



Final work carried out with the aim of obtaining the degree of Master in Biomedical Engineering by

**Florence GILSON (S191385)**

---

**Prediction of lower-body joint kinematics during running and walking using a portable inertial sensor**

---



Image source: [1]

*Supervisor :*  
Professor Cédric SCHWARTZ

University of Liège  
Faculty of Applied Sciences  
Academic year 2023 - 2024

# Acknowledgements

Before starting this thesis, I would like to express my gratitude to all those who, directly or indirectly, contributed to the completion of this work.

I sincerely thank my supervisor, Professor Cédric Schwartz, who made this project possible. Not only did he propose a fascinating and exciting topic, but he also guided me while allowing me significant freedom, which helped me develop greater autonomy. I am grateful to him for the time and effort he invested in this project and for granting me access to the Human Movement Analysis Laboratory. I would also like to thank him for allowing me to use and handle all the equipment necessary for the completion of this work.

I would like to express my gratitude to Ashraf Hassan, a PhD candidate at the University of Liège, for his invaluable assistance, insightful advice, and the time he dedicated to answering all my questions in a field, machine learning, that was almost entirely new to me.

I would also like to extend my gratitude to my friends who generously volunteered to participate in the essential measurements that formed the basis of this work. Thank you for your enthusiasm, perseverance, especially when things didn't go as planned, and for the time you so kindly devoted to me. Without them, the completion of this work would simply not have been possible.

I wish to give special mention to my boyfriend, who took the time to critically and kindly review my thesis. His insightful comments greatly improved this work, and I am deeply touched by his commitment to supporting me throughout this journey.

Finally, I would like to thank in advance the members of the jury who will take the time to read this thesis, and for their attention and consideration.

# Abstract

Analysis of joint kinematics plays a crucial role in various applications, ranging from optimizing sports performance to clinical rehabilitation. However, this analysis is traditionally conducted in controlled laboratory environments, requiring expensive and sophisticated equipment, which limits its accessibility to a broader audience. This study aims to overcome these constraints by exploring the use of more affordable portable sensors to provide accurate kinematic data in more practical and realistic settings.

The objective of this study is to predict the flexion angles of lower limb joints, both during running and walking, based on data provided by a single portable inertial sensor (IMU).

To achieve this goal, a significant portion of the work was dedicated to experimentation and the collection of data necessary for analysis. Twenty healthy volunteers participated in this study, each performing three running trials and three walking trials at different speeds on a treadmill. The participants were equipped with an optoelectronic sensor system to measure the angles of the three main joints of the lower limb, namely the hip, knee, and ankle. Additionally, two IMU sensors were placed on each participant at different locations to determine which offered the best results. These IMU sensors collected inertial data, such as linear acceleration and angular velocity.

Twelve feedforward neural networks (FNN) were then created and trained using these inertial data as model inputs and kinematic data as model outputs.

The results indicate that the middle foot and heel locations for the IMU are comparable in terms of the accuracy of lower limb kinematic prediction, with a slight preference for the middle foot location.

Furthermore, this study also demonstrates that lower limb kinematics can be predicted with satisfactory accuracy using feedforward neural networks, with minimal RMSE errors of  $2.616^\circ$  for the ankle,  $3.142^\circ$  for the knee, and  $3.8292^\circ$  for the hip during running. For walking, similar minimal RMSE errors were observed, with values of  $2.8807^\circ$  for the ankle,  $4.21^\circ$  for the knee, and  $4.5767^\circ$  for the hip.

The Pearson correlation coefficients, ranging from 0.789 to 0.930 ( $p < 0.001$ ) across all models, further validate the accuracy of the predictions.

However, a major limitation of this study is the significant variability in results between participants, making it difficult to generalize the models to a larger population. Additional research will be necessary to improve model generalization and reduce the observed errors, thus opening promising prospects for more accessible clinical and sports applications.

# Contents

<b>Introduction</b>	<b>1</b>
<b>1 State of the art</b>	<b>3</b>
1.1 Importance of joint kinematics	3
1.2 Reference technique for studying joint kinematics	5
1.3 Portable alternatives for measuring joint kinematics	7
1.3.1 Operating principle of Inertial Measurement Units (IMUs)	8
1.3.2 Approaches for analyzing data from IMUs	11
1.3.3 Use of a limited number of IMUs	12
1.4 Joint angles prediction using Artificial Neural Networks (ANNs)	13
<b>2 Materials and methods</b>	<b>18</b>
2.1 Protocol	18
2.1.1 Aims of the study	18
2.1.2 Participants	19
2.1.3 Instrumentation and marker placement	19
2.1.4 Description of the different test phases	21
2.1.5 Performing the test	23
2.2 Data processing	24
2.2.1 Preprocessing with Qualisys Track Manager®	24
2.2.2 Preprocessing with Visual3D™	25
2.2.3 Preprocessing in MATLAB®	28
2.3 Models development and optimization	30
2.3.1 Designing regression models using fitrnet	30
2.3.2 Hyperparameters optimization	32
2.3.3 Averaging hyperparameters for final model creation	35
2.4 Evaluation metrics	36
2.4.1 Root Mean Square Error (RMSE) and Mean Absolute Error (MAE)	37
2.4.2 Normalized Root Mean Square Error (nRMSE)	37
2.4.3 Violin plots	37
2.4.4 Pearson correlation coefficient	39
2.4.5 Segmentation into gait cycles	40
<b>3 Results and discussions</b>	<b>42</b>
3.1 Optimized Neural Network hyperparameters	42
3.2 Comparison of different IMU locations	43
3.2.1 Model 1: Ankle angle predictions during running	43
3.2.2 Model 2: Knee angle predictions during running	45
3.2.3 Model 3: Hip angle predictions during running	46
3.2.4 Model 4: Ankle angle predictions during walking	47
3.2.5 Model 5: Knee angle predictions during walking	48

---

3.2.6	Model 6: Hip angle predictions during walking . . . . .	49
3.3	Analysis of Pearson correlation coefficients . . . . .	50
3.3.1	Model 1: Ankle angle predictions during running . . . . .	50
3.3.2	Model 2: Knee angle predictions during running . . . . .	51
3.3.3	Model 3: Hip angle predictions during running . . . . .	51
3.3.4	Model 4: Ankle angle predictions during walking . . . . .	52
3.3.5	Model 5: Knee angle predictions during walking . . . . .	52
3.3.6	Model 6: Hip angle predictions during walking . . . . .	53
3.4	Summary of model performances . . . . .	53
3.5	Model predictions based on the gait cycle . . . . .	54
3.5.1	Model 1: Ankle angle predictions during running . . . . .	55
3.5.2	Model 2: Knee angle predictions during running . . . . .	58
3.5.3	Model 3: Hip angle predictions during running . . . . .	61
3.5.4	Model 4: Ankle angle predictions during walking . . . . .	64
3.5.5	Model 5: Knee angle predictions during walking . . . . .	68
3.5.6	Model 6: Hip angle predictions during walking . . . . .	73
<b>4</b>	<b>Limitations of this study</b>	<b>77</b>
4.1	Limitations related to the experimental phase . . . . .	77
4.2	Limitations related to data preprocessing . . . . .	78
4.3	Limitations related to Machine Learning models . . . . .	78
<b>5</b>	<b>Conclusions and prospects</b>	<b>80</b>
	<b>References</b>	<b>1</b>
	<b>Appendix: MATLAB® Codes</b>	<b>1</b>
.1	importData.m . . . . .	2
.2	main.m . . . . .	7
.3	optimizeNeuralNetworksWithSets.m . . . . .	10
.4	createSets.m . . . . .	13
.5	visualizeModelPerformances.m . . . . .	13
.6	cyclesDetection.m . . . . .	17

# List of Figures

1.1	Examples of multi-segment modeling of the human body for calculating body motion . . . . .	4
1.2	Planes and axes of movement . . . . .	5
1.3	Hip . . . . .	5
1.4	Knee . . . . .	5
1.5	Ankle . . . . .	5
1.6	Movements of the three main lower body joints in the sagittal plane . . . . .	5
1.7	Laboratory optoelectronic motion capture system, illustrating the infrared cameras and reflective markers used to measure joint kinematics . . . . .	6
1.8	Illustration of inertial measurement units (IMUs) . . . . .	7
1.9	Illustration of Newton's second law with a mass attached to springs in a 2D reference frame . . . . .	8
1.10	Acceleration measurement using a MEMS accelerometer with a single movable mass . . . . .	8
1.11	Illustration of the basic principle of a gyroscope and how the rotations (Roll: $\phi$ , Pitch $\theta$ , Yaw: $\psi$ ) occur around respective axes . . . . .	10
1.12	Inner and substrate representation relative to a moving mass . . . . .	11
1.13	Illustration of the two IMU locations compared in this work . . . . .	12
1.14	Example of an ANN: a feed-forward neural network . . . . .	13
1.15	Example of structure of a Convolutional Neural Networks (CNN) . . . . .	15
1.16	Example of structure of a Long Short-Term Memory (LSTM) network . . . . .	16
1.17	Overview of Artificial Neural Network types and Learning Approaches. . . . .	17
2.1	Schematic diagram of joint angle prediction protocol using an Artificial Neural Network (ANN). . . . .	19
2.2	Image from Visual3D showing placement of reflective markers (green) and IMU (orange): front view, back view and right side view respectively. . . . .	20
2.3	Representation of anatomical or calibration markers (blue spheres) and technical markers (green spheres) on a body segment . . . . .	20
2.4	Illustration of the forces and moments acting on the knee during a quadriceps contraction measured by a dynamometer. . . . .	22
2.5	Illustration of Qualysis calibration kit . . . . .	23
2.6	Schematic representation of the body marker structuring for the first participant during the static acquisition. . . . .	24
2.7	Scaled model for first participant. . . . .	25
2.8	Illustration of the frequency response of a Butterworth filter for different orders (n) . . . . .	26
2.9	Representation of generalized coordinates used to control the position and orientation of all body segments . . . . .	27
2.10	Schema of the "Angles.mat" table. . . . .	28
2.11	Schema of the "IMU.mat" table. . . . .	28
2.12	Illustration of running data assembly for the first three models. . . . .	29

---

2.13	sigmoid	31
2.14	tanh	31
2.15	ReLU	31
2.16	Activation functions used by <code>fitrnet</code>	31
2.17	Schematic representation of the gradient descent algorithm from	32
2.18	Illustration of the data structure for Bayesian optimization and for the first three models.	33
2.19	5-fold cross-validation.	34
2.20	Row split for the first iteration of running model (19 participants) cross-validation.	35
2.21	Row split for the first iteration of walking model (20 participants) cross-validation.	35
2.22	80-20 distribution of running data for a participant and a specific task.	36
2.23	Representation of the characteristics of whisker boxes (A) and violin diagrams (B).	38
2.24	Running data format for Pearson's coefficient calculation.	39
2.25	Example of a conventional approach for interpreting Pearson's coefficient	39
2.26	Illustrations of a walking (top) and running (bottom) cycle and the corresponding phases	40
3.1	Illustration of the Bayesian optimization process for hyperparameter selection.	42
3.2	RMSE	43
3.3	nRMSE	43
3.4	MAE	43
3.5	Analysis of error distributions: Root Mean Squared Error (RMSE), normalized Root Mean Squared Error (nRMSE) and Mean Absolute Error (MAE).	43
3.6	RMSE	45
3.7	nRMSE	45
3.8	MAE	45
3.9	Analysis of error distributions: Root Mean Squared Error (RMSE), normalized Root Mean Squared Error (nRMSE) and Mean Absolute Error (MAE).	45
3.10	RMSE	46
3.11	nRMSE	46
3.12	MAE	46
3.13	Analysis of error distributions: Root Mean Squared Error (RMSE), normalized Root Mean Squared Error (nRMSE) and Mean Absolute Error (MAE).	46
3.14	RMSE	47
3.15	nRMSE	47
3.16	MAE	47
3.17	Analysis of error distributions: Root Mean Squared Error (RMSE), normalized Root Mean Squared Error (nRMSE) and Mean Absolute Error (MAE).	47
3.18	RMSE	48
3.19	nRMSE	48
3.20	MAE	48
3.21	Analysis of error distributions: Root Mean Squared Error (RMSE), normalized Root Mean Squared Error (nRMSE) and Mean Absolute Error (MAE).	48
3.22	RMSE	49
3.23	nRMSE	49
3.24	MAE	49
3.25	Analysis of error distributions: Root Mean Squared Error (RMSE), normalized Root Mean Squared Error (nRMSE) and Mean Absolute Error (MAE).	49
3.26	Ankle	54
3.27	Knee	54
3.28	Hip	54

---

3.29	Movements of the three main lower body joints in the sagittal plane . . . . .	54
3.30	Illustration of expected ankle flexion angles during a running cycle . . . . .	55
3.31	Predicted and true mean ankle cycles for "Run100". . . . .	56
3.32	Predicted and true mean ankle cycles for "Run120" . . . . .	57
3.33	Illustration of expected knee flexion angles during a running cycle . . . . .	58
3.34	Predicted and true mean knee cycles for "Run100". . . . .	59
3.35	Predicted and true mean knee cycles for "Run120". . . . .	60
3.36	Illustration of expected hip flexion angles during a running cycle . . . . .	61
3.37	Predicted and true mean hip cycles for "Run100" . . . . .	62
3.38	Visual3D hip angle signals for participant 1 and "Run100" before filtering (left) and after filtering (right). . . . .	63
3.39	Predicted and true mean hip cycles for "Run120" . . . . .	64
3.40	Illustration of expected ankle flexion angles during a walking cycle . . . . .	65
3.41	Predicted and true mean ankle cycles for "Walk100" . . . . .	66
3.42	Predicted and true mean ankle cycles for "Walk120" . . . . .	67
3.43	Predicted and true mean ankle cycles for "Walk80" . . . . .	68
3.44	Illustration of expected knee flexion angles during a walking cycle . . . . .	69
3.45	Predicted and true mean knee cycles for "Walk100". . . . .	70
3.46	Predicted and true mean knee cycles for "Walk120". . . . .	71
3.47	Predicted and true mean knee cycles for "Walk80". . . . .	72
3.48	Illustration of expected hip flexion angles during a walking cycle . . . . .	73
3.49	Predicted and true mean hip cycles for "Walk100". . . . .	74
3.50	Predicted and true mean hip cycles for "Walk120". . . . .	75
3.51	Predicted and true mean hip cycles for "Walk80". . . . .	76



# List of Tables

2.1	Participants data . . . . .	24
2.2	Description of the models with their corresponding input, output data, and IMU locations. . . . .	29
3.1	Optimal hyperparameters for each model and IMU localization. . . . .	43
3.2	Pearson's correlations for the Model 1. . . . .	50
3.3	Pearson's correlations for the Model 2 . . . . .	51
3.4	Pearson's correlations for the Model 3 . . . . .	51
3.5	Pearson's correlations for the Model 4 . . . . .	52
3.6	Pearson's correlations for the Model 5 . . . . .	52
3.7	Pearson's correlations for the Model 6 . . . . .	53
3.8	Summary of the models' performances, including the best IMU location, and error statistics. . . . .	54

# Introduction

Human movement has always captivated attention, from the dawn of humanity to the present day. In prehistoric times, this interest was expressed through frescoes carved on rock walls, reflecting a fascination with the body in motion [2]. Today, thanks to technological advances, kinematic analysis of human movement allows for precise capture and quantification of every bodily motion, offering a vision akin to a slow-motion film where each gesture is broken down in time and space, independent of the muscular forces or underlying mechanisms that produce them.

Joint kinematics, which focuses specifically on the movements of joints, finds applications in various fields, ranging from medical rehabilitation to the optimization of athletic performance. Traditionally, these analyses are conducted in specialized laboratories using sophisticated and expensive equipment. While these methods offer high precision, they limit accessibility to a broader audience and remain somewhat unrepresentative of real-world conditions. For instance, in clinical settings, accidental falls may be rare but can have severe consequences. However, even a detailed laboratory evaluation may not always capture the precise causes of these incidents, as unforeseen events in daily life do not necessarily occur under observation. Continuous monitoring outside of a controlled environment would be necessary to obtain a comprehensive risk assessment. Similarly, for a high-level athlete seeking to optimize performance, even the most advanced treadmill cannot fully replicate the natural dynamics of outdoor running, where terrain variations, weather, and other factors may influence movements.

This thesis is framed within the context of expanding access to movement analysis beyond the confines of specialized laboratories. The proposed approach relies on using a limited number of portable inertial sensors (IMUs), which are both affordable and easy to use, combined with machine learning methods capable of handling complex and large datasets. This work focuses on the most commonly used means of locomotion by the majority of people daily, instinctively since early childhood, namely walking and its extension, running. These activities occur in the sagittal plane and primarily involve the hip, knee, and ankle joints. Therefore, the angles of these three joints will be studied here. The ultimate goal of this thesis is to accurately predict these three joint angles based on inertial data collected from a single IMU.

To achieve this objective, the work begins by delving into the context, establishing the theoretical and technical foundations essential for understanding the challenges related to predicting joint angles from inertial sensor data. This initial step also includes a critical analysis of previous research, thereby identifying the gaps that this study aims to address (Chapter 1).

The study then continues with a detailed description of the experimental protocol, carried out in a laboratory setting to ensure the precise data collection from twenty participants. Section 2.1 explores in detail the various tasks performed by the volunteers, as well as the infrastructure and instruments used. While the use of the laboratory is crucial at this stage to ensure the quality and accuracy of the collected data, the ultimate goal of this work is to develop predictive models that are robust enough to allow movement analysis outside of this controlled environment.

The following Section 2.3 explains in detail the process by which the collected data is transformed

into predictive models for estimating lower body joint angles. This process includes a step of hyperparameter optimization, ensuring that the models reach their optimal performance.

Chapter 3 then presents an analysis of the results obtained from the developed predictive models. This analysis is conducted through statistical metrics such as RMSE (Root Mean Square Error), MAE (Mean Absolute Error), nRMSE (normalized RMSE), and Pearson's correlation coefficient, as well as through biomechanical evaluation of the model predictions. The aim of this analysis is particularly to determine which IMU location provides the best predictions.

The limitations of this study are also highlighted, offering a clear perspective on the challenges encountered and the opportunities for future research (Chapter 4).

Finally, this thesis concludes by offering perspectives for the future, drawing both from the identified limitations and the potential future directions of this work (Chapter 5).

# Chapter 1

## State of the art

This first chapter provides an overview of the essential theoretical foundations necessary for understanding the prediction of joint angles using Artificial Neural Networks (ANN). It begins by exploring the significance of joint kinematics across various fields such as biomechanics, clinical diagnostics, and athletic performance. The chapter then delves into the reference technique for studying joint kinematics, namely optoelectronic systems, while highlighting their limitations, particularly in uncontrolled environments. This analysis supports the transition to more flexible and real-world applicable alternatives, such as Inertial Measurement Units (IMU). The operating principles of IMUs are subsequently explained, followed by a presentation of studies focusing on the use of a limited number of these sensors. Finally, the chapter introduces the use of ANNs for predicting joint angles, shedding light on the architectures commonly used in biomechanics. It also justifies the choice of network type selected for this study, based on comparisons with existing literature.

### 1.1 Importance of joint kinematics

Kinematics is a branch of mechanics that focuses on the movement of objects without considering the forces and moments that cause these movements [3]. In biomechanics, this tool allows for the precise analysis and quantification of human body movements, such as those observed during gait [4].

To apply the principles of kinematics in biomechanics, the human body is typically modeled as several distinct segments, as illustrated in Figure 1.1 [4]. Joint kinematics, on the other hand, specifically focuses on the relative movements between these segments. It is concerned with "how" different parts of the body move relative to one another, without addressing the causes of these movements, such as muscle forces or external influences. In other words, it focuses on how the body's segments change position and orientation, as well as their speed and acceleration, while leaving aside the "why" behind these movements [2, 3].

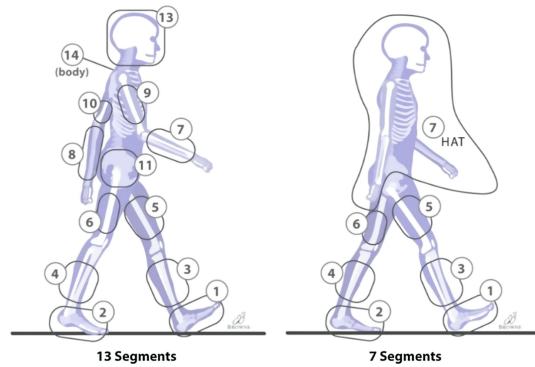


Figure 1.1: Examples of multi-segment modeling of the human body for calculating body motion [4].

The study of joint kinematics, thanks to technological advancements, now enables the decoding of complex movements, paving the way for numerous applications ranging from medical rehabilitation to the optimization of athletic performance.

In the field of rehabilitation, precise monitoring of rehabilitation programs is essential to ensure the correct execution of exercises and to measure patient progress, particularly in terms of range of motion [5]. Studies have shown that supervised rehabilitation exercises, where feedback is provided to the patient, are more effective than those performed without feedback [6, 7]. Thus, integrating an automated biofeedback system, such as an interface displaying the patient's actual movements in real-time compared to those prescribed by the clinician, can greatly enhance the effectiveness of the program. This type of feedback is not only beneficial for exercises performed at home, ensuring their proper execution, but also under clinical supervision, where visual observations can be subjective. Clinicians can also use these systems to retrospectively analyze kinematic data, such as joint trajectories or angles, to detect potential compensatory movements due to the pathology or to calculate relevant spatio-temporal parameters. Ultimately, this allows the therapeutic program to be adjusted according to the specific needs of each patient, optimizing the results.

Beyond the evaluation and monitoring of progress, the study of joint kinematics can also be considered an effective preventive tool. It allows for the diagnosis of abnormal gait or running patterns and can even serve as a simple, non-invasive, and affordable indicator of a person's general health status [8].

This is relevant not only for healthy individuals who wish to prevent the risks associated with improper gait but also for those facing aging, a natural and inevitable phenomenon that affects a growing population [8]. Aging indeed increases the risk of falls, a major concern for the elderly. Through kinematic analysis, it is possible to detect early alterations in movements, thereby enabling the implementation of preventive interventions to reduce these risks.

Finally, another key advantage of studying joint kinematics lies in the field of sports. In this context, even a slight improvement, such as a one-second increase or even a one-millimeter enhancement in an athlete's performance, can have a significant impact on the results. The collection of kinematic data during training sessions or competitions is essential for athletes and their coaches. This data allows for the precise identification of strengths and weaknesses in sporting techniques, thereby offering opportunities for performance optimization [9].

In this study, the objective is to analyze running and walking, with key movements originating from the lower body. Consequently, only the kinematics of the three main lower body joints,

namely the hip, knee, and ankle, will be examined. Additionally, these joint angles will be analyzed exclusively in the sagittal plane. This choice is a natural one, as the sagittal plane is where the primary flexion and extension movements characteristic of walking and running occur. Furthermore, it is the most studied plane in the literature for gait analysis, allowing for easier comparison of the results of this study with previous work. This plane is illustrated in red in Figure 1.2.

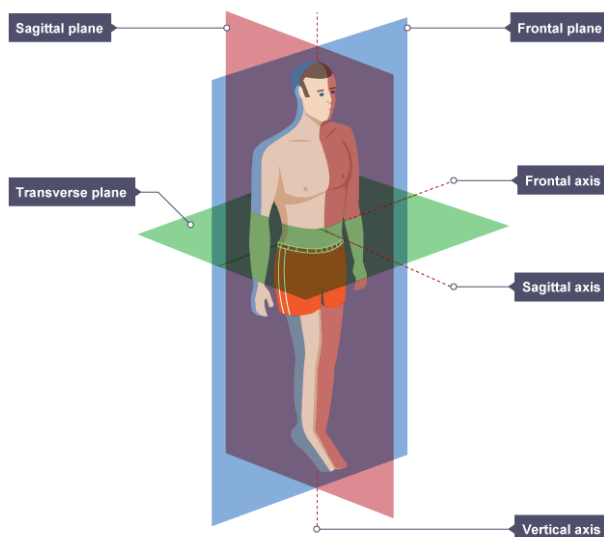


Figure 1.2: Planes and axes of movement [10].

In this plane, the movements of the hip, knee, and ankle joints are primarily flexion and extension (or dorsiflexion and plantarflexion for the ankle), as depicted in Figures 1.3, 1.4 and 1.5.

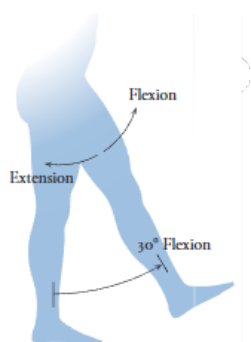


Figure 1.3: Hip

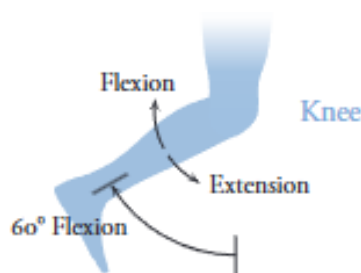


Figure 1.4: Knee

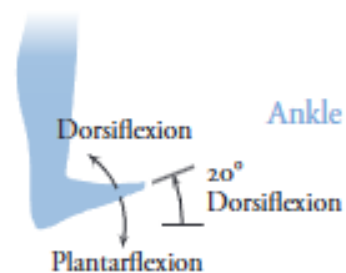


Figure 1.5: Ankle

Figure 1.6: Movements of the three main lower body joints in the sagittal plane [11].

## 1.2 Reference technique for studying joint kinematics

To accurately and non-invasively measure joint kinematics, the optoelectronic motion capture system is widely recognized as the gold standard in laboratory settings for human movement analysis. As illustrated in Figure 1.7, this system utilizes infrared cameras and reflective markers placed on specific points of the body to precisely track the trajectories of these markers in three dimensions [12].

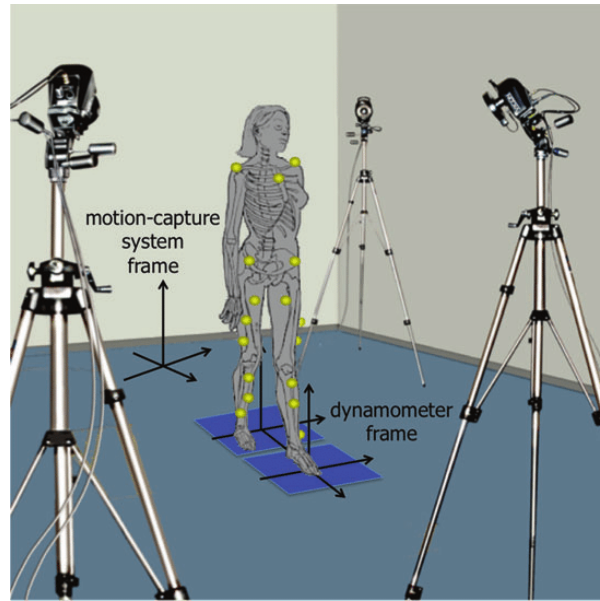


Figure 1.7: Laboratory optoelectronic motion capture system, illustrating the infrared cameras and reflective markers used to measure joint kinematics [13].

However, this system presents significant limitations for everyday use in clinical settings, at home, or outdoors. These constraints primarily stem from the need for strict control over lighting conditions to ensure that the infrared cameras operate optimally. Additionally, the installation of such a system requires a considerable amount of space to guarantee accurate motion capture, making it impractical for smaller or less controlled environments. Furthermore, the high cost of this equipment poses another significant barrier, restricting its use to specialized laboratories dedicated to human motion analysis [14].

It is also important to mention one of the main sources of error in human movement analysis, which also affects these optoelectronic systems, namely soft tissue artifacts (STA). These artifacts are errors caused by the relative movement between the markers placed on the skin and the underlying bones. Their magnitude can vary depending on several factors, including the task being performed, particularly the range of motion, the part of the body being studied, the anatomical characteristics of the subject, and the location of the markers, especially when placed near joints. Moreover, incorrect marker placement can significantly exacerbate these artifacts, highlighting the importance of precise positioning, which requires the presence of qualified technical or clinical personnel [12, 15, 16].

Additionally, to ensure optimal accuracy in measuring joint kinematics, it is often necessary to position a large number of markers and carry out calibration steps, making the process lengthy and complex [17].

These limitations underscore the need to develop more accessible alternative systems for measuring joint kinematics, allowing for more flexible use in clinical and everyday contexts [17].

In this study, the optoelectronic system will be used solely to accurately calculate the angles of the three main lower body joints. These angles will then serve as output data for machine learning models. The ultimate goal is that once these models are created and trained, the use of the optoelectronic system will no longer be necessary, as it remains difficult to employ for daily kinematic assessments due to the limitations mentioned earlier.

Regarding soft tissue artifacts (STA), which, as discussed earlier, limit the accuracy of optoelectronic systems, marker set techniques commonly used in practice will be implemented to minimize them as much as possible. These techniques will be further detailed in Section 2.1 of this work [12, 15].

Additionally, another frequent source of errors in optoelectronic systems involves systematic and random errors, also known as photogrammetric errors. These errors can be caused by improper camera calibration, noise in the data, or inappropriate lighting conditions. To mitigate these, digital filtering will also be applied here, as these errors typically manifest at high frequencies [15].

### 1.3 Portable alternatives for measuring joint kinematics

Numerous studies have highlighted the growing interest in using inertial measurement units (IMUs) as a portable alternative to traditional optoelectronic systems for measuring joint kinematics [8, 9, 14, 17–35].

IMUs offer several notable advantages. They are cost-effective and small and lightweight enough to be attached directly to various parts of the body without hindering the person’s movements, allowing for continuous monitoring in various environments, whether indoors or outdoors. Moreover, their ability to transmit data wirelessly makes them particularly well-suited for real-time monitoring, enabling immediate tracking and analysis of movements. This capacity to provide instant feedback and operate without requiring complex setup or extended preparation time makes them a valuable tool for both sports and clinical applications [17].

These small, portable sensors, illustrated in Figure 1.8, have already proven their usefulness in scientific literature, particularly in the field of sports. For example, a 2018 review, which compiled 286 studies and 23 reviews, demonstrated that IMUs are a viable solution for extending and enhancing athletes’ careers by offering better injury prevention and more tailored and specific training [35]. Furthermore, a more recent review, published in 2021, confirmed that IMUs represent a promising alternative for real-time kinematic analysis in the field, thereby facilitating a more accurate and immediate evaluation of athletic performance [9].

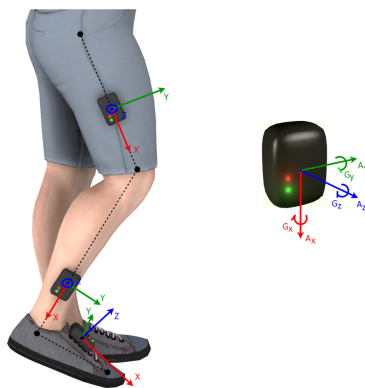


Figure 1.8: Illustration of inertial measurement units (IMUs) [36].

In a clinical context, although further research is necessary, IMUs have also shown promising potential. For example, one study revealed that IMUs can accurately predict knee kinematics in patients with osteoarthritis, although this study was limited by a sample of participants with low BMI, reducing the generalizability of the results [37]. Additionally, a review focusing on participants



with degenerative knee disorders emphasized the future need to evaluate a wider range of tasks and to expand the analysis to other joints to obtain more comprehensive results [38]. Moreover, another study explored the use of IMUs to monitor rehabilitation exercises for the hip and knee joints, although it was limited to a small sample of healthy participants [5].

### 1.3.1 Operating principle of Inertial Measurement Units (IMUs)

These portable devices combine information from several electromechanical sensors, such as accelerometers, gyroscopes, and sometimes magnetometers, to accurately estimate the orientation and dynamics of body segments.

The data from magnetometers, in addition to that from accelerometers and gyroscopes, detect the Earth’s magnetic field. This allows the IMU to orient itself with respect to a global reference system. Moreover, the data from magnetometers help correct potential errors and improve the accuracy of the measurements obtained from the accelerometers and gyroscopes [22, 39].

However, several studies have shown that magnetometers can introduce significant errors because they are highly sensitive to local magnetic field disturbances, such as the presence of metals or nearby electronic equipment [40, 41].

To avoid these disturbances, it is often recommended to exclude magnetometer data, which makes the measurements independent of variations in the environmental magnetic field [25, 26, 42, 43]. Consequently, in this study, only data from accelerometers and gyroscopes will be used. The remainder of this section will therefore focus exclusively on the functioning of these two types of sensors.

#### 1.3.1.1 Functioning of the accelerometer

The accelerometer in IMU sensors measures linear acceleration in three orthogonal directions (X, Y, and Z), which is the change in velocity in these directions. The Z-axis is generally assumed to be aligned with gravity, and measurements in the X and Y orientations provide information on the orientations during a given movement [22]. For human motion analysis, MEMS (Micro-Electro-Mechanical Systems) accelerometers are commonly used, as they are smaller and lighter compared to other types of accelerometers. The accelerometer used in these systems is typically of the capacitive type [44], which measures changes in capacitance resulting from the movement of an internal mass [45].

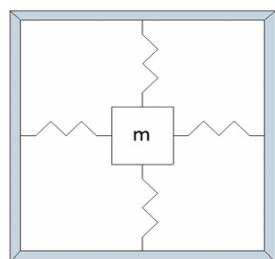


Figure 1.9: Illustration of Newton’s second law with a mass attached to springs in a 2D reference frame [46].

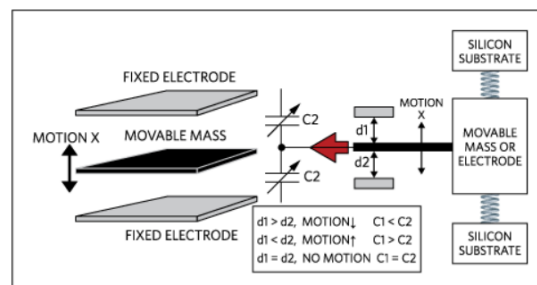


Figure 1.10: Acceleration measurement using a MEMS accelerometer with a single movable mass [47].

More specifically, the operating principle of the capacitive accelerometer is based on Newton’s second law and Hooke’s law (1.1). These laws allow for the calculation of acceleration  $\vec{x}$  by

detecting the force  $\vec{F}$ , exerted by a spring with a stiffness constant  $k$ , on a mass  $m$  suspended inside the sensor (Figure 1.9). The deformation of the spring  $\vec{x}$  and the displacement of the movable mass alter the distance between the electrodes  $d$ , thereby changing the electrical capacitance  $C$  (1.2). This change in capacitance is then converted into an electrical signal proportional to the measured acceleration, which is subsequently processed by a microcontroller [44].

$$\text{Newton's Second Law: } \vec{F} = m \vec{x}$$

$$\text{Hooke's Law: } \vec{F} = -k \vec{x}$$

$$\begin{aligned} \text{Combining the two laws: } m \vec{x} &= -k \vec{x} \\ \iff \vec{x} &= -\frac{m}{k} \vec{x} \end{aligned} \quad (1.1)$$

<b>Measured capacitance:</b> $C = \frac{\epsilon_r \epsilon_0 A}{d}$ où $d = d_0 + \vec{x}$ <span style="float: right;">(1.2)</span>
--

- $\epsilon_0$  : Permittivity of free space
- $\epsilon_r$  : Relative permittivity of the material between the plates
- $A$  : Overlapping area of the electrodes
- $d$  : Current distance between the electrodes
- $d_0$  : Initial distance between the electrodes in the absence of deformation
- $\vec{x}$  : Displacement of the moving mass

Thus, in reality, the accelerometer does not directly measure linear acceleration but does so indirectly by detecting the displacement of a mass relative to fixed electrodes [47].

Figure 1.10 shows an example in which capacitive accelerometers are arranged in a differential pair. This configuration relies on a single moving mass, held by a mechanical spring between two fixed reference electrodes. By calculating the difference between the capacitances  $C_1$  and  $C_2$ , which respectively depend on the distances  $d_1$  and  $d_2$  relative to the movement  $x$  of the mass and the fixed electrodes, it is possible to calculate the displacement of the mass and its direction [47].

The double integration of the linear acceleration signals measured by the accelerometer in the three spatial directions ultimately allows the determination of the spatial position of the moving body segment to which the IMU sensor is attached [46].

### 1.3.1.2 Functioning of the gyroscope

The gyroscopic sensor measures angular velocity around three orthogonal axes (yaw, pitch, and roll, corresponding to the X, Y, and Z axes), meaning the rate of rotation of an object around these axes. It functions by measuring the rotational movements of the frame on which the gyroscope is mounted relative to a fixed reference system [45]. Figure 1.11 illustrates the rotations around the different axes.

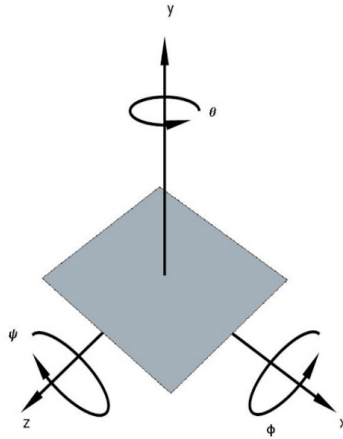


Figure 1.11: Illustration of the basic principle of a gyroscope and how the rotations (Roll:  $\phi$ , Pitch  $\theta$ , Yaw:  $\psi$ ) occur around respective axes [46].

The operation of MEMS (Micro-Electro-Mechanical Systems) gyroscopes is based on the Coriolis effect (1.3), which describes the Coriolis force  $\vec{F}_C$  generated when a mass oscillating at a certain frequency along one axis is subjected to a rotation  $\vec{\omega}$  perpendicular to this oscillation.

**Coriolis Effect:**  $\vec{F}_C = -2m(\vec{\omega} \times \vec{v})$  (1.3)

This Coriolis force, which occurs perpendicularly to the vectors  $\vec{v}$  (vibration velocity) and  $\vec{\omega}$  (rotation velocity), induces a displacement  $\vec{d}$  of the oscillating mass in a direction different from its initial movement. This displacement is used to measure rotation.

Figure 1.12 illustrates the operation of a MEMS gyroscope for a specific detection orientation. When the resonating mass, located in the inner frame, vibrates at a given frequency along the vertical axis, also known as the drive axis (yaw axis or Z-axis), and a rotation is applied around the horizontal Y-axis, which is perpendicular to the oscillation direction (pitch axis), a Coriolis force is generated in the direction of the X-axis (roll axis), perpendicular to both the vibration (Z) and rotation (Y) axes. This force induces a displacement  $\vec{d}$  of the inner frame in the direction of the X-axis.

This displacement is measured by detecting the changes in capacitance between the various electrodes of the outer frame, in a manner similar to an accelerometer (??). Once this displacement  $\vec{d}$  is measured, and knowing the spring constant  $k$ , the Coriolis force  $\vec{F}_C$  can be determined by the following relation 1.4:

$$\vec{d} = \frac{\vec{F}_C}{k} = \frac{-2m(\vec{\omega} \times \vec{v})}{k} \tag{1.4}$$

In this equation, the two remaining unknowns are the angular velocity  $\vec{\omega}$  and the vibration velocity of the mass  $\vec{v}$ . In practice, the vibration velocity is fixed and known, as the mass in these sensors is designed to vibrate at a specific predetermined frequency. The angular velocity can thus be determined and, once integrated, will provide the orientation of the body segments to which the IMU sensors are attached [47].

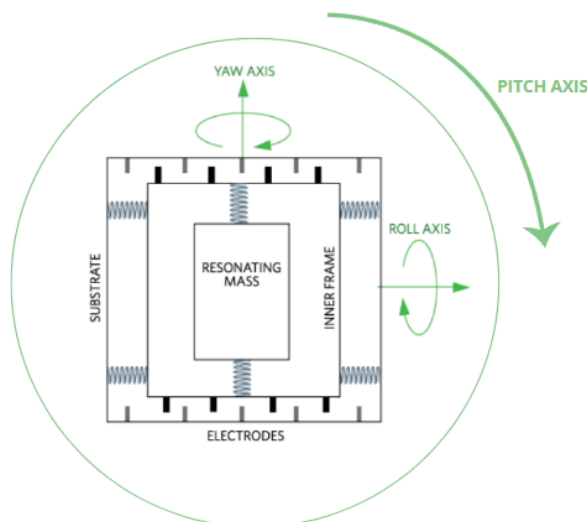


Figure 1.12: Inner and substrate representation relative to a moving mass [47].

### 1.3.2 Approaches for analyzing data from IMUs

Following the recording of signals from accelerometers and gyroscopes within Inertial Measurement Units (IMUs), two main methods for data processing are commonly used.

The first and more conventional method involves integrating and fusing data from both sensors (accelerometer and gyroscope). By integrating linear accelerations, one can obtain velocity (through single integration) and position (through double integration), while integrating angular velocities allows for determining the orientation of the body segments to which the inertial sensors are attached [44]. The fusion of this data enables even more accurate estimations of the movements of the segments on which the IMUs are placed.

To combine the data from these different sensors, sensor fusion algorithms are employed, with Kalman filters and complementary filters being the most popular in motion analysis systems [25]. The Kalman filter, for instance, uses a probabilistic model to estimate the true state of the system based on previous measurements and knowledge of the system's dynamics [48]. On the other hand, complementary filtering combines the strengths of the sensors to filter signals across different frequencies [21].

The major advantage of this fusion approach is that it compensates for the limitations that each sensor might present individually. For example, accelerometers are useful for stable long-term estimates as they provide a good inclination estimate, but they respond slowly to rapid movements. In contrast, gyroscopes are accurate in the short term but accumulate drift errors over longer periods. Thus, by combining the two, one obtains an orientation measurement that is both accurate and stable over time, while still being responsive to quick movements [45, 46, 49].

Therefore, unlike optoelectronic systems, which directly measure the absolute position of markers attached to the body, IMUs estimate the position of body segments indirectly by integrating signals from accelerometers and gyroscopes. However, these integration processes can introduce errors, such as the accumulation of bias, noise, drift, or other disturbances in the signals, which is a major drawback of this method [21, 22, 44, 46].

The second method, which is the one adopted in this work, involves directly using the raw data from accelerometers and gyroscopes to predict joint kinematics. This approach has the advantage of eliminating errors associated with integration steps. However, it requires the use of

a model capable of effectively processing this raw data and accurately inferring joint kinematics. The relationships between input data (linear accelerations and angular velocities) and output data (flexion-extension angles of the hip, knee, and ankle) are complex and non-linear, thereby necessitating the use of a machine learning model.

### 1.3.3 Use of a limited number of IMUs

Reducing the number of IMU sensors used across the body not only decreases costs and setup time but also makes the sensors less intrusive and more convenient for daily use.

In a recent review [27], focused on the use of machine learning for lower limb biomechanics during running, Xiang et al. (2022) demonstrated that one or two IMU sensors were sufficient to obtain reliable predictors or input variables for machine learning models.

However, in general, studies focusing on the prediction of lower limb joint kinematics using machine learning and inertial data, those using only a single IMU remain relatively rare.

Most of this research has focused exclusively on walking, whether on a treadmill [23, 28, 50] or over a short straight distance [30–32]. The IMU was generally placed on the sacrum, near the center of mass [28], on the lateral shank [30], at the ankle [32, 50], or on the foot [31].

On the other hand, very few studies have focused on running analysis. These studies [33, 34] have analyzed treadmill running with an IMU placed on the tibia. Meanwhile, another study [32] focused on running over short distances of a few meters, with an IMU placed at the ankle.

Furthermore, most of these studies involve a limited number of participants, typically fewer than 11 [28, 32–34, 50], highlighting a gap in the literature on this topic.

In this work, two different IMU locations will be compared, as shown in Figure 1.13. These locations, namely the ankle and the foot, were selected based on both the most commonly used positions in previous studies [27] and the insights gained from the Human Movement Analysis course taken this year [44].

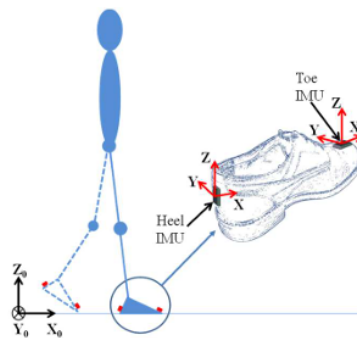


Figure 1.13: Illustration of the two IMU locations compared in this work [44].

It is important to note that the sensor positioned at the front of the foot will be placed near the shoe laces, rather than at the toe as shown in Figure 1.13. This decision was made to minimize interference from the flexion-extension movements of the forefoot, which could affect measurement accuracy.

## 1.4 Joint angles prediction using Artificial Neural Networks (ANNs)

As mentioned earlier, using raw data from accelerometers and gyroscopes has the advantage of avoiding errors associated with the signal integration steps. However, to extract accurate predictions of joint kinematics, it is essential to use a machine learning model capable of capturing the complex relationships between the measured signals and the actual joint movements.

**Machine learning** is a branch of artificial intelligence that aims to enable machines to learn and improve autonomously from their experiences, without requiring explicit programming for each task. The primary goal is to reduce human intervention in performing repetitive or secondary tasks [51]. **Artificial intelligence** thus endows systems with the ability to solve problems and make decisions in a manner similar to humans [51].

In the context of human movement analysis, a type of machine learning model often used for predicting joint kinematics based on IMU data is the **Artificial Neural Network (ANN)** [23–31, 33, 34]. ANNs are particularly well-suited to this task due to their ability to model nonlinear and complex relationships [52].

The term "Neural" in Artificial Neural Networks comes from the fact that these networks are inspired by the structure of the human brain, which is composed of vast interconnected networks of neurons working together to process and interpret sensory information [25, 53].

These ANNs are typically represented by a network diagram, as illustrated in Figure 1.14.

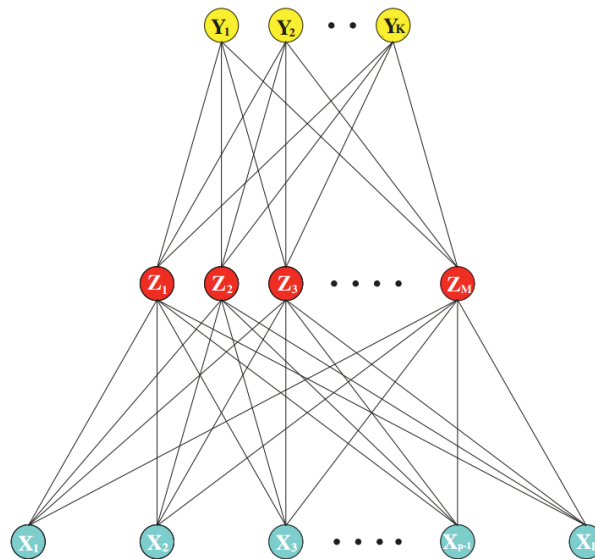


Figure 1.14: Example of an ANN: a feed-forward neural network [52].

In this diagram, each circle can be seen as a neuron. The blue circles represent the input variables ( $X_1, X_2, \dots, X_p$ ), and together they form the **first layer** of the network. The red circles represent combinations of these input variables ( $Z_1, Z_2, \dots, Z_M$ ) and are collectively known as the **hidden layers**. Finally, the yellow circles represent the model outputs ( $Y_1, Y_2, \dots, Y_k$ ) and together they constitute the **output layer**.

In this diagram, all the neurons in each layer are connected to all the neurons in the next layer, which is why these layers are described as **fully connected**. Such a network is called a **Fully-connected Neural Network**. The connections between neurons, represented by lines, each have a weight that can be adjusted during the training process, whether supervised or unsupervised,

allowing the network to learn complex data representations [52].

**Supervised learning** involves training a neural network on so-called "labeled" data, where each input is associated with a target output. The goal is to minimize the error between the model's predictions and the actual values by adjusting the weights of the neural connections [52]. Unlike unsupervised learning, which attempts to identify relationships, patterns, or groups in data without labeled outputs, supervised learning is ideal for tasks such as joint angle prediction. Here, the model learns from precise input signals (like those from IMUs) and known outputs (the measured joint angles) to produce reliable predictions on new data.

Two main techniques are used in learning: classification and regression [51]. Classification aims to predict a class or category for each input, while regression seeks to predict continuous values from one or more inputs, such as kinematic data.

Mathematically, neural networks rely on the basic principle of extracting linear combinations of the input data, known as "derived features," and then modeling the output data as a nonlinear function of these derived features [52].

If the input data is denoted by:

$$\mathbf{X}^T = (X_1, X_2, \dots, X_p),$$

where  $\mathbf{X}$  is a vector containing the different input features. Linear combinations of these inputs could take the form:

$$Z_m = \alpha_{m1}X_1 + \alpha_{m2}X_2 + \dots + \alpha_{mp}X_p,$$

with  $\alpha_{m1}, \alpha_{m2}, \dots, \alpha_{mp}$  representing the connection weights, which indicate the relative importance of each input in the creation of these new derived feature variables. These weights correspond to the connections, or lines, between the blue circles and the red circles in Figure 1.14. The derived features  $Z_m$  are then transformed using a nonlinear activation function  $\sigma$ , typically the sigmoid or ReLU function, to produce:

$$Z_m = \sigma(\alpha_{0m} + \alpha_m^T \mathbf{X}), \quad m = 1, \dots, M,$$

where  $\alpha_{0m}$  represents the constant bias term, added to give the model flexibility. Once these derived features are created, they are linearly combined using new weights  $\beta_k$  and the bias  $\beta_{0k}$  to produce the following intermediate values:

$$T_k = \beta_{0k} + \beta_k^T \mathbf{Z}, \quad k = 1, \dots, K, \quad \text{with} \quad \begin{cases} \mathbf{T} = (T_1, T_2, \dots, T_K) \\ \mathbf{Z} = (Z_1, Z_2, \dots, Z_M) \end{cases}$$

These  $\beta_k$  weights correspond to the connections, or lines, between the red circles and the yellow circles in Figure 1.14, determining the relative contribution of each derived feature to the final output variables. Finally, the final outputs can be produced by applying a nonlinear function  $g_k$  to the intermediate values  $T_k$  :

$$Y_k = g_k(\mathbf{T}), \quad k = 1, \dots, K.$$

In the context of this work, Artificial Neural Networks (ANN) are used to learn the relationships between the input data (data from IMUs) and the output data (kinematic data). To achieve this, training datasets, composed of these input data and their corresponding targets, are provided to the model so that it can train and subsequently be able to predict the output data based solely on IMU data that is different from the training data.

The three main categories of ANN commonly used for joint angles prediction are described below, along with their respective advantages and disadvantages.

**1.4.0.0.1 Feedforward Neural Networks (FNN)** represent the most classic class of ANN and are relatively simple to train. They are characterized by the transmission of information in a single direction, from input to output, as previously illustrated by Figure 1.14 and the associated mathematical equations, without any feedback loops where the model's outputs are fed back into the model itself [54]. In this type of network, each data point passes through the different layers of the network only once, with no possibility of going back. Each point is processed independently of the others, without considering the sequential order that may exist in time series data, such as those obtained from accelerometer and gyroscope sensors. This means that these networks are not inherently designed to handle the temporal dependency of the data. However, FNNs can still be used with temporal data by flattening these sequences, i.e., by combining all the data from each time step into a single vector. For example, if two 10-second running acquisitions are obtained for two different subjects by an accelerometer measuring acceleration along an X-axis at a frequency of 100 Hz,  $10 \times 100 = 1,000$  time steps will be obtained for each subject. Flattening these two sequences involves concatenating them into a single large column vector of size  $2 \times 1,000 = 2,000$ . This vector then represents an input feature for a model, where each value in this vector corresponds to an acceleration measurement at a given moment for one of the two subjects.

The main drawbacks of FNNs are their high computational cost, as well as the fact that they require inputs and outputs of the same size, in addition to requiring input normalization. These latter two drawbacks impose preprocessing steps between signal acquisition and the creation of FNN models, thus limiting their use in real-time applications [25].

Although several studies have already used these FNN models to predict kinematics from IMU sensors, they have all focused exclusively on walking [23–25, 28].

**1.4.0.0.2 Convolutional Neural Networks (CNN)** are traditionally used to process grid-structured data, typically images, which can be viewed as a 2D grid of pixels. These networks can also be applied to time series data, which can be considered as a 1D grid of samples taken at regular time intervals. As their name suggests, these networks use the mathematical operation of convolution instead of traditional matrix multiplication in at least one of their layers. This approach allows them to effectively capture spatial or temporal dependencies in the data by exploiting local relationships between neighboring elements [54]. Thus, unlike Feedforward Neural Networks, which use fully connected layers (Figure 1.14), CNNs employ convolutional layers where each neuron is connected only to a small region of the neurons in the previous layer to perform the convolution [55], as illustrated by Figure 1.15 below. Additionally, each convolutional layer applies a fixed set of weights, which differs from FNNs where each connection has its own weight [55].

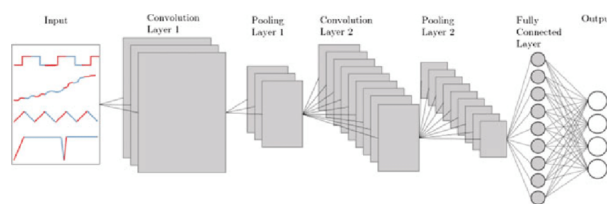


Figure 1.15: Example of structure of a Convolutional Neural Networks (CNN) [55].

One of the main drawbacks of CNNs is that they require a large dataset for effective model training, which can be challenging in biomechanics, where obtaining a sufficiently large experimental dataset is often difficult and complex [27].

Nevertheless, CNNs are widely recognized in the scientific literature as being among the most accurate models for estimating joint kinematics [24, 26, 33, 34].



**1.4.0.0.3 Recurrent Neural Networks (RNNs)** are particularly well-suited for processing sequential data with variable time periods. They are equipped with feedback loops that allow them to consider previous inputs when generating new outputs. This capability functions as an "internal memory," which stores information from previous time steps and uses it to update the network's state at each new step [55]. In the context of human motion analysis, a commonly used type of RNN is the Long Short-Term Memory (LSTM) network, which is specifically designed to better manage long-term dependencies in data. The Figure 1.16 below, illustrates the structure of such a network, including components like the cell state  $C_t$  (the LSTM's memory), the forget gate  $f_t$ , the input gate  $i_t$ , and the output gate  $o_t$ .

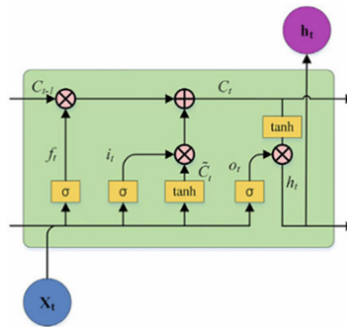


Figure 1.16: Example of structure of a Long Short-Term Memory (LSTM) network [55].

The main drawbacks of these models are their complexity in training and the large datasets they require [24].

Despite these challenges, LSTMs have been studied extensively in several research papers [25, 26, 30, 31].

The study by Mundt et al. (2021) [24] compared the performance of the three main classes of artificial neural networks for predicting kinematic data during walking, using IMU sensors for inputs and an optoelectronic system for outputs, an approach similar to that adopted in this work. The results revealed that, although FNNs are slightly less effective than CNNs in terms of performance, they have the advantage of requiring smaller datasets and simpler preprocessing steps compared to CNNs. Furthermore, FNNs demonstrated better accuracy than LSTMs in predicting joint angles.

Based on these findings, a Feedforward Neural Network (FNN), specifically a Fully-connected Neural Network (a type of FNN where all neurons are interconnected, as illustrated in Figure 1.14), will be used here. This choice offers an ideal balance between simplicity, performance, and efficiency, both in terms of the required data and the complexity of implementation for kinematic prediction.

The following diagram 1.17 provides an overview of the key concepts discussed earlier, showing how they relate to each other. Although not exhaustive, this diagram offers a useful visual structure to help understand the various concepts explored so far.

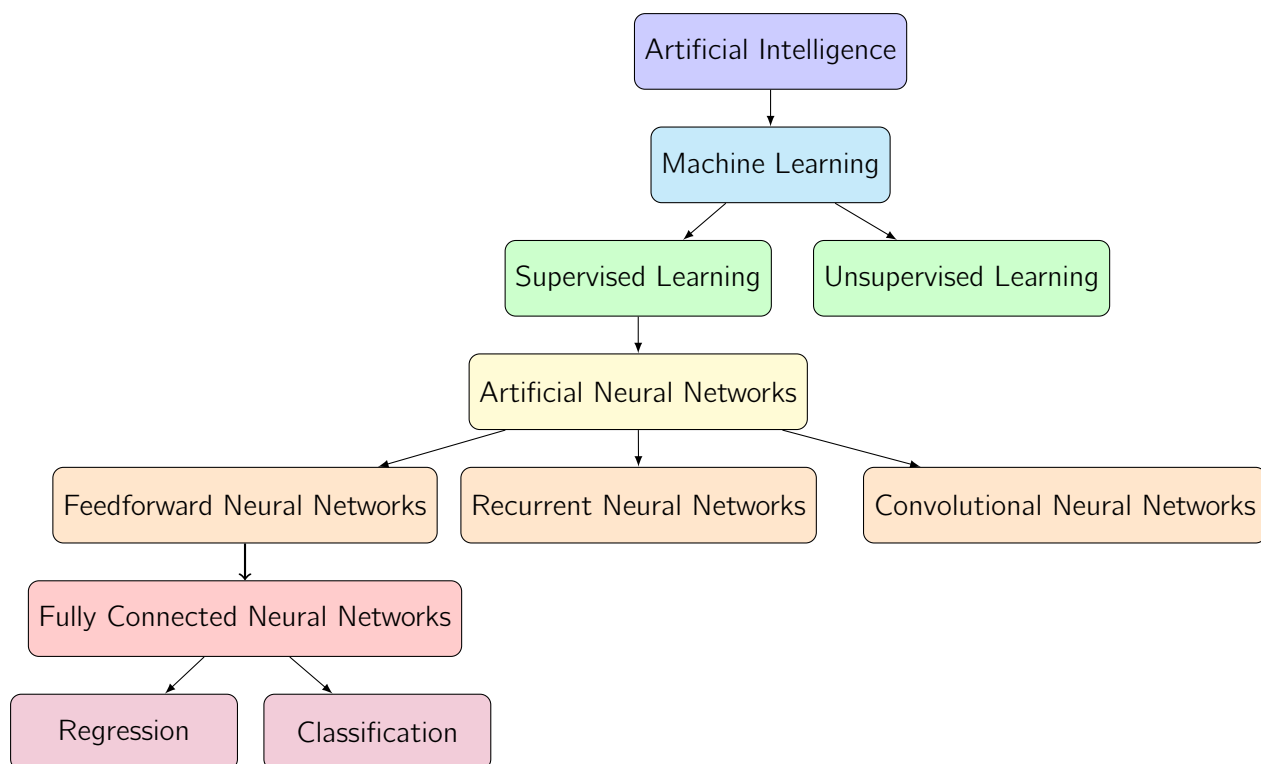


Figure 1.17: Overview of Artificial Neural Network types and Learning Approaches.

## Chapter 2

# Materials and methods

This chapter provides an in-depth overview of the essential steps that led to the collection, processing, and analysis of the data used in this study. It begins with a detailed description of the experimental protocol, including the collection of motion capture and inertial data. The characteristics of the participants, the selection and use of instrumentation, as well as the precise execution of the different test phases are thoroughly outlined. The chapter then delves into the processing of motion capture data, emphasizing the crucial preprocessing steps carried out with specialized software such as Qualisys Track Manager<sup>®</sup>, Visual3D<sup>™</sup>, and MATLAB<sup>®</sup>. The third section focuses on the development and optimization of regression models, explaining the application of the `fitrnet` function, Bayesian optimization strategies, and cross-validation methods. Finally, various evaluation metrics are presented to assess the performance of the developed models. This includes not only traditional calculations like root mean square errors (RMSE) but also advanced visualizations such as violin plots, which offer a more nuanced understanding of the results. The analysis of gait cycles helps to contextualize these results in a biomechanical perspective, providing a richer and more relevant interpretation of the predictions made.

## 2.1 Protocol

### 2.1.1 Aims of the study

This study aims, first and foremost, to collect motion capture data using an optoelectronic system, as well as inertial data through two IMU sensors. Subsequently, kinematic data will be calculated from the optical data using inverse kinematics techniques via the Visual3D software. Finally, the obtained kinematic data will be used to train an artificial neural network (ANN) model capable of predicting these same kinematic data based solely on inertial data from a single IMU sensor.

Figure 2.1 illustrates these different steps.

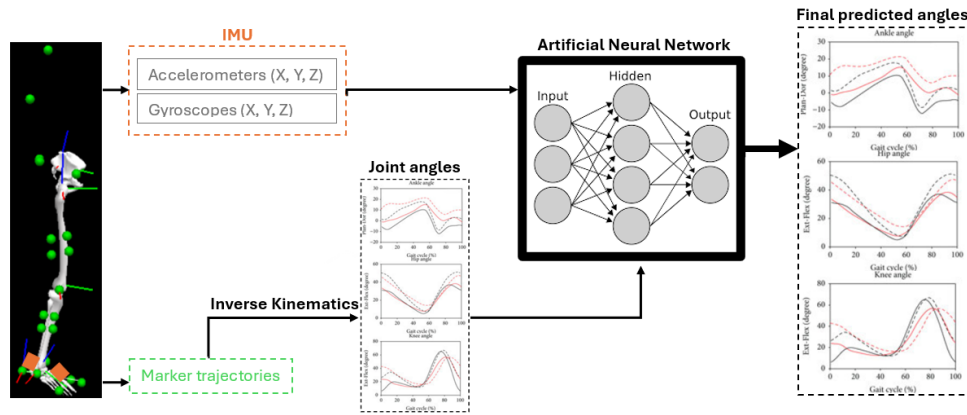


Figure 2.1: Schematic diagram of joint angle prediction protocol using an Artificial Neural Network (ANN).

### 2.1.2 Participants

Twenty healthy volunteers, including fourteen women and six men (aged  $22.6 \pm 1.93$  years, with an average weight of  $68.195 \pm 12.42$  kg and an average height of  $1.717 \pm 0.059$  m), were recruited to participate in this study. The participants met the following inclusion criteria: aged between 18 and 45 years, no recent pathology (less than 6 months) affecting the ankle, knee, or hip joints, no history of joint surgery, and no current pain in these areas.

The participants were equipped with fitted sports shoes and clothing to avoid any interference with the sensors. They attended the Human Movement Analysis Laboratory at Sart Tilman in Liège, where they provided consent for the collection of their personal data, such as weight, height, age, and the sport(s) they practice (this information is listed in Table 2.1). They also consented to the use of the data collected during the experimental sessions in the laboratory.

### 2.1.3 Instrumentation and marker placement

Three-dimensional motion analysis was conducted using 11 optoelectronic cameras (Qualisys) with a sampling frequency of 200 Hz, tracking the movements of 22 reflective markers placed on the upper body and the right leg only, as illustrated by the green spheres in Figure 2.2. For simplicity, it was assumed that participants exhibited symmetrical gait between both legs. Among these 22 markers, 14 were placed on the following anatomical landmarks: 1st and 5th metatarsal heads, hallux, Achilles tendon insertion, medial and lateral malleoli, medial and lateral femoral condyles, right and left anterior superior iliac spines, right and left posterior superior iliac spines, as well as the C7 and T8 vertebrae. The remaining eight markers were arranged in two sets of four markers each, with four markers being considered a practical compromise in this study [56]. One group was placed on the muscle mass of the tibia and the other on that of the femur.

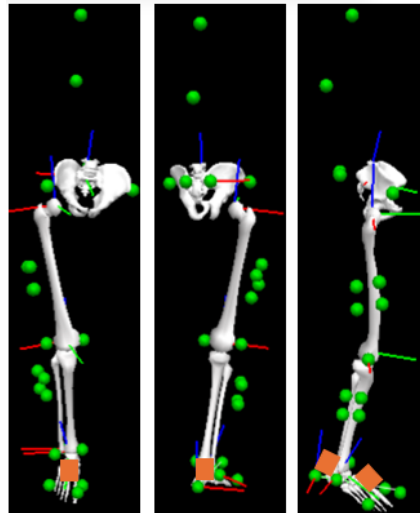


Figure 2.2: Image from Visual3D showing placement of reflective markers (green) and IMU (orange): front view, back view and right side view respectively.

Based on the recorded trajectories of these markers, the positions and orientations of the underlying bones can be reconstructed. More specifically, the 14 anatomical markers (represented by the blue spheres in Figure 2.3) are used solely for calibrating and defining the anatomical reference frame (or local reference frame) at the extremities of a segment, as they are placed where bony landmarks are palpable. However, these landmarks coincide with areas where soft tissue artifacts (STA) are most prevalent, particularly at the joints, where movement is most pronounced. To address this issue, the position of this anatomical reference frame relative to the technical reference frame (depicted in yellow in Figure 2.3), located at the center of the body segment, is assumed to be invariant over time. This assumption is based on the premise that the shape and structure of the segment are rigid and constant, regardless of the subject's movements. Therefore, this anatomical reference frame is primarily used during static calibration and is not directly involved in estimating segment movements. Nevertheless, it plays a crucial role in ensuring the reproducibility of measurements and allows for consistent comparison between subjects while expressing the results within a clinically understandable framework [12].

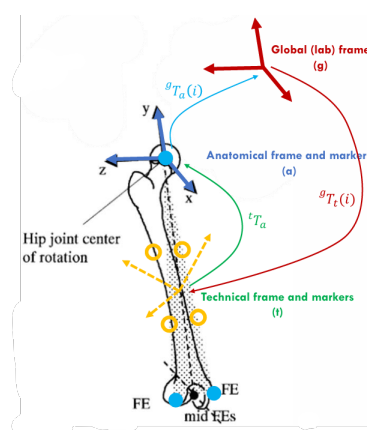


Figure 2.3: Representation of anatomical or calibration markers (blue spheres) and technical markers (green spheres) on a body segment [12].

The two groups of four technical markers (represented by the yellow spheres in Figure 2.3) are used to track segment movements [12]. While these markers, positioned in the middle of a segment rather than at the joint intersections, are less prone to soft tissue artifacts (STA) than

anatomical markers, they can still shift or deform relative to the underlying bone during movement due to the presence of passive and active soft tissues between the markers and the bone. Despite this, technical markers are crucial because they introduce redundancy into the data, enabling the application of optimization techniques to detect and minimize the impact of deformations and errors, including those affecting the anatomical markers. This redundancy arises from the fact that a given segment has six degrees of freedom (three for position and three for orientation), while each marker provides three degrees of freedom (x, y, z). Therefore, with a group of four markers on a segment, 12 degrees of freedom are available for the 6 required by the segment. Optimization can then compare minor errors or variations among these four markers, detect potential inconsistencies, and correct them [56].

The anatomical markers positioned on the metatarsal, hallux, and Achilles tendon were placed over the participants' shoes.

Simultaneously, inertial data were collected using two Trigno Avanti IMU sensors (Delsys Europe) [57], attached at two specific locations on the right leg of each participant: at the midfoot (in the laces of the shoes) and at the Achilles tendon insertion, secured with tape. These locations are illustrated by the orange rectangles, which have been deliberately enlarged for better visualization in Figures 2.16. These sensors have a sampling frequency of 2000 Hz.

Finally, a treadmill was installed at the center of the 11 cameras.

## 2.1.4 Description of the different test phases

### 2.1.4.1 Walk analysis phase

Each subject completed three 30-second walking trials on a treadmill at three different speeds, specifically determined for each participant. The first session started at the participant's preferred walking speed, considered as their usual daily walking speed and referred to as "Walk100" in this study. Based on this speed, a second speed, called "Walk80" or "slow" walking, was calculated to correspond to 80 % of the preferred speed. Similarly, a third speed, called "Walk120" or "fast" walking, was set to 120 % of the reference speed "Walk100".

The choice of these three speeds allows the experiment to be tailored to the individual capabilities of the participants, which may vary according to their physical characteristics and athletic condition. This approach ensures that each participant walks at a comfortable speed ("Walk100"), while also allowing the observation of biomechanical adaptations when the speed is slightly increased ("Walk120") or decreased ("Walk80"). Moreover, these trials at different speeds increase the amount of data available for training the ANN model. The specific choice of "80%" and "120%" values was made to avoid including too fast a walk, which would resemble running, or too slow a walk, which would not reflect realistic walking conditions.

The specific speeds for each participant are listed in Table 2.1.

This walking phase also served as a warm-up for the next phase, which focused on treadmill running.

### 2.1.4.2 Running analysis phase

Similar to the walking phase, each subject completed three 30-second running trials on a treadmill, at three individual-specific speeds. A comfortable jogging speed for the participant was used to define the reference speed "Run100", which then served to determine the "slow" running speed

("Run80") and "fast" running speed ("Run120").

To ensure participants' safety during this running phase, a safety harness was attached above the treadmill, connecting the participant to the treadmill's infrastructure, helping to prevent the risk of falling.

### 2.1.4.3 Knee extension moment analysis

The knee extension moment of the right leg was measured for each participant to assess their general physical capabilities. This moment is generated by the quadriceps force around the knee joint and thus represents an indicator of quadriceps strength. The quadriceps muscles, located at the front of the thigh, are responsible for knee extension and hip flexion, which are essential movements during walking and running [58].

Although these extension moment values were not directly used for training the machine learning models, they provide an assessment of the participants' overall physical condition and help contextualize the results obtained during the walking and running phases. Additionally, this information could prove useful in future analyses to study the links between muscle strength and movement kinematics.

For this test, a handheld dynamometer and a physiotherapy table were used. Participants sat on the table with the knee flexed at 90°, and the popliteal fossa firmly pressed against the table. The dynamometer was positioned at the participant's tibia, and the distance between the external condyle and the center of the dynamometer platform was measured to determine the lever arm.

Before the actual measurements, familiarization tests were conducted, consisting of a contraction at 50 % of maximum strength, followed by a maximum contraction. Then, three maximum contraction trials were performed, with the contraction held for 5 seconds each time. If a progressive increase in force was observed, a fourth trial was conducted. The highest value among the three or four trials was recorded as the final measure of quadriceps strength. Figure 2.4 illustrates the setup of this test as well as the forces and moments involved.



Figure 2.4: Illustration of the forces and moments acting on the knee during a quadriceps contraction measured by a dynamometer.

The knee extension moment (represented in blue in Figure 2.4) was calculated by multiplying

the force measured by the dynamometer (represented in yellow) by the lever arm (represented in purple). This moment was then normalized by the BMI  $\left(\frac{\text{Weight}[\text{kg}]}{\text{Height}^2[\text{m}^2]}\right)$  of each participant to allow comparisons between different participants. The values of these moments are summarized in Table 2.1.

### 2.1.5 Performing the test

First, the calibration of the cameras and optoelectronic markers was carefully conducted to ensure accurate motion capture on the treadmill. This calibration process for the optoelectronic system involves informing the computer of the precise locations of the cameras relative to each other and to the position of the treadmill in space. For this, an L-shaped square equipped with reflective markers (Figure 2.5) is placed at the center of the treadmill. This square helps define the X, Y, and Z axes of the calibrated space. Then, a carbon fiber wand with reflective markers at its ends is waved throughout the space occupied by the treadmill, allowing the system to locate each camera relative to the square [59].

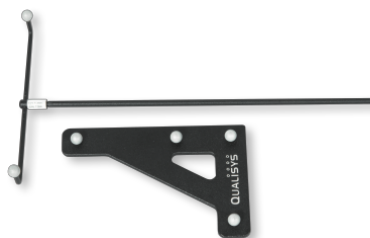


Figure 2.5: Illustration of Qualysis calibration kit [59].

Once the three-dimensional capture space was properly defined, the two IMU sensors were synchronized and placed on the participant. Then, the 22 reflective markers were installed.

The recording session began with a static trial aimed at estimating the alignments between the markers and the corresponding body segments. For this, the subject was asked to stand still on the treadmill, with arms and legs slightly apart, for a few seconds.

After the static recording was completed, the treadmill speed was gradually increased to reach the participant's comfortable walking speed, referred to as "Walk100". From this speed, the walking speeds at 80% and 120% were calculated and then successively applied to the participant, with each phase lasting 30 seconds. A few seconds of transition were given to the participant at each speed change to allow adaptation to the new pace.

Similarly, the three running phases were calculated and performed by the participant, with or without breaks between the different speed levels, depending on individual preferences.

The data were collected directly via the Qualysis Track Manager 3D analysis software for later processing.

The total duration for completing all three tests (walking, running, and quadriceps strength measurement) is approximately 1.5 hours per participant.

Table 2.1 summarizes the characteristics of each participant, as well as their strength test results and speeds for each task.



No.	Sex	Year of birth	Height [m]	Weight [kg]	Sport(s)	Lever arm [m]	Strength test [N]	Quadriceps strength [Nm/kg/m <sup>2</sup> ]	Walking speed [m/s]			Running speed [km/h]		
									100%	80%	120%	100%	80%	120%
1	M	2002	1.80	90	Weightlifting	0.34	743	9.094	3.5	2.8	4.2	10.5	8.4	12.6
2	M	2004	1.75	83	Weightlifting Wrestling	0.33	766.60	9.193	4.5	4.3	5.4	10.1	8	12.12
3	F	2001	1.65	78	Running Dance	0.30	238.00	2.492	3.4	2.72	4.08	8.8	7.04	10.56
4	F	2001	1.68	54	Synchronized Swimming Yoga Running	0.28	353.70	5.176	3.2	2.56	3.84	6.8	5.44	8.16
5	F	2001	1.68	54	Synchronized Swimming Walking Dance	0.28	342.10	5.007	4.0	3.2	4.8	6.6	5.28	7.92
6	F	1999	1.72	68	-	0.30	334.0	4.941	1.8	1.44	2.16	6.5	5.2	7.8
7	M	2001	1.79	67	Cycling Running	0.36	440.40	7.581	4.3	3.44	5.16	11.8	9.44	14.16
8	F	2004	1.8	64	Tennis Running	0.33	203.30	3.345	3.5	2.8	4.2	8.9	7.12	10.68
9	F	2004	1.67	60	Gym Running	0.33	301.20	4.62	4.4	3.52	5.28	8.8	7.04	10.56
10	F	2002	1.73	72	-	0.33	372.30	5.045	4.8	3.84	5.76	8.6	6.88	10.32
11	M	2001	1.72	70	-	0.33	280.3	3.85	3.3	2.64	3.96	7.7	6.16	9.24
12	F	2004	1.66	49	-	0.27	323.00	4.904	4	3.2	4.8	8.8	7.04	10.56
13	F	2001	1.75	65	Weightlifting	0.33	457.30	7.002	3.9	3.12	4.68	8.2	6.56	9.84
14	F	2000	1.75	68	Weightlifting Dance	0.33	454.20	6.75	3.5	2.8	4.2	7.5	6	9
15	F	2000	1.69	54	Weightlifting	0.32	390.6	6.611	2.7	2.16	3.24	6.1	4.88	7.32
16	M	1997	1.81	72	Swimming Cycling	0.31	465.20	6.561	2.6	2.08	3.12	6.8	5.44	8.16
17	F	2001	1.63	64	Weightlifting	0.31	295.00	3.796	3.8	3.04	4.56	7.5	6	9
18	F	2002	1.60	75	Running	0.3	395.00	4.449	2.9	2.32	3.48	5.2	4.16	6.24
19	F	1999	1.71	98	Walking	0.32	407.00	3.886	2	1.6	2.4	4.8	3.84	5.76
20	M	2004	1.75	58.9	-	0.32	335.90	5.588	2.7	2.16	3.24	7	5.6	8.4
<b>Average</b>	-	<b>22.6 [yrs]</b>	<b>1.717</b>	<b>68.195</b>	-	<b>0.32</b>	<b>394.91</b>	<b>5.495</b>	<b>3.44</b>	<b>2.787</b>	<b>4.128</b>	<b>7.85</b>	<b>6.276</b>	<b>9.42</b>
<b>Std Dev</b>	-	<b>1.93 [yrs]</b>	<b>0.059</b>	<b>12.42</b>	-	<b>0.02</b>	<b>142.26</b>	<b>1.81</b>	<b>0.813</b>	<b>0.714</b>	<b>0.975</b>	<b>1.749</b>	<b>1.394</b>	<b>2.099</b>

Table 2.1: Participants data

## 2.2 Data processing

### 2.2.1 Preprocessing with Qualisys Track Manager<sup>®</sup>

Qualisys Track Manager (QTM) is a motion capture software that allows for the immediate identification of markers placed on the body during acquisitions. Once the data is captured and recorded, the software also offers features to preprocess this data.

In this study, the markers placed on the participants were named and connected in a way that approximately represents the body structure. This setup was then used to process the data captured during the different tasks performed by each participant. An example of this structuring is presented in Figure 2.6 for the first participant during the static acquisition.

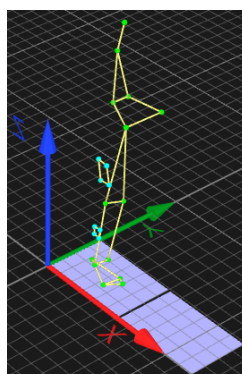


Figure 2.6: Schematic representation of the body marker structuring for the first participant during the static acquisition.

The trajectories of the markers for all tasks and all participants were visually analyzed. In case of deviations in the trajectories, these were corrected by interpolation. For deviations of 10

frames or less (as the data was sampled at 200 Hz, a 30-second trial contains 6000 frames), a polynomial interpolation was applied. For larger gaps, a relational interpolation was used. This latter method is based on selecting three markers considered fixed, relative to which the deviation of the concerned marker is corrected.

Once the marker trajectories were 100% complete, they were exported in the C3D 3D coordinate file format to undergo further preprocessing in the Visual3D software.

## 2.2.2 Preprocessing with Visual3D™

Visual3D is a biomechanical analysis software dedicated to the processing of 3D motion capture data. In this study, it was used for several key steps: first, to record the positions of the optical markers placed on an experimental subject and to match them with the markers of a musculoskeletal model by scaling the model's segments to fit the subject's anatomy. Next, Visual3D was used to filter the captured data to reduce noise. Finally, an optimization step, called inverse kinematics, was employed to calculate the kinematics of the body segments based on the trajectories of the experimental markers during movements.

### 2.2.2.1 Scaling the anatomical model

A generic Visual3D model (CMO file), provided by my supervisor, Professor Schwartz, and containing all the rigid segments of the right lower limb, such as the pelvis, femur, tibia, and foot, was scaled to match the anatomical landmarks obtained during the static trial of each participant.

In this model, the segments are explicitly linked by joint constraints, so the movement of one segment depends on the movement of others, while respecting the natural limits of the joints. For example, in the creation of this model in Visual3D, the right thigh segment was configured to have the pelvis as the parent segment. This thigh segment has no degrees of freedom in translation but can rotate around the three axes of rotation, meaning that the thigh is fixed to the pelvis without any possibility of translation but can rotate freely relative to it.

An example of this scaled model is shown in Figure 2.7 for the first participant.

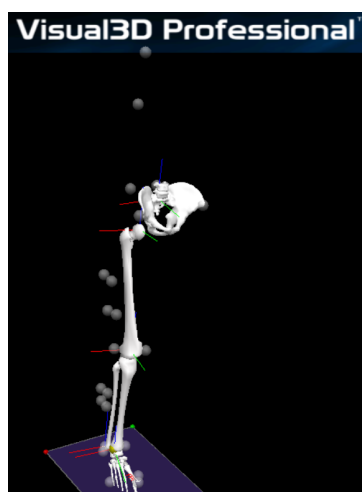


Figure 2.7: Scaled model for first participant.

During the static acquisition, which lasts a few seconds, Visual3D calculates an average of the marker positions over all the frames to compensate for noise in the data. The software then automatically adjusts each segment of the model based on the measured distances between the

anatomical markers. For example, the length of the femur is adjusted to match the distance between the markers placed on the anterior superior iliac spine (at the upper end) and the lateral or medial femoral condyle (at the lower end) of this segment.

Once this global scaling is completed, the model is associated with the other dynamic trials (Walk100, Walk80, Walk100 and Run100, Run80, Run120) by calculating the pose, i.e., the position and orientation, of each segment and the model's reference points for the movement trials. The signals from the different tasks are then ready to be processed.

### 2.2.2.2 Filtering of motion capture data

The signals corresponding to the different tasks were all filtered using a bidirectional Butterworth low-pass filter with a cutoff frequency of 10 Hz.

This filtering aims to reduce the noise present in the motion capture data, which typically occurs at higher frequencies than those associated with running or walking movements [12, 56, 60].

The low-pass Butterworth filter used here leaves frequencies below 10 Hz, associated with running and walking movements, intact while attenuating frequencies above 10 Hz. These higher frequencies may originate from soft tissue artifacts, poor marker visibility, or the motion capture system's sensitivity to electrical interference from other electronic devices [60].

The bidirectionality of the Butterworth filter means that it is applied twice, first in the forward direction (from the first to the last data point) and then in the reverse direction (from the last to the first data point). This corrects the time delay naturally present in Butterworth filters, due to the filtering being based on a weighted average of data points, which slightly delays the filtered signal relative to the raw signal. Filtering in both directions ensures proper temporal alignment of the data [60]. In Visual3D, the number of passes performed by the filter also determines the order of the Butterworth filter, that is, the degree of the differential equation describing it. In this case, it is of order 4, as is commonly used in biomechanics [12].

A 4th-order Butterworth filter is characterized by a faster and more intense attenuation of high frequencies compared to a lower-order filter [12], as illustrated in Figure 2.8.

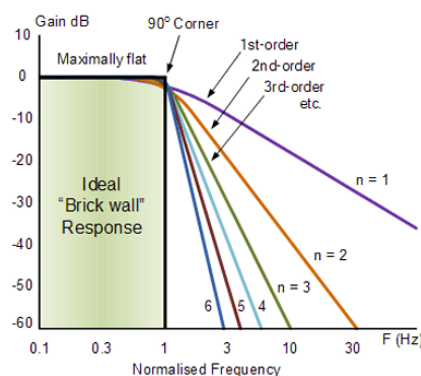


Figure 2.8: Illustration of the frequency response of a Butterworth filter for different orders ( $n$ ) [12].

### 2.2.2.3 Inverse kinematics

Inverse kinematics, in biomechanics, is an optimization process aimed at determining the positions and orientations of bone segments from the measured coordinates of markers placed on the body.

This process then allows for the calculation of joint angles between different segments.

In Visual3D, inverse kinematics is solved using a global optimization method. This method, proposed by Lu and O'Connor (1999) [61], is recognized for its accuracy and realism compared to segmental optimization methods and direct methods [12]. Global optimization treats body segments as rigid and imposes joint constraints on the model [12, 61]. This approach minimizes the weighted sum of squared distances between the positions measured during various tasks and the positions of the markers on the model, across all body segments.

Mathematically, the position of a point  $\mathbf{a}$  on a body segment, expressed in the anatomical or technical reference frame (represented in blue or yellow in Figure 2.3), can be transformed into the global reference frame (represented in red in Figure 2.3), noted as  $\bar{\mathbf{p}}$ , using the rotation matrix  $\mathbf{T}$  and the translation vector  $\mathbf{o}$  between the coordinate systems:

$$\bar{\mathbf{p}} = \mathbf{T}\mathbf{a} + \mathbf{o}. \quad (2.1)$$

Least squares optimization then minimizes the sum of squared errors between the measured position  $\mathbf{p}_i$  (motion capture data) and the calculated position  $\bar{\mathbf{p}}_i$  from the model, for each marker  $i$  on a given segment, expressed as:

$$\sum_{i=1}^m |\mathbf{p}_i - \mathbf{T}\mathbf{a}_i - \mathbf{o}|^2, \quad (2.2)$$

with  $m$  being the number of markers on the segment. This local optimization method is then extended to the entire body model by introducing the generalized coordinates  $\mathbf{q}$ . These generalized coordinates, shown in Figure 2.9, describe the overall configuration of the body, taking into account joint constraints and relationships between different segments. Thus, the two equations (2.1) and (2.2) become:

$$\bar{\mathbf{p}}_i(\mathbf{q}) = \mathbf{T}(\mathbf{q})\mathbf{a}_i + \mathbf{o}(\mathbf{q}),$$

$$\sum_{i=1}^{m_t} |\mathbf{p}_i - \mathbf{T}(\mathbf{q})\mathbf{a}_i - \mathbf{o}(\mathbf{q})|^2,$$

where  $m_t$  is the total number of markers across all segments.

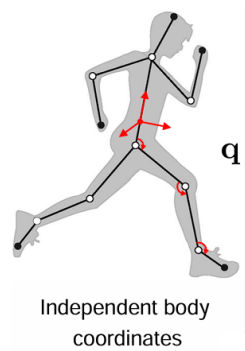


Figure 2.9: Representation of generalized coordinates used to control the position and orientation of all body segments [3].

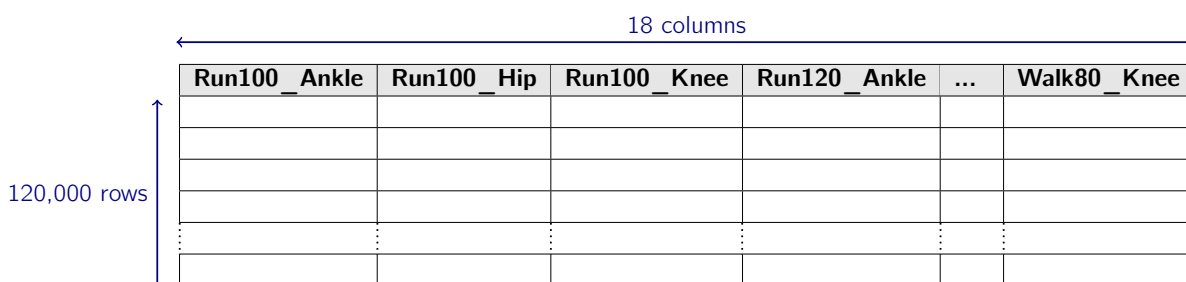
All these equations are derived from the Visual3D documentation on inverse kinematics [62].

Once the kinematic data are obtained, they are filtered again using a 10 Hz low-pass filter, and only the ankle, knee, and hip joint angle data in the sagittal plane are exported in ASCII format for subsequent analysis in MATLAB.

As for the IMU sensor data, all raw data recorded along the three axes were also exported in ASCII format.

### 2.2.3 Preprocessing in MATLAB®

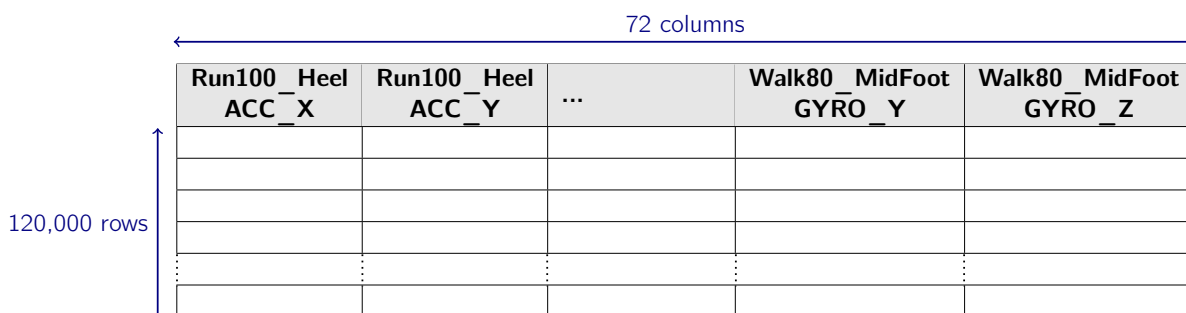
The kinematic data in ASCII format for all participants were exported as a MATLAB table. The "Angles.mat" table contains 30 seconds of acquisition at a sampling frequency of 200 Hz for 20 participants, which amounts to 120,000 rows. This table also includes 18 columns corresponding to 6 types of tasks ("Run100", "Run80", "Run120", "Walk100", "Walk80", "Walk120") and 3 types of angles (ankle, hip, knee). This table, illustrated in Figure 2.10, will serve as the output for the various models.



Run100_Angle	Run100_Hip	Run100_Knee	Run120_Angle	...	Walk80_Knee

Figure 2.10: Schema of the "Angles.mat" table.

The inertial data in ASCII format for all participants were initially sampled at 2000 Hz. To temporally synchronize this data with the kinematic data, which were sampled at 200 Hz, MATLAB's `resample` function was used to downsample the IMU data to 200 Hz. Then, similarly to the kinematic data, a table titled "IMU.mat" was created from the downsampled ASCII data. This table also contains 30 seconds of acquisition at a sampling frequency of 200 Hz for 20 participants, which represents a total of 120,000 rows. It contains 72 columns corresponding to the 6 types of tasks ("Run100", "Run80", "Run120", "Walk100", "Walk80", "Walk120"), 2 types of sensors (accelerometer and gyroscope), 3 spatial orientations (X, Y, and Z), and 2 sensor locations (heel and midfoot). This table, schematized in Figure 2.11, will serve as input to the different models.



Run100_Heel ACC_X	Run100_Heel ACC_Y	...	Walk80_MidFoot GYRO_Y	Walk80_MidFoot GYRO_Z

Figure 2.11: Schema of the "IMU.mat" table.

#### 2.2.3.1 Preparing and structuring data for learning

For the design of Fully-Connected Neural Network (FCNN) models, MATLAB's `fitrnet` function was used. Since this function only supports a single output variable per model, the data were structured accordingly.

A total of 12 distinct regression models were created to predict the 3 joint angles (ankle, knee, and hip) based on the 2 different IMU sensor positions (heel and midfoot) and whether

the data pertain to running or walking. To compare the different IMU sensor locations, models with the same input and output data but different sensor positions were grouped together in the same MATLAB data structure. This resulted in 6 model groups, each composed of the 2 different locations. Table 2.2 details the specific inputs and outputs associated with each model.

Model	INPUT (IMU)	OUTPUT (Angles)	IMU Locations
1	Running data	Ankle for running	Heel
			Mid foot
2	Running data	Knee for running	Heel
			Mid foot
3	Running data	Hip for running	Heel
			Mid foot
4	Walking data	Ankle for walking	Heel
			Mid foot
5	Walking data	Knee for walking	Heel
			Mid foot
6	Walking data	Hip for walking	Heel
			Mid foot

Table 2.2: Description of the models with their corresponding input, output data, and IMU locations.

For the assembly of running data, only the data from the "Run100" and "Run120" tasks were retained. The data from the "Run80" task, associated with a lower running speed, were excluded because some participants performed fast walking instead of running as initially intended, which would have compromised the homogeneity of the running data. Additionally, a signal loss of several seconds was observed for participant 16 during the "Run100" and "Run120" tasks. Therefore, it was decided to remove this participant from the running model analyses while keeping them for the walking models.

Next, the running data were reorganized to meet the requirements of the `fitrnet` function. The columns were structured to represent the sensor types (accelerometer or gyroscope) and their associated directions (X, Y, or Z), while the rows represented the sensor observations for all tasks combined, whether for "Run100" or "Run120". To achieve this structure, the data from "Run120" were added directly below those of "Run100", as shown in Figure 2.12 for the first three models. This organization was identical for the other models.

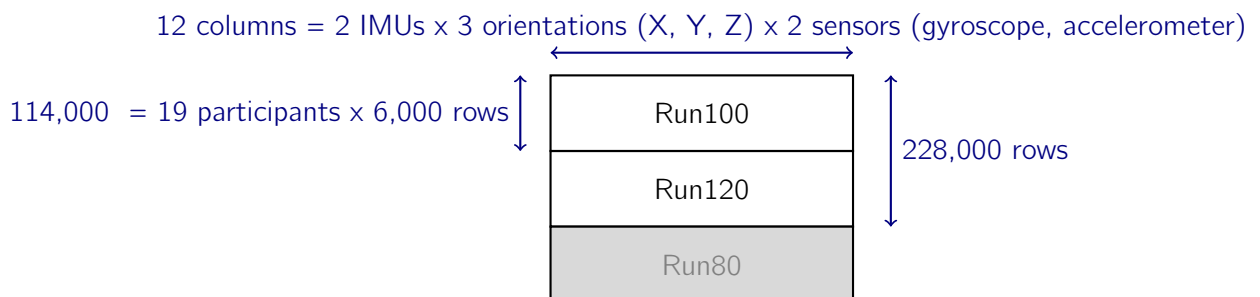


Figure 2.12: Illustration of running data assembly for the first three models.

For the assembly of walking data, the three tasks "Walk100", "Walk120", and "Walk80", as well as participant 16, were retained, resulting in input data of size 360,000 x 12 for models 4, 5, and 6.

The output data were assembled following the same logic. These have a single column and 228,000 rows for the first three models and 360,000 rows for the last three models.

### 2.2.3.2 Normalization

Feedforward fully connected neural networks require normalized input data for effective learning [25]. The normalization was performed on tables containing the different running or walking speeds stacked on top of each other (Figure 2.12), where the columns represent only the different sensors. This ensures that data from the different accelerometer and gyroscope sensors are on the same magnitude scale and contribute evenly to the model's learning.

Normalization was done by subtracting the mean of its column from each value in the table and then dividing the result by the standard deviation of that column. This operation ensures that each column or feature has a mean of 0 and a standard deviation of 1.

Scaling the input data directly impacts the final solution, as it determines the effective scaling of the weights in the input layer. Normalizing the data ensures that all features are treated uniformly, which promotes a more effective regularization process and improves the stability and convergence of the model training [52].

## 2.3 Models development and optimization

### 2.3.1 Designing regression models using `fitrnet`

For the design of Fully Connected Neural Networks (FCNN), the `fitrnet` function from MATLAB's Statistics and Machine Learning Toolbox™ [63] was used. This function allows the creation of FCNNs specifically for regression problems. It provides the flexibility to customize both the hyperparameters and the network architecture.

Hyperparameters refer to the parameters that define the structure and behavior of the neural network, which are not altered during learning [64]. They are divided into two categories: those related to training and those related to design. The former influence the learning rate and network performance during training, while the latter directly affect the model's learning capacity depending on the complexity of the task to be accomplished [64].

Among the design hyperparameters that can be optimized by the `fitrnet` function are the following:

**The number of hidden layers**, which determines the overall structure of the network and has a direct impact on performance [64]. In `fitrnet`, this parameter can vary from 1 to 3 fully connected layers.

**The number of neurons per layer** is also very important because an insufficient number of neurons limits the model's ability to capture data complexities, while too many neurons can slow down learning [64]. Generally, it is preferable to have too many rather than too few neurons per hidden layer to ensure the model is flexible enough to capture the non-linearities in the data [52]. In `fitrnet`, this number is optimized within a logarithmic interval of 1 to 300.

**The regularization hyperparameter ( $\lambda$ )** helps prevent overfitting by reducing the model's complexity, contrary to the other two hyperparameters discussed earlier, which increase it [64]. In `fitrnet`, this hyperparameter is optimized over logarithmic values ranging from  $[0.00001/n, 100000/n]$ , where  $n$  is the number of observations.

**Activation functions** play a crucial role in neural networks by introducing the necessary non-linearities to capture complex relationships in the data, as mentioned in Section 1.4. The `fitrnet` function determines the most suitable activation function for the training data from the following:

- The sigmoid function:  $f(x) = \frac{1}{1+e^{-x}}$ , with outputs ranging between 0 and 1. This is often used for binary classification (Figure 2.13).
- The hyperbolic tangent ( $\tanh$ ) function:  $f(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$ , which is centered on 0 and provides outputs between -1 and 1 (Figure 2.14).
- The following ReLu function, which produces only positive outputs (Figure 2.15):

$$f(x) = \begin{cases} 0 & \text{if } x \leq 0, \\ x & \text{if } x > 0. \end{cases}$$

- No activation function is also tested, reducing the neural network to a linear regression model for simpler modeling cases.

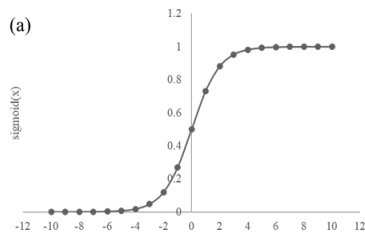


Figure 2.13: sigmoid

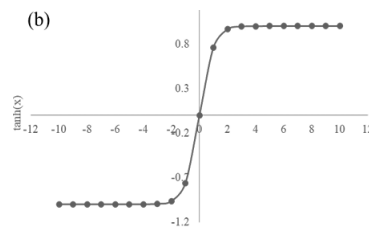


Figure 2.14: tanh

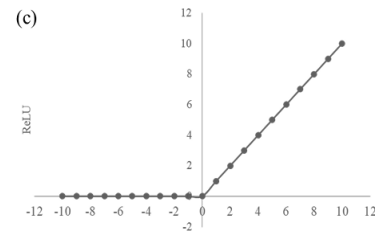


Figure 2.15: ReLu

Figure 2.16: Activation functions used by fitrnet [64].

Regarding the training-related hyperparameters, which regulate the optimization of the model's internal parameters, such as the weights of the connections between neurons [64], `fitrnet` does not allow for direct customization. Instead, this function uses an internal optimization algorithm based on a **quasi-Newton** method called **limited-memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS)**. This optimizer adapts the gradient descent algorithm to apply it to regression problems [65].

Simply put, **gradient descent** is an optimization method that aims to minimize a loss function, in this case, the mean squared error (MSE) between the model's predictions and the actual angle values, denoted by  $L$ . To minimize this function, the gradient of the loss function with respect to the model weights  $W$  is calculated ( $\frac{\partial L}{\partial W}$ ). This gradient is a vector indicating the direction of the greatest increase in the loss function [66], meaning it points to the left in Figure 2.17. Thus, moving in the opposite direction of this gradient, with a step given by the learning rate  $\alpha \in [0, 1]$ , allows one to reach the local minimum of the loss function [66]. The weights are updated at each iteration, or epoch, until the loss function reaches its minimum or the gradient is zero. This process of updating the weights via gradient descent is represented by the following equation 2.3, where  $W_{\text{new}}$  represents the new updated weights, and  $W_{\text{old}}$  represents the current weights before the update.

$$W_{\text{new}} := W_{\text{old}} - \alpha \frac{\partial L}{\partial W} \quad (2.3)$$



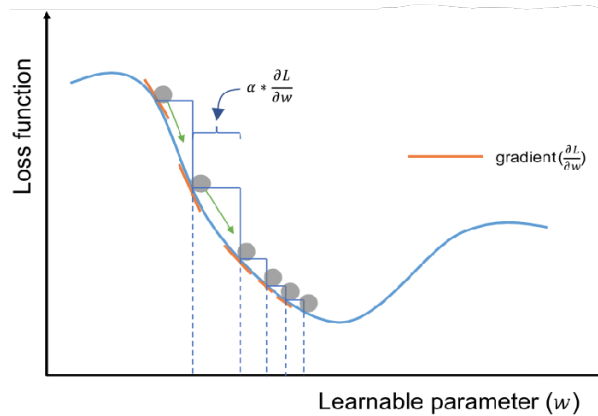


Figure 2.17: Schematic representation of the gradient descent algorithm from [66].

The **quasi-Newton LBFGS** algorithm used by `fitrnet` is a more advanced version of the classic gradient descent. In addition to using gradient information, it incorporates an approximation of the Hessian matrix, which consists of the second derivatives of the loss function and provides information about the curvature of this function at a given point. Knowing the curvature allows for adjusting not only the direction but also the magnitude of the weight updates [65]. However, directly calculating the Hessian matrix is very costly in terms of computation and memory. This is why the quasi-Newton algorithm used by LBFGS employs an approximation of this matrix. The limited-memory variant of this method, employed by `fitrnet`, reduces the amount of memory required by using only a small portion of past gradients to adjust the descent direction, making the algorithm more efficient for large-scale models [65].

### 2.3.2 Hyperparameters optimization

The `fitrnet` function in MATLAB offers three main methods for hyperparameters optimization: grid-search, random-search, and Bayesian optimization.

**Grid-search** involves performing an exhaustive search over a predefined set of hyperparameters. The user specifies a range of values for each hyperparameter in advance, and the algorithm systematically tests all possible combinations. Although this method is simple to implement, it requires strong preliminary knowledge of the optimal value ranges for each hyperparameter. Additionally, due to the high dimensionality of the hyperparameters, this method can quickly become inefficient and computationally expensive, making it unsuitable for complex cases like the one studied here [64].

**Random-search** is an improvement over grid-search, which randomly selects combinations of hyperparameters to test, based on predefined distributions of possible values. This method generally offers greater efficiency compared to grid-search, as it explores the hyperparameter space more broadly but it may still require significant time and computational resources [64].

Finally, **Bayesian optimization** is a more advanced method that does not require preliminary knowledge of the hyperparameter values or their distribution. It models the loss function (or objective function) as a posterior probability model, which is updated at each iteration based on previous results. This approach allows for faster and more efficient determination of the optimal hyperparameters. It is particularly valued for its effective balance between exploration (gathering new information) and exploitation (using current knowledge to make decisions). Given these advantages, Bayesian optimization will be the method used in this work [64].

### 2.3.2.1 Implementation of Bayesian optimization

To accelerate this Bayesian optimization process and make it less resource-intensive, the initial tables described in Figure 2.12, containing 30 seconds of acquisition per participant, were decomposed into 15 submatrices, each containing 2 seconds of signal acquisition per participant.

For example, for the first three models and for one sensor location (the heel), the initial input matrix with dimensions  $228,000 \times 6$  was split into 15 submatrices with dimensions  $15,200 \times 6$ , as illustrated in Figure 2.18. Each submatrix thus contains 2 seconds of acquisition for each participant, or 400 rows per participant, given that the data is sampled at 200 Hz. This segmentation was similarly applied to all other models and their corresponding outputs. The function "createSets.m", detailed in the appendix, was used to perform these decompositions.

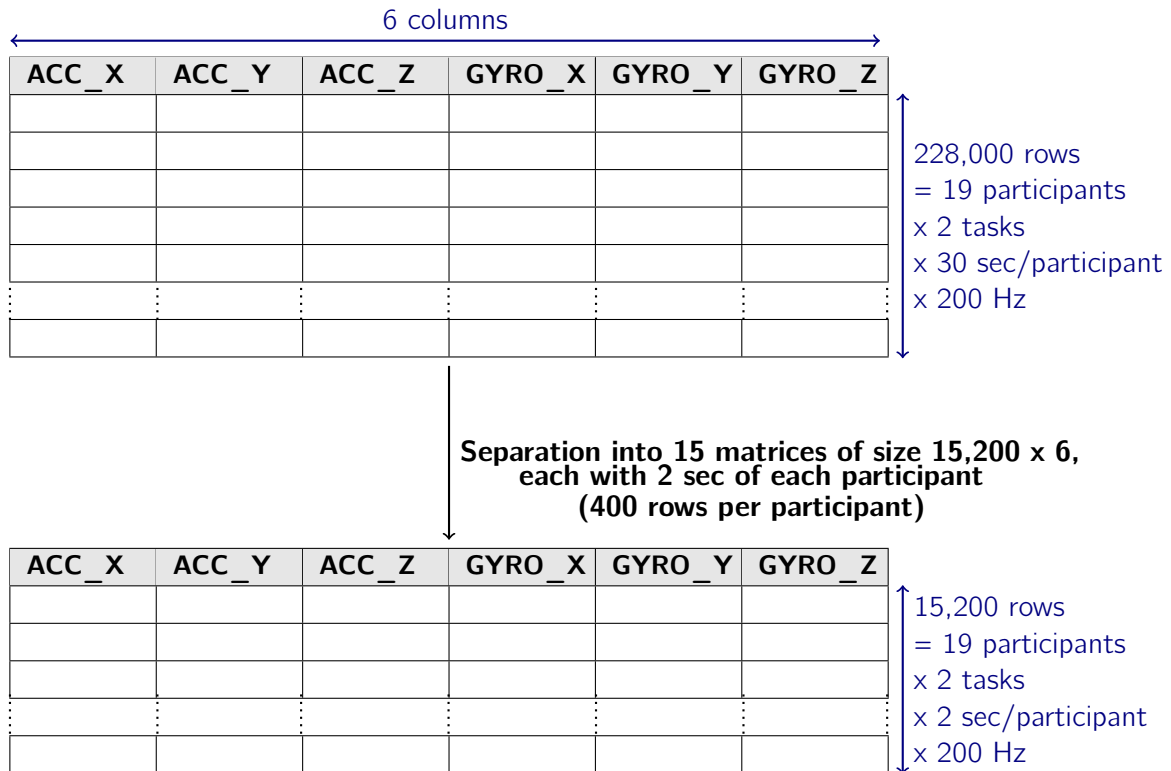


Figure 2.18: Illustration of the data structure for Bayesian optimization and for the first three models.

Bayesian optimization was independently performed on the 15 submatrices using MATLAB's `bayesopt` function, which is part of the Statistics and Machine Learning Toolbox™ [63].

In the `bayesopt` function, the chosen loss function or objective function to minimize, denoted here as  $f(x)$ , is the cross-validated loss, specifically the mean squared error (MSE) calculated over multiple validation folds (`kfoldloss`), as detailed further in the next section.

The approach used by `bayesopt` to model this objective function is based on a **Gaussian process (GP)**. Instead of predicting a single value for the objective function at each point in the search space, the Gaussian process provides a probabilistic distribution characterized by a mean and variance. The mean represents the best estimate of the objective function value, while the variance quantifies the uncertainty associated with this estimate. Using this information, the Bayesian optimization algorithm can make informed decisions, balancing the exploration of new regions of the search space (where uncertainty is high) with the exploitation of regions already identified as

promising (where the objective function is low for minimization) [67].

The algorithm also employs an acquisition function  $a(x)$ , which is also based on the Gaussian process model. This acquisition function is maximized to determine the next point  $x$  to evaluate, guiding the optimization process [67].

Initially, `bayesopt` evaluates  $y_i = f(x_i)$  for a number of points  $x_i$ , defined by the 'NumSeedPoints' parameter set to 4, to establish a foundation for the model. The algorithm then iterates, updating the Gaussian process model of  $f(x)$ , which provides a posterior distribution  $Q(f | x_i, y_i \text{ for } i = 1, \dots, t)$ . At each iteration, the next point  $x$  to evaluate is chosen by maximizing the acquisition function  $a(x)$  [67].

The `bayesopt` function offers several options for choosing the acquisition function  $a(x)$ . The 'expected-improvement-plus' option was selected for its adaptive balance between exploration and exploitation, enhancing the optimization process. This acquisition function evaluates the expected improvement in the objective function, focusing on points that could potentially lower it. Additionally, the 'plus' option detects when the algorithm is overly exploiting a local minimum region of the objective function and increases the variance, allowing the algorithm to explore new areas of the search space, ensuring a good balance between exploration and exploitation [67].

The optimization terminates when one of the stopping criteria is met, in this case, the maximum number of iterations, which is set by default to 30.

### 2.3.2.2 Cross-validation setup

To ensure an accurate and balanced estimation of the cross-validation loss used in Bayesian optimization, a `cvpartition` object was created to implement a stratified k-fold cross-validation (with  $k=5$ ).

As illustrated in Figure 2.19, 5-fold cross-validation involves splitting the dataset into 5 subsets, known as "folds." During each iteration, 4 of these folds are used to train the model, while the remaining fold serves as the test or validation set to evaluate the model's performance. This process is repeated 5 times so that each fold is used once as the test set. This ensures that every data subset is evaluated, while avoiding overlap between the training and test sets, as the training data remains independent of the test data at each iteration.

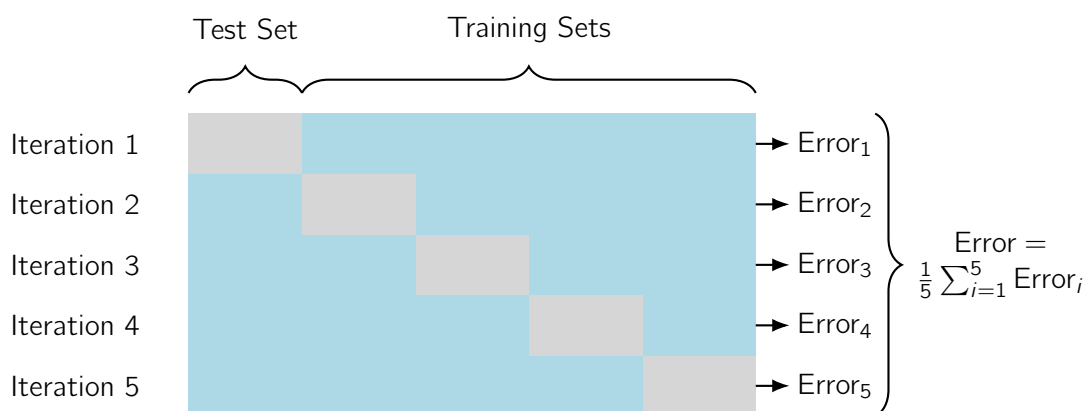


Figure 2.19: 5-fold cross-validation.

The `group` option, added to the partition object, ensured that each fold in the cross-validation

process contained a fair proportion of data from each participant and each task. To achieve this, a column vector named "group.mat" was created, with the number of rows corresponding to the number of rows in the submatrices: 15,200 rows for the running models and 24,000 rows for the walking models. Each row of this vector was then assigned to a specific group, representing a participant and a particular task. This resulted in a vector containing values ranging from 1 to 38 for the running models, and from 1 to 60 for the walking models, with each group represented by 400 consecutive rows.

For the running models, composed of 2 tasks, each of the 15 submatrices contains 15,200 rows. Therefore, each fold in the cross-validation process contains 3,040 rows, distributed to include 80 rows per participant for the "Run100" task and 80 rows per participant for the "Run120" task, as illustrated in Figure 2.20.

For the walking models, composed of 3 tasks, each of the 15 submatrices contains 24,000 rows. Thus, each fold contains 4,800 rows, distributed to include 80 rows per participant for each of the "Walk100", "Walk120", and "Walk80" tasks, as illustrated in Figure 2.21.

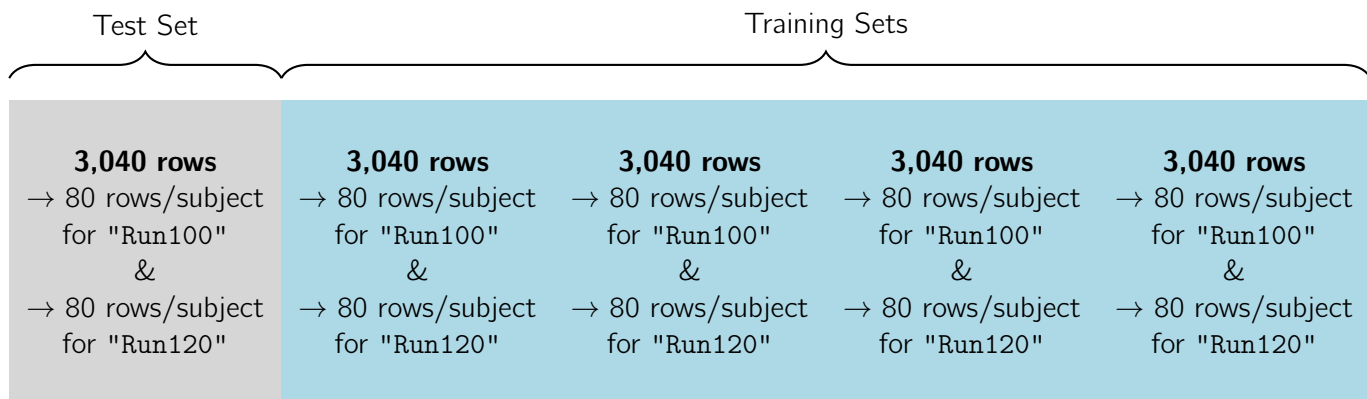


Figure 2.20: Row split for the first iteration of running model (19 participants) cross-validation.

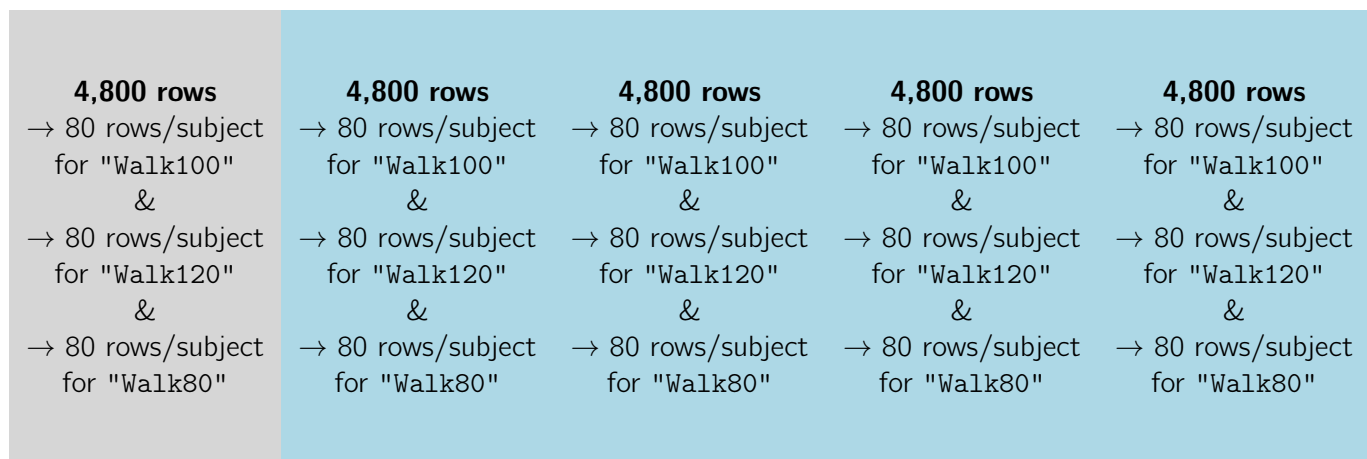


Figure 2.21: Row split for the first iteration of walking model (20 participants) cross-validation.

### 2.3.3 Averaging hyperparameters for final model creation

Given that Bayesian optimization was independently conducted on the 15 submatrices, the final hyperparameters for each regression model, built on the entire input matrix (without division into

submatrices), were determined by averaging the 15 sets of optimized hyperparameters from each submatrix. Numerical hyperparameter values were averaged and then rounded to the nearest integer using the `round` function. For categorical hyperparameters, the most frequently occurring values across the 15 optimized sets were selected for the final model creation.

Each final model was then created using the `fitrnet` function with the complete dataset, without prior division into submatrices, corresponding to the structure shown in Figure 2.12.

Given that the complete dataset is substantially larger, validation for each final model was performed using an 80/20 split, where 80% of the observations were used for training and 20% for testing, across all participants and tasks. Thus, with each participant contributing 6,000 rows per task, the first 4,800 rows were used for model training, while the remaining 1,200 rows were used to verify whether the model could accurately predict these data for each participant, as shown in Figure 2.22.

For the running models, which included 19 participants performing the two tasks "Run100" and "Run120", a total of 38 predictions were made and are presented in the following results chapter. Similarly, for the walking models, which involved 20 participants performing the three tasks "Walk100", "Walk120", and "Walk80", 60 predictions were made and are also presented in the subsequent chapter.

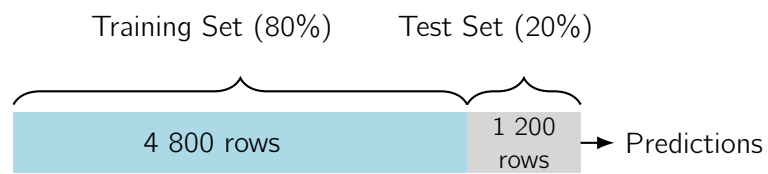


Figure 2.22: 80-20 distribution of running data for a participant and a specific task.

Finally, the remaining 20% of the test data for each participant and each task was used with MATLAB's `predict` function to generate specific predictions for each participant and task. These predictions then served as the basis for calculating various evaluation metrics to quantify the errors associated with each regression model.

The functions described and used throughout this section 2.3, including Bayesian optimization, cross-validation, and the creation of each model based on the averaged set of 15 optimized hyperparameter sets, are detailed in the appendix, specifically in the "`optimizeNeuralNetworkWithSets.m`" and "`main.m`" functions.

## 2.4 Evaluation metrics

Several metrics were employed to evaluate the performance of the different models and to compare them with previous work in the literature.

First, the Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) were calculated to provide a measure of the overall accuracy of the models [68]. These metrics are commonly used in the literature, allowing for direct comparison of the results obtained here with those from previous studies.

Next, normalizing the error based on the data amplitude through the calculation of nRMSE allowed for comparing errors across different running and walking tasks.

Violin plots were generated to visually represent the distribution of errors.

Additionally, beyond simply evaluating average errors, the Pearson correlation coefficient was calculated to verify that the model predictions follow the general trend of the actual data.

Finally, the predicted and actual angles were plotted against the percentage of the running or walking cycle, following common practices in human movement analysis, enabling a biomechanical analysis.

### 2.4.1 Root Mean Square Error (RMSE) and Mean Absolute Error (MAE)

For each prediction made for each participant, the Root Mean Square Error (RMSE) is calculated as the square root of the mean of the squared differences between the model's predicted values  $\hat{y}_i$  and the actual joint angle values  $y_i$ , according to Equation (2.4) [69]. Here,  $n$  represents the total number of observations in the test set.

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (2.4)$$

The RMSE takes into account large prediction errors, as these are amplified due to the squaring of the differences [51].

Similarly, the Mean Absolute Error (MAE) is calculated for each participant by taking the average of the absolute differences between the actual values  $y_i$  and the predicted values  $\hat{y}_i$ , as shown in Equation (2.5) below [51].

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (2.5)$$

The MAE is less sensitive to large prediction errors because it takes the absolute value of the differences rather than squaring them. These two metrics are widely used standard measures for evaluating the performance of neural network models [69]. Moreover, they have the advantage of being expressed in the same units as the model's output data, in this case degrees, which simplifies their interpretation.

### 2.4.2 Normalized Root Mean Square Error (nRMSE)

The nRMSE is a normalized version of the RMSE, calculated by normalizing the RMSE by the amplitude of the actual values in the entire test sample for a given participant. This involves taking the difference between the maximum and minimum values of this sample and then multiplying by 100 to express it as a percentage, as shown in Equation (2.6) below.

$$\text{nRMSE} = \frac{\text{RMSE}}{\max(y) - \min(y)} \times 100 \quad (2.6)$$

Normalizing by the data amplitude for each participant and each task allows for comparison of results across different participants, even if they did not run or walk at the same speed. It also enables comparison between different types of tasks, such as walking and running [68].

### 2.4.3 Violin plots

Violin plots are an effective visual method for representing the results of a machine learning model. Named for their resemblance to the shape of a violin, these plots combine features of box plots while providing a more detailed depiction of data distribution along the vertical axis [70].

Box plots, which are statistical visualization tools, summarize key characteristics of a dataset, including [70]:

**The minimum**, which is the smallest non-outlier value.

**The first quartile (Q1 or 25th percentile)**, representing the value below which 25% of the observations fall.

**The median (Q2 or 50th percentile)**, which divides the dataset into two equal parts.

**The third quartile (Q3 or 75th percentile)**, representing the value below which 75% of the observations fall.

**The maximum**, which is the largest non-outlier value.

Additionally, box plots allow for the visualization of outliers, the interquartile range (IQR) between the third and first quartiles, and skewness in the data [70]. Figure 2.23 illustrates these various features and shows how violin plots enrich this information by adding the data distribution in the form of smoothed histograms along the vertical axis [70].

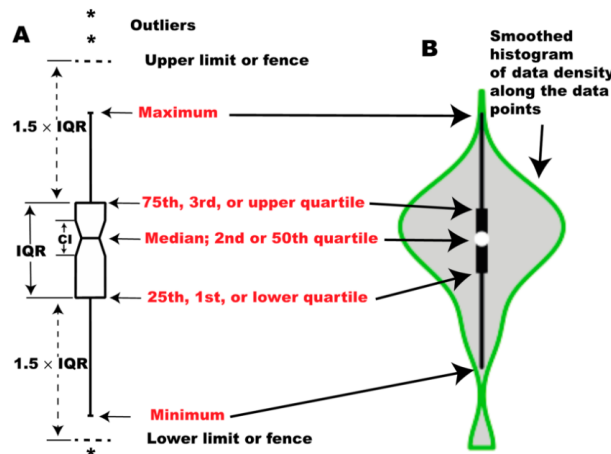


Figure 2.23: Representation of the characteristics of whisker boxes (A) and violin diagrams (B) [70].

Thus, violin plots allow for comparing the performance of different models by visualizing not only the statistical characteristics of the errors but also their distribution, offering a more comprehensive and nuanced view of each model's performance.

To generate these plots, the "violin.mat" function developed by Hoffman H. [71], available online, was used. This function generates violin plots using kernel density estimation, performed by MATLAB's `ksdensity` function.

The kernel density estimation of errors represents the probability density function (pdf) of these errors non-parametrically, meaning no assumptions are made about the distribution of this data [72]. This estimation smooths each error value  $x_i$  for  $i = 1, 2, \dots, n$  (where  $n$  is the number of errors, one per participant) with a kernel smoothing function denoted as  $\mathbf{K}(-)$ , which are all summed to obtain the estimated density  $\hat{f}_h(x)$ , as shown in the following Equation 2.7 [72]. Unlike a traditional histogram that bins values into discrete intervals, this method generates a smooth, continuous probability curve [72].

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^n \mathbf{K} \left( \frac{x - x_i}{h} \right) \quad (2.7)$$

Where  $h$  is the bandwidth, which controls the smoothness of the resulting density curve [72].

### 2.4.4 Pearson correlation coefficient

The Pearson correlation coefficient is a statistical measure that evaluates the strength and direction of the linear relationship between two continuous variables, which are typically assumed to follow a bivariate normal distribution [73]. In the context of this study, it is used to analyze the relationship between the actual joint angle values and those predicted by the machine learning models.

This coefficient, denoted as  $r$ , ranges from -1 to +1. A value close to +1 indicates a very strong positive correlation, meaning that the model's predictions effectively capture the variations in joint angles over time, with both variables increasing proportionally. A value of 0 suggests no linear relationship, while a value close to -1 indicates a perfect negative correlation, where one variable increases as the other decreases proportionally. In the latter case, this could suggest that the model fails to capture the dynamics of joint movements, resulting in predictions that are inversely related to reality [73].

To calculate this coefficient, the JASP software (JASP Team, 2024. JASP [Version 0.19.0], [Computer software]) [74] was used. The actual and predicted values for each participant were organized into a table for the two IMU locations. This table, illustrated in Figure 2.24, was then converted into a CSV file to be imported into JASP. Consequently, a Pearson correlation coefficient  $r$  was calculated for each IMU location (the last two columns of Figure 2.24) by comparing them to the actual angle values (second column).

Participant	True angles	Predicted angles (Heel)	Predicted angles (Mid foot)
1			
1			
1			
⋮	⋮	⋮	⋮
38			
38			
38			

45,600 rows  
 = 19 participants  
 x 2 tasks  
 x 1,200 predictions  
 rows

Figure 2.24: Running data format for Pearson's coefficient calculation.

Figure 2.25 below illustrates a commonly used approach for interpreting this correlation coefficient.

Absolute Magnitude of the Observed Correlation Coefficient	Interpretation
0.00–0.10	Negligible correlation
0.10–0.39	Weak correlation
0.40–0.69	Moderate correlation
0.70–0.89	Strong correlation
0.90–1.00	Very strong correlation

Figure 2.25: Example of a conventional approach for interpreting Pearson's coefficient [73].

Additionally, in JASP, the calculated Pearson coefficient is accompanied by a p-value, which helps determine whether the observed correlation is likely to be due to chance. Specifically, this p-value represents the probability of obtaining the observed correlation  $r$ , given that the null hypothesis ( $H_0$ ), which states that there is no relationship between the variables ( $r = 0$ ), is true. In other words, the p-value assesses the likelihood that the coefficient  $r$  is high purely by chance [75].



Therefore, if the p-value is low ( $p < 0.05$ ), it is highly unlikely that the observed correlation is due to chance, allowing us to reject the null hypothesis and accept the existence of a significant relationship between the variables. Conversely, if the p-value is high ( $p > 0.05$ ), even if the coefficient  $r$  is high, it will be considered non-significant and potentially due to chance [75].

### 2.4.5 Segmentation into gait cycles

To interpret the model predictions in terms of human biomechanics, it is essential to present the kinematic data in the form of gait cycles.

Figure 2.26 shows a walking and running cycle, along with their associated phases. Both cycles begin when one foot, in this case, the right foot, makes contact with the ground and end when the same foot touches the ground again. The main difference between walking and running cycles lies in the presence of a double support phase during walking, where both feet are in contact with the ground. This phase is absent in the running cycle, which consists only of the stance phase (where the right foot is in contact with the ground) and the swing phase (where the foot is not touching the ground). Thus, at no point during running are both feet simultaneously in contact with the ground.

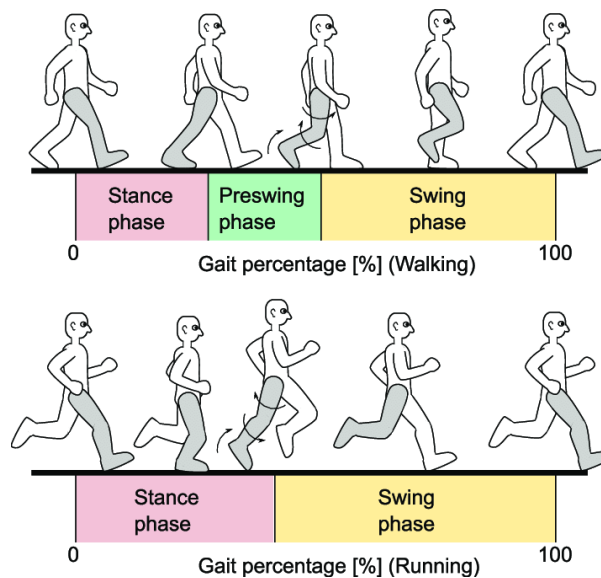


Figure 2.26: Illustrations of a walking (top) and running (bottom) cycle and the corresponding phases [76].

In this work, the beginning and end of gait cycles, corresponding to the moments when the right foot touches the ground, were identified using the event detection tool in the Visual3D application. It was assumed that all participants exhibited heel strikes during running (heel strikes are assumed to occur during walking by default), to mark each cycle's start as the moment when the heel touches the ground. The end of a cycle was then defined as the start of the next cycle. These moments were imported into MATLAB and stored in a matrix to facilitate the identification of different cycles present in the actual and predicted angle signals.

Once the cycles were identified for each participant, the mean and standard deviation were calculated for all observed cycles. These means and standard deviations are then displayed for the two IMU sensor locations for each participant. The different tasks (e.g., "Run100" and "Run120" for running models) are presented separately.

To generate these figures, MATLAB's `fill` and `flip1r` functions were used to plot the mean cycle and the associated standard deviations around this curve. The codes responsible for signal segmentation into cycles and those used to generate the figures are available in the appendix ("`visualizedModelPerformances.m`").

Interpreting these graphs provides a qualitative evaluation that complements the quantitative metrics discussed earlier. These graphs help detect anomalies in predictions, assess temporal accuracy, and verify that the general movement trends are correctly captured by the model.

It is important to note that the amplitude of movements does not depend on the subject's anthropometry. Therefore, it is unnecessary to normalize the angles according to the subject's weight and height, unlike what is often required for joint moments [23].

# Chapter 3

## Results and discussions

This third chapter provides a detailed analysis of the results obtained during this study. First, the results of Bayesian optimization aimed at determining the optimal hyperparameters for each model are presented. Next, a comparative analysis of the two possible IMU sensor placements is conducted using violin plots. The mean absolute errors are compared with existing scientific studies, particularly those mentioned in Section 1.3.3 of the first Chapter 1, which also used a single IMU sensor for predicting lower limb kinematics. In addition to the quantitative error metrics, Pearson correlation coefficients are calculated to provide a qualitative evaluation of the agreement between the predictions and the actual values. Finally, the model predictions in the form of gait cycles are presented for each participant and each task. This presentation helps identify participants who contributed to the increase in model errors. These cycles are also compared with reference cycles from the literature and analyzed biomechanically.

### 3.1 Optimized Neural Network hyperparameters

To illustrate the process and performance of Bayesian optimization, an example analysis on one of the 15 submatrices of data created for Model 4, which focuses on predicting the ankle angle during walking, is shown.

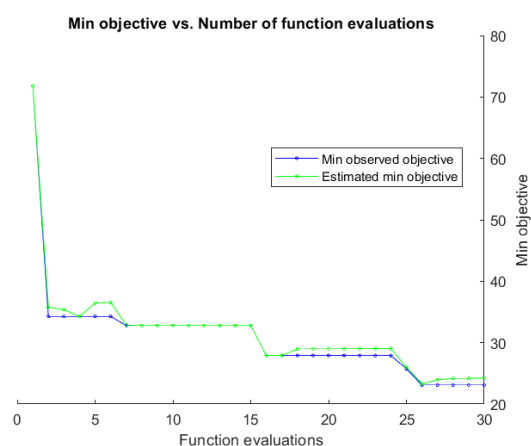


Figure 3.1: Illustration of the Bayesian optimization process for hyperparameter selection.

The above Figure 3.1 shows the evolution of the minimum value of the objective function as a function of the number of function evaluations. This curve represents the convergence of the

Bayesian optimization algorithm, highlighting both the observed minimum value of the objective function (in blue) and the estimated minimum of the objective function (in green).

The curve shows a gradual decrease in the value of the objective function, indicating that the algorithm is refining the hyperparameters over the iterations to minimize the loss.

Finally, the difference between the blue and green curves indicates that the estimation of the objective function by the Bayesian model is not always exact, as this model is based on probabilistic predictions. However, it generally converges towards the observed minimum value.

In general, the convergence curves observed for the 15 submatrices and across all models showed similar trends to those illustrated in Figure 3.1. However, the final values of the objective function varied from one model to another. Although the exact values of the objective function were not retained, they directly reflect the final errors obtained through the various model evaluation metrics, as these values are proportional to each model's performance.

The final optimal hyperparameters for each model, obtained by averaging the hyperparameters from the 15 submatrices, are summarized in the following Table 3.1.

Model	IMU locations	Number of layers	First layer size	Second layer size	Third layer size	Activation function	Regularization
1	Heel	3	216	104	53	ReLu	5.5448e-04
	Mid foot	3	149	166	112.27	ReLu	0.0022
2	Heel	2	268	167	-	Sigmoid	0.0153
	Mid foot	2	192	238	-	Tanh	0.0025
3	Heel	3	175	225	134	Tanh	0.0042
	Mid foot	2	230	163	-	Tanh	0.0047
4	Heel	3	217	199	160	Tanh	0.0043951
	Mid foot	2	253	273	-	Tanh	0.0058
5	Heel	3	284	177	91	Tanh	0.00078
	Mid foot	2	200	291	-	Tanh	0.0016
6	Heel	2	267	239	-	Relu	0.0118
	Mid foot	2	216	277	-	ReLu	4.8881e-04

Table 3.1: Optimal hyperparameters for each model and IMU localization.

## 3.2 Comparison of different IMU locations

### 3.2.1 Model 1: Ankle angle predictions during running

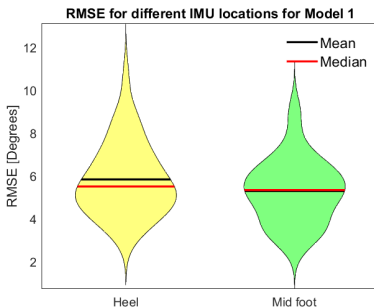


Figure 3.2: RMSE

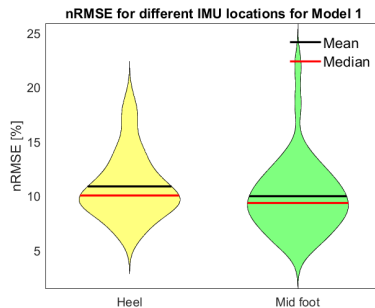


Figure 3.3: nRMSE

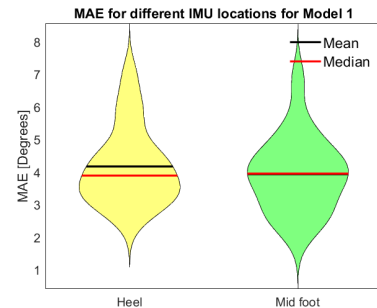


Figure 3.4: MAE

Figure 3.5: Analysis of error distributions: Root Mean Squared Error (RMSE), normalized Root Mean Squared Error (nRMSE) and Mean Absolute Error (MAE).

Figures 3.2, 3.3, and 3.4 present the distribution of errors for each IMU location in the form of violin plots for the first model, which aims to predict ankle angles during running.

Analysis of these figures reveals that both IMU placements produce relatively similar errors, suggesting that both positions are reliable for predicting ankle angles during running.

Delving deeper into the analysis, Figure 3.2, shows that the RMSE distribution for the IMU placed in the middle of the foot is more concentrated around the mean and median, indicating a lower presence of extreme errors. Moreover, the median and mean RMSE for this location are slightly lower than those for the heel placement (less than  $6^\circ$ ), suggesting better accuracy.

The difference between the average RMSEs (Figure 3.2) and the average MAEs (Figure 3.4), is not significantly large (about  $2^\circ$ ), indicating that there are not many outlier errors.

Thus, it can be concluded that the IMU placed in the middle of the foot allows for more accurate predictions of ankle angles during running. This location will therefore be preferred for comparing errors with previous studies in the literature, although both positions show relatively similar performances in terms of errors.

### 3.2.1.1 Comparison with previous studies

For the midfoot location, the smallest RMSE value obtained is  **$2.8807^\circ$**  (participant 10 for the "Run120" task), while the highest reaches  **$8.8487^\circ$**  (participant 1 for the "Run100" task).

The only comparable study that analyzed the ankle joint in the sagittal plane during running with a single IMU is that of Long et al. (2024) [32]. This study, using a random forest regression model, reported an average RMSE of  $3.22^\circ \pm 0.69$  during a jogging task over a few meters. This performance was calculated on a small sample of only four participants, each performing several repetitions of the tasks. Thus, the approach adopted by Long et al. (2024) essentially aims to develop an extremely accurate but individual-specific model, rather than a model generalizable to a larger population.

In comparison, this work, by including a larger number of participants and analyzing running tasks over longer periods, strives to develop a more generalizable model, which inevitably leads to greater error variability. However, achieving a minimum error of  $2.8807^\circ$  (lower than that of Long et al. at  $3.22^\circ$ ) suggests that although there is still room for improvement in reducing errors for some participants, this model demonstrates solid and promising performance for predicting ankle angles during running with an IMU placed in the middle of the foot.

### 3.2.2 Model 2: Knee angle predictions during running

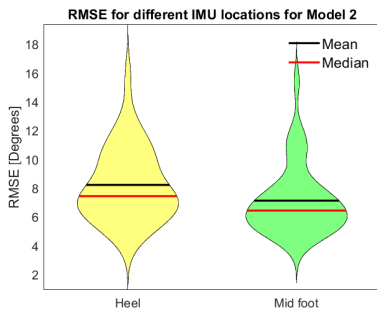


Figure 3.6: RMSE

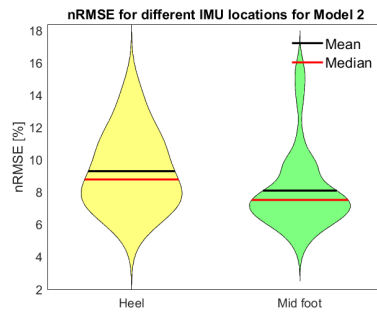


Figure 3.7: nRMSE

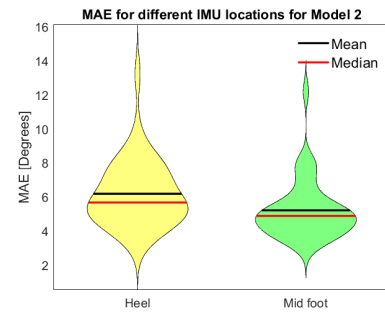


Figure 3.8: MAE

Figure 3.9: Analysis of error distributions: Root Mean Squared Error (RMSE), normalized Root Mean Squared Error (nRMSE) and Mean Absolute Error (MAE).

Figures 3.6, 3.7, and 3.8 present the distribution of errors for each IMU location in the second model, which aims to predict knee angle during running.

The observations made for the first model are even more pronounced in this second model. The distribution of RMSE (Figure 3.6) as well as MAE (Figure 3.8) show a much more pronounced concentration around the mean and median for the IMU placed on the midfoot, compared to the placement on the heel. Additionally, the difference in central values (mean and median) between the two positions is more significant this time, with the midfoot proving to be the better location, showing a mean RMSE below  $8^\circ$  and a mean MAE below  $6^\circ$ .

The difference between the mean RMSE (Figure 3.6) and mean MAE (Figure 3.8), is not significantly large (around  $2^\circ$ ), suggesting that there are not many outlier errors.

Thus, as with the first model, it can be concluded that the IMU placed on the midfoot allows for more accurate knee angle predictions during running. This location will therefore be preferred for comparing errors with previous studies in the literature, although both positions show relatively similar performance in terms of errors.

#### 3.2.2.1 Comparison with previous studies

For the midfoot location, the smallest RMSE obtained is  $4.21^\circ$  (participant 10 for the "Run120" task), while the largest is  $15.3895^\circ$  (participant 1 for the "Run100" task).

These results are generally less accurate than those reported in the literature, both in terms of the minimum value obtained and the standard deviation.

For example, the study by Long et al. (2024) [32], which aimed to predict kinematics using an IMU mounted on the ankle for 4 athletes and a random forest regression model, obtained an RMSE of  $2.66^\circ \pm 0.27$  for a jogging task over a few meters.

The study by Chow et al. (2021) [33] reported average RMSE values of  $3.2^\circ \pm 1$  for intra-participant predictions on 10 participants. This study trained its CNN model on a portion of a participant's data and then predicted other data from the same participant, based solely on gyroscope data from a single IMU placed on the tibia while using a treadmill.

Finally, the study by Chow et al. (2022) [34], with a protocol similar to the previous study [33], obtained average RMSE values of  $4.1^\circ \pm 1.2$ .

These findings clearly demonstrate, as mentioned in Section 1.4, that convolutional neural networks (CNNs) offer better performance compared to a feedforward neural network (FNN).

### 3.2.3 Model 3: Hip angle predictions during running

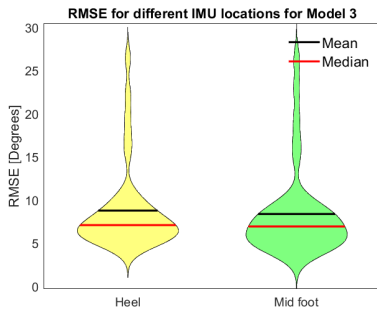


Figure 3.10: RMSE

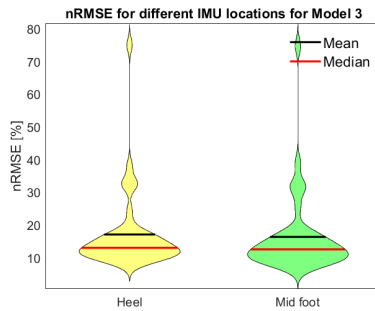


Figure 3.11: nRMSE

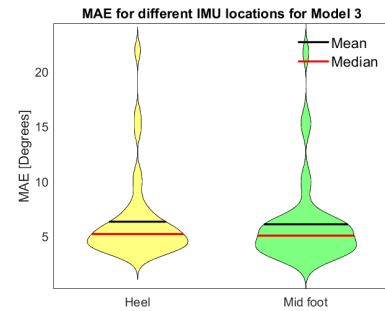


Figure 3.12: MAE

Figure 3.13: Analysis of error distributions: Root Mean Squared Error (RMSE), normalized Root Mean Squared Error (nRMSE) and Mean Absolute Error (MAE).

For the third model, which is focused on predicting hip angles during running (Figure 3.13), the violin plots for both IMU locations show relatively similar distributions, making it challenging to determine a superior IMU placement for hip angle predictions. Therefore, both locations will be considered for comparison with previous studies.

The violin shapes for this third model indicate that the error distributions are relatively concentrated around central values. However, there is a noticeable long upper tail, especially for the RMSE (Figure 3.10), suggesting that some predictions have extreme errors.

#### 3.2.3.1 Comparison with previous studies

The smallest RMSE value obtained is **3.8292°** (participant 5 for the "Run120" task with the IMU placed on the midfoot), while the highest is **26.4602°** (participant 1 for the "Run100" task with the IMU on the heel).

In comparison, the study by Chow et al. (2021) [33], which used a convolutional neural network (CNN), reported average RMSE values of  $5.2^\circ \pm 1.7$  for intra-participant predictions (10 participants), based solely on gyroscope data.

Additionally, the study by Chow et al. (2022) [34], with a protocol similar to the previous one [33], achieved average RMSE values of  $3.6^\circ \pm 1.2$ .

Thus, the minimum value obtained here ( $3.8292^\circ$ ) for hip angles during running is relatively comparable to those achieved in these two studies. However, once again, the main area for improvement in this work is the reduction of errors for certain participants, as this model still exhibits significant variability in error values across different participants.

### 3.2.4 Model 4: Ankle angle predictions during walking

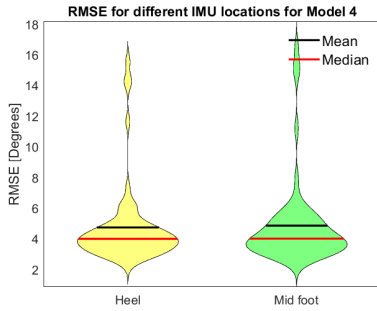


Figure 3.14: RMSE

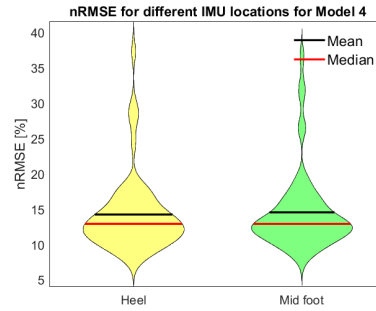


Figure 3.15: nRMSE

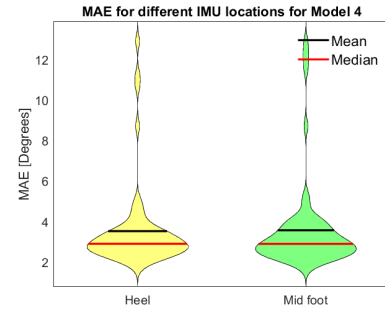


Figure 3.16: MAE

Figure 3.17: Analysis of error distributions: Root Mean Squared Error (RMSE), normalized Root Mean Squared Error (nRMSE) and Mean Absolute Error (MAE).

For the fourth model, dedicated to predicting ankle angles during walking (Figure 3.17), the violin plots for both IMU placements exhibit remarkable similarities, both in terms of the positioning of the means and medians and in the overall shape of the violins. Therefore, it is difficult to determine an optimal IMU placement for predicting ankle angles. As a result, both placements will be retained for comparison with previous studies.

In comparison with the first model dedicated to running (Figure 3.5), the normalized errors relative to the range of motion for this fourth walking model (Figure 3.15) are slightly higher ( $\pm 15\%$ ) compared to the equivalent model for running ( $\pm 11\%$ ). This suggests that the models developed here more easily capture the dynamics of running compared to walking for ankle angles.

#### 3.2.4.1 Comparison with previous studies

The results obtained for this model show significant variation in errors among participants. The smallest RMSE observed is **2.616°** (participant 11, "Walk100" task with the IMU placed at the midfoot), while the largest reaches **16.4349°** (participant 9, "Run120" task with the IMU also at the midfoot).

In comparison, the study by Long et al. (2024) [32] reports an average RMSE of  $2.19^\circ \pm 0.57$  for a short walking task, with an IMU placed on the ankle. The results of their study are comparable to those obtained here only in terms of the minimum value. However, it is crucial to note that Long et al.'s study used a much smaller sample (four athletes), which limits the generalizability of their results and could explain the lower variability compared to the errors observed in this work.

The study by Alcaraz et al. (2021) [31], using deep neural networks with an IMU on the foot, achieved RMSE values ranging from  $2.57^\circ$  to  $3.54^\circ$ . The performances of this fourth FNN model developed here fall within a similar range.

As for nRMSE, the results obtained here range from **7.9851%** (participant 19, "Walk100" task with the IMU at the midfoot) to **37.3315%** (participant 9, "Walk80" task with the IMU at the heel).

In comparison, the study by Lim et al. (2020) [28], with nRMSE values ranging from 4.36% to 5.93% for slow walking speeds and an IMU placed at the sacrum, shows overall lower errors than



those obtained here. Therefore, testing this IMU placement on the sacrum, likely less prone to artifacts related to movements of distal segments, could be an avenue for improvement in this work.

The study by Mundt et al. (2020) [23], based on an FNN and an IMU, achieved average nRMSE values of 7.39%, very close to the best performances observed here. However, it is important to note that this study used simulated data from an optical motion capture database rather than real IMU data.

Finally, the study by Sung et al. (2022), which used an LSTM model with 30 participants, reported an nRMSE of 8.52% and an intra-subject RMSE of 3.96°. Comparatively, the results obtained here with an FNN model show equivalent performances.

In conclusion, the results of this model show that, although significant errors may occur for some participants, this fourth model is capable, for some of them, of providing results comparable to or even better than those reported in the literature for predicting ankle angles during walking.

### 3.2.5 Model 5: Knee angle predictions during walking

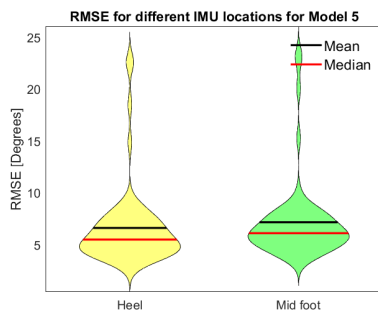


Figure 3.18: RMSE

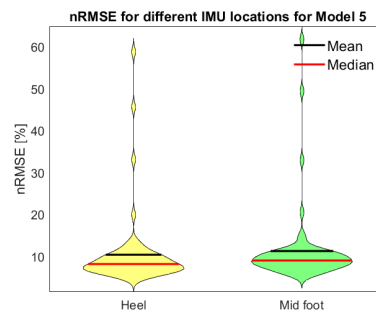


Figure 3.19: nRMSE

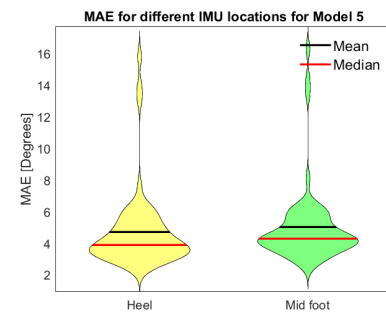


Figure 3.20: MAE

Figure 3.21: Analysis of error distributions: Root Mean Squared Error (RMSE), normalized Root Mean Squared Error (nRMSE) and Mean Absolute Error (MAE).

For the fifth model, dedicated to predicting knee angles during walking (Figure 3.21), the violin plots for both sensor placements show remarkable similarities. However, the minimum (lower end of the violin) and maximum errors (upper end of the violin) are slightly lower for the sensor placed on the heel, which justifies the choice of this location for comparison with previous studies.

When compared to the second model dedicated to running (Figure 3.9), the normalized errors relative to the amplitude of movements in this fifth walking model (Figure 3.19) are higher ( $\pm 12\%$ ) than those in the equivalent running model ( $\pm 8\%$ ) (Figure 3.7). This suggests, as with ankle angles, that the models developed here generally capture the dynamics of running more easily than those of walking for knee angles.

#### 3.2.5.1 Comparison with previous studies

For knee angle prediction using the IMU placed on the heel, the results show RMSE values ranging from **3,142°** (participant 1 for the "Walk100" task) to **22,7862°** (participant 1 for the "Walk120" task).

In comparison, the study by Long et al. (2024) [32] reports an RMSE of  $3,30^\circ \pm 0,69^\circ$  for a walking task with an IMU placed on the ankle. The results from Model 5 in this work show similar errors under certain conditions, but also a greater variation in errors.

Using an IMU on the ankle and personalized models for 10 participants, Yeung et al. (2023) [50] obtained average RMSE values of  $2,45^\circ \pm 0,65^\circ$  for personalized models, compared to  $6,77^\circ \pm 3,38^\circ$  for generalized models. This highlights the challenges of generalizing predictive models to a broader population, as observed here.

Using an IMU on the foot and deep neural networks, Alcaraz et al. (2021) [31] obtained RMSEs ranging from  $2.12^\circ$  to  $2.95^\circ$ , demonstrating the superiority of CNNs.

Regarding nRMSE values for the same location, they range from **5,2071%** (participant 10 for the "Walk80" task) to **58,8381%** (participant 9 for the "Walk100" task).

In comparison, the study by Lim et al. (2020) [28] reported nRMSE values between  $2.99\% \pm 0.80$  and  $6.04\% \pm 1.55$  for three different slow walking speeds (with errors increasing with walking speed) for 7 participants and an IMU placed on the sacrum, suggesting that this position should be considered.

The study by Mundt et al. (2020) [23] also obtained, based on an FNN and an IMU, average nRMSE values of 9.46%. However, it is important to note that this study was based on simulated data from an optical motion capture database rather than real IMU data.

Using an LSTM model with an IMU placed on the lateral shank, the study by Sung et al. (2022) [30] obtained an nRMSE of 9.3% and an RMSE of  $3.34^\circ$  for intra-subject predictions. The results from our study show that the FNN used is capable of achieving similar performance.

In conclusion, the fifth model, dedicated to predicting knee angles during walking, shows performance comparable to some previous studies, although significant variations in errors are observed depending on the participants. The results highlight the complexity of generalizing predictive models to a broader population.

### 3.2.6 Model 6: Hip angle predictions during walking

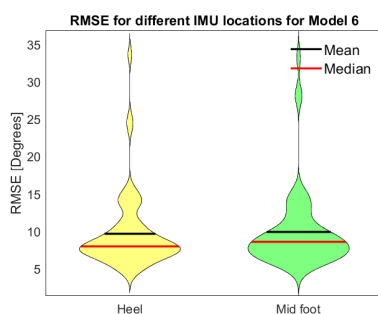


Figure 3.22: RMSE

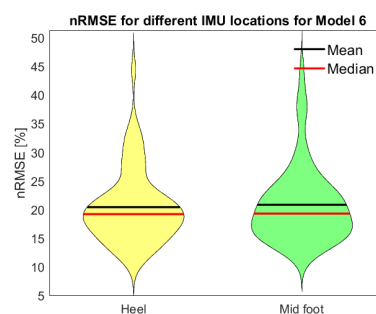


Figure 3.23: nRMSE

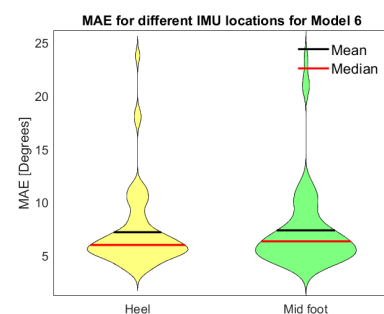


Figure 3.24: MAE

Figure 3.25: Analysis of error distributions: Root Mean Squared Error (RMSE), normalized Root Mean Squared Error (nRMSE) and Mean Absolute Error (MAE).

For the sixth model, dedicated to predicting hip angles during walking (Figure 3.25, the violin plots for both IMU placements once again show significant similarities. However, the average RMSE (Figure 3.22) and MAE (Figure 3.24) errors are slightly lower for the heel placement, justifying

the choice of this position for comparison with previous studies.

The average relative errors observed in this walking model ( around  $\pm 20\%$  of the movement amplitude) are comparable to those observed in the corresponding running model (Figure 3.11), where the average nRMSE is approximately  $\pm 19\%$  of the movement amplitude.

### 3.2.6.1 Comparison with previous studies

For the prediction of hip angles using the IMU placed on the heel, the results show RMSE values ranging from **4,6757°** (participant 11 for the "Walk100" task) to **33,5641°** (participant 9 for the "Walk80" task).

In comparison, the study by Alcaraz et al. (2021) [31] used an IMU on the foot and deep neural networks, obtaining RMSEs ranging from 1.91° to 2.68°. Compared to these results, this study shows higher errors, demonstrating the superiority of CNNs for predicting hip angles.

The nRMSE values for this location vary between **11,3581%** (participant 11 for the "Walk100" task) and **44,4002%** (participant 9 for the "Walk80" task).

In comparison, the study by Lim et al. (2020) [28] reported nRMSE values lower than those obtained here, ranging from  $5.80\% \pm 0.93$  to  $8.59\% \pm 1.11$  for different slow walking speeds, with an IMU placed on the sacrum, suggesting that the sacrum location should be considered.

The study by Mundt et al. (2020) [23], which used an FNN and simulated data from optical motion capture, obtained an average nRMSE of 10.29%. Comparatively, the results of this sixth model show higher errors.

Finally, the study by Sung et al. (2022), using an LSTM model and an IMU placed on the lateral shank, obtained nRMSE values of 9.01% and RMSEs of 5.47° for intra-subject predictions, confirming that this sixth model had more difficulty capturing the dynamics of hip angles during walking.

## 3.3 Analysis of Pearson correlation coefficients

### 3.3.1 Model 1: Ankle angle predictions during running

		Pearson's r	p
True angles	- Predicted angles (Heel)	0.915	< .001
	- Predicted angles (Mid foot)	0.931	< .001
Predicted angles (Heel)	- Predicted angles (Mid foot)	0.909	< .001

Table 3.2: Pearson's correlations for the Model 1.

For the first model (Table 3.2), the angle predictions associated with both IMU sensor locations show a "very strong" positive correlation with the actual angle values ( $r > 0.9$ ), indicating that both sub-models perform particularly well in their predictions.

The sub-model associated with the IMU placed on the midfoot exhibits a slightly higher correlation with the real angle data compared to the one associated with the IMU placed on the heel, although the difference between the two is minimal. This is consistent with the fact that this location showed lower absolute errors compared to the heel location (Figure 3.5).

Moreover, the strong correlation between the predictions of the two sub-models ( $r = 0.909$ ) suggests that these models likely share similar characteristics in how they predict joint angles.

Finally, the p-values associated with these correlations, all below 0.001, confirm that the observed correlations are statistically highly significant.

### 3.3.2 Model 2: Knee angle predictions during running

		Pearson's r	p
True angles	- Predicted angles (Heel)	0.944	< .001
	- Predicted angles (Mid foot)	0.958	< .001
Predicted angles (Heel)	- Predicted angles (Mid foot)	0.948	< .001

Table 3.3: Pearson's correlations for the Model 2

For the second model (Table 3.3), the conclusions are similar to those of the first model. The predictions of the two sub-models, associated with the two IMU locations, show a "very strong" correlation with the actual angle values ( $r = 0.944$  and  $0.958$ ), with a slight advantage for the IMU placed on the midfoot. This observation corroborates the conclusions drawn from the error distributions (Figure 3.9).

Moreover, the strong similarity between the predictions of the two sub-models ( $r = 0.948$ ) is also confirmed. Finally, all correlations are statistically significant ( $p < 0.001$ ).

### 3.3.3 Model 3: Hip angle predictions during running

		Pearson's r	p
True angles	- Predicted angles (Heel)	0.845	< .001
	- Predicted angles (Mid foot)	0.854	< .001
Predicted angles (Heel)	- Predicted angles (Mid foot)	0.912	< .001

Table 3.4: Pearson's correlations for the Model 3

For the third model (Table 3.4), although the correlations between the predictions of the two sub-models associated with the two IMU placements and the actual angle values are slightly lower than those observed for the first two models, they are still classified as "strong correlations" ( $r = 0.845$  and  $0.854$ ).

The sub-model using the IMU placed on the mid-foot shows a slight superiority, helping to identify the better IMU placement, an aspect that was not clearly established during the analysis of absolute error distributions (Figure 3.13).

The strong correlation between the predictions of the two sub-models ( $r = 0.912$ ) is also confirmed. As with the other models, all correlations are statistically significant ( $p < 0.001$ ).

### 3.3.4 Model 4: Ankle angle predictions during walking

		Pearson's r	p
True angles	- Predicted angles (Heel)	0.797	< .001
	- Predicted angles (Mid foot)	0.781	< .001
Predicted angles (Heel)	- Predicted angles (Mid foot)	0.793	< .001

Table 3.5: Pearson's correlations for the Model 4

For the fourth model (Table 3.4), which aims to predict ankle angles during walking, the correlation coefficients are lower than those observed for running, where the correlations were classified as "very strong" (Table 3.2). However, these correlations are still considered "strong," indicating a significant relationship between the predictions and the actual values.

In this model, the IMU placement on the mid-foot shows a slightly higher correlation ( $r = 0.797$ ) compared to the heel placement ( $r = 0.781$ ), reinforcing that the mid-foot is the better location for the IMU. This finding helps clarify an aspect that was not definitively established during the analysis of absolute error distributions (Figure 3.17).

As with the other models analyzed so far, all observed correlations are statistically significant ( $p < 0.001$ ).

### 3.3.5 Model 5: Knee angle predictions during walking

		Pearson's r	p
True angles	- Predicted angles (Heel)	0.928	< .001
	- Predicted angles (Mid foot)	0.918	< .001
Predicted angles (Heel)	- Predicted angles (Mid foot)	0.944	< .001

Table 3.6: Pearson's correlations for the Model 5

For this fifth model (Table 3.6), dedicated to predicting knee angles during walking, the correlation coefficients are comparable to those observed for running ( $r = 0.944$  and  $0.958$ , Table 3.3), with correlations also classified as "very strong".

In this case, the IMU placed at the heel shows a slightly higher correlation ( $r = 0.928$ ) compared to the one placed at the midfoot ( $r = 0.918$ ), which is consistent with the observations made in the distributions of absolute errors (Figure 3.21).

As with the other models analyzed so far, all observed correlations are statistically significant ( $p < 0.001$ ).

### 3.3.6 Model 6: Hip angle predictions during walking

		Pearson's r	p
True angles	- Predicted angles (Heel)	0.790	< .001
	- Predicted angles (Mid foot)	0.775	< .001
Predicted angles (Heel)	- Predicted angles (Mid foot)	0.799	< .001

Table 3.7: Pearson's correlations for the Model 6

In the case of the last model (Table 3.7), which aims to predict hip angles during walking, the correlation coefficients are slightly lower than those observed for running ( $r = 0.845$  and  $0.854$ , Table 3.4), but these correlations are still classified as "strong".

For this model, the IMU placed at the heel shows a slightly higher correlation ( $r = 0.790$ ) compared to the one placed at the midfoot ( $r = 0.775$ ), which is consistent with the observations made in the distributions of absolute errors (Figure 3.25).

As with the other models analyzed so far, all observed correlations are statistically significant ( $p < 0.001$ ).

## 3.4 Summary of model performances

In conclusion, the analysis of absolute (RMSE, MAE) and relative (nRMSE) error distributions showed that although the mid-foot location generally offers the best results in terms of errors for most models (with the exception of the last two), this performance difference compared to the heel location is not significant. This suggests that both IMU positions can be effectively used for predicting lower limb kinematics. The selection of the mid-foot as the optimal location may be explained by the fact that, for most participants with heel strikes, this location is the first to touch the ground, which could introduce more noise into the data.

The comparison between the walking models, which included a greater number of tasks per participant ("Walk100", "Walk120", "Walk80"), and the running models ("Run100", "Run120"), reveals some interesting findings. Although the inclusion of more tasks might have been expected to reduce prediction errors, it instead tends to increase the variability of errors, with higher maximum values. This highlights the challenge of creating models that can be generalized to a broader population, as opposed to personalized models that would be more specific to an individual.

Overall, the ankle joint during walking proved to be the easiest to predict, while the hip joint during walking presented the greatest challenges. This complexity could be linked to the movements of the hip, which involve several degrees of freedom. Additionally, optoelectronic markers placed at the pelvis are often difficult to position precisely due to the challenge of accurately locating anatomical landmarks in this region. These markers, often placed over participants' clothing, may move independently of the pelvis, introducing noise into the measurements and reducing data accuracy. Furthermore, the markers located at the pelvis are particularly prone to being obscured by the arms during walking or running, where the arms naturally swing. These obstructions can cause intermittent interruptions in the data recorded by the cameras. Although the marker trajectories were interpolated to fill gaps, this method is less reliable than continuous, unobstructed capture.

The following Table 3.8 summarizes the optimal IMU placements obtained for each model, as well as the associated minimum and maximum error values (RMSE). Pearson correlation coefficients are also included.

	Running models			Walking models		
	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6
<b>Predicted angle</b>	Ankle	Knee	Hip	Ankle	Knee	Hip
<b>Best IMU location</b>	Midfoot	Midfoot	Midfoot	Midfoot	Heel	Heel
<b>Minimum value</b>	2.8807°	4.21°	3.8292°	<b>2.616°</b>	3.142°	4.6757°
<b>Maximum value</b>	8.8487°	15.3895°	26.4602°	16.4349°	22.7862°	<b>33.5641°</b>
<b>Average value</b>	6.0°	9.8°	14.1°	9.5°	12.4°	19.1°
<b>Pearson's coefficient</b>	0.931	0.958	0.854	0.781	0.928	0.790
<b>p-value</b>	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001

Table 3.8: Summary of the models' performances, including the best IMU location, and error statistics.

The next section, which will detail the predictions made by each model for each participant, placement, and task, will be particularly interesting as it will allow us to precisely identify the participants or tasks that contribute to the large variability observed so far across all models.

### 3.5 Model predictions based on the gait cycle

In this section, the flexion angles of the ankle, knee, and hip, measured in degrees in the sagittal plane, will be presented as a function of the percentage of the gait cycle. Figure 3.29 precisely shows the measurement points for these angles.

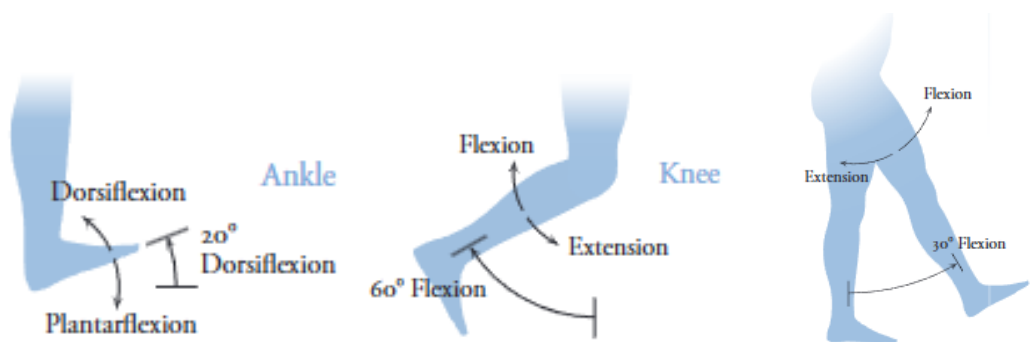


Figure 3.26: Ankle

Figure 3.27: Knee

Figure 3.28: Hip

Figure 3.29: Movements of the three main lower body joints in the sagittal plane [11].

The ankle angle is traditionally measured between the front of the tibia and the back of the foot in biomechanics. The neutral position of the ankle, where the angle is zero, corresponds to a position where the foot is perpendicular to the leg. When the foot moves toward the tibia (upward), the angle becomes positive, a movement known as dorsiflexion. Conversely, when the foot moves away from the tibia (downward), the angle becomes negative, corresponding to plantarflexion.

The knee angle is generally measured between the tibia and the femur. A full extension of the leg, where it is completely straight, corresponds to an angle of 0°, while maximum flexion, corresponding to a positive angle, is reached when the tibia moves closer to the back of the thigh.

The hip angle is measured between the femur and the torso. The reference position, at  $0^\circ$ , corresponds to the neutral anatomical position, where the person is standing upright, with the body straight, feet aligned under the hips, and legs extended. Hip flexion corresponds to a reduction in the angle between the thigh and the torso.

### 3.5.1 Model 1: Ankle angle predictions during running

Figure 3.31 presents the mean ankle cycles during the "Run00" task, expressed as a percentage of the running cycle. Each graph represents a participant, with the black curves corresponding to the actual ankle angle values, and the orange and blue curves representing the predictions of the two models associated with the two IMU placements (heel and midfoot). The shaded areas around the curves indicate the standard deviation, reflecting the variability of the predictions. The expected shape of these cycles for this joint is illustrated in Figure 3.30.

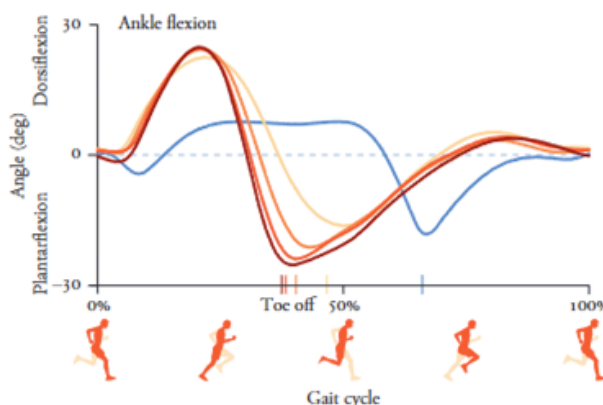


Figure 3.30: Illustration of expected ankle flexion angles during a running cycle [11].

As shown in this reference Figure 3.30, since most participants in this study had a heel strike during running, the ankle cycles begin with an almost neutral position (angle close to  $0^\circ$ ) at the initial ground contact. Next, the first positive peak observed on these curves (maximum dorsiflexion) corresponds to the body moving over the supporting foot, preparing for the transition to the swing phase. Finally, the gradual decrease in the dorsiflexion angle to the minimum peak (corresponding to maximum plantarflexion) occurs when the heel lifts off the ground, leaving only the toe in contact with the ground just before entering the swing phase.

In Figure 3.31, it can be noted that participants 1, 2, 9, and 19 likely contributed to the increased errors of the first model, as they exhibit greater variability (wider shaded areas) in both predicted and actual angle values.

As expected, the two curves associated with the different IMU placements follow the actual values quite similarly, reflecting the fact that the difference between the average errors of the two models is not significant.

Moreover, the average cycles displayed for some participants show a slight shift compared to the expected reference cycle. This discrepancy is explained by the use of a fixed threshold value to segment the walking and running cycles for all participants. In the context of this study, this threshold was applied uniformly to all participants, based on an average of all thresholds specific to each participant. Ideally, however, a personalized threshold should have been determined for each participant and each specific task. This remark applies to all the figures presented in the subsequent sections of this work.



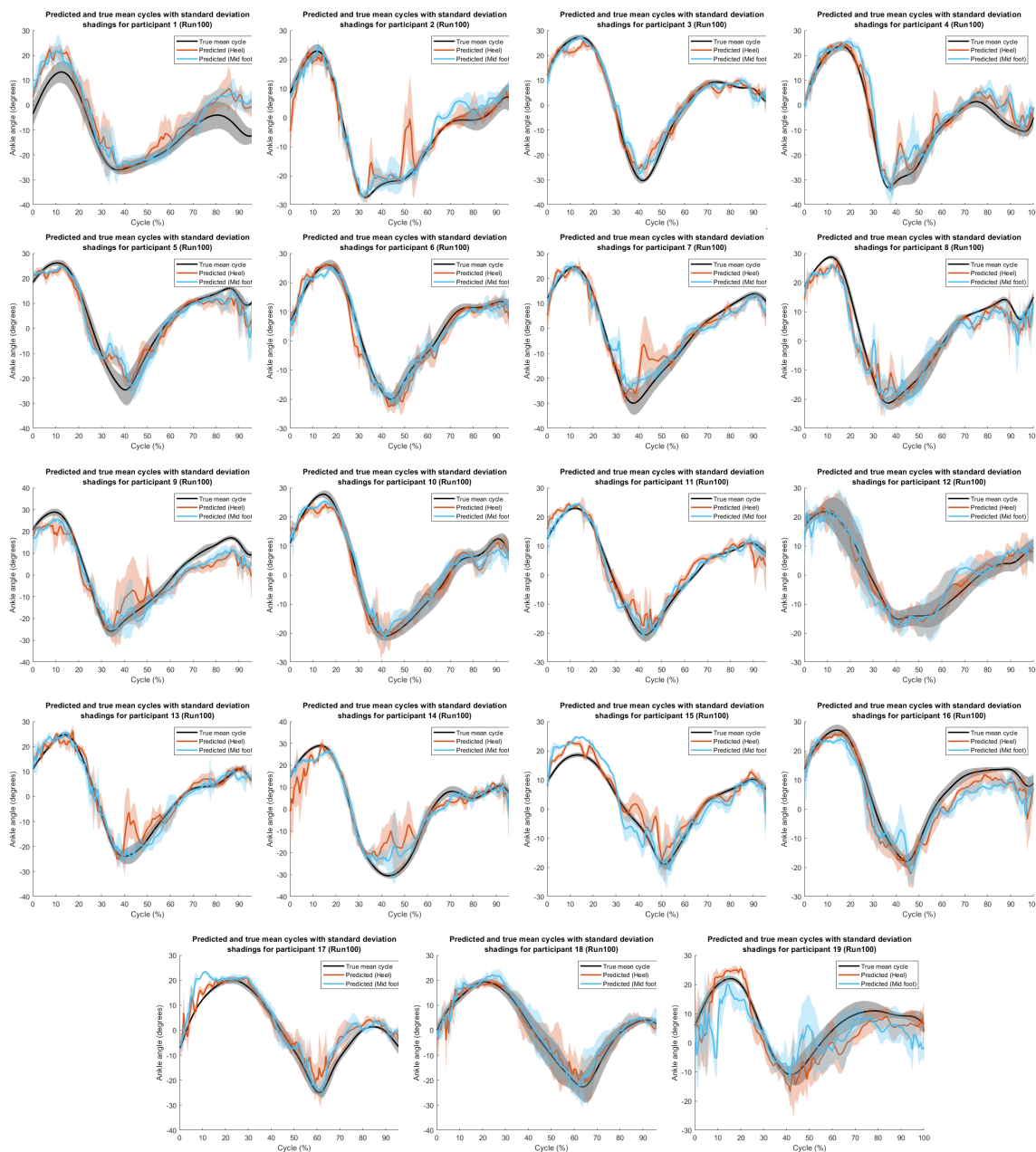


Figure 3.31: Predicted and true mean ankle cycles for "Run100".

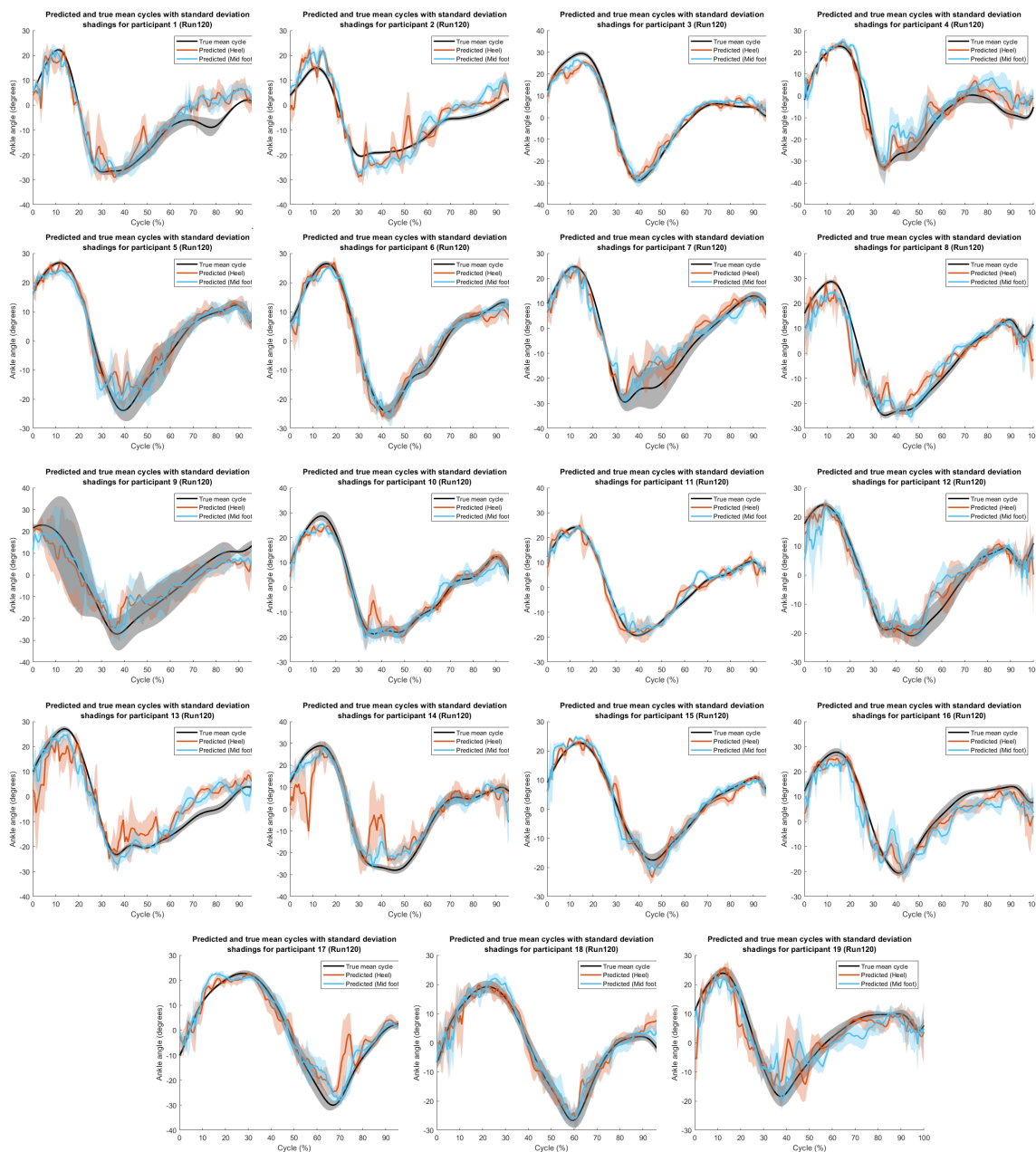


Figure 3.32: Predicted and true mean ankle cycles for "Run120"

Figure 3.32 is similar to the previous Figure 3.31, but it pertains to the "Run120" running task. This analysis allows us to examine how the model performs under conditions of faster running.

Overall, the general observations remain consistent with those made for the "Run100" task. The primary differences noted are a slight increase in the peak values of the ankle angles for the majority of participants, likely due to the greater range of joint motion induced by the higher running speed. This is in line with the reference Figure 3.30 which shows more pronounced peaks for the darker curves associated with higher speeds.

Regarding the variability in angle values, it does not significantly increase compared to the "Run100" task for each individual participant. This suggests that these variabilities are not specifically attributed to higher running speeds but are rather inherent to each participant.

### 3.5.2 Model 2: Knee angle predictions during running

Figure 3.34 shows the predicted and actual mean cycles of knee flexion angles for the two IMU sensor locations, as well as for the 19 participants during the "Run100" running task. The associated standard deviations are also represented.

The expected shape of these cycles for this joint is illustrated in Figure 3.33.

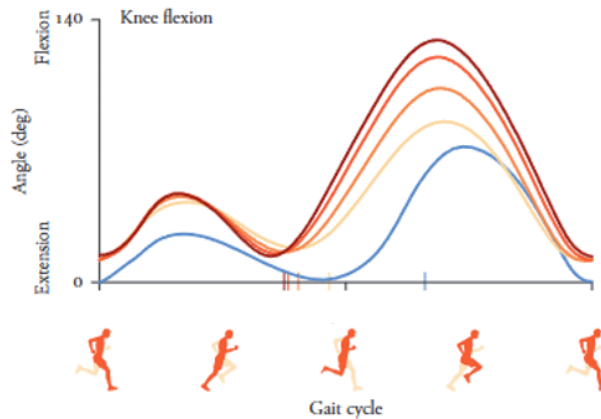


Figure 3.33: Illustration of expected knee flexion angles during a running cycle [11].

On these reference curves (Figure 3.33), the first flexion peak typically occurs shortly after the foot touches the ground during the stance phase, reflecting the slight knee bend to absorb the impact. The near-zero angle indicates the moment when the leg reaches almost full extension, allowing effective propulsion before the foot lifts off the ground to enter the swing phase. The second, higher flexion peak occurs during the swing phase when the foot is in the air. At this point, the knee bends more sharply than during the first peak, shortening the leg length to enable a quick foot swing forward in preparation for the next step.

The predicted mean cycles derived from data captured by the two sensor locations satisfactorily follow the main dynamics of knee flexion movement. The participants with the greatest variability in actual and predicted angle values are participants 1, 2, 12, 18, and 19.

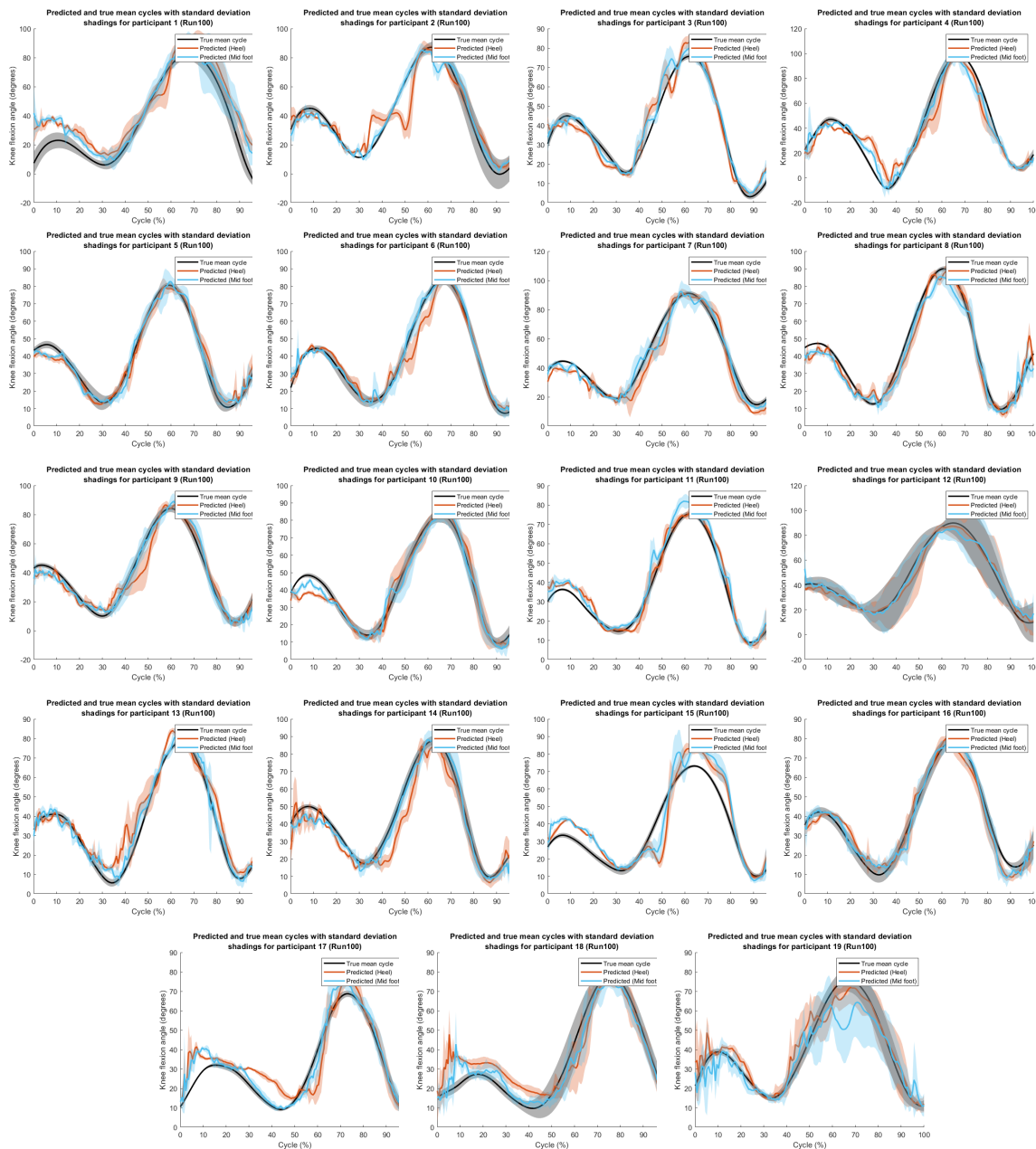


Figure 3.34: Predicted and true mean knee cycles for "Run100".

Figure 3.35 is similar to the previous 3.34, but for the "Run120" running task. Its analysis allows for examining how the model performs under faster running conditions.

The main observation is a slight increase in peak knee angle values for the majority of participants, which is attributed to a greater range of joint movements induced by the increased running speed.

The variability observed in the predicted and actual angle curves within the same participant also persists for this task, but does not show a significant increase compared to the "Run100" task. This suggests that these variations are more dependent on the individual characteristics of the participants rather than the speed at which the task is performed.

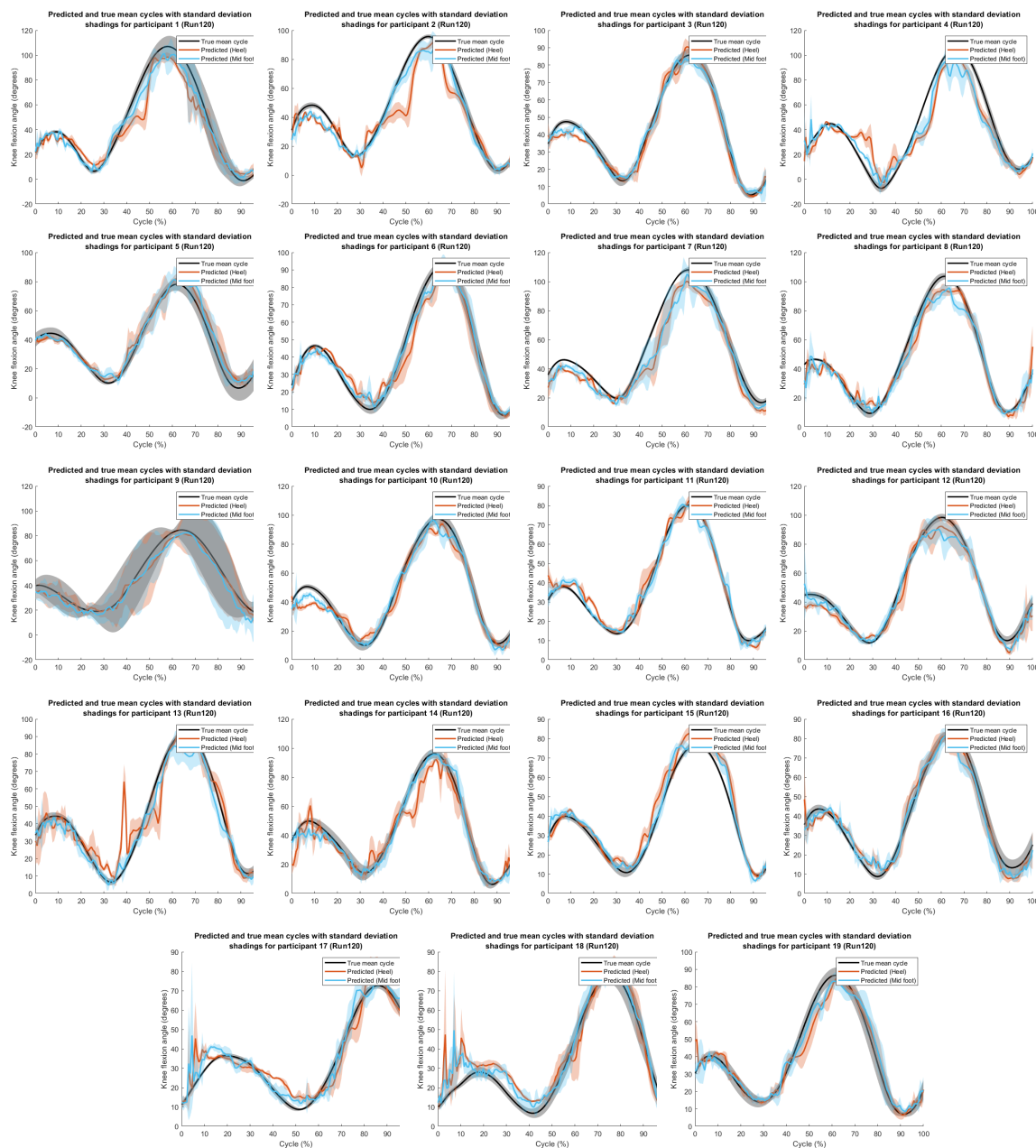


Figure 3.35: Predicted and true mean knee cycles for "Run120".

### 3.5.3 Model 3: Hip angle predictions during running

Figure 3.37 presents the predicted and actual mean hip angle cycles for the two IMU sensor locations and for the 19 participants during the "Run100" task, along with the associated standard deviations.

The expected shape of these cycles for this joint is illustrated in Figure 3.36.

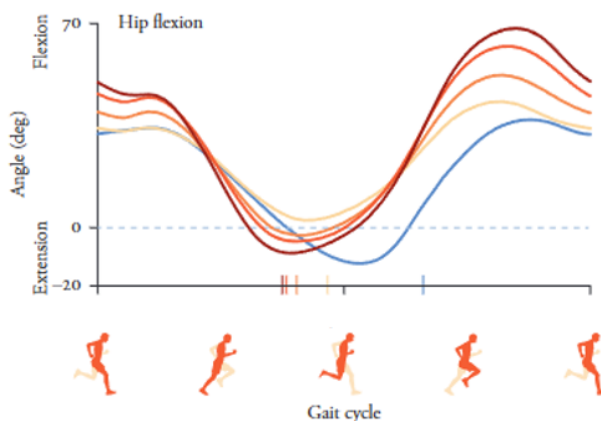


Figure 3.36: Illustration of expected hip flexion angles during a running cycle [11].

In these reference curves (Figure 3.36), the first positive peak corresponds to the moment when the foot touches the ground, with the hip flexed to absorb the impact and stabilize the body. The dip around  $0^\circ$ , occurring after the first peak, represents the maximum hip extension. This happens as the body moves over the supporting foot, preparing to leave the ground and enter the swing phase. Finally, the second peak, usually the highest, corresponds to the swing phase, where the hip is in maximum flexion to bring the foot forward in preparation for the next ground contact.

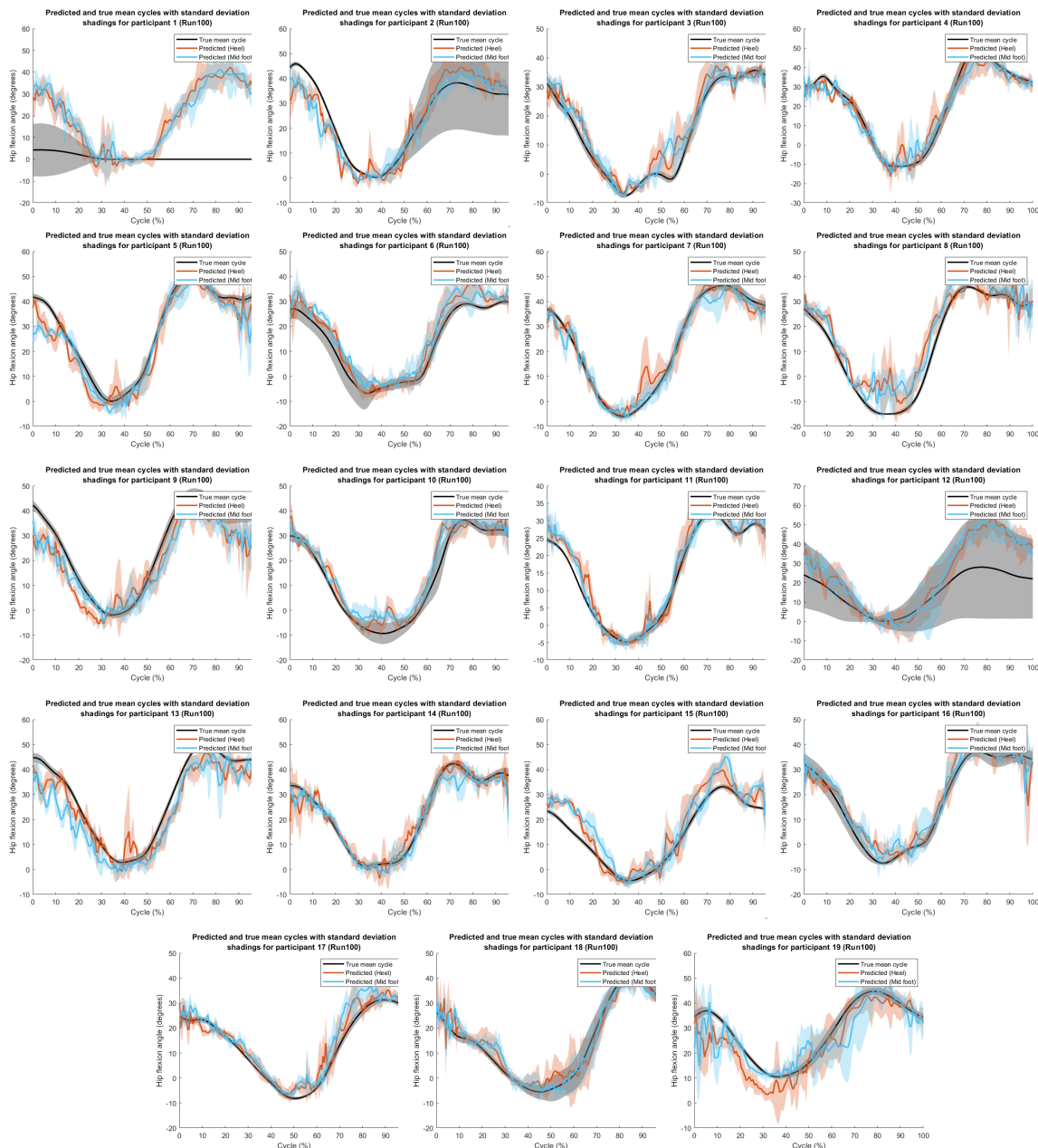


Figure 3.37: Predicted and true mean hip cycles for "Run100"

A notable observation is that participants 1, 2, and 12 exhibit particularly high variability in the actual angle values (very wide shaded black areas). This variability is responsible for participant 4's reduced performance of this third model, as already noted in the previous section (Table 3.8). Indeed, with such variability in the actual angle values, the model will struggle during training to capture a stable and coherent relationship between the input data features (IMU sensors) and the predicted outputs (joint angles).

Moreover, for the first participant, the hip angle signal is completely absent. The following Figure 3.38 shows that, after the filtering step in Visual3D, the last 5 seconds of the signal were lost. This phenomenon is known as "edge artifacts" and is frequently encountered when applying digital filters [77]. These artifacts occur because a low-pass filter, to smooth a given point in a signal, uses the values on either side of that point to calculate an average. However, at the signal's edges, the filter lacks future values needed to perform this calculation accurately. In this specific

case, Visual3D may have been unable to generate a reliable value for these last seconds, perhaps due to excessive noise or other anomalies in the data, leading to the deletion of these segments.

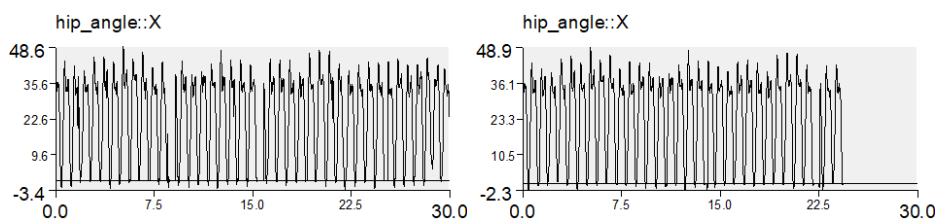


Figure 3.38: Visual3D hip angle signals for participant 1 and "Run100" before filtering (left) and after filtering (right).

However, despite this loss of signal, the model, having been trained on the initial full signal, managed to capture and predict what the signal should have been, as evidenced by the orange and blue curves corresponding to the predictions.

These observations explain why this third model exhibits higher errors and distributions with heavy tails (Figure 3.13) compared to the first two models. Despite these findings, the predicted curves for most other participants show a good match with the actual hip angle values. This suggests that if the real data had been of better quality, the model's performance would have been significantly improved. Figure 3.39 is similar to the previous Figure 3.37, but for the "Run120" running task.

Once again, some participants exhibit significant variability in the actual hip angle values. However, it is noteworthy that these are not exactly the same participants as before, confirming that this variability is more attributable to the individual characteristics of the participants rather than the speed of the task.

A slight increase in the peak hip angle values can also be observed for the majority of participants.



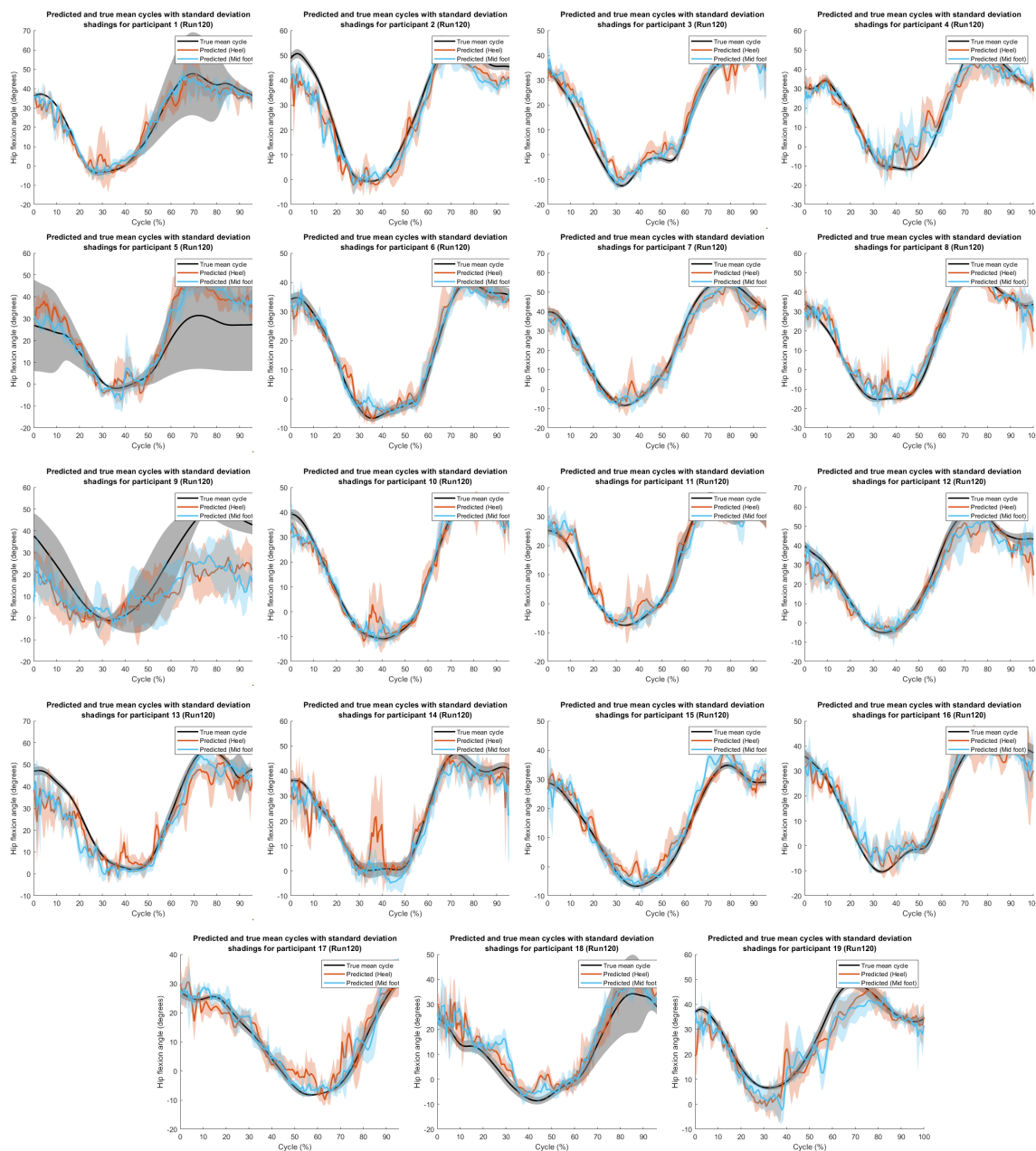


Figure 3.39: Predicted and true mean hip cycles for "Run120"

### 3.5.4 Model 4: Ankle angle predictions during walking

Figure 3.41 shows the average ankle cycles during the "Walk100" task, expressed as a percentage of the gait cycle. The expected shape of these cycles for this joint is illustrated in Figure 3.40.

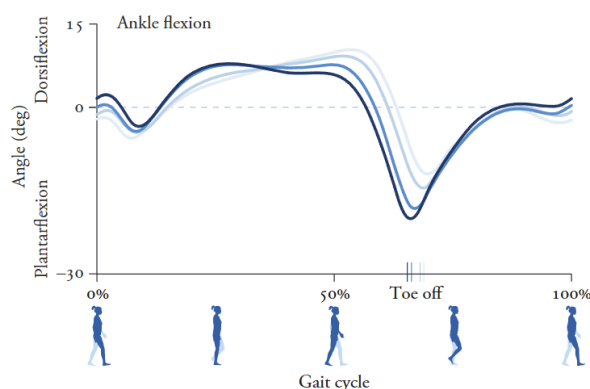


Figure 3.40: Illustration of expected ankle flexion angles during a walking cycle [11].

While some participants exhibit curves that closely match this reference (Figure 3.40), a significant number of curves deviate from the expected pattern. The main issue here lies not so much in the model predictions, but rather in the actual angle measurements (black curves), particularly in their segmentation into cycles, which do not consistently align with the expected curve shape. Additionally, these real signals show considerable variability, as indicated by the wide shaded areas, which in turn affects the accuracy of the model predictions and contributes to the large absolute errors noted in the previous section (Table 3.8).

Regarding the gait cycle, the ankle initially remains in a neutral position as the foot makes contact with the ground (angle close to  $0^\circ$ ), then transitions into dorsiflexion (positive angle) as the tibia advances over the supporting foot. The ankle subsequently shifts to a negative angle as the toe lifts off the ground, resulting in a negative peak, albeit with a smaller amplitude compared to what is observed during running.



Figure 3.41: Predicted and true mean ankle cycles for "Walk100"

Figure 3.42 is similar to the previous Figure 3.41, but for the "Walk120" task.

As previously observed for "Walk100", a high degree of variability is also notably present in the real cycles.



Figure 3.42: Predicted and true mean ankle cycles for "Walk120"

Figure 3.43 is similar to the previous Figures 3.41 and 3.42, but this time focuses on the "Walk80" task.

The observations remain consistent with those made for the "Walk100" and "Walk120" tasks. There is significant variability in the real cycles, which suggests that this variability is not specifically related to the walking speed, as it is present in both "Walk120" and "Walk80", but is rather linked to the individual participants.

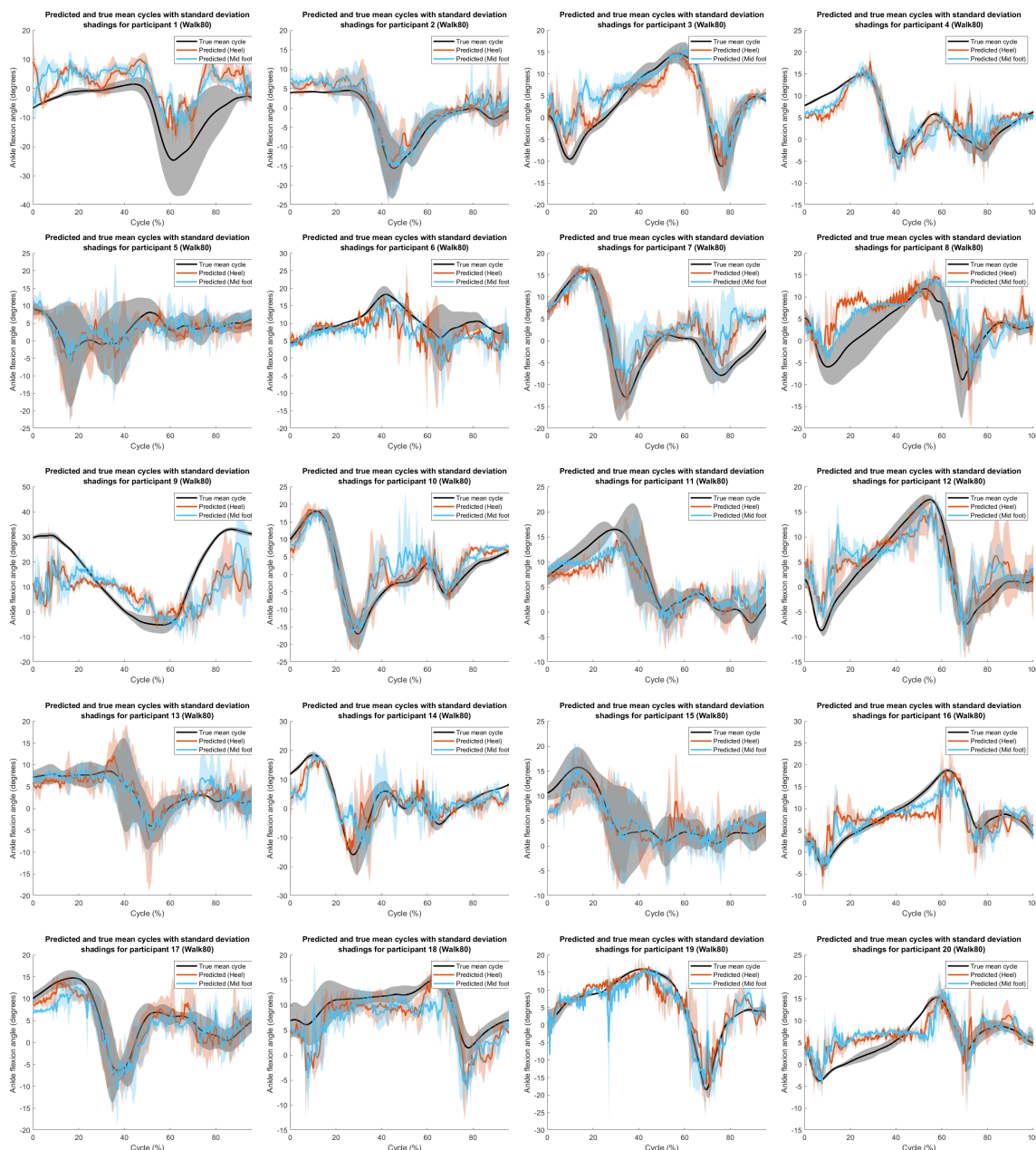


Figure 3.43: Predicted and true mean ankle cycles for "Walk80"

### 3.5.5 Model 5: Knee angle predictions during walking

Figure 3.45 presents the average knee cycles during the "Walk100" walking task, expressed as a percentage of the gait cycle. The expected shape of these cycles for this joint is illustrated in Figure 3.44.

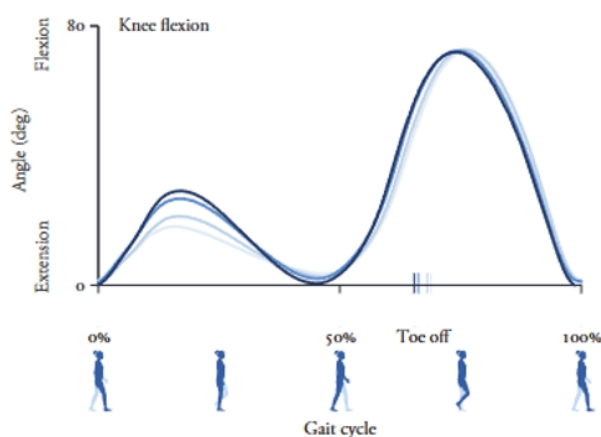


Figure 3.44: Illustration of expected knee flexion angles during a walking cycle [11].

These reference curves (Figure 3.33) illustrating the knee flexion angle during walking are quite similar to those observed during running (Figure 3.33), although the angle amplitudes are significantly lower here. The first flexion peak generally occurs shortly after the foot makes contact with the ground. The near-zero angle marks the moment when the leg reaches almost full extension, just before the foot lifts off the ground, coinciding with the phase where both feet are in contact with the ground (a phase that is not present during running). The second, more pronounced flexion peak occurs during the swing phase when the foot is in the air. At this point, the knee bends more than during the first peak, shortening the leg length and facilitating the forward movement of the foot in preparation for the next step.

Compared to the curves obtained for the same joint during running (Figure 3.34), these show greater variability in the measured (real) angles between participants.

Additionally, for participant 9, the real angle signal was also lost due to edge artifacts, a problem similar to what was previously mentioned for model 3 (Figure 3.37). This underscores the need to improve the digital filtering techniques used in this study to minimize these artifacts and preserve the integrity of the data.

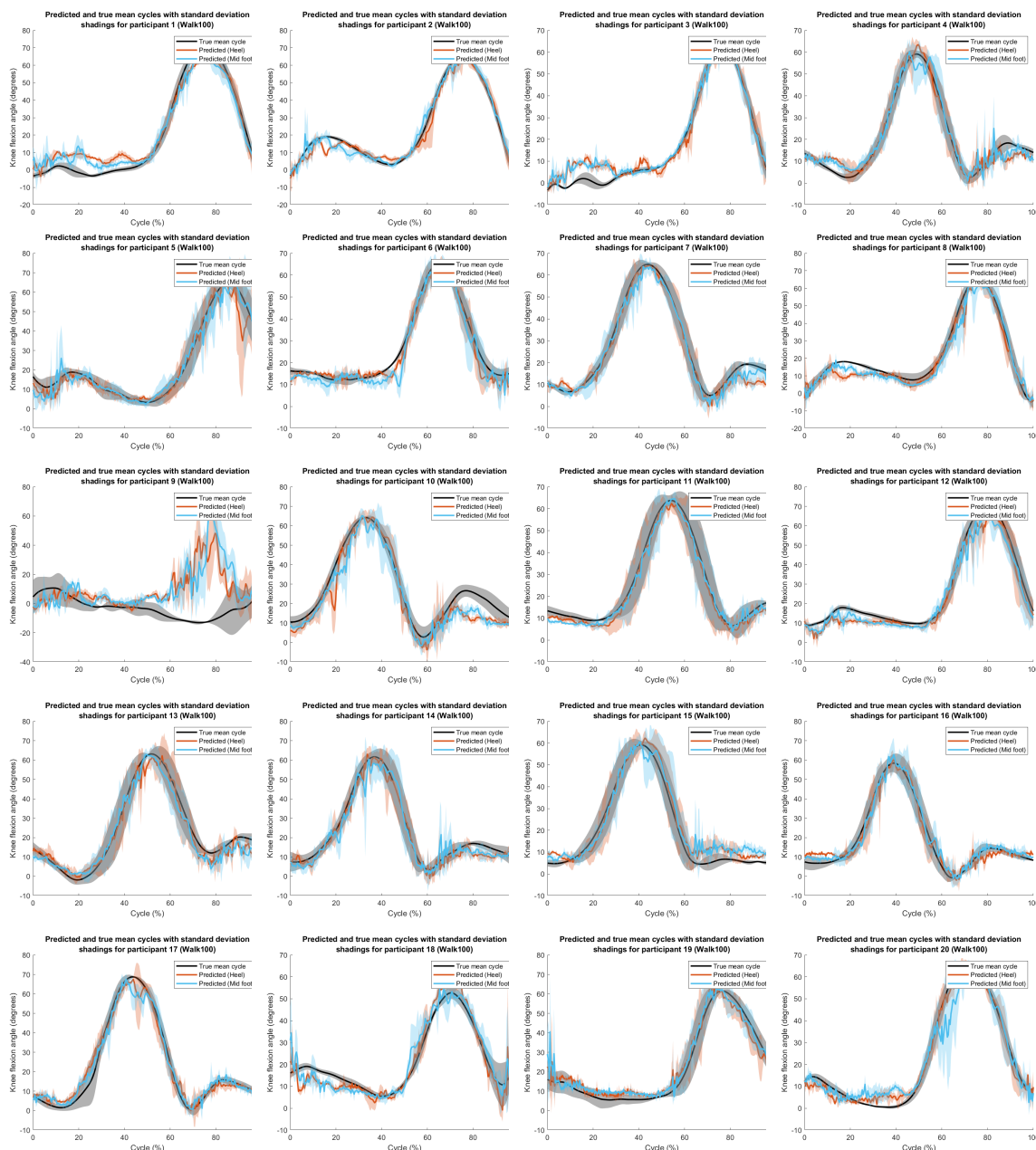


Figure 3.45: Predicted and true mean knee cycles for "Walk100".

Figure 3.46 is similar to the previous Figure 3.45, but for the "Walk120" task.

As previously observed for "Walk100", a high degree of variability is also notably present in the real cycles.

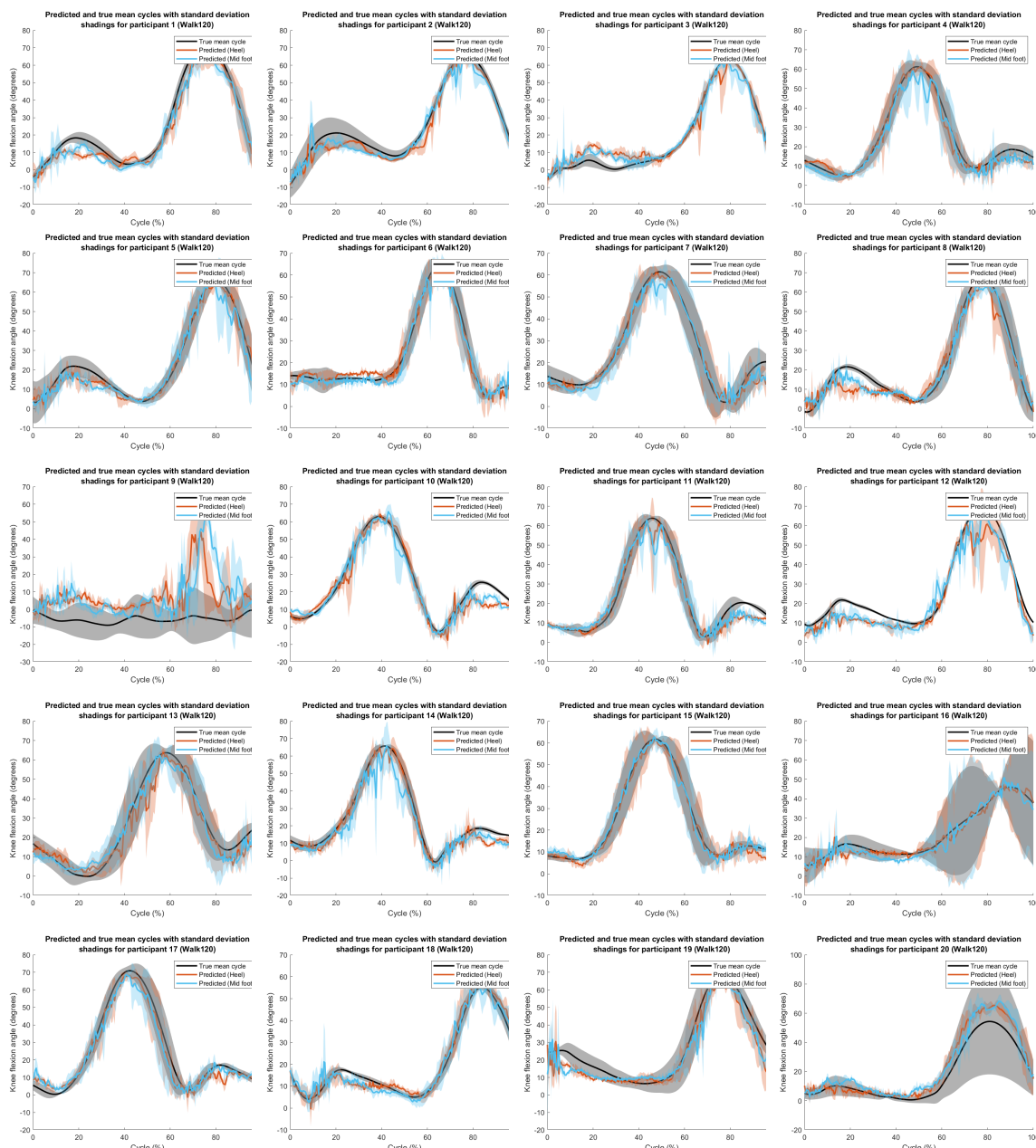


Figure 3.46: Predicted and true mean knee cycles for "Walk120".

Figure 3.47 is similar to the previous Figures 3.45 and 3.46, but this time focuses on the "Walk80" task.

This Figure 3.47 shows the most variability, even though it was performed at speeds lower than the "Walk100" and "Walk120" tasks, which suggests that this variability is not specifically linked to walking speed.

Added to this variability is the fact that the cycles have not been segmented in the right place.





Figure 3.47: Predicted and true mean knee cycles for "Walk80".

### 3.5.6 Model 6: Hip angle predictions during walking

The Figure 3.49 shows the mean hip cycles during the "Walk100" task, expressed as a percentage of the walking cycle. The expected form of these cycles for this joint is illustrated in Figure 3.48.

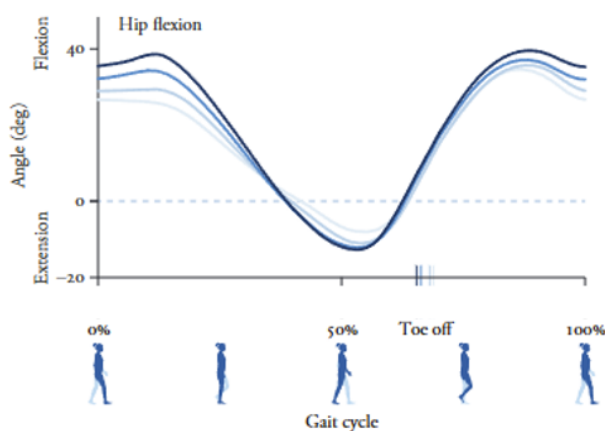


Figure 3.48: Illustration of expected hip flexion angles during a walking cycle [11].

When analyzing the walking cycle, the hip is initially in a flexed position when the foot makes contact with the ground (corresponding to the maximum peak), with the foot slightly ahead of the trunk. Then, during the stance phase, the flexion angle decreases as the foot passes under the body, reaching a peak of extension (minimum peak) when both feet are in contact with the ground (a phase not present during running), just before entering the swing phase. During this swing phase, as the foot moves forward again, the hip flexion increases accordingly.

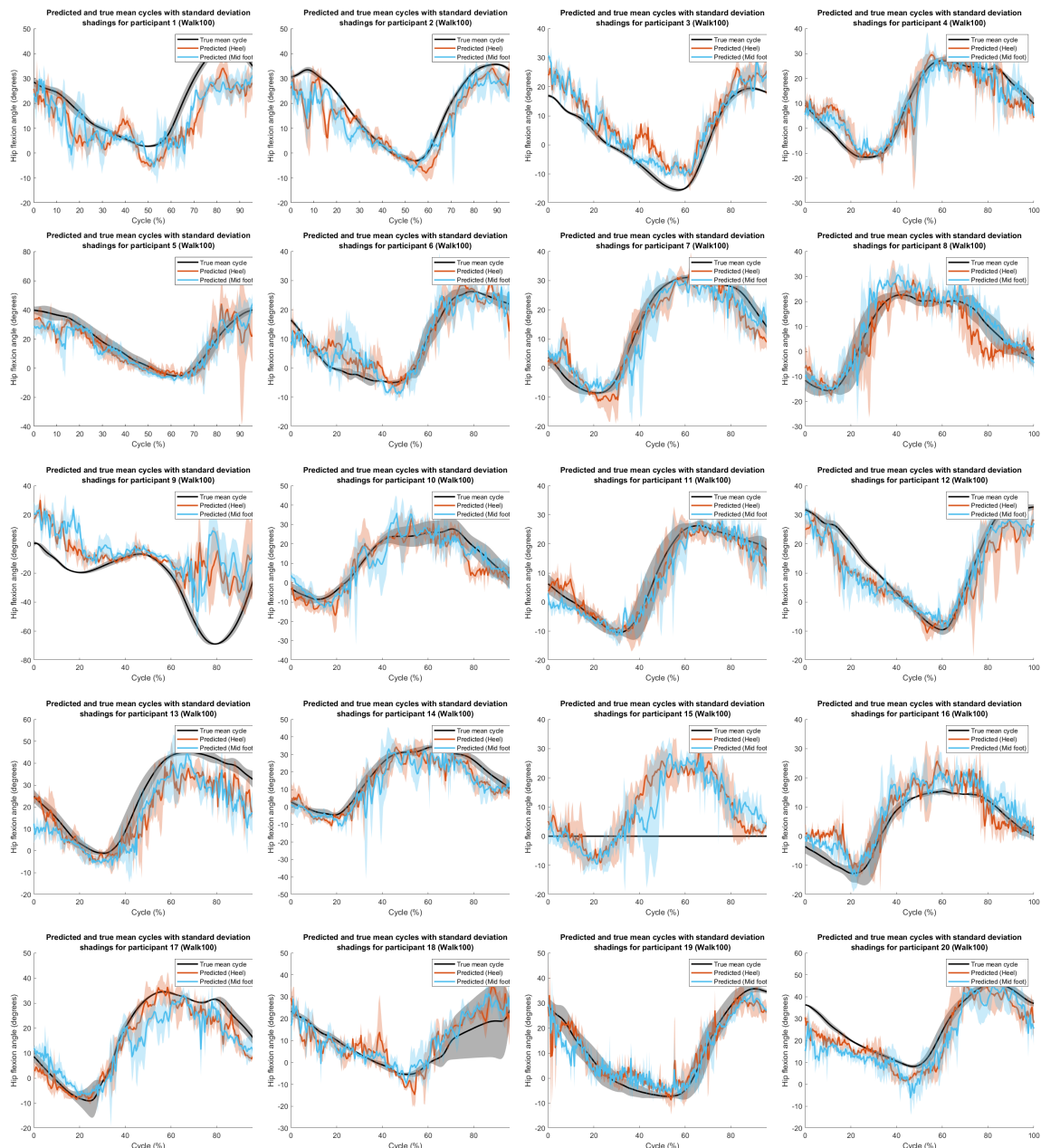


Figure 3.49: Predicted and true mean hip cycles for "Walk100".

Figure 3.50 is similar to the previous Figure 3.49, but for the "Walk120" task.

As previously observed for "Walk100", a high degree of variability is also notably present in the real cycles.

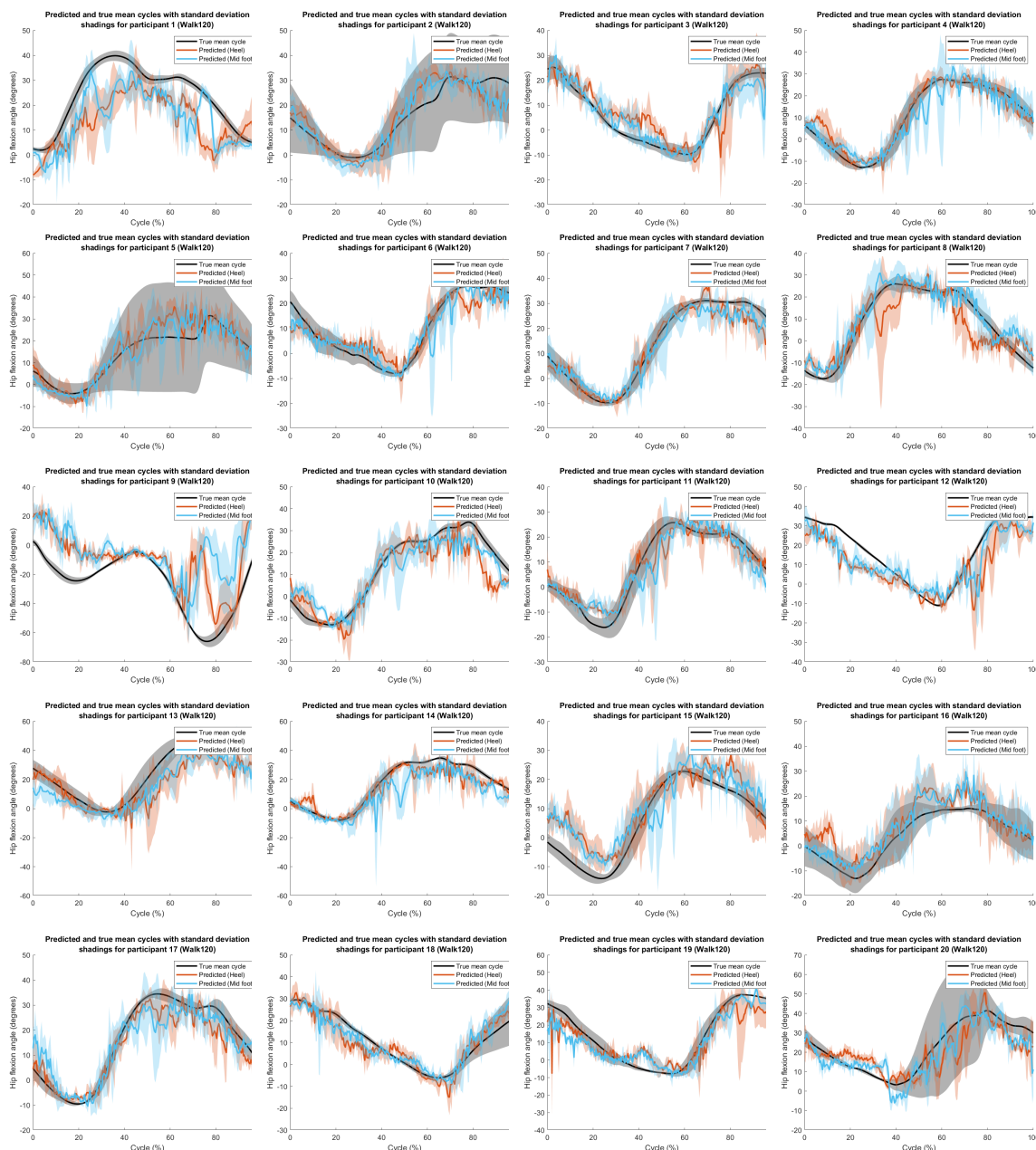


Figure 3.50: Predicted and true mean hip cycles for "Walk120".

Figure 3.51 is similar to the previous Figures 3.49 and 3.50, but this time focuses on the "Walk80" task.

The observations remain consistent with those made for the "Walk100" and "Walk120" tasks. There is significant variability in the real cycles, which suggests that this variability is not specifically related to the walking speed, as it is present in both "Walk120" and "Walk80", but is rather linked to the individual participants.

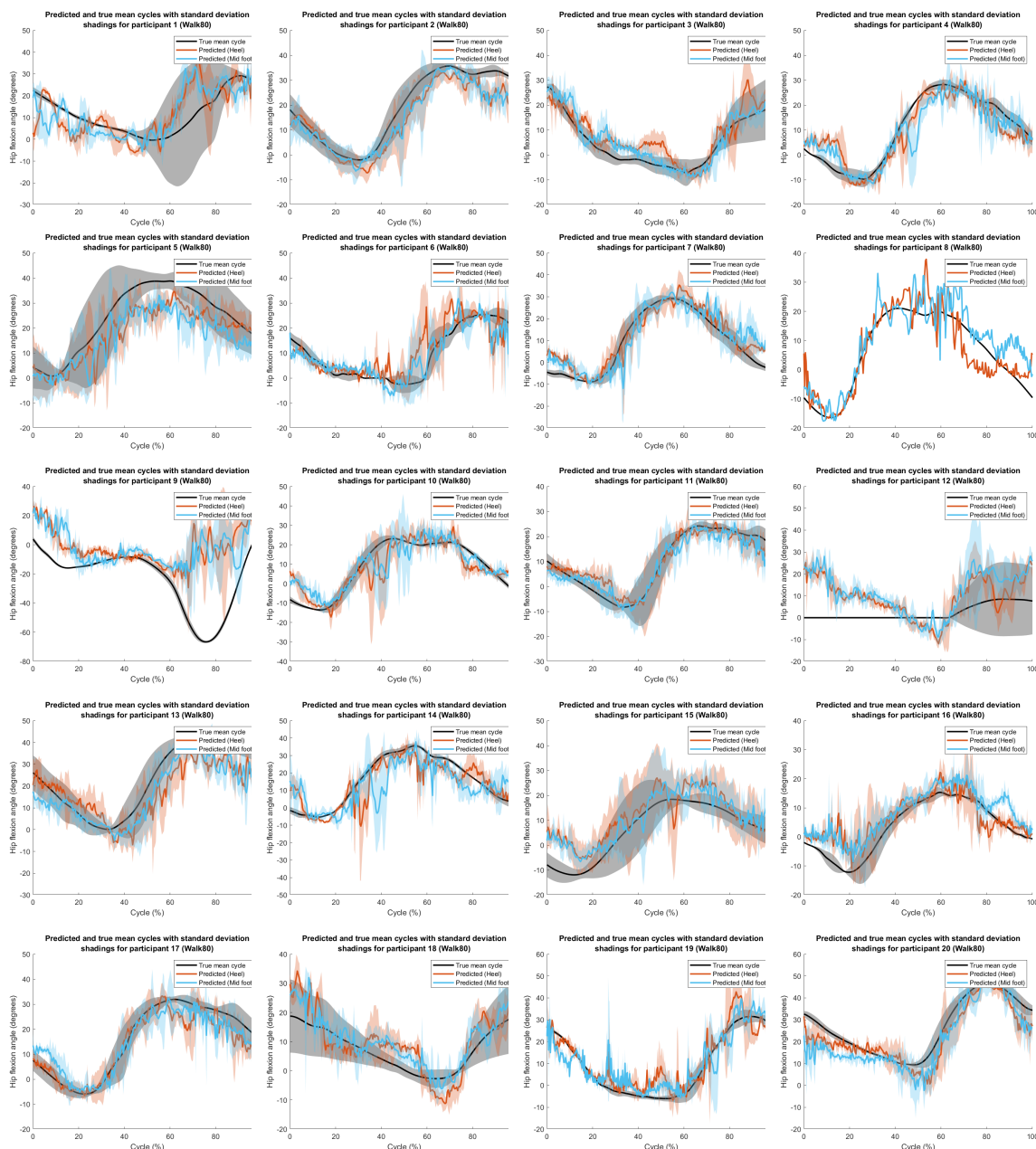


Figure 3.51: Predicted and true mean hip cycles for "Walk80".

## Chapter 4

# Limitations of this study

This fourth chapter provides an in-depth analysis of the limitations encountered during this work, while also viewing them as opportunities for improvement in future research. It is divided into three main sections. The first section is devoted to experimental limitations. The second section addresses the limitations related to data preprocessing, particularly concerning the choices of digital filtering. Finally, the last section discusses the inherent limitations of the machine learning models used.

### 4.1 Limitations related to the experimental phase

The main limitation of the experimental phase, and likely the most significant limitation of this study, lies in the substantial variability observed among individual participants, which inevitably leads to significant differences between them. This variability was particularly evident during the evaluation of absolute (RMSE, MAE) and relative (nRMSE) errors. It was noted that two participants within the same model and for the same task could exhibit very different error values. Additionally, these errors tended to increase in proportion to the number of tasks included in the models. For instance, the walking models, which incorporated a greater number of tasks ("Walk100", "Walk120", "Walk80"), showed significant discrepancies between minimum and maximum errors compared to the running models, which included fewer tasks ("Run100", "Run120").

This observation was further confirmed during the analysis of the average cycles for each participant and each type of task. The notable variations in the average measured angles, and consequently in the model predictions, strongly suggest that these differences stem from the experimental phase itself, particularly the capture of optical movements.

Several hypotheses can be proposed to explain this variability.

Firstly, variability in the placement of markers between participants, despite being conducted with great care, remains a potential source of error. Additionally, certain marker locations, such as T8, the iliac spines, etc., were often positioned over the participants' clothing, which can lead to marker displacements independent of adjacent segments, even when the clothing is well-fitted to the skin.

Secondly, as previously mentioned, optoelectronic systems are subject to noise. Although this noise was minimized through the application of digital filters, residual errors in the measurement of joint angles may persist, which could subsequently influence the accuracy of the predictive models.

Finally, the fact that some participants were not particularly athletic, and even less accus-

tomed to using a treadmill, could explain part of the observed variability. Their walking or running technique, potentially irregular due to this lack of familiarity with the measurement devices, likely contributed to this variability.

To reduce variability within the same participant in future studies, and thereby develop more generalizable predictive models, capable of adapting to a wide range of behaviors and physical characteristics, several improvements can be considered in the experimental protocol. For example, rather than using all the tasks performed by the participants, as was done here, it would be preferable to include only those that demonstrate sufficient regularity. This would help minimize the impact of abnormal fluctuations on the overall performance of the models. Additionally, it would be beneficial to increase the familiarization time for participants with the treadmill before data collection. The time allocated in this study, ranging from 30 seconds to 1 minute, could be extended to allow participants to fully adapt to the walking or running pace, thereby reducing variations due to initial awkwardness.

## 4.2 Limitations related to data preprocessing

The choice of the cutoff frequency for the Butterworth low-pass filter can have a significant impact on the filtered kinematic data and, consequently, on the final results of the models [60]. Generally, this cutoff frequency is selected to preserve the majority of the signal's power, typically between 95% and 99% [60]. Since the power of a signal is proportional to the square of its amplitude, the higher the task speed, the greater the amplitude of movement, and therefore, the higher this frequency should be. However, in this study, a fixed value was used for all tasks, both walking and running, which is not optimal and may introduce more noise and/or fail to capture the full essence of the signal.

Moreover, as discussed during the analysis of hip angle predictions during running (Section 3.5.3), digital filtering can introduce "edge artifacts," making the ends of the angle signals potentially less reliable. To improve this work, it would be beneficial to adopt approaches to minimize these artifacts. A simple solution would be to artificially extend the initial signal before filtering by adding extrapolated signal segments. Although these segments do not provide new information, they allow the filter to have the necessary data to function correctly. After filtering, these additional segments can be cut off to retain only the reliable part of the signal [77]. Alternatively, another option would be to extend the acquisition phases during the experimental phase to obtain a longer signal, from which only a portion free of artifacts would be selected for analysis.

Another limitation of this study relates to the segmentation of the walking and running cycles. A fixed threshold value was used uniformly across all participants to segment these cycles. While this approach simplified the process, it introduced slight discrepancies in the alignment of the cycles when compared to the expected reference cycle. Ideally, a personalized threshold should have been determined for each participant and for each specific task to ensure more accurate segmentation. This limitation is particularly important and should be considered when interpreting the figures and results presented throughout this work.

## 4.3 Limitations related to Machine Learning models

Although the training and test data were correctly separated to avoid any bias in the predictions, the hyperparameter optimization via Bayesian optimization was performed on the entire dataset before this separation. This means that the model's structure was influenced by the overall characteristics of the input and output data, which could introduce a bias in the hyperparameter selection,

as they are adjusted based on both the training and test data, potentially leading to an overestimation of the performance of the various models evaluated.

Furthermore, the choice of the regression model used in this study could be improved. As mentioned in Section 1.4, exploring more sophisticated models, such as convolutional neural networks (CNN), would be relevant. When comparing the absolute and relative errors of the models developed here, CNNs have shown superior accuracy. Additionally, recurrent neural networks (RNN), which have shown performance comparable to the models developed here, could be considered for real-time applications, as they offer the ability to capture the temporal dependencies of the data.

Moreover, one of the limitations of the FNN models, as observed here, is that they require fully processed data (with normalized inputs and input and output sequences of the same length), which requires that the data be fully recorded. This constraint makes FNNs unsuitable for real-time applications, a field toward which this work could evolve to make kinematic analysis more accessible to the general public. In contrast, CNNs and RNNs, although more complex to implement, are better suited for these real-time applications.

However, one of the reasons why researchers may hesitate to adopt deep networks such as CNNs and LSTMs is their need for large training datasets, a requirement that is difficult to meet in the field of biomechanics. To overcome this challenge, an emerging approach in the literature involves artificially augmenting datasets based on existing data [25, 26].



## Chapter 5

# Conclusions and prospects

The analysis of joint kinematics during running and walking has numerous applications, ranging from rehabilitation to enhancing sports performance. Traditionally conducted in laboratories, this analysis is not accessible to the general public and may not accurately reflect real-world conditions. Therefore, this final study aimed to make joint kinematics analysis more accessible by utilizing portable inertial sensors (IMUs). These sensors, being cost-effective and compact, allow for use outside of motion analysis laboratories. The ultimate goal of this work was to predict lower limb kinematics during walking and running using a single IMU.

Given the complexity of data from inertial sensors and the nonlinear relationships with joint kinematics, several feedforward neural network models were developed to accurately predict these joint angles.

To train and validate these neural networks, an experimental protocol was established, involving the collection of data from twenty healthy participants. These participants completed six 30-second trials at different speeds, including three walking trials and three running trials, on a treadmill at the Sart Tilman Movement Analysis Laboratory. An optoelectronic system, considered the gold standard in movement analysis, was used to obtain lower body joint angles, which served as a reference for training the models. Simultaneously, two IMU sensors were placed on the participants to determine the optimal location for predicting joint angles.

The raw accelerometer and gyroscope data from the inertial sensors were used as inputs for the machine learning models. The 3D data from the optoelectronic system were preprocessed through interpolation, scaling, digital filtering, and optimization (inverse kinematics) processes to obtain the joint angles.

The neural network models were designed using MATLAB's `fitrnet` function, with hyperparameter optimization performed through Bayesian optimization, minimizing the stratified k-fold cross-validation loss (k=5).

For each participant, several performance indicators were calculated, such as the Root Mean Square Error (RMSE), the Mean Absolute Error (MAE), the normalized Root Mean Square Error (nRMSE), and the Pearson correlation coefficient. These metrics were then synthesized into violin plots for each model, allowing for clear and intuitive visualization of the error distribution. The results revealed that the two sensor positions tested were generally similar for predicting lower body kinematics, with a slight preference for the mid-foot location. The ankle flexion angles during running showed the highest accuracy, with a minimum RMSE of 2.616° for running and 2.8807° for walking. Predictions for knee and hip angles also showed respectable minimum RMSEs, with 3.142° and 4.21° for the knee in walking and running, respectively, as well as RMSE errors of

3.8292° and 4.5767° for the hip in running and walking, respectively. Additionally, the Pearson correlation coefficients indicated a correlation ranging from 'strong' to 'very strong' between the predictions and the actual values, with p-values below 0.001 for all developed models.

On the other hand, the analysis of these errors also highlighted a major limitation of this work, namely the difficulty in generalizing the models to a wide range of participants. Indeed, the discrepancies between two participants performing the same task for the same model resulted in RMSE error variation ranges from 6° to 28.8884°. This underscores the complexity of uniformly modeling joint kinematics across a diverse population.

Furthermore, the models' predictions in terms of the percentage of the gait cycle not only allowed for an in-depth biomechanical analysis but also clearly identified the participants and tasks that contribute the most to the increase in average RMSE values per model. These variations are primarily due to the experimental phase and data processing, thereby suggesting areas for improvement in future studies.

In conclusion, this work represents a further step towards the use of a single IMU sensor for analyzing joint kinematics during running and walking, highlighting the importance of the experimental phase and data processing. However, further research will be necessary to improve the generalization of the models and reduce the observed errors, thereby opening up promising prospects for more accessible clinical and sports applications.

# Bibliography

- [1] Siddharth Hans. A protocol for two-dimensional running gait analysis. <https://simplifaster.com/articles/gait-analysis/>, 2024. Accessed: August 18, 2024.
- [2] Kai-Nan An. Kinematic analysis of human movement. *Annals of biomedical engineering*, 12:585–597, 1984.
- [3] Olivier Brùls. Musculoskeletal modelling, part 1, 2024. Lecture presented as part of BIOM0631 - Human Movement Analysis, University of Liège, Belgium.
- [4] Kristen Nicholson. Kinematics and kinetics: Technique and mechanical models. In *Cerebral Palsy*, pages 1339–1353. Springer, 2020.
- [5] Vincent Bonnet, Vladimir Joukov, Dana Kulić, Philippe Fraisse, Nacim Ramdani, and Gentiane Venture. Monitoring of hip and knee joint angles using a single inertial measurement unit during lower limb rehabilitation. *IEEE Sensors journal*, 16(6):1557–1564, 2015.
- [6] Martin Friedrich, Thomas Cermak, and Petra Maderbacher. The Effect of Brochure Use Versus Therapist Teaching on Patients Performing Therapeutic Exercise and on Changes in Impairment Status. *Physical Therapy*, 76(10):1082–1088, 10 1996.
- [7] Agnes WK Lam, Ahmed HajYasien, and Dana Kulic. Improving rehabilitation exercise performance through visual guidance. In *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1735–1738. IEEE, 2014.
- [8] Sumit Majumder, Tapas Mondal, and M. Jamal Deen. A simple, low-cost and efficient gait analyzer for wearable healthcare applications. *IEEE Sensors Journal*, 19(6):2320–2329, 2019.
- [9] Manju Rana and Vikas Mittal. Wearable sensors for real-time kinematics analysis in sports: A review. *IEEE Sensors Journal*, 21(2):1187–1207, 2021.
- [10] BBC Bitesize. Movement analysis in sport - eduqas: Planes and axes of movement, 2024. Accessed: 2024-08-12.
- [11] Thomas K. Uchida, Scott Delp, and David B. Delp. *Biomechanics of movement : the science of sports, robotics, and rehabilitation*. The MIT Press, Cambridge, Massachusetts, 2020.
- [12] Schwartz Cedric. Biom0631-1 human movement analysis, part 2: Kinematics. Lecture presented at the University of Liège, 2024.
- [13] Valentina Camomilla, Aurelio Cappozzo, and Giuseppe Vannozzi. Three-dimensional reconstruction of the human skeleton in motion. *Handbook of human motion*, pages 1–29, 2017.
- [14] Rafael Caldas, Marion Mundt, Wolfgang Potthast, Fernando Buarque de Lima Neto, and Bernd Markert. A systematic review of gait analysis methods based on inertial sensors and adaptive algorithms. *Gait & Posture*, 57:204–210, 2017.

- [15] Daniel L Benoit, Michael Damsgaard, and Michael Skipper Andersen. Surface marker cluster translation, rotation, scaling and deformation: Their contribution to soft tissue artefact and impact on knee joint kinematics. *Journal of biomechanics*, 48(10):2124–2129, 2015.
- [16] Rita Stagni, Silvia Fantozzi, Angelo Cappello, and Alberto Leardini. Quantification of soft tissue artefact in motion analysis by combining 3d fluoroscopy and stereophotogrammetry: a study on two subjects. *Clinical Biomechanics*, 20(3):320–329, 2005.
- [17] Elodie Piche, Marine Guilbot, Frédéric Chorin, Olivier Guerin, Raphaël Zory, and Pauline Gerus. Validity and repeatability of a new inertial measurement unit system for gait analysis on kinematic parameters: Comparison with an optoelectronic system. *Measurement*, 198:111442, 2022.
- [18] Alessandro Filippeschi, Norbert Schmitz, Markus Miezal, Gabriele Bleser, Emanuele Ruffaldi, and Didier Stricker. Survey of motion tracking methods based on inertial sensors: A focus on upper limb human motion. *Sensors*, 17(6):1257, 2017.
- [19] Irvin Hussein Lopez-Nava and Angelica Munoz-Melendez. Wearable inertial sensors for human motion analysis: A review. *IEEE Sensors Journal*, 16(22):7821–7834, 2016.
- [20] Daniel Tik-Pui Fong and Yue-Yan Chan. The use of wearable inertial motion sensors in human lower limb biomechanics studies: A systematic review. *Sensors*, 10(12):11556–11565, 2010.
- [21] Ive Weygers, Manon Kok, Marco Konings, Hans Hallez, Henri De Vroey, and Kurt Claeys. Inertial sensor-based lower limb joint kinematics: A methodological systematic review. *Sensors*, 20(3), 2020.
- [22] Isabelle Poitras, Frédérique Dupuis, Mathieu Biemann, Alexandre Campeau-Lecours, Catherine Mercier, Laurent J Bouyer, and Jean-Sébastien Roy. Validity and reliability of wearable sensors for joint angle estimation: A systematic review. *Sensors*, 19(7):1555, 2019.
- [23] Marion Mundt, Wolf Thomsen, Tom Witter, Arnd Koeppel, Sina David, Franz Bamer, Wolfgang Potthast, and Bernd Markert. Prediction of lower limb joint angles and moments during gait using artificial neural networks. *Medical & biological engineering & computing*, 58:211–225, 2020.
- [24] Marion Mundt, William R Johnson, Wolfgang Potthast, Bernd Markert, Ajmal Mian, and Jacqueline Alderson. A comparison of three neural network approaches for estimating joint angles and moments from inertial measurement units. *Sensors*, 21(13):4535, 2021.
- [25] Marion Mundt, Arnd Koeppel, Sina David, Tom Witter, Franz Bamer, Wolfgang Potthast, and Bernd Markert. Estimation of gait mechanics based on simulated and measured imu data using an artificial neural network. *Frontiers in Bioengineering and Biotechnology*, 8, 2020.
- [26] Eric Rapp, Soyong Shin, Wolf Thomsen, Reed Ferber, and Eni Halilaj. Estimation of kinematics from inertial measurement units using a combined deep learning and optimization framework. *Journal of Biomechanics*, 116:110229, 2021.
- [27] Liangliang Xiang, Alan Wang, Yaodong Gu, Liang Zhao, Vickie Shim, and Justin Fernandez. Recent machine learning progress in lower limb running biomechanics with wearable technology: A systematic review. *Frontiers in Neurorobotics*, 16, 2022.
- [28] Hyerim Lim, Bumjoon Kim, and Sukyung Park. Prediction of lower limb kinetics and kinematics during walking by a single imu on the lower back using machine learning. *Sensors*, 20(1), 2020.

- [29] Myunghyun Lee and Sukyung Park. Estimation of three-dimensional lower limb kinetics data during walking using machine learning from a single imu attached to the sacrum. *Sensors*, 20(21), 2020.
- [30] JooHwan Sung, Sungmin Han, Heesu Park, Hyun-Myung Cho, Soree Hwang, Jong Woong Park, and Inchan Youn. Prediction of lower extremity multi-joint angles during overground walking by using a single imu with a low frequency based on an lstm recurrent neural network. *Sensors*, 22(1), 2022.
- [31] Javier Conte Alcaraz, Sanam Moghaddamnia, and Jürgen Peissig. Efficiency of deep neural networks for joint angle modeling in digital gait assessment. *EURASIP Journal on Advances in Signal Processing*, 2021(1):10, 2021.
- [32] Ting Long, Jereme Outerleys, Ted Yeung, Justin Fernandez, Mary L Boussein, Irene S Davis, Miriam A Bredella, and Thor F Besier. Predicting ankle and knee sagittal kinematics and kinetics using an ankle-mounted inertial sensor. *Computer Methods in Biomechanics and Biomedical Engineering*, 27(9):1057–1070, 2024.
- [33] Daniel Hung Kay Chow, Luc Tremblay, Chor Yin Lam, Adrian Wai Yin Yeung, Wilson Ho Wu Cheng, and Peter Tin Wah Tse. Comparison between accelerometer and gyroscope in predicting level-ground running kinematics by treadmill running kinematics using a single wearable sensor. *Sensors*, 21(14), 2021.
- [34] Daniel Hung-Kay Chow, Zaheen Ahmed Iqbal, Luc Tremblay, Chor-Yin Lam, and Rui-Bin Zhao. Cross-leg prediction of running kinematics across various running conditions and drawing from a minimal data set using a single wearable sensor. *Symmetry*, 14(6), 2022.
- [35] Valentina Camomilla, Elena Bergamini, Silvia Fantozzi, and Giuseppe Vannozzi. Trends supporting the in-field use of wearable inertial sensors for sport performance evaluation: A systematic review. *Sensors*, 18(3), 2018.
- [36] Amin Ahmadi, Francois Destelle, David Monaghan, Kieran Moran, Noel E O’Connor, Luis Unzueta, and Maria Teresa Linaza. Human gait monitoring using body-worn inertial sensors and kinematic modelling. In *2015 IEEE SENSORS*, pages 1–4. IEEE, 11 2015.
- [37] Jay-Shian Tan, Sawitchaya Tippaya, Tara Binnie, Paul Davey, Kathryn Napier, JP Caneiro, Peter Kent, Anne Smith, Peter O’Sullivan, and Amity Campbell. Predicting knee joint kinematics from wearable sensor data in people with knee osteoarthritis and clinical considerations for future machine learning models. *Sensors*, 22(2):446, 2022.
- [38] Rob van der Straaten, Liesbet De Baets, Ilse Jonkers, and Annick Timmermans. Mobile assessment of the lower limb kinematics in healthy persons and in persons with degenerative knee disorders: A systematic review. *Gait & posture*, 59:229–241, 2018.
- [39] Angelo M Sabatini. Quaternion-based extended kalman filter for determining orientation by inertial and magnetic sensing. *IEEE transactions on Biomedical Engineering*, 53(7):1346–1356, 2006.
- [40] WHK De Vries, HEJ Veeger, CTM Baten, and FCT Van Der Helm. Magnetic distortion in motion labs, implications for validating inertial magnetic sensors. *Gait & posture*, 29(4):535–541, 2009.
- [41] Wolfgang Teufl, Markus Miezal, Bertram Taetz, Michael Fröhlich, and Gabriele Bleser. Validity of inertial sensor based 3d joint kinematics of static and dynamic sport and physiotherapy specific movements. *PLoS one*, 14(2):e0213064, 2019.

- [42] Pengfei Gui, Liqiong Tang, and Subhas Mukhopadhyay. Mems based imu for tilting measurement: Comparison of complementary and kalman filter based data fusion. In *2015 IEEE 10th conference on Industrial Electronics and Applications (ICIEA)*, pages 2004–2009. IEEE, 2015.
- [43] Wolfgang Teufl, Markus Miezal, Bertram Taetz, Michael Fröhlich, and Gabriele Bleser. Validity, test-retest reliability and long-term stability of magnetometer free inertial sensor based 3d joint kinematics. *Sensors*, 18(7):1980, 2018.
- [44] Mohamed Boutaayamou. Ambulatory systems for quantitative gait analysis. Lecture, Human Movement Analysis (BIOM0631-1), University of Liège, Belgium., November 2024. Lecture given at the University of Liège, LAM - Motion Lab, Department of Electricity, Electronics, and Computer Science.
- [45] I Arun Faisal, T Waluyo Purboyo, and A Siswo Raharjo Ansori. A review of accelerometer sensor and gyroscope sensor in imu sensors on motion capture. *J. Eng. Appl. Sci*, 15(3):826–829, 2019.
- [46] Erik Grahn. Evaluation of mems accelerometer and gyroscope for orientation tracking n-trunner functionality, 2017.
- [47] Majid Dadafshar. Accelerometer and gyroscopes sensors: operation, sensing, and applications. *Maxim Integrated [online]*, 2014.
- [48] Kristel Çoçoli and Leonardo Badia. A comparative analysis of sensor fusion algorithms for miniature imu measurements. In *2023 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, pages 239–244, 2023.
- [49] Hendrik J Luinge. Inertial sensing of human movement. 2002.
- [50] Ted Yeung, Astrid Cantamessa, Andreas W Kempa-Liehr, Thor Besier, and Julie Choisne. Personalized machine learning approach to estimating knee kinematics using only shank-mounted imu. *IEEE Sensors Journal*, 23(11):12380–12387, 2023.
- [51] Abhishek V Tatachar. Comparative assessment of regression models based on model evaluation metrics. *International Journal of Innovative Technology and Exploring Engineering*, 8(9):853–860, 2021.
- [52] Trevor Hastie, Robert Tibshirani, and Jerome H Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*, volume 2. Springer New York, NY, 2009.
- [53] Saeed Mouloudi, Hadi Rahmanpanah, Colin Burvill, Soheil Gohari, and Helen M. S. Davies. Experimental, regression learner, numerical, and artificial neural network analyses on a complex composite structure subjected to compression loading. *Mechanics of Advanced Materials and Structures*, 29(17):2437–2453, 2022.
- [54] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [55] Amal Mahmoud and Ammar Mohammed. A survey on deep learning for time-series forecasting. *Machine learning and big data analytics paradigms: analysis, applications and challenges*, pages 365–392, 01 2021.
- [56] Aurelio Cappozzo, Angelo Cappello, U Della Croce, and Francesco Pensalfini. Surface-marker cluster design criteria for 3-d bone movement reconstruction. *IEEE Transactions on Biomedical Engineering*, 44(12):1165–1174, 1997.

- [57] Delsys Europe. Trigno avanti sensor. <https://delsyseurope.com/trigno-avanti/>. Accessed on July 1, 2024.
- [58] Cédric Schwartz. Human movement analysis, part 5: Muscle biomechanics. <https://www.uliege.be>, 2023. Course material, University of Liège, Belgium.
- [59] Qualisys. Calibration kit assembly, 2024. Accessed: 2024-08-10.
- [60] Jonathan Sinclair, Paul John Taylor, and Sarah Jane Hobbs. Digital filtering of three-dimensional lower extremity kinematics: An assessment. *Journal of human kinetics*, 39(1):25–36, 2013.
- [61] T.-W. Lu and J.J. O’Connor. Bone position estimation from skin marker co-ordinates using global optimisation with joint constraints. *Journal of Biomechanics*, 32(2):129–134, 1999.
- [62] HAS Motion. Inverse kinematics, 2024. Accessed: August 8, 2024.
- [63] The MathWorks Inc. Statistics and machine learning toolbox, 2024.
- [64] Tong Yu and Hong Zhu. Hyper-parameter optimization: A review of algorithms and applications. *arXiv preprint arXiv:2003.05689*, 2020.
- [65] Stephen J Wright. Numerical optimization, 2006.
- [66] Kamyab Keshtkar. Convolutional neural networks in computer-aided diagnosis of colorectal polyps and cancer: A review. 10 2021.
- [67] MathWorks. Bayesian optimization algorithm. Accessed: 2024-07-02.
- [68] Aryan Jadon, Avinash Patil, and Shruti Jadon. A comprehensive survey of regression-based loss functions for time series forecasting. In *International Conference on Data Management, Analytics & Innovation*, pages 117–147. Springer, 2024.
- [69] Timothy O Hodson. Root mean square error (rmse) or mean absolute error (mae): When to use them or not. *Geoscientific Model Development Discussions*, 2022:1–10, 2022.
- [70] Kejin Hu. Become competent within one day in generating boxplots and violin plots for a novice without prior r experience. *Methods and Protocols*, 3(4), 2020.
- [71] H Hoffmann et al. violin. m-simple violin plot using matlab default kernel density estimation. *INRES (University of Bonn), Katzenburgweg*, 5:53115 (Germany), 2015.
- [72] MathWorks. *Kernel Distribution Documentation*, 2024. Accessed: 2024-08-15.
- [73] Patrick Schober, Christa Boer, and Lothar A Schwarte. Correlation coefficients: appropriate use and interpretation. *Anesthesia & analgesia*, 126(5):1763–1768, 2018.
- [74] JASP Team. JASP (Version 0.19.0.0)[Computer software], 2024.
- [75] Mark A Goss-Sampson. *Statistical Analysis in JASP: A Guide for Students*. JASP Team, Version JASP v0.9.1, October 2018, 2nd edition edition, 2018.
- [76] Tetsuro Miyazaki, Toshihiro Kawase, Takahiro Kanno, Maina Sogabe, Yoshikazu Nakajima, and Kenji Kawashima. Running motion assistance using a soft gait-assistive suit and its experimental validation. *IEEE Access*, PP:1–1, 06 2021.
- [77] Roemer van der Meij and Jan-Mathijs Schoffelen. *Digital Filtering*, pages 1222–1228. Springer New York, New York, NY, 2022.

# Appendix: MATLAB® Codes

## .1 importData.m

```
function [angles, IMU, outputRun_Ankle, outputRun_Knee, outputRun_Hip, ...
        outputWalk_Ankle, outputWalk_Knee, outputWalk_Hip, ...
        inputRun_table, inputRun_matrix, inputWalk_table, inputWalk_matrix, ...
        inputRun80_matrix, output_Run80] = ImportData()

% This function imports angles and IMU data from text files for different
% models (run and walk).

%% Import angles data (OUTPUTS)
path = 'C:\\Users\\flore\\OneDrive\\Documents\\Master 2\\TFE\\QTM_Projet\\Data';
opts = delimitedTextImportOptions("NumVariables", 19);
opts.DataLines = [6, Inf];
opts.Delimiter = "\t";
opts.VariableNames = ["ITEM", "Run100_Ankle", "Run100_Hip", "Run100_Knee", ...
                    "Run120_Ankle", "Run120_Hip", "Run120_Knee", ...
                    "Run80_Ankle", "Run80_Hip", "Run80_Knee", ...
                    "Walk100_Ankle", "Walk100_Hip", "Walk100_Knee", ...
                    "Walk120_Ankle", "Walk120_Hip", "Walk120_Knee", ...
                    "Walk80_Ankle", "Walk80_Hip", "Walk80_Knee"];
opts.VariableTypes = repmat("double", 1, 19);
opts.ExtraColumnsRule = "ignore";
opts.EmptyLineRule = "read";

angles = zeros(120000, 19);
angles = array2table(angles, "VariableNames", opts.VariableNames);
rowIdx = 1;

for participant = 1:20
    participantPath = sprintf('%s\\participant%d\\', path, participant);
    participant_angles = readtable(...
        sprintf('%sparticipant%d_angles.txt', participantPath, participant), ...
        opts);
    save(...
        sprintf('%sparticipant%d_angles.mat', participantPath, participant), ...
        'participant_angles');

    angles(rowIdx:rowIdx+6000-1, :) = participant_angles;
    rowIdx = rowIdx + 6000;
end

angles = removevars(angles, 'ITEM');

% Output for the MODEL1 : Running ankle angles
angles_Run_Ankle = angles(:, {'Run100_Ankle', 'Run120_Ankle'});
outputRun_Ankle = reshape(angles_Run_Ankle, [], 1);
```



```

% Remove participant16 rows
outputRun_Ankle(210001:216000, :) = []; % For Run120
outputRun_Ankle(90001:96000, :) = []; % For Run100

% Output for the MODEL2 : Running knee angles
angles_Run_Knee = angles(:, {'Run100_Knee', 'Run120_Knee'});
outputRun_Knee = reshape(angles_Run_Knee, [], 1);
% Remove participant16 rows
outputRun_Knee(210001:216000, :) = []; % For Run120
outputRun_Knee(90001:96000, :) = []; % For Run100

% Output for the MODEL3 : Running hip angles
angles_Run_Hip = angles(:, {'Run100_Hip', 'Run120_Hip'});
outputRun_Hip = reshape(angles_Run_Hip, [], 1);
% Remove participant16 rows
outputRun_Hip(210001:216000, :) = []; % For Run120
outputRun_Hip(90001:96000, :) = []; % For Run100

% Output for the MODEL4 : Walking ankle angles
angles_Walk_Ankle = angles(:, {'Walk100_Ankle', 'Walk120_Ankle', 'Walk80_Ankle'});
outputWalk_Ankle = reshape(angles_Walk_Ankle, [], 1);

% Output for the MODEL5 : Walking knee angles
angles_Walk_Knee = angles(:, {'Walk100_Knee', 'Walk120_Knee', 'Walk80_Knee'});
outputWalk_Knee = reshape(angles_Walk_Knee, [], 1);

% Output for the MODEL6 : Walking hip angles
angles_Walk_Hip = angles(:, {'Walk100_Hip', 'Walk120_Hip', 'Walk80_Hip'});
outputWalk_Hip = reshape(angles_Walk_Hip, [], 1);

% Output to test the model (Run_80)
output_Run80 = angles(:, {'Run80_Ankle', 'Run80_Knee', 'Run80_Hip'});
%output_Run80 = reshape(output_Run80, [], 1);

clear opts
%% Import IMU data (INPUTS)

IMU = zeros(120000, 73);
IMU = array2table(IMU, "VariableNames", cellstr({ ...
    "ITEM", "Run100_Heel_ACC_X", "Run100_Heel_ACC_Y", ...
    "Run100_Heel_ACC_Z", "Run100_Heel_GYRO_X", "Run100_Heel_GYRO_Y", ...
    "Run100_Heel_GYRO_Z", "Run100_MidFoot_ACC_X", "Run100_MidFoot_ACC_Y", ...
    "Run100_MidFoot_ACC_Z", "Run100_MidFoot_GYRO_X", "Run100_MidFoot_GYRO_Y", ...
    "Run100_MidFoot_GYRO_Z", "Run120_Heel_ACC_X", "Run120_Heel_ACC_Y", ...
    "Run120_Heel_ACC_Z", "Run120_Heel_GYRO_X", "Run120_Heel_GYRO_Y", ...
    "Run120_Heel_GYRO_Z", "Run120_MidFoot_ACC_X", "Run120_MidFoot_ACC_Y", ...
    "Run120_MidFoot_ACC_Z", "Run120_MidFoot_GYRO_X", "Run120_MidFoot_GYRO_Y", ...
    "Run120_MidFoot_GYRO_Z", "Run80_Heel_ACC_X", "Run80_Heel_ACC_Y", ...
    "Run80_Heel_ACC_Z", "Run80_Heel_GYRO_X", "Run80_Heel_GYRO_Y", ...
    "Run80_Heel_GYRO_Z", "Run80_MidFoot_ACC_X", "Run80_MidFoot_ACC_Y", ...
    "Run80_MidFoot_ACC_Z", "Run80_MidFoot_GYRO_X", "Run80_MidFoot_GYRO_Y", ...
    "Run80_MidFoot_GYRO_Z", "Walk100_Heel_ACC_X", "Walk100_Heel_ACC_Y", ...
    "Walk100_Heel_ACC_Z", "Walk100_Heel_GYRO_X", "Walk100_Heel_GYRO_Y", ...
    "Walk100_Heel_GYRO_Z", "Walk100_MidFoot_ACC_X", "Walk100_MidFoot_ACC_Y", ...
    "Walk100_MidFoot_ACC_Z", "Walk100_MidFoot_GYRO_X", "Walk100_MidFoot_GYRO_Y", ...
    "Walk100_MidFoot_GYRO_Z", "Walk120_Heel_ACC_X", "Walk120_Heel_ACC_Y", ...
    "Walk120_Heel_ACC_Z", "Walk120_Heel_GYRO_X", "Walk120_Heel_GYRO_Y", ...
    "Walk120_Heel_GYRO_Z", "Walk120_MidFoot_ACC_X", "Walk120_MidFoot_ACC_Y", ...
    "Walk120_MidFoot_ACC_Z", "Walk120_MidFoot_GYRO_X", "Walk120_MidFoot_GYRO_Y", ...
}));

```

```

"Walk120_MidFoot_GYRO_Z", "Walk80_Heel_ACC_X", "Walk80_Heel_ACC_Y", ...
"Walk80_Heel_ACC_Z", "Walk80_Heel_GYRO_X", "Walk80_Heel_GYRO_Y", ...
"Walk80_Heel_GYRO_Z", "Walk80_MidFoot_ACC_X", "Walk80_MidFoot_ACC_Y", ...
"Walk80_MidFoot_ACC_Z", "Walk80_MidFoot_GYRO_X", "Walk80_MidFoot_GYRO_Y", ...
"Walk80_MidFoot_GYRO_Z"));

for participant = 1:2
    opts = delimitedTextImportOptions("NumVariables", 109);
    opts.DataLines = [6, Inf];
    opts.Delimiter = "\t";
    opts.VariableNames = ["ITEM", ...
        "Run100_DistalTibia_ACC_X", "Run100_DistalTibia_ACC_Y", ...
        "Run100_DistalTibia_ACC_Z", "Run100_DistalTibia_GYRO_X", ...
        "Run100_DistalTibia_GYRO_Y", "Run100_DistalTibia_GYRO_Z", ...
        "Run100_Heel_ACC_X", "Run100_Heel_ACC_Y", "Run100_Heel_ACC_Z", ...
        "Run100_Heel_GYRO_X", "Run100_Heel_GYRO_Y", "Run100_Heel_GYRO_Z", ...
        "Run100_MidFoot_ACC_X", "Run100_MidFoot_ACC_Y", "Run100_MidFoot_ACC_Z", ...
        "Run100_MidFoot_GYRO_X", "Run100_MidFoot_GYRO_Y", ...
        "Run100_MidFoot_GYRO_Z", "Run120_DistalTibia_ACC_X", ...
        "Run120_DistalTibia_ACC_Y", "Run120_DistalTibia_ACC_Z", ...
        "Run120_DistalTibia_GYRO_X", "Run120_DistalTibia_GYRO_Y", ...
        "Run120_DistalTibia_GYRO_Z", ...
        "Run120_Heel_ACC_X", "Run120_Heel_ACC_Y", "Run120_Heel_ACC_Z", ...
        "Run120_Heel_GYRO_X", "Run120_Heel_GYRO_Y", "Run120_Heel_GYRO_Z", ...
        "Run120_MidFoot_ACC_X", "Run120_MidFoot_ACC_Y", "Run120_MidFoot_ACC_Z", ...
        "Run120_MidFoot_GYRO_X", "Run120_MidFoot_GYRO_Y", ...
        "Run120_MidFoot_GYRO_Z", "Run80_DistalTibia_ACC_X", ...
        "Run80_DistalTibia_ACC_Y", "Run80_DistalTibia_ACC_Z", ...
        "Run80_DistalTibia_GYRO_X", "Run80_DistalTibia_GYRO_Y", ...
        "Run80_DistalTibia_GYRO_Z", ...
        "Run80_Heel_ACC_X", "Run80_Heel_ACC_Y", "Run80_Heel_ACC_Z", ...
        "Run80_Heel_GYRO_X", "Run80_Heel_GYRO_Y", "Run80_Heel_GYRO_Z", ...
        "Run80_MidFoot_ACC_X", "Run80_MidFoot_ACC_Y", "Run80_MidFoot_ACC_Z", ...
        "Run80_MidFoot_GYRO_X", "Run80_MidFoot_GYRO_Y", "Run80_MidFoot_GYRO_Z", ...
        "Walk100_DistalTibia_ACC_X", "Walk100_DistalTibia_ACC_Y", ...
        "Walk100_DistalTibia_ACC_Z", "Walk100_DistalTibia_GYRO_X", ...
        "Walk100_DistalTibia_GYRO_Y", "Walk100_DistalTibia_GYRO_Z", ...
        "Walk100_Heel_ACC_X", "Walk100_Heel_ACC_Y", "Walk100_Heel_ACC_Z", ...
        "Walk100_Heel_GYRO_X", "Walk100_Heel_GYRO_Y", "Walk100_Heel_GYRO_Z", ...
        "Walk100_MidFoot_ACC_X", "Walk100_MidFoot_ACC_Y", ...
        "Walk100_MidFoot_ACC_Z", "Walk100_MidFoot_GYRO_X", ...
        "Walk100_MidFoot_GYRO_Y", "Walk100_MidFoot_GYRO_Z", ...
        "Walk120_DistalTibia_ACC_X", "Walk120_DistalTibia_ACC_Y", ...
        "Walk120_DistalTibia_ACC_Z", "Walk120_DistalTibia_GYRO_X", ...
        "Walk120_DistalTibia_GYRO_Y", "Walk120_DistalTibia_GYRO_Z", ...
        "Walk120_Heel_ACC_X", "Walk120_Heel_ACC_Y", "Walk120_Heel_ACC_Z", ...
        "Walk120_Heel_GYRO_X", "Walk120_Heel_GYRO_Y", "Walk120_Heel_GYRO_Z", ...
        "Walk120_MidFoot_ACC_X", "Walk120_MidFoot_ACC_Y", ...
        "Walk120_MidFoot_ACC_Z", "Walk120_MidFoot_GYRO_X", ...
        "Walk120_MidFoot_GYRO_Y", "Walk120_MidFoot_GYRO_Z", ...
        "Walk80_DistalTibia_ACC_X", "Walk80_DistalTibia_ACC_Y", ...
        "Walk80_DistalTibia_ACC_Z", "Walk80_DistalTibia_GYRO_X", ...
        "Walk80_DistalTibia_GYRO_Y", "Walk80_DistalTibia_GYRO_Z", ...
        "Walk80_Heel_ACC_X", "Walk80_Heel_ACC_Y", "Walk80_Heel_ACC_Z", ...
        "Walk80_Heel_GYRO_X", "Walk80_Heel_GYRO_Y", "Walk80_Heel_GYRO_Z", ...
        "Walk80_MidFoot_ACC_X", "Walk80_MidFoot_ACC_Y", "Walk80_MidFoot_ACC_Z", ...
        "Walk80_MidFoot_GYRO_X", "Walk80_MidFoot_GYRO_Y", "Walk80_MidFoot_GYRO_Z"];
    opts.VariableTypes = repmat("double", 1, 109);
    opts.ExtraColumnsRule = "ignore";
    opts.EmptyLineRule = "read";

```

```

participantPath = sprintf('%s\\participant%d\\', path, participant);
participant_IMU = readtable(...
    sprintf('%sparticipant%d_IMU.txt', participantPath, participant), opts);
participant_IMU = removevars(participant_IMU, ...
    cellstr({...
        "Run100_DistalTibia_ACC_X", "Run100_DistalTibia_ACC_Y", ...
        "Run100_DistalTibia_ACC_Z", "Run100_DistalTibia_GYRO_X", ...
        "Run100_DistalTibia_GYRO_Y", "Run100_DistalTibia_GYRO_Z", ...
        "Run120_DistalTibia_ACC_X", "Run120_DistalTibia_ACC_Y", ...
        "Run120_DistalTibia_ACC_Z", "Run120_DistalTibia_GYRO_X", ...
        "Run120_DistalTibia_GYRO_Y", "Run120_DistalTibia_GYRO_Z", ...
        "Run80_DistalTibia_ACC_X", "Run80_DistalTibia_ACC_Y", ...
        "Run80_DistalTibia_ACC_Z", "Run80_DistalTibia_GYRO_X", ...
        "Run80_DistalTibia_GYRO_Y", "Run80_DistalTibia_GYRO_Z", ...
        "Walk100_DistalTibia_ACC_X", "Walk100_DistalTibia_ACC_Y", ...
        "Walk100_DistalTibia_ACC_Z", "Walk100_DistalTibia_GYRO_X", ...
        "Walk100_DistalTibia_GYRO_Y", "Walk100_DistalTibia_GYRO_Z", ...
        "Walk120_DistalTibia_ACC_X", "Walk120_DistalTibia_ACC_Y", ...
        "Walk120_DistalTibia_ACC_Z", "Walk120_DistalTibia_GYRO_X", ...
        "Walk120_DistalTibia_GYRO_Y", "Walk120_DistalTibia_GYRO_Z", ...
        "Walk80_DistalTibia_ACC_X", "Walk80_DistalTibia_ACC_Y", ...
        "Walk80_DistalTibia_ACC_Z", "Walk80_DistalTibia_GYRO_X", ...
        "Walk80_DistalTibia_GYRO_Y", "Walk80_DistalTibia_GYRO_Z"}));
participant_IMU = resample(table2array(participant_IMU), 6000, 60000);
save(sprintf('participant%d_IMU.mat', participantPath, participant), 'participant_IMU')

IMU((participant-1)*6000+1:participant*6000,:) = array2table(participant_IMU);
end

for participant = 3:20
    if participant == 16 % having only Run120, Walk100, Walk120 et Walk80
        opts = delimitedTextImportOptions("NumVariables", 49);
        opts.DataLines = [6, Inf];
        opts.Delimiter = "\t";
        opts.VariableNames = ["ITEM", ...
            "Run120_Heel_ACC_X", "Run120_Heel_ACC_Y", "Run120_Heel_ACC_Z", ...
            "Run120_Heel_GYRO_X", "Run120_Heel_GYRO_Y", "Run120_Heel_GYRO_Z", ...
            "Run120_MidFoot_ACC_X", "Run120_MidFoot_ACC_Y", ...
            "Run120_MidFoot_ACC_Z", "Run120_MidFoot_GYRO_X", ...
            "Run120_MidFoot_GYRO_Y", "Run120_MidFoot_GYRO_Z", ...
            "Walk100_Heel_ACC_X", "Walk100_Heel_ACC_Y", "Walk100_Heel_ACC_Z", ...
            "Walk100_Heel_GYRO_X", "Walk100_Heel_GYRO_Y", ...
            "Walk100_Heel_GYRO_Z", "Walk100_MidFoot_ACC_X", ...
            "Walk100_MidFoot_ACC_Y", "Walk100_MidFoot_ACC_Z", ...
            "Walk100_MidFoot_GYRO_X", "Walk100_MidFoot_GYRO_Y", ...
            "Walk100_MidFoot_GYRO_Z", "Walk120_Heel_ACC_X", ...
            "Walk120_Heel_ACC_Y", "Walk120_Heel_ACC_Z", ...
            "Walk120_Heel_GYRO_X", "Walk120_Heel_GYRO_Y", ...
            "Walk120_Heel_GYRO_Z", "Walk120_MidFoot_ACC_X", ...
            "Walk120_MidFoot_ACC_Y", "Walk120_MidFoot_ACC_Z", ...
            "Walk120_MidFoot_GYRO_X", "Walk120_MidFoot_GYRO_Y", ...
            "Walk120_MidFoot_GYRO_Z", "Walk80_Heel_ACC_X", ...
            "Walk80_Heel_ACC_Y", "Walk80_Heel_ACC_Z", ...
            "Walk80_Heel_GYRO_X", "Walk80_Heel_GYRO_Y", ...
            "Walk80_Heel_GYRO_Z", "Walk80_MidFoot_ACC_X", ...
            "Walk80_MidFoot_ACC_Y", "Walk80_MidFoot_ACC_Z", ...
            "Walk80_MidFoot_GYRO_X", "Walk80_MidFoot_GYRO_Y", ...
            "Walk80_MidFoot_GYRO_Z"];
        opts.VariableTypes = repmat("double", 1, 49);
        opts.ExtraColumnsRule = "ignore";
        opts.EmptyLineRule = "read";
    end
end

```

```

participantPath = sprintf('%s\\participant%d\\', path, participant);
participant_IMU = readtable(...
sprintf("%sparticipant%d_IMU.txt", participantPath, participant), opts);
participant_IMU = removevars(participant_IMU, cellstr({"ITEM", ...
    "Run120_Heel_ACC_X", "Run120_Heel_ACC_Y", "Run120_Heel_ACC_Z", ...
    "Run120_Heel_GYRO_X", "Run120_Heel_GYRO_Y", ...
    "Run120_Heel_GYRO_Z", "Run120_MidFoot_ACC_X", ...
    "Run120_MidFoot_ACC_Y", "Run120_MidFoot_ACC_Z", ...
    "Run120_MidFoot_GYRO_X", "Run120_MidFoot_GYRO_Y", ...
    "Run120_MidFoot_GYRO_Z"})));

participant_IMU = resample(table2array(participant_IMU), 6000, 60000);
save(...
    sprintf('%sparticipant%d_IMU.mat', participantPath, participant), ...
    'participant_IMU')
IMU((participant-1)*6000+1:participant*6000,38:73) = ...
    array2table(participant_IMU);
continue;
end

opts = delimitedTextImportOptions("NumVariables", 73);
opts.DataLines = [6, Inf];
opts.Delimiter = "\t";
opts.VariableNames = ["ITEM", "Run100_Heel_ACC_X", "Run100_Heel_ACC_Y", ...
    "Run100_Heel_ACC_Z", "Run100_Heel_GYRO_X", "Run100_Heel_GYRO_Y", ...
    "Run100_Heel_GYRO_Z", "Run100_MidFoot_ACC_X", "Run100_MidFoot_ACC_Y", ...
    "Run100_MidFoot_ACC_Z", "Run100_MidFoot_GYRO_X", "Run100_MidFoot_GYRO_Y", ...
    "Run100_MidFoot_GYRO_Z", "Run120_Heel_ACC_X", "Run120_Heel_ACC_Y", ...
    "Run120_Heel_ACC_Z", "Run120_Heel_GYRO_X", "Run120_Heel_GYRO_Y", ...
    "Run120_Heel_GYRO_Z", "Run120_MidFoot_ACC_X", "Run120_MidFoot_ACC_Y", ...
    "Run120_MidFoot_ACC_Z", "Run120_MidFoot_GYRO_X", "Run120_MidFoot_GYRO_Y", ...
    "Run120_MidFoot_GYRO_Z", "Run80_Heel_ACC_X", "Run80_Heel_ACC_Y", ...
    "Run80_Heel_ACC_Z", "Run80_Heel_GYRO_X", "Run80_Heel_GYRO_Y", ...
    "Run80_Heel_GYRO_Z", "Run80_MidFoot_ACC_X", "Run80_MidFoot_ACC_Y", ...
    "Run80_MidFoot_ACC_Z", "Run80_MidFoot_GYRO_X", "Run80_MidFoot_GYRO_Y", ...
    "Run80_MidFoot_GYRO_Z", "Walk100_Heel_ACC_X", "Walk100_Heel_ACC_Y", ...
    "Walk100_Heel_ACC_Z", "Walk100_Heel_GYRO_X", "Walk100_Heel_GYRO_Y", ...
    "Walk100_Heel_GYRO_Z", "Walk100_MidFoot_ACC_X", "Walk100_MidFoot_ACC_Y", ...
    "Walk100_MidFoot_ACC_Z", "Walk100_MidFoot_GYRO_X", "Walk100_MidFoot_GYRO_Y", ...
    "Walk100_MidFoot_GYRO_Z", "Walk120_Heel_ACC_X", "Walk120_Heel_ACC_Y", ...
    "Walk120_Heel_ACC_Z", "Walk120_Heel_GYRO_X", "Walk120_Heel_GYRO_Y", ...
    "Walk120_Heel_GYRO_Z", "Walk120_MidFoot_ACC_X", "Walk120_MidFoot_ACC_Y", ...
    "Walk120_MidFoot_ACC_Z", "Walk120_MidFoot_GYRO_X", "Walk120_MidFoot_GYRO_Y", ...
    "Walk120_MidFoot_GYRO_Z", "Walk80_Heel_ACC_X", "Walk80_Heel_ACC_Y", ...
    "Walk80_Heel_ACC_Z", "Walk80_Heel_GYRO_X", "Walk80_Heel_GYRO_Y", ...
    "Walk80_Heel_GYRO_Z", "Walk80_MidFoot_ACC_X", "Walk80_MidFoot_ACC_Y", ...
    "Walk80_MidFoot_ACC_Z", "Walk80_MidFoot_GYRO_X", ...
    "Walk80_MidFoot_GYRO_Y", "Walk80_MidFoot_GYRO_Z"];
opts.VariableTypes = repmat("double", 1, 73);
opts.ExtraColumnsRule = "ignore";
opts.EmptyLineRule = "read";

participantPath = sprintf('%s\\participant%d\\', path, participant);
participant_IMU = readtable(...
    sprintf("%sparticipant%d_IMU.txt", participantPath, participant), opts);
participant_IMU = resample(table2array(participant_IMU), 6000, 60000);
save(sprintf('%sparticipant%d_IMU.mat', participantPath, participant), ...
    'participant_IMU')

IMU((participant-1)*6000+1:participant*6000,:) = array2table(participant_IMU);
end

```

```

IMU = removevars(IMU, 'ITEM');

%% Process IMU running data

% Input for the MODEL 1,2,3 (Running)
inputRun = array2table(zeros(360000, 12), 'VariableNames', ...
    {'Run_Heel_ACC_X', 'Run_Heel_ACC_Y', 'Run_Heel_ACC_Z', ...
    'Run_Heel_GYRO_X', 'Run_Heel_GYRO_Y', 'Run_Heel_GYRO_Z', ...
    'Run_MidFoot_ACC_X', 'Run_MidFoot_ACC_Y', 'Run_MidFoot_ACC_Z', ...
    'Run_MidFoot_GYRO_X', 'Run_MidFoot_GYRO_Y', 'Run_MidFoot_GYRO_Z'});

rowIdx = 1;
for i = 1:3
    IMU_Run = IMU(:, (i-1)*12+1:i*12);
    IMU_Run = reshape(IMU_Run, [], 12);
    inputRun(rowIdx:rowIdx+119999, :) = array2table(IMU_Run);
    rowIdx = rowIdx + 120000;
end

% Remove participant16 rows
inputRun(330001:336000, :) = []; % For Run80
inputRun(210001:216000, :) = []; % For Run120
inputRun(90001:96000, :) = []; % For Run100

inputRun80_table = inputRun(228001:342000,:);
inputRun80_matrix = inputRun80_table{:, :};
inputRun_table = inputRun(1:228000,:);
inputRun_matrix = inputRun_table{:, :};

% Input for the MODEL 4,5,6 (Walking)
inputWalk_table = array2table(zeros(360000, 12), 'VariableNames', ...
    {'Walk_Heel_ACC_X', 'Walk_Heel_ACC_Y', 'Walk_Heel_ACC_Z', ...
    'Walk_Heel_GYRO_X', 'Walk_Heel_GYRO_Y', 'Walk_Heel_GYRO_Z', ...
    'Walk_MidFoot_ACC_X', 'Walk_MidFoot_ACC_Y', 'Walk_MidFoot_ACC_Z', ...
    'Walk_MidFoot_GYRO_X', 'Walk_MidFoot_GYRO_Y', 'Walk_MidFoot_GYRO_Z'});

rowIdx = 1;
for i = 4:6
    IMU_Walk = IMU(:, (i-1)*12+1:i*12);
    IMU_Walk = reshape(IMU_Walk, [], 12);
    inputWalk_table(rowIdx:rowIdx+119999, :) = array2table(IMU_Walk);
    rowIdx = rowIdx + 120000;
end

inputWalk_matrix = inputWalk_table{:, :};

end

```

## .2 main.m

```

function [angles, IMU, inputRun_table, inputWalk_table, ...
    resultsModel1, resultsModel2, resultsModel3, ...
    resultsModel4, resultsModel5, resultsModel6] = main()

% Main function to create the neural networks models

```

```

%% Data importation

[angles, IMU, ...
 outputRun_Ankle, outputRun_Knee, outputRun_Hip, ...
 outputWalk_Ankle, outputWalk_Knee, outputWalk_Hip, ...
 inputRun_table, inputRun_matrix, ...
 inputWalk_table, inputWalk_matrix, ...
 inputRun80_matrix, output_Run80, ...
 ] = ImportData();

%% Normalization of inputs

normalizedInputRun_matrix = (inputRun_matrix - mean(inputRun_matrix)) ...
    ./ std(inputRun_matrix);
normalizedInputWalk_matrix = (inputWalk_matrix - mean(inputWalk_matrix)) ...
    ./ std(inputWalk_matrix);
%normalizedInputRun80_matrix = (inputRun80_matrix - mean(inputRun80_matrix)) ...
%
%    ./ std(inputRun80_matrix);

%% Creation and bayesian optimization of Neural Networks

numSets = 15;
secPerParticipant = 2;
%numParticipants = 38; % Total number of participants for running tasks
numParticipants = 60; % Total number of participants for walking tasks

% MODEL1: Run -> Ankle
resultsModel1 = cell(1, 2);
[resultsRunAnkleHeel] = OptimizedNeuralNetworkWithSets(...
    normalizedInputRun_matrix(:, 1:6), ...
    outputRun_Ankle, ...
    numSets, secPerParticipant, numParticipants);
resultsModel1{1,1} = resultsRunAnkleHeel;
save("resultsModel1.mat", "resultsModel1");

[resultsRunAnkleMidFoot] = OptimizedNeuralNetworkWithSets(...
    normalizedInputRun_matrix(:, 7:12), ...
    outputRun_Ankle, ...
    numSets, secPerParticipant, numParticipants);
resultsModel1{1,2} = resultsRunAnkleMidFoot;
save("resultsModel1.mat", "resultsModel1");

% MODEL2: Run -> Knee
resultsModel2 = cell(1, 2);
[resultsRunKneeHeel] = OptimizedNeuralNetworkWithSets(...
    normalizedInputRun_matrix(:, 1:6), ...
    outputRun_Knee, ...
    numSets, secPerParticipant, numParticipants);
resultsModel2{1,1} = resultsRunKneeHeel;
save("resultsModel2.mat", "resultsModel2");

[resultsRunKneeMidFoot] = OptimizedNeuralNetworkWithSets(...
    normalizedInputRun_matrix(:, 7:12), ...
    outputRun_Knee, ...
    numSets, secPerParticipant, numParticipants);
resultsModel2{1,2} = resultsRunKneeMidFoot;
save("resultsModel2.mat", "resultsModel2");

% MODEL3: Run -> Hip (Heel)

```

```
resultsModel3 = cell(1, 2);
[resultsRunHipHeel] = OptimizedNeuralNetworkWithSets(...
    normalizedInputRun_matrix(:, 1:6), ...
    outputRun_Hip, ...
    numSets, secPerParticipant, numParticipants);
resultsModel3{1,1} = resultsRunHipHeel;
save("resultsModel3.mat", "resultsModel3");

[resultsRunHipMidFoot] = OptimizedNeuralNetworkWithSets(...
    normalizedInputRun_matrix(:, 7:12), ...
    outputRun_Hip, ...
    numSets, secPerParticipant, numParticipants);
resultsModel3{1,2} = resultsRunHipMidFoot;
save("resultsModel3.mat", "resultsModel3");

% MODEL4: Walk -> Ankle
resultsModel4 = cell(1, 2);
[resultsWalkAnkleHeel] = OptimizedNeuralNetworkWithSets(...
    normalizedInputWalk_matrix(:, 1:6), ...
    outputWalk_Ankle, ...
    numSets, secPerParticipant, numParticipants);
resultsModel4{1,1} = resultsWalkAnkleHeel;
save("resultsModel4.mat", "resultsModel4");

[resultsWalkAnkleMidFoot] = OptimizedNeuralNetworkWithSets(...
    normalizedInputWalk_matrix(:, 7:12), ...
    outputWalk_Ankle, ...
    numSets, secPerParticipant, numParticipants);
resultsModel4{1,2} = resultsWalkAnkleMidFoot;
save("resultsModel4.mat", "resultsModel4");

% MODEL5: Walk -> Knee
resultsModel5 = cell(1, 2);
[resultsWalkKneeHeel] = OptimizedNeuralNetworkWithSets(...
    normalizedInputWalk_matrix(:, 1:6), ...
    outputWalk_Knee, ...
    numSets, secPerParticipant, numParticipants);
resultsModel5{1,1} = resultsWalkKneeHeel;
save("resultsModel5.mat", "resultsModel5");

[resultsWalkKneeMidFoot] = OptimizedNeuralNetworkWithSets(...
    normalizedInputWalk_matrix(:, 7:12), ...
    outputWalk_Knee, ...
    numSets, secPerParticipant, numParticipants);
resultsModel5{1,2} = resultsWalkKneeMidFoot;
save("resultsModel5.mat", "resultsModel5");

% MODEL6: Walk -> Hip
resultsModel6 = cell(1, 2);
[resultsWalkHipHeel] = OptimizedNeuralNetworkWithSets(...
    normalizedInputWalk_matrix(:, 1:6), ...
    outputWalk_Hip, ...
    numSets, secPerParticipant, numParticipants);
resultsModel6{1,1} = resultsWalkHipHeel;
save("resultsModel6.mat", "resultsModel6");

[resultsWalkHipMidFoot] = OptimizedNeuralNetworkWithSets(...
    normalizedInputWalk_matrix(:, 7:12), ...
    outputWalk_Hip, ...
```

```

    numSets, secPerParticipant, numParticipants);
resultsModel6{1,2} = resultsWalkHipMidFoot;
save("resultsModel6.mat","resultsModel6");

```

```
end
```

### .3 optimizeNeuralNetworksWithSets.m

```

function [results] = OptimizedNeuralNetworkWithSets(X, Y, numSets, ...
    secPerParticipant, numParticipants)
% This function performs Bayesian optimization to minimize the cross-validation
% loss of the fitrnet function by optimizing its hyperparameters.

%% Divide data into 'numSets' to speed up the process

[Xsets, Ysets] = createSets(X, Y, numSets, secPerParticipant, numParticipants);
numObservations = size(Xsets{1,1}, 1);

%% Define the hyperparameters to optimize
% (similar to: https://nl.mathworks.com/help/stats/hyperparameter-
% optimization-in-regression-learner-app.html)

rng default

numLayers = optimizableVariable('NumLayers', [1, 3], 'Type', 'integer');
firstLayerSize = optimizableVariable('Layer_1_Size', [1, 300], 'Type', ...
    'integer', 'Transform', 'log');
secondLayerSize = optimizableVariable('Layer_2_Size', [1, 300], 'Type', ...
    'integer', 'Transform', 'log');
thirdLayerSize = optimizableVariable('Layer_3_Size', [1, 300], 'Type', ...
    'integer', 'Transform', 'log');
activation = optimizableVariable('Activations', {'reLU', 'tanh', ...
    'none', 'sigmoid'}, 'Type', 'categorical');
regularization = optimizableVariable('Regularization', ...
    [0.00001/numObservations, 100000/numObservations], 'Transform', 'log');

vars = [numLayers, firstLayerSize, secondLayerSize, ...
    thirdLayerSize, activation, regularization];

%% Optimize hyperparameters on each set

results = cell(numSets, 1);

for i = 1:numSets
    X_set = Xsets{i, 1};
    Y_set = Ysets{i, 1};

    numObservations = size(X_set, 1);
    groups = zeros(numObservations, 1);
    rowsPerParticipant = 200 * secPerParticipant;
    for j = 1:numParticipants
        startIdx = (j-1) * rowsPerParticipant + 1;
        endIdx = min(j * rowsPerParticipant, numObservations);
        if startIdx <= numObservations
            groups(startIdx:endIdx) = j;
        end
    end
end

```



```

c = cvpartition(groups, "Kfold", 5);

fun = @(x)kfoldLoss(fitrnet(X_set, Y_set, ...
    'CVPartition', c, ...
    'LayerSizes', getCorrectLayerSizes(x.NumLayers, x.Layer_1_Size, ...
        x.Layer_2_Size, x.Layer_3_Size), ...
    'Activations', char(x.Activations), ...
    'Lambda', x.Regularization, ...
    'IterationLimit', 1000));

results{i} = bayesopt(fun, vars, ....
    'Verbose', 0, ...
    'AcquisitionFunctionName', 'expected-improvement-plus', ...
    'UseParallel', false);
end

%% Visualization and averaging of the 15 sets of obtained hyperparameters

numSets = length(results);
hyperparamTable = table('Size', [numSets, 6], ...
    'VariableTypes', {'double', 'double', 'double', 'double', ...
        'categorical', 'double'}, ...
    'VariableNames', {'NumLayers', 'Layer_1_Size', 'Layer_2_Size', ...
        'Layer_3_Size', 'Activations', 'Regularization'});

for i = 1:9
    bestParams = results{i}.XAtMinObjective;
    hyperparamTable.NumLayers(i) = bestParams.NumLayers;
    hyperparamTable.Layer_1_Size(i) = bestParams.Layer_1_Size;
    hyperparamTable.Layer_2_Size(i) = bestParams.Layer_2_Size;
    hyperparamTable.Layer_3_Size(i) = bestParams.Layer_3_Size;
    hyperparamTable.Activations(i) = bestParams.Activations;
    hyperparamTable.Regularization(i) = bestParams.Regularization;
end

disp('Hyperparameters for each set:');
disp(hyperparamTable);

meanHyperparams = varfun(@mean, hyperparamTable, 'InputVariables', ...
    {'NumLayers', 'Layer_1_Size', 'Layer_2_Size', ...
        'Layer_3_Size', 'Regularization'});
meanActivation = mode(hyperparamTable.Activations);

disp('Mean Hyperparameters:');
disp(meanHyperparams);
disp(['Mean Activation Function: ', char(meanActivation)]);

%% Create and train the model with the best hyperparameters found

rng("default")

secPerParticipant = 30;
rowsPerParticipant = 200 * secPerParticipant;
indices = reshape(1:(rowsPerParticipant * numParticipants), ...
    rowsPerParticipant, numParticipants);

XTrain = [];
YTrain = [];
XTest = [];
YTest = [];

for j = 1:numParticipants

```

```

    participantIndices = indices(:, j);

    trainIndices=participantIndices(1:round(0.8*length(participantIndices)));
    testIndices=participantIndices(round(0.8*length(participantIndices))+1:end);

    XTrain = [XTrain; X(trainIndices, :)];
    YTrain = [YTrain; Y(trainIndices, :)];
    XTest = [XTest; X(testIndices, :)];
    YTest = [YTest; Y(testIndices, :)];
end

model = fitrnet(XTrain, YTrain, ...
    'LayerSizes', [round(meanHyperparams.mean_Layer_1_Size), ...
        round(meanHyperparams.mean_Layer_2_Size)] , ...
    'Activations', char(meanActivation), ...
    'Lambda', meanHyperparams.mean_Regularization, ...
    'IterationLimit', 1000);

predictions = struct('Participant', {}, 'Y_Test', {}, 'Y_Pred', {}, ...
    'RMSE', {}, 'MAE' , {});

for i = 1:numParticipants
    participant = i;
    participantTestIdx = indices(round(0.8 * length(indices(:, i))) + 1:end, i);

    X_test_participant = X(participantTestIdx, :);
    Y_test_participant = Y(participantTestIdx, :);

    Y_pred_participant = predict(model, X_test_participant);

    RMSE = sqrt(mean((Y_pred_participant - Y_test_participant).^2));
    MAE = mean(abs(Y_pred_participant - Y_test_participant));

    predictions(i).Participant = participant;
    predictions(i).Y_Test = Y_test_participant;
    predictions(i).Y_Pred = Y_pred_participant;
    predictions(i).RMSE = RMSE;
    predictions(i).MAE = MAE;

end

results = struct('Model', model, 'PredictionsForAllParticipants', predictions);
end

function layerSizes = getCorrectLayerSizes(numLayers, layer1, layer2, layer3)
    % Function to handle the size of layers based on the optimal number of layers
    % returned by 'numLayers'
    switch numLayers
        case 1
            layerSizes = [layer1];
        case 2
            layerSizes = [layer1 layer2];
        case 3
            layerSizes = [layer1 layer2 layer3];
    end
end
end

```

## .4 createSets.m

```
function [XsegmentedMatrices, YsegmentedMatrices] = createSets(X, Y, ...
    numSets, secPerParticipant, numParticipants)
    % This function divides the input matrices (X and Y) into 'numSets'
    % matrices, each containing 'secPerParticipant' seconds of data from each
    % participant.

    numRowsPerParticipant = 200 * secPerParticipant;
    setRows = numParticipants * numRowsPerParticipant;

    %% Segmentation of X into 'numSets' matrices
    XsegmentedMatrices = cell(numSets, 1);

    for i = 1:numSets
        matrix = zeros(setRows, size(X, 2));
        setStartIdx = (i-1) * numRowsPerParticipant + 1;
        setEndIdx = i * numRowsPerParticipant;

        for j = 1:numParticipants
            participantStartIdx = (j-1) * 6000 + setStartIdx;
            participantEndIdx = (j-1) * 6000 + setEndIdx;

            matrix((j-1)*numRowsPerParticipant+1 : j*numRowsPerParticipant, :) ...
                = X(participantStartIdx:participantEndIdx, :);
        end
        XsegmentedMatrices{i} = matrix;
    end

    %% Segmentation of Y into 'numSets' matrices
    YsegmentedMatrices = cell(numSets, 1);

    for i = 1:numSets

        matrix = zeros(setRows, size(Y, 2));
        setStartIdx = (i-1) * numRowsPerParticipant + 1;
        setEndIdx = i * numRowsPerParticipant;

        for j = 1:numParticipants
            participantStartIdx = (j-1) * 6000 + setStartIdx;
            participantEndIdx = (j-1) * 6000 + setEndIdx;

            matrix((j-1)*numRowsPerParticipant+1 : j*numRowsPerParticipant, :) ...
                = Y(participantStartIdx:participantEndIdx, :);
        end
        YsegmentedMatrices{i} = matrix;
    end
end
```

## .5 visualizeModelPerformances.m

```
%% FIGURE: Violin plot for RMSE, MAE, and nRMSE

% Parameters to adapt
modelNb = 3;
model = resultsModel3;
```

```

numParticipants = 38; % Models for running tasks
%numParticipants = 60; % Models for walking tasks

IMULocations = {'Heel', 'Mid foot'};

% Extract RMSE and MAE for all participants by model
RMSE_values = zeros(numParticipants, 2);
MAE_values = zeros(numParticipants, 2);
for j = 1:numParticipants
    RMSE_values(j, 1) = model{1,1}.PredictionsForAllParticipants(j).RMSE;
    RMSE_values(j, 2) = model{1,2}.PredictionsForAllParticipants(j).RMSE;

    MAE_values(j, 1) = model{1,1}.PredictionsForAllParticipants(j).MAE;
    MAE_values(j, 2) = model{1,2}.PredictionsForAllParticipants(j).MAE;
end

% Extract nRMSE for all participants by model
nRMSE_values = zeros(numParticipants, 2);
for j = 1:numParticipants
    Y_Test_Heel = model{1,1}.PredictionsForAllParticipants(j).Y_Test;
    Y_Test_MidFoot = model{1,2}.PredictionsForAllParticipants(j).Y_Test;

    amplitude_Heel = max(Y_Test_Heel) - min(Y_Test_Heel);
    amplitude_MidFoot = max(Y_Test_MidFoot) - min(Y_Test_MidFoot);

    nRMSE_values(j, 1) = (RMSE_values(j, 1) / amplitude_Heel) * 100;
    nRMSE_values(j, 2) = (RMSE_values(j, 2) / amplitude_MidFoot) * 100;
end

% Plot RMSE
figure;
violin(RMSE_values, 'facecolor',[1 1 0;0 1 0;.3 .3 .3;0 0.3 0.1]);
title(sprintf('RMSE for different IMU locations for Model %d', modelNb));
ylabel('RMSE [Degrees]');
xticks(1:length(IMULocations));
xticklabels(IMULocations);

% Plot nRMSE
figure;
violin(nRMSE_values, 'facecolor',[1 1 0;0 1 0;.3 .3 .3;0 0.3 0.1]);
title(sprintf('nRMSE for different IMU locations for Model %d', modelNb));
ylabel('nRMSE [%]');
xticks(1:length(IMULocations));
xticklabels(IMULocations);

% Plot MAE
figure;
violin(MAE_values, 'facecolor',[1 1 0;0 1 0;.3 .3 .3;0 0.3 0.1]);
title(sprintf('MAE for different IMU locations for Model %d', modelNb));
ylabel('MAE [Degrees]');
xticks(1:length(IMULocations));
xticklabels(IMULocations);

%% FIGURE: Predicted and true angle cycles over time for each model

% Parameters to adapt
numModel = 3; % Model number (1, 2, 3, 4, 5, or 6)
model = resultsModel3; % Model
numParticipants = 38; % 38 for running, 60 for walking
realColor = [0.0, 0.0, 0.0]; % Black color
predColors = {[0.8500, 0.3250, 0.0980], [0.3010, 0.7450, 0.9330], ...
    [0.4940, 0.1840, 0.5560]};

```

```

samplingRate = 200;
IMULocations = {'Heel', 'Mid foot'};
numNetworks = 2; % Number of sub-models

for k = 1:numParticipants
    figure;
    hold on
    for i = 1:numNetworks
        predictions = model{1,i}.PredictionsForAllParticipants;

        YTest = predictions(k).Y_Test;
        YPred = predictions(k).Y_Pred;

        participantIdx = mod(k-1, 19) + 1;

        if (numModel == 1 || numModel == 2 || numModel == 3)
            if k <= 15
                cyclesTimesPerParticipant=cycleTimes{participantIdx,1}.Run100;
                task = "Run100";
            elseif k <= 19
                cyclesTimesPerParticipant=cycleTimes{participantIdx+1,1}.Run100;
                task = "Run100";
            elseif k > 19
                if participantIdx <= 15
                    cyclesTimesPerParticipant = cycleTimes{participantIdx,...
                        1}.Run120;
                    task = "Run120";
                else
                    cyclesTimesPerParticipant = cycleTimes{participantIdx+1,...
                        1}.Run120;
                    task = "Run120";
                end
            end
        elseif (numModel == 4 || numModel == 5 || numModel == 6)
            if k <= 20
                cyclesTimesPerParticipant=cycleTimes{participantIdx, 1}.Walk100;
                task = "Walk100";
            elseif k <= 40
                cyclesTimesPerParticipant=cycleTimes{participantIdx, 1}.Walk120;
                task = "Walk120";
            else
                cyclesTimesPerParticipant=cycleTimes{participantIdx, 1}.Walk80;
                task = "Walk80";
            end
        end

        % Convert times to indices
        cyclesIndices = round(cyclesTimesPerParticipant * samplingRate);
        cyclesIndices=cyclesIndices(~isnan(cyclesIndices) & cyclesIndices>0);
        adjustedIndices = cyclesIndices(cyclesIndices >= 4800 & ...
            cyclesIndices <= 6000) - 4800 + 1;

        [trueCycleCells, predictedCycleCells, ...
            meanTrueCycle, stdTrueCycle, ...
            meanPredictedCycle, stdPredictedCycle] = cyclesDetection(...
                adjustedIndices, YTest, YPred);

        timeVector = linspace(0, 100, length(meanTrueCycle));

        % Real cycle

```

```

    if i == 1
        plot(timeVector, meanTrueCycle, 'LineWidth', 2, ...
            'Color', realColor,...
            'DisplayName', 'True mean cycle');
        fill([timeVector, fliplr(timeVector)], ...
            [meanTrueCycle + stdTrueCycle, ...
            fliplr(meanTrueCycle - stdTrueCycle)], ...
            realColor, 'FaceAlpha', 0.3, 'EdgeColor', ...
            'none', 'HandleVisibility', 'off');
    end

    % Predicted cycle
    plot(timeVector, meanPredictedCycle, 'LineWidth', 2, ...
        'Color', predColors{i},...
        'DisplayName', sprintf('Predicted (%s)', IMULocations{i}));
    fill([timeVector, fliplr(timeVector)], ...
        [meanPredictedCycle + stdPredictedCycle, ...
        fliplr(meanPredictedCycle - stdPredictedCycle)], ...
        predColors{i}, 'FaceAlpha', 0.3, 'EdgeColor', ...
        'none', 'HandleVisibility', 'off');

    end

    xlabel('Cycle (%)');
    ylabel('Hip flexion angle (degrees)'); % To adapt
    title(sprintf(...
        'Predicted and true mean cycles with standard ...
        deviation\nshadings for participant %d (%s)', ...
        participantIdx, task));
    legend();
    hold off;
end

%% Conversion in CSV format for JASP

% Parameters to adapt
numParticipants = 38;
numNetworks = 2;
IMULocations = {'Heel', 'Mid foot'};

participantList = [];
YTestList = [];
YPredModel1List = [];
YPredModel2List = [];

for k = 1:numParticipants
    tempYPredModel1 = [];
    tempYPredModel2 = [];

    for i = 1:numNetworks
        predictions = resultsModel{1,i}.PredictionsForAllParticipants;

        YTest = predictions(k).Y_Test;
        YPred = predictions(k).Y_Pred;

        if i == 1
            tempYPredModel1 = YPred;
        end

        if i == 2
            tempYPredModel2 = YPred;
        end
    end
end

```

```

end

nDataPoints = length(YTest);
participantList = [participantList; repmat(k,nDataPoints,1)];
YTestList = [YTestList;YTest];
YPredModel1List = [YPredModel1List; tempYPredModel1];
YPredModel2List = [YPredModel2List; tempYPredModel2];
end

dataTable_Model = table(participantList, YTestList, ...
    YPredModel1List, YPredModel2List, ...
    'VariableNames', {'Participant', 'YTest', 'YPred_ModelHeel',...
    'YPred_ModelMidFoot'});

writetable(dataTable_Model, 'model_predictions_full.csv');

```

## .6 cyclesDetection.m

```

function [trueCycles, predictedCycles, meanTrueCycle, ...
    stdTrueCycle, meanPredictedCycle, stdPredictedCycle] = cyclesDetection(...
    Idx, YTest, YPred)

numCycles = length(Idx) - 1;
trueCycles = cell(numCycles, 1);
predictedCycles = cell(numCycles, 1);

for i = 1:numCycles
    if i < length(Idx)
        startIdx = max(1, round(Idx(i)));
        endIdx = round(Idx(i+1)) - 1;

        trueCycles{i} = YTest(startIdx:endIdx);
        predictedCycles{i} = YPred(startIdx:endIdx);
    end
end

% Criteria for excluding cycles if too small
minCycleLengthThreshold = 0.5;

cyclesToRemove = true;
while ~isempty(cyclesToRemove) && numCycles > 2
    cycleLengths = cellfun(@length, trueCycles);
    meanCycleLength = mean(cycleLengths);

    cyclesToRemove = find(cycleLengths < ...
        minCycleLengthThreshold * meanCycleLength);

    if ~isempty(cyclesToRemove)
        trueCycles(cyclesToRemove) = [];
        predictedCycles(cyclesToRemove) = [];
        numCycles = numCycles - length(cyclesToRemove);
    end
end

%Conversion into a matrix to compute the average
minTrueCycleLength = min(cellfun(@length, trueCycles));
trueCycleMatrix = cell2mat(cellfun(@(x) x(1:minTrueCycleLength), ...

```

```
    trueCycles, 'UniformOutput', false)');

minPredictedCycleLength = min(cellfun(@length, predictedCycles));
predictedCycleMatrix = cell2mat(cellfun(@(x) x(1:minPredictedCycleLength), ...
    predictedCycles, 'UniformOutput', false)');

meanTrueCycle = mean(trueCycleMatrix, 2);
meanPredictedCycle = mean(predictedCycleMatrix, 2);

stdTrueCycle = std(trueCycleMatrix, 0, 2);
stdPredictedCycle = std(predictedCycleMatrix, 0, 2);

meanTrueCycle = meanTrueCycle';
meanPredictedCycle = meanPredictedCycle';

stdTrueCycle = stdTrueCycle';
stdPredictedCycle = stdPredictedCycle';
end
```