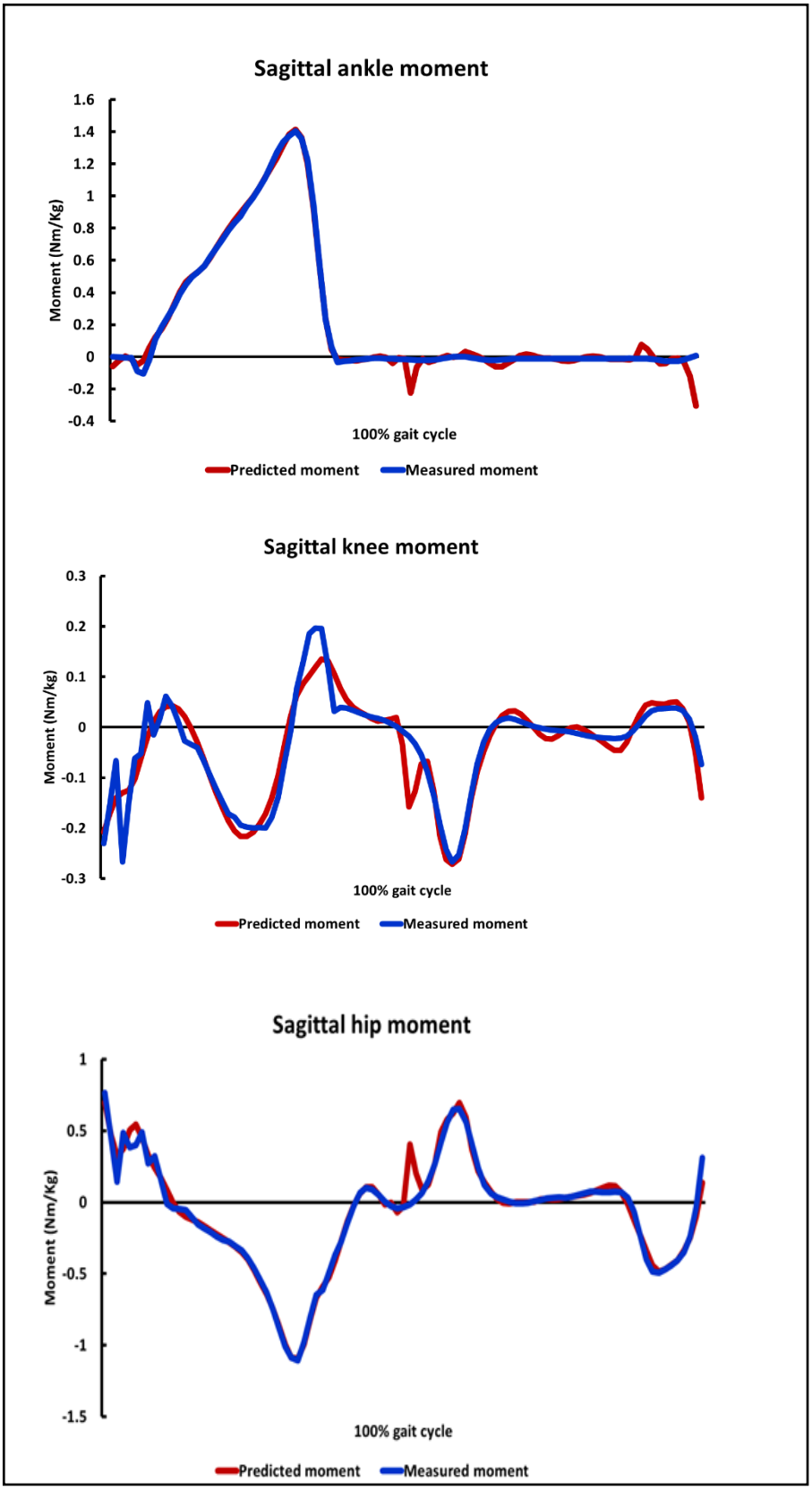

Figure 3.2 The prediction ability of the FFANN trained by data extracted from marker trajectories in unimpaired gait, with RMSEs of 0.046, 0.064 and 0.034 Nm/Kg for ankle, knee and hip sagittal plane moment. The predicted joint moments (red) of gait were comparable to joint moments of the participant calculated by inverse dynamic method (blue). A good prediction of stance phase is shown especially at hip and ankle joint.

3.2 Pilot study 2. Prediction of joint moment of gait by an FFANN trained with joint angles obtained by an IMMU system (Xsens)

Participant preparation

Two healthy participants (P1: a 27 year old male, body height 1.85 metres, body mass 102.25 kg and P2: a 26 years old female, body height 1.56 metres, body mass 53.25 kg.) volunteered to join the pilot study. Informed consent was obtained verbally after the study protocol was explained. The volunteers were contacted and given the detail of the pilot study protocol to make their decision to participate the study one week before the study was conducted. The department risk assessment protocol (appendix 1.) was strictly followed throughout the study. Both participants were asked to wear a tight fitting t-shirt, a pair of tight shorts and their own comfortable trainers. The following body dimensions: shoulder height, shoulder width, hip height, hip width, knee height, knee width, ankle height, ankle width and foot length were measured in accordance with the Xsens sensor instructions (Xsens Technologies B.V., Enschede, The Netherlands).

Data collection, processing and analysis

The gait data of each participant were collected separately on different days of the same week. Firstly, seven MTw2 wireless data trackers (Xsens Awinda, Xsens Technologies B.V., The Netherlands) were securely placed, using the elastic Velcro straps, on sacrum, lateral aspect of both thighs, anterior surface of both shins and the dorsal side of both feet over the shoe tongue. Thereafter, 26 reflective markers were placed on bony landmarks according to the HBM (lower body and trunk) (van den Bogert et al., 2013) using medical grade doubled-sided adhesive tape (figure 3.3).

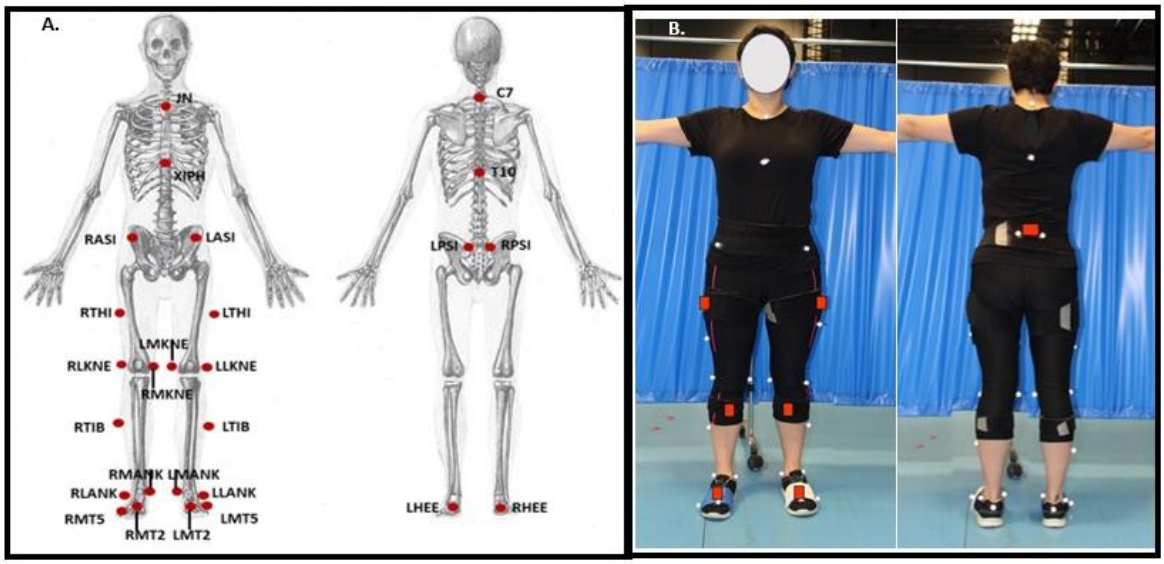

Figure 3.3 A. An illustration of the Human Body Model (HBM). B. The demonstration of the participant preparation with both motion capture systems: Xsens Awinda sensors (orange square boxes) and the HBM reflective markers.

Since the gait data were going to be collected on an instrumented treadmill (Motekforce Link, The Netherlands) the Xsens sensor calibration process was recommended to be carried out on a short distance walk. The calibration was, therefore, conducted in an unoccupied area of the laboratory close by the treadmill. The participant was then brought up to the treadmill and was fitted with a safety harness which was securely attached to the ceiling mounted cable as part of the laboratory hazards prevention protocol. The participant started to walk on the treadmill at comfortable self-selected speed using the self-paced mode of the treadmill for a short period in order to become familiar with the treadmill walk. A one-minute long gait data were then collected from each participant.

The reflective marker signals were streamed in real time from Vicon Nexus 2.5 (Oxford Metrices Ltd, Oxford, UK) to D-Flow software (Motekforce Link, The Netherlands) and were instantly processed for kinematics and kinetics of gait and

saved as text files. Similarly, joint angles were also recorded from the Xsens sensors by Xsens MVN studio software (Xsens Technologies B.V., Enschede, The Netherlands). The joint angles were estimated using the relation of a proximal and distal segment obtained from the anatomical segment frame (data from each IMMU sensor) which was compared with a priori frame of the same segments. The anatomical frame was originally calculated in a relation with the global magnetic north as its reference. As time dependent data, an appropriate beginning data point of gait data recorded by the two system was identified by the curve registration technique for identifying an event in gait described by Sadeghi et al. (2000). A mean registered curve was calculated from five consecutive gait cycles of the gait data obtained from both systems. The mean stance phase was calculated. The data from the two motion capture systems were aligned with the mean registered curve and the peak angle matching was then identified as a cut off for the identical data point and time recorded by both systems.

The recorded gait data of each participant were then extracted in Matlab 2017 software (The MathWorks Inc, Massachusetts, USA) beginning with 1,000 consecutive data points of eighteen input variables from hip, knee and ankle joint angles (all three planes of motion from both legs, i.e. 2 legs x 3 angles x 3 planes = 18 variables) derived from the Xsens system, followed by the corresponding 1,000 data points of left hip abduction moment (normalised by body mass) derived from HBM in D-Flow software as measured output. The FFANN was trained using input data from P2 to predict the hip abduction moment of the same participant with the Levenberg-Marquardt algorithm (Matlab2017, The MathWorks Inc, Massachusetts, USA), and 10 hidden neurons. The data were divided into 70% for training the network, 15% for

validation and 15% for testing the performance of the FFANN. Thereafter, the trained FFANN was used to predict the hip abduction moment of the other participant (P1). The prediction ability of the FFANN was quantified by the RMSE between the measured and the predicted output.

Results

The FFANN showed the accurate prediction of P2 hip abduction moment when it was trained by the joint angles obtained by the Xsens sensor with the RMSE of 0.17 Nm/Kg at the validation and testing part which were the gait data that the FFANN did not see during training (the unseen data) shown in figure 3.4. However, poor performance was found when the FFANN trained by data from P2 was used to predict the hip abduction moment of the other participant (P1) with an RMSE of the unseen part of 0.67 Nm/Kg (figure 3.5). The results suggested that the FFANN could not predict an unseen set of data unless it was trained by the data of that particular participant.

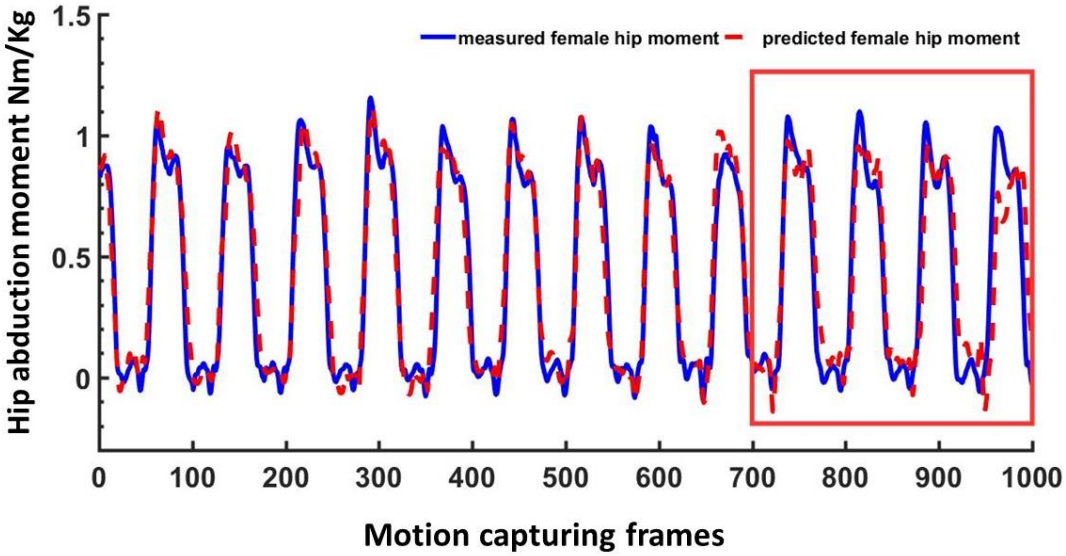

Figure 3.4 The FFANN trained by joint angles of P2 obtained by Xsens system could accurately predict hip abduction moment of the same participant. The figure also shows the generalisation ability of the FFANN to predict the unseen gait data used for validation and testing (red box) with RMSE 0.17 Nm/Kg.

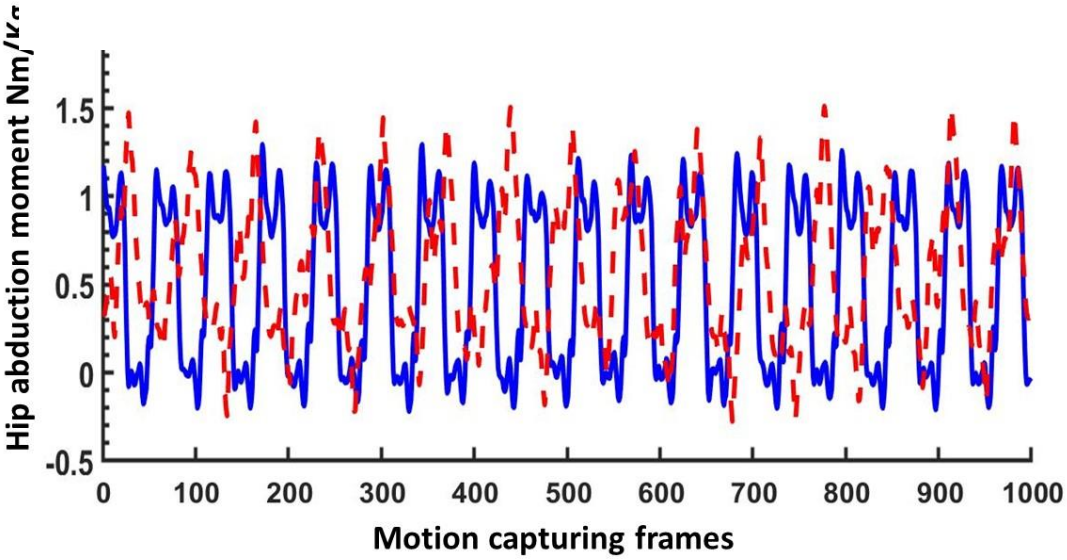

Figure 3.5 The FFANN trained by joint angles of P2 obtained by Xsens system was not able to predict hip abduction moment of the other participant (P1), thus reflecting poor generalisation ability to predict unseen gait data.

3.3 Pilot study 3. Prediction of joint moment of another participant by an FFANN trained with gait data of a different person

Data collection, processing and analysis

The data used in this pilot study were the gait data of P1 and P2 from pilot study 2, in combination with gait data obtained from a third 32 year old male volunteer (P3, 1.78 metres and 65.75 Kg). Gait data of P3 were recorded, processed and extracted in the same fashion with P1 and P2 as described in pilot study 2. Hip, knee and ankle angles from both sides of the three participants were used as input to train the network while the sagittal plane moments (normalised by body mass) of both sides hip, knee and ankle were used as target.

The pilot study was divided into two steps. First, the FFANN was trained with the gait data split into 80% for training, 10% for validation and 10% for testing the performance, with 10 hidden neurons. Data from two participant (P1 and P2) were used for training the FFANN while the gait data of P3 were used for validation and testing the FFANN performance. Therefore, the FFANN was never trained with the data of P3 before, thus making these data unseen by the FFANN. On the other hand, in step two, the FFANN was trained by gait data from all three participants with a smaller proportion of the data from P3 compared to P1 and P2 (demonstrated by a smaller and fader letter in the right block diagram (figure 3.6). The FFANN was validated and tested by the data from P3 only. At this step, the gait data of P3 were presented to the FFANN before which might affect the performance of the FFANN. Normalised root mean square error (NRMSE) to 100% was calculated to examine the performance of the FFANN.

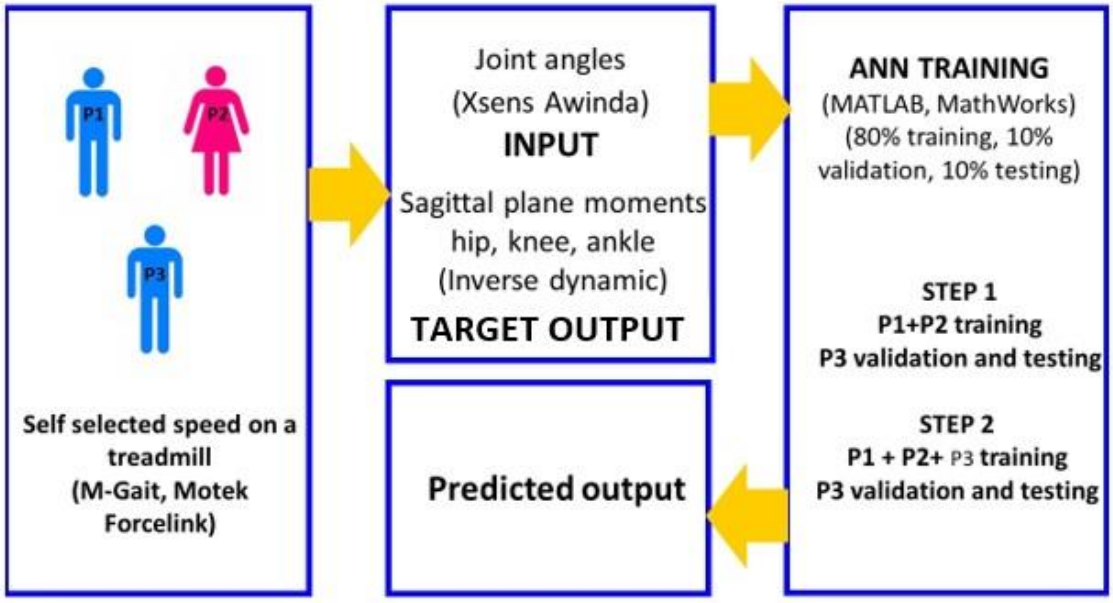

Figure 3.6 Pilot study 3 method illustrated by a block diagram. The generalisation ability of the FFANN was examined from the two steps of the training process. Step 1, the FFANN was trained by gait data of P1 an P2 and tested by gait data of P3. Step 2, some gait data of P3 were included in training, followed by testing with all gait data of P3.

Results

The FFANN was able to predict unseen data of P3 when it was trained by the input-output pairs of P1 and P2 shown by the NRMSEs at 12%, 19% and 24% for the right hip, knee and ankle sagittal moment respectively and 11%, 40% and 20% for the left hip, knee and ankle sagittal moment (figure 3.7). Moreover, the FFANN showed a better performance when approximately 17% of P3 input-output pairs were included in training, compared to the FFANN trained only by data from P1 and P2. The NRMSEs were 6%, 10% and 11% for the right hip, knee and ankle sagittal plane moment and 7%, 13% and 11% for the left side hip, knee and ankle sagittal plane moment respectively (figure 3.8).

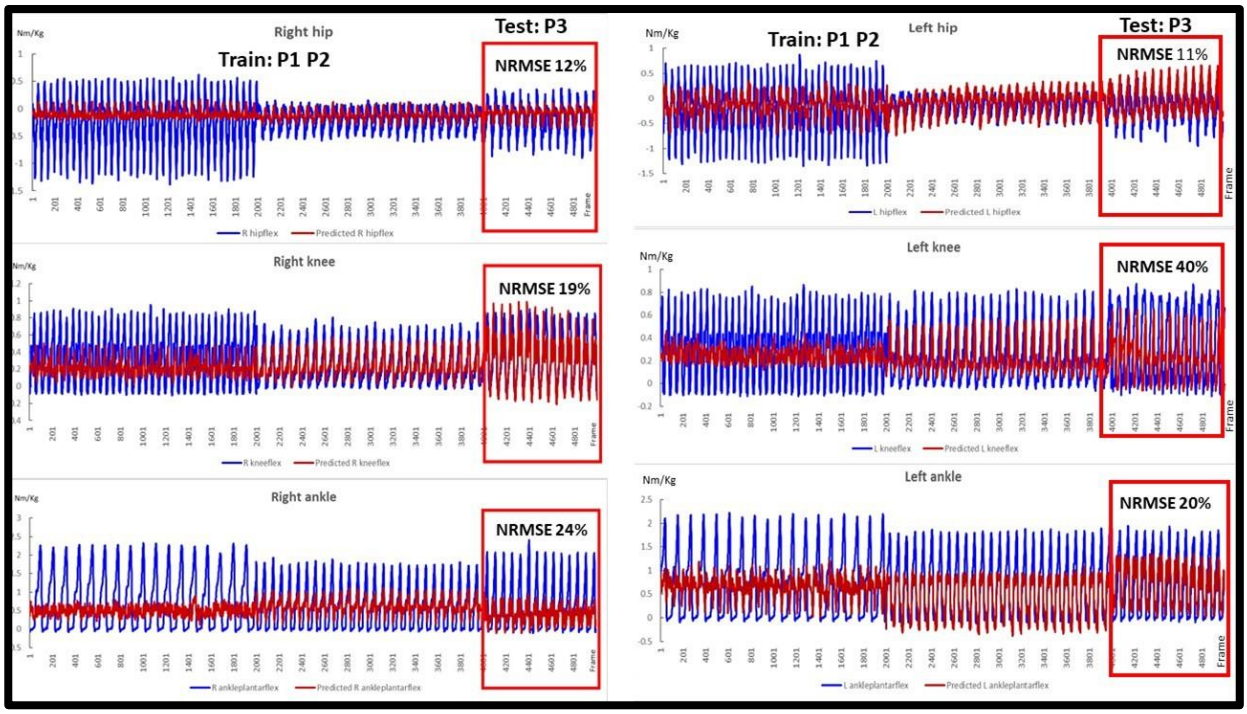

Figure 3.7 The poor performances of the FFANN in the prediction of sagittal plane joint moments of an unseen individual are shown in the testing part (red box). This indicates that the FFANN trained by data of P1 and P2 was not sufficient to cover the gait pattern of P3 and as a result the FFANN could not provide an accurate prediction of joint moments.

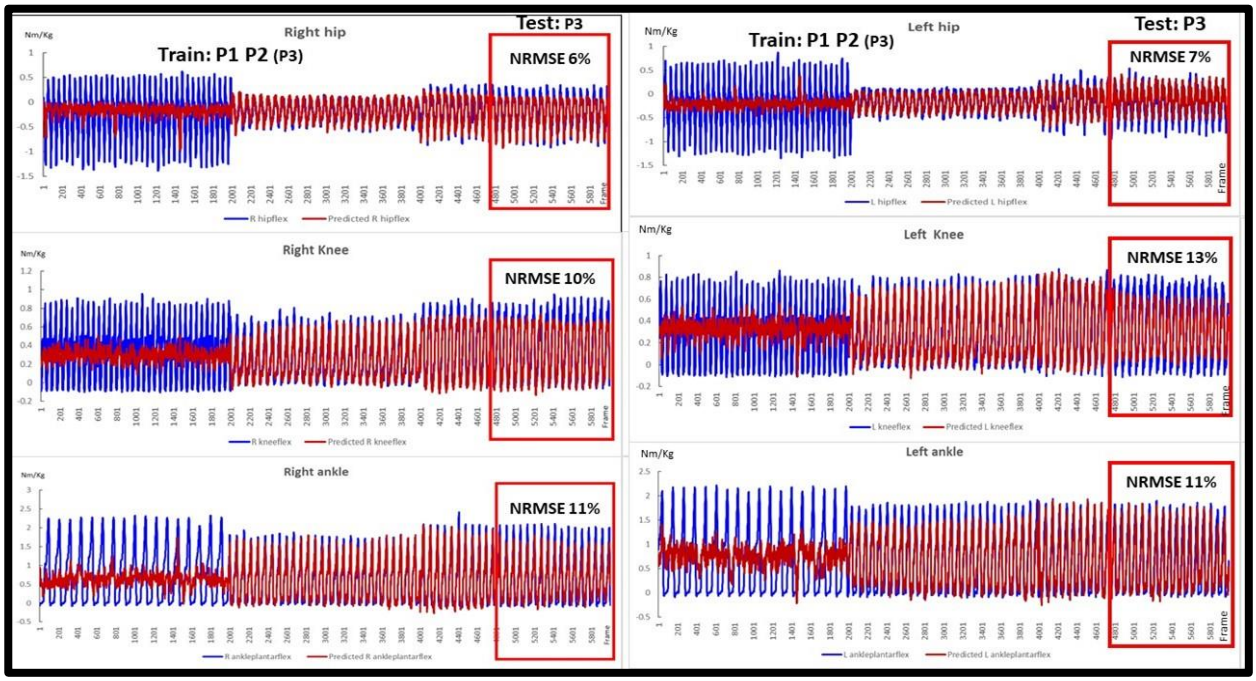

Figure 3.8 Better performance of the FFANN was shown when some data of P3 were included in training. Subsequently, the FFANN could predict more accurately the joint moments during testing with unseen data, shown by lower NRMSEs.

Discussion

The pilot studies show that the FFANN trained with Levenberg-Marquardt algorithm was capable of predicting joint moments during gait whether the input data were obtained from marker coordinates or the joint angles obtained from Xsens sensors were used as inputs. This demonstrated the relationship between the inputs (kinematics) and outputs (kinetics) well and can possibly imply that the relationship would come from the classic: the movements (joint angles) are created by the forces (joint moments) (Winter, 2009).

The FFANN trained by marker coordinates – joint moments pairs obtained from one lower extremity could predict joint moments of the same side fairly accurately, shown in pilot study 1. More importantly, the FFANN could also predict the joint moments of the contralateral limb when it was presented with input variables obtained from the opposite leg. Thus, indicating that the FFANN had generalisation ability to predict the output that was not directly related with the input data. Also, the data obtained from marker coordinates appeared to be a reasonable input to train an ANN to predict joint moment of gait, as they represent the original gait data that expressing the exact position of a particular body segments without much data processing for gait analysis (Federolf, 2013; Federolf, 2016). From the promising results of the pilot studies, later in this research, the marker coordinates will be used as input to train the FFANN and compared with training the FFANN by joint angles.

Similarly, the FFANN trained by the Xsens joint angles as inputs showed a good prediction of joint moments in both frontal plane and sagittal plane. The FFANN did not generalise well in pilot study 2 as the trained FFANN from P2 gait data provided a poor prediction of P1 hip abduction moment even though the joint moments were

normalised to body mass. This could be explained by the extreme difference of anthropometric parameters including the body height and body mass between P1 and P2 which could cause large differences in their individual gait patterns due to the differently developed gait habits thus creating the distinctive joint reaction forces.

The generalisation ability of the FFANN was explained better with the results from pilot study 3 when gait data of a third person were used to test the network. Similar to pilot study 2, the FFANN trained by data of P1 and P2 could not generalise well to predict the joint moments of P3 as seen by a high NRMSEs of the testing part of the FFANN. In contrast, the FFANN showed better performance when a small part of P3 gait data were presented to the network during training, shown by lower NRMSEs.

The success of using joint angles obtained from the Xsens sensors to predict joint moments in pilot study 2 and pilot study 3 supported the principle of conducting gait analysis outside a gait laboratory. The results were promising for the clinical application in the future which will consequently provide opportunities for firstly, the people who needed the investigation but being restricted by any reason such as independent ambulation.

Secondly, the researchers could be able to collect a genuine gait in real life that represents the typical walk of an individual. Nevertheless, more gait data obtained from a variety of gait patterns and walking speeds are required to train the FFANN to increase chances for the network to be able to recognise a wider range of possible gait patterns of the general population.

Moreover, gait data from either healthy and pathologic gait with different physical configurations should also be included to train the FFANN to provide a better chance

for the FFANN to generalise to an unseen gait pattern of a new individual resulting in the effective prediction and the achievement of conducting an ideal gait analysis outside of the gait laboratory (Stetter et al., 2020).

Chapter 4: Validity of Inertial Measurement Units for lower extremity joint kinematics in gait analysis

Background

There are several advantages of using an IMU system in order to quantify gait compared with the optoelectronic motion capture that is typically used in a standard gait laboratory, especially the possibility of using the inertial system to conduct a gait analysis outside the lab. Additionally, in situations where the real time gait kinematics are required, as the gait data collected using the traditional method needs processing which takes longer time for the data to be readily used (Picerno, 2017). The IMU system, was therefore validated against the optoelectronics system to test the capability of the alternative motion capture for practically use.

The performance of Xsens sensors, a commercial IMU system, (Xsens Technologies B.V., Netherlands) for CGA has been validated against standard motion capture. Cutti et al. (2010) applied the 'OUTWALK' protocol with the Xsens system to compare against the Calibrated Anatomical System Technique ('CAST') gait model recorded by an optoelectronic motion capture system in order to measure gait kinematics. They found that both pairs of the protocols and motion captures could be used interchangeably with very good coefficient of multiple correlation (CMC) at > 0.88 for sagittal plane movement of hip, knee and ankle joints and the frontal plane movement of the hip (CMC: 0.65-0.75 = moderate, 0.75-0.85 = good, 0.85-0.95 = very good and 0.95-1 = excellent) (Ferrari et al., 2010).

Similar results were published by Zhang et al. (2013) and Al-Amri et al. (2018), without using the 'OUTWALK' protocol, the latter groups showed that Xsens accurately provided hip, knee and ankle flexion/extension angles at CMC 0.98, however, the CMCs were lower in the frontal and transverse planes (ranged from 0.5-0.85).

It is difficult to compare the results of those hallmark studies because the experiments were conducted using different equipment, gait models and movement activities. Since joint angles obtained from Xsens Awinda were used entirely in our study, it was necessary to validate the kinematics obtained from the system against the conventional gait model (CGM) (Kadaba, Ramakrishnan and Wootten, 1990; Davis et al., 1991) with an optoelectronic motion capture system (Vicon Motion Systems, Oxford Metrics Group Ltd.) which are normally used in our gait laboratory.

This study highlighted the validity of using Xsens Awinda system in order to measure kinematics of lower extremity in gait analysis compared to the standard optoelectronic motion capture system.

Materials and Methods

The study took place in the Movement Function Research Laboratory (MFRL) at the Tom Reilly Building of Liverpool John Moores University (LJMU).

Research participants

Participants were recruited from staff and students in the Faculty of Science, LJMU. The study was approved by the university research ethics committee (reference number 18/SPS/005). Ten volunteers participated in this study (six females and four males, age: 27.9±5.32 years, body height 1.69±0.11 m, body mass 63.95±10.8 kg). Written informed consent was obtained prior to data collection.

Participant preparation

At the beginning, body mass and body height were recorded. The following anthropometric data were measured from both sides using a tape measure and an anthropometer: hip height, hip width, ankle height, knee height, foot length, shoe thickness, shoulder height, shoulder width and arm span from each participant according to the instructions by Xsens (Xsens, 2020). Ankle width and knee width were also recorded for joint centre calculations in the PiG model.

Figure 4.1 An Xsens MTw-2 tracker, dimensions 47 x 30 x 13 mm.

Firstly, seven MTw-2 data trackers (figure 4.1) (Xsens Technologies B.V., Netherlands) were attached to a participant at the sacrum, lateral aspect of both thighs, the medial surface of lower legs using elastic Velcro straps following the Xsens Awinda user manual (Xsens, 2020). The feet trackers were placed at the dorsum side over the shoe tongue and were secured by self-adhesive elastic bandage. Thereafter, 16 reflective markers were placed on bony landmarks according to the PiG lower body model (Davis et al., 1991) shown in figure 3.1. The reflective markers were placed after the Xsens sensors to prevent them from being knocked and dropped off from their positions when the MTw-2 and Velcro straps were applied.

Data collection

Kinematics data were recorded at 120 Hz by 12 Vero cameras and Vicon Nexus 2.5 software (Vicon, Oxford Metrics Group Ltd.) and at 100 Hz by Xsens MVN analyse 2018 (Xsens Technologies B.V., Netherlands). Both motion capture systems were synchronised using a trigger start signal sent from Xsens MVN Analyze 2018 via two BNC cables (appendix 3).

The gait data was collected while a participant was walking on a split belt treadmill (M-Gait, Motekforce Link, Netherlands). Therefore, the Xsens sensors were calibrated in an unoccupied area of the MFRL to allow the participant to have a small walk following the Xsens calibration protocol. This short walk is crucial for the software to examine and stabilise ferromagnetic signals around the experimental area, however, due to the limited space on the treadmill the volunteers were asked to walk on the level floor close by following the recommendations from Xsens. Thereafter, the participant was then brought up to stand in the centre of the treadmill, ready to start a walking trial. A safety harness was fitted to the participant which was securely connected to a safety hook and attached to the laboratory ceiling mounted cable according to the risk assessment scheme of the MFRL. The treadmill was operated via D-Flow software (Motekforce Link, Netherlands).

Once the safety harness was securely put on the participant and attached firmly to the hook, the calibration for the PiG model was performed whilst the participant was in a T-pose: the position where the participant was asked to stand still with both feet close together and both arms horizontally outstretched. The position was chosen as it was suitable with the treadmill environment to allow the Vicon system to visualise

the reflective markers thoroughly, avoiding markers getting obscured by the handrails.

Thereafter, the participant was asked to walk on the treadmill at their self-selected speed for a few minutes to get familiar with the treadmill gait. A few walking speeds were adjusted by the researcher for the participant to individually identify the most comfortable speed similar to their usual walk in daily living. Once the normal (self-selected) speed was determined, gait data were collected at three different speeds for replicating the heterogeneity of gait: normal, fast (40% faster than normal speed) and slow (40% slower than normal speed) (Orendurff et al., 2004; Fukuchi, Fukuchi and Duarte, 2018). The treadmill was stopped only when the speed needed changing or a marker fell off.

At each walking speed, the participant was asked to walk continuously on the treadmill, however, sets of gait data (45 seconds long) were recorded by the Xsens and Vicon systems simultaneously without stopping the treadmill. During data collection, the participant was regularly asked if there was a loosening of the sensor or marker detachment.

Data collected by the Vicon system were automatically labelled and reconstructed using the PiG model, the gait data were then exported as a c3d file for further processing with Visual3D v6 (C-Motion, MD, USA). The data were filtered with a 6 Hz low pass second order Butterworth filter. Five continuous gait cycles were identified using the GRF signal at the initial contact of a gait cycle thereafter following five consecutive gait cycles were included. The selected gait data were normalised to 100% for all three walking speeds. Kinematics data of each gait cycle were computed and recorded for the validation.

For Xsens, the segment orientations were determined firstly, relative to the initial calibration pose. The change of body segment postures was then subsequently calculated to determine joint angles from the orientation of the body segment (Al-Amri et al., 2018). In this study, joint angles processed by the Xsens MVN Analyze 2018 were exported in .mvnx files. The data were upsampled to 120 Hz using the spline function in Matlab (Matlab 2018b, The Mathworks Inc., MA, USA) to match the sampling rate of the gait data recorded by the Vicon system. Subsequently the relevant gait cycles of each set of gait data from the Xsens (the corresponding gait cycles with the gait data obtained from Vicon system) were extracted and also normalised to 100% of the gait cycle using a custom Matlab script.

Hip, knee and ankle joint angles of both legs from each gait cycle derived from the Xsens and the Vicon system were compared to validate the gait kinematics from the Xsens biomechanics model against the PiG model. The new formulation of CMCs (Ferrari, Cutti and Cappello, 2010) were calculated in another Matlab script (appendix 4).

Coefficient of multiple correlation

The computational method to examine the similarity of waveforms was originally described by Kadaba et al. (1990). In this recent study, a Matlab script for CMC calculation based on the original Matlab script from Ferrari, A., Cutti, A.G. and Cappello, A. (2010) was written to evaluate the validity of the Xsens system compared to PiG model. The formula was adapted from the original to be used for waveform similarity determination between different gait protocols when the effect of the protocol on the waveform similarity is the only interest. The kinematics data were collected continuously and synchronously between P methods (protocols), G gait cycles at F

frames were included for analysis. Following are the steps to describe how the CMC were calculated in this study (Ferrari, Cutti and Cappello, 2010).

1. Find variability between data at a particular frame of a gait cycle in a protocol (Y_{gpf}) and the mean value of the data at exactly the same frame of all protocols ($\bar{Y}_{gf}$). Sum the variability for all protocols and all gait cycles together and normalise by the degree of freedom due to the change of F from one gait cycle to the next.
2. Find the variability between Y_{gpf} and the grand mean ($\bar{Y}_g$), where $\bar{Y}_g$ represents the average value of the gait data of all protocols of a gait cycle. Sum the variability and normalise by the degree of freedom.
3. Take the square root of 1 minus the ratio of step 1 over step 2.
4. The similarity of the waveforms was quantified as CMC value between 0.65-0.74 = moderate, 0.75-0.84 = good, 0.85-0.94 = very good and 0.95-1 = excellent (Ferrari et al., 2010; Zhang et al., 2013).

$$CMC = \sqrt{1 - \frac{\sum_{g=1}^{G}\left[\sum_{p=1}^{P}\sum_{f=1}^{F}\left(Y_{gpf} - \bar{Y}_{gf}\right)^2 / GF_g\,(P-1)\right]}{\sum_{g=1}^{G}\left[\sum_{p=1}^{P}\sum_{f=1}^{F}\left(Y_{gpf} - \bar{Y}_{g}\right)^2 / G\left(PF_g - 1\right)\right]}}$$

$$\bar{Y}_{gf} = \frac{1}{P}\sum_{p=1}^{P} Y_{gpf}$$

$$\bar{Y}_{g} = \frac{1}{PF}\sum_{p=1}^{P}\sum_{f=1}^{F} Y_{gpf}$$

The Linear fit method

Despite the CMC being one of the extensively used indicators to quantify gait waveform similarity, researchers have been made aware of its varied reliability which depends on related factors such as marker misplacement and the range of motion (ROM) of interested joints where the CMC provides more accurate waveform similarity for a joint with larger ROM (McGinley et al., 2009; Røislien et al., 2012). The linear fit method (LFM), with its rather simplified calculation steps, provides more reliable gait waveform comparison by finding a linear relationship (Y_a) between two different gait curves (kinematics) e.g. a joint angle derived from two different gait protocols (Iosa, Cereatti and Cappozzo, 2009; Iosa et al., 2014). Three parameters from the LFM equation explain the similarity of the two waveforms as followed.

$$Y_a = a_1 \cdot P_{ref} + a_0$$

Y_a is the linear function which approximates P_a values (gait data from the new measure protocol at a time point) by means of a linear transformation of P_{ref} (gait data from the standard protocol at the corresponding time point).

According to the linear function, the slope a_1 is the angular coefficient (amplitude scalar factor) which indicates the mean difference between each data point from both protocols. It expresses the quantity that is required to be multiplied to match P_{ref} to P_a. On the other hand, a_0 is the intercept of the fitting line to add on to the equation for the value of P_a when P_{ref} is zero. Also, coefficient of determination (R^2) is the goodness of fit that determines the strength of the linear relationship between P_a and P_{ref} by quantifying the proportion of variance in P_a that can be matched with P_{ref}. It is a parameter for the shape similarity of the waveform. The value coincides

with the square of the Pearson's correlation coefficient of the waveforms (Iosa et al., 2014). The goodness of fit was classified as excellent if $R^2 > 0.75$, fair to high if $R^2 > 0.4$-0.74 and poor if $R^2 < 0.39$.

Results

The average walking speeds were 1.1±0.23 m/s, 1.54±0.32 m/s and 0.66±0.14 m/s for normal, fast and slow respectively. The CMCs were similar for the right and left side of the body. Sagittal plane angles show the best similarity of waveforms between both motion capture methods compared to frontal and transverse plane. This pattern is consistent amongst data from all participants and all speeds. Walking at fast speed shows the highest correlations between waveforms followed by normal and slow speed. Poor to moderate agreement of the waveforms were found in the frontal plane joint angles from all joints at all speeds. Shown in figure 4.2, the mean CMC of the sagittal angle of the knee joint was 'excellent' at 0.96 (ranged between 0.86-0.99), followed by the hip joint at 0.88 (ranged between 0.55-0.99) and the ankle joint at 0.77 (ranged between 0.04-0.96). The mean CMCs of the joint angles in the frontal plane were lower than the sagittal plane angles with the highest mean CMC at 0.62 (ranged between 0.09-0.94) for the ankle joint followed by the hip joint at 0.58 (ranged between 0.05-0.93) and the knee joint at 0.51 (ranged between 0.007-0.72). The CMC values of the waveform similarity in transverse plane could not be reported since they were non-real numbers. Example waveform similarity of the three walking speeds from a participant are shown in figure 4.3 and 4.4 for the sagittal and frontal plane respectively.

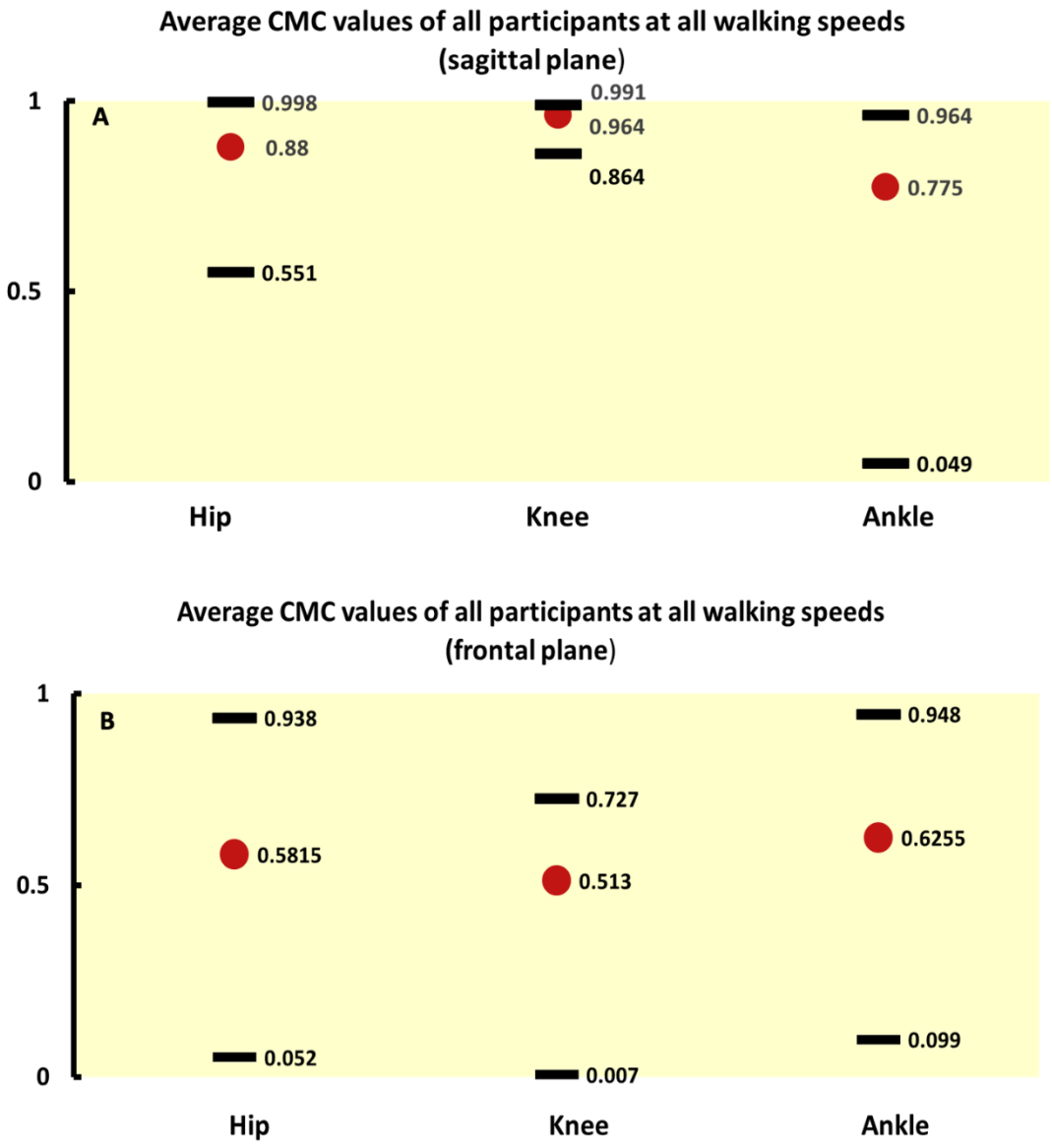

Figure 4.2 The average coefficient of multiple correlation of all walking speeds are generally higher for the sagittal plane joint angles (A) compared to the frontal plane (B). Overall, the similarity of the waveforms in sagittal plane obtained from Xsens and Vicon system vary from good to excellent, however, there is poor similarity of the waveforms in frontal plane. There is a remarkably wide range of the CMCs in the frontal plane joint angles which varied from poor to excellent.

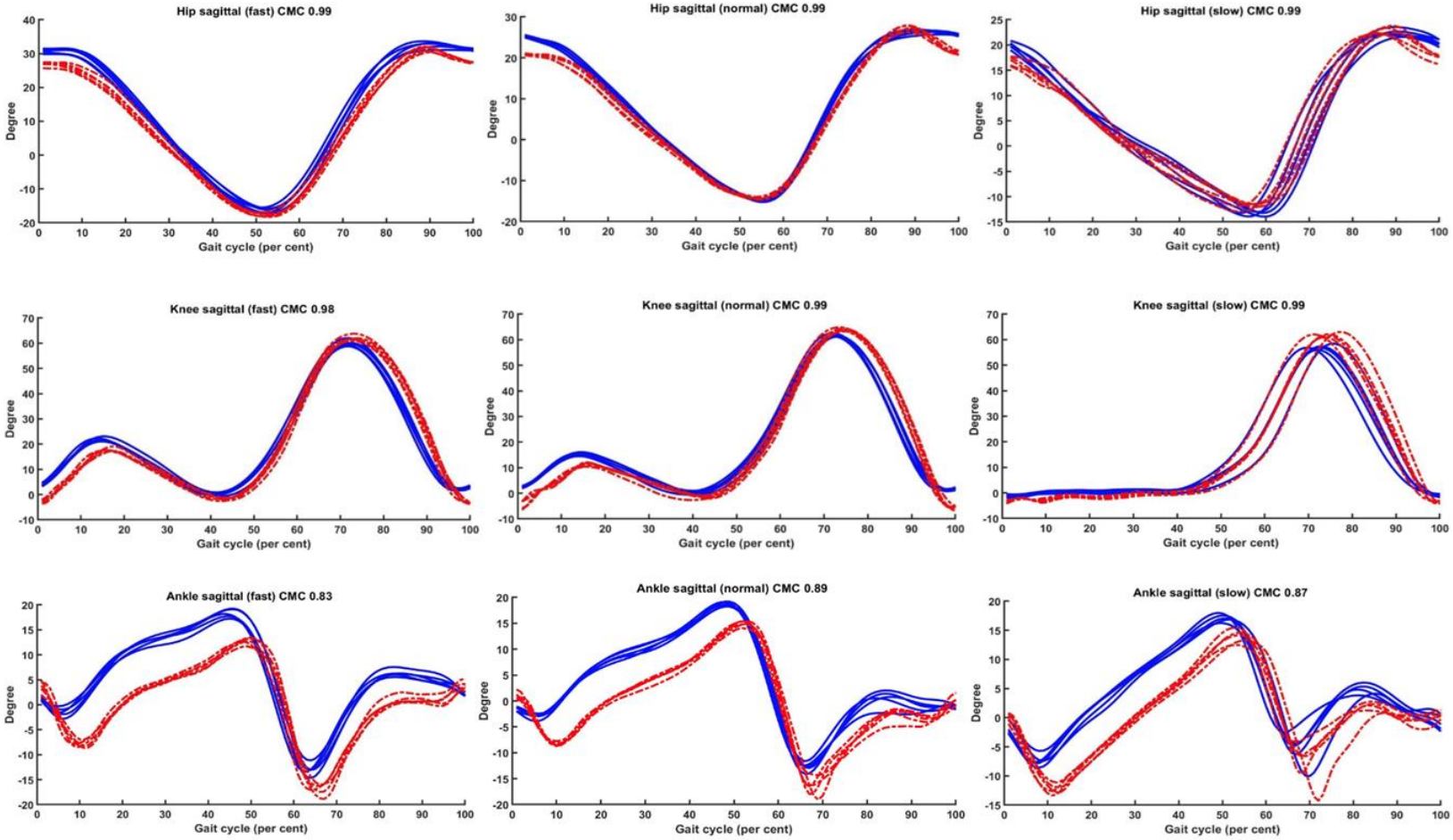

Figure 4.3 Waveform similarity and CMC of joint angles (sagittal plane) of a participant obtained from Vicon system (blue) and the joint angles obtained from the Xsens sensors (red) at fast (left column), normal (middle column) and slow speed (right column) of walking.

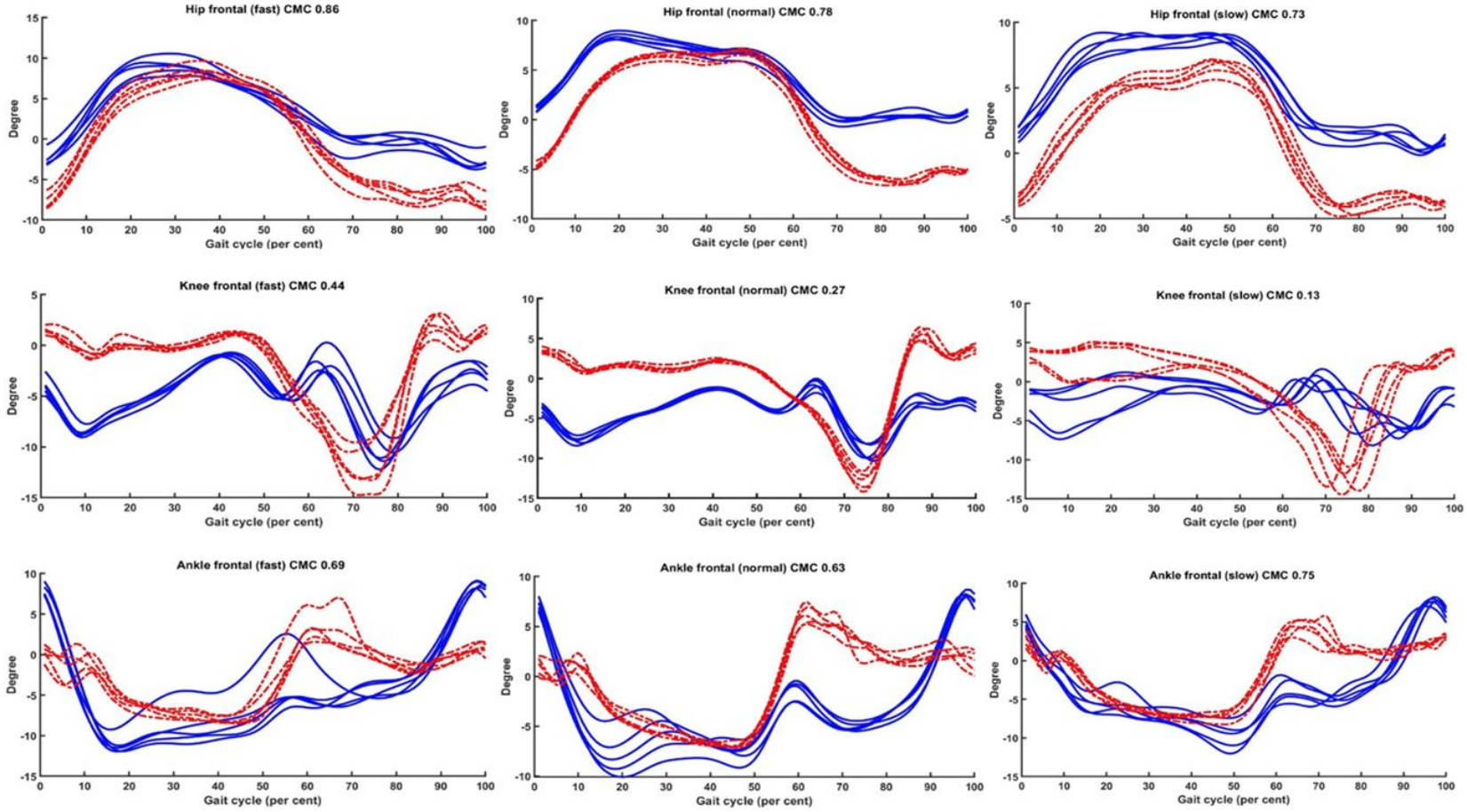

Figure 4.4 Waveform similarity and CMC of joint angles (frontal plane) of a participant obtained from Vicon system (blue) and the joint angles obtained from the Xsens sensors (red) at fast, normal and slow sped of walking.

Correspondingly, the LFM demonstrated that the joint angles obtained by the Xsens Awinda were comparable with the conventional method in sagittal plane motion followed by frontal and transverse plane. Average R^2 values were higher in angles of larger joints. Data recorded from slow walking speed showed poorer results than the others (table 4.1).

There was excellent similarity of sagittal plane joint angles in most of the joints at normal and fast speed according to the R^2 (0.71-0.98). However, it was generally fair to high at the slow speed walks. The R^2 values of frontal plane joint angles ranged from poor to fair to high. Poor similarity was observed in the transverse plane motion. The angular coefficient (a_1) of the linear regression of joint angles shown in table 4.2 represents a constant number that is needed for P_{ref} (a data point obtained from Vicon system) to be equivalent with P_a (data point obtained from Xsens).

Additionally, table 4.3 reveals the offsets (a_0) that are required to add on and make equivalent joint values with the Vicon system.

Table 4.1 Average R^2 +-SD of the linear regression model of kinematics data obtained from the Xsens Awinda compared with PiG model and Vicon system.

Average R^2 ±SD of sagittal plane motion at

Joint angle	**Normal speed**		**Fast speed**		**Slow speed**	
	Left	Right	Left	Right	Left	Right
Ankle	0.88±0.06	0.77±0.11	0.83±0.08	0.71±0.13	0.64±0.36	0.56±0.30
Knee	0.95±0.02	0.96±0.01	0.93±0.03	0.94±0.03	0.71±0.43	0.71±0.04
Hip	0.98±0.02	0.98±0.02	0.98±0.01	0.96±0.04	0.74±0.04	0.72±0.43

Average R^2 ±SD of frontal plane motion at

Joint angle	**Normal speed**		**Fast speed**		**Slow speed**	
	Left	Right	Left	Right	Left	Right
Ankle	0.67±0.25	0.49±0.29	0.61±0.21	0.40±0.29	0.48±0.32	0.41±0.29
Knee	0.27±0.20	0.40±0.28	0.33±0.27	0.36±0.26	0.28±0.23	0.36±0.28
Hip	0.48±0.32	0.27±0.30	0.49±0.31	0.36±0.34	0.45±0.34	0.39±0.34

Average R^2 ±SD of transverse plane motion at

Joint angle	**Normal speed**		**Fast speed**		**Slow speed**	
	Left	Right	Left	Right	Left	Right
Ankle	0.10±0.10	0.18±0.19	0.18±0.18	0.22±0.22	0.18±0.16	0.23±0.21
Knee	0.17±0.24	0.23±0.23	0.28±0.27	0.27±0.21	0.30±0.28	0.27±0.23
Hip	0.23±0.20	0.16±0.20	0.24±0.22	0.14±0.19	0.28±0.26	0.21±0.20

Table 4.2 The angular coefficient (a_1) of the linear regression model

Average a_1±SD of sagittal plane motion at

Joint angle	**Normal speed**		**Fast speed**		**Slow speed**	
	Left	Right	Left	Right	Left	Right
Ankle	1.23±0.21	1.08±0.18	1.24±0.23	0.99±0.16	1.05±0.57	0.82±0.44
Knee	1.05±0.04	1.06±0.08	1.03±0.05	1.04±0.07	0.79±0.46	0.80±0.46
Hip	0.86±0.09	0.83±0.10	0.87±0.09	0.85±0.12	0.67±0.37	0.64±0.35

Average a_1±SD of frontal plane motion at

Joint angle	**Normal speed**		**Fast speed**		**Slow speed**	
	Left	Right	Left	Right	Left	Right
Ankle	0.70±0.32	0.55±0.33	0.59±0.28	0.41±0.28	0.66±0.43	0.43±0.38
Knee	0.41±0.64	0.58±1.12	0.31±0.75	0.53±0.88	0.49±0.76	0.40±0.28
Hip	0.94±0.37	0.75±0.68	0.97±0.34	0.81±0.72	0.77±0.53	0.71±0.81

Average a_1±SD of transverse plane motion at

Joint angle	**Normal speed**		**Fast speed**		**Slow speed**	
	Left	Right	Left	Right	Left	Right
Ankle	-0.08±0.37	0.19±0.28	-0.22±0.41	0.22±0.27	-0.09±0.24	0.18±0.24
Knee	0.26±0.39	0.00±0.60	0.36±0.44	0.13±0.60	0.40±0.64	0.11±0.77
Hip	-0.04±0.27	-0.09±0.19	-0.03±0.31	-0.04±0.17	0.01±0.20	-0.16±0.33

Table 4.3 The offsets (a_0) to make equivalent joint angle between gait data that obtained from Xsens and PiG and Vicon system.

Average a_0±SD of sagittal plane motion at

Joint angle	**Normal speed**		**Fast speed**		**Slow speed**	
	Left	Right	Left	Right	Left	Right
Ankle	-7±4.60	-4.26±4.06	-7.07±4.69	-4.52±3.53	-5±8.50	-2.32±5.52
Knee	-2.84±5.47	0.39±6.15	-0.54±6.40	0.55±5.91	-0.13±7.74	1.57±7.86
Hip	-5.47±3.55	-4.39±5.59	-4.33±2.88	-4.73±5.28	-2.84±5.26	-2.08±7.50

Average a_0±SD of frontal plane motion at

Joint angle	**Normal speed**		**Fast speed**		**Slow speed**	
	Left	Right	Left	Right	Left	Right
Ankle	6.90±5.46	2.03±2.80	4.58±5.70	0.82±3.21	7.71±7.60	2.26±3.59
Knee	2.07±3.35	0.22±7.50	0.57±4.13	-0.26±5.86	1.33±4.28	-0.92±9.45
Hip	0.41±4.39	1.06±4.66	1.49±3.62	3.36±5.25	1.68±4.48	3.47±6.02

Average a_0±SD of transverse plane motion at

Joint angle	**Normal speed**		**Fast speed**		**Slow speed**	
	Left	Right	Left	Right	Left	Right
Ankle	-4.72±6.22	3.22±04.56	-5.29±6.63	1.83±6.12	-3.51±3.80	1.09±5.45
Knee	-1.27±5.19	-5.02±8.05	0.82±6.59	-2.41±11.63	0.96±9.22	-0.01±13.34
Hip	3.80±7.40	-3.03±8.24	6.13±9.14	-5.44±8.14	4.00±4.66	-3.27±7.31

Discussion

The findings from this chapter showed good to excellent waveform similarity at the sagittal plane motion of ankle, hip and knee respectively, also corresponded with the correlation coefficient values. However, the waveform similarity at the frontal planes of the two motion capture systems were less identical. The results were similar with some previous studies.

Ferrari et al. (2010) reported excellent accuracy of using Xsens in all planes of motion in the thoraco-pelvic, hip, knee and ankle joints for gait analysis compared to the standard motion capture system using the 'Outwalk' protocol. The hallmark study has encouraged the possibility of using an IMU system in gait analysis to provide a clinically accessible motion laboratory toward the needs as it is portable and less expensive than the conventional motion capture system.

Similar to Ferrari et al. (2010), high CMC values from our results suggested the capability of Xsens for the estimation of gait kinematics, particularly, in joints with a wide range of motion (ROM) during gait. Excellent correlation between knee sagittal angles was found at all different walking speeds. Similarly, the sagittal movement of the hip and ankle joints obtained from the Xsens system were comparable with the Vicon system confirmed by good to very good CMC values. Poor to moderate correlations were observed in the frontal plane and the CMCs of transverse plane motion could not be reported due to their non-real number results which are similar to the studies from Zhang et al. (2013) and Al-Amri et al. (2018) The studies that validated Xsens against the conventional motion capture system without using the 'Outwalk' protocol demonstrated poor correlation of the kinematics in the frontal

and transverse plane. One possible explanation would be the non-reliable marker placement at a smaller joint and skin movement artefact.

More importantly, there is a difference between the biomechanical models of the two motion capture systems compared. Ferrari et al. (2010), in particular, used the "Outwalk" protocol as a functional calibration for the Xsens system and showed the superior results. Ideallly, joint angles derived from the Xsens system should be identical with the joint angles derived from the Vicon system, however, there could be some discrepancy of data influenced by marker placement error (Al-Amri et al., 2018) between the two systems (the virtual markers of the Xsens are developed from the biomechanical model and data obtained from a sensor unit). In addition, errors can be accumulated during data collection due to the potential magnetic disturbances in the motion laboratory and in integration drift despite the good calibration was shown at the beginning (Roetenberg, Slycke and Veltink, 2007). Skin artefact can also be a factor to interfere data recording accuracy (Mundt et al., 2019). Therefore, Xsens was not recommended to use interchangeably with the standard motion capture for motion analysis (Al-Amri et al., 2018). The Xsens Awinda's ability to capture a movement has been improved remarkably in terms of the accuracy and the magnetic immunity by utilising the Strap-Down Integration algorithm combination with Kalman filter (the sensor fusion algorithm). None of the Xsens calibration failed due to magnetic disturbance during the course of the study although the trackers were used on a treadmill (on average, at least 6 cm from the foot sensor to the treadmill belt) which is a magnetically challenging environment (Schepers, Giuberti and Bellusci, 2018).

In this study the accuracy of the Xsens was validated in order to examine the suitability of using this motion capture system when compared with the standard method and also addressing the possibility of using such system outside laboratory. Coefficient of multiple correlation was used as it is one of the recognised techniques to represent waveform similarity between different data collection methods of gait analysis. Iosa et al. (2014) and Røislien et al. (2012) recommended to avoid using CMC due to its non-reliable values. In contrast, they rather suggested using another method such as LFM to overcome issues considered to affect the interpretation of waveform similarity calculated by CMC for example, inadequate number of the participants, the data recorded with high sampling rate and the consistency of marker placement between raters. In this study, we did not find an obvious difference between CMC and LFM. Both methods showed similar results with good to excellent correlation in sagittal plane of movement especially in normal and fast speed. The other parameters of the LFM also showed the corresponding results with the R^2. In particular, if a_1 is equal to 1, it could be stated that joint angles obtained from the two motion capture systems were perfectly matched. It can be seen that a_1 values in sagittal plane angles were approximately between 0.8-1.2 whereas the a_1 values in frontal plane and transverse plane angles were at a wider range between 0.3-0.9 and -0.2-0.4 respectively.

In our study, we aimed to examine the ability of Xsens to estimate gait kinematics which will be the inputs to train an artificial neural network. Therefore, we conducted only a single experiment with no different day testing or different raters involved. Since the study's results are comparable with results from Zhang et al. (2013) and Al-Amri et al. (2018) gait kinematics derived from Xsens system are not completely

equivalent with the standard motion capture system, instead, Xsens describes gait characteristics that were processed using a different biomechanical model. Subsequently, an individual's gait obtained from the two motion capture system are not recommended to be used interchangeably. The feasibility of using the Xsens system in clinical gait will be further discussed in the next chapter when the joint angles obtained from the system were used as input to train an ANN.

Chapter 5: Estimation of the knee abduction moment during gait using an artificial neural network from joint angles obtained by inertial magnetic measurement units

Background

Gait analysis is an objective measurement of human walking that also includes a study of movement or joint angles (kinematics) and a study of forces as joint loading or joint moment (kinetics) and power (Mayich et al., 2014). The kinetics of gait are the measurement of forces that cause the movement. The abnormality of the forces that repetitively act on the joint can ultimately initiate musculoskeletal problems. A common health condition of the long-term effect of altered joint loading is knee osteoarthritis mostly affecting the elderly population (Maly, 2008).

Conducting gait analysis using IMUs can be combined with additional instruments such as instrumented shoes, portable pressure sensors and a transportable force plate in order to acquire gait kinetics (Koch et al., 2016). A pressure measurement insole provides a thin and less bulky option for a mobile kinetic measurement in however, the major drawback is that it provides only vertical GRF and the centre of pressure (CoP) is defined as the centre of the pressure distribution over its surface (Liedtke et al., 2007). Low validity, high sensitivity and especially poor durability were also reported as its shortcomings by several studies (Abdul Razak et al., 2012; Koch et al., 2016; Shahabpoor and Pavic, 2017). Joint moments and GRFs could also be estimated by the calculation from data of an accelerometer and mass of a body segment following Newton-Euler dynamic algorithms, based on the recursive formulation of force, moment balance equations and the open kinematic chains of the Newton-Euler dynamic algorithm (Ohtaki, Sagawa and Inooka, 2001). Despite providing a promising result of the estimated joint moments in gait, the technique lacked accuracy due to the inaccuracies in kinematic calculation (Ancillao et al., 2018).

Using an ANN to estimate joint moments has also become an area of interest (Ancillao et al., 2018). Hahn and O'Keefe, 2008b estimated sagittal plane moments of hip, knee and ankle joint from EMG signals and anthropometric data of 19 healthy volunteers using a three layers ANN. They demonstrated great performance of the ANN by coefficient of determination (r^2) at 0.90 for hip and knee moments and 0.95 for the ankle moment.

Ardestani et al. (2014) used GRFs and EMG signals as inputs to train two types of ANNs: FFANN and a wavelet neural network (WNN) and compared their ability to predict lower extremity joint moments. All planes of hip moment, knee sagittal moments and ankle sagittal and transverse plane moments were estimated by the two ANNs. The result showed a better performance of WNN than the FFANN at normalised RMSE < 10% and cross correlation coefficient > 0.94. Likewise, Mundt et al. (2018) firstly, predicted hip, knee, and ankle joint moments in normal gait from a trained long-short term memory based recurrent ANN using hip, knee and ankle joint angles obtained from an optoelectronic motion capture system, gait velocity and anthropometric data as inputs. They showed comparable results with the study from Ardestani et al. (2014).

Aiming to use IMUs for gait analysis outside the laboratory, Mundt et al. (2019) later investigated the ability of using LSTM to estimate normal gait kinematics and kinetics from simulated IMU data. The simulated linear acceleration and angular velocity were computed from gait data obtained from a standard motion capture system and used as inputs to train the LSTM compared with a FFANN. The simulated IMU data were claimed to be better from the data originally derived IMU because there was no skin movement artefact. Although the FFANN was generally superior, both

networks addressed high correlation coefficient at 0.98 in the sagittal plane moment and 0.80 in the other planes (Mundt et al., 2019).

Including the results from pilot studies in chapter 3, it has been shown that ANNs are able to predict joint moments in gait from various types of input that are related, mathematically, to the target joint moments such as joint angles obtained from the optoelectronic method, simulated IMU data, EMG signals and the individual's anthropometric parameters.

In this study, the internal KAM was estimated by using a FFANN, from an array of gait kinematics obtained from an Xsens Awinda IMU motion tracking system.

Materials and methods

Research participants

Fifteen healthy volunteers were recruited from staff and students in the Faculty of Sciences, LJMU (the same group of participants in chapter 4). The study was approved by the university research ethics committee (reference number 18/SPS/005). Written informed consent was obtained prior to data collection.

Participant preparation

In a tight T-shirt, tight cycling style shorts and their own comfortable training shoes the participants anthrometric data were measured and seven MTw2 data trackers (Xsens Technologies B.V., The Netherlands) were attached on body segments in the same fashion as described previously in chapter 4. Followed by the reflective marker placement according to the HBM model as shown in figure 3.3. To prevent the

reflective markers from being knocked off the participant during the placement of the MTw sensors using Velcro straps, the markers were placed on the anatomical landmarks after all MTw sensors were completely in place. A safety harness, thereafter, was fitted on the participant's body and was attached to the participant's clothes by using tape where necessary to prevent its movement which could potentially obscure some reflective markers during the test.

Data collection

The Xsens sensors were calibrated in an unoccupied area before the walking was performed at three different speeds on the treadmill followed the process described in chapter 4. Markers signals were then fed through Vicon Nexus to D-Flow ready for data collection.

Kinematics data and GRFs were recorded by the combination of Mocap and Treadmill module via a custom D-Flow application (D-Flow, Motekforce Link, the Netherlands) saved into two files of different formats (.mox and .txt). The kinematics data were captured by Vicon Nexus 2.5 at 120 Hz through 12 Vero cameras (Vicon, Oxford Metrics Group Ltd.) and streamed in real time to D-Flow. Kinematics data were recorded simultaneously at 100 Hz by Xsens MVN analyse 2018 (Xsens Technologies B.V., Netherlands) as .mvnx files. Both systems were synchronised by the technique described in appendix 3.

The participant was asked to walk on the treadmill at three different speeds: normal (self-selected), fast (40% faster than normal speed) and slow (40% slower than normal speed). Data collection began after the self-selected speed was determined and recorded as described in the validation study (Chapter 4).

Data processing and analysis

Joint movement from frontal (abduction/adduction), sagittal (flexion/extension) and transverse (internal/external) plane of motion of the right hip, right knee and right ankle were directly extracted from the saved Xsens files. The data obtained from D-Flow were computed for gait kinematics and kinetics using the HBM model in Gait Offline Analysis Tool (GOAT, Motekforce Link, the Netherlands) smoothed with low-pass filter at 10 Hz and were saved as .mox files. Right KAM was then extracted from the data at the starting time point indicated by the rising edge generated by the Xsens system which was seen in the added analogue channel. One set of gait data contained 1000 data points, to assure the sufficient data for the FFANN training process, was selected for each walking speed from the beginning of each set of data (defined by the rising edge) were saved as .xls files to be used in the next step of data analysis.

The Levenberg-Marquard backpropagation neural network, a three layers FFANN, was chosen to estimate joint moment in this study. A modified Matlab script (appendix 5) was created with the Neural Network Toolbox in Matlab 2019b (The MathWorks Inc., MA, USA). The ANN was trained using kinematics (right hip, knee, and ankle angles in frontal, transverse and sagittal planes) as inputs and right KAM obtained from inverse dynamics by GOAT as target outputs. Therefore, there were 9 variables X 1000 data points of inputs and 1 variable X 1000 data points of target outputs for each set of gait data included in the neural network training and prediction process.

Two sub-studies were conducted according to walking speed. Firstly, data from all three walking speeds were presented to the ANN at the same time. This was to see the performance of the ANN when it was trained by data, which included gait

variability due to different gait speeds of an individual's gait pattern. Secondly, the ANN was trained with data captured at each gait speed separately to examine the ability of the ANN to predict joint moment at a particular walking speed.

In order to cover all available data to train neural network models without bias, a k-fold validation was applied to this study (k=15) (Fushiki, 2011). To estimate the joint moment, the data of 13 participants were allocated to train the ANN, data from the 14th participant were used for validating and data from the 15th (last) participant were used for testing the performance of the ANN which is the set of unseen data that the FFANN never sees during training (this made 6.67% of data for testing the FFANN performance) (figure 5.1).

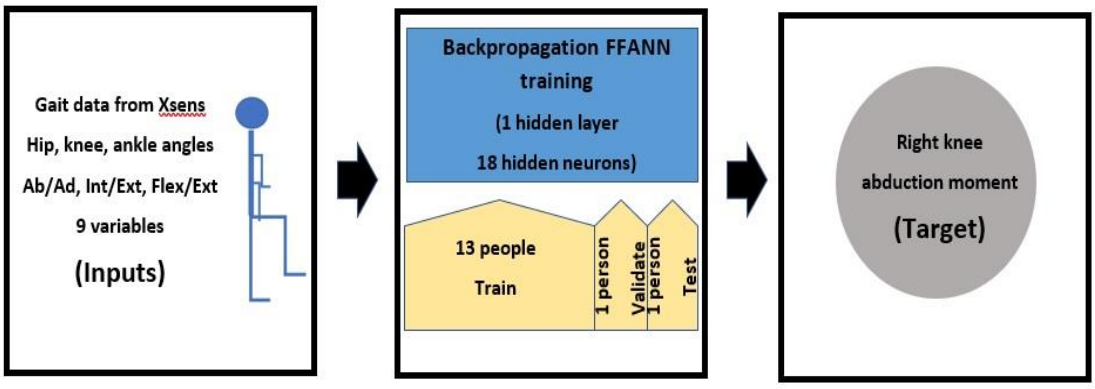

Figure 5.1 The block diagram shows the order of FFANN training process, the inputs were right hip, knee, ankle angles in three planes of motion. The FFANN was trained by data from 13 people, validated by data of one person and tested by data of one person. Data from each participant were used to test the ANN one at a time in 15 sets of training-validation-test data. Right knee abduction moment was the target output.

Data from each participant were used to test the performance of the FFANN, one at a time, consequently there were 15 ANN architectures for the first sub-study where the data of all walking speeds were trained together in one ANN and 45 ANN architectures for the second sub-study where the data of each walking speed were trained separately (3 speeds x 15 sets). Root mean square error and normalised root mean square error were calculated to examine the difference between the FFANN outputs and target output. The NRMSE was used in this study to compare with previous studies reported in the literature.

The mean differences between the target and the predicted KAM of all walking speeds and sub studies were analysed using a one way analysis of variance test (ANOVA) with the null hypothesis that there was no significant difference within and between the groups, at the significance level of 0.05. The normal distribution assumption and the homogeneity of the variances were performed using Shapiro-Wilk and Levene's test. Welsh's test would be used for data analysis if the data violated the normal distribution and homogeneity of the variances. The data were analysed by a statistical package for the social sciences (SPSS) for Windows software version 26 (SPSS Chicago, IL, USA).

Results

Fifteen volunteers participated in the study, average age of 27.6±4.39 (22-35) year, height 1.7±0.10 (1.51-1.87) metres and body mass 66.62±11.89 (41.55-89.25) kg. The average walking speeds were 1.14±0.21 (0.6-1.4), 1.60±0.30 (0.84-1.96) and 0.68±0.13 (0.36-1.84) m/s for normal, fast and slow speed respectively.

The results showed that the FFANN was capable to predict the right knee abduction moment when it was trained by either kinematic-kinetic pairs from all walking speeds

at the same time (the first sub-study) or when the FFANN was trained by data from each walking speed separately (the second sub-study).

Firstly, when data from all speeds were used for training the FFANN, the highest performance was found for predicting KAM at normal speed, followed by slow and fast speed respectively. Average RMSE of the training part in the first sub-study was 0.129±0.013 (0.104-0.170) Nm/Kg and the average RMSE of the testing part (where the FFANN did not see the data before) were at 0.132±0.074 (0.047-0.313), 0.171±0.105 (0.069-0.496) and 0.135±0.070 (0.063-0.290) Nm/Kg for normal, fast and slow speed respectively.

Similar results were found when the ANNs of each gait speed were trained separately, the average RMSEs of the testing parts of the FFANN training were 0.141±0.059 (0.092-0.302), 0.174±0.106 (0.087-0.497) and 0.131±0.059 (0.038-0.239) Nm/Kg for normal, fast and slow speed (figure 5.2). The RMSEs of training parts also corresponded to the results in the testing parts (figure 5.3).

In addition, there were wide ranges of NRMSE at the testing parts at 13%-92%, 10%-79% and 2%-62% for normal, fast and slow walk of the first sub-study. In the second sub-study the ranges were at 6%-70%,14%-80% and 3%-68% for normal, fast and slow speed (figure 5.4).

Due to the non-homogeneity of the variances, Welch test with Game-Howell post hoc test were used for statistical analysis within and between the studies and walking speeds. There were significant differences between the KAMs predicted by sub-study 1 compared to sub-study 2 at normal ($F(1,1998) = 48.54$, $p = 0.00$) and slow speed $F(1,1998) = 328.87$, $p = 0.00$), the effect sizes were 0.02 and 0.14 respectively, however there was no significant difference between the KAMs from fast speed that were predicted by both sub-studies ($F(1,1998) = 3.24$, $p = 0.07$)

(table 5.1). There were significant differences between the KAMs from each speed that were predicted when data from all speeds were used to train the FFANN in the first sub-study ($F(2,2997) = 469.23$, $p = 0.00$). The Games-Howell post hoc analysis showed the predicted KAMs from fast speed was significantly different from the other walking speeds when the FFANN was trained using gait data from each speed separately ($p = 0.00$).

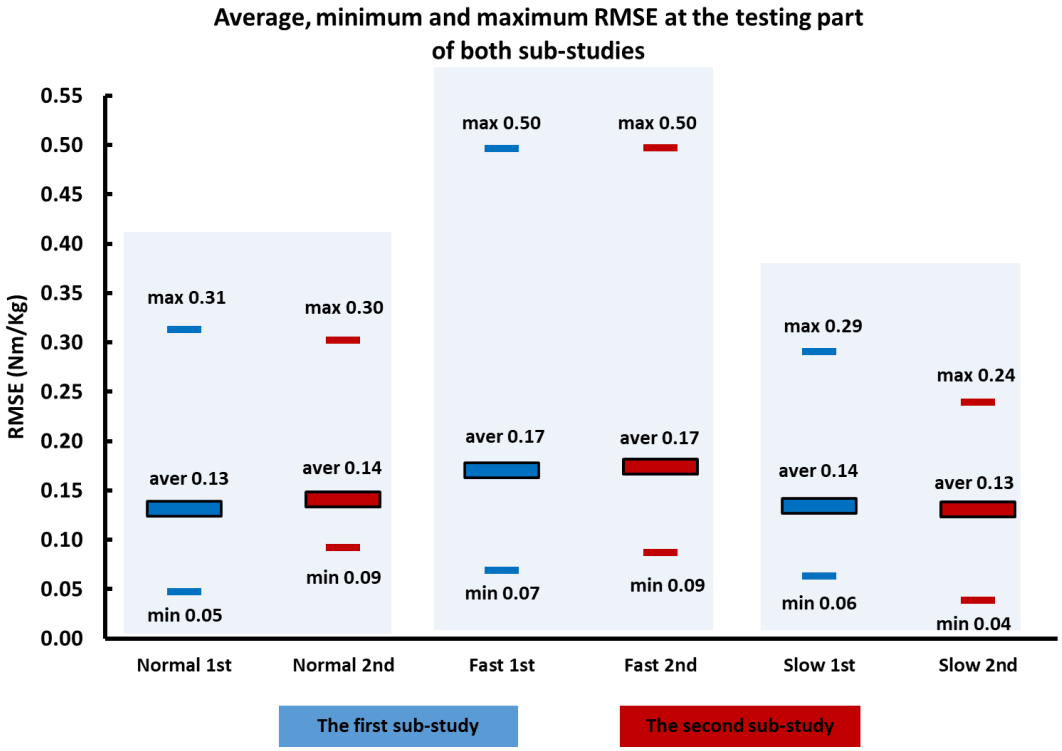

Figure 5.2 The average RMSEs between the target output and the predicted output at the testing part of the FFANN prediction. The comparable results were found when the FFANN was trained by data of all speeds together (sub-study1) (blue) and when it was trained by data of each walking speed separately (sub-study 2) (red). There was a wide range of the RMSE amongst the individuals seen by the minimum and maximum RMSE values.

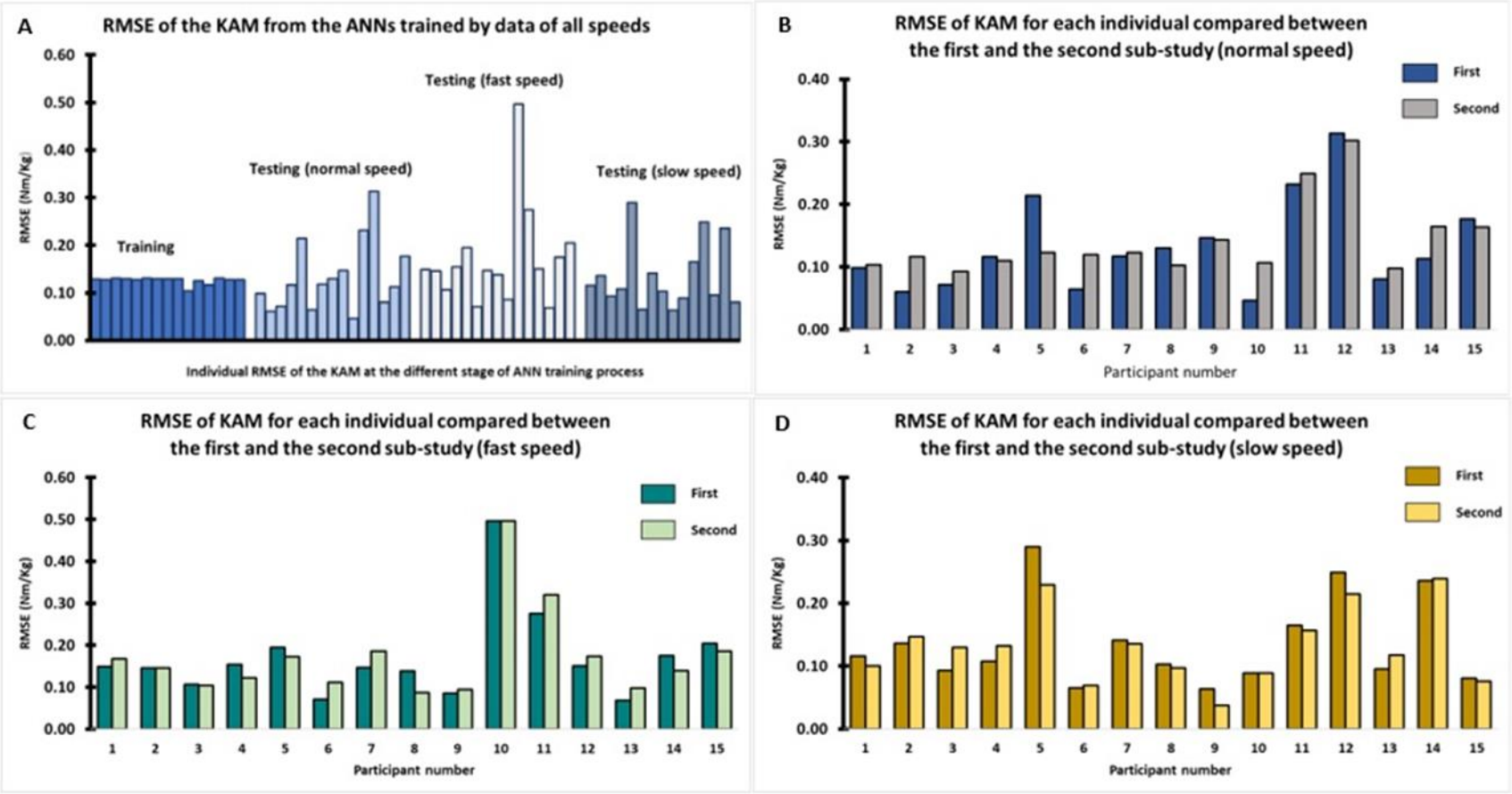

Figure 5.3 The bar charts show RMSEs between target outputs and predicted outputs. 5.3 A shows the general trend of FFANN prediction when FFANNs were trained using data from all walking speeds (the first sub-study) of participant 1-15 respectively. RMSEs of the training part are similar but the variation amongst RMSEs at the testing part was observed at all walking speeds. That implies a lack of generalisation ability of the FFANN. 5.3B-D illustrate the KAM predictions at the testing part between the first sub-study and the second sub-study where the data of each walking speed were separately trained. The variation of the RMSEs of the KAM at the testing part could be seen by both FFANN training methods.

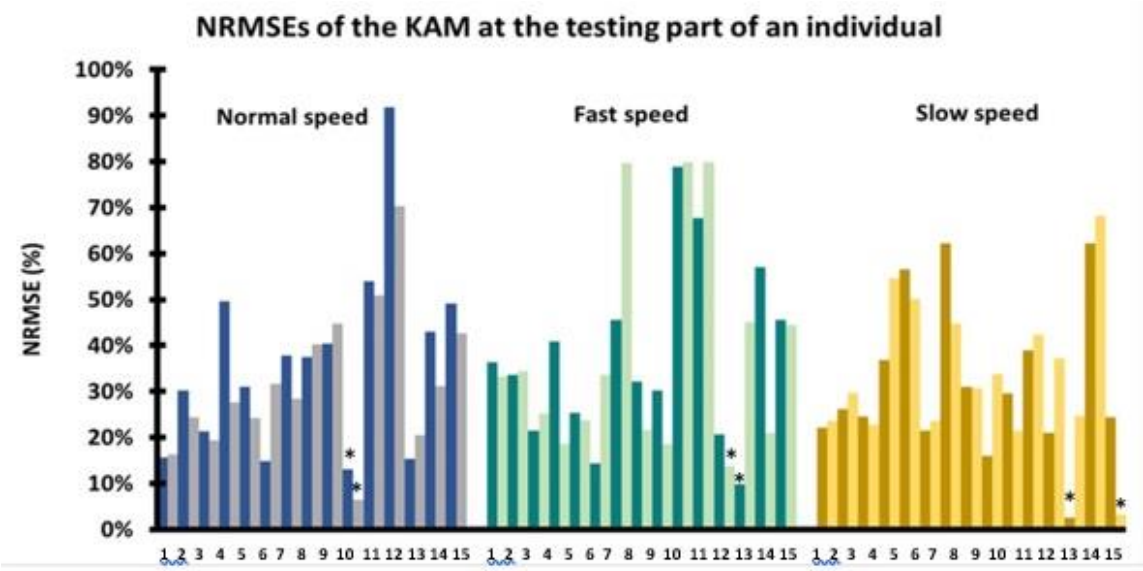

Figure 5.4 NRMSEs of the KAM of an individual predicted by the first sub study (dark) and the second sub-study (light shade) of normal speed (blue), fast speed (green) and slow speed (yellow). The ranges of NRMSE were 13%-92%, 10%-79% and 2%-62% for normal, fast and slow walk of the first sub-study and 6%-70%,14%-80% and 3%-68% for normal, fast and slow speed in the second sub-study. Asterisks denote the best performance of the FFANN.

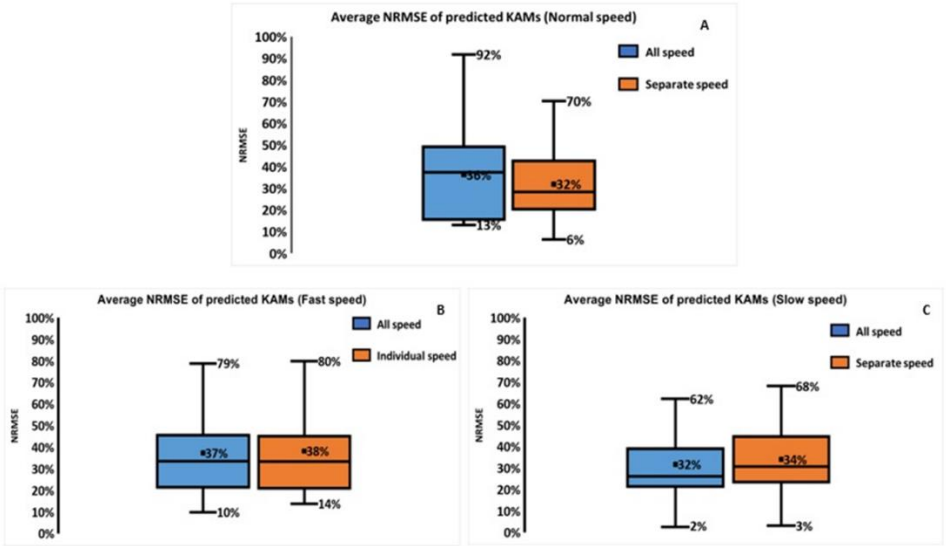

Figure 5.5 Box and whisker plots demonstrate ranges of the NRMSEs of the KAM prediction at the testing part of the FFANN comparing between the first sub-study (blue) when all walking speeds were trained together and the second sub-study (orange) when the three walking speeds were trained separately. The FFANN performances were significantly different between the two sub-studies when predicting the KAM at normal and slow speed ($p < 0.05$), A = normal speed, B = fast speed and C = slow speed.

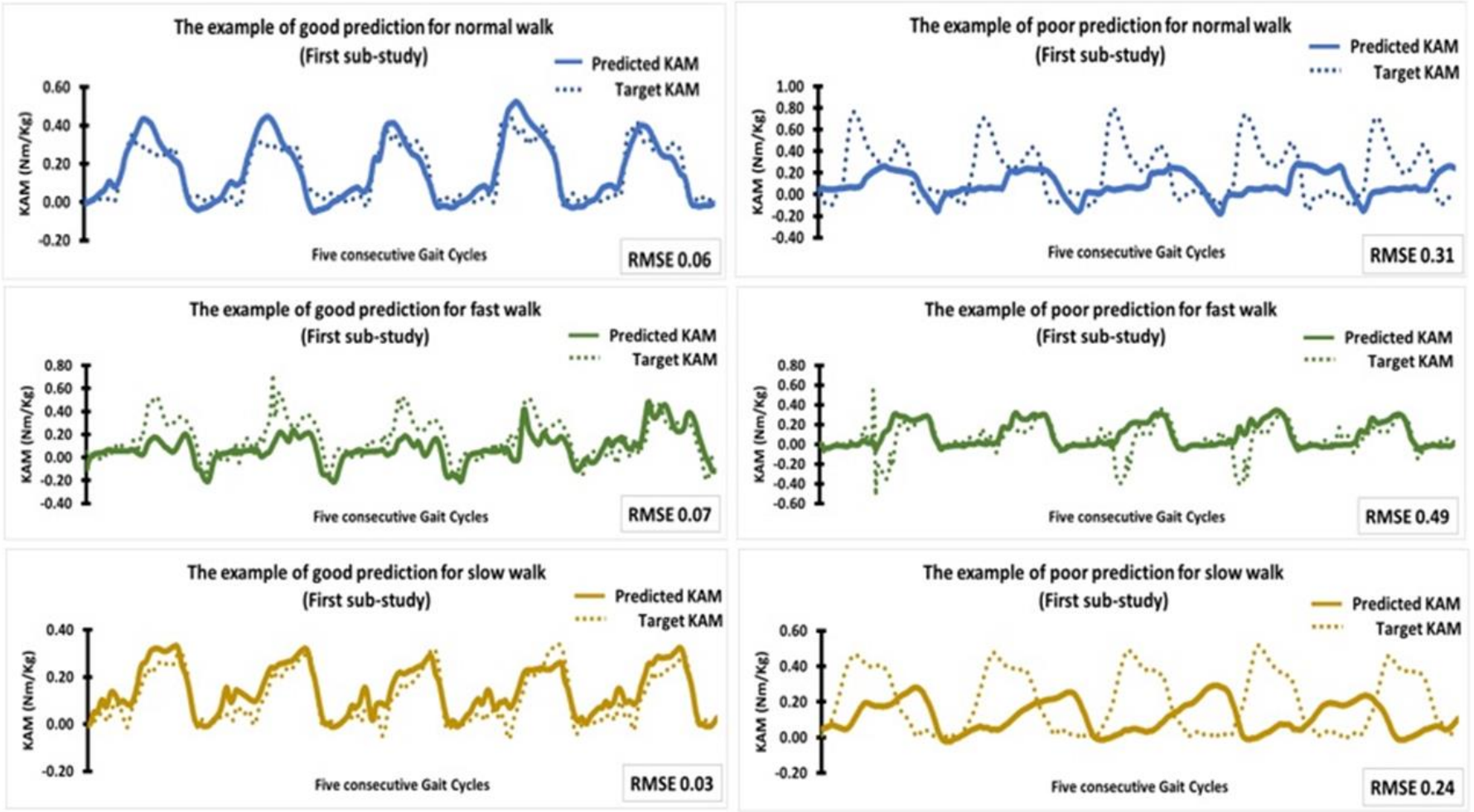

Figure 5.6 Example of good and poor KAM prediction in five consecutive gait cycles from the testing part of the first sub-study (all speeds were trained together). At good prediction (left column), the amplitude and shape of the predicted KAMs are similar to the target KAMs (dotted line).

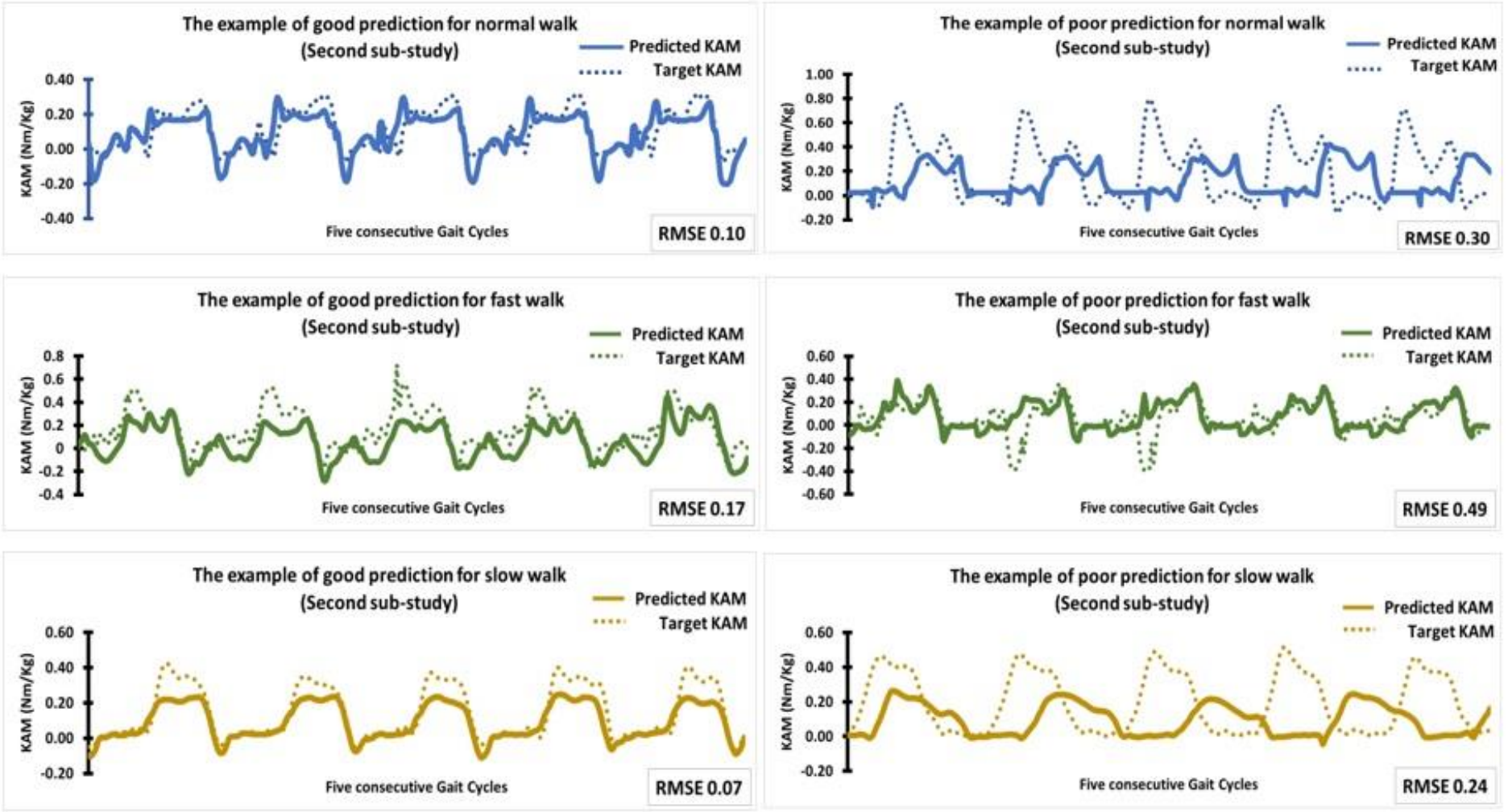

Figure 5.7 Example of good and poor KAM prediction in five consecutive gait cycles from the testing part of the second sub-study (each speed was trained separately). At good prediction (left column), the amplitude and shape of the predicted KAMs are similar to the target KAMs (dotted line).

Table 5.1 The differences between the KAM from normal, fast and slow speeds that were predicted by two sub-studies at p value < 0.05 (speed 1 = fast, speed 2 = normal, speed 3 = slow).

ANOVA						
		Sum of Squares	df	Mean Square	F	Sig.
Fast	Between Groups	0.003	1	0.003	3.248	0.072
	Within Groups	1.671	1998	0.001		
	Total	1.674	1999			
Normal	Between Groups	0.030	1	0.030	48.547	0.000
	Within Groups	1.253	1998	0.001		
	Total	1.283	1999			
Slow	Between Groups	0.207	1	0.207	328.879	0.000
	Within Groups	1.260	1998	0.001		
	Total	1.467	1999			

Multiple Comparisons							
Games-Howell							
Dependent Variable	(I) Speed	(J) Speed	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
						Lower Bound	Upper Bound
The 1st sub-study	1.000	2.000	-.006*	0.001	0.000	-0.009	-0.003
		3.000	-.035*	0.001	0.000	-0.038	-0.032
	2.000	1.000	.006*	0.001	0.000	0.003	0.009
		3.000	-.0293*	0.001	0.000	-0.032	-0.026
	3.000	1.000	.035*	0.001	0.000	0.032	0.038
		2.000	.0293*	0.001	0.000	0.026	0.032
The 2nd sub-study	1.000	2.000	-.011*	0.001	0.000	-0.014	-0.008
		3.000	-.012*	0.001	0.000	-0.015	-0.010
	2.000	1.000	.011*	0.001	0.000	0.008	0.014
		3.000	-0.001	0.001	0.519	-0.004	0.001
	3.000	1.000	.0124*	0.001	0.000	0.010	0.015
		2.000	0.001	0.001	0.519	-0.001	0.004

* The mean difference is significant at the 0.05 level.

Discussion

The evolution of inertial motion capture has matured to a level where the concept of out of the lab gait analysis has become a real possibility. In this chapter, the feasibility of estimating the kinetics of gait, knee abduction moment in particular, was investigated using the FFANN trained by joint angles provided by the IMU system. Low RMSEs between the target KAM and the predicted KAM were shown in both training and testing part, especially at the testing part of the slow walk. Thus reflecting good generalisation ability of the FFANN model used in terms of estimation the KAM that the FFANN did not see during training. The FFANN could practically be used to complement the joint angles obtained from an IMU system, bringing the possibility of portable gait analysis closer.

At normal and slow speed, significant differences of the predicted KAMs were found when the FFANNs were trained by the data of all walking speeds together (first sub-study) and when the data of each speed were separately used to train the FFANN (second sub-study). However, no significant difference of the predicted KAM was found at fast speed when the FFANN was trained by either method. One reason why the FFANN predicted the KAM from normal and slow speed differently could be the fact that in the first sub-study, there was a variety of KAM from the 15 participants from the three walking speeds which were presented to the FFANN at the same time. Therefore, the FFANNs were able to recognise from different patterns to predict the unseen KAM at the testing part while in the second sub-study the FFANN could only recognise a particular pattern from each walking speed. However, for fast speed, there was more noise obtained when collecting data which could affect the overall

pattern of gait for the FFANN to see the distinctive pattern when training in both methods.

Our results, in general, show that the FFANN performed better to predict the KAM of slower speed walk with a small average RMSE, ranged from 0.131-0.135 Nm/Kg (minimum 0.038 Nm/Kg) at slow speed of both sub-studies and 0.132-0.141 Nm/Kg (minimum 0.047 Nm/Kg) at normal speed. In spite of showing a good KAM prediction at the lowest NRMSE of 2% for the first sub-study and 3% for the second sub-study at slow speed, there was still a wide variation of results in this study in order to predict the KAM where the largest NRMSEs, at the same speed, were shown at 62% and 68% for the first and the second sub study respectively. Comparing with a previous study, Aljaff et al. (2016) trained an FFANN by using joint angles as inputs to predict the KAM in AKU gait, our result is slightly better in terms of RMSE as their best performance was shown with RMSE at 0.074 Nm/Kg. The walking speed was not mentioned in their study therefore it is difficult to directly compare with our RMSE from each walking speed, presumably that the AKU patients could not walk too fast due to their nature of disease. Correspondingly, the FFANN architecture used in our study should be able to predict KAM in clinical practice for abnormal gait especially in some population e.g. osteoarthritis and elderly who typically walk slower than normal.

Average NRMSEs amongst all walking speeds when the FFANN was trained by data of all speeds at the same time (the first sub-study) is at 35.66% (2%-92%) which is higher than the study from Mundt et al. (2018) that predicted joint moments of gait by training a long short-term memory based recurrent ANN (LSTM) using joint angles obtained from an optoelectronic system, walking speeds and anthropometric

parameters including height, weight, foot length, shank length and thigh length as inputs. They reported the average NRMSE of the knee adduction moment at 11.35% of the testing part where the joint moments of unseen data were predicted. The obvious difference of the NRMSE between the two studies could be due to a few reasons. Firstly, the anthropometric data were included as inputs so the data can provide extra characteristics of gait for the ANN to learn while training. Most importantly, the inputs were shuffled (randomized) before training which would have less bias and then become more generalised when the ANN is predicting a new set of data that are unseen. Moreover, it would be difficult to directly compare the performance of the two ANNs that were used in these studies due to the difference of the ANN architectures used.

The performance of FFANN and LSTM in order to predict joint moment in gait were later compared in a previous study by Mundt et al. (2019). They concluded that the FFANN provided better results than the LSTM. The group simulated IMU data from the gait data obtained from a standard motion capture system and used them as inputs to train the two ANNs. Their overall results were better than this chapter with the average NRMSE of the frontal plane knee joint moment was at 10.58%. However, on closer examination, a similar pattern was found from the FFANN's performances as there was a wide variation of the prediction demonstrated by a wide range of the NRMSEs between 6%-80%. More reliable prediction was found in the sagittal plane joint moment with a high correlation coefficient at 0.98 from both types of the ANN. With simulated IMU data, the researchers strongly claimed that the data were better than using the data obtained from actual IMU system because there was no soft tissue movement artefact that could affect data processing as well as avoiding the

ferromagnetic disturbance when the IMU data are collected. They concluded that there was a high relation between the predicted and the actual moment regarding the figures reported. Therefore, an FFANN seems to be a suitable tool for joint moment estimation in gait analysis.

In comparison to pilot study (chapter 3), we recruited more participants to this study which made a remarkable impact to the performance of the FFANN according to the recent results. We have found from chapter 3 that the FFANN performed better when some of the expected targets were introduced to the FFANN at the training part. So, if the ANN is trained with more instances of inputs-outputs, it should be able to estimate outputs that the ANN did not see before when training which is presented as the testing part. This is to demonstrate the generalisation of the ANN in predicting the joint moments of a new participant that has never been seen.

Walking speeds has known to affect knee joint loading in gait, in general, the sagittal and frontal plane joint moment increase when an individual walks at a high speed (learner 2014). Walking speed also affected the results. The FFANN provided best results for the slow speed but less accurate predictions in the fast speed walk in both sub-studies. This can be explained by the gait pattern being more consistent with a longer double support phase duration, stride length and shorter step width (Lee 2017 age related) when people walk slowly thus having less effect to joint moments due to the smaller impact on GRFs. In contrast, a fast speed walk could affect the GRFs especially on the treadmill gait compared to a slow walk (Nilsson and Thorstensson, 1989). Moreover, more noise on force data can be created while the instrumented treadmill was on fast speed due to the combination of more produced electrical noise and mechanical noise in the system (Sinitski, Lemaire and Baddour, 2015) thus

disturbing the calculation of the actual joint moments which were used as target outputs. The pattern of noise in joint moment could interfere with the ANN's ability to model the true kinematics-kinetics relationship while training.

There were many previous studies which proposed various machine learning or ANN algorithms to predict joint moments in gait from a variety of inputs e.g. EMG signals, GRFs and kinematics obtained from the standard optical motion capture system and revealed good results from their research strategies. This suggests the capability of the ANN approach in general to be able to model a complex relationship between a range of inputs captured by different methods as long as the inputs relate to the target outputs. In our study, we considered that kinematics data directly related to joint moments, so we decided not to incorporate another type of signals to train the ANN. However, from the variability of the predicted KAMs, there is a crucial area to improve the performance of the FFANN to give more reliable prediction.

Importantly, the results from this chapter confirmed that joint angles obtained from the Xsens provided sufficient information content to describe an individual's gait, despite the suboptimal match between the joint angles obtained by the Xsens and the Vicon system shown by the low CMC values in chapter 4. The recognition of biomechanical and mathematical relationship between joint angles obtained from the Xsens and the corresponding target KAMs by the FFANN indicated a legitimate expression of the joint movements of gait by the IMU system which should theoretically be created by the forces that reacted to the joints. Therefore, the Xsens system could be considered as a motion capture system that has its own biomechanical description to quantify human walking.

Chapter 6: Enhancing the FFANN performance for knee abduction moment estimation by leave-one-out cross validation and data randomisation

Background

The ability to generalise is an indicator of a successful artificial neural network (ANN) model (Halilaj et al., 2018). It means that the network can interpolate the new input patterns to predict an accurate output (Zurada, 1992). In other words, the ANN works successfully to estimate the output that has never been seen by the ANN during training. In the early studies of this project, the FFANN showed its efficiency in generalisation by being able to predict the unseen knee abduction moment when it was trained by using joint angles obtained from an inertial measurement unit as input during gait. Step by step, new sets of gait data were introduced to the FFANN to examine the prediction ability of the model. Some evidence was found that the FFANN could generalise well if input-target pairs similar to the ones used in testing were included in the training process.

Data splitting is a technique to balance and minimise the variance and bias amongst data that are used to train the ANN. It, therefore, improves the generalisation ability for an ANN model (Reitermanová, 2010). Moreover, with the data that were split by an appropriate method, the generalisation can be assessed by the testing part of the ANN training process where the unseen data are used to evaluate the ANN performance. One of the most well-known methods of reducing bias, K-fold cross validation, is a technique where the data are divided equally to *k* parts and then k-1 folds of the data are used for training the ANN and the remaining fold is used for testing the ANN's performance (Ghojogh and Crowley, 2019). Each part of the data then will be used for testing by assigning another slice of the data for testing, in k steps. The advantage of this method is that the training and testing set of data are systematically rotated in every k^{th} training process which should minimise bias and

variance because overall all data are used both for training and testing in a balanced way.

For a small group of data, leave-one-out cross validation can be applied. The method is similar to K-fold validation, only one set of the sample is reserved for testing (Myles et al., 1997). According to the prior results (chapter 5) when the leave-one-out cross validation was utilised, poor performance of the FFANN was observed. The explanation could be that each fold comprised of gait data collected from a particular participant who had different body mass, height and body anthropometrics, for example the leg length that can provide a different kinematics and kinetics of gait. Consequently, the FFANN was expected to generalise to a set of unseen data beyond the scope that the FFANN was trained on. Apparently, bias and variance amongst data need to be minimised to enhance the usage of K-fold cross validation. One of the efficient methods of data splitting is simple random sampling (SRS). This most common sample selecting technique provides an equal and uniform distribution of the dataset leading to the minimum bias in data selection. However, it is suitable only with less complex data rather than the high complexity (non-uniformly distributed) type (Reitermanová, 2010).

In the previous chapter, the FFANN was capable of prediction the KAM when joint angles were used as inputs to train the network and data of each participant were left out to test the performance. However, there was a lack of consistency of the results which indicated some restriction of the performance of this particular FFANN to generalise to new gait data. The differences of predicted KAMs obtained from the FFANN compared to the target KAM amongst data from each participant at three walking speeds, subsequently, were the essential issue to be improved. The main

focus of the study was to evaluate the effect of randomising the input-target pairs of data using the simple random sampling technique. In addition, more participants were included and gait data of each participant were left out during training to evaluate the performance of the FFANN in order to estimate the KAM from randomised joint angles at the testing part of the data (leave-one-out K-fold cross validation).

Materials and Methods

The study took place in the Movement Function Research Laboratory (MFRL) at Tom Reilly Building, Liverpool John Moors University (LJMU).

Research participants

Healthy volunteers aged between 18-35 year who have no known gait problems, were recruited from staff and students in the Faculty of Science, LJMU. The study was approved by the university research ethics committee (reference number 18/SPS/005). Gait data were collected from 19 participants. Written informed consent was obtained prior to data collection.

Participant preparation

The participants were prepared in the same fashion as described in chapter 5, the anthropometric data were measured and recorded following the instruction by Xsens. Seven MTw2 data trackers (Xsens Technologies B.V., Netherlands) were attached to the participant at sacrum, the lower third of lateral aspect of both thighs, proximal third of medial surface of the lower legs using elastic Velcro straps. The foot trackers were placed at the dorsal side over the shoe tongue and were secured by a self-adhesive elastic bandage. Followed by the attachment of 26 reflective markers for the HBM model (van den Bogert et al., 2013) (figure 3.3).

Data collection

After Xsens sensors calibration was completed at an unoccupied area, the participant was moved to the instrumented split belt treadmill. Then the HBM model was calibrated and streamed through Vicon Nexus 2.5 (receiving the reflective signals from 12 Vero cameras, Vicon, Oxford Metrics Group Ltd, UK) into a custom D-Flow application (D-Flow software, Motekforce Link, the Netherlands) for being further processed by Gait offline analysis tools (GOAT, Motekforce Link, the Netherlands). Both systems were synchronised following the technique described in appendix 3.

With the same process described in chapter 5, the participants were asked to walk at three different walking speeds which started with the self-selected comfortable speed followed by fast and slow speed respectively.

Data processing and analysis

Data extraction from the original data files

The first 2000 data points of the instantly processed kinematics data composed of the three orthogonal components of right hip, knee and ankle angles which were directly extracted from saved files in the Xsens MVN Analyze software. One set of gait data from each speed was included from each participant. The extracted kinematics data were, thereafter, up-sampled through a Matlab script to 120 Hz using the spline function (appendix 6) to be equivalent with the corresponding kinetics data (KAMs) sampled at 120 Hz in D-Flow. Data recorded from D-Flow were transferred to GOAT where both kinematics and kinetics data were computed for all participants. Knee abduction moment of each set of gait data that corresponded to the extracted Xsens joint angles were selected and filtered using a second order Butterworth filter at 6 Hz cut off frequency. Individually, there were 10 gait variables created including nine

variables of joint angles: the right side hip, knee, and ankle in all planes of movement that were obtained from Xsens sensors (flexion/extension, abduction/adduction, and internal/external rotation) and one variable of right KAM obtained from GOAT, at each walking speed. All variables of gait data of an individual were gathered to a file arranged as normal, fast and slow speed respectively thus making 30 variables for one participant.

Leave-one-out cross validation to prepare input and output pair for each FFANN architecture

A custom Matlab (Mathworks Inc., MA, USA) script was created for leave-one-out cross validation to generate an individual set of gait data a chance to test the FFANN performance (appendix 7). Table 6.1 demonstrates the leave one out cross validation and the data randomisation technique applied in this study. The variables were arranged from joint angles of normal speed, KAM of normal speed, joint angles of fast speed, KAM of fast speed, joint angles of slow speed and KAM of slow speed respectively. Then data from each participant at a walking speed were concatenated beginning with the data of participant 1 and continuously to participant 19 The data of the last participant of the set were used for testing the FFANN. Similar to chapter 5, nineteen folds of data were created. Overall, there were 19 sets of data created with each set comprising of 30 rows (10 rows per walking speed: nine rows for angles and rotations of hip, knee and ankle joint, one for the target KAM) with the different order of participant to make the leave-one-out cross validation for the FFANN performance evaluation. The data were then systemically randomised in a separate Matlab script described below, thus making each set of data in each fold include gait data (as a data point) obtained from any participant. Therefore, the FFANN was being

trained by variation of the healthy gait as well as being tested by a variety gait data. This aimed to reduce variance and bias of the gait data obtained from such small group of participants in order to improve the FFANN's performance.

Table 6.1 The demonstration of the leave-one-out cross validation with 19 folds (F1-19) that was applied to the FFANN training.

A.																	Tr_o	V_o	T_o
F1	*F2*	*F3*	*F4*	*F5*	*F6*	*F7*	*F8*	*F9*	*F10*	*F11*	*F12*	*F13*	*F14*	*F15*	*F16*	*F17*	*F18*	*F19*	
P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P1	
P1	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P2	
P1	P2	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P3	
									- - -										
P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	

B.																	Tr_{rm}	V_{rm}	T_{rm}
F1	*F2*	*F3*	*F4*	*F5*	*F6*	*F7*	*F8*	*F9*	*F10*	*F11*	*F12*	*F13*	*F14*	*F15*	*F16*	*F17*	*F18*	*F19*	
P1-19 2	P1-19 3	P1-19 4	P1-19 5	P1-19 6	P1-19 7	P1-19 8	P1-19 9	P1-19 10	P1-19 11	P1-19 12	P1-19 13	P1-19 14	P1-19 15	P1-19 16	P1-19 17	P1-19 18	P1-19 19	P1-19 1	
P1-19 1	P1-19 3	P1-19 4	P1-19 5	P1-19 6	P1-19 7	P1-19 8	P1-19 9	P1-19 10	P1-19 11	P1-19 12	P1-19 13	P1-19 14	P1-19 15	P1-19 16	P1-19 17	P1-19 18	P1-19 19	P1-19 2	
P1-19 1	P1-19 2	P1-19 4	P1-19 5	P1-19 6	P1-19 7	P1-19 8	P1-19 9	P1-19 10	P1-19 11	P1-19 12	P1-19 13	P1-19 14	P1-19 15	P1-19 16	P1-19 17	P1-19 18	P1-19 19	P1-19 3	
									- - -										
P1-19 1	P1-19 2	P1-19 3	P1-19 4	P1-19 5	P1-19 6	P1-19 7	P1-19 8	P1-19 9	P1-19 10	P1-19 11	P1-19 12	P1-19 13	P1-19 14	P1-19 15	P1-19 16	P1-19 17	P1-19 18	P1-19 19	

(A.) First, gait data of 17 participants (F1-17) were used to train the FFANN (Tr_o), one for validation (F18, V_o) and one for testing the FFANN performance (F19, T_o). Data of each participant were used to test the network. (B.) Second, the gait data were randomised and then divided into 19 folds to be used as training (F1-17), validation (F18) and testing (F19). Each fold contained gait data from every single participant due to randomisation (P1-19 n where n = 1-19).

(Subscript *o* represents original gait data and *rm* represents randomised data)

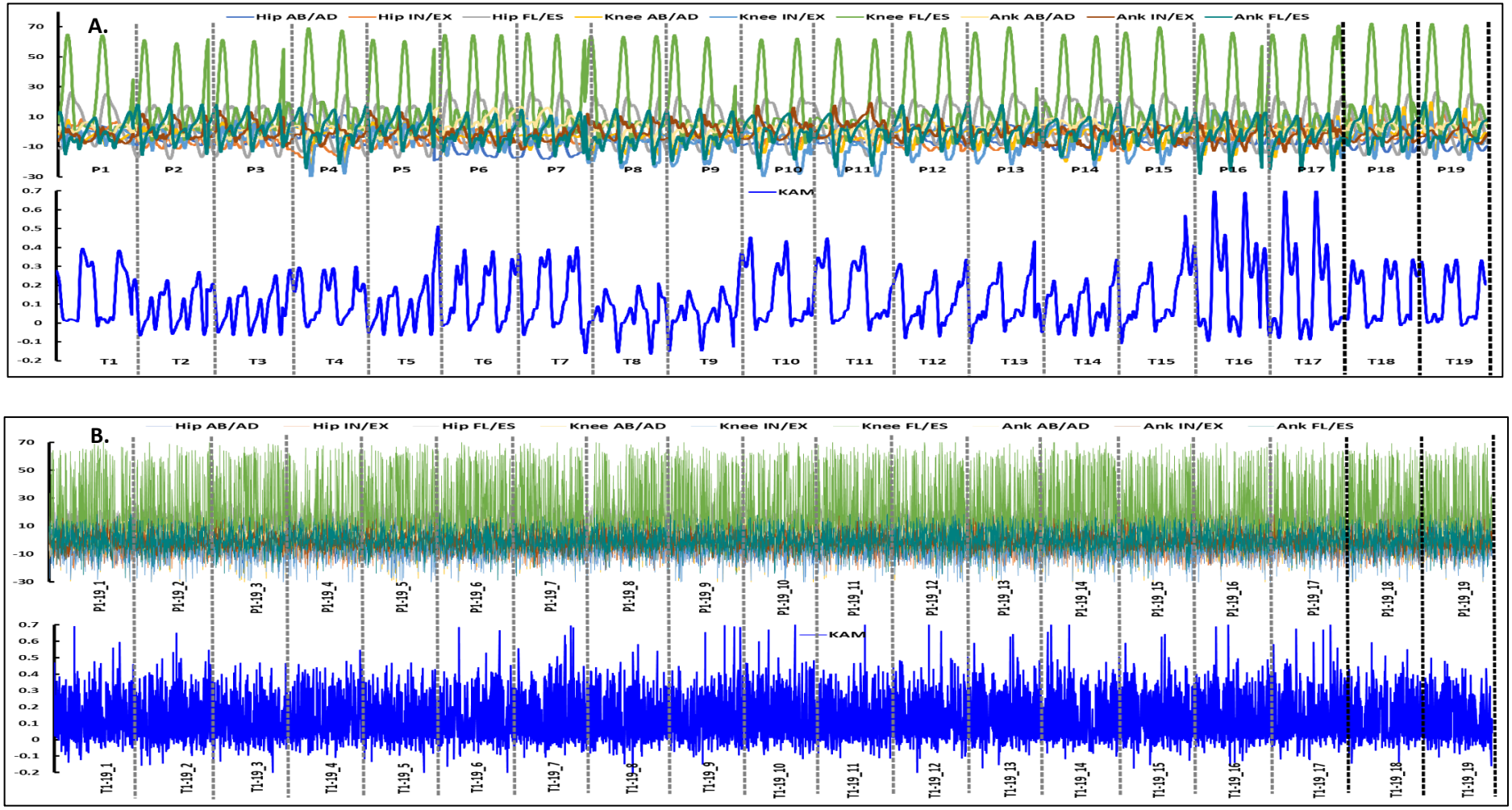

Figure 6.1 A. The original gait data of each participant were prepared for the FFANN training (chapter 5). Joint angle obtained from Xsens were used as input data: P (top) to predict the target KAM: T (bottom). Data of 17 people were used to train the FFANN, one for validation and one for testing the performance, according to leave-one out cross validation data of each participant were used to test the performance (unseen data). Randomised data are shown in B. Both inputs and output were systematically randomised. Each input-output pair was reordered form the original position (A.), the random data were also divided into 19 sets which possibly contained data pairs of all 19 participants. Similarly, the first 17 sets were used to train the FFANN, one for validation and the last set for testing the FFANN thus the unseen data from an individual was still used to test the network. The figures demonstrate the data preparation of one speed, the same method was also used for the other two speeds.

Training the FFANN

Another custom Matlab script (appendix 8.) was created for FFANN training when each fold contained data of one of the 19 participants. Before the training began, the SRS method was implemented to the script using the random permutation (randperm) function (figure 6.1). The data points in the 19 folds of the gait data were randomised to a new order which applied to all 30 rows (10 rows for each walking speed). The script was written to randomise the data using the first seed of random generator number (rng) function to secure the same order of the new rearranged data set for each individual fold. The Levenberg-Marquard algorithm was used for this FFANN training with one hidden layer of 18 hidden neurons. The nine variables of joint angle and rotation were presented to the ANN as input and the actual KAM obtained from GOAT were the target output. The data of each walking speed were separately used to train the FFANN.

In the training process, each FFANN architecture (of each left out fold) was run 20 times (20 different seeds for randomisation of initial weights) in order to identify the best performance of that particular FFANN which was indicated by the lowest mean square error (MSE). The FFANN model at each speed was run separately, taking rows 1-10, 11-20 and 21-30, within the same Matlab script. Next, the three FFANNs (for normal, slow and fast speed) were carried out with the best seed number obtained previously. The predicted KAMs were obtained however they were still in the random order and while this would allow a numerical analysis, the visualisation of the KAM would not be possible. Subsequently, the predicted KAMs at all walking speeds needed reversing to their original position and represented as usual KAM pattern (Peralta, Gutierrez and Sanchis de Miguel, 2009) for comparison with the measured

KAM. Normalised root mean square error, root mean square error and SMAPE were calculated to quantify the difference between target KAM and predicted KAM. Pearson's correlation coefficient was used to determine the relationship between the target and the predicted KAM. The stopwatch function of Matlab (tic and toc) was operated in the FFANN training to measure the operation time. In general, nineteen Matlab scripts were written to carry out the KAM prediction of each left out fold while the data of three different walking speeds were trained separately in the same Matlab script.

Results

The average time of KAM prediction at all walking speeds together was 38.63 ± 11.01 (range 26.25-72.12) seconds (ASUS laptop X455L series, Intel Core i3 4030U, 1.9 GHz, 4 installed RAM with Windows 10 Pro 20H2 version). The average RMSEs of the training part were 0.068 ± 0.002 Nm/Kg, 0.064±0.002 Nm/Kg and 0.083 ± 0.004 Nm/Kg for normal, fast and slow speed respectively. At the testing part where the FFANN did not see the gait data before, the average RMSEs were 0.063 ± 0.025 Nm/Kg, 0.061 ± 0.017 Nm/Kg and 0.072 ± 0.040 Nm/Kg for normal, fast and slow speed. The average NRMSEs at the testing part were 11.65 ± 3.861% at normal speed followed by 10.71 ± 2.759% for fast speed and 13.82 ± 6.127% for slow speed (figure 6.2). The average SMAPEs were at 35.95± 6.888 %, 32.02 ± 5.780 % and 36.39 ± 6.431 % for normal, fast and slow respectively.

There was a strong positive correlation between the average measured KAM and the average predicted KAM amongst all walking speeds with the r values between 0.86-0.9. The average correlation coefficients (r) were 0.86 ± 0.176, 0.90 ± 0.068 and 0.87± 0.141 for normal, fast and slow speed respectively. The r values were approximately 0.99 when they were calculated over the normalised gait cycles and stance phases (table 6.2).

The Bland-Altman plots visualise the distribution of the difference as a function of the average KAMs of all participants are shown in figure 6.3. The plots were relevant to the r values demonstrated by a very small bias (mean of the differences) between the measured and the predicted KAM at -0.0002, 0.014 and 0.001 Nm/kg for normal, fast and slow speed respectively. There were good agreements between the measured and predicted KAM in all speeds, as shown in the scatter plots, suggesting that the

majority of the differences between measured and predicted KAM were within the limit of agreement, especially in the gait cycle and stance phase data. However, the predicted KAM of the fast speed showed the best results compared with the other two speeds seen by the narrowest of the limits of agreement.

The paired t-test showed a non-significant difference between the predicted KAM and the target KAM at fast speed walk when the FFANN was trained with the randomised data ($p=0.475$). There were significant differences between the predicted KAM and the target KAM in the other walking speeds ($p<0.05$).

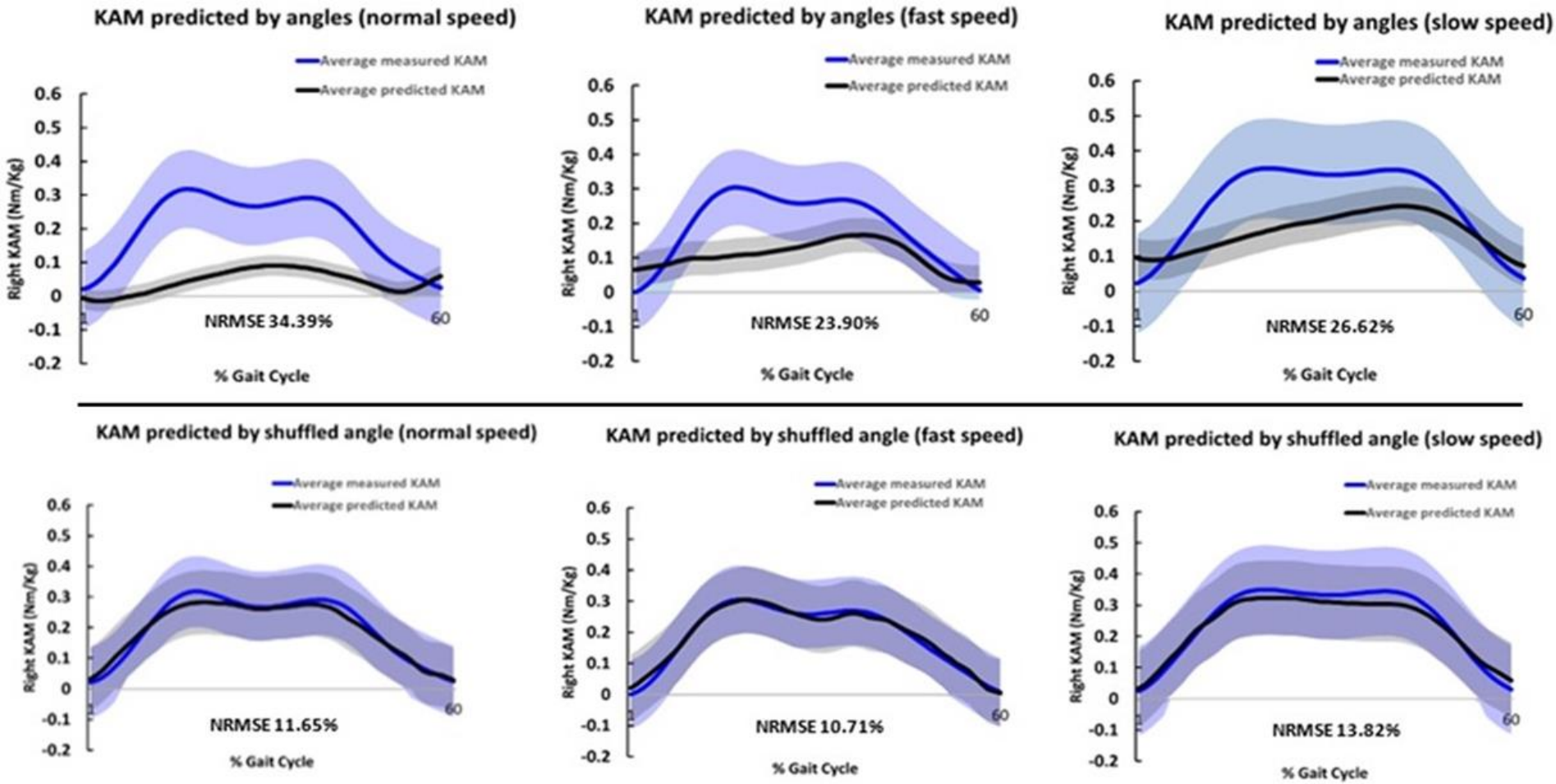

Figure 6.2 The comparison between the target KAM (blue) and the predicted KAM (black) during stance phase (60% of the gait cycle). Less similarity is shown at the top row as the KAMs were predicted using the non-randomised gait data. A better match of the KAMs is observed at the bottom row which shows the results when training the FFANN by randomised gait data.

Table 6.2 Correlation coefficient (r), slope of line of best fit and bias and standard deviation (SD) between measured and predicted KAM using randomised data for training.

Values	**r**	**Slope of line of best fit**	**Bias (mean of the differences ± SD, Nm/kg)**
Normal speed (2000 data points)	0.85	0.74	0.0014 (±0.0153)
Normal speed (Gait cycles)	0.99	0.89	0.0018 (±0.0160)
Normal speed (Stance phases)	0.99	0.86	0.0075 (±0.0178)
Fast speed (2000 data points)	0.90	0.84	0.0002 (±0.0134)
Fast speed (Gait cycles)	0.99	0.97	-0.0005 (±0.0104)
Fast speed (Stance phases)	0.99	0.93	-0.0002 (±0.0119)
Slow speed (2000 data points)	0.84	0.64	-0.0010 (±0.0192)
Slow speed (Gait cycles)	0.99	0.84	-0.0020 (±0.0238)
Slow speed (Stance phases)	0.99	0.82	0.0117 (±0.0210)

Bland-Altman plots (Randomised angles)

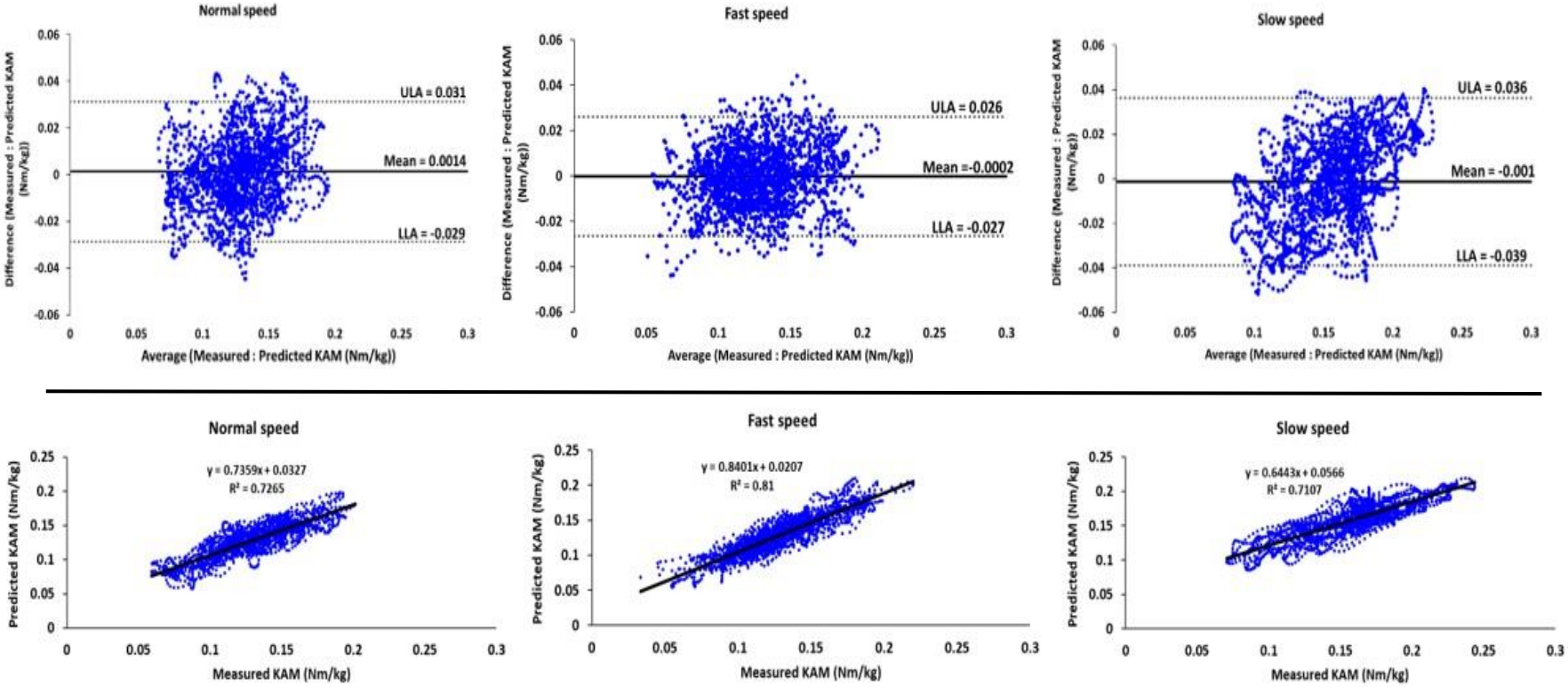

Figure 6.3 Bland-Altman plots (top) demonstrate good agreement between the target KAM (measured) and the predicted KAM by the FFANN trained with the randomised joint angles. The bottom row shows strong relationship between the target and the predicted KAM.

Discussion

Training the FFANN with systematically randomised gait data showed superior results compared to when the FFANN was trained by gait data of individuals without randomising (chapter 5), even though the method required longer training time (averagely, 38.6 seconds compared to 6.29 seconds). At the same speed, the NRMSE values calculated in this chapter were approximately 50% smaller than the NRMSEs shown in the previous chapter. Superior results were found with the r values and SMAPEs as well. The strong relationship was observed with average r values at the testing part from all walking speeds. Moreover, even stronger relationship was found when focusing on the normalised gait cycle and especially when the r was calculated at stance phase. These are likely to be explained by using the mean of all the normalised gait cycles as this removes the cycle-to-cycle variability of gait. A further reduction of the amount of data to the stance phase only resulted in even better performance as the ANN had to fit its multivariate solution to less variable data. Reducing training data dimension has been reported to be beneficial in order to improve the model accuracy (Sivakumar et al., 2016).

The results exhibited the ability of the FFANN to generalise to the unseen set of gait data when the FFANN was trained by the randomised data. Good predictive results were reported in a previous study using a long short-term memory based recurrent ANN trained with the randomised kinematics, gait velocity and anthropometric data to predict joint moments. For the unseen participant, the average NRMSE of all joint moments was 11.33% with a strong correlation between the actual joint moments and the predicted joint moment at $r > 0.9$. In particular, the NRMSE of the KAM was comparable to this recent study at 14.96% (Mundt et al., 2018).

However, in practice, the FFANN would be expected to be able to predict KAM of a new participant that the FFANN has never seen before which referred to the ability to generalise to a new individual (Halilaj et al., 2018). From chapter 5, the FFANN was used to predict the KAM of 15 participants. The FFANN was trained with hip, knee and ankle joint angles in all three planes of motion to predict the KAM. The leave-one-out cross validation was used and data of each individual were used to estimate the model's performance (at the testing part of the training process). Even though the results from the previous chapter showed that the FFANN was able to predict individual unseen KAM, the NRMSEs of the testing part varied from low to notably high NRMSE from a prediction of some individuals. The NRMSEs of the testing part at normal speed walk ranged between 14.69%-55.71%, between 16.40%-35.00% for fast speed and between 14.32%-41.86% for slow speed. This reflects that the gait pattern of a new person can be much different from the 13 others that were used to train the ANN.

The question still presents whether the FFANN can generalise well when it is used in real life. The answer would be that the FFANN should be able to predict the unseen KAM of an unknown participant if the FFANN was trained by the gait data from a large number of participants, as many as to present all kinds of gait patterns of the world population to the FFANN. The FFANN will then be able to recognise the new gait pattern and will provide more accurate prediction of unseen KAM. However, when there were only 13 gait patterns to train the FFANN, it was inevitable to evidence a poor generalisaton due to the variances amongst each gait pattern. To reduce those variances, in case of having a limited number of gait patterns, randomising the gait data could effectively even the variability of the gait data thus balancing the data in

each fold to be more comparable with less bias (individuals' difference) (Reitermanová, 2010).

A better performance of the FFANN observed from the results in this chapter compared to the former chapter could be due to the FFANN was trained by a more uniform gait patterns, therefore it could predict more precise KAMs. Also, the participants' gait patterns were not violated by the method and the principle of testing the performance with the unseen gait data was still preserved as each fold of the data equally had potential to contain data of every single participant therefore the gait data from all participants were used to test the FFANN performance in a balanced way. The advantages of randomising data before training the FFANN were firstly, to eliminate the issue of having a small number of data or participants that reflect less variety. Secondly, it was proven that the FFANN can be used to predict the KAM of gait if the input-output pairs were formulated properly. The FFANN worked quite well in order to predict the KAM if the individual difference was removed. The drawback of the randomisation of the data in this present study was that the technique might not yet be practically used as in the gait laboratory the participant visits the laboratory in person presented as an unseen gait data. Ideally, the FFANN will be expected to predict such unseen gait data instantaneously. Therefore, the FFANN needs to be trained with data of more participants, perhaps up until the point where the FFANN trained by gait data of individuals and the FFANN trained by the randomised data can provide equivalent results. Furthermore, the two studies were conducted using data from unimpaired gait to train the FFANN, however there are pathological gait patterns that need to be introduced to the FFANN,

especially when the method is applied in practice in the future. Therefore, obviously, more participants are required to maximise the FFANN performance.

Chapter 7: Enhancing the FFANN performance for knee abduction moment estimation using marker coordinate data as input and the effective number of hidden neuron and input variables

Background

The previous results have shown that joint angles obtained from Xsens sensors alone can be used to successfully train the FFANN to predict KAM during gait. The randomisation technique and appropriate data splitting were proven to increase prediction ability and generalise to the unseen set of data. Joint angles were used as input to train the FFANN in this project since they are a crucial parameter in gait analysis that can be obtained instantly from the IMU system which offers flexibility for gait analysts to analyse an individual's gait without the requirement of laboratory-based motion capture equipment (Sivakumar et al., 2016). However, from the previous part of the study, there was a possibility to improve the FFANN performance since only predicted KAM at fast speed were comparable with the target KAM. Some other methods, for instance using a different set of inputs to train the network should be considered.

In practice, when gait analysis is produced, the joint angles are reconstructed in a three-dimensional (3D) space using signals from reflective markers that were attached on a participants' body according to a particular 3D gait model that had been chosen. Body segments, thereafter, are created from the model and joint angles of rotation are calculated from the orientation of a lower body segment relative to the adjacent proximal segment specifically described for each gait model. To identify 3D joint angles (relative orientation of neighbouring segments), the order of segment rotations about axes needs to be determined in the 3D space beginning with an axis then the following axes which depend on the first axis, therefore, the possible sequence of axes could be any of XYZ, XZY, YXZ, YZX, ZXY, or ZYX. The axes' sequences

are known as the Cardan rotation sequences, as a result there is a variety of possibilities to define joint angles (Lees, Barton and Robinson, 2010).

Generally, the sequence is arbitrarily chosen or is chosen following the clinical conventional description (Baker, 2001). Although, all of the rotational sequences provide mathematically equivalent joint angles: an evidence from a previous study in football motion analysis has shown that movement data collected from those potential rotational sequences, one that has flexion-extension plane at the second order showed the greatest divergences from the other data set (Lees, Barton and Robinson, 2010).

Despite a commonly used rotational sequence to quantify joint angles of gait, a choice has to be made which rotational sequence to use but the reasons for this choice vary, leading to different angular representations of the same movement. Therefore, it might be difficult to rationalise selection of one of the possible joint angles that were processed from the typical gait analysis as the best information to train the FFANN due to the possibility of being calculated from various Cardan rotation sequences. Alternatively, the signals from marker coordinates could perhaps, represent more unambiguous gait information since they are the original signals that are directly measured from a participant's gait without the need for further choices about the calculation process of angles. The marker coordinates were reported to be able to represent gait data for healthy and pathological gait analysis (Federolf, Roos and Nigg, 2013). A previous study showed an advantage over using joint angles for validating the Movement Deviation Profile (MDP) to indicate gait deviations in the progressive condition of alkaptonuria (Barton et al., 2015).

One important factor of a successful output prediction by an artificial neural network is to choose an appropriate set of input variables. In FFANN training, it is necessary that to some extent, both input and target output variables should relate to each other for the FFANN to be able to capture the non-linear relationship between the input-target pairs (Fernando, Maier and Dandy, 2009). Using marker coordinates as input to train the network could give the network another chance to predict KAM better as the marker coordinates are the most fundamental raw pieces of information obtained from a motion capture system. Besides having a relevant set of input variables, the quantity of the input is also vital to enhance the FFANN performance. An excessive number of inputs will cause the model unnecessary work, for instance, the size and complexity of the model will require more memory to operate the task and also disturb the model calibration and weight adjustment.

Another factor that influences the FFANN performance is the number of hidden neurons. It has been known that one hidden layer is sufficient for the neural network to accomplish a prediction task (Sivakumar et al., 2016). However, too few or too many hidden neurons could significantly affect the FFANN performance; the former would slow down the learning process and the latter would limit the performance through overtraining (Agatonovic-Kustrin and Beresford, 2000). This study will focus on the feasibility of using marker coordinates as inputs to train the FFANN and to find an appropriate ratio of hidden neurons for efficient FFANN training to predict knee abduction moment during gait.

7.1 Minimum number of input variables and the suitable number of hidden neurons

Data processing and analysis

Data extraction and pre-processing

The gait data obtained from the Xsens sensors were exported as a .c3d file, containing virtual markers generated by the Xsens software. The data of virtual marker coordinates in X, Y and Z axes were selected from the virtual markers of the right leg including ASIS, PSIS, greater trochanter, lateral knee epicondyle, tibial tuberosity, heel and toe using Visual3D V.6 software (C-Motion, MD, USA); they were then exported as a .txt file to be another set of input for the FFANN training. The marker coordinate data were up sampled from 100 Hz to 120 Hz using the spline function in Matlab as in the prior chapter, to be equivalent with the KAM output data. Similarly, the marker coordinate data were divided in 19 folds in accordance with data from each participant the dataset would be systematically randomised in the next step. As well as the former chater, the marker coordinate data of all walking speeds were included in the set of input-target output data.

The FFANN training strategy

In order to identify the appropriate number of the input variables and hidden neurons, an experiment has been conducted in this study. Firstly, to investigate how efficient the FFANN was in predicting the KAM when the FFANN was trained by input data selected from a specific number of the markers. Secondly, with a specific number of inputs, to find what the suitable number of hidden neurons that would provide the most efficient performance of the FFANN was to be. Therefore, in this

study, gait data were extracted from all 19 participants and only data from one participant (28 years old female, body height 1.67 metres and body mass 65 kg) was chosen to test the FFANN's performance (with unseen data) for the experiment. The maximum number of the marker coordinates was limited at seven to be equivalent to the conventional gait model.

The following strategy was applied (figure 7.1)

1. Trained the FFANN with data from all seven markers (ASIS, PSIS, greater trochanter, lateral knee epicondyle, tibial tuberosity, heel and toe) as they were markers that represented each segment of one lower extremity (pelvis, thigh, leg and foot) and resemble the conventional gait model's markers. There were 21 input variables to train the FFANN (X, Y and Z coordinates of the seven markers).
2. Trained the FFANN with data from five markers (ASIS, greater trochanter, lateral knee epicondyle, heel and toe) to examine the effect of reducing the number of input variables when the direct gait data from marker coordinates were used. There were 15 input variables to train the FFANN (X, Y and Z coordinates of the five markers).
3. Different ratios of hidden neurons were used to train the FFANN model of the seven markers and the five markers.
   3.1 An FFANN trained with the number of hidden neurons at a half of input variables and target output (11 for the seven markers (21+1)/2 and 8 for the five markers (15+1)/2).

3.2 An FFANN trained with the number of hidden neurons at two third of input variables doubled (28 for the seven markers (2/3 of 42) and 20 for the five markers (2/3 of 30).

3.3 An FFANN trained with the number of hidden neurons at double of input variables (42 for the seven markers and 30 for the five markers).

3.4 An FFANN trained with a high number of hidden neurons at 56 for the seven markers and 40 for the five markers (doubled input (42 and 30) plus a third of itself (plus 13 for seven marker and plus 10 for five markers).

4. At this stage, the most effective ratio of hidden neurons could be identified and followed thereafter in the next steps of the study.
5. Trained the FFANN with data from four markers (ASIS, greater trochanter, lateral knee epicondyle and toe) to examine the effect of reducing input number when the direct gait data from marker coordinates were used. There were 12 input variables to train the FFANN (X, Y and Z coordinates of the four markers).
6. Finally, trained the FFANN with data from three markers (greater trochanter, lateral knee epicondyle and toe) to examine the effect of reducing input number when the direct gait data from marker coordinates were used. There were 9 input variables to train the FFANN (X, Y and Z coordinates of the three markers).

A new Matlab script (appendix 9) was customised to train the FFANN described in this strategy. All variables were randomised by the randperm function and to maintain the identical randomised order of each FFANN the first randomised order was chosen for the data preparation. Gait data of three walking speeds were used to train the FFANN separately. The temporal order of the predicted KAM was reversed back for

visualisation of the usual KAM pattern and FFANN performances were computed and recorded as NRMSE, RMSE and SMAPE values.

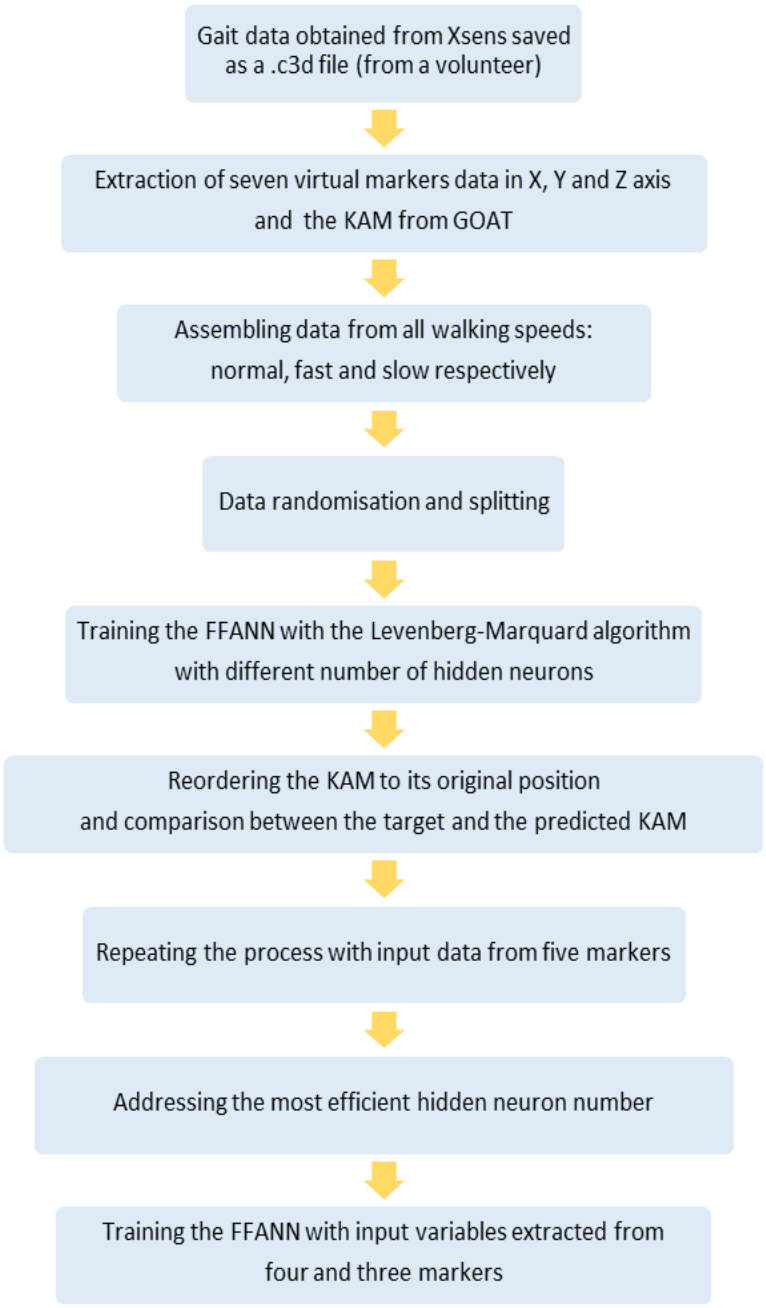

Figure 7.1 The flow chart illustrates the order of data preparation and the strategies that were used to train the FFANN with the virtual marker coordinates obtained from the Xsens sensors. Gait data of one volunteer (28 years old female) were used for testing in this study.

7.2 The effectiveness of using a minimum number of input variables in FFANN training for knee abduction moment prediction

Data processing and analysis

Input variables selected from the marker coordinates

For further examination of the FFANN performance, minimum number of input variables were used to train the FFANN. The number of the variables were selected from the previous part of the study, therefore coordinate data from four (ASIS, greater trochanter, lateral knee epicondyle and toe) and three (greater trochanter, lateral knee epicondyle and toe) markers were used as input to train the FFANN.

A Matlab script (appendix 10.) was created in order to gather gait data from marker coordinates and to create 19 folds cross validation where each fold was the data from each participant. The data from four and three marker coordinates were prepared separately. The data up sampling and randomisation were performed similarly to the previous part of the study. Overall, 19 FFANN architectures were created to predict KAM of each participant. The predicted KAM were then reversed to the original KAM pattern at the end of the training process. The Levenberg-Marquard algorithm was used for the FFANN training with one hidden layer of 24 and 18 hidden neurons (double of the input variables for four and three marker coordinates respectively). The FFANN were trained for 20 times to identify the best performance and then the KAM prediction was carried out following the chosen best performance. The FFANN training process described in part 7.1 was applied in this part of the study for four and three marker coordinates input.

Results

7.1 Minimum number of input variables and the suitable number of hidden neurons

Training the FFANN with data extracted from seven marker coordinates required minimally five times longer operation time compared to smaller number of markers. Similarly, more operation time was required in the FFANN models having more hidden neurons. At seven markers, the operation times were 348, 575, 2,529 and 4,124 seconds for 11, 28, 42 and 56 hidden neurons respectively. On average it took 5 times less when the FFANNs were trained by data from coordinates of five markers at 38, 261, 559 and 659 seconds for 8, 20, 30 and 40 hidden neurons respectively (ASUS laptop X455L series, Intel Core i3 4030U, 1.9 GHz, 4 installed RAM with Windows 10 Pro 20H2 version).

Overall, the FFANN generalised to the unseen gait data better when the model was trained by more hidden neurons (table 7.1). At normal walking speed, using coordinate of seven markers, RMSEs were 0.026, 0.015, 0.014 and 0.013 Nm/Kg for 11, 28, 42 and 56 hidden neurons respectively. The same trend was observed for NRMSE values at 6.33%, 3.61%, 3.35% and 3.12% and the SMAPEs at 26.97%, 20.88%, 19.59% and 18.83% for 11, 28, 42 and 56 hidden neurons respectively. A strong relationship between the target and the predicted KAM was found with correlation coefficients between 0.95-0.99 amongst the prediction from the FFANN with different number of hidden neurons. Similar results were shown at the other walking speeds.

Table 7.1 The FFANN's performances when they were trained by X, Y and Z coordinates of seven and five markers with various numbers of hidden neurons (Asterix indicates the appropriate ratio applied in the next step of the study).

	Normal speed				**Fast speed**				**Slow speed**				**Training time** (second)
7 markers (21 inputs) with different number of hidden neurons (rounded)	**RMSE (Nm/kg)**	**NRMSE (%)**	**SMAPE (%)**	**r**	**RMSE (Nm/kg)**	**NRMSE (%)**	**SMAPE (%)**	**r**	**RMSE (Nm/kg)**	**NRMSE (%)**	**SMAPE (%)**	**r**	
11 *(input+output)/2*	**0.03**	**6.33**	**26.98**	**0.95**	**0.03**	**5.75**	**21.95**	**0.96**	**0.02**	**5.73**	**23.00**	**0.97**	**348**
28 *(2input - 1/3input)*	**0.02**	**3.61**	**20.88**	**0.98**	**0.01**	**3.35**	**16.13**	**0.99**	**0.02**	**4.68**	**20.45**	**0.98**	**576**
42 *(2input)**	**0.01**	**3.36**	**19.60**	**0.99**	**0.01**	**2.99**	**14.80**	**0.99**	**0.02**	**4.16**	**19.09**	**0.99**	**2530**
56 *(2input + 1/3input)*	**0.01**	**3.12**	**18.84**	**0.99**	**0.01**	**2.87**	**14.74**	**0.99**	**0.02**	**3.73**	**16.59**	**0.99**	**4124**
5 markers (15 inputs) with different number of hidden neurons (rounded)													
8 *(input+output)/2*	**0.03**	**7.35**	**27.52**	**0.94**	**0.04**	**9.77**	**32.29**	**0.90**	**0.03**	**7.99**	**26.82**	**0.94**	**38**
20 *(2input - 1/3input)*	**0.02**	**4.62**	**23.28**	**0.97**	**0.02**	**5.23**	**20.55**	**0.97**	**0.02**	**5.39**	**21.89**	**0.98**	**262**
30 *(2input)**	**0.01**	**3.58**	**21.50**	**0.98**	**0.02**	**3.95**	**18.67**	**0.98**	**0.02**	**5.13**	**20.25**	**0.98**	**560**
40 *(2input + 1/3input)*	**0.02**	**5.10**	**23.46**	**0.97**	**0.02**	**3.83**	**17.60**	**0.98**	**0.02**	**4.46**	**18.25**	**0.98**	**659**

The FFANN model that was trained by data from coordinates of five markers with different number of hidden neurons generalised well for all walking speeds, although the error values were slightly higher than the results from seven marker coordinates. At normal speed, RMSEs were 0.03, 0.02, 0.01 and 0.02 Nm/Kg for 8, 20, 30 and 40 hidden neurons respectively. The NRMSEs were 7.35%, 4.62%, 3.58% and 5.1% and the SMAPEs were 27.52, 23.28, 21.49 and 23.46 for 8, 20, 30 and 40 hidden neurons respectively. Strong relationship between the target and the predicted KAM was shown with correlation coefficients between 0.93-0.98 at all number of hidden neurons used. Similar results were shown at fast and slow speed.

Overall, the results showed that the prediction ability of the FFANN improved when it was trained using more hidden neurons. However, the improvement of the prediction did not clearly improve at the hidden neurons at more than twice the number of input variables while a considerably increased operation time was required to accomplish the training task. Therefore, in the following steps of the experiment the FFANNs were trained by input variables extracted from 12 coordinates of four and 9 coordinates of three markers using the hidden neurons at double number of the inputs (24 and 18 hidden neurons respectively). A one-way analysis of variance test (ANOVA) was performed to examine the difference between the performance of the FFANNs predicting KAM compared to the actual measured KAM. No statistically significant difference was found between the actual KAM and the KAM predicted by FFANNs trained by data of seven and five marker coordinates with all hidden neurons number ratios ($p < 0.05$) at all walking speeds. There was no statistically significant difference between the KAM predicted by the FFANNs trained

with data from three, four, five and seven coordinates with hidden neurons at twice number of the input variables ($p<0.05$).

The FFANN trained by inputs from four marker coordinates exhibited good generalisation to this particular gait pattern at RMSE 0.02 Nm/Kg, NRMSE 4.84%, SMAPE 22.93 and r=0.97 for normal speed, RMSE 0.02 Nm/Kg, NRMSE 4.98%, SMAPE 19.25 and r=0.97 for fast speed and RMSE 0.03 Nm/Kg, NRMSE 6.51%, SMAPE 23.52 and r=0.96 for slow speed. The results found when the FFANN was trained by inputs from three marker coordinates also showed good generalisation at RMSE 0.04 Nm/Kg, NRMSE 8.56%, SMAPE 35.99 and r=0.92 for normal speed, RMSE 0.03 Nm/Kg, NRMSE 7.59%, SMAPE 28.47 and r=0.93 for fast speed and RMSE 0.03 Nm/Kg, NRMSE 7.31%, SMAPE 27.92 and r=0.96 for slow speed (figure 7.2).

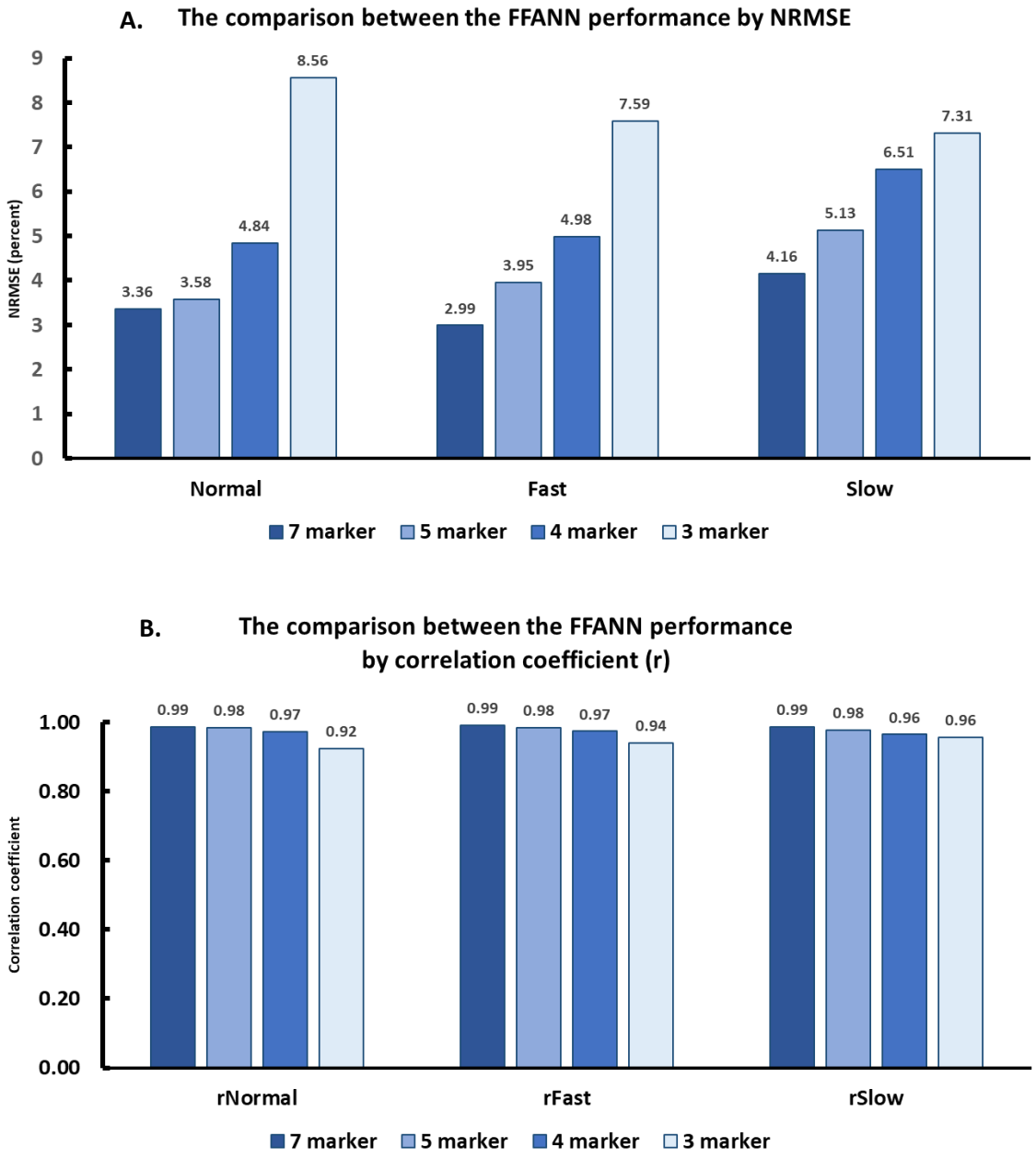

Figure 7.2 The FFANN's performance to predict unseen KAM using a variety of number of input variables extracted from X, Y and Z coordinates of seven, five, four and three markers with the same ratio of hidden neurons (double number of input variables): NRMSEs (A.) and the correlation coefficient (r) (B.). Note that prediction of the KAM with data from four markers provides close results to the seven and five markers.

7.2 The effectiveness of using a minimum number of input variables in FFANN training for knee abduction moment prediction

The networks trained by input variables extracted from X, Y and Z coordinates of three markers (greater trochanter, lateral knee epicondyle and toe) and X, Y and Z coordinates of four markers (ASIS, greater trochanter, lateral knee epicondyle and toe) could predict the KAM of gait. The results from four markers were slightly superior to using three markers, also both sets of input variables exhibited good generalisation ability towards the unseen set of data (figure 7.3). At three markers, the average time to operate the KAM prediction task was 52.58±10.65 s (the KAM of all three walking speeds were predicted by one Matlab script in a sequence). Average RMSEs at the testing part were 0.053±0.025, 0.049±0.015 and 0.059±0.041 Nm/Kg for normal, fast and slow speed respectively. The average NRMSEs were 9.76±3.30%, 8.69±2.80% and 11.27±6.58%, the average SMAPEs were 33.66±7.16, 30.31±6.37 and 34.38±6.21 for normal, fast and slow speed. Amongst the 19 participants, the generalisation of the FFANN predicting individual KAM as unseen data was varied. There was a wide range of correlation coefficient values (r) computed between the target and predicted output, especially in normal and slow speed walk, ranged at 0.14-0.98 and 0.20-0.98 respectively and 0.69-0.98 for fast speed (figure 7.4).

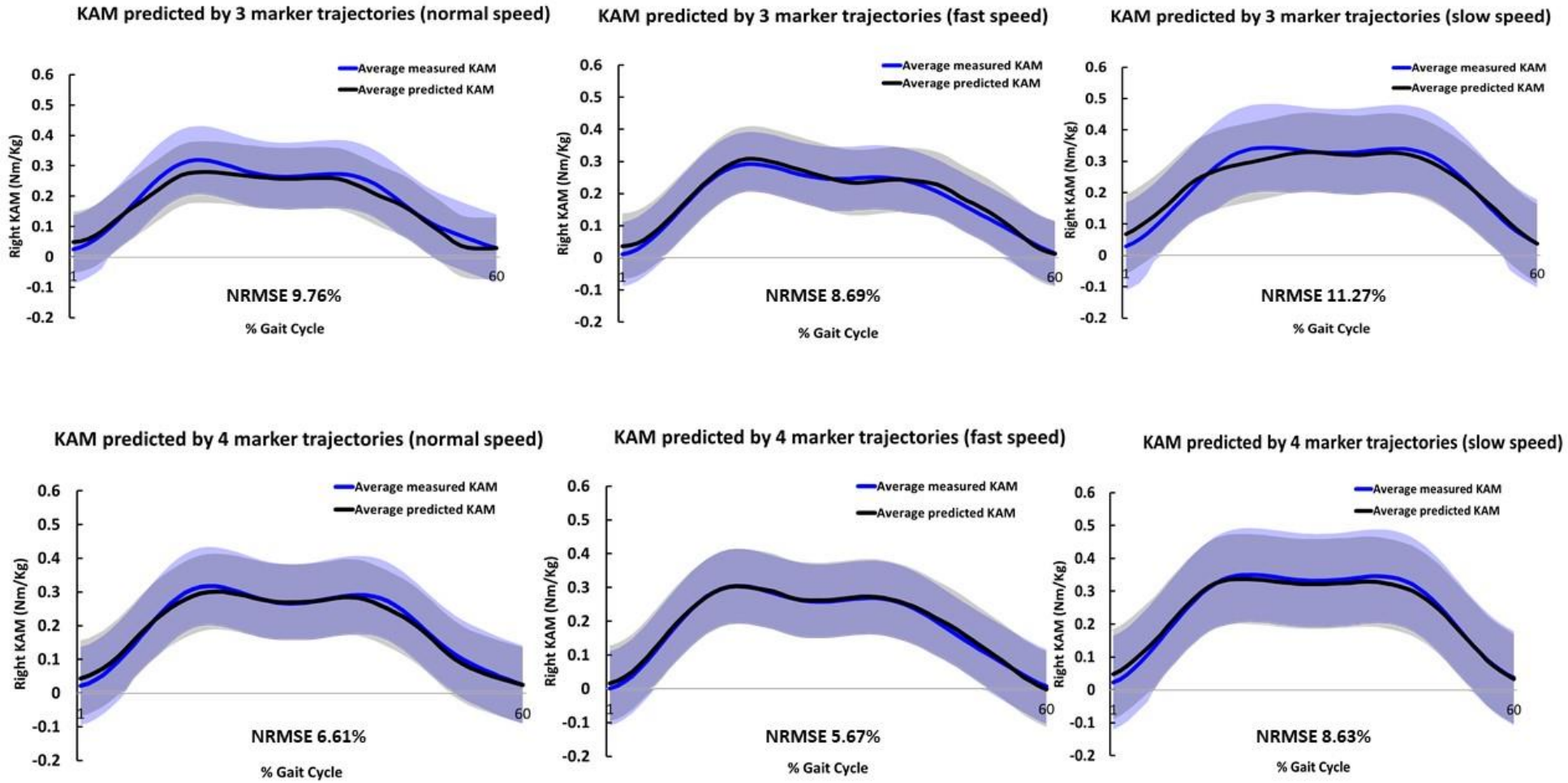

Figure 7.3 The comparison between the target KAM (blue) and the predicted KAM (black) extracting from the testing part of the FFANN at stance phase when the FFANNs were trained by data from three (top row) and four (bottom row) marker coordinates. The gait graphs demonstrate good generalisation of the model to predict the unseen data, lower NRMSEs were observed when the FFANN was trained by data of four markers compared to three markers.

Bland-Altman plots (three markers)

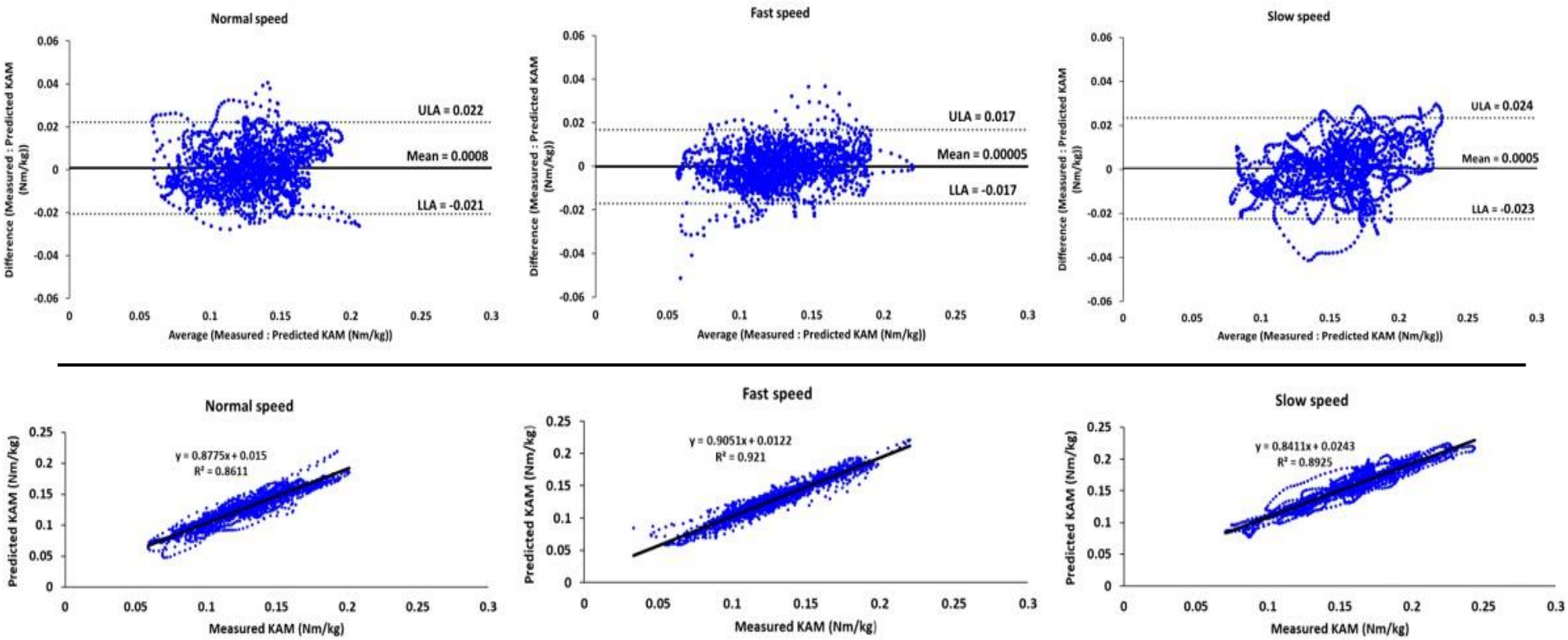

Figure 7.4 Bland-Altman plots (top) demonstrated good agreement between the target KAM (measured) and the predicted KAM when the FFANNs were trained by inputs from X, Y and Z coordinates of three markers. The prediction of the KAM at fast speed walk was the best amongst all walking speeds. The bottom row shows good correlation relationship between the target and the predicted KAM.

Bland-Altman plots (four markers)

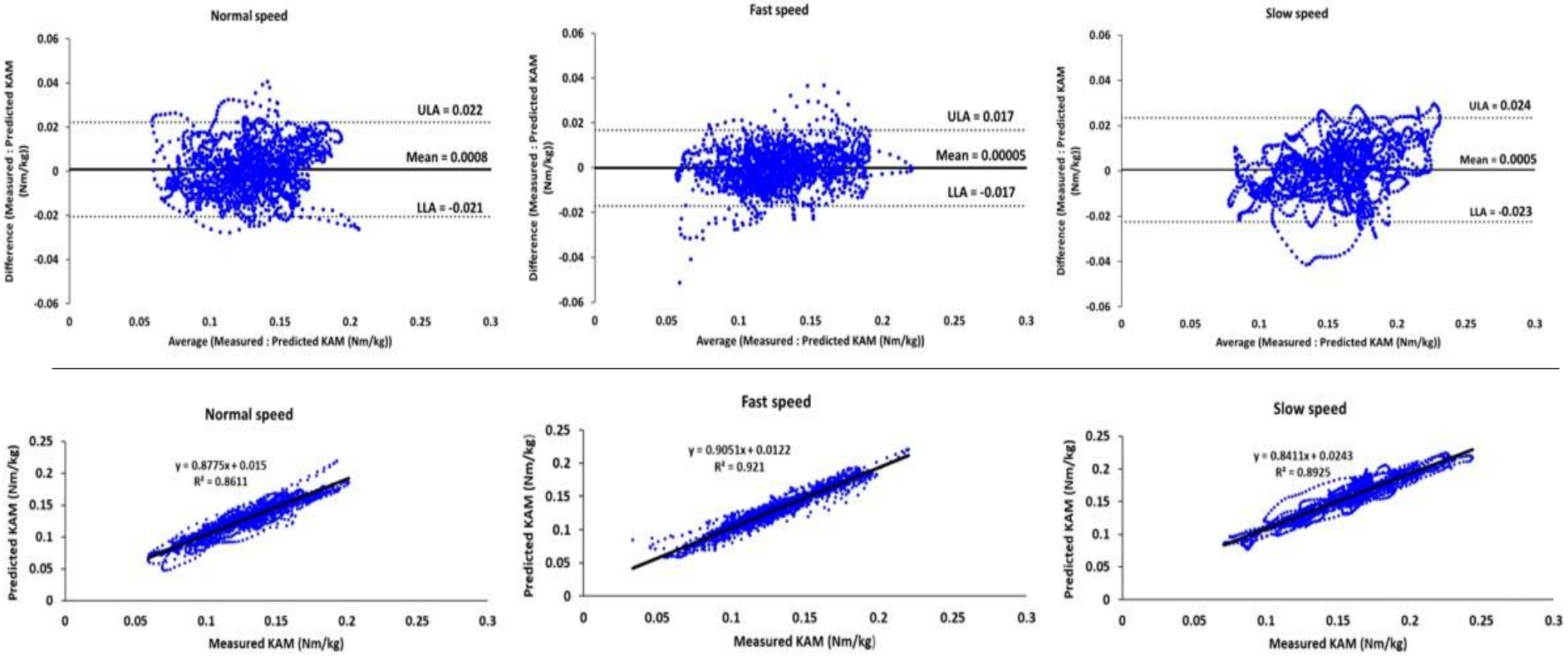

Figure 7.5 Bland-Altman plots (top) demonstrate good agreement between the target KAM (measured) and the predicted KAM when the FFANNs were trained by inputs from four markers. The better results were shown when compared with the KAM predicted by the FFANNs that trained with three markers (figure 6.7). The prediction of the KAM at fast speed walk was the best amongst all walking speeds. The bottom row shows good correlation relationship between the target and the predicted KAM.

For four markers, the average operation time was 36.42±7.77 s that was faster than the former result (with three markers). Average RMSEs at the testing part were 0.037±0.025, 0.032±0.011 and 0.045±0.032 Nm/Kg for normal, fast and slow speed respectively.

The average NRMSEs were 6.61±2.86%, 5.67±1.09% and 8.63±5.11%, the average SMAPEs were 26.21±7.22, 22.69±4.05 and 30.29±5.28 for normal, fast and slow speed. The correlation coefficient showed a better relationship between the target and the predicted output than the prediction with three markers, with average values of 0.94 (0.45-0.99), 0.97 (0.92-0.99) and 0.94 (0.69-0.99) for normal, fast and slow walk respectively (figure 7.5).

The FFANN architecture of the three marker coordinate data was identical to the FFANN architecture used in chapter 6 where the randomised joint angles were used as input: nine input variables, one hidden layer with eighteen hidden neurons and one output variable. Using the marker coordinates showed slightly better results according to the smaller average NRMSE values and higher correlation coefficients. However, training the FFANN using the three marker coordinates required longer operation time than the randomised joint angle.

Discussion

The results showed that gait data directly obtained from the marker coordinates can be used efficiently as input variables in order to train the FFANN to predict the KAM. The results corresponded to the preliminary finding from pilot study1 in chapter 3 where the y marker coordinate data were used to train the FFANN and showed the ability to predict ipsilateral and contralateral hip, knee and ankle joint moments of the overground gait. Superior KAM prediction was observed in the study compared to the KAM predicted by the FFANN trained by randomised joint angles shown in chapter 6. Comparing with an identical FFANN architecture (nine input variables, one hidden layer, and 18 hidden neurons), FFANNs trained by data of three markers performed better than the FFANNs trained by joint angles obtained by the Xsens system (when the FFANN was trained by the randomised joint angles) regarding to the NRMSEs by 16%, 19% and 18% for normal, fast and slow speed respectively.

It has been shown that, when marker coordinates were used to train the FFANN, more inputs provided better KAM prediction and better generalisation to the unseen set of data compared to the FFANN trained by joint angles in chapter 5 and 6. In the same direction, a higher number of hidden neurons in the FFANN model provides a better prediction and generalisation ability. However, using a large amount of inputs or hidden neurons created a complex model, as a result, it came with higher computational cost resulting in a considerably higher operational time to finish the prediction task when the FFANNs were trained by inputs from seven marker coordinates. Another potentially negative effect of using more input data is the risk of overfitting (Mundt et al., 2019), but the results from training the FFANN by data of

seven marker coordinates were not significantly different from the results obtained from the model trained by data of the three or four marker coordinates.

Nevertheless, it required more than 60 minutes to complete the prediction task when data from seven marker coordinates were used, compared to 29.34 seconds for the four marker coordinates (training with the same hidden neuron ratio). For this reason, using a large number of inputs to train the network is less likely to benefit gait analysis in practice. It has been debated how many hidden neurons would be appropriate for training an ANN model and still inconclusive as this depends on what is the desired output (Hirose, Yamashita and Hijiya, 1991; Pasini, 2015): trial and error is one of the techniques. From the findings in this study, double of the input variables was the most effective ratio according to operation time required and the capability of generalisation.

Using an appropriate number of input variables is one of the important factors for effective prediction. Either superfluous or too few inputs can affect the prediction ability of the FFANN (Ardestani et al., 2014). In previous studies, techniques were chosen to help reducing the number of the input variables in order to enhance neural network performance (Hahn and O'Keefe, 2008; Aljaaf et al., 2016; Mundt et al., 2019). It has been confirmed that smaller number of input variables worked more effectively to predict the KAM. The best performances in terms of providing a comparable generalisation to unseen data and reasonable prediction time were found when the FFANN was trained by data from the coordinates of four markers. The prediction ability of the FFANN was slightly lower when the network was trained by input extracted from three markers than four markers when the same ratio of hidden neurons was applied to the model. This finding provides a great potential to

adopt this technique and conduct a gait analysis outside the gait laboratory hence it is less complex, does not require long prediction time and most importantly the network can generalise well to new data.

There were similar studies that successfully predicted knee frontal plane moment in gait by an ANN using different types of input data. Overall, the results from this study are 50% superior when compared to those reports. Average NRMSEs between the target and the predicted outputs obtained from the FFANN trained by data of four marker coordinates were 6.61%, 5.67% and 8.63% at normal, fast and slow speed respectively while the correlation coefficients showed strong relationships between the two at 0.94, 0.97 and 0.94 for normal, fast and slow walk. These are better results than those by Mundt et al. (2018) who reported NRMSE of 14.96% and correlation coefficient at 0.95 for the unseen data of a healthy volunteer when the frontal knee joint moment was predicted by a long short-term memory (LSTM)-based recurrent ANN. Smaller NRMSE was also shown when the KAM was predicted by an FFANN model using simulated inertial measurement unit gait data as input at 10.58% and correlation coefficient at 0.98 (Mundt et al., 2019). Similarly, Favre et al. (2012) reported strong relationships between the measured KAM and the predicted KAM of asymptomatic osteoarthritis volunteers when it was predicted by the same FFANN algorithm as in this study, using 11 inputs, although deriving from the ground reaction force and the mechanical axis alignment at 0.97, 0.97 and 0.96 for normal, fast and slow speed walk and also showed a slightly better prediction of the fast speed over the other two walking speeds.

Chapter 8: Enhancing the FFANN performance for knee abduction moment estimation using simulated two-dimensional gait kinematics as inputs to train the FFANN for knee abduction moment prediction

Background

Three-dimensional gait analysis (3DGA) has been accepted as a standard method to quantify gait in order to identify the underlying gait pathology. However, to be practically used, 3DGA is limited to be conducted in a well-established motion laboratory under complex instruments, at high cost and requires a specialist to operate (Simon, 2004; Ugbolue et al., 2013; Zult et al., 2019). Two-dimensional gait analysis (2DGA), in contrast, requires less resources and can be carried out more comfortably with uncomplicated setting. For a century, the 2DGA has been used to measure joint angles in gait, the joint angles obtained from the 2DGA were highly correlated with the 3DGA, particularly knee and ankle joint angles (Michelini, Eshraghi and Andrysek, 2020). In areas where the 3DGA is not applicable, the low cost 2DGA can be the investigation of choice to help researchers to identify the abnormality of gait (Zult et al., 2019). It has been shown in previous studies that the reliability and validity of using 2DGA are varied when compared to the standard. Joint angles quantification of knees and ankles were reliably measured by the 2DGA while the hip and pelvic joint were less reliable (Michelini, Eshraghi and Andrysek, 2020).

It has been shown in the former chapters that reducing the input quantity can improve the FFANN performance. Processing 2D gait analysis requires fewer markers compared to 3DGA, therefore the gait data could be suitable to use for the FFANN training. Moreover, in a situation where 3DGA is not readily available, 2DGA can play a major role for gait analysis. Therefore, it is worth examining if the data obtained from 2DGA can be used to train the FFANN in order to predict the knee abduction moment and its performance compared with using the 3DGA data as inputs. This study aimed to investigate the possibility of using two-dimensional gait data derived

from marker coordinates of the virtual markers produced by the Xsens sensors to train the FFANN and then predict knee KAM during gait.

Data processing and analysis

Data extraction from the original data files

Data used in this study were originally from the data set in chapter 7. The gait data obtained from the Xsens sensors were exported as a .c3d file, containing virtual markers generated by the Xsens software. The data of four virtual marker coordinates in the laboratory's global coordinate system in the X, Y and Z directions were selected from the virtual markers of the right leg, according to the previous finding that data of four marker coordinates was the most efficient to predict the KAM, including ASIS, greater trochanter, lateral knee epicondyle and toe using Visual3D V.6 software (C-Motion, MD, USA). The extracted data were then up sampled from 100 Hz to 120 Hz using the spline function. The KAM of the right leg was obtained from GOAT.

Data from the four markers were then arranged to create a new set of data that is equivalent to the 2DGA data in the sagittal, frontal or transverse planes. From the basic movement analysis theory, a movement of gait is recorded and referenced by a spatial reference system depending on the processing software. Generally, X axis represents the direction of progression (anterior-posterior), Y axis represents the vertical direction and Z is represents the sideways direction (medial-lateral). Each plane of movement is described by two axes as followed, movements between the X and Y axes towards X direction depicts the sagittal plane motion, movements between Y and Z axes towards Y direction depicts the frontal plane motion and lastly, movements created between X and Z towards Z direction depicts the transverse

plane motion (Winter, 2009). However, in this chapter, the data were extracted from the Xsens system, therefore the 2D planes were adapted corresponding to the Xsens axes as the movement about the Z axis is the sagittal plane, the movement about the X axis is the frontal plane and the movement about Y axis is the transverse plane (Schepers, Giuberti and Bellusci, 2018). The 2D gait data of the four markers were created in the frontal plane, sagittal plane and transverse plane, thus creating a two variable data set for each plane of motion.

Similar to chapter 7, data of all walking speeds were separately used to train the FFANN using the same Matlab script after data preparation to create 19 left out folds. Fifty-seven Matlab scripts were written for KAM prediction using inputs from three set of data (19 left out folds of frontal, sagittal and transverse planes (19 x 3)) as input and data of the three walking speeds were trained in the same Matlab script). The data from the two axes of four markers were used, the FFANNs were then trained with one hidden layer and 16 hidden neurons. The data were also systematically randomised as described in chapter 6 and the randomised KAM series were reversed back to the normal order at the end of the process.

Results

The FFANNs performed well compared to the former part of the study, average operation time was 40.55 ± 8.31 seconds. They could generalise well to an unseen set of gait data shown by small NRMSEs, especially when the sagittal plane data were used as inputs at 9.59%, 9.28% and 11.60 % for normal, fast and slow speed respectively. In accordance with the NRMSEs, a strong relationship was found between the predicted KAM and the measured KAM with correlation coefficients at 0.88, 0.91 and 0.90 for normal, fast and slow speed. The FFANNs trained by frontal

plane data could also generalise to unseen gait data with average NRMSEs at 11.36%, 10.03% and 13.21% for normal, fast and slow speed, strong relationships were observed with correlation coefficients at 0.85, 0.91 and 0.88 for normal, fast and slow speed respectively. Similarly, the FFANNs trained by transverse plane data could generalise to an unseen set of gait data with average NRMSEs at 11.56%, 10.65% and 12.94% (figure 8.1). Strong relationships between the predicted KAMs and the measured KAMs were also shown by average correlation coefficients at 0.84, 0.88 and 0.88 for normal, fast and slow speed respectively (figure 8.2).

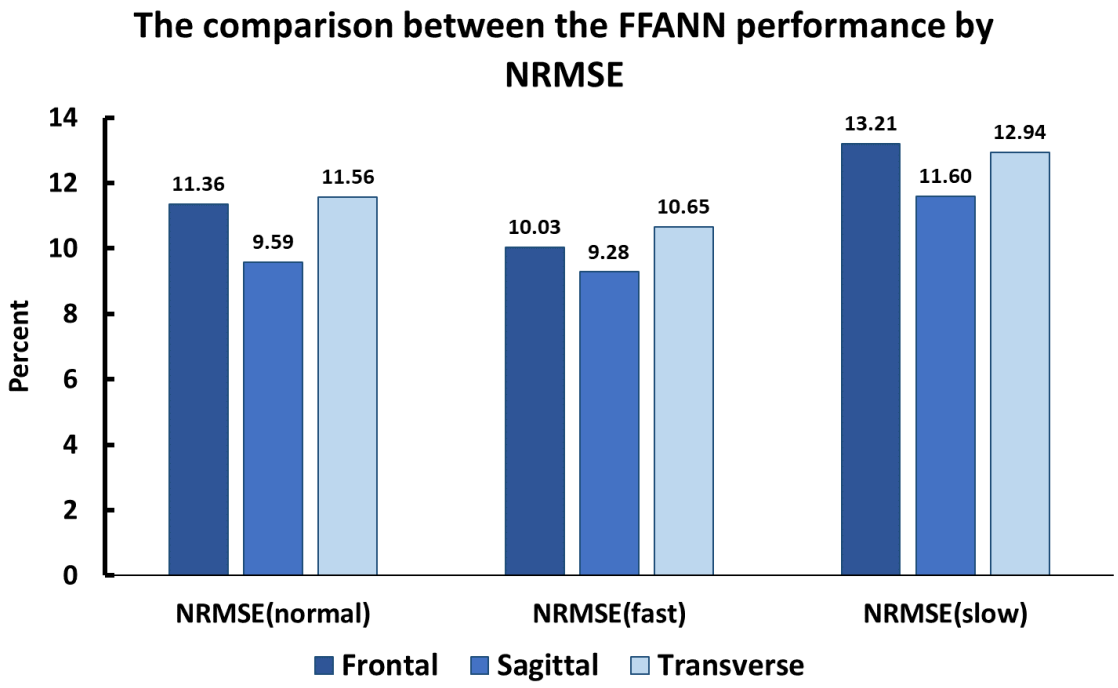

Figure 8.1 The bar charts show the generalisation ability of the FFANN trained by simulated 2D data in the frontal, sagittal and transverse planes. The average NRMSEs of the testing part of the network at all walking speeds range from 9.28% to 13.21% with the best performance obtained from the prediction using simulated sagittal plane data at fast speed.

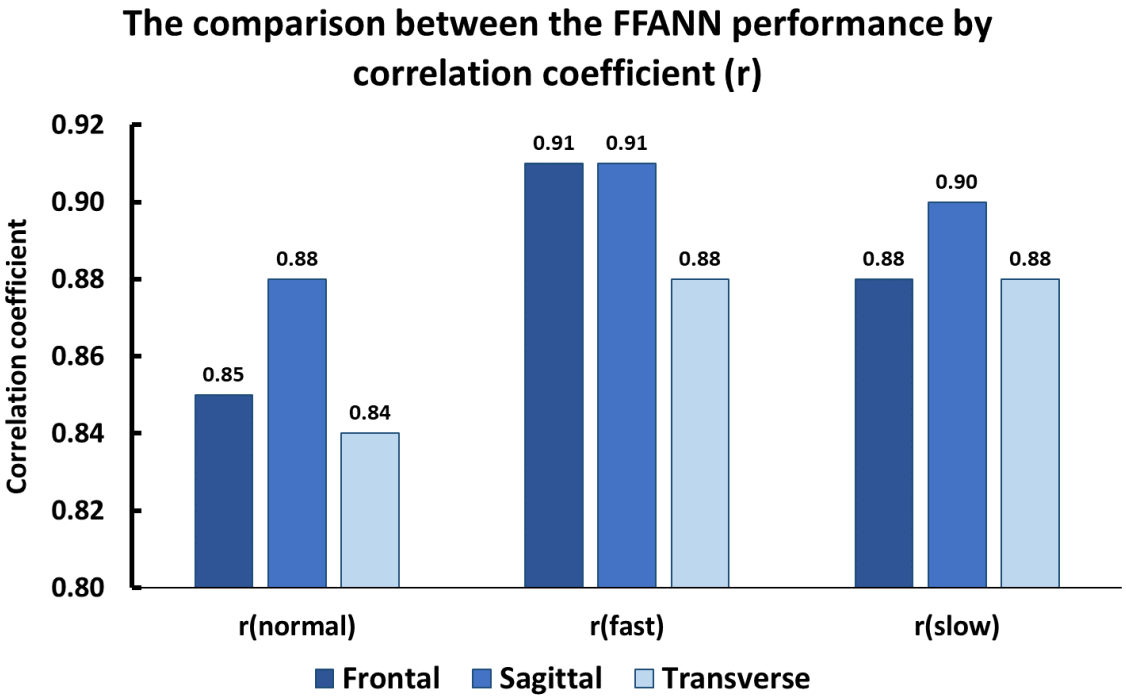

Figure 8.2 Strong relationships between the predicted KAM and the measured KAM are depicted corresponding with the FFANN performance (NRMSE) in figure 8.1. The FFANN trained by data from the 2D sagittal plane generally performed better than the other two motion planes.

Discussion

The results from this study have shown the feasibility of using 2D marker coordinates as inputs to train the FFANN. The results also suggested that using smaller size of input variables could improve the FFANN performance. Prediction of the KAM by an FFANN trained by 2D data as input required slightly longer operation time when compared with using randomised 3D marker coordinate data. Average NRMSEs between the measured and the predicted KAM obtained from the 2D data at all walking speeds were approximately 5% larger than the NRMSEs of the 3D data thus indicating that the FFANN performed better when the 3D data were used as inputs. Amongst the three planes of movement, using data of the frontal plane was expected to show the best prediction. In general, the KAM is calculated from inverse dynamic method using joint reaction force by the distance between the force and the joint centre (Winter, 2009). Therefore, mathematically, there should be a relationship between the frontal plane motion and the KAM that the FFANN can recognise by arriving at a suitable set of weights during training, in order to predict the output (Hodas and Stinis, 2018). However, from this study, using 2D sagittal plane motion provided a slightly better prediction than the frontal and the transverse plane. This could be due to the pattern of motion in sagittal plane which has a large range of motion and can be captured more accurately and reliably than in the other planes. As a result, the FFANN could perform better in this sagittal plane. Moreover, randomisation of the input data before training the FFANN could potentially help eliminating biases and variances of the input data for the FFANN to perform better (Peralta, Gutierrez and Sanchis de Miguel, 2009) as the results shown in chapter 6.

Based on the results from the study, it can be stated that the 2DGA marker coordinates in sagittal plane can be used as inputs to train the FFANN to predict the KAM. This can benefit reserchers where the 3DGA is not readily available as well as in clinical practice where gait analysis is required but has a limited laboratory space for all 3DGA instruments or without sufficient training

Chapter 9: The effectiveness of reducing the input dimensions using Principal Component Analysis to train the FFANN for knee abduction moment prediction

Background

The performance of the FFANN could be improved when a suitable number of inputs were used to train the network. The results from the previous chapters showed a more effective prediction when data from fewer markers (four markers compared with five and seven markers) were used as inputs for the FFANN training in terms of the accuracy and computational cost (time to operate the task). Besides reduction of input number whilst a particular set of inputs is used to train the FFANN, for instance the joint angles and marker coordinates as inputs, there is another method that could be used to remove the redundant information in the data (Daffertshofer et al., 2004). Principal component analysis (PCA) is an established mathematical algorithm that has been used to perform orthogonal linear transformation of multidimensional data to new dimensions based on the covariance matrix computed from the original data (Hui et al., 2005; Bisele et al., 2017). Generally, the essential features of the data are transformed in order, from the most relevant to the least relevant data variables to the original data. In other words, the first principal component has the largest associated variance while the last principal component has the smallest variance (Jones, Holt and Beynon, 2008). However, depending on the research question, each orthogonal dimensionality component of the PCA space can be a representative of important characters of the data (Bisele et al., 2017).

The PCA algorithm has been utilised to reduce the number of input features and optimise the artificial neural network performance to predict kinematics and kinetics of gait. There are several studies using a PCA to classify gait features and help with developing a machine learning algorithm in gait analysis (Eskofier et al., 2013; Federolf, Boyer and Andriacchi, 2013; Bisele et al., 2017). Mundt et al. (2020) applied

a PCA to extract input data obtained from simulated IMU sensors and hypothesised that reduction of the input data by feature extraction computing by a PCA would provide a better performance of the long short-term memory neural network. The conclusion could not be drawn since such result was not observed, in addition, an increased variance in the accuracy of the prediction over all joint motion planes was found (Mundt et al., 2020). However, the PCA has not been used to optimise the FFANN in order to predict joint moment of gait before. This study focused at the possibility of improving the FFANN performance by using the input data feature extraction with PCA and compare with the results from chapter 6 where the FFANN was trained by the randomised joint angles.

Data processing and analysis

Data extraction from the original data files and data preparation

The same set of data from chapter 6 were used in this study. The input variables comprised of consecutive 2,000 data points of each of the nine joint angles and rotation: hip, knee and ankle in flexion/extension, abduction/adduction, and internal/external rotation obtained from Xsens sensors. The data were up sampled using the spline function in a custom Matlab script to match with the target output's sampling rate. The target output variable was the KAM obtained from GOAT. All input and output variables were from the right leg. Input and target output data of each participant were then concatenated from normal, fast and slow speed respectively. Therefore, a data matrix of 10 (9 inputs, 1 target) x 38,000 was created for each walking speed. The data were prepared separately for the leave-one-out cross validation as stated in chapter 6.

The data pre-processing by a PCA was conducted using the PCA function in Matlab (appendix 11.). The scores were named as score 1 for input data of normal speed, score 2 for fast speed and score 3 for slow speed. Each score expressed the nine new data dimensions (PC1, PC2, PC3, …, PC9) of the input variables of each walking speed. The first PC represented the majority of variance (the most relevant according to the covariance matrix) of the input data and the last PC represented the least relevant data. The transformed data were consequently selected to be inputs to train the FFANN.

Principal component analysis algorithm can be further explained in the following paragraph (modified and adapted to this task from (Eskofier et al., 2013)). The principle of using a PCA to pre-process a set of data aims to extract features and retain the essential characteristics of the data and change it into a new set of variables of principal components (PCs). It can be carried out in steps beginning with a data matrix M $\in \Re^{38000\times9}$ where 2,000 data points of joint angles obtained by Xsens sensors at all planes of motion of hip, knee and ankle of the 19 participants were concatenated. Eigenvalues were decomposed from the original matrix and a new correlation matrix was created as M^t M ($\in \Re^{9\times9}$), followed by creating eigenvectors e_k ($\in \Re^{9\times1}$ k = 1, …, 9) of this matrix. The eigenvectors expressed characteristic vectors of a linear transformation of the original input data. The matrix M was then multiplied by the eigenvectors thus creating the principal component vector p_k($\in \Re^{38000\times1}$) which would later be ordered corresponding to the magnitudes of the eigenvalues formerly calculated. As a result, the first few principal components describe the major variations of the input variables.

Thereafter, the input data were systematically randomised in the same method described in chapter 6, then an FFANN architecture was created using the Levenberg-Marquard algorithm with one hidden layer and the hidden neurons ratio (double of input variables) was used following the result from chapter 7. The FFANNS were trained using three different sizes of input data from 1. PC1 only, 2. PC1 and PC2, 3. PC1-PC3 in separate architectures (two, four and six hidden neurons respectively). Therefore, 19 FFANNs were created to predict the KAM for each set of the input thus making overall 57 FFANNs in this study (19 folds times three FFANN architectures).

Results

Amongst the nine principal components, the first three PCs, altogether, represented on average 84.57% of the variance (85.97%, 83.05%, 84.67% for normal, fast and slow speed respectively) and there was approximately 15.44% for the combination of the remaining PCs (figure 9.1). The first PC (PC1) depicted on average 56.88% of the input data.

Using inputs of the PCs to train the FFANN to predict KAM of all three walking speeds required a short operative time of 2.64 ± 0.75, 3.51 ± 0.48 and 6.48 ± 1.49 seconds from PC1, PC1 and PC2, and PC1-PC3 respectively (ASUS laptop X455L series, Intel Core i3 4030U, 1.9 GHz, 4 installed RAM with Windows 10 Pro 20H2 version). Good performances of the FFANN in order to predict unseen KAM of the left-out participant were shown at average NRMSEs 18.60%, 16.48% and 18.13% for normal, fast and slow speed for the FFANN trained by PC1-PC3 (84.57%) (figure 9.2). Additionally, the correlation coefficients were 0.67, 0.77 and 0.73 for normal, fast and slow speed respectively (figure 9.3).

Similar results were shown in the prediction of the unseen KAM from FFANNs trained by PC1 and PC2 (77.35%) with the average NMRSEs 19.19%, 17.17% and 18.78% and correction coefficients 0.66, 0.75 and 0.73 for normal, fast and slow speed respectively. The FFANN trained by PC1 alone showed less accurate predictions with average NRMSEs at 22.20%, 20.25% and 23.67% and correlation coefficients at 0.56, 0.63 and 0.59 for normal, fast and slow speed respectively. The predicted KAM of normal and slow speed by the FFANN trained using PCA showed significant differences to the result in chapter 6 where the joint angles (randomised) were used as inputs. However, there was no significant difference ($p > 0.05$) of the predicted KAM at fast speed.

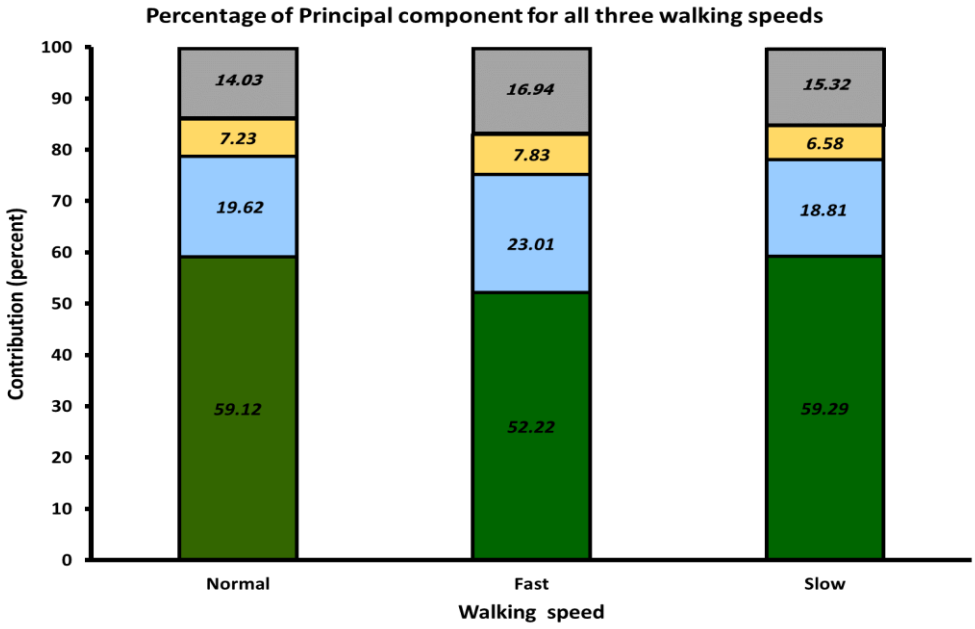

Figure 9.1 The first three principal components represented over 80% variance of the data, the PC1 (green) describes the largest variability between 52-59%, the PC2 (blue) describes between 18-23% and PC3 (yellow) describes approximately 7% of the data. The rest of PCs (grey), altogether, illustrate approximately 15% variance of the data. Similar distribution is seen in the input data of all walking speeds.

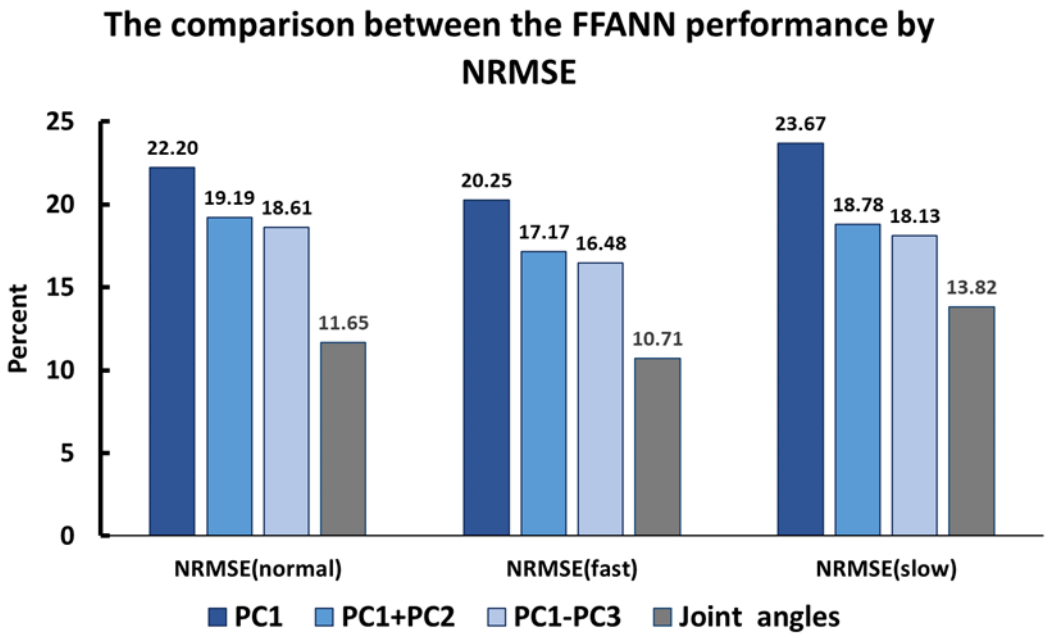

Figure 9.2 The better performances of the FFANNs trained by randomised joint angles are demonstrated by the lower NRMSEs.

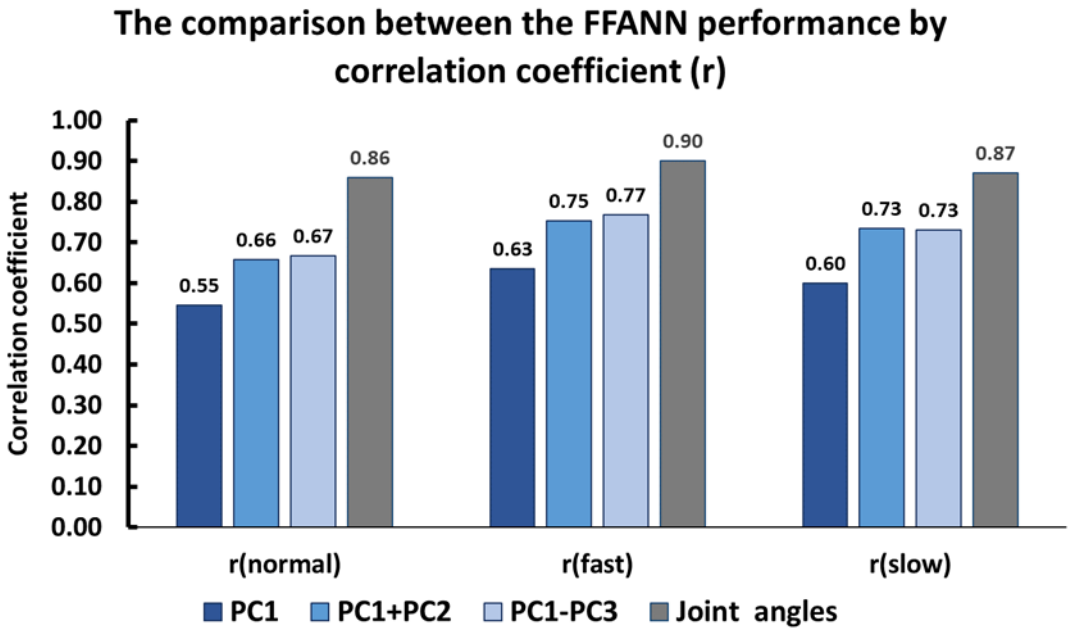

Figure 9.3 Moderate correlations were found between the predicted KAM and the measured KAM at all walking speeds. The FFANNs trained by randomised joint angles provide a better KAM prediction than the FFANNs trained by PCA data.

Discussion

The gait data obtained from the data dimensionality reduction or data feature extraction by a PCA can be used to train the FFANNs to predict the KAM. This training procedure required less operation time than prior methods conducted in chapter 6. This is the straightforward advantage of using PCA in terms of reducing computational cost considering that the original data were transformed in new dimensions computed from the variance of the data. However, to achieve the comparable prediction with using randomised joint angles to predict the KAM (chapter 6), more than 80% of input data representation was required (PC1-PC3). Using only PC1 to train the FFANN was not sufficient to introduce gait patterns to the network to predict KAM accurately compared to previous technique used in this study. Compared with the results from chapter 6, the FFANN trained by the randomised joint angles performed better than the FFANN trained by the PCA data. The NRMSE values were 10% -15% different between the two methods. The average operation time was six times faster in the PCA method than the randomised joint angle which was most likely due to the smaller number of input variables and the uncorrelated nature of the PCA data. Also shown by NRMSEs, the best prediction of this part was from the FFANNs trained with PC1-PC3. The NRMSEs were larger than the results from the randomisation technique at all walking speeds. However, there was no significant difference between the two methods at the KAM prediction of fast walk. The results also showed the moderate correlation between the predicted KAM and the measured KAM at all walking speeds; the best relationships were shown in the results obtained from the FFANN trained by PC1-PC3 especially at fast speed. This suggested that more data were required to predict the KAM from joint angles

obtained from the Xsens sensors. In contrast, inputs obtained by data dimensionality reduction were not suitable for this task. One could try to add more PCs to gain more gait characteristics to train the FFANN and receive better KAM prediction, however, it would simply indicate, by doing so, that more data were needed and feature extraction is no longer necessary. A previous study reported the use of 95% data representation to predict gait kinematics (hip, knee and ankle joint angle) using simulated IMU data to train a long short-term memory neural network. They showed the strong relationships between the predicted and the actual joint angles with NRMSE ranged between 2-5% (Mundt et al., 2020). It is worth noting that in PC1 and PC2, the major contribution of the coefficient were derived from hip and knee sagittal plane respectively. This could be due to the joints with larger range of motion will have a better signal-to-noise ratio than the ones with a small range of motion once they are normalised.

Chapter 10: Enhancing the FFANN performance Discussion and Conclusion

The results obtained from chapter 5 have exhibited the ability of the FFANN model to predict the KAM of gait in healthy individual. However, due to a wide range of generalisation capacity when the FFANN was trained by joint angles derived from Xsens system with the NRMSE of 13%-92% for normal speed, 10%-79% for fast speed and 2%-62% for slow speed. This indicates the necessity of improving the FFANN performance to achieve the aim of this project. The FFANN performance could be improved by means of data manipulations in chapter6-9 : randomised joint angles, marker coordinates,2D marker coordinates and PCA data (when the input were from PCs represented more than 75% of the joint angles) compared with the results of chapter 5 that the FFANN was trained by the joint angles obtained directly from the Xsens system. Moreover, the results from this study suggested that appropiate number of hidden neurons that provided the efficient KAM prediction was double number of the input variables. The operation time for the FFANN to perform the prediction varied from 2.65 seconds when only the PC1 was used as input to 52.58 seconds when data of three marker coordinates were used as input (figure 10.1). Number of the input variables and hidden neurons were the main factors determining the operation time. Comparing between the inputs that used to train the FFANN, using marker coordinates showed the best KAM prediction whereas using joint angles obtained directly from the Xsens (chapter 5) showed the poorest performance (figure 10.2).

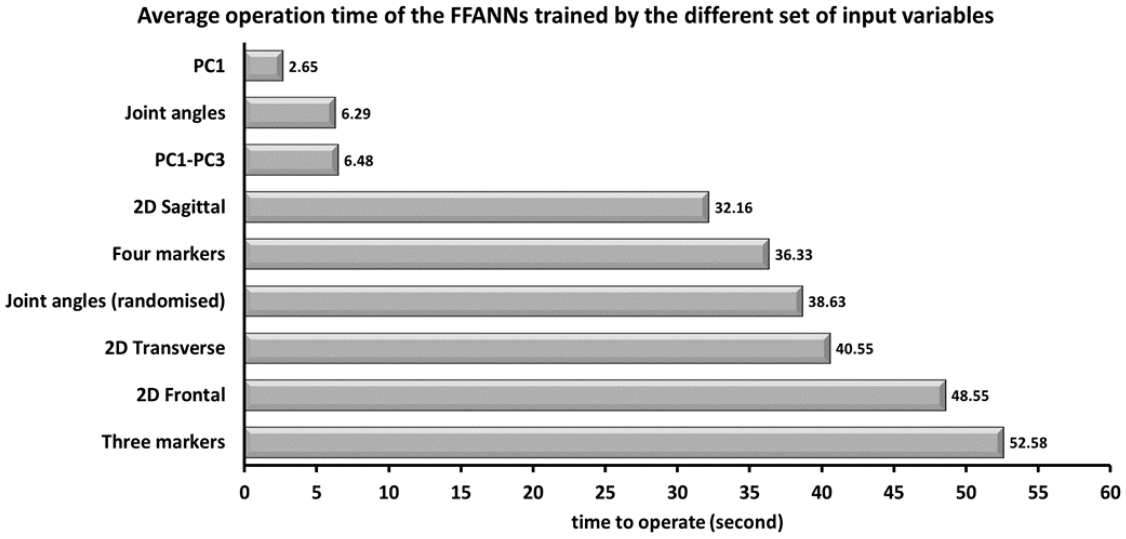

Figure 10.1 The time required to finish the KAM prediction task when different sets of input variables were used to train the FFANN ranged from slow to fast operation time. Using PCA data required remarkably less time than the other inputs.

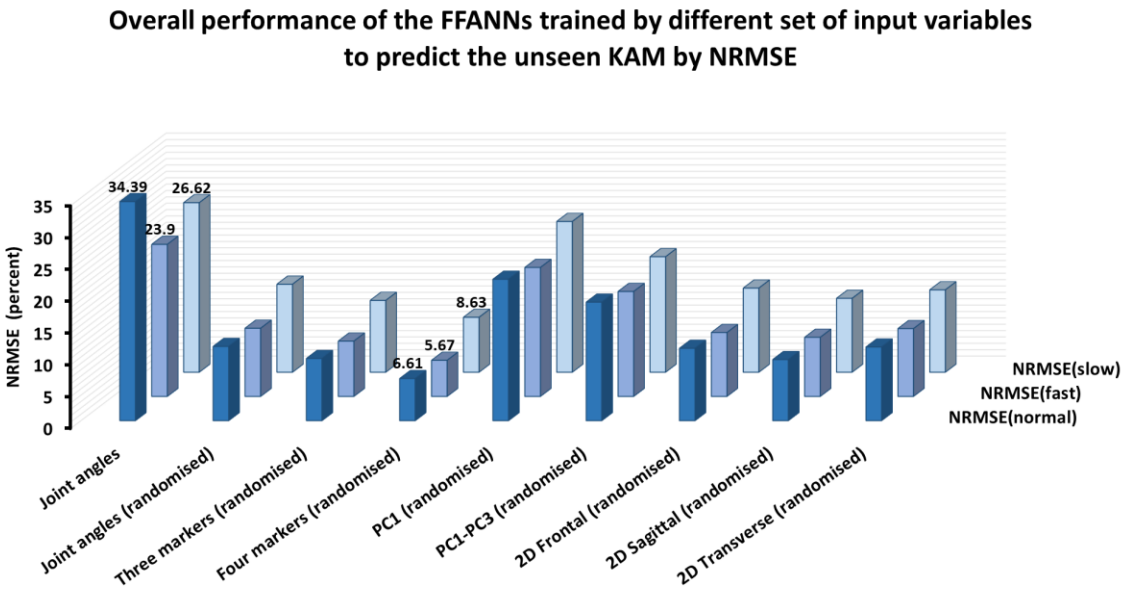

Figure 10.2 The generalisation ability of the FFANN to predict an unseen set of KAM shown by NRMSEs, comparing between different inputs. At all walking speeds using data of four marker trajectories to train the FFANN provides the best prediction amongst the others while using joint angle obtained from Xsens sensors without randomisation provided the highest NRMSEs.

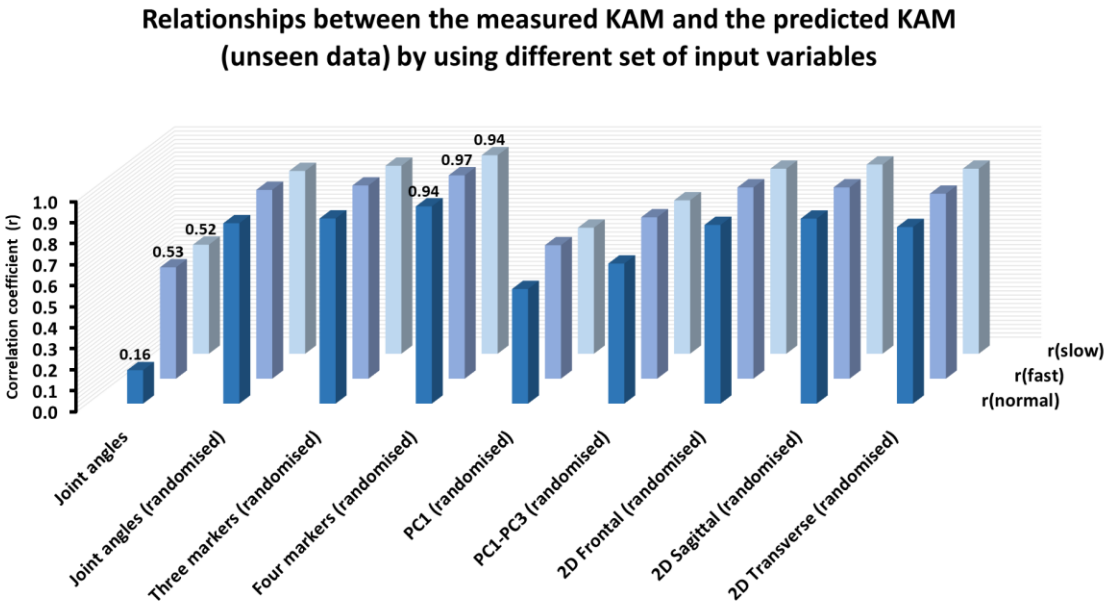

Figure 10.3 Strong relationships between the predicted KAM and the measured KAM are shown when the FFANNs were trained using randomised joint angles, marker coordinates and simulated 2D data of all planes of movement.

Data from four marker coordinates were the best input variable to train the FFANN to predict KAM of all three walking speeds that could possibly be due to the marker coordinate data presented a pattern of gait that the FFANN could recognise with an appropiate amount of noise which will help the FFANN in order to learn the pattern of data and predict the KAM, especially, at the testing part where the FFANN predicted the KAM from the unseen data. Moreover, the marker position data could contain more information than any 3D angles as the data carry full information that were recorded from a movement before being selectively processed to as joint angles. Although the prediction using data from more marker coordinates (five and seven) provided lower NRMSE than the four markers, it was not practical to apply in clinical gait analysis since the considerably long operation time was required and also possibly required a high performance computer to conduct the KAM prediction task

(this study was conducted using an ASUS laptop X455L series, Intel Core i3 4030U, 1.9 GHz, 4 installed RAM with Windows 10 Pro 20H2 version).

Similarly, the correlation coefficients showed the same trend as NRMSEs. The predicted KAM was less correlated to the measured KAM when it was trained using the joint angles as inputs without randomisation. Good to excellent correlations were exhibited when the KAM was predicted using randomised joint angles, marker trajectories, 2D marker coordinates and PCA data (figure 10.3).

Training the FFANNs by using the 2D gait data required reasonable operational time when compared with the prediction using joint angles or marker coordinates as input. The strong relationship between the predicted and the measured KAM in this study were shown to be comparable with the results when the FFANNs were trained using randomised joint angle and marker coordinates as inputs. The comparison of the FFANN performance using the different sets of input data showed the capability of the FFANNs trained by any plane of 2D data in order to predict KAM of the unseen person when compared to the FFANNs trained by joint angle alone or PCA data. The FFANN performed better at fast speed gait data compared to normal and slow speed and the KAM prediction results were comparable amongst the majority of inputs (table 10.1). No significant differences were shown between the KAM predicted by the FFANN trained by randomised joint angles and the majority of input variables including three and four marker coordinates, 2D frontal and sagittal plane marker coordinates.

Table 10.1 The illustration of the differences amongst predicted KAMs from the different input data used to train the FFANNs. Asterisks indicate the significant differences between the KAM predicted by the FFANN trained by randomised joint angles and the particular set of inputs ($p<0.05$).

	Joint angles	3 markers	4 markers	PC1-PC3	2D frontal	2D sagittal	2D Transverse
Normal	0.000*	0.000*	0.053	0.000*	0.009*	0.003*	0.872
Fast	0.000*	0.848	0.634	0.583	0.912	0.872	0.000*
Slow	0.000*	0.581	0.000*	0.000*	0.000*	0.126	0.000*

From the findings in these chapters, marker coordinates appear to be the best prediction input variables to train the FFANN to predict the KAM in gait. Some limitations in this study are, firstly, the marker coordinates were extracted from the Xsens virtual makers, therefore it might be difficult to imply that the similar outcome would be received if the data from real reflective markers were used to train the network. In addition, there should be some differences between skin movement artefact when the Xsens sensors were applied on a body segment and the virtual marker coordinates which could affect the input data pattern when they were presented to the FFANN. However, the purpose of the study was to extend the use of an IMU system for out-of-the gait laboratory, therefore, this virtual marker

coordinates can still be an input variable of choice to train the FFANN to predict joint moment of gait. Secondly, the number of the markers to be used as input needs further study as there could be a better combination of the markers that could provide more accurate prediction. Also, the studies were conducted using data of unimpaired individuals without the report of gait problems, the FFANN, therefore, should be futher trained using the data from people with gait problems to examine the FFANN performance (generalisation ability).

In conclusion, there were strategies to improve the performance of the FFANN used in this research project in order to predict the knee abduction moment. Using data from marker coordinates would be recommended to be an effective input for the FFANN. The 2DGA sets of input which were derived from the marker coordinates also illustrated prediction and generalisation ability of the FFANN. Randomisation of the input was proved to help more efficient KAM prediction in the study with small number of the inputs to reduce the bias and variance between data of individuals. In addition, using appropriate size of input variables, in other words the number of the inputs, was proved to enhance the FFANN performance, however this was not applicable when the PCA data were used to train the FFANN. One hidden layer was sufficient for the FFANN to predict the KAM, however, the recommended number of the hidden neurons that provided an accurate prediction and computational cost-efficient would be double the number of input variables.

Chapter 11: General discussion

11.1 Overview

The inertial motion capture system has been developed and used for several kind of motion analyses (Iosa et al., 2016; van der Kruk and Reijne, 2018). The system offers a freedom in recording a movement and produces joint angles (kinematics) while offering independence from a camera set up which allows the motion capture to be conducted outside the restrictions of the typical optoelectronics method used in a traditional motion laboratory. The main purpose of this study was to further extend the practicality of the IMU system, especially in gait analysis. To complete a gait analysis by using an IMU system, several methods have been introduced to quantify the forces that cause the movement in combination with the kinematics obtained from the IMU system. The frontal plane knee joint moment was focused due to this biomechanical parameter's central role as it is an important risk factor for the common degenerative disease of the knee (knee osteoarthritis or OA knee) (Teng et al., 2015). Despite good outcomes were reported when machine learning such as an artificial neural network is used to predict joint moments in gait (Hahn and O'Keefe, 2008; Favre et al., 2012; Ardestani et al., 2014), there was a gap in knowledge on using joint angles directly obtained from an IMU system as inputs to train an ANN for predicting a joint moment during gait.

11.2 Summary of the experimental findings

The results found in pilot studies shaded light on the feasibility of using the FFANN to predict joint moments of gait. Either marker coordinates or joint angles could be selected as input to train the network shown by pilot study 1 and 2 respectively. The

FFANN could learn an individual gait pattern and predict the unseen KAM of the same participant, thus reflecting the generalisation ability of the FFANN used in this study. However, due to too few number of participants in the pilot studies, the FFANN was not able to generalise to estimate the KAM of a different individual shown in pilot study 2. The FFANN needed to be trained by part of a new set of data to be able to predict KAM of the rest of the data, reported in pilot study 3. This finding was important to indicate the necessity of presenting as many input-output patterns as possible to FFANN, for it to be able to recognise and predict the most accurate output. Therefore, a larger number of the participants was required to train the FFANN.

The results from chapter 5 where the FFANN was trained using gait data of 13 participants to predict the KAM of the unseen participant, at the testing part of the data, were very encouraging confirmed by a low NRMSE and high correlation coefficient. However, a wide range of the two values was found which highlighted a lack of uniform gait pattern between the gait patterns of those 13 individuals that were used in training. As a result, it was difficult for the FFANN to recognise and provide the accurate prediction to the unseen data of some individuals. From the findings, several strategies were experimented to improve the FFANN performance in chapter 6-9. The bias and variance in gait data obtained form different individuals were reduced using systematic randomisation which was applied to indiscriminate the input-output pairs to create a more uniform set of data (Reitermanová, 2010). The remarkable improvement of the FFANN performances was shown by halved NRMSEs in the randomised method compared to using joint angle as input to train the network. Moreover, the marker coordinate data were also used as input to train

the FFANN further unwrapping the promising initial results of pilot study 1. The ability to predict the unseen KAM was best when marker coordinates were used to train the network compared to the joint angles. This might be explained by the different characteristics of the marker coordinate data and the joint angles. Generally, data of the marker coordinates are less processed which could represent the primary data for individuals' gait as opposed to using joint angles, which although based on the marker coordinate signals, are complicated by using an Euler rotation sequence which involves compulsory decisions inevitably leading to loss of information content. The better performance of ANN training and testing confirmed that the relationship between the inputs and the target KAM was easier to establish for the FFANN. The better prediction of the KAM were also found when the FFANNs were trained using 2D markers and principal components derived from the original data as input compared with the KAM predicted by the FFANN trained by joint angles. The findings, therefore, supported the adequacy of information content obtained from the marker coordinates since the FFANN did not require input data of too many markers to train yet giving more accurate prediction than using joint angles as input. Data from only two to three markers were sufficient to train the network as shown in chapter 7 and 8. Another possible explanation would be down to the randomisation method (Reitermanová, 2010) that was integrated to the data preparation before training the FFANN from chapter 6 onward.

11.3 Clinical implications of the study

11.3.1 Accuracy and reliabilty of the Xsens system

Inspite of being developed for decades, the IMU system has lagged behind optoelectronic motion capture to use clinically. One of the reasons is the accuracy of the motion capture system compared to the standard that affects the potential use of the IMUs in clinics in order to help making vital decisions to provide appropiate management to patients who suffer from gait abnormality (Poitras et al., 2019). The system has been validated against the standard motion capture system, and the findings showed that the IMU system provides comparable joint angles with the standard motion capture in the sagittal plane which has larger range of motion than in the frontal and transverse planes during gait (Zhang et al., 2013; Al-Amri et al., 2018). This was explained by the effect of differences in the underlying biomechanical model used by the two motion capture systems (Poitras et al., 2019). A similar finding was observed in chapter 4 where the Xsens Awinda was validated against the Vicon system. However, the result from a previous study showed that the between-rater and within-rater reliability when using the Xsens system were better than using the standard motion capture system, even though one of the rater never had an experience of using the Xsens sensors before (Al-Amri et al., 2018). It was suggested that the IMU system could be appropiately used for clinical gait analysis but it cannot be interchangeably used with the standard system. The similar suggestion would be implied from the results in chapter 5-8 where the FFANN could predict the corresponding KAM from the joint angles and marker coordinate data derived from Xsens system. Thus indicating that the Xsens system provided rich information content which could be linked mathematically to a dynamic joint moment by a

complex ANN model, in spite of the suboptimal match between the joint angles derived from the standard motion capture system.

11.3.2 Possibility of the out-of-the lab concept

The results from chapter 5 and 6 have shown that the FFANN was able to predict KAM of gait when it was trained by input variables derived from the IMU system. The marker coordinate data appeared to be the most efficient input to train the FFANN. Therefore, it is a logically possible concept for the combination of IMU system and the machine learning algorithm to be applied to conduct gait analysis outside the typical gait laboratory. The combination could be superior to using another method to record kinetics of gait such as using a wearable plantar pressure device or the instrumented shoes due to their known poor accuracy and cumbersome practical limitation due to the added size and weight of equipment which affect the individual's gait (Shahabpoor and Pavic, 2017). To succesfully use the FFANN to predict the KAM, however, requires acquisition of a lot more gait data to present to the network during training in order to completely cover normal and abnormal gait patterns of human gait. In this study, training the FFANN by the different walking speeds was one way to extend the variability of gait in this small group of participants. More gait patterns are still needed to improve the generalisation ability of the FFANN. Also due to the small number of participants, randomisation of the data were adopted in order to create more uniform and even gait pattern to train the network. Randomisation was probably the most effective data manipulation confirmed by substantially increase generalisation performance. The lower generalisation performance when using non-randomised data could be explained by having only 13 gait patterns for the FFANN to learn while training, so there was a high chance of the

FFANN to be unable to interpolate the knowledge from the training process to predict unseen gait pattern (at the testing part) especially if the 15th gait pattern differed substantially from the others used for training. To be perfectly generalised to predict the joint moment of an unforeseen individual, the FFANN, therefore, has to be trained by more gait patterns. In addition, to be more applicable to clinical practice, healthy and pathologic gait of different age groups will be necessary to train an ANN for a superior performance. As the FFANN used in this study is suitable to train a small group of data, another type of FFANN could be considered to appropiately and efficiently predict joint moment of gait when it was trained by substantial amount of gait data in a clinic.

11.4 Limitations and future directions

The small number of the participants was the first limitation of the study. The FFANN should perform better with more participants and more gait variations. The results could not be implied to abnormal gait because the study was conducted using data from healthy volunteers. Moreover, it might also be difficult to totally imply the results from the study with overgound gait due to the gait data were recorded using the instrumented treadmill that alters the kinetics of gait (van der Krogt et al., 2015). The marker coordinates that were used to train the network throughout the study were not the actual data collected form reflective markers, in fact they were the virtual markers extracted from the Xsens data and this may result in subtle differences due to skin movement artefacts affecting the IMU sensors and markers differently. Thus indicating that the results from chapter 7 should be carefully interpreted under its conditions and might not be comparable with a result in case the actual markers were used.

There might be some effect of the difference in the gait curve from the time delayed for the signals recorded by the Xsens and the Vicon system due to the synchronisation technique in pilot study 2 and 3 (Sadeghi et al., 2000). Therefore the results from those pilot studies might not be perfectly accurate. Only knee frontal plane moment was the main predictive outcome of this study, it was chosen due to the importance in terms of the association with the most common degenerative joint disease of the knee joint. However, it would be important to be able to predict all of the joint moments of gait such as the frontal hip moment and the sagittal ankle moment to broaden the use of the IMU combined with the FFANN prediction to be able to evaluate causes of abnormal gait in different gait deteriorative diseases such as hip diseases and cerebral palsy.

11.5 Conclusion

The practicality of using an IMU system can be broadened when the FFANN is used to estimate the kinetics of gait from the IMU data. The results of this study have shown that marker coordinates could be the input variable of choice to use in the FFANN training process, followed by the randomised joint angle that obtained from the IMU system. The 2D and PCA data can be used to train the artificial neural network with acceptable results when a 3D system or fast processing power are not available. The most efficient ratio of the hidden neurons used would be two times of the input variables.

References

Abdul Razak, A.H., Zayegh, A., Begg, R.K. and Wahab, Y. (2012) Foot Plantar Pressure Measurement System: A Review. *Sensors*, 12 (7), 9884-9912.

Agatonovic-Kustrin, S. and Beresford, R. (2000) Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. *J Pharm Biomed Anal*, 22 (5), 717-727.

Al-Amri, M., Nicholas, K., Button, K., Sparkes, V., Sheeran, L. and Davies, J.L. (2018) Inertial Measurement Units for Clinical Movement Analysis: Reliability and Concurrent Validity. *Sensors (Basel)*, 18 (3).

Aljaaf, A.J., Hussain, A.J., Fergus, P., Przybyla, A. and Barton, G.J. (2016) Evaluation of machine learning methods to predict knee loading from the movement of body segments. *2016 International Joint Conference on Neural Networks (IJCNN)*, 24-29 July 2016 of Conference.

Amin, S., Luepongsak, N., McGibbon, C.A., LaValley, M.P., Krebs, D.E. and Felson, D.T. (2004) Knee adduction moment and development of chronic knee pain in elders. *Arthritis Rheum*, 51 (3), 371-376.

Ancillao, A., Tedesco, S., Barton, J. and O'Flynn, B. (2018) Indirect Measurement of Ground Reaction Forces and Moments by Means of Wearable Inertial Sensors: A Systematic Review. *Sensors (Basel)*, 18 (8).

Ardestani, M., Zhang, X., Wang, L., Lian, Q., Liu, Y., He, J., Li, D. and Jin, Z. (2014) Human lower extremity joint moment prediction: A wavelet neural network approach. *Expert Systems with Applications*, 41, 4422–4433.

Ardestani, M.M. and Hornby, T.G. (2020) Effect of investigator observation on gait parameters in individuals with stroke. *Journal of Biomechanics*, 100, 109602.

Bachmann, E.R., Xiaoping, Y. and Peterson, C.W. (2004) An investigation of the effects of magnetic variations on inertial/magnetic orientation sensors. *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, 26 April-1 May 2004 of Conference.

Baker, R. (2001) Pelvic angles: a mathematically rigorous definition which is consistent with a conventional clinical understanding of the terms. *Gait Posture*, 13 (1), 1-6.

Baker, R. (2006) Gait analysis methods in rehabilitation. *J Neuroeng Rehabil*, 3, 4.

Baker, R. (2007) The history of gait analysis before the advent of modern computers. *Gait Posture*, 26 (3), 331-342.

Barrios, J.A., Crossley, K.M. and Davis, I.S. (2010) Gait retraining to reduce the knee adduction moment through real-time visual feedback of dynamic knee alignment. *Journal of Biomechanics*, 43 (11), 2208-2213.

Barton, G., Hawken, M., King, S., Robinson, M. and Ranganath, L. (2015) Quantification of gait deviations in alkaptonuria: A comparison of joint angles versus marker coordinates. *Gait & Posture*, 42, S24.

Bisele, M., Bencsik, M., Lewis, M.G.C. and Barnett, C.T. (2017) Optimisation of a machine learning algorithm in human locomotion using principal component and discriminant function analyses. *PLoS One*, 12 (9), e0183990.

Cappozzo, A., Della Croce, U., Leardini, A. and Chiari, L. (2005) Human movement analysis using stereophotogrammetry. Part 1: theoretical background. *Gait Posture*, 21 (2), 186-196.

Chen, L., Zheng, J.J.Y., Li, G., Yuan, J., Ebert, J.R., Li, H., Papadimitriou, J., Wang, Q., Wood, D., Jones, C.W. and Zheng, M. (2020) Pathogenesis and clinical management of obesity-related knee osteoarthritis: Impact of mechanical loading. *Journal of orthopaedic translation*, 24, 66-75.

Chiari, L., Della Croce, U., Leardini, A. and Cappozzo, A. (2005) Human movement analysis using stereophotogrammetry. Part 2: instrumental errors. *Gait Posture*, 21 (2), 197-211.

Daffertshofer, A., Lamoth, C.J., Meijer, O.G. and Beek, P.J. (2004) PCA in studying coordination and variability: a tutorial. *Clin Biomech (Bristol, Avon)*, 19 (4), 415-428.

Davids, J.R., Cung, N.Q., Sattler, K., Boakes, J.L. and Bagley, A.M. (2019) Quantitative Assessment of Muscle Strength Following “Slow” Surgical Lengthening of the Medial Hamstring Muscles in Children With Cerebral Palsy. *Journal of Pediatric Orthopaedics*, 39 (5), e373-e379.

Davis, R.B. (1988) Clinical gait analysis. *IEEE Engineering in Medicine and Biology Magazine*, 7 (3), 35-40.

Davis, R.B., Õunpuu, S., Tyburski, D. and Gage, J.R. (1991) A gait analysis data collection and reduction technique. *Human Movement Science*, 10 (5), 575-587.

de Vries, W.H., Veeger, H.E., Baten, C.T. and van der Helm, F.C. (2009) Magnetic distortion in motion labs, implications for validating inertial magnetic sensors. *Gait Posture*, 29 (4), 535-541.

Du, K.L. (2010) Clustering: A neural network approach. *Neural Networks*, 23 (1), 89-107.

Eskofier, B.M., Federolf, P., Kugler, P.F. and Nigg, B.M. (2013) Marker-based classification of young-elderly gait pattern differences via direct PCA feature extraction and SVMs. *Comput Methods Biomech Biomed Engin*, 16 (4), 435-442.

Farizawani, A.G., Puteh, M., Marina, Y. and Rivaie, A. (2020) A review of artificial neural network learning rule based on multiple variant of conjugate gradient approaches. *Journal of Physics: Conference Series*, 1529, 022040.

Favre, J., Hayoz, M., Hledik, J.C.E. and Andriacchi, T.P. (2012) A neural network model to predict knee adduction moment during walking based on ground reaction force and anthropometric measurements. *Journal of Biomechanics*, 45 (4), 692-698.

Favre, J. and Jolles, B.M. (2016) Gait analysis of patients with knee osteoarthritis highlights a pathological mechanical pathway and provides a basis for therapeutic interventions. *EFORT Open Rev*, 1 (10), 368-374.

Federolf, P., Roos, L. and Nigg, B.M. (2013) Analysis of the multi-segmental postural movement strategies utilized in bipedal, tandem and one-leg stance as quantified by a principal component decomposition of marker coordinates. *Journal of Biomechanics*, 46 (15), 2626-2633.

Federolf, P.A., Boyer, K.A. and Andriacchi, T.P. (2013) Application of principal component analysis in clinical gait research: identification of systematic differences between healthy and medial knee-osteoarthritic gait. *J Biomech*, 46 (13), 2173-2178.

Fernando, T.M.K.G., Maier, H.R. and Dandy, G.C. (2009) Selection of input variables for data driven models: An average shifted histogram partial mutual information estimator approach. *Journal of Hydrology*, 367 (3), 165-176.

Ferrari, A., Benedetti, M.G., Pavan, E., Frigo, C., Bettinelli, D., Rabuffetti, M., Crenna, P. and Leardini, A. (2008) Quantitative comparison of five current protocols in gait analysis. *Gait Posture*, 28 (2), 207-216.

Ferrari, A., Cutti, A.G. and Cappello, A. (2010) A new formulation of the coefficient of multiple correlation to assess the similarity of waveforms measured synchronously by different motion analysis protocols. *Gait Posture*, 31 (4), 540-542.

Ferrari, A., Cutti, A.G., Garofalo, P., Raggi, M., Heijboer, M., Cappello, A. and Davalli, A. (2010) First in vivo assessment of "Outwalk": a novel protocol for clinical gait analysis based on inertial and magnetic sensors. *Med Biol Eng Comput*, 48 (1), 1-15.

Foucher, K.C. and Wimmer, M.A. (2012) Contralateral hip and knee gait biomechanics are unchanged by total hip replacement for unilateral hip osteoarthritis. *Gait & Posture*, 35 (1), 61-65.

Foxlin, E. (1996) *Inertial Head-Tracker Sensor Fusion by a Complementary Separate-Bias Kalman Filter*.

Fukuchi, C.A., Fukuchi, R.K. and Duarte, M. (2018) A public dataset of overground and treadmill walking kinematics and kinetics in healthy individuals. *PeerJ*, 6, e4640.

Fushiki, T. (2011) Estimation of prediction error by using K-fold cross-validation. *Statistics and Computing*, 21 (2), 137-146.

Ghojogh, B. and Crowley, M. (2019) The theory behind overfitting, cross validation, regularization, bagging, and boosting: tutorial. *arXiv preprint arXiv:1905.12787*.

Hahn, M. and O'Keefe, K. (2008) A neural network model for estimation of net joint moments during normal gait. *Journal of Musculoskeletal Research*, 11.

Halilaj, E., Rajagopal, A., Fiterau, M., Hicks, J.L., Hastie, T.J. and Delp, S.L. (2018) Machine learning in human movement biomechanics: Best practices, common pitfalls, and new opportunities. *J Biomech*, 81, 1-11.

Hirose, Y., Yamashita, K. and Hijiya, S. (1991) Back-propagation algorithm which varies the number of hidden units. *Neural Networks*, 4 (1), 61-66.

Hodas, N.O. and Stinis, P. (2018) Doing the Impossible: Why Neural Networks Can Be Trained at All. *Frontiers in psychology*, 9, 1185-1185.

Hui, Y., Guang-min, S., Wen-xing, S. and Xiao, L. (2005) Human motion recognition based on neural network. *Proceedings. 2005 International Conference on Communications, Circuits and Systems, 2005.*, 27-30 May 2005 of Conference.

Iosa, M., Cereatti, A. and Cappozzo, A. (2009) A linear method for curve comparison in gait data. *Gait & Posture*, 30, S53-S54.

Iosa, M., Cereatti, A., Merlo, A., Campanini, I., Paolucci, S. and Cappozzo, A. (2014) Assessment of waveform similarity in clinical gait data: the linear fit method. *BioMed research international*, 2014, 214156-214156.

Iosa, M., Picerno, P., Paolucci, S. and Morone, G. (2016) Wearable inertial sensors for human movement analysis. *Expert Review of Medical Devices*, 13 (7), 641-659.

Jacquelin Perry, J.M.B. (2010) *Gait Analysis Normal and Pathological Function*. 2nd ed. USA: SLACK Incorporated.

Johnson, W.R., Mian, A., Donnelly, C.J., Lloyd, D. and Alderson, J. (2018) Predicting athlete ground reaction forces and moments from motion capture. *Medical & Biological Engineering & Computing*.

Jones, L., Holt, C.A. and Beynon, M.J. (2008) Reduction, classification and ranking of motion analysis data: an application to osteoarthritic and normal knee function data. *Computer Methods in Biomechanics and Biomedical Engineering*, 11 (1), 31-40.

Kadaba, M.P., Ramakrishnan, H.K. and Wootten, M.E. (1990) Measurement of lower extremity kinematics during level walking. *J Orthop Res*, 8 (3), 383-392.

Khouri, N. and Desailly, E. (2017) Contribution of clinical gait analysis to single-event multi-level surgery in children with cerebral palsy. *Orthopaedics & Traumatology: Surgery & Research*, 103 (1, Supplement), S105-S111.

King, S.L., Barton, G.J. and Ranganath, L.R. (2017) Interpreting sources of variation in clinical gait analysis: A case study. *Gait Posture*, 52, 1-4.

Kobsar, D., Charlton, J.M., Tse, C.T.F., Esculier, J.F., Graffos, A., Krowchuk, N.M., Thatcher, D. and Hunt, M.A. (2020) Validity and reliability of wearable inertial sensors in healthy adult walking: a systematic review and meta-analysis. *J Neuroeng Rehabil*, 17 (1), 62.

Koch, M., Lunde, L.-K., Ernst, M., Knardahl, S. and Veiersted, K.B. (2016) Validity and reliability of pressure-measurement insoles for vertical ground reaction force assessment in field situations. *Applied Ergonomics*, 53, 44-51.

Korjus, K., Hebart, M.N. and Vicente, R. (2016) An Efficient Data Partitioning to Improve Classification Performance While Keeping Parameters Interpretable. *PLoS One*, 11 (8), e0161788-e0161788.

Leardini, A., Chiari, L., Della Croce, U. and Cappozzo, A. (2005) Human movement analysis using stereophotogrammetry. Part 3. Soft tissue artifact assessment and compensation. *Gait Posture*, 21 (2), 212-225.

Lees, A., Barton, G. and Robinson, M. (2010) The influence of Cardan rotation sequence on angular orientation data for the lower limb in the soccer kick. *J Sports Sci*, 28 (4), 445-450.

Loose, H. and Orlowski, K. (2015) Gait patterns in standard scenarios: Using Xsens MTw inertial measurement units. *2015 16th International Conference on Research and Education in Mechatronics (REM)*, 18-20 Nov. 2015 of Conference.

Luinge, H.J. and Veltink, P.H. (2005) Measuring orientation of human body segments using miniature gyroscopes and accelerometers. *Medical and Biological Engineering and Computing*, 43 (2), 273-282.

Luinge, H.J., Veltink, P.H. and Baten, C.T.M. (1999) Estimation of orientation with gyroscopes and accelerometers. *Proceedings of the First Joint BMES/EMBS Conference. 1999 IEEE Engineering in Medicine and Biology 21st Annual Conference and the 1999 Annual Fall Meeting of the Biomedical Engineering Society (Cat. N*, 13-16 Oct. 1999 of Conference.

Maly, M.R. (2008) Abnormal and cumulative loading in knee osteoarthritis. *Curr Opin Rheumatol*, 20 (5), 547-552.

Mayich, D.J., Novak, A., Vena, D., Daniels, T.R. and Brodsky, J.W. (2014) Gait analysis in orthopedic foot and ankle surgery--topical review, part 1: principles and uses of gait analysis. *Foot Ankle Int*, 35 (1), 80-90.

McGinley, J.L., Baker, R., Wolfe, R. and Morris, M.E. (2009) The reliability of three-dimensional kinematic gait measurements: a systematic review. *Gait Posture*, 29 (3), 360-369.

Michelini, A., Eshraghi, A. and Andrysek, J. (2020) Two-dimensional video gait analysis: A systematic review of reliability, validity, and best practice considerations. *Prosthetics and Orthotics International*, 44 (4), 245-262.

Müller, B., Ilg, W., Giese, M.A. and Ludolph, N. (2017) Validation of enhanced kinect sensor based motion capturing for gait assessment. *PLoS One*, 12 (4), e0175813.

Mundt, M., Koeppe, A., Bamer, F., David, S. and Markert, B. (2020) Artificial Neural Networks in Motion Analysis-Applications of Unsupervised and Heuristic Feature Selection Techniques. *Sensors (Basel)*, 20 (16).

Mundt, M., Koeppe, A., Bamer, F., Potthast, W. and Markert, B. (2018) *Prediction of joint kinetics based on joint kinematics using neural networks*

Mundt, M., Thomsen, W., Witter, T., Koeppe, A., David, S., Bamer, F., Potthast, W. and Markert, B. (2019) Prediction of lower limb joint angles and moments during gait using artificial neural networks. *Medical & Biological Engineering & Computing*, 58.

Myles, A.J., Murray, A.F., Wallace, A.R., Barnard, J. and Smith, G. (1997) Estimating MLP generalisation ability without a test set using fast, approximate leave-one-out cross-validation. *Neural computing & applications*, 5 (3), 134-151.

Nilsson, J. and Thorstensson, A. (1989) Ground reaction forces at different speeds of human walking and running. *Acta Physiologica Scandinavica*, 136 (2), 217-227.

Oh, S.E., Choi, A. and Mun, J.H. (2013) Prediction of ground reaction forces during gait based on kinematics and a neural network model. *Journal of Biomechanics*, 46 (14), 2372-2380.

Orendurff, M.S., Segal, A.D., Klute, G.K., Berge, J.S., Rohr, E.S. and Kadel, N.J. (2004) The effect of walking speed on center of mass displacement. *J Rehabil Res Dev*, 41 (6a), 829-834.

Parsons, H.M. (1974) What Happened at Hawthorne? *Science*, 183 (4128), 922-932.

Pasini, A. (2015) Artificial neural networks for small dataset analysis. *Journal of thoracic disease*, 7 (5), 953-960.

Pauli, C., Graf, E., Wirz, M. and Bauer, C. (2019) *Can off-the-shelf inertial measurement units measure gait kinematics in patients with gait limitations?*

Paulich, M., Schepers, M., Rudigkeit, N. and Bellusci, G. (2018) *Xsens MTw Awinda: Miniature Wireless Inertial-Magnetic Motion Tracker for Highly Accurate 3D Kinematic Applications*.

Pedersen, S.M., Jorgensen, J.S. and Pedersen, J.B. (1996) Use of neural networks to diagnose acute myocardial infarction. II. A clinical application. *Clin Chem*, 42 (4), 613-617.

Peralta, J., Gutierrez, G. and Sanchis de Miguel, A. (2009) *Shuffle Design to Improve Time Series Forecasting Accuracy*.

Picerno, P. (2017) 25 years of lower limb joint kinematics by using inertial and magnetic sensors: A review of methodological approaches. *Gait Posture*, 51, 239-246.

Picerno, P., Cereatti, A. and Cappozzo, A. (2008) Joint kinematics estimate using wearable inertial and magnetic sensing modules. *Gait Posture*, 28 (4), 588-595.

Poitras, I., Dupuis, F., Bielmann, M., Campeau-Lecours, A., Mercier, C., Bouyer, L.J. and Roy, J.-S. (2019) Validity and Reliability of Wearable Sensors for Joint Angle Estimation: A Systematic Review. *Sensors (Basel, Switzerland)*, 19 (7), 1555.

Rafiq, M.Y., Bugmann, G. and Easterbrook, D.J. (2001) Neural network design for engineering applications. *Computers & Structures*, 79 (17), 1541-1552.

Reitermanová, Z. (2010) Data Splitting. *WDS 2010 - Proceedings of Contributed Papers. Proceedings of the 19th Annual Conference of Doctoral Students*of Conference.

Roetenberg, D., Baten, C.T. and Veltink, P.H. (2007) Estimating body segment orientation by applying inertial and magnetic sensing near ferromagnetic materials. *IEEE Trans Neural Syst Rehabil Eng*, 15 (3), 469-471.

Roetenberg, D., Luinge, H.J., Baten, C.T. and Veltink, P.H. (2005) Compensation of magnetic disturbances improves inertial and magnetic sensing of human body segment orientation. *IEEE Trans Neural Syst Rehabil Eng*, 13 (3), 395-405.

Roetenberg, D., Slycke, P.J. and Veltink, P.H. (2007) Ambulatory position and orientation tracking fusing magnetic and inertial sensing. *IEEE Trans Biomed Eng*, 54 (5), 883-890.

Røislien, J., Skare, Ø., Opheim, A. and Rennie, L. (2012) Evaluating the properties of the coefficient of multiple correlation (CMC) for kinematic gait data. *Journal of Biomechanics*, 45 (11), 2014-2018.

Sabatini, A.M. (2011) Estimating three-dimensional orientation of human body parts by inertial/magnetic sensing. *Sensors (Basel)*, 11 (2), 1489-1525.

Sadeghi, H., Allard, P., Shafie, K., Mathieu, P.A., Sadeghi, S., Prince, F. and Ramsay, J. (2000) Reduction of gait data variability using curve registration. *Gait & Posture*, 12 (3), 257-264.

Sandau, M., Koblauch, H., Moeslund, T.B., Aanæs, H., Alkjær, T. and Simonsen, E.B. (2014) Markerless motion capture can provide reliable 3D gait kinematics in the sagittal and frontal plane. *Medical Engineering & Physics*, 36 (9), 1168-1175.

Sapna, S., Tamilarasi, A. and Kumar, M.P. (2012) Backpropagation learning algorithm based on Levenberg Marquardt Algorithm. *Comp Sci Inform Technol (CS and IT)*, 2, 393-398.

Schepers, M., Giuberti, M. and Bellusci, G. (2018) *Xsens MVN: Consistent Tracking of Human Motion Using Inertial Sensing*.

Shahabpoor, E. and Pavic, A. (2017) Measurement of Walking Ground Reactions in Real-Life Environments: A Systematic Review of Techniques and Technologies. *Sensors (Basel)*, 17 (9).

Shepherd, H., Barton, G., Robinson, M. and Ranganath, L. (2020) Taking a moment to consider the medial knee thrust: Gait modifications to reduce the 3D knee moment. *Gait & Posture*, 81, 344-345.

Simon, S.R. (2004) Quantification of human motion: gait analysis-benefits and limitations to its application to clinical problems. *Journal of Biomechanics*, 37 (12), 1869-1880.

Sinitski, E., Lemaire, E. and Baddour, N. (2015) Evaluation of motion platform embedded with force plate-instrumented treadmill. *The Journal of Rehabilitation Research and Development*, 52, 221-234.

Sivakumar, S., Gopalai, A.A., Gouwanda, D. and Hann, L.K. (2016) ANN for gait estimations: A review on current trends and future applications. *2016 IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, 4-8 Dec. 2016 of Conference.

Stetter, B.J., Krafft, F.C., Ringhof, S., Stein, T. and Sell, S. (2020) A Machine Learning and Wearable Sensor Based Approach to Estimate External Knee Flexion and Adduction Moments During Various Locomotion Tasks. *Front Bioeng Biotechnol*, 8, 9.

Svozil, D., Kvasnicka, V. and Pospichal, J.í. (1997) Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, 39 (1), 43-62.

Teng, H.L., MacLeod, T.D., Kumar, D., Link, T.M., Majumdar, S. and Souza, R.B. (2015) Individuals with isolated patellofemoral joint osteoarthritis exhibit higher mechanical loading at the knee during the second half of the stance phase. *Clin Biomech (Bristol, Avon)*, 30 (4), 383-390.

Ugbolue, U.C., Papi, E., Kaliarntas, K.T., Kerr, A., Earl, L., Pomeroy, V.M. and Rowe, P.J. (2013) The evaluation of an inexpensive, 2D, video based gait assessment system for clinical use. *Gait & Posture*, 38 (3), 483-489.

van den Bogert, A.J., Geijtenbeek, T., Even-Zohar, O., Steenbrink, F. and Hardin, E.C. (2013) A real-time system for biomechanical analysis of human movement and muscle function. *Medical & Biological Engineering & Computing*, 51 (10), 1069-1077.

van den Noort, J., van der Esch, M., Steultjens, M.P.M., Dekker, J., Schepers, M., Veltink, P.H. and Harlaar, J. (2013) Ambulatory measurement of the knee adduction moment in patients with osteoarthritis of the knee. *Journal of Biomechanics*, 46 (1), 43-49.

van der Krogt, M.M., Sloot, L.H., Buizer, A.I. and Harlaar, J. (2015) Kinetic comparison of walking on a treadmill versus over ground in children with cerebral palsy. *Journal of Biomechanics*, 48 (13), 3577-3583.

van der Kruk, E. and Reijne, M.M. (2018) Accuracy of human motion capture systems for sport applications; state-of-the-art review. *European Journal of Sport Science*, 18 (6), 806-819.

van Tunen, J.A.C., Dell'Isola, A., Juhl, C., Dekker, J., Steultjens, M., Thorlund, J.B. and Lund, H. (2018) Association of malalignment, muscular dysfunction, proprioception, laxity and abnormal joint loading with tibiofemoral knee osteoarthritis - a systematic review and meta-analysis. *BMC musculoskeletal disorders*, 19 (1), 273-273.

Vastola, R., Medved, V., Albano, D., Coppola, S. and Sibilio, M. (2016) Use of Optoelectronic Systems for the Analysis of Technique in Trials. *Journal of Sports Science*, 4.

Veltink, P.H., Liedtke, C., Droog, E. and Kooij, H.v.d. (2005) Ambulatory measurement of ground reaction forces. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 13 (3), 423-427.

Vincent, K.R., Conrad, B.P., Fregly, B.J. and Vincent, H.K. (2012) The pathophysiology of osteoarthritis: a mechanical perspective on the knee joint. *Pm r*, 4 (5 Suppl), S3-9.

Vos, T., Flaxman, A.D., Naghavi, M., Lozano, R., Michaud, C., Ezzati, M., Shibuya, K., Salomon, J.A., Abdalla, S. and Aboyans, V. (2012) Years lived with disability (YLDs) for 1160 sequelae of 289 diseases and injuries 1990–2010: a systematic analysis for the Global Burden of Disease Study 2010. *The lancet*, 380 (9859), 2163-2196.

Washabaugh, E.P., Kalyanaraman, T., Adamczyk, P.G., Claflin, E.S. and Krishnan, C. (2017) Validity and repeatability of inertial measurement units for measuring gait parameters. *Gait & posture*, 55, 87-93.

Whatling, G.M., Biggs, P.R., Elson, D.W., Metcalfe, A., Wilson, C. and Holt, C. (2020) High tibial osteotomy results in improved frontal plane knee moments, gait patterns and patient-reported outcomes. *Knee surgery, sports traumatology, arthroscopy : official journal of the ESSKA*, 28 (9), 2872-2882.

Willemsen, A.T., van Alsté, J.A. and Boom, H.B. (1990) Real-time gait assessment utilizing a new way of accelerometry. *J Biomech*, 23 (8), 859-863.

Winkelstein, B.A. (2013) Orthopaedic biomechanics.

Winter, D.A. (1990) *Biomechanics and motor control of human movement*. 2 ed.

Winter, D.A. (2009) *Biomechanics and Motor Control of Human Movement, Fourth Edition*. John Wiley & Sons, Inc.

Wren, T.A., Otsuka, N.Y., Bowen, R.E., Scaduto, A.A., Chan, L.S., Dennis, S.W., Rethlefsen, S.A., Healy, B.S., Hara, R., Sheng, M. and Kay, R.M. (2013) Outcomes of lower extremity orthopedic surgery in ambulatory children with cerebral palsy with and without gait analysis: results of a randomized controlled trial. *Gait Posture*, 38 (2), 236-241.

Wren, T.A.L., Otsuka, N.Y., Bowen, R.E., Scaduto, A.A., Chan, L.S., Sheng, M., Hara, R. and Kay, R.M. (2011) Influence of gait analysis on decision-making for lower extremity orthopaedic surgery: Baseline data from a randomized controlled trial. *Gait & Posture*, 34 (3), 364-369.

Wren, T.A.L., Tucker, C.A., Rethlefsen, S.A., Gorton, G.E., 3rd and Õunpuu, S. (2020) Clinical efficacy of instrumented gait analysis: Systematic review 2020 update. *Gait Posture*, 80, 274-279.

Xsens (2020) MVN User Manual: User Guide MVN, MVN BIOMECH MVN Link, MVN Awinda. Available online: https://www.xsens.com/hubfs/Downloads/Manuals/MVN_User_Manual.pdf?hsCtaTracking=1eacb2ea-b005-43d0-a2b7-55dbf7f6a4f7%7Cb08403d4-214e-42c7-9920-18ab2f35e714.

Yelin, E., Murphy, L., Cisternas, M.G., Foreman, A.J., Pasta, D.J. and Helmick, C.G. (2007) Medical care expenditures and earnings losses among persons with arthritis and other rheumatic conditions in 2003, and comparisons with 1997. *Arthritis Rheum*, 56 (5), 1397-1407.

Zayani, R., Bouallegue, R. and Roviras, D. (2008) Levenberg-Marquardt learning neural network for adaptive predistortion for time-varying HPA with memory in OFDM systems. *2008 16th European Signal Processing Conference*, 25-29 Aug. 2008 of Conference.

Zhang, J.T., Novak, A.C., Brouwer, B. and Li, Q. (2013) Concurrent validation of Xsens MVN measurement of lower limb joint angular kinematics. *Physiol Meas*, 34 (8), N63-69.

Zult, T., Allsop, J., Tabernero, J. and Pardhan, S. (2019) A low-cost 2-D video system can accurately and reliably assess adaptive gait kinematics in healthy and low vision subjects. *Scientific Reports*, 9 (1), 18385.

Zurada, J.M. (1992) *Introduction to artificial neural systems*. West St. Paul.

Appendix 1. Risk assessment protocol of the Movement Function Research Laboratory

LIVERPOOL JOHN MOORES UNIVERSITY

Risk Assessment – Core Procedures

Core Procedure Details

Procedure*	2D and 3D Motion Capture	**School***	School of Sport & Exercise Sciences
Procedure Short Code*	2D3DMoCap	**Laboratory/Room***	No Specific Laboratory/Room Affiliated
Procedure Category*	Kinematic and Kinetic Analyses	**Review Period (months)***	12

Conductor and Consultee Details (LJMU Staff Details Only)

Conductor Name*	Gibbon, Karl	**Conductor Email Address***	k.c.gibbon@ljmu.ac.uk
Consultee 1 Name*	Barton, Gabor	**Consultee 1 Email Address***	g.j.barton@ljmu.ac.uk
Consultee 2 Name	Robinson, Mark	**Consultee 2 Email Address**	m.a.robinson@ljmu.ac.uk
Consultee 3 Name	Lake, Mark	**Consultee 3 Email Address**	m.j.lake@ljmu.ac.uk

External Consultation (if applicable)

External Consultant(s)	N/A

* **Indicates required field**

Advisory Notices:

Note 1. Printed copies of this document may not be most recent version. The definitive version is held at https://teams.ljmu.ac.uk/2/sps/ResInstSES/RA_Database_CoreProcedures/RA_SPS_Core_2D3DMoCap_060818.docx.

Note 2. Governance Policy.
All risk assessments are to be conducted in accordance with the Health and Safety Code of Practice SCP18 - Risk Assessment (available at https://policies.ljmu.ac.uk/UserHome/Policies/PolicyView.aspx?c=5).

Note 3. Risk Assessment Review.
As per Section 5.5 of SCP 18 – Risk Assessment, all risk assessments are to be reviewed at least every 12 months. More frequent reviews can/may be requested upon approval review or as reasonably necessitated by procedural changes, incidents, adverse events etc.

Note 4. Implementation Responsibility.
Proper implementation of any and all risk control measures specified below is the responsibility of the person undertaking the activity specified within this risk assessment.

Note 5. Childbearing Age Consideration.
As per Section 5.2.1 of SCP18 – Risk Assessment, all risk assessments must include consideration of risks to women of childbearing age as some activities may have an effect on a foetus or woman's reproductive system.

Note 6. Out of Hours Working.
For all activities performed outside of normal working hours (09:00-17:00 Monday to Friday) please refer to the hazards & risk control measures outlined in the section 'Out of Hours and Lone Working Hazards & Risk Control Measures' below. In addition you should familiarise yourself with Health and Safety Code of Practice (SCP) for 1) Fire evacuation and assisted evacuation (SCP2 and SCP37), 2) First Aid (SCP30), and 3) Lone working (SCP25).

Note 7. Lone Working.
LJMU staff and students are not permitted to work alone when the activity being undertaken is specifically prohibited within section 4 of SCP25 – Lone Working. Please refer to the hazards & risk control measures outlined in the section 'Out of Hours and Lone Working Hazards & Risk Control Measures' below.

Note 8. Risk Assessment Responsibility and Training.
As per section 4.1 of SCP 18 – Risk Assessment, in every case the risk assessment is the responsibility of the person supervising or directing the activity. This task may be delegated but the responsibility remains entirely with the supervising person. As such it is recommended that all supervisors attend the Risk Assessment Training Course offered by the Health & Safety Unit. If you wish to attend one of these courses please consult the training planner available at https://www2.ljmu.ac.uk/HSU/65140.htm.

Document Control Information (for office use only)

Filename	RA_SPS_Core_2D3DMoCap_060818.docx	**Last Modified**	06/08/2018 15:10:41	**Approval Status**	Draft	**Reviewed By**	[Reviewer]
				Reviewed On	[Last Reviewed]	**Expiry Date**	[Expiry Date]

LIVERPOOL JOHN MOORES UNIVERSITY

Risk Assessment – Core Procedures

Activity/Procedure Specific Hazards & Risk Control Measures

1. What are the Hazards?	2(a). Who might be harmed and how?	3(a). What are you already doing?	3(b). What further action is needed?	4. How will you put the assessment into action?
Skin irritants and allergenic allergic substances.	For marker based motion capture retroreflective markers are typically attached directly to the skin. Dependent on the attachment method used exposure to skin irritants and allergenic substances is possible.	Only medically approved double sided adhesive tape is to be used. If required, pre-tape spray (e.g. Mueller Tuffner Pre-Tape Adherent or Muller Mueller Quick-Drying Adherent) can be used. Glue may also be used if the application of double sided tape in conjunction with pre-tape adherent is ineffective. This glue must be formulated for use on skin (e.g. medical dermal adhesive). The use of cyanoacrylate glue (i.e. superglue) is not permitted. Should signs of an adverse reaction, such as excessive reddening, inflammation, hives or pruritus be observed/reported the activity is to cease and the adhesive tape is to be removed immediately.	When pre-tape spray is used, manufacturer's guidelines for application/removal must be followed.	Steps 1, 2, 3a and 3b are being taken to ensure minimum risk for participants and experimenters. Additionally this risk assessment is to be stored digitally and provided openly to both students and staff members.
Restricted blood supply to tissues (ischemia)	When using material wrap to secure articles to the body (e.g. rigid marker clusters) for motion capture excessive tightening can result in ischemia.	No non-elastic material is permitted for use in this application. Only latex-free elastic compression bandages, such as SuperWrap™ or Coban™ are permitted. A capillary refill time check of distal appendages should be conducted. Wrap is e removed and reapplied if refill time is excessive. Typical refill time is 2 seconds.	Should signs/symptoms of ischemia (e.g. paraesthesia, swelling, discolouration) be observed/reported the activity is to cease and the material wrap be removed immediately.	As above.

Document Control Information (for office use only)

Filename	RA_SPS_Core_2D3DMoCap_060818.docx	**Last Modified**	06/08/2018 15:10:41	**Approval Status**	Draft	**Reviewed By**	[Reviewer]
				Reviewed On	[Last Reviewed]	**Expiry Date**	[Expiry Date]

Trailing cables	Dependent upon the type of capture system used trailing cables may be unavoidable and could increase trip risk to anyone passing through/around the capture environment.	Loose/trailing cables are to be managed as carefully as possible in terms of ensuring cable length; • Is sufficient enough to traverse the perimeter of the capture volume • Is not excessive by coiling and stowing where practicable • Is routed (as much as is possible) away from access routes, walkways and thoroughfare. Where cabling across access routes, walkways and thoroughfares is not avoidable use of cable covers is required.	None.	As above.
Lighting	Additional lighting may be required to properly illuminate the reflective markers attached to the participant's skin. Particularly when video based camera technology is used. This lighting can be bright and staring into bright lights can cause retinal damage.	Within a laboratory environment ceiling/truss mounted floodlights are to be used initially. This ensure the direction of light is not aimed directly at the participant during typical ground-level activities. If additional/custom lighting is required this should be positioned such that the principal direction of light is not directly facing the participant and is above the typical gaze pattern if possible.	None.	As above.
Childbearing	No known contraindications exist to performing 2D or 3D motion capture on pregnant individuals or individuals of childbearing age.	N/A	None	N/A

Out of Normal Working Hours and Lone Working Hazards & Risk Control Measures

1. What are the Hazards?	2(a). Who might be harmed and how?	3(a). What are you already doing?	3(b). What further action is needed?	4. How will you put the assessment into action?

a). Lone Working: Reduced ability to summon emergency assistance from LJMU First Aid trained staff. **b). Out of Normal Working Hours (09:00-17:00 Monday-Friday) Working:** Reduced availability of emergency assistance from LJMU First Aid trained staff. **c). Inability to leave the building unaided when required to evacuate in an emergency.**	Research study participants and/or LJMU staff and students **a).** There is an increased risk associated with any activity performed alone due to a reduced ability to summon emergency assistance should it be required. For clarification, lone working is the absence of other LJMU staff or students as research study participants (who may be members of the public or other visitors) cannot be expected to be familiar with LJMU emergency procedures. **b).** There is an increased risk associated with any activity performed outside of 'normal working hours' due to the reduced availability of emergency assistance. **c).** Individuals with reduced mobility working alone and outside of normal working hours may be trapped in the event of an emergency requiring evacuation.	**a).** LJMU Health and Safety Code of Practice SCP25 (Lone Working) must be read and understood by all individuals working alone. **a & b).** Signage must be in place containing direct contact information for LJMU 'First Aid Trained Staff' and detailing the 'Emergency Procedure for Calling an Ambulance'. **b).** LJMU Health and Safety Code of Practice SCP11 (Out of Hours Working) must be read and understood by all individuals working outside of normal working hours. For these activities the out of hours approval form (located in the appendix of SCP11) must be completed and signed by the academic supervisor and Director (Head) of School prior to final review by the Dean of Faculty. Subsequently security services must be informed of your presence in and departure from the workplace. **c).** As part of the out of hours approval process individuals requiring assistance to leave the building in the event of an emergency must inform security of this need.	**a).** None **b).** None **c).** None	**a & b).** Maintain an awareness of the laboratory signage and procedures for obtaining First Aid assistance from; •□Staff based in the Tom Reilly Building (refer to 'First Aid Trained Staff' list). •□Security Lodge (emergency extension number 2222, 0151 231 2222 from a mobile phone). Maintain an awareness of the laboratory signage and procedures for calling an ambulance; Call 999 from a mobile phone or call 9999 from an LJMU internal phone. **b & c).** Individuals working alone outside of normal working hours are to inform security services of their presence in and departure from the workplace. **c).** Maintain an awareness of emergency refuge points with direct communication systems located within the building.

Appendix 2. A Matlab script used for the prediction hip, knee and ankle joint moment from marker coordinates data

```
% Solve an Input-Output Fitting problem with a Neural Network
% Script generated by Neural Fitting app
% Created 25-Apr-2017 18:38:05
%
% This script assumes these variables are defined:
%
%   walk3leftinput - input data.
%   walk3rightankhipkneemoment - target data.

x = walk3leftinput';
t = walk3rightankhipkneemoment';

% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for challenging problems.
% 'trainscg' uses less memory. Suitable in low memory situations.
trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.

% Create a Fitting Network
hiddenLayerSize = 10;
net = fitnet(hiddenLayerSize,trainFcn);

% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Train the Network
[net,tr] = train(net,x,t);

% Test the Network
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y)

% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, ploterrhist(e)
%figure, plotregression(t,y)
%figure, plotfit(net,x,t)
```

Appendix 3. The synchronization method used in this PhD project

The Xsens and Vicon system were synchronised using a trigger start and stop signal sent from the Xsens Awinda station via two BNC cables (figure 12.1). An extra analogue channel was wired to the force signal receiver that sends the analogue force data from the treadmill to the Vicon system. Subsequently when the data were recorded, the start signal of each set of gait was sent from the Xsens Awinda station towards Vicon. The same signals were then streamed through D-Flow where the signal of the extra analogue channel could be seen as a unique spike in the analogue signals (figure 12.2.2) that would later be saved as a .txt file in D-Flow.

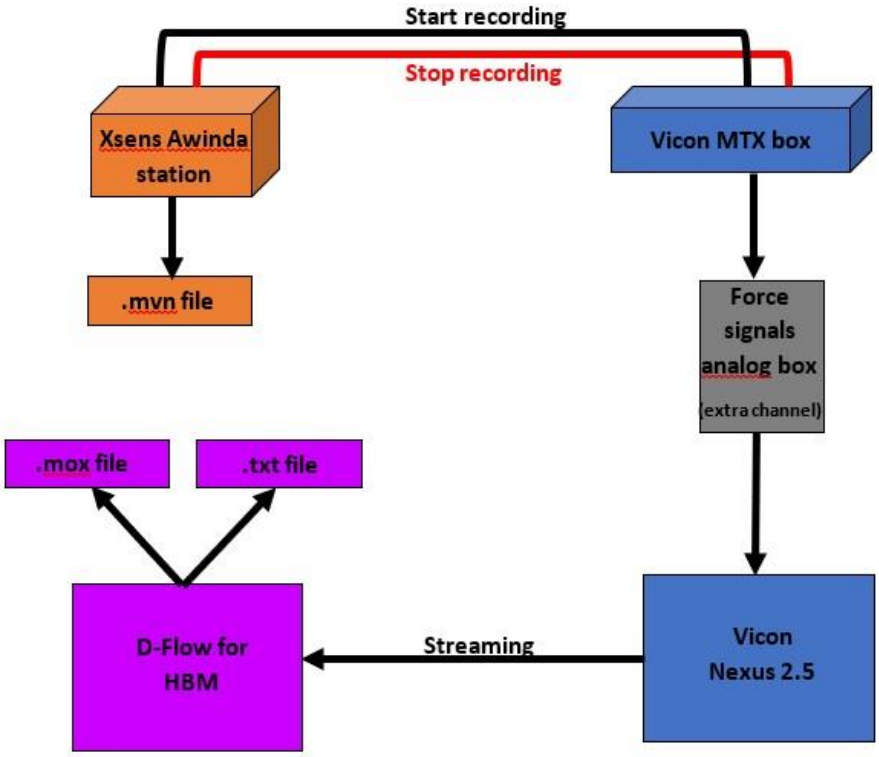

Figure 12.1 The block diagram illustrates the synchronisation of the two motion capture systems (Xsens and HBM via Vicon Nexus 2.5). An extra analogue channel was wired to the force analogue box which was connected from the instrumented treadmill to Vicon Nexus. Once the Xsens data collection started, the data were collected by both systems, including the extra analogue channel. Finally, all the data were streamed through D-Flow where the gait data for the HBM model were saved in two formats (.mox and .txt). The common starting point for the Xsens and Vicon data was the rising edge of the extra channel in the saved .txt file.

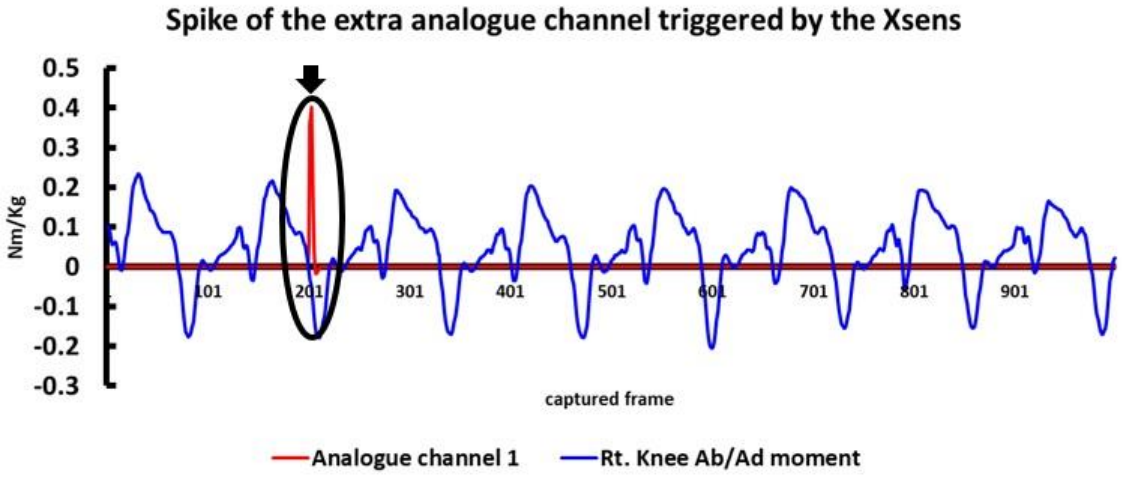

Figure 12.2 The spike of the extra analogue signal, channel 1 (red), triggered by the Xsens system when the data recording was begun (black circle). The signal was saved as a .txt file as channel 1. The gait data, for instant, the KAM (blue) was saved as a .mox file from GOAT. Both channel 1 and KAM had been recorded via the Vicon system and streamed through D-Flow a few seconds before the Xsens system. The peak of the spike (black arrow) indicates the first frame that the data were recorded by the Xsens and was used as the first data point to align data from Xsens and D-Flow by the captured frame for synchronisation.

Appendix 4. A Matlab script used to calculate coefficient of correlation (CMC) for gait data of a participant

```
% this script is to calculate the mean joint angle estimation error
and
% the coefficient of multiple correlation for p8 normal001
% there are 5 gait cycles for each trial
% there are 3 trials for each speed
% there are 3 speeds for a participant (fast, normal, slow)
% there are 2 sides to study (right,left)
% there are 3 joints to study (ankle,hip,knee)
% there are 3 axes for each joint (ab,fl,in)
% ? is an angle at a particular captured frame
% ?vf is an angle from vicon at a particular frame
% ?xf is an angle from xsens at a particular frame
% f = a captured frame
% F = 101 (the total captured frames per cycle)
% P = 2 (number of motion capturing methods/ 1= vicon, 1= xsens)
% G = 5 (number of gait cycles)

p8vileftanklenormal001in_f = p8vileftanklenormal001in_f *-1;

%%%the mean joint estimation error (MJEANKLE or ?e)
% Left ankle
MJEANKLEG1ab = Np8vileftanklenormal001ab(:,1) -
Np8xsleftanklenormal001ab(:,1);
MJEANKLEG1ab = abs(MJEANKLEG1ab);
MJEANKLEG1ab = sum(MJEANKLEG1ab);
MJEANKLEG1ab = MJEANKLEG1ab *1/101;
MJEANKLEG2ab = Np8vileftanklenormal001ab(:,2) -
Np8xsleftanklenormal001ab(:,2);
MJEANKLEG2ab = abs(MJEANKLEG2ab);
MJEANKLEG2ab = sum(MJEANKLEG2ab);
MJEANKLEG2ab = MJEANKLEG2ab *1/101;
MJEANKLEG3ab = Np8vileftanklenormal001ab(:,3) -
Np8xsleftanklenormal001ab(:,3);
MJEANKLEG3ab = abs(MJEANKLEG3ab);
MJEANKLEG3ab = sum(MJEANKLEG3ab);
MJEANKLEG3ab = MJEANKLEG3ab *1/101;
MJEANKLEG4ab = Np8vileftanklenormal001ab(:,4) -
Np8xsleftanklenormal001ab(:,4);
MJEANKLEG4ab = abs(MJEANKLEG4ab);
MJEANKLEG4ab = sum(MJEANKLEG4ab);
MJEANKLEG4ab = MJEANKLEG4ab *1/101;
MJEANKLEG5ab = Np8vileftanklenormal001ab(:,5) -
Np8xsleftanklenormal001ab(:,5);
MJEANKLEG5ab = abs(MJEANKLEG5ab);
MJEANKLEG5ab = sum(MJEANKLEG5ab);
MJEANKLEG5ab = MJEANKLEG5ab *1/101;
MJEANKLEG1fl = Np8vileftanklenormal001fl(:,1) -
Np8xsleftanklenormal001fl(:,1);
MJEANKLEG1fl = abs(MJEANKLEG1fl);
MJEANKLEG1fl = sum(MJEANKLEG1fl);
MJEANKLEG1fl = MJEANKLEG1fl *1/101;
MJEANKLEG2fl = Np8vileftanklenormal001fl(:,2) -
Np8xsleftanklenormal001fl(:,2);
MJEANKLEG2fl = abs(MJEANKLEG2fl);
MJEANKLEG2fl = sum(MJEANKLEG2fl);
MJEANKLEG2fl = MJEANKLEG2fl *1/101;
MJEANKLEG3fl = Np8vileftanklenormal001fl(:,3) -
Np8xsleftanklenormal001fl(:,3);
```

```
MJEANKLEG3fl = abs(MJEANKLEG3fl);
MJEANKLEG3fl = sum(MJEANKLEG3fl);
MJEANKLEG3fl = MJEANKLEG3fl *1/101;
MJEANKLEG4fl = Np8vileftanklenormal001fl(:,4) -
Np8xsleftanklenormal001fl(:,4);
MJEANKLEG4fl = abs(MJEANKLEG4fl);
MJEANKLEG4fl = sum(MJEANKLEG4fl);
MJEANKLEG4fl = MJEANKLEG4fl *1/101;
MJEANKLEG5fl = Np8vileftanklenormal001fl(:,5) -
Np8xsleftanklenormal001fl(:,5);
MJEANKLEG5fl = abs(MJEANKLEG5fl);
MJEANKLEG5fl = sum(MJEANKLEG5fl);
MJEANKLEG5fl = MJEANKLEG5fl *1/101;
MJEANKLEG1in = Np8vileftanklenormal001in(:,1) -
Np8xsleftanklenormal001in(:,1);
MJEANKLEG1in = abs(MJEANKLEG1in);
MJEANKLEG1in = sum(MJEANKLEG1in);
MJEANKLEG1in = MJEANKLEG1in *1/101;
MJEANKLEG2in = Np8vileftanklenormal001in(:,2) -
Np8xsleftanklenormal001in(:,2);
MJEANKLEG2in = abs(MJEANKLEG2in);
MJEANKLEG2in = sum(MJEANKLEG2in);
MJEANKLEG2in = MJEANKLEG2in *1/101;
MJEANKLEG3in = Np8vileftanklenormal001in(:,3) -
Np8xsleftanklenormal001in(:,3);
MJEANKLEG3in = abs(MJEANKLEG3in);
MJEANKLEG3in = sum(MJEANKLEG3in);
MJEANKLEG3in = MJEANKLEG3in *1/101;
MJEANKLEG4in = Np8vileftanklenormal001in(:,4) -
Np8xsleftanklenormal001in(:,4);
MJEANKLEG4in = abs(MJEANKLEG4in);
MJEANKLEG4in = sum(MJEANKLEG4in);
MJEANKLEG4in = MJEANKLEG4in *1/101;
MJEANKLEG5in = Np8vileftanklenormal001in(:,5) -
Np8xsleftanklenormal001in(:,5);
MJEANKLEG5in = abs(MJEANKLEG5in);
MJEANKLEG5in = sum(MJEANKLEG5in);
MJEANKLEG5in = MJEANKLEG5in *1/101;

%%CMC
% ? is an angle at a particular captured frame
% ?vf is an angle from vicon at a particular frame
% ?xf is an angle from xsens at a particular frame
% f = a captured frame
% F = 101 (the total captured frames per cycle)
% P = 2 (number of motion capturing methods/ 1= vicon, 1= xsens)
% G = 5 (number of gait cycles)

%GFg(P-1) =   5*101(2-1) = 505
%GFg(P-1) =   4*101(2-1) = 404  %% p17 slow speed only
%G(PFg-1) = 5(2*101-1) = 1005
%G(PFg-1) = 4(2*101-1) = 804%% p17 slow speed only

% the mean angle at frame f (Mf) between angles measured by the 2
systems for the gait cycle g
%Mf = Mf/2;  (2 is P)

%%%%leftanklefl
MfANKLEG1FL = Np8vileftanklenormal001fl(:,1) +
Np8xsleftanklenormal001fl(:,1);
MfANKLEG1FL = MfANKLEG1FL/2;
```

```
MfANKLEG2FL = Np8vileftanklenormal001fl(:,2) +
Np8xsleftanklenormal001fl(:,2);
MfANKLEG2FL = MfANKLEG2FL/2;
MfANKLEG3FL = Np8vileftanklenormal001fl(:,3) +
Np8xsleftanklenormal001fl(:,3);
MfANKLEG3FL = MfANKLEG3FL/2;
MfANKLEG4FL = Np8vileftanklenormal001fl(:,4) +
Np8xsleftanklenormal001fl(:,4);
MfANKLEG4FL = MfANKLEG4FL/2;
MfANKLEG5FL = Np8vileftanklenormal001fl(:,5) +
Np8xsleftanklenormal001fl(:,5);
MfANKLEG5FL = MfANKLEG5FL/2;

%% the grand mean (Mg) for the gait cycle "g" among these two
methods

MgankleFLnormal001 = sum(Np8vileftanklenormal001fl)+
sum(Np8xsleftanklenormal001fl);
MgankleFLnormal001 = MgankleFLnormal001/202; % devided by 1/2F;

CMCLeftankleviFLG1 = Np8vileftanklenormal001fl(:,1)- MfANKLEG1FL;
CMCLeftankleviFLG1 = CMCLeftankleviFLG1.^2;
CMCLeftankleviFLG1 = sum(CMCLeftankleviFLG1);
CMCLeftankleviFLG1 = CMCLeftankleviFLG1/505;
CMCLeftanklexsFLG1 = Np8xsleftanklenormal001fl(:,1)- MfANKLEG1FL;
CMCLeftanklexsFLG1 = CMCLeftanklexsFLG1.^2;
CMCLeftanklexsFLG1 = sum(CMCLeftanklexsFLG1);
CMCLeftanklexsFLG1 = CMCLeftanklexsFLG1/505;
CMCLeftankleviFLG2 = Np8vileftanklenormal001fl(:,2)- MfANKLEG2FL;
CMCLeftankleviFLG2 = CMCLeftankleviFLG2.^2;
CMCLeftankleviFLG2 = sum(CMCLeftankleviFLG2);
CMCLeftankleviFLG2 = CMCLeftankleviFLG2/505;
CMCLeftanklexsFLG2 = Np8xsleftanklenormal001fl(:,2)- MfANKLEG2FL;
CMCLeftanklexsFLG2 = CMCLeftanklexsFLG2.^2;
CMCLeftanklexsFLG2 = sum(CMCLeftanklexsFLG2);
CMCLeftanklexsFLG2 = CMCLeftanklexsFLG2/505;
CMCLeftankleviFLG3 = Np8vileftanklenormal001fl(:,3)- MfANKLEG3FL;
CMCLeftankleviFLG3 = CMCLeftankleviFLG3.^2;
CMCLeftankleviFLG3 = sum(CMCLeftankleviFLG3);
CMCLeftankleviFLG3 = CMCLeftankleviFLG3/505;
CMCLeftanklexsFLG3 = Np8xsleftanklenormal001fl(:,3)- MfANKLEG3FL;
CMCLeftanklexsFLG3 = CMCLeftanklexsFLG3.^2;
CMCLeftanklexsFLG3 = sum(CMCLeftanklexsFLG3);
CMCLeftanklexsFLG3 = CMCLeftanklexsFLG3/505;
CMCLeftankleviFLG4 = Np8vileftanklenormal001fl(:,4)- MfANKLEG4FL;
CMCLeftankleviFLG4 = CMCLeftankleviFLG4.^2;
CMCLeftankleviFLG4 = sum(CMCLeftankleviFLG4);
CMCLeftankleviFLG4 = CMCLeftankleviFLG4/505;
CMCLeftanklexsFLG4 = Np8xsleftanklenormal001fl(:,4)- MfANKLEG4FL;
CMCLeftanklexsFLG4 = CMCLeftanklexsFLG4.^2;
CMCLeftanklexsFLG4 = sum(CMCLeftanklexsFLG4);
CMCLeftanklexsFLG4 = CMCLeftanklexsFLG4/505;
CMCLeftankleviFLG5 = Np8vileftanklenormal001fl(:,5)- MfANKLEG5FL;
CMCLeftankleviFLG5 = CMCLeftankleviFLG5.^2;
CMCLeftankleviFLG5 = sum(CMCLeftankleviFLG5);
CMCLeftankleviFLG5 = CMCLeftankleviFLG5/505;
CMCLeftanklexsFLG5 = Np8xsleftanklenormal001fl(:,5)- MfANKLEG5FL;
CMCLeftanklexsFLG5 = CMCLeftanklexsFLG5.^2;
CMCLeftanklexsFLG5 = sum(CMCLeftanklexsFLG5);
CMCLeftanklexsFLG5 = CMCLeftanklexsFLG5/505;

CMCLeftankleFLG1 = CMCLeftankleviFLG1 + CMCLeftanklexsFLG1;
```

```
CMCLeftankleFLG2 = CMCLeftankleviFLG2 + CMCLeftanklexsFLG2;
CMCLeftankleFLG3 = CMCLeftankleviFLG3 + CMCLeftanklexsFLG3;
CMCLeftankleFLG4 = CMCLeftankleviFLG4 + CMCLeftanklexsFLG4;
CMCLeftankleFLG5 = CMCLeftankleviFLG5 + CMCLeftanklexsFLG5;

CMCLeftankleFL1 = CMCLeftankleFLG1 + CMCLeftankleFLG2 +
CMCLeftankleFLG3 + CMCLeftankleFLG4 + CMCLeftankleFLG5;

CMC2LeftankleviFLG1 = Np8vileftanklenormal001fl(:,1)-
MgankleFLnormal001(1,1);
CMC2LeftankleviFLG1 = CMC2LeftankleviFLG1.^2;
CMC2LeftankleviFLG1 = sum(CMC2LeftankleviFLG1);
CMC2LeftankleviFLG1 = CMC2LeftankleviFLG1/1005;
CMC2LeftanklexsFLG1 = Np8xsleftanklenormal001fl(:,1)-
MgankleFLnormal001(1,1);
CMC2LeftanklexsFLG1 = CMC2LeftanklexsFLG1.^2;
CMC2LeftanklexsFLG1 = sum(CMC2LeftanklexsFLG1);
CMC2LeftanklexsFLG1 = CMC2LeftanklexsFLG1/1005;
CMC2LeftankleviFLG2 = Np8vileftanklenormal001fl(:,2)-
MgankleFLnormal001(1,2);
CMC2LeftankleviFLG2 = CMC2LeftankleviFLG2.^2;
CMC2LeftankleviFLG2 = sum(CMC2LeftankleviFLG2);
CMC2LeftankleviFLG2 = CMC2LeftankleviFLG2/1005;
CMC2LeftanklexsFLG2 = Np8xsleftanklenormal001fl(:,2)-
MgankleFLnormal001(1,2);
CMC2LeftanklexsFLG2 = CMC2LeftanklexsFLG2.^2;
CMC2LeftanklexsFLG2 = sum(CMC2LeftanklexsFLG2);
CMC2LeftanklexsFLG2 = CMC2LeftanklexsFLG2/1005;
CMC2LeftankleviFLG3 = Np8vileftanklenormal001fl(:,3)-
MgankleFLnormal001(1,3);
CMC2LeftankleviFLG3 = CMC2LeftankleviFLG3.^2;
CMC2LeftankleviFLG3 = sum(CMC2LeftankleviFLG3);
CMC2LeftankleviFLG3 = CMC2LeftankleviFLG3/1005;
CMC2LeftanklexsFLG3 = Np8xsleftanklenormal001fl(:,3)-
MgankleFLnormal001(1,3);
CMC2LeftanklexsFLG3 = CMC2LeftanklexsFLG3.^2;
CMC2LeftanklexsFLG3 = sum(CMC2LeftanklexsFLG3);
CMC2LeftanklexsFLG3 = CMC2LeftanklexsFLG3/1005;
CMC2LeftankleviFLG4 = Np8vileftanklenormal001fl(:,4)-
MgankleFLnormal001(1,4);
CMC2LeftankleviFLG4 = CMC2LeftankleviFLG4.^2;
CMC2LeftankleviFLG4 = sum(CMC2LeftankleviFLG4);
CMC2LeftankleviFLG4 = CMC2LeftankleviFLG4/1005;
CMC2LeftanklexsFLG4 = Np8xsleftanklenormal001fl(:,4)-
MgankleFLnormal001(1,4);
CMC2LeftanklexsFLG4 = CMC2LeftanklexsFLG4.^2;
CMC2LeftanklexsFLG4 = sum(CMC2LeftanklexsFLG4);
CMC2LeftanklexsFLG4 = CMC2LeftanklexsFLG4/1005;
CMC2LeftankleviFLG5 = Np8vileftanklenormal001fl(:,5)-
MgankleFLnormal001(1,5);
CMC2LeftankleviFLG5 = CMC2LeftankleviFLG5.^2;
CMC2LeftankleviFLG5 = sum(CMC2LeftankleviFLG5);
CMC2LeftankleviFLG5 = CMC2LeftankleviFLG5/1005;
CMC2LeftanklexsFLG5 = Np8xsleftanklenormal001fl(:,5)-
MgankleFLnormal001(1,5);
CMC2LeftanklexsFLG5 = CMC2LeftanklexsFLG5.^2;
CMC2LeftanklexsFLG5 = sum(CMC2LeftanklexsFLG5);
CMC2LeftanklexsFLG5 = CMC2LeftanklexsFLG5/1005;

CMC2LeftankleFLG1 = CMC2LeftankleviFLG1 + CMC2LeftanklexsFLG1;
CMC2LeftankleFLG2 = CMC2LeftankleviFLG2 + CMC2LeftanklexsFLG2;
CMC2LeftankleFLG3 = CMC2LeftankleviFLG3 + CMC2LeftanklexsFLG3;
```

```
CMC2LeftankleFLG4 = CMC2LeftankleviFLG4 + CMC2LeftanklexsFLG4;
CMC2LeftankleFLG5 = CMC2LeftankleviFLG5 + CMC2LeftanklexsFLG5;


CMC2LeftankleFL1 = CMC2LeftankleFLG1 + CMC2LeftankleFLG2 +
CMC2LeftankleFLG3 + CMC2LeftankleFLG4 + CMC2LeftankleFLG5;

CMCleftankleflp8normal001 = CMCLeftankleFL1/ CMC2LeftankleFL1;
CMCleftankleflp8normal001 = 1- CMCleftankleflp8normal001;
CMCleftankleflp8normal001 = sqrt(CMCleftankleflp8normal001)


%%%%leftankleab

% the mean angle at frame f (Mf) between angles measured by the 2
systems for the gait cycle g
%Mf = Mf/2;  (2 is P)

MfANKLEG1AB = Np8vileftanklenormal001ab(:,1) +
Np8xsleftanklenormal001ab(:,1);
MfANKLEG1AB = MfANKLEG1AB/2;
MfANKLEG2AB = Np8vileftanklenormal001ab(:,2) +
Np8xsleftanklenormal001ab(:,2);
MfANKLEG2AB = MfANKLEG2AB/2;
MfANKLEG3AB = Np8vileftanklenormal001ab(:,3) +
Np8xsleftanklenormal001ab(:,3);
MfANKLEG3AB = MfANKLEG3AB/2;
MfANKLEG4AB = Np8vileftanklenormal001ab(:,4) +
Np8xsleftanklenormal001ab(:,4);
MfANKLEG4AB = MfANKLEG4AB/2;
MfANKLEG5AB = Np8vileftanklenormal001ab(:,5) +
Np8xsleftanklenormal001ab(:,5);
MfANKLEG5AB = MfANKLEG5AB/2;

%% the grand mean (Mg) for the gait cycle "g" among these two
methods

MgankleABnormal001 = sum(Np8vileftanklenormal001ab)+
sum(Np8xsleftanklenormal001ab);
MgankleABnormal001 = MgankleABnormal001/202; % devided by 1/2F;

CMCLeftankleviABG1 = Np8vileftanklenormal001ab(:,1)- MfANKLEG1AB;
CMCLeftankleviABG1 = CMCLeftankleviABG1.^2;
CMCLeftankleviABG1 = sum(CMCLeftankleviABG1);
CMCLeftankleviABG1 = CMCLeftankleviABG1/505;
CMCLeftanklexsABG1 = Np8xsleftanklenormal001ab(:,1)- MfANKLEG1AB;
CMCLeftanklexsABG1 = CMCLeftanklexsABG1.^2;
CMCLeftanklexsABG1 = sum(CMCLeftanklexsABG1);
CMCLeftanklexsABG1 = CMCLeftanklexsABG1/505;
CMCLeftankleviABG2 = Np8vileftanklenormal001ab(:,2)- MfANKLEG2AB;
CMCLeftankleviABG2 = CMCLeftankleviABG2.^2;
CMCLeftankleviABG2 = sum(CMCLeftankleviABG2);
CMCLeftankleviABG2 = CMCLeftankleviABG2/505;
CMCLeftanklexsABG2 = Np8xsleftanklenormal001ab(:,2)- MfANKLEG2AB;
CMCLeftanklexsABG2 = CMCLeftanklexsABG2.^2;
CMCLeftanklexsABG2 = sum(CMCLeftanklexsABG2);
CMCLeftanklexsABG2 = CMCLeftanklexsABG2/505;
CMCLeftankleviABG3 = Np8vileftanklenormal001ab(:,3)- MfANKLEG3AB;
CMCLeftankleviABG3 = CMCLeftankleviABG3.^2;
CMCLeftankleviABG3 = sum(CMCLeftankleviABG3);
CMCLeftankleviABG3 = CMCLeftankleviABG3/505;
```

```
CMCLeftanklexsABG3 = Np8xsleftanklenormal001ab(:,3)- MfANKLEG3AB;
CMCLeftanklexsABG3 = CMCLeftanklexsABG3.^2;
CMCLeftanklexsABG3 = sum(CMCLeftanklexsABG3);
CMCLeftanklexsABG3 = CMCLeftanklexsABG3/505;
CMCLeftankleviABG4 = Np8vileftanklenormal001ab(:,4)- MfANKLEG4AB;
CMCLeftankleviABG4 = CMCLeftankleviABG4.^2;
CMCLeftankleviABG4 = sum(CMCLeftankleviABG4);
CMCLeftankleviABG4 = CMCLeftankleviABG4/505;
CMCLeftanklexsABG4 = Np8xsleftanklenormal001ab(:,4)- MfANKLEG4AB;
CMCLeftanklexsABG4 = CMCLeftanklexsABG4.^2;
CMCLeftanklexsABG4 = sum(CMCLeftanklexsABG4);
CMCLeftanklexsABG4 = CMCLeftanklexsABG4/505;
CMCLeftankleviABG5 = Np8vileftanklenormal001ab(:,5)- MfANKLEG5AB;
CMCLeftankleviABG5 = CMCLeftankleviABG5.^2;
CMCLeftankleviABG5 = sum(CMCLeftankleviABG5);
CMCLeftankleviABG5 = CMCLeftankleviABG5/505;
CMCLeftanklexsABG5 = Np8xsleftanklenormal001ab(:,5)- MfANKLEG5AB;
CMCLeftanklexsABG5 = CMCLeftanklexsABG5.^2;
CMCLeftanklexsABG5 = sum(CMCLeftanklexsABG5);
CMCLeftanklexsABG5 = CMCLeftanklexsABG5/505;

CMCLeftankleABG1 = CMCLeftankleviABG1 + CMCLeftanklexsABG1;
CMCLeftankleABG2 = CMCLeftankleviABG2 + CMCLeftanklexsABG2;
CMCLeftankleABG3 = CMCLeftankleviABG3 + CMCLeftanklexsABG3;
CMCLeftankleABG4 = CMCLeftankleviABG4 + CMCLeftanklexsABG4;
CMCLeftankleABG5 = CMCLeftankleviABG5 + CMCLeftanklexsABG5;

CMCLeftankleAB1 = CMCLeftankleABG1 + CMCLeftankleABG2 +
CMCLeftankleABG3 + CMCLeftankleABG4 + CMCLeftankleABG5;

CMC2LeftankleviABG1 = Np8vileftanklenormal001ab(:,1)-
MgankleABnormal001(1,1);
CMC2LeftankleviABG1 = CMC2LeftankleviABG1.^2;
CMC2LeftankleviABG1 = sum(CMC2LeftankleviABG1);
CMC2LeftankleviABG1 = CMC2LeftankleviABG1/1005;
CMC2LeftanklexsABG1 = Np8xsleftanklenormal001ab(:,1)-
MgankleABnormal001(1,1);
CMC2LeftanklexsABG1 = CMC2LeftanklexsABG1.^2;
CMC2LeftanklexsABG1 = sum(CMC2LeftanklexsABG1);
CMC2LeftanklexsABG1 = CMC2LeftanklexsABG1/1005;
CMC2LeftankleviABG2 = Np8vileftanklenormal001ab(:,2)-
MgankleABnormal001(1,2);
CMC2LeftankleviABG2 = CMC2LeftankleviABG2.^2;
CMC2LeftankleviABG2 = sum(CMC2LeftankleviABG2);
CMC2LeftankleviABG2 = CMC2LeftankleviABG2/1005;
CMC2LeftanklexsABG2 = Np8xsleftanklenormal001ab(:,2)-
MgankleABnormal001(1,2);
CMC2LeftanklexsABG2 = CMC2LeftanklexsABG2.^2;
CMC2LeftanklexsABG2 = sum(CMC2LeftanklexsABG2);
CMC2LeftanklexsABG2 = CMC2LeftanklexsABG2/1005;
CMC2LeftankleviABG3 = Np8vileftanklenormal001ab(:,3)-
MgankleABnormal001(1,3);
CMC2LeftankleviABG3 = CMC2LeftankleviABG3.^2;
CMC2LeftankleviABG3 = sum(CMC2LeftankleviABG3);
CMC2LeftankleviABG3 = CMC2LeftankleviABG3/1005;
CMC2LeftanklexsABG3 = Np8xsleftanklenormal001ab(:,3)-
MgankleABnormal001(1,3);
CMC2LeftanklexsABG3 = CMC2LeftanklexsABG3.^2;
CMC2LeftanklexsABG3 = sum(CMC2LeftanklexsABG3);
CMC2LeftanklexsABG3 = CMC2LeftanklexsABG3/1005;
CMC2LeftankleviABG4 = Np8vileftanklenormal001ab(:,4)-
MgankleABnormal001(1,4);
```

```
CMC2LeftankleviABG4 = CMC2LeftankleviABG4.^2;
CMC2LeftankleviABG4 = sum(CMC2LeftankleviABG4);
CMC2LeftankleviABG4 = CMC2LeftankleviABG4/1005;
CMC2LeftanklexsABG4 = Np8xsleftanklenormal001ab(:,4)-
MgankleABnormal001(1,4);
CMC2LeftanklexsABG4 = CMC2LeftanklexsABG4.^2;
CMC2LeftanklexsABG4 = sum(CMC2LeftanklexsABG4);
CMC2LeftanklexsABG4 = CMC2LeftanklexsABG4/1005;
CMC2LeftankleviABG5 = Np8vileftanklenormal001ab(:,5)-
MgankleABnormal001(1,5);
CMC2LeftankleviABG5 = CMC2LeftankleviABG5.^2;
CMC2LeftankleviABG5 = sum(CMC2LeftankleviABG5);
CMC2LeftankleviABG5 = CMC2LeftankleviABG5/1005;
CMC2LeftanklexsABG5 = Np8xsleftanklenormal001ab(:,5)-
MgankleABnormal001(1,5);
CMC2LeftanklexsABG5 = CMC2LeftanklexsABG5.^2;
CMC2LeftanklexsABG5 = sum(CMC2LeftanklexsABG5);
CMC2LeftanklexsABG5 = CMC2LeftanklexsABG5/1005;

CMC2LeftankleABG1 = CMC2LeftankleviABG1 + CMC2LeftanklexsABG1;
CMC2LeftankleABG2 = CMC2LeftankleviABG2 + CMC2LeftanklexsABG2;
CMC2LeftankleABG3 = CMC2LeftankleviABG3 + CMC2LeftanklexsABG3;
CMC2LeftankleABG4 = CMC2LeftankleviABG4 + CMC2LeftanklexsABG4;
CMC2LeftankleABG5 = CMC2LeftankleviABG5 + CMC2LeftanklexsABG5;


CMC2LeftankleAB1 = CMC2LeftankleABG1 + CMC2LeftankleABG2 +
CMC2LeftankleABG3 + CMC2LeftankleABG4 + CMC2LeftankleABG5;

CMCleftankleabp8normal001 = CMCLeftankleAB1/ CMC2LeftankleAB1;
CMCleftankleabp8normal001 = 1- CMCleftankleabp8normal001;
CMCleftankleabp8normal001 = sqrt(CMCleftankleabp8normal001)

%%%%leftanklein

% the mean angle at frame f (Mf) between angles measured by the 2
systems for the gait cycle g
%Mf = Mf/2;  (2 is P)

MfANKLEG1IN = Np8vileftanklenormal001in(:,1) +
Np8xsleftanklenormal001in(:,1);
MfANKLEG1IN = MfANKLEG1IN/2;
MfANKLEG2IN = Np8vileftanklenormal001in(:,2) +
Np8xsleftanklenormal001in(:,2);
MfANKLEG2IN = MfANKLEG2IN/2;
MfANKLEG3IN = Np8vileftanklenormal001in(:,3) +
Np8xsleftanklenormal001in(:,3);
MfANKLEG3IN = MfANKLEG3IN/2;
MfANKLEG4IN = Np8vileftanklenormal001in(:,4) +
Np8xsleftanklenormal001in(:,4);
MfANKLEG4IN = MfANKLEG4IN/2;
MfANKLEG5IN = Np8vileftanklenormal001in(:,5) +
Np8xsleftanklenormal001in(:,5);
MfANKLEG5IN = MfANKLEG5IN/2;

%% the grand mean (Mg) for the gait cycle "g" among these two
methods

MgankleINnormal001 = sum(Np8vileftanklenormal001in)+
sum(Np8xsleftanklenormal001in);
MgankleINnormal001 = MgankleINnormal001/202; % devided by 1/2F;
```

```
CMCLeftankleviING1 = Np8vileftanklenormal001in(:,1)- MfANKLEG1IN;
CMCLeftankleviING1 = CMCLeftankleviING1.^2;
CMCLeftankleviING1 = sum(CMCLeftankleviING1);
CMCLeftankleviING1 = CMCLeftankleviING1/505;
CMCLeftanklexsING1 = Np8xsleftanklenormal001in(:,1)- MfANKLEG1IN;
CMCLeftanklexsING1 = CMCLeftanklexsING1.^2;
CMCLeftanklexsING1 = sum(CMCLeftanklexsING1);
CMCLeftanklexsING1 = CMCLeftanklexsING1/505;
CMCLeftankleviING2 = Np8vileftanklenormal001in(:,2)- MfANKLEG2IN;
CMCLeftankleviING2 = CMCLeftankleviING2.^2;
CMCLeftankleviING2 = sum(CMCLeftankleviING2);
CMCLeftankleviING2 = CMCLeftankleviING2/505;
CMCLeftanklexsING2 = Np8xsleftanklenormal001in(:,2)- MfANKLEG2IN;
CMCLeftanklexsING2 = CMCLeftanklexsING2.^2;
CMCLeftanklexsING2 = sum(CMCLeftanklexsING2);
CMCLeftanklexsING2 = CMCLeftanklexsING2/505;
CMCLeftankleviING3 = Np8vileftanklenormal001in(:,3)- MfANKLEG3IN;
CMCLeftankleviING3 = CMCLeftankleviING3.^2;
CMCLeftankleviING3 = sum(CMCLeftankleviING3);
CMCLeftankleviING3 = CMCLeftankleviING3/505;
CMCLeftanklexsING3 = Np8xsleftanklenormal001in(:,3)- MfANKLEG3IN;
CMCLeftanklexsING3 = CMCLeftanklexsING3.^2;
CMCLeftanklexsING3 = sum(CMCLeftanklexsING3);
CMCLeftanklexsING3 = CMCLeftanklexsING3/505;
CMCLeftankleviING4 = Np8vileftanklenormal001in(:,4)- MfANKLEG4IN;
CMCLeftankleviING4 = CMCLeftankleviING4.^2;
CMCLeftankleviING4 = sum(CMCLeftankleviING4);
CMCLeftankleviING4 = CMCLeftankleviING4/505;
CMCLeftanklexsING4 = Np8xsleftanklenormal001in(:,4)- MfANKLEG4IN;
CMCLeftanklexsING4 = CMCLeftanklexsING4.^2;
CMCLeftanklexsING4 = sum(CMCLeftanklexsING4);
CMCLeftanklexsING4 = CMCLeftanklexsING4/505;
CMCLeftankleviING5 = Np8vileftanklenormal001in(:,5)- MfANKLEG5IN;
CMCLeftankleviING5 = CMCLeftankleviING5.^2;
CMCLeftankleviING5 = sum(CMCLeftankleviING5);
CMCLeftankleviING5 = CMCLeftankleviING5/505;
CMCLeftanklexsING5 = Np8xsleftanklenormal001in(:,5)- MfANKLEG5IN;
CMCLeftanklexsING5 = CMCLeftanklexsING5.^2;
CMCLeftanklexsING5 = sum(CMCLeftanklexsING5);
CMCLeftanklexsING5 = CMCLeftanklexsING5/505;

CMCLeftankleING1 = CMCLeftankleviING1 + CMCLeftanklexsING1;
CMCLeftankleING2 = CMCLeftankleviING2 + CMCLeftanklexsING2;
CMCLeftankleING3 = CMCLeftankleviING3 + CMCLeftanklexsING3;
CMCLeftankleING4 = CMCLeftankleviING4 + CMCLeftanklexsING4;
CMCLeftankleING5 = CMCLeftankleviING5 + CMCLeftanklexsING5;

CMCLeftankleIN1 = CMCLeftankleING1 + CMCLeftankleING2 +
CMCLeftankleING3 + CMCLeftankleING4 + CMCLeftankleING5;

CMC2LeftankleviING1 = Np8vileftanklenormal001in(:,1)-
MgankleINnormal001(1,1);
CMC2LeftankleviING1 = CMC2LeftankleviING1.^2;
CMC2LeftankleviING1 = sum(CMC2LeftankleviING1);
CMC2LeftankleviING1 = CMC2LeftankleviING1/1005;
CMC2LeftanklexsING1 = Np8xsleftanklenormal001in(:,1)-
MgankleINnormal001(1,1);
CMC2LeftanklexsING1 = CMC2LeftanklexsING1.^2;
CMC2LeftanklexsING1 = sum(CMC2LeftanklexsING1);
CMC2LeftanklexsING1 = CMC2LeftanklexsING1/1005;
```

```
CMC2LeftankleviING2 = Np8vileftanklenormal001in(:,2)-
MgankleINnormal001(1,2);
CMC2LeftankleviING2 = CMC2LeftankleviING2.^2;
CMC2LeftankleviING2 = sum(CMC2LeftankleviING2);
CMC2LeftankleviING2 = CMC2LeftankleviING2/1005;
CMC2LeftanklexsING2 = Np8xsleftanklenormal001in(:,2)-
MgankleINnormal001(1,2);
CMC2LeftanklexsING2 = CMC2LeftanklexsING2.^2;
CMC2LeftanklexsING2 = sum(CMC2LeftanklexsING2);
CMC2LeftanklexsING2 = CMC2LeftanklexsING2/1005;
CMC2LeftankleviING3 = Np8vileftanklenormal001in(:,3)-
MgankleINnormal001(1,3);
CMC2LeftankleviING3 = CMC2LeftankleviING3.^2;
CMC2LeftankleviING3 = sum(CMC2LeftankleviING3);
CMC2LeftankleviING3 = CMC2LeftankleviING3/1005;
CMC2LeftanklexsING3 = Np8xsleftanklenormal001in(:,3)-
MgankleINnormal001(1,3);
CMC2LeftanklexsING3 = CMC2LeftanklexsING3.^2;
CMC2LeftanklexsING3 = sum(CMC2LeftanklexsING3);
CMC2LeftanklexsING3 = CMC2LeftanklexsING3/1005;
CMC2LeftankleviING4 = Np8vileftanklenormal001in(:,4)-
MgankleINnormal001(1,4);
CMC2LeftankleviING4 = CMC2LeftankleviING4.^2;
CMC2LeftankleviING4 = sum(CMC2LeftankleviING4);
CMC2LeftankleviING4 = CMC2LeftankleviING4/1005;
CMC2LeftanklexsING4 = Np8xsleftanklenormal001in(:,4)-
MgankleINnormal001(1,4);
CMC2LeftanklexsING4 = CMC2LeftanklexsING4.^2;
CMC2LeftanklexsING4 = sum(CMC2LeftanklexsING4);
CMC2LeftanklexsING4 = CMC2LeftanklexsING4/1005;
CMC2LeftankleviING5 = Np8vileftanklenormal001in(:,5)-
MgankleINnormal001(1,5);
CMC2LeftankleviING5 = CMC2LeftankleviING5.^2;
CMC2LeftankleviING5 = sum(CMC2LeftankleviING5);
CMC2LeftankleviING5 = CMC2LeftankleviING5/1005;
CMC2LeftanklexsING5 = Np8xsleftanklenormal001in(:,5)-
MgankleINnormal001(1,5);
CMC2LeftanklexsING5 = CMC2LeftanklexsING5.^2;
CMC2LeftanklexsING5 = sum(CMC2LeftanklexsING5);
CMC2LeftanklexsING5 = CMC2LeftanklexsING5/1005;

CMC2LeftankleING1 = CMC2LeftankleviING1 + CMC2LeftanklexsING1;
CMC2LeftankleING2 = CMC2LeftankleviING2 + CMC2LeftanklexsING2;
CMC2LeftankleING3 = CMC2LeftankleviING3 + CMC2LeftanklexsING3;
CMC2LeftankleING4 = CMC2LeftankleviING4 + CMC2LeftanklexsING4;
CMC2LeftankleING5 = CMC2LeftankleviING5 + CMC2LeftanklexsING5;


CMC2LeftankleIN1 = CMC2LeftankleING1 + CMC2LeftankleING2 +
CMC2LeftankleING3 + CMC2LeftankleING4 + CMC2LeftankleING5;

CMCleftankleinp8normal001 = CMCLeftankleIN1/ CMC2LeftankleIN1;
CMCleftankleinp8normal001 = 1- CMCleftankleinp8normal001;
CMCleftankleinp8normal001 = sqrt(CMCleftankleinp8normal001)

%%%%end of the calculation

%%%hip
p8vilefthipnormal001in_f = p8vilefthipnormal001in_f *-1;

%%%the mean joint estimation error (MJEHIP or ?e)
% Left hip
```

```
MJEHIPG1ab = Np8vilefthipnormal001ab(:,1) -
Np8xslefthipnormal001ab(:,1);
MJEHIPG1ab = abs(MJEHIPG1ab);
MJEHIPG1ab = sum(MJEHIPG1ab);
MJEHIPG1ab = MJEHIPG1ab *1/101;
MJEHIPG2ab = Np8vilefthipnormal001ab(:,2) -
Np8xslefthipnormal001ab(:,2);
MJEHIPG2ab = abs(MJEHIPG2ab);
MJEHIPG2ab = sum(MJEHIPG2ab);
MJEHIPG2ab = MJEHIPG2ab *1/101;
MJEHIPG3ab = Np8vilefthipnormal001ab(:,3) -
Np8xslefthipnormal001ab(:,3);
MJEHIPG3ab = abs(MJEHIPG3ab);
MJEHIPG3ab = sum(MJEHIPG3ab);
MJEHIPG3ab = MJEHIPG3ab *1/101;
MJEHIPG4ab = Np8vilefthipnormal001ab(:,4) -
Np8xslefthipnormal001ab(:,4);
MJEHIPG4ab = abs(MJEHIPG4ab);
MJEHIPG4ab = sum(MJEHIPG4ab);
MJEHIPG4ab = MJEHIPG4ab *1/101;
MJEHIPG5ab = Np8vilefthipnormal001ab(:,5) -
Np8xslefthipnormal001ab(:,5);
MJEHIPG5ab = abs(MJEHIPG5ab);
MJEHIPG5ab = sum(MJEHIPG5ab);
MJEHIPG5ab = MJEHIPG5ab *1/101;
MJEHIPG1fl = Np8vilefthipnormal001fl(:,1) -
Np8xslefthipnormal001fl(:,1);
MJEHIPG1fl = abs(MJEHIPG1fl);
MJEHIPG1fl = sum(MJEHIPG1fl);
MJEHIPG1fl = MJEHIPG1fl *1/101;
MJEHIPG2fl = Np8vilefthipnormal001fl(:,2) -
Np8xslefthipnormal001fl(:,2);
MJEHIPG2fl = abs(MJEHIPG2fl);
MJEHIPG2fl = sum(MJEHIPG2fl);
MJEHIPG2fl = MJEHIPG2fl *1/101;
MJEHIPG3fl = Np8vilefthipnormal001fl(:,3) -
Np8xslefthipnormal001fl(:,3);
MJEHIPG3fl = abs(MJEHIPG3fl);
MJEHIPG3fl = sum(MJEHIPG3fl);
MJEHIPG3fl = MJEHIPG3fl *1/101;
MJEHIPG4fl = Np8vilefthipnormal001fl(:,4) -
Np8xslefthipnormal001fl(:,4);
MJEHIPG4fl = abs(MJEHIPG4fl);
MJEHIPG4fl = sum(MJEHIPG4fl);
MJEHIPG4fl = MJEHIPG4fl *1/101;
MJEHIPG5fl = Np8vilefthipnormal001fl(:,5) -
Np8xslefthipnormal001fl(:,5);
MJEHIPG5fl = abs(MJEHIPG5fl);
MJEHIPG5fl = sum(MJEHIPG5fl);
MJEHIPG5fl = MJEHIPG5fl *1/101;
MJEHIPG1in = Np8vilefthipnormal001in(:,1) -
Np8xslefthipnormal001in(:,1);
MJEHIPG1in = abs(MJEHIPG1in);
MJEHIPG1in = sum(MJEHIPG1in);
MJEHIPG1in = MJEHIPG1in *1/101;
MJEHIPG2in = Np8vilefthipnormal001in(:,2) -
Np8xslefthipnormal001in(:,2);
MJEHIPG2in = abs(MJEHIPG2in);
MJEHIPG2in = sum(MJEHIPG2in);
MJEHIPG2in = MJEHIPG2in *1/101;
MJEHIPG3in = Np8vilefthipnormal001in(:,3) -
Np8xslefthipnormal001in(:,3);
MJEHIPG3in = abs(MJEHIPG3in);
```

```
MJEHIPG3in = sum(MJEHIPG3in);
MJEHIPG3in = MJEHIPG3in *1/101;
MJEHIPG4in = Np8vilefthipnormal001in(:,4) -
Np8xslefthipnormal001in(:,4);
MJEHIPG4in = abs(MJEHIPG4in);
MJEHIPG4in = sum(MJEHIPG4in);
MJEHIPG4in = MJEHIPG4in *1/101;
MJEHIPG5in = Np8vilefthipnormal001in(:,5) -
Np8xslefthipnormal001in(:,5);
MJEHIPG5in = abs(MJEHIPG5in);
MJEHIPG5in = sum(MJEHIPG5in);
MJEHIPG5in = MJEHIPG5in *1/101;

%%CMC
% ? is an angle at a particular captured frame
% ?vf is an angle from vicon at a particular frame
% ?xf is an angle from xsens at a particular frame
% f = a captured frame
% F = 101 (the total captured frames per cycle)
% P = 2 (number of motion capturing methods/ 1= vicon, 1= xsens)
% G = 5 (number of gait cycles)

%GFg(P-1) =   5*101(2-1) = 505
%GFg(P-1) =   4*101(2-1) = 404  %% p17 slow speed only
%G(PFg-1) = 5(2*101-1) = 1005
%G(PFg-1) = 4(2*101-1) = 804%% p17 slow speed only

% the mean angle at frame f (Mf) between angles measured by the 2
systems for the gait cycle g
%Mf = Mf/2;  (2 is P)

%%%%lefthipfl
MfHIPG1FL = Np8vilefthipnormal001fl(:,1) +
Np8xslefthipnormal001fl(:,1);
MfHIPG1FL = MfHIPG1FL/2;
MfHIPG2FL = Np8vilefthipnormal001fl(:,2) +
Np8xslefthipnormal001fl(:,2);
MfHIPG2FL = MfHIPG2FL/2;
MfHIPG3FL = Np8vilefthipnormal001fl(:,3) +
Np8xslefthipnormal001fl(:,3);
MfHIPG3FL = MfHIPG3FL/2;
MfHIPG4FL = Np8vilefthipnormal001fl(:,4) +
Np8xslefthipnormal001fl(:,4);
MfHIPG4FL = MfHIPG4FL/2;
MfHIPG5FL = Np8vilefthipnormal001fl(:,5) +
Np8xslefthipnormal001fl(:,5);
MfHIPG5FL = MfHIPG5FL/2;

%% the grand mean (Mg) for the gait cycle "g" among these two
methods

MghipFLnormal001 = sum(Np8vilefthipnormal001fl)+
sum(Np8xslefthipnormal001fl);
MghipFLnormal001 = MghipFLnormal001/202; % devided by 1/2F;

CMCLefthipviFLG1 = Np8vilefthipnormal001fl(:,1)- MfHIPG1FL;
CMCLefthipviFLG1 = CMCLefthipviFLG1.^2;
CMCLefthipviFLG1 = sum(CMCLefthipviFLG1);
CMCLefthipviFLG1 = CMCLefthipviFLG1/505;
CMCLefthipxsFLG1 = Np8xslefthipnormal001fl(:,1)- MfHIPG1FL;
CMCLefthipxsFLG1 = CMCLefthipxsFLG1.^2;
CMCLefthipxsFLG1 = sum(CMCLefthipxsFLG1);
```

```
CMCLefthipxsFLG1 = CMCLefthipxsFLG1/505;
CMCLefthipviFLG2 = Np8vilefthipnormal001fl(:,2)- MfHIPG2FL;
CMCLefthipviFLG2 = CMCLefthipviFLG2.^2;
CMCLefthipviFLG2 = sum(CMCLefthipviFLG2);
CMCLefthipviFLG2 = CMCLefthipviFLG2/505;
CMCLefthipxsFLG2 = Np8xslefthipnormal001fl(:,2)- MfHIPG2FL;
CMCLefthipxsFLG2 = CMCLefthipxsFLG2.^2;
CMCLefthipxsFLG2 = sum(CMCLefthipxsFLG2);
CMCLefthipxsFLG2 = CMCLefthipxsFLG2/505;
CMCLefthipviFLG3 = Np8vilefthipnormal001fl(:,3)- MfHIPG3FL;
CMCLefthipviFLG3 = CMCLefthipviFLG3.^2;
CMCLefthipviFLG3 = sum(CMCLefthipviFLG3);
CMCLefthipviFLG3 = CMCLefthipviFLG3/505;
CMCLefthipxsFLG3 = Np8xslefthipnormal001fl(:,3)- MfHIPG3FL;
CMCLefthipxsFLG3 = CMCLefthipxsFLG3.^2;
CMCLefthipxsFLG3 = sum(CMCLefthipxsFLG3);
CMCLefthipxsFLG3 = CMCLefthipxsFLG3/505;
CMCLefthipviFLG4 = Np8vilefthipnormal001fl(:,4)- MfHIPG4FL;
CMCLefthipviFLG4 = CMCLefthipviFLG4.^2;
CMCLefthipviFLG4 = sum(CMCLefthipviFLG4);
CMCLefthipviFLG4 = CMCLefthipviFLG4/505;
CMCLefthipxsFLG4 = Np8xslefthipnormal001fl(:,4)- MfHIPG4FL;
CMCLefthipxsFLG4 = CMCLefthipxsFLG4.^2;
CMCLefthipxsFLG4 = sum(CMCLefthipxsFLG4);
CMCLefthipxsFLG4 = CMCLefthipxsFLG4/505;
CMCLefthipviFLG5 = Np8vilefthipnormal001fl(:,5)- MfHIPG5FL;
CMCLefthipviFLG5 = CMCLefthipviFLG5.^2;
CMCLefthipviFLG5 = sum(CMCLefthipviFLG5);
CMCLefthipviFLG5 = CMCLefthipviFLG5/505;
CMCLefthipxsFLG5 = Np8xslefthipnormal001fl(:,5)- MfHIPG5FL;
CMCLefthipxsFLG5 = CMCLefthipxsFLG5.^2;
CMCLefthipxsFLG5 = sum(CMCLefthipxsFLG5);
CMCLefthipxsFLG5 = CMCLefthipxsFLG5/505;

CMCLefthipFLG1 = CMCLefthipviFLG1 + CMCLefthipxsFLG1;
CMCLefthipFLG2 = CMCLefthipviFLG2 + CMCLefthipxsFLG2;
CMCLefthipFLG3 = CMCLefthipviFLG3 + CMCLefthipxsFLG3;
CMCLefthipFLG4 = CMCLefthipviFLG4 + CMCLefthipxsFLG4;
CMCLefthipFLG5 = CMCLefthipviFLG5 + CMCLefthipxsFLG5;

CMCLefthipFL1 = CMCLefthipFLG1 + CMCLefthipFLG2 + CMCLefthipFLG3 +
CMCLefthipFLG4 + CMCLefthipFLG5;

CMC2LefthipviFLG1 = Np8vilefthipnormal001fl(:,1)-
MghipFLnormal001(1,1);
CMC2LefthipviFLG1 = CMC2LefthipviFLG1.^2;
CMC2LefthipviFLG1 = sum(CMC2LefthipviFLG1);
CMC2LefthipviFLG1 = CMC2LefthipviFLG1/1005;
CMC2LefthipxsFLG1 = Np8xslefthipnormal001fl(:,1)-
MghipFLnormal001(1,1);
CMC2LefthipxsFLG1 = CMC2LefthipxsFLG1.^2;
CMC2LefthipxsFLG1 = sum(CMC2LefthipxsFLG1);
CMC2LefthipxsFLG1 = CMC2LefthipxsFLG1/1005;
CMC2LefthipviFLG2 = Np8vilefthipnormal001fl(:,2)-
MghipFLnormal001(1,2);
CMC2LefthipviFLG2 = CMC2LefthipviFLG2.^2;
CMC2LefthipviFLG2 = sum(CMC2LefthipviFLG2);
CMC2LefthipviFLG2 = CMC2LefthipviFLG2/1005;
CMC2LefthipxsFLG2 = Np8xslefthipnormal001fl(:,2)-
MghipFLnormal001(1,2);
CMC2LefthipxsFLG2 = CMC2LefthipxsFLG2.^2;
CMC2LefthipxsFLG2 = sum(CMC2LefthipxsFLG2);
```

```
CMC2LefthipxsFLG2 = CMC2LefthipxsFLG2/1005;
CMC2LefthipviFLG3 = Np8vilefthipnormal001fl(:,3)-
MghipFLnormal001(1,3);
CMC2LefthipviFLG3 = CMC2LefthipviFLG3.^2;
CMC2LefthipviFLG3 = sum(CMC2LefthipviFLG3);
CMC2LefthipviFLG3 = CMC2LefthipviFLG3/1005;
CMC2LefthipxsFLG3 = Np8xslefthipnormal001fl(:,3)-
MghipFLnormal001(1,3);
CMC2LefthipxsFLG3 = CMC2LefthipxsFLG3.^2;
CMC2LefthipxsFLG3 = sum(CMC2LefthipxsFLG3);
CMC2LefthipxsFLG3 = CMC2LefthipxsFLG3/1005;
CMC2LefthipviFLG4 = Np8vilefthipnormal001fl(:,4)-
MghipFLnormal001(1,4);
CMC2LefthipviFLG4 = CMC2LefthipviFLG4.^2;
CMC2LefthipviFLG4 = sum(CMC2LefthipviFLG4);
CMC2LefthipviFLG4 = CMC2LefthipviFLG4/1005;
CMC2LefthipxsFLG4 = Np8xslefthipnormal001fl(:,4)-
MghipFLnormal001(1,4);
CMC2LefthipxsFLG4 = CMC2LefthipxsFLG4.^2;
CMC2LefthipxsFLG4 = sum(CMC2LefthipxsFLG4);
CMC2LefthipxsFLG4 = CMC2LefthipxsFLG4/1005;
CMC2LefthipviFLG5 = Np8vilefthipnormal001fl(:,5)-
MghipFLnormal001(1,5);
CMC2LefthipviFLG5 = CMC2LefthipviFLG5.^2;
CMC2LefthipviFLG5 = sum(CMC2LefthipviFLG5);
CMC2LefthipviFLG5 = CMC2LefthipviFLG5/1005;
CMC2LefthipxsFLG5 = Np8xslefthipnormal001fl(:,5)-
MghipFLnormal001(1,5);
CMC2LefthipxsFLG5 = CMC2LefthipxsFLG5.^2;
CMC2LefthipxsFLG5 = sum(CMC2LefthipxsFLG5);
CMC2LefthipxsFLG5 = CMC2LefthipxsFLG5/1005;

CMC2LefthipFLG1 = CMC2LefthipviFLG1 + CMC2LefthipxsFLG1;
CMC2LefthipFLG2 = CMC2LefthipviFLG2 + CMC2LefthipxsFLG2;
CMC2LefthipFLG3 = CMC2LefthipviFLG3 + CMC2LefthipxsFLG3;
CMC2LefthipFLG4 = CMC2LefthipviFLG4 + CMC2LefthipxsFLG4;
CMC2LefthipFLG5 = CMC2LefthipviFLG5 + CMC2LefthipxsFLG5;


CMC2LefthipFL1 = CMC2LefthipFLG1 + CMC2LefthipFLG2 + CMC2LefthipFLG3
+ CMC2LefthipFLG4 + CMC2LefthipFLG5;

CMClefthipflp8normal001 = CMCLefthipFL1/ CMC2LefthipFL1;
CMClefthipflp8normal001 = 1- CMClefthipflp8normal001;
CMClefthipflp8normal001 = sqrt(CMClefthipflp8normal001)


%%%%lefthipab

% the mean angle at frame f (Mf) between angles measured by the 2
systems for the gait cycle g
%Mf = Mf/2;  (2 is P)

MfHIPG1AB = Np8vilefthipnormal001ab(:,1) +
Np8xslefthipnormal001ab(:,1);
MfHIPG1AB = MfHIPG1AB/2;
MfHIPG2AB = Np8vilefthipnormal001ab(:,2) +
Np8xslefthipnormal001ab(:,2);
MfHIPG2AB = MfHIPG2AB/2;
MfHIPG3AB = Np8vilefthipnormal001ab(:,3) +
Np8xslefthipnormal001ab(:,3);
```

```
MfHIPG3AB = MfHIPG3AB/2;
MfHIPG4AB = Np8vilefthipnormal001ab(:,4) +
Np8xslefthipnormal001ab(:,4);
MfHIPG4AB = MfHIPG4AB/2;
MfHIPG5AB = Np8vilefthipnormal001ab(:,5) +
Np8xslefthipnormal001ab(:,5);
MfHIPG5AB = MfHIPG5AB/2;

%% the grand mean (Mg) for the gait cycle "g" among these two
methods

MghipABnormal001 = sum(Np8vilefthipnormal001ab)+
sum(Np8xslefthipnormal001ab);
MghipABnormal001 = MghipABnormal001/202; % devided by 1/2F;

CMCLefthipviABG1 = Np8vilefthipnormal001ab(:,1)- MfHIPG1AB;
CMCLefthipviABG1 = CMCLefthipviABG1.^2;
CMCLefthipviABG1 = sum(CMCLefthipviABG1);
CMCLefthipviABG1 = CMCLefthipviABG1/505;
CMCLefthipxsABG1 = Np8xslefthipnormal001ab(:,1)- MfHIPG1AB;
CMCLefthipxsABG1 = CMCLefthipxsABG1.^2;
CMCLefthipxsABG1 = sum(CMCLefthipxsABG1);
CMCLefthipxsABG1 = CMCLefthipxsABG1/505;
CMCLefthipviABG2 = Np8vilefthipnormal001ab(:,2)- MfHIPG2AB;
CMCLefthipviABG2 = CMCLefthipviABG2.^2;
CMCLefthipviABG2 = sum(CMCLefthipviABG2);
CMCLefthipviABG2 = CMCLefthipviABG2/505;
CMCLefthipxsABG2 = Np8xslefthipnormal001ab(:,2)- MfHIPG2AB;
CMCLefthipxsABG2 = CMCLefthipxsABG2.^2;
CMCLefthipxsABG2 = sum(CMCLefthipxsABG2);
CMCLefthipxsABG2 = CMCLefthipxsABG2/505;
CMCLefthipviABG3 = Np8vilefthipnormal001ab(:,3)- MfHIPG3AB;
CMCLefthipviABG3 = CMCLefthipviABG3.^2;
CMCLefthipviABG3 = sum(CMCLefthipviABG3);
CMCLefthipviABG3 = CMCLefthipviABG3/505;
CMCLefthipxsABG3 = Np8xslefthipnormal001ab(:,3)- MfHIPG3AB;
CMCLefthipxsABG3 = CMCLefthipxsABG3.^2;
CMCLefthipxsABG3 = sum(CMCLefthipxsABG3);
CMCLefthipxsABG3 = CMCLefthipxsABG3/505;
CMCLefthipviABG4 = Np8vilefthipnormal001ab(:,4)- MfHIPG4AB;
CMCLefthipviABG4 = CMCLefthipviABG4.^2;
CMCLefthipviABG4 = sum(CMCLefthipviABG4);
CMCLefthipviABG4 = CMCLefthipviABG4/505;
CMCLefthipxsABG4 = Np8xslefthipnormal001ab(:,4)- MfHIPG4AB;
CMCLefthipxsABG4 = CMCLefthipxsABG4.^2;
CMCLefthipxsABG4 = sum(CMCLefthipxsABG4);
CMCLefthipxsABG4 = CMCLefthipxsABG4/505;
CMCLefthipviABG5 = Np8vilefthipnormal001ab(:,5)- MfHIPG5AB;
CMCLefthipviABG5 = CMCLefthipviABG5.^2;
CMCLefthipviABG5 = sum(CMCLefthipviABG5);
CMCLefthipviABG5 = CMCLefthipviABG5/505;
CMCLefthipxsABG5 = Np8xslefthipnormal001ab(:,5)- MfHIPG5AB;
CMCLefthipxsABG5 = CMCLefthipxsABG5.^2;
CMCLefthipxsABG5 = sum(CMCLefthipxsABG5);
CMCLefthipxsABG5 = CMCLefthipxsABG5/505;

CMCLefthipABG1 = CMCLefthipviABG1 + CMCLefthipxsABG1;
CMCLefthipABG2 = CMCLefthipviABG2 + CMCLefthipxsABG2;
CMCLefthipABG3 = CMCLefthipviABG3 + CMCLefthipxsABG3;
CMCLefthipABG4 = CMCLefthipviABG4 + CMCLefthipxsABG4;
CMCLefthipABG5 = CMCLefthipviABG5 + CMCLefthipxsABG5;
```

```
CMCLefthipAB1 = CMCLefthipABG1 + CMCLefthipABG2 + CMCLefthipABG3 +
CMCLefthipABG4 + CMCLefthipABG5;

CMC2LefthipviABG1 = Np8vilefthipnormal001ab(:,1)-
MghipABnormal001(1,1);
CMC2LefthipviABG1 = CMC2LefthipviABG1.^2;
CMC2LefthipviABG1 = sum(CMC2LefthipviABG1);
CMC2LefthipviABG1 = CMC2LefthipviABG1/1005;
CMC2LefthipxsABG1 = Np8xslefthipnormal001ab(:,1)-
MghipABnormal001(1,1);
CMC2LefthipxsABG1 = CMC2LefthipxsABG1.^2;
CMC2LefthipxsABG1 = sum(CMC2LefthipxsABG1);
CMC2LefthipxsABG1 = CMC2LefthipxsABG1/1005;
CMC2LefthipviABG2 = Np8vilefthipnormal001ab(:,2)-
MghipABnormal001(1,2);
CMC2LefthipviABG2 = CMC2LefthipviABG2.^2;
CMC2LefthipviABG2 = sum(CMC2LefthipviABG2);
CMC2LefthipviABG2 = CMC2LefthipviABG2/1005;
CMC2LefthipxsABG2 = Np8xslefthipnormal001ab(:,2)-
MghipABnormal001(1,2);
CMC2LefthipxsABG2 = CMC2LefthipxsABG2.^2;
CMC2LefthipxsABG2 = sum(CMC2LefthipxsABG2);
CMC2LefthipxsABG2 = CMC2LefthipxsABG2/1005;
CMC2LefthipviABG3 = Np8vilefthipnormal001ab(:,3)-
MghipABnormal001(1,3);
CMC2LefthipviABG3 = CMC2LefthipviABG3.^2;
CMC2LefthipviABG3 = sum(CMC2LefthipviABG3);
CMC2LefthipviABG3 = CMC2LefthipviABG3/1005;
CMC2LefthipxsABG3 = Np8xslefthipnormal001ab(:,3)-
MghipABnormal001(1,3);
CMC2LefthipxsABG3 = CMC2LefthipxsABG3.^2;
CMC2LefthipxsABG3 = sum(CMC2LefthipxsABG3);
CMC2LefthipxsABG3 = CMC2LefthipxsABG3/1005;
CMC2LefthipviABG4 = Np8vilefthipnormal001ab(:,4)-
MghipABnormal001(1,4);
CMC2LefthipviABG4 = CMC2LefthipviABG4.^2;
CMC2LefthipviABG4 = sum(CMC2LefthipviABG4);
CMC2LefthipviABG4 = CMC2LefthipviABG4/1005;
CMC2LefthipxsABG4 = Np8xslefthipnormal001ab(:,4)-
MghipABnormal001(1,4);
CMC2LefthipxsABG4 = CMC2LefthipxsABG4.^2;
CMC2LefthipxsABG4 = sum(CMC2LefthipxsABG4);
CMC2LefthipxsABG4 = CMC2LefthipxsABG4/1005;
CMC2LefthipviABG5 = Np8vilefthipnormal001ab(:,5)-
MghipABnormal001(1,5);
CMC2LefthipviABG5 = CMC2LefthipviABG5.^2;
CMC2LefthipviABG5 = sum(CMC2LefthipviABG5);
CMC2LefthipviABG5 = CMC2LefthipviABG5/1005;
CMC2LefthipxsABG5 = Np8xslefthipnormal001ab(:,5)-
MghipABnormal001(1,5);
CMC2LefthipxsABG5 = CMC2LefthipxsABG5.^2;
CMC2LefthipxsABG5 = sum(CMC2LefthipxsABG5);
CMC2LefthipxsABG5 = CMC2LefthipxsABG5/1005;

CMC2LefthipABG1 = CMC2LefthipviABG1 + CMC2LefthipxsABG1;
CMC2LefthipABG2 = CMC2LefthipviABG2 + CMC2LefthipxsABG2;
CMC2LefthipABG3 = CMC2LefthipviABG3 + CMC2LefthipxsABG3;
CMC2LefthipABG4 = CMC2LefthipviABG4 + CMC2LefthipxsABG4;
CMC2LefthipABG5 = CMC2LefthipviABG5 + CMC2LefthipxsABG5;
```

```
CMC2LefthipAB1 = CMC2LefthipABG1 + CMC2LefthipABG2 + CMC2LefthipABG3
+ CMC2LefthipABG4 + CMC2LefthipABG5;

CMClefthipabp8normal001 = CMCLefthipAB1/ CMC2LefthipAB1;
CMClefthipabp8normal001 = 1- CMClefthipabp8normal001;
CMClefthipabp8normal001 = sqrt(CMClefthipabp8normal001)

%%%%lefthipin

% the mean angle at frame f (Mf) between angles measured by the 2
systems for the gait cycle g
%Mf = Mf/2;  (2 is P)

MfHIPG1IN = Np8vilefthipnormal001in(:,1) +
Np8xslefthipnormal001in(:,1);
MfHIPG1IN = MfHIPG1IN/2;
MfHIPG2IN = Np8vilefthipnormal001in(:,2) +
Np8xslefthipnormal001in(:,2);
MfHIPG2IN = MfHIPG2IN/2;
MfHIPG3IN = Np8vilefthipnormal001in(:,3) +
Np8xslefthipnormal001in(:,3);
MfHIPG3IN = MfHIPG3IN/2;
MfHIPG4IN = Np8vilefthipnormal001in(:,4) +
Np8xslefthipnormal001in(:,4);
MfHIPG4IN = MfHIPG4IN/2;
MfHIPG5IN = Np8vilefthipnormal001in(:,5) +
Np8xslefthipnormal001in(:,5);
MfHIPG5IN = MfHIPG5IN/2;

%% the grand mean (Mg) for the gait cycle "g" among these two
methods

MghipINnormal001 = sum(Np8vilefthipnormal001in)+
sum(Np8xslefthipnormal001in);
MghipINnormal001 = MghipINnormal001/202; % devided by 1/2F;

CMCLefthipviING1 = Np8vilefthipnormal001in(:,1)- MfHIPG1IN;
CMCLefthipviING1 = CMCLefthipviING1.^2;
CMCLefthipviING1 = sum(CMCLefthipviING1);
CMCLefthipviING1 = CMCLefthipviING1/505;
CMCLefthipxsING1 = Np8xslefthipnormal001in(:,1)- MfHIPG1IN;
CMCLefthipxsING1 = CMCLefthipxsING1.^2;
CMCLefthipxsING1 = sum(CMCLefthipxsING1);
CMCLefthipxsING1 = CMCLefthipxsING1/505;
CMCLefthipviING2 = Np8vilefthipnormal001in(:,2)- MfHIPG2IN;
CMCLefthipviING2 = CMCLefthipviING2.^2;
CMCLefthipviING2 = sum(CMCLefthipviING2);
CMCLefthipviING2 = CMCLefthipviING2/505;
CMCLefthipxsING2 = Np8xslefthipnormal001in(:,2)- MfHIPG2IN;
CMCLefthipxsING2 = CMCLefthipxsING2.^2;
CMCLefthipxsING2 = sum(CMCLefthipxsING2);
CMCLefthipxsING2 = CMCLefthipxsING2/505;
CMCLefthipviING3 = Np8vilefthipnormal001in(:,3)- MfHIPG3IN;
CMCLefthipviING3 = CMCLefthipviING3.^2;
CMCLefthipviING3 = sum(CMCLefthipviING3);
CMCLefthipviING3 = CMCLefthipviING3/505;
CMCLefthipxsING3 = Np8xslefthipnormal001in(:,3)- MfHIPG3IN;
CMCLefthipxsING3 = CMCLefthipxsING3.^2;
CMCLefthipxsING3 = sum(CMCLefthipxsING3);
CMCLefthipxsING3 = CMCLefthipxsING3/505;
CMCLefthipviING4 = Np8vilefthipnormal001in(:,4)- MfHIPG4IN;
CMCLefthipviING4 = CMCLefthipviING4.^2;
```

```
CMCLefthipviING4 = sum(CMCLefthipviING4);
CMCLefthipviING4 = CMCLefthipviING4/505;
CMCLefthipxsING4 = Np8xslefthipnormal001in(:,4)- MfHIPG4IN;
CMCLefthipxsING4 = CMCLefthipxsING4.^2;
CMCLefthipxsING4 = sum(CMCLefthipxsING4);
CMCLefthipxsING4 = CMCLefthipxsING4/505;
CMCLefthipviING5 = Np8vilefthipnormal001in(:,5)- MfHIPG5IN;
CMCLefthipviING5 = CMCLefthipviING5.^2;
CMCLefthipviING5 = sum(CMCLefthipviING5);
CMCLefthipviING5 = CMCLefthipviING5/505;
CMCLefthipxsING5 = Np8xslefthipnormal001in(:,5)- MfHIPG5IN;
CMCLefthipxsING5 = CMCLefthipxsING5.^2;
CMCLefthipxsING5 = sum(CMCLefthipxsING5);
CMCLefthipxsING5 = CMCLefthipxsING5/505;

CMCLefthipING1 = CMCLefthipviING1 + CMCLefthipxsING1;
CMCLefthipING2 = CMCLefthipviING2 + CMCLefthipxsING2;
CMCLefthipING3 = CMCLefthipviING3 + CMCLefthipxsING3;
CMCLefthipING4 = CMCLefthipviING4 + CMCLefthipxsING4;
CMCLefthipING5 = CMCLefthipviING5 + CMCLefthipxsING5;

CMCLefthipIN1 = CMCLefthipING1 + CMCLefthipING2 + CMCLefthipING3 +
CMCLefthipING4 + CMCLefthipING5;

CMC2LefthipviING1 = Np8vilefthipnormal001in(:,1)-
MghipINnormal001(1,1);
CMC2LefthipviING1 = CMC2LefthipviING1.^2;
CMC2LefthipviING1 = sum(CMC2LefthipviING1);
CMC2LefthipviING1 = CMC2LefthipviING1/1005;
CMC2LefthipxsING1 = Np8xslefthipnormal001in(:,1)-
MghipINnormal001(1,1);
CMC2LefthipxsING1 = CMC2LefthipxsING1.^2;
CMC2LefthipxsING1 = sum(CMC2LefthipxsING1);
CMC2LefthipxsING1 = CMC2LefthipxsING1/1005;
CMC2LefthipviING2 = Np8vilefthipnormal001in(:,2)-
MghipINnormal001(1,2);
CMC2LefthipviING2 = CMC2LefthipviING2.^2;
CMC2LefthipviING2 = sum(CMC2LefthipviING2);
CMC2LefthipviING2 = CMC2LefthipviING2/1005;
CMC2LefthipxsING2 = Np8xslefthipnormal001in(:,2)-
MghipINnormal001(1,2);
CMC2LefthipxsING2 = CMC2LefthipxsING2.^2;
CMC2LefthipxsING2 = sum(CMC2LefthipxsING2);
CMC2LefthipxsING2 = CMC2LefthipxsING2/1005;
CMC2LefthipviING3 = Np8vilefthipnormal001in(:,3)-
MghipINnormal001(1,3);
CMC2LefthipviING3 = CMC2LefthipviING3.^2;
CMC2LefthipviING3 = sum(CMC2LefthipviING3);
CMC2LefthipviING3 = CMC2LefthipviING3/1005;
CMC2LefthipxsING3 = Np8xslefthipnormal001in(:,3)-
MghipINnormal001(1,3);
CMC2LefthipxsING3 = CMC2LefthipxsING3.^2;
CMC2LefthipxsING3 = sum(CMC2LefthipxsING3);
CMC2LefthipxsING3 = CMC2LefthipxsING3/1005;
CMC2LefthipviING4 = Np8vilefthipnormal001in(:,4)-
MghipINnormal001(1,4);
CMC2LefthipviING4 = CMC2LefthipviING4.^2;
CMC2LefthipviING4 = sum(CMC2LefthipviING4);
CMC2LefthipviING4 = CMC2LefthipviING4/1005;
CMC2LefthipxsING4 = Np8xslefthipnormal001in(:,4)-
MghipINnormal001(1,4);
CMC2LefthipxsING4 = CMC2LefthipxsING4.^2;
```

```
CMC2LefthipxsING4 = sum(CMC2LefthipxsING4);
CMC2LefthipxsING4 = CMC2LefthipxsING4/1005;
CMC2LefthipviING5 = Np8vilefthipnormal001in(:,5)-
MghipINnormal001(1,5);
CMC2LefthipviING5 = CMC2LefthipviING5.^2;
CMC2LefthipviING5 = sum(CMC2LefthipviING5);
CMC2LefthipviING5 = CMC2LefthipviING5/1005;
CMC2LefthipxsING5 = Np8xslefthipnormal001in(:,5)-
MghipINnormal001(1,5);
CMC2LefthipxsING5 = CMC2LefthipxsING5.^2;
CMC2LefthipxsING5 = sum(CMC2LefthipxsING5);
CMC2LefthipxsING5 = CMC2LefthipxsING5/1005;

CMC2LefthipING1 = CMC2LefthipviING1 + CMC2LefthipxsING1;
CMC2LefthipING2 = CMC2LefthipviING2 + CMC2LefthipxsING2;
CMC2LefthipING3 = CMC2LefthipviING3 + CMC2LefthipxsING3;
CMC2LefthipING4 = CMC2LefthipviING4 + CMC2LefthipxsING4;
CMC2LefthipING5 = CMC2LefthipviING5 + CMC2LefthipxsING5;


CMC2LefthipIN1 = CMC2LefthipING1 + CMC2LefthipING2 + CMC2LefthipING3
+ CMC2LefthipING4 + CMC2LefthipING5;

CMClefthipinp8normal001 = CMCLefthipIN1/ CMC2LefthipIN1;
CMClefthipinp8normal001 = 1- CMClefthipinp8normal001;
CMClefthipinp8normal001 = sqrt(CMClefthipinp8normal001)

%%%%end of the calculation

%%%knee

p8vileftkneenormal001in_f = p8vileftkneenormal001in_f *-1;

%%%the mean joint estimation error (MJEKNEE or ?e)
% Left knee
MJEKNEEG1ab = Np8vileftkneenormal001ab(:,1) -
Np8xsleftkneenormal001ab(:,1);
MJEKNEEG1ab = abs(MJEKNEEG1ab);
MJEKNEEG1ab = sum(MJEKNEEG1ab);
MJEKNEEG1ab = MJEKNEEG1ab *1/101;
MJEKNEEG2ab = Np8vileftkneenormal001ab(:,2) -
Np8xsleftkneenormal001ab(:,2);
MJEKNEEG2ab = abs(MJEKNEEG2ab);
MJEKNEEG2ab = sum(MJEKNEEG2ab);
MJEKNEEG2ab = MJEKNEEG2ab *1/101;
MJEKNEEG3ab = Np8vileftkneenormal001ab(:,3) -
Np8xsleftkneenormal001ab(:,3);
MJEKNEEG3ab = abs(MJEKNEEG3ab);
MJEKNEEG3ab = sum(MJEKNEEG3ab);
MJEKNEEG3ab = MJEKNEEG3ab *1/101;
MJEKNEEG4ab = Np8vileftkneenormal001ab(:,4) -
Np8xsleftkneenormal001ab(:,4);
MJEKNEEG4ab = abs(MJEKNEEG4ab);
MJEKNEEG4ab = sum(MJEKNEEG4ab);
MJEKNEEG4ab = MJEKNEEG4ab *1/101;
MJEKNEEG5ab = Np8vileftkneenormal001ab(:,5) -
Np8xsleftkneenormal001ab(:,5);
MJEKNEEG5ab = abs(MJEKNEEG5ab);
MJEKNEEG5ab = sum(MJEKNEEG5ab);
MJEKNEEG5ab = MJEKNEEG5ab *1/101;
```

```
MJEKNEEG1fl = Np8vileftkneenormal001fl(:,1) -
Np8xsleftkneenormal001fl(:,1);
MJEKNEEG1fl = abs(MJEKNEEG1fl);
MJEKNEEG1fl = sum(MJEKNEEG1fl);
MJEKNEEG1fl = MJEKNEEG1fl *1/101;
MJEKNEEG2fl = Np8vileftkneenormal001fl(:,2) -
Np8xsleftkneenormal001fl(:,2);
MJEKNEEG2fl = abs(MJEKNEEG2fl);
MJEKNEEG2fl = sum(MJEKNEEG2fl);
MJEKNEEG2fl = MJEKNEEG2fl *1/101;
MJEKNEEG3fl = Np8vileftkneenormal001fl(:,3) -
Np8xsleftkneenormal001fl(:,3);
MJEKNEEG3fl = abs(MJEKNEEG3fl);
MJEKNEEG3fl = sum(MJEKNEEG3fl);
MJEKNEEG3fl = MJEKNEEG3fl *1/101;
MJEKNEEG4fl = Np8vileftkneenormal001fl(:,4) -
Np8xsleftkneenormal001fl(:,4);
MJEKNEEG4fl = abs(MJEKNEEG4fl);
MJEKNEEG4fl = sum(MJEKNEEG4fl);
MJEKNEEG4fl = MJEKNEEG4fl *1/101;
MJEKNEEG5fl = Np8vileftkneenormal001fl(:,5) -
Np8xsleftkneenormal001fl(:,5);
MJEKNEEG5fl = abs(MJEKNEEG5fl);
MJEKNEEG5fl = sum(MJEKNEEG5fl);
MJEKNEEG5fl = MJEKNEEG5fl *1/101;
MJEKNEEG1in = Np8vileftkneenormal001in(:,1) -
Np8xsleftkneenormal001in(:,1);
MJEKNEEG1in = abs(MJEKNEEG1in);
MJEKNEEG1in = sum(MJEKNEEG1in);
MJEKNEEG1in = MJEKNEEG1in *1/101;
MJEKNEEG2in = Np8vileftkneenormal001in(:,2) -
Np8xsleftkneenormal001in(:,2);
MJEKNEEG2in = abs(MJEKNEEG2in);
MJEKNEEG2in = sum(MJEKNEEG2in);
MJEKNEEG2in = MJEKNEEG2in *1/101;
MJEKNEEG3in = Np8vileftkneenormal001in(:,3) -
Np8xsleftkneenormal001in(:,3);
MJEKNEEG3in = abs(MJEKNEEG3in);
MJEKNEEG3in = sum(MJEKNEEG3in);
MJEKNEEG3in = MJEKNEEG3in *1/101;
MJEKNEEG4in = Np8vileftkneenormal001in(:,4) -
Np8xsleftkneenormal001in(:,4);
MJEKNEEG4in = abs(MJEKNEEG4in);
MJEKNEEG4in = sum(MJEKNEEG4in);
MJEKNEEG4in = MJEKNEEG4in *1/101;
MJEKNEEG5in = Np8vileftkneenormal001in(:,5) -
Np8xsleftkneenormal001in(:,5);
MJEKNEEG5in = abs(MJEKNEEG5in);
MJEKNEEG5in = sum(MJEKNEEG5in);
MJEKNEEG5in = MJEKNEEG5in *1/101;

%%CMC
% ? is an angle at a particular captured frame
% ?vf is an angle from vicon at a particular frame
% ?xf is an angle from xsens at a particular frame
% f = a captured frame
% F = 101 (the total captured frames per cycle)
% P = 2 (number of motion capturing methods/ 1= vicon, 1= xsens)
% G = 5 (number of gait cycles)

%GFg(P-1) =   5*101(2-1) = 505
%GFg(P-1) =   4*101(2-1) = 404  %% p17 slow speed only
%G(PFg-1) = 5(2*101-1) = 1005
```

```
%G(PFg-1) = 4(2*101-1) = 804%% p17 slow speed only

% the mean angle at frame f (Mf) between angles measured by the 2
systems for the gait cycle g
%Mf = Mf/2;  (2 is P)

%%%%leftkneefl
MfKNEEG1FL = Np8vileftkneenormal001fl(:,1) +
Np8xsleftkneenormal001fl(:,1);
MfKNEEG1FL = MfKNEEG1FL/2;
MfKNEEG2FL = Np8vileftkneenormal001fl(:,2) +
Np8xsleftkneenormal001fl(:,2);
MfKNEEG2FL = MfKNEEG2FL/2;
MfKNEEG3FL = Np8vileftkneenormal001fl(:,3) +
Np8xsleftkneenormal001fl(:,3);
MfKNEEG3FL = MfKNEEG3FL/2;
MfKNEEG4FL = Np8vileftkneenormal001fl(:,4) +
Np8xsleftkneenormal001fl(:,4);
MfKNEEG4FL = MfKNEEG4FL/2;
MfKNEEG5FL = Np8vileftkneenormal001fl(:,5) +
Np8xsleftkneenormal001fl(:,5);
MfKNEEG5FL = MfKNEEG5FL/2;

%% the grand mean (Mg) for the gait cycle "g" among these two
methods

MgkneeFLnormal001 = sum(Np8vileftkneenormal001fl)+
sum(Np8xsleftkneenormal001fl);
MgkneeFLnormal001 = MgkneeFLnormal001/202; % devided by 1/2F;

CMCLeftkneeviFLG1 = Np8vileftkneenormal001fl(:,1)- MfKNEEG1FL;
CMCLeftkneeviFLG1 = CMCLeftkneeviFLG1.^2;
CMCLeftkneeviFLG1 = sum(CMCLeftkneeviFLG1);
CMCLeftkneeviFLG1 = CMCLeftkneeviFLG1/505;
CMCLeftkneexsFLG1 = Np8xsleftkneenormal001fl(:,1)- MfKNEEG1FL;
CMCLeftkneexsFLG1 = CMCLeftkneexsFLG1.^2;
CMCLeftkneexsFLG1 = sum(CMCLeftkneexsFLG1);
CMCLeftkneexsFLG1 = CMCLeftkneexsFLG1/505;
CMCLeftkneeviFLG2 = Np8vileftkneenormal001fl(:,2)- MfKNEEG2FL;
CMCLeftkneeviFLG2 = CMCLeftkneeviFLG2.^2;
CMCLeftkneeviFLG2 = sum(CMCLeftkneeviFLG2);
CMCLeftkneeviFLG2 = CMCLeftkneeviFLG2/505;
CMCLeftkneexsFLG2 = Np8xsleftkneenormal001fl(:,2)- MfKNEEG2FL;
CMCLeftkneexsFLG2 = CMCLeftkneexsFLG2.^2;
CMCLeftkneexsFLG2 = sum(CMCLeftkneexsFLG2);
CMCLeftkneexsFLG2 = CMCLeftkneexsFLG2/505;
CMCLeftkneeviFLG3 = Np8vileftkneenormal001fl(:,3)- MfKNEEG3FL;
CMCLeftkneeviFLG3 = CMCLeftkneeviFLG3.^2;
CMCLeftkneeviFLG3 = sum(CMCLeftkneeviFLG3);
CMCLeftkneeviFLG3 = CMCLeftkneeviFLG3/505;
CMCLeftkneexsFLG3 = Np8xsleftkneenormal001fl(:,3)- MfKNEEG3FL;
CMCLeftkneexsFLG3 = CMCLeftkneexsFLG3.^2;
CMCLeftkneexsFLG3 = sum(CMCLeftkneexsFLG3);
CMCLeftkneexsFLG3 = CMCLeftkneexsFLG3/505;
CMCLeftkneeviFLG4 = Np8vileftkneenormal001fl(:,4)- MfKNEEG4FL;
CMCLeftkneeviFLG4 = CMCLeftkneeviFLG4.^2;
CMCLeftkneeviFLG4 = sum(CMCLeftkneeviFLG4);
CMCLeftkneeviFLG4 = CMCLeftkneeviFLG4/505;
CMCLeftkneexsFLG4 = Np8xsleftkneenormal001fl(:,4)- MfKNEEG4FL;
CMCLeftkneexsFLG4 = CMCLeftkneexsFLG4.^2;
CMCLeftkneexsFLG4 = sum(CMCLeftkneexsFLG4);
CMCLeftkneexsFLG4 = CMCLeftkneexsFLG4/505;
```

```
CMCLeftkneeviFLG5 = Np8vileftkneenormal001fl(:,5)- MfKNEEG5FL;
CMCLeftkneeviFLG5 = CMCLeftkneeviFLG5.^2;
CMCLeftkneeviFLG5 = sum(CMCLeftkneeviFLG5);
CMCLeftkneeviFLG5 = CMCLeftkneeviFLG5/505;
CMCLeftkneexsFLG5 = Np8xsleftkneenormal001fl(:,5)- MfKNEEG5FL;
CMCLeftkneexsFLG5 = CMCLeftkneexsFLG5.^2;
CMCLeftkneexsFLG5 = sum(CMCLeftkneexsFLG5);
CMCLeftkneexsFLG5 = CMCLeftkneexsFLG5/505;

CMCLeftkneeFLG1 = CMCLeftkneeviFLG1 + CMCLeftkneexsFLG1;
CMCLeftkneeFLG2 = CMCLeftkneeviFLG2 + CMCLeftkneexsFLG2;
CMCLeftkneeFLG3 = CMCLeftkneeviFLG3 + CMCLeftkneexsFLG3;
CMCLeftkneeFLG4 = CMCLeftkneeviFLG4 + CMCLeftkneexsFLG4;
CMCLeftkneeFLG5 = CMCLeftkneeviFLG5 + CMCLeftkneexsFLG5;

CMCLeftkneeFL1 = CMCLeftkneeFLG1 + CMCLeftkneeFLG2 + CMCLeftkneeFLG3
+ CMCLeftkneeFLG4 + CMCLeftkneeFLG5;

CMC2LeftkneeviFLG1 = Np8vileftkneenormal001fl(:,1)-
MgkneeFLnormal001(1,1);
CMC2LeftkneeviFLG1 = CMC2LeftkneeviFLG1.^2;
CMC2LeftkneeviFLG1 = sum(CMC2LeftkneeviFLG1);
CMC2LeftkneeviFLG1 = CMC2LeftkneeviFLG1/1005;
CMC2LeftkneexsFLG1 = Np8xsleftkneenormal001fl(:,1)-
MgkneeFLnormal001(1,1);
CMC2LeftkneexsFLG1 = CMC2LeftkneexsFLG1.^2;
CMC2LeftkneexsFLG1 = sum(CMC2LeftkneexsFLG1);
CMC2LeftkneexsFLG1 = CMC2LeftkneexsFLG1/1005;
CMC2LeftkneeviFLG2 = Np8vileftkneenormal001fl(:,2)-
MgkneeFLnormal001(1,2);
CMC2LeftkneeviFLG2 = CMC2LeftkneeviFLG2.^2;
CMC2LeftkneeviFLG2 = sum(CMC2LeftkneeviFLG2);
CMC2LeftkneeviFLG2 = CMC2LeftkneeviFLG2/1005;
CMC2LeftkneexsFLG2 = Np8xsleftkneenormal001fl(:,2)-
MgkneeFLnormal001(1,2);
CMC2LeftkneexsFLG2 = CMC2LeftkneexsFLG2.^2;
CMC2LeftkneexsFLG2 = sum(CMC2LeftkneexsFLG2);
CMC2LeftkneexsFLG2 = CMC2LeftkneexsFLG2/1005;
CMC2LeftkneeviFLG3 = Np8vileftkneenormal001fl(:,3)-
MgkneeFLnormal001(1,3);
CMC2LeftkneeviFLG3 = CMC2LeftkneeviFLG3.^2;
CMC2LeftkneeviFLG3 = sum(CMC2LeftkneeviFLG3);
CMC2LeftkneeviFLG3 = CMC2LeftkneeviFLG3/1005;
CMC2LeftkneexsFLG3 = Np8xsleftkneenormal001fl(:,3)-
MgkneeFLnormal001(1,3);
CMC2LeftkneexsFLG3 = CMC2LeftkneexsFLG3.^2;
CMC2LeftkneexsFLG3 = sum(CMC2LeftkneexsFLG3);
CMC2LeftkneexsFLG3 = CMC2LeftkneexsFLG3/1005;
CMC2LeftkneeviFLG4 = Np8vileftkneenormal001fl(:,4)-
MgkneeFLnormal001(1,4);
CMC2LeftkneeviFLG4 = CMC2LeftkneeviFLG4.^2;
CMC2LeftkneeviFLG4 = sum(CMC2LeftkneeviFLG4);
CMC2LeftkneeviFLG4 = CMC2LeftkneeviFLG4/1005;
CMC2LeftkneexsFLG4 = Np8xsleftkneenormal001fl(:,4)-
MgkneeFLnormal001(1,4);
CMC2LeftkneexsFLG4 = CMC2LeftkneexsFLG4.^2;
CMC2LeftkneexsFLG4 = sum(CMC2LeftkneexsFLG4);
CMC2LeftkneexsFLG4 = CMC2LeftkneexsFLG4/1005;
CMC2LeftkneeviFLG5 = Np8vileftkneenormal001fl(:,5)-
MgkneeFLnormal001(1,5);
CMC2LeftkneeviFLG5 = CMC2LeftkneeviFLG5.^2;
CMC2LeftkneeviFLG5 = sum(CMC2LeftkneeviFLG5);
```

```
CMC2LeftkneeviFLG5 = CMC2LeftkneeviFLG5/1005;
CMC2LeftkneexsFLG5 = Np8xsleftkneenormal001fl(:,5)-
MgkneeFLnormal001(1,5);
CMC2LeftkneexsFLG5 = CMC2LeftkneexsFLG5.^2;
CMC2LeftkneexsFLG5 = sum(CMC2LeftkneexsFLG5);
CMC2LeftkneexsFLG5 = CMC2LeftkneexsFLG5/1005;

CMC2LeftkneeFLG1 = CMC2LeftkneeviFLG1 + CMC2LeftkneexsFLG1;
CMC2LeftkneeFLG2 = CMC2LeftkneeviFLG2 + CMC2LeftkneexsFLG2;
CMC2LeftkneeFLG3 = CMC2LeftkneeviFLG3 + CMC2LeftkneexsFLG3;
CMC2LeftkneeFLG4 = CMC2LeftkneeviFLG4 + CMC2LeftkneexsFLG4;
CMC2LeftkneeFLG5 = CMC2LeftkneeviFLG5 + CMC2LeftkneexsFLG5;


CMC2LeftkneeFL1 = CMC2LeftkneeFLG1 + CMC2LeftkneeFLG2 +
CMC2LeftkneeFLG3 + CMC2LeftkneeFLG4 + CMC2LeftkneeFLG5;

CMCleftkneeflp8normal001 = CMCLeftkneeFL1/ CMC2LeftkneeFL1;
CMCleftkneeflp8normal001 = 1- CMCleftkneeflp8normal001;
CMCleftkneeflp8normal001 = sqrt(CMCleftkneeflp8normal001)


%%%%leftkneeab

% the mean angle at frame f (Mf) between angles measured by the 2
systems for the gait cycle g
%Mf = Mf/2;  (2 is P)

MfKNEEG1AB = Np8vileftkneenormal001ab(:,1) +
Np8xsleftkneenormal001ab(:,1);
MfKNEEG1AB = MfKNEEG1AB/2;
MfKNEEG2AB = Np8vileftkneenormal001ab(:,2) +
Np8xsleftkneenormal001ab(:,2);
MfKNEEG2AB = MfKNEEG2AB/2;
MfKNEEG3AB = Np8vileftkneenormal001ab(:,3) +
Np8xsleftkneenormal001ab(:,3);
MfKNEEG3AB = MfKNEEG3AB/2;
MfKNEEG4AB = Np8vileftkneenormal001ab(:,4) +
Np8xsleftkneenormal001ab(:,4);
MfKNEEG4AB = MfKNEEG4AB/2;
MfKNEEG5AB = Np8vileftkneenormal001ab(:,5) +
Np8xsleftkneenormal001ab(:,5);
MfKNEEG5AB = MfKNEEG5AB/2;

%% the grand mean (Mg) for the gait cycle "g" among these two
methods

MgkneeABnormal001 = sum(Np8vileftkneenormal001ab)+
sum(Np8xsleftkneenormal001ab);
MgkneeABnormal001 = MgkneeABnormal001/202; % devided by 1/2F;

CMCLeftkneeviABG1 = Np8vileftkneenormal001ab(:,1)- MfKNEEG1AB;
CMCLeftkneeviABG1 = CMCLeftkneeviABG1.^2;
CMCLeftkneeviABG1 = sum(CMCLeftkneeviABG1);
CMCLeftkneeviABG1 = CMCLeftkneeviABG1/505;
CMCLeftkneexsABG1 = Np8xsleftkneenormal001ab(:,1)- MfKNEEG1AB;
CMCLeftkneexsABG1 = CMCLeftkneexsABG1.^2;
CMCLeftkneexsABG1 = sum(CMCLeftkneexsABG1);
CMCLeftkneexsABG1 = CMCLeftkneexsABG1/505;
CMCLeftkneeviABG2 = Np8vileftkneenormal001ab(:,2)- MfKNEEG2AB;
CMCLeftkneeviABG2 = CMCLeftkneeviABG2.^2;
```

```
CMCLeftkneeviABG2 = sum(CMCLeftkneeviABG2);
CMCLeftkneeviABG2 = CMCLeftkneeviABG2/505;
CMCLeftkneexsABG2 = Np8xsleftkneenormal001ab(:,2)- MfKNEEG2AB;
CMCLeftkneexsABG2 = CMCLeftkneexsABG2.^2;
CMCLeftkneexsABG2 = sum(CMCLeftkneexsABG2);
CMCLeftkneexsABG2 = CMCLeftkneexsABG2/505;
CMCLeftkneeviABG3 = Np8vileftkneenormal001ab(:,3)- MfKNEEG3AB;
CMCLeftkneeviABG3 = CMCLeftkneeviABG3.^2;
CMCLeftkneeviABG3 = sum(CMCLeftkneeviABG3);
CMCLeftkneeviABG3 = CMCLeftkneeviABG3/505;
CMCLeftkneexsABG3 = Np8xsleftkneenormal001ab(:,3)- MfKNEEG3AB;
CMCLeftkneexsABG3 = CMCLeftkneexsABG3.^2;
CMCLeftkneexsABG3 = sum(CMCLeftkneexsABG3);
CMCLeftkneexsABG3 = CMCLeftkneexsABG3/505;
CMCLeftkneeviABG4 = Np8vileftkneenormal001ab(:,4)- MfKNEEG4AB;
CMCLeftkneeviABG4 = CMCLeftkneeviABG4.^2;
CMCLeftkneeviABG4 = sum(CMCLeftkneeviABG4);
CMCLeftkneeviABG4 = CMCLeftkneeviABG4/505;
CMCLeftkneexsABG4 = Np8xsleftkneenormal001ab(:,4)- MfKNEEG4AB;
CMCLeftkneexsABG4 = CMCLeftkneexsABG4.^2;
CMCLeftkneexsABG4 = sum(CMCLeftkneexsABG4);
CMCLeftkneexsABG4 = CMCLeftkneexsABG4/505;
CMCLeftkneeviABG5 = Np8vileftkneenormal001ab(:,5)- MfKNEEG5AB;
CMCLeftkneeviABG5 = CMCLeftkneeviABG5.^2;
CMCLeftkneeviABG5 = sum(CMCLeftkneeviABG5);
CMCLeftkneeviABG5 = CMCLeftkneeviABG5/505;
CMCLeftkneexsABG5 = Np8xsleftkneenormal001ab(:,5)- MfKNEEG5AB;
CMCLeftkneexsABG5 = CMCLeftkneexsABG5.^2;
CMCLeftkneexsABG5 = sum(CMCLeftkneexsABG5);
CMCLeftkneexsABG5 = CMCLeftkneexsABG5/505;

CMCLeftkneeABG1 = CMCLeftkneeviABG1 + CMCLeftkneexsABG1;
CMCLeftkneeABG2 = CMCLeftkneeviABG2 + CMCLeftkneexsABG2;
CMCLeftkneeABG3 = CMCLeftkneeviABG3 + CMCLeftkneexsABG3;
CMCLeftkneeABG4 = CMCLeftkneeviABG4 + CMCLeftkneexsABG4;
CMCLeftkneeABG5 = CMCLeftkneeviABG5 + CMCLeftkneexsABG5;

CMCLeftkneeAB1 = CMCLeftkneeABG1 + CMCLeftkneeABG2 + CMCLeftkneeABG3
+ CMCLeftkneeABG4 + CMCLeftkneeABG5;

CMC2LeftkneeviABG1 = Np8vileftkneenormal001ab(:,1)-
MgkneeABnormal001(1,1);
CMC2LeftkneeviABG1 = CMC2LeftkneeviABG1.^2;
CMC2LeftkneeviABG1 = sum(CMC2LeftkneeviABG1);
CMC2LeftkneeviABG1 = CMC2LeftkneeviABG1/1005;
CMC2LeftkneexsABG1 = Np8xsleftkneenormal001ab(:,1)-
MgkneeABnormal001(1,1);
CMC2LeftkneexsABG1 = CMC2LeftkneexsABG1.^2;
CMC2LeftkneexsABG1 = sum(CMC2LeftkneexsABG1);
CMC2LeftkneexsABG1 = CMC2LeftkneexsABG1/1005;
CMC2LeftkneeviABG2 = Np8vileftkneenormal001ab(:,2)-
MgkneeABnormal001(1,2);
CMC2LeftkneeviABG2 = CMC2LeftkneeviABG2.^2;
CMC2LeftkneeviABG2 = sum(CMC2LeftkneeviABG2);
CMC2LeftkneeviABG2 = CMC2LeftkneeviABG2/1005;
CMC2LeftkneexsABG2 = Np8xsleftkneenormal001ab(:,2)-
MgkneeABnormal001(1,2);
CMC2LeftkneexsABG2 = CMC2LeftkneexsABG2.^2;
CMC2LeftkneexsABG2 = sum(CMC2LeftkneexsABG2);
CMC2LeftkneexsABG2 = CMC2LeftkneexsABG2/1005;
CMC2LeftkneeviABG3 = Np8vileftkneenormal001ab(:,3)-
MgkneeABnormal001(1,3);
```

```
CMC2LeftkneeviABG3 = CMC2LeftkneeviABG3.^2;
CMC2LeftkneeviABG3 = sum(CMC2LeftkneeviABG3);
CMC2LeftkneeviABG3 = CMC2LeftkneeviABG3/1005;
CMC2LeftkneexsABG3 = Np8xsleftkneenormal001ab(:,3)-
MgkneeABnormal001(1,3);
CMC2LeftkneexsABG3 = CMC2LeftkneexsABG3.^2;
CMC2LeftkneexsABG3 = sum(CMC2LeftkneexsABG3);
CMC2LeftkneexsABG3 = CMC2LeftkneexsABG3/1005;
CMC2LeftkneeviABG4 = Np8vileftkneenormal001ab(:,4)-
MgkneeABnormal001(1,4);
CMC2LeftkneeviABG4 = CMC2LeftkneeviABG4.^2;
CMC2LeftkneeviABG4 = sum(CMC2LeftkneeviABG4);
CMC2LeftkneeviABG4 = CMC2LeftkneeviABG4/1005;
CMC2LeftkneexsABG4 = Np8xsleftkneenormal001ab(:,4)-
MgkneeABnormal001(1,4);
CMC2LeftkneexsABG4 = CMC2LeftkneexsABG4.^2;
CMC2LeftkneexsABG4 = sum(CMC2LeftkneexsABG4);
CMC2LeftkneexsABG4 = CMC2LeftkneexsABG4/1005;
CMC2LeftkneeviABG5 = Np8vileftkneenormal001ab(:,5)-
MgkneeABnormal001(1,5);
CMC2LeftkneeviABG5 = CMC2LeftkneeviABG5.^2;
CMC2LeftkneeviABG5 = sum(CMC2LeftkneeviABG5);
CMC2LeftkneeviABG5 = CMC2LeftkneeviABG5/1005;
CMC2LeftkneexsABG5 = Np8xsleftkneenormal001ab(:,5)-
MgkneeABnormal001(1,5);
CMC2LeftkneexsABG5 = CMC2LeftkneexsABG5.^2;
CMC2LeftkneexsABG5 = sum(CMC2LeftkneexsABG5);
CMC2LeftkneexsABG5 = CMC2LeftkneexsABG5/1005;

CMC2LeftkneeABG1 = CMC2LeftkneeviABG1 + CMC2LeftkneexsABG1;
CMC2LeftkneeABG2 = CMC2LeftkneeviABG2 + CMC2LeftkneexsABG2;
CMC2LeftkneeABG3 = CMC2LeftkneeviABG3 + CMC2LeftkneexsABG3;
CMC2LeftkneeABG4 = CMC2LeftkneeviABG4 + CMC2LeftkneexsABG4;
CMC2LeftkneeABG5 = CMC2LeftkneeviABG5 + CMC2LeftkneexsABG5;


CMC2LeftkneeAB1 = CMC2LeftkneeABG1 + CMC2LeftkneeABG2 +
CMC2LeftkneeABG3 + CMC2LeftkneeABG4 + CMC2LeftkneeABG5;

CMCleftkneeabp8normal001 = CMCLeftkneeAB1/ CMC2LeftkneeAB1;
CMCleftkneeabp8normal001 = 1- CMCleftkneeabp8normal001;
CMCleftkneeabp8normal001 = sqrt(CMCleftkneeabp8normal001)

%%%%leftkneein

% the mean angle at frame f (Mf) between angles measured by the 2
systems for the gait cycle g
%Mf = Mf/2;  (2 is P)

MfKNEEG1IN = Np8vileftkneenormal001in(:,1) +
Np8xsleftkneenormal001in(:,1);
MfKNEEG1IN = MfKNEEG1IN/2;
MfKNEEG2IN = Np8vileftkneenormal001in(:,2) +
Np8xsleftkneenormal001in(:,2);
MfKNEEG2IN = MfKNEEG2IN/2;
MfKNEEG3IN = Np8vileftkneenormal001in(:,3) +
Np8xsleftkneenormal001in(:,3);
MfKNEEG3IN = MfKNEEG3IN/2;
MfKNEEG4IN = Np8vileftkneenormal001in(:,4) +
Np8xsleftkneenormal001in(:,4);
MfKNEEG4IN = MfKNEEG4IN/2;
```

```
MfKNEEG5IN = Np8vileftkneenormal001in(:,5) +
Np8xsleftkneenormal001in(:,5);
MfKNEEG5IN = MfKNEEG5IN/2;

%% the grand mean (Mg) for the gait cycle "g" among these two
methods

MgkneeINnormal001 = sum(Np8vileftkneenormal001in)+
sum(Np8xsleftkneenormal001in);
MgkneeINnormal001 = MgkneeINnormal001/202; % devided by 1/2F;

CMCLeftkneeviING1 = Np8vileftkneenormal001in(:,1)- MfKNEEG1IN;
CMCLeftkneeviING1 = CMCLeftkneeviING1.^2;
CMCLeftkneeviING1 = sum(CMCLeftkneeviING1);
CMCLeftkneeviING1 = CMCLeftkneeviING1/505;
CMCLeftkneexsING1 = Np8xsleftkneenormal001in(:,1)- MfKNEEG1IN;
CMCLeftkneexsING1 = CMCLeftkneexsING1.^2;
CMCLeftkneexsING1 = sum(CMCLeftkneexsING1);
CMCLeftkneexsING1 = CMCLeftkneexsING1/505;
CMCLeftkneeviING2 = Np8vileftkneenormal001in(:,2)- MfKNEEG2IN;
CMCLeftkneeviING2 = CMCLeftkneeviING2.^2;
CMCLeftkneeviING2 = sum(CMCLeftkneeviING2);
CMCLeftkneeviING2 = CMCLeftkneeviING2/505;
CMCLeftkneexsING2 = Np8xsleftkneenormal001in(:,2)- MfKNEEG2IN;
CMCLeftkneexsING2 = CMCLeftkneexsING2.^2;
CMCLeftkneexsING2 = sum(CMCLeftkneexsING2);
CMCLeftkneexsING2 = CMCLeftkneexsING2/505;
CMCLeftkneeviING3 = Np8vileftkneenormal001in(:,3)- MfKNEEG3IN;
CMCLeftkneeviING3 = CMCLeftkneeviING3.^2;
CMCLeftkneeviING3 = sum(CMCLeftkneeviING3);
CMCLeftkneeviING3 = CMCLeftkneeviING3/505;
CMCLeftkneexsING3 = Np8xsleftkneenormal001in(:,3)- MfKNEEG3IN;
CMCLeftkneexsING3 = CMCLeftkneexsING3.^2;
CMCLeftkneexsING3 = sum(CMCLeftkneexsING3);
CMCLeftkneexsING3 = CMCLeftkneexsING3/505;
CMCLeftkneeviING4 = Np8vileftkneenormal001in(:,4)- MfKNEEG4IN;
CMCLeftkneeviING4 = CMCLeftkneeviING4.^2;
CMCLeftkneeviING4 = sum(CMCLeftkneeviING4);
CMCLeftkneeviING4 = CMCLeftkneeviING4/505;
CMCLeftkneexsING4 = Np8xsleftkneenormal001in(:,4)- MfKNEEG4IN;
CMCLeftkneexsING4 = CMCLeftkneexsING4.^2;
CMCLeftkneexsING4 = sum(CMCLeftkneexsING4);
CMCLeftkneexsING4 = CMCLeftkneexsING4/505;
CMCLeftkneeviING5 = Np8vileftkneenormal001in(:,5)- MfKNEEG5IN;
CMCLeftkneeviING5 = CMCLeftkneeviING5.^2;
CMCLeftkneeviING5 = sum(CMCLeftkneeviING5);
CMCLeftkneeviING5 = CMCLeftkneeviING5/505;
CMCLeftkneexsING5 = Np8xsleftkneenormal001in(:,5)- MfKNEEG5IN;
CMCLeftkneexsING5 = CMCLeftkneexsING5.^2;
CMCLeftkneexsING5 = sum(CMCLeftkneexsING5);
CMCLeftkneexsING5 = CMCLeftkneexsING5/505;

CMCLeftkneeING1 = CMCLeftkneeviING1 + CMCLeftkneexsING1;
CMCLeftkneeING2 = CMCLeftkneeviING2 + CMCLeftkneexsING2;
CMCLeftkneeING3 = CMCLeftkneeviING3 + CMCLeftkneexsING3;
CMCLeftkneeING4 = CMCLeftkneeviING4 + CMCLeftkneexsING4;
CMCLeftkneeING5 = CMCLeftkneeviING5 + CMCLeftkneexsING5;

CMCLeftkneeIN1 = CMCLeftkneeING1 + CMCLeftkneeING2 + CMCLeftkneeING3
+ CMCLeftkneeING4 + CMCLeftkneeING5;
```

```
CMC2LeftkneeviING1 = Np8vileftkneenormal001in(:,1)-
MgkneeINnormal001(1,1);
CMC2LeftkneeviING1 = CMC2LeftkneeviING1.^2;
CMC2LeftkneeviING1 = sum(CMC2LeftkneeviING1);
CMC2LeftkneeviING1 = CMC2LeftkneeviING1/1005;
CMC2LeftkneexsING1 = Np8xsleftkneenormal001in(:,1)-
MgkneeINnormal001(1,1);
CMC2LeftkneexsING1 = CMC2LeftkneexsING1.^2;
CMC2LeftkneexsING1 = sum(CMC2LeftkneexsING1);
CMC2LeftkneexsING1 = CMC2LeftkneexsING1/1005;
CMC2LeftkneeviING2 = Np8vileftkneenormal001in(:,2)-
MgkneeINnormal001(1,2);
CMC2LeftkneeviING2 = CMC2LeftkneeviING2.^2;
CMC2LeftkneeviING2 = sum(CMC2LeftkneeviING2);
CMC2LeftkneeviING2 = CMC2LeftkneeviING2/1005;
CMC2LeftkneexsING2 = Np8xsleftkneenormal001in(:,2)-
MgkneeINnormal001(1,2);
CMC2LeftkneexsING2 = CMC2LeftkneexsING2.^2;
CMC2LeftkneexsING2 = sum(CMC2LeftkneexsING2);
CMC2LeftkneexsING2 = CMC2LeftkneexsING2/1005;
CMC2LeftkneeviING3 = Np8vileftkneenormal001in(:,3)-
MgkneeINnormal001(1,3);
CMC2LeftkneeviING3 = CMC2LeftkneeviING3.^2;
CMC2LeftkneeviING3 = sum(CMC2LeftkneeviING3);
CMC2LeftkneeviING3 = CMC2LeftkneeviING3/1005;
CMC2LeftkneexsING3 = Np8xsleftkneenormal001in(:,3)-
MgkneeINnormal001(1,3);
CMC2LeftkneexsING3 = CMC2LeftkneexsING3.^2;
CMC2LeftkneexsING3 = sum(CMC2LeftkneexsING3);
CMC2LeftkneexsING3 = CMC2LeftkneexsING3/1005;
CMC2LeftkneeviING4 = Np8vileftkneenormal001in(:,4)-
MgkneeINnormal001(1,4);
CMC2LeftkneeviING4 = CMC2LeftkneeviING4.^2;
CMC2LeftkneeviING4 = sum(CMC2LeftkneeviING4);
CMC2LeftkneeviING4 = CMC2LeftkneeviING4/1005;
CMC2LeftkneexsING4 = Np8xsleftkneenormal001in(:,4)-
MgkneeINnormal001(1,4);
CMC2LeftkneexsING4 = CMC2LeftkneexsING4.^2;
CMC2LeftkneexsING4 = sum(CMC2LeftkneexsING4);
CMC2LeftkneexsING4 = CMC2LeftkneexsING4/1005;
CMC2LeftkneeviING5 = Np8vileftkneenormal001in(:,5)-
MgkneeINnormal001(1,5);
CMC2LeftkneeviING5 = CMC2LeftkneeviING5.^2;
CMC2LeftkneeviING5 = sum(CMC2LeftkneeviING5);
CMC2LeftkneeviING5 = CMC2LeftkneeviING5/1005;
CMC2LeftkneexsING5 = Np8xsleftkneenormal001in(:,5)-
MgkneeINnormal001(1,5);
CMC2LeftkneexsING5 = CMC2LeftkneexsING5.^2;
CMC2LeftkneexsING5 = sum(CMC2LeftkneexsING5);
CMC2LeftkneexsING5 = CMC2LeftkneexsING5/1005;

CMC2LeftkneeING1 = CMC2LeftkneeviING1 + CMC2LeftkneexsING1;
CMC2LeftkneeING2 = CMC2LeftkneeviING2 + CMC2LeftkneexsING2;
CMC2LeftkneeING3 = CMC2LeftkneeviING3 + CMC2LeftkneexsING3;
CMC2LeftkneeING4 = CMC2LeftkneeviING4 + CMC2LeftkneexsING4;
CMC2LeftkneeING5 = CMC2LeftkneeviING5 + CMC2LeftkneexsING5;


CMC2LeftkneeIN1 = CMC2LeftkneeING1 + CMC2LeftkneeING2 +
CMC2LeftkneeING3 + CMC2LeftkneeING4 + CMC2LeftkneeING5;

CMCleftkneeinp8normal001 = CMCLeftkneeIN1/ CMC2LeftkneeIN1;
```

```
CMCleftkneeinp8normal001 = 1- CMCleftkneeinp8normal001;
CMCleftkneeinp8normal001 = sqrt(CMCleftkneeinp8normal001)

%%%%end of the calculation
```

Appendix 5. A Matlab script used for KAM prediction (joint angle obtained directly from Xsens)

```
perf= zeros(30,2);

 seed = 11 % best of 300 with 1000f
% for seed=1:30
    %rng(0) % good with 500
    rng(seed) % good with 1999
%xnm = input/angle for normalspeed
%tnm = target/moment for normalspeed
%xPn_1 = angle for participant n at normal speed
%xPn_2 = angle for participant n at fast speed
%xPn_3 = angle for participant n at slow speed
%a = right side knee abduction moment
%b = left side knee abduction moment

 xP2_1 = xlsread('Formatlab\P2\normal001rightsideangles')';
 xP2_1 = xP2_1(:,1:1000);
 xP2_2 = xlsread('Formatlab\P2\fast001rightsideangles')';
 xP2_2 = xP2_2(:,1:1000);
 xP2_3 = xlsread('Formatlab\P2\slow001rightsideangles')';
 xP2_3 = xP2_3(:,1:1000);
 xP3_1 = xlsread('Formatlab\P3\normal001rightsideangles')';
 xP3_1 = xP3_1(:,1:1000);
 xP3_2 = xlsread('Formatlab\P3\fast001rightsideangles')';
 xP3_2 = xP3_2(:,1:1000);
 xP3_3 = xlsread('Formatlab\P3\slow001rightsideangles')';
 xP3_3 = xP3_3(:,1:1000);
 xP4_1 = xlsread('Formatlab\P4\normal001rightsideangles')';
 xP4_1 = xP4_1(:,1:1000);
 xP4_2 = xlsread('Formatlab\P4\fast001rightsideangles')';
 xP4_2 = xP4_2(:,1:1000);
 xP4_3 = xlsread('Formatlab\P4\slow001rightsideangles')';
 xP4_3 = xP4_3(:,1:1000);
 xP5_1 = xlsread('Formatlab\P5\normal001rightsideangles')';
 xP5_1 = xP5_1(:,1:1000);
 xP5_2 = xlsread('Formatlab\P5\fast001rightsideangles')';
 xP5_2 = xP5_2(:,1:1000);
 xP5_3 = xlsread('Formatlab\P5\slow001rightsideangles')';
 xP5_3 = xP5_3(:,1:1000);
 xP6_1 = xlsread('Formatlab\P6\normal001rightsideangles')';
 xP6_1 = xP6_1(:,1:1000);
 xP6_2 = xlsread('Formatlab\P6\fast001rightsideangles')';
 xP6_2 = xP6_2(:,1:1000);
 xP6_3 = xlsread('Formatlab\P6\slow001rightsideangles')';
 xP6_3 = xP6_3(:,1:1000);
 xP9_1 = xlsread('Formatlab\P9\normal001rightsideangles')';
 xP9_1 = xP9_1(:,1:1000);
 xP9_2 = xlsread('Formatlab\P9\fast001rightsideangles')';
 xP9_2 = xP9_2(:,1:1000);
 xP9_3 = xlsread('Formatlab\P9\slow001rightsideangles')';
 xP9_3 = xP9_3(:,1:1000);
 xP11_1 = xlsread('Formatlab\P11\normal001rightsideangles')';
 xP11_1 = xP11_1(:,1:1000);
 xP11_2 = xlsread('Formatlab\P11\fast001rightsideangles')';
 xP11_2 = xP11_2(:,1:1000);
 xP11_3 = xlsread('Formatlab\P11\slow001rightsideangles')';
 xP11_3 = xP11_3(:,1:1000);
 xP12_1 = xlsread('Formatlab\P12\normal001rightsideangles')';
 xP12_1 = xP12_1(:,1:1000);
```

```
xP12_2 = xlsread('Formatlab\P12\fast001rightsideangles')';
xP12_2 = xP12_2(:,1:1000);
xP12_3 = xlsread('Formatlab\P12\slow001rightsideangles')';
xP12_3 = xP12_3(:,1:1000);
xP13_1 = xlsread('Formatlab\P13\normal001rightsideangles')';
xP13_1 = xP13_1(:,1:1000);
xP13_2 = xlsread('Formatlab\P13\fast001rightsideangles')';
xP13_2 = xP13_2(:,1:1000);
xP13_3 = xlsread('Formatlab\P13\slow001rightsideangles')';
xP13_3 = xP13_3(:,1:1000);
xP14_1 = xlsread('Formatlab\P14\normal001rightsideangles')';
xP14_1 = xP14_1(:,1:1000);
xP14_2 = xlsread('Formatlab\P14\fast001rightsideangles')';
xP14_2 = xP14_2(:,1:1000);
xP14_3 = xlsread('Formatlab\P14\slow001rightsideangles')';
xP14_3 = xP14_3(:,1:1000);
xP15_1 = xlsread('Formatlab\P15\normal001rightsideangles')';
xP15_1 = xP15_1(:,1:1000);
xP15_2 = xlsread('Formatlab\P15\fast001rightsideangles')';
xP15_2 = xP15_2(:,1:1000);
xP15_3 = xlsread('Formatlab\P15\slow001rightsideangles')';
xP15_3 = xP15_3(:,1:1000);
xP16_1 = xlsread('Formatlab\P16\normal001rightsideangles')';
xP16_1 = xP16_1(:,1:1000);
xP16_2 = xlsread('Formatlab\P16\fast001rightsideangles')';
xP16_2 = xP16_2(:,1:1000);
xP16_3 = xlsread('Formatlab\P16\slow001rightsideangles')';
xP16_3 = xP16_3(:,1:1000);
xP17_1 = xlsread('Formatlab\P17\normal001rightsideangles')';
xP17_1 = xP17_1(:,1:1000);
xP17_2 = xlsread('Formatlab\P17\fast001rightsideangles')';
xP17_2 = xP17_2(:,1:1000);
xP17_3 = xlsread('Formatlab\P17\slow001rightsideangles')';
xP17_3 = xP17_3(:,1:1000);
xP18_1 = xlsread('Formatlab\P18\normal001rightsideangles')';
xP18_1 = xP18_1(:,1:1000);
xP18_2 = xlsread('Formatlab\P18\fast001rightsideangles')';
xP18_2 = xP18_2(:,1:1000);
xP18_3 = xlsread('Formatlab\P18\slow001rightsideangles')';
xP18_3 = xP18_3(:,1:1000);
xP8_1 = xlsread('Formatlab\P8\normal001rightsideangles')';
xP8_1 = xP8_1(:,1:1000);
xP8_2 = xlsread('Formatlab\P8\fast001rightsideangles')';
xP8_2 = xP8_2(:,1:1000);
xP8_3 = xlsread('Formatlab\P8\slow001rightsideangles')';
xP8_3 = xP8_3(:,1:1000);


x1 = [xP2_1 xP2_2 xP2_3 xP3_1 xP3_2 xP3_3 xP4_1 xP4_2 xP4_3 xP5_1
xP5_2 xP5_3 xP6_1 xP6_2 xP6_3 xP9_1 xP9_2 xP9_3 xP11_1 xP11_2 xP11_3
xP12_1 xP12_2 xP12_3 xP13_1 xP13_2 xP13_3 xP14_1 xP14_2 xP14_3
xP15_1 xP15_2 xP15_3 xP16_1 xP16_2 xP16_3 xP17_1 xP17_2 xP17_3
xP18_1 xP18_2 xP18_3 xP8_1 xP8_2 xP8_3];
x2 = x1;



tP2_1a = xlsread('Formatlab\P2\normal001rightkneemo')';
tP2_1a = tP2_1a(:,1:1000);
tP2_2a = xlsread('Formatlab\P2\fast001rightkneemo')';
tP2_2a = tP2_2a(:,1:1000);
tP2_3a = xlsread('Formatlab\P2\slow001rightkneemo')';
```

```
tP2_3a = tP2_3a(:,1:1000);
tP3_1a = xlsread('Formatlab\P3\normal001rightkneemo')';
tP3_1a = tP3_1a(:,1:1000);
tP3_2a = xlsread('Formatlab\P3\fast001rightkneemo')';
tP3_2a = tP3_2a(:,1:1000);
tP3_3a = xlsread('Formatlab\P3\slow001rightkneemo')';
tP3_3a = tP3_3a(:,1:1000);
tP4_1a = xlsread('Formatlab\P4\normal001rightkneemo')';
tP4_1a = tP4_1a(:,1:1000);
tP4_2a = xlsread('Formatlab\P4\fast001rightkneemo')';
tP4_2a = tP4_2a(:,1:1000);
tP4_3a = xlsread('Formatlab\P4\slow001rightkneemo')';
tP4_3a = tP4_3a(:,1:1000);
tP5_1a = xlsread('Formatlab\P5\normal001rightkneemo')';
tP5_1a = tP5_1a(:,1:1000);
tP5_2a = xlsread('Formatlab\P5\fast001rightkneemo')';
tP5_2a = tP5_2a(:,1:1000);
tP5_3a = xlsread('Formatlab\P5\slow001rightkneemo')';
tP5_3a = tP5_3a(:,1:1000);
tP6_1a = xlsread('Formatlab\P6\normal001rightkneemo')';
tP6_1a = tP6_1a(:,1:1000);
tP6_2a = xlsread('Formatlab\P6\fast001rightkneemo')';
tP6_2a = tP6_2a(:,1:1000);
tP6_3a = xlsread('Formatlab\P6\slow001rightkneemo')';
tP6_3a = tP6_3a(:,1:1000);
tP9_1a = xlsread('Formatlab\P9\normal001rightkneemo')';
tP9_1a = tP9_1a(:,1:1000);
tP9_2a = xlsread('Formatlab\P9\fast001rightkneemo')';
tP9_2a = tP9_2a(:,1:1000);
tP9_3a = xlsread('Formatlab\P9\slow001rightkneemo')';
tP9_3a = tP9_3a(:,1:1000);
tP11_1a = xlsread('Formatlab\P11\normal001rightkneemo')';
tP11_1a = tP11_1a(:,1:1000);
tP11_2a = xlsread('Formatlab\P11\fast001rightkneemo')';
tP11_2a = tP11_2a(:,1:1000);
tP11_3a = xlsread('Formatlab\P11\slow001rightkneemo')';
tP11_3a = tP11_3a(:,1:1000);
tP12_1a = xlsread('Formatlab\P12\normal001rightkneemo')';
tP12_1a = tP12_1a(:,1:1000);
tP12_2a = xlsread('Formatlab\P12\fast001rightkneemo')';
tP12_2a = tP12_2a(:,1:1000);
tP12_3a = xlsread('Formatlab\P12\slow001rightkneemo')';
tP12_3a = tP12_3a(:,1:1000);
tP13_1a = xlsread('Formatlab\P13\normal001rightkneemo')';
tP13_1a = tP13_1a(:,1:1000);
tP13_2a = xlsread('Formatlab\P13\fast001rightkneemo')';
tP13_2a = tP13_2a(:,1:1000);
tP13_3a = xlsread('Formatlab\P13\slow001rightkneemo')';
tP13_3a = tP13_3a(:,1:1000);
tP14_1a = xlsread('Formatlab\P14\normal001rightkneemo')';
tP14_1a = tP14_1a(:,1:1000);
tP14_2a = xlsread('Formatlab\P14\fast001rightkneemo')';
tP14_2a = tP14_2a(:,1:1000);
tP14_3a = xlsread('Formatlab\P14\slow001rightkneemo')';
tP14_3a = tP14_3a(:,1:1000);
tP15_1a = xlsread('Formatlab\P15\normal001rightkneemo')';
tP15_1a = tP15_1a(:,1:1000);
tP15_2a = xlsread('Formatlab\P15\fast001rightkneemo')';
tP15_2a = tP15_2a(:,1:1000);
tP15_3a = xlsread('Formatlab\P15\slow001rightkneemo')';
tP15_3a = tP15_3a(:,1:1000);
tP16_1a = xlsread('Formatlab\P16\normal001rightkneemo')';
tP16_1a = tP16_1a(:,1:1000);
```

```
tP16_2a = xlsread('Formatlab\P16\fast001rightkneemo')';
tP16_2a = tP16_2a(:,1:1000);
tP16_3a = xlsread('Formatlab\P16\slow001rightkneemo')';
tP16_3a = tP16_3a(:,1:1000);
tP17_1a = xlsread('Formatlab\P17\normal001rightkneemo')';
tP17_1a = tP17_1a(:,1:1000);
tP17_2a = xlsread('Formatlab\P17\fast001rightkneemo')';
tP17_2a = tP17_2a(:,1:1000);
tP17_3a = xlsread('Formatlab\P17\slow001rightkneemo')';
tP17_3a = tP17_3a(:,1:1000);
tP18_1a = xlsread('Formatlab\P18\normal001rightkneemo')';
tP18_1a = tP18_1a(:,1:1000);
tP18_2a = xlsread('Formatlab\P18\fast001rightkneemo')';
tP18_2a = tP18_2a(:,1:1000);
tP18_3a = xlsread('Formatlab\P18\slow001rightkneemo')';
tP18_3a = tP18_3a(:,1:1000);
tP8_1a = xlsread('Formatlab\P8\normal001rightkneemo')';
tP8_1a = tP8_1a(:,1:1000);
tP8_2a = xlsread('Formatlab\P8\fast001rightkneemo')';
tP8_2a = tP8_2a(:,1:1000);
tP8_3a = xlsread('Formatlab\P8\slow001rightkneemo')';
tP8_3a = tP8_3a(:,1:1000);


t1 = [tP2_1a tP2_2a tP2_3a tP3_1a tP3_2a tP3_3a tP4_1a tP4_2a
tP4_3a tP5_1a tP5_2a tP5_3a tP6_1a tP6_2a tP6_3a tP9_1a tP9_2a
tP9_3a tP11_1a tP11_2a tP11_3a tP12_1a tP12_2a tP12_3a tP13_1a
tP13_2a tP13_3a tP14_1a tP14_2a tP14_3a tP15_1a tP15_2a tP15_3a
tP16_1a tP16_2a tP16_3a tP17_1a tP17_2a tP17_3a tP18_1a tP18_2a
tP18_3a tP8_1a tP8_2a tP8_3a];
t2 = t1;

  % Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for challenging problems.
% 'trainscg' uses less memory. Suitable in low memory situations.
trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.

% Create a Fitting Network
hiddenLayerSize = 5;
net = fitnet(hiddenLayerSize,trainFcn);

% Setup Division of Data for Training, Validation, Testing
netnm = fitnet(hiddenLayerSize,trainFcn);


netnm.input.processFcns = {'removeconstantrows','mapminmax'};

    netnm.output.processFcns = {'removeconstantrows','mapminmax'};
    %net.divideFcn = 'dividerand';  % Divide data randomly
    netnm.divideFcn = 'divideind';
    netnm.divideMode = 'sample';  % Divide up every sample
%     net.divideParam.trainRatio = 86.6/100;
%     net.divideParam.valRatio = 6.6/100;
%     net.divideParam.testRatio = 6.6/100;
%%
    netnm.divideParam.trainInd = [1:round(1/15*13*size(x2,2))];
    netnm.divideParam.valInd =  [round(1/15*13*size(x2,2))+1 :
round(1/15*size(x2,2))];
```

```
    netnm.divideParam.testInd = [round(1/15*size(x2,2))+1 :
size(x2,2)];
%%
    netnm.performFcn = 'mse';  % Mean Squared Error
    netnm.plotFcns = {'plotperform','plottrainstate','ploterrhist',
...
        'plotregression', 'plotfit'};
% Train the Network
[netnm,tr] = trainlm(netnm,x2,t2);

% Test the Network
y2 = netnm(x2);
enm = gsubtract(t2,y2);
performance1 = perform(netnm,t2,y2)

% View the Network
% view(net)


    seed
    testPerformance = perform(netnm,performance1,y2);

    perf(seed,1) = seed;
    perf(seed,2) = testPerformance;
    %[seed testPerformance]

t2train = t2(1:round(1/15*13*size(t2,2)));
y2train = y2(1:round(1/15*13*size(y2,2)));

%%
%t13test = t13(:,42030:45000);
t2test = t2(:,14*3000+1:45000);
y2test = y2(:,14*3000+1:45000);
RMSE_Normal = sqrt(mean((t2test(1:1000)-y2test(1:1000)).^2))
RMSE_Fast =   sqrt(mean((t2test(1001:2000)-y2test(1001:2000)).^2))
RMSE_Slow =   sqrt(mean((t2test(2001:3000)-y2test(2001:3000)).^2))
RMSE_Train =   sqrt(mean((t2train-y2train).^2))
%%

NRMSEnormal = RMSE_Normal/(max(y2test(1:1000))-min(y2test(1:1000)))
NRMSEfast = RMSE_Fast/(max(y2test(1001:2000))-
min(y2test(1001:2000)))
NRMSEslow = RMSE_Slow/(max(y2test(2001:3000))-
min(y2test(2001:3000)))


%t14val = t14(:,39000:42000);
%y14val = y14(:,39000:42000);

%t14ns = [t14val t14test];
%y14ns = [y14val y14test];

% Plots

% figure
% plot (t2test)
% hold on
% plot (y2test)

% figure
```

```
% plot (t16)
% hold on
% plot (y16)
%
% figure
% plot (t16ns)
% hold on
% plot (y16ns)


% end
```

Appendix 6. A Matlab script used for data extraction and up sampled

```
%%% This script is for extracting Xsens data from the starting point
%(frame 0)%%%
% each participant (20 in total), 1900 datapoint per trial (then
extract 2000 upsampled datapoint)
% only 1 trial will be extract for each speed
% some of these trials were not at the same order as in the very
first data
% collection, to see the original file the researcher needs to look
up in
% the original data sheet
% to be used in joint moment prediction using an artificial neural
network

% the first 9 columns are the right side angles and the other 9
columns are the left side angles
% the columns are as the following order
% Right Hip Abduction/Adduction, Hip Internal/External Rotation, Hip
Flexion/Extension
% Right Knee Abduction/Adduction, Knee Internal/External Rotation,
Knee Flexion/Extension
% Right Ankle Abduction/Adduction, Ankle Internal/External Rotation,
Ankle Dorsiflexion/Plantarflexion
% Left Hip Abduction/Adduction, Hip Internal/External Rotation, Hip
Flexion/Extension
% Left Knee Abduction/Adduction, Knee Internal/External Rotation,
Knee Flexion/Extension
% Left Ankle Abduction/Adduction, Ankle Internal/External Rotation,
Ankle Dorsiflexion/Plantarflexion

% the second part of the script is to upsample the extracted files
from 100 Hz to 120 Hz
% to be equivalent with the HBM files

%%Participant2

p2002xs = xlsread('Participant2\normal-002angle');
p2002xsrt = p2002xs(2:1901,1:9);
p2002xslf = p2002xs(2:1901,10:18);
p2006xs = xlsread('Participant2\fast-001angle');
p2006xsrt = p2006xs(2:1901,1:9);
p2006xslf = p2006xs(2:1901,10:18);
p2009xs = xlsread('Participant2\slow-001angle');
p2009xsrt = p2009xs(2:1901,1:9);
p2009xslf = p2009xs(2:1901,10:18);

%Participant3

p3001xs = xlsread('Participant3\normal-001angle');
p3001xsrt = p3001xs(2:1901,1:9);
p3001xslf = p3001xs(2:1901,10:18);
p3006xs = xlsread('Participant3\fast-001angle');
p3006xsrt = p3006xs(2:1901,1:9);
p3006xslf = p3006xs(2:1901,10:18);
p3009xs = xlsread('Participant3\slow-001angle');
p3009xsrt = p3009xs(2:1901,1:9);
p3009xslf = p3009xs(2:1901,10:18);
```

```
%Participant4

p4001xs = xlsread('Participant4\normal-001angle');
p4001xsrt = p4001xs(2:1901,1:9);
p4001xslf = p4001xs(2:1901,10:18);
p4006xs = xlsread('Participant4\fast-001angle');
p4006xsrt = p4006xs(2:1901,1:9);
p4006xslf = p4006xs(2:1901,10:18);
p4009xs = xlsread('Participant4\slow-001angle');
p4009xsrt = p4009xs(2:1901,1:9);
p4009xslf = p4009xs(2:1901,10:18);

%Participant5

p5001xs = xlsread('Participant5\normal-001angle');
p5001xsrt = p5001xs(2:1901,1:9);
p5001xslf = p5001xs(2:1901,10:18);
p5006xs = xlsread('Participant5\fast-001angle');
p5006xsrt = p5006xs(2:1901,1:9);
p5006xslf = p5006xs(2:1901,10:18);
p5009xs = xlsread('Participant5\slow-001angle');
p5009xsrt = p5009xs(2:1901,1:9);
p5009xslf = p5009xs(2:1901,10:18);

%Participant6

p6001xs = xlsread('Participant6\normal-001angle');
p6001xsrt = p6001xs(2:1901,1:9);
p6001xslf = p6001xs(2:1901,10:18);
p6006xs = xlsread('Participant6\fast-001angle');
p6006xsrt = p6006xs(2:1901,1:9);
p6006xslf = p6006xs(2:1901,10:18);
p6009xs = xlsread('Participant6\slow-001angle');
p6009xsrt = p6009xs(2:1901,1:9);
p6009xslf = p6009xs(2:1901,10:18);

%Participant8

p8001xs = xlsread('Participant8\normal-001angle');
p8001xsrt = p8001xs(2:1901,1:9);
p8001xslf = p8001xs(2:1901,10:18);
p8006xs = xlsread('Participant8\fast-001angle');
p8006xsrt = p8006xs(2:1901,1:9);
p8006xslf = p8006xs(2:1901,10:18);
p8009xs = xlsread('Participant8\slow-001angle');
p8009xsrt = p8009xs(2:1901,1:9);
p8009xslf = p8009xs(2:1901,10:18);

%Participant9

p9001xs = xlsread('Participant9\normal-001angle');
p9001xsrt = p9001xs(2:1901,1:9);
p9001xslf = p9001xs(2:1901,10:18);
p9006xs = xlsread('Participant9\fast-001angle');
p9006xsrt = p9006xs(2:1901,1:9);
p9006xslf = p9006xs(2:1901,10:18);
p9009xs = xlsread('Participant9\slow-001angle');
p9009xsrt = p9009xs(2:1901,1:9);
p9009xslf = p9009xs(2:1901,10:18);

%Participant11
```

```
p11001xs = xlsread('Participant11\normal-001angle');
p11001xsrt = p11001xs(2:1901,1:9);
p11001xslf = p11001xs(2:1901,10:18);
p11006xs = xlsread('Participant11\fast-001angle');
p11006xsrt = p11006xs(2:1901,1:9);
p11006xslf = p11006xs(2:1901,10:18);
p11009xs = xlsread('Participant11\slow-001angle');
p11009xsrt = p11009xs(2:1901,1:9);
p11009xslf = p11009xs(2:1901,10:18);

%Participant12

p12001xs = xlsread('Participant12\normal-001angle');
p12001xsrt = p12001xs(2:1901,1:9);
p12001xslf = p12001xs(2:1901,10:18);
p12006xs = xlsread('Participant12\fast-001angle');
p12006xsrt = p12006xs(2:1901,1:9);
p12006xslf = p12006xs(2:1901,10:18);
p12009xs = xlsread('Participant12\slow-001angle');
p12009xsrt = p12009xs(2:1901,1:9);
p12009xslf = p12009xs(2:1901,10:18);

%Participant14

p14001xs = xlsread('Participant14\normal-001angle');
p14001xsrt = p14001xs(2:1901,1:9);
p14001xslf = p14001xs(2:1901,10:18);
p14006xs = xlsread('Participant14\fast-001angle');
p14006xsrt = p14006xs(2:1901,1:9);
p14006xslf = p14006xs(2:1901,10:18);
p14009xs = xlsread('Participant14\slow-001angle');
p14009xsrt = p14009xs(2:1901,1:9);
p14009xslf = p14009xs(2:1901,10:18);

%Participant15

p15001xs = xlsread('Participant15\normal-001angle');
p15001xsrt = p15001xs(2:1901,1:9);
p15001xslf = p15001xs(2:1901,10:18);
p15006xs = xlsread('Participant15\fast-001angle');
p15006xsrt = p15006xs(2:1901,1:9);
p15006xslf = p15006xs(2:1901,10:18);
p15009xs = xlsread('Participant15\slow-001angle');
p15009xsrt = p15009xs(2:1901,1:9);
p15009xslf = p15009xs(2:1901,10:18);

%Participant16

p16001xs = xlsread('Participant16\normal-001angle');
p16001xsrt = p16001xs(2:1901,1:9);
p16001xslf = p16001xs(2:1901,10:18);
p16006xs = xlsread('Participant16\fast-001angle');
p16006xsrt = p16006xs(2:1901,1:9);
p16006xslf = p16006xs(2:1901,10:18);
p16009xs = xlsread('Participant16\slow-001angle');
p16009xsrt = p16009xs(2:1901,1:9);
p16009xslf = p16009xs(2:1901,10:18);

%Participant17
```

```
p17001xs = xlsread('Participant17\normal-001angle');
p17001xsrt = p17001xs(2:1901,1:9);
p17001xslf = p17001xs(2:1901,10:18);
p17006xs = xlsread('Participant17\fast-001angle');
p17006xsrt = p17006xs(2:1901,1:9);
p17006xslf = p17006xs(2:1901,10:18);
p17009xs = xlsread('Participant17\slow-001angle');
p17009xsrt = p17009xs(2:1901,1:9);
p17009xslf = p17009xs(2:1901,10:18);

%Participant18

p18001xs = xlsread('Participant18\normal-001angle');
p18001xsrt = p18001xs(2:1901,1:9);
p18001xslf = p18001xs(2:1901,10:18);
p18006xs = xlsread('Participant18\fast-001angle');
p18006xsrt = p18006xs(2:1901,1:9);
p18006xslf = p18006xs(2:1901,10:18);
p18009xs = xlsread('Participant18\slow-001angle');
p18009xsrt = p18009xs(2:1901,1:9);
p18009xslf = p18009xs(2:1901,10:18);

%Participant21

p21001xs = xlsread('Participant21\normal-001angle');
p21001xsrt = p21001xs(2:1901,1:9);
p21001xslf = p21001xs(2:1901,10:18);
p21006xs = xlsread('Participant21\fast-001angle');
p21006xsrt = p21006xs(2:1901,1:9);
p21006xslf = p21006xs(2:1901,10:18);
p21009xs = xlsread('Participant21\slow-001angle');
p21009xsrt = p21009xs(2:1901,1:9);
p21009xslf = p21009xs(2:1901,10:18);

%Participant22

p22001xs = xlsread('Participant22\normal-001angle');
p22001xsrt = p22001xs(2:1901,1:9);
p22001xslf = p22001xs(2:1901,10:18);
p22006xs = xlsread('Participant22\fast-001angle');
p22006xsrt = p22006xs(2:1901,1:9);
p22006xslf = p22006xs(2:1901,10:18);
p22009xs = xlsread('Participant22\slow-001angle');
p22009xsrt = p22009xs(2:1901,1:9);
p22009xslf = p22009xs(2:1901,10:18);
p22009xxs = xlsread('Participant22\slow03-001angle');
p22009xxsrt = p22009xxs(2:1901,1:9);
p22009xxslf = p22009xxs(2:1901,10:18);

%Participant23

p23001xs = xlsread('Participant23\normal-001angle');
p23001xsrt = p23001xs(2:1901,1:9);
p23001xslf = p23001xs(2:1901,10:18);
p23006xs = xlsread('Participant23\fast01-001angle');
p23006xsrt = p23006xs(2:1901,1:9);
p23006xslf = p23006xs(2:1901,10:18);
p23009xs = xlsread('Participant23\slow01-001angle');
p23009xsrt = p23009xs(2:1901,1:9);
p23009xslf = p23009xs(2:1901,10:18);
```

```
%Participant24

p24002xs = xlsread('Participant24\normal-002angle');
p24002xsrt = p24002xs(2:1901,1:9);
p24002xslf = p24002xs(2:1901,10:18);
p24006xs = xlsread('Participant24\fast-001angle');
p24006xsrt = p24006xs(2:1901,1:9);
p24006xslf = p24006xs(2:1901,10:18);
p24009xs = xlsread('Participant24\slow-001angle');
p24009xsrt = p24009xs(2:1901,1:9);
p24009xslf = p24009xs(2:1901,10:18);

%Participant25

p25002xs = xlsread('Participant25\normal-002angle');
p25002xsrt = p25002xs(2:1901,1:9);
p25002xslf = p25002xs(2:1901,10:18);
p25006xs = xlsread('Participant25\fast-001angle');
p25006xsrt = p25006xs(2:1901,1:9);
p25006xslf = p25006xs(2:1901,10:18);
p25009xs = xlsread('Participant25\slow-001angle');
p25009xsrt = p25009xs(2:1901,1:9);
p25009xslf = p25009xs(2:1901,10:18);

%Participant28

p28001xs = xlsread('Participant28\walk001-001angle');
p28001xsrt = p28001xs(2:1901,1:9);
p28001xslf = p28001xs(2:1901,10:18);
p28006xs = xlsread('Participant28\fast001-001angle');
p28006xsrt = p28006xs(2:1901,1:9);
p28006xslf = p28006xs(2:1901,10:18);
p28009xs = xlsread('Participant28\slow001-001angle');
p28009xsrt = p28009xs(2:1901,1:9);
p28009xslf = p28009xs(2:1901,10:18);

%% upsampling

t  = 1:1:1900;
tI = 1:0.833:1900;

% x2 = spline(t,x,tI);

p2002xsrt = spline(t,p2002xsrt',tI);
p2002xslf = spline(t,p2002xslf',tI);
p2006xsrt = spline(t,p2006xsrt',tI);
p2006xslf = spline(t,p2006xslf',tI);
p2009xsrt = spline(t,p2009xsrt',tI);
p2009xslf = spline(t,p2009xslf',tI);

p3001xsrt = spline(t,p3001xsrt',tI);
p3001xslf = spline(t,p3001xslf',tI);
p3006xsrt = spline(t,p3006xsrt',tI);
p3006xslf = spline(t,p3006xslf',tI);
p3009xsrt = spline(t,p3009xsrt',tI);
p3009xslf = spline(t,p3009xslf',tI);

p4001xsrt = spline(t,p4001xsrt',tI);
p4001xslf = spline(t,p4001xslf',tI);
p4006xsrt = spline(t,p4006xsrt',tI);
p4006xslf = spline(t,p4006xslf',tI);
```

```
p4009xsrt = spline(t,p4009xsrt',tI);
p4009xslf = spline(t,p4009xslf',tI);

p5001xsrt = spline(t,p5001xsrt',tI);
p5001xslf = spline(t,p5001xslf',tI);
p5006xsrt = spline(t,p5006xsrt',tI);
p5006xslf = spline(t,p5006xslf',tI);
p5009xsrt = spline(t,p5009xsrt',tI);
p5009xslf = spline(t,p5009xslf',tI);

p6001xsrt = spline(t,p6001xsrt',tI);
p6001xslf = spline(t,p6001xslf',tI);
p6006xsrt = spline(t,p6006xsrt',tI);
p6006xslf = spline(t,p6006xslf',tI);
p6009xsrt = spline(t,p6009xsrt',tI);
p6009xslf = spline(t,p6009xslf',tI);

p8001xsrt = spline(t,p8001xsrt',tI);
p8001xslf = spline(t,p8001xslf',tI);
p8006xsrt = spline(t,p8006xsrt',tI);
p8006xslf = spline(t,p8006xslf',tI);
p8009xsrt = spline(t,p8009xsrt',tI);
p8009xslf = spline(t,p8009xslf',tI);

p9001xsrt = spline(t,p9001xsrt',tI);
p9001xslf = spline(t,p9001xslf',tI);
p9006xsrt = spline(t,p9006xsrt',tI);
p9006xslf = spline(t,p9006xslf',tI);
p9009xsrt = spline(t,p9009xsrt',tI);
p9009xslf = spline(t,p9009xslf',tI);

p11001xsrt = spline(t,p11001xsrt',tI);
p11001xslf = spline(t,p11001xslf',tI);
p11006xsrt = spline(t,p11006xsrt',tI);
p11006xslf = spline(t,p11006xslf',tI);
p11009xsrt = spline(t,p11009xsrt',tI);
p11009xslf = spline(t,p11009xslf',tI);

p12001xsrt = spline(t,p12001xsrt',tI);
p12001xslf = spline(t,p12001xslf',tI);
p12006xsrt = spline(t,p12006xsrt',tI);
p12006xslf = spline(t,p12006xslf',tI);
p12009xsrt = spline(t,p12009xsrt',tI);
p12009xslf = spline(t,p12009xslf',tI);

p14001xsrt = spline(t,p14001xsrt',tI);
p14001xslf = spline(t,p14001xslf',tI);
p14006xsrt = spline(t,p14006xsrt',tI);
p14006xslf = spline(t,p14006xslf',tI);
p14009xsrt = spline(t,p14009xsrt',tI);
p14009xslf = spline(t,p14009xslf',tI);

p15001xsrt = spline(t,p15001xsrt',tI);
p15001xslf = spline(t,p15001xslf',tI);
p15006xsrt = spline(t,p15006xsrt',tI);
p15006xslf = spline(t,p15006xslf',tI);
p15009xsrt = spline(t,p15009xsrt',tI);
p15009xslf = spline(t,p15009xslf',tI);

p16001xsrt = spline(t,p16001xsrt',tI);
p16001xslf = spline(t,p16001xslf',tI);
```

```
p16006xsrt = spline(t,p16006xsrt',tI);
p16006xslf = spline(t,p16006xslf',tI);
p16009xsrt = spline(t,p16009xsrt',tI);
p16009xslf = spline(t,p16009xslf',tI);

p17001xsrt = spline(t,p17001xsrt',tI);
p17001xslf = spline(t,p17001xslf',tI);
p17006xsrt = spline(t,p17006xsrt',tI);
p17006xslf = spline(t,p17006xslf',tI);
p17009xsrt = spline(t,p17009xsrt',tI);
p17009xslf = spline(t,p17009xslf',tI);

p18001xsrt = spline(t,p18001xsrt',tI);
p18001xslf = spline(t,p18001xslf',tI);
p18006xsrt = spline(t,p18006xsrt',tI);
p18006xslf = spline(t,p18006xslf',tI);
p18009xsrt = spline(t,p18009xsrt',tI);
p18009xslf = spline(t,p18009xslf',tI);

p21001xsrt = spline(t,p21001xsrt',tI);
p21001xslf = spline(t,p21001xslf',tI);
p21006xsrt = spline(t,p21006xsrt',tI);
p21006xslf = spline(t,p21006xslf',tI);
p21009xsrt = spline(t,p21009xsrt',tI);
p21009xslf = spline(t,p21009xslf',tI);

p22001xsrt = spline(t,p22001xsrt',tI);
p22001xslf = spline(t,p22001xslf',tI);
p22006xsrt = spline(t,p22006xsrt',tI);
p22006xslf = spline(t,p22006xslf',tI);
p22009xsrt = spline(t,p22009xsrt',tI);
p22009xslf = spline(t,p22009xslf',tI);
p22009xxsrt = spline(t,p22009xxsrt',tI);
p22009xxslf = spline(t,p22009xxslf',tI);

p23001xsrt = spline(t,p23001xsrt',tI);
p23001xslf = spline(t,p23001xslf',tI);
p23006xsrt = spline(t,p23006xsrt',tI);
p23006xslf = spline(t,p23006xslf',tI);
p23009xsrt = spline(t,p23009xsrt',tI);
p23009xslf = spline(t,p23009xslf',tI);

p24002xsrt = spline(t,p24002xsrt',tI);
p24002xslf = spline(t,p24002xslf',tI);
p24006xsrt = spline(t,p24006xsrt',tI);
p24006xslf = spline(t,p24006xslf',tI);
p24009xsrt = spline(t,p24009xsrt',tI);
p24009xslf = spline(t,p24009xslf',tI);

p25002xsrt = spline(t,p25002xsrt',tI);
p25002xslf = spline(t,p25002xslf',tI);
p25006xsrt = spline(t,p25006xsrt',tI);
p25006xslf = spline(t,p25006xslf',tI);
p25009xsrt = spline(t,p25009xsrt',tI);
p25009xslf = spline(t,p25009xslf',tI);
p28001xsrt = spline(t,p28001xsrt',tI);
p28001xslf = spline(t,p28001xslf',tI);
p28006xsrt = spline(t,p28006xsrt',tI);
p28006xslf = spline(t,p28006xslf',tI);
p28009xsrt = spline(t,p28009xsrt',tI);
p28009xslf = spline(t,p28009xslf',tI);
```

Appendix 7. A Matlab script used for leave-one-out cross validation

```
%%creating matrix for shuffle
%%% from xtp
%row 1-9 = xnormal, row 10 = tnormal
%row 11-19 = xfast, row 20 = tfast
%row 21-29 = xslow, row 30 = tslow
%saved as 'shuffle_2.mat'

load 'jointmoments_shuffle.mat';
load 'xsensdataextraction_shuffle.mat';

%1
xnormalp28 = [p2002xsrt' p3001xsrt' p4001xsrt' p5001xsrt' p6001xsrt'
p8001xsrt' p9001xsrt' p11001xsrt' p12001xsrt' p14001xsrt'
p16001xsrt' p17001xsrt' p18001xsrt' p21001xsrt' p22001xsrt'
p23001xsrt' p24002xsrt' p25002xsrt' p28001xsrt'];

xfastp28 = [p2006xsrt' p3006xsrt' p4006xsrt' p5006xsrt' p6006xsrt'
p8006xsrt' p9006xsrt' p11006xsrt' p12006xsrt' p14006xsrt'
p16006xsrt' p17006xsrt' p18006xsrt' p21006xsrt' p22006xsrt'
p23006xsrt' p24006xsrt' p25006xsrt' p28006xsrt'];

xslowp28 = [p2009xsrt' p3009xsrt' p4009xsrt' p5009xsrt' p6009xsrt'
p8009xsrt' p9009xsrt' p11009xsrt' p12009xsrt' p14009xsrt'
p16009xsrt' p17009xsrt' p18009xsrt' p21009xsrt' p22009xsrt'
p23009xsrt' p24009xsrt' p25009xsrt' p28009xsrt'];

tnormalp28 = [p2001' p3001' p4001' p5001' p6001' p8001' p9001'
p11001' p12001' p14001' p16001' p17001' p18001' p21001' p22001'
p23001' p24002' p25001' p28001'];

tfastp28 = [p2006' p3006' p4006' p5006' p6006' p8006' p9006' p11006'
p12006' p14006' p16006' p17006' p18006' p21006' p22006' p23006'
p24006' p25006' p28006'];

tslowp28 = [p2009' p3009' p4009' p5009' p6009' p8009' p9009' p11009'
p12009' p14009' p16009' p17009' p18009' p21009' p22009' p23009'
p24009' p25009' p28009'];

xtp28 = [xnormalp28; tnormalp28; xfastp28; tfastp28; xslowp28;
tslowp28];

%2
xnormalp25 = [p2002xsrt' p3001xsrt' p4001xsrt' p5001xsrt' p6001xsrt'
p8001xsrt' p9001xsrt' p11001xsrt' p12001xsrt' p14001xsrt'
p16001xsrt' p17001xsrt' p18001xsrt' p21001xsrt' p22001xsrt'
p23001xsrt' p24002xsrt' p28001xsrt' p25002xsrt'];

xfastp25 = [p2006xsrt' p3006xsrt' p4006xsrt' p5006xsrt' p6006xsrt'
p8006xsrt' p9006xsrt' p11006xsrt' p12006xsrt' p14006xsrt'
p16006xsrt' p17006xsrt' p18006xsrt' p21006xsrt' p22006xsrt'
p23006xsrt' p24006xsrt' p28006xsrt' p25006xsrt'];

xslowp25 = [p2009xsrt' p3009xsrt' p4009xsrt' p5009xsrt' p6009xsrt'
p8009xsrt' p9009xsrt' p11009xsrt' p12009xsrt' p14009xsrt'
p16009xsrt' p17009xsrt' p18009xsrt' p21009xsrt' p22009xsrt'
p23009xsrt' p24009xsrt' p28009xsrt' p25009xsrt'];
```

```
tnormalp25 = [p2001' p3001' p4001' p5001' p6001' p8001' p9001'
p11001' p12001' p14001' p16001' p17001' p18001' p21001' p22001'
p23001' p24002' p28001' p25001'];

tfastp25 = [p2006' p3006' p4006' p5006' p6006' p8006' p9006' p11006'
p12006' p14006' p16006' p17006' p18006' p21006' p22006' p23006'
p24006' p28006' p25006'];

tslowp25 = [p2009' p3009' p4009' p5009' p6009' p8009' p9009' p11009'
p12009' p14009' p16009' p17009' p18009' p21009' p22009' p23009'
p24009' p28009' p25009'];

xtp25 = [xnormalp25; tnormalp25; xfastp25; tfastp25; xslowp25;
tslowp25];

%3
xnormalp24 = [p2002xsrt' p3001xsrt' p4001xsrt' p5001xsrt' p6001xsrt'
p8001xsrt' p9001xsrt' p11001xsrt' p12001xsrt' p14001xsrt'
p16001xsrt' p17001xsrt' p18001xsrt' p21001xsrt' p22001xsrt'
p23001xsrt' p25002xsrt' p28001xsrt' p24002xsrt'];

xfastp24 = [p2006xsrt' p3006xsrt' p4006xsrt' p5006xsrt' p6006xsrt'
p8006xsrt' p9006xsrt' p11006xsrt' p12006xsrt' p14006xsrt'
p16006xsrt' p17006xsrt' p18006xsrt' p21006xsrt' p22006xsrt'
p23006xsrt' p25006xsrt' p28006xsrt' p24006xsrt'];

xslowp24 = [p2009xsrt' p3009xsrt' p4009xsrt' p5009xsrt' p6009xsrt'
p8009xsrt' p9009xsrt' p11009xsrt' p12009xsrt' p14009xsrt'
p16009xsrt' p17009xsrt' p18009xsrt' p21009xsrt' p22009xsrt'
p23009xsrt' p25009xsrt' p28009xsrt' p24009xsrt'];

tnormalp24 = [p2001' p3001' p4001' p5001' p6001' p8001' p9001'
p11001' p12001' p14001' p16001' p17001' p18001' p21001' p22001'
p23001' p25001' p28001' p24002'];

tfastp24 = [p2006' p3006' p4006' p5006' p6006' p8006' p9006' p11006'
p12006' p14006' p16006' p17006' p18006' p21006' p22006' p23006'
p25006' p28006' p24006'];

tslowp24 = [p2009' p3009' p4009' p5009' p6009' p8009' p9009' p11009'
p12009' p14009' p16009' p17009' p18009' p21009' p22009' p23009'
p25009' p28009' p24009'];

xtp24 = [xnormalp24; tnormalp24; xfastp24; tfastp24; xslowp24;
tslowp24];

%4
xnormalp23 = [p2002xsrt' p3001xsrt' p4001xsrt' p5001xsrt' p6001xsrt'
p8001xsrt' p9001xsrt' p11001xsrt' p12001xsrt' p14001xsrt'
p16001xsrt' p17001xsrt' p18001xsrt' p21001xsrt' p22001xsrt'
p24002xsrt' p25002xsrt' p28001xsrt' p23001xsrt'];

xfastp23 = [p2006xsrt' p3006xsrt' p4006xsrt' p5006xsrt' p6006xsrt'
p8006xsrt' p9006xsrt' p11006xsrt' p12006xsrt' p14006xsrt'
p16006xsrt' p17006xsrt' p18006xsrt' p21006xsrt' p22006xsrt'
p24006xsrt' p25006xsrt' p28006xsrt' p23006xsrt'];

xslowp23 = [p2009xsrt' p3009xsrt' p4009xsrt' p5009xsrt' p6009xsrt'
p8009xsrt' p9009xsrt' p11009xsrt' p12009xsrt' p14009xsrt'
```

```
p16009xsrt' p17009xsrt' p18009xsrt' p21009xsrt' p22009xsrt'
p24009xsrt' p25009xsrt' p28009xsrt' p23009xsrt'];

tnormalp23 = [p2001' p3001' p4001' p5001' p6001' p8001' p9001'
p11001' p12001' p14001' p16001' p17001' p18001' p21001' p22001'
p24002' p25001' p28001' p23001'];

tfastp23 = [p2006' p3006' p4006' p5006' p6006' p8006' p9006' p11006'
p12006' p14006' p16006' p17006' p18006' p21006' p22006' p24006'
p25006' p28006' p23006'];

tslowp23 = [p2009' p3009' p4009' p5009' p6009' p8009' p9009' p11009'
p12009' p14009' p16009' p17009' p18009' p21009' p22009' p24009'
p25009' p28009' p23009'];

xtp23 = [xnormalp23; tnormalp23; xfastp23; tfastp23; xslowp23;
tslowp23];

%5
xnormalp22 = [p2002xsrt' p3001xsrt' p4001xsrt' p5001xsrt' p6001xsrt'
p8001xsrt' p9001xsrt' p11001xsrt' p12001xsrt' p14001xsrt'
p16001xsrt' p17001xsrt' p18001xsrt' p21001xsrt' p23001xsrt'
p24002xsrt' p25002xsrt' p28001xsrt' p22001xsrt'];

xfastp22 = [p2006xsrt' p3006xsrt' p4006xsrt' p5006xsrt' p6006xsrt'
p8006xsrt' p9006xsrt' p11006xsrt' p12006xsrt' p14006xsrt'
p16006xsrt' p17006xsrt' p18006xsrt' p21006xsrt' p23006xsrt'
p24006xsrt' p25006xsrt' p28006xsrt' p22006xsrt'];

xslowp22 = [p2009xsrt' p3009xsrt' p4009xsrt' p5009xsrt' p6009xsrt'
p8009xsrt' p9009xsrt' p11009xsrt' p12009xsrt' p14009xsrt'
p16009xsrt' p17009xsrt' p18009xsrt' p21009xsrt' p23009xsrt'
p24009xsrt' p25009xsrt' p28009xsrt' p22009xsrt'];

tnormalp22 = [p2001' p3001' p4001' p5001' p6001' p8001' p9001'
p11001' p12001' p14001' p16001' p17001' p18001' p21001' p23001'
p24002' p25001' p28001' p22001'];

tfastp22 = [p2006' p3006' p4006' p5006' p6006' p8006' p9006' p11006'
p12006' p14006' p16006' p17006' p18006' p21006' p23006' p24006'
p25006' p28006' p22006'];

tslowp22 = [p2009' p3009' p4009' p5009' p6009' p8009' p9009' p11009'
p12009' p14009' p16009' p17009' p18009' p21009' p23009' p24009'
p25009' p28009' p22009'];

xtp22 = [xnormalp22; tnormalp22; xfastp22; tfastp22; xslowp22;
tslowp22];

%6
xnormalp21 = [p2002xsrt' p3001xsrt' p4001xsrt' p5001xsrt' p6001xsrt'
p8001xsrt' p9001xsrt' p11001xsrt' p12001xsrt' p14001xsrt'
p16001xsrt' p17001xsrt' p18001xsrt' p22001xsrt' p23001xsrt'
p24002xsrt' p25002xsrt' p28001xsrt' p21001xsrt'];

xfastp21 = [p2006xsrt' p3006xsrt' p4006xsrt' p5006xsrt' p6006xsrt'
p8006xsrt' p9006xsrt' p11006xsrt' p12006xsrt' p14006xsrt'
p16006xsrt' p17006xsrt' p18006xsrt' p22006xsrt' p23006xsrt'
p24006xsrt' p25006xsrt' p28006xsrt' p21006xsrt'];
```

```
xslowp21 = [p2009xsrt' p3009xsrt' p4009xsrt' p5009xsrt' p6009xsrt'
p8009xsrt' p9009xsrt' p11009xsrt' p12009xsrt' p14009xsrt'
p16009xsrt' p17009xsrt' p18009xsrt' p22009xsrt' p23009xsrt'
p24009xsrt' p25009xsrt' p28009xsrt' p21009xsrt'];

tnormalp21 = [p2001' p3001' p4001' p5001' p6001' p8001' p9001'
p11001' p12001' p14001' p16001' p17001' p18001' p22001' p23001'
p24002' p25001' p28001' p21001'];

tfastp21 = [p2006' p3006' p4006' p5006' p6006' p8006' p9006' p11006'
p12006' p14006' p16006' p17006' p18006' p22006' p23006' p24006'
p25006' p28006' p21006'];

tslowp21 = [p2009' p3009' p4009' p5009' p6009' p8009' p9009' p11009'
p12009' p14009' p16009' p17009' p18009' p22009' p23009' p24009'
p25009' p28009' p21009'];

xtp21 = [xnormalp21; tnormalp21; xfastp21; tfastp21; xslowp21;
tslowp21];

%7
xnormalp18 = [p2002xsrt' p3001xsrt' p4001xsrt' p5001xsrt' p6001xsrt'
p8001xsrt' p9001xsrt' p11001xsrt' p12001xsrt' p14001xsrt'
p16001xsrt' p17001xsrt' p21001xsrt' p22001xsrt' p23001xsrt'
p24002xsrt' p25002xsrt' p28001xsrt' p18001xsrt'];

xfastp18 = [p2006xsrt' p3006xsrt' p4006xsrt' p5006xsrt' p6006xsrt'
p8006xsrt' p9006xsrt' p11006xsrt' p12006xsrt' p14006xsrt'
p16006xsrt' p17006xsrt' p21006xsrt' p22006xsrt' p23006xsrt'
p24006xsrt' p25006xsrt' p28006xsrt' p18006xsrt'];

xslowp18 = [p2009xsrt' p3009xsrt' p4009xsrt' p5009xsrt' p6009xsrt'
p8009xsrt' p9009xsrt' p11009xsrt' p12009xsrt' p14009xsrt'
p16009xsrt' p17009xsrt' p21009xsrt' p22009xsrt' p23009xsrt'
p24009xsrt' p25009xsrt' p28009xsrt' p18009xsrt'];

tnormalp18 = [p2001' p3001' p4001' p5001' p6001' p8001' p9001'
p11001' p12001' p14001' p16001' p17001' p21001' p22001' p23001'
p24002' p25001' p28001' p18001'];

tfastp18 = [p2006' p3006' p4006' p5006' p6006' p8006' p9006' p11006'
p12006' p14006' p16006' p17006' p21006' p22006' p23006' p24006'
p25006' p28006' p18006'];

tslowp18 = [p2009' p3009' p4009' p5009' p6009' p8009' p9009' p11009'
p12009' p14009' p16009' p17009' p21009' p22009' p23009' p24009'
p25009' p28009' p18009'];

xtp18 = [xnormalp18; tnormalp18; xfastp18; tfastp18; xslowp18;
tslowp18];

%8
xnormalp17 = [p2002xsrt' p3001xsrt' p4001xsrt' p5001xsrt' p6001xsrt'
p8001xsrt' p9001xsrt' p11001xsrt' p12001xsrt' p14001xsrt'
p16001xsrt' p18001xsrt' p21001xsrt' p22001xsrt' p23001xsrt'
p24002xsrt' p25002xsrt' p28001xsrt' p17001xsrt'];

xfastp17 = [p2006xsrt' p3006xsrt' p4006xsrt' p5006xsrt' p6006xsrt'
p8006xsrt' p9006xsrt' p11006xsrt' p12006xsrt' p14006xsrt'
```

```
p16006xsrt' p18006xsrt' p21006xsrt' p22006xsrt' p23006xsrt'
p24006xsrt' p25006xsrt' p28006xsrt' p17006xsrt'];

xslowp17 = [p2009xsrt' p3009xsrt' p4009xsrt' p5009xsrt' p6009xsrt'
p8009xsrt' p9009xsrt' p11009xsrt' p12009xsrt' p14009xsrt'
p16009xsrt' p18009xsrt' p21009xsrt' p22009xsrt' p23009xsrt'
p24009xsrt' p25009xsrt' p28009xsrt' p17009xsrt'];

tnormalp17 = [p2001' p3001' p4001' p5001' p6001' p8001' p9001'
p11001' p12001' p14001' p16001' p18001' p21001' p22001' p23001'
p24002' p25001' p28001' p17001'];

tfastp17 = [p2006' p3006' p4006' p5006' p6006' p8006' p9006' p11006'
p12006' p14006' p16006' p18006' p21006' p22006' p23006' p24006'
p25006' p28006' p17006'];

tslowp17 = [p2009' p3009' p4009' p5009' p6009' p8009' p9009' p11009'
p12009' p14009' p16009' p18009' p21009' p22009' p23009' p24009'
p25009' p28009' p17009'];

xtp17 = [xnormalp17; tnormalp17; xfastp17; tfastp17; xslowp17;
tslowp17];

%9
xnormalp16 = [p2002xsrt' p3001xsrt' p4001xsrt' p5001xsrt' p6001xsrt'
p8001xsrt' p9001xsrt' p11001xsrt' p12001xsrt' p14001xsrt'
p17001xsrt' p18001xsrt' p21001xsrt' p22001xsrt' p23001xsrt'
p24002xsrt' p25002xsrt' p28001xsrt' p16001xsrt'];

xfastp16 = [p2006xsrt' p3006xsrt' p4006xsrt' p5006xsrt' p6006xsrt'
p8006xsrt' p9006xsrt' p11006xsrt' p12006xsrt' p14006xsrt'
p17006xsrt' p18006xsrt' p21006xsrt' p22006xsrt' p23006xsrt'
p24006xsrt' p25006xsrt' p28006xsrt' p16006xsrt'];

xslowp16 = [p2009xsrt' p3009xsrt' p4009xsrt' p5009xsrt' p6009xsrt'
p8009xsrt' p9009xsrt' p11009xsrt' p12009xsrt' p14009xsrt'
p17009xsrt' p18009xsrt' p21009xsrt' p22009xsrt' p23009xsrt'
p24009xsrt' p25009xsrt' p28009xsrt' p16009xsrt'];

tnormalp16 = [p2001' p3001' p4001' p5001' p6001' p8001' p9001'
p11001' p12001' p14001' p17001' p18001' p21001' p22001' p23001'
p24002' p25001' p28001' p16001'];

tfastp16 = [p2006' p3006' p4006' p5006' p6006' p8006' p9006' p11006'
p12006' p14006' p17006' p18006' p21006' p22006' p23006' p24006'
p25006' p28006' p16006'];

tslowp16 = [p2009' p3009' p4009' p5009' p6009' p8009' p9009' p11009'
p12009' p14009' p17009' p18009' p21009' p22009' p23009' p24009'
p25009' p28009' p16009'];

xtp16 = [xnormalp16; tnormalp16; xfastp16; tfastp16; xslowp16;
tslowp16];

%11
xnormalp14 = [p2002xsrt' p3001xsrt' p4001xsrt' p5001xsrt' p6001xsrt'
p8001xsrt' p9001xsrt' p11001xsrt' p12001xsrt' p16001xsrt'
p17001xsrt' p18001xsrt' p21001xsrt' p22001xsrt' p23001xsrt'
p24002xsrt' p25002xsrt' p28001xsrt' p14001xsrt'];
```

```
xfastp14 = [p2006xsrt' p3006xsrt' p4006xsrt' p5006xsrt' p6006xsrt'
p8006xsrt' p9006xsrt' p11006xsrt' p12006xsrt' p16006xsrt'
p17006xsrt' p18006xsrt' p21006xsrt' p22006xsrt' p23006xsrt'
p24006xsrt' p25006xsrt' p28006xsrt' p14006xsrt'];

xslowp14 = [p2009xsrt' p3009xsrt' p4009xsrt' p5009xsrt' p6009xsrt'
p8009xsrt' p9009xsrt' p11009xsrt' p12009xsrt' p16009xsrt'
p17009xsrt' p18009xsrt' p21009xsrt' p22009xsrt' p23009xsrt'
p24009xsrt' p25009xsrt' p28009xsrt' p14009xsrt'];

tnormalp14 = [p2001' p3001' p4001' p5001' p6001' p8001' p9001'
p11001' p12001' p16001' p17001' p18001' p21001' p22001' p23001'
p24002' p25001' p28001' p14001'];

tfastp14 = [p2006' p3006' p4006' p5006' p6006' p8006' p9006' p11006'
p12006' p16006' p17006' p18006' p21006' p22006' p23006' p24006'
p25006' p28006' p14006'];

tslowp14 = [p2009' p3009' p4009' p5009' p6009' p8009' p9009' p11009'
p12009' p16009' p17009' p18009' p21009' p22009' p23009' p24009'
p25009' p28009' p14009'];

xtp14 = [xnormalp14; tnormalp14; xfastp14; tfastp14; xslowp14;
tslowp14];

%12
xnormalp12 = [p2002xsrt' p3001xsrt' p4001xsrt' p5001xsrt' p6001xsrt'
p8001xsrt' p9001xsrt' p11001xsrt' p14001xsrt' p16001xsrt'
p17001xsrt' p18001xsrt' p21001xsrt' p22001xsrt' p23001xsrt'
p24002xsrt' p25002xsrt' p28001xsrt' p12001xsrt'];

xfastp12 = [p2006xsrt' p3006xsrt' p4006xsrt' p5006xsrt' p6006xsrt'
p8006xsrt' p9006xsrt' p11006xsrt' p14006xsrt' p16006xsrt'
p17006xsrt' p18006xsrt' p21006xsrt' p22006xsrt' p23006xsrt'
p24006xsrt' p25006xsrt' p28006xsrt' p12006xsrt'];

xslowp12 = [p2009xsrt' p3009xsrt' p4009xsrt' p5009xsrt' p6009xsrt'
p8009xsrt' p9009xsrt' p11009xsrt' p14009xsrt' p16009xsrt'
p17009xsrt' p18009xsrt' p21009xsrt' p22009xsrt' p23009xsrt'
p24009xsrt' p25009xsrt' p28009xsrt' p12009xsrt'];

tnormalp12 = [p2001' p3001' p4001' p5001' p6001' p8001' p9001'
p11001' p14001' p16001' p17001' p18001' p21001' p22001' p23001'
p24002' p25001' p28001' p12001'];

tfastp12 = [p2006' p3006' p4006' p5006' p6006' p8006' p9006' p11006'
p14006' p16006' p17006' p18006' p21006' p22006' p23006' p24006'
p25006' p28006' p12006'];

tslowp12 = [p2009' p3009' p4009' p5009' p6009' p8009' p9009' p11009'
p14009' p16009' p17009' p18009' p21009' p22009' p23009' p24009'
p25009' p28009' p12009'];

xtp12 = [xnormalp12; tnormalp12; xfastp12; tfastp12; xslowp12;
tslowp12];

%13
xnormalp11 = [p2002xsrt' p3001xsrt' p4001xsrt' p5001xsrt' p6001xsrt'
p8001xsrt' p9001xsrt' p12001xsrt' p14001xsrt' p16001xsrt'
```

```
p17001xsrt' p18001xsrt' p21001xsrt' p22001xsrt' p23001xsrt'
p24002xsrt' p25002xsrt' p28001xsrt' p11001xsrt'];

xfastp11 = [p2006xsrt' p3006xsrt' p4006xsrt' p5006xsrt' p6006xsrt'
p8006xsrt' p9006xsrt' p12006xsrt' p14006xsrt' p16006xsrt'
p17006xsrt' p18006xsrt' p21006xsrt' p22006xsrt' p23006xsrt'
p24006xsrt' p25006xsrt' p28006xsrt' p11006xsrt'];

xslowp11 = [p2009xsrt' p3009xsrt' p4009xsrt' p5009xsrt' p6009xsrt'
p8009xsrt' p9009xsrt' p12009xsrt' p14009xsrt' p16009xsrt'
p17009xsrt' p18009xsrt' p21009xsrt' p22009xsrt' p23009xsrt'
p24009xsrt' p25009xsrt' p28009xsrt' p11009xsrt'];

tnormalp11 = [p2001' p3001' p4001' p5001' p6001' p8001' p9001'
p12001' p14001' p16001' p17001' p18001' p21001' p22001' p23001'
p24002' p25001' p28001' p11001'];

tfastp11 = [p2006' p3006' p4006' p5006' p6006' p8006' p9006' p12006'
p14006' p16006' p17006' p18006' p21006' p22006' p23006' p24006'
p25006' p28006' p11006'];

tslowp11 = [p2009' p3009' p4009' p5009' p6009' p8009' p9009' p12009'
p14009' p16009' p17009' p18009' p21009' p22009' p23009' p24009'
p25009' p28009' p11009'];

xtp11 = [xnormalp11; tnormalp11; xfastp11; tfastp11; xslowp11;
tslowp11];

%14
xnormalp9 = [p2002xsrt' p3001xsrt' p4001xsrt' p5001xsrt' p6001xsrt'
p8001xsrt' p11001xsrt' p12001xsrt' p14001xsrt' p16001xsrt'
p17001xsrt' p18001xsrt' p21001xsrt' p22001xsrt' p23001xsrt'
p24002xsrt' p25002xsrt' p28001xsrt' p9001xsrt'];

xfastp9 = [p2006xsrt' p3006xsrt' p4006xsrt' p5006xsrt' p6006xsrt'
p8006xsrt' p11006xsrt' p12006xsrt' p14006xsrt' p16006xsrt'
p17006xsrt' p18006xsrt' p21006xsrt' p22006xsrt' p23006xsrt'
p24006xsrt' p25006xsrt' p28006xsrt' p9006xsrt'];

xslowp9 = [p2009xsrt' p3009xsrt' p4009xsrt' p5009xsrt' p6009xsrt'
p8009xsrt' p11009xsrt' p12009xsrt' p14009xsrt' p16009xsrt'
p17009xsrt' p18009xsrt' p21009xsrt' p22009xsrt' p23009xsrt'
p24009xsrt' p25009xsrt' p28009xsrt' p9009xsrt'];

tnormalp9 = [p2001' p3001' p4001' p5001' p6001' p8001' p11001'
p12001' p14001' p16001' p17001' p18001' p21001' p22001' p23001'
p24002' p25001' p28001' p9001'];

tfastp9 = [p2006' p3006' p4006' p5006' p6006' p8006' p11006' p12006'
p14006' p16006' p17006' p18006' p21006' p22006' p23006' p24006'
p25006' p28006' p9006'];

tslowp9 = [p2009' p3009' p4009' p5009' p6009' p8009' p11009' p12009'
p14009' p16009' p17009' p18009' p21009' p22009' p23009' p24009'
p25009' p28009' p9009'];

xtp9 = [xnormalp9; tnormalp9; xfastp9; tfastp9; xslowp9; tslowp9];

%15
```

```
xnormalp8 = [p2002xsrt' p3001xsrt' p4001xsrt' p5001xsrt' p6001xsrt'
p9001xsrt' p11001xsrt' p12001xsrt' p14001xsrt' p16001xsrt'
p17001xsrt' p18001xsrt' p21001xsrt' p22001xsrt' p23001xsrt'
p24002xsrt' p25002xsrt' p28001xsrt' p8001xsrt'];

xfastp8 = [p2006xsrt' p3006xsrt' p4006xsrt' p5006xsrt' p6006xsrt'
p9006xsrt' p11006xsrt' p12006xsrt' p14006xsrt' p16006xsrt'
p17006xsrt' p18006xsrt' p21006xsrt' p22006xsrt' p23006xsrt'
p24006xsrt' p25006xsrt' p28006xsrt' p8006xsrt'];

xslowp8 = [p2009xsrt' p3009xsrt' p4009xsrt' p5009xsrt' p6009xsrt'
p9009xsrt' p11009xsrt' p12009xsrt' p14009xsrt' p16009xsrt'
p17009xsrt' p18009xsrt' p21009xsrt' p22009xsrt' p23009xsrt'
p24009xsrt' p25009xsrt' p28009xsrt' p8009xsrt'];

tnormalp8 = [p2001' p3001' p4001' p5001' p6001' p9001' p11001'
p12001' p14001' p16001' p17001' p18001' p21001' p22001' p23001'
p24002' p25001' p28001' p8001'];

tfastp8 = [p2006' p3006' p4006' p5006' p6006' p9006' p11006' p12006'
p14006' p16006' p17006' p18006' p21006' p22006' p23006' p24006'
p25006' p28006' p8006'];

tslowp8 = [p2009' p3009' p4009' p5009' p6009' p9009' p11009' p12009'
p14009' p16009' p17009' p18009' p21009' p22009' p23009' p24009'
p25009' p28009' p8009'];

xtp8 = [xnormalp8; tnormalp8; xfastp8; tfastp8; xslowp8; tslowp8];

%16
xnormalp6 = [p2002xsrt' p3001xsrt' p4001xsrt' p5001xsrt' p8001xsrt'
p9001xsrt' p11001xsrt' p12001xsrt' p14001xsrt' p16001xsrt'
p17001xsrt' p18001xsrt' p21001xsrt' p22001xsrt' p23001xsrt'
p24002xsrt' p25002xsrt' p28001xsrt' p6001xsrt'];

xfastp6 = [p2006xsrt' p3006xsrt' p4006xsrt' p5006xsrt' p8006xsrt'
p9006xsrt' p11006xsrt' p12006xsrt' p14006xsrt' p16006xsrt'
p17006xsrt' p18006xsrt' p21006xsrt' p22006xsrt' p23006xsrt'
p24006xsrt' p25006xsrt' p28006xsrt' p6006xsrt'];

xslowp6 = [p2009xsrt' p3009xsrt' p4009xsrt' p5009xsrt' p8009xsrt'
p9009xsrt' p11009xsrt' p12009xsrt' p14009xsrt' p16009xsrt'
p17009xsrt' p18009xsrt' p21009xsrt' p22009xsrt' p23009xsrt'
p24009xsrt' p25009xsrt' p28009xsrt' p6009xsrt'];

tnormalp6 = [p2001' p3001' p4001' p5001' p8001' p9001' p11001'
p12001' p14001' p16001' p17001' p18001' p21001' p22001' p23001'
p24002' p25001' p28001' p6001'];

tfastp6 = [p2006' p3006' p4006' p5006' p8006' p9006' p11006' p12006'
p14006' p16006' p17006' p18006' p21006' p22006' p23006' p24006'
p25006' p28006' p6006'];

tslowp6 = [p2009' p3009' p4009' p5009' p8009' p9009' p11009' p12009'
p14009' p16009' p17009' p18009' p21009' p22009' p23009' p24009'
p25009' p28009' p6009'];

xtp6 = [xnormalp6; tnormalp6; xfastp6; tfastp6; xslowp6; tslowp6];

%17
```

```
xnormalp5 = [p2002xsrt' p3001xsrt' p4001xsrt' p6001xsrt' p8001xsrt'
p9001xsrt' p11001xsrt' p12001xsrt' p14001xsrt' p16001xsrt'
p17001xsrt' p18001xsrt' p21001xsrt' p22001xsrt' p23001xsrt'
p24002xsrt' p25002xsrt' p28001xsrt' p5001xsrt'];

xfastp5 = [p2006xsrt' p3006xsrt' p4006xsrt' p6006xsrt' p8006xsrt'
p9006xsrt' p11006xsrt' p12006xsrt' p14006xsrt' p16006xsrt'
p17006xsrt' p18006xsrt' p21006xsrt' p22006xsrt' p23006xsrt'
p24006xsrt' p25006xsrt' p28006xsrt' p5006xsrt'];

xslowp5 = [p2009xsrt' p3009xsrt' p4009xsrt' p6009xsrt' p8009xsrt'
p9009xsrt' p11009xsrt' p12009xsrt' p14009xsrt' p16009xsrt'
p17009xsrt' p18009xsrt' p21009xsrt' p22009xsrt' p23009xsrt'
p24009xsrt' p25009xsrt' p28009xsrt' p5009xsrt'];

tnormalp5 = [p2001' p3001' p4001' p6001' p8001' p9001' p11001'
p12001' p14001' p16001' p17001' p18001' p21001' p22001' p23001'
p24002' p25001' p28001' p5001'];

tfastp5 = [p2006' p3006' p4006' p6006' p8006' p9006' p11006' p12006'
p14006' p16006' p17006' p18006' p21006' p22006' p23006' p24006'
p25006' p28006' p5006'];

tslowp5 = [p2009' p3009' p4009' p6009' p8009' p9009' p11009' p12009'
p14009' p16009' p17009' p18009' p21009' p22009' p23009' p24009'
p25009' p28009' p5009'];

xtp5 = [xnormalp5; tnormalp5; xfastp5; tfastp5; xslowp5; tslowp5];

%18
xnormalp4 = [p2002xsrt' p3001xsrt' p5001xsrt' p6001xsrt' p8001xsrt'
p9001xsrt' p11001xsrt' p12001xsrt' p14001xsrt' p16001xsrt'
p17001xsrt' p18001xsrt' p21001xsrt' p22001xsrt' p23001xsrt'
p24002xsrt' p25002xsrt' p28001xsrt' p4001xsrt'];

xfastp4 = [p2006xsrt' p3006xsrt' p5006xsrt' p6006xsrt' p8006xsrt'
p9006xsrt' p11006xsrt' p12006xsrt' p14006xsrt' p16006xsrt'
p17006xsrt' p18006xsrt' p21006xsrt' p22006xsrt' p23006xsrt'
p24006xsrt' p25006xsrt' p28006xsrt' p4006xsrt'];

xslowp4 = [p2009xsrt' p3009xsrt' p5009xsrt' p6009xsrt' p8009xsrt'
p9009xsrt' p11009xsrt' p12009xsrt' p14009xsrt' p16009xsrt'
p17009xsrt' p18009xsrt' p21009xsrt' p22009xsrt' p23009xsrt'
p24009xsrt' p25009xsrt' p28009xsrt' p4009xsrt'];

tnormalp4 = [p2001' p3001' p5001' p6001' p8001' p9001' p11001'
p12001' p14001' p16001' p17001' p18001' p21001' p22001' p23001'
p24002' p25001' p28001' p4001'];

tfastp4 = [p2006' p3006' p5006' p6006' p8006' p9006' p11006' p12006'
p14006' p16006' p17006' p18006' p21006' p22006' p23006' p24006'
p25006' p28006' p4006'];

tslowp4 = [p2009' p3009' p5009' p6009' p8009' p9009' p11009' p12009'
p14009' p16009' p17009' p18009' p21009' p22009' p23009' p24009'
p25009' p28009' p4009'];

xtp4 = [xnormalp4; tnormalp4; xfastp4; tfastp4; xslowp4; tslowp4];

%19
```

```
xnormalp3 = [p2002xsrt' p4001xsrt' p5001xsrt' p6001xsrt' p8001xsrt'
p9001xsrt' p11001xsrt' p12001xsrt' p14001xsrt' p16001xsrt'
p17001xsrt' p18001xsrt' p21001xsrt' p22001xsrt' p23001xsrt'
p24002xsrt' p25002xsrt' p28001xsrt' p3001xsrt'];

xfastp3 = [p2006xsrt' p4006xsrt' p5006xsrt' p6006xsrt' p8006xsrt'
p9006xsrt' p11006xsrt' p12006xsrt' p14006xsrt' p16006xsrt'
p17006xsrt' p18006xsrt' p21006xsrt' p22006xsrt' p23006xsrt'
p24006xsrt' p25006xsrt' p28006xsrt' p3006xsrt'];

xslowp3 = [p2009xsrt' p4009xsrt' p5009xsrt' p6009xsrt' p8009xsrt'
p9009xsrt' p11009xsrt' p12009xsrt' p14009xsrt' p16009xsrt'
p17009xsrt' p18009xsrt' p21009xsrt' p22009xsrt' p23009xsrt'
p24009xsrt' p25009xsrt' p28009xsrt' p3009xsrt'];

tnormalp3 = [p2001' p4001' p5001' p6001' p8001' p9001' p11001'
p12001' p14001' p16001' p17001' p18001' p21001' p22001' p23001'
p24002' p25001' p28001' p3001'];

tfastp3 = [p2006' p4006' p5006' p6006' p8006' p9006' p11006' p12006'
p14006' p16006' p17006' p18006' p21006' p22006' p23006' p24006'
p25006' p28006' p3006'];

tslowp3 = [p2009' p4009' p5009' p6009' p8009' p9009' p11009' p12009'
p14009' p16009' p17009' p18009' p21009' p22009' p23009' p24009'
p25009' p28009' p3009'];

xtp3 = [xnormalp3; tnormalp3; xfastp3; tfastp3; xslowp3; tslowp3];

%20
xnormalp2 = [p3001xsrt' p4001xsrt' p5001xsrt' p6001xsrt' p8001xsrt'
p9001xsrt' p11001xsrt' p12001xsrt' p14001xsrt' p16001xsrt'
p17001xsrt' p18001xsrt' p21001xsrt' p22001xsrt' p23001xsrt'
p24002xsrt' p25002xsrt' p28001xsrt' p2002xsrt'];

xfastp2 = [p3006xsrt' p4006xsrt' p5006xsrt' p6006xsrt' p8006xsrt'
p9006xsrt' p11006xsrt' p12006xsrt' p14006xsrt' p16006xsrt'
p17006xsrt' p18006xsrt' p21006xsrt' p22006xsrt' p23006xsrt'
p24006xsrt' p25006xsrt' p28006xsrt' p2006xsrt'];

xslowp2 = [p3009xsrt' p4009xsrt' p5009xsrt' p6009xsrt' p8009xsrt'
p9009xsrt' p11009xsrt' p12009xsrt' p14009xsrt' p16009xsrt'
p17009xsrt' p18009xsrt' p21009xsrt' p22009xsrt' p23009xsrt'
p24009xsrt' p25009xsrt' p28009xsrt' p2009xsrt'];

tnormalp2 = [p3001' p4001' p5001' p6001' p8001' p9001' p11001'
p12001' p14001' p16001' p17001' p18001' p21001' p22001' p23001'
p24002' p25001' p28001' p2001'];

tfastp2 = [p3006' p4006' p5006' p6006' p8006' p9006' p11006' p12006'
p14006' p16006' p17006' p18006' p21006' p22006' p23006' p24006'
p25006' p28006' p2006'];

tslowp2 = [p3009' p4009' p5009' p6009' p8009' p9009' p11009' p12009'
p14009' p16009' p17009' p18009' p21009' p22009' p23009' p24009'
p25009' p28009' p2009'];

xtp2 = [xnormalp2; tnormalp2; xfastp2; tfastp2; xslowp2; tslowp2];
```

Appendix 8. A Matlab script used for KAM prediction (randomised joint angle inputs)

```
%%% This script is to evaluate the ability of LMtrain (a BPFF ANN)
to predict %%% shuffled data (randomly arranged data points from the
original data)
%%% use data in 'shuffle_2.mat'
%%% the order of participants to be presented to train, validate and
test the %%% ANN are as followed (19 in total)
%%% the second last order is to validate the ANN
%%% the last order is for testing the ANN
%%% the rest are for training

%%% p2 p3 p4 p5 p6 p8 p9 p11 p12 p14 p16 p17 p18 p21 p22 p23 p24 p25
p28 %%%

%%% xtp2 is defined as
%%% input (xp2) (right hip, knee, and ankle abd/add, int/ext and
flex/extens)
%%% angles and target output (tp2) (right knee abduction moment)
%%% WHEN PARTICIPANT2 is used to test the ANN.

% THERFORE THE ORDER FOR xtp2 is
% p3 p4 p5 p6 p8 p9 p11 p12 p14 p16 p17 p18 p21 p22 p23 p24 p25 p28
p2

% As we adopt k-fold validation, so every single participant will be
left out % for testing the ANN

% THEREFORE, the order for xtp3 is
% p2 p4 p5 p6 p8 p9 p11 p12 p14 p16 p17 p18 p21 p22 p23 p24 p25 p28
p3
% so on and so forth

% FOR xtp28, the order is
% p2 p3 p4 p5 p6 p8 p9 p11 p12 p14 p16 p17 p18 p21 p22 p23 p24 p25
p28

%% normal, fast and slow are the walking speed.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
%% This part of the script is to shuffle the data points using the
FIRST
% random result of randperm function of Matlab
% According to the number of data points for every participant left
out
% (xtp2, xtp3, xtp4...,xtp28) are equal (30 rows x 38000 columns)
% [M,N] will be the exact same number for each participant,
% consequently the random order will also be exactly the same for
% each left out participant

tic %%% to calculate the prediction time for all speeds)

[M,N] = size(xtp8);

rng(1)
```

```
ind = randperm(N);

k = ind(1,36001:38000);
%%% as mentioned above 'ind' will be the same for every participant

xtp8_rand = xtp8(:,ind); %%% shuffled xtp %%%

xp8normalsf = xtp8_rand(1:9,:); %%% shuffled input (normal speed)
tp8normalsf = xtp8_rand(10,:);  %%% shuffled target output (normal
speed)
tN = xtp8(10,:); %%% the original (preshuffled) target for normal
speed

xp8fastsf = xtp8_rand(11:19,:); %%% shuffled input (fast speed)
tp8fastsf = xtp8_rand(20,:); %%% shuffled target output (fast speed)
tF = xtp8(20,:); %%% the original target for fast speed

xp8slowsf = xtp8_rand(21:29,:); %%% shuffled input (slow speed)
tp8slowsf = xtp8_rand(30,:); %%% shuffled target output (slow speed)
tS = xtp8(30,:); %%% the original target for slow speed

%% ANN training using the angles as inputs and the KAM as targets
% 'trainlm' is usually the fastest (the most efficient).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

xNormalsf = xp8normalsf;
tNormalsf = tp8normalsf;

xFastsf = xp8fastsf;
tFastsf = tp8fastsf;

xSlowsf = xp8slowsf;
tSlowsf = tp8slowsf;


%%NORMAL SPEED%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

perfnormal= zeros(20,2);

 seednormal = 1 % best of 20

% for seednormal=1:20
    rng(seednormal)


trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.

% Create a Fitting Network
hiddenLayerSize = 18;
net = fitnet(hiddenLayerSize,trainFcn);

% Setup Division of Data for Training, Validation, Testing
 netnormal = fitnet(hiddenLayerSize,trainFcn);
    netnormal.input.processFcns =
{'removeconstantrows','mapminmax'};
    netnormal.output.processFcns =
{'removeconstantrows','mapminmax'};
```

```
    %net.divideFcn = 'dividerand';  % Divide data randomly
    netnormal.divideFcn = 'divideind';
    netnormal.divideMode = 'sample';  % Divide up every sample
%     net.divideParam.trainRatio = 80/100;
%     net.divideParam.valRatio = 10/100;
%     net.divideParam.testRatio = 10/100;
%%

    netnormal.divideParam.trainInd =
[1:round(1/19*17*size(xNormalsf,2))];
    netnormal.divideParam.valInd =
[round(1/19*17*size(xNormalsf,2))+1 :
round(1/19*18*size(xNormalsf,2))];
    netnormal.divideParam.testInd =
[round(1/19*18*size(xNormalsf,2))+1 : size(xNormalsf,2)];
    netnormal.performFcn = 'mse';  % Mean Squared Error
    netnormal.plotFcns =
{'plotperform','plottrainstate','ploterrhist', ...
        'plotregression', 'plotfit'};
% Train the Network
[netnormal,tr] = trainlm(netnormal,xNormalsf,tNormalsf);

% Test the Network
yNormalsf = netnormal(xNormalsf);
enm = gsubtract(tNormalsf,yNormalsf);
performancenormal = perform(netnormal,tNormalsf,yNormalsf)

%%% Since yNormalsf is the shuffled predicted output

y1 = yNormalsf; %%% rename yNormalsf to y1 then we can reverse the
shuffled yNormalsf afterward

%%% NOW reversing the shuffled predicted output to original order

yNormalsf(ind) = yNormalsf;

yN = yNormalsf; %%% yN = the reversed yNormalsf to the original gait
cycle (the original order of the predicted output)

yNormalsf = y1; %%% yNormalsf at this line is the shuffled predicted
output


% view the Network
% view(net)


    seednormal

    testPerformance =
perform(netnormal,performancenormal,yNormalsf);

    perfnormal(seednormal,1) = seednormal;
    perfnormal(seednormal,2) = testPerformance;

    %[seed testPerformance]

% end
```

```
%% IDENTIFY training and testing set of data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%   SHUFFLED  %%%

% training part of the shuffled target output

tNormalsftrain = tNormalsf(1:round(1/19*17*size(tNormalsf,2)));

% the training part of the shuffled predicted output

yNormalsftrain = yNormalsf(1:round(1/19*17*size(yNormalsf,2)));

% the validation part of the shuffled target output

tNormalsfval = tNormalsf(round(1/19*17*size(tNormalsf,2))+1 :
round(1/19*18*size(tNormalsf,2)));

% the validation part of the shuffled predicted output

yNormalsfval = yNormalsf(round(1/19*17*size(yNormalsf,2))+1 :
round(1/19*18*size(yNormalsf,2)));

% the testing part of the shuffled target output

tNormalsftest = tNormalsf(round(1/19*18*size(tNormalsf,2))+1 :
size(tNormalsf,2));

% the testing part of the shuffled predicted output

yNormalsftest = yNormalsf(round(1/19*18*size(yNormalsf,2))+1 :
size(yNormalsf,2));

%%% ORIGINAL or REVERSE SHUFFLED %% as this is the original gait
cycle graph


% training part of the original target output

tNtrain = tN(1:round(1/19*17*size(tN,2)));

% the training part of the reverse shuffled predicted output

yNtrain = yN(1:round(1/19*17*size(yN,2)));

% the validation part of the original target output

tNval = tN(round(1/19*17*size(tN,2))+1 : round(1/19*18*size(tN,2)));

% the validation part of the reverse shuffled predicted output

yNval = yN(round(1/19*17*size(yN,2))+1 : round(1/19*18*size(yN,2)));

% the testing part of the original target output

tNtest = tN(round(1/19*18*size(tN,2))+1 : size(tN,2));
```

```
% the testing part of the reverse shuffled predicted output

yNtest = yN(round(1/19*18*size(yN,2))+1 : size(yN,2));


%% RMSE shuffled
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

RMSEtrainNormalsf = sqrt(mean((tNormalsftrain - yNormalsftrain).^2))
RMSEtestNormalsf = sqrt(mean((tNormalsftest - yNormalsftest).^2))

%%% RMSE original or reversed shuffled
RMSEtrainN = sqrt(mean((tNtrain - yNtrain).^2))
RMSEtestN = sqrt(mean((tNtest -yNtest).^2))


%%


figure
plot (tNormalsftrain)
hold on
plot (yNormalsftrain)

figure
plot (tNormalsftest)
hold on
plot (yNormalsftest)

figure
plot (tNtrain)
hold on
plot (yNtrain)

figure
plot (tNtest)
hold on
plot (yNtest)



%%FAST SPEED%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

perffast= zeros(20,2);

 seedfast = 6 % best of 20

% for seedfast=1:20
    rng(seedfast)


trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.

% Create a Fitting Network
hiddenLayerSize = 18;
net = fitnet(hiddenLayerSize,trainFcn);
```

```
% Setup Division of Data for Training, Validation, Testing
 netfast = fitnet(hiddenLayerSize,trainFcn);
    netfast.input.processFcns = {'removeconstantrows','mapminmax'};
    netfast.output.processFcns = {'removeconstantrows','mapminmax'};
    %net.divideFcn = 'dividerand';  % Divide data randomly
    netfast.divideFcn = 'divideind';
    netfast.divideMode = 'sample';  % Divide up every sample
%     net.divideParam.trainRatio = 80/100;
%     net.divideParam.valRatio = 10/100;
%     net.divideParam.testRatio = 10/100;
%%

    netfast.divideParam.trainInd =
[1:round(1/19*17*size(xFastsf,2))];
    netfast.divideParam.valInd = [round(1/19*17*size(xFastsf,2))+1 :
round(1/19*18*size(xFastsf,2))];
    netfast.divideParam.testInd = [round(1/19*18*size(xFastsf,2))+1
: size(xFastsf,2)];
    netfast.performFcn = 'mse';  % Mean Squared Error
    netfast.plotFcns =
{'plotperform','plottrainstate','ploterrhist', ...
        'plotregression', 'plotfit'};
% Train the Network
[netfast,tr] = trainlm(netfast,xFastsf,tFastsf);

% Test the Network
yFastsf = netfast(xFastsf);
enm = gsubtract(tFastsf,yFastsf);
performancefast = perform(netfast,tFastsf,yFastsf)

%%% Since yFastsf is the shuffled predicted output

y2 = yFastsf; %%% rename yFastsf to y2 then we can reverse the
shuffled yFastsf afterward

%%% NOW reversing the shuffled predicted output to original order

yFastsf(ind) = yFastsf;

yF = yFastsf; %%% yF = the reversed yFastsf to the original gait
cycle (the original order of the predicted output)

yFastsf = y2; %%% yFastsf at this line is the shuffled predicted
output


% view the Network
% view(net)


    seedfast

    testPerformance = perform(netfast,performancefast,yFastsf);

    perffast(seedfast,1) = seedfast;
    perffast(seedfast,2) = testPerformance;

    %[seed testPerformance]

% end
```

```
%% IDENTIFY training and testing set of data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%   SHUFFLED  %%%

% training part of the shuffled target output

tFastsftrain = tFastsf(1:round(1/19*17*size(tFastsf,2)));

% the training part of the shuffled predicted output

yFastsftrain = yFastsf(1:round(1/19*17*size(yFastsf,2)));

% the validation part of the shuffled target output

tFastsfval = tFastsf(round(1/19*17*size(tFastsf,2))+1 :
round(1/19*18*size(tFastsf,2)));

% the validation part of the shuffled predicted output

yFastsfval = yFastsf(round(1/19*17*size(yFastsf,2))+1 :
round(1/19*18*size(yFastsf,2)));

% the testing part of the shuffled target output

tFastsftest = tFastsf(round(1/19*18*size(tFastsf,2))+1 :
size(tFastsf,2));

% the testing part of the shuffled predicted output

yFastsftest = yFastsf(round(1/19*18*size(yFastsf,2))+1 :
size(yFastsf,2));


%%% ORIGINAL or REVERSE SHUFFLED %% as this is the original gait
cycle graph


% training part of the original target output

tFtrain = tF(1:round(1/19*17*size(tF,2)));

% the training part of the reverse shuffled predicted output

yFtrain = yF(1:round(1/19*17*size(yF,2)));

% the validation part of the original target output

tFval = tF(round(1/19*17*size(tF,2))+1 : round(1/19*18*size(tF,2)));

% the validation part of the reverse shuffled predicted output

yFval = yF(round(1/19*17*size(yF,2))+1 : round(1/19*18*size(yF,2)));
```

```
% the testing part of the original target output

tFtest = tF(round(1/19*18*size(tF,2))+1 : size(tF,2));

% the testing part of the reverse shuffled predicted output

yFtest = yF(round(1/19*18*size(yF,2))+1 : size(yF,2));


%% RMSE shuffled
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

RMSEtrainFastsf = sqrt(mean((tFastsftrain - yFastsftrain).^2))
RMSEtestFastsf = sqrt(mean((tFastsftest - yFastsftest).^2))

%%% RMSE original or reversed shuffled
RMSEtrainF = sqrt(mean((tFtrain - yFtrain).^2))
RMSEtestF = sqrt(mean((tFtest -yFtest).^2))


%%


figure
plot (tFastsftrain)
hold on
plot (yFastsftrain)

figure
plot (tFastsftest)
hold on
plot (yFastsftest)

figure
plot (tFtrain)
hold on
plot (yFtrain)

figure
plot (tFtest)
hold on
plot (yFtest)


%%SLOW SPEED%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

perfslow= zeros(20,2);

 seedslow = 19 % best of 20

% for seedslow=1:20
    rng(seedslow)


trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.

% Create a Fitting Network
hiddenLayerSize = 18;
net = fitnet(hiddenLayerSize,trainFcn);
```

```
% Setup Division of Data for Training, Validation, Testing
 netslow = fitnet(hiddenLayerSize,trainFcn);
    netslow.input.processFcns = {'removeconstantrows','mapminmax'};
    netslow.output.processFcns = {'removeconstantrows','mapminmax'};
    %net.divideFcn = 'dividerand';  % Divide data randomly
    netslow.divideFcn = 'divideind';
    netslow.divideMode = 'sample';  % Divide up every sample
%     net.divideParam.trainRatio = 80/100;
%     net.divideParam.valRatio = 10/100;
%     net.divideParam.testRatio = 10/100;
%%

    netslow.divideParam.trainInd =
[1:round(1/19*17*size(xSlowsf,2))];
    netslow.divideParam.valInd = [round(1/19*17*size(xSlowsf,2))+1 :
round(1/19*18*size(xSlowsf,2))];
    netslow.divideParam.testInd = [round(1/19*18*size(xSlowsf,2))+1
: size(xSlowsf,2)];
    netslow.performFcn = 'mse';  % Mean Squared Error
    netslow.plotFcns =
{'plotperform','plottrainstate','ploterrhist', ...
        'plotregression', 'plotfit'};
% Train the Network
[netslow,tr] = trainlm(netslow,xSlowsf,tSlowsf);

% Test the Network
ySlowsf = netslow(xSlowsf);
enm = gsubtract(tSlowsf,ySlowsf);
performanceslow = perform(netslow,tSlowsf,ySlowsf)

%%% Since ySlowsf is the shuffled predicted output

y3 = ySlowsf; %%% rename ySlowsf to y3 then we can reverse the
shuffled ySlowsf afterward

%%% NOW reversing the shuffled predicted output to original order

ySlowsf(ind) = ySlowsf;

yS = ySlowsf; %%% yS = the reversed ySlowsf to the original gait
cycle (the original order of the predicted output)

ySlowsf = y3; %%% ySlowsf at this line is the shuffled predicted
output


% view the Network
% view(net)


    seedslow

    testPerformance = perform(netslow,performanceslow,ySlowsf);

    perfslow(seedslow,1) = seedslow;
    perfslow(seedslow,2) = testPerformance;

    %[seed testPerformance]
```

```
% end


%% IDENTIFY training and testing set of data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%   SHUFFLED  %%%

% training part of the shuffled target output

tSlowsftrain = tSlowsf(1:round(1/19*17*size(tSlowsf,2)));

% the training part of the shuffled predicted output

ySlowsftrain = ySlowsf(1:round(1/19*17*size(ySlowsf,2)));

% the validation part of the shuffled target output

tSlowsfval = tSlowsf(round(1/19*17*size(tSlowsf,2))+1 :
round(1/19*18*size(tSlowsf,2)));

% the validation part of the shuffled predicted output

ySlowsfval = ySlowsf(round(1/19*17*size(ySlowsf,2))+1 :
round(1/19*18*size(ySlowsf,2)));

% the testing part of the shuffled target output

tSlowsftest = tSlowsf(round(1/19*18*size(tSlowsf,2))+1 :
size(tSlowsf,2));

% the testing part of the shuffled predicted output

ySlowsftest = ySlowsf(round(1/19*18*size(ySlowsf,2))+1 :
size(ySlowsf,2));

%%% ORIGINAL or REVERSE SHUFFLED %% as this is the original gait
cycle graph


% training part of the original target output

tStrain = tS(1:round(1/19*17*size(tS,2)));

% the training part of the reverse shuffled predicted output

yStrain = yS(1:round(1/19*17*size(yS,2)));

% the validation part of the original target output

tSval = tS(round(1/19*17*size(tS,2))+1 : round(1/19*18*size(tS,2)));

% the validation part of the reverse shuffled predicted output

ySval = yS(round(1/19*17*size(yS,2))+1 : round(1/19*18*size(yS,2)));

% the testing part of the original target output
```

```
tStest = tS(round(1/19*18*size(tS,2))+1 : size(tS,2));

% the testing part of the reverse shuffled predicted output

yStest = yS(round(1/19*18*size(yS,2))+1 : size(yS,2));


%% RMSE shuffled
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

RMSEtrainSlowsf = sqrt(mean((tSlowsftrain - ySlowsftrain).^2))
RMSEtestSlowsf = sqrt(mean((tSlowsftest - ySlowsftest).^2))

%%% RMSE original or reversed shuffled
RMSEtrainF = sqrt(mean((tStrain - yStrain).^2))
RMSEtestS = sqrt(mean((tStest -yStest).^2))


%%


figure
plot (tSlowsftrain)
hold on
plot (ySlowsftrain)

figure
plot (tSlowsftest)
hold on
plot (ySlowsftest)

figure
plot (tStrain)
hold on
plot (yStrain)

figure
plot (tStest)
hold on
plot (yStest)


toc
%
% writematrix(yNtest','results\yNtestp8.xlsx')
% writematrix(yFtest','results\yFtestp8.xlsx')
% writematrix(yStest','results\yStestp8.xlsx')
% writematrix(tNtest','results\tNtestp8.xlsx')
% writematrix(tFtest','results\tFtestp8.xlsx')
% writematrix(tStest','results\tStestp8.xlsx')


%%This part is to calculate normalised RMSE (NRMSE) and systemic
mean absolute percentage error (SMAPE)
%%%NRMSE = RMSE divides by average t (the average value of the
measured data)

NRMSEnormal = 100.*RMSEtestN/(max(tNtest)-min(tNtest));
```

```
NRMSEfast = 100.*RMSEtestF/(max(tFtest)-min(tFtest));
NRMSEslow = 100.*RMSEtestS/(max(tStest)-min(tStest));

SMAPEnormal1 = abs(yNtest-tNtest);
SMAPEnormal2 = abs(tNtest)+abs(yNtest);
SMAPEnormal3 = SMAPEnormal1./SMAPEnormal2;
SMAPEnormal = sum(SMAPEnormal3);
SMAPEnormal = (SMAPEnormal/2000)*100;

SMAPEfast1 = abs(yFtest-tFtest);
SMAPEfast2 = abs(tFtest)+abs(yFtest);
SMAPEfast3 = SMAPEfast1./SMAPEfast2;
SMAPEfast = sum(SMAPEfast3);
SMAPEfast = (SMAPEfast/2000)*100;

SMAPEslow1 = abs(yStest-tStest);
SMAPEslow2 = abs(tStest)+abs(yStest);
SMAPEslow3 = SMAPEslow1./SMAPEslow2;
SMAPEslow = sum(SMAPEslow3);
SMAPEslow = (SMAPEslow/2000)*100;

%%% then calculate correlation coefficient


TN =   tNtest';
YN =   yNtest';
rN =   corrcoef(TN,YN)

TF =   tFtest';
YF =   yFtest';
rF =   corrcoef(TF,YF)

TS =   tStest';
YS =   yStest';
rS =   corrcoef(TS,YS)
```

Appendix 9. A Matlab script used to train the FFANN with data of seven marker coordinates at 56 hidden neurons

```
%%%this script is to evaluate the ability of LMtrain to predict
shuffled data
%%from shuffled markers trajectories
%%use data in 'marker7.mat'
tic
[M,N] = size(xtp8);

rng(1)
ind = randperm(N);
xtp8_rand = xtp8(:,ind);

xp8normal = xtp8_rand(1:21,:);
tp8normal = xtp8_rand(22,:);
t1 = xtp8(22,:);

xp8fast = xtp8_rand(23:43,:);
tp8fast = xtp8_rand(44,:);
t2 = xtp8(44,:);

xp8slow = xtp8_rand(45:65,:);
tp8slow = xtp8_rand(66,:);
t3 = xtp8(66,:);

%% ANN training using the angles as inputs and the KAM as targets
% 'trainlm' is usually fastest.

xNormal = xp8normal;
tNormal = tp8normal;

xFast = xp8fast;
tFast = tp8fast;

xSlow = xp8slow;
tSlow = tp8slow;


%% train normal speed
perf1= zeros(20,2);

 seed1 = 16 % best of 20
%  for seed1=1:20
    rng(seed1)


trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.

% Create a Fitting Network
hiddenLayerSize = 56;
net = fitnet(hiddenLayerSize,trainFcn);

% Setup Division of Data for Training, Validation, Testing
 netnm1 = fitnet(hiddenLayerSize,trainFcn);
    netnm1.input.processFcns = {'removeconstantrows','mapminmax'};
    netnm1.output.processFcns = {'removeconstantrows','mapminmax'};
    %net.divideFcn = 'dividerand';  % Divide data randomly
```

```
    netnm1.divideFcn = 'divideind';
    netnm1.divideMode = 'sample';  % Divide up every sample
%     net.divideParam.trainRatio = 80/100;
%     net.divideParam.valRatio = 10/100;
%     net.divideParam.testRatio = 10/100;
%%

    netnm1.divideParam.trainInd =
[1:round(1/19*17*size(xNormal,2))];
    netnm1.divideParam.valInd =  [round(1/19*17*size(xNormal,2))+1 :
round(1/19*18*size(xNormal,2))];
    netnm1.divideParam.testInd = [round(1/19*18*size(xNormal,2))+1 :
size(xNormal,2)];
    netnm1.performFcn = 'mse';  % Mean Squared Error
    netnm1.plotFcns = {'plotperform','plottrainstate','ploterrhist',
...
        'plotregression', 'plotfit'};
% Train the Network
[netnm1,tr] = trainlm(netnm1,xNormal,tNormal);

% Test the Network
yNormal = netnm1(xNormal);
enm = gsubtract(tNormal,yNormal);
performance1 = perform(netnm1,tNormal,yNormal)

%%% yNormal is the shuffled predicted output
%%% rename yNormal to y1
y1 = yNormal;

%%% yNormal after this line will be the inverse from shuffled of the
predicted output
yNormal(ind) = yNormal;

% View the Network
% view(net)


%     seed1
%     testPerformance = perform(netnm1,performance1,yNormal);
%
%     perf1(seed1,1) = seed1;
%     perf1(seed1,2) = testPerformance;
    %[seed testPerformance]

%  end



%% identify traing and testing set of data
%%%shuffled
tNormaltrain = tNormal(1:round(1/19*17*size(tNormal,2))); %target
output at training part
y1train = y1(1:round(1/19*17*size(y1,2))); %predicted output at
training part

tNormaltest = tNormal(round(1/19*18*size(tNormal,2))+1 :
size(tNormal,2)); %target output at testing part
y1test = y1(round(1/19*18*size(y1,2))+1 : size(y1,2)); %predicted
output at testing part
```

```
%%%non shuffled
t1train = t1(1:round(1/19*17*size(t1,2))); %target output at
training part before shuffled
yNormaltrain = yNormal(1:round(1/19*17*size(yNormal,2))); %reversed
predicted output at training part

t1test = t1(round(1/19*18*size(t1,2))+1 : size(t1,2)); %target
output at testing part before shuffled
yNormaltest = yNormal(round(1/19*18*size(yNormal,2))+1 :
size(yNormal,2)); %reversed predicted output at testing part

%%%shuffled
RMSEtrain1Sf = sqrt(mean((tNormaltrain -y1train).^2))
RMSEtestNormalSf = sqrt(mean((tNormaltest -y1test).^2))

%%%non shuffled
RMSEtrain1 = sqrt(mean((t1train -yNormaltrain).^2))
RMSEtestNormal = sqrt(mean((t1test -yNormaltest).^2))

%%



% figure
% plot (tNormaltrain)
% hold on
% plot (y1train)
%
% figure
% plot (tNormaltest)
% hold on
% plot (y1test)
%
% figure
% plot (t1train)
% hold on
% plot (yNormaltrain)
%
% figure
% plot (t1test)
% hold on
% plot (yNormaltest)
%

%% train fast speed

% perf2= zeros(20,2);

 seed2 = 7 % best of 20
% for seed2=1:20
    rng(seed2)

trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.

% Create a Fitting Network
hiddenLayerSize = 56;
net = fitnet(hiddenLayerSize,trainFcn);

% Setup Division of Data for Training, Validation, Testing
 netnm2 = fitnet(hiddenLayerSize,trainFcn);
    netnm2.input.processFcns = {'removeconstantrows','mapminmax'};
    netnm2.output.processFcns = {'removeconstantrows','mapminmax'};
```

```
    %net.divideFcn = 'dividerand';  % Divide data randomly
    netnm2.divideFcn = 'divideind';
    netnm2.divideMode = 'sample';  % Divide up every sample
%     net.divideParam.trainRatio = 80/100;
%     net.divideParam.valRatio = 10/100;
%     net.divideParam.testRatio = 10/100;
%%

    netnm2.divideParam.trainInd =
[1:round(1/19*17*size(xNormal,2))];
    netnm2.divideParam.valInd =  [round(1/19*17*size(xNormal,2))+1 :
round(1/19*18*size(xNormal,2))];
    netnm2.divideParam.testInd = [round(1/19*18*size(xNormal,2))+1 :
size(xNormal,2)];
    netnm2.performFcn = 'mse';  % Mean Squared Error
    netnm2.plotFcns = {'plotperform','plottrainstate','ploterrhist',
...
        'plotregression', 'plotfit'};
% Train the Network
[netnm2,tr] = trainlm(netnm2,xFast,tFast);

% Test the Network
yFast = netnm2(xFast);
enm = gsubtract(tFast,yFast);
performance2 = perform(netnm2,tFast,yFast)

%%% y2 isthe prediced output
y2 = yFast;

%%% yFast after this line will be the inverse shuffled of the
predictedoutput
yFast(ind) = yFast;

% View the Network
% view(net)


%     seed2
%     testPerformance2 = perform(netnm2,performance2,yFast);
%
%     perf2(seed2,1) = seed2;
%     perf2(seed2,2) = testPerformance2;
    %[seed testPerformance]
%  end


%% identify traing and testing set of data
%%%shuffled
tFasttrain = tFast(1:round(1/19*17*size(tFast,2))); %target output
at training part
y2train = y2(1:round(1/19*17*size(y2,2))); %predicted output at
training part

tFasttest = tFast(round(1/19*18*size(tFast,2))+1 : size(tFast,2));
%target output at testing part
y2test = y2(round(1/19*18*size(y2,2))+1 : size(y2,2)); %predicted
output at testing part

%%%non shuffled
t2train = t2(1:round(1/19*17*size(t2,2))); %target output at
training part before shuffled
```

```
yFasttrain = yFast(1:round(1/19*17*size(yFast,2))); %reversed
predicted output at training part

t2test = t2(round(1/19*18*size(t2,2))+1 : size(t2,2)); %target
output at testing part before shuffled
yFasttest = yFast(round(1/19*18*size(yFast,2))+1 : size(yFast,2));
%reversed predicted output at testing part

%%%shuffled
RMSEtrain1Sf = sqrt(mean((tFasttrain -y2train).^2))
RMSEtestFastSf = sqrt(mean((tFasttest -y2test).^2))

%%%non shuffled
RMSEtrain1 = sqrt(mean((t2train -yFasttrain).^2))
RMSEtestFast = sqrt(mean((t2test -yFasttest).^2))


%%


% figure
% plot (tFasttrain)
% hold on
% plot (y2train)
%
% figure
% plot (tFasttest)
% hold on
% plot (y2test)
%
% figure
% plot (t2train)
% hold on
% plot (yFasttrain)
%
% figure
% plot (t2test)
% hold on
% plot (yFasttest)


%% train slow speed

perf3= zeros(20,2);

 seed3 = 17 % best of 20
% for seed3=1:20
    rng(seed3)
trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.

% Create a Fitting Network
hiddenLayerSize = 56;
net = fitnet(hiddenLayerSize,trainFcn);

% Setup Division of Data for Training, Validation, Testing
 netnm3 = fitnet(hiddenLayerSize,trainFcn);
    netnm3.input.processFcns = {'removeconstantrows','mapminmax'};
    netnm3.output.processFcns = {'removeconstantrows','mapminmax'};
    %net.divideFcn = 'dividerand';  % Divide data randomly
    netnm3.divideFcn = 'divideind';
```

```
    netnm3.divideMode = 'sample';  % Divide up every sample
%     net.divideParam.trainRatio = 80/100;
%     net.divideParam.valRatio = 10/100;
%     net.divideParam.testRatio = 10/100;
%%

    netnm3.divideParam.trainInd =
[1:round(1/19*17*size(xNormal,2))];
    netnm3.divideParam.valInd =  [round(1/19*17*size(xNormal,2))+1 :
round(1/19*18*size(xNormal,2))];
    netnm3.divideParam.testInd = [round(1/19*18*size(xNormal,2))+1 :
size(xNormal,2)];
    netnm3.performFcn = 'mse';  % Mean Squared Error
    netnm3.plotFcns = {'plotperform','plottrainstate','ploterrhist',
...
        'plotregression', 'plotfit'};
% Train the Network
[netnm3,tr] = trainlm(netnm3,xSlow,tSlow);

% Test the Network
ySlow = netnm3(xSlow);
enm = gsubtract(tSlow,ySlow);
performance3 = perform(netnm3,tSlow,ySlow)

%%% y3 isthe prediced output
y3 = ySlow;

%%% ySlow after this line will be the inverse shuffled of the
predictedoutput
ySlow(ind) = ySlow;

% View the Network
% view(net)


%     seed3
%     testPerformance3 = perform(netnm3,performance3,ySlow);
%
%     perf3(seed3,1) = seed3;
%     perf3(seed3,2) = testPerformance3;
    %[seed testPerformance]
% end

%% identify traing and testing set of data
%%%shuffled
tSlowtrain = tSlow(1:round(1/19*17*size(tSlow,2))); %target output
at training part
y3train = y3(1:round(1/19*17*size(y3,2))); %predicted output at
training part

tSlowtest = tSlow(round(1/19*18*size(tSlow,2))+1 : size(tSlow,2));
%target output at testing part
y3test = y3(round(1/19*18*size(y3,2))+1 : size(y3,2)); %predicted
output at testing part

%%%non shuffled
t3train = t3(1:round(1/19*17*size(t3,2))); %target output at
training part before shuffled
ySlowtrain = ySlow(1:round(1/19*17*size(ySlow,2))); %reversed
predicted output at training part
```

```
t3test = t3(round(1/19*18*size(t3,2))+1 : size(t3,2)); %target
output at testing part before shuffled
ySlowtest = ySlow(round(1/19*18*size(ySlow,2))+1 : size(ySlow,2));
%reversed predicted output at testing part

%%%shuffled
RMSEtrain1Sf = sqrt(mean((tSlowtrain -y3train).^2))
RMSEtestSlowSf = sqrt(mean((tSlowtest -y3test).^2))

%%%non shuffled
RMSEtrain1 = sqrt(mean((t3train -ySlowtrain).^2))
RMSEtestSlow = sqrt(mean((t3test -ySlowtest).^2))

%%


% figure
% plot (tSlowtrain)
% hold on
% plot (y3train)
%
% figure
% plot (tSlowtest)
% hold on
% plot (y3test)
%
% figure
% plot (t3train)
% hold on
% plot (ySlowtrain)
%
% figure
% plot (t3test)
% hold on
% plot (ySlowtest)

%  end
toc

%%This part is to calculate normalised RMSE (NRMSE) and systemic
mean absolute percentage error (SMAPE)
%%%NRMSE = RMSE divides by average t (the average value of the
measured data)

NRMSEnormal = 100.*RMSEtestNormal/(max(t1test)-min(t1test));
NRMSEfast = 100.*RMSEtestFast/(max(t2test)-min(t2test));
NRMSEslow = 100.*RMSEtestSlow/(max(t3test)-min(t3test));

SMAPEnormal1 = abs(yNormaltest-t1test);
SMAPEnormal2 = abs(t1test)+abs(yNormaltest);
SMAPEnormal3 = SMAPEnormal1./SMAPEnormal2;
SMAPEnormal = sum(SMAPEnormal3);
SMAPEnormal = (SMAPEnormal/2000)*100;

SMAPEfast1 = abs(yFasttest-t2test);
SMAPEfast2 = abs(t2test)+abs(yFasttest);
SMAPEfast3 = SMAPEfast1./SMAPEfast2;
SMAPEfast = sum(SMAPEfast3);
SMAPEfast = (SMAPEfast/2000)*100;

SMAPEslow1 = abs(ySlowtest-t3test);
```

```
SMAPEslow2 = abs(t3test)+abs(ySlowtest);
SMAPEslow3 = SMAPEslow1./SMAPEslow2;
SMAPEslow = sum(SMAPEslow3);
SMAPEslow = (SMAPEslow/2000)*100;

%%% then calculate correlation coefficient


TN =   t1test';
YN =   yNormaltest';
rN =   corrcoef(TN,YN)

TF =   t2test';
YF =   yFasttest';
rF =   corrcoef(TF,YF)

TS =   t3test';
YS =   ySlowtest';
rS =   corrcoef(TS,YS)
```

Appendix 10. A Matlab script used to predict KAM (four marker coordinates, 24 hidden neurons)

```
%%%this script is to evaluate the ability of LMtrain to predict
shuffled data
%%from shuffled markers trajectories
%%use data in 'markers4.mat'
tic
[M,N] = size(xtp8);

rng(1)
ind = randperm(N);
xtp8_rand = xtp8(:,ind);

xp8normal = xtp8_rand(1:12,:);
tp8normal = xtp8_rand(13,:);
t1 = xtp8(13,:);

xp8fast = xtp8_rand(14:25,:);
tp8fast = xtp8_rand(26,:);
t2 = xtp8(26,:);

xp8slow = xtp8_rand(27:38,:);
tp8slow = xtp8_rand(39,:);
t3 = xtp8(39,:);

%% ANN training using the angles as inputs and the KAM as targets
% 'trainlm' is usually fastest.

xNormal = xp8normal;
tNormal = tp8normal;


xFast = xp8fast;
tFast = tp8fast;

xSlow = xp8slow;
tSlow = tp8slow;


%% train normal speed
perf1= zeros(20,2);

 seed1 = 7 % best of 20
% for seed1=1:20
    rng(seed1)


trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.

% Create a Fitting Network
hiddenLayerSize = 24;
net = fitnet(hiddenLayerSize,trainFcn);

% Setup Division of Data for Training, Validation, Testing
 netnm1 = fitnet(hiddenLayerSize,trainFcn);
    netnm1.input.processFcns = {'removeconstantrows','mapminmax'};
    netnm1.output.processFcns = {'removeconstantrows','mapminmax'};
    %net.divideFcn = 'dividerand';  % Divide data randomly
```

```
    netnm1.divideFcn = 'divideind';
    netnm1.divideMode = 'sample';  % Divide up every sample
%     net.divideParam.trainRatio = 80/100;
%     net.divideParam.valRatio = 10/100;
%     net.divideParam.testRatio = 10/100;
%%

    netnm1.divideParam.trainInd =
[1:round(1/19*17*size(xNormal,2))];
    netnm1.divideParam.valInd =  [round(1/19*17*size(xNormal,2))+1 :
round(1/19*18*size(xNormal,2))];
    netnm1.divideParam.testInd = [round(1/19*18*size(xNormal,2))+1 :
size(xNormal,2)];
    netnm1.performFcn = 'mse';  % Mean Squared Error
    netnm1.plotFcns = {'plotperform','plottrainstate','ploterrhist',
...
        'plotregression', 'plotfit'};
% Train the Network
[netnm1,tr] = trainlm(netnm1,xNormal,tNormal);

% Test the Network
yNormal = netnm1(xNormal);
enm = gsubtract(tNormal,yNormal);
performance1 = perform(netnm1,tNormal,yNormal)

%%% yNormal is the shuffled predicted output
%%% rename yNormal to y1
y1 = yNormal;

%%% yNormal after this line will be the inverse from shuffled of the
predicted output
yNormal(ind) = yNormal;

% View the Network
% view(net)


    seed1
    testPerformance = perform(netnm1,performance1,yNormal);

    perf1(seed1,1) = seed1;
    perf1(seed1,2) = testPerformance;
    %[seed testPerformance]

%  end



%% identify traing and testing set of data
%%%shuffled
tNormaltrain = tNormal(1:round(1/19*17*size(tNormal,2))); %target
output at training part
y1train = y1(1:round(1/19*17*size(y1,2))); %predicted output at
training part

tNormaltest = tNormal(round(1/19*18*size(tNormal,2))+1 :
size(tNormal,2)); %target output at testing part
y1test = y1(round(1/19*18*size(y1,2))+1 : size(y1,2)); %predicted
output at testing part
```

```
%%%non shuffled
t1train = t1(1:round(1/19*17*size(t1,2))); %target output at
training part before shuffled
yNormaltrain = yNormal(1:round(1/19*17*size(yNormal,2))); %reversed
predicted output at training part

t1test = t1(round(1/19*18*size(t1,2))+1 : size(t1,2)); %target
output at testing part before shuffled
yNormaltest = yNormal(round(1/19*18*size(yNormal,2))+1 :
size(yNormal,2)); %reversed predicted output at testing part

%%%shuffled
RMSEtrain1Sf = sqrt(mean((tNormaltrain -y1train).^2))
RMSEtestNormalSf = sqrt(mean((tNormaltest -y1test).^2))

%%%non shuffled
RMSEtrain1 = sqrt(mean((t1train -yNormaltrain).^2))
RMSEtestNormal = sqrt(mean((t1test -yNormaltest).^2))

%%



% figure
% plot (tNormaltrain)
% hold on
% plot (y1train)
%
% figure
% plot (tNormaltest)
% hold on
% plot (y1test)
%
% figure
% plot (t1train)
% hold on
% plot (yNormaltrain)
%
% figure
% plot (t1test)
% hold on
% plot (yNormaltest)


%% train fast speed

perf2= zeros(20,2);

 seed2 = 11 % best of 20
% for seed2=1:20
    rng(seed2)

trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.

% Create a Fitting Network
hiddenLayerSize = 24;
net = fitnet(hiddenLayerSize,trainFcn);

% Setup Division of Data for Training, Validation, Testing
 netnm2 = fitnet(hiddenLayerSize,trainFcn);
    netnm2.input.processFcns = {'removeconstantrows','mapminmax'};
```

```
    netnm2.output.processFcns = {'removeconstantrows','mapminmax'};
    %net.divideFcn = 'dividerand';  % Divide data randomly
    netnm2.divideFcn = 'divideind';
    netnm2.divideMode = 'sample';  % Divide up every sample
%     net.divideParam.trainRatio = 80/100;
%     net.divideParam.valRatio = 10/100;
%     net.divideParam.testRatio = 10/100;
%%

    netnm2.divideParam.trainInd =
[1:round(1/19*17*size(xNormal,2))];
    netnm2.divideParam.valInd =  [round(1/19*17*size(xNormal,2))+1 :
round(1/19*18*size(xNormal,2))];
    netnm2.divideParam.testInd = [round(1/19*18*size(xNormal,2))+1 :
size(xNormal,2)];
    netnm2.performFcn = 'mse';  % Mean Squared Error
    netnm2.plotFcns = {'plotperform','plottrainstate','ploterrhist',
...
        'plotregression', 'plotfit'};
% Train the Network
[netnm2,tr] = trainlm(netnm2,xFast,tFast);

% Test the Network
yFast = netnm2(xFast);
enm = gsubtract(tFast,yFast);
performance2 = perform(netnm2,tFast,yFast)

%%% y2 isthe prediced output
y2 = yFast;

%%% yFast after this line will be the inverse shuffled of the
predictedoutput
yFast(ind) = yFast;

% View the Network
% view(net)


    seed2
    testPerformance2 = perform(netnm2,performance2,yFast);

    perf2(seed2,1) = seed2;
    perf2(seed2,2) = testPerformance2;
    %[seed testPerformance]
% end


%% identify traing and testing set of data
%%%shuffled
tFasttrain = tFast(1:round(1/19*17*size(tFast,2))); %target output
at training part
y2train = y2(1:round(1/19*17*size(y2,2))); %predicted output at
training part

tFasttest = tFast(round(1/19*18*size(tFast,2))+1 : size(tFast,2));
%target output at testing part
y2test = y2(round(1/19*18*size(y2,2))+1 : size(y2,2)); %predicted
output at testing part

%%%non shuffled
```

```
t2train = t2(1:round(1/19*17*size(t2,2))); %target output at
training part before shuffled
yFasttrain = yFast(1:round(1/19*17*size(yFast,2))); %reversed
predicted output at training part

t2test = t2(round(1/19*18*size(t2,2))+1 : size(t2,2)); %target
output at testing part before shuffled
yFasttest = yFast(round(1/19*18*size(yFast,2))+1 : size(yFast,2));
%reversed predicted output at testing part

%%%shuffled
RMSEtrain1Sf = sqrt(mean((tFasttrain -y2train).^2))
RMSEtestFastSf = sqrt(mean((tFasttest -y2test).^2))

%%%non shuffled
RMSEtrain1 = sqrt(mean((t2train -yFasttrain).^2))
RMSEtestFast = sqrt(mean((t2test -yFasttest).^2))


%%


% figure
% plot (tFasttrain)
% hold on
% plot (y2train)
%
% figure
% plot (tFasttest)
% hold on
% plot (y2test)
%
% figure
% plot (t2train)
% hold on
% plot (yFasttrain)
%
% figure
% plot (t2test)
% hold on
% plot (yFasttest)


%% train slow speed

perf3= zeros(20,2);

 seed3 = 7 % best of 20
% for seed3=1:20
    rng(seed3)
trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.

% Create a Fitting Network
hiddenLayerSize = 24;
net = fitnet(hiddenLayerSize,trainFcn);

% Setup Division of Data for Training, Validation, Testing
 netnm3 = fitnet(hiddenLayerSize,trainFcn);
    netnm3.input.processFcns = {'removeconstantrows','mapminmax'};
    netnm3.output.processFcns = {'removeconstantrows','mapminmax'};
```

```
    %net.divideFcn = 'dividerand';  % Divide data randomly
    netnm3.divideFcn = 'divideind';
    netnm3.divideMode = 'sample';  % Divide up every sample
%     net.divideParam.trainRatio = 80/100;
%     net.divideParam.valRatio = 10/100;
%     net.divideParam.testRatio = 10/100;
%%

    netnm3.divideParam.trainInd =
[1:round(1/19*17*size(xNormal,2))];
    netnm3.divideParam.valInd =  [round(1/19*17*size(xNormal,2))+1 :
round(1/19*18*size(xNormal,2))];
    netnm3.divideParam.testInd = [round(1/19*18*size(xNormal,2))+1 :
size(xNormal,2)];
    netnm3.performFcn = 'mse';  % Mean Squared Error
    netnm3.plotFcns = {'plotperform','plottrainstate','ploterrhist',
...
        'plotregression', 'plotfit'};
% Train the Network
[netnm3,tr] = trainlm(netnm3,xSlow,tSlow);

% Test the Network
ySlow = netnm3(xSlow);
enm = gsubtract(tSlow,ySlow);
performance3 = perform(netnm3,tSlow,ySlow)

%%% y3 isthe prediced output
y3 = ySlow;

%%% ySlow after this line will be the inverse shuffled of the
predictedoutput
ySlow(ind) = ySlow;

% View the Network
% view(net)


    seed3
    testPerformance3 = perform(netnm3,performance3,ySlow);

    perf3(seed3,1) = seed3;
    perf3(seed3,2) = testPerformance3;
    %[seed testPerformance]
% end

%% identify traing and testing set of data
%%%shuffled
tSlowtrain = tSlow(1:round(1/19*17*size(tSlow,2))); %target output
at training part
y3train = y3(1:round(1/19*17*size(y3,2))); %predicted output at
training part

tSlowtest = tSlow(round(1/19*18*size(tSlow,2))+1 : size(tSlow,2));
%target output at testing part
y3test = y3(round(1/19*18*size(y3,2))+1 : size(y3,2)); %predicted
output at testing part

%%%non shuffled
t3train = t3(1:round(1/19*17*size(t3,2))); %target output at
training part before shuffled
```

```
ySlowtrain = ySlow(1:round(1/19*17*size(ySlow,2))); %reversed
predicted output at training part

t3test = t3(round(1/19*18*size(t3,2))+1 : size(t3,2)); %target
output at testing part before shuffled
ySlowtest = ySlow(round(1/19*18*size(ySlow,2))+1 : size(ySlow,2));
%reversed predicted output at testing part

%%%shuffled
RMSEtrain1Sf = sqrt(mean((tSlowtrain -y3train).^2))
RMSEtestSlowSf = sqrt(mean((tSlowtest -y3test).^2))

%%%non shuffled
RMSEtrain1 = sqrt(mean((t3train -ySlowtrain).^2))
RMSEtestSlow = sqrt(mean((t3test -ySlowtest).^2))

%%



% figure
% plot (tSlowtrain)
% hold on
% plot (y3train)
%
% figure
% plot (tSlowtest)
% hold on
% plot (y3test)
%
% figure
% plot (t3train)
% hold on
% plot (ySlowtrain)
%
% figure
% plot (t3test)
% hold on
% plot (ySlowtest)

%  end
toc


%%This part is to calculate normalised RMSE (NRMSE) and systemic
mean absolute percentage error (SMAPE)
%%%NRMSE = RMSE divides by average t (the average value of the
measured data)

NRMSEnormal = 100.*RMSEtestNormal/(max(t1test)-min(t1test));
NRMSEfast = 100.*RMSEtestFast/(max(t2test)-min(t2test));
NRMSEslow = 100.*RMSEtestSlow/(max(t3test)-min(t3test));

SMAPEnormal1 = abs(yNormaltest-t1test);
SMAPEnormal2 = abs(t1test)+abs(yNormaltest);
SMAPEnormal3 = SMAPEnormal1./SMAPEnormal2;
SMAPEnormal = sum(SMAPEnormal3);
SMAPEnormal = (SMAPEnormal/2000)*100;

SMAPEfast1 = abs(yFasttest-t2test);
SMAPEfast2 = abs(t2test)+abs(yFasttest);
SMAPEfast3 = SMAPEfast1./SMAPEfast2;
```

```
SMAPEfast = sum(SMAPEfast3);
SMAPEfast = (SMAPEfast/2000)*100;

SMAPEslow1 = abs(ySlowtest-t3test);
SMAPEslow2 = abs(t3test)+abs(ySlowtest);
SMAPEslow3 = SMAPEslow1./SMAPEslow2;
SMAPEslow = sum(SMAPEslow3);
SMAPEslow = (SMAPEslow/2000)*100;

%%% then calculate correlation coefficient


TN =   t1test';
YN =   yNormaltest';
rN =   corrcoef(TN,YN)

TF =   t2test';
YF =   yFasttest';
rF =   corrcoef(TF,YF)

TS =   t3test';
YS =   ySlowtest';
rS =   corrcoef(TS,YS)
```

Appendix 11. A Matlab script used to create PCA data and predict the KAM using the PCA data

```
%%%this script is to evaluate the ability of LMtrain to predict
shuffled data
%%use data in 'shuffle_2.mat'
tic
[M,N] = size(xtp8);

rng(1)
ind = randperm(N);
xtp8_rand = xtp8(:,ind);

xp8normal = xtp8_rand(1:9,:);
tp8normal = xtp8_rand(10,:);
t1 = xtp8(10,:);

xp8fast = xtp8_rand(11:19,:);
tp8fast = xtp8_rand(20,:);
t2 = xtp8(20,:);

xp8slow = xtp8_rand(21:29,:);
tp8slow = xtp8_rand(30,:);
t3 = xtp8(30,:);

%% ANN training using the angles as inputs and the KAM as targets
% 'trainlm' is usually fastest.

% for normal speed

xNormal = xp8normal;
tNormal = tp8normal;

xFast = xp8fast;
tFast = tp8fast;

xSlow = xp8slow;
tSlow = tp8slow;

[coeff1,score1,latent1,tsquared1,explained1,mu1] = pca(xNormal');

xNormal = score1(:,1:3);
xNormal = xNormal';

[coeff2,score2,latent2,tsquared2,explained2,mu2] = pca(xFast');

xFast = score2(:,1:3);
xFast = xFast';

[coeff3,score3,latent3,tsquared3,explained3,mu3] = pca(xSlow');

xSlow = score3(:,1:3);
xSlow = xSlow';


%% train normal speed
perf1= zeros(20,2);
```

```
 seed1 = 5 % best of 20
% for seed1=1:20
    rng(seed1)


trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.

% Create a Fitting Network
hiddenLayerSize = 6;
%%using 6 because the first 3 columns of score1 represented approx
90% data in PCA
%%and the number of hidden layers is number of input variables x 2%%
net = fitnet(hiddenLayerSize,trainFcn);

% Setup Division of Data for Training, Validation, Testing
 netnm1 = fitnet(hiddenLayerSize,trainFcn);
    netnm1.input.processFcns = {'removeconstantrows','mapminmax'};
    netnm1.output.processFcns = {'removeconstantrows','mapminmax'};
    %net.divideFcn = 'dividerand';  % Divide data randomly
    netnm1.divideFcn = 'divideind';
    netnm1.divideMode = 'sample';  % Divide up every sample
%     net.divideParam.trainRatio = 80/100;
%     net.divideParam.valRatio = 10/100;
%     net.divideParam.testRatio = 10/100;
%%


    netnm1.divideParam.trainInd =
[1:round(1/19*17*size(xNormal,2))];
    netnm1.divideParam.valInd =  [round(1/19*17*size(xNormal,2))+1 :
round(1/19*18*size(xNormal,2))];
    netnm1.divideParam.testInd = [round(1/19*18*size(xNormal,2))+1 :
size(xNormal,2)];
    netnm1.performFcn = 'mse';  % Mean Squared Error
    netnm1.plotFcns = {'plotperform','plottrainstate','ploterrhist',
...
        'plotregression', 'plotfit'};
% Train the Network
[netnm1,tr] = trainlm(netnm1,xNormal,tNormal);

% Test the Network
yNormal = netnm1(xNormal);
enm = gsubtract(tNormal,yNormal);
performance1 = perform(netnm1,tNormal,yNormal)

%%% yNormal is the shuffled predicted output
%%% rename yNormal to y1
y1 = yNormal;

%%% yNormal after this line will be the inverse from shuffled of the
predicted output
yNormal(ind) = yNormal;

% View the Network
% view(net)


    seed1
    testPerformance = perform(netnm1,performance1,yNormal);
```

```
    perf1(seed1,1) = seed1;
    perf1(seed1,2) = testPerformance;
    %[seed testPerformance]

%  end



%% identify traing and testing set of data
%%%shuffled
tNormaltrain = tNormal(1:round(1/19*17*size(tNormal,2))); %target
output at training part
y1train = y1(1:round(1/19*17*size(y1,2))); %predicted output at
training part

tNormaltest = tNormal(round(1/19*18*size(tNormal,2))+1 :
size(tNormal,2)); %target output at testing part
y1test = y1(round(1/19*18*size(y1,2))+1 : size(y1,2)); %predicted
output at testing part

%%%non shuffled
t1train = t1(1:round(1/19*17*size(t1,2))); %target output at
training part before shuffled
yNormaltrain = yNormal(1:round(1/19*17*size(yNormal,2))); %reversed
predicted output at training part

t1test = t1(round(1/19*18*size(t1,2))+1 : size(t1,2)); %target
output at testing part before shuffled
yNormaltest = yNormal(round(1/19*18*size(yNormal,2))+1 :
size(yNormal,2)); %reversed predicted output at testing part

%%%shuffled
RMSEtrain1Sf = sqrt(mean((tNormaltrain -y1train).^2))
RMSEtestNormalSf = sqrt(mean((tNormaltest -y1test).^2))

%%%non shuffled
RMSEtrain1 = sqrt(mean((t1train -yNormaltrain).^2))
RMSEtestNormal = sqrt(mean((t1test -yNormaltest).^2))

%%



% figure
% plot (tNormaltrain)
% hold on
% plot (y1train)
%
% figure
% plot (tNormaltest)
% hold on
% plot (y1test)
%
% figure
% plot (t1train)
% hold on
% plot (yNormaltrain)
%
% figure
% plot (t1test)
% hold on
% plot (yNormaltest)
```

```
%% train fast speed

perf2= zeros(20,2);

 seed2 = 18 % best of 20
% for seed2=1:20
    rng(seed2)

trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.

% Create a Fitting Network
hiddenLayerSize = 6;
net = fitnet(hiddenLayerSize,trainFcn);

% Setup Division of Data for Training, Validation, Testing
 netnm2 = fitnet(hiddenLayerSize,trainFcn);
    netnm2.input.processFcns = {'removeconstantrows','mapminmax'};
    netnm2.output.processFcns = {'removeconstantrows','mapminmax'};
    %net.divideFcn = 'dividerand';  % Divide data randomly
    netnm2.divideFcn = 'divideind';
    netnm2.divideMode = 'sample';  % Divide up every sample
%     net.divideParam.trainRatio = 80/100;
%     net.divideParam.valRatio = 10/100;
%     net.divideParam.testRatio = 10/100;
%%

    netnm2.divideParam.trainInd =
[1:round(1/19*17*size(xNormal,2))];
    netnm2.divideParam.valInd =  [round(1/19*17*size(xNormal,2))+1 :
round(1/19*18*size(xNormal,2))];
    netnm2.divideParam.testInd = [round(1/19*18*size(xNormal,2))+1 :
size(xNormal,2)];
    netnm2.performFcn = 'mse';  % Mean Squared Error
    netnm2.plotFcns = {'plotperform','plottrainstate','ploterrhist',
...
        'plotregression', 'plotfit'};
% Train the Network
[netnm2,tr] = trainlm(netnm2,xFast,tFast);

% Test the Network
yFast = netnm2(xFast);
enm = gsubtract(tFast,yFast);
performance2 = perform(netnm2,tFast,yFast)

%%% y2 isthe prediced output
y2 = yFast;

%%% yFast after this line will be the inverse shuffled of the
predictedoutput
yFast(ind) = yFast;

% View the Network
% view(net)


    seed2
    testPerformance2 = perform(netnm2,performance2,yFast);
```

```
    perf2(seed2,1) = seed2;
    perf2(seed2,2) = testPerformance2;
    %[seed testPerformance]
% end


%% identify traing and testing set of data
%%%shuffled
tFasttrain = tFast(1:round(1/19*17*size(tFast,2))); %target output
at training part
y2train = y2(1:round(1/19*17*size(y2,2))); %predicted output at
training part

tFasttest = tFast(round(1/19*18*size(tFast,2))+1 : size(tFast,2));
%target output at testing part
y2test = y2(round(1/19*18*size(y2,2))+1 : size(y2,2)); %predicted
output at testing part

%%%non shuffled
t2train = t2(1:round(1/19*17*size(t2,2))); %target output at
training part before shuffled
yFasttrain = yFast(1:round(1/19*17*size(yFast,2))); %reversed
predicted output at training part

t2test = t2(round(1/19*18*size(t2,2))+1 : size(t2,2)); %target
output at testing part before shuffled
yFasttest = yFast(round(1/19*18*size(yFast,2))+1 : size(yFast,2));
%reversed predicted output at testing part

%%%shuffled
RMSEtrain1Sf = sqrt(mean((tFasttrain -y2train).^2))
RMSEtestFastSf = sqrt(mean((tFasttest -y2test).^2))

%%%non shuffled
RMSEtrain1 = sqrt(mean((t2train -yFasttrain).^2))
RMSEtestFast = sqrt(mean((t2test -yFasttest).^2))


%%


% figure
% plot (tFasttrain)
% hold on
% plot (y2train)
%
% figure
% plot (tFasttest)
% hold on
% plot (y2test)
%
% figure
% plot (t2train)
% hold on
% plot (yFasttrain)
%
% figure
% plot (t2test)
% hold on
% plot (yFasttest)
```

```
%% train slow speed

perf3= zeros(20,2);

 seed3 = 16 % best of 20
% for seed3=1:20
    rng(seed3)
trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.

% Create a Fitting Network
hiddenLayerSize = 6;
net = fitnet(hiddenLayerSize,trainFcn);

% Setup Division of Data for Training, Validation, Testing
 netnm3 = fitnet(hiddenLayerSize,trainFcn);
    netnm3.input.processFcns = {'removeconstantrows','mapminmax'};
    netnm3.output.processFcns = {'removeconstantrows','mapminmax'};
    %net.divideFcn = 'dividerand';  % Divide data randomly
    netnm3.divideFcn = 'divideind';
    netnm3.divideMode = 'sample';  % Divide up every sample
%     net.divideParam.trainRatio = 80/100;
%     net.divideParam.valRatio = 10/100;
%     net.divideParam.testRatio = 10/100;
%%

    netnm3.divideParam.trainInd =
[1:round(1/19*17*size(xNormal,2))];
    netnm3.divideParam.valInd =  [round(1/19*17*size(xNormal,2))+1 :
round(1/19*18*size(xNormal,2))];
    netnm3.divideParam.testInd = [round(1/19*18*size(xNormal,2))+1 :
size(xNormal,2)];
    netnm3.performFcn = 'mse';  % Mean Squared Error
    netnm3.plotFcns = {'plotperform','plottrainstate','ploterrhist',
...
        'plotregression', 'plotfit'};
% Train the Network
[netnm3,tr] = trainlm(netnm3,xSlow,tSlow);

% Test the Network
ySlow = netnm3(xSlow);
enm = gsubtract(tSlow,ySlow);
performance3 = perform(netnm3,tSlow,ySlow)

%%% y3 isthe prediced output
y3 = ySlow;

%%% ySlow after this line will be the inverse shuffled of the
predictedoutput
ySlow(ind) = ySlow;

% View the Network
% view(net)


    seed3
    testPerformance3 = perform(netnm3,performance3,ySlow);

    perf3(seed3,1) = seed3;
    perf3(seed3,2) = testPerformance3;
```

```
    %[seed testPerformance]
% end

%% identify traing and testing set of data
%%%shuffled
tSlowtrain = tSlow(1:round(1/19*17*size(tSlow,2))); %target output
at training part
y3train = y3(1:round(1/19*17*size(y3,2))); %predicted output at
training part

tSlowtest = tSlow(round(1/19*18*size(tSlow,2))+1 : size(tSlow,2));
%target output at testing part
y3test = y3(round(1/19*18*size(y3,2))+1 : size(y3,2)); %predicted
output at testing part

%%%non shuffled
t3train = t3(1:round(1/19*17*size(t3,2))); %target output at
training part before shuffled
ySlowtrain = ySlow(1:round(1/19*17*size(ySlow,2))); %reversed
predicted output at training part

t3test = t3(round(1/19*18*size(t3,2))+1 : size(t3,2)); %target
output at testing part before shuffled
ySlowtest = ySlow(round(1/19*18*size(ySlow,2))+1 : size(ySlow,2));
%reversed predicted output at testing part

%%%shuffled
RMSEtrain1Sf = sqrt(mean((tSlowtrain -y3train).^2))
RMSEtestSlowSf = sqrt(mean((tSlowtest -y3test).^2))

%%%non shuffled
RMSEtrain1 = sqrt(mean((t3train -ySlowtrain).^2))
RMSEtestSlow = sqrt(mean((t3test -ySlowtest).^2))

%%
% figure
% plot (tSlowtrain)
% hold on
% plot (y3train)
%
% figure
% plot (tSlowtest)
% hold on
% plot (y3test)
%
% figure
% plot (t3train)
% hold on
% plot (ySlowtrain)
%
% figure
% plot (t3test)
% hold on
% plot (ySlowtest)

%  end

%%%NRMSE calculation

NRMSEnormal = 100.*RMSEtestNormal/(max(t1test)-min(t1test))
NRMSEfast = 100.*RMSEtestFast/(max(t2test)-min(t2test))
```

```
NRMSEslow = 100.*RMSEtestSlow/(max(t3test)-min(t3test))

TN =   t1test';
YN =   yNormaltest';
rN =   corrcoef(TN,YN)

TF =   t2test';
YF =   yFasttest';
rF =   corrcoef(TF,YF)

TS =   t3test';
```