Polytechnic of Turin

Department of Control and Computer Engineering (DAUIN)
Master's course in Mechatronic Engeeniring

Master's degree Thesis

Kinematic analysis of the gait for the man-machine interface of a future lower-limb exoskeleton

Candidates:
Menegazzi Luca
Floris Federico

Thesis advisor:
Morisio Maurizio

Research supervisors:
Agostini Valentina
Ghislieri Marco
Menga Giuseppe

Accademic Year 2020/2021

Contents

Abstract

Robotics is increasingly being used in the medicine to solve a wide range of problems in a multitude of different fields. Robots are becoming metallic allies for the benefit of the vulnerable, and apparently their place in the healthcare industry is increasing every day. Some of the fields in which these new technological tools have been incorporated are surgery, telemedicine, sterilization, invasive medicine, and rehabilitation.

In the last field, exoskeletons are used as orthoses to assist or rehabilitate individuals with locomotor impairments. One type are lower-limb exoskeletons, which are used to rehabilitate gait in patients suffering from stroke, accidents, or spinal injuries. These must help the wearer walk, supporting its posture maintaining balance and guaranteeing patient compliance.

Unfortunately, no exoskeletons that combine these two purposes perfectly have been designed yet. Some impose physiological movements on the joints in a passive manner, without taking the patient's will into account (Lokomat by Hokoma); some focus only on maintaining balance while walking (EKSO GT by Ekso Bionics); others are synchronized with the wearer's desired movements through electromyography (EMG) signals (HAL by Cyberdyne).

Our thesis focuses on mapping the kinematics on gait and on correlating EMG signals to lower limb motion, for the future development of a man-machine interface in a haptic exoskeleton for postural rehabilitation, which simultaneously offers compliance to the patient for voluntary control of multiple joints and balance.

Two experimental datasets containing joint patterns and associated EMG were used, one obtained during gait on a treadmill and the other on the ground.

The kinematics analysis is carried to offer a structured environment to the experimental data, and to reconstruct certain information not readily available directly from the data. First, the balance was evaluated, recreating from the gait data the trajectories of the zero-moment point (ZMP), of the center of mass (COM) and of the meta of the feet, using the linearized inverse pendulum model; then an estimation algorithm was used to fit the previously calculated trajectories and the joint angles from the dataset to a kinematic model. In this way, it was possible to model the kinematics that aims at both a stable gait and a physiological joint motion.

Artificial neural networks were used to correlate muscular activity (patient voluntary action) and joint's motion of lower limbs. These networks have been trained to predict the angular positions of leg joints from EMG signals. Two recurrent types of neural network have been tested for this purpose, the non-linear autoregressive network with exogenous inputs and the long and short memory neural network. The EMG signals were properly treated to eliminate the unnecessary parts, by means of the extraction of the principal muscle activation, the computation of the envelopes and the muscle synergy, to apply them to the neural networks.

Introduction

An exoskeleton for rehabilitation is designed to aid recovery and improve the mobility of the wearer subject to some type of locomotor deficit. Many people lose their physical mobility as a result of various accidents, illnesses or simply due to aging. The use of rehabilitation robots could help their recovery, reducing the burden on therapists, realizing data detection during training, and aiding the quantitative evaluation of improvements in a controllable and repeatable manner.

Exoskeletons are useful for muscle and joint recovery by mobilizing the patient in a controlled way. They can be also used in tailored therapies to activate the physiological mechanism of *neuroplasticity*. This is a process in which the nervous system rearranges its structure, function and connections between neurons and between groups of neurons in response to damage and stimuli from outside or inside the body. Through controlled training, these neurological recovery mechanisms come into action, leading to a recovery of physiological muscle activity and an increase in *proprioception* (from the Latin *proprius*, meaning belonging to oneself, it is defined as the sense of position and movement of the limbs and body that one has independently of sight).

In the case of gait impairment, lower-limb exoskeletons can be used for therapies aimed at restoring locomotion abilities. They must be able to guarantee dynamic balance, a correct posture and the joints' physiological movement during all phases of walking. Exoskeletons of this type that have been developed so far fail in guaranteeing optimum balance and patient compliance at the same time.

In this thesis, the kinematics of bipedal gait was studied to find a solution for a man-machine interface for a lower-limb exoskeleton that guarantees both elements that characterize human walking.

Complete gait trials were analyzed to better understand the lower limbs and torso movements during a walk. The dynamic equilibrium was evaluated using one of the well-know stability criteria for the bipeds' gait, the zero moment point (ZMP) of Vukobratovic [48] [47]. To explore the connection between the ZMP and the center of mass (CoM) trajectory, we employed the Kajita linear inverted pendulum (LIMP) model [21] [22], used to model the gait of bipedal robots. Then, in order to analyze the kinematics of walking, a dedicated model was developed to allow the evaluation and observation of the relationships between the physiological movement of the joints and the CoM.

Neural networks were designed to relate electromyography (EMG) signals from limbs muscles with the joint motion, aiming at a synchronization between muscles activity and the motion of the exoskeleton. Thanks to these, high patient compliance (i.e. high adherence of the patient with the movements of the exoskeleton) could be guaranteed.

1.1 Rehabilitative exoskeletons: brief history and review

In the animal kingdom an exoskeleton is the external skeleton that supports and protect the body of certain species. In usage, some of the larger kinds of exoskeleton are known as shells. Some examples of animals with this type of skeletal structure are grasshoppers, crabs, tortoises, snails, etc. Instead an internal skeleton is referred as endoskeleton. Some animals can have both types.

The exoskeletons mentioned here are intended for human use, and can be defined as wearable robotic structures designed to give mechanical benefits to the user [24] [36]. They have to match the anthropomorphic form of the wearer and to be synchronized with its movements. An exoskeleton is *powered* if improves movements of the limbs through electro-mechanical systems connected to a power source, and *passive* if can still add strength to the wearer but is not powered.

We can classify the exoskeletons into four main groups depending on their specific application field and the use case: consumer, industrial, military and medical. Consumer exoskeletons are designed to assist in daily activities. In the case of industrial exoskeletons, the primary goal is to augment human capabilities and to prevent weakness or injuries while performing repetitive actions. Military exoskeletons enhance the physical strength and endurance of the soldiers. In the end, medical exoskeletons are deployed for restoration of the physical movements of physically handicapped patients and are designed to help their recovery at a faster rate.

In this chapter is reported a briefly review of some exoskeletons, with a particular focus on the powered ones used for rehabilitation of the lower limbs.

The history of the earliest exoskeleton design can be traced back to 1890, when a Russian engineer N. Yagn developed the first exoskeleton-like device. It consisted in a long bow operating in parallel to the user's legs by enhancing his lower limbs, helping the wearer in walking, running and jumping.

In 2004 the Human Engineering and Robotics Laboratory of Berkley presented the Berkley Lower Extremity Exoskeleton (BLEEX, Fig. 1.1a) the first functional energetically autonomous load carrying exoskeleton. It provides augmented strength and endurance to its wearer during locomotion, with a walking speed range from 0.9 m/s when loaded to 1.3 m/s when without load. Joint torques of hips, knees and ankles, are controlled through inverse dynamic without user interaction.

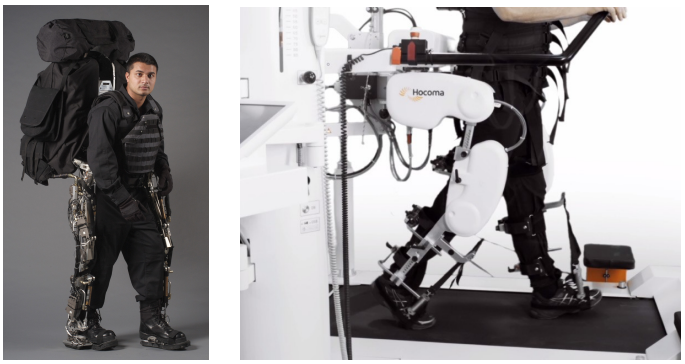
After BLEEX a multitude of model has been developed, each one with its proper characteristics in terms of structure and control design. Medical exoskeletons have an extra challenge due to the different mobility disorders that a patient can have. Stability and supports are the main features that an exoskeleton for lower limb rehabilitation has to provide.

Two different lines of design have been followed: the former involves body weight support systems (BWS) in addition to a treadmill for guaranteeing the safety of people lacking of lower-limb force, the latter allows the exoskeleton to transfer all the weight to the ground.

Lokomat by Hokoma is an example of the first design class (Fig. 1.1b). It is a passive

system, because without direct user interaction, and is used for rehabilitation on a stationary position. The knee flexion-extension of hip and knee is actuated, while the ankle plantar flexion is limited by elastic constrains. The control system imposes a position or a trajectory of the lower-limb joints to realize a gait as near as possible to the patient physiological one.

Instead, there are many different designs of exoskeletons that discharge all the weight on the ground, depending on the specific medical use case. RoboKnee of Yobotics Inc. supports the knee with a series elastic actuator at the joint. The control system controls the actuators depending on the knees' torque necessary to maintain a stable posture. The torque is computed evaluating the reaction forces on the foot plant.



(a) (b)

Figure 1.1: (a) BLEEX; (b) Lokomat;

The Hybrid Assistive Limb (HAL) is the name of a series of exoskeletons developed by University of Tsukuba in collaboration with the robotics company Cyberdyne. The medical exoskeleton HAL-ML05 (Fig. 1.2a) is an actively controlled by the electromyography (EMG) signals from user's muscles. It also gets joint angles and plantar force data. It has four active degrees of freedom (DOF) and is used by people affected by SCI (Spinal Cord Injury) and other neurological and musculoskeletal injuries in which the patients have a sufficient residual force. However its use is limited to assist gait training and not for sit-to-stands or stair-climbing movements.

EKSO GT (Fig. 1.2b) is the medical exoskeleton of Ekso Bionics, designed for post stroke patients, complete SCI patients and traumatic brain injury patients. It has four actuated joints in hips and knees, two sprung joints in the ankles' sagittal planes, and sensed crutches for helping the sagittal balance. It has a control method based on finite state which identify particular gait phases, determined by the sensing of the foot pressure, hip angles, knee angles, arm angles and crutch load. The actuation can be controlled by a therapist or by the patient actively or passively.

The Indego (Fig. 1.2c) of the Parker Hannifin Corporation exoskeleton is designed for SCI patients rehabilitation and mobility assistance too. Its control architecture is supervised by a finite-state machine, and is provided of joint-level controllers which actively rotate the knees and hips angles in the sagittal plane. The state are divided in static states, which depend on the joints angles, and transitions state, controlled by the center

of pressure (CoP) movement. The exoskeleton lack of ground sensors, thus depends only by the user's posture and tilt.

ReWalk Robotics, formerly Argo Medical Technology, is the developer of the ReWalk exoskeleton (Fig. 1.2d). This motorized exoskeleton is designed for users with disabilities such as paraplegia, quadriplegia, hemiplegia, polio-resultant paralysis and other mobility diseases. It has hip and knee powered joints, with four motors acting on the sagittal plane, commanded by a main unit that adapts the stance depending on the upper body position. The measurement of the tilt and of the ground reaction forces enables a real-time control of the gait phases.

(a) (b) (c) (d)

Figure 1.2: (a) HAL; (b) EKSO; (c) Indego; (d) ReWalk;

However, the exoskeletons mentioned above do not manage to perfectly combine postural balance control with patient compliance. Each of them has its limitations: some exoskeletons controls systems are based on checking the equilibrium with criteria such as the CoP and CoM trajectories or the tilt of the trunk; others prefer to actuate the joints by considering the patient's voluntary action through detection of muscle activity by EMG. Most of them needs some kind of feature for granting the patients stability, such as crutches, or to be linked to a stationary structure. What we want to achieve with the

path started in this thesis, is an exoskeleton capable of taking into account the voluntary action of the patient, and at the same time, to always self-regulate itself to ensure balance, without the need for external support elements and only with the joint action.

1.2 Thesis content

During the course of this thesis project, we to studied the interactions between muscles activity, posture and equilibrium of the patient, displacements and angular speeds of lower limb joints, and feet' trajectories. In this direction, we performed an analysis of the gait kinematics aimed at the evaluation of main features of the walk, leaving the dynamic to a second moment.

The first chapter was an introduction to the project: it talked about rehabilitation exoskeletons for the lower limbs, their desirable characteristics and the aims of the thesis.

This is followed by the two parts that constitute the body of the thesis: the first "Background analysis" serves to provide the reader with the tools and knowledge related to the study of bipedal locomotion and neural networks, necessary for the second part "Simulations", where the processes and tests developed to analyze the kinematics of human gait are illustrated.

The first parts is formed by:

- The second chapter, which provides the basis for the study of gait biomechanics and kinematics, introducing the main terms of gait analysis, providing the fundamentals for the design of a kinematic model and for the motion evaluation, offering a overview of the stability criteria and presenting a review of gait measurement systems.
- The third chapter, which focus on bipedal balance during walking, how to maintain and evaluate it, discussing the relationships between ZMP and CoM. This can be reduced to the behavior of a linear inverse pendulum (LIPM).
- The fourth chapter provides an overview of the elements that make up an artificial neural network, the mathematical principles that govern it, and how the training and validation processes work.

Whereas the second part by:

- The fifth capital, in which the experimental datasets used in the various tests are described.
- The sixth chapter offers a report on analysis of the experimental information, and explains the processes of generating ZMP, foot and CoM trajectories from data processing, according to the LIMP model.
- The seventh chapter illustrates the modeling of a bipedal walker, and the development of an algorithm based on the kinematic relationships between the joints of the biped and its configuration in space, which would make it walk using the experimental data and the trajectories generated in the previous chapter.

- The eighth chapter deals with tests carried out to correlate EMG signals and joint movement through neural networks. Treatments carried out on EMG signals to clean it up, the use of muscle synergies and the results obtained from recursive neural networks are explained.
- The ninth chapter deals with an experiment in which the equilibrium of a bipedal model controlled with angular joint signals derived from neural networks is evaluated in terms of ZMP and the trend of its CoM.

The thesis document concludes with a chapter in which the results of the experiments are reviewed, the pros and cons of the procedures described in the chapter on simulations are evaluated, and ways to improve them or their possible implementations are proposed. This ends with a section in which the various contributions that made the thesis possible are reported and the role of each of its two authors is described.

Part I

Background analysis

Fundamentals of gait analysis

2.1 Human biomechanics

The list of fields interested in human movement is quite long: orthopedic, surgical, prosthetic, kinesiological, sports, rehabilitation, psychiatric and so on. The science that evaluates human and non-human motion in every aspect is called *kinesiology* (from the greek *kìnesis*, movement and *logos*, study). It blends knowledge from psychology, motor learning and physiology, as well biomechanics.

Biomechanics is the inter-discipline which studies, analyses and evaluates human movement by applying mechanics. It is a combination of medical and mathematical sciences. It requires assessments from multiple disciplines, such as physics, mathematics, chemistry, anatomy and physiology, to develop comprehensive analyses that bring together various views.

2.1.1 The human gait

Human gait consists in a series of alternating movements of the legs in a rhythmic motion that make the body move forward with minimal energy expenditure. It is mainly influenced by the shape, position and function of neuromuscular and musculoskeletal structures as well as by the ligamentous and capsular constraints of the joints.

The gait cycle

A complete gait cycle is defined as the period of gait comprised between two successive strike of a foot (see Fig. 2.1). The heel contact is the starting and finishing event.

The gait cycle can be broken in two distinct phases depending on the mutual contact of both feet with the ground: the stance phase and the swing phase.

- The *stance phase* of gait begins with the first touch of a foot and ends when the same foot leaves the ground. It lasts around sixty percent of the gait cycle.
- The *swing phase* of gait is when the foot is in the air, begins when it leaves the ground and ends once the same foot touches the ground again. This phase lasts around the other forty percent of the gait cycle.

When both feet are in contact with the ground the *double support phase* (*DSP*) is set. It happens two times during gait cycle, at the begging and at the end of the stance phase. When running the double support phase disappear.

Each double support phase takes up about ten percent or more of the gait cycle of a walk, and its duration depends on the walking speed.

The period during which only one limb is in contact with the ground is the single support phase. It last as long as the swinging phase.

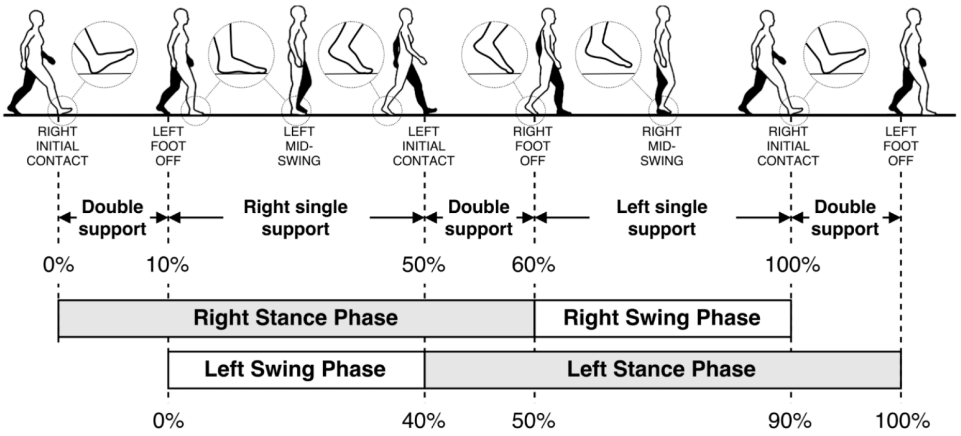


Figure 2.1: The gait cycle

The tasks of gait

A physiological gait must comply three task: the *weight acceptance*, the *single leg support* and the *limb progression*. The gait cycle can be split in more refined sub-phases depending on the support phases and the task which is being executed (see Fig. 2.2):

- Initial contact (IC)
- Loading response (LR)
- Midstance (MSt)
- Terminal stance (TSt)
- Preswing (PSw)
- Initial swing (ISw)
- Midswing (MSw)
- Terminal swing (TSw)

The stance event has five phases and the swing event has three phases, divided as shown in Fig. 2.2. The preswing phase adapts the body for going ahead, thus it can be also included in the swing period.

The weight acceptance is performed during the initial contact and loading phases. In the last phase the weight passes from the posterior to the anterior limb. This is the most demanding task and must fulfill the shock absorption, the initial limb stability and the preservation of progression.

Lifting the foot for swing, begins the singles support phase. It is carried out during midstance and terminal stance, and one limb supports all the body weight.

The limb or body advancement is executed during the following phases: preswing, initial swing, midswing and terminal swing. The preswing is the transition from single support to limb progression, when the stability maintained during the single support is broken. Then swinging leg advances with the body towards the next contact point.

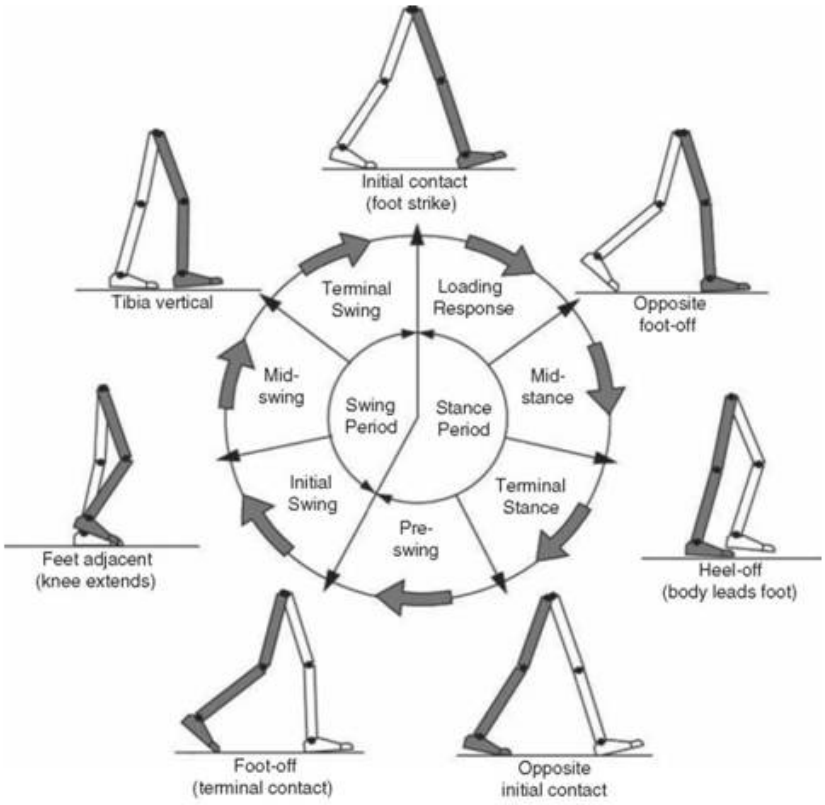


Figure 2.2: Relationship between the main phases and the sub-phases of the gait cycle

Stride and Step

The gait cycle has been identified in literature by the term stride, which must not be confused with the inappropriate word step.

The *stride* is the equivalent of a gait cycle. It is based on the actions of one limb. The duration of a stride is the interval between two sequential initial floor contacts by the same limb (i.e.,right IC and the next right IC).

Step refers to the timing between the two limbs. In each stride (or gait cycle) there is a right and a left step. At the midpoint of one stride the other foot touches the ground to begin its next stance period. The interval between an initial contact by each foot is a step (i.e., left and then right). The same offset in timing will be repeated in reciprocal fashion throughout the walk (Perry J., [20]).

On the basis of these definitions are established some variables employed for describing the gait:

- *Stride length*: the distance between two successive IC points of the same foot, the sum of two step length belonging to two different feet.
- *Step length*: the distance between the IC point of one foot and the IC point of the opposite foot.
- *Cadence* or *walking rate*: the number of steps executed in a minute during a gait.
- *Velocity* is the product of step length and cadence, and is the distance covered per unit of time.
- *Step width*: also called *base of support*, is the distance measured between the heels of the two feet during the support phase. It is estimated either between the medial-most borders of the two heels or between lines through the midline of the two heels. The average stride width for adults is between 3-8 cm.

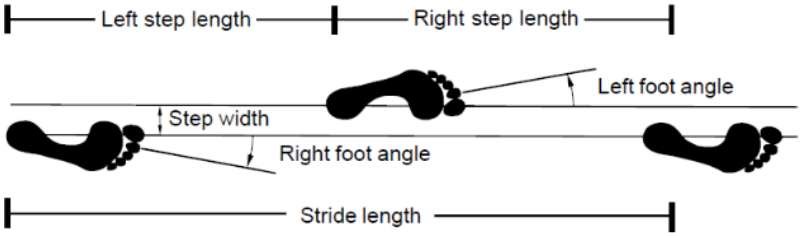


Figure 2.3: Gait cycle, stride length, and step length and width

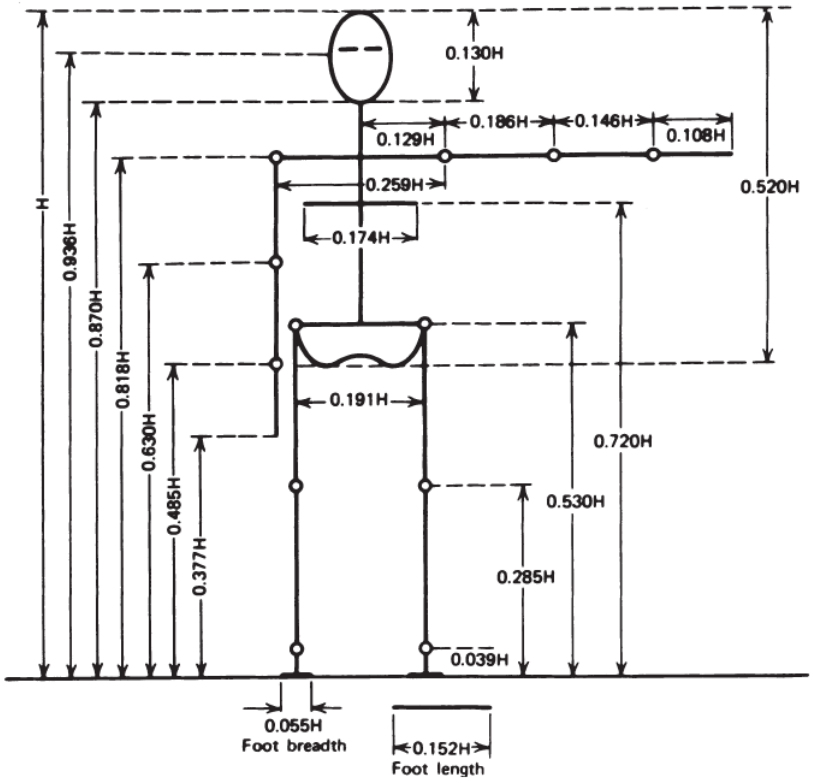
2.1.2 Anthropomorphic human dimensions, volume and weight distribution

To build a proper kinematic model of an human-exoskeleton system, human dimensions must be considered as reference. Kinetics analyses require also data regarding mass distributions, mass centers, moments of inertia, and the like. It is also necessary to identify the exact location of the body joints centers of rotation.

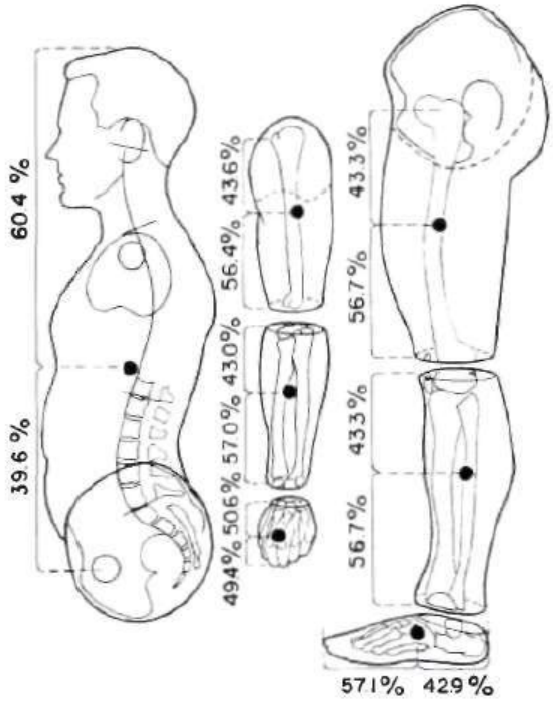
Anthropometry is the major branch of anthropology that studies the physical measurements of the human body to determine differences in individuals and groups. It relates the body characteristics described above with some of their determinants such as race, sex, age and body type, thanks to a wide variety of measurements.

Dempster and coworkers (1955,1959)[11] have summarized estimates of segments lengths and joint center locations relative to anatomical landmarks. Drillis and Continini (1966)[12] reported an average set of segment lengths expressed as a percentage of the body total height, used as relative unit of measure (Fig. 2.4a). By reasoning in this manner, they also computed the position of the center of mass of each body segment, and expressed it as a percentage of the total length of the segment. These segment proportion could be useful as a good approximation in absence of more accurate data, better if measured directly from the individual (Winter D., [49]).

The moments of inertia of a body segment are defined about an axis of rotation which, in most studies, is passing through its center of gravity. Occasionally are defined as passing through an estimated joint center of rotation.



(a)



(b)

Figure 2.4: (a) Body segment lengths expressed as a fraction of body height H. (b) Location of mass centers of body segment

2.1.3 Kinematics

Reference systems conventions

Kinematic variables are involved in the description of the movement, and include linear and angular displacements, velocities and accelerations. In order to keep track of all the kinematic variables, it is important to establish the convention of reference systems in which represent them.

Trajectories are usually represented w.r.t. an absolute global spatial reference system, with fixed origin and axis directions. The one utilized throughout this text is commonly used to describe gait. It is based on the *body planes*, hypothetical geometric planes used to divide body into sections, used in anatomical terminology (see Fig. 2.5).

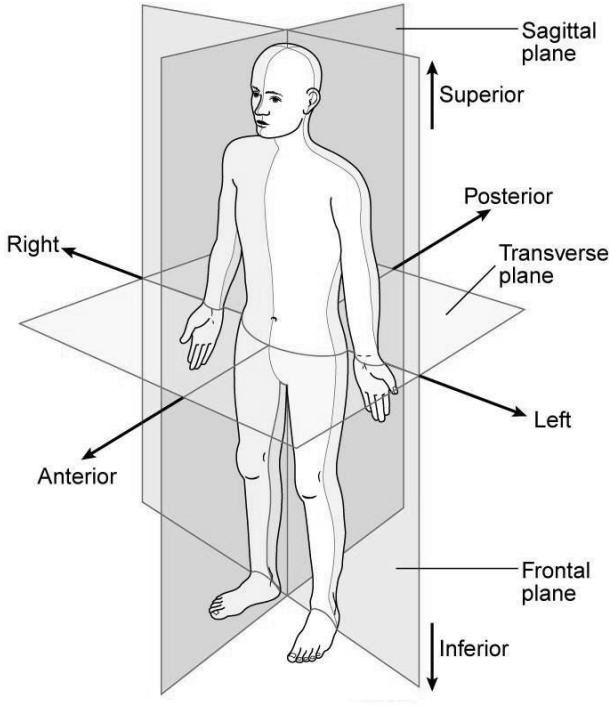


Figure 2.5: Anatomical body planes and directions

The *frontal plane* is the vertical plane dividing the body in anterior and posterior part, the *trasversal o medial* plane is horizontal, and splits the body in upper and lower part, the *sagittal plane* is vertical, and separates the body in left and right part.

The axis of our reference system are X in the direction of progression at the intersection of the sagittal and traverse planes, Y in the sideways direction at the intersection of the transverse and frontal plane and Z in the vertical direction at the intersection of the transverse and frontal plane. X is positive in the anterior direction, Y in the left direction and Z in the superior direction. The angles are measured starting from the direction of the first axis that expresses the plane (e.g. for the XY the angle starts from 0°), increasing counterclockwise. The origin of this absolute reference system is located on the body *center of gravity* or of *mass* (*CoG* or *CoM*), positioned at this point at the start of the walk, and then held fixed in space.

In some cases it has been useful to represent the kinematic variables not w.r.t. to the center of mass, but w.r.t. the ground. The new reference system will have as its center the projection of the CoG on the soil, and the XY plane will no longer correspond to the transverse (or medial) plane but to the ground surface.

Joint angular displacements and velocities are expressed w.r.t. a local frame, centered in the center of the articulation (see Fig. 2.6). It has as its axes the axis of joint rotation, an axis in directed along one of the body segments connected by the joint and the last one positioned according to the right hand rule. This convention is very close to the one used for describing the manipulators in robotics.

In the anatomical literature is established a definite convention to describe the angular motion of the joints, w.r.t. the body planes described above.

- *Flexion* and *extension*: for joint rotations in the sagittal plane. With flexion the articulation angle is decreasing while during extension increases.
- *Abduction* and *adduction*: for joint rotations in the frontal plane. During abduction the body segment is moved away from the medial line, and in the adduction the opposite occurs.
- *External* and *internal rotation*: rotations of a limb around its axis on the horizontal plane. The former brings the limb closer to the medial line, in the latter further.

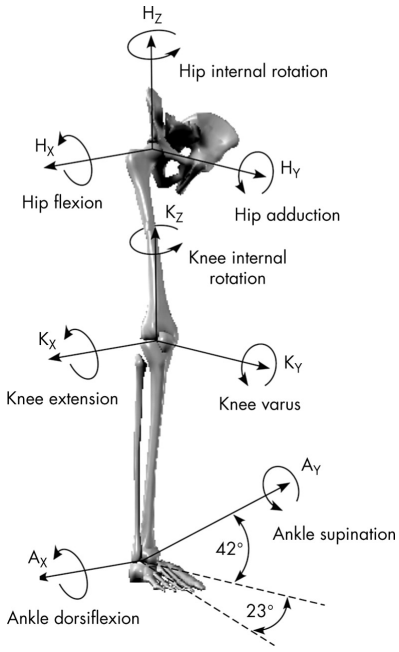


Figure 2.6: Local frames of the joints of a leg.

Human trajectories

Human walking trajectories are studied in order to evaluate the correct motion of each link and joint during gait. The focus is on the trajectory of the legs, the angles of the torso and foot in relation to the ground, and the movement of the CoG.

The *stick diagram* (see Fig. 2.7) is a simple method to represent these trajectories in the plane of the movement, in which each corporeal segment is displayed as a straight line or stick. To plot it, the coordinate data of the pelvis, knees, ankles, heels and toes are required. Thus, and adequate walking pattern should be generated.

Joint angular kinematics

The ankles, knees, and hips articulations motion generate the gait. Each one possesses a certain number of *degrees of freedom* (*dof*) in specific body planes, with definite range of motion. Although each joint can rotate along three axes, some degrees of freedom have such small rotation ranges that they can be ignored.

The ankle joint performs *plantar flexion-extension* and *prone-supination* (i.e. internal and external rotations), the knee joint can execute only flexion-extension, while the pelvis is able to rotate in all the three body planes.

Joint angles can be represented either w.r.t. the time or w.r.t. the gait cycle percentage. The latter allows the evaluation of events during a single cycle, highlighting their dependence on angles. In Fig. 2.8 are reported the ankle, knee and hip flexion-extension joint angles w.r.t the gait cycle percentage of the subject AB01 in a plane treadmill trial belonging to the datasets in [18]. In each plot the green line represents the mean angle, while the yellow lines are the various gait cycles angles.

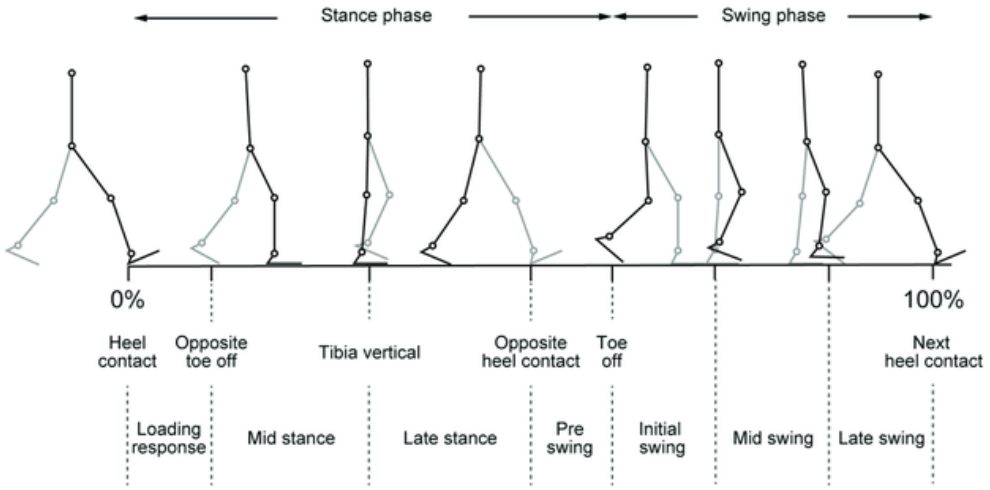


Figure 2.7: Example of stick diagram on the sagittal plane.

2.2 Biped locomotion stability criteria

Apart from the execution of the gait in kinematic terms, the most important task that a biped must carry out during motion is to preserve its dynamic balance, named by most authors *stability*. It has been the subject of many studies in the field of gait biomechanics and biped robotics.

The contact between foot and ground can be modeled as an *unilateral* (since attractive forces are not present) and *underactuated* (or *passive*, since it cannot be controlled

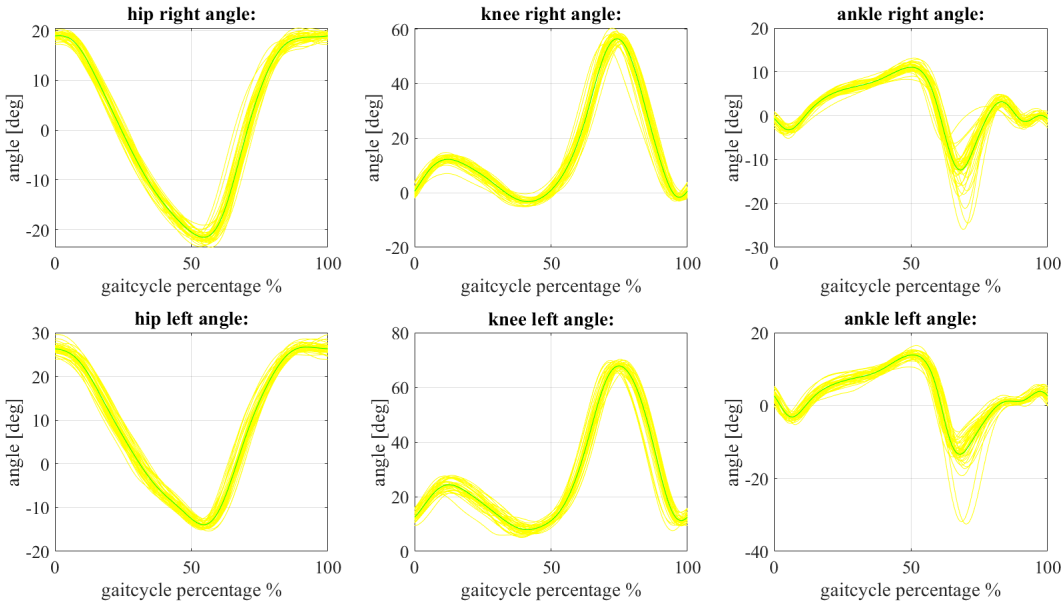


Figure 2.8: Flexion-extension joint angles

directly) joint. Thus every joint of a biped is powered and directly controlled, except for the one representing the ground contact. Consequently, it is essential for the realization of the walk that stability is ensured by an appropriate dynamic of the body over the foot, according to some kind of criteria.

The main parameters evaluated in the balance of a biped have been the *center of pressure* (*CoP*) and the *"Zero-Moment Point"* (*ZMP*) introduced by Vukobratovic et al. [48][47]. The ZMP has been one of the most used criterion for evaluating the dynamic equilibrium of walking biped robots, and several other methods for assessing the stability of a biped have been derived from it. Currently, it has gone from being an element of study belonging only to the world of robotics, to also being examined in the context of human walking, due to its affinity with the CoP.

Other criteria for assessing the stability of a biped, mainly used in robotics, are the *"Foot Rotation Indicator"* (*FRI*) proposed by Dr. Ambarish Goswami [17], which is an extension of the ZMP theory, and the *"Contact Wrench Cone"* (*CWC*) developed by Dr. Hirohisa Hirukawa and coworkers from the AIST [19].

2.2.1 Zero Moment Point (ZMP)

The ZMP has been proved to be helpfull in the field of robotics, as a dynamic stability criterion, but also for the analysis and control of human gait in rehabilitation robotics. It was a turning point in gait planning and control, and it has been time the only criterion for the synthesis of biped gait for a long. Before clarifying what is its meaning and what it implies, the concepts of support polygon and CoP must be introduced.

During each phase of walk, some points of the feet are in contact with the ground. The *support polygon* is a convex 2-D or 3-D polygon which is defined by the foot points that are touching the ground. In the single support phase of the gait the support polygon corresponds to the area of the supporting foot in contact with the terrain, while in the

double support phase is the area between the points of the feet in contact with the ground (see Fig. 2.9).

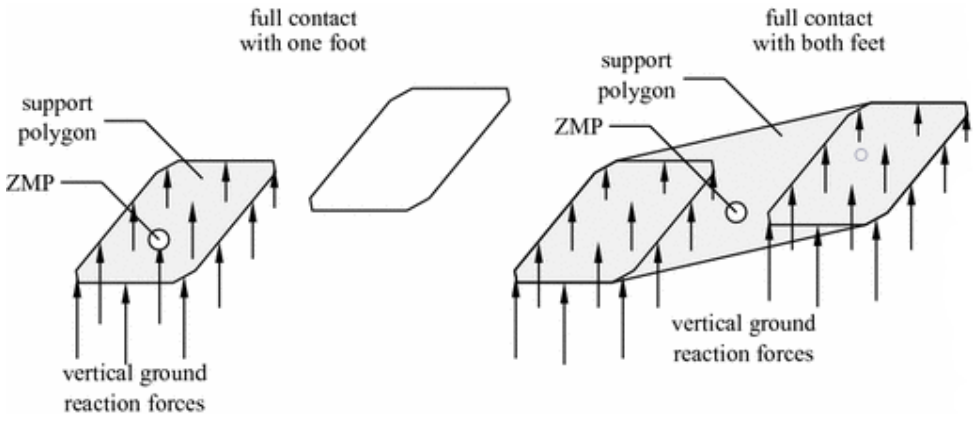


Figure 2.9: Biped body and forces acting on the support foot

The CoP has been defined by Vukobratovic et al. [48] as:

Definition 1 (The notion of the CoP)**.** *CoP represents the point on the support polygon at which the resultant of distributed foot ground reaction forces acts.*

Thus CoP represents the application point of the resultant of the ground reaction forces. In gait analysis it is computed using measurements from force platform or pressure mats. During the stance phase of human locomotion the CoP generally moves from the heel towards the tip. The formulas for finding the CoP position are the following:

$$CoP_x = \frac{-M_{Ry}}{F_{Rz}} \quad CoP_y = \frac{-M_{Rx}}{F_{Rz}} \quad CoP_z = 0 \tag{2.1}$$

Where M_{Ri} and F_{Ri} are the moments and forces resultants along the i-th axis.

Along the definition of the CoP, Vukobratovic et al. have defined precisely the ZMP notion [48]:

Definition 2 (The notion of the ZMP)**.** *The pressure under supporting foot can be replaced by the appropriate reaction force acting at a certain point of the mechanism's foot. Since the sum of all moments of active forces with respect to this point is equal to zero, it is termed the zero-moment point (ZMP).*

Dynamic balance can be achieved by ensuring that the foot's whole area, and not only the edge, is in contact with the ground. The foot relies freely on the support, and it is related to the environment via the friction force and the vertical force of the ground reaction.

From now on will be reported the logic flow of Vukobratovic et al. in [47] for introducing the concept of ZMP. Let us consider the single support phase, with the whole foot touching the ground. By neglecting the body from the ankle upwards we can replace its influence by the force $\mathbf{F}_{AK}$ and the torque $\mathbf{M}_{AK}$ (see Fig. 2.10).

The weight of the foot itself acts at its CoM. At point P the reaction force of the ground $\mathbf{R}$ acts on the foot to provide the equilibrium.

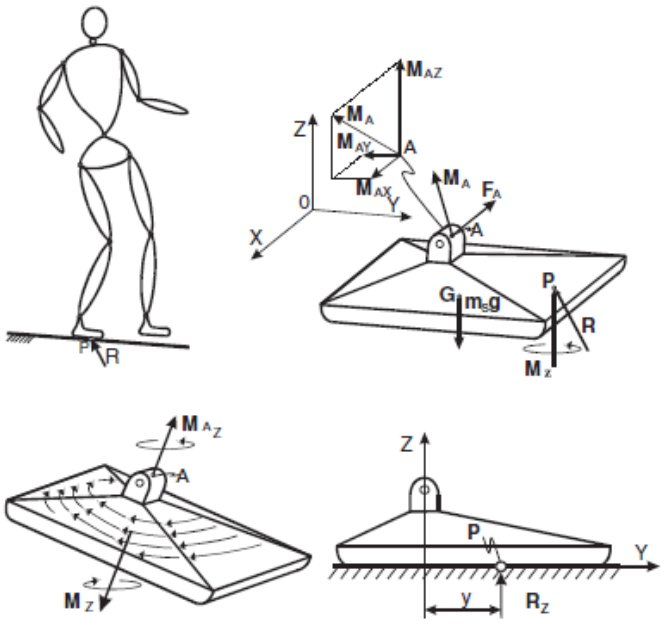


Figure 2.10: Biped body and forces acting on the support foot

The resultant of the reaction force and torque in P respectively are $\mathbf{F} = (R_x, R_y, R_z)$ and $\mathbf{M} = (M_x, M_y, M_z)$. The horizontal component of the force $\mathbf{F}_A$ applied in the ankle is balanced by the horizontal components of the reaction force (R_x, R_y), which are the result of friction. The vertical reaction moment M_z represents the torque due to friction forces which compensates the vertical component of $\mathbf{M}_A$ and the moment induced by the force $\mathbf{F}_A$.

Thus, assuming a foot-floor contact without sliding, the static friction will compensate for the horizontal components of the force (R_x, R_y) and the vertical component of the moment M_z. Instead the vertical reaction force R_z will be the component of the reaction that balances the vertical forces. Since the foot-ground contact produces unidirectional reaction forces that can be only oriented upwards, the horizontal components of all active moments can be compensated only by changing position of the reaction force $\mathbf{R}$ within the support polygon. Therefore, the horizontal component of $\mathbf{M}_{AK}$ will shift the reaction force to the corresponding position, to balance the additional load.

Consequently, exists only M_z at point P. If the support polygon is not large enough to allow the force $\mathbf{R}$ to act in the appropriate position, the force will act on the support's edge to balance the action of external moments. The uncompensated horizontal component of the moment $\mathbf{M}_A$ will cause a rotation of the body about the support edge, which can result in the body's overturning.

Therefore, we can say that necessary and sufficient condition for granting the body equilibrium is that for the point P:

$$M_x = 0 \quad M_y = 0 \tag{2.2}$$

Because the horizontal components of the ground reaction moment acting on the foot are equal to zero, the point P was named the point of zero momentum or ZMP. In other

words, the reaction of the ground due to the foot resting on it can always be reduced to $\mathbf{R}$ and to M_z, the vertical component of the moment; the point P at which the reaction force is acting represents ZMP.

To ensure dynamic stability with the ZMP criterion is a fundamental prerequisite that the support foot is fully resting on the floor. The static equilibrium for the supporting foot is defined by the following equations:

$$\mathbf{R} + \mathbf{F}_A + \mathbf{mg} = 0 \tag{2.3}$$

$$\overrightarrow{\mathbf{OP}} \times \overrightarrow{\mathbf{R}} + \overrightarrow{\mathbf{OG}} \times \mathbf{mg} + \mathbf{M}_A + M_z + \overrightarrow{\mathbf{OA}} \times \mathbf{F}_A = 0, \tag{2.4}$$

where $\overrightarrow{\mathbf{OP}}$, $\overrightarrow{\mathbf{OG}}$, $\overrightarrow{\mathbf{OA}}$ are radius vectors from the origin $O_{x,y,z}$ of the coordinate system to the ground reaction force acting point P, the foot mass center is CoM, the ankle joint center is located in A, and m is the mass of the support foot. If we place the origin of the coordinate system at point P and project Eq. 2.4 onto the z-axis, the vertical component of the ground reaction moment (can be also called ground friction moment) will be:

$$\mathbf{M}_z = -(M_A^z + (\overrightarrow{\mathbf{OA}} \times \mathbf{F}_A)^z \tag{2.5}$$

Generally, this moment differs from zero, and it can be reduced to zero only by the appropriate dynamics of the body. Eq. 2.4 onto the horizontal plane gives:

$$(\overrightarrow{\mathbf{OP}} \times \overrightarrow{\mathbf{R}})^H + \overrightarrow{\mathbf{OG}} \times \mathbf{mg} + \mathbf{M}_A^H + (\overrightarrow{\mathbf{OA}} \times \mathbf{F}_A)^H = 0 \tag{2.6}$$

This last equation is a basis for computing the position of P, where the ground reaction forces are applied. It represents the equation of the foot equilibrium, and permits to find out the suitable ZMP position to ensure dynamic equilibrium. When P is inside the support polygon, the body is in a dynamically stable equilibrium, but accordingly to Eq. 2.6 it can be also outside the polygon. In this case, the point is defined fictitious ZMP (FZMP). Therefore, the ZMP exists only when inside the support polygon.

For a dynamically balanced gait CoP and ZMP coincide. The CoP is where the resultants of the distributed pressure force between the foot and the ground acts. The pressure acting point is the ZMP if it balances all the active forces acting on the mechanism during the motion. However, when the gait is not dynamically balanced, the ZMP doesn't exist and collapses on the edge. In summary the ZMP coincides always with the CoP, because it exists only for stable gaits, while the CoP is not always equal to the ZMP. The FZMP never coincides with the CoP, because the latter is always inside the support polygon.

The ZMP projection on the sagittal, frontal and ground planes during a walk with constant step width will appear like shown in Fig. 2.11. Each one is a broken line.

The constant traits in the sagittal plane projection represent gait phases in which the ZMP is stationary on the sole of the support foot (thus they are referred to the single stance phase), while in the sloping ones it moves forward from the rear foot towards the front one (thus they are referred to the single stance phase). The same happens in the Frontal projection, where is possible differentiate between constant single support traits and sloped double support traits. The combination of the two will produce a graph representing the projection of the ZMP on the ground.

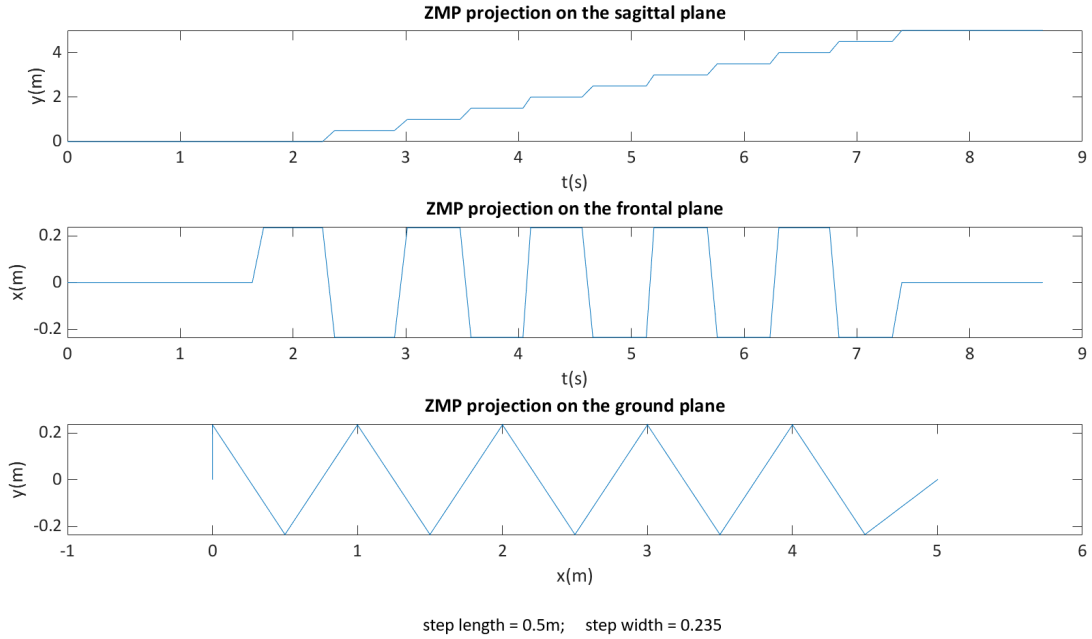


Figure 2.11: Projection of the ZMP for a constant step walk on the Sagittal, Frontal and ground planes

2.2.2 Comparison with other stability criteria

In this subsection are reported two of the most efficient stability criteria used in biped robot gait planning and control. However their application in the field of human gait analysis is still unexplored, and they are briefly illustrated only to underline what are the limits of the ZMP, through a comparison.

We will start the comparison from a direct heir of the ZMP, the FRI point. It has been defined by Dr. Ambarish Goswami[17] as:

Definition 3 (The notion of FRI)**.** *The foot rotation indicator (FRI) point is a point on the foot/ground contact surface, within or outside the convex hull (support polygon) of the foot support area, at which the resultant moment of the force/torque impressed on the foot is normal to the surface.*

The FRI point gives us useful information about the foot rotation and the stability of a biped. It allows to find out the occurrence and the direction of the foot rotation. Its location returns the magnitude of the unbalanced moment of the foot. It also indicates the stability margin of the robot, i.e. the minimum distance of the support polygon boundary from the current position of the FRI point within the footprint. When the FRI point is outside the footprint, this minimum distance is a measure of the instability of the robot. An imminent foot rotation is linked to a displacement of the FRI point towards the boundary of the support polygon.

The FIR point and the CoP are coincident when:

a the foot is at rest or has uniform linear and angular speeds.

b the foot has null mass and inertia.

c the segment between the FRI and the center of mass is parallel to the force applied on the foot CoM, and the foot inertia is zero.

When the robot is stable and stationary, the FIR point coincides with the CoP and with the projection of the robot CoG onto the ground. Instead for stationary and unstable configurations the ZMP and the projection of the CoG are coincident and outside the support polygon, while the CoP is on the boundary, because it can never leave it.

However Vukobratovic et al. in [48] disprove the novelty of this criteria. Even if they appreciate the considerations of Dr. Ambarish Goswami about the foot force components, they clearly demonstrate the coincidence of the FIR with the previously defined FZMP.

In the CWC method, Let us consider the set of contact force and torque applied from the ground to the foot of the biped, $(\mathbf{f}_c, \tau_c)$. As reported by Dr. Hirohisa Hirukawa and coworkers this set forms a polyhedral cone in the space of the contact force and torque, which can be called *polyhedral convex cone of the contact wrench* (*CWC*). The following equations define the CWC:

$$\mathbf{f}_c = \sum_{k=1}^{K} \sum_{l=1}^{L} \epsilon_k^l \mathbf{p}_k \times (\mathbf{n}_k + \mu_k \mathbf{t}_k^l) \tag{2.7}$$

$$\tau_\mathbf{c} = \sum_{k=1}^{K} \sum_{l=1}^{L} \epsilon_k^l \mathbf{p}_k (\mathbf{n}_k + \mu_k \mathbf{t}_k^l) \tag{2.8}$$

where the friction cone at $\mathbf{p}_k$ is approximated by a L-polyhedral cone, $\mathbf{t}_k^l$ is a unit tangent vector to make $\mathbf{n}_k + \mu_k \mathbf{t}_k^l$ be the l-th edge of the polyhedral cone, μ_k the friction coefficient and ϵ_k^l a non negative scalar which gives the magnitude of the force of the l-th edge of the approximated friction cone at the k-th contact point.

The contact between the robot and the environment is *strongly stable* when $(\mathbf{f}_c, \tau_c)$ grants the dynamic stability of the robot and its is *weakly stable* when the robot is unstable but the contact may be stable to $(\mathbf{f}_c, \tau_c)$. The contact will be *strongly unstable* when it is impossible granting stability.

For assessing the dynamic equilibrium, two criteria based on the states of the stability have been proposed:

Theorem 1 (Strong stability criterion)**.** *If* $(-\mathbf{f}_c, -\tau_c)$ *is an internal element of the polyhedral convex cone of the contact wrench, then the contact is strongly stable to* $(\mathbf{f}_c, \tau_c)$*.*

Theorem 2 (Weak stability criterion)**.** *If* $(-\mathbf{f}_c, -\tau_c)$ *is an element of a proper subset of the polyhedral convex cone of the contact wrench, then the contact is called sufficiently weakly stable to* $(\mathbf{f}_c, \tau_c)$*.*

The CWC based stability criterion can be used to establish the strong stability of the foot contact even for robots walking on various terrains, that are not necessarily horizontal and flat. It is equivalent to check if the ZMP is inside the support polygon of the feet when the robot walks on a horizontal plane with sufficient friction. The criterion can also be used to determine if the foot contact is sufficiently weakly stable when the friction follows a physical law.

The ZMP is a rigorous stability criterion for a flat plane walk with sufficient friction. Instead, the CWC has been proposed as a suitable solution for a wider range of situations.

In the case of humanoid walking on flat surface it is demonstrated from the last thirty years, that the ZMP criterion is robust enough, theoretically and experimentally. On the other hand, the CWC could be a powerful tool, or at least in theory, due its capability of granting a criterion also for critical cases at uneven terrain with or without multi-contact points.

2.3 Instrumentation

In gait analysis the biomechanics of human motion, can be evaluated with different methods and instrumentation. These methods can be classified according to various criteria. For example they can be categorized into *non-wearable* or *wearable* sensors:

- *Non-wearable* sensors, are those that cannot be placed on a specific part of the body. However, they interact with the patient through their contact. Some examples in this category are force plates and instrumented treadmills.
- *Wearable* sensors, of several types, with different measurement accuracy and purposes. We can find markers, accelerometers, gyroscopes, magnetometers, electrodes for EMG and many others among them.

2.3.1 Non-wearable sensors

Force transducers and force plates

To measure the force exerted by the human body on an external body or load, we need a suitable force-measuring device. Such a device, called force transducer, generates an electrical signal proportional to the applied force. There are many kinds available: strain gauge, piezoelectric, piezoresistive, capacitive, and others. All these work on the principle that the applied force causes a specific strain within the transducer [49].

- For the *strain gauge type*, a calibrated metal plate or beam within the transducer undergoes to a tiny change (strain) in one of its dimensions.

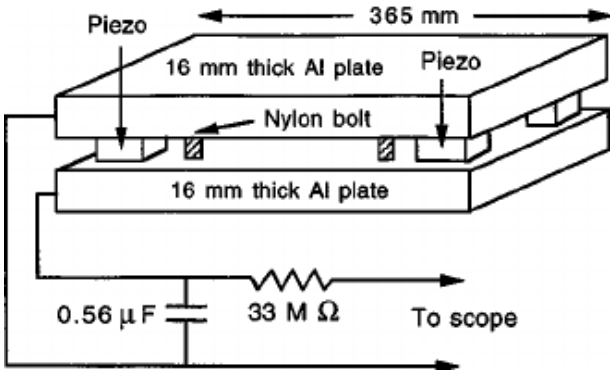


Figure 2.12: Scheme of a piezoelectrical force plate

- *Piezoelectrical*, require slight of the deformations of the atomic structure within a block of special crystalline material, such as quartz. Deformation of its crystalline structure changes the electrical characteristics such that the electrical charge across appropriate surfaces of the block is altered and can be translated via suitable electronics to a signal proportional to the applied force.

- The *piezoresistive* types exhibit a change in resistance which, like the strain gauge, upset the balance of a bridge circuit.

Treadmills

Recently is becoming more popular to use treadmills for gait studies. They can be combined with cameras, markers systems, and force platforms can be inserted directly under the rollers. The use of treadmills in gait analysis has made possible to record straight line walking that can go beyond the distance covered in a laboratory limited by the cameras. It's also much easier to set a walking speed and it's also possible to conduct experiments with different inclinations w.r.t. the ground.

However, depending on the future use of the recorded data, one of the disadvantages may be that the space-temporal recordings of the markers remain confined in the treadmill dimension's instead of advancing meter by meter like a real walk, so this data may need further adjustments [18].

Figure 2.13: Example of a modern system for gait analysis with the use of a treadmill

2.3.2 Wearable sensors

Optoelectronic stereophotogrammetry

It's a technique that involves cameras to capture the trajectory of spherical retroreflective markers attached to the desired locations of the body. With stereophotogrammetry we can evaluate, with a good precision, movement and orientation of each body segment. It enables realistic reconstructions and representations of the musculoskeletal system during a certain motion task. For these reasons it is considered one of the best instrumentation for gait analysis [38].

By the way it also suffers a bit from trajectory gaps, it takes longs time for preparation and the space for analysis is restricted to the area in which the cameras are operating. In addiction it is expensive.

Accelerometer

It's a measurement device whose output consist of the proper acceleration, the acceleration of the body on which is attached w.r.t. its instantaneous coordinate frame. Accelerometers can be single or multi axis and detect magnitude and direction of the acceleration, seen as a vector quantity.

In most accelerometers, the physical principle exploited to measure the acceleration is based on the inertia of a mass subjected to an acceleration. An elastic element suspends a mass, and this mass, in case of acceleration, moves from its rest position. Equating Hooke's law to Newtons law we have $kx = ma$ and see that the displacement of the elastic element is proportional to the mass acceleration.

A displacement-sensitive sensor transforms the it into an electrical signal [38]. There are many types of accelerometers, such as capacitive, strain gauge, piezoresistive and piezoelectric. For gait analysis the most commonly used are capacitive and piezoresistive (Fig. 2.15a and Fig. 2.15b respectively).

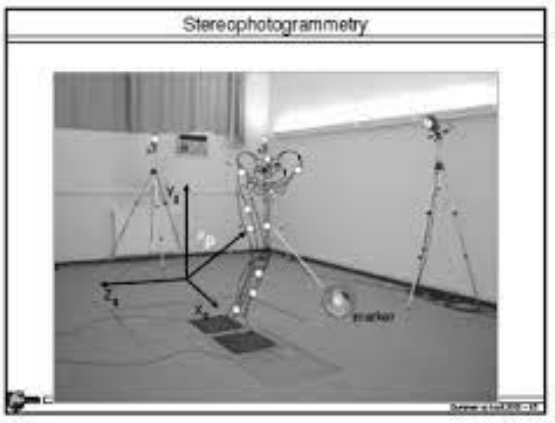
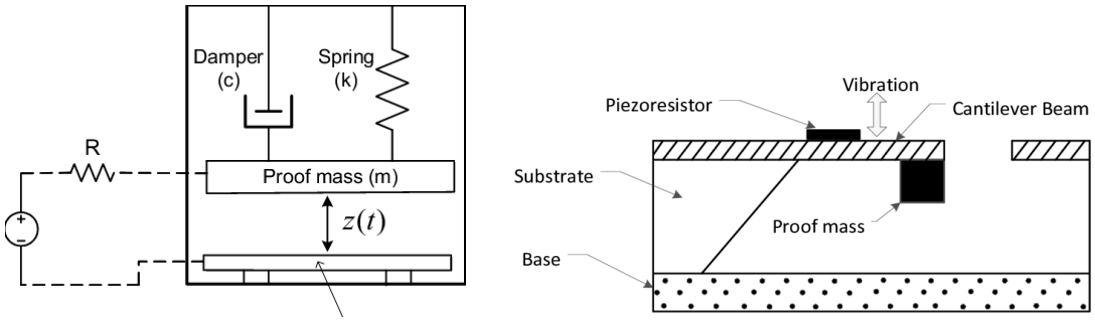


Figure 2.14: Stereophotogrammetry cameras system



(a) (b)

Figure 2.15: (a) Scheme of a capacitive accelerometer (b) Scheme of a piezoresistive accelerometer

Gyroscope

They measure the angular velocity around their sensing axis. Typically they are mechanical and consist of a rotating device which maintains fixed its rotating axis exploiting the conservation of angular momentum law. A 3D gyroscope can be described as a wheel mounted in three gimbals, which are the pivoted supports that enable the rotation around three different axes. The fundamental equation describing a rotating rigid system is the following one.

$$M = \frac{dL}{dt} = \frac{d(I\omega)}{dt} \tag{2.9}$$

Where M is the torque, L the momentum, I the inertia and ω the angular velocity. The derived motion is the precession, and the reaction force induces the gyroscope to rotate around a fixed axis, called spin axis, which does not change its direction even if the support varies its orientation. Thanks to the development of MEMS[1], miniaturizes gyroscopes can become widespread. They consist of a vibrating element that, if subjected to a rotation, is also affected by a vibration in the orthogonal direction to the original one, according to the Coriolis effect [38]:

$$F = -2m(\omega \times v) \tag{2.10}$$

F is the Coriolis force, ω the angular velocity and v the linear velocity of the mass m.

Magnetometer

A magnetometer is a measuring device that detect a magnetic field. A *scalar* magnetometers measure the magnitude of the magnetic field directly, while the *vectorial* ones measure the direction and the strength of the magnetic field detecting the component along a particular axis. Using a three axial magnetometer, thus knowing the components of the magnetic field in three different and independent directions, allows to determine the vector in 3D space [38].

[1]Microelectromechanical systems

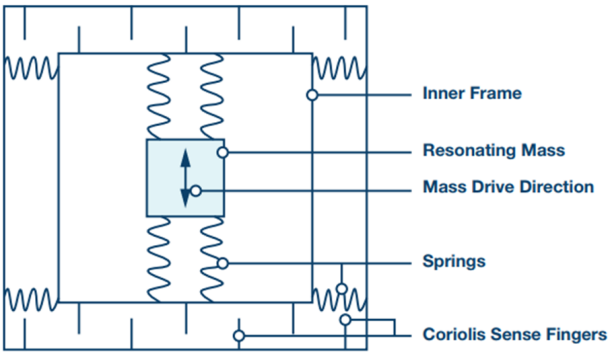


Figure 2.16: MEMS gyroscope scheme

$$h = (h_{earth} + h_{external})n \tag{2.11}$$

Eq. 2.11 represent a single axis magnetometer model, where n is the sensing axes. The most common of the magnetometers is the compass, which points in the direction of the Earth's magnetic north.

Inertial Measurement Units

In many fields, such as navigation, robotics and motion analysis, we need to know as much precise as possible the angular position in the space of objects. So for an accurate estimation of the orientation of a rigid body, w.r.t. an inertial frame, we can use an Inertial Measurement Units (IMU). They are composed by two sensors, a gyroscope which measures the angular rate, and an accelerometer which measures the linear and gravity acceleration With these two sensors an IMU can estimate its attitude.

But since the accelerometer is not sensitive to the rotation around the gravity axis, an additional reference vector is needed to estimate the heading direction. Recent studies have discovered that combining an accelerometer with a magnetometer makes possible to find out both attitude and heading directions. This system is called Magneto-Inertial Measurement Units or MIMU [38]).

Electrogoniometers

An electrogoniometer is an electronic device that uses angle sensors, such as potentiometers, strain gauges and, more recently, accelerometers, appropriately positioned across a joint to measure its angle. It gives good results when used for body movements where we have limited speed and amplitude [26]. The most common electrogoniometers employ one of the following three sensor schemes:

- In *Potentiometric Electrogoniometer* an electrical resistance can be used to determine the angle between the joints. These types of electrogoniometers are somewhat bulky and restrict patient movement.
- For *Flexible Electrogoniometer* the strain gauge mechanism is housed inside a spring, which changes its electrical resistance proportionally to the variation of the angle between the plastic end blocks' longitudinal axes.

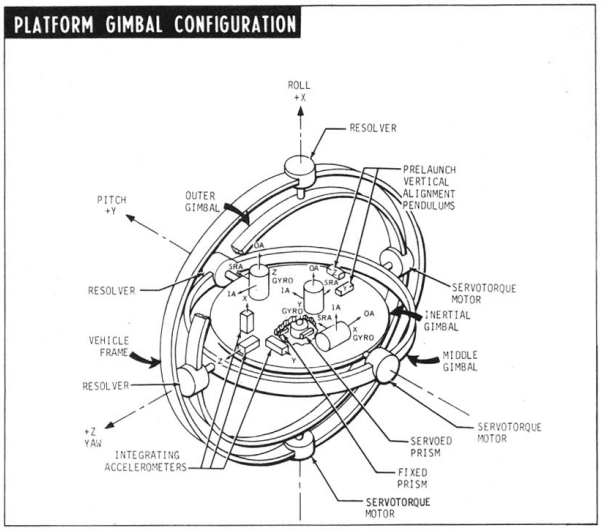
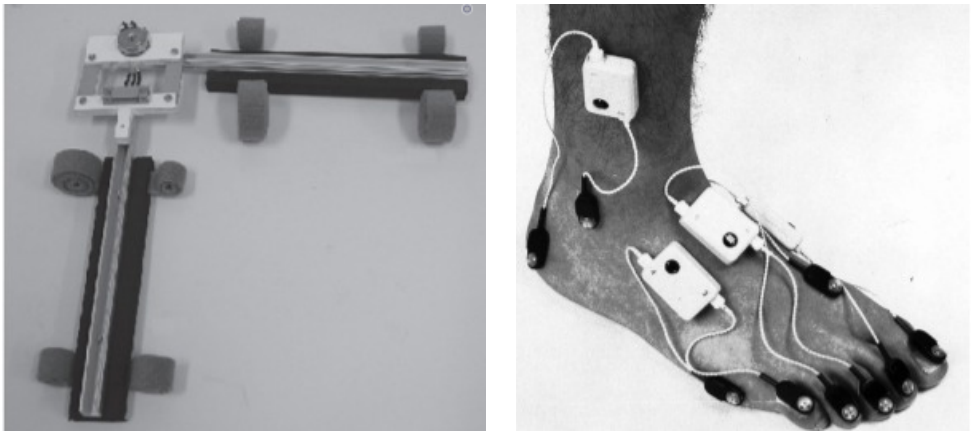


Figure 2.17: Inertial Measurement Units system

- *Optoelectronic Systems* are video systems that use one or more video cameras to track bright markers placed at various locations on the patient's body. The system keeps track of the vertical and horizontal coordinates of each marker, and a software processes this information to determine the angle on the body segments of interest.



(a) (b)

Figure 2.18: (a) A potentiometric electrogoniometer (b) Optoeletronic system positioned on a patient

Optical fiber sensors

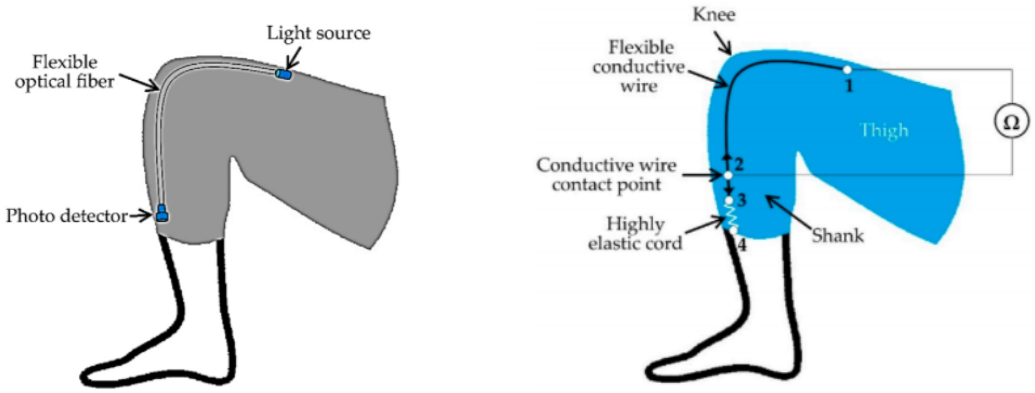
These sensors are made of flexible plastic optical fibers (OFS) through which optical signals are transmitted. The basic components of an OFS-based system are a light source, a flexible optical fiber and a photodetector. The light source at one of the extremities generates the optical signal, which travels through the flexible optical fiber and is received

by the photodetector at the other extremity of the fiber. By measuring the attenuation of the optical signal, it is possible to determine the bending angle of the fiber.

Due to this simple sensing principle and structure, OFS can be easily integrated into a monitoring system for measuring human joint angles. The main benefits of OFS are high resolution, flexibility, light-weight and immunity to electromagnetic interference [14].

Textile-based sensors

Textile-based sensors are very suitable for developing a wearable joint monitoring system. The working principles of all these sensors are similar. In all cases, changes of resistance are measured, and these changes are directly related to the corresponding joint angles. To develop a long-term and regular wearable monitoring device, textile-based sensors can be a good choice because of their flexibility and simple sensing principle. Furthermore, they can be easily integrated into stretchable skin-tight fabrics around the joints. The measurement parameter is the resistivity change of the conductive wire w.r.t. the joints movement [14].



(a) (b)

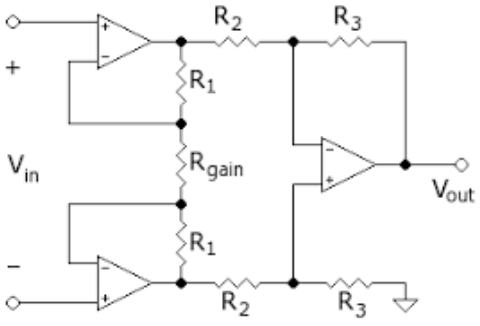
Figure 2.19: (a) Scheme of an optical fiber sensor. (b) Scheme of wearable wire sensor.

EMG signals

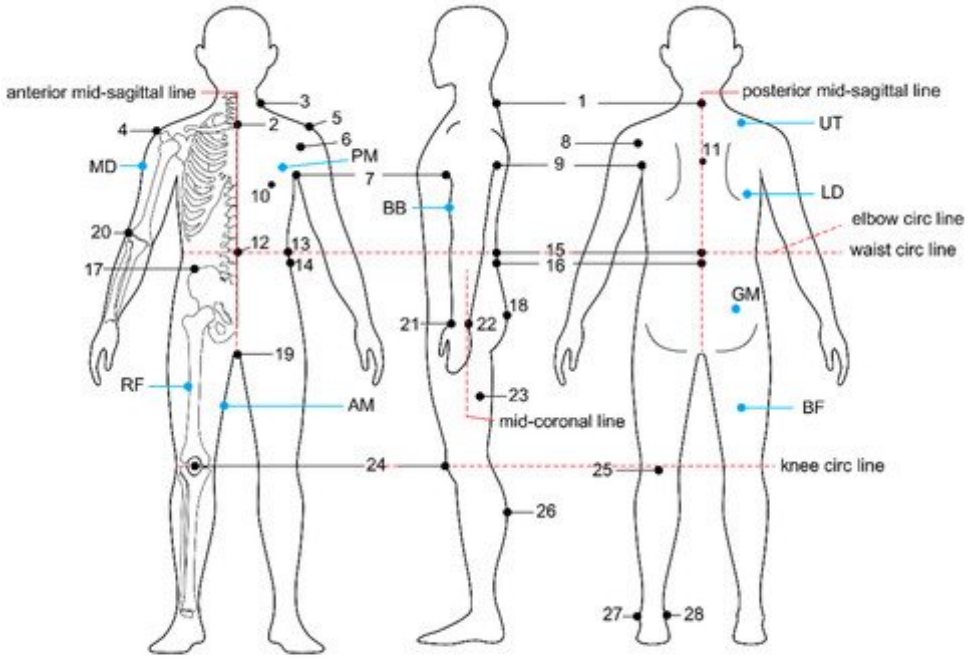
The electrical signal associated with the contraction of a muscle is called an electromyogram, or EMG. The study of EMGs, is called electromyography. An EMG signal increases in amplitude as the intensity of the voluntary muscle activity it quantifies increases.

Electrodes are used for their recording and can be divided into two main groups, surface and indwelling electrodes. For both groups, the basic function is linked to the correct positioning on the patient (position and surface of contact) and the appropriate adjustment of the amplifier with which they operate (Instrumentation Amplifier). EMG signals, depending on the application and specifications of the acquisition system, once recorded are processed with, for example, filters and rectifiers. Their treatment will be explained in more detail in the following chapters, as it is a subject of interest in this thesis.

They are actually under analysis in many studies, which aim to a better understanding of the meaning of these body signals, especially when we talk about muscle synergies, patients with neuromuscular diseases and the discoveries that can be obtained by linking these aspects. EMG signals are a fundamental tool in the analysis of muscle behavior associated with a particular task: for this reason they are widely used in gait analysis[49].



(a)



Anthropometric landmarks (AP)

1	Back neck	15	Back waist	UT	Upper trapezius
2	Front neck	16	Back waist (omphalion)	MD	Middle deltoid
3	Side neck	17	Anteroir superior lilac spine	BB	Biceps brachii
4	Acromion	18	Buttock protrusion	PM	Pectoralis major
5	Lateral shoulder	19	Crotch	LD	Latissimus dorsi
6	Anterior midaxilla	20	Radiale	GM	Gluteus maximus
7	Anterior axillary fold	21	Radial styloid	RF	Rectus femoris
8	Posterior midaxilla	22	Ulnar styloid	AM	Adductor magnus
9	Posterior axillary fold	23	Midthigh	BF	Biceps femoris
10	Bust point	24	Midpatella		
11	Axillary level at midspine	25	Posterior juncture of calf and thigh		
12	Front waist	26	Calf protrusion		
13	Side waist	27	Lateral malleous		
14	Side waist (omphalion)	28	Medial malleous		

EMG electrode points (EP)

(b)

Figure 2.20: (a) Scheme of an instrumentation amplifier (b) Example of a possible placement of electrodes on patient bodies

Gait modeling

Human walking accomplishes a series of motor tasks: the propulsion, the stance stability, the shock absorption and the energy conservation [20]. Although it is limiting to compare the complex movement and dynamics involved in human gait with that of a bipedal robot, they must provide the same tasks listed above and some similarities can be found.

In the first part of this chapter, an analysis of the various types of robotic walking has been carried out in order to evaluate the similarities between them and human gait. Then the possible models used in gait generation are introduced, in order to better understand how an exoskeleton can produce a human-like gait, synchronized with the user's muscular activity, and how it can grant a correct posture and dynamic equilibrium during the locomotion.

3.1 Gait types

In this section, the various categories of walk will be outlined. Two criteria are used for establishing the different types of gait: the first criterion is the type of actuation and the second is the motion speed. Each category has its proper properties and limits, that make it more suitable for certain tasks.

Passive and active walking

Depending on how the walk is actuated, it is possible to distinguish between *active* and *passive* walking.

A passive walker exploits the potential field of gravity for walking. This kind of walk has also been defined cyclic walking. Underactuated and semi underactuated legged robots can walk in this way.

McGeer [27] demonstrated that walking can be a natural mode of a simple mechanical device, such as four bars linked together, in the shape of the skeleton of the lower half of the human body. It will be able of walking unaided down a slight line, and as long the lengths and the masses of the components are tuned correctly, it will produce a fluid and human like-walking. The produced walk is a repetitive motion generated by the passive interaction between gravity and the inertia of the system. The downhill slope can be seen as the only energy source, and joint actuation is required. However, its strength is also its weak spot: even if the gait of such as device is human-like, it is only restricted to the case of a walk down a slope.

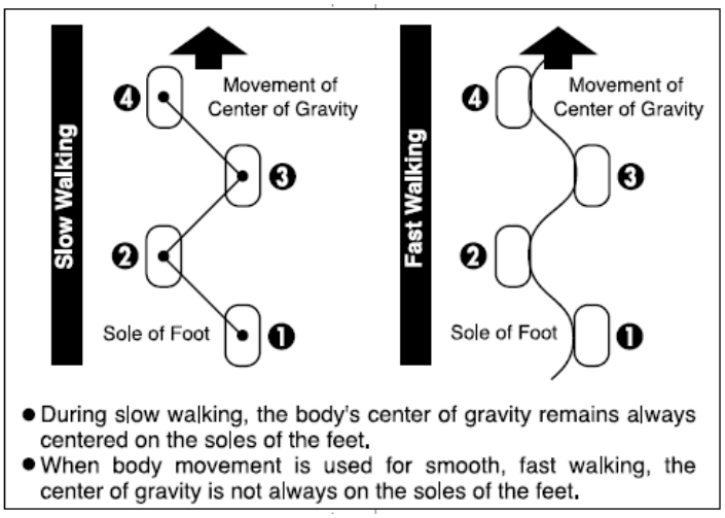


Figure 3.1: Static and dynamic gait

Instead a biped is an active walker when each one of its joints is actuated. In a passive gait the swinging feet fall on landing motion, while during an active walk the feet motion is controlled. In this way the impact force is reduced considerably, increasing stability.

For generating the active walking motion of a biped, suitable patterns should be developed taking in account the dynamics and stability during the step. In order to generate the motion patterns, a model representing the biped dynamics is required: it can be a mass distributed model and a mass concentrated model. Some of the best-known mass concentrated models are the "Inverted Pendulum" and the "Cart-Table". Later in the chapter, the features and the working principle of both models will be described.

Static and dynamic gait

When the walking motion is done at slow speed, the gait is defined *static*, when its speed is equal or higher than the normal human walking speed, the gait is defined *dynamic*.

In the static gait, the body CoM projection on the ground will be always close to the middle of the support polygon, and it is almost the same as the ZMP. Therefore, the motion of the CoM in the frontal plane is increased considerably compared to the normal walking motion, and the ZMP almost correspond to the CoM projection on the walking surface. Static torques (due to gravity) have a stronger effect during a static gait compared to dynamic torques (due to inertia).

This kind of gait is commonly used on humanoid robots of reduced scale (height less than 0.5m), which move many times with pre-planned patterns without whole-body control (WBC), because the inertial and structural effects do not cause disturbances during motion.

In a dynamic gait the projection of the CoM can be also outside the support polygon when taking a step, but the ZMP must be always inside. A dynamic gait is more representative of the human motion, in which inertial effects help to maintain biped stability during the step, because the CoM acceleration locates the ZMP closer to the middle of the ZMP. To generate a dynamic gait, a proper modeling of the biped dynamic is needed.

We focused on the inverted pendulum derived models (Kajita et al.), which are described more in detail in the next chapter.

3.2 Gait models

Many models have been proposed to generate a walking motion, each one with some advantages and certain weak points. We can differentiate between *mass distributed models*, which take into account the whole or partial body inertia and masses of the humanoid robot, and *mass concentrated models*, which simplify the whole body dynamics to the CoM motion, by concentrating the body mass in that point.

The mass distributed models are very accurate, and permit the evaluation of the whole body motion during the gait, as well its dynamics; however their computational cost is really high, so they are used in offline simulations. Examples of this type of model are the Two Masses Inverted Pendulum Mode (TMIPM) and the Multiple Masses Inverted Pendulum Mode (MMIPM) proposed by Albert et al. [3], which simulate the performance of the biped walking motion by adding pendular motions on each biped link.

Experimentally is demonstrated that it is not necessary to obtain a walking pattern considering a complex dynamic model, and that it can be used only for the evaluation of the inertial effects of the real robot. The mass concentrated models are better for computing stable of walking patterns in real-time, due to their shorter computation time.

The human CoM motion looks like the motion of the mass of an inverted pendulum. Thus in these category the dynamic models are derived from the inverted pendulum model, such as the three dimensional linear inverted pendulum model and the cart-table model proposed by Kajita et al. [21][22]. These will be described in the next sections, and they are used for generating the variables needed for our research simulations.

3.2.1 The 3D Linear Inverted Pendulum Model

The 3D Linear Inverted Pendulum Model (3D-LIPM) consists in a point mass and a massless staff. Let be $p = (x, y, z)$ the position of the mass, r the length of the staff and θ_p and θ_r the angles of the staff w.r.t. the axis x and y respectively. The pendulum position equations are:

$$x = r sin(\theta_p) \tag{3.1}$$

$$y = -r \sin(\theta_r) \tag{3.2}$$

$$z = r\sqrt{(1 - sin(\theta_r)^2 - sin(\theta_p)^2)} \tag{3.3}$$

While the motion equation is:

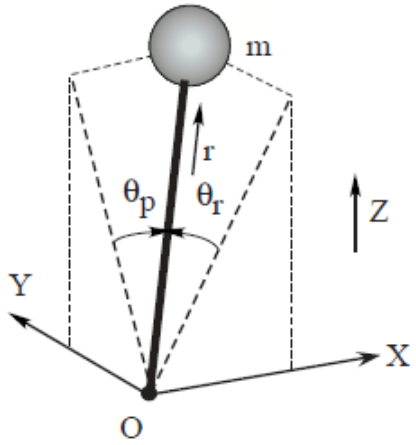


Figure 3.2: 3D pendulum

$$\begin{pmatrix}\tau_r\\ \tau_p\\ f\end{pmatrix} = m\begin{pmatrix}0 & -rcos(\theta_r) & -\frac{rcos(\theta_r)sin(\theta_r)}{\sqrt{(1-sin(\theta_r)^2-sin(\theta_p)^2)}}\\ rcos(\theta_p) & 0 & -\frac{rcos(\theta_p)sin(\theta_p)}{\sqrt{(1-sin(\theta_r)^2-sin(\theta_p)^2)}}\\ sin(\theta_p) & -sin(\theta_r) & -\sqrt{(1-sin(\theta_r)^2+sin(\theta_p)^2)}\end{pmatrix} + \\ + mg\begin{pmatrix}-\frac{rcos(\theta_r)sin(\theta_r)}{\sqrt{(1-sin(\theta_r)^2-sin(\theta_p)^2)}}\\ -\frac{rcos(\theta_p)sin(\theta_p)}{\sqrt{(1-sin(\theta_r)^2-sin(\theta_p)^2)}}\\ -\sqrt{(1-sin(\theta_r)^2+sin(\theta_p)^2)}\end{pmatrix} \tag{3.4}$$

The dynamic along the x-axis is:

$$m(z\ddot{x} - x\ddot{z}) = \frac{\sqrt{(1-sin(\theta_r)^2-sin(\theta_p)^2)}}{cos\theta_p}\tau_p + mgx \tag{3.5}$$

And the equation for the dynamics along the y-axis is:

$$m(-z\ddot{y} + y\ddot{z}) = \frac{\sqrt{(1-sin(\theta_r)^2-sin(\theta_p)^2)}}{cos\theta_r}\tau_r - mgy \tag{3.6}$$

The pendulum motion can be constrained in the xy plane, considering that the oscillations around the axis z are small compared to the others. The constrain plane is represented by the respective normal vector $(k_x, k_y, -1)$ and by its z intersection z_c as [22]:

$$z = k_x x + k_y y + z_c \tag{3.7}$$

Replacing 3.7 and its second derivative in 3.5 and 3.6 we get:

$$\ddot{x} = \frac{g}{z_c}x + \frac{k_y}{z_c}(x\ddot{y} - \ddot{x}y) + \frac{1}{mz_c}\frac{\sqrt{(1-sin(\theta_r)^2-sin(\theta_p)^2)}}{cos\theta_p}\tau_p \tag{3.8}$$

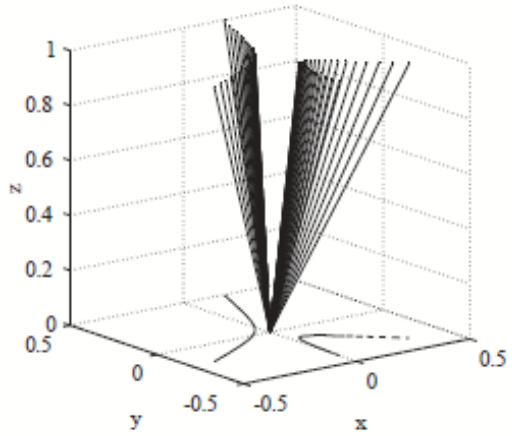


Figure 3.3: 3D Linear Inverted Pendulum Mode

$$\ddot{y} = \frac{g}{z_c}y - \frac{k_x}{z_c}(x\ddot{y} - \ddot{x}y) - \frac{1}{mz_c}\frac{\sqrt{(1 - sin(\theta_r)^2 - sin(\theta_p)^2)}}{cos\theta_r}\tau_r \tag{3.9}$$

If the above equation allow the pendulum motion in any plane and slope, constraining that to a flat plane ($k_x = k_y = 0$) gives:

$$\ddot{x} = \frac{g}{z_c}x + \frac{1}{mz_c}\frac{\sqrt{(1 - sin(\theta_r)^2 - sin(\theta_p)^2)}}{cos\theta_p}\tau_p \tag{3.10}$$

$$\ddot{y} = \frac{g}{z_c}y - \frac{1}{mz_c}\frac{\sqrt{(1 - sin(\theta_r)^2 - sin(\theta_p)^2)}}{cos\theta_r}\tau_r \tag{3.11}$$

Considering small oscillations around the axes x and y, the formulas can be simplified further:

$$\ddot{x} = \frac{g}{z_c}x + \frac{1}{mz_c}\tau_p \tag{3.12}$$

$$\ddot{y} = \frac{g}{z_c}y - \frac{1}{mz_c}\tau_r \tag{3.13}$$

These are linear equations, and the pendulum dynamic is governed only by the parameter z_c. Even in the case of a sloped constrain where ($k_x = k_y \neq 0$) we can obtain the same dynamics by applying the additional constrain:

$$\tau_x x + \tau_y y = 0 \tag{3.14}$$

We can conclude that the plane inclination never affects the horizontal motion. Solving the Eqs.3.12 and 3.13 with zero input torques (i.e. $\tau_r = \tau_p = 0$) the trajectories of the pendulum ball motion is obtained in the field of gravity, such as the two examples in Fig 3.3.

For the 3D-LIPM constrained horizontally ($k_x = k_y = 0$), we can easily calculate the ZMP position on the floor (p_x, p_y):

$$p_x = -\frac{\tau_y}{mg} \tag{3.15}$$

$$p_y = \frac{\tau_x}{mg} \tag{3.16}$$

By substituting 3.15 and 3.16 in 3.12 and 3.13

$$\ddot{x} = \frac{g}{z_c}(x + p_x) \tag{3.17}$$

$$\ddot{y} = \frac{g}{z_c}(y - p_y) \tag{3.18}$$

3.2.2 The cart-table model

To control the ZMP it has to be the output of the systems. In the previous section we have described the relationship that exists between the ZMP position and the 3D-LIMP model in Eqs. 3.17 and 3.18. Rewriting them to have the ZMP as their output we obtain:

$$p_x = x - \frac{z_c}{g}\ddot{x} \tag{3.19}$$

$$p_y = y - \frac{z_c}{g}\ddot{y} \tag{3.20}$$

A model that directly corresponds to these equations is the cart-table model. It consists in a running cart of mass m on a pedestal table whose mass is negligible. As depicted in the figure, the foot of the table is too small to let the cart stay on the edge. However, if the cart accelerates with a proper rate, the table can keep upright for a while. At this moment, the ZMP exists inside the table foot. The ZMP moments must be zero, thus the torque around the x-axis will be:

$$\tau_{zmp} = mg(x - p_x) - m\ddot{x}z_c = 0 \tag{3.21}$$

In the same way can be found the ZMP torque equation for th y-axis. The motions on x and y are uncoupled, and planar motion occurs at z_c. If the ZMP control is taken into account as a servo control problem, it is possible to put 3.19 in the form of a state variable, including the time derivative of the acceleration as input as shown by:

$$\begin{aligned} \frac{d}{dt}\begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix} &= \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} u_x \\ p_x &= \begin{pmatrix} 1 & 0 & \frac{z_c}{g} \end{pmatrix}\begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix} \end{aligned} \tag{3.22}$$

Katayama et al. [23] have proposed the optimal preview servo controller technique to obtain the CoG pattern which tracks the ZMP reference. First the system in the equations 3.22 must be discretized with a sampling time of t_s as:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ p(k) &= Cx(k) \end{aligned} \tag{3.23}$$

where:

$$\begin{aligned} x(k) &= \begin{pmatrix} x(kT) & \dot{x(kT)} & \ddot{xT} \end{pmatrix}^T, \\ u(k) &= u_x(kT), \\ p(k) &= p_x(kT), \\ A &= \begin{pmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{pmatrix} \\ C &= \begin{pmatrix} 1 & 0 & -\frac{z_c}{g} \end{pmatrix} \end{aligned} \tag{3.24}$$

In this way is obtained the state-space representation of the dynamics of the cart-table model. With the given ZMP reference $p^{ref}(k)$, the performance index is specified as:

$$J = \sum_{i=k}^{n} (Q_e e(i)^2 + \Delta x^T Q_x \Delta x(i) + R\Delta u^2) \tag{3.25}$$

where

$$\begin{aligned} e(i) = p(i) - p^r f(i) &\text{ is a servo error} \\ Q_e, R > 0, Q_x &\text{ is a symmetric non-negative definitive matrix} \\ \Delta x(k) = x(k) - x(k-1) &\text{ is the incremental state vector} \\ \Delta x(k) = u(k) - u(k-1) &\text{ is the incremental input} \end{aligned} \tag{3.26}$$

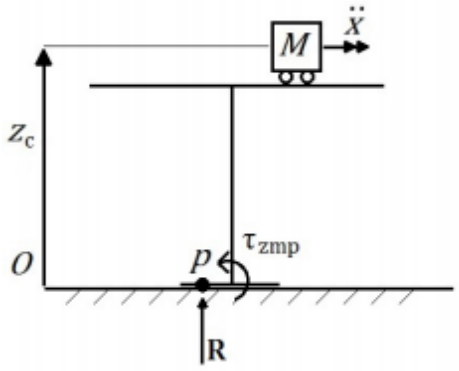


Figure 3.4: Cart-Table-model

When the ZMP reference can be previewed for N_L step future, the optimal controller proposed by Kajita et al. which minimizes the performance index J 3.25 at every sampling time is:

$$u(k) = -G_i \sum_{i=0}^{k} e(k) - G_x x(k) - \sum_{j=1}^{N_L} G_p(j) p^{ref}(k+j) \tag{3.27}$$

The gains G_i, G_x and $G_p(j)$ are calculated from the weights Q_e, Q_x and R, and from the systems parameter of Eq. 3.23.

The problem solved by Kajita et al. [22] is a discrete-time, infinite-horizon LQR problem. This method has an high computational cost, but generalized the specification as the optimization of a quadratic cost which could balance ZMP tracking against CoM acceleration, giving a more robust CoM output. Three terms are included in the preview controls (Eq. 3.27): the integral action on the tracking error, the state feedback and the preview action on the future reference.

3.2.3 Discussion about the LIMP and cart-table models

In this paragraph, some remarks will be made order to highlight some important concept that have determined the choices regarding the simulations approaches.

The LIPM model is a good approximation of the humanoid robot motion dynamics which results in a natural motion of the CoM, but has not a direct link with the position of the ZMP on the ground. It is true that the acceleration of the CoM is strictly related to its position and to that of the ZMP, as reported by 3.17 and 3.18, but the inverse equations that compute the position of the ZMP from the acceleration and position of CoM corresponds to a different model, the cart-table. Furthermore, there is a discontinuity during the double support phase, when changing stance leg. The jerk (the time derivative of the acceleration) of the CoM is not taken into account, thus we can have low performances at high speeds, when it influences more the biped dynamics.

From the cart-model is possible to derive a controller based on the ZMP preview and granting a balanced CoM motion like the 3.27. The preview control optimize also the jerk of the CoM, and the cart-table is continuous all the time, no matter the gait phase. For these reasons the cart-table model is usually preferred for the development of a controller.

There are other approaches for solving the cart-table optimal control problem, like the solution proposed by Choi et al. in [8]. The discrete-time LQR problem proposed by Kajita [21][22] has been of great impact: its greater contribution to bipedal control theory is that he developed a method that takes into account the ZMP preview as well the quality of the CoM motion dynamics. From its method, have been derived other approaches for optimal controllers of the ZMP preview, such as the Russ Tedrake et al. [41] optimal ZMP tracking controller, which solves the continuous-time LQR problem in an iterative manner without taking into account the jerk. This last approach is used in this thesis for the CoM trajectory generation.

Artificial Neural Networks

Neural networks are computing systems coming from the field of machine learning, which in turn is part of the artificial intelligence field. Machine learning's studies aim to develop computer algorithms that, thanks to experience, can solve and interpret situations and conditions that the system has never experienced before.

And that is what neural networks do. Using some known data that we call training data[1], they autonomously build a model that can manage and react appropriately to new and never seen data if the training data are enough to fulfill the network's knowledge.

Neural networks are used to build those models that are analytically difficult to calculate, which is why we rely on instead on training data to generate them.

4.1 General structure

4.1.1 Neurons, connections and layers

The structure of an artificial neural network is inspired by animal brains structures, in which we can find nodes called neurons.

Connected, they can build a more or less complex net in which signals, typically numbers, can propagate from a side to the other (feed-forward), just like in a human brain, where we have a collection of neurons that constitute a synaptic system.

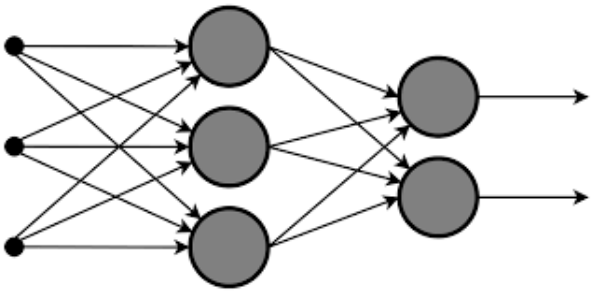


Figure 4.1: Neurons and connections

This model is organized into layers, and we can find three types of them:

- The *input layer*, in which our signals can enter the model.

[1]It is a part of the data set intended for training. There are also test and validation data

- The *output layer*, where there are our output signals elaborated from the network.
- The *hidden layer*, which are all the layers that are located between input and output ones.

It can also be helpful to enumerate them from first to last. To adapt the behavior of the network to multiple situations and couples of input-output data, the layer can be customized.

The number of inputs can be different from the outputs one, and the number of hidden layers will be proportional to the level of complexity needed for the system. The more hidden layer we have, and more complex will be the model. Starting from scratch it is possible to define three types of neural network structure:

- The *single layer neural network*, where we have just input and output layer, without the hidden ones.
- The *shallow layer neural network*, in which we can find one hidden layer.
- The *deep neural network*, which nowadays represents the standard way of thinking about a network, since this type regroup all the network with more then one hidden layer.

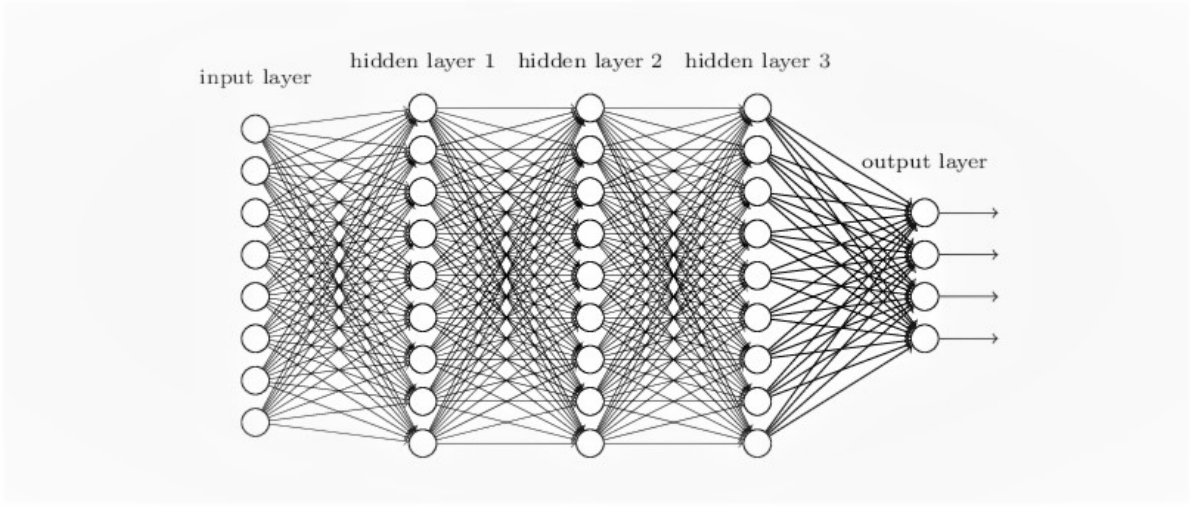


Figure 4.2: Input, output and hidden layers of a multilayer perceptron network

As shown in Fig. 4.2, each successive neuron in the layers structure is connected to all the neurons of the previous layer.

4.1.2 Weights, biases and activation function

Any neuron in the net, except for the input ones, can contribute from each neuron in the previous layer. The following three features influence this interaction between neurons and layers:

- The *weights*, which define the level of contribution of each neuron to neurons of the successive layer.
- The *biases* define how much high the threshold of neuron needs to be activated.
- The *activation function* scales the output of a neuron. Thus it depends on the application. Typically is a sigmoid function[2].

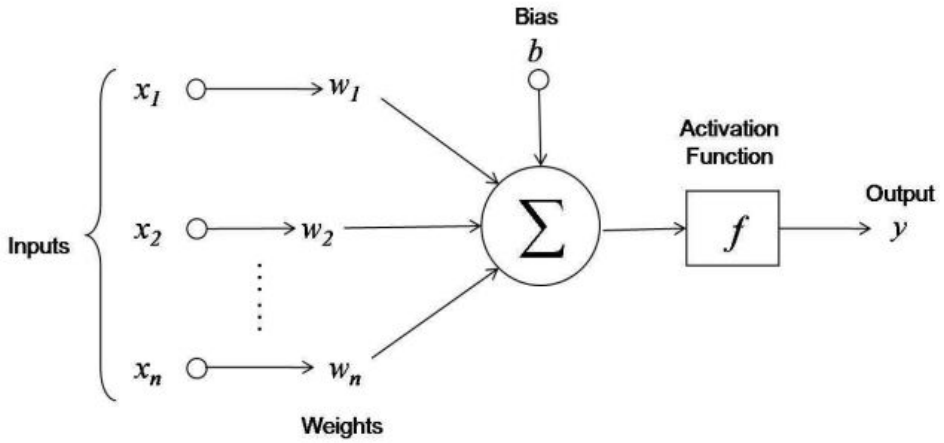


Figure 4.3: Working principle of a neurons

In Fig. 4.3, it is shown how to combine together all the ingredients that constitute the behavior of a single neuron. As already mentioned before, each neuron is influenced by the neurons of previous layer, and each connection has its corresponding weight.

As a result we have that entering in a neuron. We find the activation function processing the weighted sum of the previous layer neuron's outputs, plus the correlate bias. The activation function determines the shape of the output. For example, using a sigmoid function our output will be a number between zero and one (3Blue1Brown,[1] and M. Nielsen,[30]).

$$y = f(x_1 w_1 + x_2 w_2 + ... + x_n w_n + b) \tag{4.1}$$

Depending on the application it could be better to work with positive number, negative or both. In general it is useful to use small numbers because computation for the network will result easier and so faster. In any case, after the process, output signals can always be reshaped.

4.2 Working principles

4.2.1 Cost function and gradient descent

To recap, the last section explains that it's possible to see the neural network as a function, in which weights and biases are its coefficients and its parameters.

[2] $\frac{1}{1+e^{-x}}$, squeezes the input between zero and one

How to find their values? To find them we train the network using the backpropagation rules. But before it is necessary to introduce the concepts of cost function, gradient descent and show all the steps needed to teach the network how to learn.

Cost function

A cost function in machine learning and mathematical optimization is a function that tells us how much wrong or right the created model is interpreting the input-output relationship. In other words the cost function evaluates the performance of a model returning a number(M. Nielsen,[32]).

If we define y_i as the output vector of our system and $\hat{y}_i$ as the desired output vector, where i goes from one to n and n is the number of outputs, the cost function will be defined as the sum of the squares of the differences between y_i and $\hat{y}_i$.

$$Cost = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{4.2}$$

The higher the values of $Cost$, the worst the model performances, so our goal is to lead the model to minimize the cost function. To achieve the minimum, we rely on the gradient's information (M. Nielsen,[29]).

Gradient descent

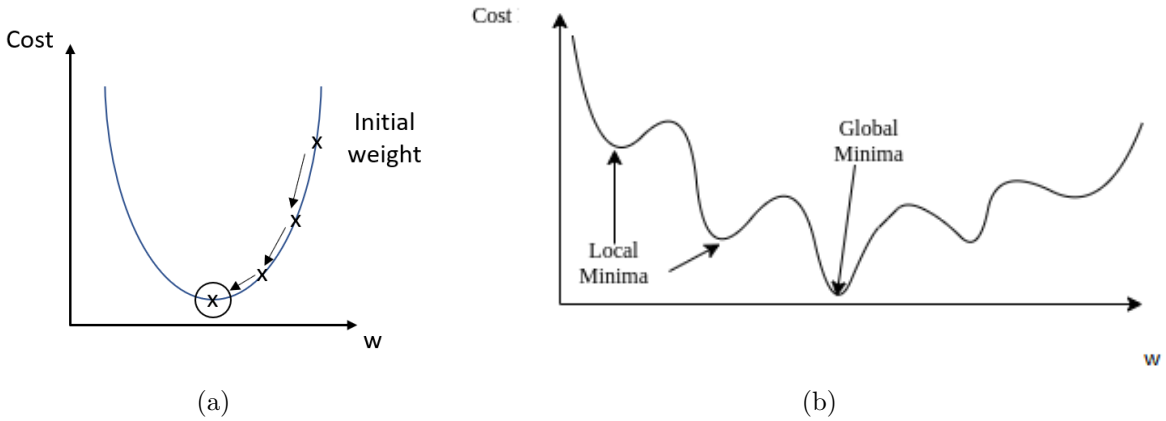
The gradient of a function of a scalar p, from a computational point of view, is the partial derivative in the point p w.r.t. all the variables contained in the function $(x_1...x_n)$. Typically for its notation we use the nabla symbol ∇.

$$\nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_n}(p) \end{bmatrix} \tag{4.3}$$

Besides how it's computed, the gradient is nothing more than a vector. This vector gives the direction in which the function is going to increase w.r.t to the point p. In the previous paragraph, we have discussed the cost function and the necessities of minimizing it. Thus, since we want to find minima, instead of being interested in where the function will increase, we want to know where it will decrease.

If we find where the function is decreasing we can lead our way to the minima, and it is possible to do it by taking the negative of the gradient. This is the concept of gradient descent (3Blue1Brown,[2]). To better understand the idea we can consider for the moment a model with just one input; in this case it's possible to map the function on a plane with two axes, just like in Fig. 4.4a.

First of all, we randomly choose the initial weight, compute the negative gradient, and then move along. To move from where we are, of course, we have to change the values of the weights. Since in this particular case the gradient is the slope of the function, we can say that, moving away from the initial position. The slope starts to get flatter and flatter. The flatter it becomes, and the smaller our movement must be. Otherwise, we could miss the minima or its surrounding area too much.



(a) (b)

Figure 4.4: (a) Gradient descent (b) Global and local minima

It is also necessary to notice that we have no guarantee that this solution is the best one obtainable for the optimization problem once we have reached a minimum. If it is true, we have stumbled upon a local minima, a solution that is not bad but maybe not as good as the global minima solution could be. The global minima is the best result obtainable (Fig. 4.4b).

Depending on the solving algorithm, it's possible to partially overcome the problem of being stuck in local minima and finding the global one. But this task is tough to manage, and lots of researches still need to be done. Now let's turn back to the neural network where we have as many weights and biases instead of one input. The following section will present a step-by-step procedure for training a neural network and how the weights of a single-layer structure can be possibly adjusted.

4.2.2 Learning procedure

Before entering into the steps we must follow to perform proper training, it is essential to say that there are many methods to train a neural network. For our purpose we can use the supervised learning method, which consists of training the network forcing it to return the desired output. This means that our problem solution is already known, and it can be used for the learning process (F. Nuruzzaman, [34]).

Now we can resume the training procedure through the following steps (part of this has already been mentioned in the previous subsection):

- *Step 1 :* Randomly initialize the weights and biases.
- *Step 2 :* Calculate the error between desired and obtained outputs (cost function).
- *Step 3 :* Calculate the weights updates according to the error through the gradient.
- *Step 4 :* Adjust the weighs updates.
- *Step 5 :* Repeat from 2 to 4 for all the training data, an epoch[3].

[3]An epoch is a full round of training data

- *Step 6 :* Repeat from 2 to 5 until the error has reached an acceptable level.

Focusing on *Step 3* and *Step 4*, we are going discus the learning rules[4]. But before, let's discuss how it works in the simplest situation.

Delta rule

Considering a single layer neural network, the way it computes the updates of weights and adjusts them can be resumed in the Delta rule[5].

$$w_{ij} \leftarrow w_{ij} + \alpha e_i x_j \tag{4.4}$$

The left side represents the updated weight in the formula, while the right side is the weight that has to be updated, plus the computation for updating it. w_{ij} is the weight associated with the j-th input neuron and i-th output one. α is the learning rate, its value is used to tune how far from the previous weight we want to go. If α is too high the output wanders around the expected solution, while if it is too low it can fail to reach the desired output. Typically its value lay between zero and one. e_i is the error of i-th output neuron, and x_j is the j-th input of one.

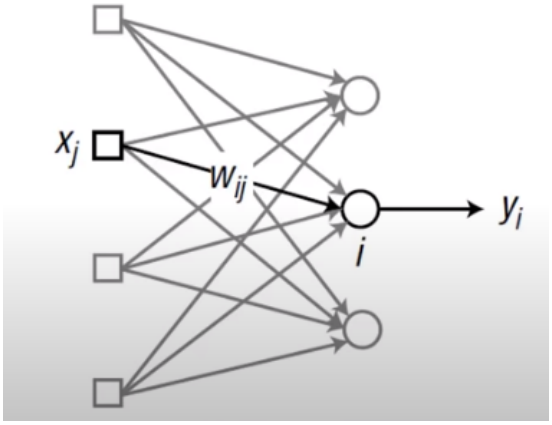


Figure 4.5: Single layer architecture example

In this case, we are considering no biases and a linear activation function. For non-linear activation functions we speak of the generalized delta rule.

$$\delta_i = \phi'(v_i)e_i \tag{4.5}$$

In the equation, δ_i is equal to the derivative of the activation function ϕ of v_i, which is the weighted sum of i-th neuron, times the error. With this notion we can obtain the generalized delta rule.

$$w_{ij} \leftarrow w_{ij} + \alpha \delta_i x_j \tag{4.6}$$

Note that in equation 4.6, if we have a linear activation function δ_i is equal to e_i. For a linear function ϕ' is equal to one. Thus we return to equation 4.4

[4]It is a set of equations that defines how the network changes its weights and biases

[5]the learning rule of the single layer structure

4.2.3 Backpropagation rules

The delta rule is not enough to compute the weights updates for all the layers with deep neural network structure. To do that, we can use the backpropagation algorithm. To recap the situation, we want to minimize the cost function by adjusting weights and biases with this algorithm. Thus, it is necessary to evaluate the partial derivative of the cost function w.r.t. weights and biases to put it simply, since the mathematics behind the backpropagation is complex, in this subsection will be presented the four equations of the algorithm in a simple way and without going into details, since this is not the thesis focus.

Before describing the equations, we have to define δ_j^l as the error of j-th neuron in l-th layer. By the way, its value is also related to the partial derivative of the cost function w.r.t. weights and biases [31].

Equation for the error in the output layer

$$\delta_j^l = \nabla_a C \odot \sigma'(z)_j^l \tag{4.7}$$

Here $\nabla_a C$ is equal to $\partial C / \partial a_j^l$, where C is the cost function, and a_j^l is the weighted sum of the outputs, which gives the activation of j-th neuron. The other term is the derivative of the activation function σ of z_j^l, which is the weighted sum input of the j-th neuron in the l-th layer. These two elements are multiplied by the Hadamard product $\odot$ which means that is an element wise product (in the equation the elements are vector by the way). So trying to get its meaning, we can say that the error δ is equal to the rate of change of the cost function w.r.t. the output activation multiplied by how fast the activation function changes at the input.

Equation for the error in terms of the error in the next layer

$$\delta_j^l = ((w_j^{l+1})^T \delta_j^{l+1}) \odot \sigma'(z)_j^l \tag{4.8}$$

In the equation $(w_j^{l+1})^T$ is the transpose of the weighted matrix of the next layer, which can be thought of as moving the error backward through the network. The other element, the same as the previous equation, moves the error backward through the activation function. The combination of the two elements give us the measure of the error at output and input weighted sum for the l-th layer.

Combining eq. 4.7 and eq. 4.8 it is possible to compute the error δ for any layer in the network moving backward.

Equation for the rate of change of the cost with respect to any bias in the network

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \tag{4.9}$$

Here we can just conclude that δ is equal to the rate of change of the cost function w.r.t. the bias. So the error δ is evaluated at the same neuron of the bias b.

Equation for the rate of change of the cost with respect to any weight in the network

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1}\delta_j^l \tag{4.10}$$

The rate of change of the cost function w.r.t. the weight w_{jk}^l of the l-th layer, from the k-th neuron to the j-th neuron is equal to the activation coming from the last neuron times the error in the neuron we are considering.

An important thing that we can notice from these equations is that to have an efficient and fast learning algorithm we have to be able to act on the neurons with the higher values. Because the equations show us that the weights updates depend on the value of the activation of a neuron. finding the higher ones and acting on them will have a stronger effect than the effect that the neurons with smaller activation values would have. By doing that we can modify the weights more freely and faster.

4.2.4 Training methods and performances

Training methods

As already mentioned there exist several ways to train a network. Here we present the most used ones.

- The *SGD method*, which stay for stochastic gradient descent, is a fast method since we compute and update the weight for each training trial. But its behavior is also random and so not much convenient.
- The *Batch method* calculates the weights updates, but the update is computed with the average of updates. As a result we update the weights once in an epoch.

  $$\delta w_{ij} = \frac{1}{N}\sum_{k=1}^{N}\delta w_{ij}(k) \tag{4.11}$$

  N is the total number of data. The formula stabilizes the training, but the flaw of this method is that it takes a long time.
- The *Mini Batch method* is the combination of SGD and Batch methods. We use the batch only for a section of the training data at a time and not for all the data at once. This method inherits the speed of SGD and the stability of batch.

Performances

As we have seen previously, the rules behind network behavior are complex to figure out, and depending on the model we are trying to build, poor performances can occur [33]. The most evident ones are:

- The *vanishing gradient*: it happens when the error propagation fails to reach the farthest neurons and updates the weights. Nowadays, some more robust architectures w.r.t. this kind of problem, such as LSTM[6] neural networks.

[6]Long Shot Term Memory neural network

- The *overfitting*: in a certain case we can have really huge and complex structures and it is impossible to satisfy each requirement of each neuron. In order to solve it, we satisfy only some of the neurons request.
- The *computational load*: the training time of big networks can exponentially explode. To improve it we can use GPU instead of CPU.

Those are just some of the problems that can affect construction and behavior of a neural network and a lots of issues are still waiting to be solved or improved.

Part II

Simulations

Experimental datasets

In order to conduct the simulations concerning kinematics, dynamics and neural networks, two different datsets, containing recordings of biometric, kinematic and kinetic signals during various gait trials, were used. One of them was provided by the Biolab laboratory of the Department of Electronics and Telecommunications of the Politecnico di Torino. The other is the result of an experimental protocol approved by the Institutional Review Board at the University of Texas at Dallas. It can be found online, as open source on the IEEE Dataport website.

5.1 Biolab dataset

Gait data were recorded by means of a multichannel system specifically developed for clinical gait analysis (STEP32, Medical Technology, Italy). the signal recordings were acquired from twenty-two healthy subjects, free from lower limb damage and neuromuscular disease. twenty of the subjects had a dominant right leg and the last two had a dominant left leg. All the subjects walked barefoot for 5 minutes at self-selected speed, back and forth on a 10 meters straight walkway [15]. As just mentioned, in the main study data was acquired from twenty-two subjects, in this dataset there are three of them.

Author: V.Agostini, M.Ghislieri.

Date: 28/10/2020.

Data format: struct *.m *.mat

Number os subjects: 3.

Sampling frequency: 2000 $[Hz]$.

Subject informations :

1. Subject: description of the volunteer.
2. Info: description of the walking task.
3. Cadence: average cadence $[cycles/min]$.
4. Age: subject's age $[years]$.
5. Height: subject's height $[cm]$.
6. Weight: subject's weight $[kg]$.

7. Gender: Male or Female.

Signals :

1. BasoLeft: an array with basographic signal acquired of the left foot, by means a switch (size: 10 mm x 10 mm x 0.5 mm, activation force: 3 N).
2. BasoRight: an array with basographic signal acquired of the right foot, by means a switch (size: 10 mm x 10 mm x 0.5 mm, activation force: 3 N).
3. KneeLeft: an array with signal acquired by the left knee electrogoniometer [$degrees$].
4. KneeRight: an array with signal acquired by the right knee electrogoniometer [$degrees$].
5. HipLeft: an array with signal acquired by the left hip electrogoniometer [$degrees$].
6. HipRight: an array with signal acquired by the right hip electrogoniometer [$degrees$].
7. KneeLeft_Mean: an array with the mean of signal acquired by the left knee electrogoniometer [$degrees$].
8. KneeRight_Mean: an array with the mean of signal acquired by the right knee electrogoniometer [$degrees$].
9. HipLeft_Mean: an array with the mean of signal acquired by the left hip electrogoniometer [$degrees$].
10. HipRight_Mean: an array with the mean of signal acquired by the right hip electrogoniometer [$degrees$].
11. sMEG: an array with filtered sEMG signals acquired from 10 muscles of the right lower limb [μV]. By means of active probes (configuration: single differential, size: 19 mm x 17 mm x 7 mm, Ag-disks diameter: 4 mm, inter-electrode distance: 12 mm, gain: variable in the range from 60 dB to 86 dB). In particular, each column represent the electromyographic activity of left Tibialis Anterior (TA), left Lateral Gastrocnemius (LGS), left Rectus Femoris (RF), left Lateral Hamstring (LH), left Gluteus Medius (GMD), right Tibialis Anterior (TA), right Lateral Gastrocnemius (LGS), right Rectus Femoris (RF), right Lateral Hamstring (LH), and right Gluteus Medius (GMD), respectively.
12. sEMG_Activ: an array with muscle activation signals computed by means of the muscle activity detector [μV]. In particular, each column represent the electromyographic activity of left Tibialis Anterior (TA), left Lateral Gastrocnemius (LGS), left Rectus Femoris (RF), left Lateral Hamstring (LH), left Gluteus Medius (GMD), right Tibialis Anterior (TA), right Lateral Gastrocnemius (LGS), right Rectus Femoris (RF), right Lateral Hamstring (LH), and right Gluteus Medius (GMD), respectively.

Matlab struct organization :

Control_(subject).(datatype).(variable)

1. subject (X): ranges from 1 to 3, is the number of subjects.

2. datatype:
    - Subject (*char*).
    - Info (*char*).
    - Cadence (*double*).
    - Age (*double*).
    - Height (*double*).
    - Weight (*double*).
    - Gender (*char*).
    - Signals (*struct*).
3. variable (only for signals datatype):
    - sEMG ($10x367140$ *double*).
    - sEMG_active ($10x367140$ *double*).
    - KneeLeft ($1x367140$ *double*).
    - HipLeft ($1x367140$ *double*).
    - KneeLeft_Mean ($1000x1$ *double*).
    - HipLeft_Mean ($1000x1$ *double*).
    - KneeRight ($1x367140$ *double*).
    - HipRight ($1x367140$ *double*).
    - KneeRight_Mean ($1000x1$ *double*).
    - HipRight_Mean ($1000x1$ *double*).
    - BasoLeft ($1x367140$ *double*).
    - BasoRight ($1x367140$ *double*).

5.2 Incline experiment (IEEE) dataset

Ten able-bodied subjects walked at steady speeds and inclines on a Bertec instrumented treadmill for one minute per trial. Each subject walked at every combination of the speeds 0.8 m/s, 1 m/s, and 1.2 m/s and inclines from -10 *degrees* to +10 *degrees* at 2.5 *degree* increments, for a total of 27 trials. During each trial, a 10-camera Vicon motion capture system recorded leg kinematics[46], while force plates in the Bertec treadmill recorded ground reaction forces [5][6], and a Delsys Trigno EMG system recorded muscle activation of the rectus femoris, biceps femoris, tibialis anterior, and gastrocnemius [18][13].

For marker signals the world frame is positioned at the base of the treadmill. The data were saved and organised with two different temporal criteria, *Continuous* and *Gaitcycle*. In the former, the recording takes place normally and its progression is marked by the succession of temporal instants. In *Gaitcycle*, on the other hand, the recording is not time dependent, but indeed, is defined in terms of percentage with respect to the walking cycle.

Author: K. Embry, D. Villarreal, R. Macaluso, and R. Gregg.

Date: 07/16/2020.

Data format: struct *.m *.mat

Number os subjects: 10.

Sampling frequency: 100 $[Hz]$.

Continuous

Subject Details :

1. Gender: 1 for male, 2 for female.
2. Age: subject's age $[years]$.
3. Height: subject's height $[mm]$.
4. Weight: subject's weight $[kg]$.
5. Leg Length: subject's legs length $[mm]$.

Trial Details :

1. Speed: treadmill's speed $[m/s]$.
2. Incline: treadmill's inclination $[degrees]$.

Signals :

1. Time: an array with the time since the beginning of the experiment for every frame $[s]$.
2. Kinematics:
   - Markers: an array with world-frame positions of all motion-capture markers, located on the Anterior Superior Iliac Spine (asi), Posterior Superior Iliac Spine (psi), the thigh, knee, tibia, ankle, heel, and toe of both legs. Results are split into three component directions: x, y, or z $[mm]$.
   - Joint angles: an array with the joint angles for the pelvis, hip, knee, ankle, and foot as calculated by Vicon Plug-in Gait (Vicon, Oxford, UK). Results are split into three component directions: x, y, or z $[degrees]$.
3. EMG data:
   - EMG: delsys EMG sensors (Model:Trigno wireless system, Delsys, Natick, MA) were attached to the rectus femoris (RF), biceps femoris (BF), tibialis anterior (TA), and gastrocnemius (GC). The EMG signals have been rectified and low-pass filtered (fc=40 Hz) with a zero-phase digital filter (MathWorks, Natick, MA)$[V]$.
   - Accelerations: each Delsys EMG also contains a 3-axis accelerometer that reports an acceleration vector in the local frame. Results are split into three component directions: x, y, or z $[m/s^2]$.
4. Kinetics:
   - Joint power: an array of the power generated by each joint, determined by Plug-In Gait (Vicon) $[W/kg]$.

- Joint force: an array of the force applied at each joint, determined by Plug-In Gait (Vicon). Results are split into three component directions: x, y, or z $[N/kg]$.
- Joint moment: an array of the moment generated by each joint, determined by Plug-In Gait (Vicon). Results are split into three component directions: x, y, or z $[Nmm/kg]$.
- Forceplate force: a 3D force vector from force plates in the split belt instrumented treadmill (Bertec, Columbus, OH). These signals have been low-pass filtered (fc=40 Hz) with a zero-phase digital filter (MathWorks). Results are split into three component directions: x, y, or z $[N]$.
- Forceplate moment: a 3D moment vector from force plates in the split belt instrumented treadmill (Bertec). Results are split into three component directions: x, y, or z $[Nmm]$.
- Forceplate cop: the center of pressure location (world-frame) from force plates in the split belt instrumented treadmill (Bertec). Results are split into three component directions: x, y, or z $[mm]$.

Matlab struct organization :

Continuous.(subject).(trial).(datatype).(leg).(variable)

1. subject (ABXX): ranges from 01 to 10, is the number of subjects.
2. trial:
   - subjectdetails ($6x3cell$):
     - Gender (*double*).
     - Age (*double*).
     - Height (*double*).
     - Weight (*double*).
     - Left Leg Length (*double*).
     - Right Leg Length (*double*).
   - (sXXi/dYY): XX is the speed of the trial (0x8, 1 and 1x2[1] $[m/s]$). YY is the incline of walking (10, 7x5, 5, 2x5 and 0 $[degrees]$), "i" stands for incline the "d" for decline.
3. datatype:
   - description ($2x3cell$):
     - Speed (*double*).
     - Incline (*double*).
   - kinematics :
     - markers : cartesian signals (kinematiks.marker).
     - jointangles: angular signals (kinematics.jointangles).
   - time ($6000x1$ *double*).
   - kinetics:

[1]Decimal point are replace with "x"

- jointpower: power generated signals (kinetics.jointpower).
- jointforce: applied force signals (kinetics.jointforce).
- jointmoment: momente generated signals (kinetics.jointmoment:).
- forceplate: force plate signals (kinetics.forceplate).

- emgdata:
  - emg: EMG signals (emgdata.emg).
  - accel: acceleration vector signals (emgdata.accel).

4. leg:
   - right: indicates the right leg.
   - left: indicates the left leg.
5. variable (depends on datatype, see Fig. 5.1):
   - marker:
     - asi ($6000x3$ *double*).
     - psi ($6000x3$ *double*).
     - thigh ($6000x3$ *double*).
     - knee ($6000x3$ *double*).
     - tibia ($6000x3$ *double*).
     - ankle ($6000x3$ *double*).
     - heel ($6000x3$ *double*).
     - toe ($6000x3$ *double*).
   - joint:
     - hip ($6000x3$ *double*).
     - knee ($6000x3$ *double*).
     - ankle ($6000x3$ *double*).
     - pelvis ($6000x3$ *double*).
     - foot ($6000x3$ *double*).
   - muscle:
     - RF ($6000x3$ *double* for accelerations $6000x1$ *double* for EMG).
     - BF ($6000x3$ *double* for accelerations $6000x1$ *double* for EMG).
     - TA ($6000x3$ *double* for accelerations $6000x1$ *double* for EMG).
     - GC ($6000x3$ *double* for accelerations $6000x1$ *double* for EMG).
   - forceplate:
     - force ($6000x3$ *double*).
     - moment ($6000x3$ *double*).
     - cop ($6000x3$ *double*).

Gaitcycle

For the gait cycle standard deviations and signal averages were also measured for all muscles considered for EMG, and for all spatial directions for all other kinematic and kinetic recordings.

Subject Details :

1. Gender: 1 for male, 2 for female.
2. Age: subject's age [$years$].
3. Height: subject's height [mm].
4. Weight: subject's weight [kg].
5. Leg Length: subject's legs length [mm].

Trial Details :

1. Speed: treadmill's speed [m/s].
2. Incline: treadmill's inclination [$degrees$].

Signals :

1. Steps out: contains a vector of strides that we have identified to be outliers, as defined by having kinematics 3 standard deviations from the mean.
2. Cycles time: an array with the same dimensions as the other Gaitcycle data that indicates the time since the beginning of the corresponding stride [s].
3. Cycles frame: a vector that indicates what frame each heel strike occurred on.
4. Kinematics:
   - Markers: an array with world-frame positions of all motion-capture markers, located on the Anterior Superior Iliac Spine (asi), Posterior Superior Iliac Spine (psi), the thigh, knee, tibia, ankle, heel, and toe of both legs. Results are split into three component directions: x, y, or z [mm].
   - Joint angles: an array with the joint angles for the pelvis, hip, knee, ankle, and foot as calculated by Vicon Plug-in Gait (Vicon, Oxford, UK). Results are split into three component directions: x, y, or z [$degrees$].
5. EMG data:
   - EMG: delsys EMG sensors (Model:Trigno wireless system, Delsys, Natick, MA) were attached to the rectus femoris (RF), biceps femoris (BF), tibialis anterior (TA), and gastrocnemius (GC). The EMG signals have been rectified and low-pass filtered (fc=40 Hz) with a zero-phase digital filter (MathWorks, Natick, MA)[V].
   - accelerations: Each Delsys EMG also contains a 3-axis accelerometer that reports an acceleration vector in the local frame. Results are split into three component directions: x, y, or z [m/s^2].
6. Kinetics:
   - Joint power: an array of the power generated by each joint, determined by Plug-In Gait (Vicon) [W/kg].
   - Joint force: an array of the force applied at each joint, determined by Plug-In Gait (Vicon). Results are split into three component directions: x, y, or z [N/kg].

- Joint moment: an array of the moment generated by each joint, determined by Plug-In Gait (Vicon). Results are split into three component directions: x, y, or z $[Nmm/kg]$.
- Forceplate force: a 3D force vector from force plates in the split belt instrumented treadmill (Bertec, Columbus, OH). These signals have been low-pass filtered (fc=40 Hz) with a zero-phase digital filter (MathWorks). Results are split into three component directions: x, y, or z $[N]$.
- Forceplate moment: a 3D moment vector from force plates in the split belt instrumented treadmill (Bertec). Results are split into three component directions: x, y, or z $[Nmm]$.
- Forceplate cop: the center of pressure location (world-frame) from force plates in the split belt instrumented treadmill (Bertec). Results are split into three component directions: x, y, or z $[mm]$.

Matlab struct organization for signals :

Gaitcycle.(subject).(trial).(datatype).(leg).(variable)

1. subject (ABXX): ranges from 01 to 10, is the number of subjects.
2. trial:
   - subjectdetails ($6x3cell$):
     - Gender ($double$).
     - Age ($double$).
     - Height ($double$).
     - Weight ($double$).
     - Left Leg Length ($double$).
     - Right Leg Length ($double$).
   - (sXXi/dYY): XX is the speed of the trial (0x8, 1 and 1x2[2] $[m/s]$). YY is the incline of walking (10, 7x5, 5, 2x5 and 0 $[degrees]$), "i" stands for incline the "d" for decline.
3. datatype:
   - description ($2x3cell$):
     - Speed ($double$).
     - Incline ($double$).
   - kinematics :
     - markers : cartesian signals (kinematiks.marker).
     - jointangles: angular signals (kinematics.jointangles).
   - stepsout: vector of strides that we have identified to be outliers.
   - cycle:
     - time (cycles.time, $150xM$[3] $double$).
     - frame (cycles.frame, $1x(M+1)$ $double$).

[2]Decimal point are replace with "x"

[3]M is the number of strides

- kinetics:
  - jointpower: power generated signals (kinetics.jointpower).
  - jointforce: applied force signals (kinetics.jointforce).
  - jointmoment: momente generated signals (kinetics.jointmoment:).
  - forceplate: force plate signals (kinetics.forceplate).
- emgdata:
  - emg: EMG signals (emgdata.emg).
  - accel: acceleration vector signals (emgdata.accel).

4. leg:
   - right: indicates the right leg.
   - left: indicates the left leg.
5. variable (depends on datatype, see Fig. 5.2):
   - marker:
     - asi ($150xM$ *double* for the three component directions $150x1$ *double* for mean and standard deviation).
     - psi ($150xM$ *double* for the three component directions $150x1$ *double* for mean and standard deviation).
     - knee ($150xM$ *double* for the three component directions $150x1$ *double* for mean and standard deviation).
     - tibia ($150xM$ *double* for the three component directions $150x1$ *double* for mean and standard deviation).
     - ankle ($150xM$ *double* for the three component directions $150x1$ *double* for mean and standard deviation).
     - heel ($150xM$ *double* for the three component directions $150x1$ *double* for mean and standard deviation).
     - toe ($150xM$ *double* for the three component directions $150x1$ *double* for mean and standard deviation).
   - joint:
     - hip ($150xM$ *double* for the angular position $150x1$ *double* for mean and standard deviation).
     - knee ($150xM$ *double* for the angular position $150x1$ *double* for mean and standard deviation).
     - ankle ($150xM$ *double* for the angular position $150x1$ *double* for mean and standard deviation).
     - pelvis ($150xM$ *double* for the angular position $150x1$ *double* for mean and standard deviation).
     - foot ($150xM$ *double* for the angular position $150x1$ *double* for mean and standard deviation).
   - muscle:
     - RF ($150xM$ *double* for the three component directions and EMG $150x1$ *double* for mean and standard deviation).

  - BF ($150xM$ *double* for the three component directions and EMG $150x1$ *double* for mean and standard deviation).
  - TA ($150xM$ *double* for the three component directions and EMG $150x1$ *double* for mean and standard deviation).
  - GC ($150xM$ *double* for the three component directions and EMG $150x1$ *double* for mean and standard deviation).
- forceplate:
  - force ($150xM$ *double* for the three component directions $150x1$ *double* for mean and standard deviation).
  - moment ($150xM$ *double* for the three component directions $150x1$ *double* for mean and standard deviation).
  - cop ($150xM$ *double* for the three component directions $150x1$ *double* for mean and standard deviation).

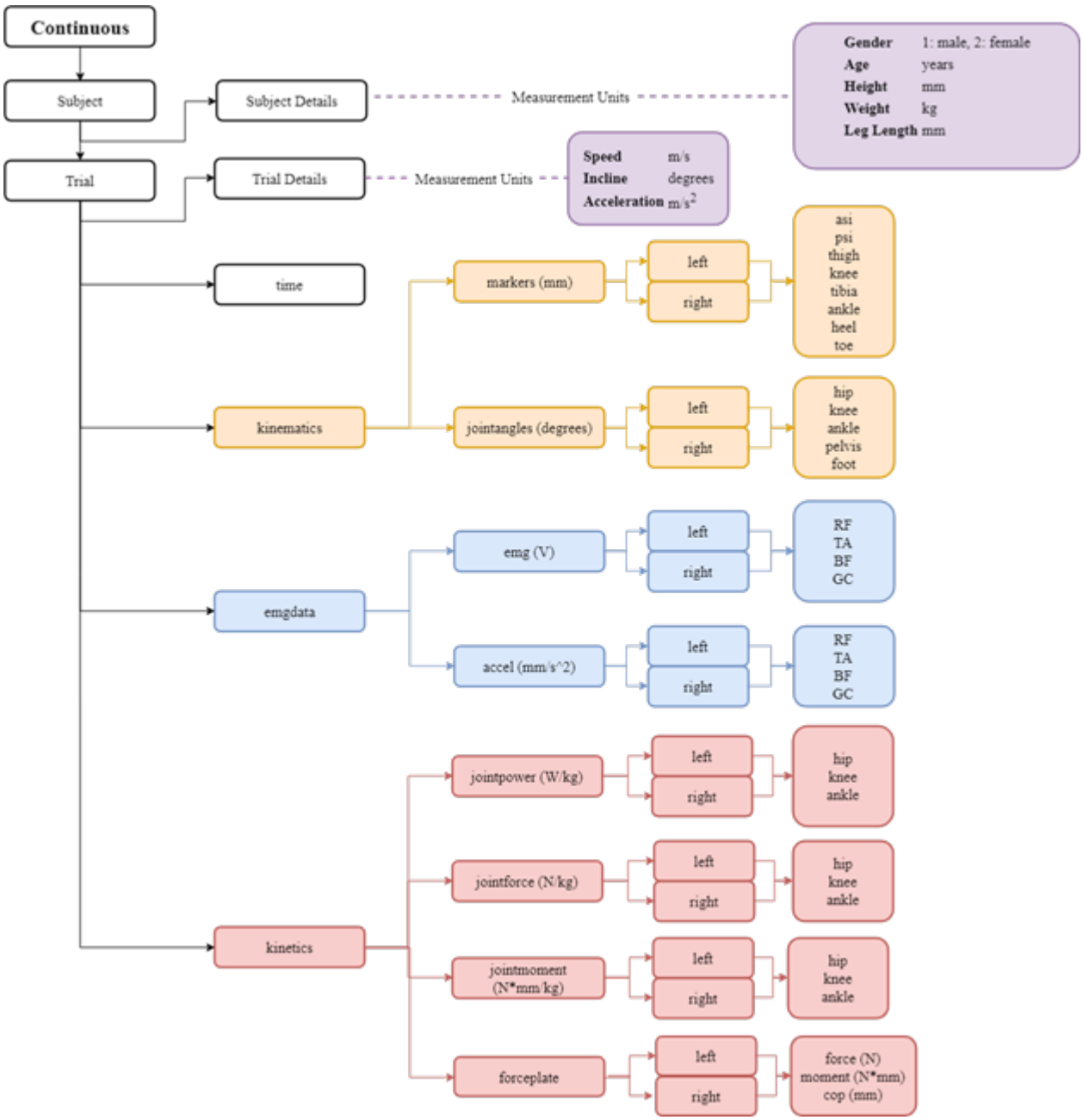


Figure 5.1: Continuous data Hierarchy

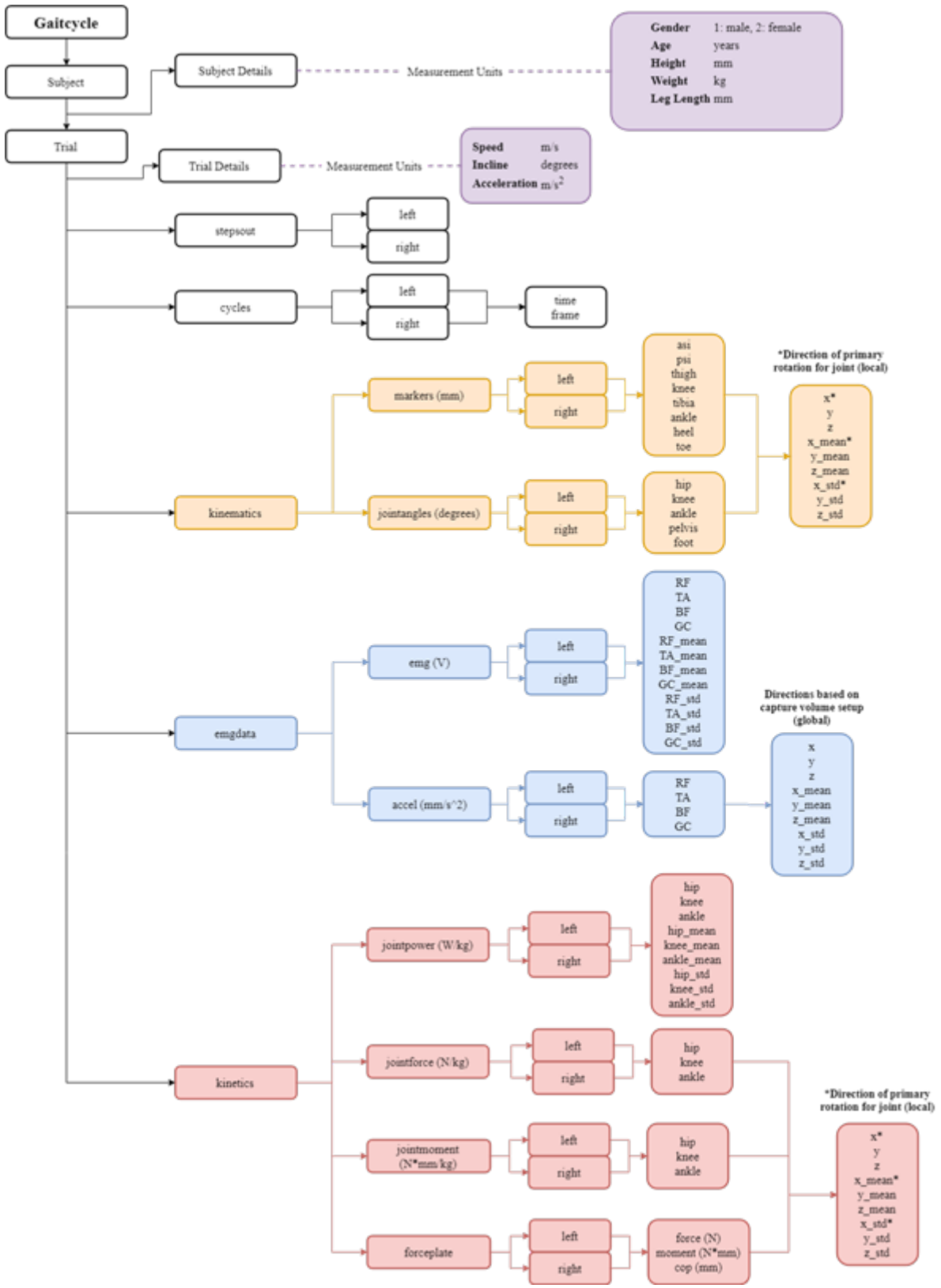


Figure 5.2: Gaitcycle data Hierarchy

Data analysis and reference variables generation

The first step in the development of our simulation of bipedal kinematics during walking is to generate the necessary input variables for the kinematic cycle that will be described in the next chapter. These are the CoM and the meta trajectories and the feet and trunk angles w.r.t. the ground plane and their respective speeds.

The first stage of the reference variables generation process consists in obtaining of the ZMP trajectory from the gait events and the feet and steps characteristics. Such events are recognized with a state-machine and determined from the feet markers contained in the experimental dataset. The feet markers gave us the step length and width information, and the feet plant and dorso dimensions.

From the feet events we determined the motion of the meta of the feet, expressed in terms of their position in time and their angle w.r.t. the ground [8]. The CoM position and speed were computed from the ZMP trajectory according to the LIMP model theory [41].

Some treatments were made on the torso and knee angles and the reference speeds were obtained from the respective trajectories and angles by derivation. Finally, the trajectories and speeds variables were packed together to be used in the kinematics cycle.

The graphs, tables and data used in this chapter and in the next to illustrate the various processes are derived from the experimental data of the trial s1i0 of the subject AB01 (1 m/s of treadmill's speed, treadmill not inclined), taken from the dataset of the section 5.2.

6.1 Event based data analysis

The data of zero slope trials from dateset of the section 5.2 were analyzed and processed using MATLAB. From the foot marker data, the various walking events at which the support point changes were detected. It was therefore possible to determine the state of the support during the entire duration of the trials. When walking, the trajectories of the feet and of the ZMP depend on how the foot is placed on the ground and on the position of the CoM. From the sequence of events it was possible to find out their course using a proper modeling, such as the LIMP and cart-table models.

The continuous variables of the dataset were segmented into vectors according to the state of the support, creating a correlation between the support of the foot and the trend of the joints and markers.

The joint centers and the links between them are also reconstructed from the markers: in this way it was possible to calculate the lengths of the body segments of the lower limbs of the subject of the trial.

State machine for support state and event recognition

The different gait events can be determined from the motion of the feet relative to the ground. From the recordings of the toe and heel markers it was possible to establish the support point (heel and toe) and the support leg (left or right) at each moment of the trial. Thus it can be also assessed when the subject was in double stance or in single stance.

A simple state machine was developed for detecting the support state during the entire trial. It has two states variables: the *stance leg*, which can be right or left, and the *stance point*, that can be the heel or the toe of the supporting leg. These determine four states alternating cyclically in a precise order in a regular gait: *left heel support*, *left tip support*, *right heel support*, *right tip support*. The states depend on the markers' coordinate along the world reference frame vertical axis z, $z_{foot-point}$. The index foot is R or L depending on the leg, while the index point is H or L when referring to the heel or toe marker respectively.

The events that must occur in order to change state are:

A *The change of the support point*: when $z_{stancefoot-H} \geq z_{stancefoot-T}$ the toe marker is at the same level or below the heel marker, the foot must be flat or the heel must be lifted w.r.t. the terrain. The support point is changing from the heel to the toe of the stance foot.

B *The change of the support leg*: if there is a local minimum of $z_{swingingfoot-H}$ while standing on the toe of the opposite foot, it corresponds to an heel-strike of the swinging foot. The support leg changes, as well the support point, and a step was completed.

The first state is determined by the different time plots of $z_{foot-point}$, and depends on which marker is lowest, while the others depend on events.

Sometimes the subject walks without placing the heel on the ground, especially at high walking speed. However $z_{swingingfoot-H}$ has a local minima also in this cases, so a short period of heel support will always be accounted before switching to the tip support state, even if only of one sample. In addition the condition for event A makes possible to change support point even if the heel and the toe do not lie together on the ground for a moment.

It has been verified that the double support period corresponds to the time spent in heel support, while the single support period corresponds to the time passed in toe stance, in the case of a slow walk. Thus it is possible not only to determine the stance leg and the support point, but also if the subject is in double or single stance.

At the end of this process, is generated an array with the following information:

- Stance leg.
- Support point.

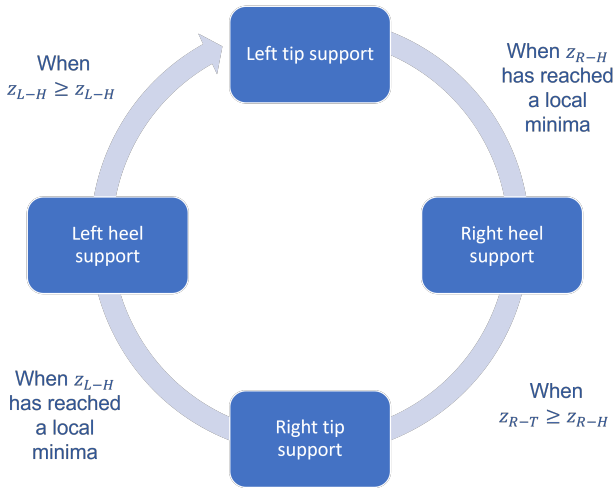


Figure 6.1: State machine representation for the subject AB01 and trial s0x8i0.

- Step number. Together with the support point it gives the support state.
- The initial instant of the time period spent in a certain state.
- The final instant of the time period spent in a certain state.
- An array providing the current state of stance, single or double for the whole period spent in a certain state.
- The time array of the period, starting from zero and ending at the final time less the initial time.

Using this array it is possible to segment the data according to the status of the media.

Evaluation of kinematic variables and body segment lengths

In the gait analysis it is fundamental the evaluation of the antropometric data of the subject, and if it is conducted on a kinematic level, the information on the correct lengths of the body segments and the position of the joint centers are essential.

Due to the lack of measurements on the lower limb segments, their length were calculated from the markers, following the procedure given by the Vicon Plug-in Gait Reference guide [46].

The Newington-Cage [4] model is used to define the positions of the hip joint centers in the pelvis segment. The interAsis distance is computed as the value between the left ASI (LASI) and the right ASI (RASI) markers. The distances from the Asis to the Trocanter are calculated independently for each leg, using the following formula:

$$AsisTrocDist = 0.1288 LegLength - 48.56 \tag{6.1}$$

The offsets vectors for the two hip joint centers (the left LHJC and the right RHJC) are calculated as follow:

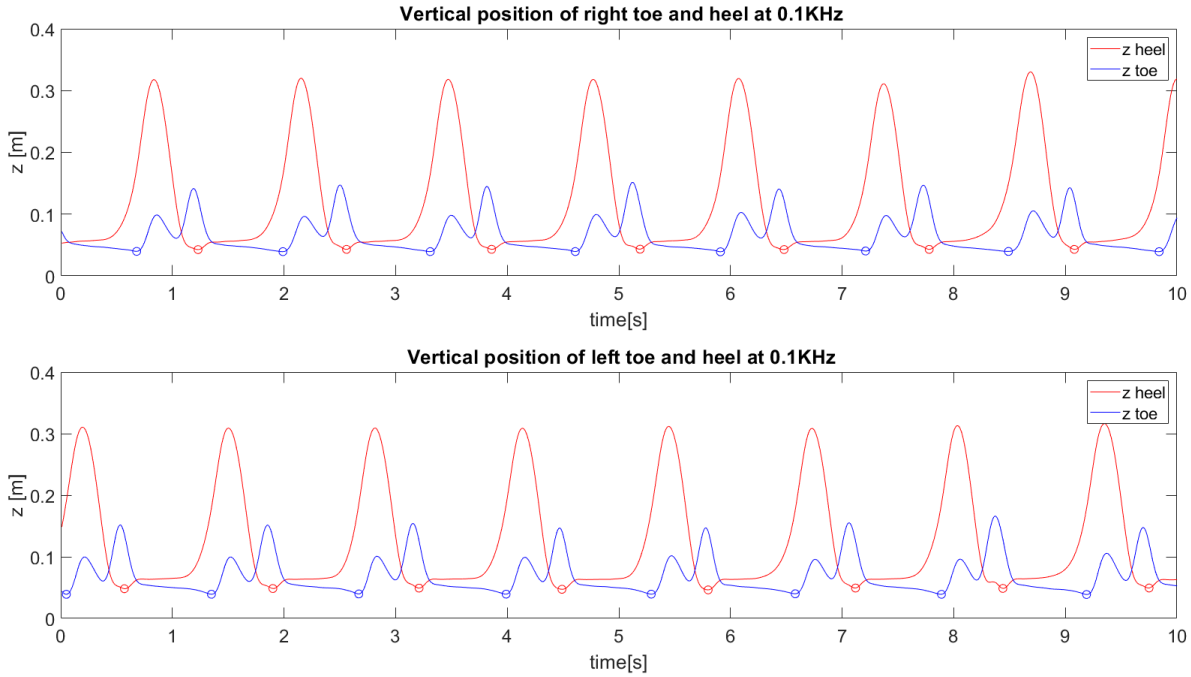


Figure 6.2: Position along the vertical axis z of the heel and toe markers for the subject AB01 during the trial s1i0. The minima are highlighted with circles

$$\begin{aligned} X =& C\cos(\theta)sin(\beta) - (AsisTrocDist + mm)\cos(\beta) \\ Y =& -(C\sin(\theta) - aa) \\ Z =& -C\cos(\theta)\cos(\beta) - ()AsisTrocDist + mm)\sin(\beta) \end{aligned} \tag{6.2}$$

where:

$$\begin{aligned} C &= 0.115 MeanLegLength - 15.3 \\ aa &\quad \text{is the interAsis distance} \\ \theta &\quad \text{is taken as 0.5 rad} \\ \beta &\quad \text{is taken as 0.314 rad} \end{aligned} \tag{6.3}$$

For the right joint center, the Y is negated, since Y is in the lateral direction for the pelvis embedded coordinate system.

To define the joint centers of the ankles and knees, a function called *chord* has been developed, starting from the homonym in the Vicon guide [46], where the way it works is described in details.

A plane is defined from a set of three points, a previously calculated joint center, the required joint center, and marker attached to the body segments that links them (the *plane definition marker*) as shown in Fig. 6.3). The joint center to be calculated is located at a some known, perpendicular distance (the *joint center offset*) from the respective joint marker.

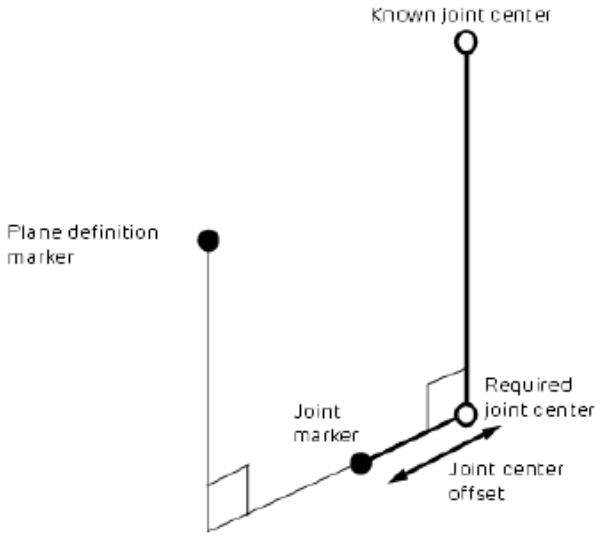


Figure 6.3: General geometric relationship between two successive joint centers and related markers

The joint center offsets of the knee joint and of the ankle joint centers (KJC and AJC) can be calculated by adding half the measured joint width and respective joint marker diameter. This function is called chord because the three points (two joint centers and the joint marker) lie on the periphery of the same circle.

In the function chord the joint markers lie in the same plane of the two joint centers and the plane definition marker. A modified version of the function calculates the required joint center position when the plane definition marker is rotated out of this plane by a known angle around the proposed joint center axis.

After positioning the joint centers, the length of the body segments can be determined. The pelvis segment stays between the two HJCs, the femur segment goes from the HJC to the KJC, while the tibial from the KJC to the AJC.

The foot plant is represented by the segment between the heel and the toe markers, while the foot dorso is the segment which links the AJC and the toe marker. Projecting the AKJ on the foot plant the meta position is obtained.

Considering that each body segment can be represented by a vector connecting two successive joints, and that a joint rotates two segments one w.r.t. the other, the rotation around a particular joint center can be calculated from the relative orientation of the vectors starting from that joint.

Let be a and b the unit vectors representing the orientation of two consecutive body segment vectors linked by a joint. The rotation matrix R representing their relative orientation can be calculated as:

$$R = I + [v]_{skew} + [v]^2_{skew} \frac{1-c}{s^2} \tag{6.4}$$

where:

Data: $R_{(3\times3)}$
Result: φ,ϑ,ψ
if $R_{3,1} \neq 1$ *and* $R_{3,1} \neq -1$ **then**
  $\psi_1 = -\arcsin R_{3,1}$;
  $\psi_2 = \pi - \arcsin R_{3,1}$;
  $\varphi_1 = Atan2(R_{3,2}/\cos\psi_1, R_{3,3}/\cos\psi_1)$;
  $\varphi_2 = Atan2(R_{3,2}/\cos\psi_2, R_{3,3}/\cos\psi_2)$;
  $\vartheta_1 = Atan2(R_{2,1}/\cos\psi_1, R_{1,1}/\cos\psi_1)$;
  $\vartheta_1 = Atan2(R_{2,1}/\cos\psi_2, R_{1,1}/\cos\psi_2)$;
  $\psi = min(\psi_1, \psi_2)$;
  $\varphi = min(\varphi_1, \varphi_2)$;
  $\vartheta = min(\vartheta_1, \vartheta_2)$;
else
  $\vartheta = 0$;
  if $R_{3,1} = -1$ **then**
    $\psi = \pi/2$;
    $\varphi = \vartheta + Atan2(R_{1,2}, R_{1,3})$;
  else
    $\psi = -\pi/2$;
    $\varphi = -\vartheta + Atan2(-R_{1,2}, -R_{1,3})$;
  end
end

Algorithm 1: Computing rotation angles from a given rotation matrix R.

$$\begin{aligned} v &= a \times b \\ s &= \|v\| \text{ (i.e. the sine of the angle)} \\ c &= a\dot{b} \text{ (i.e. the cosine of the angle)} \end{aligned} \tag{6.5}$$

I is the eye matrix $[v]_{skew}$ is the skew-symmetric cross-product matrix of $v = (v_1, v_2, v_3)$,

$$[v]_{skew} = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \tag{6.6}$$

From the rotation matrix it is possible to compute the relative angles, computed with an apposite MATLAB function that operates according to the Algorithm 1.

To use the chord or modified chord functions are required the value of the joint center offsets or at least the diameter of the articulations, but these are not present in the experimental data.

For this reason we opted for a trial and error process using chord, thus assuming that the joint marker is not rotated w.r.t. the plane defined by the joint centers and their respective plane definition marker. First we have established an appropriate value of the joint offsets and second we have compared the joint angles computed using the method presented above to those from the dataset. If they coincide, the joint offsets were

accepted as valid. It is important to emphasize that although this option is not the best or even the most suitable, it gave us acceptable results, and the body segments found were proportionate.

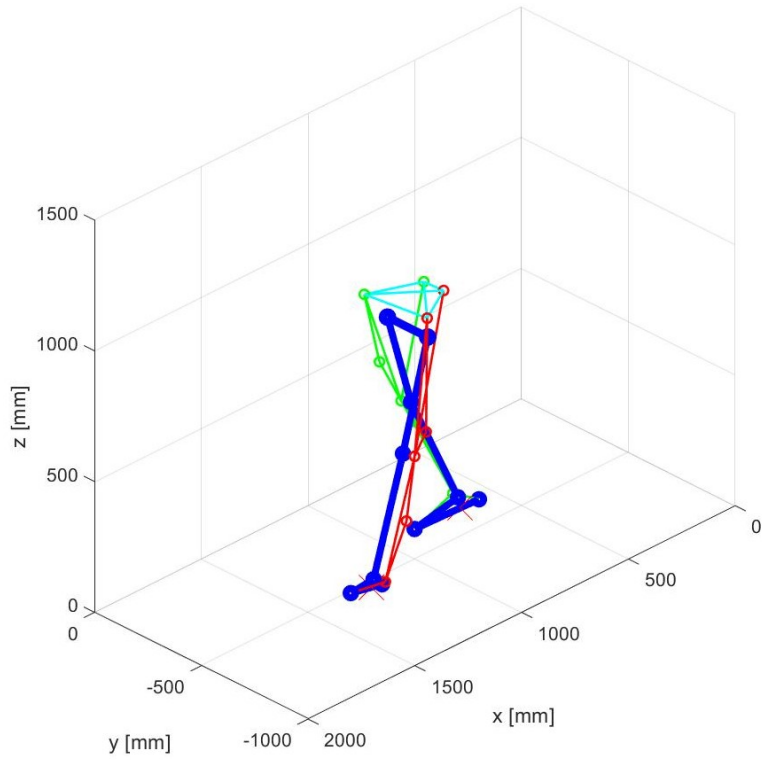


Figure 6.4: Markers and body segments represented in the dataset world frame

Outputs

The extensions of the tibial segments, of the femur segments, of the plant of the feet and of the pelvic segment are calculated at each instant. The time-averaged length of each body segment was determined, and the standard deviation of each time sample.

To determine whether the values of the joint offsets have been chosen correctly, the standard deviations must not exceed a certain threshold: this would mean that the lengths vary excessively over time. In any case, it is impossible to obtain the time invariant length of the body segments, because of the same measurement methodology. The motion capture system may not measure the position of the markers correctly, and markers may shift during walking. There will always be a few values that deviate significantly from the average, even in the best situation.

As output to this process we have the data obtained from the state machine described in the first paragraph and the average values of the body segments.

Joint	Joint Offsets
Left KJC	40 mm
Right KJC	40 mm
Left AKJ	36 mm
Right AJC	35 mm

Table 6.1: Chosen center offsets for the subject AB01 and trial s1i0.

Body Segment	Time-Averaged Length
Pelvis	187 mm
Right femur	409 mm
Left femur	400 mm
Right tibia	425 mm
Left tibia	448 mm
Right feet plant	250 mm
Left feet plant	245 mm
Right feet dorso	187 mm
Left feet dorso	174 mm
Right AJC-meta distance	39 mm
Left AJC-meta distance	32 mm

Table 6.2: Time-averaged extension of the body segments for the subject AB01 and trial s1i0.

6.2 Cartesian variables generation

The output from the event based data were used to compute the reference variables needed for the kinematic analysis. Our reasoning starts from the paper of SangJoo Known et al. [25] in which it is explained how to adapt the generation of the trajectories of the ZMP and feet developed by Choi et al. [8] to produce human-like trajectories instead. Then the trajectory of the ZMP is used to reconstruct the COG pattern, according to the LIPM modeling. The remaining reference variables needed for the kinematic analysis, i.e. trunk angles w.r.t. the ground and knee joint angles, are taken directly from the experimental data.

6.2.1 ZMP and feet trajectories generation

In order to describe the trajectories of the ZMP and the meta of the feet, a series of variables obtained from the outputs of the state machine described in the previous chapter and the markers' data are required. First of all, the incomplete steps were deleted from the recordings, to start and end the trial with a flat-foot event. Then, starting from the first flat-foot event of the trial, the following parameters are found iteratively:

- *stepN*: the step number, assigned from supportinfos, deleting the first incomplete step.

- t_{to} : time between flat-foot and toe-off events of the support foot.
- t_{hs} : time between toe-off and the heel-strike events of the support foot.
- t_{ff} : time between heel-strike and flat foot events of the support foot.
- Δx_{step}: the longitudinal distance between the meta of the feet, calculated at the flat-foot event.
- Δy_{step}: the lateral distance between the meta of the feet, calculated at the flat-foot event.
- l_f: the longitudinal distance between the markers of the meta and of the toe of the supporting foot during the flat-foot event.
- l_b: the longitudinal distance between the markers of the meta and of the heel of the supporting foot during the flat-foot event.

The first six parameters vary with each step, while the last two l_f and l_b are constant, because they depend only on the length of the foot.

ZMP generation from the support states

In flat-footed walking, the ZMP remains fixed in the middle of the sole of the supporting foot until it abruptly moves to the opposite foot during the double stance phase. However, in human walking the ZMP always travels from the heel to the toe of the supporting foot, and changes foot more gradually. The ZMP longitudinal trajectory between two flat-footed events occurring at time t_i and time t_f respectively is given by:

$$ZMP_x(t) = \begin{cases} ZMP_x(t_i) + (l_f/t_{ff})t & \text{for } t_i < t \leq t_i + t_{ff}, \\ ZMP_x(t_i) + (l_b/t_{to})t - l_b + \Delta(x_{step}) & \text{for } t_i + t_{ff} < t \leq t_f. \end{cases} \tag{6.7}$$

Instead in the lateral direction is given by:

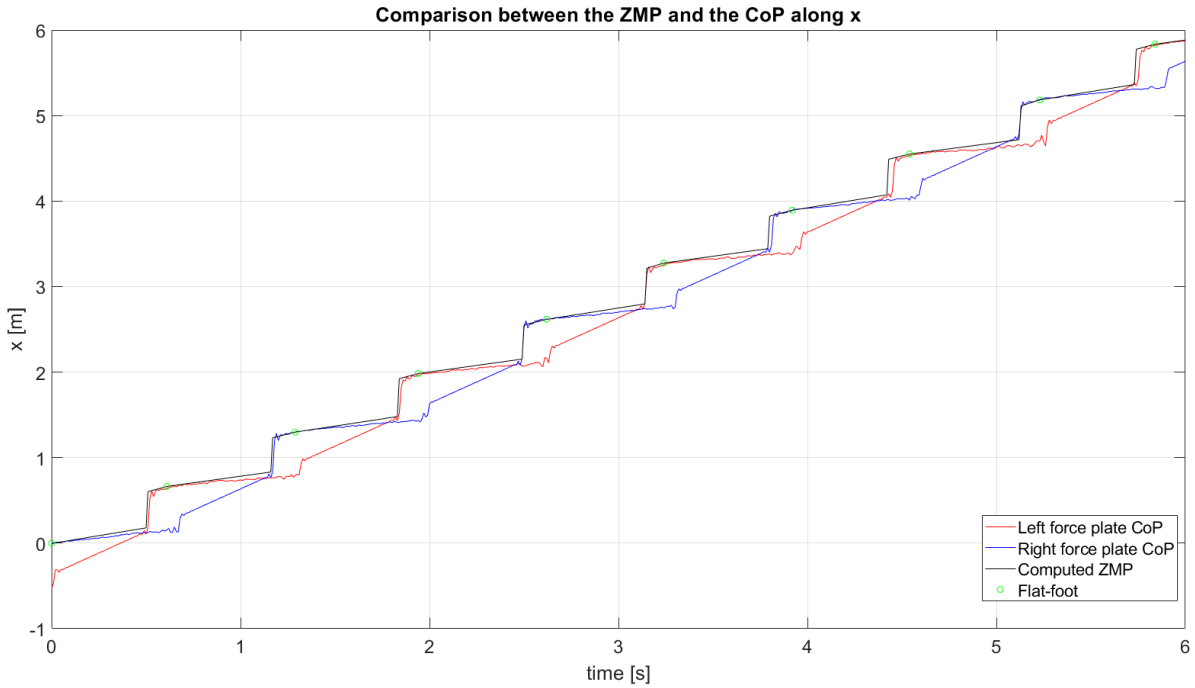
$$ZMP_y(t) = \begin{cases} \Delta(y_{step}) & \text{for } t_i < t \leq t_i + t_{ff}, \\ (K_f - K_y)(t/t_{to}) - K_f & \text{for } t_i + t_{ff} < t \leq t_f. \end{cases} \tag{6.8}$$

The parameter K_y is defined at each steps and is calculated as:

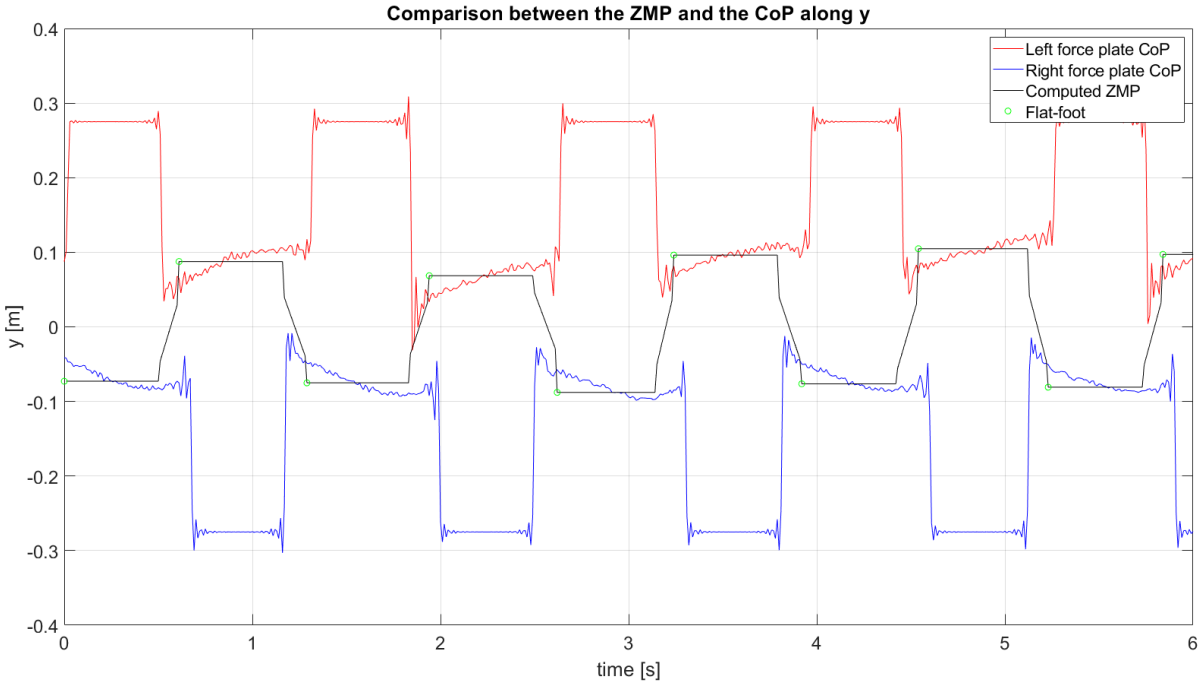
$$K_y = \frac{\Delta(y_{step}) w_n \tanh(w_n(t_{to}))}{1 + \Delta(y_{step}) w_n \tanh(w_n(t_{to}))} \tag{6.9}$$

where $w_n = \sqrt{g/z_{CoG}}$ is the natural frequency of the inverted pendulum that from the ZMP (its base) to the CoM. This formula was derived from the CoG trajectory constrain along the lateral direction defined by Choi et al. in [8]. The parameter K_f is equal to Ky calculated for the next step.

It can be noticed that the time difference between t_i and t_f (i.e. the time between two flat-foot events) is equal to the sum of t_{ff} and t_d.



(a)



(b)

Figure 6.5: Comparison between the generated ZMP (black line) and the CoPs (blue and red lines) taken from experimental data along the longitudinal direction (a) and the lateral direction(b). (subject AB01 trial s1i0)

The ZMP position is finally compared with the CoP data measured by the two force plates (one for each side) placed under the Bertec treadmill used in the trials (Fig. 6.5a and 6.5b). The CoP from the experimental data is represented in the local frames of the force plates [6], while the ZMP in the global frames located on the ground defined in the section 2.1.3. The CoP reference frame is changed from local to global adding the distance covered by the treadmill over time (equal to its its speed times the elapsed time) to the CoP position in the longitudinal direction, and by subtracting (for the right side) or adding (for the left side) half of the treadmill width to the opposite of the CoP's lateral position.

The comparison shows that the calculated ZMP has a similar trend to that of the CoP in the longitudinal direction x and they almost overlap completely. In the lateral direction y, however, the situation is different: the lines parallel to the time axis of the ZMP and those of the CoP corresponding to the single support phase do not coincide. This occurs for two reasons: during the double stance phase, force platforms are unable to measure the displacement of the CoP from one foot to the other, and during the single stance phases, the CoP moves from the inner part of the sole to the outer part due to the pronation of the stance foot.

The green circles associated in Fig. 6.5a and 6.5b represent the meta position of the support foot when it is flat. In the longitudinal direction the flat-foot event corresponds to a change in the slope of the ZMP, and in the lateral direction to the completion of the displacement of the ZMP from the previous support foot to the next.

The double stance phases correspond to the intervals between a heel-strike event and a flat foot event of the same foot.

The generated trajectory of the ZMP can be seen as a continuous piecewise linear functions. Its break-points have been recorded to be used in the CoG generation process to compute the angular coefficients of the ZMP linear traits.

Meta trajectories generation from support states

In human walking the swinging leg goes through take-off, swing and landing, while the opposite leg is in contact with the terrain, supporting the body. To define the meta trajectory of a foot, the following notations are defined:

- ϕ_{take} : the foot angle around the lateral axis w.r.t. the terrain at the toe-off event.
- ϕ_{land} : the foot angle around the lateral axis w.r.t. the terrain at the heel-strike event.
- h_0 : maximum height of the meta during the swing phase.
- (x_{take}, z_{take}) : horizontal and vertical positions of the meta at the toe-off event.
- (x_{land}, z_{land}) : horizontal and vertical positions of the meta at the heel-strike event.
- T : the time period between two flat foot-events of the swinging foot equal to the sum of t_{to}, t_{hs} and t_{ff}.
- Dx : the distance covered between two consecutive flat-foot events of the swinging foot along x.

- Dy : the distance covered between two consecutive flat-foot events of the swinging foot along y.

The value of ϕ_{take}, ϕ_{land} and h_0 are constant and established appropriately by observing of the measurements of the feet's angles w.r.t. the floor and the position of the meta during the swing phase. Instead, the values of (x_{take}, z_{take}) and (x_{land}, z_{land}) are calculated as shown in 6.14. The feet angles and their trajectories are defined in a time interval lasting T. The angle of the feet w.r.t. the terrain is described by 3rd order polynomials to get a continuous differentiable pose, as given by:

$$\phi(t) = \begin{cases} 3\phi_{take}(t/t_{to})^2 - 2\phi_{take}(t/t_{to})^3 \quad \text{for } 0 < t \leq t_{to}; \\ \phi_{take} - 3(\phi_{take} + \phi_{land})(t/t_{hs})^2 + 2(\phi_{take} + \phi_{land}))((t - t_{ho}/t_{hs})) \\ \text{for } t_{to} \leq t < t_{to} + t_{hs}; \\ 3\phi_{land}((t - T + t_{ff})/t_{ff})^2 - 2\phi_{land}((t - T + t_{ff})/t_{ff})^3 - \phi_{land} \\ \text{for } t_{to} + t_{hs} \leq t \leq T; \end{cases} \tag{6.10}$$

The position of the meta along the longitudinal and vertical directions depends on the angle of the foot and is determined by simple sinusoidal functions (Eqs. 6.11 and 6.13). Instead the lateral trajectories are computed by a sigmoidal function (Eq. 6.12).

$$r_x(t) = \begin{cases} l_f(i)(1 - \cos\phi(t)) & \text{for } 0 < t \leq t_{to}; \\ x_{take} + M(1 - \cos wr(t - t_{to})) & \text{for } t_{to} \leq t < t_{to} + t_{hs}; \\ Dx - l_b(1 - \cos\phi(t)) & \text{for } t_{to} + t_{hs} \leq t \leq T; \end{cases} \tag{6.11}$$

$$r_y(t) = \begin{cases} 0 & \text{for } 0 < t \leq t_{to}; \\ Dy/(1 + e^{-0.23(t - t_{to} + t_{hs}/2)}) & \text{for } t_{to} \leq t < t_{to} + t_{hs}; \\ Dy & \text{for } t_{to} + t_{hs} \leq t \leq T; \end{cases} \tag{6.12}$$

$$r_z(t) = \begin{cases} l_f(\sin(\phi(t))) & \text{for } 0 < t \leq t_{to}; \\ z_{take} + N(1 - \cos 2w_r(t - t_{to})) & \text{for } t_{to} \leq t < t_m \\ z_{land} + Q(1 - \cos 2w_r(t - t_{to})) & \text{for } t_m \leq t < t_{to} + t_{hs}; \\ -l_b(1 - \cos\phi(t)) & \text{for } t_{to} + t_{hs} \leq t \leq t_f; \end{cases} \tag{6.13}$$

where:

$$\begin{aligned} x_{take} &= l_f(1 - \cos\phi_{take}) & z_{take} &= l_f \sin\phi_{take} \\ x_{land} &= l_b(1 - \cos\phi_{land}) & z_{land} &= l_b \sin\phi_{land} \\ w_r &= 180/t_{hs} & M &= l_{step} - (x_{take} + x_{land})/2 \\ N &= (h_0 - z_{take})/2 & Q &= (h_0 - z_{land})/2 \\ t_m &= t_{hs}/2 + t_{to} & T_{step} &= t_{to} + t_{ff}(i) + t_{hs} \end{aligned} \tag{6.14}$$

Outputs

The outputs of the processes defined previously in the section are the ZMP trajectory, the meta trajectories and the angles of the feet w.r.t. the ground.

To these variables a first trait was added to simulate the transition from standing with both feet touching the ground to the first half-step. This ensures that the biped starts from a stable position.

The recorded ZMP break-points are collected to be used for reconstructing the continuous piece-wise linear function which matches the ZMP trajectory.

Initialized Value	Value	
ϕ_{take}	15	mm
ϕ_{land}	10	mm
h_0	20	cm

Table 6.3: Constant values chosen for generating the feet trajectories for the subject AB01 and trial s0x8i0.

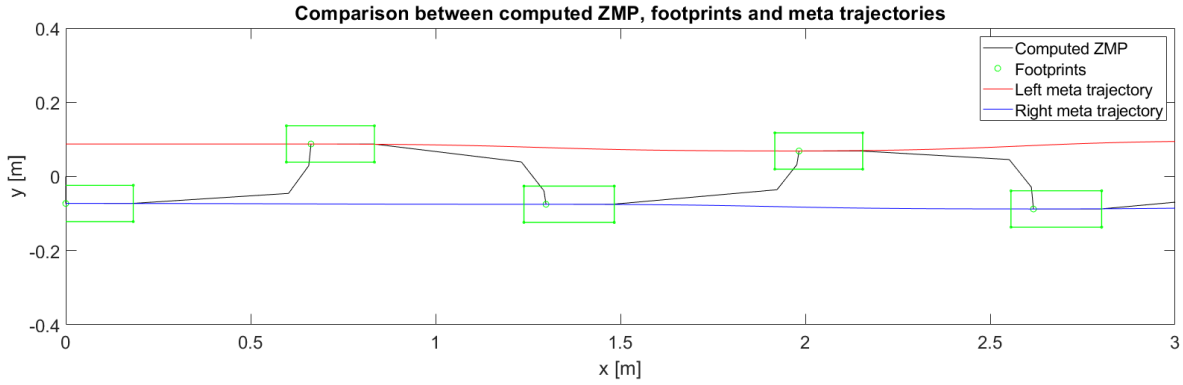


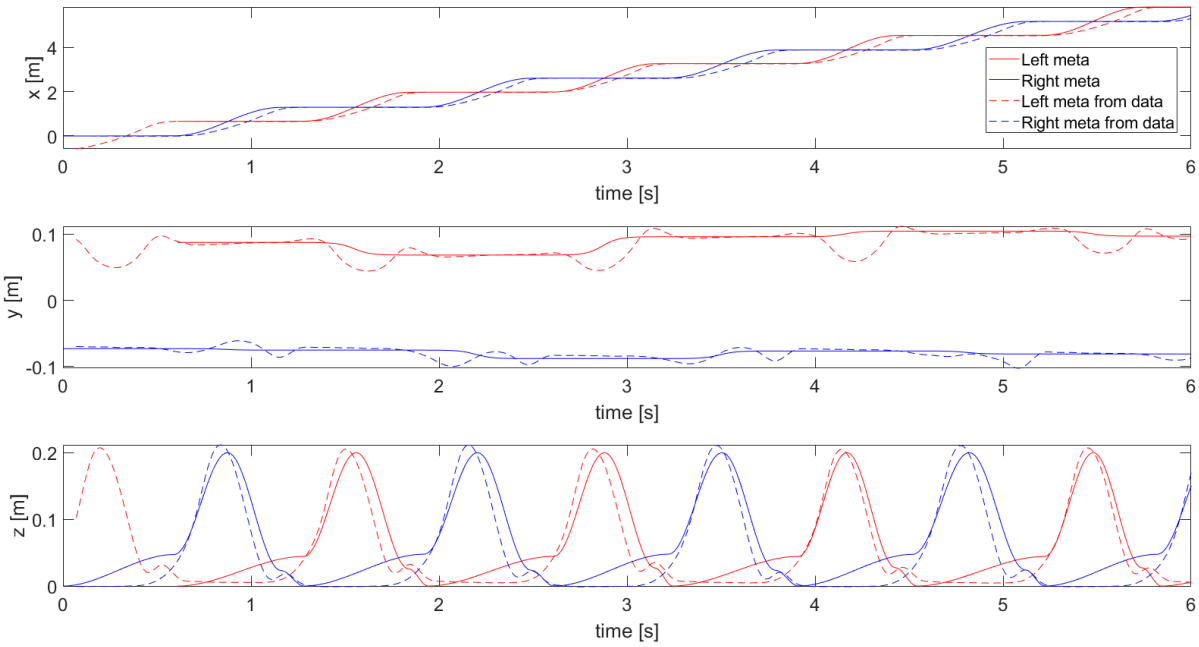
Figure 6.6: ZMP, feet trajectories and footprints in the ground plane (subject AB01 and trial s0x8i0).

6.2.2 CoG computation

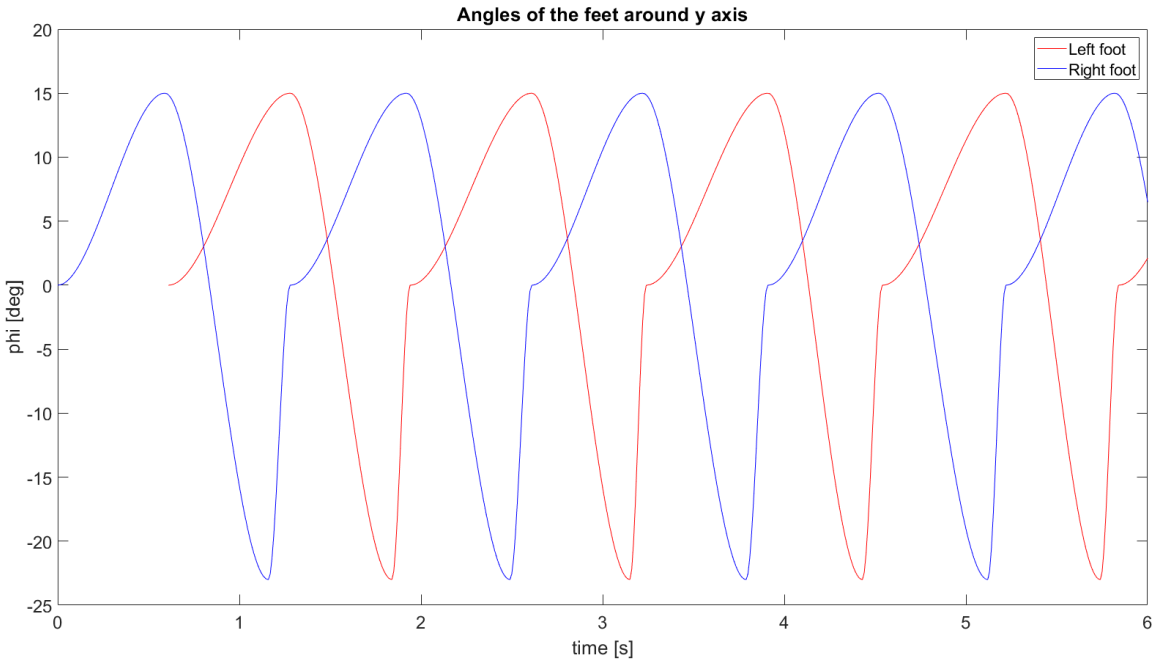
The CoG motion can be derived from the ZMP as stated by Kajita et al. [22]. He proposed to solve numerically an infinite-horizon LQR problem to stabilize the ZMP, optimizing the quadratic cost of the ZMP tracking against the CoG acceleration. The weakness of this method is the high computation time. For this reason, the method of Russ Tedrake et al. [41], which proposes an iterative method for finding solutions to the LQR problem, was preferred for calculating the CoG trajectory and verifying the proper ZMP tracking at the same time. The CoM and ZMP dynamics of a legged rigid body systems con be written in the state space form as:

$$\dot{x} = Ax + Bu = \begin{bmatrix} 0_{2x2} & I_{2x2} \\ 0_{2x2} & 0_{2x2} \end{bmatrix} x + \begin{bmatrix} 0_{2x2} \\ I_{2x2}, \end{bmatrix} u \tag{6.15}$$

$$\dot{y} = Cx + Du(x,u) = \begin{bmatrix} I_{2\times 2} & 0_{2\times 2} \end{bmatrix} x + \frac{-z_{com}}{\ddot{z}_{com}} I_{2\times 2} u. \tag{6.16}$$



(a)



(b)

Figure 6.7: Comparison between left and right meta trajectories, generated or from the marker analysis (a), and generated feet angles around the lateral axis y (b) (subject AB01 and trial s0x8i0).

where $x = [x_{CoG}, x_{CoG}, \dot{x}_{CoG}, \dot{x}_{CoG}]^T$, $u = [\ddot{x}_{CoG}, \ddot{x}_{CoG}]^T$, $y = [x_{ZMP}, x_{ZMP}]^T$, g the constant gravitational acceleration and z_{CoM} the center of gravity vertical position. Unlike the formulation commonly found in literature [22], 3^{rd}-order derivatives of the CoG trajectory are not considered. The terms $D(x, u)$ becomes $D(u)$ assuming a constant CoG height z_{CoG}.

The desired ZMP trajectories, $y_d(t)$ can be described by a continuous piecewise polynomial of degree k with n breaks at t_j (with $t_0 = 0$ and $t_n = t_f$):

$$y_d(t) = \sum_{i=0}^{k} c_{j,i}(t - t_j)^i \quad \text{for } j = 0, \ldots, n-1 \quad \text{and} \quad \forall t \in [t_j, t_{j+1}). \tag{6.17}$$

The ZMP used in the simulations in this document is a piece-wise linear function (i.e. of degree $k = 1$), thus $c_{j,i}$ will be the angular coefficient of each trait, determined from its breakpoints obtained like described previously.

LQR design

Given the desired trajectory of the ZMP $y_d(t)$ the optimal ZMP tracking controller can be obtained solving a continuous-time LQR problem:

$$\begin{aligned} \underset{u(t)}{\text{minimize}} \int_0^{\inf} & (||y(t) - y_d(t)||_Q^2 + ||u(t)||_R^2)dt \\ \text{subject to} \quad & Q = Q^T > 0 \\ & R = R^T > 0 \\ & y_d(t) = y(t_f), \forall t \geq t_f \\ & \dot{x(t)} = Ax(t) + Bu(t) \\ & y(t) = Cx(t) + Du(t) \end{aligned} \tag{6.18}$$

and initial conditions $x(0) = x_0$. Here Q and R explicitly trade off ZMP tracking performance against the cost of accelerating the CoG. Note that Q has to be positive defined. Observing the third constrain in 6.21, the problem can be rewritten with a cost on state in coordinates relative to the final conditions:

$$\bar{x}(t) = x(t) - \begin{pmatrix} y_d(t_f) \\ 0_{2\times 1} \end{pmatrix} \tag{6.19}$$

$$\bar{y}(t) = y_d(t) - y_d(t_f) \tag{6.20}$$

Thus the LQR problem become:

$$
\begin{aligned}
&\underset{u(t)}{\text{minimize}} \int_0^{\inf} g(\bar{x}(t), u(t))dt \\
&\text{subject to} \quad Q = Q^T > 0 \\
&\qquad\qquad\quad R = R^T > 0 \\
&\qquad\qquad\quad y_d(t) = y(t_f), \forall t \geq t_f \\
&\qquad\qquad\quad \dot{x(t)} = Ax(t) + Bu(t) \\
&\qquad\qquad\quad y(t) = Cx(t) + Du(t)
\end{aligned}
\tag{6.21}
$$

where:

$$
g(\bar{x}(t), u(t)) = \bar{x}^T Q_1 \bar{x} + \bar{x}^T q_2(t) + q_3(t) + u^T(t) R_1 u(t) + u(t)^T r_2(t) + 2\bar{x}^T(t) N u(t) \tag{6.22}
$$

and:

$$
\begin{array}{lll}
Q_1 = C^T Q C & q_2(t) = -2C^T Q \bar{y}_d(t) & q_3 = ||\bar{y}_d(t)||^2_Q \\
R_1 = R + D^T Q D & r_2(t) = -2DQ\bar{y}_d(t) & N = C^T Q D
\end{array}
\tag{6.23}
$$

Note that this implies that $\lim_{t \to \inf} \bar{x}(t) = 0$ in order for the cost to be finite. From standard LQR theory [40] the optimal cost-to-go for this problem has the general form:

$$
J(\bar{x}(t), t) = \bar{x}^T(t) S_1(t) \bar{x}(t) + \bar{x}^T(t) s_2(t) + s_3(t) \tag{6.24}
$$

The optimal controller id defined as:

$$
u^*(t) = -R_1(N_B \bar{x}(t) + r_s(t)), \tag{6.25}
$$

where $N_B = N^T + B^T S_1$ and $r_s = (1/2)(r_2(t) + B^T s_2)$ and the S_1, s_2 and s_3 terms are computed via the Riccardi differential equation:

$$
\begin{aligned}
\dot{S}_1 &= -Q_1 + N_B^T R_1^{-1} N_B - S_1 A - A^T S_1 \\
\dot{s}_2 &= -q_2 + 2N_B^T R_1^{-1} r_s(t) - A^T s_2(t) \\
\dot{s}_3 &= -q_3 + r_s^T(t) R_1^{-1} r_s(t)
\end{aligned}
\tag{6.26}
$$

It is possible to observe that $\dot{S}_1$ is time-independent, so S_1 is a constant term, given by the steady-state solution of the algebraic Riccati equation. Furthermore, it depends on Q, R and z_{CoG} but does not depend on the ZMP trajectory, thus it can be computed offline and simply used as a constant at runtime. The optimal feedback controller 6.25 can therefore be expressed as:

$$
u^*(t) = K_1 \bar{x}(t) + k_2(t), \tag{6.27}
$$

where the feedback matrix K_1 is a constant and:

$$
k_2(t) = -R_1^{-1} \left(\frac{1}{2} B^T s_2(t) - DQ\bar{y}_d(t) \right). \tag{6.28}
$$

Computing $s_2(t)$

The time-varyng linear term in the cost-to-go is given by the linear differential equation:

$$\dot{s}_2(t) = A_2 s_2(t) + B\bar{y}_2(t), \quad s_2(t_f) = 0, \tag{6.29}$$

with:

$$\begin{aligned} A_2 &= N_B^T R_1^{-1} B^T - A^T \\ B_2 &= 2(C^T - N_B^T R_1^{-1} D)Q. \end{aligned} \tag{6.30}$$

Assuming $\bar{y}_d(t)$ is described by a continuous piecewise polynomial, this system has an explicit solution given by:

$$s_2(t) = e^{A_2(t-t_j)}\alpha_j + \sum_{i=0}^{k} \beta_{j,i}(t-t_j)^i, \quad \text{for } \forall t \in [t_j, t_{j+1}) \tag{6.31}$$

with α_j and $\beta_{j,i}$ vectors parameters to be solved.

The second term of the controller in 6.27 can be straightforwardly computed given the solution of $s_w(t)$

$$k_2(t) = -\frac{1}{2}R_1^{-1}B^T e^{A_2(t-t_j)}\alpha_j - \sum_{i=0}^{k} \gamma_{j,i}(t-t_j)^i, \tag{6.32}$$

where:

$$\gamma_{j,i} = R_1^{-1}DQc_{j,i} - \frac{1}{2}R_1^{-1}B^T\beta_{j,i}. \tag{6.33}$$

The Algorithm 2 taken from [41] solves the parameters of $s_2(t)$ and $k_2(t)$ backwards in time. Its complexity is $O(nk)$, therefore, when the trajectory is piecewise linear ($k = 1$) the computation time will be linear in the number of different traits too. To apply the algorithm it is mandatory that A_2 is full rank. The A_2 can be rewritten in block diagonal form:

$$A_2 = \begin{bmatrix} 0 & (S_{1,2} + QD)R_1^{-1} \\ -I & S_{2,2}^T R_1^{-1} \end{bmatrix} \tag{6.34}$$

where:

$$S_1 = \begin{bmatrix} S_{1,1} & S_{1,2} \\ S_{1,2}^T & S_{2,2} \end{bmatrix}. \tag{6.35}$$

Its inverse can be expressed as:

$$A_2^{-1} = \begin{bmatrix} ((S_{1,2} + QD)R_1^{-1})^{-1} S_{2,2}^T R_1^{-1} & -I \\ ((S_{1,2} + QD)R_1^{-1})^{-1} & 0 \end{bmatrix} \tag{6.36}$$

We then have that $A2$ is full rank if and only if $(S_{1,2} + QD)$ is invertible. Recall that $Q = Q^T > 0$ and D is a positive scaling of the identity matrix. Therefore, $A2$

is invertible if $S_{1,2}$ is full rank. Since $S1$ is a constant, this property can be ensured at design time (empirically $S_{1,2}$ is always full rank for admissible costs, but a formal statement is difficult to make since the continuous algebraic Riccati equation does not have a closed-form solution).

Data: A_2, B_2, degree k piecewise polynomial $\bar{y}_d(t)$ with n breaks
Result: α_j,$\beta_{j,i}$,$\gamma_{j,i}$,$\forall j \in (1,\ldots,n)$,$\forall i \in (0,\ldots,k)$
for $j = n,\ldots,1$ **do**
  $\beta_{j,k} = -A_2^{-1}B_2c_{j,k}$;
  $\gamma_{j,k} = R_1^{-1}DQc_{j,k} - (1/2)R_1^{-1}B^T\beta_{j,k}$;
  for $j = n,\ldots,1$ **do**
    $\beta_{j,i} = A_2^{-1}((i+1)\beta_{j,i+1} - B_2c_{j,k})$;
    $\gamma_{j,i} = R_1^{-1}DQc_{j,k} - (1/2)R_1^{-1}B^T\beta_{j,i}$
  end
  if $j = n$ **then**
    $\alpha_j = e^{A_2(t_n - t_j)} / \left(-\sum_{i=0}^{k-1} \beta_{j,i}(t-t_j)^i\right)$
  else
    $\alpha_j = e^{A_2(t_{j+1} - t_j)} / \left(\alpha_{j+1} + \beta_{j+1,i} - \sum_{i=0}^{k-1} \beta_{j,i}(t-t_j)^i\right)$;
  end
end

Algorithm 2: Solve for parameters of $s_2(t)$ and $k_2(t)$.

Computing the CoG trajectory

To compute the CoG trajectory that corresponds to the desired ZMP trajectory, the optimal input must be applied to the state space equations representing the body dynamics. Substituting 6.27 in 6.15 we have:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + B(K_1x(t) + k_2(t)) \\ &= (A + BK_1)x(t) + Bk_2(t)\end{aligned} \tag{6.37}$$

Since $k_2(t)$ is the result of another linear system cascaded in front of this one, it is conceptually simplest to solve them jointly. We define:

$$\begin{aligned}z(t) &= \begin{bmatrix} x(t) \\ s_2(t) \end{bmatrix} \\ \dot{y}(t) &= A_z z(t) + B_z \bar{y}_d(t),\end{aligned} \tag{6.38}$$

where

$$\begin{aligned}A_z &= \begin{bmatrix} A + BK_1 & -\frac{1}{2}BR_1^{-1}B^T \\ 0 & A_2 \end{bmatrix} \\ B_z &= \begin{bmatrix} BR_1^{-1}DQ \\ B_2 \end{bmatrix}\end{aligned} \tag{6.39}$$

The solution to this systems, as in the above, has the general form:

$$z(t) = e^{A_z(t-t_j)}a_j + \sum_{i=0}^{k} b_{j,i}(t-t_j)^i \tag{6.40}$$

Algorithm 3 solves for the coefficients of 6.40 forward in time. Note that it is possible to reuse the parameters, $\beta_{j,i}$, returned from Algorithm 1 and thereby only solve for the top half of $b_{j,i}$:

Data: $x(0)$, A_z, B_z, degree k piecewise polynomial $\bar{y}_d(t)$ with n breaks
Result: a_j, $b_{j,i}$, $\forall j \in (1,\dots,n)$, $\forall i \in (0,\dots,k)$
for $j = n,\dots,1$ **do**
  $\beta_{j,k} = -A_z^{-1}B_z c_{j,k}$;
  for $i = k-1,\dots,0$ **do**
    $b_{j,i} = A_z^{-1}((i+1)b_{j,i+1} - B_z c_{j,i})$;
  end
  $a_j = \begin{bmatrix} x - b_{j,1} \\ \alpha_j \end{bmatrix}$;
  $x = \begin{bmatrix} I \\ 0 \end{bmatrix} e^{A_z(t_{j+1}-t_j)}a_j + \sum_{i=0}^{k-1} b_{j,i}(t_{j,i} - t_i)^i$;
end
$b_{j,1}[1:2] = b_{j,1}[1:2] + y(t_f)$;;

Algorithm 3: Solve for the CoG trajectory $x(t)$.

Outputs

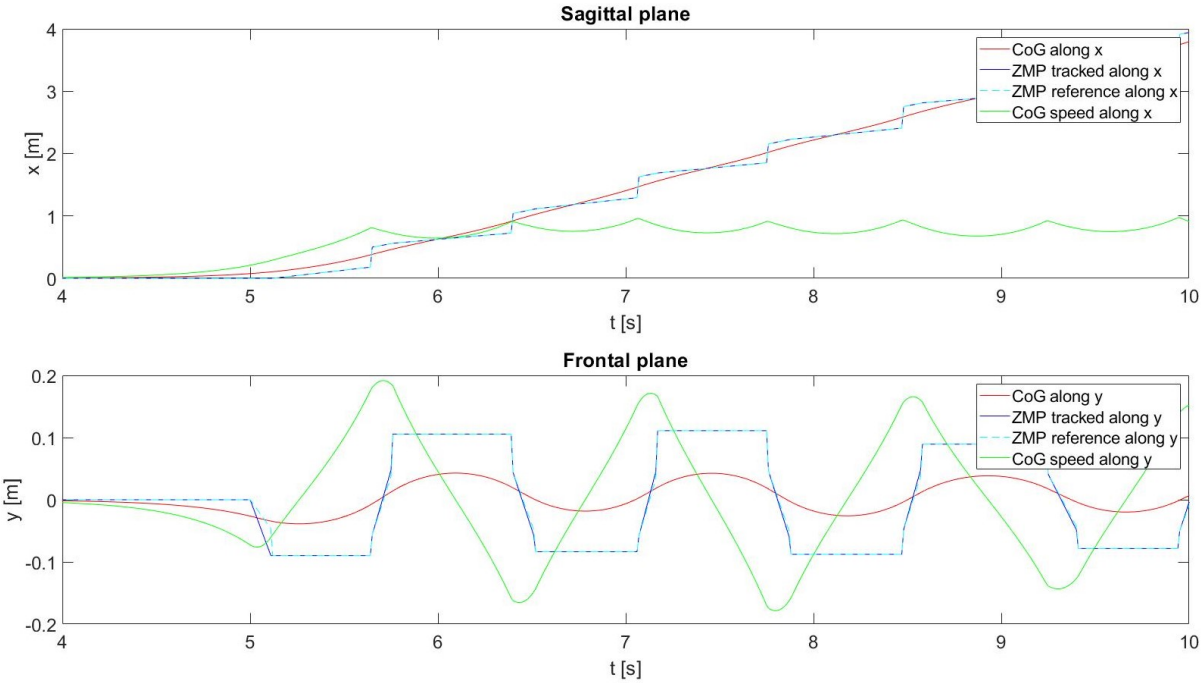
The process outputs are the optimal input u, the output y and the state variables x of the state space system in 6.15. Using the ZMP generated from the support states and the LQR gains in 6.41 the plots in 6.8a and 6.8b are obtained.

$$Q = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix} \quad R = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix} \tag{6.41}$$

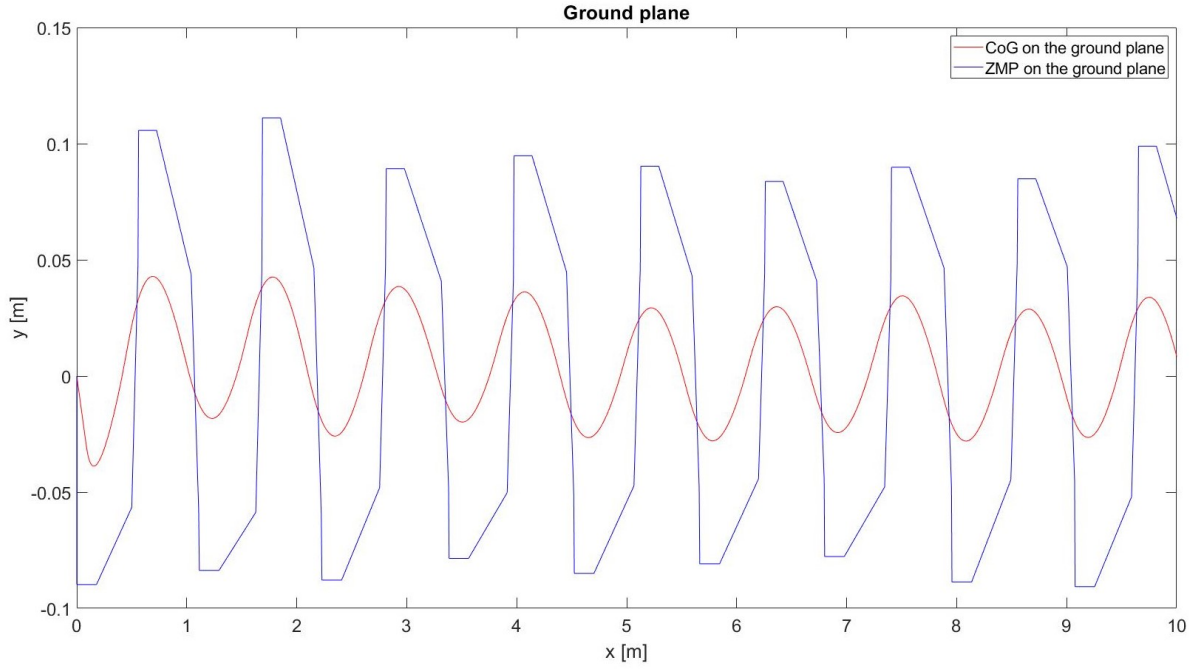
The reference ZMP was perfectly tracked thanks to the use of a rather large time horizon of about sixty seconds. The longer the reference ZMP, the more accurate the tracking.

The motion of the CoG is stable, and never moves outside the line traced by the ZMP, except during the first half step (see Fig. 6.41). Along the longitudinal direction its trajectory seems a straight line, while in the lateral a sinusoid which peaks approach the ZMP during the single support phase.

The velocity of the CoG along x starts from zero, increases, and then begins to oscillate around a constant value, equal to the trial's walking speed, reported in the experimental data. Instead the CoG's speed along y is represented by a a triangular wave with rounded edges, and reach is peak during the double support phase, when the position of the CoG pass from a side the other.



(a)



(b)

Figure 6.8: The comparison of the reference and tracked ZMP with the generated CoG along x, y (a) and on the xy ground plane (b) (subject AB01 and trial s0x8i0).

6.3 Final treatments on reference variables

Before being used as reference variables, the angles of the knee and torso were treated to adapt them to the needs of the simulation. The torso angles taken into account are rotations around the longitudinal x-axis and lateral y-axis of the floor. Torso rotations around z have not been taken into account, because they are small and cannot be considered.

The knees angles were adjusted, adding an offset. Occasionally, joint angles appear to have some measurement error, leading to articular over-extension or a pattern that does not respect physiological limits of the joints. It should be noted that offset values are always around a few degrees.

Both the corners of the knees and the trunk were cut off at the first and last flat-footed events to erase incomplete steps. A first section was added to simulate the body standing erected on both feet, followed by another for the first half-step, connected to the rest of the records, and created using splines. The splines allow a smooth transition from a standing position to walking.

The feet trajectories, and the reference angles are time-derived to obtain the respective speeds, that will be the input of the kinematic cycle. These are packed together with the generated CoG trajectories and speeds in the structure *cartesian_vars*.

Fitting experimental data to a kinematic bipedal model

As previously stated, the aim of this thesis is to develop a system for a lower limbs exoskeleton which is able to move according to the voluntary action of the wearer and capable to guarantee balance during walk.

The generated and treated kinematic variables were used to create a kinematic model of a biped, walking in a rectilinear direction. Some of the variables come from experimental data, others have been generated using LIMP theory, thus following the rules for a balanced gait.

To guarantee the *Whole Body Coordination* (WBC) of the model, i.e. an unambiguous and consistent configuration of the kinematic chain of the biped, it is fundamental that the number of constrain is at least equal to the model degrees of freedom. It is not important whether these constraints are defined partly in Cartesian space and partly in joint space.

In our case the kinematics is redundant, i.e. the number of constraints exceeds the number of degrees of freedom: some of them are defined in Cartesian space, specifically generated to ensure the balance of the biped, and others in the joint space for the synchronized movement of the model with the experimental data. The aim is to check whether it is possible to achieve a movement of the model coordinated with that of the subject of the trial and balanced at the same time.

A dedicated kinematic cycle was created to fit the experimental data and the data generated to the biped model. During a series of cycles, differential inverse kinematics and direct kinematics alternate, making the biped walk in a balanced manner and respecting the constraints imposed on the joints by the experimental input data.

7.1 Kinematic modelling

In order to develop the kinematic cycle, it is necessary to develop a model that can adequately represents the lower limbs' kinematics. This model will be a kinematic chain representing the various body segments and the articulations that links them. We opted for the development of four models, each representing a kinematic chain that starts from the support point and ends with the swinging foot. The models were designed as systems of rigid bodies interconnected by joints, using MATLAB *rigidBodyTree* objects.

Definition 4 (rigidBodyTree)**.** *The rigidBodyTree is a representation of the connectivity of rigid bodies with joints. It is made up of rigid bodies as rigidBody objects. Each rigid*

body has a rigidBodyJoint object associated with it that defines how it can move relative to its parent body.

Four possible support points were considered, right and left heel and right and left toe. In the name of each model, the support point and the wording *rpy* appear, indicating that the ground contact has been modelled with a joint with three rotational DOF (i.e. stays for roll, pitch and yaw):

- robot_model_left_heel_rpy
- robot_model_left_tip_rpy
- robot_model_right_heel_rpy
- robot_model_right_tip_rpy

Model generation is divided into two stages. In the first stage, a Simulink models are created in which the joints are represented with their own reference systems and the bodies that connect them. Reference systems in joints are used to represent the relative movement of the two bodies linked by the joint.

In the second phase, the model is refined by manually adding reference systems not related to the joints, which are used to assess the position and orientation of particular parts of the biped.

7.1.1 Simulink models design and import

The models were designed on Simulink using the following blocks (represented in Fig. 7.1) from Simscape Multibody[1] [44]:

- World Frame:
  - Library: Simscape / Multibody / Frames and Transforms.
  - Description: represents the global reference frame in a model, this frame is inertial and at absolute rest.
- Mechanism Configuration:
  - Library: Utilities.
  - Description: provides mechanical and simulation parameters to a mechanism.
- Solver Configuration:
  - Library: Utilities.
  - Description: specifies the solver parameters that the model needs before you can begin simulation.
- Rigid Transform:

[1]It is a Matlab toolbox that provides a multibody simulation environment for 3D mechanical systems, such as robots, vehicle suspensions, construction equipment, and aircraft landing gear.

  - Library: Simscape / Multibody / Frames and Transforms.
  - Description: applies a time-invariant transformation between two frames.

- Reference Frame:
  - Library: Simscape / Multibody / Frames and Transforms.
  - Description: represents a reference frame with respect to which you can define other frames.

- Revolute Joint:
  - Library: Joints.
  - Description: represents a joint with one rotational degree of freedom.

- Brick Solid:
  - Library: Simscape / Multibody / Body Elements.
  - Description: is a prismatic shape with geometry center coincident with the reference frame origin and prismatic surfaces normal to the reference frame x, y, and z axes.

- Cylindrical Solid:
  - Library: Simscape / Multibody / Body Elements.
  - Description: is a cylindrical shape with geometry center coincident with the reference frame origin and symmetry axis coincident with the reference frame z axis.

- Connection Port:
  - Library: Simscape / Utilities.
  - Description: transfers a physical connection or signal across subsystem boundaries.

- Subsystem:
  - Library: Simulink / Ports & Subsystems.
  - Description: contains a subset of blocks within a model or system.

The development of the Simulink models begins by placing the *World Frame* block, to establish the starting point of the kinematic chain, together with the *Solver Configuration* block and the *Mechanism Configuration* that defines the mechanical parameters (Fig. 7.2a). The convention the World Frame is attached to the joints representing the contact between the support point and the ground. In this way a Cartesian reference is fixed on the contact point, oriented with the y-axis in the walking direction, the x-axis in the lateral direction and the z-axis in the vertical direction. This is called the *Base Frame*, and is the initial reference system of the kinematic chain, with the origin fixed to the contact point, not allowed to rotate w.r.t. the floor. Due to this design choice,

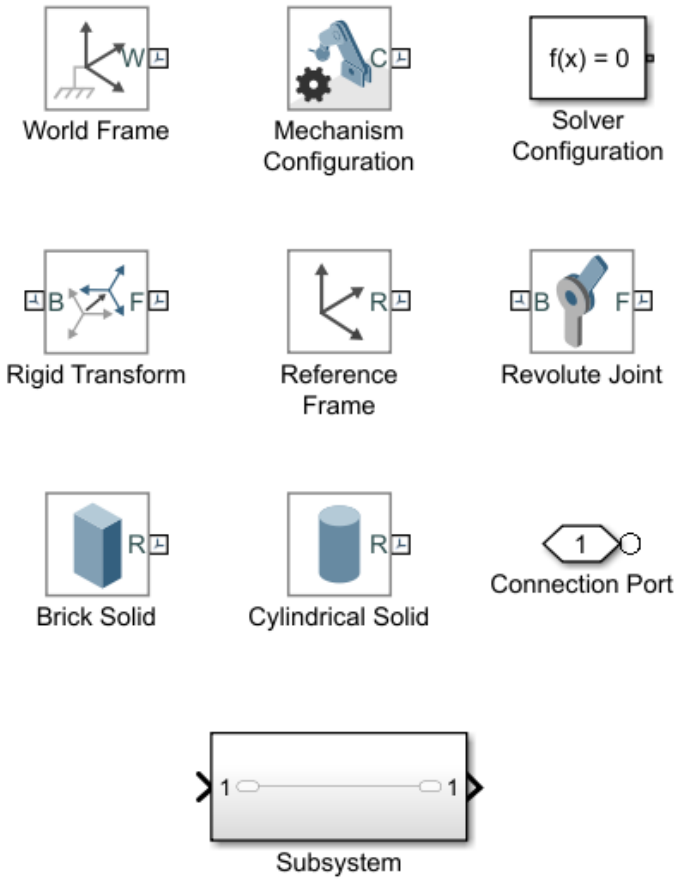


Figure 7.1: Simulink blocks used for the generation of the models

a possible sliding of the support point will not be taken into account in our kinematics simulations.

As a result the rotations perpendicular to the x-y transverse plane will occur around the z-axis (yaw) the rotations perpendicular to the y-z frontal plane around the y-axis (roll) and finally those perpendicular to the z-x sagittal plane along the x-axis (pitch). Tab. **??** shows the rotations that have been applied to the various joints.

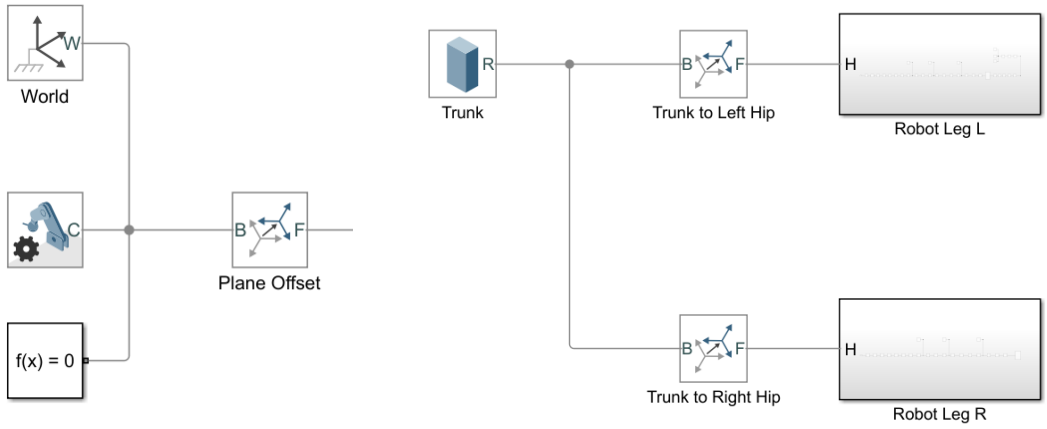
Revolute Joint blocks are implemented to simulate the joints. Each one of these blocks introduces one DOF, thus for simulating joints with more than one DOF they are put in cascade. For example, to simulate the foot-ground contact three rotations are required, so three blocks of this type are combined, oriented along the three directions and centered in the same point. Each *Revolute Joint* block has its own intrinsic reference frame with rotation take place around the z-axis.

To orient properly the joints' rotational axis and to position the joint centers inside the rigid bodies *Rigid Transform* blocks are used. They can handle both translation and orientation transformations, and manage the transition from one joint to another.

After the kinematic chain construction, consisting on the alternation of *Revolute Joint* and *Rigid Trasmorm* blocks, *Brick Solid* blocks are inserted to model feet and torso, and *Cylindrical Solid* blocks are used to model the thighs and tibia. The last two blocks are

only blocks used to have a visual feedback, as meshes. The *Reference Frame* block is used to add additional reference systems, such as the non-supporting foot one.

In the end, using the *Subsystem* and *Connection Port* blocks, the Simulink is divided in sub-blocks, one for each leg, to give the scheme a better look and a organize it in layers (Fig. 7.2b). The figure below shows the model diagram of *robot_model_left_heel_rpy* as an example.



(a) (b)

Figure 7.2: (a) World Frame, Solver and Mechanism blocks (b) Model organization into sub-systems.

Body segment lengths are used within the *Rigid Transform* and the solid blocks to determine translation transformations and mesh sizes. Each solid blocks is set with the dimensions and the weights of the respective body part. The body segment lengths are determined following the antropomethric rules in the section 2.1.2, or as described in the previous chapter, i.e. from the marker analysis. The remaining dimensions have been chosen appropriately for a properly proportioned appearance of the Simulink model. The weights and mass center positions are computed w.r.t. the subject height, using the height normalized values given in [12] and described in the figure 2.4b.

7.1.2 Additional reference frames and model CoG position

Once the Simulink was created, it can be imported in the MATLAB workspace from the Simulink as a *rigidBodyTree* object [43].

The reference frames on the CoG, on the feet' tips and heels and on the meta are added manually to the imported model. The frames are fixed on the model, thus they cannot move from their position inside the kinematic chain, but their motion depends on the movements of the body to which they are attached. For all the added references frames, their orientation was chosen, for convenience, as that of the World Frame (z pointing straight up, y in the traveling direction and x perpendicular to the sagittal plane).

The new fixed reference frame are create and then added to the rigid body tree derived from Simulink. In order to be added in the correct place, a homogeneous transformation

must be defined for each reference frame, which specifies the orientation and distance of one from the other.

Each segment of the rigid body has its own center of mass, and its own weight, inherited from the Simulink model. Finally the CoM position of the model is computed. By knowing its coordinates, it was possible to fix its position in the model, associating it with a reference system.

7.2 Adaptation of variables to the model based on the kinematic cycle

The kinematics with redundancies guarantees the WBC of the model, and is structured to take into account together the Cartesian variables related to the balance of the model while walking, and the angular variables of the joints to guarantee the synchronous movement of the biped with the experimental test data.

Initially, the direct kinematics based on homogeneous transformations will be explained. Then the functioning of differential inverse kinematics in the presence of redundancies will be illustrated, and how its solution can be obtained thanks to the least squares method. Finally these will be combined in a single loop where they alternate at each iteration.

7.2.1 Direct Kinematics

The *direct kinematics* is a function $K(\theta)$ that relates the linear and angular displacements of the model in the Cartesian space to its configuration, i.e. the set of its joint angles θ.

$$x = K(\theta) \tag{7.1}$$

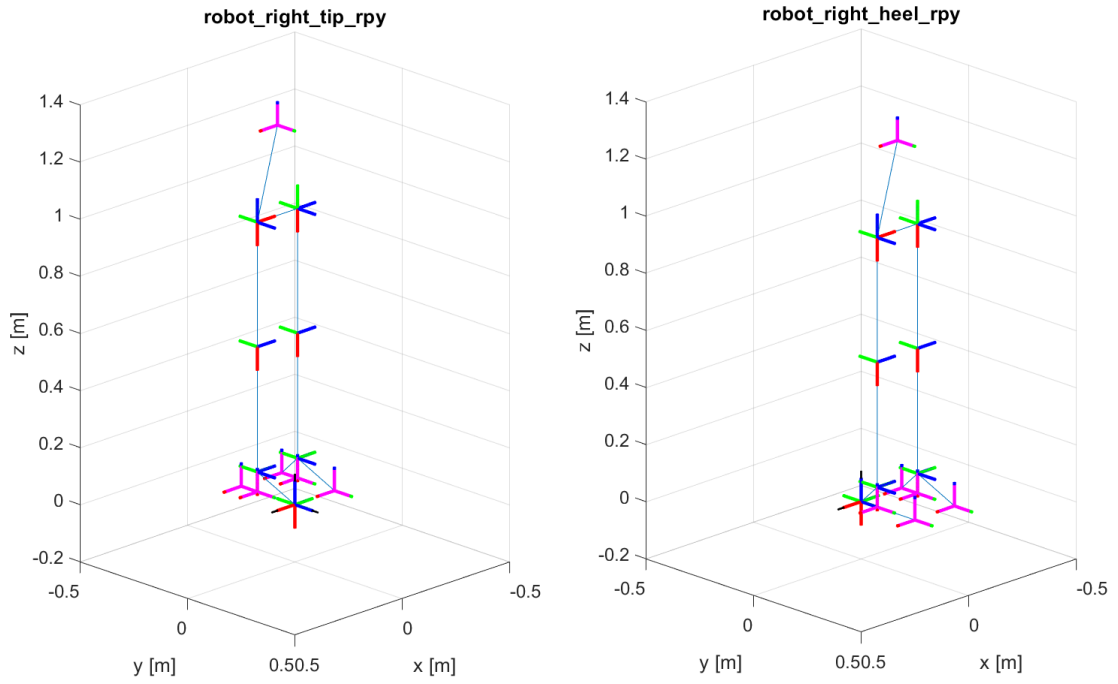
Direct kinematics creates a link between joint space variables θ and Cartesian space variables x. The position of a rigid body in space is expressed in terms of the position of a suitable point on the body w.r.t. a reference frame (translation), while its orientation is expressed in terms of the components of the unit vectors of a frame attached to the body — with origin in the above point — w.r.t. the same reference frame (rotation) [7].

Consider two reference systems RF_0, RF_1, and an arbitrary point P in space. Let p_0 be the vector of coordinates of P w.r.t. the reference frame RF_0. Instead let p_1 be the vector representing the position of P w.r.t. RF_1. Let $d_{(0\to1)}$ be the distance of the origin of RF_1 from the origin of RF_0, and $R_{(0\to1)}$ be the rotation matrix of RF_1 with respect to RF_0. The position p_0 of P in RF_0 is given by:

$$p_0 = d_{(0\to1)} + R_{(0\to1)}p_1 \tag{7.2}$$

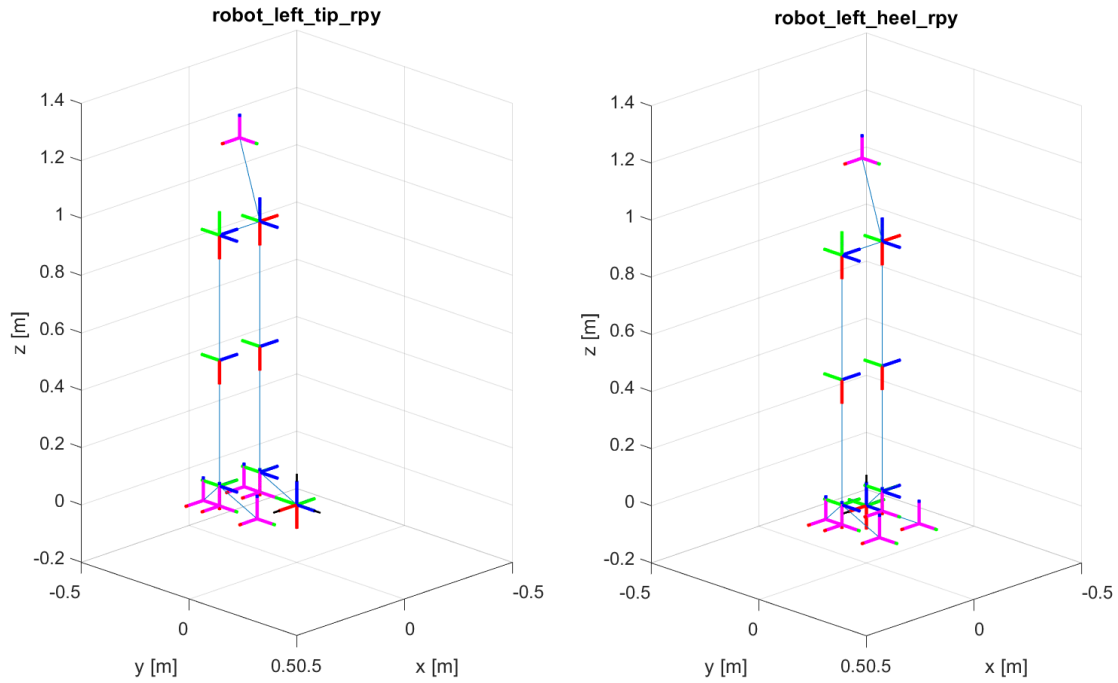
Hence, 7.2 represents the *coordinate transformation* (translation + rotation) of a bound vector between two frames. In order to achieve a compact representation of the relationship between the coordinates of the same point in two different frames, the *homogeneous representation* of a generic vector p can be introduced as:

$$\tilde{p} = \begin{bmatrix} p \\ 1 \end{bmatrix} \tag{7.3}$$



(a)

(b)



(c)

(d)

Figure 7.3: (a) rigidBodyTree model with left heel as support (b) rigidBodyTree model with left tip as support (c) rigidBodyTree model with right heel as support (d) rigidBodyTree model with right tip as support

By adopting this representation for the vectors p_0 and p_1 in 7.2, the coordinate transformation can be compactly rewritten in terms of the *homogeneous transformation matrix* $T_{(0\to1)}$:

$$p_0 = T_{(0\to1)} p_1 \tag{7.4}$$

where:

$$T_{(0\to1)} = \begin{bmatrix} R_{(0\to1)} & p_{(0\to1)} \\ 0^T & 1 \end{bmatrix} \tag{7.5}$$

From the elements of a generic rotation matrix R we can compute the ZYX Euler angles also known as Tait-Bryan angles $\phi_{ZYX} = \begin{bmatrix} \varphi & \vartheta & \psi \end{bmatrix}^T$:

$$R(\phi_{ZYX}) = R_z(\varphi) R_y(\vartheta) R_x(\psi) = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{bmatrix} \tag{7.6}$$

$$\begin{aligned} \varphi &= Atan2(r_{2,1}, r_{1,1}) \\ \vartheta &= Atan2(-r_{3,1}, \sqrt{r_{3,2}^2 + r_{3,3}^2}) \\ \psi &= Atan2(r_{3,2}, r_{3,3}) \end{aligned} \tag{7.7}$$

These represent the rotation angles relative to the axes of a fixed frame w.r.t. happen the rotations.

An homogeneous transformation matrix gives us all the information needed to describe the direct kinematics: it is a suitable instrument for describing the orientation and position of a body in space w.r.t a reference frame, in terms of its ZYX angles ϕ (calculated from R) and its translation vector p, giving all the Cartesian space variables x:

$$x = \begin{bmatrix} \phi \\ p \end{bmatrix} = \begin{bmatrix} \psi \\ \vartheta \\ \varphi \\ p_x \\ p_y \\ p_z \end{bmatrix} \tag{7.8}$$

We have computed the homogeneous transformations of the reference frames of the meta of the supporting foot (F_1), of the model's CoG (CoG) and of the meta of the swinging foot (F_2). These are obtained by the product of a series of homogeneous transformations starting from the *base frame* (i.e. the Cartesian reference system) to the corresponding point:

$$\begin{aligned} T_{(0\to F1)} &= T_{(0\to1)} T_{(1\to2)} T_{(2\to3)} T_{(3\to F1)} \\ T_{(0\to CoG)} &= T_{(0\to3)} T_{(3\to4)} T_{(4\to5)} T_{(5\to6)} T_{(6\to7)} T_{(7\to8)} T_{(8\to9)} T_{(9\to CoG)} \\ T_{(0\to F2)} &= T_{(9\to10)} T_{(10\to11)} T_{(11\to12)} T_{(12\to13)} T_{(13\to14)} T_{(14\to15)} T_{(15\to F2)} \end{aligned} \tag{7.9}$$

where a generic homogeneous transformation $T_{i\in j}$ express how the frame j-th is represented w.r.t. the frame i-th.

Given a model such as those defined in the previous section, we consider fixed the support point of the foot until it is changed. In each joint and point of interest there's a local frame, like described in the previous section. The value 0 indicates the reference frame located on the support point and attached to the terrain, while the other numbers indicates the other reference frames located on the joints. Each of these, except the last of each formula in 7.9 ($T_{(3\to F1)}$, $T_{(9\to CoG)}$ which are constant transformation, $T_{(15\to F2)}$), depends on the i-th joint angle. Thus the position of F_1 depends only on the angles of the support foot w.r.t. the ground. Instead F_2 and CoG position depends also on the other joint variables.

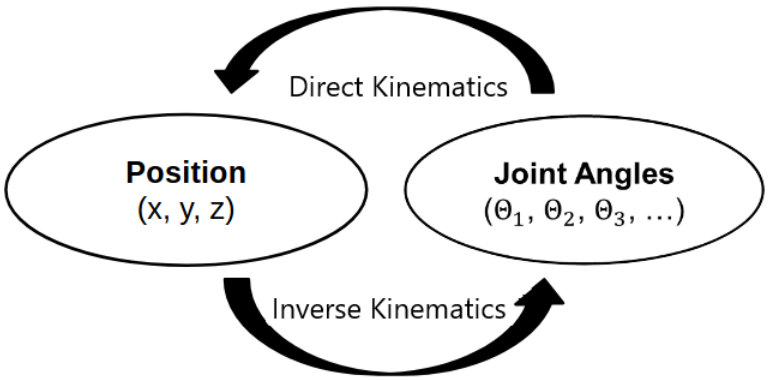


Figure 7.4: Direct and inverse kinematics.

7.3 Differential inverse kinematics in the presence of redundancies

The *differential direct kinematics* gives us the relationship between the joint velocities and the linear and angular velocities of the various points forming the model. It is expressed by the following formula:

$$\dot{x} = J_A(\theta)\dot{\theta} \tag{7.10}$$

where J_A is the analytical Jacobian and:

$$J_A(\theta) = \frac{\partial K(\theta)}{\partial \theta} \tag{7.11}$$

The matrix $J_A(\theta)$ can be obtained from the partial derivatives of the ZYX angles ϕ and of the vector p, coming from both the homogeneous transformation representing the direct kinematics, w.r.t. the joint angles θ. Given a set of three linear and three angular velocities ($\dot{p}$ and $\dot{\phi}$) to be computed, the analytical Jacobian is a 6×6 matrix such that:

$$\dot{x}_{ref} = \begin{bmatrix} \dot{\phi} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} J_\phi(\theta) \\ J_P(\theta) \end{bmatrix} \dot{\theta} = J_A(\theta)\dot{\theta} \tag{7.12}$$

This formula represents the linear relationship between the velocities in Cartesian space and those at the joints.

In our case, we want to track precisely the motion of the CoG, of F_1 and of F_2, the orientation and position of their reference frames in space, and how their time evolution. Their Cartesian linear and rotational velocities are grouped within the vector $\dot{x}_{ref}$, the vector of the reference speeds.

The *differential direct kinematic* that gives the set of reference variables x_{ref} can be written as follows:

$$\dot{x}_{ref} = \begin{bmatrix} \dot{\phi}_{CoG} \\ \dot{p}_{CoG} \\ \dot{\phi}_{F1} \\ \dot{\phi}_{F2} \\ \dot{p}_{F2} \end{bmatrix} = J_{A(15\times15)}(\theta) \begin{bmatrix} \dot{\theta}_{1-F_1} \\ \dot{\theta}_{2-F_1} \\ \dot{\theta}_{3-F_1} \\ \dot{\theta}_{1-Ankle_1} \\ \dot{\theta}_{2-Ankle_1} \\ \dot{\theta}_{Knee_1} \\ \dot{\theta}_{1-Hip_1} \\ \dot{\theta}_{2-Hip_1} \\ \dot{\theta}_{3-Hip_1} \\ \dot{\theta}_{3-Hip_2} \\ \dot{\theta}_{2-Hip_2} \\ \dot{\theta}_{1-Hip_2} \\ \dot{\theta}_{Knee_2} \\ \dot{\theta}_{2-Ankle_2} \\ \dot{\theta}_{1-Ankle_2} \end{bmatrix} \tag{7.13}$$

There the Cartesian speed variables obtained like described in the chapter 6 are related to the joint angles. The matrix J_A is composed by the analytical Jacobians of the CoG, of F_1 and of F_2:

$$J_{A(15\times15)}(\theta) = \begin{bmatrix} J_{\phi CoG}(\theta) \\ J_{PCoG}(\theta) \\ J_{\phi F1}(\theta) \\ J_{\phi F2}(\theta) \\ J_{PF2}(\theta) \end{bmatrix} \tag{7.14}$$

To obtain the redundancy of constrains, the number of degree of freedom has to be lower than the reference variables. Constrain can be added in both joint space and Cartesian space. The knee angles were added on the left side of Eq. 7.13 as additional reference variable, obtaining:

$$\begin{bmatrix} \dot{\phi}_{CoG} \\ \dot{p}_{CoG} \\ \dot{\phi}_{F1} \\ \dot{\phi}_{F2} \\ \dot{p}_{F2} \\ \dot{\theta}_{Knee_1} \\ \dot{\theta}_{Knee_2} \end{bmatrix} = J_{A(17\times15)} \begin{bmatrix} \dot{\theta}_{1-F_1} \\ \dot{\theta}_{2-F_1} \\ \dot{\theta}_{3-F_1} \\ \dot{\theta}_{1-Ankle_1} \\ \dot{\theta}_{2-Ankle_1} \\ \dot{\theta}_{Knee_1} \\ \dot{\theta}_{1-Hip_1} \\ \dot{\theta}_{2-Hip_1} \\ \dot{\theta}_{3-Hip_1} \\ \dot{\theta}_{3-Hip_2} \\ \dot{\theta}_{2-Hip_2} \\ \dot{\theta}_{1-Hip_2} \\ \dot{\theta}_{Knee_2} \\ \dot{\theta}_{2-Ankle_2} \\ \dot{\theta}_{1-Ankle_2} \end{bmatrix} \tag{7.15}$$

where the analytical Jacobian was expanded:

$$J_{A(17\times15)}(\theta) = \begin{bmatrix} J_{\phi CoG}(\theta) \\ J_{PCoG}(\theta) \\ J_{\phi F1}(\theta) \\ J_{\phi F2}(\theta) \\ J_{PF2}(\theta) \\ J_{AKnee1} \\ J_{AKnee2} \end{bmatrix} \tag{7.16}$$

The angular velocities of the knees depend only on the value of the respective joint speed, i.e. on $\dot{\theta}_{Knee_1}$ and on $\dot{\theta}_{Knee_2}$. Thus the matrices J_{AKnee1} and J_{AKnee2} are two 15 elements rows of zeros, except a one in the 6-th position of J_{AKnee1} and another one in the 13-th position of J_{AKnee2}. The analytical Jacobian J_A passed from being square to being rectangular, with more rows than columns. When the Jacobian is square, it can be inverted to obtain the inverse differential kinematics. Instead if it is rectangular and the number of rows exceeds the number of columns, the inversion is not possible, and instead is used the *left pseudo-inverse*[2].

When the number of constrains exceed the number of DOF, the inverse kinematics has not solution. Thus an approximate solution can be obtained using the *weighted least square method*. Multiplying both sides of 7.10 for a diagonal matrix W of weights we have:

$$W\dot{x}_{ref} = WJ_A\dot{\theta} \tag{7.17}$$

The matrix W is a 17×17 square matrix, with different weights for each of the reference variables. The weighted least-square method gives the pseudo-inverse of J_A:

[2]The pseudoinverse $A^\dagger$ of a matrix A depends on then number of rows r and columns c. If $r < c$ and has rank r the *right pseudo-inverse* of A is $A^\dagger = A^T(AA^T)^{-1}$. If $c < r$ and has rank c the *left pseudo-inverse* of A is $A^\dagger = (A^TA)^{-1}A^T$

$$\dot{\theta} = J_A^{\dagger} W \dot{x}_{ref} = (J_A^T W J_A)^{-1} J_A^T W \dot{x}_{ref} \tag{7.18}$$

The least square method gives the solution which minimizes $||WJ_A(\theta)\dot{\theta} - \dot{x}_{ref}||$ and minimizes $||Wx_{ref}||$. The higher the value of a weight, the more the respective Cartesian speed variable given by the result of the least squares method $\dot{\theta}$ will be similar to the respective reference variable, to the detriment of the variables with a lower weight. Solving the inverse kinematics with the computed pseudo-inverse we obtain the joint speeds:

$$\dot{\theta} = J_A^{\dagger} \dot{x}_{ref} \tag{7.19}$$

The joint angles can be obtained integrating the inverse differential kinematic solution $\dot{\theta}$.

7.4 Fitting the data to the model

A series of kinematics cycles is used for fitting the experimental data to the biped model. The fitting process follows the flowchart shown in the figure 7.5. The aim is to ensure that there is simultaneous tracking of the generated variables in the Cartesian space and of the joint angles experimental data. The selected angles are those at the knees joints, but these can be replaced by different ones or others can be added to increase the constraints.

As the number of reference variables is greater than the number of DOF, it is not possible for all of them to be tracked accurately. Thus, the weighted least squares method was used as described in the inverse kinematic section: it enables the choice of which variables should be tracked more precisely by giving them a higher weight.

The experimental data and the data obtained from the processes described in the chapter 6 are both sampled at 100Hz. Let us consider that these forms s_f-sample long recordings, with initial sample s_0. The simulation begins by placing a model in a standing position, with the feet spaced the same distance apart as indicated in the reference data of the foot trajectories at s0. The home configuration is thus determined, which will be used to calculate the first Jacobian $J_A(\theta(s_0))$. It can be noted that the upright position does not correspond to any of the various models defined above, as the feet touch the ground with the entire sole. If both feet of the model touch the ground, a closed kinematic chain is formed, whereas our models correspond to open kinematic chains. The first model selected was the one corresponding to the first support other than both feet lying on the terrain, and the home configuration was established using it.

A global reference system was positioned at the point where the CoG was projected onto the floor, with the longitudinal axis in the direction of the walk, the lateral axis in the direction of the right foot and the vertical axis in the direction of the CoG (the same global frame described in the subsection 2.1.3). The distances between the origins of the model's base frame and the global frame just defined are calculated. These distances allow the change of coordinate system from the base frame to the global reference frame, by adding them to the value of the variables represented in the local base frame.

The values of the elements of the diagonal matrices W, G and G_i are then chosen. The first is the already mentioned matrix of weights used in the least squares method, the others will play a role in correcting the reference variables from their tracking error.

W, G and G_i are to be chosen for best results in fitting inverse kinematics with constrain redundancy.

The series kinematic cycle begins at the second sample ($s_0 + 1$), involves alternating differential inverse kinematics and direct kinematics, and continues until the last sample s_f is reached. In the first step of the cycle, the state of the support is determined from the state vector obtained as shown in section 6.1. According to the determined state, the proper model will be selected.

The values of the elements of J_A are calculated according to the chosen model and the current configuration. The current configuration is the value of theta obtained at the previous iteration of the cycle ($\theta(s-1)$). Depending on the support state, the kinematic chain, its starting frame anchored to the ground contact point and the swinging foot change. This is followed by the resolution of the inverse kinematics thanks to the least squares method (Eq.7.18). The solution $\dot{\theta}$ does not guarantee a perfect tracking of all reference speeds, but a better tracking of the variables with a high weight in W, at the expense of those with a low weight in W.

By integrating $\dot{\theta}(s)$, the next configuration $\theta(s)$ is obtained. With the an apposite MATLAB command we obtain the matrices of the homogeneous transformation of the CoG, $F1$ and $F2$ reference frames w.r.t. the base frame. The homogeneous transformations can be used to determine the vector $x(s)$ containing the Cartesian variables of interest, as explained in the section 7.2.1, dedicated to direct kinematics.

It should be noted that θ is not the same for all the four models, i.e. it differs according to the kinematic chain. A matrix containing the model configurations at each sample was defined for each model: depending on the support point we have θ_{L-H} (left heel support), θ_{L-T} (left tip support), θ_{R-H} (right heel support), θ_{R-T} (right tip support). This matrix is an n*m, where n is the number of configurations variables and m is the number of samples. The first three values of a columns depend on the rotations on the support point. The value of θ does not change when selecting a new model with the base frame located on a support point on the same side of the body ($\theta_{L-H} = \theta_{L-T}$ and $\theta_{R-H} = \theta_{R-T}$). Instead when the stance leg is swapped, the configuration vector changes along with the model. The three values of the configuration depending on the orientation of the new stance foot are calculated from the ZYX angles of its meta w.r.t. the base frame of the previous model, while the remainder are the same but with their order is reversed.

To make the Cartesian quantities converge in time to the values of the reference trajectories, a proportional integral loop is closed on the Cartesian positions. The reference speeds are corrected with the reaction:

$$\dot{x}_{ref}(s+1) = \dot{x}_{ref}(s+1) + G(x_{ref}(s) - x(s)) + G_i \int_0^t (x_{ref}(s) - x(s))dt \tag{7.20}$$

where G and G_i are diagonal matrices of gains and $(x_{ref} - x)$ is the tracking error. High gain values impose a higher error consideration, and therefore more severe correction. The presence of the integrating action is justified by the fact that the integration in time of the velocities leads to drifts in time.

When a model with a different support point than the previous one is selected, the distance to the global reference frame is recalculated, adding the distance between the

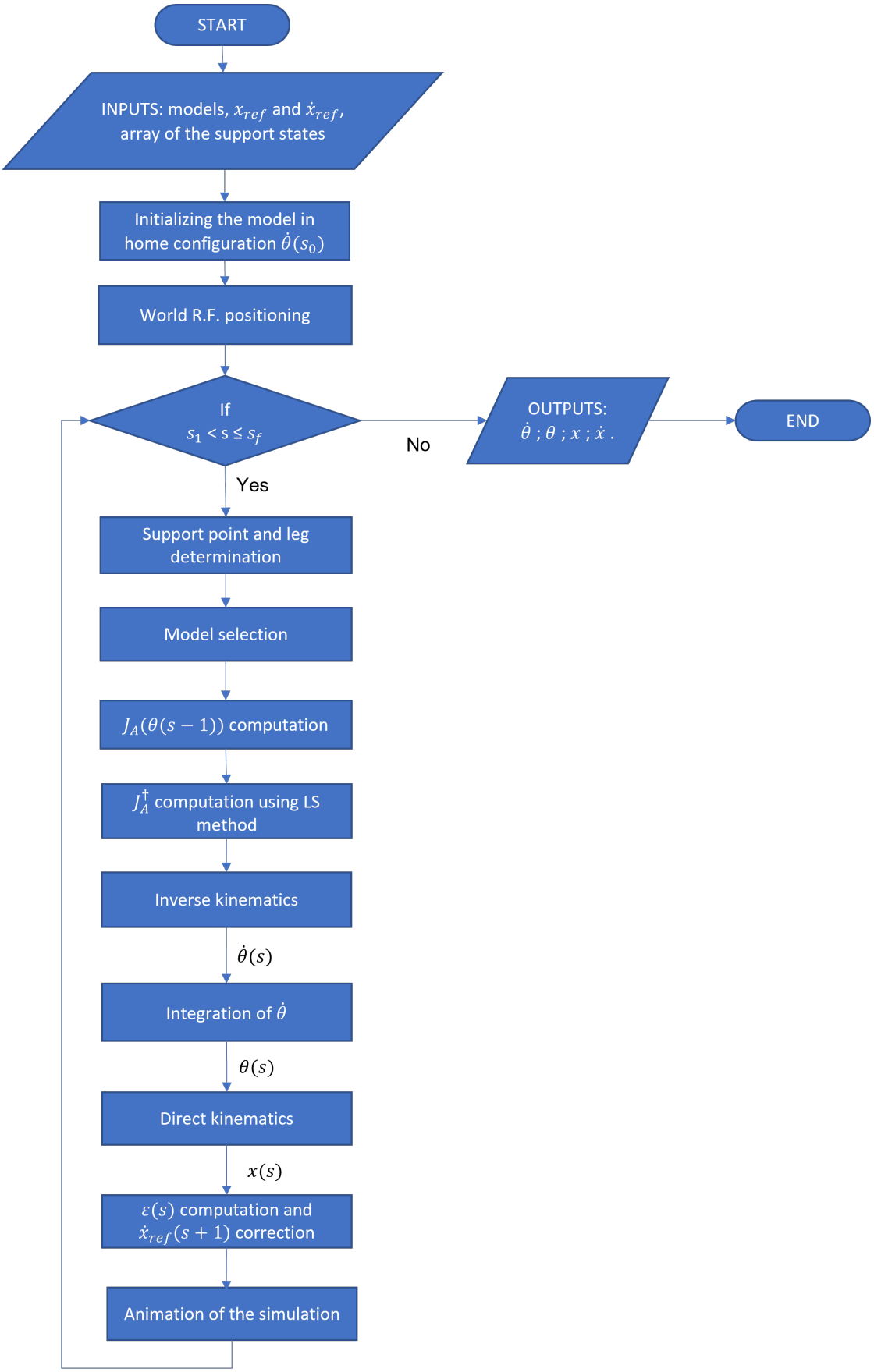


Figure 7.5: Flow diagram representation of the fitting process.

new and old support point. The global frame distance is added to the variables in the local reference frame (the base frame), obtaining a representation in the global one.

When changing support leg, the angular velocities of the joints at the new support point are not available. These are obtained by deriving the values of the angles in time obtained previously for the new support foot.

7.4.1 Simulation and results

The results shown here are obtained from experimental data from trial s1i0 of the subject AB01, from which the simulation inputs and models were derived.

The simulation inputs are the reference speeds $\dot{x}_{ref}$, and the respective angles and position contained in x_{ref}. The reference velocities are the angular and linear velocities of the CoM and of the feet, and the rotational speed of the knees. The values of the linear velocities of CoM and feet, and the angles of the feet around the lateral y-axis were obtained as explained in chapter 6. The torso angular speeds about the y-axis and the z-axis, and the knee angular speeds were derived directly from the experimental data. To complete the set of reference variables, the angular speeds of the feet around the axes x and z and those of the trunk around z are missing. These have been set to zero because no lateral oscillation of the feet due to a prone-supination action is wanted, and the rotations around z are small and cannot be considered while walking in a straight line.

The values chosen for the gains (for the data coming from the trial s1i0, subject AB01) are as following:

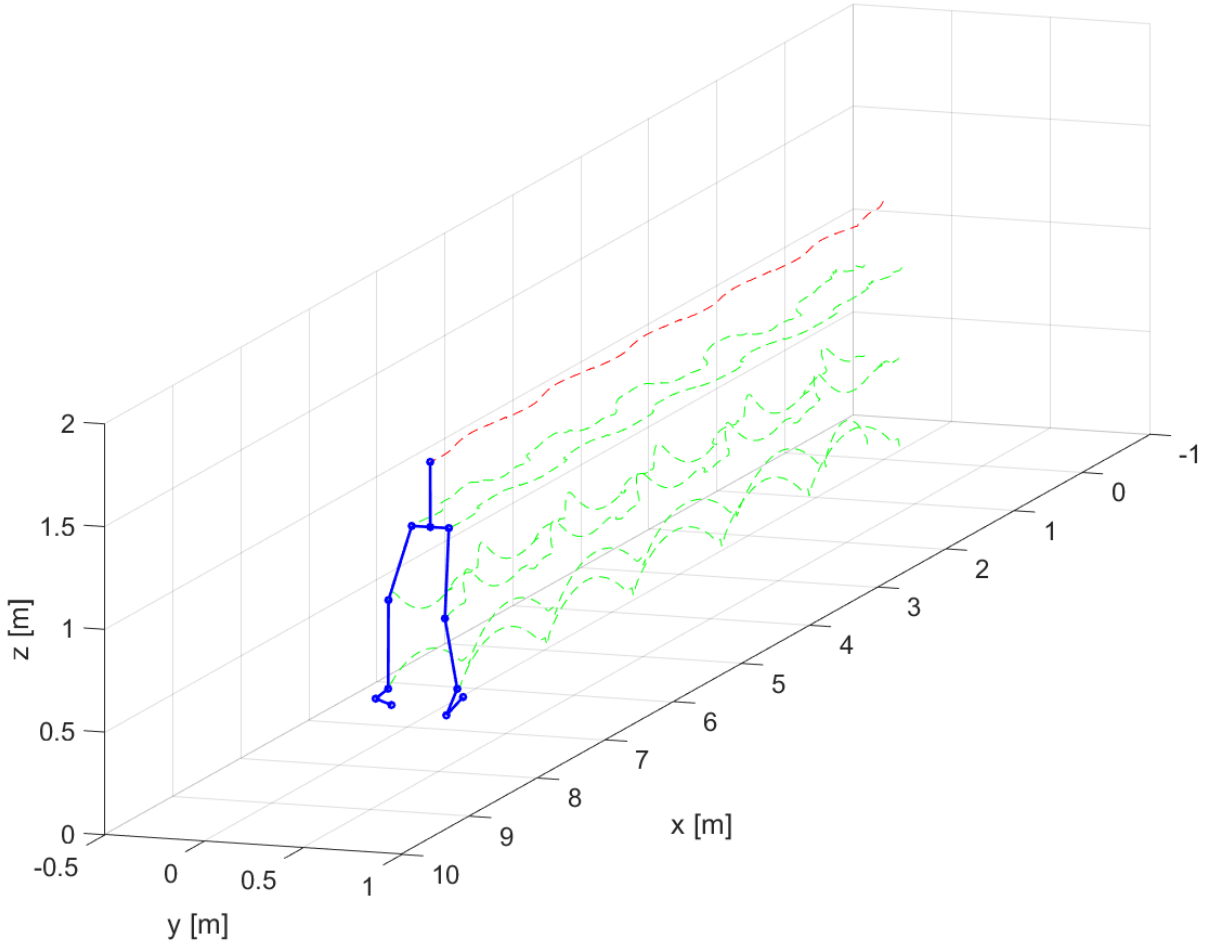
$$\begin{aligned} W =& diag(10^2, 10^2, 10^2, 10^3, 10^3, 10^3, 10^3, 10^3, 10^3, 10^2, 10^4, 10^4, 10^3, 10^2, 10^9, 70, 70); \\ G =& diag(2, 3.5, 3, 3, 1, 3, 0.5, 0.8, 1, 1, 1, 1, 1, 1, 5.3, 0.6, 0.6); \\ G_i =& diag(0.1, 0.15, 0.4, 0.1, 0.1, 0.3, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.61, 0.01, 0.01); \end{aligned} \tag{7.21}$$

These have been chosen to ensure good tracking (the perfect is impossible) of all reference variables.

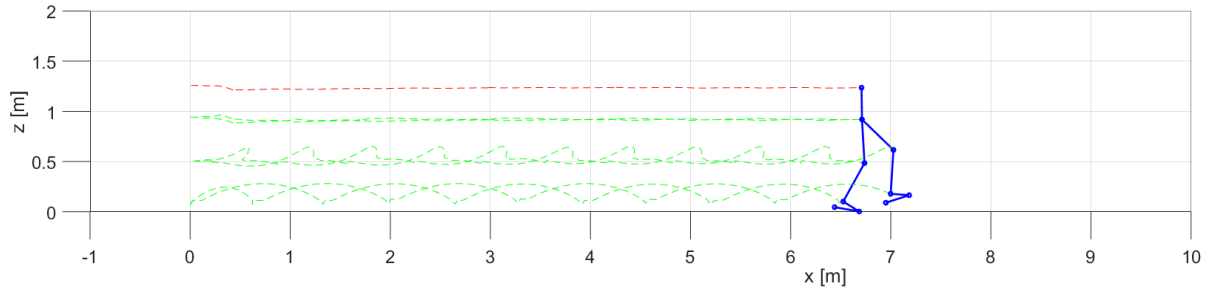
It was not possible track precisely the input angles ar the knees and at the same time of the trajectories of the CoM and of the feet. Other gains were also tried, in an attempt to obtain a pattern of the knee angles similar to those in x_{ref}: unfortunately it was not possible to obtain a good result, and the other reference variables deviated significantly from those in the data. This could be caused by a discrepancy in the origin of some Cartesian variables w.r.t. others. As we already mentioned, some are obtained from the trial measurement while others are generated according to the LIMP model. When the variables from the dataset are valued by giving them high weights, this is to the detriment of the variables generated, and vice versa. Finally, with the resulting data, a real-time animation was developed to show the evolution of walking over time.

In the following pages are shown the reference variables compared with the respective outputs of the fitting. The only ones not represented are the foot angles around z and y, with their respective speeds. These variables are rather small and negligible, and do not vary much w.r.t. the null references. In general, the plots of reference trajectories and angles have two lines: one named "Reference", which refers to the simulation inputs, and one "Effective", which represents the actual movements made by the biped (i.e. the

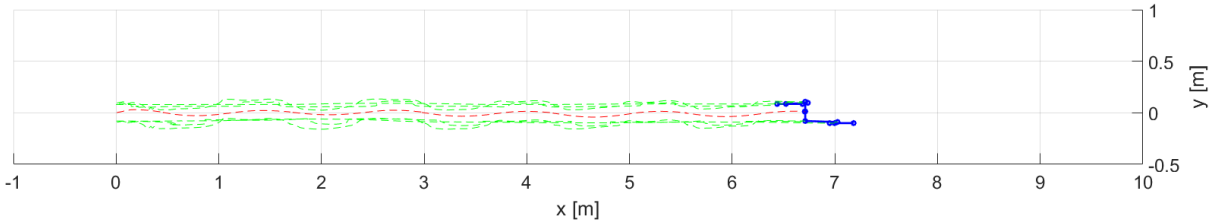
outputs of the fitting process). On the other hand, as regards the speed graphs, a third line, "Corrected reference", is added to the above-mentioned ones, representing the input speed corrected by the feedback in 7.20.



(a)



(b)



(c)

Figure 7.6: Biped walker animation and respective lateral view (b) and top view (c). The position of the CoM (red line) and of the joint centers (green lines),

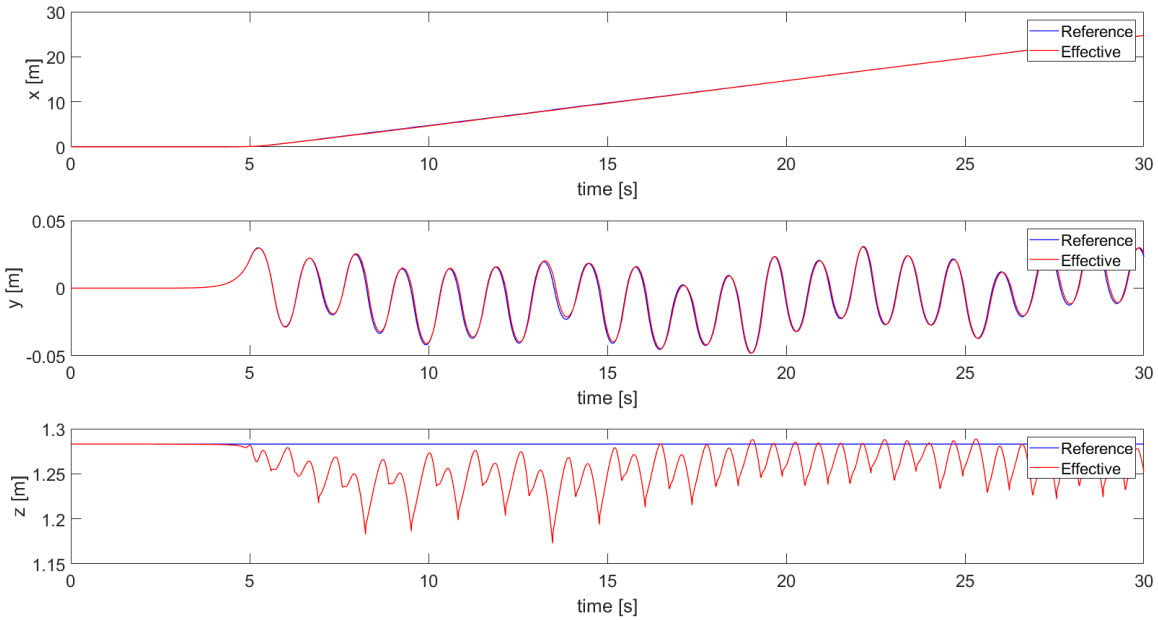


Figure 7.7: From top to bottom: CoM trajectory along x-axis, y-axis and z-axis.

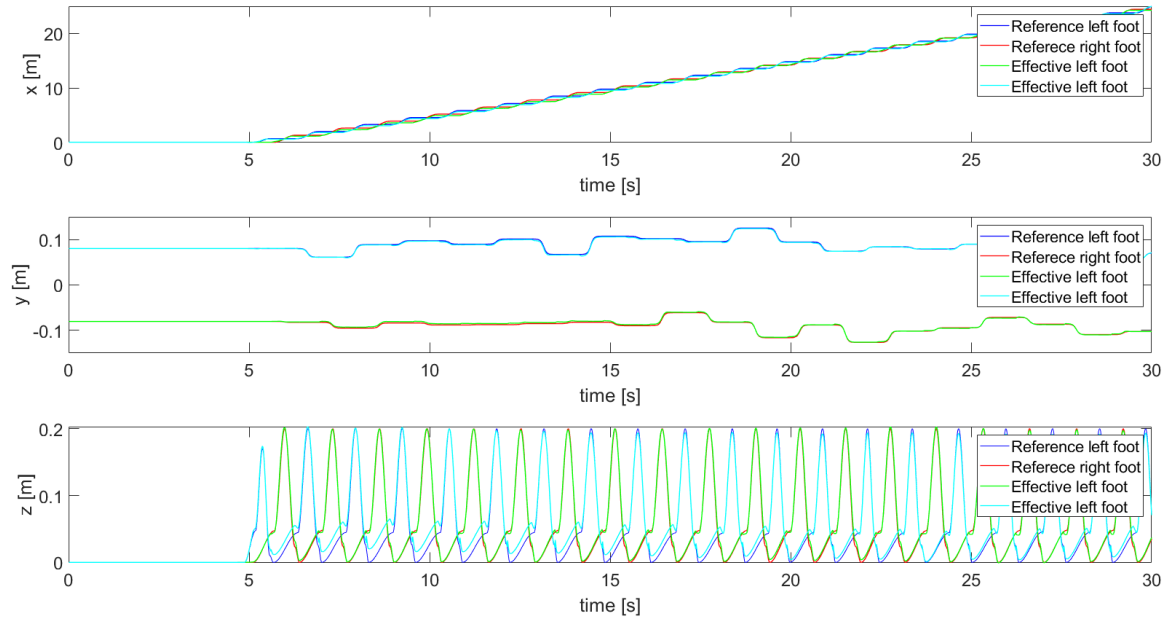


Figure 7.8: From top to bottom: feet trajectories along x-axis, y-axis and z-axis.

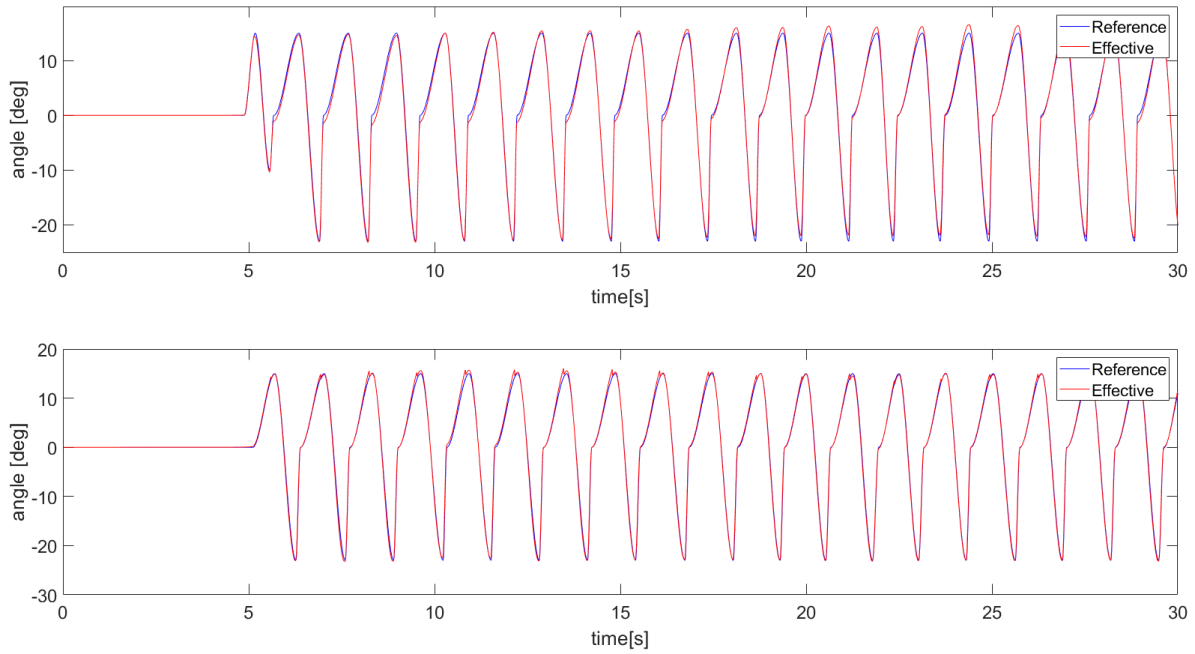


Figure 7.9: From top to bottom: left and right feet angle around y-axis.

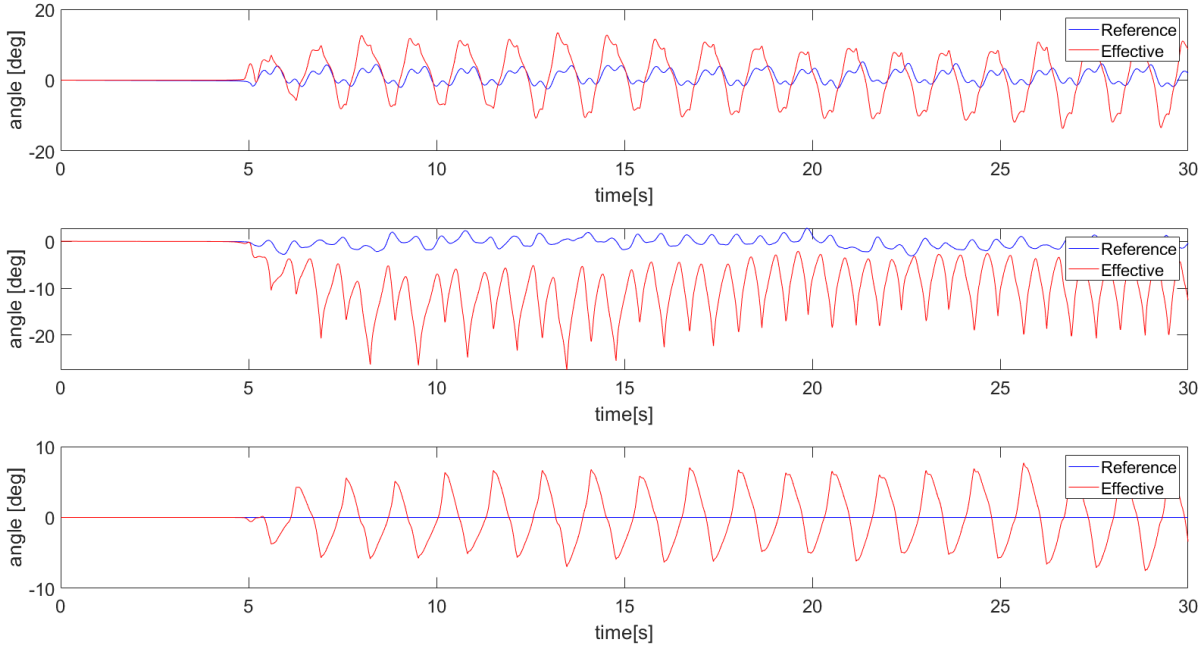


Figure 7.10: From top to bottom: trunk angles around x-axis, y-axis and z-axis.

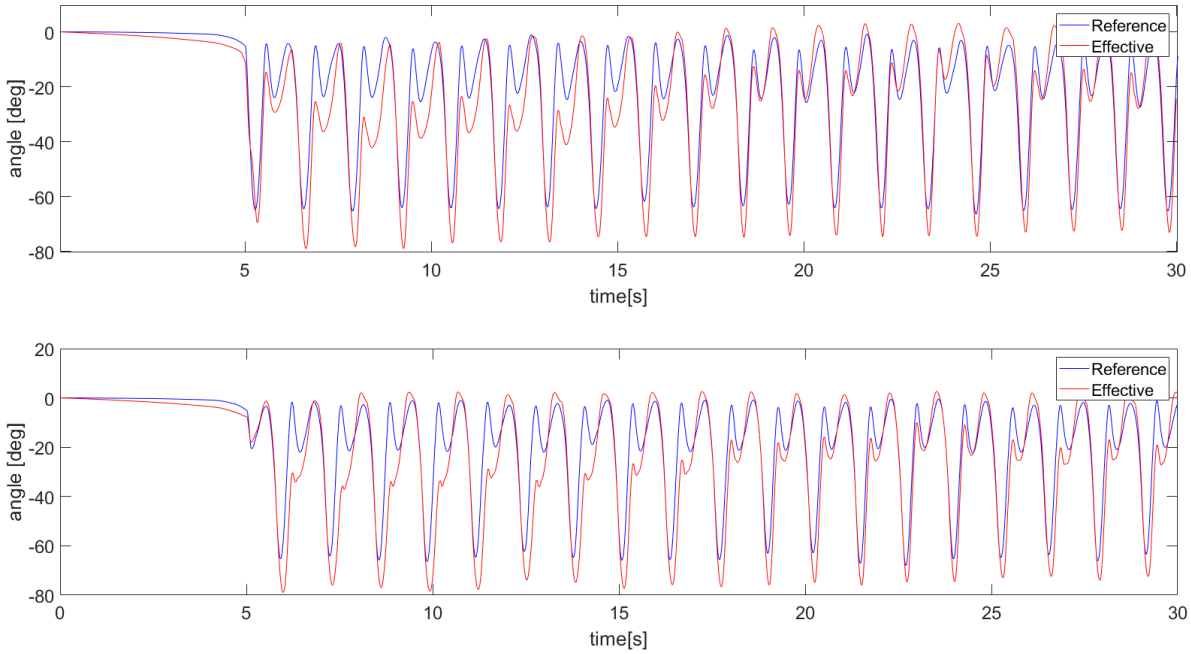


Figure 7.11: From top to bottom: left and right knee joints angles.

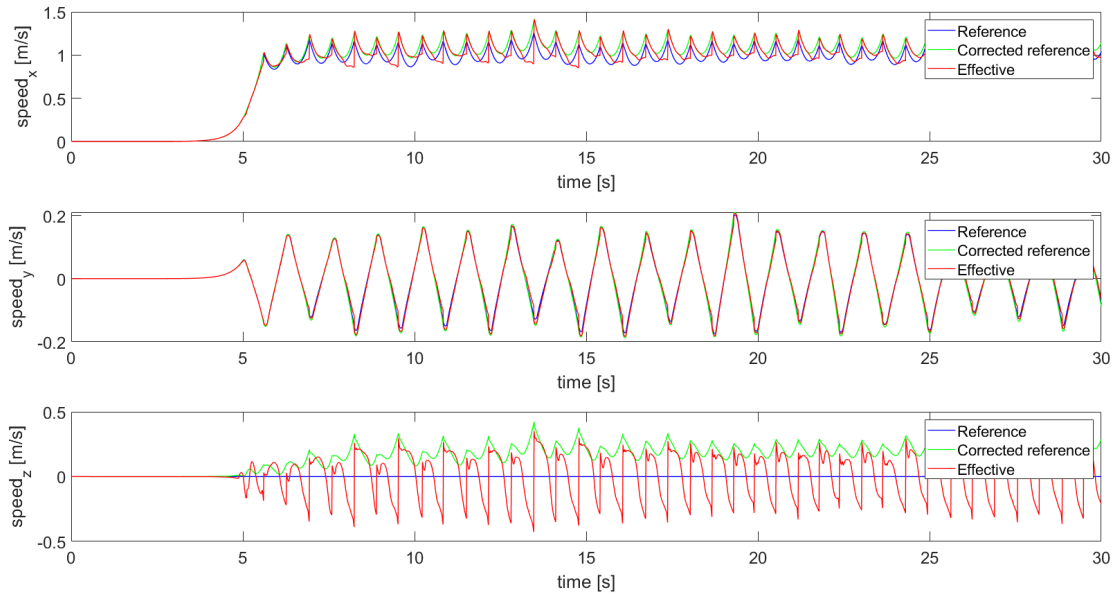


Figure 7.12: From top to bottom: CoM speed along x-axis, y-axis and z-axis.

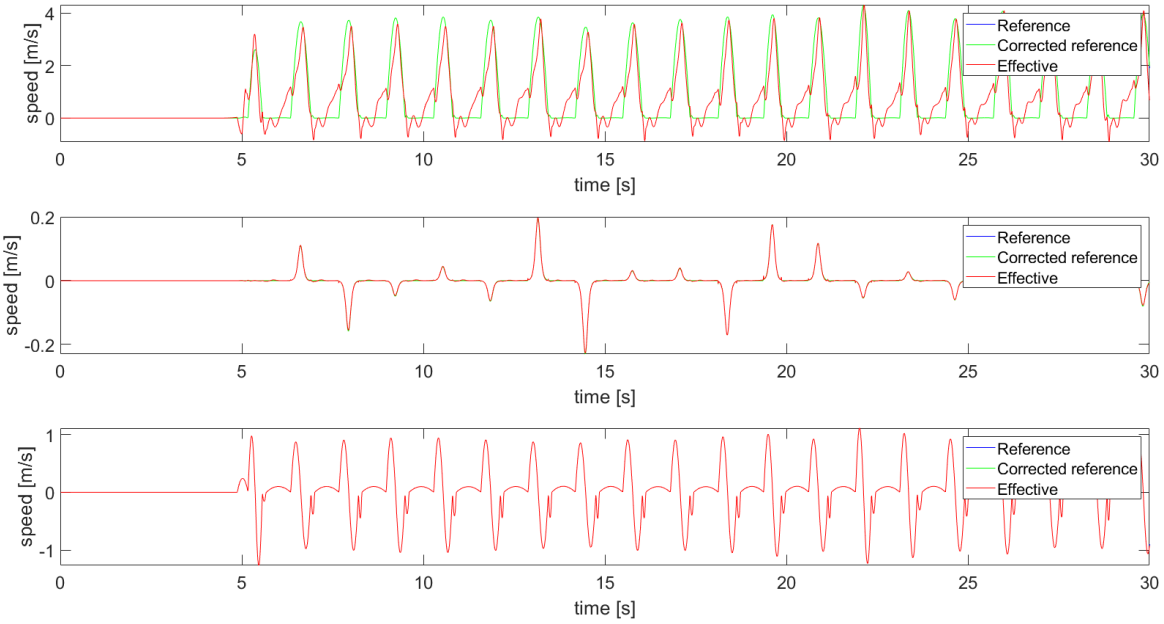


Figure 7.13: From top to bottom: left foot speed along x-axis, y-axis and z-axis.

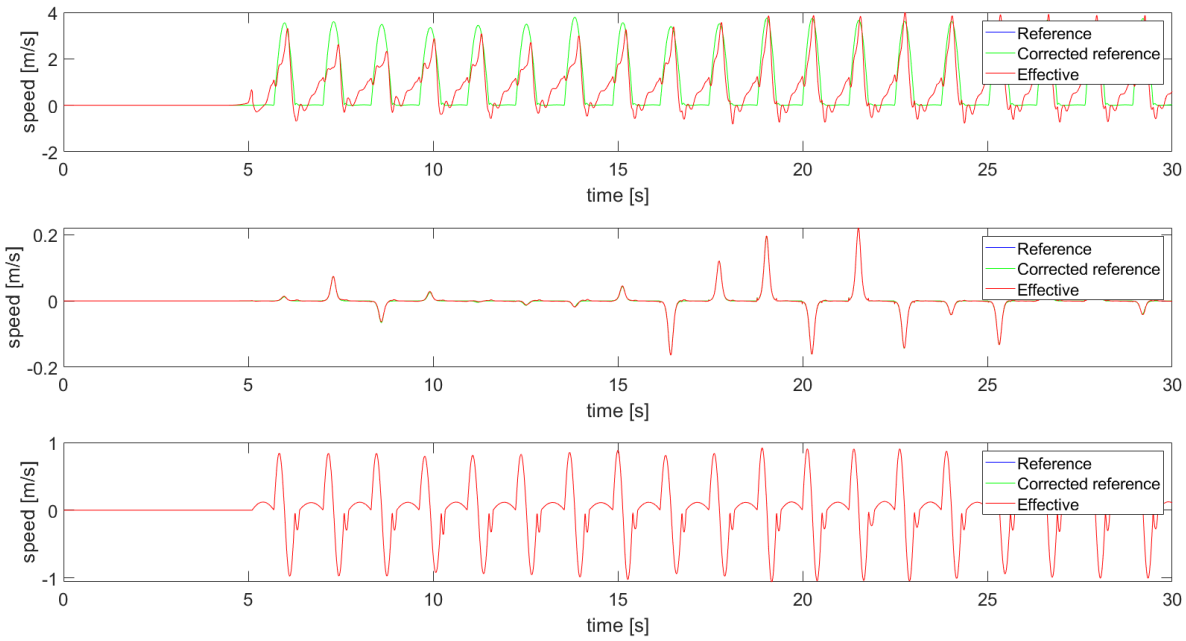


Figure 7.14: From top to bottom: right foot speed along x-axis, y-axis and z-axis.

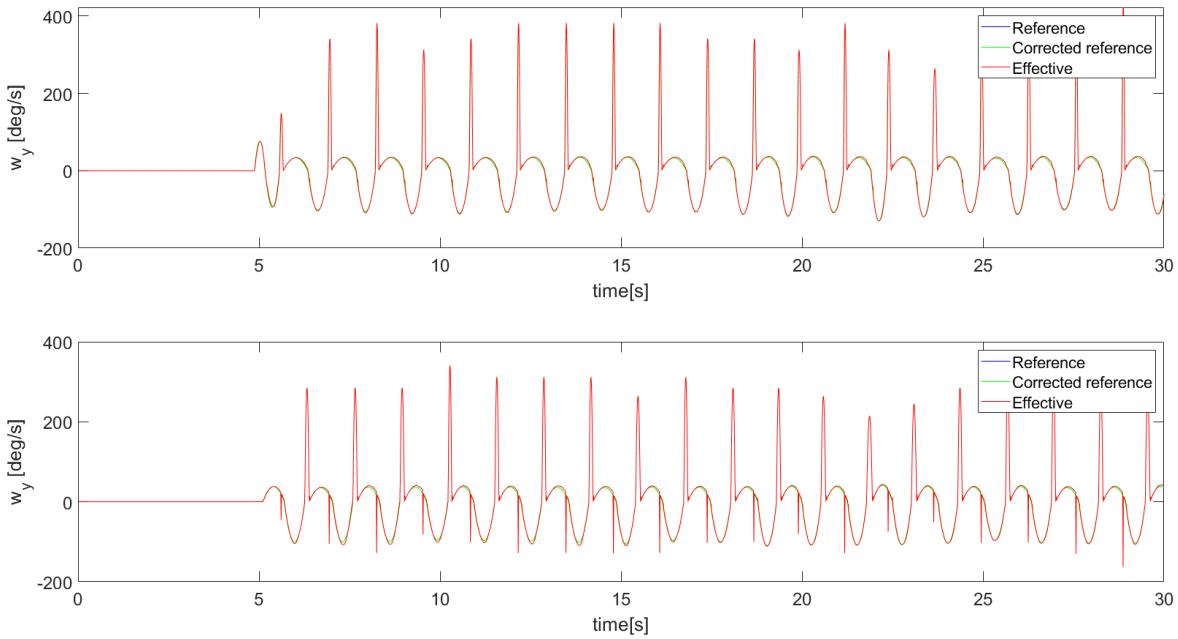


Figure 7.15: From top to bottom: left and right feet angular speeds around y-axis.

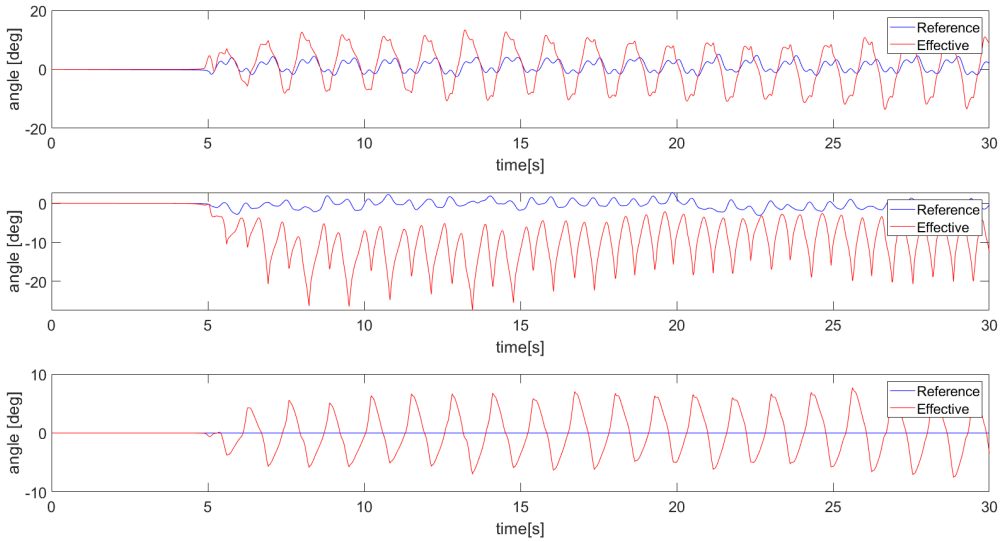


Figure 7.16: From top to bottom: trunk angular speeds around x-axis, y-axis and z-axis.

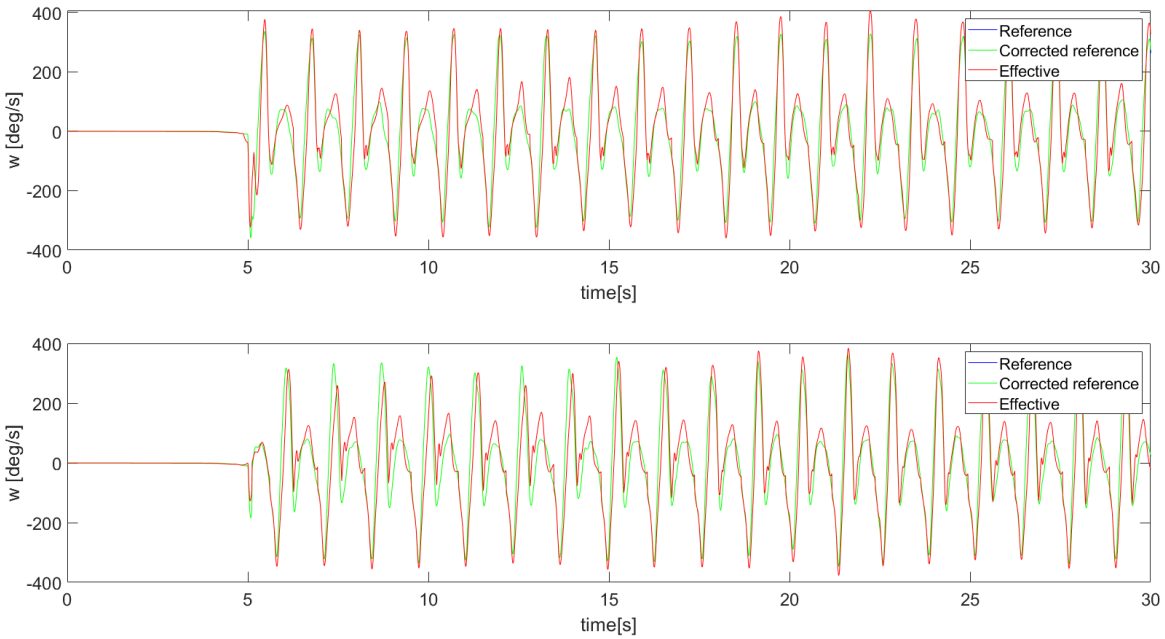


Figure 7.17: From top to bottom: left and right knee joints angular speed.

Neural Networks Performances

As mentioned in the introduction, we can distinguish exoskeletons into two significant subgroups: passive and active exoskeletons for rehabilitation. Passive exoskeletons are those that do not involve a contribution from the patient but are limited to the execution of a certain task that is already defined before being performed. while active exoskeletons are able to engage the patient neurologically. This interaction between man and machine is achieved by involving biometric signals in the exoskeleton's operating and control system. In our case, the signals taken into consideration are electromyographic or EMG signals.

These signals, of an electrical nature, are directly involved in the muscular contractions performed by the human body to execute a certain movement. It is therefore natural to take them into consideration when it comes to finding a quantifiable signal for human movement. There are already exoskeletons, such as HAL, that use these signals to pursue a rehabilitation that is neurologically effective. Therefore, in this chapter, we will discuss the construction of a possible model that is able to relate the EMG signals to the angular positions of the joints of the lower body limbs. The model proposed for this purpose is a neural network [28].

8.1 EMG signals

8.1.1 Action potential, motor unit and disturbances

All cells can generate electrical potentials at rest. Still, some cells, known as excitable cells, do not just respond passively to electrical stimuli but respond actively by generating an active response is called an action potential. If we consider a nerve cell, it has a resting membrane potential of -70 mV at rest. Therefore in its resting condition the membrane of the cell is polarized because the cytoplasmic side is negatively charged compared to the negatively charged compared to the positively charged outside.

In excitable cells such as nerve cells, if the depolarizing stimulus stimulus exceeds a different threshold value for each cell type, the cell's active response is triggers in the form of action potentials. Once the membrane has reached the threshold, a very rapid depolarization occurs, exceeding zero until polarity is reversed. The peak of the action potential reaches +50 mV. Subsequently, the membrane potential returns to the values of the resting potential almost as quickly as it depolarized. If the cell receives a depolarizing stimulus (entry of positive charges into the cell) high enough to shift the membrane potential to a threshold potential, the action potential is triggered. potential, the action potential is triggered, which is an "all-or-nothing" self-regenerative response, i.e. the

process proceeds spontaneously and does not stop until it returns to its initial conditions.

That is, the potential begins to increase rapidly, crosses zero, reverses the sign and reaches a maximum value (between +30 and +65 mV), the overshoot. If a more intense stimulus is applied than is sufficient to reach the threshold, the amplitude and shape of the action potential do not change. In practice, a stimulus is either incapable of eliciting an action potential (sub-threshold stimulus) or causes a full action potential. The set of action potentials constitutes the motor unit activation potential.

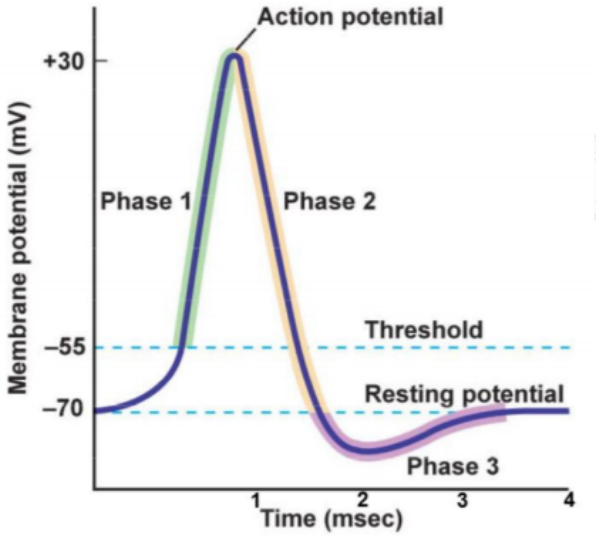


Figure 8.1: Evolution of an action potential

Motor unit activation potential

In the human muscular system, the strain and tension of a muscle is directly proportional to the intensity of the corresponding EMG signal. Each muscle fiber generates its own *motor unit action potential*,(m.u.a.p.) , all these peaks of myoelectric activation combined together make up the EMG signal.

Definition 5 (The notion of a motor unit action potential)**.** *A motor unit (m.u.), represents the anatomical and functional element of the neuromuscular system. They are formed by the alpha spinal motor neuron and its innervated set of muscular cells. The electrical changes that occur during muscular movements are called motor unit action potential.*

Chemical references

Chemically, when an m.u.a.p. starts, there is a release of Ca^{2+} ions in the sarcoplasmic reticulum. these ions rapidly diffuses to the contractile filaments of actin and myosin, where ATP[1] is hydrolyzed to produce ADP[2] plus heat plus mechanical energy (tension).

[1]Adenosine triphosphate is the main molecule for storing and releasing energy when needed

[2]Adenosine diphosphate is a nucleotide of ATP that has released a phosphate group releasing energy

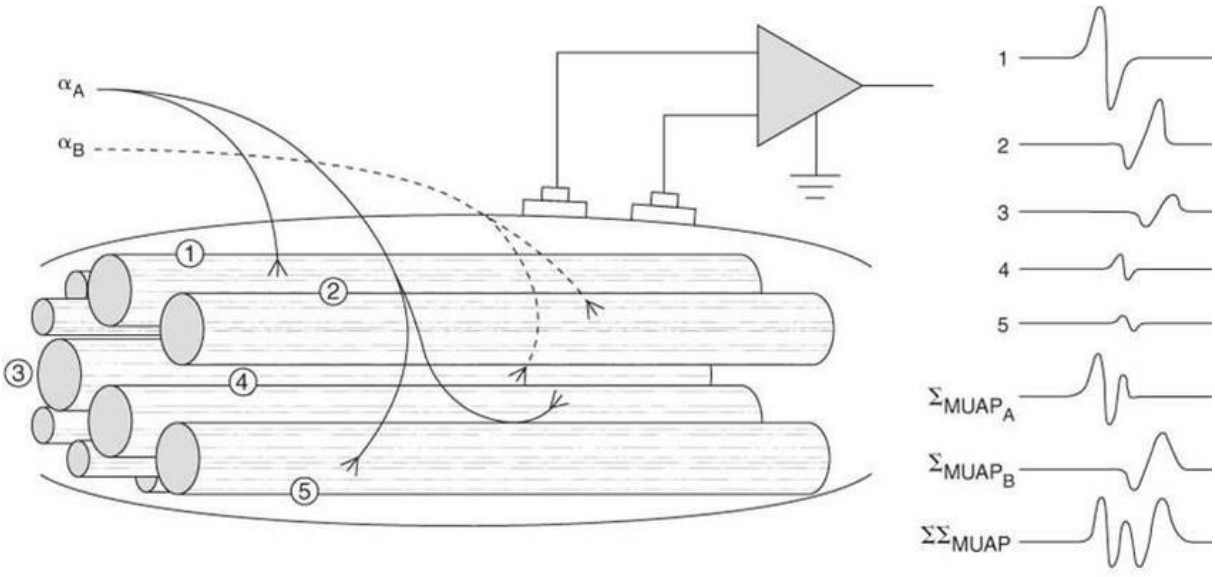


Figure 8.2: MUAP recorded by surface EMG electrodes

The mechanical energy manifests itself as an impulsive force at the cross-bridges of the contractile element [49].

Disturbance factors for EMG signal acquisition

During the recording of the EMG signal, several factors can occur that disturb the acquired signal [9].

- Cross-talk: in addition to taking the action potential of the desired muscle, the electrodes also record the potentials of neighboring muscles or the residual disturbance of the previous action potential.
- Electrode placement and size: during placement, the electrodes may not have adhered well to the surface. They may also be the wrong size to record the activity of a given muscle.
- Acquisition dynamics: there is a possibility that the distance between the signal origin and the detection point varies over time.
- Surrounding environment: the environment in which the acquisition is made may be electronically noisy, affecting the quality of the recording.

Since the aim is to use the signals to train the neural networks, EMGs have to be processed before their use. Eliminating unwanted signal sections, extracting the main activations and giving them a less noisy appearance to be more interpretable for the network This is done by calculating the envelope and muscle synergies.

8.1.2 Principal activation

Principal activations are those activations that are strictly necessary to perform the motor task, then there are secondary activations, which have an auxiliary function, such as correcting the movement and posture of the body segment [16]. In order to simplify the EMG signals, one possibility is to extract the main activations and to reduce the contribution of the secondary activations to zero.

Principal activation for Biolab dataset

In this dataset, the activations were already provided. They were calculated by means of the CIMAP algorithm (Clustering for Identification of Muscle Activation Patterns), a methodology developed by the Biolab research laboratory of the Politecnico di Torino. The algorithm is based on hierarchical clustering, groups together the gait cycles sharing similar EMG onset-offset activation patterns. For each cluster, the cluster prototype is defined as the median timing pattern. Then, principal activations are extracted from the representative clusters, computed as the intersection of the clusters' prototypes [16].

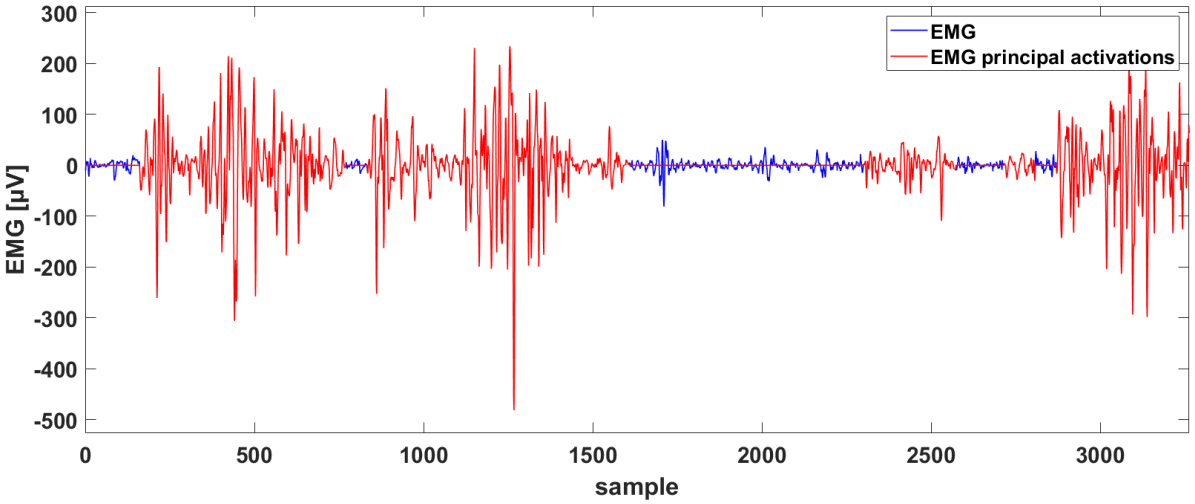


Figure 8.3: Principal activation and raw EMG of Biolab dataset

Principal activation for Incline Experiment dataset

In this dataset, the EMG signals were previously filtered with a low-pass filter (cut-off frequency 40 Hz) and rectified. Due to this treatment, it was not possible to use CIMAP to extract principal activation from this data as well, as the algorithm network was not trained to analyze strictly positive signals. A double threshold detector was used to calculate the activation in this case.

The detector uses two parameters to identify the signal tracts to be eliminated. The first is a threshold calculated by taking into account a signal stretch present between two peaks, which is then considered as noise only, and the mean plus K times the standard deviation of the noise signal is calculated. K is a number that is typically chosen between 3 and 5 simply by testing, it can take higher values if necessary.

$$T = \bar{x} + K\sigma \tag{8.1}$$

$$\bar{x} = \frac{\sum_{i=1}^{N}(x_i)}{N} \tag{8.2}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N}} \tag{8.3}$$

In eq. 8.1, 8.2 and 8.3 x_i is the part of the signal identified as noise. The second parameter is a temporal one, i.e. once a set of values below the threshold has been identified, to be considered as noise and therefore eliminated, these values must have a sufficient time duration to suggest that it is indeed just noise. In this way, long stretches of the signal below the threshold are discarded, while short stretches between peaks, which are part of the activation, are retained.

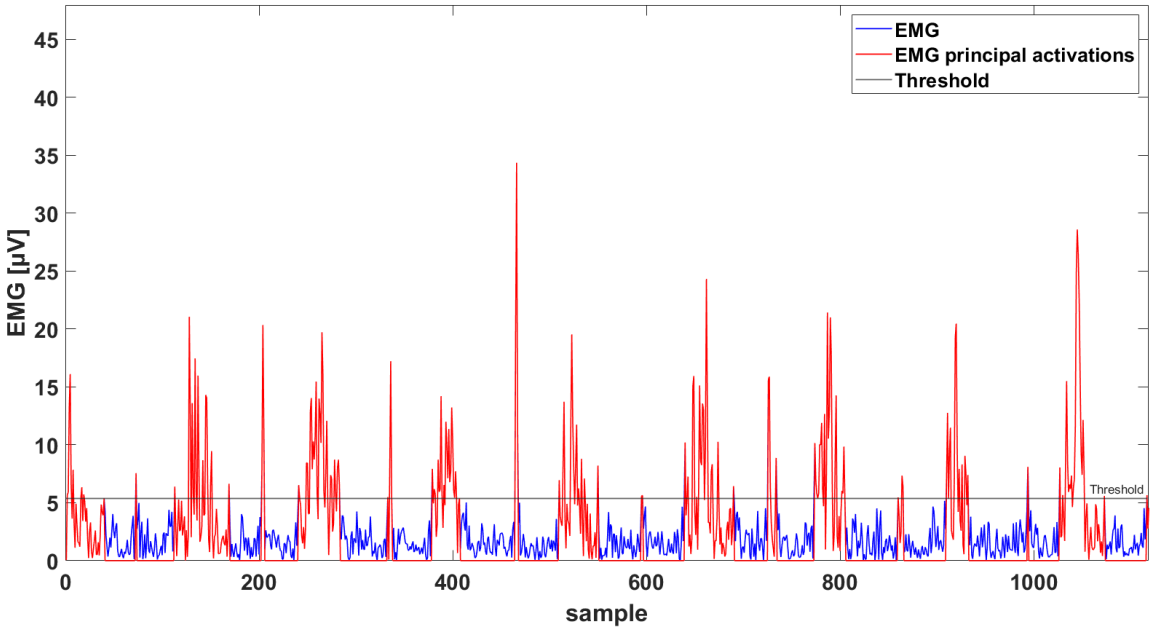


Figure 8.4: Principal activation and raw EMG of Incline Experiment dataset

8.1.3 Envelope computation

The electromyographic signal is by nature a noisy signal and can therefore appear almost accidental. As it is subsequently used for training neural networks, its random nature can make this task difficult. For this purpose, the envelope of the EMG signal is calculated, typically using filters, creating a smooth and less chaotic signal call envelope.

Envelope computation for Biolab dataset

In the Biolab dataset, the following cascade filters were applied to calculate the envelope [16].

- High-pass butterworth filter of the 8^{th} order with cut-off frequency of 35 Hz, to remove motion artifact.

$$|H_{hp}(j\omega)| = \frac{1}{\sqrt{1+(\omega_c/\omega)^{2n}}} = \frac{1}{\sqrt{1+(2\pi 35/\omega)^{16}}} = \begin{cases} 0 & \omega = 1 \\ 1/\sqrt{2} & \omega = 2\pi 35 \\ 1 & \omega = \infty \end{cases} \tag{8.4}$$

- Rectifier to obtain a non-negative signal (compute the absolute value of the signal).

$$|x| = \begin{cases} x & if\ x \geq 0 \\ -x & if\ x < 0 \end{cases} \tag{8.5}$$

- Low-pass butterworth filter of the 5^{th} order with cut-off frequency of 12 Hz, to remove high frequency disturbances.

$$|H_{lp}(j\omega)| = \frac{1}{\sqrt{1+(\omega/\omega_c)^{2n}}} = \frac{1}{\sqrt{1+(\omega/2\pi 12)^{10}}} = \begin{cases} 1 & \omega < 2\pi 12 \\ 0 & \omega > 2\pi 12 \end{cases} \tag{8.6}$$

Thanks to MATLAB, all the filters used were all rendered as zero phase filters, to avoid introducing delays into the signal.

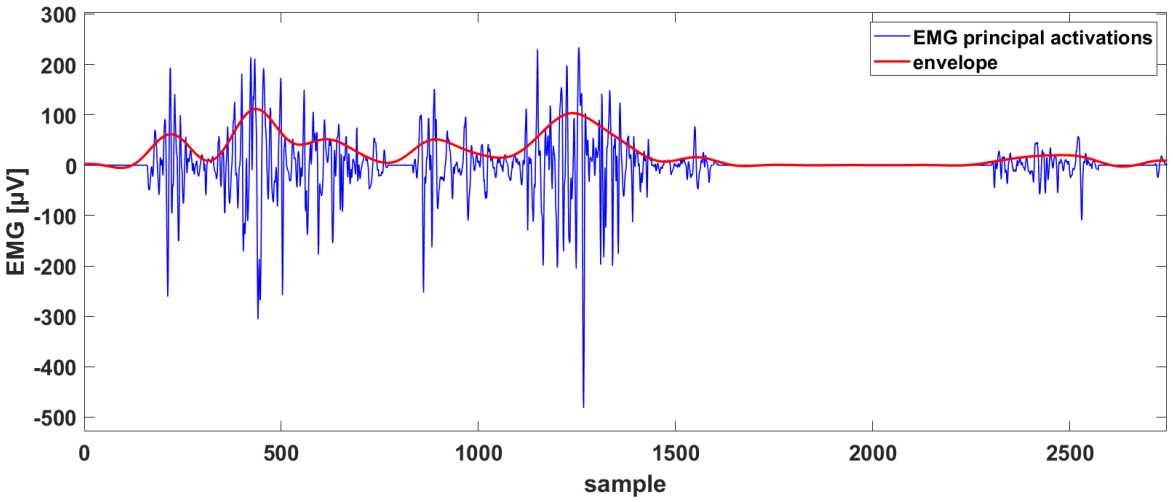


Figure 8.5: Envelope and principal activations of Biolab dataset

Envelope computation for Incline Experiment dataset

For the Incline Experiment dataset, as already mentioned, low-pass filtering and rectification had already been performed. However, despite the filtering, the signal still appeared noisy and random, so it was decided to treat it further with a smooth function with a Gaussian-weighted moving average filter.

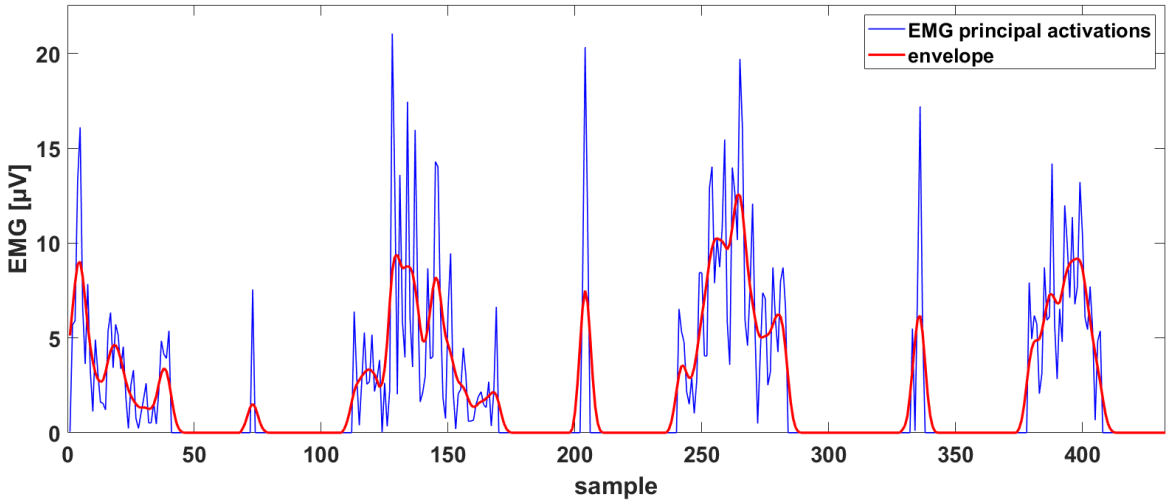


Figure 8.6: Envelope and principal activations of Incline Experiment dataset

8.2 Synergies

One of the main purposes of the central nervous system is to appropriately coordinate the activation of muscles that are used for a certain movement task. However, at present, how muscles are coordinated, in terms of time and intensity of activation, is not entirely clear despite many years of study [39].

The long-standing idea is that motor control may be simplified by a modular organization. Under this hypothesis, the control problem is reduced to modulating an appropriate selection of an adequate number of motor modules, also called muscle synergies, resulting in a simplified control of movement [10]. As far as neural networks are concerned, the idea is the same. The aim is to reduce complexity by reducing the number of inputs to the network and then use synergies instead of EMG signals to train neural networks.

Over the years, several techniques have been proposed for the extraction of muscle synergies. Such as artificial neural networks, auto-encoders, or through matrix factorization, Principal Component Analysis (PCA), Singular Value Decomposition (SVD), Independent Component Analysis (ICA) or Non-Negative Matrix Factorization (NNMF) [28]. In this study, the proposed extraction methodology is NNMF since it has demonstrated superior performance in the extraction of muscle synergies [37].

8.2.1 Non-negative matrix factorization

Basically, this algorithm models the original EMG signals (M) as the linear combination of the time-independent muscle synergy weights (W) and the time-dependent activation coefficients (H) as described in eq. 8.7 :

$$M(t) = \sum_{i=1}^{N} H(t)_i W_i + e \tag{8.7}$$

Here N represent the optimal number of muscle synergies needed to describe the motor task and e represents the prediction error of the factorization algorithm. In substance

what we get from the factorization are the two matrices $H(t)$ and W, whose dimensions depend precisely on N. To choose an acceptable factor of N, we use the formula of the coefficient of determination (R^2).

If N is not large enough, the synergies will not be able to carry the salient features of the source signals. On the other hand, if N is too small, the number of signals remains high, and overall there is no particular simplification of the signal system. $H(t)$ is the time-varying synergy excitation primitives matrix, the synergies in practice. W is the time-independent muscle weightings matrix, which contains quantitative information about the contribution of the various muscles in the synergy.

NNMF with MATLAB

Factorization was carried out in MATLAB using the Statistics and Machine Learning Toolbox [45]. After choosing the number N of synergies, the procedure involves calculating the W and $H(t)$ matrices by factoring with the sequence of iteration cycles, using two different algorithms. We start with ten cycles of ten iterations each, using the multiplicative algorithm. The purpose of this first phase of cycles is to identify the optimal solution generically. It is followed by a final cycle of one thousand iterations using the alternating least squares algorithm. This takes the best of the results calculated in the previous ten cycles and performs a final optimization of the factorization to return the best final result.

The matrix $H(t)$, the one we are interested in, is returned normalized with a unit norm, so the synergies need to be multiplied by the norm of the EMG signals before being used.

8.2.2 Coefficient of determination

It is the proportion of the variance in the dependent variable that is predictable from the independent variable. In other words, the coefficient of determination returns a number between zero and one. The closer this number is to one, the better the reconstructed variables approximate the original source variables. On the contrary, if it is close to zero, the reconstruction performed does not explain the original signal.

$$R^2 = 1 - \frac{\sum_{j=1}^{n}(M_j - M_j^R)^2}{\sum_{j=1}^{n}(M_j - \bar{M})^2} \tag{8.8}$$

In the formula, M_j^R is the reconstructed signal, and $\bar{M}$ is the mean of the original signal. Consequently, if the error e is small enough, the value of R^2 will be close to one and therefore, the choice of N will have been appropriate. Typically, the value of N is obtained by increasing it by one, until the self-determination coefficient reaches a value of at least 0.9. Notice that when the coefficient is equal to one, the number of synergies N is equal to the number of starting signals.

8.2.3 Synergies computation

It should be noted that in addition to taking into account when it reaches a value of at least 0.9 for the coefficient of determination, the needs of the neural networks must also be

taken into account when selecting N. According to the two datasets, the network starting from a certain number of EMG signals must be able to reproduce a certain number of joint signals. If, through the factorization, the number of inputs of the network becomes much lower than those of the outputs, the network could lose efficiency. It is sufficient to think of a limit case in which the network, starting from two inputs, must be able to return ten outputs. In this case, the system could be too simplified with the resulting loss of meaning.

Synergies computation for Biolab dataset

The neural network has to use ten EMG signals, five per leg (TA, LGS, RF, LH, GMD for both legs), to reconstruct four angular signals, two per leg (hip flexion-extension and knee flexion-extension for both legs).

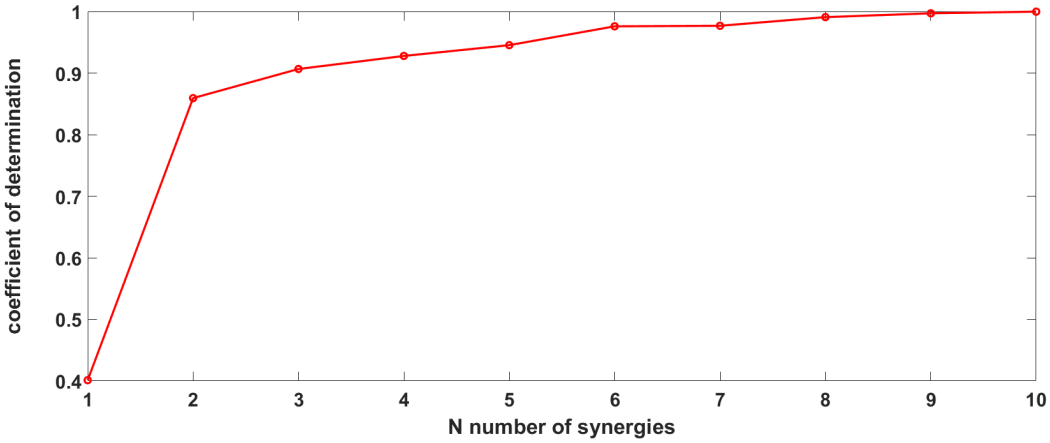


Figure 8.7: Trend of the coefficient of determination for Biolab dataset

As shown in Fig. 8.7, N equal to three is already enough to reconstruct the signals efficiently, but to give the neural network a sufficient number of input it is better to choose N equal to six. So that with six input the network have to reconstruct four output.

Synergies computation for Incline Experiment dataset

The neural network has to use eight EMG signals, four per leg (RF, BF, TA, GC for both legs), to reconstruct ten angular signals, five per leg (hip flexion-extension, hip adduction-abduction, knee flexion-extension, ankle flexion-extension, ankle pronation-supination for both legs).

Has it can be seen from Fig. 8.8, N equal to three is already enough to reconstruct the signals efficiently, but to give to the neural network a sufficient number of input it is better to choose N equal to six. So that with six input the network have to reconstruct four output. Fig. 8.9a and Fig. 8.9b show block diagrams summarizing the process from raw EMG signals to muscle synergies for both datasets.

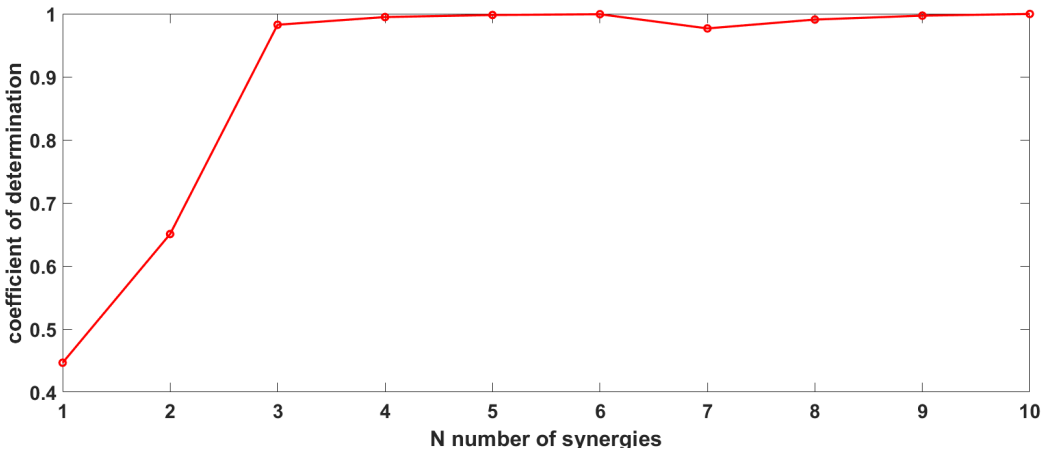
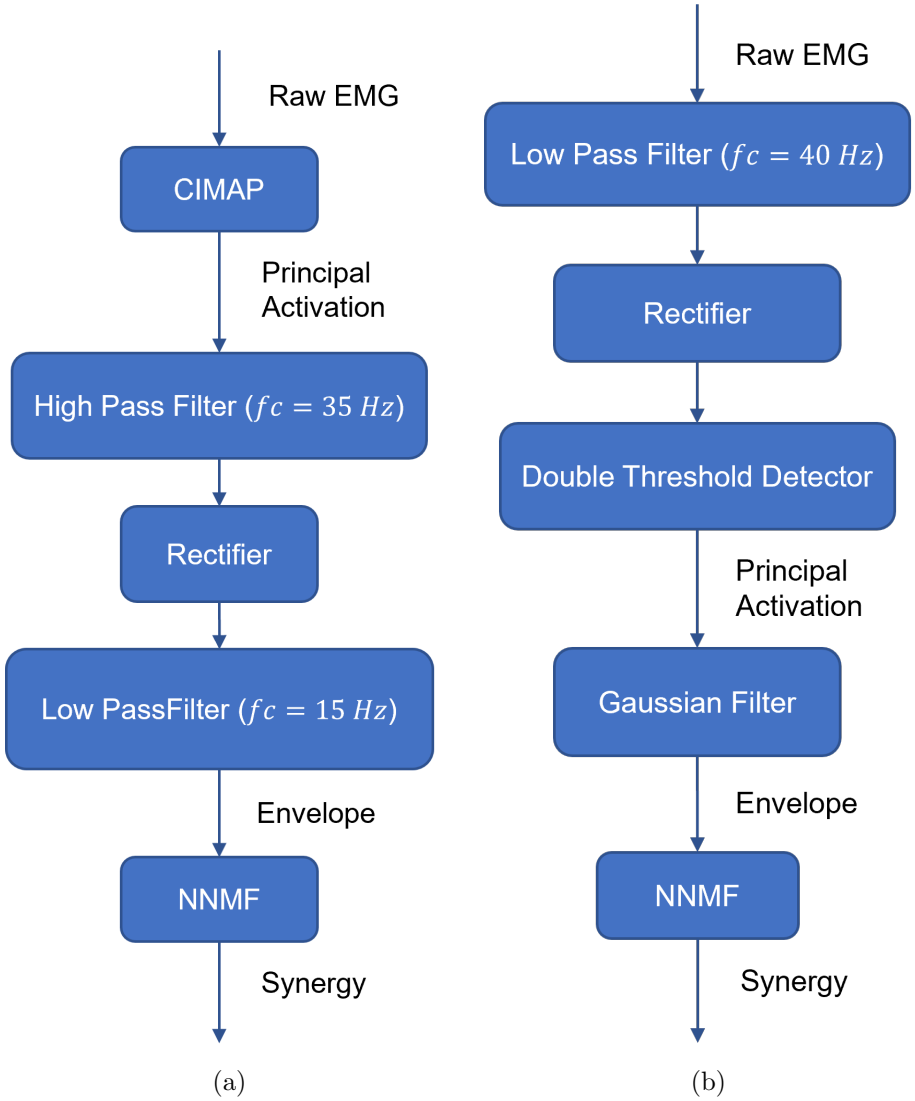


Figure 8.8: Trend of the coefficient of determination for Incline Experiment dataset



(a) (b)

Figure 8.9: (a) Block diagram to obtain synergies from the Biolab dataset (b) Block diagram to obtain synergies from the Incline Experiment dataset

8.3 Recurrent neural network

8.3.1 RNN introduction

In this type of network, connections between nodes form a directed graph along a temporal sequence. This characteristic makes them suitable for working with time series and in particular with future predictions of time sequences. The general structure of these networks is shown in Fig. 8.10.

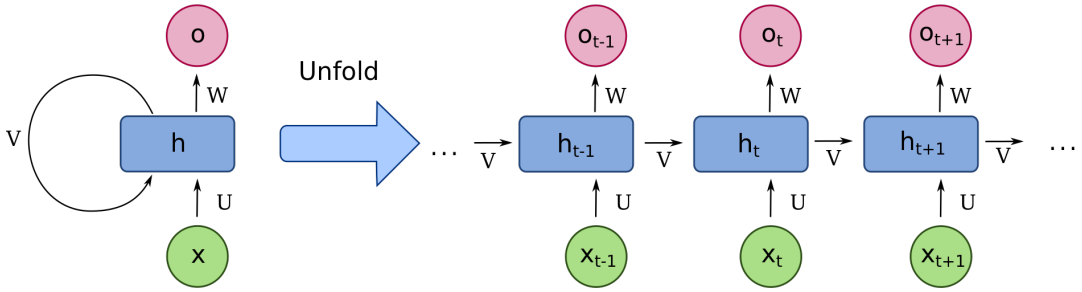


Figure 8.10: General structure of Recurrent neural networks

In a Recurrent neural network, the cells represent the fundamental brick. Each cell is equal to itself and is recurring (hence the name of these networks), and takes care of the mathematical calculations to be made to obtain an output. The input data, at a certain time, x_t are associated with input weighted matrix U. The output o_t is associated with matrix W and the hidden state h_t with matrix V. The hidden state vector keeps memory of the cell at a certain time and so, it is used by the next cell for the calculation of o_t, taking into account the state of the previous cell.

$$h_t = f(Ux_t + Wh_{t-1}) \tag{8.9}$$

$$o_t = f(Vh_t) \tag{8.10}$$

In eq. 8.9 the calculation of the hidden state to be used as information in the next cell and for calculating of the output is explained. The hidden state and the input are multiply by their respective weight matrices and summed up together. Then the sum is processed by the activation function. Typically the hyperbolic tangent is chosen as the activation since it shapes its output between one and minus one. Thus, distributing the gradient and trying to tackle its vanish or its explosion.

While in eq. 8.10, it is explained that the output is nothing more than the hidden state multiply by its weighted matrix and processed by the output activation function, that depend on the network application [35].

Known issues of basic RNN

The performance of these networks concerning future predictions of time series is affected by the following problems:

- No long term memory.
- Can't use information from the distant past.
- Can't learn patterns with long dependencies.

8.3.2 NARX feedback neural network

NARX means that the model is non-linear and auto-regressive, [3] which has exogenous[4] inputs. In other words, this network correlates the prediction made at a certain time with the predictions made at previous times and with the values of the time series, whose behavior we want to predict, at previous times.

From a mathematical point of view, this relationship is described in eq. 8.11.

$$y(t) = F(y(t-1), y(t-2), \ldots, u(t-n), u(t-1), u(t-2), \ldots, u(t-n)) + e \qquad (8.11)$$

F is the function representing the neural network, y is the predicted time series, u is the series for which we want to create a relationship with the predictions and e represents the error between predictions and known data. Network training involves two successive steps, one feed-forward and one with feedback.

Open loop training

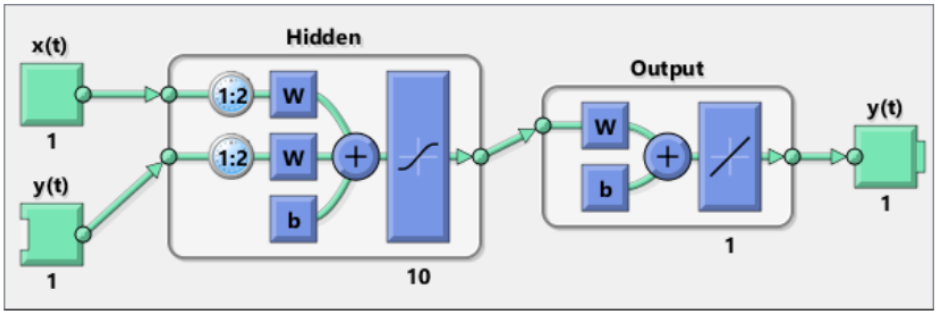
At this stage, there is no feedback in the network, and predictions are calculated in feed-forward using the true data of the time series to be predicted as input. Thanks to this first step, the weights and biases of the network are initialized. Since at the moment the predictions are made using as input the real-time series to be predicted, these parameters have already been directed towards the optimal solution. The open-loop it is also named series-parallel architecture and it is represented in Fig.8.11a.

Closed loop training

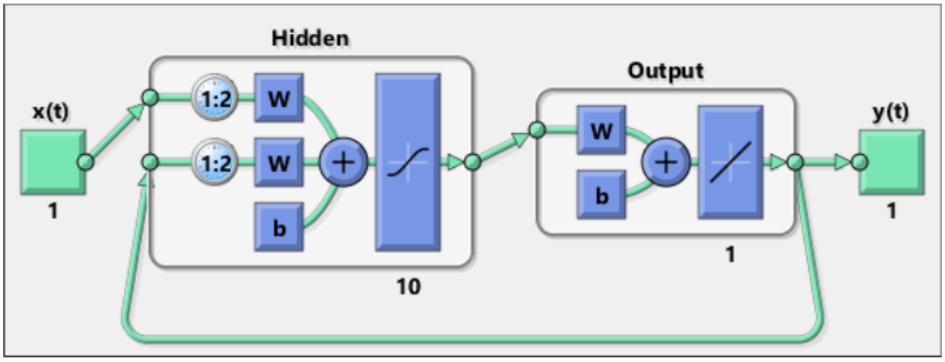
Here the loop on the network is closed and predictions are made using as input the same predictions made in previous time instants. The closed-loop is also named parallel architecture, and it is represented in Fig.8.11b.

[3]Input-output correlation is not proportional, and the model represents a random process, which provides an instant output value based on the previous input and output values.

[4]The input changes due to events outside the model that it cannot explain



(a)



(b)

Figure 8.11: (a) NARX network in open-loop (b) NARX network in closed-loop

MATLAB procedure for NARX

The Deep Learning Toolbox [42] was used to train NARX networks on MATLAB. The procedure followed is summarized in the following points:

- Partition of the data: for the network, it was decided to use three-quarters of the data for training and one quarter to test the trained network. Training data was divided into:
  - Training data: these are presented to the network during training, and the network is adjusted according to its error. Seventy per cent of the training data was set for all networks.
  - Validation data: these are used to measure network generalization, and to halt training when generalization, stops improving. Fifteen per cent of the training data was set for all networks.
  - Testing data: these have no effect on training and so provide an independent measure of network performance during and after training. Fifteen per cent of the training data was set for all networks.
- Set the layers and the delay: decide on the number of hidden layer and number of previous time event to use for the predictions (number of inputs and outputs are automatically set by the network as it evaluates input data size). A number of 6 hidden layers and 5 previous time events was set for all networks.

- Set the options: Decide on the parameters to be used in the network.
  - Activation function for hidden layers: a linear saturation function between one and minus one was decided for all the networks.
  - Activation function for output layers: a linear function was decided for all the networks.
- Training: training involves the following steps.
  - Open-loop training.
  - Close the loop on the network.
  - Closed-loop training.
- Predictions: compute the values predicted by the network using test data.
- Coefficient of determination: use the coefficient of determination to evaluate the performance of the network by observing how close the predicted data are to the test data.

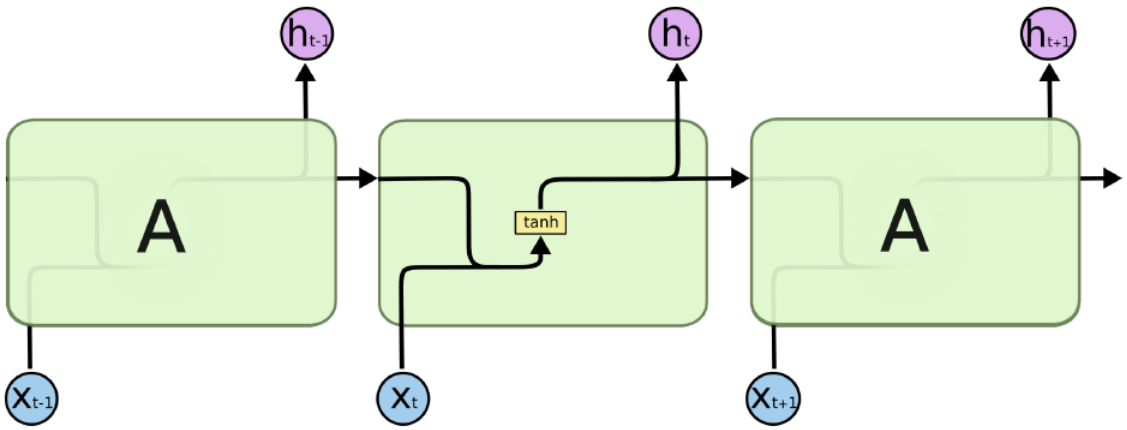
$$\overline{DataTest} = \frac{\sum_{i=1}^{N}(DataTest_i)}{N} \tag{8.12}$$

$$R^2\_narx = 1 - \frac{\sum_{j=1}^{n}(DataTest_j - DataPred_j)^2}{\sum_{j=1}^{n}(DataTest_j - \overline{DataTest})^2} \tag{8.13}$$

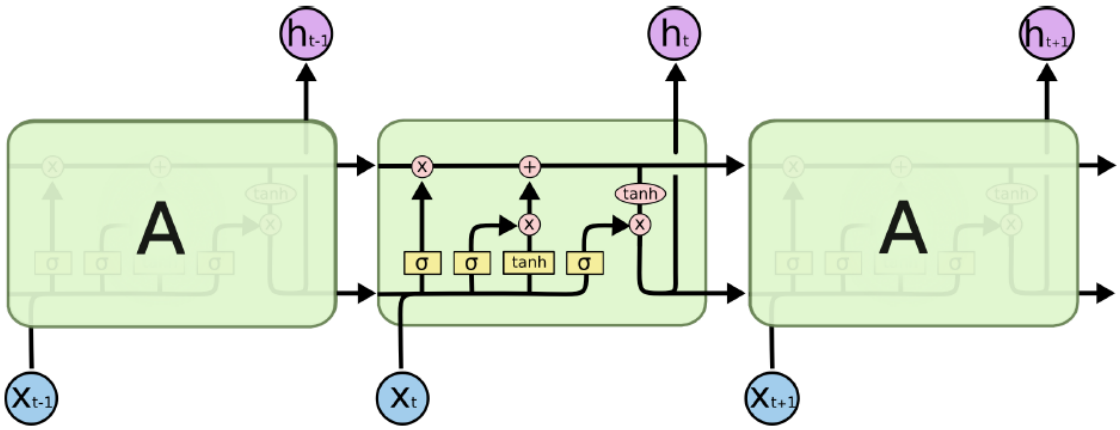
- Repeat the training: The closed-loop training is repeated ten times and ten neural networks with different weights and biases are calculated. After calculating the respective coefficients of determination, the best of the networks is identified and reused for another cycle of ten pieces of training. With the difference that between one cycle, the intensity at which the weights and biases are disturbed is chosen. As one approaches a model that is able to predict the behavior of the signals well enough, the intensity with which to perturb the network parameters has to be chosen smaller and smaller. In this way, we try to locate a minimum point at which the network parameters are able to return faithful temporal predictions. The procedure is repeated until a good coefficient of determination is reached, or until a local minimum is presumed to have been reached, and no further steps can be taken.

8.3.3 LSTM neural network

These types of recurrent networks are designed to counter the vanishing/exploding gradient problems through the use of memory cells. So, to build an LSTM network, it is sufficient to replace the normal cells of an RNN with LSTM cells like in Fig 8.12b. In addition to having the regular hidden state vector $h_{(t)}$, LSTM cells pass data through another signal $C_{(t)}$ called the cell state. This additional vector is responsible for storing long-term information at a given time instant. The new cell structure makes use of three gates, which act as filters, to regulate the passage of data [35]. The difference between a normal RNN cell and an LSTM cell are visible when comparing Fig 8.12a and Fig 8.12b.



(a)



(b)

Figure 8.12: (a) General RNN's cell structure (b) LSTM's cell structure

Fig 8.13 summarises and explains the flow of data through the cell, with mathematical operations such as sum, element-wise product and *tanh*[5] highlighted in red. The yellow blocks represent the dense layers with the corresponding activation functions. A dense layer is a set of neurons with the corresponding weights and biases. Finally, it should be mentioned that when two black lines converge into one, the result is the concatenation[6] of the two signals present in the two lines.

With these arrangements, it is now possible to understand the operating principle based on the three gates. The *input gate*, the *forget gate* and the *output gate*.

Input Gate

Control whenever the memory cell is updated by multiplying $i_{(t)}$ with $\bar{C}_{(t)}$. Variable $\bar{C}_{(t)}$ is responsible for changes in the state of the cell, as it is the only element that is added to $C_{(t)}$. The variable $i_{(t)}$ is a matrix resulting from the concatenation of $x_{(t)}$ (the input data) and $h_{(t-1)}$ (the hidden state of the previous time instant, computed by the previous

[5] $\frac{e^x - e^{-x}}{e^x + e^{-x}}$, the function of hyperbolic tangent that returns an output value between plus and minus one

[6] In mathematics, to concatenate means to merge two or more numbers to form a new one.

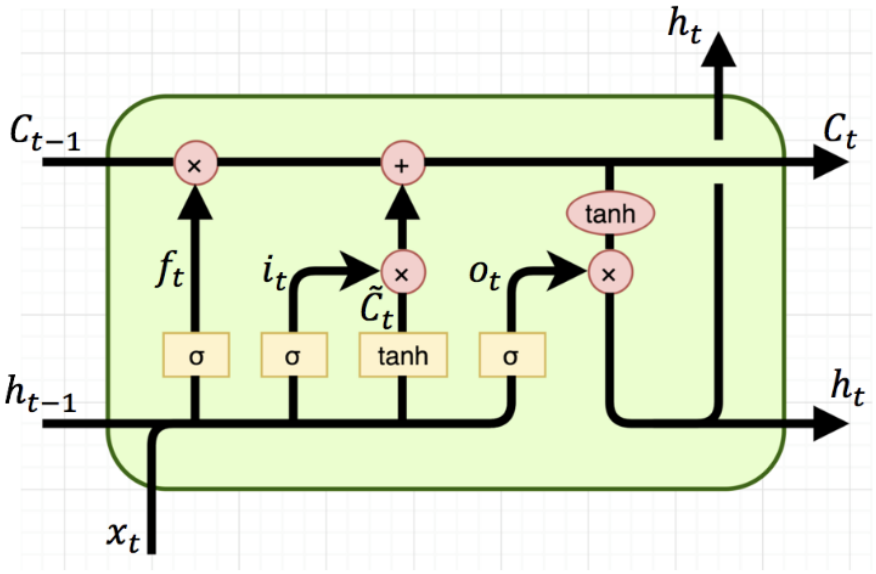


Figure 8.13: Signals inside an LSTM cell

cell) multiplied by the respective matrix of weights W_i and added to the respective biases b_i. Then, due to the application of a sigmoid function, matrix $i_{(t)}$ contains numbers between zero and one. Consequently, it decides which elements of $\bar{C}_{(t)}$ remain unchanged and which are attenuated or even reduced to zero.

$$i_{(t)} = \sigma(W_i[h_{(t-1)}, x_{(t)}] + b_i) \tag{8.14}$$

Forget Gate

Control when the memory cell is reset to zero. The variable $f_{(t)}$ is a matrix resulting from the concatenation of the input and the previous hidden state multiplied by the respective matrix of weights W_f and added to the respective biases b_f. Then, the sigmoid function shrinks matrix $f_{(t)}$ between zero and one. This matrix multiplies the state of the previous cell. As a result, elements of $C_{(t-1)}$ that are multiplied by numbers close to one are kept, while elements multiplied by values close to zero are forgotten.

$$f_{(t)} = \sigma(W_f[h_{(t-1)}, x_{(t)}] + b_f) \tag{8.15}$$

Output Gate

Control if the memory cell information is made visible or not. This matrix is also derived from the concatenation of the input and the previous hidden state multiplied by the respective matrix of weights W_o and added to the respective biases b_o. Then, the matrix $o_{(t)}$ is used to make a selection of the elements of the current cell state $C_{(t)}$. The selection is again made using a sigmoid as an activation function for the output gate, which makes the elements of the matrix numbers between zero and one. Subsequently, from this operation, the new hidden state is calculated (eq. 8.18).

$$o_{(t)} = \sigma(W_o[h_{(t-1)}, x_{(t)}] + b_o) \tag{8.16}$$

State Equations

The coordination of the three gates then results in the cell state and the hidden state to be used in the next cell for the next time instant.

$$C_{(t)} = f_{(t)}C_{(t-1)} + i_{(t)}\bar{C}_{(t)} \tag{8.17}$$

it is a matrix the cell state is influenced by the forget and the input gate. As already mentioned, the input gate controls the memory update, so it is multiplied by $\bar{C}$. The forget gate control how much of the old cell state is preserved, so it is multiplied by $C_{(t-1)}$.

$$h_{(t)} = \tanh(C_{(t)}o_{(t)}) \tag{8.18}$$

According to the cell state, the hidden vector is obtained by the multiplication of the output gate and the cell state, passing through the activation of *tanh*.

$$\bar{C}_{(t)} = \tanh(W_C[h_{(t-1)}, x_{(t)}] + b_C) \tag{8.19}$$

Only $\bar{C}_{(t)}$ can update the cell state, represent a new candidate value to apply to the cell state. $\bar{C}_{(t)}$ is the concatenation of the input and the previous hidden state multiplied by the respective matrix of weights W_C and added to the respective biases b_C. In this case, the *tanh* function is chosen as activation. The *tanh* distribute the gradient preventing it from vanishing or exploding.

MATLAB procedure for LSTM

The Deep Learning Toolbox [42] was used to train LSTM networks on MATLAB. The procedure followed is summarized in the following points:

- Partition of the data: for the network, it was decided to use three-quarters of the data for training and one quarter to test the trained network.
- standardize data: for a better fit and to prevent the training from diverging, train and test data have been standardized to have zero mean and unit variance. To do this, we use the mean and standard deviation.

$$StanData = \frac{(Data - \overline{Data})}{Data\_\sigma} \tag{8.20}$$

$$\overline{Data} = \frac{\sum_{i=1}^{N}(Data_i)}{N} \tag{8.21}$$

$$Data\_\sigma = \sqrt{\frac{\sum_{i=1}^{N}(Data_i - \overline{Data})^2}{N}} \tag{8.22}$$

- Set the layers: decide on the number of inputs and outputs (they change depending on the dataset and depending on the methodology used on the EMG signal), and the number of hidden units. A number of 400 hidden units was set for all networks.
- Set the options: Decide on the parameters to be used in the network.
  - Maximum number of epochs: number of the epochs for the training, 400 for all the networks.
  - Initial learning rate: if the learning rate is too low, then training takes a long time. If the learning rate is too high, then training might reach a sub-optimal result or diverge. A learning rate of 0.01 was decided for all the networks.
  - Learning rate drop period: deciding after how many epochs the learning rate changes. A learning rate drop period of 100 epochs was decided for all the networks.
  - Learning rate drop factor: deciding how much the learning rate is halved after the learning rate drop period. A learning rate drop factor of 0.2 epoch was decided for all the networks.
- Training: train the network with the established parameters and the training data.
- Predictions: compute the values predicted by the network using test data and simultaneously update the net state.
- De-standardize the predictions: de-standardize predictions to compare them with test data.

$$DeStanPred = Pred\_\sigma \cdot Pred + \overline{Pred} \tag{8.23}$$

$$\overline{Pred} = \frac{\sum_{i=1}^{N}(Pred_i)}{N} \tag{8.24}$$

$$Pred\_\sigma = \sqrt{\frac{\sum_{i=1}^{N}(Pred_i - \overline{Pred})^2}{N}} \tag{8.25}$$

- Coefficient of determination: use the coefficient of determination to evaluate the performance of the network by observing how close the predicted data are to the test data.

$$R^2\_lstm = 1 - \frac{\sum_{j=1}^{n}(DataTest_j - DeStanPred_j)^2}{\sum_{j=1}^{n}(DataTest_j - \overline{DataTest})^2} \tag{8.26}$$

8.4 Performances

The training of the networks was carried out, taking into account raw EMGs, envelopes and synergies, both for NARX and LSTM networks, on both datasets. As already mentioned, the coefficient of determination was used to evaluate the results obtained.

The coefficient was assessed on signal sections that the neural networks had not processed during training and not on sections that the networks had already analyzed. Consequently, the training of the networks was targeted on the part of the signal to be tested

and not on the signal used for training. Table 8.1 shows the results of the coefficients of determination calculated on the data that were also used for training. On the other hand, in table 8.2, the coefficients shown were evaluated on test data, which the network had never seen before.

Dataset	Type of input	Type of RNN	
		NARX	LSTM
Biolab	EMG	0.520	0.992
	Envelope	0.815	0.999
	Synergy	0.841	0.999
Inc. Exp.	EMG	0.651	0.996
	Envelope	0.782	0.994
	Synergy	0.540	0.998

Table 8.1: Coefficients of determination with training data

Dataset	Type of input	Type of RNN	
		NARX	LSTM
Biolab	EMG	0.470	0.985
	Envelope	0.754	0.975
	Synergy	0.756	0.983
Inc. Exp.	EMG	0.530	0.985
	Envelope	0.705	0.983
	Synergy	0.429	0.968

Table 8.2: Coefficients of determination with testing data

When exiting the neural network, the angular signals are noisy, so to cancel out the high-frequency components, the signals are passed through a low-pass filter.

8.4.1 Angular positions for Biolab dataset

Control_1 was chosen to train the neural networks among the subjects of the dataset. The recorded gait was not straight but had curved lines. Therefore, for the purpose of the simulations, a portion of the signal without curved stretches were selected and used to conduct the experiments. This portion comprised 11394 samples at a frequency of 2000 Hz. Of these samples, 8545 were used as training data and the remaining 2849 as testing data. In Fig. 8.14, Fig. 8.15, Fig. 8.17 and Fig. 8.18 angular signals are shown for right knee flexion-extension, left knee flexion-extension, right hip flexion-extension and left hip flexion-extension, from the top to the bottom and for all figures.

Angular positions from NARX

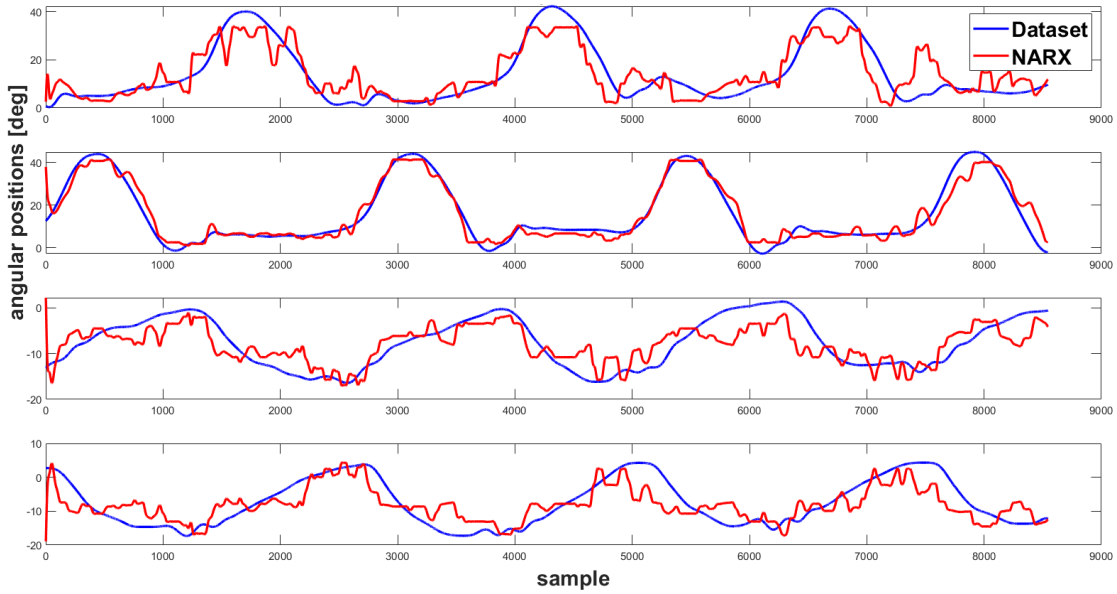


Figure 8.14: Angular positions and reconstructed angular positions with training data using NARX for the Biolab dataset

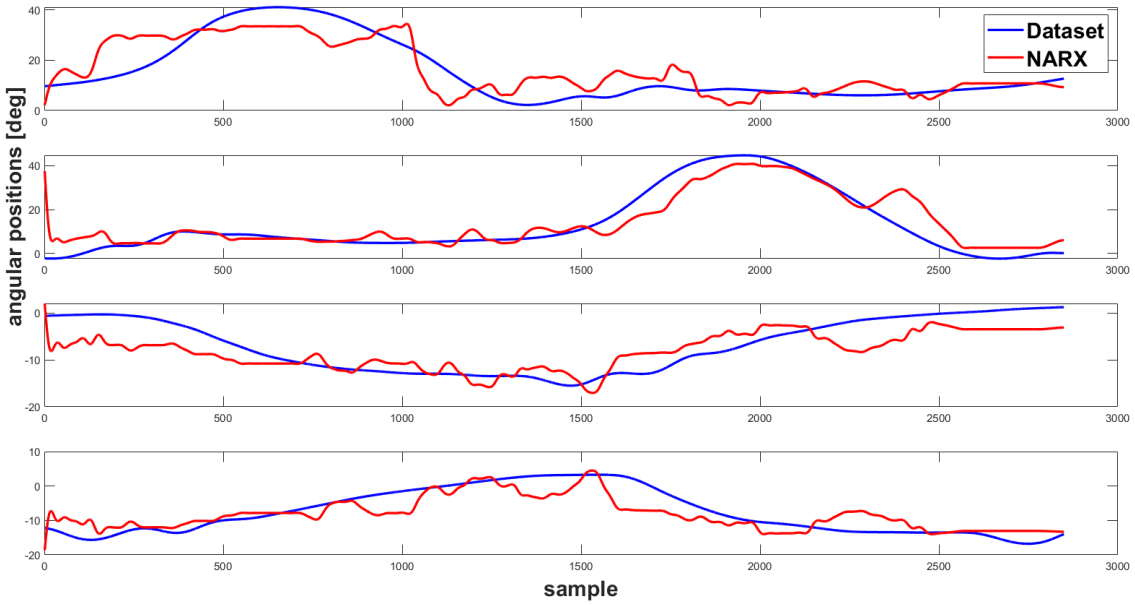


Figure 8.15: Angular positions and reconstructed angular positions with testing data using NARX for the Biolab dataset

The angular positions of the Biolab dataset passed through NARX network, were filtered with 5^{th} low-pass batterworth with a cut-off frequency of 3 Hz.

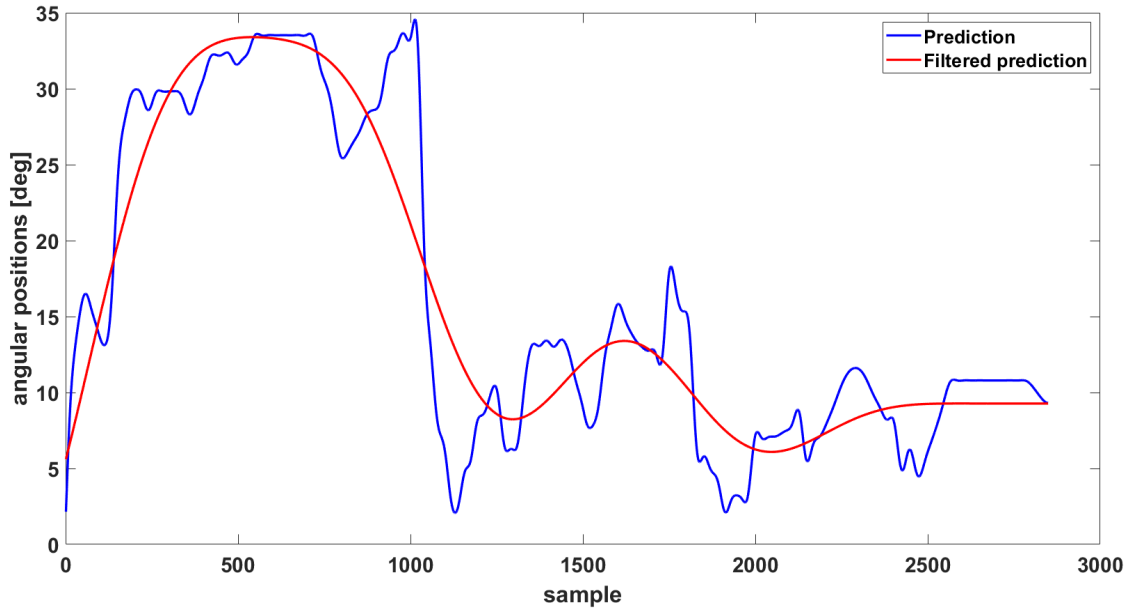


Figure 8.16: Reconstructed angular positions before and after low-pass filtering using NARX for the Biolab dataset

Angular positions from LSTM

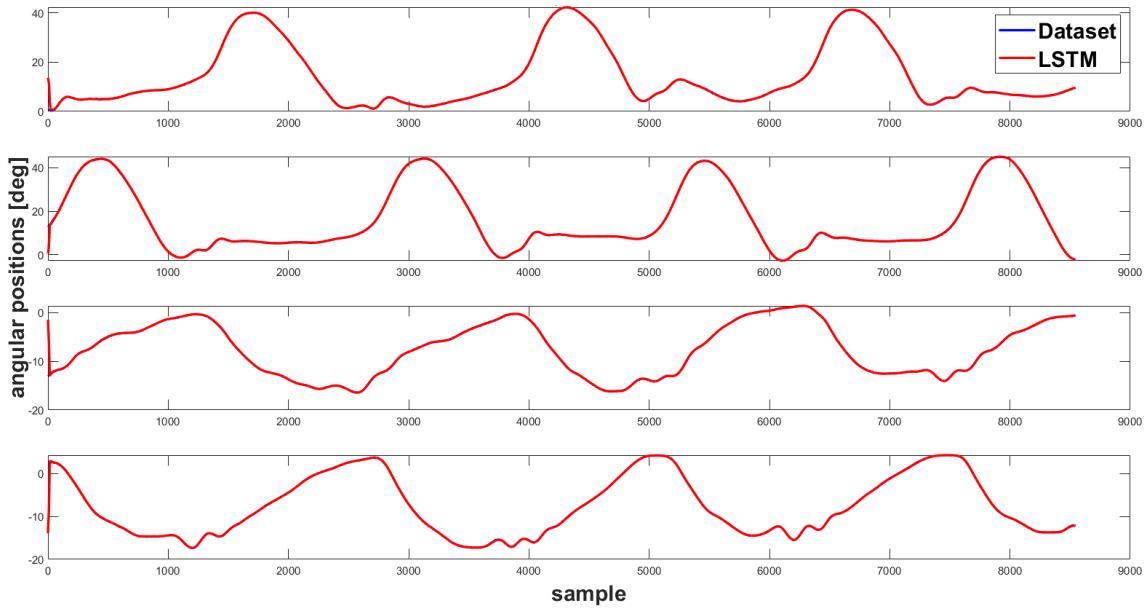


Figure 8.17: Angular positions and reconstructed angular positions with training data using LSTM for the Biolab dataset

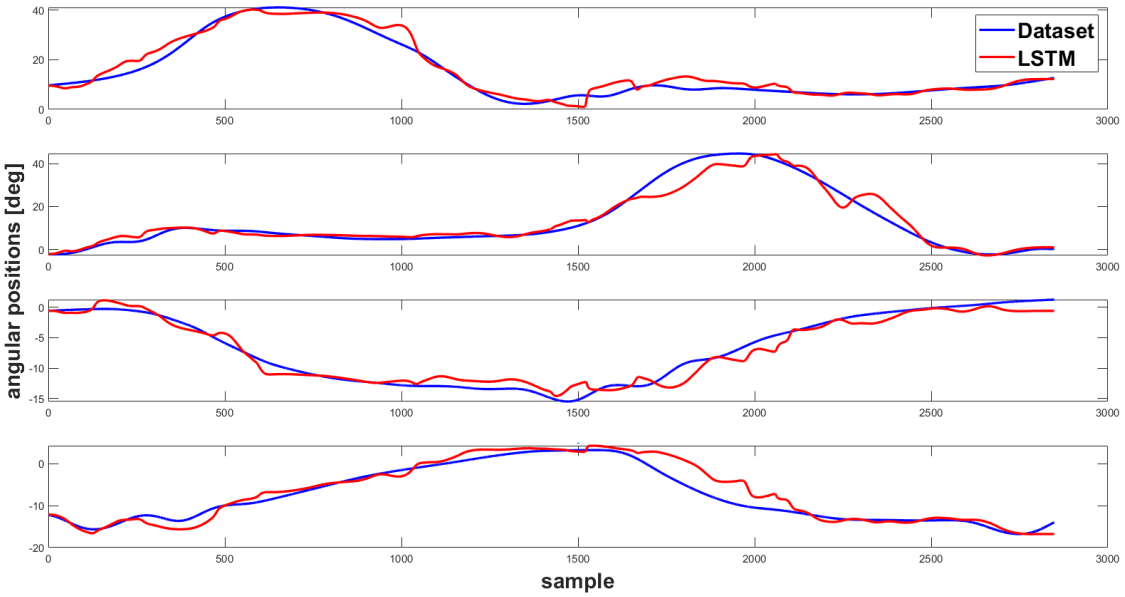


Figure 8.18: Angular positions and reconstructed angular positions with testing data using LSTM for the Biolab dataset

The angular positions of the Biolab dataset passed through LSTM network, were filtered with 5^{th} low-pass batterworth with a cut-off frequency of 5 Hz.

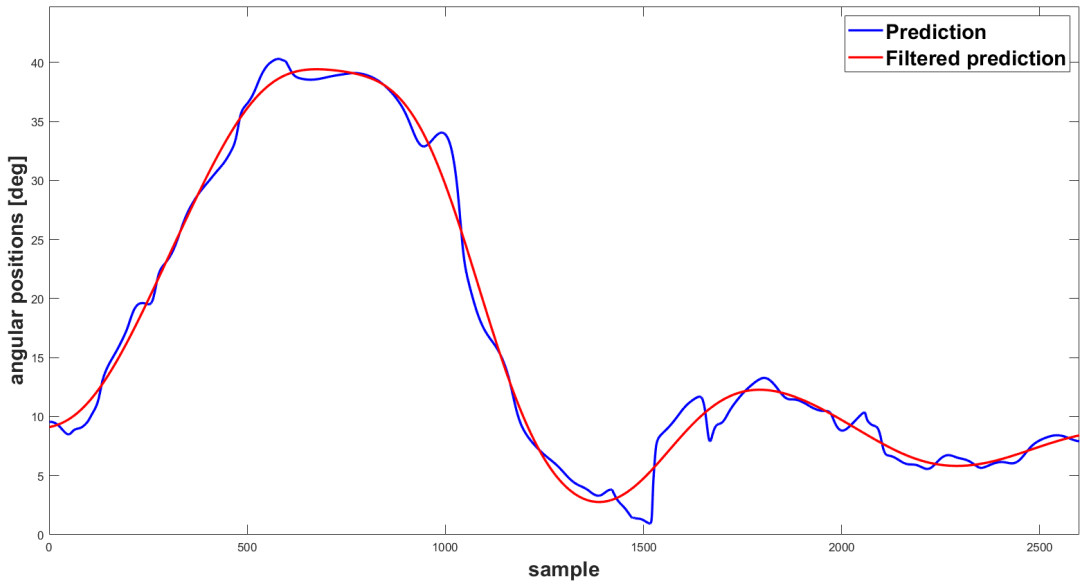


Figure 8.19: Reconstructed angular positions before and after low-pass filtering using LSTM for the Biolab dataset

8.4.2 Angular positions for Incline Experiment dataset

For this dataset, data from patient *AB01*, at a speed of 1 m/s and with the inclination equal to that of the ground were chosen. In this case, all walking was straight, and it was not necessary to cut the recordings. Of 6000 samples, 4500 were for training and the remaining 1500 for testing, with a sampling rate of 100 Hz. From Fig. 8.20, to Fig. 8.23 and from Fig. 8.25, to Fig. 8.28 angular signals are shown for hip adduction-abduction, hip flexion-extension, knee flexion-extension, ankle flexion-extension and ankle pronation-supination, from the top to the bottom and for all figures.

Angular positions from NARX

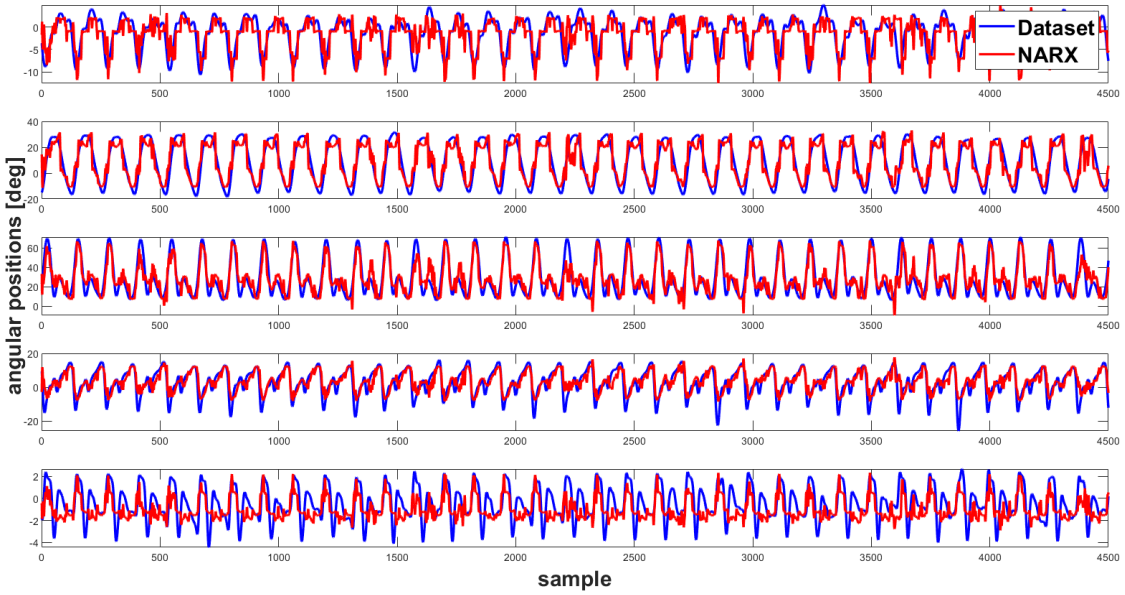


Figure 8.20: Angular positions and reconstructed angular positions of the left leg with training data using NARX for the Incline Experiment dataset

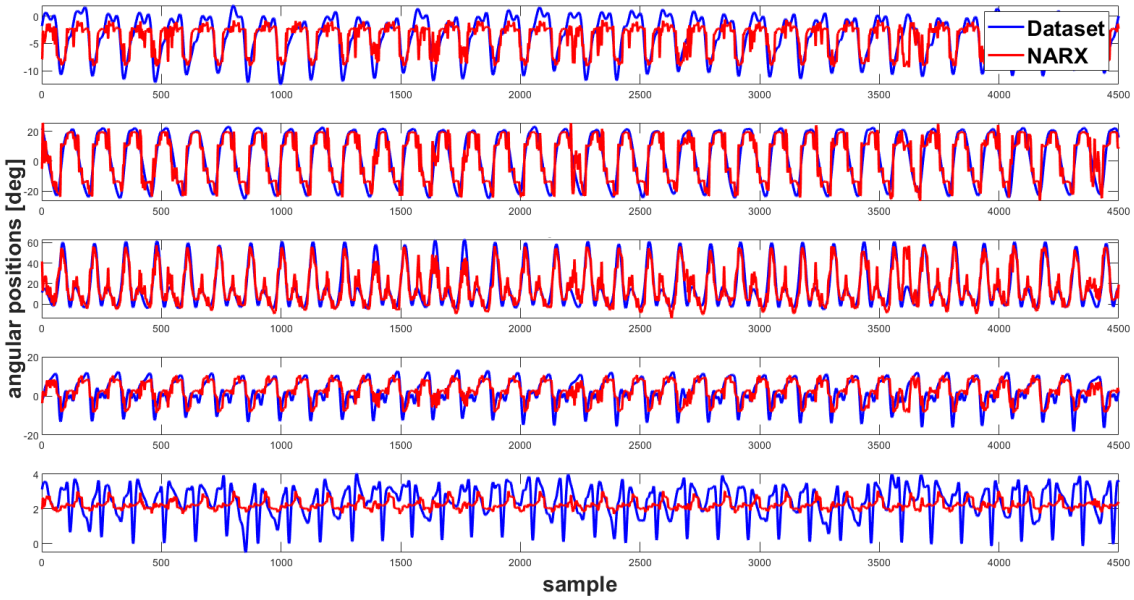


Figure 8.21: Angular positions and reconstructed angular positions of the right leg with training data using NARX for the Incline Experiment dataset

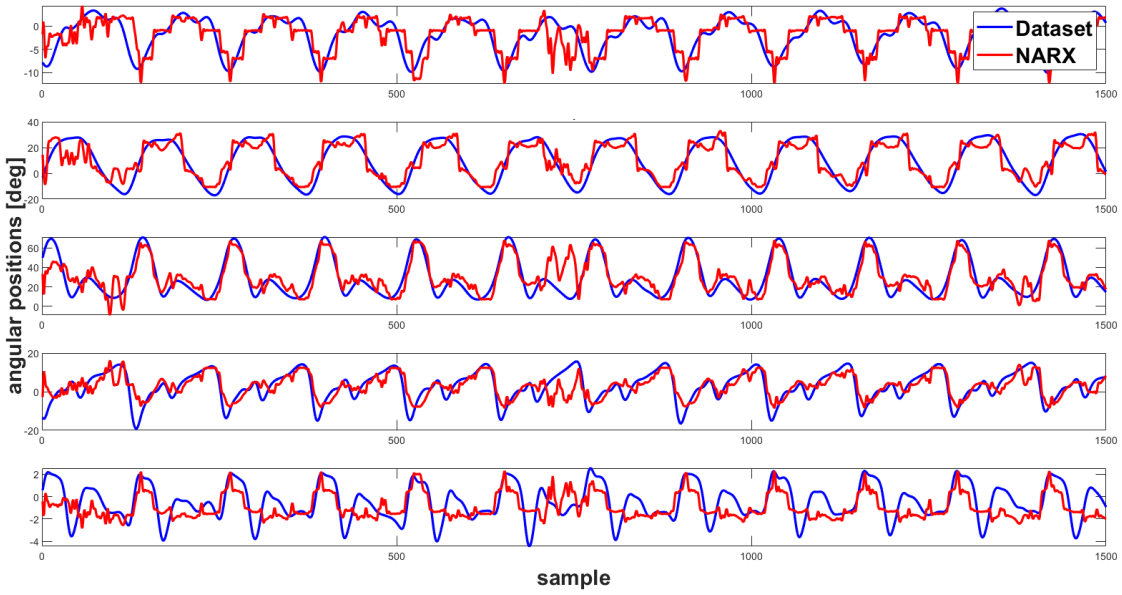


Figure 8.22: Angular positions and reconstructed angular positions of the left leg with testing data using NARX for the Incline Experiment dataset

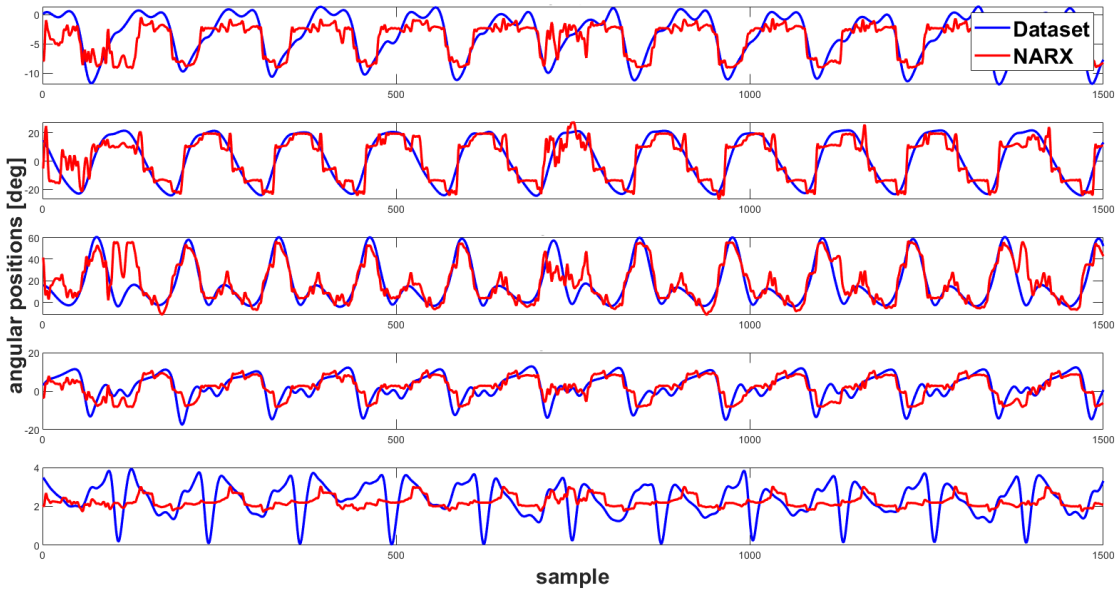


Figure 8.23: Angular positions and reconstructed angular positions of the right leg with testing data using NARX for the Incline Experiment dataset

The angular positions of the Incline Experiment dataset passed through NARX network, were filtered with 5^{th} low-pass batterworth with a cut-off frequency of 100 Hz.

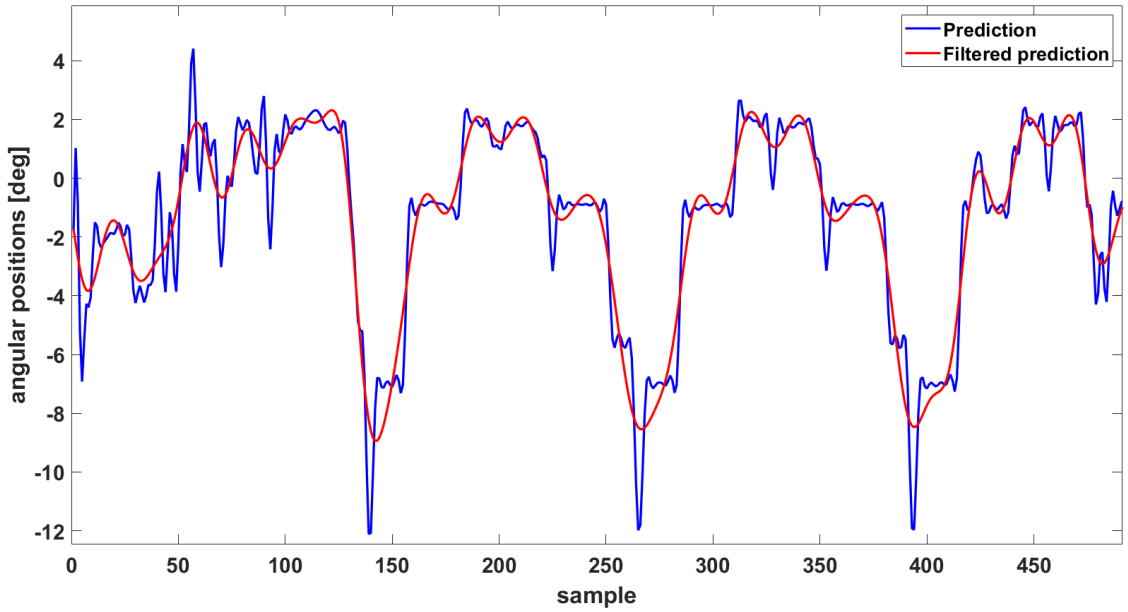


Figure 8.24: Reconstructed angular positions before and after low-pass filtering using NARX for the Incline Experiment dataset

Angular positions from LSTM

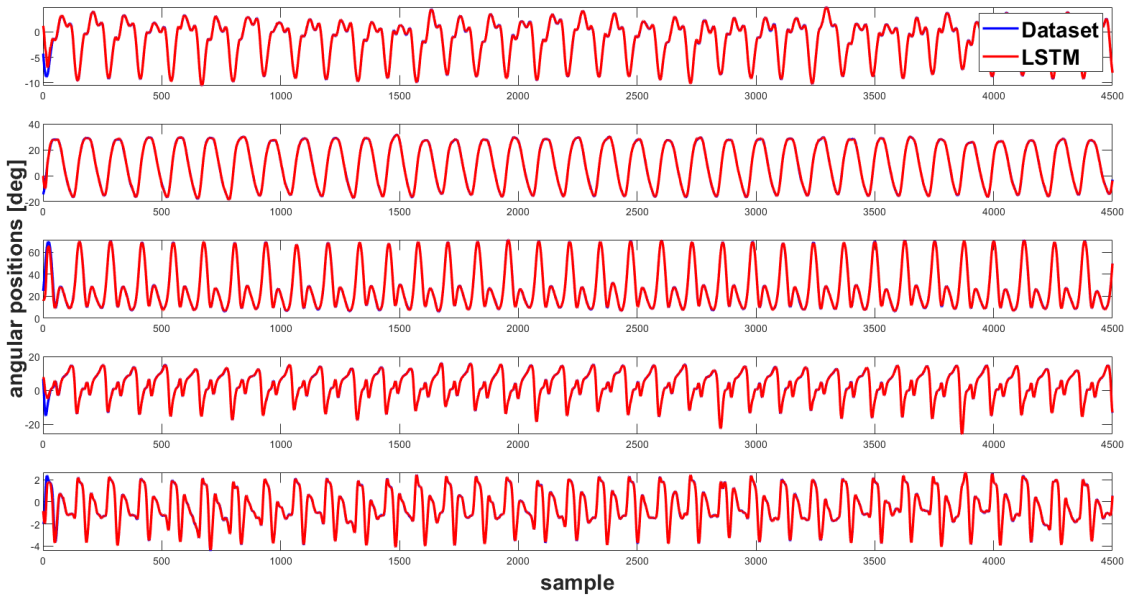


Figure 8.25: Angular positions and reconstructed angular positions of the left leg with training data using LSTM for the Incline Experiment dataset

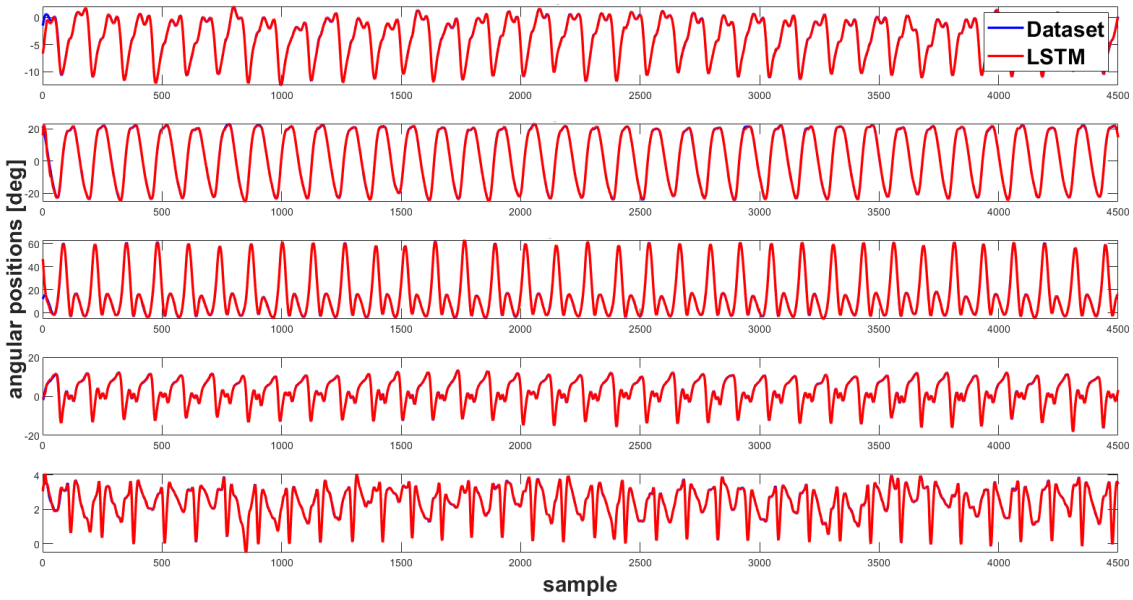


Figure 8.26: Angular positions and reconstructed angular positions of the right leg with training data using LSTM for the Incline Experiment dataset

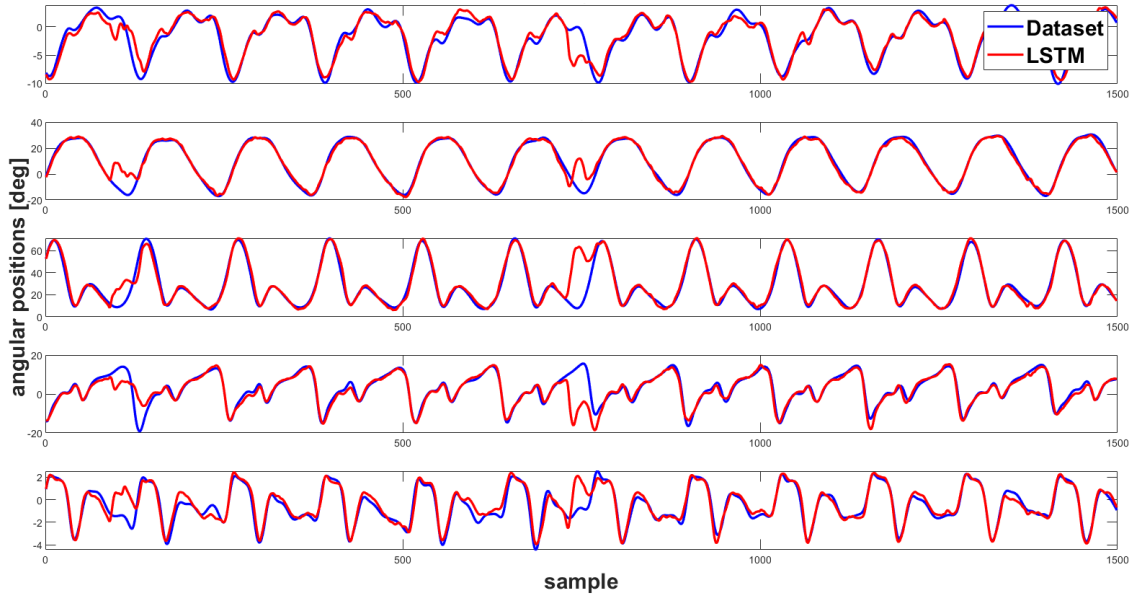


Figure 8.27: Angular positions and reconstructed angular positions of the left leg with testing data using LSTM for the Incline Experiment dataset

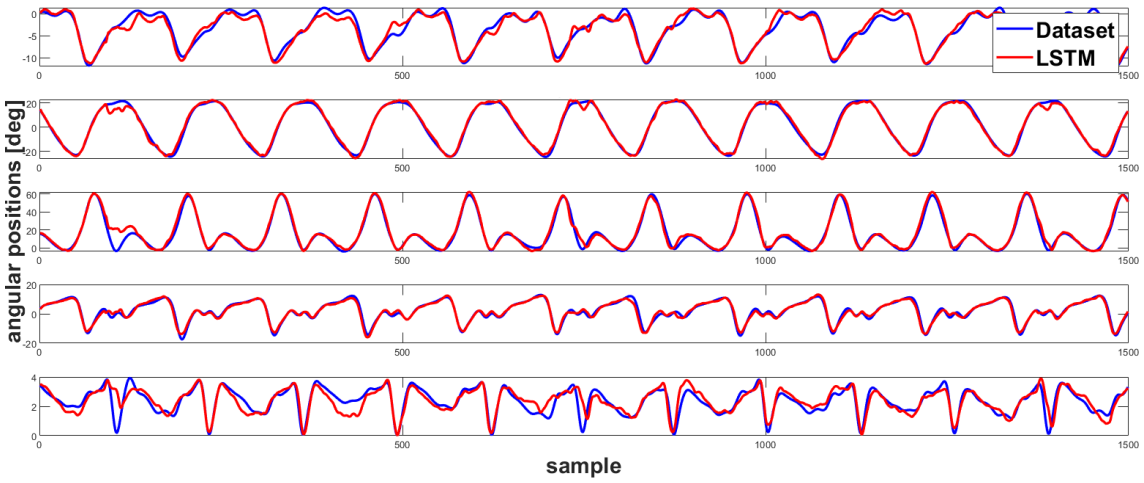


Figure 8.28: Angular positions and reconstructed angular positions of the right leg with testing data using LSTM for the Incline Experiment dataset

The angular positions of the Incline Experiment dataset passed through LSTM network, were filtered with 5^{th} low-pass batterworth with a cut-off frequency of 70 Hz.

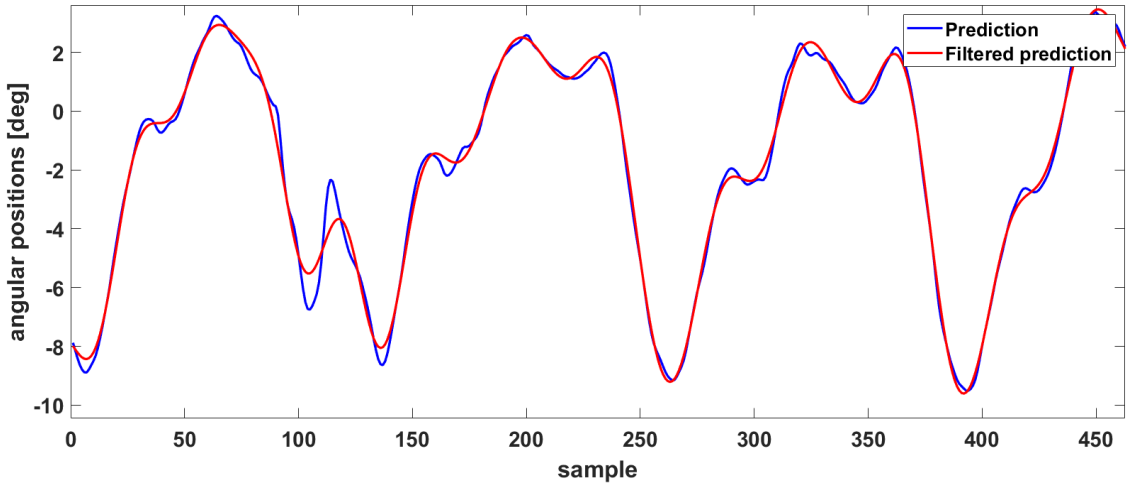


Figure 8.29: Reconstructed angular positions before and after low-pass filtering using LSTM for the Incline Experiment dataset

Conclusions, contributions and future works

9.1 Conclusions

The experiments and simulations discussed up to now have produced results that are far from perfect, but the methodologies chosen to conduct them have proved to be more than adequate. Most of the limitations in fact seem to be linked to the lack of more precise data and of some information that would certainly improve the overall quality of the project. Unfortunately, the development of this thesis was concomitant with the COVID pandemic: it limited the possibility of creating a dataset, using a suitable measurement protocol, to obtain all the necessary information. These have been reconstructed according to standard rules, or through various procedures, but this is an obvious cause of errors.

In sixth chapter the kinematic data were processed to obtain the information needed for the generation of the ZMP, of the feet' meta and CoM trajectories, and in the following chapter for the construction of the kinematic model.

The ZMP determined from the evaluation of feet events, derived from the measured positions of the heel and toe markers, gave very good results along the straight walking direction. It can be seen in 6.5a how it almost overlaps with the position data of the CoP in the same direction. The choice of simulating the displacement of the ZMP from heel to toe, as proposed by SangJoo Kwon et al. in [25], proved to be consistent with the experimental data. However, in the lateral direction, as shown in 6.5b, ZMP and CoP do not coincide. While from the experimental data we can deduce the pronation of the support foot, with consequent displacement of the CoP from the inside to the outside of the sole, the ZMP remains in the centre. Anyway, the position of the ZMP along both axes moves congruently with the double and single stance phases, determined from the experimental data and evaluated by observing the CoP trend.

The generated trajectories of the meta are very similar with the behaviour of the same obtained from the markers of the dataset. The formulation used to calculate them proved to be suitable for modelling their motion, in accordance with human locomotion.

The trajectory of the CoM was determined according to the LIMP model proposed by Kajita et al. in [21], with the method developed by Tedrake et al. in [41]. This involves the tracking of a reference ZMP, and the generation of a CoM trajectory consistent with the LIMP model: both prove to be optimal. The respective speed of the CoM along the direction of locomotion has an average value equal to the measured speed of the walk, which can be found among the information about the various experimental trials.

Pros	Cons
• Redundant kinematics makes it possible to consider both the natural behaviour of the patient and the balancing action of the variables generated in Cartesian space. • The kinematic model of the biped is very close to the real one in terms of DOF.	• The number of values requiring tuning is high, partly because of the high number of DOF. • The use of different kinematic chains depending on the support state of requires more operations than using a single kinematic chain. • Possible slippage of the support point is not taken into account. • The feet prono-supination cannot be taken into account.

Table 9.1: Pros and cons of the simulation for fitting the data to the kinematic model.

Moreover, its course is always between the two feet: if it is outside one of them, the risk of falling would be elevated. Therefore, the speeds and trajectories generated by the CoM are typical of a balanced walk.

In the seventh chapter, an attempt was made to fit to a kinematic model of a biped the generated trajectories and velocities in combination with the experimental information on knees' rotations. The aim was to assess whether it is possible to have a CoM motion that ensures balance when walking, and a pattern of knees' angles similar to that of the subjects of the trials at the same time. The various models were sized on the basis of anthropometric data obtained from marker analysis, and to guarantee all the DOF and rotations typical of the joints of a human's lower limbs. The kinematic cycle developed to fit the variables to the models proved to be suitable for the purpose. It allows you to decide how to adjust the contributions of the reference inputs through the choice of a series of gains, and thus decide whether to favour a joint movement that is synchronous with the physiological joint patterns or a kinematically balanced locomotion. Unfortunately, this strength also proved to be a weakness: the large number of values to be tuned made it hard to find the most suitable ones immediately.

Although the Cartesian variables generated seem to be coherent with the gait of the subjects of the various trials, they contrast with the measured physiological joints' rotations. Several possible causes for this problem have been identified. One of these could be an incorrect models sizing: for its development, it was necessary to start from the few experimental information provided about it, and the extensions of the various body segments were obtained by processing the available body markers data. Another problem could be that it has been chosen to consider null the rotations on the medial plane of the body and those on the frontal plane of the foot (i.e. its prono-supination). Although these

are small, their contributions could add up and modify the configuration of the model. We remind that the models used are four, representing different kinematic chains that start from the various possible support points. Are required a series of passages when switching from one to the other while maintaining a certain continuity: these can be a source of error. It is not possible to simulate foot slippage because the support points are considered to be fixed to the ground and are therefore it is not taken into account.

In chapter eight, the results showed that the LSTM neural network could perform better than the NARX neural network. This is due to the ability of the LSTM network to counteract the disappearance and explosion of the gradient. It was also possible to use a high number of hidden layers without requiring too much time for training. On the other hand, for the NARX network, it was not possible to set a high number of hidden layers because training became too long. Furthermore, as the number of connections of the NARX network could not be increased, the ability to reproduce a model for interacting EMG and joint signals was limited.

The LSMT network again showed no significant differences for training on either dataset. The NARX network, on the other hand, performed much better with the Biolab dataset. The segments to reproduce in the Biolab dataset were less than the input ones. On the contrary, for the Incline Experiment dataset, the number of the outputs exceeded the number of the inputs. Therefore, the NARX network ran into difficulties when the ratio of input to output was not even one to one. Consequently, for this type of experiment, the number of input data must be at least equal to the number of output data, preferably greater.

As expected, for both networks, the performance was higher for the training data. For the NARX neural network, the difference between training and testing data was considerable, while for the LSTM network, numerically speaking, the difference between training and testing was lighter. In any case, a good coefficient of determination for training data does not guarantee a good network prediction on never before seen data (testing data).

Finally, taking into account the type of input used for training, with the LSTM network, the simulations were excellent regardless of whether raw EMG signals, envelopes or synergies were used as input. In contrast, for the NARX network, on average, the best results were obtained when envelopes were used as input. As it turns out, the performance of the networks (at least for NARX networks) is significantly improved by cleaning the EMG signals of noise and excessive segments. As far as synergies are concerned, the result of their use for training is visible in the performance of the NARX network. In the Biolab dataset, the number of EMGs exceeded the number of angular signals, so a reduction in the number of inputs simplified the network's task. In fact, for this dataset, the performances obtained with the synergies (R^2 0.841) exceeded those obtained with the envelopes (R^2 0.815), at least for what concerns the training data. On the other hand, in the Incline Experiment dataset, where the number of EMG signals was lower than the number of angular signals, synergies to reduce the number of inputs only impoverished an already poor input situation. In fact, the performance of the envelopes (R^2 0.705), in this case, exceeded that of the synergies (R^2 0.429). It follows that synergies with the right data can be used as envelopes for training neural networks, sometimes even achieving better results.

9.2 Future works

The various limitations and advantages that can be observed from the results of the various experiments were underlined. For the future we want to make a series of proposals to improve the work already done, and to continue the development of a rehabilitative exoskeleton for the lower limbs:

- It would be advisable to build a new experimental dataset, in order to have more accurate sizing for the model. The various measurements of the joint angles and EMG signals should be accompanied by the precise size of the body segments, in order to build models that adequately represent the human body. The use of motion capture systems is one of the most suitable solutions for this purpose: it allows both the angles at the joints and the positions in space of the markers to be measured in real time. In order to also have the necessary data for training and validating the neural networks, an EMG signal measurement system, with a good number of channels, must be used in parallel.

- Starting from the direct and inverse kinematics proposed, a control system can be developed for a dynamic model. The criterion for assessing bipedal equilibrium might be the ZMP. This can be measured in real time on the soles of the feet, and using the LIMP it is possible evaluate the CoM motion and thus the dynamic balance.

- Although the performance for the LSTM network was high,it must be considered that the signals are taken into account, although not precise, had a certain periodicity. Therefore, it would be interesting to record EMG and angular joint signals for free body exercises with various movements and to train the neural networks on these recordings. Probably with this new data set, the performance of the network would drop at the moment. In this regard, the network should be trained again with more accurate parameters, with particular attention to the number of hidden layers, the maximum number of epochs, the initial learning rate, and the learning rate drop factor. To do this, it is sufficient to construct a code in MATLAB consisting of a series of nested *for* cycles, the purpose of which is to test all possible combinations of parameters, defined within certain limits, and then return the best configuration. The time required for this simulation could take several days, so it must be run on a computer that can handle the amount of work.

- About muscle synergies,a possible study could take EMG signals from unhealthy patients and then reconstruct the synergies from these signals. From the obtained synergies, one could look for a correlation between the type of injury of the patient and the synergy(s) that lose relevance. In this respect, the interest would be to check whether in patients with a particular injury, all or only some of the synergies are affected by the injury. To conduct this experiment, we would need a good number of subjects with similar and non-similar injuries, as well as a good number of healthy subjects, from which to define standards for healthy synergies. To be able to make a comparison with the synergies of injured subjects.

- Carry out an experiment highlighting the effect that the angles generated by neural networks have on equilibrium. During the walking phase, the patient operated in a closed chain and therefore made imperceptible changes to maintain equilibrium in the contingent conditions of that walk. The reconstruction of the joint angles and therefore of the gait from the EMG signals is essentially in open chain. Therefore without any reference to equilibrium which is slowly lost due to numerical approximations. The introduction of feedback with perturbation of the signals provided by the EMG signals must represent those contingent corrections (in the simulation) which are otherwise lost.

9.3 Contributions

In these last lines we would like to thank the people who helped us along this path, providing the necessary tools and listening to our doubts, and to describe the personal contributions of the two authors to this work.

On the biomedical side, we would like to thank the Professor Valeria Agostini and Marco Ghisleri, who shared their knowledge with us in the field of EMG signal processing, on the gait analysis and on the design of neural networks. In addition, they offered us a database of measurements collected by their group.

Our research supervisor, the Professor Giuseppe Menga, followed us through the various stages of project's development: he introduced us to the modelling of bipedal walking, to the various balance criteria, and the development of the kinematics was under his supervision and according to his indications and suggestions. He also suggested to model the relationship between EMG and angular signals using neural networks.

Luca Menegazzi was responsible for the generation of the foot CoM and ZMP trajectories, the related data processing, and the development of the systems for fitting them to the models. Federico Floris worked on the development of neural networks to correlate EMG signals with limb rotations, on the processing of the respective signals, and on all the tests carried out to assess the possibility of developing a control system based on muscular activity. We both worked on the study of the databases on which the whole project was based and on the creation of the various models used.

The path that led us to these final conclusions was by no means easy: the global pandemic, which forced us to remain distant, made collaboration much more difficult. The two authors would like to thank each other: the desire to cooperate and to do their utmost for a common goal has often made what could have been laborious and lacking in enthusiasm more enjoyable and engaging, and a proactive attitude has always been displayed by both.

Bibliography

[1] 3Blue1Brown. *But what is a neural network? — Chapter 1, Deep learning*. 2017. URL: `https://youtu.be/aircAruvnKk` (visited on 07/16/2021).

[2] 3Blue1Brown. *Gradient descent, how neural networks learn — Chapter 2, Deep learning*. 2017. URL: `https://youtu.be/IHZwWFHWa-w` (visited on 07/16/2021).

[3] Amos Albert and Wilfried Gerth. "Analytic path planning algorithms for bipedal robots without a trunk". In: *Journal of Intelligent and Robotic Systems: Theory and Applications* 36.2 (2003), pp. 109–127.

[4] Richard Baker et al. "Handbook of Human Motion". In: *Handbook of Human Motion*. Ed. by Springer. 2018. Chap. The Conven.

[5] BERTEC Corporation. *Bertec Force Plates*. March. 2012, pp. 13–13.

[6] BERTEC Corporation. *Instrumented Treadmill User Manual*. 2013.

[7] Siciliano Bruno et al. *Robotics Modelling, Planning and Control*. 2009.

[8] Youngjin Choi et al. "Posture/walking control for humanoid robot based on kinematic resolution of CoM Jacobian with embedded motion". In: *IEEE Transactions on Robotics* 23.6 (2007), pp. 1285–1293.

[9] A Covallero. *Confronto tra metodi per l'analisi di manifestazioni elettriche di fatica muscolare in pazienti diabetici, con e senza vasculopatia periferica, e soggetti sani durante camminata su treadmill*. Univesità Degli Studi di Padova, 2014, pp. 1–108.

[10] d'Avella D. Torricelli et al. "Muscle Synergies in Clinical Practice: Theoretical and Practical Implications". In: 10 (2016), pp. 65–85.

[11] Wilfred Dempster. *Space requirements of the seated operator geometrical, kinematic, and mechanical aspects of the body*. 1955.

[12] R. Drillis, R. Contini, and M. Bluestein. "Body Segment Parameters; a Survey of Measurement Techniques." In: *J. Bone Joint Surg.* 48-A (1964).

[13] Kyle R. Embry et al. "Modeling the Kinematics of Human Locomotion over Continuously Varying Speeds and Inclines". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 26.12 (2018), pp. 2342–2350.

[14] Abu Ilius Faisal et al. "Monitoring Methods of Human Body Joints :" in: *Sensors (Switzerland)* (2019).

[15] M Ghislieri et al. "A Machine Learning Approach for Muscle Activity Detection". In: ed. by GNR. 2020, pp. 10–13.

[16] Marco Ghislieri, Valentina Agostini, and Marco Knaflitz. "Muscle Synergies Extracted Using Principal Activations: Improvement of Robustness and Interpretability". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 28.2 (2020), pp. 453–460.

[17] Ambarish Goswami. "Postural stability of biped robots and the foot-rotation indicator (FRI) point". In: *International Journal of Robotics Research* 18.6 (1999), pp. 523–533.

[18] R Macaluso; K Embry; D Villarreal; R Gregg. *Human Leg Kinematics, Kinetics, and EMG during Phase-Shifting Perturbations at Varying Inclines*. 2020. URL: `https://dx.doi.org/10.21227/12hp-e249` (visited on 11/13/2020).

[19] Hirohisa Hirukawa et al. "A universal stability criterion of the foot contact of legged robots - Adios ZMP". In: *Proceedings - IEEE International Conference on Robotics and Automation* 2006.May (2006), pp. 1976–1983.

[20] Perry Jacquelin. *Gait Analysis, Normal and Pathological Function*. SLACK Incorporated, 1992.

[21] Shuuji Kajita et al. "A realtime pattern generator for biped walking". In: *Proceedings - IEEE International Conference on Robotics and Automation* 1.March 2014 (2002), pp. 31–37.

[22] Shuuji Kajita et al. "Biped walking pattern generation by using preview control of zero-moment point". In: *Proceedings - IEEE International Conference on Robotics and Automation* 2 (2003), pp. 1620–1626.

[23] Tohru Katayama et al. "Design of an optimal controller for a discrete-time system subject to previewable demand". In: *International Journal of Control* 41.3 (1985), pp. 677–699.

[24] Vikash Kumar, Yogesh V. Hote, and Shivam Jain. "Review of Exoskeleton: History, Design and Control". In: *2019 3rd International Conference on Recent Developments in Control, Automation and Power Engineering, RDCAPE 2019* (2019), pp. 677–682.

[25] Sang Joo Kwon and Jinhee Park. "Kinesiology-based robot foot design for human-like walking". In: *International Journal of Advanced Robotic Systems* 9 (2012).

[26] Leonardo Mangiapelo. "Implementing an Electrogoniometer Using Freescale's low g accelerometers". In: *It's making the world a smarter place* 4 (2009), pp. 57–60.

[27] Tad McGeer. "Passive walking with knees". In: *School of Engeeniring Scienze, Simon Fraser University* (1990).

[28] Giuseppe Menga and Jie Geng. "LOWER LIMB EXOSKELETON HAPTIC DEVICE FOR REHABILITATION OR WALKING , WITH IMPROVED EQUILIBRIUM CONTROL , MAN-MACHINE INTERFACE". In: *Robotics* (2019), pp. 1–13.

[29] Michael Nielsen. "Learning with gradient descent". In: *Neural Netw. Deep Learn.* Determination Press, 2015. Chap. 1. URL: `http://neuralnetworksanddeeplearning.com/chap1.html%7B%5C#%7Dlearning_with_gradient_descent` (visited on 07/16/2021).

[30] Michael Nielsen. "Perceptrons". In: *Neural Netw. Deep Learn.* Determination Press, 2015. Chap. 1. URL: http://neuralnetworksanddeeplearning.com/chap1.html%7B%5C#%7Dperceptrons (visited on 07/16/2021).

[31] Michael Nielsen. "The four fundamental equations behind backpropagation". In: *Neural Netw. Deep Learn.* Determination Press, 2015. Chap. 2. URL: http://neuralnetworksanddeeplearning.com/chap2.html%5C#the_four_fundamental_equations_behind_backpropagation (visited on 07/16/2021).

[32] Michael Nielsen. "The two assumptions we need about the cost function". In: *Neural Netw. Deep Learn.* Determination Press, 2015. Chap. 2. URL: http://neuralnetworksanddeeplearning.com/chap2.html%5C#the_two_assumptions_we_need_about_the_cost_function (visited on 07/16/2021).

[33] Nuruzzaman Faruqui. *Deep Learning using Matlab*. 2019. URL: https://youtu.be/HSDyiCuViWg (visited on 07/16/2021).

[34] Nuruzzaman Faruqui. *Neural Network using Matlab*. 2019. URL: https://youtu.be/WoLBeWclYxo (visited on 07/16/2021).

[35] Christopher Olah. "Understanding LSTM Networks [Blog]". In: *GitHub* (2015), pp. 1–13. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[36] Alberto Plaza et al. "Lower-limb medical and rehabilitation exoskeletons: A review of the current designs". In: *IEEE Reviews in Biomedical Engineering* (2021).

[37] Mohammad Fazle Rabbi et al. "Non-negative matrix factorisation is the most appropriate method for extraction of muscle synergies in walking and running". In: *Scientific Repeports* 10.1 (2020), pp. 1–11.

[38] Rachele Rossanigo. *Analysis of spatial parameters during gait with magneto-inertial sensors and infrared proximity sensors*. Politecnico di Torino, 2019.

[39] Juri Taborri et al. "Feasibility of muscle synergy outcomes in clinics, robotics, and sports: A systematic review". In: *Appl. Bionics Biomech.* April (2018).

[40] Russ Tedrake. *Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation (Course Notes for MIT 6.832)*. URL: http://underactuated.mit.edu/ (visited on 07/04/2021).

[41] Russ Tedrake et al. "A closed-form solution for real-time ZMP gait generation and feedback stabilization". In: *IEEE-RAS International Conference on Humanoid Robots* 2015-December (2015), pp. 936–940.

[42] Inc. The MathWorks. *Deep Learning Toolbox (R2020b)*. Natick, Massachusetts, United State, 2018. URL: https://it.mathworks.com/help/deeplearning/ (visited on 07/16/2021).

[43] Inc. The MathWorks. *Robotics System Toolbox (R2020b)*. Natick, Massachusetts, United State, 2018. URL: https://it.mathworks.com/help/robotics/ (visited on 07/16/2021).

[44] Inc. The MathWorks. *Simscape (R2020b)*. Natick, Massachusetts, United State, 2018. URL: https://it.mathworks.com/help/physmod/simscape/ (visited on 07/16/2021).

[45] Inc. The MathWorks. *Statistics and Machine Learning Toolbox (R2020b)*. Natick, Massachusetts, United State, 2019. URL: https://it.mathworks.com/help/stats/ (visited on 07/16/2021).

[46] Vicon Motion Systems Ltd. *Plug-in Gait Reference Guide*. 2017. URL: http://www.crcnetbase.com/doi/10.1201/b10400-14 (visited on 07/16/2021).

[47] Miomir Vukobratovic and Branislav Borovac. "Zero-Moment Point — Thirty Five Years of Its Life". In: *International Journal of Humanoid Robotics* 01.01 (2004), pp. 157–173.

[48] Miomir Vukobratovic and Dragoljub Surdilovic. *Zero-Moment Point - Proper Interpretation and New Applications*. 2001.

[49] David A. Winter. *Biomechanics and Motor Control of Human Movement: Fourth Edition*. John Wiley and Sons Inc, 2009.