# POLITECNICO DI TORINO

Master's Degree in Mechatronic Engineering (Ingegneria Meccatronica)

Master's Degree Thesis

## Study of the control of an industrial exoskeleton with pneumatic actuation


**Advisor:**
Prof. Luigi Mazza

**Candidate:**
Drocco Giacomo

**Co-Advisors:**
Prof. Terenziano Raparelli
Prof. Gabriella Eula
Dr. Giuseppe Pepe


Academic Year 2019/2020

*Vorrei dedicare questo lavoro conclusivo del mio percorso universitario a tutte le persone che mi sono state vicino in questi anni. Vorrei ringraziare i nonni Pippo per avermi insegnato a non abbattersi mai e a rialzarsi più forti di prima e sempre con il sorriso per trasformare una difficoltà in un'opportunità, vorrei poi ringraziare anche i nonni Billo per avermi insegnato ad affrontare la vita con passione e dedizione e soprattutto vorrei dedicare questo lavoro a mia mamma e mio papà per avermi cresciuto ed educato con valori forti e sinceri, per essere stati la mia fonte di ispirazione per la vita e per aver reso possibile il raggiungimento di questo traguardo grazie ai loro insegnamenti e al loro esempio.*

*Infine, vorrei ringraziare la mia fidanzata Lucia che mi è stata vicina durante l'intero periodo universitario aiutandomi nei momenti più difficili non facendomi mai mancare affetto e spensieratezza.*

# Table of contents

# List of figures

# Abstract

The objectives of this work are the study, design and simulation of the control logic of an active exoskeleton for industrial applications. The device, thanks to the pneumatic actuation, has the task of producing a torque, equal to 30% of the torque produced by the human being, able to reduce human muscle fatigue at the hip joint. The introductory part presents the state of art and the basic concepts of the devices on the market. The thesis is divided into six chapters: the first section introduces the subject focusing on the introductory control scheme presented in previous works. In the next chapter it is analysed the various types of controllers that were suitable for our device. The third and fourth chapters present the model of control logic developed through studies in MATLAB, more precisely through the Simscape Multibody and StateFlow interface. The fifth chapter contains all the simulation results with the relative analysis. Then, in the last section the possible future developments and the conclusion of the work were indicated.

# 1 Introduction

## 1.1 Motivation

The main goal of this thesis is to study, design and optimize the control of an industrial hip joint exoskeleton that factory workers could wear, since they are constantly exposed to lower back and hip pain. This exoskeleton is aiming to reduce fatigue while performing trunk bending, and to assist defined tasks. There is also the possibility that these exoskeletons could facilitate earlier return-to-work, after an injury. Nowadays, wearable robots such as exoskeletons, which are designed to be used in an industrial field are the fastest growing research topic. Our plan is to define some of the aspects that are dealing with the controllability, for a high-level design. In practice, we try to figure out which are the most effective control scheme for better functional requirements, that could be used to make the exoskeleton comfortable for the wearers and efficient from an industrial point of view and then we are focus also on the design of models to simulate the different operating condition of the device.

## 1.2 What exoskeleton means?

The term exoskeleton initially represented a biological term referring to the external skeleton of an animal or vegetal that supports and protects the internal organs of the body (Fig. 1.1). From the industrial point of view (Fig. 1.2), the term will refer to a wearable, external mechanical structure that enhances the power of a person.

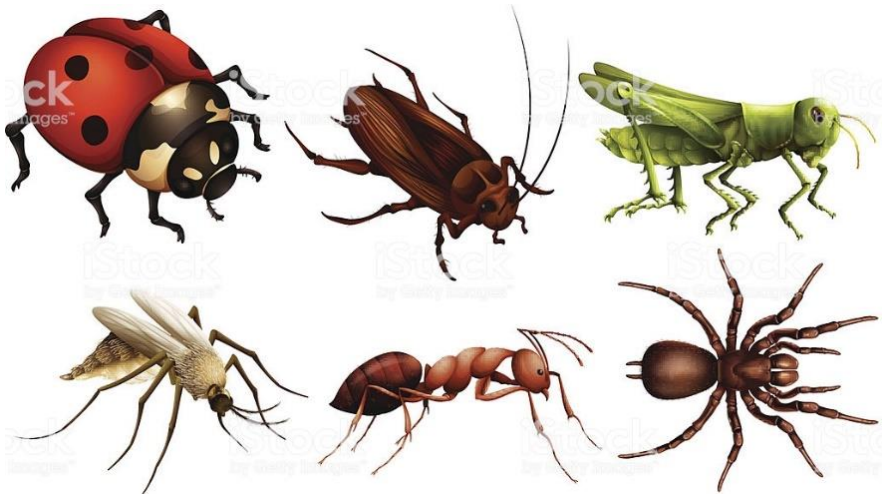


*Figure 1.1: Animal exoskeleton [1]*


*Figure 1.2: Industrial exoskeleton [2]*

These are devices worn and controlled by a human to compensate the mass of the device and payload, and to amplify its forces, integrating only low-level control. Contrary to classical industrial manipulators, exoskeletons high level control is designed with Human-in-the-loop (HIL) enabling for the operator to take high level decision. Many different terms have been used to describe exoskeletons, such as “back support,” “lift assist,” “lumbar support,” “hip orthosis,” “spinal exoskeleton.” For the sake of simplicity, we refer to them as “back-support exoskeletons.”

### 1.2.1 History of wearable exoskeleton

At the beginning of the 1800 the inventors started to think to a mechanical device able to assist the human body. The initial concept was a steam-powered wearable device designed to help people to walk (Fig.1.3a), which has inspired many others since then. For example, around 1890, Nicholas Yagn designed a device to assist Russian soldiers in jumping higher and running faster (Fig. 1.3b).

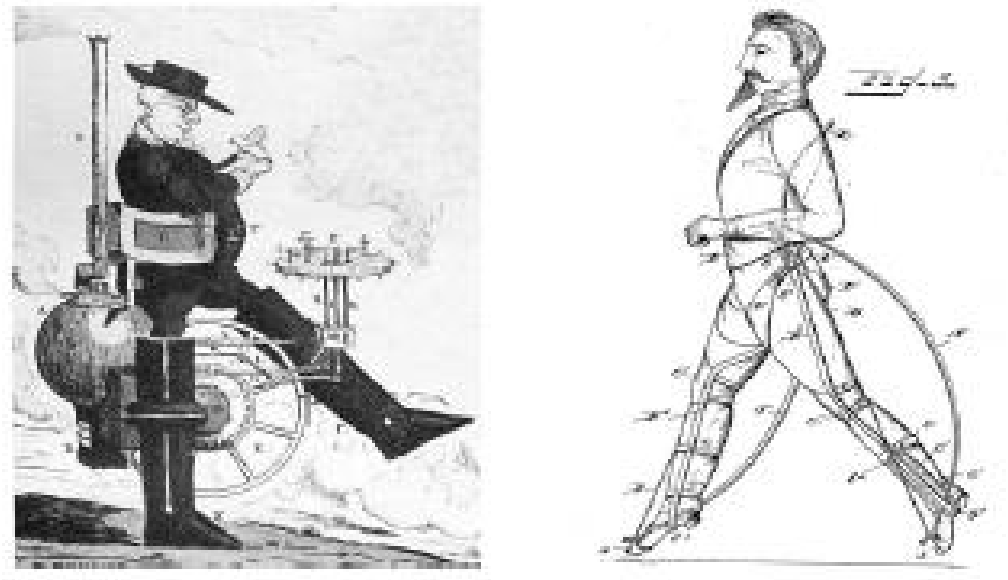


*Figure 1.3: (a) Steam powered legs [3], R.Seymour (b)Soldier’s passive assistance legs, N.Yagn [4]*

Obviously during these years, the technology was not so advanced and so these ideas remained concept and they were not thorough. The first modern

exoskeleton, shown in the figure below (Fig. 1.4), was designed around the 1965 by General Electric to help people to carry heavy loads.

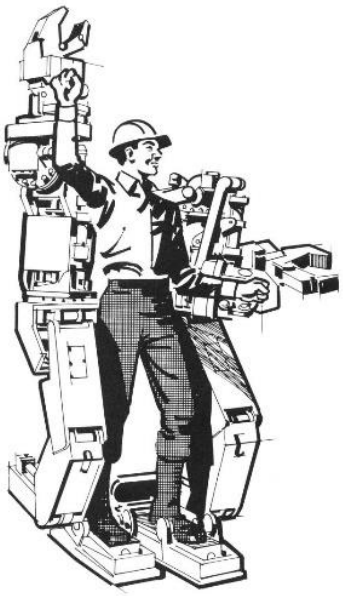


*Figure 1.4: The Hardiman [5]*

This exoskeleton had a lot of limits like the considerable mass (≈700 Kg), the difficulties in detecting human motion intent and many others that stopped all the future development. Nowadays, despite the on-going trend in automation and mechanisation in industry, many workers are still exposed to physical workloads due to material handling (over 30% of the work population in the EU), repetitive movements (63%) and awkward body postures (46%) (Eurofound 2012). In the European Union, yearly more than 40% of the workers suffer from low back pain or neck and shoulder pain (Eurofound 2012). The same situation also characterizes the United States where approximately 2 363 960 people are employed under the designation of Manual Freight, Stock, and Material Movers (Bureau of Labor Statistics). For example, the shoulder was involved in 13% of Work-related musculoskeletal disorders (WMSD) cases reported in 2011 (Bureau of Labour Statistics 2012), second only to the back (42% of the cases). Shoulder WMSDs may be relatively more severe, and recent evidence indicates that they led to a median of 23 lost workdays compared to 11 lost workdays across all body regions (Bureau of Labour Statistics 2012). The National Institute of Occupational Safety and Health (NIOSH) recommend

maximum lifting loads for repetitive lifting by healthy workers, capped at 223 N (22,73 Kg), and reduced based on several risk factors. NIOSH recommends a maximum spinal compression of 3400 N during lifting, called the action limit. Note, during bending or squatting, a moment will be generated about the hips (axis of rotation) due to the weight of the body superior to the hips and any load being lifted. 85 to 95 percent of low back injuries occur at the L5/S1 and L4/L5 discs and are evenly split between the two. Existing lift assist systems are unacceptably slow for many tasks thus reducing productivity. So, with significant advances in robotics technology in recent years, for manual handling task wearable lifting assist devices (LADs) (i.e., exoskeletons in the robotics field) have been suggested as a tool to help industrial athletes and to prevent lower back pain (LBP). It is possible to see the growth in the research about the exoskeleton analysing the data given by PatentInspiration® which illustrate the number of publications (Fig. 1.5) and the patent distribution around the world of exoskeletons (Fig. 1.6).

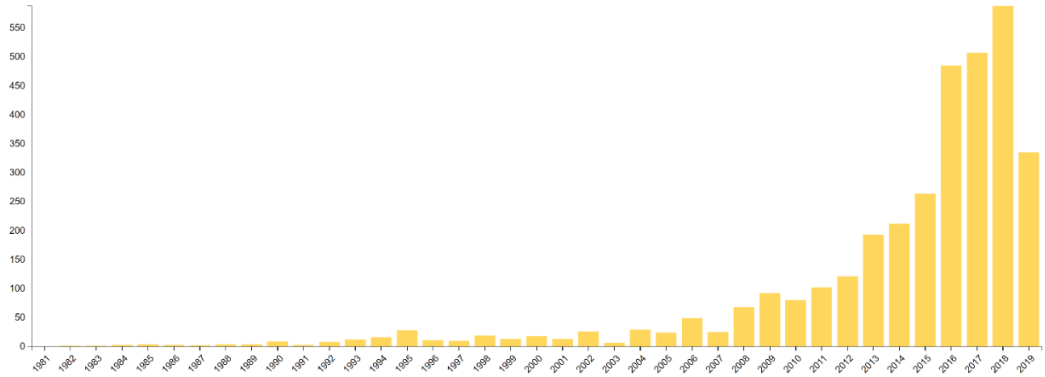



*Figure 1.5: Trend in the world of exoskeleton publications*

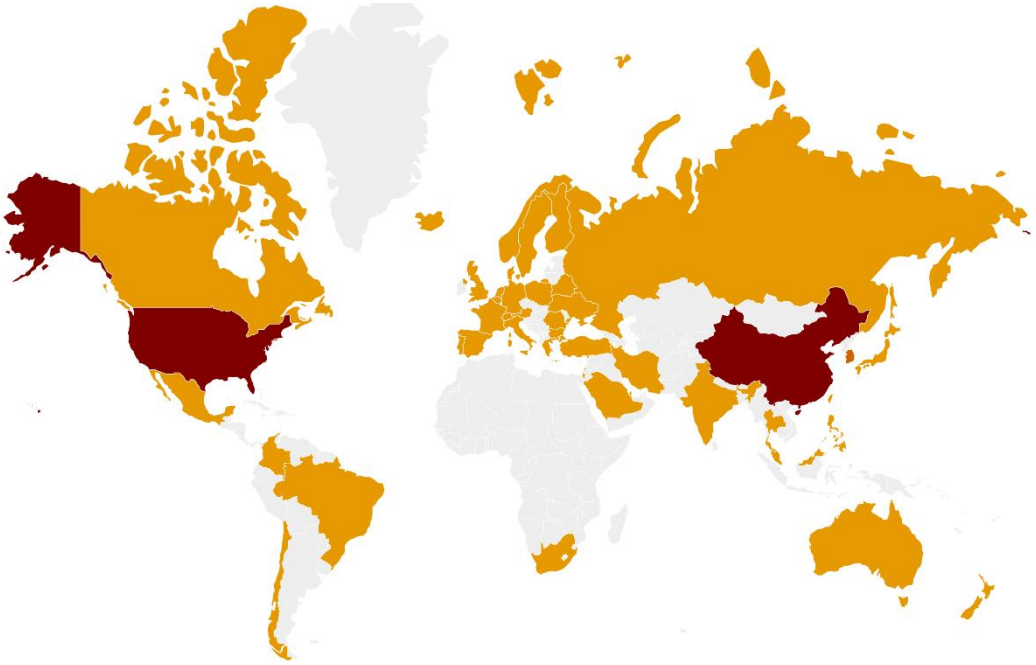


*Figure 1.6: Countries inventors of patents on exoskeletons*

As you can see, exoskeletons are spreading more and more and consequently the number of manufacturers grows with the time. In the figure below you can see a part of the producers divided by fields on which they are more concentrated.


*Figure 1.7: Exoskeleton producers [6]*

### 1.2.2 Classification of exoskeleton

Exoskeletons can be classified as:

1. **Active exoskeleton:** it is an external structural mechanism with joints and links, powered with technologies to mimic the movement of the corresponding to those of the human body. This type of exoskeletons contains electromagnetic, electrohydraulic, electropneumatic, or other types of actuators that generate the external forces. The action of these actuators is controlled during operation by a computer program based on sensor information. Usually they are equipped with onboard batteries.

*Figure 1.8: The Muscle Suit [7]*

A possible approach is characterized using electromagnetic clutches which can decuple the actuator from the exoskeleton when no assistance is necessary. The advantage of this approach is the reduced energy consumption and reduced undesirable actuator dynamics.

2. **Passive exoskeleton:** this type of system does not use any actuator but use intrinsically passive elements such as springs, which store energy during negative work (i.e., lowering movements) to be recycled during positive work (i.e., lifting movements). Also, the simpler passive devices are not yet widely used in practice probably because they are not so comfort when are wearing. Existing passive back-support exoskeletons employ elastic elements of different types. Elastic bands are used on several devices, including the Personal Lift Augmentation Device (PLAD). On the other hand, the SPEXOR device (Koopman et al., 2018) employs flexible beams to transfer the assistive torque between the pelvis and the torso.

*Figure 1.9: SPEXOR, a passive exoskeleton [8]*

The passive devices can be used for several task like:

- Weight re-distribution.
- Damping.
- Locking.
- Energy capture.

3. **Semi-active or quasi-passive:** A semi-active/quasi-passive exoskeleton is a device in which the coupling/decoupling or the mechanical properties of passive elements (e.g., spring or dampers)

could be modulated automatically during operation. For example, the C-Brace model which alternatively unlocks slows down the swing of the leg and locks depending on the position of the leg in the gait cycle as determined by the integrated sensors.

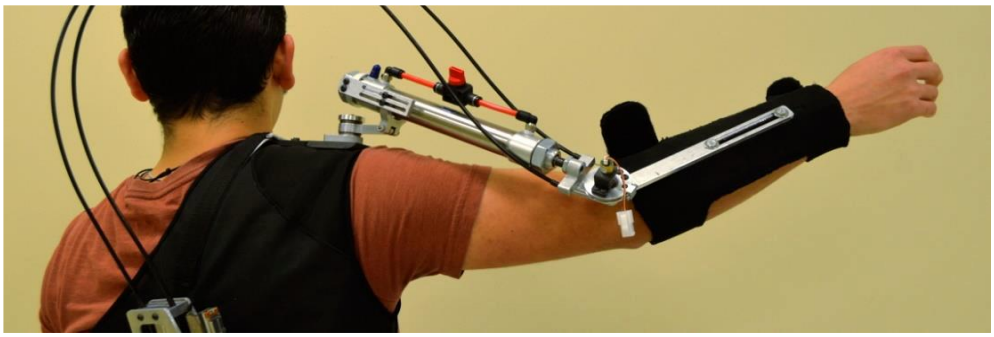


*Figure 1.10: Pseudo-passive exoskeleton [9]*

### 1.2.3 Application area

The main application area of exoskeletons has been for medical /rehabilitation purposes where the devices are aimed to support physically weak, injured, or disabled people to perform a wide range of motions involved in activities of daily living. A small number of exoskeletons have also been designed for military applications for soldiers to lift or carry heavy loads and for industrial application.

*Figure 1.11: Medical exoskeleton [10]*

*Figure 1.12: (a)Military exoskeleton [11], (b)Industrial exoskeleton [12]*

In this work we are focus on the field of the industrial application.

## 1.3 Industrial exoskeleton

Industrial exoskeletons are the name correlated to the majority of biomechanical devices that worn by workers. It replicates the structure of the worker`s limbs, muscles, and joints working in tandem with them. It is useful to think the industrial exoskeletons as wearable robots that take advantage of the worker intelligence and the strength of industrial robots. Exoskeletons are like the traditional robots, but the difference is that they work on the idea of physically demanding sensing given by the worker body with some semi-automated operations. At that point we are trying to take the advantages of both traditional industrial robots with entering action done by the human intelligence. The manufacturing industry is a labour-intensive sector with extensive manual work that could benefit from the use of exoskeletons. More specifically, the automobile industry could greatly benefit from exoskeletons because it still strongly relies on manual operations that are difficult to robotize. Despite considerable efforts made to adapt the design of vehicle parts, human operators often overcome robots thanks to their intelligence and ability to climb into the car or to move rapidly around the assembly. Exoskeletons provide a flexible alternative

when no other solution is appropriate. Testing processes are therefore being developed for industrial implementation.

*Figure 1.13: Exoskeleton uses in automotive industries [13] [14]*

### 1.3.1 Classification of industrial robot

1. **Full, upper and lower body:** there are many different type of exoskeleton on the market and we can distinguish them by the supported body part(s): providing power or support to the lower limbs (lower body exoskeletons, Fig. 1.14a), to the upper extremities (upper body exoskeletons, Fig. 1.14b) and to both upper and lower extremities (full-body exoskeletons, Fig. 1.14c).

*Figure 1.14: (a) Low-body [15] , (b) Upper-body [16], (c) Full-body [17]*

2. **Active and passive:** an active exoskeleton comprises one of more actuators that augments the human's power and helps in actuating the human joints. A strictly passive system does not use any type of actuator, but rather uses materials, springs, or dampers with the ability to store energy harvested by human motion and to use this as required to support a posture or a motion
3. **Rigid and soft:** Soft exoskeletons (also known as exosuits) are devices consisting of garments worn on body segments adjacent to the joint that is assisted, for example the thigh and shank for a knee exosuit. Assistance is generated by using the garments to pull two body segments together, typically via a cable or strap (Fig. 1.15a).
   Rigid exoskeletons are built with hard articulated structures that connect actuators to garments worn by the user. These articulated rigid structures run in parallel with body segments and apply forces perpendicular to them (Fig. 1.15b). These types of exoskeletons also tend to use space lateral to the user's body, increasing the lateral footprint in some scenarios.

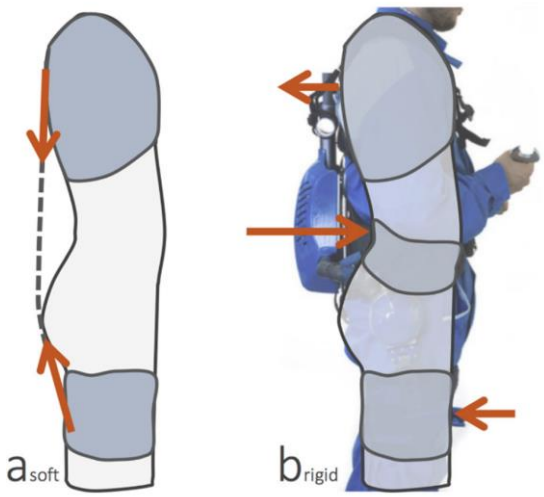



*Figure 1.15: Rigid and soft exoskeleton*

4. **Fully anthropomorphic and non-anthropomorphic:** anthropomorphic exoskeletons have exoskeleton joints with rotational axes that are aligned with the rotational movement of the human joints, which is not the case in the non-anthropomorphic

types. A fully anthropomorphic type enables the exoskeleton robot to make the same motions as the wearer thereby offering a large freedom of motion. But these systems pose major design challenges to ensure close fit for different size users while simultaneously accommodating natural movements by the user. Non-anthropomorphic types are generally simpler and can be designed to have an optimised structure for specific tasks to be performed allowing more effective energy consumption than anthropomorphic systems.

## 1.4 Control scheme

As mentioned earlier, in this work we will focus on the design of the exoskeleton control scheme. The models listed above make use of a PID controller which is the controller used in 95% of cases due to its flexibility to simplicity. Below we will analyze a control scheme of an exoskeleton found in bibliography from which we will take cues. The case in question is the LAD control strategy that is visible in the picture and which consists of a high-level (outer) and low-level (inner) control loops.

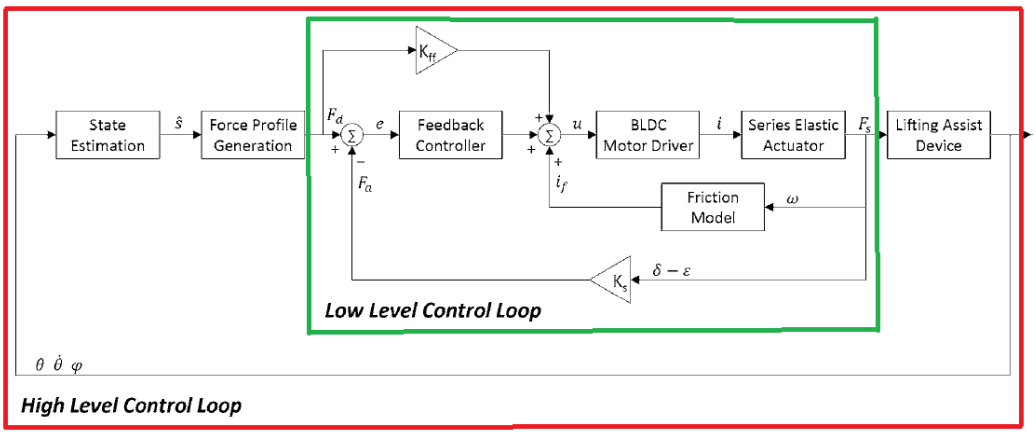



Figure 1.16: Bibliografy control scheme [18]

- The high-level controller: it is responsible for estimating the state of the wearer and generating the desired output force profile of the SEA.
    - State estimation: a finite state machine is designed to estimate the wearer's state during the lifting cycle. The finite state

machine divides the lifting cycle into three states: standing, lowering, and lifting. A transition is made from one state to another based on the values of the FE joint angles, angular velocity, and trunk inclination with respect to gravity, which can be acquired from sensors embedded in the LAD. The FE joint angles are calculated by measuring $\delta$ with the linear potentiometers and using the simple relation $\vartheta = {}^{\delta}/_{r}$ . The angular velocities are obtained by using the finite difference derivative method with an infinite impulse response low-pass filter. During operation, the standing state transitions to the lowering state occurs when the FE joints are flexed over 60° relative to the reference position. The lowering state transitions to the lifting state it happens when the FE joints are extended at a joint angular velocity of over 40deg/s. Finally, the lifting state transitions to the standing state occurs when the trunk inclination is below 10° relative to the reference position.

  - Force profile generation: during the lowering state, the SEA stays at a constant stiffness. After noticing that excessive stiffness impedes the lowering movement and causes discomfort to the wearer, we set the stiffness as low as possible to support the wearer's upper body weight while maintaining comfort. Then, we set the desired output force of the SEA to the maximum during the lifting state. During the standing state, the desired output force was set to zero.

- Low level controller: The purpose of the low-level controller is to track the desired output force coming from the high-level controller. The reference current input $u$ for the BLDC motor driver is designed as follows: $u = K_{ff}F_d + i_f + K_p e$. The first term $K_{ff}F_d$ is a feedforward term that represents the current input required to generate the desired output force Fd under the static condition. The feedforward gain $K_{ff}$ can be calculated from the specifications of the

BLDC motor with the ball-screw transmission and is given by: $K_{ff} = \frac{P}{2\pi K_t}$ where $P$ is the pitch of the ball-screw transmission and $K_t$ is the torque constant of the BLDC motor. The second term $i_f$ is a friction compensation term to eliminate the intrinsic friction behaviour in the BLDC motor with the ball-screw transmission. The third term $K_p e$ represents a proportional feedback control law, where Kp is the proportional gain and was set to $K_p = 0.0001$ in this study. is the difference between the desired output force $F_d$ and actual output force $F_a$ .

## 1.5 Our mission

The starting point, studied in the previous thesis, is a pneumatic exoskeleton realized with the aim of reduce by about 30% the wearer's muscle fatigue and so to reduce musculoskeletal problems of the lumbar area. The base structure is divided into three parts: the leg-link, the backframe and the motorized hip joint. A summary scheme is presented below.

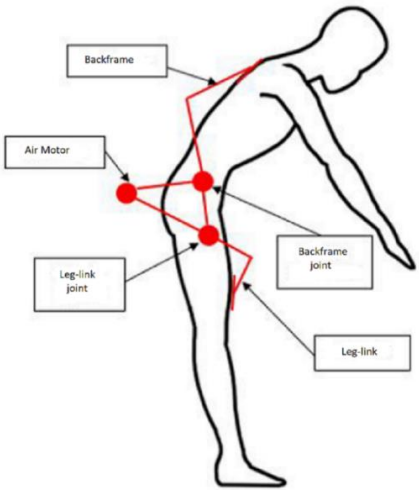



*Figure 1.17: Diagram of the pneumatic exoskeleton*

The main objective of the thesis will be to analyse the control part of the exoskeleton, the initial focus will be on the optimization of the scheme proposed by the previous student and integrating it with other types of

controller. Subsequently, define the Simscape Multibody models for the human body and for the exoskeleton. Then project a StateFlow control logic to simulate some real-life situations. The mechanical part of the exoskeleton was developed earlier, and different views of the CAD model are shown in the following figures.

*Figure 1.18: CAD model*

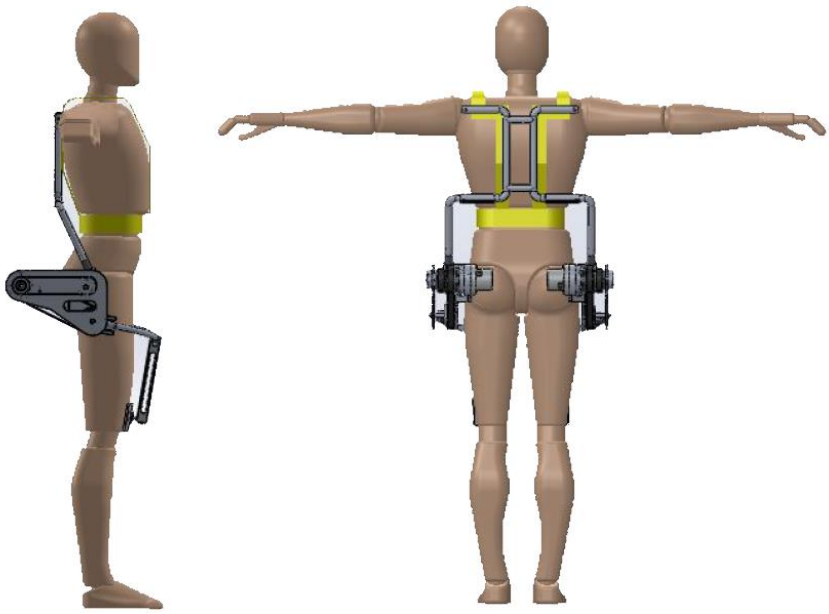


Figure 1.19: (a) Side view, (b) Rear view

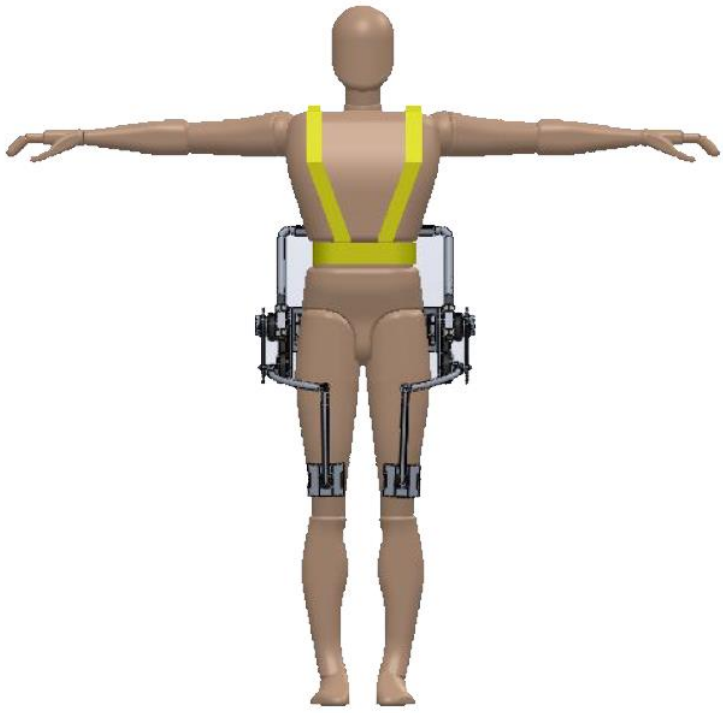


*Figure 1.20: Front view*

In conclusion, we deal with some simulations to show the behaviour of the device and then analyse the possible future development and conclusion of the thesis.

# 2 Mathematical model

As mentioned above, the objective of our work is to study a torque position control scheme, which allows us, based on the angular position, angular speed and angular acceleration detected by sensors, specially placed on the exoskeleton, to determine the driving torque produced by the pneumatic motor needed to support 30% of the muscle torque generated by the human body. In some previous works a starting scheme was presented, which in a rough way dealt with the torque control. The control scheme is shown in the following figure:

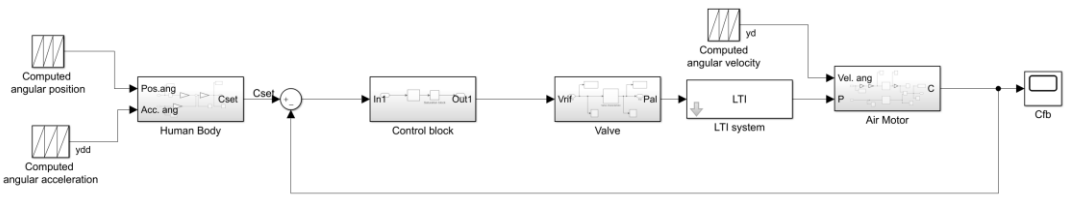



*Figure 2.1: Mathematical model*

The system, shown in Figure 2.1, is relative to the entire torso and therefore serves to simulate the user's bending operation. The $C_{SET}$ signal is the produced torque based on the angular position and angular velocity of the wearer multiplied by k=0.30 and it must be compared with the torque produced by the air motor that corresponds to the signal of feedback (FB). Below will be analysed the various blocks making up the system and the graphs of the numerical results obtained will be presented.

## 2.1 Input quantities

Some studies, conducted within the DIMEAS of the Polytechnic of Turin, similar to that done by the University of Tsukuba for the development of HALL, allowed by video analysis to characterize the bending movement in terms of bending angle as a function of time and it was concluded that the complete human bending is attributable to a cycloidal function. Subsequently, a work cycle was defined, characterised by four phases, each with its own duration and law of motion, whose characteristics are shown in the table (Table 2.1).

| Phase | Time (s) | Law of motion |
|---|---|---|
| *Bending* | 3 | Cycloidal |
| *Working* | 30 | Costant |
| *Lifting* | 3 | Cycloidal |
| *Resting* | 24 | Costant |

*Table 2.1: Work cycle phases*

**Bending phase**

About the first phase, as can be seen from the table, the law of motion is cycloidal, therefore the angular position, angular velocity and angular acceleration will be represented respectively by the following expressions:

$$\theta_f = \left(\tau - \frac{1}{2\pi}\sin(2\pi\tau)\right) * h \tag{2.1}$$

$$\dot{\theta}_f = \omega = (1 - \cos(2\pi\tau)) * \frac{h}{t} \tag{2.2}$$

$$\ddot{\theta}_f = (2\pi * \sin(2\pi\tau)) * \frac{h}{t^2} \tag{2.3}$$

The figure 2.2 shows the trends, obtained with MATLAB of the three quantities treated so far on a time horizon of 3s, that is the expected duration for this phase.

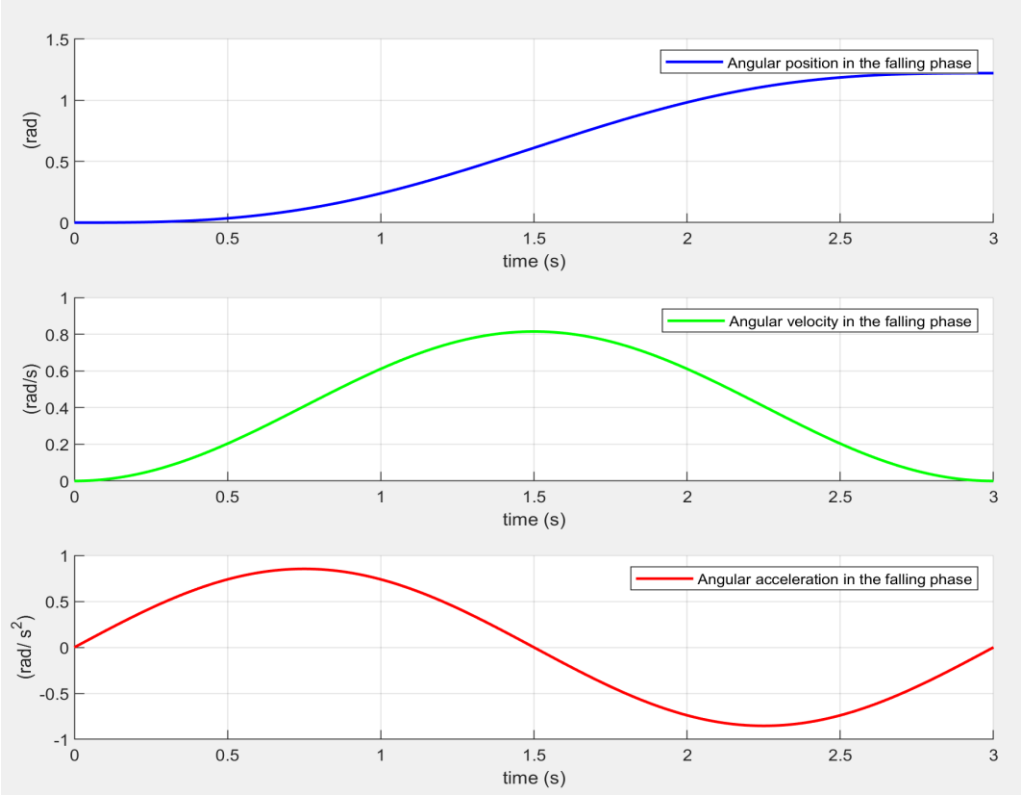



*Figure 2.2: Input quantities trend in bending phase*

**Working phase**

In the second phase the law of motion is constant, therefore the angular position, angular velocity and angular acceleration will be represented by the following expressions respectively:

$$\theta_w = h \tag{2.4}$$

$$\dot{\theta_w} = \omega = 0 \tag{2.5}$$

$$\ddot{\theta_w} = 0 \tag{2.6}$$

Subsequently, as was done for the previous phase, the trends of the three quantities treated so far are shown in the figure 2.3 over a period ranging from 3 s to 33 s, i.e. the expected duration for this phase.

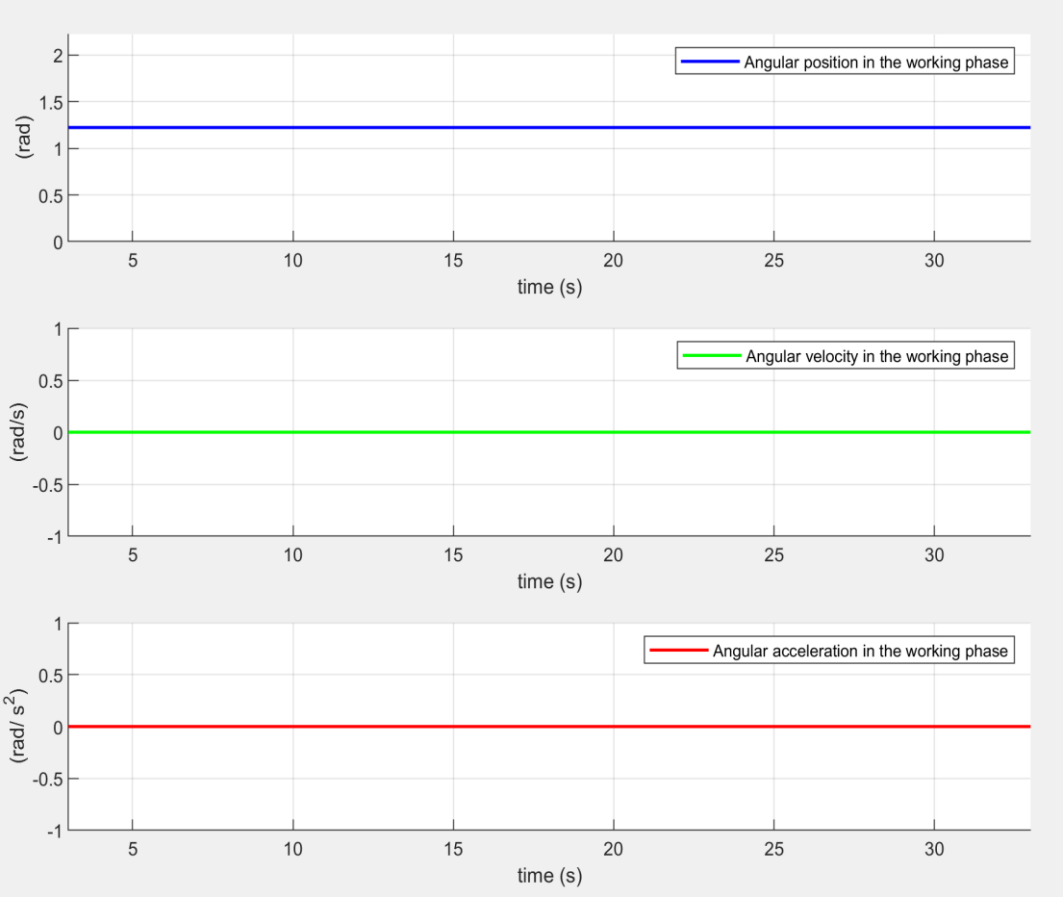



*Figure 2.3: Input quantities trend in working phase*

**Lifting phase**

As far as the third phase is concerned, the law of motion is a cycloidal one with a reverse trend to the first phase. The angular position, angular velocity and angular acceleration will be represented respectively by the following expressions:

$$\theta_a = 12 - \left(\tau - \frac{1}{2\pi}\sin(2\pi\tau)\right) * h \tag{2.7}$$

$$\dot{\theta_a} = \omega = -(1 - \cos(2\pi\tau)) * \frac{h}{t} \tag{2.8}$$

$$\ddot{\theta_a} = -(2\pi * \sin(2\pi\tau)) * \frac{h}{t^2} \tag{2.9}$$

The figure 2.4 shows the trends, obtained with MATLAB of the three quantities treated so far on a time horizon ranging from 33s to 36s, i.e. the expected duration for this phase.

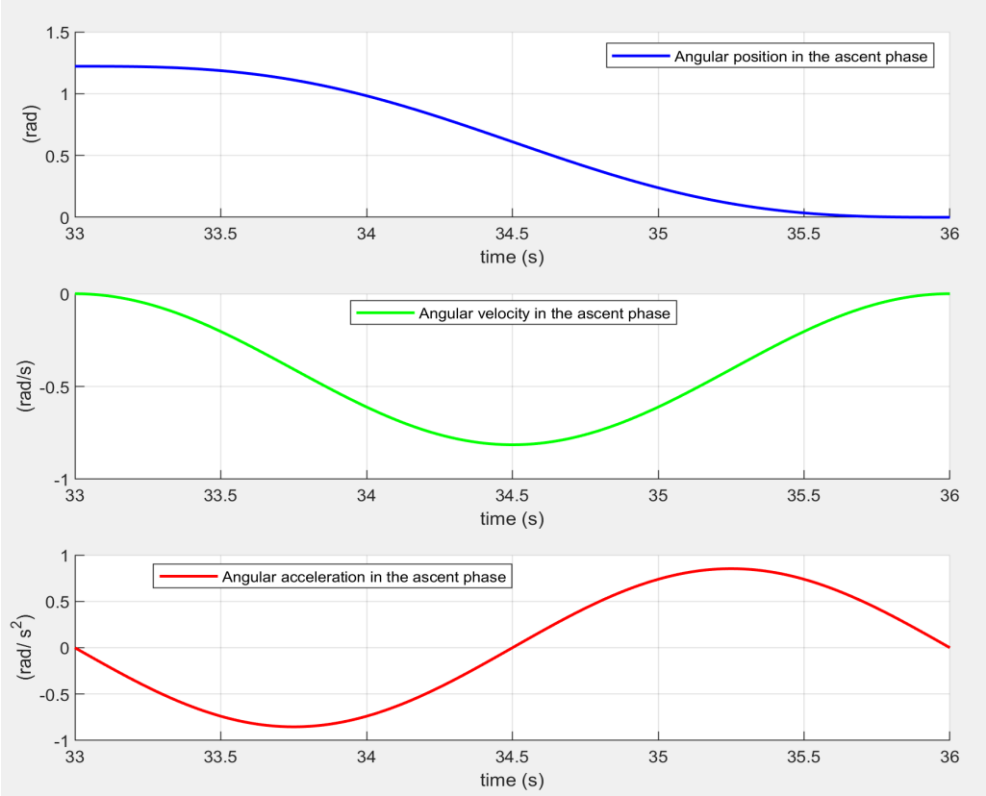



*Figure 2.4: Input quantities trend in lifting phase*

**Resting phase**

As for the last phase, the law of motion is constant and equal to zero. The three quantities will be represented respectively by the following expressions:

$$\theta_r = \dot{\theta}_r = \ddot{\theta}_r = 0 \tag{2.10}$$

The figure 2.5 shows the trends, obtained with MATLAB of the three sizes treated so far on an interval ranging from 36s to 60s, that is the expected duration for this phase.

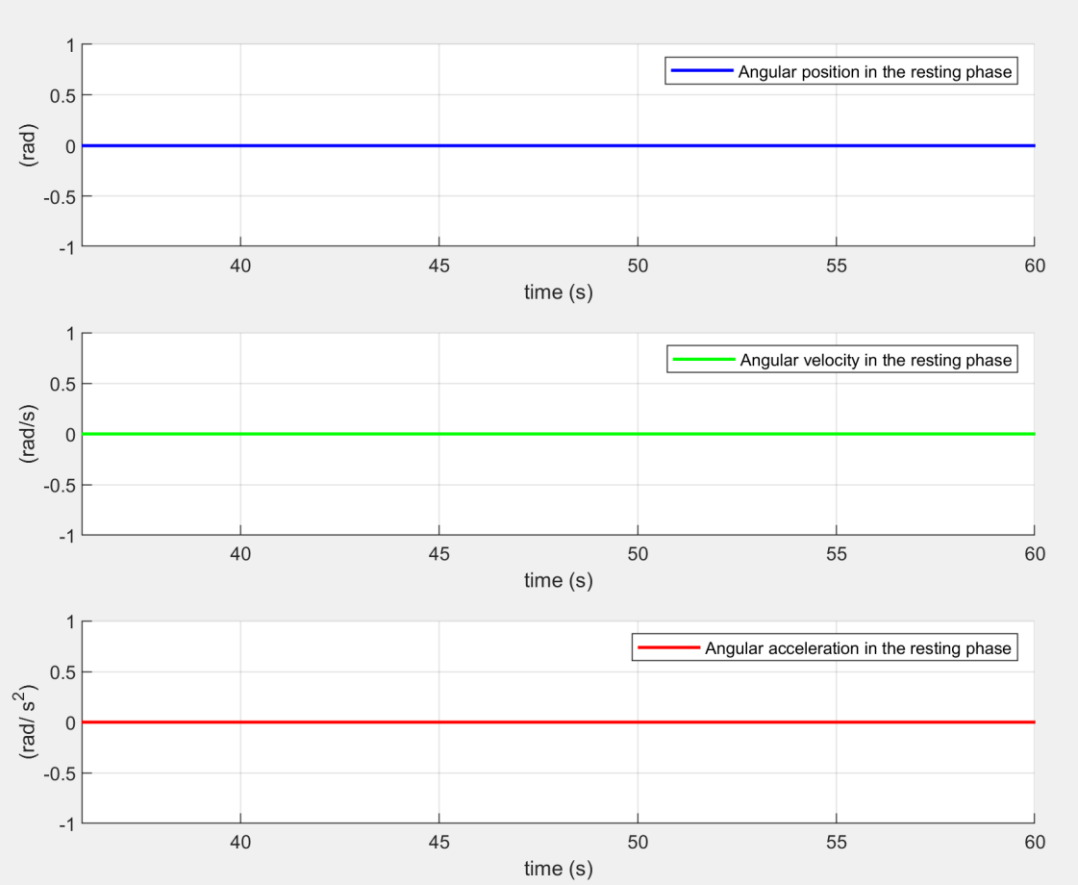



*Figure 2.5: Input quantities trend in resting phase*

**Complete work cycle**

To summarize everything, the complete work cycle representing the trend of the three quantities is shown in the figure 2.6.

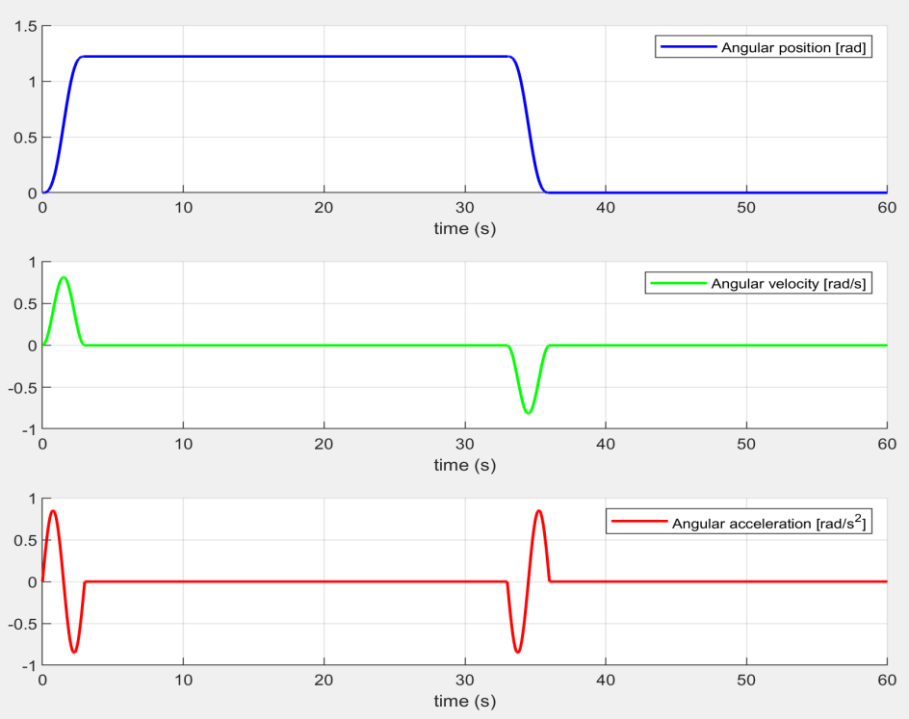



*Figure 2.6: Input quantities trend*

## 2.2 Human body

The calculation of the muscle torque produced by the human body was implemented through Simulink and then the result obtained was multiplied by 0.3 to obtain the reference signal for the motor torque generated. The specifications of the human body are compliant with the standard ISO 7250-2. The diagram is shown in the figure (Fig. 2.7) below:

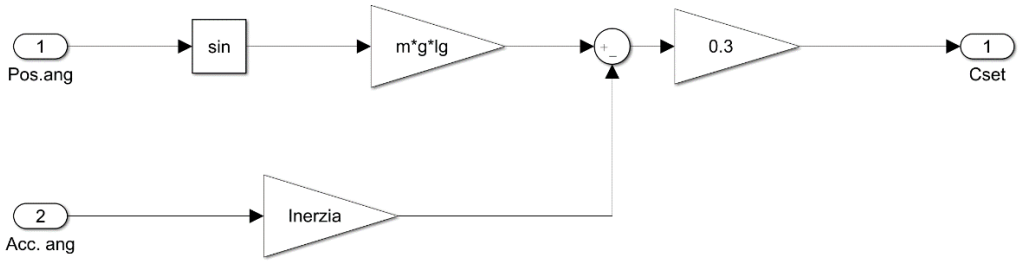



*Figure 2.7: Simulink of human body*

As said, the output is the quantity $C_{SET}$ which is described by the following equation:

$$C_{SET} = mgL_g \sin\theta \; - \; (I \; + \; \mathrm{mL_g}^2) * \ddot{\theta} \qquad (2.11)$$

The trend is shown in the following figure:

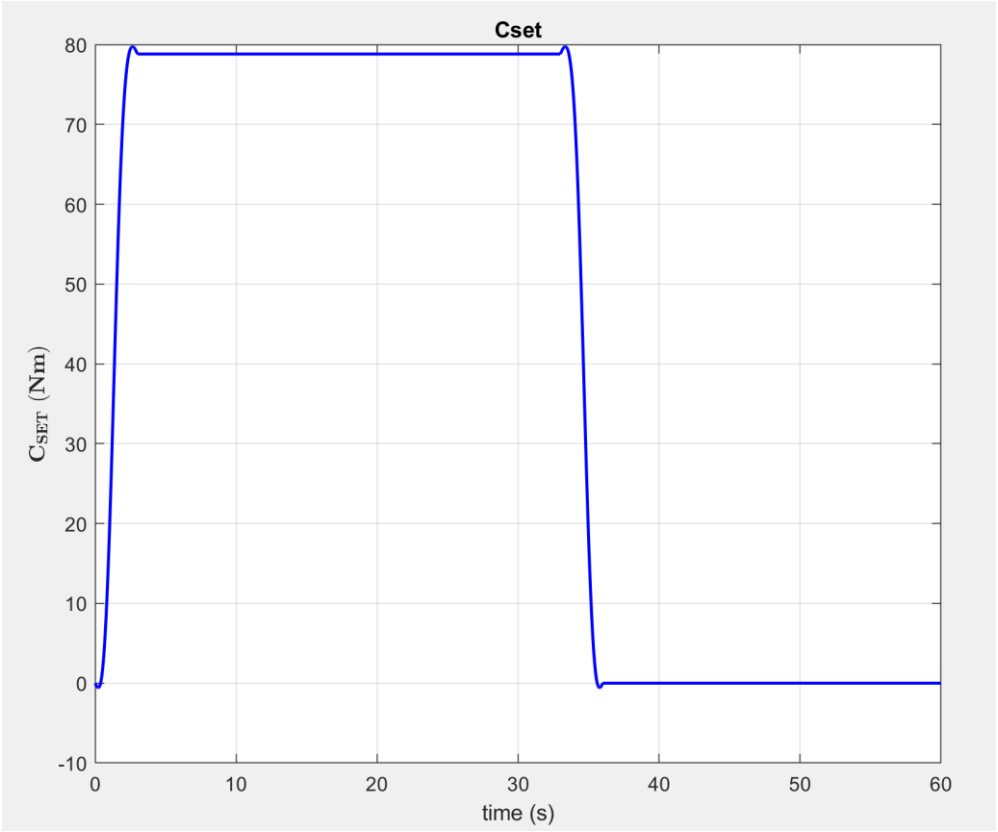



*Figure 2.8: $C_{SET}$ trend*

## 2.3 Controller

The automatic control of a given dynamic system (e.g. a motor, an industrial plant…) aims to modify the behaviour of the system to be controlled (i.e. its "outputs") by manipulating appropriate input quantities. It may be required that the output remains constant at a set value when the input changes (simple control or regulation) or faithfully follows the dynamics of the input itself (slave or control system) without amplifications and delays. The automatic control of a system is possible only if the system itself is reachable and observable. Each system can have one or more inputs and one or more outputs. With the term SISO we mean a system with single input and single output, while with the term MIMO we mean a system with multiple inputs and multiple outputs (in our work we are deal with SISO system). Each variation of the input variables is followed by a certain response of the system and the most common input variables variations are the Dirac pulse, the step, the ramp and the sinusoid.

The input variables differ in:

- manipulable variables: they have the characteristic of always being measurable
- disturbances: they may also be non-measurable, and their presence is undesirable from a control point of view.

The output variables include:

- performance variables: they are the controlled variables, not to be confused with control variables, and can be measured directly or indirectly
- Intermediate variables: are physical measurable variables that can be used for the indirect measurement of performance variables.

Direct measurement of the variables to be controlled is called primary measurement, while indirect measurement of the variables to be controlled is called secondary measurement. Examples of secondary measurement are cascading control, adaptive control and inferential control.

The control scheme can be of two types:

- Open loop
- Closed loop

As far as the first one is concerned, i.e. open loop control (either forward or predictive or feedforward), it is based on an input processing performed without knowing the output value of the controlled system, since some properties of the system to be controlled are known. In this case it is fundamental to have a good mathematical model that describes with good precision the behaviour of the system. The more the mathematical model on which the action of the feedforward control is based is exact, the more reliable this type of control is. The second one, i.e. closed-loop control (or backwards or feedback), is more complex but much more flexible than the first because it can make a system stable that is not stable at all. In this case the control loop brings back to the input of the system you want to control or make stable an output function that must be algebraically added to the signal already present at the input. You can distinguish the control in:

- positive feedback: the feedback signal is added to the reference signal, and the sum is sent to the system input.
- negative feedback: the feedback signal is subtracted from the reference signal, so that the so-called tracking error signal (the difference between output and reference) is input to the system.

In general, positive feedback leads to unstable systems, while negative feedback opens the way to very effective control strategies for achieving system stability and improving system performance: speed in reaching the

desired output value, zero error in the case of constant input or input with linear variations over time.

At the beginning to get an idea of how the mathematical model behaves (Fig. 2.1) we chose to use a standard regulator, the PID because is the most widely used control method in industrial applications. In the following chapters the structure and operation of the various controllers will be analysed in detail, comparing performance. This is to be able to choose the most suitable controller for our system.

## 2.4 Proportional valve

The pneumatic motor responsible for generating the torque to support the muscle fatigue is controlled by a proportional valve. This type of interface is characterized by the fact that the output signal (pressure) is proportional to the electrical input signal (generally a voltage, but it can also be a current). In order to carry out the Simulink scheme shown in the figure 2.9, it was necessary to have the voltage-pressure characteristic of the valve (Fig. 2.10).

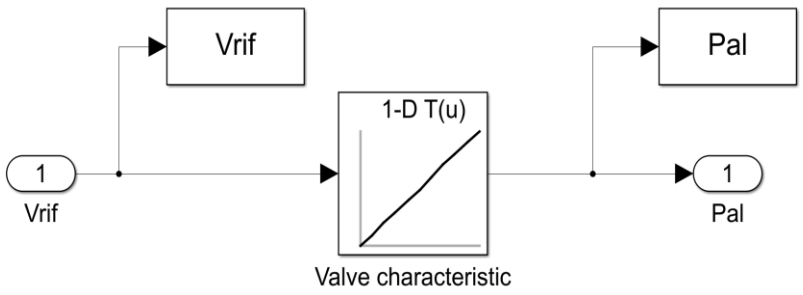



*Figure 2.9: Simulink of proportional valve*

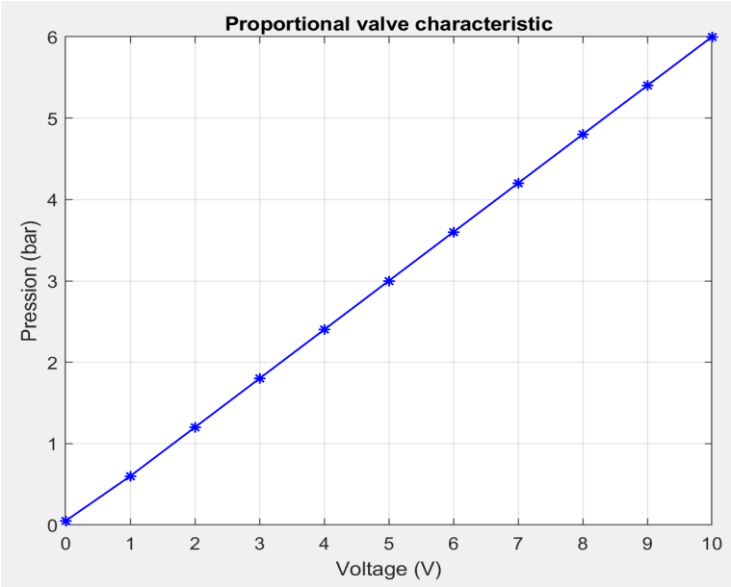



*Figure 2.10: Valve characteristic*

## 2.5 Air motor

The actuator chosen for the exoskeleton is, as already mentioned, pneumatic. Although most rotary motors are electric, in this case this choice has been made for the advantages it offers, such as: high power-to-weight ratio, indifference to overload and stall conditions, it can work in dirty and explosive atmospheres, high rotation speed, easy energy storage and are easily reversible [19]. The control system works in torque control, it is reported in detail the trend of the angular torque-speed characteristic (Fig. 2.11a) and the trend of the correction factor for torque only (Fig. 2.11b).

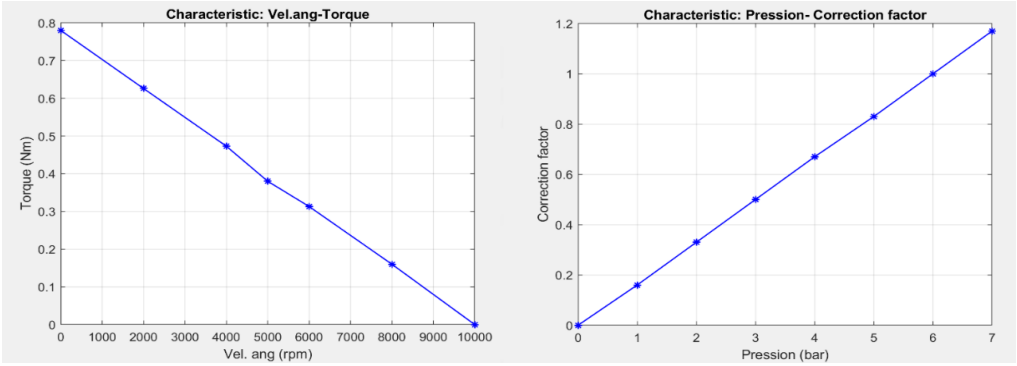



*Figure 2.11: (a)Torque-Vel.ang characteristic, (b) Pression-Correction factor characteristic*

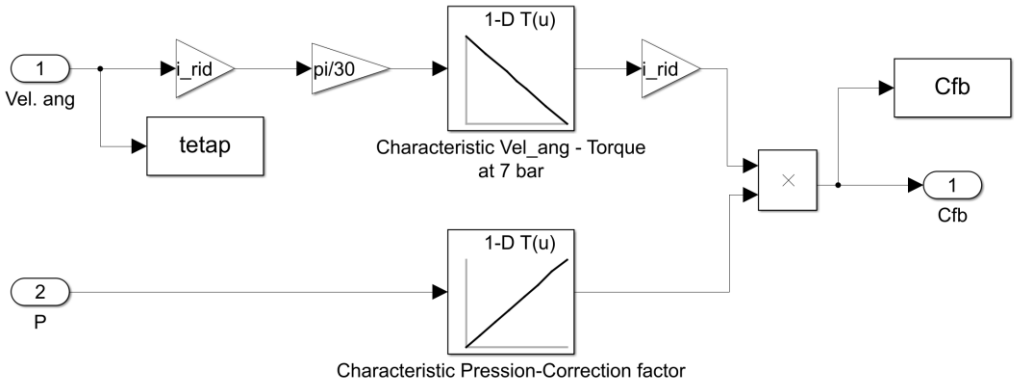



*Figure 2.12: Simulink model of air motor*

## 2.6 Results

After defining all the blocks in the diagram (Fig. 2.1), the result is analysed (i.e. the comparison between the feedback torque and the set torque). The two curves, if everything went well, should overrun unless an error that must be almost negligible. The following graphs show the results:

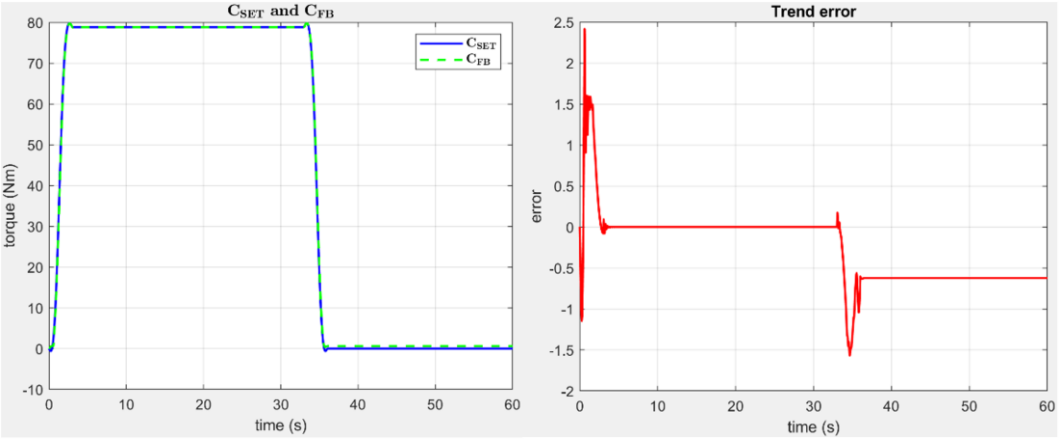



*Figure 2.13: Final results*

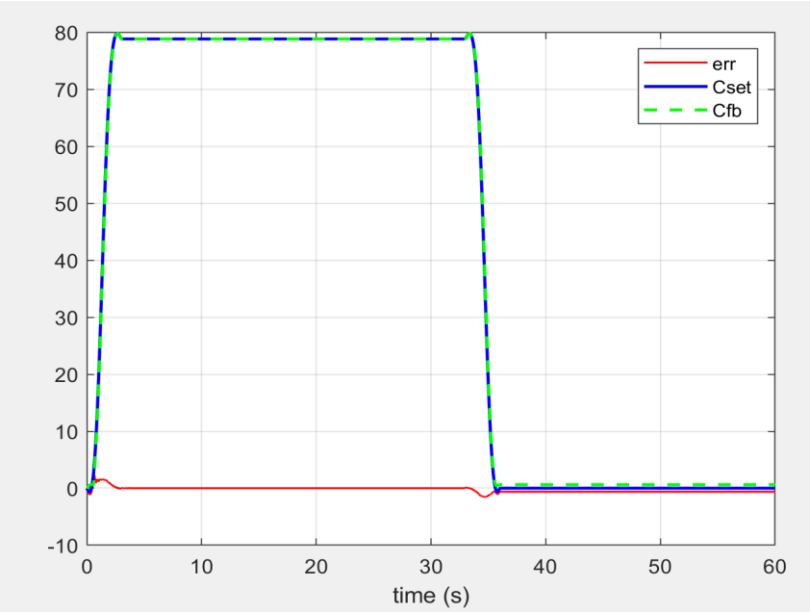



*Figure 2.14: Final comparison*

From the figure 2.15 it is possible to see that the error is negligible respect to the torque and this is an acceptable result.

# 3 Control scheme

As already described in section 2.3 there are different types of controllers that can be used for our exoskeleton. Below we will illustrate the chosen types describing them from a theoretical and implementation point of view, then we report a tuning of the fundamental parameters to see the performance trend.

## 3.1 PID controller

A proportional-integral-derivative controller (PID controller or three-term controller) is a control loop feedback mechanism requiring continuously modulated control. A PID controller continuously calculates an error value $e(t)$ as the difference between a desired setpoint and a measured process variable and applies a correction based on proportional, integral, and derivative terms (denoted P, I, and D respectively), hence the name. The distinguishing feature of the PID controller is the ability to use the three

control terms of proportional, integral and derivative influence on the controller output to apply accurate and optimal control. The block diagram shown in the following figure shows the principles of how these terms are generated and applied.

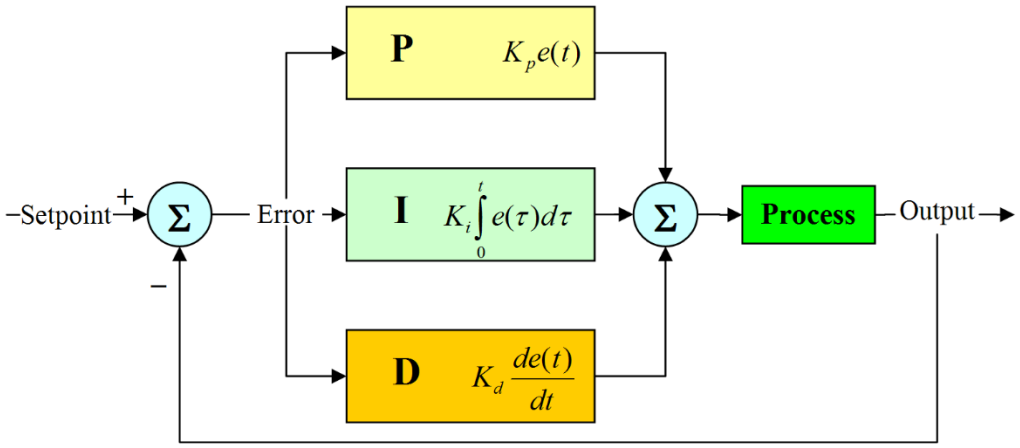



*Figure 3.1: PID controller structure*

In general, we can apply three different control action that can be combined: a proportional action P, an integral action I and a derivative action D.

- Proportional P: proportional action is linked with the quickness of the system. If we increase the weight of the P action the response is faster and the steady state error is decreased, but at the same time increase the instability of the system because an increase of the weight of P provide higher oscillations of the system;
- Integral I: the principal advantage of I action is that the steady state error belongs to zero, but at the same time, since it is by itself unstable an higher weight of I action increase the instability of the system, that must be compensated with an higher weight of P action for instance;
- Derivative D: it increments the phase margin of the system and consequently also the damping. As a consequence, the system increase its quickness and it is possible to augment the weight of P action.

### 3.1.1 Mathematical model

The equations that govern the system of a PID controller are:

$$C(s) = K_p + \frac{K_i}{s} + K_d = K_p * (1 + \frac{1}{T_i * s} + T_d * s) \qquad (3.1)$$

$K_p, K_i, K_d$ are called proportional, integral and derivative gains respectively, while $T_i \; e \; T_d$ are the integral and derivative time constants, they are defined according to the following relation:

$$T_d = {K_d}/{K_p} \qquad T_i = {K_p}/{K_i} \qquad (3.2)$$

The value of the derivative time constant $T_d$ is the time required for the proportional action to equal the derivative action in case the error increases linearly over time from zero. Instead, the value of the integral time constant $T_i$ is the time required for the integral action to equal the proportional action if the error is constant over time. In the complete PID controller, the proportional gain $K_p$ is mainly chosen to decrease the rise time, the integral gain $K_i$ to eliminate the static error, and the derivative gain $K_d$ to decrease the over elongation.

**Effects of the different PID contributes**

- Proportional: this term produces an output value that is proportional to the current error value. The proportional response can be adjusted by multiplying the error by a constant $K_p$, called the proportional gain constant. The proportional term is given by:

$$u(t) = K_p * e(t) \qquad (3.3)$$

A high proportional gain results in a large change in the output for a given change in the error. If the proportional gain is too high, the system can become unstable. In contrast, a small gain results in a small output response to a large input error, and a less responsive or less sensitive controller. If the proportional gain is too low, the control action may be too small when responding to system disturbances. Because a non-zero error is required to drive it, a proportional controller generally operates with a so-called steady- state error (SSE) which is proportional to the process gain and inversely proportional to proportional gain.

- The contribution from the integral term is proportional to both the magnitude and the duration of the error. The integral in a PID controller is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously. The accumulated error is then multiplied by the integral gain $K_i$ and added to the controller output. The integral term is given by:

$$u(t) = K_i \int_0^t e(\tau)\, \partial\tau \tag{3.4}$$

The integral term accelerates the movement of the process towards setpoint and eliminates the residual steady-state error that occurs with a pure proportional controller. However, since the integral term responds to accumulated errors from the past, it can cause the present value to overshoot the setpoint value

- The derivative of the process error is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain $K_d$. The magnitude of the contribution of the derivative term to the overall control action is termed the derivative gain, $K_d$. The derivative term is given by:

$$u(t) = K_d \frac{de(t)}{dt} \tag{3.5}$$

Derivative action predicts system behaviour and thus improves settling time and stability of the system. An ideal derivative is not causal, so that implementations of PID controllers include an additional low-pass filtering for the derivative term to limit the high-frequency gain and noise.

There is only a problem: if you try to implement the PID transfer function we cannot do it since it is not proper! In order to build a constructible "block" we must do some improvements. There is a net called LEAD compensator which can well approximates the derivate behaviour of the controller. Starting from the transfer function of a simple PD controller in the Laplace domain given by $G_{PD} = K_p + K_d * s$ we can obtain the equivalent approximate transfer function of the lead compensator as follows:

$$\begin{aligned} G_{PD} \approx\ G_{LEAD} &= K_p + K_d \frac{P\ s}{s + P} \ -->\ \ G_{LEAD} \\ &= K_c \frac{s\ + Z}{s + P} \end{aligned} \tag{3.6}$$

Where:

- $K_c = K_p + K_d P$
- $Z = \frac{K_p P}{K_p + K_d P}$

### 3.1.2 Parameters tuning

Tuning a control loop is the adjustment of its control parameters to the optimum values for the desired control response. Stability (no unbounded oscillation) is a basic requirement, but beyond that, different systems have different behaviour, different applications have different requirements, and requirements may conflict with one another. Designing and tuning a PID controller appears to be conceptually intuitive, but can be hard in practice,

if multiple (and often conflicting) objectives such as short transient and high stability are to be achieved. We have performed the simplest technique to tune the PID controller parameters, which is called manual tuning and the relative considerations are shown in the next table:

| **Parameters** | $T_r$ | $\hat{s}$ | $T_s$ | **SSE** | **Stability** |
|---|---|---|---|---|---|
| $K_p$ | decrease | increase | small change | decrease | degrade |
| $K_i$ | decrease | increase | increase | eliminate | degrade |
| $K_d$ | minor change | decrease | decrease | no effects | improve if $K_d$ small |

In the case analysed in the previous work, a PI controller is made and a manual calibration of the parameters has been carried out, which foresees that we go by trial and error and choose the proportional, integral and derivative gain values in order to obtain a satisfactory control of the process, making sure that the system responds in the shortest possible time and without having an error at full capacity.

For the calibration of the controller the values found manually for the stability of the system and to cancel the error at full speed are:

$$K_p = 0.8;\ \ K_i = 5;\ K_d = 0; \tag{3.7}$$

Thus, obtaining a PI controller described by the following equation:

$$G(s) = K_p + \frac{K_i}{s} = 0.8 + \frac{5}{s} \tag{3.8}$$

These parameter choices lead to the following results:

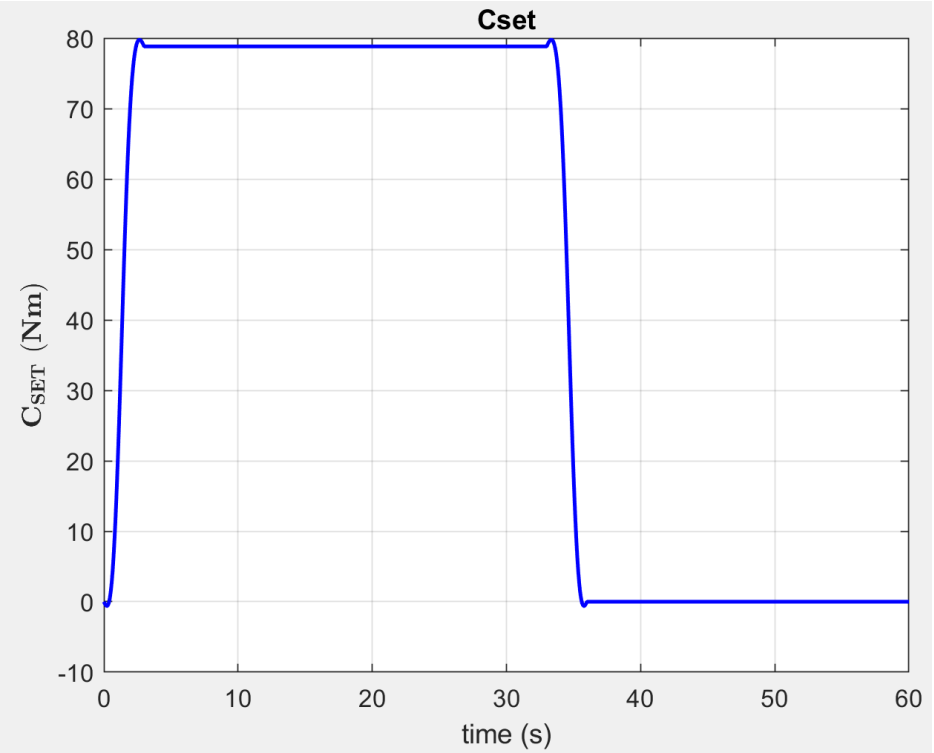



*Figure 3.2: Reference torque*

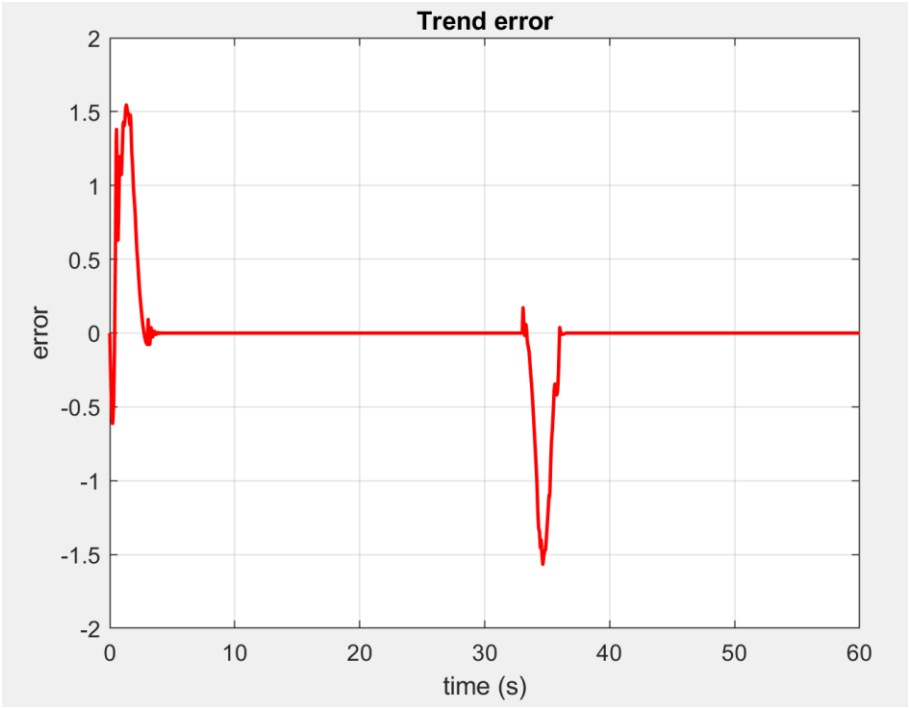



*Figure 3.3: Error trend first PI controller*

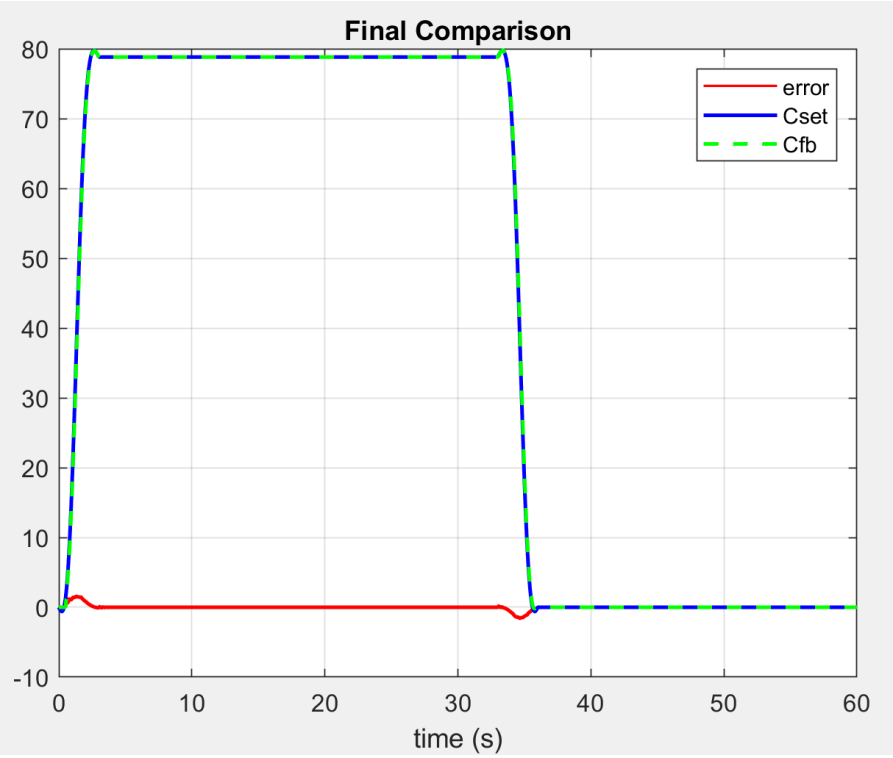



*Figure 3.4: Final comparison first PI controller*

**Time requirements:** $\hat{s} = 50.6\%$ $T_r = 0.0489s$ $T_s = 0.898s$

The error is negligible with respect to the driving torque. It is characterized by a very high overshoot and consequently a trend influenced by many oscillations during the descent and ascent phase. During the work phase and the rest phase the error, correctly, is 0.

Through the MATLAB PID tuner tool, an automatic tuning of the parameters has been performed to optimize the $K_p$ and $K_i$ values.

The following values have been obtained:

$$K_p = 0.124;\ K_i = 0.952;\ K_d = 0; \tag{3.9}$$

Thus, obtaining a PI controller described by the following equation:

$$G(s) = K_p + \frac{K_i}{s} = 0.124 + \frac{0.952}{s} \tag{3.10}$$

These parameter choices lead to the following results:

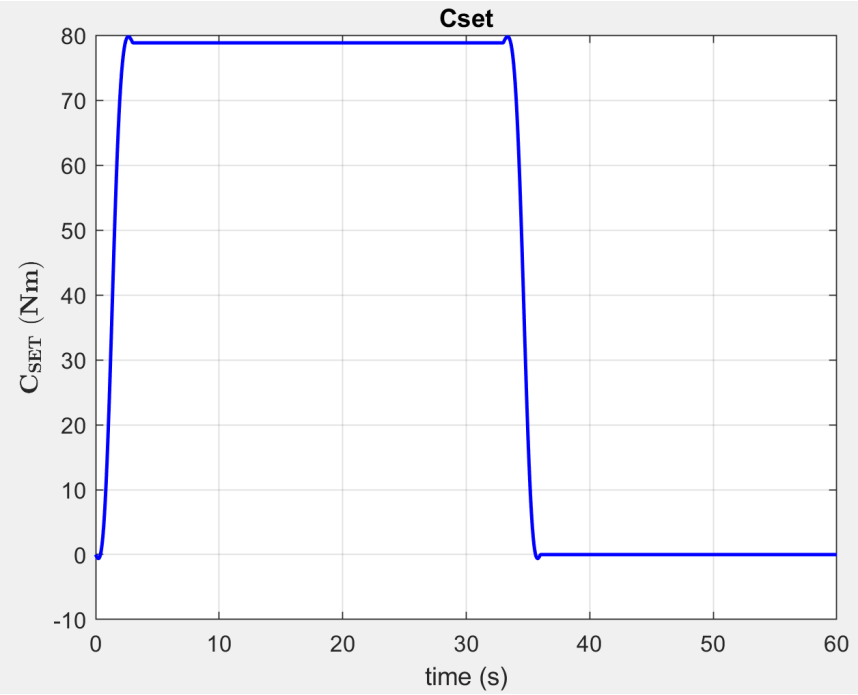



*Figure 3.5: Reference torque*

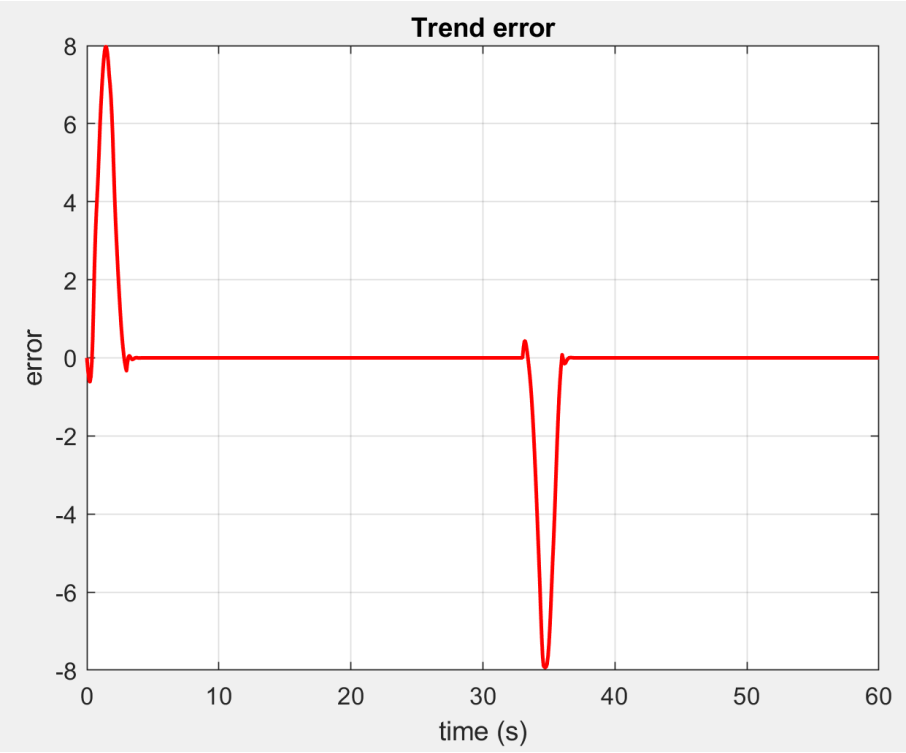



*Figure 3.6: Error trend second PI controller*

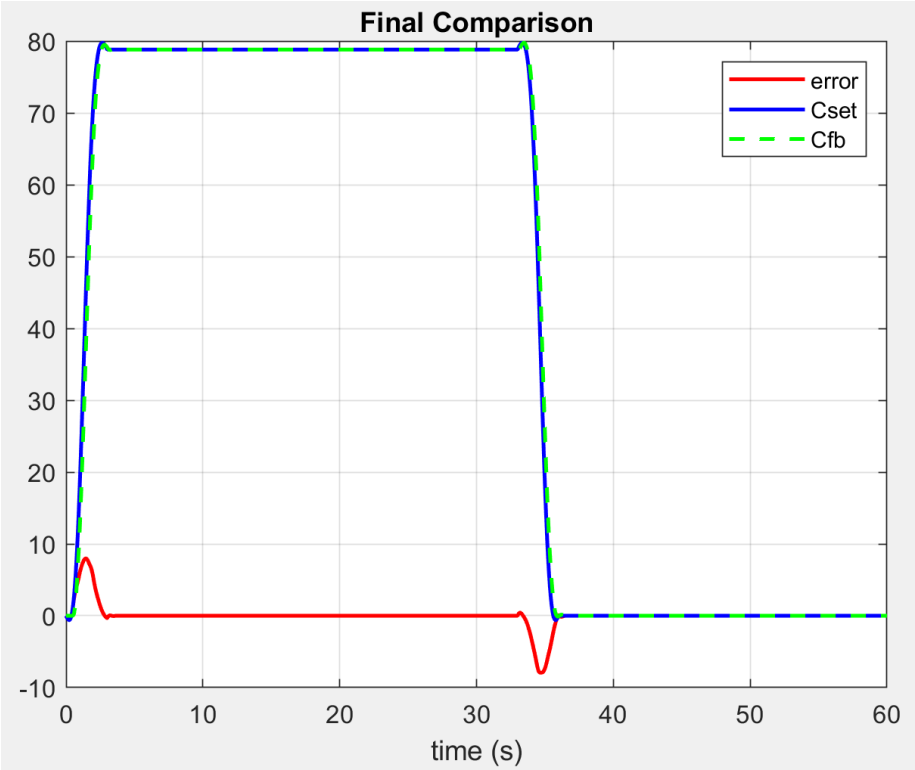



*Figure 3.7: Final comparison second PI controller*

**Time requirements:** $\hat{s} = 6.32\%$ $T_r = 0.165s$ $T_s = 0.727s$

With these parameters the oscillations disappear but you will have a slightly larger error that can always be neglected with respect to the driving torque. In addition, you get a slower and smoother system for the wearer who will feel the most comfortable exoskeleton.

Always using the PID tuner tool, a PID controller has been realized, i.e. with the presence of the derivative constant.

The following values have been obtained:

$$K_p = 0.308 \quad K_i = 2.137 \quad K_d = 0.011 \quad N = 1547.85 \quad (3.11)$$

Thus, obtaining a PI controller described by the following equation:

$$G(s) = K_p + \frac{K_i}{s} + K_d \frac{N}{1 + N * \frac{1}{s}} \quad (3.12)$$

These parameter choices lead to the following results:

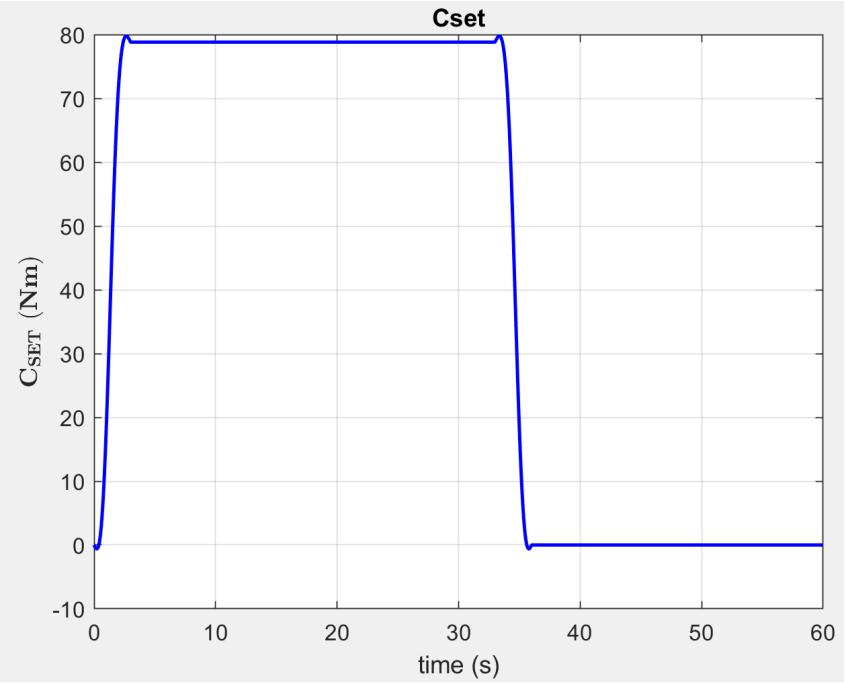



*Figure 3.8: Reference torque*

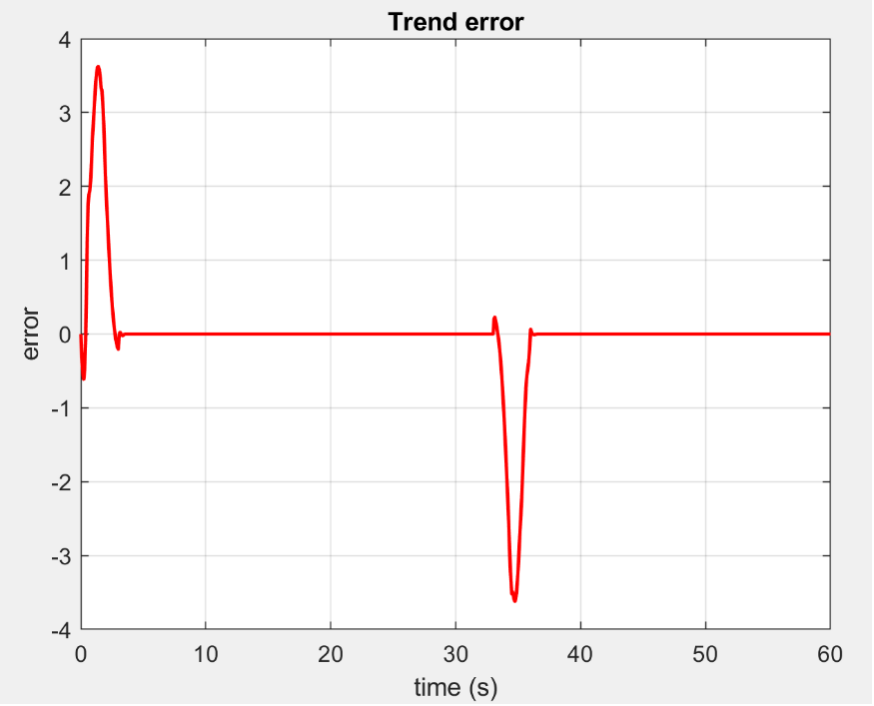



*Figure 3.9: Error trend PID controller*

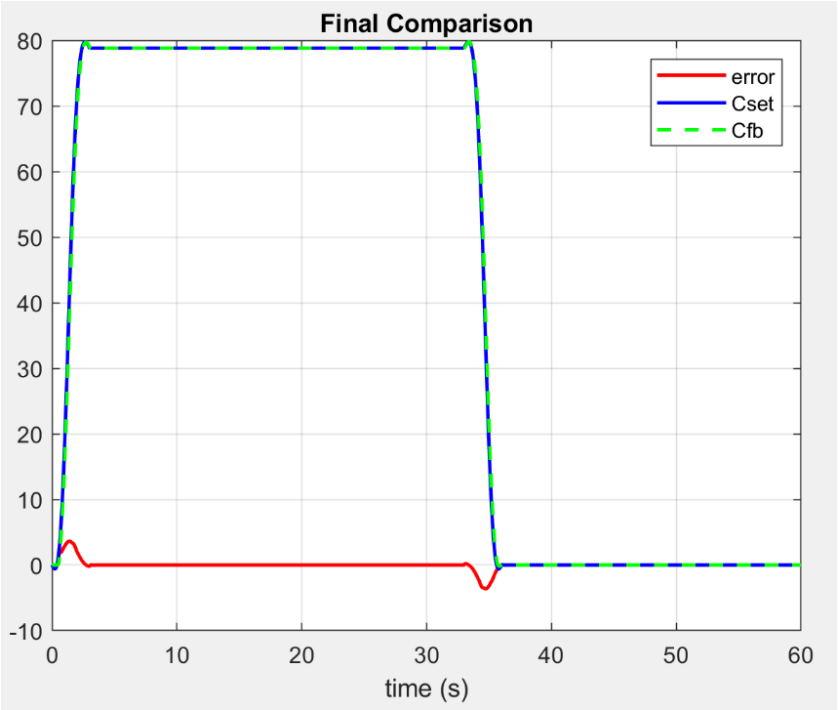



*Figure 3.10: Final comparison PID controller*

**Time requirements:** $\hat{s} = 6.59\%$ $T_r = 0.11s$ $T_s = 0.516s$

The time performance is very similar to the PI controller obtained through MATLAB's PID tuner but in this case the maximum peak of the error, in absolute value, is lower. Consequently, if you opt for a PID controller, the choice will fall on the latter.

## 3.2 LQR controller

In the previous section we discussed the design of the PID controller, and we found some acceptable results. Now we will analyse the Linear quadratic (LQ) optimal control that can be used to resolve some of the previous issue, by specifying performance objective function to be optimized. Other optimal control objectives, besides the LQ type, can also be used to resolve issues of trade-offs and extra design freedom.

### 3.2.1 Theory introduction

Considering a dynamical system described by the state space representation:

$$\dot{x}(t) = Ax(t) + Bu(t), \qquad x(t) \in R^n, u(t) \in R^p \tag{3.13}$$

The optimal control problem consists in finding a suitable input $u(t)$ as the solution to the following optimization problem:

$$\min_{u(t),t\in[0,t_f]} J(u) = \min_{u(t),t\in[0,t_f]} x^t(t_f)Su(t_f) + \int_0^{t_f} (x^t(\tau)\, Qx(\tau) + u^t(\tau)Ru(\tau))\, \partial\tau \tag{3.14}$$

$$Q, S \in R^n : Q = Q^t \geq 0, S = S^t \geq 0, R \in R^p : R = R^t > 0$$

This approach is referred to as the finite horizon linear quadratic (LQ) optimal control problem.

In practice this type of approach is limited because the obtained control law:

- Is made up by a time invariant controller
- Can be applied only on a finite time horizon

So, we can extend the previous problem to an infinite interval of time in the following way:

$$\min_{u(t),t\in[0,\infty)} J(u) = \min_{u(t),t\in[0,\infty)} \int_0^{\infty} (x^t(\tau)\, Qx(\tau) + u^t(\tau)Ru(\tau))\, \partial\tau \tag{3.15}$$

$$Q \in R^n : Q = Q^t \geq 0, R \in R^p : R = R^t > 0$$

The main difference between the two approaches is that the term $x^t(t_f)Su(t_f)$ has been removed. The infinite horizon LQ optimal control problem can be formulated as:

$$\min_{u(t),t\in[0,\infty)} J(u) = \min_{u(t),t\in[0,\infty)} \int_0^{\infty} (x^t(\tau)\, Qx(\tau) + u^t(\tau)Ru(\tau))\, \partial\tau \quad \text{s.t. } \dot{x}(t) = Ax(t) + Bu(t) \tag{3.16}$$

The optimal solution $u*$ and the optimal cost J* are denoted as:

$$\mathrm{u}*(\mathrm{t}) = \arg \min_{u(t),t\in[0,\mathrm{t_f}]} \mathrm{J(u)}, \quad \mathrm{J}* = \mathrm{J(u}*(\mathrm{t}))) \tag{3.17}$$

The matrices $Q = Q^t \geq 0$ and $R = R^t > 0$ are the design parameters chosen according to the desired performance trade-off. Given the LTI system described by the state space representation and if the system is reachable the optimal solution $u*(\mathrm{t})$ to the infinite LQ optimal control problem is given by:

$$\mathrm{u}*(\mathrm{t}) = -R^{-1} * B^t P x(t) = -Kx(t), \quad K = R^{-1} * B^t P \in R^{p,n} \tag{3.18}$$

where $P = P^t > 0$ is the solution to the Algebraic Riccati Equation.

### 3.2.2 Mathematical model

The static state feedback control architecture can be represented as:

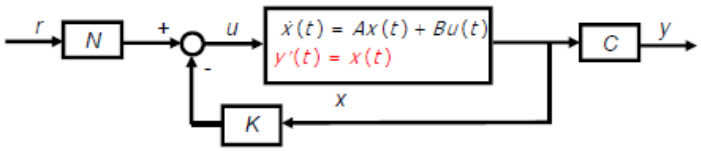



*Figure 3.11: Static state feedback architecture*

The "input" $u = -Kx + Nr$ represents a static state feedback control law. The parameters N can be chosen to make unitary the dc-gain of the controlled system guaranteeing zero tracking error in the presence of a constant reference signal. The plant state equation and transfer function are:

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

$$\xrightarrow{in\ our\ case} \begin{cases} \dot{x}(t) = \begin{bmatrix} -14 & -12.5 \\ 8 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 4 \\ 0 \end{bmatrix} u(t) \\ y(t) = \begin{bmatrix} 0 \\ 3.125 \end{bmatrix} x(t) \end{cases} \quad (3.19)$$

$$G(s) = C(sI - A)^{-1}B \xrightarrow{in\ our\ case} G(s) = \frac{100}{s^2 + 14s + 100} \quad (3.20)$$

The dc-gain of the controlled system is given by:

$$K_p = \lim_{s \to 0} NC[sI - (A - BK)]^{-1} B = NC[I - (A - BK)]^{-1}B = [3.71 \quad 7.06] \quad (3.21)$$

$$N = [C[I - (A - BK)]^{-1}B]^{-1} = 3.26 \quad (3.22)$$

The complete scheme for the LQR regulator adapts to our problem is illustrated in the following picture:

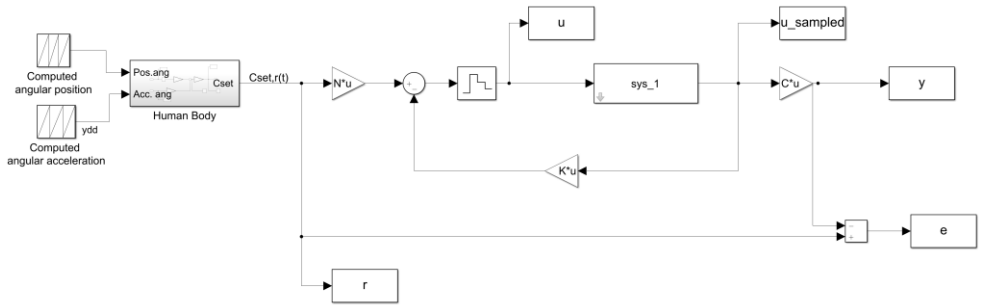



Figure 3.12: LQR Simulink scheme

### 3.2.3 Parameters tuning

Tuning the LQ regulators implies choosing the weight matrices Q and R and this is usually done through a trial and error procedure. In general, the two matrices are chosen as diagonal matrices, so that for a system that have n state and p controls we have $n + p$ parameters to define. The diagonal values are chosen according to the relative importance of each state and control variable. Through MATLAB we perform some tuning of these

parameters in order to suitably choose the correct one. Below are shown the various steps taken. At the beginning we define two vectors with possible values for the terms $Q_{11}$ and $Q_{22}$.

$$Q_{11} = [0.001\ ;\ 0.01\ ;\ 0.1\ ;\ 1\ ;\ 10\ ;\ 100]$$
$$Q_{22} = [1000\ ;\ 100\ ;\ 10\ ;\ 1;\ 0.1\ ;\ 0.01\ ] \qquad (3.23)$$

So, the matrix Q become:

$$Q = \begin{bmatrix} Q_{11} & 0 \\ 0 & Q_{22} \end{bmatrix} \qquad (3.24)$$

Then, we proceed in steps and at each step we fixed the R value and analyse the changes in error and torque. At the end when we have defined the proper value for Q, we start a tuning of R to choose the most suitable couple (Q, R) of values.

**Value of R:** $R = 1$

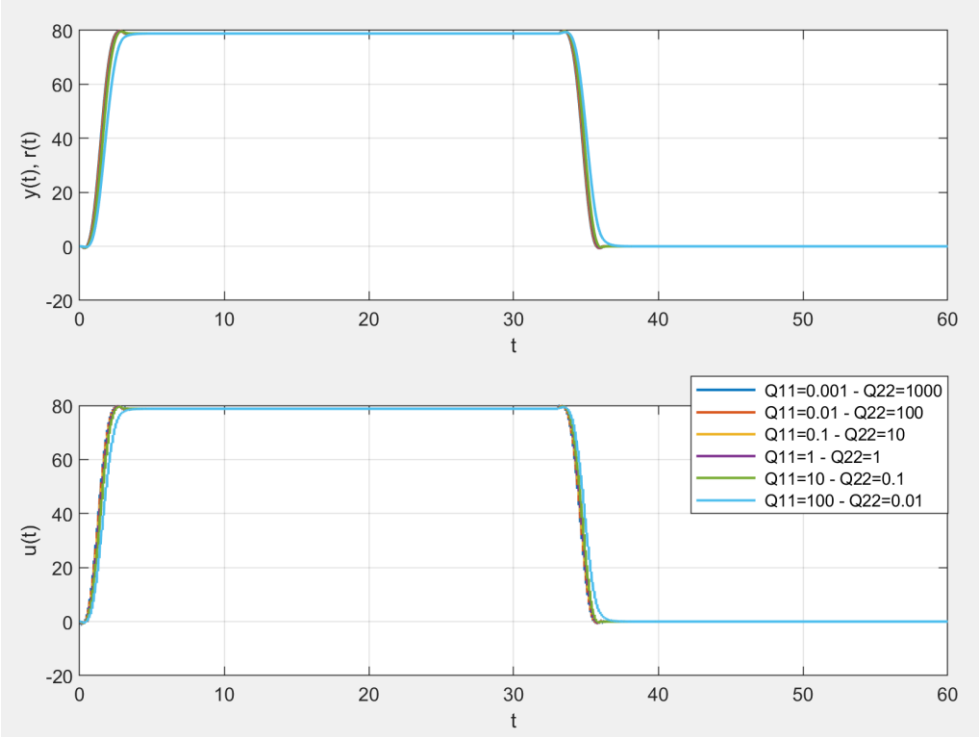



*Figure 3.13:Trend of torque with R=1*

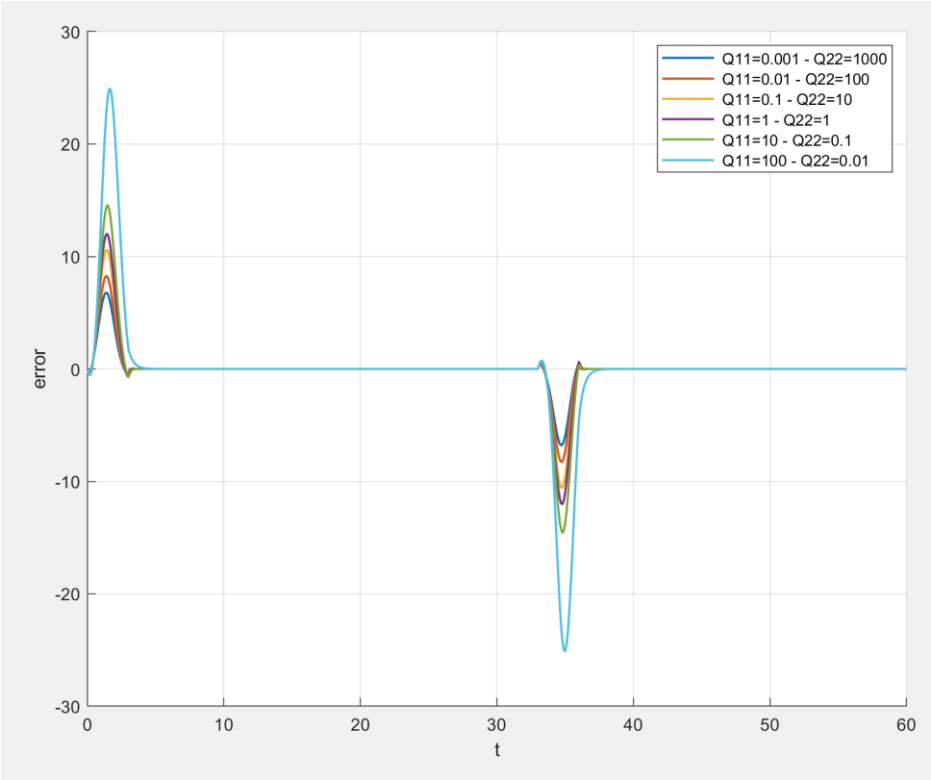



*Figure 3.14:Trend of error with R=1*

From the graphs illustrated above, the choice of parameters $Q_{11}$ and $Q_{22}$ can falls on values $Q_{11} = 0.001$ and $Q_{22} = 1000$ or on values $Q_{11} = 0.01$ and $Q_{22} = 100$. This is reasonable because the matrix Q is relative to the weights of the various quantities and consequently, we must assign a greater weight to the second variable of state that influences the produced torque or the input of our system. Now, as said before, let's do a tuning of R with fixed Q to analyse the behaviour of the error. So, the matrices Q and R become:

$$Q = \begin{bmatrix} 0.001 & 0 \\ 0 & 1000 \end{bmatrix}, \quad R = [0.01\,;\ 0.1\,;\ 1\,;\ 10\,;100] \qquad (3.25)$$

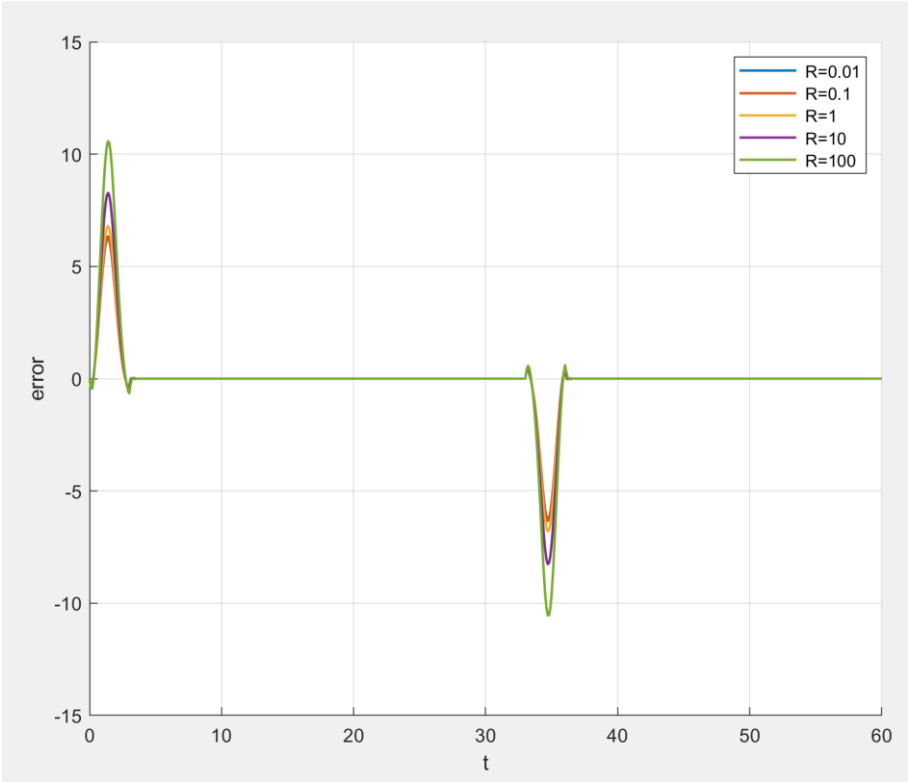



*Figure 3.15: First trend of the error with different values of R*

$$Q = \begin{bmatrix} 0.01 & 0 \\ 0 & 100 \end{bmatrix}, \quad R = [0.01\ ;\ 0.1\ ;\ 1\ ;\ 10\ ;\ 100] \tag{3.26}$$

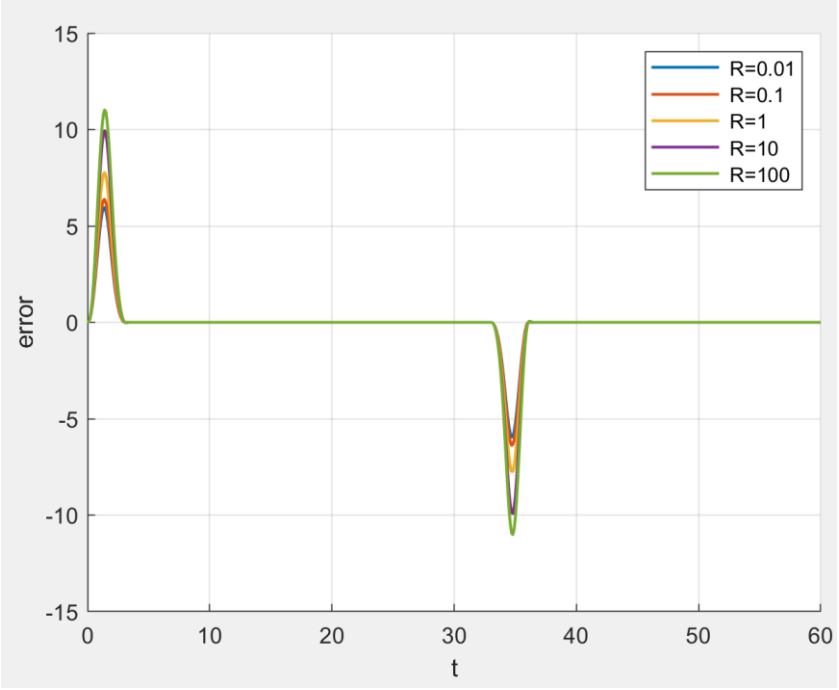



*Figure 3.16: Second trend of the error with different R*

Now in order to choose the proper value I need to perform a simulation of the system with as input a step, in order to verify in which case, the time requirements are better.

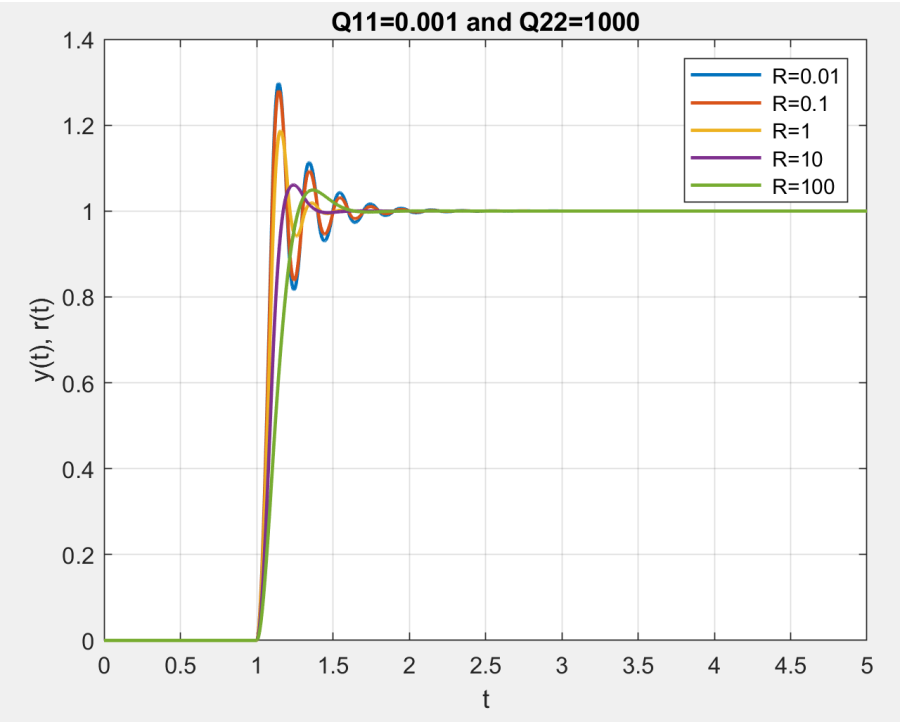



*Figure 3.17: First case step response*

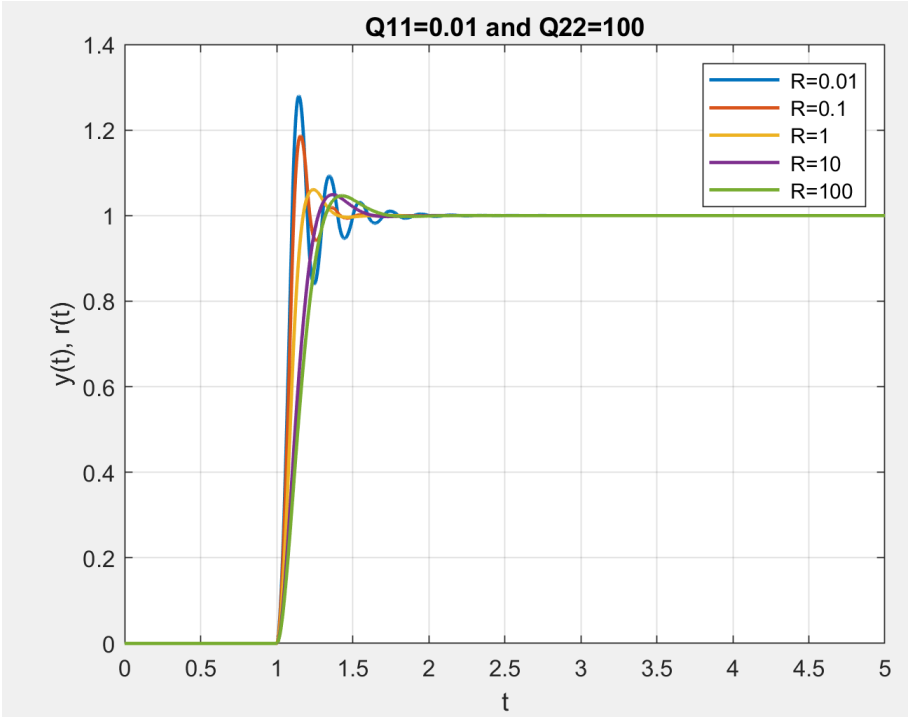



*Figure 3.18: Second case step response*

From the previous analysis it is possible to see how in the second case with the same R-value the time requirements (overshoot, time rise, time settling) are better.

### 3.2.4 Results

The final choice fell on the following values:

$$Q = \begin{bmatrix} 0.01 & 0 \\ 0 & 100 \end{bmatrix}, \quad R = 1 \tag{3.27}$$

As far as the time requirements are concerned, the following performances are obtained:

$$\hat{s} = 6.07\% \qquad T_r = 0.174\ s \qquad T_{s,5\%} = 0.28\ s$$

The figure 3.24 shows the trend of the three sizes (set torque, feedback torque and error).

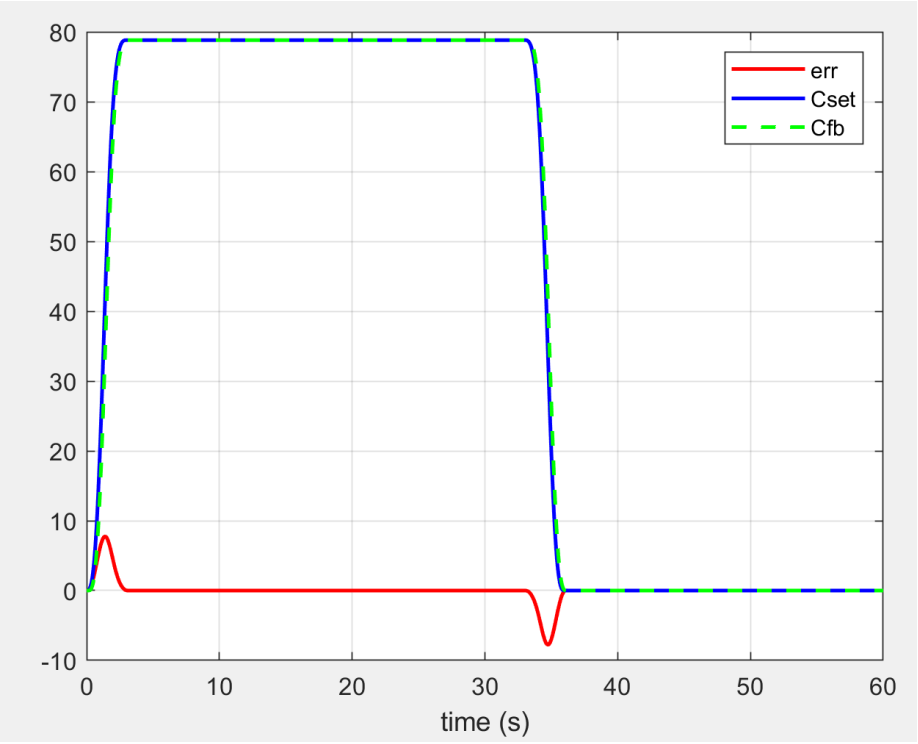



*Figure 3.19: Final comparison LQR regulator*

## 3.3 MPC controller

In the predictive control the requirements are described by a performance index which depends on the prediction of the behaviour of the controlled variable. The control action is founded optimizing on a finite horizon the performance index. We obtain an action control sequence (open loop) and we use only the first one as input of the next step (receding horizon). Every step time the procedure is repeated. The Quadratic programme (QP) that must be solved is:

$$\min_{u(k)} \sum_{i=0}^{H_p-1} x^t(k+i|k)Qx(k+i|k) + u^t(k+i|k)Ru(k+i|k) + x^t(k + H_p|k)Sx(k+H_p|k) \tag{3.28}$$

$$u_{min} \leq u(k+i|k) \leq u_{max}$$

The QP problem can handle also with output constraints:

$$y_{min} \leq y(k+i|k) \leq y_{max}$$

The greater is the prediction horizon $H_p$, the more degrees of freedom are in the optimization. To reduce the number of optimizations variables we predict the system over the prediction horizon $H_p$ only optimizing $H_c \leq H_p$ control action ($H_c$ is the control horizon). Output tracking can be handled adding the following term

$$(y(k+i|k) - r(i))^T Q_y (y(k+i|k) - r(i))$$

in the cost function.

We try to design the best MPC controller for our problem and we do this through MATLAB and Simulink. Below is shown the scheme that we use for our computation.

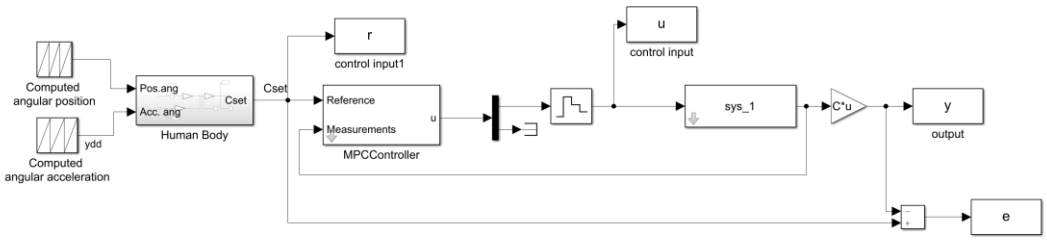



*Figure 3.20: MPC Simulink scheme*

### 3.3.1 Parameters tuning

The parameters that are free in this type of design are the Q and R weight matrices and the value of $Hp$ and $Hc$. The Q matrix is referred to the output tracking error, consequently, is referred to the controlled output (in our case the controlled output is composed only by one state and so the Q matrix is a 1x1 matrix). The R matrix, instead, is referred to the input, so if we need to make the system faster, we put less weight to reduce its effects. Finally, $Hp$ and $Hc$ are the prediction horizon and horizon control, respectively. The initial values chosen are:

$$\mathrm{R} = 1\ ;\ \mathrm{Q} = 0.01\ ;\ \mathrm{H_p} = 5\ ;\ \mathrm{H_c} = 2$$

This choice brings to the following results:

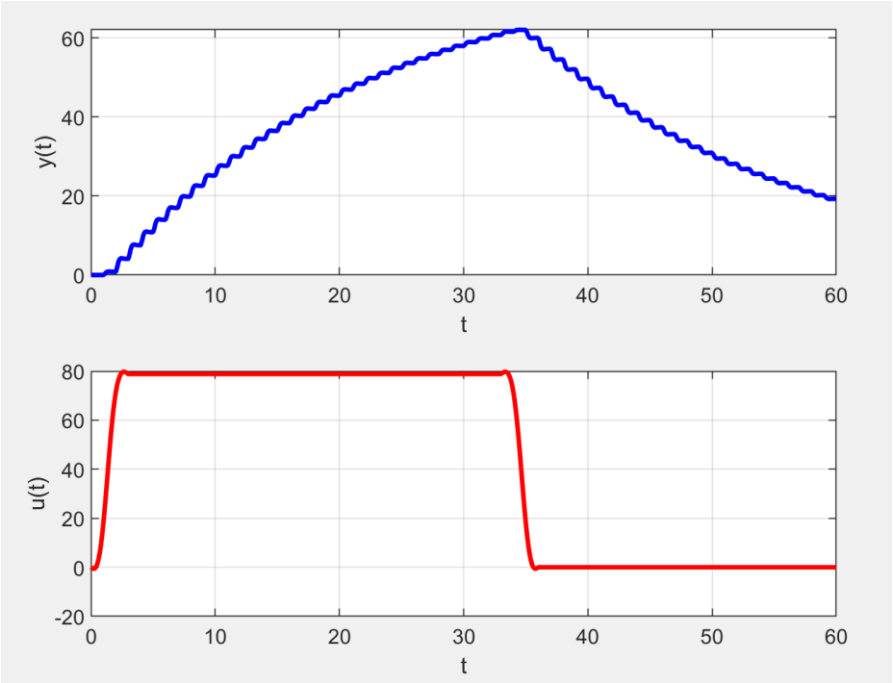



*Figure 3.21: Trend of the torque with Q=0.01*

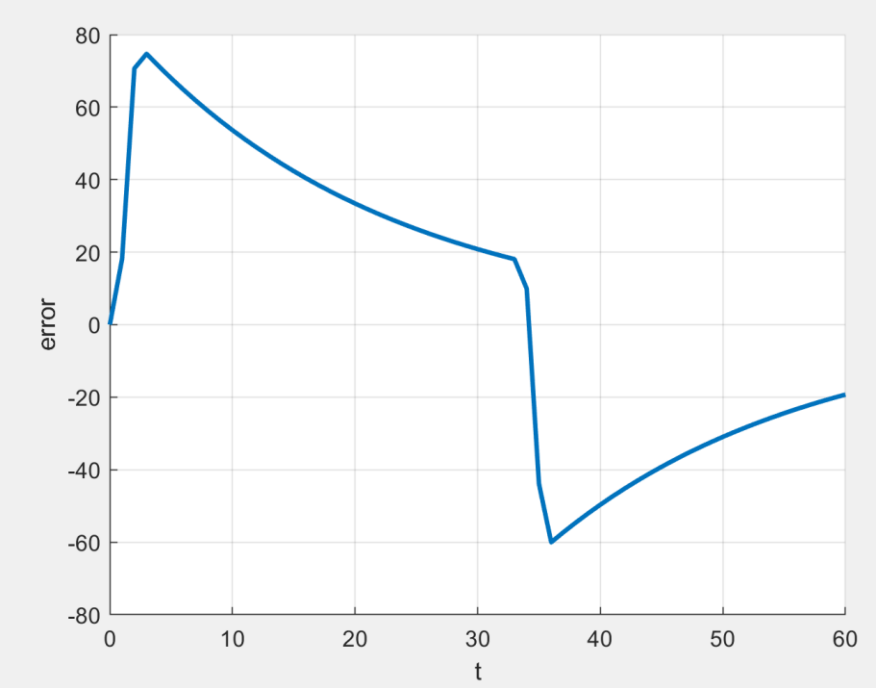



*Figure 3.22: Trend of the error with Q=0.01*

Now, to improve the performances, we tuned the parameters, leaving R=1 fixed and varying the Q value. The following results were obtained:

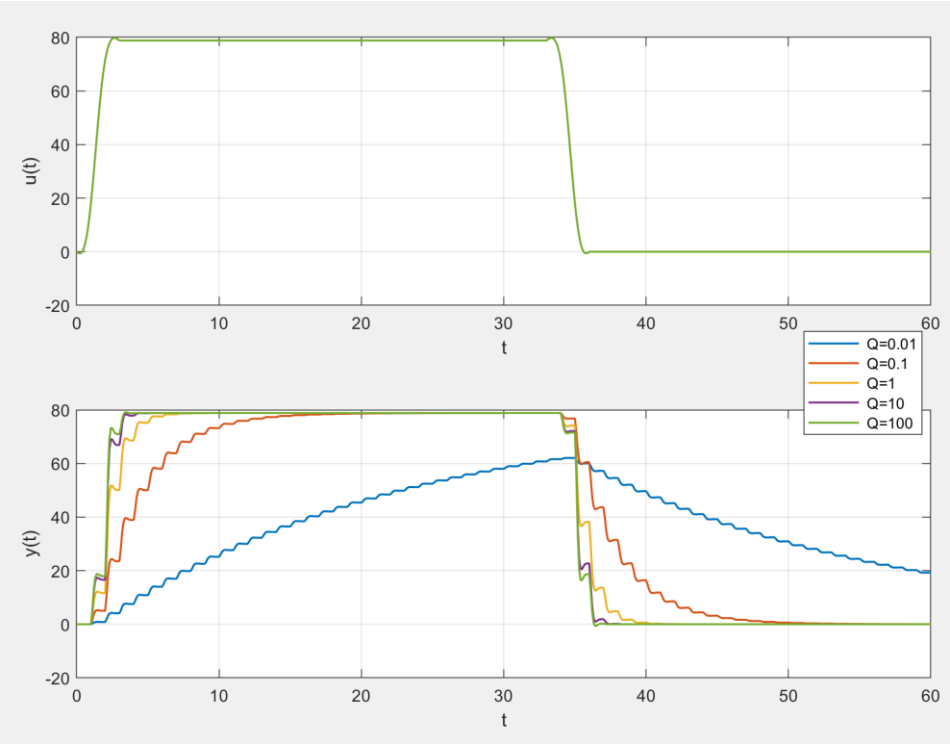



*Figure 3.23: Trend of torque with MPC*

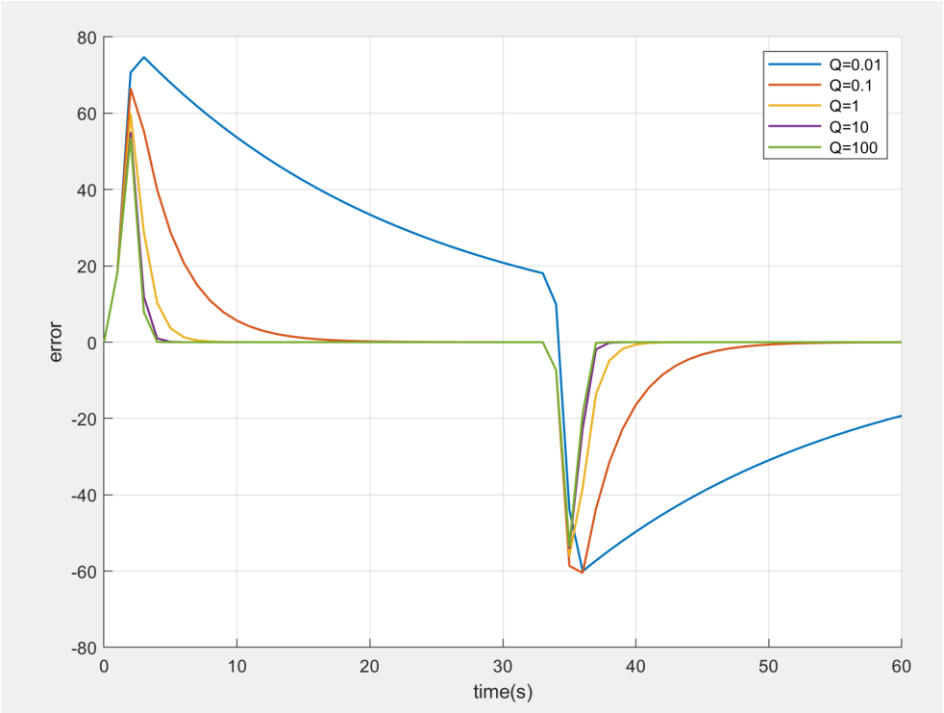



*Figure 3.24: Trend of the error with MPC*

### 3.3.2 Results

In this case, as can be seen from the graphic representations listed above, the MPC control does not allow to obtain the results previously obtained with the PID or the LQR. This is because with MPC it is essential to have a good mathematical model that describes with good accuracy the system behaviour. The more the mathematical model on which the action of the feedforward control is based is exact, the more reliable this type of control is. Despite this, the best results are obtained with the following parameters.

$$Q = 10\,;\; R = 1\,;\mathrm{H_p} = 5\,;\; \mathrm{H_c} = 2$$

In the following figure is possible to see the final comparison between the three fundamental quantities.

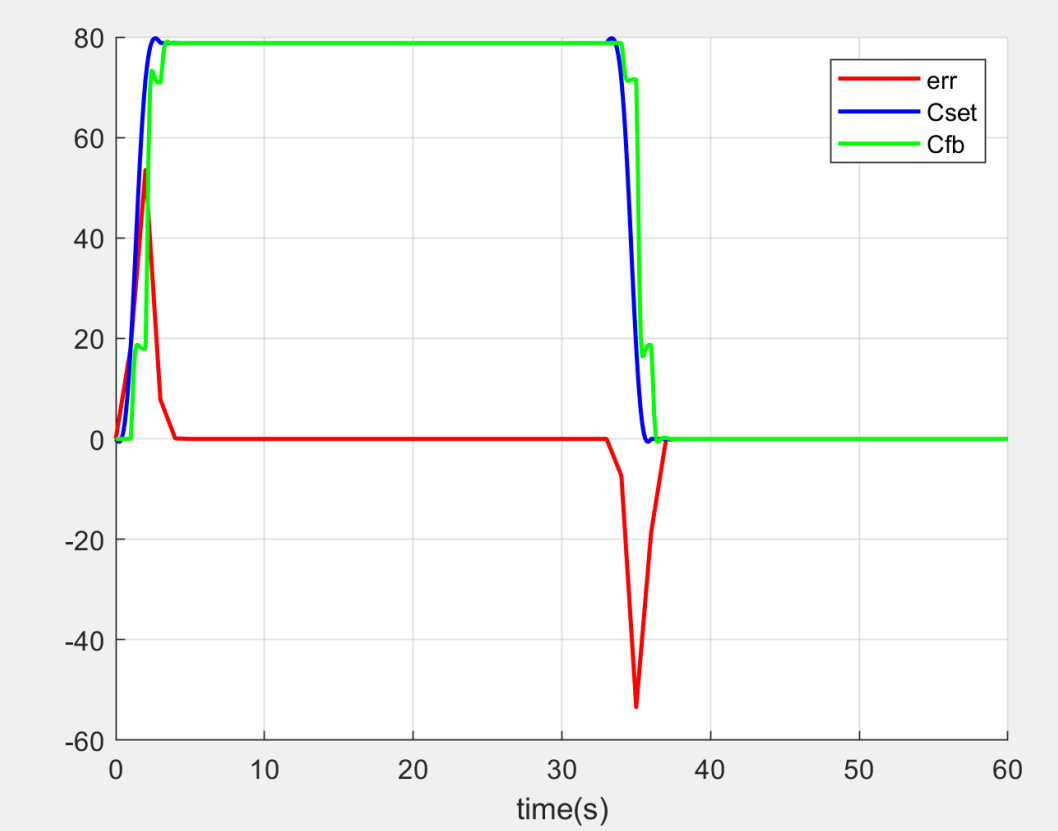



*Figure 3.25: Final comparison with MPC*

## 3.4 Final choice

After having analysed the step response and the behaviour of the three different types of controller it was possible to choose the one with the best performance. The choice fell on the PID controller which allows us to obtain the following results.

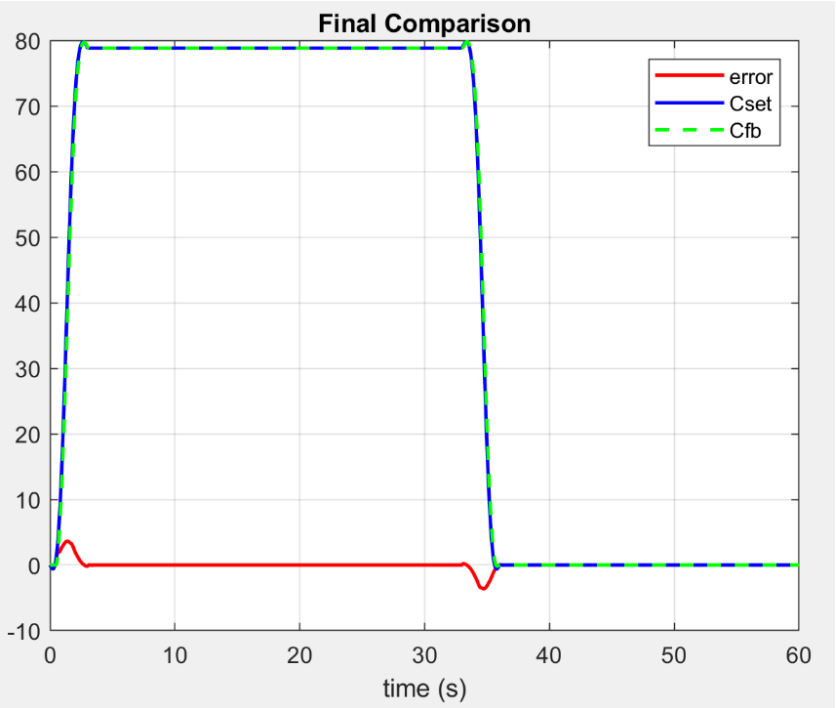



*Figure 3.26: Final controller choice*

**Time requirements:** $\hat{s} = 6.59\%$ $T_r = 0.11s$ $T_s = 0.516s$

# 4 Simscape model

The following chapter describes the model of our exoskeleton on MATLAB Simscape that allows us to simulate the gravity, mechanical movements of the mechanism, measure the speeds and torques applied to the joints. We must perform this simulation to compare the results obtained with those obtained by Simulink.

## 4.1 Simscape Multibody

In MATLAB there is an extension called Simscape Multibody that provides a simulation environment for three-dimensional mechanical systems, such as robots, vehicle suspension, construction equipment, etc. You can model different types of systems using blocks representing bodies, joints, constraints, or even force and sensor elements without using the equations describing the system. In addition, this extension gives the ability to import complete CAD assemblies, including constraints already made in another

program where you can create 3D geometry and couplings between parts (e.g. SolidWorks®, AutoCAD, Creo). Once you have imported the system and defined an implementation system, you can also generate a 3D animation that allows you to visualize the dynamics of the system including the presence of gravity. Simscape is designed to help in develop control systems and test their performance. Using the extension in conjunction with SolidWorks® allow us to automatically transfer to the simulation model the mass, inertia, and the center of gravity position. Constraints are also automatically transferred to the simulation model. This procedure is adopted in order to have a further confirmation of the calculations made in the design before buying the components and making the wearable device. Simscape Multibody allows to define rigid and flexible parts using 3D shapes parameterized and allows to create custom parts by defining 2D profiles in MATLAB. It is possible, in the 3D interface, to position and orient the parts that allow the connections through joints that act as a reference for the measurements, because in the joints there are sensors that allow to report the trends as a function of time. With this interface it is possible to define the forces on the joints, or you can give damping rigidity or torque to the joints according to their movement. The uniform gravity fixed or specified by an input signal can be applied to the entire mechanism. In a system built on Simscape there must necessarily be three blocks:

- Solver configuration.

- World frame.

- Mechanism configuration.

Each physical network represented by a connected Simscape block diagram requires solver settings information for simulation. The Solver configuration block specifies the solver parameters that your model needs before you can begin simulation. Each topologically distinct Simscape block diagram requires exactly one Solver configuration block to be connected to it. The block has one conserving port. You can add this block anywhere on

a physical network circuit by creating a branching point and connecting it to the only port of the Solver configuration block [20].

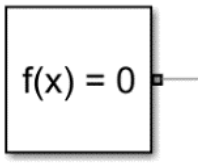



Figure 4.1: Solver configuration block [20]

Instead the following block represents the global reference frame in a model. This frame is inertial and at absolute rest. Rigidly connecting a frame to the World frame makes that frame inertial. Frame axes are orthogonal and arranged according to the right-hand rule. In a frame network, the World frame is the ultimate reference frame. Directly or indirectly, all other frames are defined with respect to the World frame. If multiple World frame blocks connect to the same frame network, those blocks identify the same frame. If no World frame block connects to a frame network, a copy of an existing frame, frozen in its initial position and orientation, serves as the World frame.

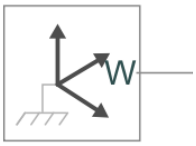



Figure 4.2: World reference block [20]

Finally, the Mechanism configuration block provides mechanical and simulation parameters to a mechanism, i.e., a self-contained group of interconnected Simscape Multibody blocks. Parameters include gravity and a linearization delta for computing numerical partial derivatives during linearization. These parameters apply only to the target mechanism. The

Mechanism configuration block is optional. If you omit it, the gravitational acceleration vector is set to zero. Use only one instance of this block per mechanism, setting uniform gravity to None if that mechanism contains one or more gravitational field blocks. The predefined vector for the gravity is $[0\ 0\ -9.80665]\ {}^{m}/_{s^2}$ that identify the gravity along the -z axis. In our case we define the gravity along the inverse of y axis and so the vector become $[0\ -9.80665\ 0]\ {}^{m}/_{s^2}$.

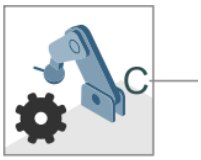



Figure 4.3: Mechanism configuration block [20]

The three blocks listed above are connected in the following way in each Simscape scheme:

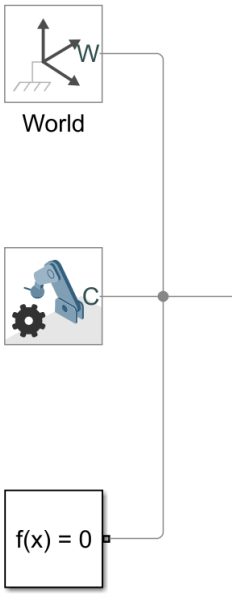



Figure 4.4: Initial configuration

In addition to these blocks that must always be present there are other basic blocks that are used as well:

- Rigid transform
- Solid
- Revolute joint

The Rigid transform block applies a time-invariant transformation between two frames. The transformation rotates and translates the follower port frame (F) with respect to the base port frame (B). The frames remain fixed with respect to each other during simulation, moving only as a single unit. Combine Rigid transform and Solid blocks to model compound rigid bodies.


Figure 4.5: Rigid transform block

In general, frames are triads of axes that encode position and orientation data in a Simscape Multibody 3D model. Each triad consists of three perpendicular axes that intersect at an origin. Each solid component has one or more local references to which it is rigidly fixed. By positioning and orienting the component references, a rigid transformation block sets the rotation and translation offsets between the axes.

The Solid block adds to the attached frame a solid element with geometry, inertia, and colour. The solid element can be a simple rigid body or part of a compound rigid body a group of rigidly connected solids, often separated in space through rigid transformations. Combine Solid and Rigid

transform blocks to model a compound rigid body. Geometry parameters include shape and size. You can choose from a list of predefined shapes or import a custom shape from an external file in STL or STEP format (Solidworks files). By default, for all but STL-derived shapes, the block automatically computes the mass properties of the solid from the specified geometry and either mass or mass density. You can change this setting in the Inertia > Type block parameter. A reference frame encodes the position and orientation of the solid. In the default configuration, the block provides only the reference frame. A frame-creation interface provides the means to define additional frames based on solid geometry features. You access this interface by selecting the create button in the frames expandable area.


Figure 4.6: Solid block

To insert a connection between two solids, such as a joint that allows for rotation, the revolute joint block must be used. This block represents a joint with a degree of freedom of rotation provided by a revolution primitive.


Figure 4.7: Revolute joint block

The origins of the reference system B and F remain coincident during the simulation, in this case the block represents the movement between the two

reference systems as a single transformation variable over time. Internal mechanics include linear spring torques, accounting for energy storage, and linear damping torques, accounting for energy dissipation. You can ignore internal mechanics by keeping spring stiffness and damping coefficient values at 0. We can directly act on 3 parameters:

- Equilibrium position: enter the spring equilibrium position. This is the rotation angle between base and follower frames at which the spring torque is zero. The default value is `0`. Select or enter a physical unit. The default is `deg`.

- Spring stiffness: enter the linear spring constant. This is the torque required to rotate the joint primitive by a unit angle. The default is `0`. Select or enter a physical unit. The default is `N*m/deg`.

- Damping coefficient: enter the linear damping coefficient. This is the torque required to maintain a constant joint primitive angular velocity between base and follower frames. The default is `0`. Select or enter a physical unit. The default is `N*m/(deg/s)`.

In addition, implementation options for the revolution joint can be managed, which include torque and movement. If you select the "Provided by Input" part from the list, you specify an input signal to be provided for the actuation mode in question. Three torque settings can be selected:

- None: no actuation torque.

- Provided by input: actuation torque from physical signal input. The signal provides the torque acting on the follower frame with respect to the base frame about the joint primitive axis. An equal and opposite torque acts on the base frame.

- Automatically computed: actuation torque from automatic calculation. Simscape Multibody computes and applies the actuation torque based on model dynamics.

On the other hand, to provide some movement there are two possibilities for implementation:

- Provided by input: joint primitive motion from physical signal input. The signal provides the desired trajectory of the follower frame with respect to the base frame along the joint primitive axis.

- Automatically computed: joint primitive motion from automatic calculation. Simscape Multibody computes and applies the joint primitive motion based on model dynamics.

If you have a joint block, you can also use the joint block as a sensor. For each joint primitive, the interface provides a detection menu with parameters that can be measured. Each measurement provides the value of a parameter for the reference system F in relation to the base frame of the joint. If the coupling is rotary or spherical, the measured parameters correspond to the angle of rotation, angular velocity, and angular acceleration, respectively. If the joint is prismatic, the parameters correspond to offset distance, linear velocity and linear acceleration respectively. In order to better understand how these tools work, research has been carried out using tutorials available on the site [20] starting from simple models. Below there is a model of a machine suspension carried out using the Simscape Multibody interface of MATLAB. The three initial blocks are always present and they are connected to the various subsystems representing the components of the system (chassis, wheel, arms, shock absorber) that have been realized combining the basic blocks listed above. The final system and two of its graphical representations are illustrated in the figures below.

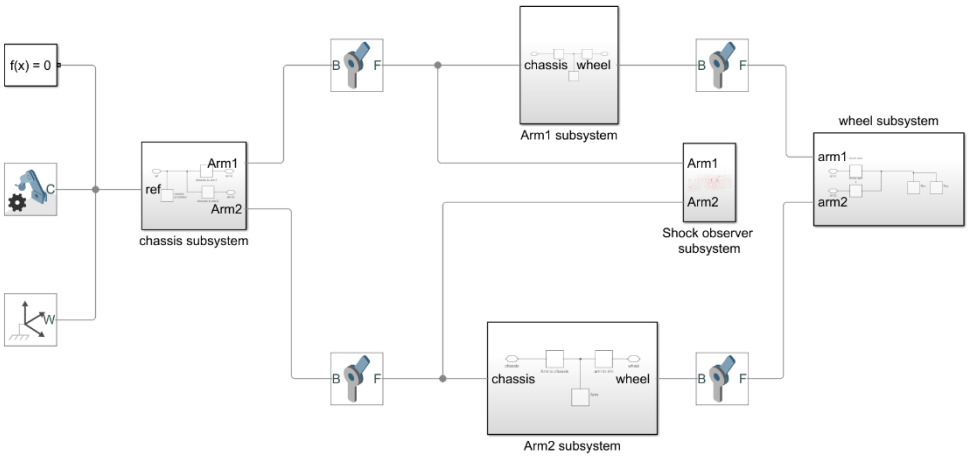



Figure 4.8: Suspension car Simscape scheme

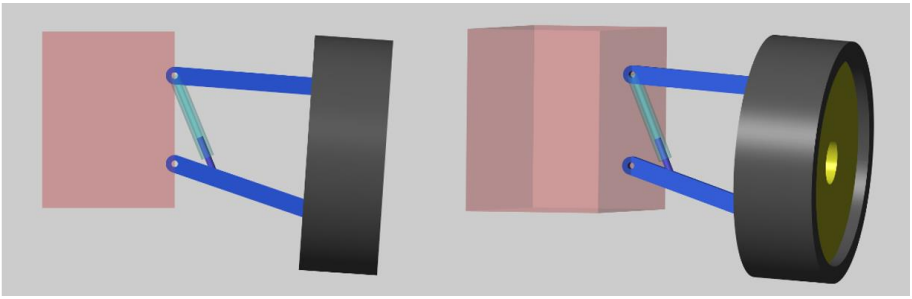


Figure 4.9: Graphical representation of a suspension

Through this example it was possible to understand how these blocks work. After studying the functioning of the blocks from the Mathworks site [20] as reported above the realization of this scheme has allowed to better understand how to insert the mass and inertia of the individual components and the position of the centre of gravity, since we are dealing with very simple components. Besides this, it was also possible to understand well how to use the rigid transformation block and the revolution joint. In addition, a value related to the damping coefficient and from the 10-second Simscape Multibody motion interface it is possible to see the effects of the presence of damping. In the next section is shown the Simscape model of our exoskeleton in a simplified way to better understand how it works and then in the following sections there will be the implementation of the final model by exporting the various components from Solidworks.

## 4.2 Simplified Simscape Multibody model

As said before, at the beginning there is the model of our system realized in a schematize way with simple links. The names of the three main components are:

- Back-frame
- Leg-link
- Traction system

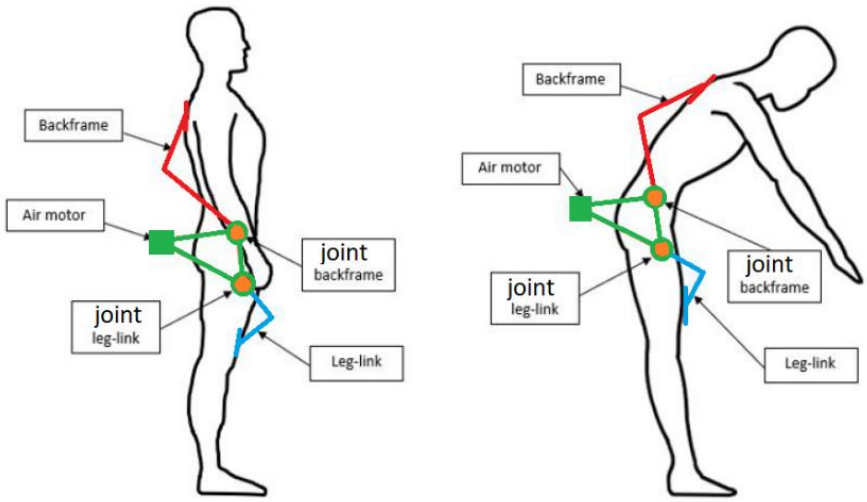



Figure 4.10: Schematic view of the simplified model

The solid blocks have been used to represent the various links. We will analyse as an example the block named “BackFrame SX low” which represents a part of the structure. For the geometry, we have chosen to use the brick shape which is a prismatic shape with geometry centre coincident with the reference frame origin and prismatic surfaces normal to the reference frame x, y, and z axes. Subsequently, about the Inertia parameterization, we choose Calculate from Geometry which enables the block to automatically calculate the rotational inertia properties from the solid geometry specifying the mass. Finally, we choose a colour to highlight all the different block.

The measurements and quantities, such as link length and mass, were taken from the analysis and tables in the past student's thesis on mechanical design.

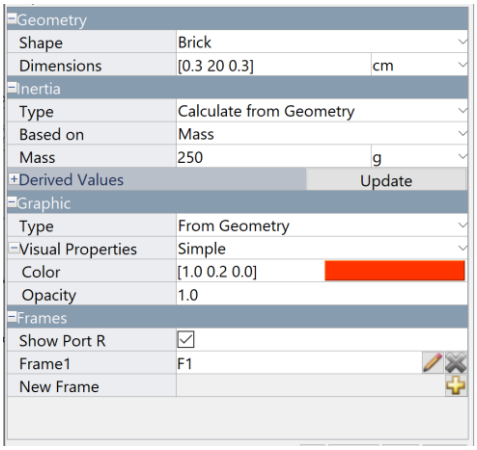
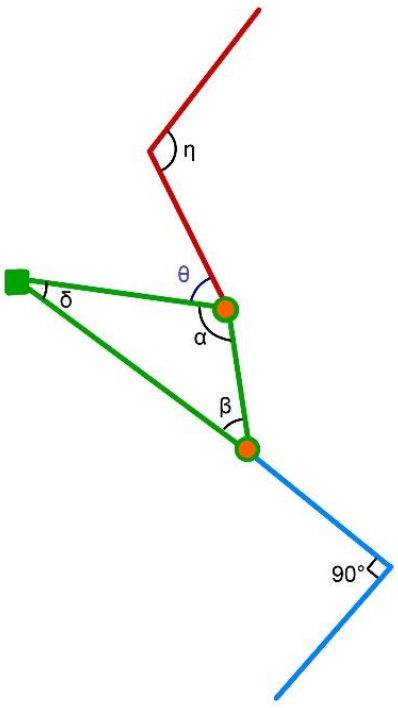



Figure 4.11: Parameters setting for Solid block

Through the Rigid transform block, it has been possible to orient the various components in the correct way by moving and/or rotating the various links of certain quantities. The table below shows the angles of the main rotations.

| ANGLE | DEGREE |
|---|---|
| α | 102.81° |
| β | 52.12° |
| δ | 25.05° |
| θ | 47.18° |
| η | 120° |



### 4.2.1 Backframe

This component has the function of supporting the bending of the trunk by transmitting the torque of muscle support to the operator to whom it is connected by means of shoulder straps; it is a rigid structure and is positioned along the user's back. The Simscape diagram and graphic representation are illustrated in the figures below:

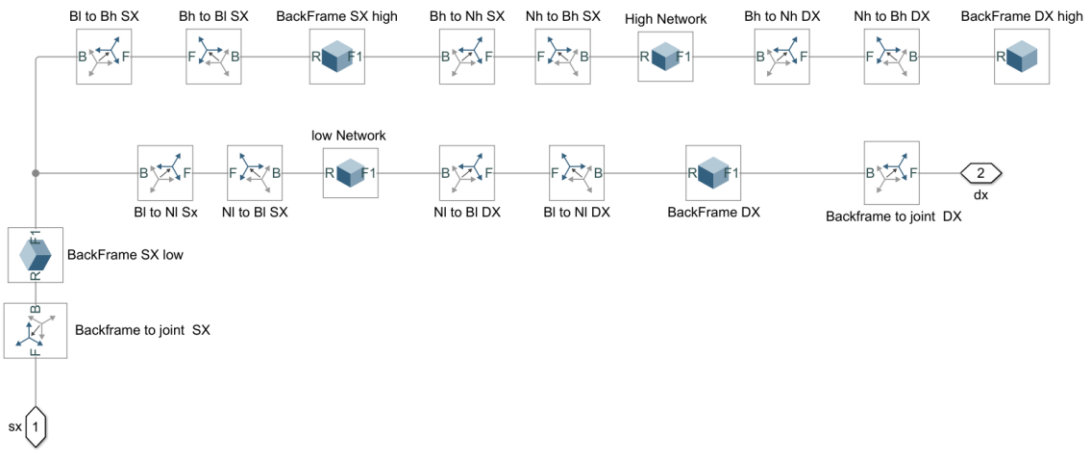



Figure 4.12: Backframe Simscape scheme

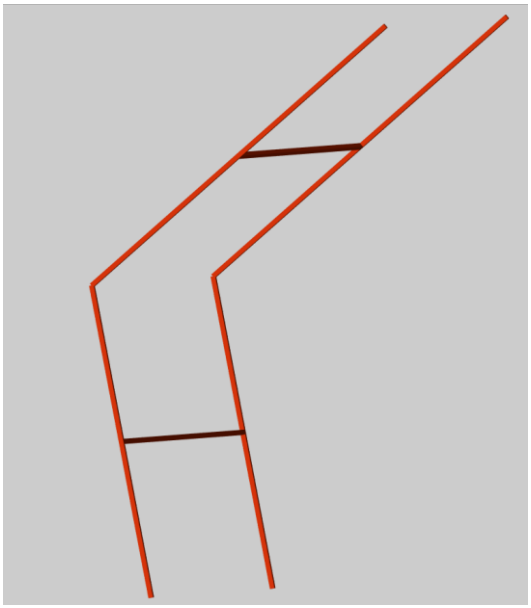


Figure 4.13: Backframe graphical representation

### 4.2.2 Leg-Link

This component represents the part directly connected to the operator's legs by means of an appropriate support to guarantee its comfort; its task is to discharge on the legs the difference between the torque generated by the engine and the one deriving from the backframe. The Simscape scheme and the graphic representation are illustrated in the figures below:

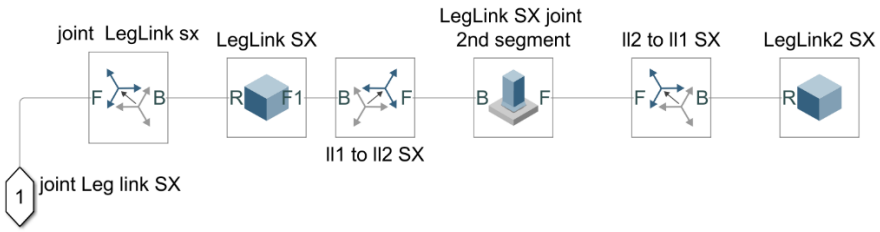



Figure 4.14: Left leg-link Simscape scheme

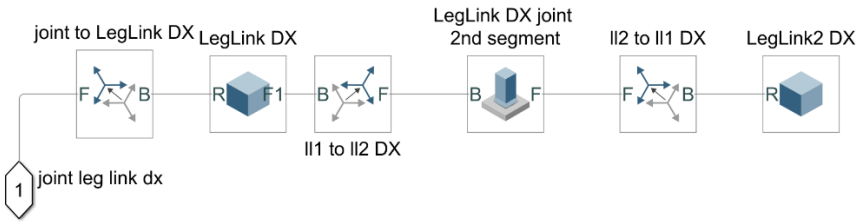



Figure 4.15: Right leg-link Simscape scheme

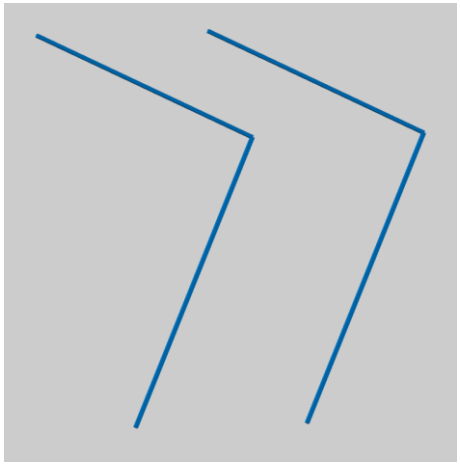


Figure 4.16: Leg-links graphical representation

### 4.2.3 Traction System

This component constitutes the fulcrum of the device as it is precisely there that the air motor, the harmonic drive for torque multiplication and the motion transmission system for the remaining two subsystems are located; moreover, it is directly connected to the human being through an appropriate support that will have to be later on realized to guarantee the operator's comfort. The Simscape diagram and graphic representation are illustrated in the figures below:

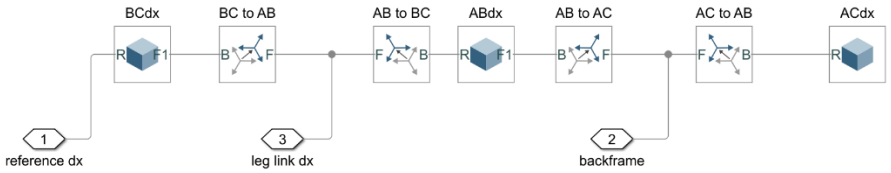



Figure 4.17: Left traction system Simscape scheme

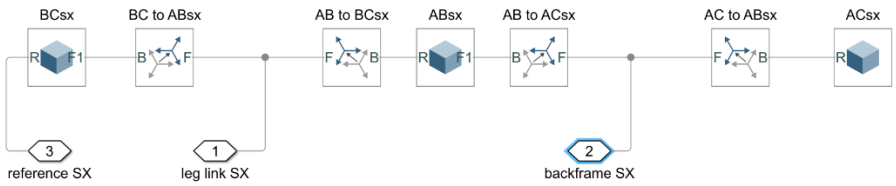



Figure 4.18: Right traction system Simscape scheme

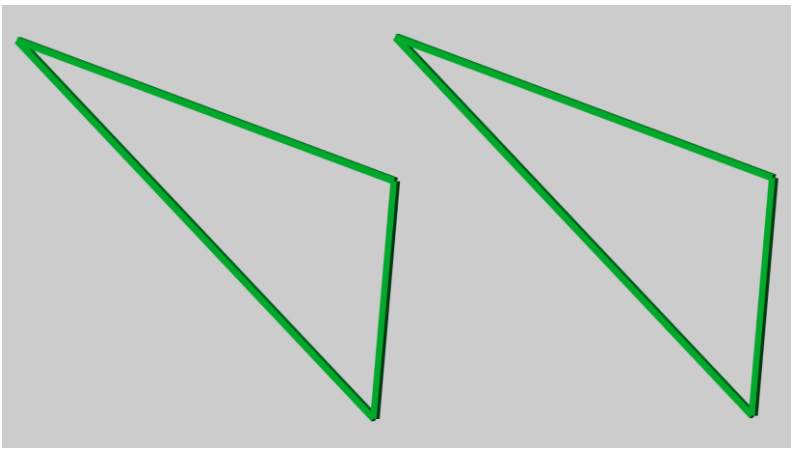


Figure 4.19: Traction systems graphical representation

### 4.2.4 Complete simplified exoskeleton

The three macro-components listed above have been grouped within various subsystem blocks and subsequently some revolute joints have been added to allow movement. Two revolute joints have been placed at the point of attack between the back frame and traction systems allowing the backframe to rotate correctly. The set of the various parameters of the revolute joints was as follows:

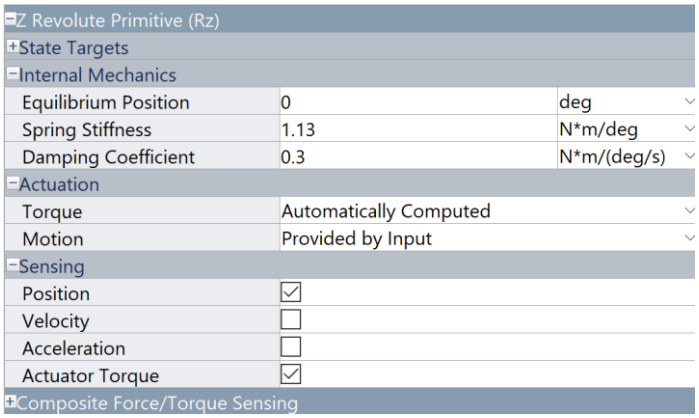



Figure 4.20: Parameters setting for revolute joint block

As far as the internal mechanics section is concerned, we have set the following parameters:

- Spring Stiffness = 1.13 Nm/deg
  This is the torque required to rotate the joint primitive by a unit angle.

- Damping Coefficient = 0.3 Nm/(deg/s)
  This is the torque required to maintain a constant joint primitive angular velocity between base and follower frames.

This is the best way to model the internal dynamics of the motor. Subsequently as far as the implementation is concerned, we have put as input the following quantities

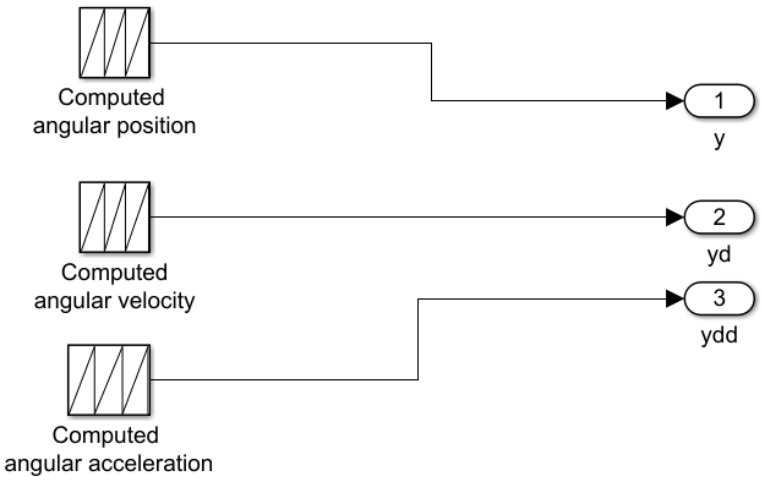



Figure 4.21: Input quantities

Finally, the torque produced by the mechanism is measured using special sensors, inside the revolute joints, to compare it with the driving torque obtained in the previous chapters. Below there is the complete diagram and graphic representation.

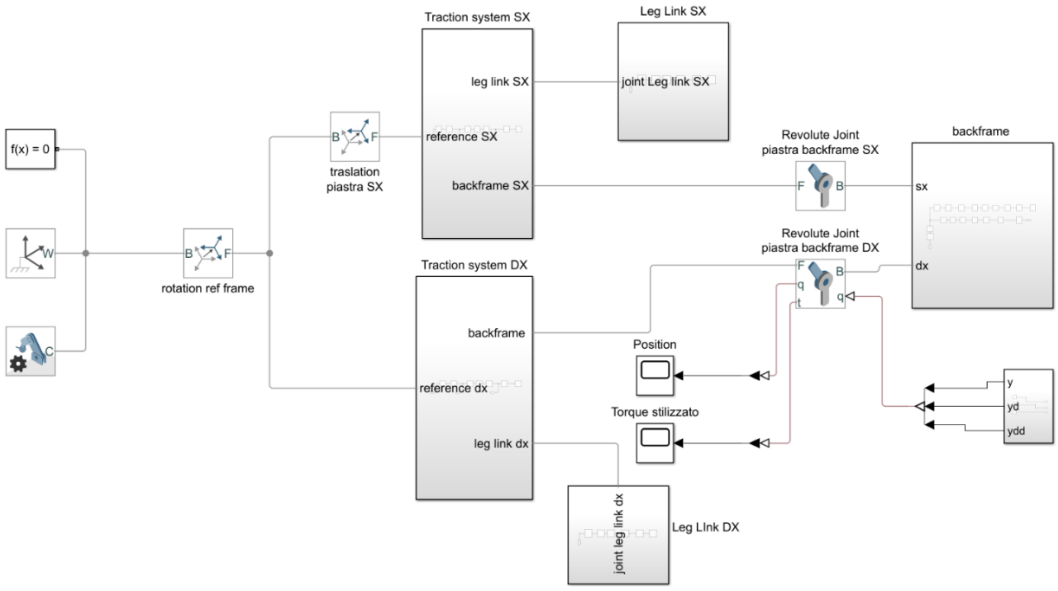



Figure 4.22: Simplified Simscape scheme

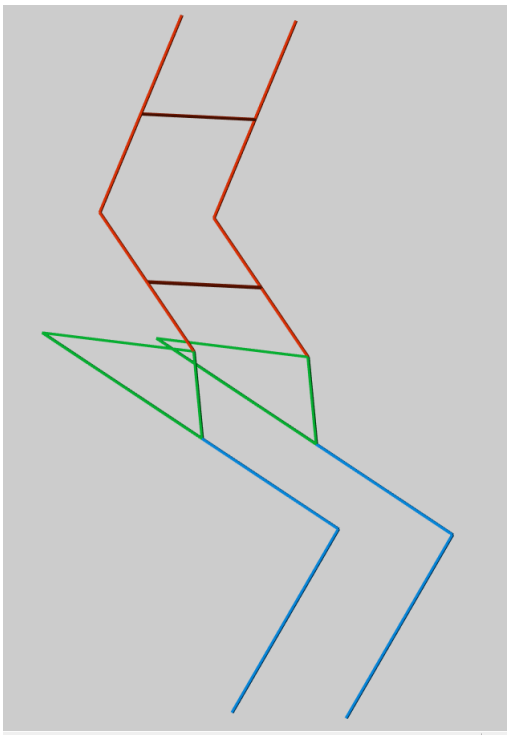


Figure 4.23: Simplified exoskeleton graphical representation

Finally, the trend of the torque produced at the joints is observed and compared with the control torque to verify that the device behaves correctly.

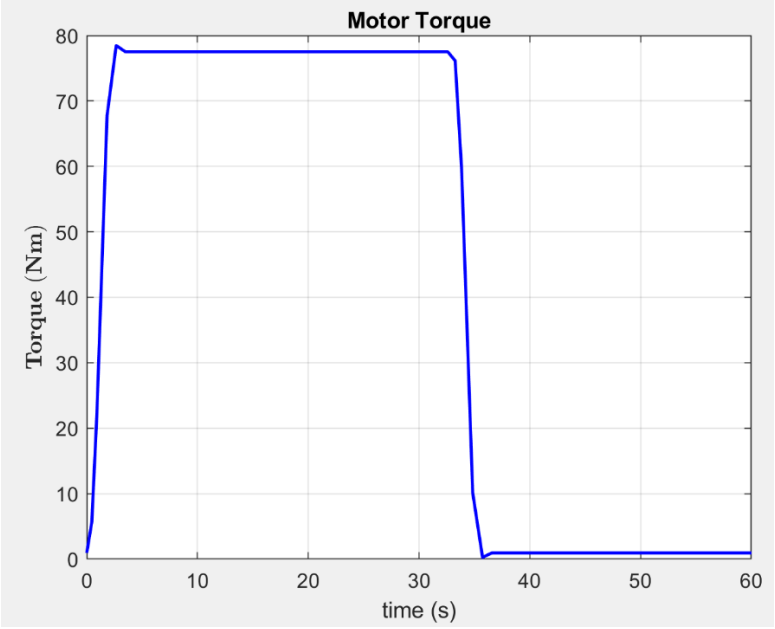



Figure 4.24: Simscape motor torque

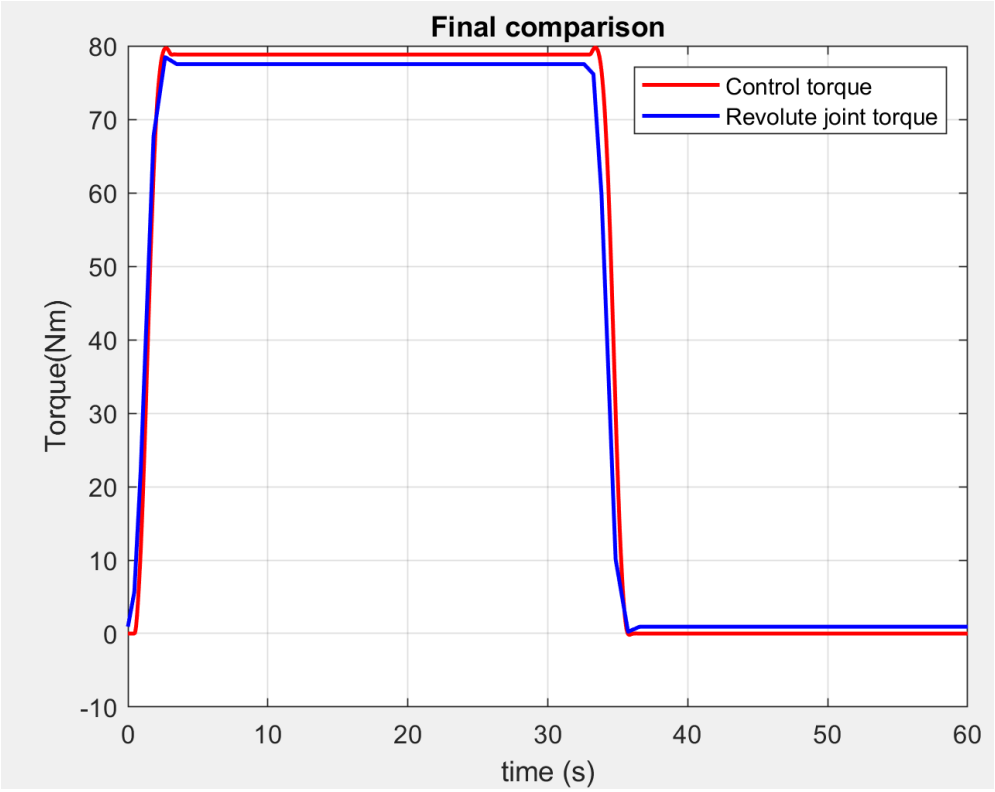



Figure 4.25: Torque comparison

As can be seen from the graphs the trends are similar, they are not the same because in this case the inertia of the various components and couplings is considered and consequently the final trend will be less linear. In the next section we will report the model of the exoskeleton in Simscape exporting the various components from Solidworks, in this way the final trend of the torque should improve because the inertia of the joints and components should be more correct and also the mass and the physical dimensions of all the blocks are the proper one and no more concentrate in one point for what regard the mass distribution.

## 4.3 Human Simscape Multibody model

The Simscape model of the human body is now analysed. The body is represented in a three-dimensional way, but in reality the study is two-dimensional, in fact it is considered a single hip joint in which it are apply the trajectories (chapter 2) that are the one that the man must follow. For what regard the equilibrium torque it is calculated with a trunk mass equal to 57 Kg (value defined by ISO-7250-2 [21]). After having realized the anthropometric parts of the man on SolidWorks® the file was imported to create a Simscape Multibody model.

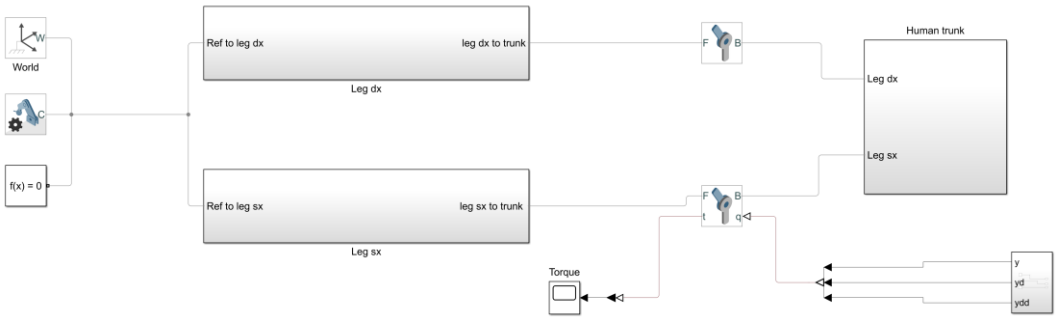



Figure 4.26: Human's Simscape scheme

In this diagram there are three subsystems blocks representing the three macro-areas of the human body, respectively the trunk and the two legs. Below is illustrated the detail of the scheme about one leg (the other is equal) and the trunk.

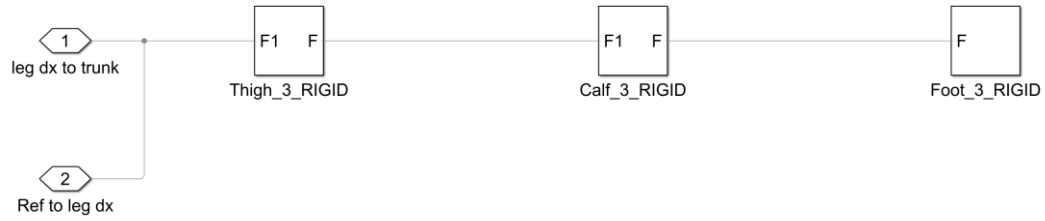



Figure 4.27: Leg scheme

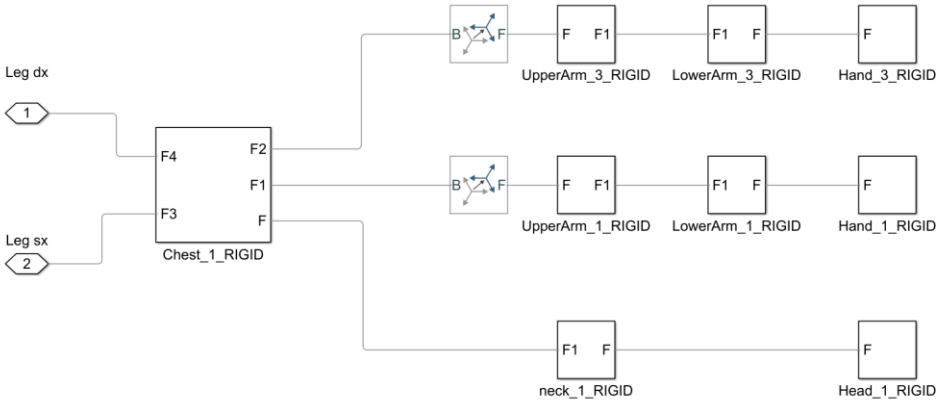



Figure 4.28: Trunk scheme

The rigid transformation blocks in the trunk subsystem were used to rotate the arms by a certain angle. The two leg blocks are connected to the revolute joints so that the position of the revolute joint is located exactly in the hip joint where the torque must be applied, the same revolution joints are also connected to the man's trunk so that the position of the hip joint is the same between the leg and trunk and here it is applied the appropriate torque to the complete bend. In this case, the role of the revolute joint block, in which the angular position is inserted at the input during complete bending, is very important. In the case analysed, the input given to the joint is provided through the cycloidal functions of the angular position; in fact, as has been mentioned previously in the kinematics analysis, the flexion of man can be traced back to a cycloidal function over time (Figure 2.6). From this block it is possible to read through a sensor, incorporated in it, the torque that the muscles must provide to restore balance during flexion.

The torque produced by the human body on its hip joint is characterized by the following formula:

$$C_{MUSC} = mgL_G \sin\theta \; - \; (I \; + \; mL_G{}^2)\ddot{\theta} \qquad (4.1)$$

About the values of the various quantities, those defined in ISO 7250-2 [21] have been used. The trend of the torque during the whole bending time respects the hypothesis made in the previous chapters in which the complete bending is composed of a bending phase, a working phase in

bending position, an ascent phase and finally a rest phase in standing position as is shown in the figure below:

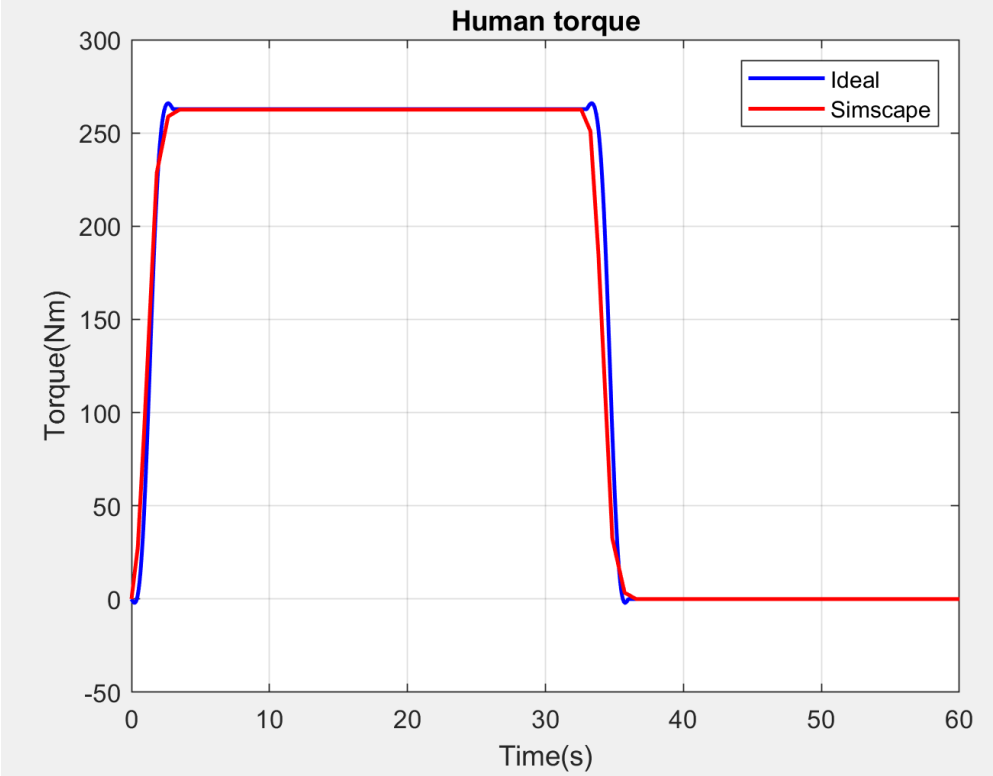



Figure 4.29: Trend of human torque

Analysing the graph, it is possible to see that in the part where the operator is in a flexed position, a constant torque equivalent to 262.8 Nm is produced by the muscles. Instead, the maximum torque is equal to 265.91 Nm and it is provided in the final part of the flexion and in the initial part of the extension, exactly at the moment in which the man is about to stop because he has completed the flexion, or he must provide a higher torque because he must return to the standing position. As you can see from the graph there is a slight discrepancy between the ideal trend and the one obtained in Simscape, this is due to the fact that through Simscape we tried to approximate the dynamics of the human body by inserting in the joint a coefficient equal to 3.97 (Nm)/deg for spring stiffness and a coefficient equal to 1.8 (Nm)/(deg/s) for damping.

Then, it is shown some frames of the video part of the Simscape Multibody interface in which the bending of the man in stand and flexed position is represented.

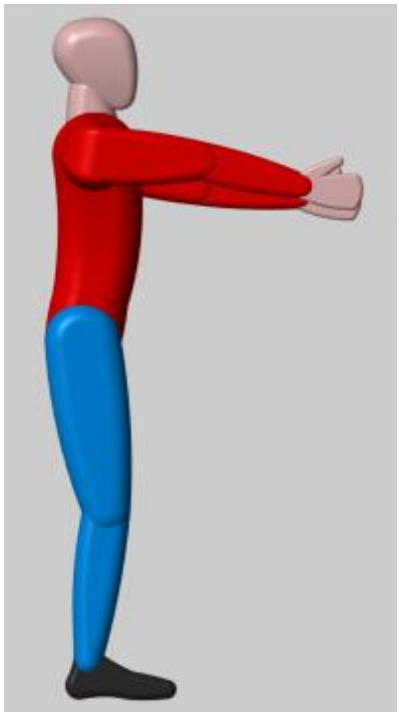


Figure 4.30: Simscape interface, stand up position

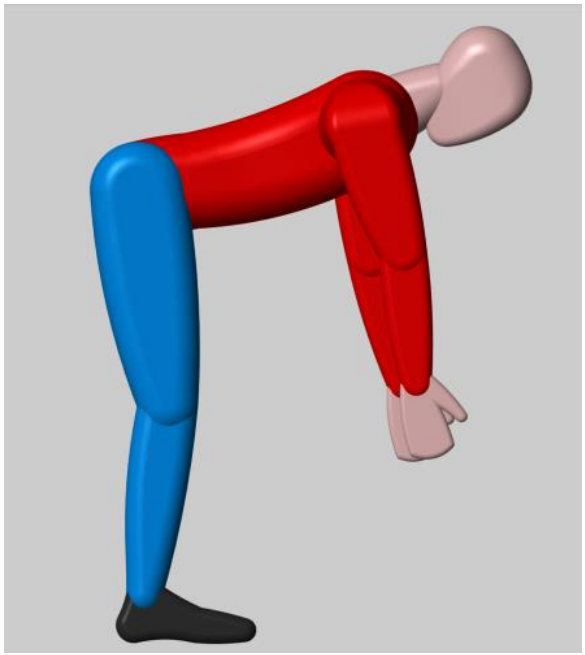


Figure 4.31:Simscape interface, complete flexion

## 4.4 Complete Simscape Multibody model

In this section is analysed the Simscape scheme characterized by the presence of the man and the exoskeleton. After the components are realized on SolidWorks® with the corresponding couplings, the model is imported and adapted on Simscape Multibody. Changes have been made to the model to simplify numerical analysis and the assumptions that have been made are consistent with the previous ones and show that the legs are fixed and have no angular displacement, the torso can only rotate around the hip, the exoskeleton is coupled with the torso on the shoulders and the axis of the electric motor coincides with the axis of the hip. By making these changes, there are no longer any cylindrical joints and, in addition, two revolution joints where the angular position could be applied through the input during complete bending were enough. The final diagram is shown in the figure below.

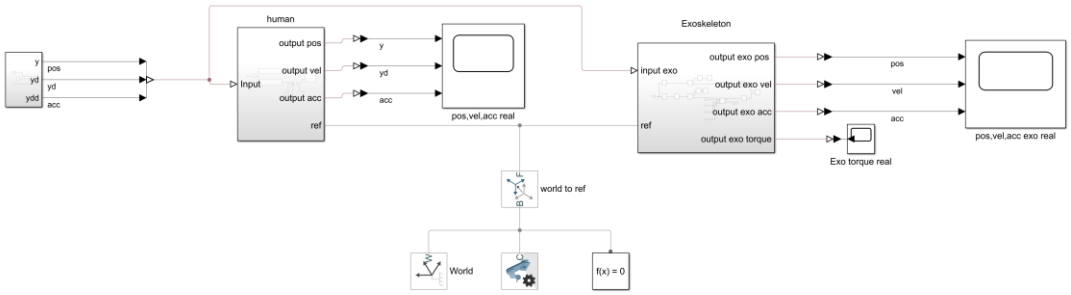
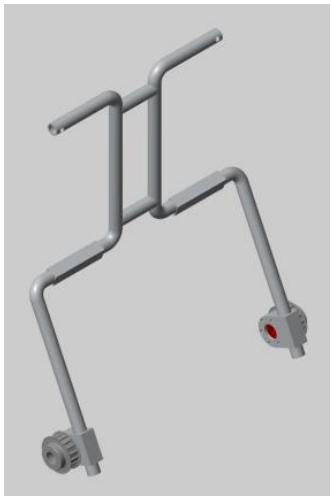



Figure 4.32: Simscape complete scheme

In addition to the various components of the exoskeleton, this diagram also includes the human body block, which has been described in the previous section. The only addition was a block of Rigid transform to allow the human body to be correctly positioned with respect to the exoskeleton. Let us now see the graphical implementation of the various components of the exoskeleton, remembering that now all the components have been imported from Solidworks.

**Backframe**



**Leg-link**

**Traction system**

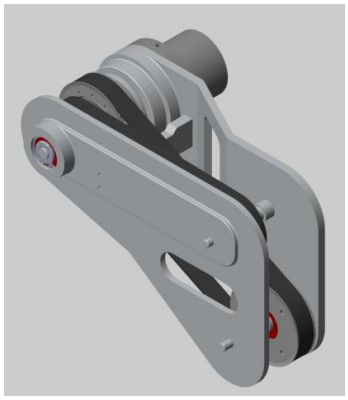


Figure 4.33: Complete exoskeleton

Having imported components from Solidworks, they are characterized by parameters defined at the design stage, so the number of approximations we need to make in the Inertia section of the various blocks is much lower. This leads to equal values for damping and spring stiffness coefficients in the couplings and an improvement in the torque produced by the motors.

| Geometry | | |
|---|---|---|
| Shape | From File | |
| File Type | STEP | |
| File Name | Flangia di collegamento puleggia_Default_sl... | |
| Inertia | | |
| Type | Custom | |
| Mass | smiData.Solid(3).mass | kg |
| Center of M... | smiData.Solid(3).CoM | mm |
| Moments of... | smiData.Solid(3).MoI | kg*mm^2 |
| Products of ... | smiData.Solid(3).PoI | kg*mm^2 |
| Graphic | | |

Figure 4.34: General parameter description

In one of the two revolution joints the input angular position provided by a MATLAB function has been set, and from that block the torque is read by a torque sensor built into the revolute joint. By imposing the presence of

gravity, it is possible to see a 60 second movement of the operator's complete work cycle.

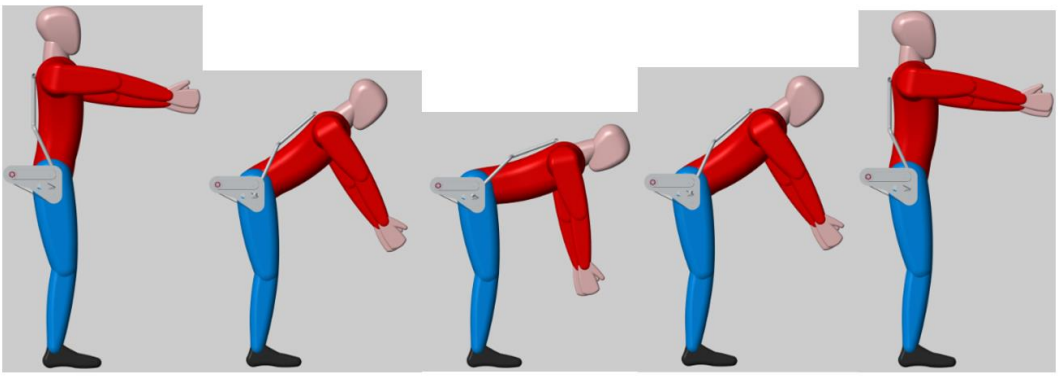


Figure 4.35: Moving sequence

Through the scope mounted on the model and through the built-in sensor in the revolute joint it is possible to acquire the torque provided by the pneumatic actuation system.

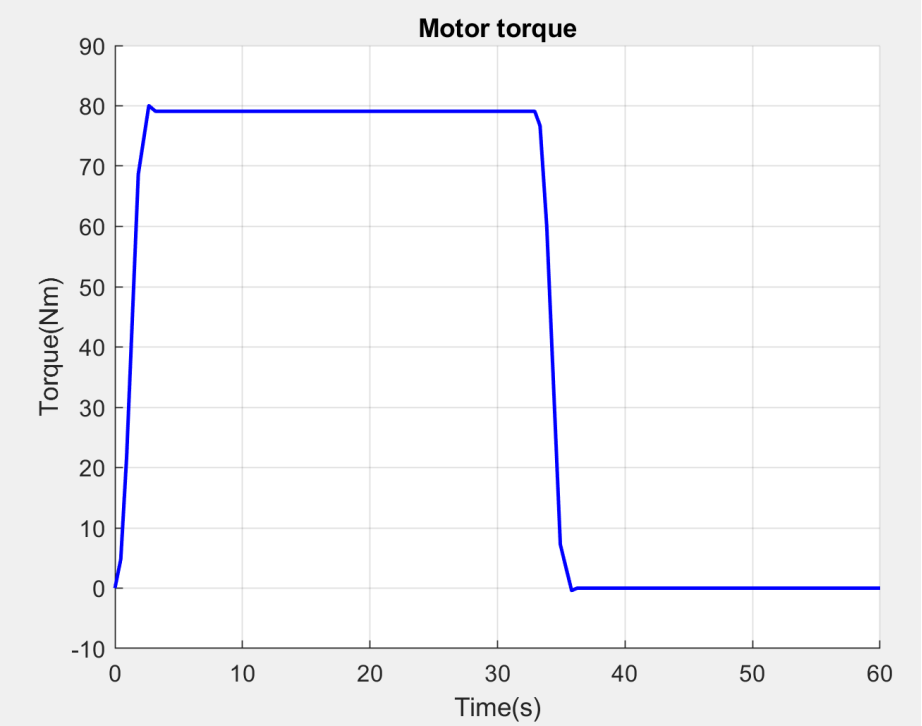



Figure 4.36: Simscape motor torque

From this trend it is possible to see that when the operator arrives in the flexed position, at 70° with respect to the vertical, there is a torque value of 79.1 Nm which is about 30% with respect to the torque required for balance. It can be observed that in the initial part of the flexion and in the final part of the extension it is not necessary to make the air motor work because the torque it gives is negative and would hinder the movement, so the normal work of the muscles is enough. Instead, the maximum torque produced by the actuation system (as already seen above) is in the final part of the bending, when the operator has to stop the movement, and in the initial part of the trunk extension, when the man intends to return to a stand position. It is possible to make a comparison between the torque produced by the muscles without the help of the exoskeleton and the torque produced by the motor; in this way, it is possible to obtain the torque produced by the muscles in the presence of our device instant by instant.

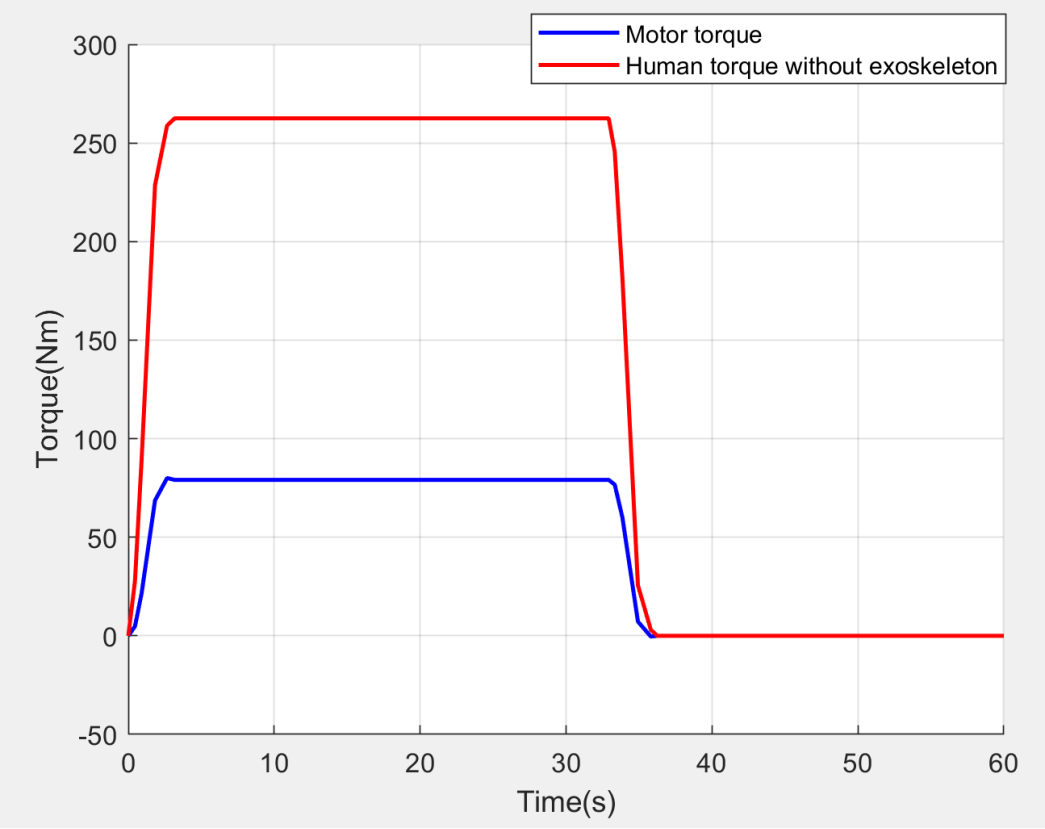



Figure 4.37: Comparison between human and motor torque

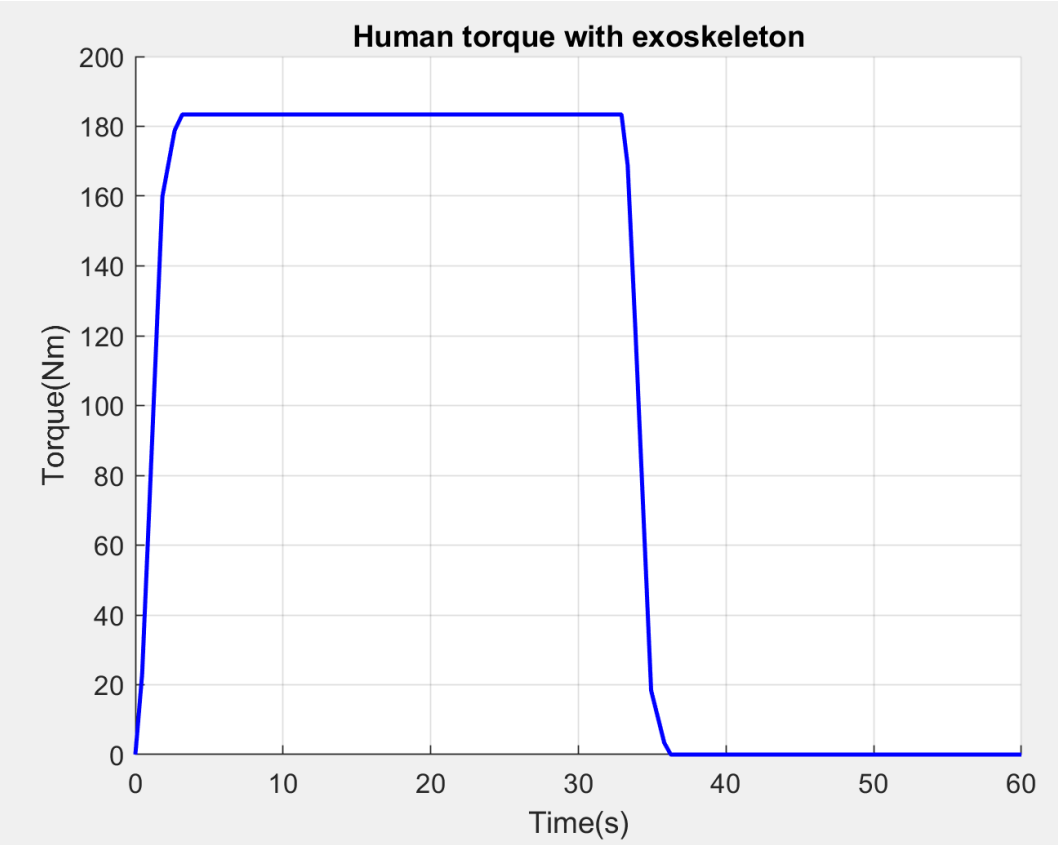



Figure 4.38: Effective human torque

From the graph it is possible to observe that in the part where the downturn ends the human arrives at the flexed position at 70° where we can see that the constant torque value is 183.4 Nm. This value is consistent with the assumptions previously made because it is 70% compared to the constant torque value without the presence of the pneumatic actuator which, as shown above (Fig. 4.29) is equal to 262.8Nm. It is possible from these considerations to understand that the assumptions that the motor provides only 30% operator support are satisfied as can be seen from the numerical analysis.

# 5 StateFlow control

This chapter analyzes the control scheme built with StateFlow, a MATLAB toolbox that enables modeling and simulation of state machines and flowcharts. We must carry out this control scheme to determine in response to which events the exoskeleton moves from one stage of the work cycle to another.

## 5.1 StateFlow

StateFlow is a MATLAB toolbox that can be used for different purposes due to the many functions it possesses. The following are the various fields of use of StateFlow:

- StateFlow® provides a graphical language that includes state transition diagrams, flow charts, state transition tables, and truth tables.

- StateFlow describes how MATLAB® algorithms and Simulink® models react to input signals, events, and time-based conditions.
- StateFlow enables you to design and develop supervisory control, task scheduling, fault management, communication protocols, user interfaces, and hybrid systems.
- It is possible to model combinatorial and sequential decision logic that can be simulated as a block within a Simulink model or executed as an object in MATLAB. Graphical animation enables you to analyse and debug your logic while it is executing. Edit-time and run-time checks ensure design consistency and completeness before implementation.

StateFlow's development environment is also graphic like that of Simulink. It is characterized by a more linear operation and yet there are far fewer types of blocks to know and use. This does not reflect, however, a lower flexibility of behaviour as each block is then programmable independently, thus allowing an infinite expressive potential. StateFlow can work and interact without any problem with Simulink by exchanging data and constant values in real time, during simulation or control of embedded applications.

There are two elements behind the operation of StateFlow:

- State
- Transition

To understand how it works, a very simple example can be analysed: a light bulb that can take two states, off or on. To each state you can associate a transaction that allows you to move in the other state.

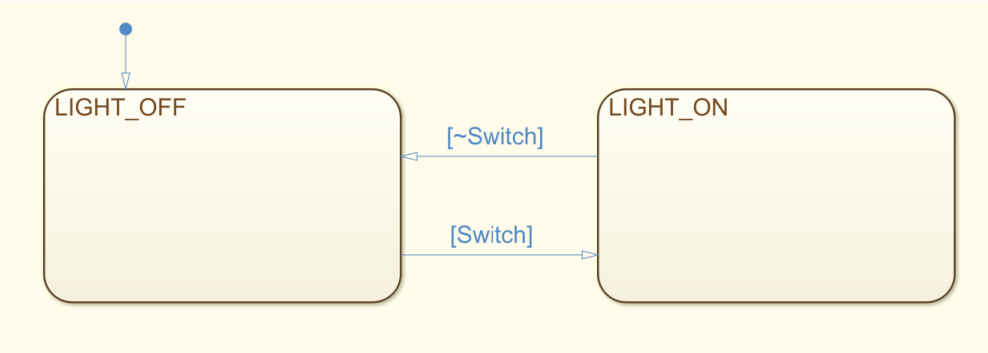



*Figure 5.1: StateFlow light example*

As far as the states are concerned, they are stable positions in which the system persists until a new external event occurs. Each state must be defined by a unique label for the whole machine. The evolution function, determined by the transitions, can procedures at logical conditions determined by input or status variables. Each state has three main phases: input (entry), duration (during) and output (exit). For each phase it is possible to determine in similar code C any reassignment of internal and/or output variables. The additional transaction (called history) that determines the input status of a machine must never be missing.

Transitions, on the other hand, are oriented branches that define the conditions under which a transition from one state to another can occur. Each transition is either defined by a conditional or event-based syntax as described below:

*event[condition]{action on condition;}/action on transition;*

The syntax of the transitions is divided into four parts:

1. Events can be generated externally or raised using the same syntax.
2. Conditions, like "if", identify a logical condition that if true determines the transition.
3. Actions on condition are performed whenever the condition test turns out to be true.

4. Transition actions are performed whenever StateFlow decides to make the transition.

However, states and transitions are not the only elements present in the StateFlow schemes but there are also many others that are listed in the figure below.

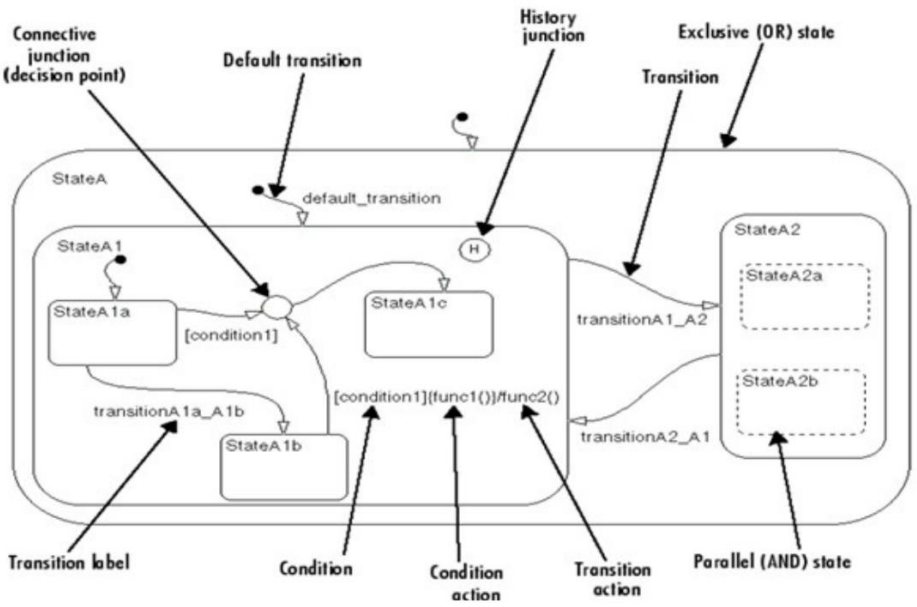



*Figure 5.2: StateFlow elements*

A special mention should be made for a block that is not present in the figure above, the graphical function block.

A graphical function in a StateFlow® chart is a graphical element that helps you reuse control-flow logic and iterative loops. You create graphical functions with flow charts that use connective junctions and transitions. You can call a graphical function in the actions of states and transitions. With graphic function, is to possible to:

- Create modular, reusable logic that you can call anywhere in your chart.
- Track simulation behaviour visually during chart animation.

For example, this graphical function has the name f1. It takes three arguments (a, b and c) and returns three output values (x, y and z). The function contains a flow chart that computes three different products of the arguments.

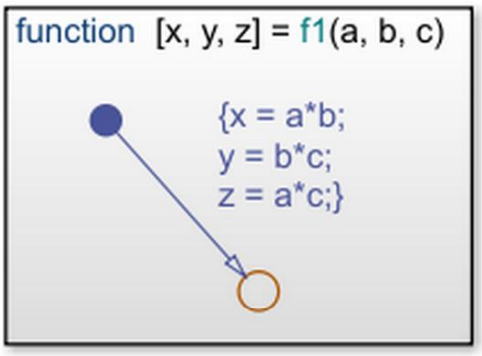



*Figure 5.3: Function block*

## 5.2 StateFlow control scheme

It has been chosen to realize this control scheme through StateFlow in order to manage the transitions between the four phases of the work cycle defined in the previous chapters, paying particular attention to the aspects related to the oscillations due to the wearer's walking (they prevent an activation of the exoskeleton if the angular position remains lower than 5°), to the presence of the power supply, the emergency case and finally to the user's willingness to operate the exoskeleton or not.

About the design of the entire control logic using StateFlow, it was decided to divide the work into three macro-areas:

- Definition of input signals.
- Definition of the diagram with relative states and transitions.
- Definition of output signals.

### 5.2.1 Definition of input signals.

The input conditions of the control logic are represented by three switches and the 3 angular quantities coming from the human body, i.e. angular position, angular speed and angular acceleration. The three switches are used to simulate the presence of the power supply, the exoskeleton ignition and the emergency button. The Simulink block you used is the manual switch which, as can be seen from Figure 5.4, requires two input signals.

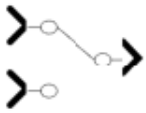


*Figure 5.4: Manual switch*

As input it has been chosen to use two constant blocks assigning the values of 0 and 1. In the case of the simulation of the presence of the power supply 0 represents the lack of air while 1 represents the presence of air, while in the case of the ignition of the device and consequent production of torque by the engines 0 represents the device off and 1 instead the device on. For what regards the emergency button when the switch is set to 1 means that there is a problem instead when it is set to 0 means that everything works well.

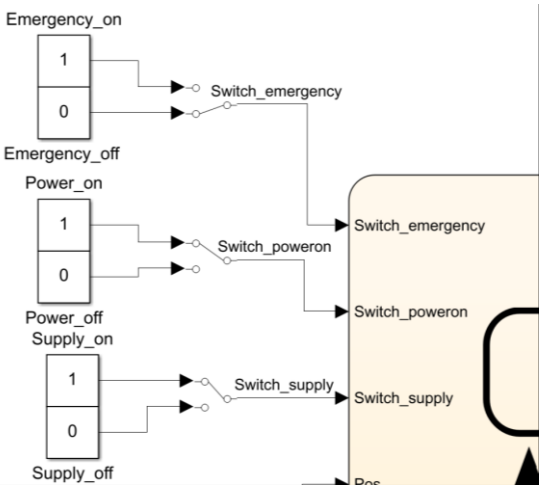



*Figure 5.5: Input switches*

The three quantities: angular position, angular velocity and angular acceleration that describe the behaviour of the human being correspond to the output signals of the human Simscape model. In the final project, i.e. in reality, these quantities will be provided by the sensors specially positioned on the shoulder straps of the exoskeleton. This part has not been studied within this work but has been left as a possible field for further investigation in the future. To make up for the lack of sensors and to avoid assigning as input the ideal signals obtained from the cycloidal laws, it has been chosen to use the outputs of the Simscape model of the human being because it takes into account the non-idealities of the human body. It is possible to see a comparison between the ideal trajectories and the ones that are taken from the Simscape human model in the following figure.

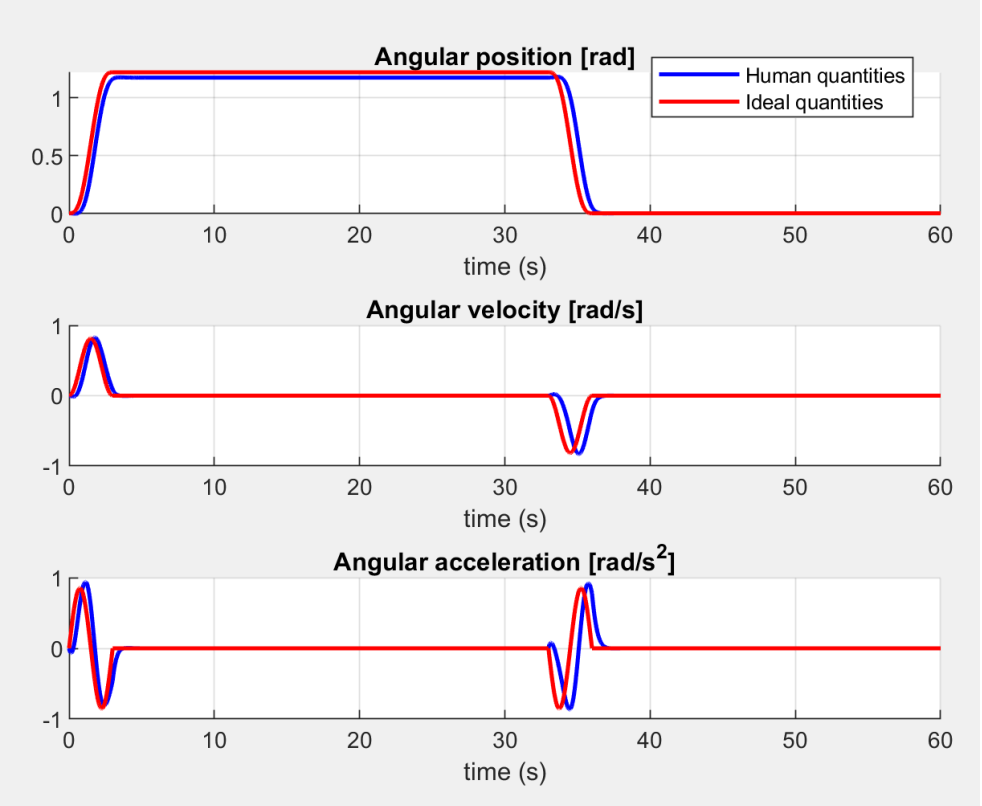



*Figure 5.6:Input quantities comparison*

This difference leads to a discrepancy between the ideal torque, calculated as explained in Chapter 2 and the torque produced by the human being.

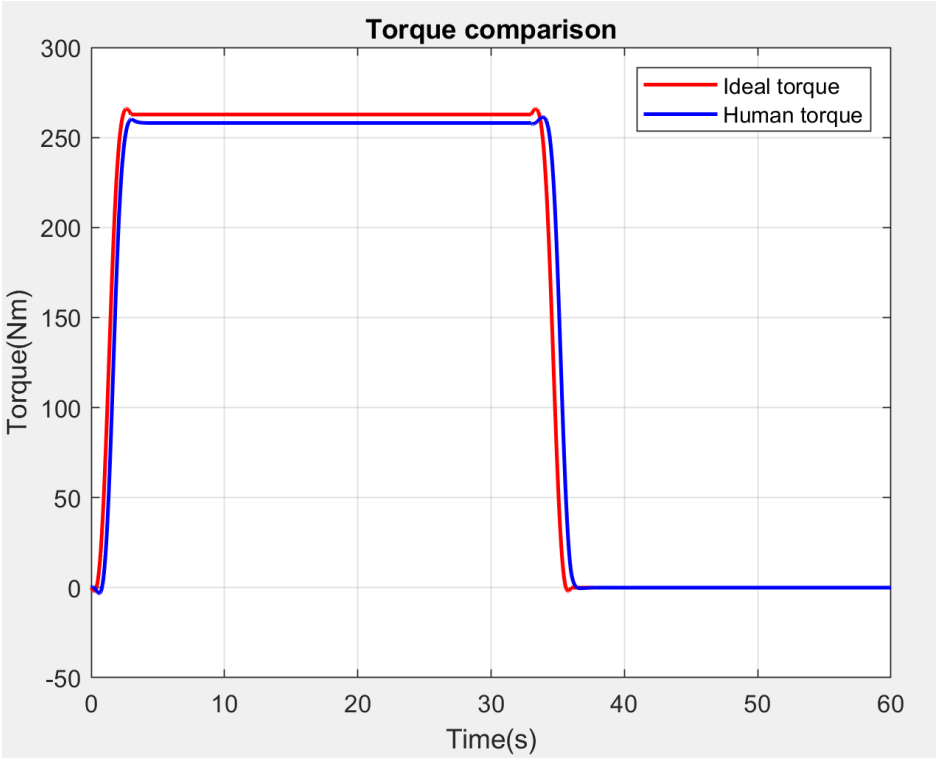



*Figure 5.7: Torque comparison*

The result containing all the inputs to the diagram is visible in the figure below:

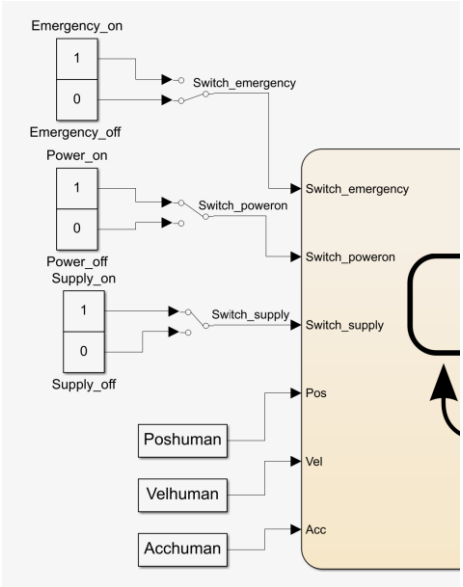



*Figure 5.8: Diagram inputs*

### 5.2.2 Definition of the diagram with relative states and transitions.

After analysing the inputs to the StateFlow diagram, the realization of the diagram itself and the logic of operation are deepened. The control scheme realized in StateFlow is visible in the figure below.

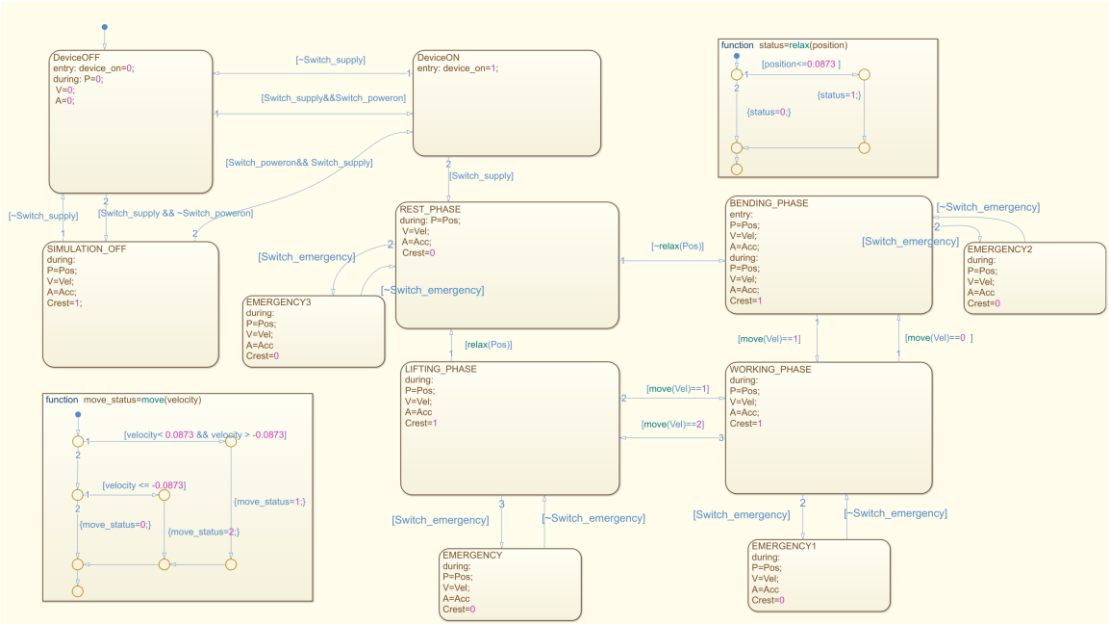



*Figure 5.9: StateFlow scheme*

As you can see in the complete diagram there are eleven states and two function blocks with their transitions. Analysing more in detail the state blocks we have the two states called Deviceoff and Deviceon that serve to the simulation of the presence or not of the power supply. The transition between them is governed by the value of Switch_supply (0: supply not present, 1: supply present). Is present then, a state called SIMULATION_OFF in which instead you can enter and exit based on the value of the variable Switch_poweron and Switch_supply, this is because when you are inside this state it means that the device is powered but the wearer is not interested in using it. Then, there are also four identical states that are responsible of the emergency case, in order to enter in these states we need to press the emergency button that is an input of the system and

once we are there we set the variable Crest equal to zero, this means that in case of emergency the exoskeleton produce zero torque. Finally there are four blocks, called respectively REST_PHASE, BENDING_PHASE, WORKING_PHASE, LIFTING_PHASE that indicate the 4 phases that constitute the entire work cycle that was used for the study of the device.

For the cycle to be able to begin there is need that is the variable Switch_supply that Switch_poweron are set to 1, after which the transitions that govern the passage from one phase to the other of the cycle are defined through the use of two functions.

As mentioned above, it was decided to take into account the small involuntary oscillations detected by the sensors during the walk and to manage this detection to ensure that the exoskeleton is activated, or produce a torque of help, only when the measured angular position is greater than 5°. This behaviour is managed by defining and using the function:

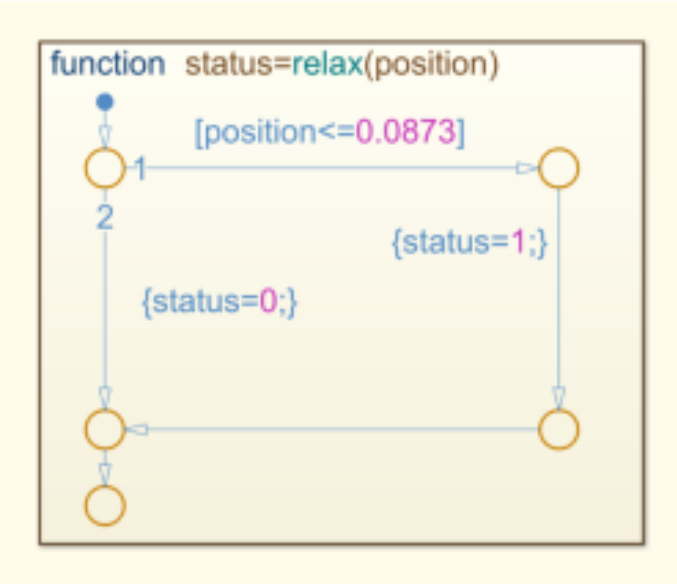



*Figure 5.10: Function status=relax(position)*

This function depends on the value of the angular position that is detected by the trajectory leaving the Simscape block of the human being, in reality by the sensors, sets the value of the status variable equal to 0 when the

angular position is less than 5°.The status variable governs the transition between REST_PHASE and BENDING_PHASE (status=1) or the transition between LIFTING_PHASE and REST_PHASE (status=0). As for the management of the transitions relative to the WORKING_PHASE state, another function has been created:

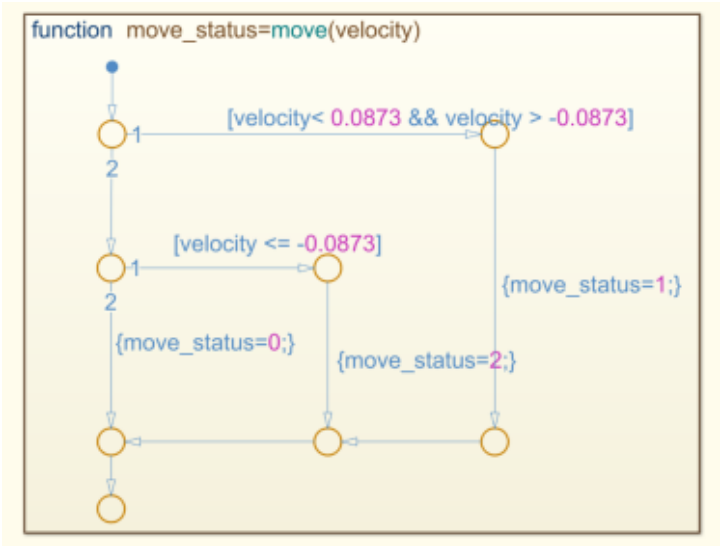



*Figure 5.11: Function move_status=move(velocity)*

This function analyses the value of the angular velocity and, depending on it, sets the value of the move_status variable to 0, 1, 2.

- 2: the speed is less than or equal to -5 °/s, this means that the human being has undertaken the phase of ascent and consequently when the variable move_status=2 happens the passage between WORKING_PHASE to LIFTING_PHASE.
- 1: The speed is between 5°/s and -5°/ s, therefore, the wearer is stopping its rise or bend phase and then when move_status=1 occurs either the transition from BENDING_PHASE to WORKING_PHASE or the transition from LIFTING_PHASE to WORKING_PHASE.

- 0: The speed is greater than or equal to 5°/s and then you have the transition from WORKING_PHASE to BENDING_PHASE this means that the human starts to bend.

As mentioned above when the angular position is less than 5°, the torque produced by the device must be zero, To realize in practical terms this condition has created a variable called Crest that is set to 0 when you are inside REST_PHASE while it is always left equal to 1 when you are in other states. The effects of this setting will be visible in the next chapter when it is analysed the complete simulation scheme realized by Simulink.

### 5.2.3 Definition of output signals.

The output conditions of the control logic are represented by four variables and one display. The four variables are the 3 angular quantities that originally come from the human body, i.e. angular position, angular velocity and angular acceleration but now re-elaborated in the StateFlow chart and the variable of the previous chapter Crest.

The display is used to show the value of the device_on variable which can take two values:

- 0 when our device is inside the DeviceOFF state which means the device is off.
- 1 when the exoskeleton is active and power on and as a result, that means that we are in the DeviceON state.

The Simulink block that is used is the display block which, as can be seen from Figure 5.12, requires one input signal. As input, as mentioned above, we have decided to use the variable device_on

*Figure 5.12: Display block*

The result containing all the outputs to the diagram is visible in the figure below:

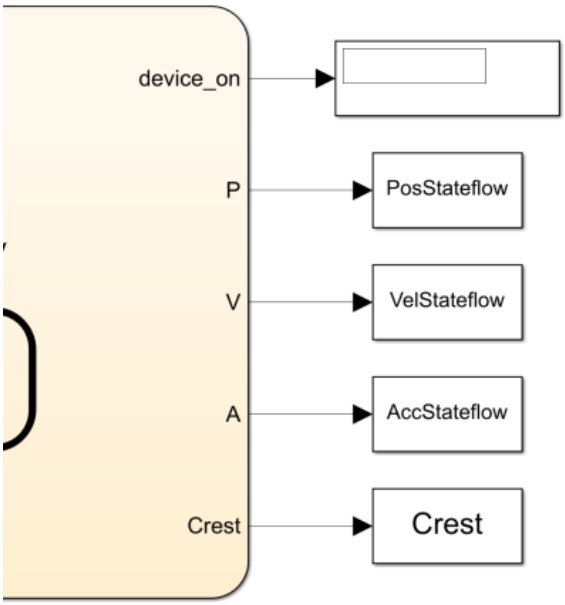



*Figure 5.13: Output quantities*

The three quantities: angular position, angular velocity and angular acceleration that describe the behaviour of the human being correspond to the output signals of the human Simscape model but now they are re-elaborated. Thanks to the Crest variable it was possible to manage the oscillations during the human walk by making the exoskeleton produce torque zero. The comparison between the ideal torque and the one produced by the exoskeleton is visible in the figure.

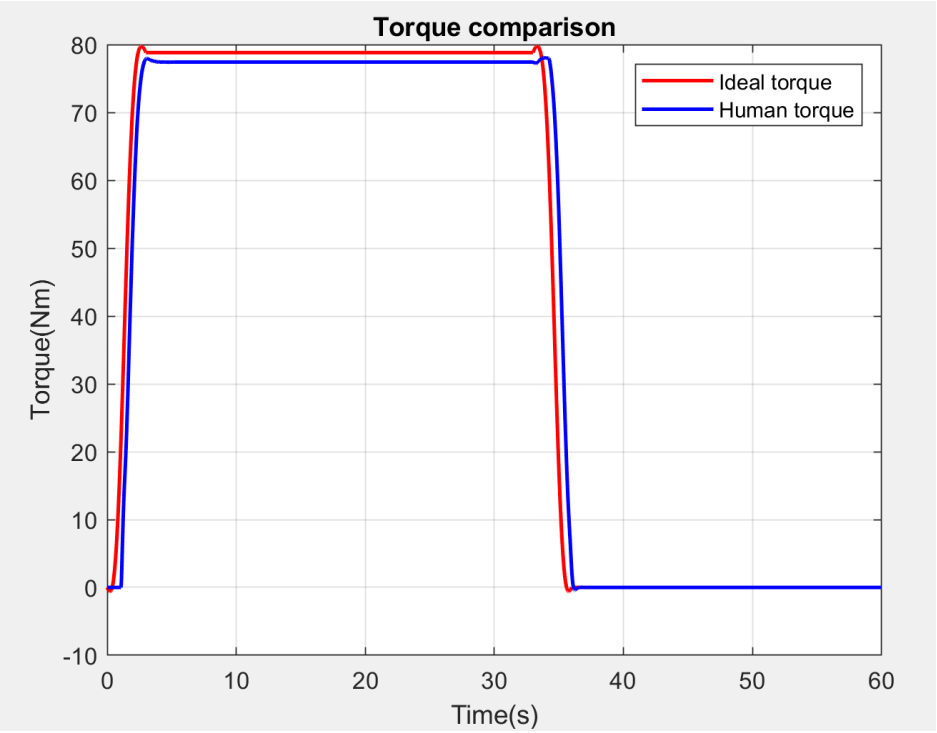



*Figure 5.14: Torque comparison*

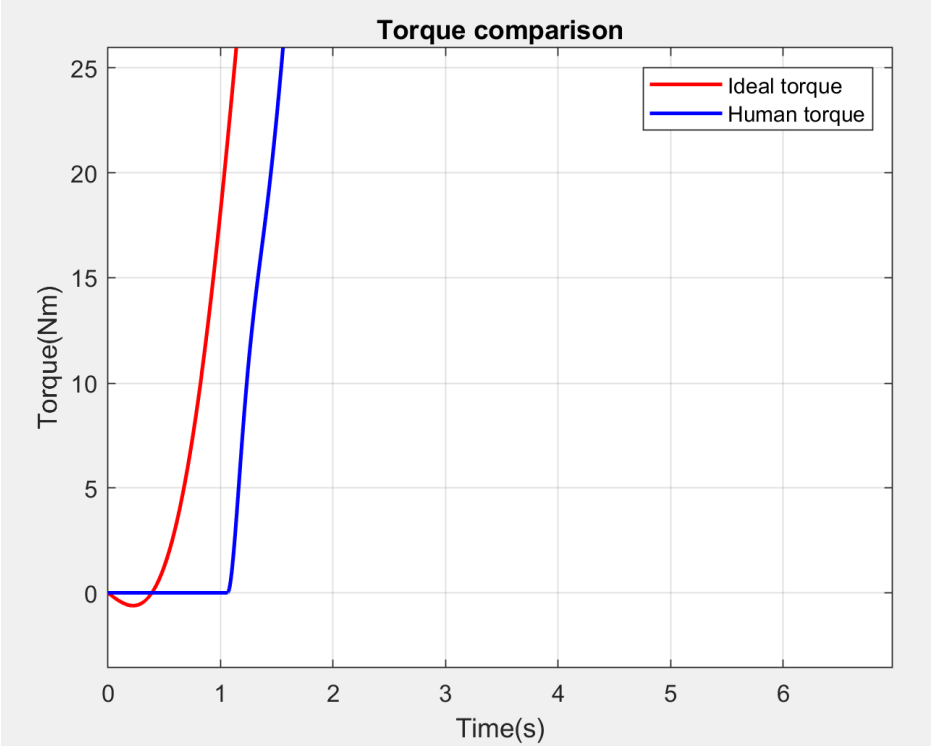



*Figure 5.15: Detail of the zero torque production*

## 5.3 Complete scheme

After analysing the different section that composed our StateFlow scheme is shown the complete scheme where it is possible to see the complete architecture of the control logic.

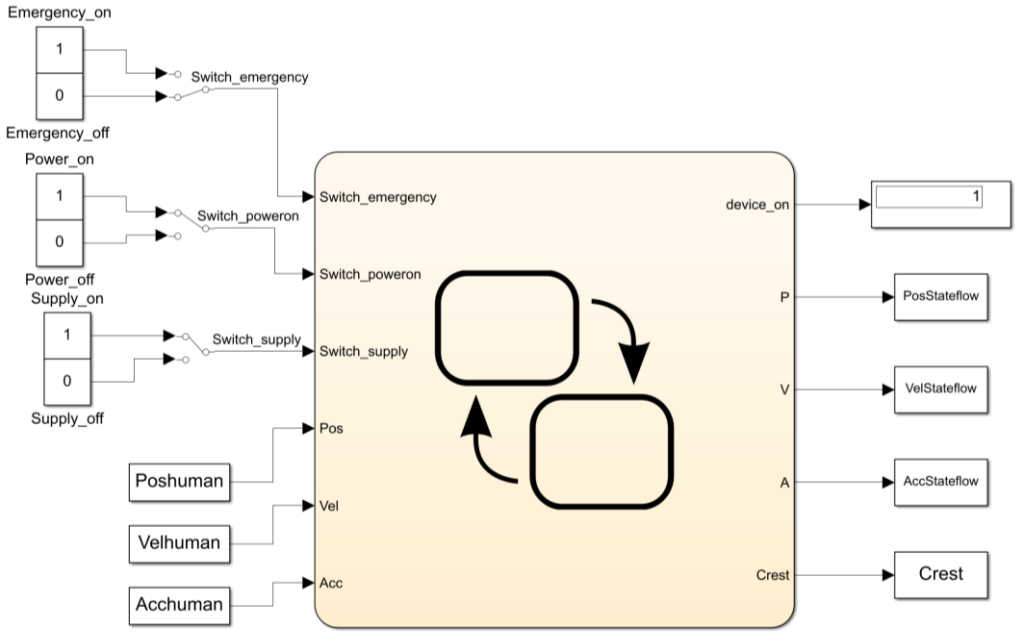



*Figure 5.16: Complete StateFlow scheme*

# 6 Simulation results

Once the design phase of the Simscape models and the StateFlow diagrams has been completed, we move on to the simulation phase to verify the behaviour of the exoskeleton in certain situations, trying to simulate some aspects that will occur during the normal operation of the device. The Simulink toolbox then defined a simulation scheme that included the Simscape models of man and exoskeleton analysed in Chapter 4 and the StateFlow diagram studied in Chapter 5. The final diagram is shown in Figure 6.1. The inputs of this scheme are the angular position, the angular velocity and the angular acceleration obtained from the cycloidal laws while the final outputs are the same three quantities but relative to the exoskeleton. In the middle, the inputs will initially be sent to the Simscape model of man to account for the non-idealities due to the human body, then processed by StateFlow and finally, through a torque generation block you will get the reference torque of the torque control loop useful to cleanse the motor torque produced by the exoskeleton.

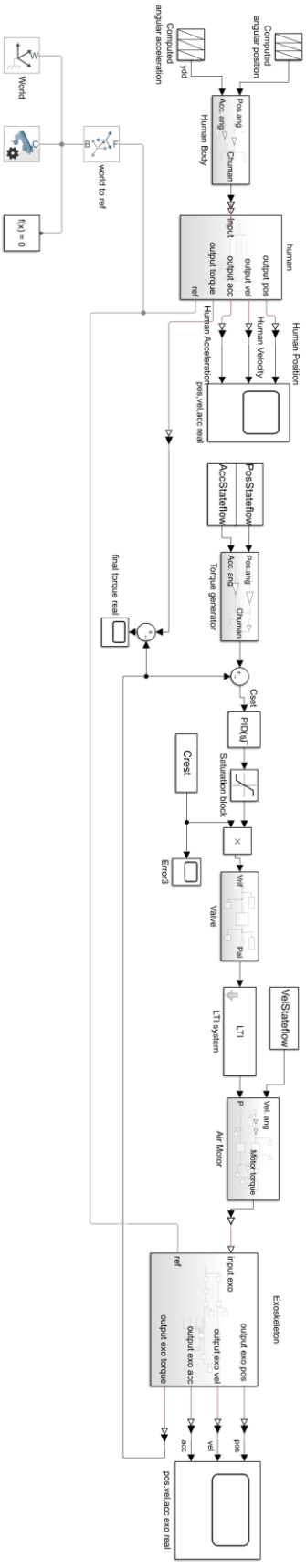



*Figure 6.1: Simulation scheme*

Analysing the scheme in more detail, the blocks responsible for the simulation of the movement of the human body are the following:

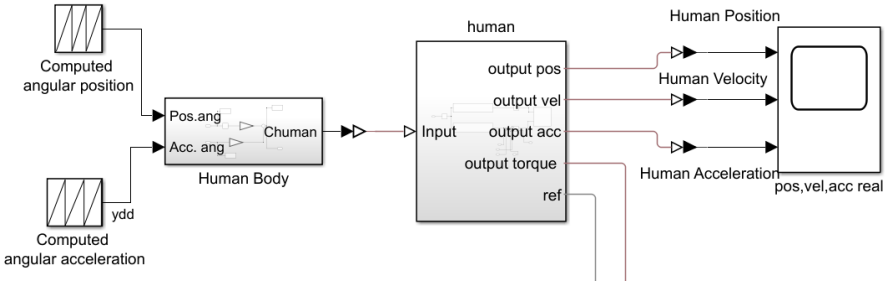



*Figure 6.2: Human's blocks*

These blocks are inserted in the diagram as they allow us to modify and make more real the ideal cycloidal quantities. This is possible because given the angular acceleration and position trajectories the torque is calculated by means of a torque generation block and the latter is applied to the revolute joint (i.e. the hip joint) of the Simscape model of man.

As output they take the three angular quantities, through the sensing function inside the revolute joint in Simscape, and they send them to the StateFlow diagram in order to elaborate them and to define the movement of the exoskeleton according to the situations. The output torque instead, is used simply to obtain the resulting torque of the complete system through a summing node. The angular position, angular velocity and angular acceleration are treated within the StateFlow diagram as presented in Chapter 5 and then used or to define the reference torque of the control loop (position and acceleration) or as input for the pneumatic engine model (speed). The torque control is carried out as follows:

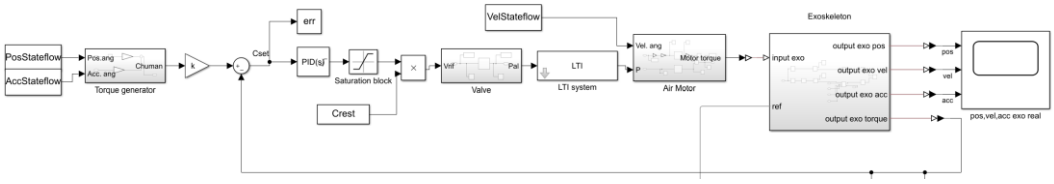



*Figure 6.3: Torque control loop*

To obtain the reference torque, that is 30% of the torque produced by the human being, it is necessary to include at the output of the torque generation block a constant K=0.3. At the exit from the motor block you will get the torque that must be applied to the hip joint present in the exoskeleton. The outputs will then be the three quantities relative to the exoskeleton that must have a trend very similar to the initial ideals but with a certain number of non-ideal present inside. The torque produced at the hip joint of the exoskeleton will be sent to the summing node together with the torque produced by the man to obtain the resulting system torque.

### 6.1.1 First simulation: a single bending of 70°

Up to now we have always had as input to the system a set of trajectories, visible in figure 2.6, that simulated a bending of 70°, that is the maximum angle of bending for the trunk of the human being. The situation is described by the following sequence of images:

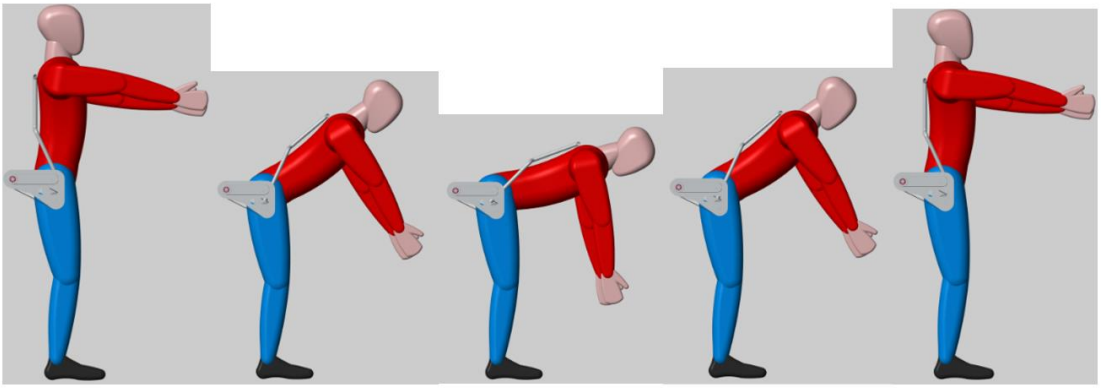


*Figure 6.4: Sequence of motion*

After applying these inputs to the system, you can immediately see the deviation between the ideal case and the trajectories produced by the human body.

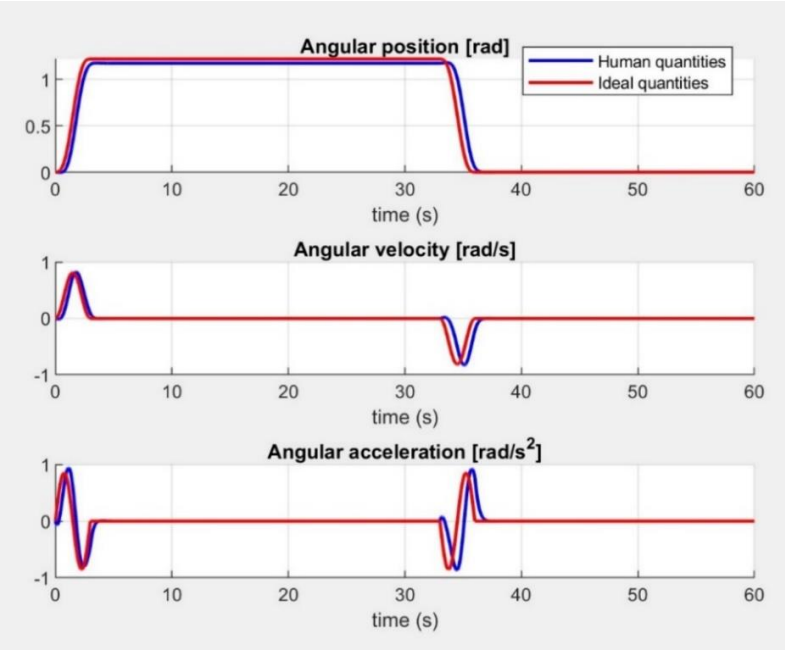



*Figure 6.5: Ideal quantities-Human quantities*

Given these inputs and thanks to the simulation scheme, realized on Simulink, we obtain the torque produced by the human body, the torque produced by the exoskeleton and the resulting torque of the system, characterized by the following trends:

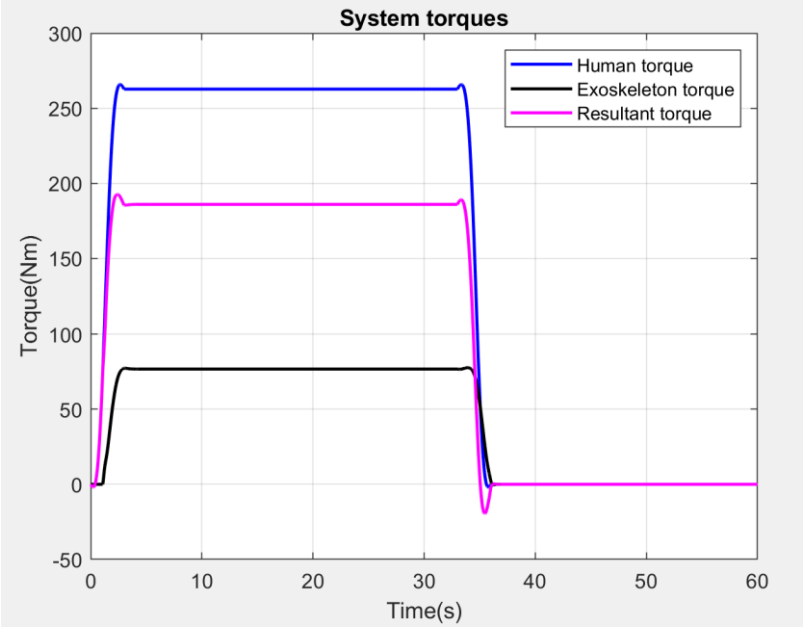



*Figure 6.6: System's torques*

Given the torque produced by the exoskeleton at the hip joint, we can go back to the relative positions, velocities and accelerations at the

exoskeleton joint and compare them with those produced by the human to verify that everything is compliant.

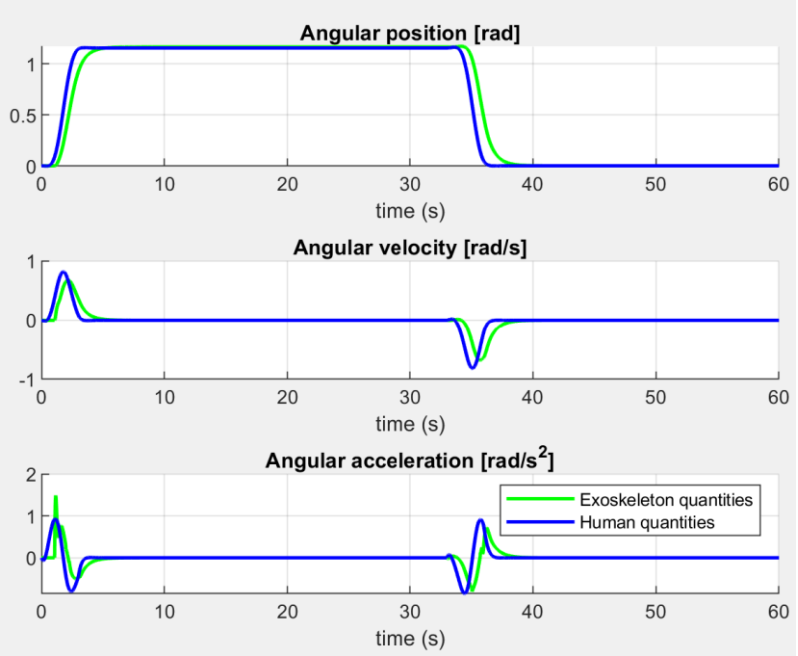



*Figure 6.7:Human quantities- Exoskeleton quantities*

The tracking error, that is the difference between the reference signal and the feedback signal, has the following trend:

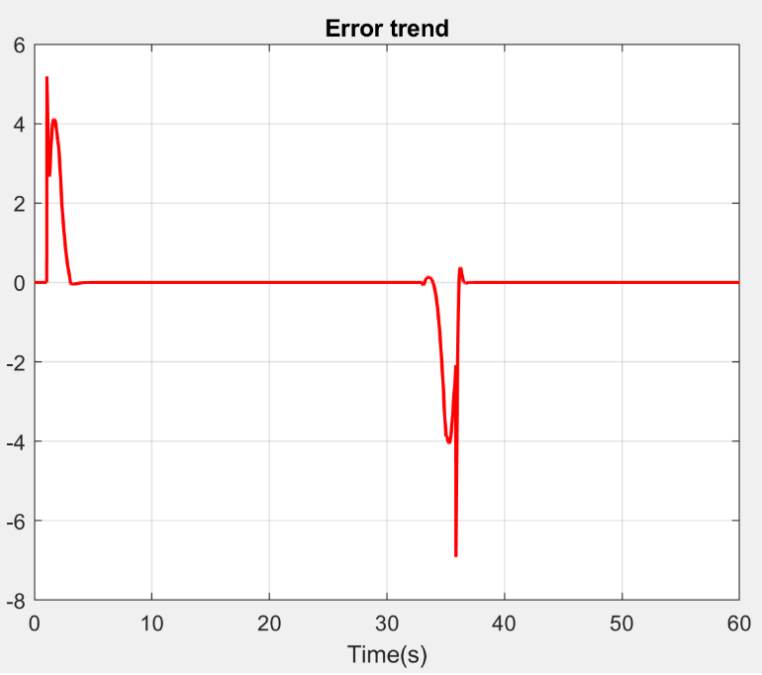



*Figure 6.8: Error trend*

### 6.1.2 Second simulation: multiple bending and lifting

Now to consider a case different from the one previously analysed and to obtain results more in line with the real case, the following ideal trajectories have been designed:

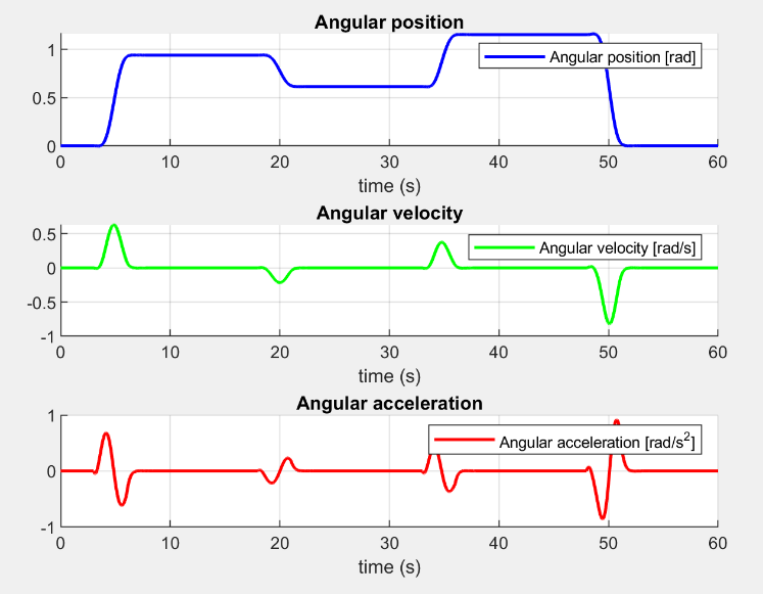



*Figure 6.9: Input quantities*

The angular position and consequently the two remaining quantities are characterized by a first bending at 50 degrees with subsequent phase at constant position to allow the processing of the pieces, then there is a rise of 20 degrees resulting in a phase of work at constant position. Then, there is a new bending of 40°, that allows us to reach the maximum angle of bending of the human trunk and finally an ascent up to the resting position. In the figure below you can see the sequence of the movement.

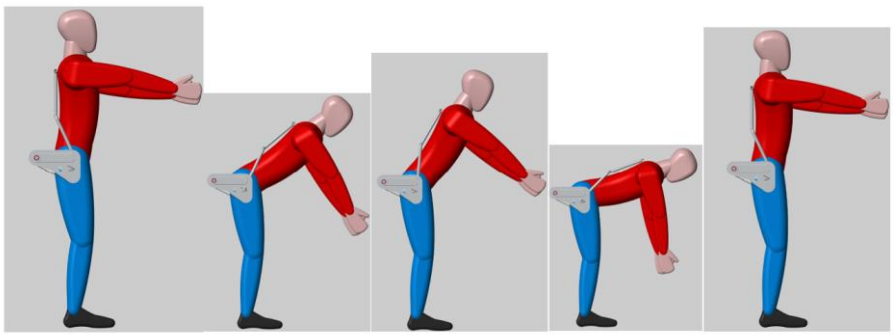


Figure 6.10: Sequence of motion

Also, in this case the trajectories produced by the human body diverge slightly from the ideal trajectories as it is visible in the figure below:

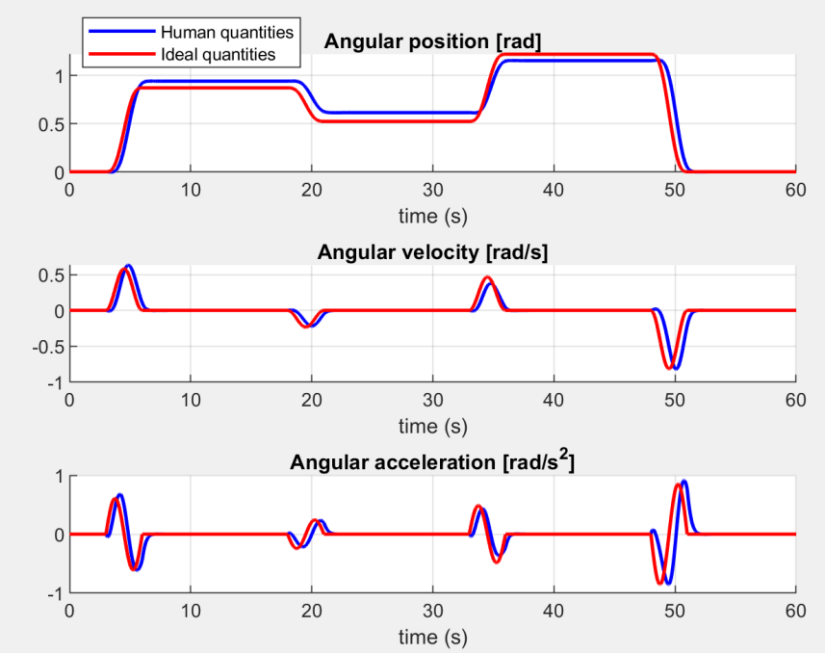



*Figure 6.11:Ideal quantities-Human quantities*

The torques in play this time will have the following trends:

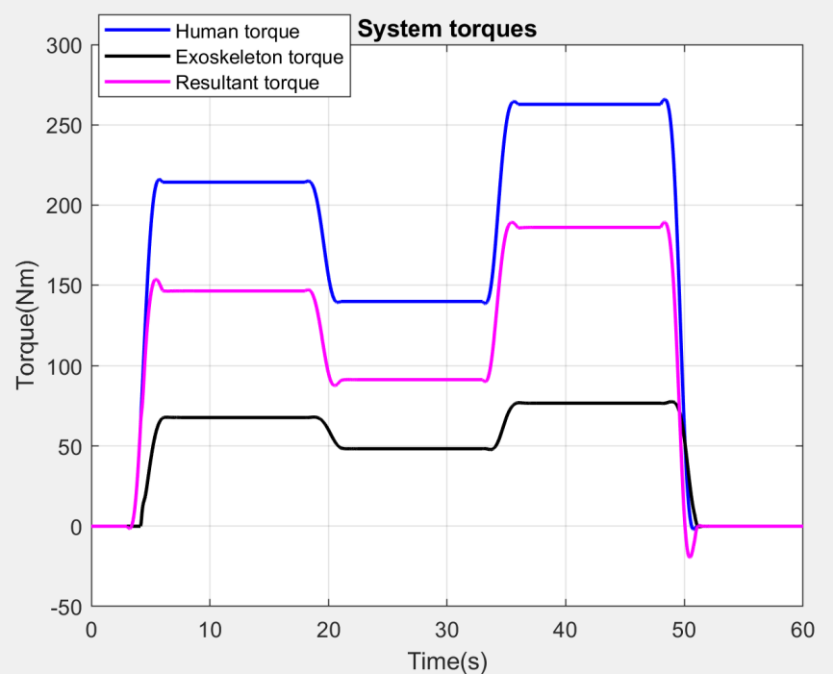



*Figure 6.12: System's torque*

You can notice how in all cases the exoskeleton torque (black curve) starts slightly later as this is due to the design and management of the oscillations that lead the exoskeleton to produce zero torque until the angular position is less than 5°. For the final positions of the exoskeleton and the tracking error, the following trends are obtained.

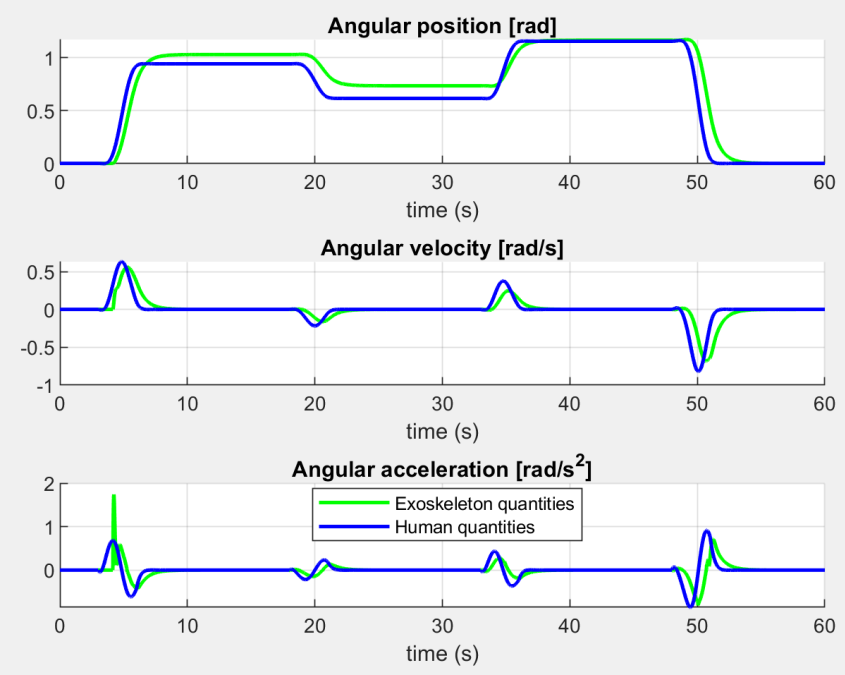



*Figure 6.13: Human quantities- Exoskeleton quantities*

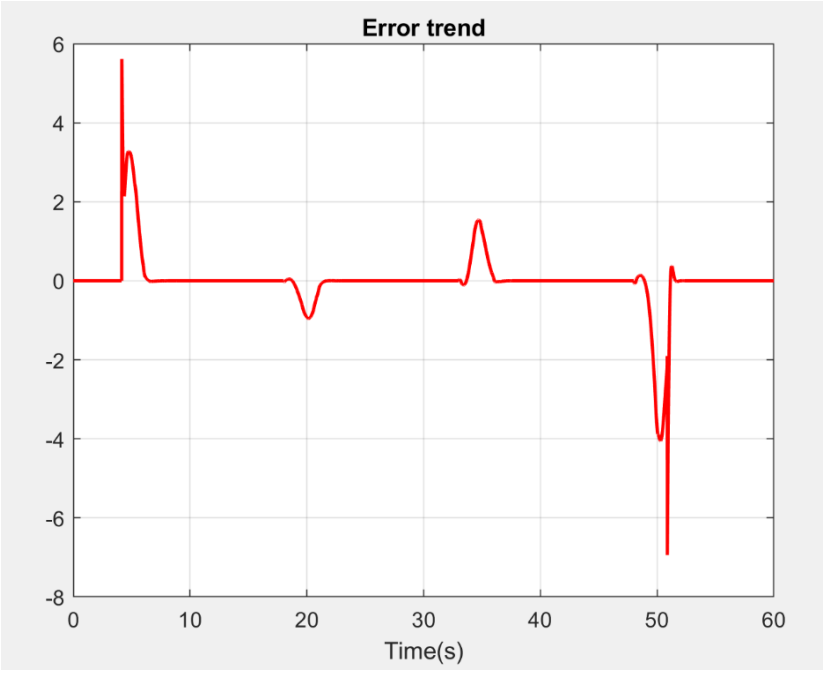



*Figure 6.14: Error trend*

### 6.1.3 Third simulation: single bending of 4°

As it has been said in the previous chapters when the human being walks it is wanted that the exoskeleton is activated and consequently produces a torque of support only and exclusively if the angular position is greater than 5°. To test this situation the following quantities were supplied as input to the system( a very small bending of a angle equal to 4°). In the figure you can also see the discrepancy between the ideal sizes and those coming out of the model Simscape wearer.

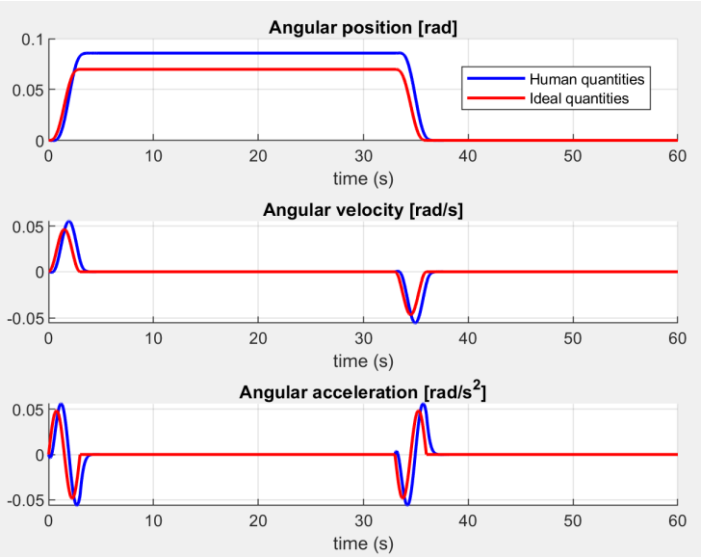



*Figure 6.15: Ideal quantities-Human quantities*

These inputs are translatable in the following sequence of motion.

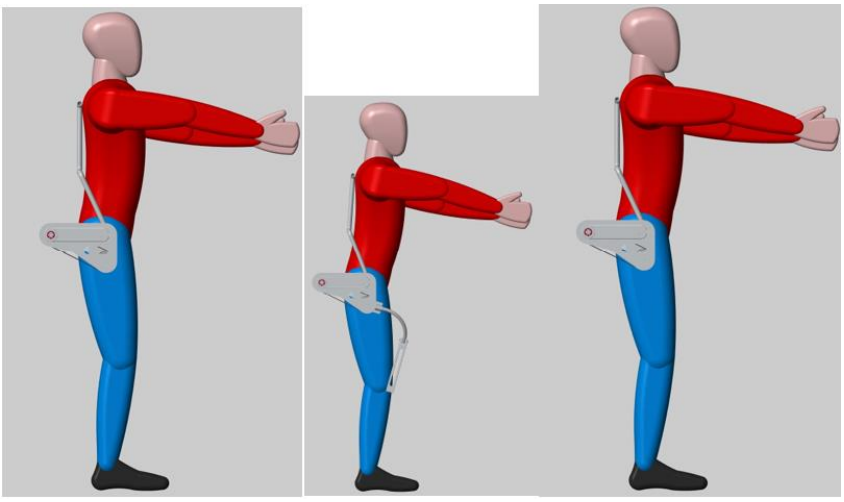


Figure 6.16:Moving sequence

Given these inputs and thanks to the simulation scheme it is possible to obtain the torque produced by the human being, the torque produced by the exoskeleton and the resulting torque of the system, characterized by the following trends:

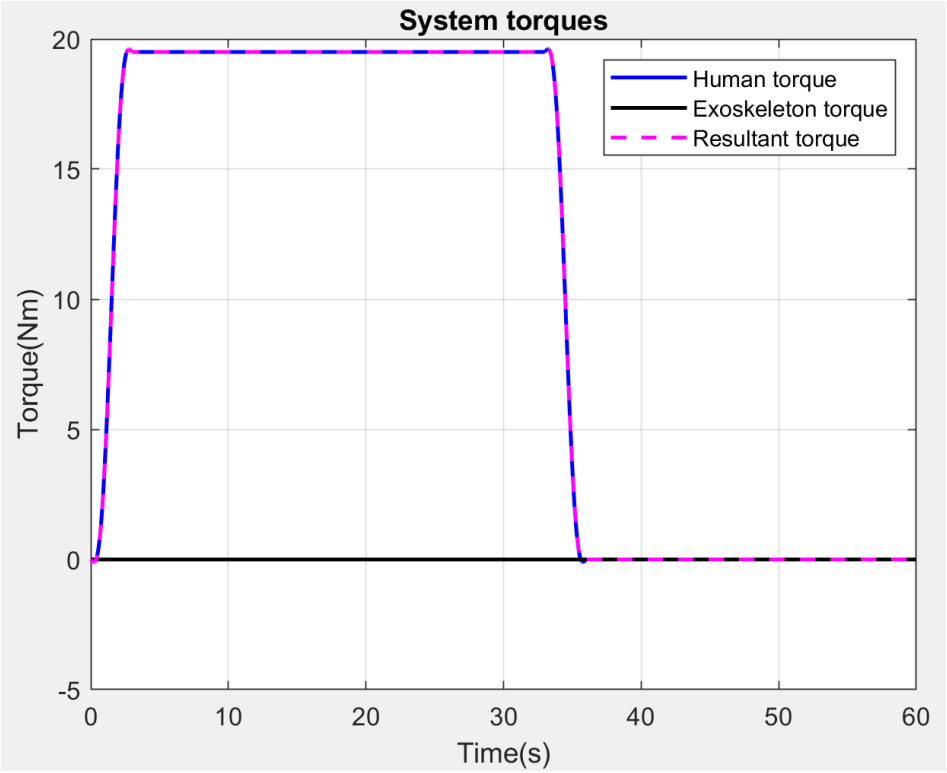



*Figure 6.17: System's torques*

As you can see from the picture above, the torque produced by the exoskeleton is zero for the entire working cycle because the angular position never exceeds 5°. The direct consequence of this behaviour is the equality between the torque produced by the human body and the resulting one of the systems. As for the final positions of the exoskeleton and the tracking error, the following trends are obtained.

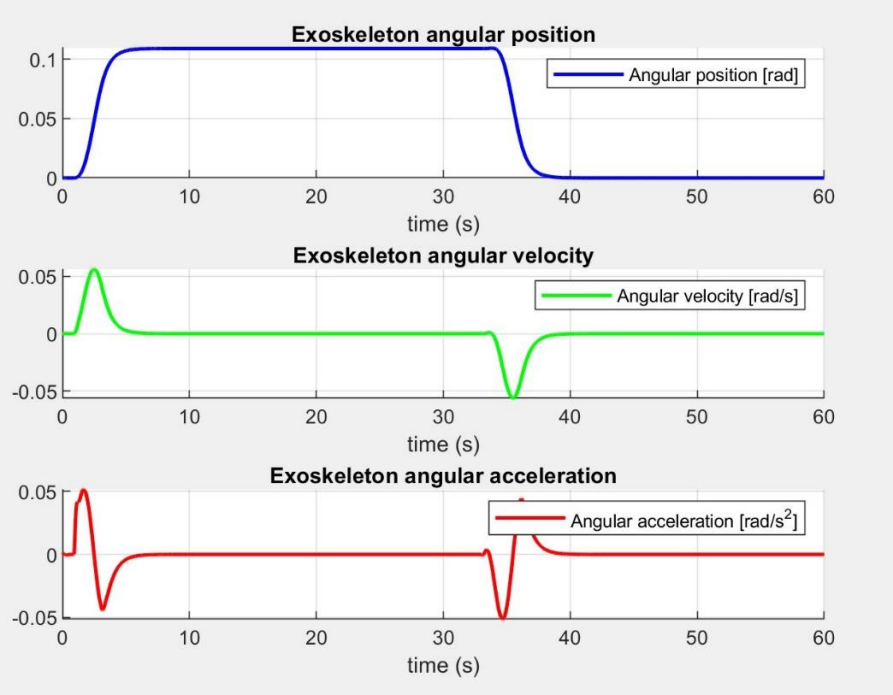



Figure 6.18: Exoskeleton quantities

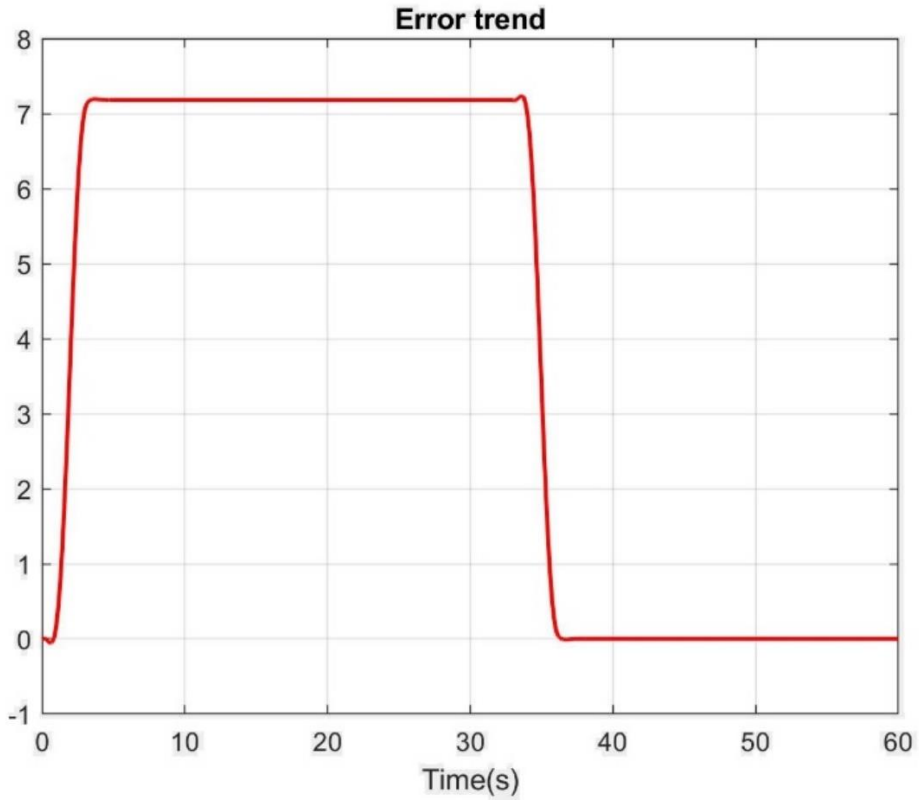



*Figure 6.19: Error trend*

### 6.1.4 Fourth simulation: the emergency case

Until now we have analysed cases in which everything worked correctly but, as it has been described in chapter 5, the StateFlow diagram is able to manage also the emergencies with consequent imposition to the exoskeleton to produce a torque equal to zero. The inputs that are provided are the following:

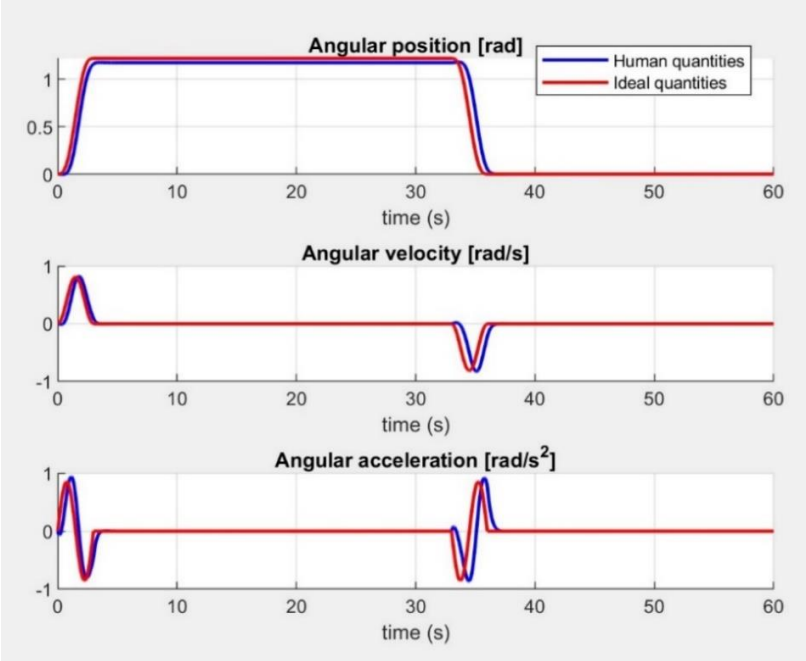



*Figure 6.20: Ideal quantities-Human quantities*

Regarding the sequence of movement, the results will be analogous to what is shown in the first simulation and consequently for the photographic vision of the movement we refer to figure 6.4. As for the speech related to the torque produced within the system this time we will observe a different trend because in the middle of the work phase was pressed the emergency button for a certain time, then the exoskeleton torque, immediately after pressing the button, it goes to zero and then return to the standard value once the button is released.

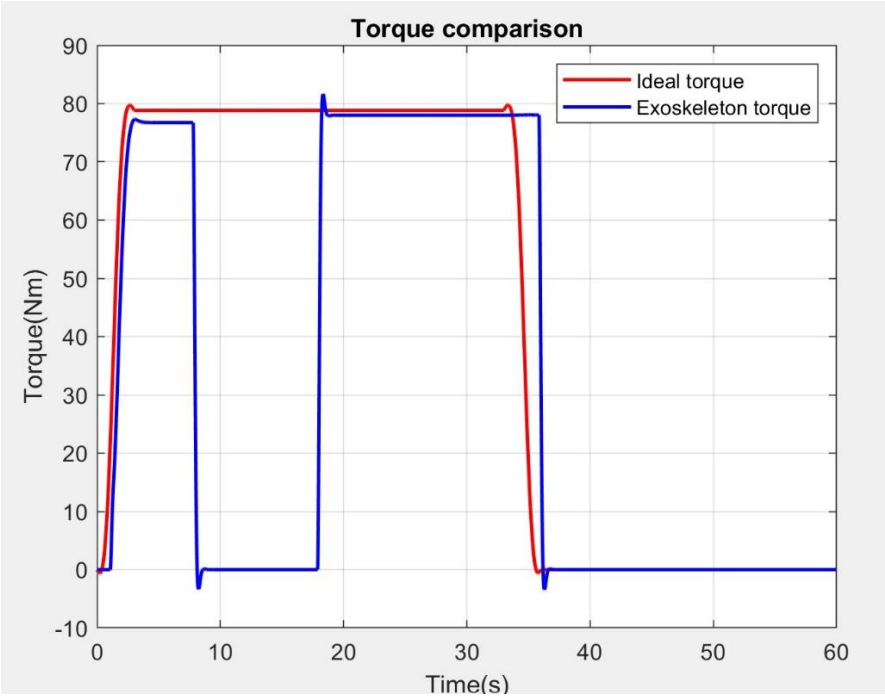



*Figure 6.21: Exoskeleton torque behaviour*

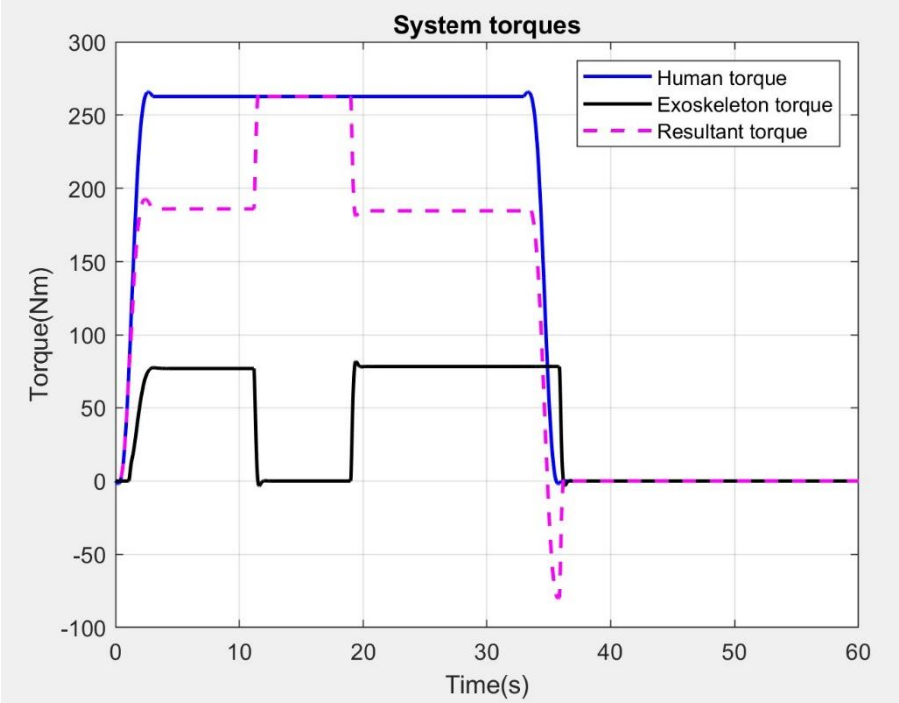



*Figure 6.22: System's torques*

Finally, for what regard the final positions of the exoskeleton and the tracking error, the following trends are obtained.

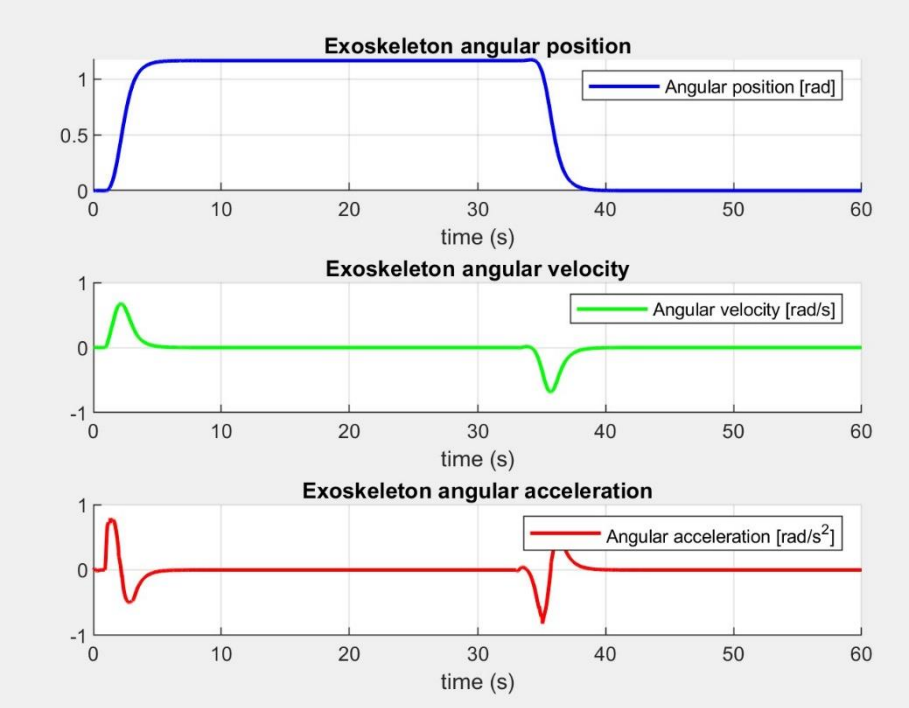



*Figure 6.23: Exoskeleton torque*

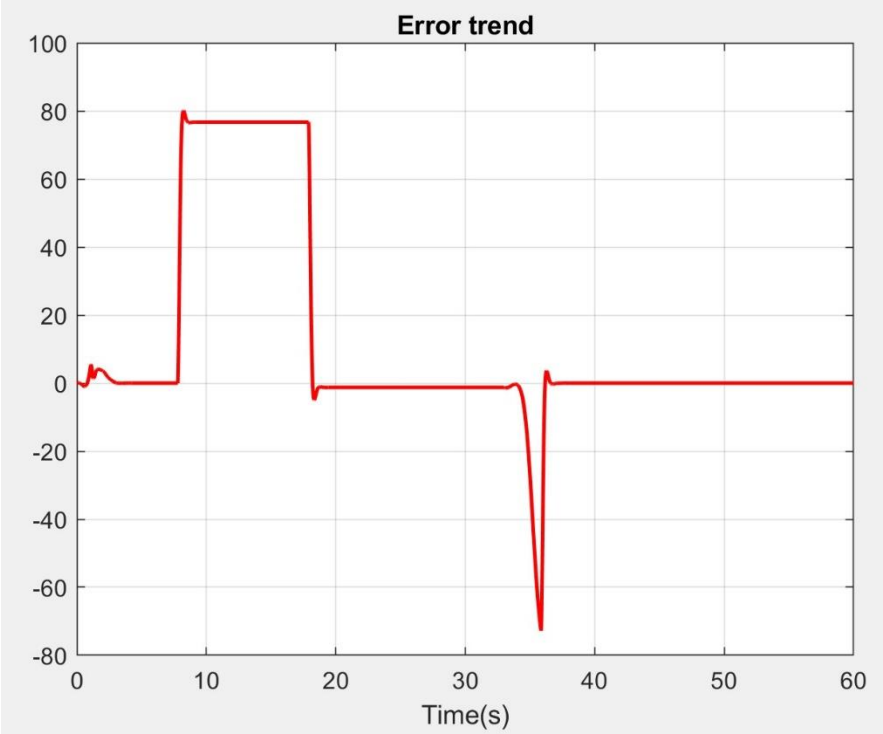



*Figure 6.24: Error trend*

# 7 Conclusion and future development

## 7.1 Conclusions

A study of the exoskeletons of the past and present has been carried out, analysing well everything that surrounds wearable robotics with an operational focus on the hip joint and that also involves the trunk. In addition to this, the drive system and the control system were analysed with interest. This research has led to a broad view of the state of the art, allowing you to set the main objectives for the design and optimal development of the work. As a reference subject for this study was chosen by legislation a human who was confirmations with the standard ISO 7250-2, with the aim of designing a device that could be worn by both male and female workers without any problem. The exoskeleton has been tested through a cycle of work defined by four phases: the phase of rest, in which the human is standing and does not perform any action, is then present the phase of bending in which the human body bends to reach the desired position, there is then the work phase in which the wearer keeps his position constant for a certain time and performs the action set and finally there is the rise phase that serves to return to the rest position to finish the work cycle. Once the work cycle has been defined and the mathematical model sketched out in previous works has been perfected, a study has been made of the possible controllers to find the most suitable for our device. The PID, LQR and MPC controllers were compared, considering the absolute error and the step response, and the choice fell on the PID which is then the most used controller in industrial applications. Having defined the type of controller to be used, the model of the human body and the exoskeleton has been exported by Solidworks in Simscape, a MATLAB tool that allows simulating 3D mechanical systems, simulating gravity and measuring speed and torque at the joints. Finally, a StateFlow diagram was designed, a MATLAB tool that defines a graphical language that includes state transition diagrams, flow charts, state transition tables, and truth tables. Through StateFlow various variables have been managed and elaborated to simulate situations similar to reality as the

button for the emergency, the presence of the supply and the oscillations due to the walking of the human being which must not lead to a production of support torque for the exoskeleton. It should be remembered that the exoskeleton must produce 30% of the human muscular torque needed to perform a bend. The simulations carried out test the operation of the device in different situations and state that the system is able to return the torque values sought through the use of the pneumatic actuator and to meet all the criteria of displacement, speed and torque. The objective of the project, in fact, was to design an implementation system able to perform the necessary actions required by man safely without speed and couples too high or dangerous for the user.

## 7.2 Future developments

The industrial pneumatic exoskeleton, presented in this paper by the candidate, is a prototype that was designed to improve user comfort by reducing muscle effort but, unlike those already on the market, is characterized by the presence of a second rotating joint. Since the device is still in the prototypical phase, it is proposed to follow some activities and analyses that could be further developed. The system can still be refined starting from the verification of the operation of the designed control logic. To do this, it is proposed to develop in the future a control program with the calibrated parameters in this elaborated through some device with interface normally used for these applications (e.g. Arduino, myRIO). Subsequently, the sensing functions within the revolute joints were used in the elaboration because the sensory part was not studied, this will certainly be a possibility for future development by going to decide the best location for the sensors and going to optimize the variables to be measured to ensure that the system can function properly. It is also possible to improve the dynamic model of pneumatic motor, thus making it possible to use discarded controllers as they require an accurate mathematical model (MPC). Finally, it must be realized and physically mounted in the laboratory and tested to see if the results obtained are consistent with those obtained through the

various software programs used. At this point a work like that carried out in this elaboration should be performed to simulate the behaviour of the exoskeleton when placed on the test bench previously designed in the past works.

# References

[1] [Online]. Available: https://www.istockphoto.com/za/vector/different-insects-gm513318037-47521890.

[2] [Online]. Available: https://insights.globalspec.com/article/1151/exoskeleton-technology-takes-a-step-forward.

[3] H. Kazerooni, «Exoskeletons for human performance augmentation,» in *Springer Handbook of Robotics*, Springer, 2008, pp. 773-793.

[4] M. K. Vukobratovic, «When were active exoskeletons actually born?,» in *International Journal of Humanoid Robotics*, vol 4, 2007, pp. 459-486.

[5] [Online]. Available: https://www.pinterest.it/pin/566468459366434220/.

[6] [Online]. Available: https://exoskeletonreport.com/2015/02/businesses-that-have-or-are-exploring-exoskeleton-products-in-alphabetical-order/.

[7] [Online]. Available: https://exoskeletonreport.com/product/muscle-suit/.

[8] [Online]. Available: http://www.spexor.eu/.

[9] [Online]. Available: http://hart.berkeley.edu/exoskeletons.html.

[10] [Online]. Available: https://www.wseit.edu.pl/en/news/presentation-of-a-new-rehabilitation-method-using-robotic-exoskeleton.

[11] [Online]. Available: https://www.militaryaerospace.com/computers/article/16726953/army-reaches-out-to-industry-for-new-ideas-on-exoskeletons-to-help-warfighters-lift-heavy-loads.

[12] [Online]. Available: https://www.engineering.com/AdvancedManufacturing/ArticleID/18274/Wearable-Robots-Exoskeletons-Coming-to-a-Workplace-Near-You.aspx.

[13] [Online]. Available: https://spectrum.ieee.org/the-human-os/biomedical/bionics/ford-assembly-line-workers-try-out-exoskeleton-tech-to-boost-performance.

[14] [Online]. Available: https://www.plasticstoday.com/automotive-and-mobility/carbon-fiber-chairless-chair-improves-ergonomics-audis-production-plants/30367266621892.
[15] [Online]. Available: https://www.digitalengineering247.com/article/exoskeletons-on-the-move.
[16] [Online]. Available: https://exoskeletonreport.com/product/arke/.
[17] [Online]. Available: https://www.pinterest.it/pin/301881981258604325/.
[18] J.-w. Gyoosuk!Kim, «Design and!Control of!a!Lifting Assist Device for!Preventing Lower Back Injuries in!Industrial Athletes,» 2019.
[19] B. P., Pneumatic Drives-System Design, Modelling and Control, Berlino: Springer, 2007.
[20] [Online]. Available: https://it.mathworks.com/.
[21] I. O. f. Standardization., ISO/TR 7250-2:2010 Basic human body measurements for technological design-Part 2: Statistical summaries of body measurements from national populations, 2010.