CONTROL SYSTEM DESIGN FOR EXOSKELETON OF THE RIGHT LOWER LIMB

PROJETO DO SISTEMA DE CONTROLE DE UM EXOESQUELETO DO MEMBRO INFERIOR DIREITO

MARLON WINSTON KOENDJBIHARIE

DISSERTAÇÃO DE MESTRADO EM SISTEMAS MECATRÔNICOS
DEPARTAMENTO DE ENGENHARIA MECÂNICA

FACULDADE DE TECNOLOGIA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA - UnB
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA MECÂNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS MECATRÔNICOS

CONTROL SYSTEM DESIGN FOR EXOSKELETON OF THE RIGHT LOWER LIMB

PROJETO DO SISTEMA DE CONTROLE DE UM EXOESQUELETO DO MEMBRO INFERIOR DIREITO

MARLON WINSTON KOENDJBIHARIE

Dissertação apresentada ao Departamento de Engenharia Mecânica da Faculdade de Tecnologia da Universidade de Brasília como parte dos requisitos necessários para obtenção do grau de Mestre em Sistemas Mecatrônicos

APROVADA POR:

Prof. Dr. Daniel Maurício Muñoz Arboleda, (UnB/ENM)

Prof. Dr. Gilmar Silva Beserra, (FGA/UnB)

Prof. Dr. Daniel Chaves Café (UnB/ENE)

The fear of the LORD is the beginning of wisdom,

and knowledge of the Holy One is understanding.

Proverbs 9:10

ACKNOWLEDGEMENTS

I thank the Holy One for this achievement and for always being present, in times of joy and in times of sorrow.

I thank my sweet and lovely wife Josmela for her unconditional support. I thank my children Roberta, Lyoni and Miguel for the privilege of being their father and the many nice memories.

I thank my parents Johanes and Marlene for being there for me when necessary and for the opportunity given to me to get an education.

I thank my brothers and sisters, Dennis, Elaine, Eurdice, Desmond, Earl and Melvin for a nice childhood spent together.

I thank my Advisor Dr. Daniel Muñoz for his guidance and support during the entire Graduate Programme and for giving me the opportunity to be part of the LEIA lab team at GRACO at the Universidade de Brasília.

I greatly appreciate the critical comments of the examining board, Dr Daniel Muñoz, Dr. Gilmar Beserra and Dr. Daniel Café. It has turned me into a more critical person regarding my work.

I would like to also thank Dr. Rudi van Els for the advice and support to me and my wife before and during my stay in Brazil.

I also thank Dr. Carlos Llanos for involving me in the activities of the LEIA lab and I also send “um grande abraço” to my colleagues Renato, Sérgio, Oscar and Joao of the LEIA lab. I really appreciated their presence and valuable tips during my time spent in the lab.

Last but not least I would like to thank Ambassador Marlon Mohamed Hoesein and Minister Counsellor Natasha Halfhuid of the Embassy of Suriname in Brazil, former Minister of Foreign Affairs of Suriname Winston Lackin for the opportunity given to support my family during my stay in Brazil.

RESUMO

Nesta pesquisa o modelo de um exoesqueleto do membro inferior direita para melhorar a mobilidade do usuário e seu sistema de controle foram desenvolvidos. O projeto físico do modelo do exoesqueleto consiste em três partes principais: um quadril e a parte superior e inferior da perna conectados um com o outro por juntas revolutas. Cada uma das juntas é atuado por um motor *Brushless DC* (BLDC) com caixa de redução para aumentar torque. Os motores a serem usados na construção possuem sensores de velocidade e de posição para fornecer os dados necessários para o sistema de controle. Solidworks *Computer Aided Design* (CAD) software é usado para desenvolver o modelo do exoesqueleto, que é salvo em formato *extensible markup language* (XML) para depois ser importado em Simmechanics, permitindo a integração de modelos de corpos físicos com componentes de Simulink.

A cinemática inversa do exoesqueleto é desenvolvido e projetado em *Very high speed integrated circuit Hardware Description Language* (VHDL) usando aritmética em ponto flutuante para ser executado a partir de um dispositivo *Field Programmable Gate Array* (FPGA). Quatro representações diferentes do projeto de hardware do modelo cinematico do exoesqueleto foram desenvolvidos fazendo análise de erro com *Mean Square Error* (MSE) e *Average Relative Error* (ARE). Análise de trade-off de desempenho e área em FPGA é feito.

A estratégia de controle *Proportional-Integrative-Derivative* (PID) é escolhido para desenvolver o sistema de controle do exoesqueleto por ser relativamente simples e eficiente para desenvolver e por ser amplamente usado em muitas áreas de aplicação. Duas estratégias de sistemas de controle combinado de posiçaõ e velocidade são desenvolvidos e comparados um com o outro. Cada sistema de controle consiste em dois controladores de velocidade e dois de posição. Os parâmetros PID são calculados usando os métodos de sintonização Ziegler-Nichols e *Particle Swarm Optimization* (PSO).

PSO é um método de sintonização relativamente simples porém eficiente que é aplicado em muitos problemas de otimização. PSO é baseado no comportamento supostamente inteligente de cardumes de peixes e bandos de aves em procura de alimento. O algoritmo, junto com o método Ziegler-Nichols, é usado para achar parâmetros PID apropriados para os blocos de controle nas duas estratégias te controle desenvolvidos. A resposta do sistema de controle é avaliada, analisando a resposta a um step input.

Simulação da marcha humana é também feito nos dois modelos de sistema de controle do exoesqueleto fornecendo dados de marcha humana ao modelo e analisando visualmente os movimentos do exoesqueleto em Simulink. Os dados para simulação da marcha humana são extraídos de uma base de dados existente e adaptados para fazer simulações nos modelos de sistema de controle do exoesqueleto.

Palavras-chave: ARE, controle de posição, controle de velocidade, controle PID, FPGA, FSM, modelo cinematico, MSE, PSO.

ABSTRACT

In this research a model of an exoskeleton of the right lower limb for user mobility enhancement and its control system are designed. The exoskeleton design consists of three major parts: a hip, an upper leg and a lower leg part, connected to one another with revolute joints. The joints will each be actuated by Brushless DC (BLDC) Motors equipped with gearboxes to increase torque. The motors are also equipped with velocity and position sensors which provide the necessary data for the designed control systems. Solidworks Computer Aided Design (CAD) software is used to develop a model of the exoskeleton which is then exported in extensible markup language (XML) format to be imported in Simmechanics, enabling the integration of physical body components with Simulink components.

The inverse kinematics of the exoskeleton model is calculated and designed in Very high speed integrated circuit Hardware Description Language (VHDL) using floating-point numbers, to be executed from a Field Programmable Gate Array (FPGA) Device. Four different bit width representations of the hardware design of the kinematics model of the exoskeleton are developed, performing error analysis with the Mean Square Error (MSE) and the Average Relative Error (ARE) approaches. Trade-off analysis is then performed against performance and area on FPGA.

The Proportional-Integrative-Derivative (PID) control strategy is chosen to develop the control system for the exoskeleton for its relatively simple design and proven efficient implementation in a very broad range of real life application areas. Two control system strategies are developed and compared to one another. Each control system design is comprised of two velocity- and two position controllers. PID parameters are calculated using the Ziegler-Nichols method and Particle Swarm Optimization (PSO).

PSO is a relatively simple yet powerful optimization method that is applied in many optimization problem areas. It is based on the seemingly intelligent behaviour of fish schools and bird flocks in search of food. The algorithm, alongside the Ziegler-Nichols method, is used to find suitable PID parameters for control system blocks in the two designs. The system response of the control systems is evaluated analyzing step response.

Human gait simulation is also performed on the developed exoskeleton control systems by observing the exoskeleton model movements in Simulink. The gait simulation data is extracted from a human gait database and adapted to be fed as input to the exoskeleton control system models.

Keywords: ARE, FPGA, FSM, kinematics model, MSE, PSO, PID control, position control, velocity control.

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

ALEX	Active Leg EXoskeleton
ALU	Arithmetic Logic Unit
ARE	Average Relative Error
BFO	Bacterial Foraging Optimization
BLDC	Brushless Direct Current
BLEEX	Berkeley Lower Extremity Exoskeleton
CAD	Computer Aided Design
CoP	Center of Pressure
CORDIC	COordinate Rotation DIgital Computer
D	Derivative
DC	Direct Current
DH	Denavit-Hartenberg
DOF	Degree of Freedom
DUT	Device Under Test
EDK	Embedded software Development Kit
ENE	Departamento de Engenharia Elétrica
ENM	Departamento de Engenharia Mecânica
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GA	Genetic Algorithm
GRACO	Grupo de Automação e Controle
HAL	Hybrid Assistive Limb
HC	Homogeneous Coordinates
HDL	Hardware Description Language
I	Integrative
IMU	Inertial Measurement Unit
IP	Intellectual Property
ISE	Integrated Software Environment
KNEXO	KNee EXOskeleton
LE	Logical Element
LEIA	Laboratory of Embedded Systems and Integrated Circuits Applications
Li-Po	Lithium Polymer
LOPES	LOwer-extremity PoweredExoSkeleton
LQR	Linear–Quadratic Regulator
MHz	Megahertz
MSE	Mean Square Error
P	Proportional
PC	Personal Computer
PD	Proportional-Derivative
PI	Proportional-Integrative
PID	Proportional-Integrative-Derivative
PSO	Particle Swarm Optimization
PWM	Pulse-Width Modulation
SoC	System on Chip

UnB	Universidade de Brasília
VHDL	Very High Speed Integrated Circuit Hardware Description Language
WSE	Walking Supporting Exoskeleton
XML	Extinsible Markup Language

CONTENTS

1. INTRODUCTION

In this research a model of an exoskeleton of the right lower limb and its control system are designed. The exoskeleton is designed for user mobility enhancement. Two approaches to the design of the control system are developed and compared to one another. The control system designs are then tested with human gait data in a simulated environment. In this section, the context of this research is presented.

1.1.PROBLEM DISCRIPTION

The designed exoskeleton model is developed for user mobility enhancement in persons that suffer from hemiplegia, the permanent paralysis of one side of the body. According to [1] the most common cause of hemiplegia is Cardiovascular Accident (CVA), popularly known as stroke. Stroke continues to be the leading cause of death and disability in the Brazil. Studies have indicated an annual incidence of 108 cases per 100.000 inhabitants [2].

This research is part of a project that is being developed in the LEIA lab (GRACO - Universidade de Brasilia) with the objective to create a low cost exoskeleton that will, among others, be adaptable to the user anatomy, host complex control algorithms and execute movements using PID control for the actuators of the exoskeleton. The control system will ultimately be developed as a System on Chip (SoC), taking advantage of the robustness of hardware design and the flexibility of implementation in software.

In particular, this research involves the design of an exoskeleton of the right lower limb and its control system. The exoskeleton design was developed to improve mobility to its wearer and, besides the skeleton structure, consist of a number of sensors and actuators. Among the components purchased for the construction of the exoskeleton are MAXTRON motors [3] and gears [4] with its respective driver for velocity control. However, when the exoskeleton is applied for mobility improvement, position control is necessary in order to maintain control of the gait. This fact shows the need to develop hybrid control of position and velocity for the exoskeleton motors, alongside the need to accelerate the calculation of direct and inverse kinematics in hardware.

To design the control system of the exoskeleton two models for combined position and velocity control were developed and tested in Simulink and compared to one another.

Other important aspects of this exoskeleton will be its low computational cost, while offering robustness, performance and flexibility for implementation of complex algorithms, attributes that can be met by a SoC. Implementation in hardware inherently offers robust design. Robustness is essential to prevent failure which may result in serious injuries or even worse. Another important aspect of the design is the possibility to parameterize the kinematics and dynamics control algorithms according to each individual user.

1.2.JUSTIFICATION

This research is primarily concerned with the control system design of a right lower limb exoskeleton. Essential parts of this exoskeleton are actuators, sensors and the control system. The control system will gather the necessary sensor data, such as motor axes angle and velocity for position and velocity control. This data is used by the control system of the exoskeleton to calculate error values for position control.

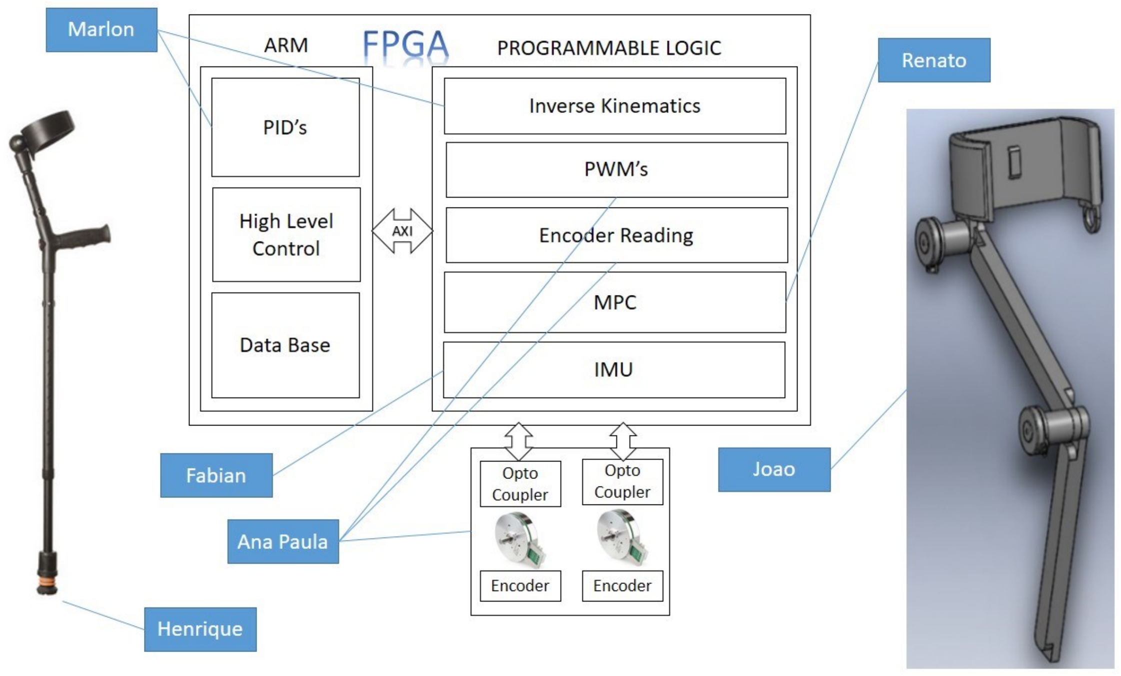


Figure 1.1 Exoskeleton SoC Implementation proposal

The exoskeleton is being developed by a team of research professors, graduate- and undergraduate students and will ultimately integrate the control system of the exoskeleton on a SoC on a Field Programmable Gate Array (FPGA). This choice allows exploration of parallel architectures for processing sensor data simultaneously, enhancing exoskeleton control. It also takes advantage of the parallelism of kinematics algorithms. Parallel implementation of the PID controllers is hereby also facilitated. Model Predictive Control (MPC) offers the ability to anticipate future events to take control actions accordingly. The exoskeleton system will consist of two BLDC motors equipped with gears, position and velocity sensors and left forearm crutch with activation buttons. To estimate gait phase the crutch will be equipped with an IMU and pressure sensor. For this purpose there will be an additional IMU for the left leg. An FPGA device with ARM processor will be used to host the PID controllers, inverse kinematics, dedicated IMU filter, PWM's and encoder reading module.

SoC's also have some other advantages. Besides the performance benefits, SoC's offer portability and low energy consumption. PC's, though capable of offering great performance and flexibility for implementation of complex algorithms, consume far more energy than microcontrollers and SoC's and are not portable. Microcontrollers on the other hand, are very energy efficient but have limited resources compared to PC's and SoC's. They also offer the possibility to prototype electronic circuits and can be programmed and reprogrammed as desired. An entire hardware design can be modified by only modifying the hardware design code making them very suitable to prototype complex systems. Equipped with processor they also enable hardware/software co-design.

1.3.OBJECTIVE

The general objective of this research is to design the control system of an exoskeleton of the right lower limb using FPGAs to accelerate the execution of the involved algorithms.

1.4.SPECIFIC OBJECTIVES

The specific objectives of this research are:

- Develop the kinematics model of the exoskeleton in Very High Speed Hardware Description Language (VHDL) analyzing the trade-off between bit-width, performance, precision and resource consumption on FPGA.
- The development of a model of the exoskeleton in Solidworks representing the exoskeleton as much as possible taking into account its measurements and degrees of freedom.
- Development of the control system in Simulink, comparing different strategies for integrated position and velocity control of the exoskeleton.

- Validation of the control system models by evaluating of the performance of the control system and simulation of the human gait movement.

1.5.METHODOLOGY

In this research the state-of-the-art in the design of existing lower limb exoskeletons is investigated, reviewing publications about exoskeletons that have already been developed. The focus is on lower limb exoskeletons that enhance autonomy for their users. This review gives valuable insight in the construction of exoskeletons from their early conception to implementation, testing and evaluation.

A Computer Aided Design (CAD) model of the exoskeleton is then developed in Solidworks. The kinematic model of the exoskeleton is then set up. The developed kinematic model is then imported in Simulink for the development of the control system of the exoskeleton and for human gait simulation.

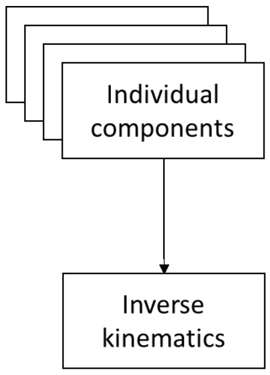


Figure 1.2 Bottom-up approach for inverse kinematics design

The synthesis of hardware components follows a bottom-up approach where the designer may develop custom components using VHDL. VHDL also allows the integration of IP's. Both floating and fixed point arithmetic can be customized for the SoC by using either IP cores or custom developed libraries offering a range of possibilities to optimize for performance and accuracy. In the case of the exoskeleton kinematics the operators used are floating-point IP components developed at the LEIA lab. The development of the inverse kinematics follows a bottom-up approach where parts of the system, such as the arctan2 and arithmetic operators using floating-point arithmetic are first developed to be integrated into the inverse kinematics design using a Finite State Machine (FSM) (Figure 1.1)

The kinematics model developed in VHDL can be validated using the output values of an automatic testbench. For the validation of the kinematics model of the exoskeleton designed in VHDL, the output results of the automatic testbench is compared and validated against the output of a kinematics model designed in Matlab, generating the Mean Square Error (MSE) and Average Relative Error (ARE) values. For this purpose the kinematics model is developed using four different bit widths which are then compared to one another.

The control system of the exoskeleton was designed in Simulink allowing efficient testing. For the control system Proportional–Integral–Derivative (PID) controllers are chosen for their relatively simple design the fact that they are used in most automatic process control applications in industry today to regulate flow, temperature, pressure, level, and many other industrial process variables. To adjust the PID parameters several approaches were used including Ziegler-Nichols and Particle Swarm Optimization (PSO). The performance of the control system design is validated using Simulink. To further validate the control system an existing human gait database was adapted to the exoskeleton design. This data is then fed as input to the exoskeleton design in Simulink enabling to observe the movements of the exoskeleton if it resembles the human gait.

1.6.CONTRIBUTIONS OF THIS WORK

Exoskeleton Inverse kinematics developed in VHDL for implementation on FPGA

Several research papers have proven that implementation of the inverse kinematics on FPGA yield good acceleration results in hardware. In this research the inverse kinematics for an exoskeleton was designed in VHDL to be implemented on an FPGA device using floating-point IP cores that were developed by the LEIA lab. Additionally the atan2 function, necessary for implementation of the inverse kinematics was also designed with these IP cores. Both atan2 and the inverse kinematics were designed using FSM’s to efficiently reduce hardware area usage.

Exoskeleton model human gait simulation using human gait database

The human gait database from Winter [5] was used to generate gait data suitable to be used for gait simulation for the exoskeleton. The data used from the aforementioned database was the temporal angle position data of the upper and the lower legs to be used for position control of the exoskeleton leg. The human leg structure is considerably more complex than the proposed exoskeleton leg especially at the joints, and more specifically at the knee joint, making it impossible to directly map the human joint data onto the exoskeleton for gait simulation. Moreover, the reference frame data for the human gait model from Winter is completely different from that of the exoskeleton structure. The joint angle data from the Winter database was first mapped onto the joint angle data of the exoskeleton. The joint angle data was then converted from degrees into radians, and the reference frame of the Winter model was then mapped onto the reference frame of the exoskeleton. Using the direct kinematics of the exoskeleton the angular data was then converted into end-effector data. The generated data was suitable to be used to simulate human gait movement.

Development of object function for exoskeleton control system for tuning with PSO

Two control system models of the exoskeleton were developed. The models are based on combined position and velocity control models presented in academic literature. PSO and the classic Ziegler-Nichols approach were used to find the PID parameters in the first model. For the second model Ziegler-Nichols and manual fine tuning were used.

For tuning of the control system of the exoskeleton with PSO an object function that seeks to minimize the error value between the reference and the output signal, minimize the overshoot and the rise time was developed for each motor. The general object function used was the sum of the aforementioned object functions.

Conference Paper

With respect to this research a first paper entitled "Control System Design for an Exoskeleton of the Right Lower Limb" was submitted and accepted for the (Brazilian Congress of Mechanical Engineering) COBEM 2017 event of the 24th (Brazilian Association of Engineering and Mechanical Sciences) ABCM International Congress of Mechanical Engineering to be held in Curitiba, at the **Pontifícia Universidade Católica do Paraná** (**PUCPR**) in December 2017. **COBEM** is the **ABCM** International Congress of Mechanical Engineering which takes place every two years in a Brazilian city. The extended paper extract is included in the appendix.

1.7.DOCUMENT ORGANIZATION

This document is organized as follows. Chapter 1, the introductory chapter presents the scope of the research.

In chapter 2, Theoretical Review, the most important theoretical topics related to this work is out, presenting background information to the reader that may elicit the concepts, methods and other material that is exposed in the next chapters.

Chapter 3 presents the various design aspects of the exoskeleton and the design process that was used.

In the fourth chapter hardware design of the kinematics model of the exoskeleton and the hardware simulation results are presented, such as error and performance analysis.

Chapter 5 exposes the development of two control system strategies where the Simmechanics model of the exoskeleton is integrated with Simulink components in order to build the two control systems with combined position- and velocity control. The two strategies are compared to one another.

Chapter 6 is dedicated to conclusions and recommendations and Appendix A presents program code that was used during this research.

2. THEORETICAL REVIEW

In this chapter several exoskeletons involving lower limbs that have been developed will be highlighted. Design characteristics such as actuation, control system, hardware platforms and sensor usage are gathered into a comprehensive table that characterizes the exoskeleton being developed with respect to exoskeletons that have already been developed.

A short but extensive review about kinematic modelling is then presented, followed by a brief description of applications of implementation of kinematic models in FPGA from academic literature for acceleration in hardware.

In the final part of this chapter the applied tuning methods in this research PSO and Ziegler-Nichols are presented.

2.1.EXOSKELETONS

Exoskeletons, sometimes referred to as wearable robots, are devices that can be worn by its users to enhance physical performance, improve user autonomy, or aid in rehabilitation. Though dozens of exoskeletons and prototypes of exoskeletons have been developed since the 1960's, it is only in the beginning of the early 1990's that considerable advances have been made in this area [6]. But still, research in this area is in its early development stage [7].

Dozens of efforts have been made to develop exoskeletons and prototypes of exoskeletons. Papers [8] and [9] have made reviews of several exoskeletons, each of them highlighting different aspects, the first reference focusing on detailed mechanical properties and the second reference focusing on a wider range of properties.

The number of exoskeleton and exoskeleton prototypes is very extensive. For this review exoskeletons that have been described in scientific resources have been chosen. Furthermore the resources were also chosen to be relatively recent, of year 2000 and above. The results are summarized in table 2.1 not following any particular order. Table 2.1 shows general information about each exoskeleton giving insight in the overall context of each exoskeleton including where it was developed, what it is used for and its overall structure. Most of the exoskeletons are adjustable and are made of lightweight material. The exoskeletons for gait rehabilitation are normally fixed structures on a treadmill requiring no special limitations on energy consumption.

Not all characteristics of the exoskeletons are available in the reviewed documents. This is caused by the difference in focus between the papers that were reviewed. This is why a number of the fields in table 2.1 are empty, which does not mean that the feature is not present. The actuators and sensors present on an exoskeleton are a measure for the complexity of its control system. Of the reviewed exoskeletons the BLEEX and the HAL exoskeleton are among the most technologically advanced. In the last row of table 2.1 the exoskeleton to be build is also presented in red.

Table 2.1 Description of existing lower limb exoskeleton

Exoskeleton/Year publication	Application	Bodypart	External support	Adjustable	DOF's	Control	Actuators	Sensors
LOPES Lower extrimity powered exoskeleton University of Twente/2007	Gait rehabilitation	Lower limbs	Yes: actuated support located at pelvis height	Yes	8:4 per leg	External PC	Elastic and bowden cable for all rotary joints; open-loop force controllable linear actuator; Linmot POl-37X240; Berger Lahr SER3910; Kollmorgery Danaher AKM22C servomotors	Force sensor; electromyography (EMG) measurements of the actication patters of eight major leg muscles; PIT-VZ4000; tracking motion with camaras
ALEX II Active leg exoskeleton University of Delaware/2011	Gait rehabilitation	One leg, right or left	Yes: back support	Yes: left, right, user measurements	4	DSPACE 11034 control system	Cables and pulleys; two Kollmorgen ACM22C2 rotary motors	Encoder or ankle (joint angle); three interlink electronics FSR 4065 pressure sensors (offline processing)
BLEEX Berkeley Lower extremity exoskeleton University of California, Berkeley/2006	Force augmentation	Lower limbs	No		14: 7 per leg	PC104 compliant computer; Network of RIOMs with ADCs and FPGA's; Three FPGA control units	Servo valves; linear hydraulic actuators	Encoders; ecceleroemeter; joint angle, angular velocity and angular acceleration; load distribution sensor (operator weight distribution); indinometer (overall orientation in relation to gravity); single axis force sensor; foot switches
WSE Walking supporting exoskeleton/2014	Autonomy	Lower limbs	Yes: underarm crutches	Yes	4: 2 per leg	ATHM800 (VIA Mark processor at 800MHz) PCI104; Adaptive network based fuzzy logic control (ANFLC); preprogramme d motion control (PMC)	DC servomotors; Vexta AHX5100KC servomotor 24V; Limit switches	Tekscan A201 model flexifore (force sensors in footsole); Hall effect sensors pre-integrated in the DC motors (angle)
HAL-3 Hybrid assistive limb third generation University of Tsukuba/2003	Autonomy	Lower limbs	No		4: 2 per leg	PC Celeron 566MHz RT Linux; PD controllers in hip and knee joints; Phase sequence control	DC servomotors; gear	Rotary encoder; EMG sensor; Floor reaction force sensor
KNEXO Knee exoskeleton Vrije Universiteit Brussel/2010	Gait rehabilitation	One leg	Yes: supportive arm	Yes	2	National Instruments PCI-6229 data acquisition board; impedance control; PID control	Pleated pneumatical artificial muscles (PPMA's); Kolvenbach KPS3/400 pressure regulating valves	Avagotech AEDEA 3300-TE1 hihg resolution incremental encoder; gauge pressure sensor; force-sensitive resistors (footsole)
HAL with instrumented cane Hybrid assistive limb University of Tsukuba/2014	Autonomy	One leg	Yes: instrumented cane		2	PC Celeron 566MHz RT Linux; synergy based control		Floor reaction force sensor in foot and cane; Inertial motion sensor
Mina Institute for Human and Machine Cognition (IHMC)/2011	Autonomy	Lower limbs	Yes: forearm crutches (optional)	Yes	6: 3 per leg	Embedded computer system (running Solaris); Control software in real-time java; PD control	BLDC motor (MoogBN 34-25EU)	Encoder (Avago HEDL-5640#A13); Renishaw RGH-24 linear encoder
[No Name] Yonsei University/2013	Autonomy	Lower limbs	No	No	14: 7 per leg	BLDC motor controller	BLDC motors	Force sensors; Inclinometer; angle sensors; potentiometers
HAL-5 Hybrid assistive limb fifth generation University of Tsukuba/2006	Force augmentation; user mobility enhancement	Upper and lower limbs	No				DC Motors	EMG signals; floor reaction force sensors
LEIA Exo (yet to be build) Universidade de Brasília	User mobility enhancement	Right/left lower limb	Yes: cane with activation button	Yes	2	PD/PID control system on FPGA	BLDC motors	Inertial Measurement Unit (IMU); encoders; FRS

The BLEEX exoskeleton (Figure 2.1) is a lower limb exoskeleton that was developed for strength, endurance and weight carrying augmentation. It uses a PC104 compliant computer and has seven DOF's on each leg [10], [11]. Of the reviewed exoskeletons it is the only one that has two different energy sources. Its hydraulic actuators are powered by a combustion engine, while the electronics are powered by a battery. The control system consists of an exoBrain that manages the exoskeleton's communication network, Remote IO Modules (RIOM's) are interconnected by Supervisory IO Module (SIOM) boards and there is also an external Graphical User Interface present for monitoring and configuration of the exoskeleton. The complex custom designed network infrastructure is designed for speed of communication and reduction of wiring. There is also one transceiver and one FPGA control unit for every RIOM and SIOM. The BLEEX exoskeleton was developed by the University of California Berkely.

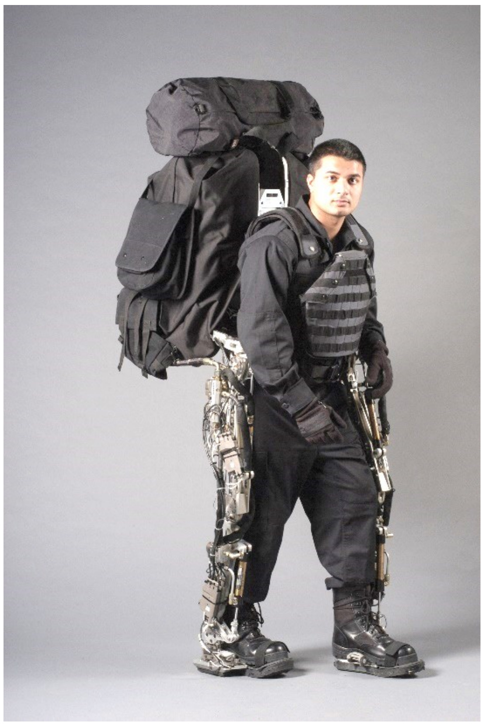


Figure 2.1 BLEEX exoskeleton (year 2006) [10]

The LOPES exoskeleton (Figure 2.2) is a lower limb exoskeleton with eight DOF's that are electrically powered, designed for rehabilitation. The exoskeleton is fixed on a structure while the user walks on a treadmill [12]. It is driven by Bowden and elastic cables and servomotors, and is controlled by an external PC. It can be operated in patient-in-charge and robot-in-charge mode. The exoskeleton was developed by the University of Twente in the Netherlands.

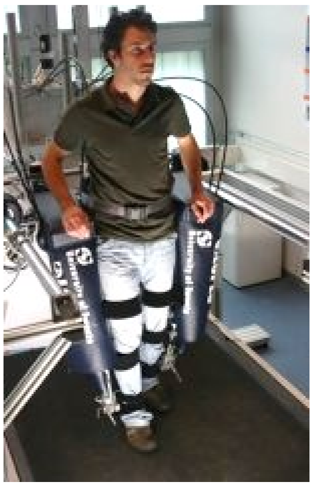


Figure 2.2 LOPES exoskeleton (year 2007) [12]

The ALEX II exoskeleton (Figure 2.3) was designed for gait rehabilitation and can be worn on either the right or left leg while being adjustable to the user's size. It is suspended on a back support structure bearing the weight of the user while walking on a treadmill [13]. It is powered by the electricity network and controlled by a dSPACE 1103 [14] control system. dSPACE is a company specialized in the development of control systems for automotive, aero spatial and industrial control. The exoskeleton was developed by the University of Delaware.

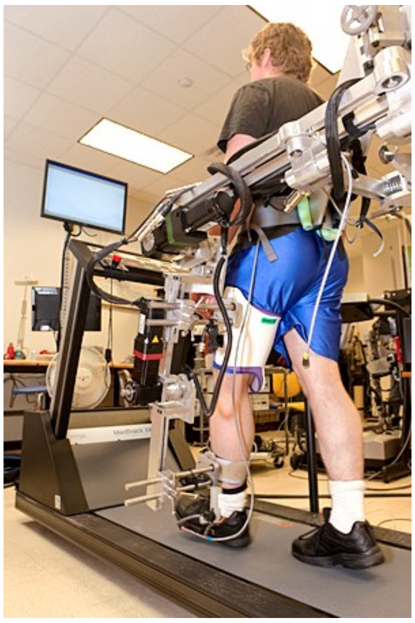


Figure 2.3 ALEX II exoskeleton (year 2011) [13]

The WSE (Figure 2.4) is a lower limb exoskeleton to aid in autonomy, designed with the idea of using lightweight material for user comfort and energy efficiency. It is used with underarm crutches and actuated by servomotors. The processing is performed by a VIA Mark

Processor at 800MHz. It uses two 23.5 V10 Ah Li-Po battery packs [15]. The device was developed by the Necmettin Erbakan University in Turkey.

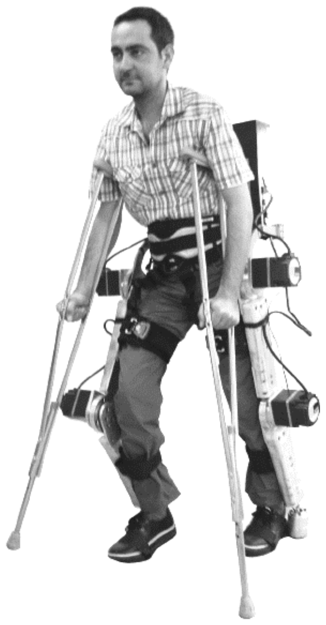


Figure 2.4 WSE exoskeleton (year 2014) [14]

The HAL exoskeleton (Figure 2.5) is among the most versatile ones. There are several versions of this exoskeleton available, ranging from complete exoskeletons of the upper and lower limbs to exoskeletons of only one lower limb, some for strength augmentation and others to promote autonomy. The authors provide a detailed description of the tools and methods in the design of the device [16].

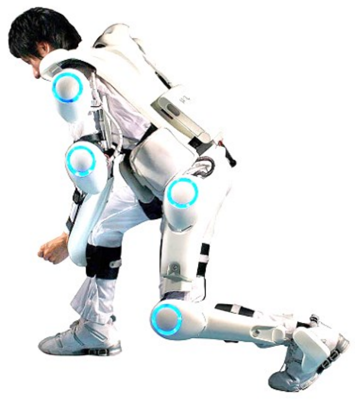


Figure 2.5 HAL exoskeleton (year 2003) [15]

The HAL-3 [17] is a lower limb version of the HAL exoskeleton, designed for patients with hemiplegia. It aids the user in standing up, walking and climbing stairs. These tasks are divided in a number of phases allowing phase sequenced control. The device is powered by battery and has a PC Celeron 566 MHz RT Linux based processing system. The HAL exoskeleton was developed by Japan's Tsukuba University and the robotics company Cyberdyne.

The HAL version with instrumented cane (Figure 2.6) was designed to promote autonomy for hemiplegic patients. The cane is equipped with force sensors on the bottom [18], and senses angle and velocity. It is equipped with a command button to start or stop the gait cycle. The cane is also equipped with the main unit and an Inertial Measurement Unit (IMU). The main unit receives the aforementioned sensor data from the cane, the sensor data from the IMU's on the thigh and shank, and from the force sensor data from the bottom of the shoes by Bluetooth. The data is then processed on the IMU, and the control commands are communicated through Bluetooth to the Wi-Fi unit on the back of the exoskeleton for its control.

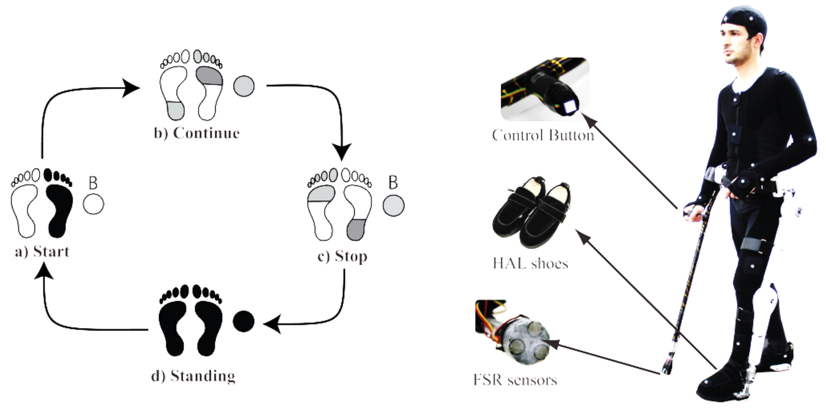


Figure 2.6 HAL exoskeleton with instrumented cane (year 2014) [17]

The KNEXO exoskeleton (Figure 2.7) is a single leg exoskeleton designed for gait rehabilitation. It is suspended by a supported arm while the user walks on a treadmill. What is particular about this exoskeleton is the actuator system composed of pleated pneumatic artificial muscles connected to a pressurized air supply system [19]. The exoskeleton was developed by the Vrije Universiteit Brussel in Belgium.

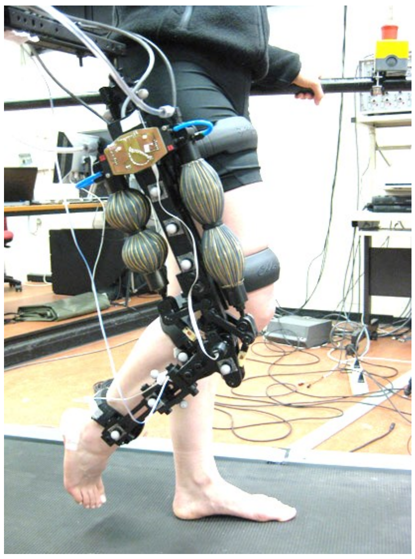


Figure 2.7 KNEXO exoskeleton (year 2010) [18]

Mina (Figure 2.8) is a lower limb exoskeleton designed to aid autonomy in paraplegia and paraparesis, optionally supported by forearm crutches [20]. The processing in this device is performed by an embedded computer system running Solaris, and the control software is written in real-time java. The actuators are brushless DC motors (BLDC) type Moog BN 34-25EU. Position and torque control are achieved with PD control. Mina was developed by the NASA Johnson Space Center and IHMC Robotics.

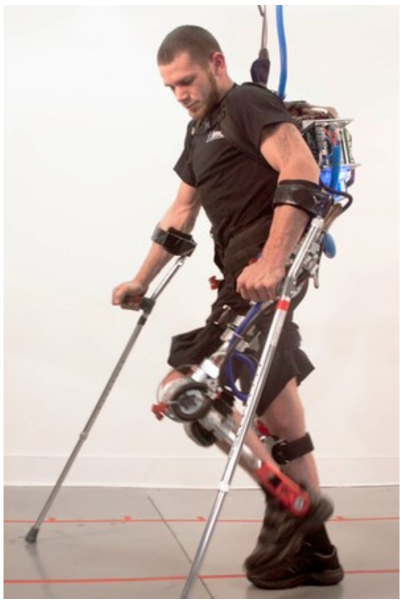


Figure 2.8 Mina exoskeleton (year 2011) [19]

An assistive exoskeleton for the lower limbs was also designed by the Yonsei University in South-Korea (Figure 2.9). The Center of Pressure (CoP) is used in the operation of this device to detect the human intention to walk and to verify stability [21]. The exoskeleton is driven by

Brushless Direct Current (BLDC) electric motors and a BLDC controller. The authors provide a detailed description of the tools and methods in the design of the device.

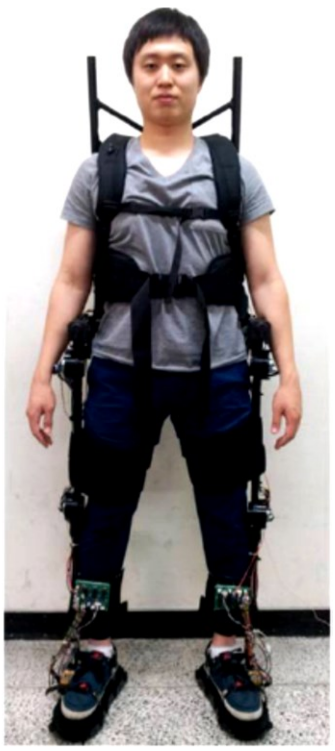


Figure 2.9 Yonsei University exoskeleton (year 2013) [20]

The design of exoskeletons is still in its early stages. Lots of research still needs to be done in this field. This is due in part to the many possible uses and design and implementation possibilities. Developments in technology, in the field of energy, mechanics, electronics, computer science, biomechanics, robotics and other related fields will also influence the developments in the field of exoskeletons.

In most of the designs of exoskeletons it is stated that it is possible to add devices to the exoskeleton if it is necessary from a therapeutic point of view. Care is also taken in most cases that the designs be adjustable to the user’s anatomy.

2.2.APPLICATION OF FPGA’S IN EXOSKELETON DESIGN

Rahman, et al. [22] the control architecture for a seven DOF upper limb robot is developed on a FPGA in conjunction with real-time PC (RT-PC). The control strategy used was to implement in FPGA the part that requires a higher sampling rate while the other part was implemented in RT. RT is a real time embedded controller developed by National Instruments (NI) [23]. The authors state that experiments have shown excellent tracking performance of the controller. The for the control architecture furthermore sliding mode control with exponential reaching law (ERL), a non-linear control strategy, was used.

Kumar et al. [24] presented the control architecture for GaExoD exoskeleton prototype that was developed using NI Lab VIEW, Robotics, FPGA and a RT (NI Real time embedded controller) module to promote shorter development time. Other important consideration is the real time and parallel processing of the control architecture, which, in this research is supported by a FPGA device. The main task of the FPGA device is to process the input information and update the actuator's position connected to the RT module by a high speed bus.

The BLEEX exoskeleton [25] is a very complex lower limb exoskeleton for force augmentation with autonomous energy supply. It uses a multivariable nonlinear algorithm for robust control behavior. The exoskeleton electronics system was designed to simplify and reduce cabling to sensors and actuators while a built-in FPGA manages data transaction and filtering.

In [26] the authors tend to prove the concept of designing a controller that is stand-alone, portable, programmable and easily maintainable using a prototype of a robotic arm. To meet these requirements a FPGA device is used and design is carried out using Verilog. The control system design is very simple using relays to activate actuators. Exoskeleton joint movements are provided using sensors near the joints of the human arm which are then transformed to digital signals. Based on the magnitude of these signals the FPGA provides the appropriate output signals to activate the relays that drive the actuators (DC motors). A commercially available robotic arm with five DOF's was used to build the prototype.

In [27] the hardware implementation of the control and interface between a master and a slave robot is designed using two FPGA's, one for collecting data from the master robot, basically a motion capture device, and the second one on the slave robot for controlling its motions. A total of twelve PID controllers were used, having each PID controller a total of 5 multipliers and 5 adders. Communication modules between the two robots using RS232 serial communication protocol, encoder counters for sensor data and PWM generators are also implemented on FPGA. The authors furthermore argue that FPGA's do not allow floating-point arithmetic and to overcome this, the smallest integer approximations were used for setting PID gains.

2.3.KINEMATIC MODEL

Kinematics is the branch of classical mechanics that describes the motion of points, bodies (objects) and systems of bodies (groups of objects) without consideration of the causes of motion [28]. In kinematics robots are modeled as chains of rigid bodies, connected to each other by joints that provide pure rotation and translation. The purpose of kinematics is to promote computer control, calculating forces and torques.

2.3.1.Forward kinematics basic concepts

Forward kinematics is concerned with determining the position of the end-effector of a robot, given the orientation of each of the consecutive links of the robot. Robot location can be expressed in any coordinate system, e.g. Cartesian, Cylindrical or Spherical. Orientation can be represented by a rotation matrix R (equation 2.1).

$$R = \begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} \tag{2.1}$$

The vectors *n*, *o* and *a,* are the unit vectors of the rotated system in the original reference frame. A rotation matrix, when multiplied by a vector, changes only the direction of this vector leaving the length of the vector unchanged.

In robotics practice, the movement *m* of a robot can be described in terms of a translation *t* and a rotation *R* (equation 2.2).

$$m=t+R \tag{2.2}$$

$$m=\begin{bmatrix} n_x & o_x & a_x & t_x \\ n_y & o_y & a_y & t_y \\ n_z & o_z & a_z & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

A more compact form to represent this movement can be achieved by the use of homogeneous coordinates, also called HC's (equation 2.3).

More complex movements (translations and rotations) can be expressed as a series of movements, i.e. several combined movements can be achieved by multiplying their HC's. HC's can only be applied to single joints. To express the movement of multiple joints, expressed as the succession of transformations, the Denavit-Hartenberg (DH) convention can be used.

Table 2.2 Example Denavit-Hartenberg representation

Link	Parameters	θ	d	a	α
1	θ1	θ1	d1	0	-pi/2
2	θ2	θ2	0	L2	0
3	θ3	θ3	0	L3	-pi/2
4	θ4	θ4	d4	0	pi/2
5	θ5	θ5	d5	0	-pi/2
6	θ6	θ6	d6	0	pi/2

The DH transformation between two successive joints *i-1* and *i* can be expressed by the following matrix:

$$T_{i-1}^{i}=\begin{bmatrix} cos\theta_i & -cos\alpha_i sin\theta_i & sin\alpha_i sin\theta_i & a_i cos\theta_i \\ sin\theta_i & cos\alpha_i cos\theta_i & -sin\alpha_i cos\theta_i & a_i sin\theta_i \\ 0 & sin\alpha_i & sin\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

Where,θ_i is the angle of rotation from X_{i-1} to X_i about the axis Z_{i-1},α_i is the angle of rotation from the Z_{i-1} axis to the Z_i axis about the X_i axis. Note that in figure 2.10 this angle is zero, a_i is the distance from the intersection of the Z_{i-1} axis and the X_i axis to the origin of the i^{th} coordinate system along the X_i axis, and d_i is the distance from the origin of the $_{i-1}th$ coordinate system to the intersection of the Z_{i-1} axis and the X_i axis along the Z_{i-1} axis and is the i joint variable for prismatic joints.

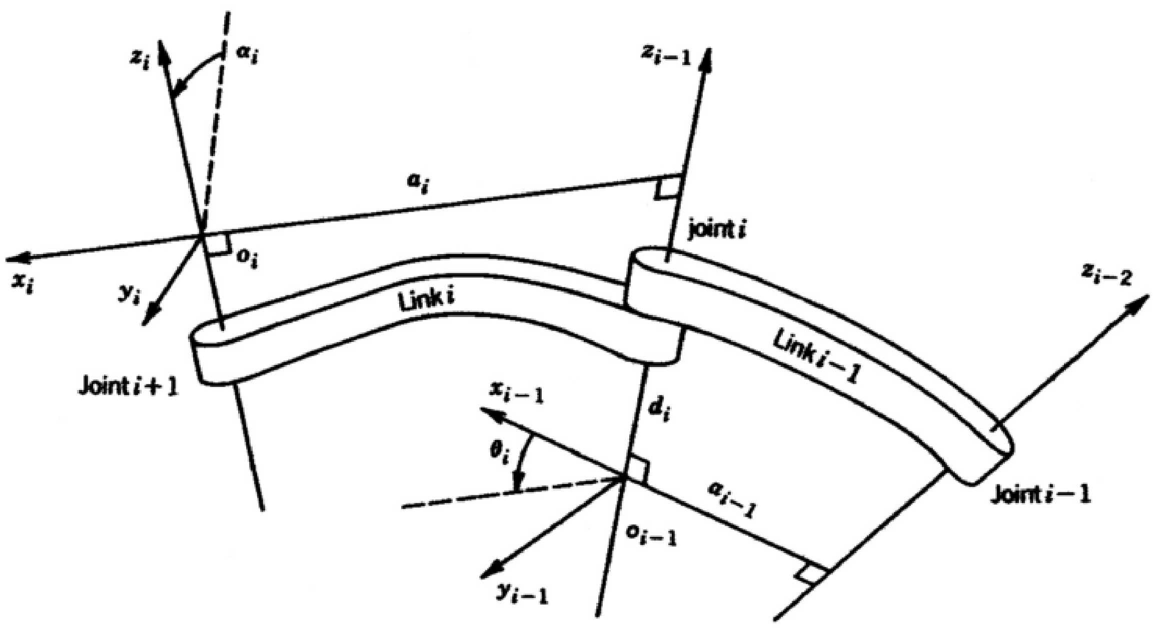


Figure 2.10 Coordinate system assignments with Denavit-Hartenberg convention [28]

For joints that allow only one degree of freedom such as the revolute and the prismatic joint can be represented directly by the transformation of equation 2.4. When there are multiple degrees of freedom single transformation matrices are combined as in equation 2.5.

$$T_0^n = T_0^1 T_1^2 \dots \dots . T_{n-1}^n \tag{2.5}$$

Rotation and translation can be extracted for any of the sub-transformations of T_0^n.

2.3.2.Inverse kinematics basic concepts

Inverse kinematics is concerned with resolving the position of the consecutive joints of a robot, given a desired end-effector position. This process is more difficult than forward kinematics and can have an infinite number of solutions, depending on the number of joints. To calculate inverse kinematics there are two approaches, the algebraic and the geometric approach [29]. End-effector position must thus be calculated departing from Cartesian space into joint position and orientation space.

In the geometric approach the end-effector position and orientation is calculated by trigonometry in terms of joint angles and lengths. This approach works well for simple robotic structures of up to two degrees of freedom (DOF's) with revolute joints in two dimensions.

For more complex structures the algebraic approach is applied. In this solution equation 2.5 is successively multiplied with the inverse transformation matrices $[T_{n-1}^{n}]^{-1}$. Some equations that are used in the trigonometric approach are also used in this approach. The solution of the inverse kinematics equation thus depends upon the robotic structure.

2.3.3.Kinematic model simplification

Equation 2.5 is the basis for the forward and inverse kinematic model. The above mentioned kinematic models are very complex in computational terms since they can involve a significant number of variables, depending on the structure of the robot. A robot with more than two revolute joint links is considered a complex structure. The addition, multiplication, sine and cosine operations add up to the computational complexity of these structures. Nevertheless, these kinematic models can all be dramatically simplified because in real-life applications some of the variables become constants.

The kinematic models are well susceptible to parallelism, allowing them to be efficiently implemented in FPGA's, providing high computational performance, and portability, which are basic requirements of the exoskeleton control system.

2.3.4.Kinematics in FPGA

In [30] the inverse kinematics for a ten DOF biped robot with angle equations that include the arctangent function by implementing the CORDIC algorithm in FPGA, offering a simplified method for reducing computational time and power consumption. The design of the inverse kinematics was spread over functional modules. Accuracy is tested comparing the FPGA results with a software based implementation. Best accuracy was found to be in the magnitude of 10e-4 for the angles.

In [31] the authors develop the inverse kinematics and the servo controller for a manipulator robot on FPGA. To reduce the usage of the logic elements (LE's) in FPGA, a finite state machine (FSM) was used in order to share the operators that implement the inverse kinematics. Simulations have also shown considerably faster performance in implementation in hardware than on a Nios II soft-processor.

In [32] a FPGA board with an Altera Cyclone IV FPGA chip was used to accelerate the position control of a parallel robot for milling. The kinematics equations were implemented in C using a Nios II soft-processor. The square root and the square root of the sum of squares operations were chosen to accelerate in hardware due to their high execution time. These custom designed operations became part of the Nios II Arithmetic Logic Unit (ALU) executing in the same way as the native microprocessor instructions. The calculation speed increased almost five times whereas the number of used logical elements increased by 11% and 65% for a first and a second set of accelerated hardware instructions.

2.4.ZIEGLER NICHOLS TUNING METHODS FOR PID CONTROL

The Ziegler-Nichols tuning method for PID controllers is the most widely used tuning method for PID controllers and was developed by John G. Ziegler and Nathaniel B. Nichols [33]. PID controllers are relatively simple controllers having only three parameters to be tuned and are widely used in many industrial applications. The method relies solely on the step response of the plant having no need to develop a model of the plant, thus making it relatively easy to apply. On the other hand it is only suitable for systems with monotonic step response [34]. The equation for the control signal of the PID controller is as follows:

$$u(t) = K_p \left[e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{d}{dt} e(t) \right] \tag{2.6}$$

Equation 2.6 presents the PID control model where *u(t)* is the control signal, *e(t)* is the error signal and K_p, T_i and T_d are the parameters to be tuned. From equation 2.6 it can be observed that the control signal is proportional to the error signal, the integral of the error signal to its derivative. The PID controller is able to eliminate the steady state error of the step response signal because of its integral action and it also has the ability to anticipate changes in the output (derivative action) [35].

The procedure for calculating the PID parameters using Ziegler-Nichols step response method is as follows [34]:

1. Obtain the step response of the plant
2. Draw the steepest straight line tangent to the response
3. Measure a and L as shown in figure 2.11

4. Calculate the parameters according to table 2.3

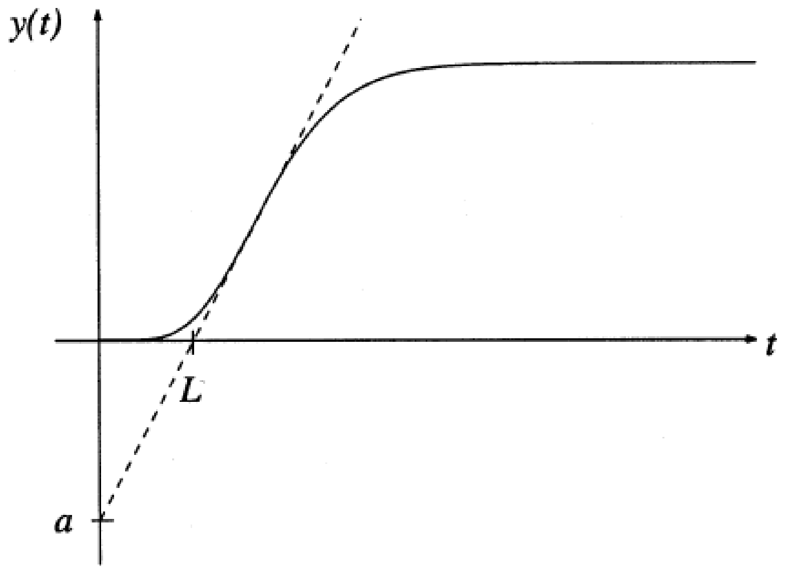


Figure 2.11 Parameters for Ziegler-Nichols step response method [34]

Table 2.3 P, PI and PID parameter values for Ziegler-Nichols step response method

Controller type	K_p	T_i	T_d
P	1/a		
PI	0.9/a	3L	
PID	1.2/a	2L	L/2

The procedure for calculating the PID parameters using Ziegler-Nichols self-oscillation method is as follows [36]:

1. Use PID P component only
2. Crank up P until oscillation
3. Determine the ultimate gain *Ku* and the ultimate period *Tu* (figure 2.12)
4. Calculate parameters according to table 2.4

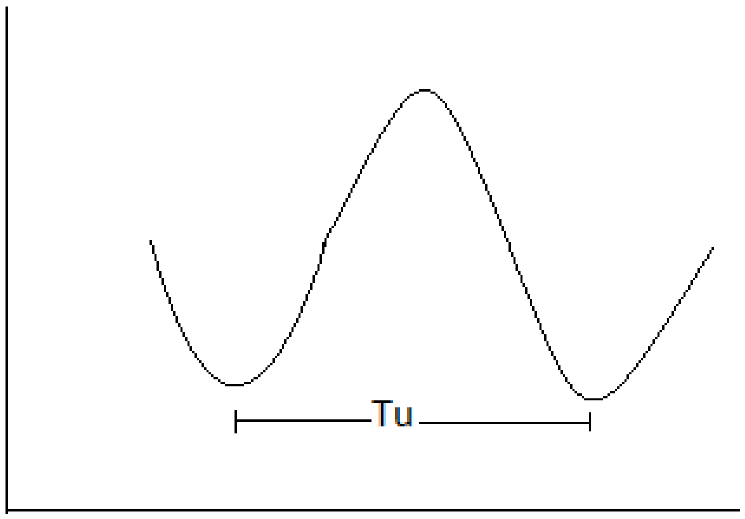


Figure 2.12 Parameters for Ziegler-Nichols self-oscillation method

Table 2.4 P, PI and PID parameter values for Ziegler-Nichols self-oscillation method

Controller type	K_p	T_i	T_d
P	0.5 Ku		
PI	0.4 Ku	0.8 Tu	
PID	0.6 Ku	0.5 Tu	0.125 Tu

2.5.PSO ALGORITHM

Particle Swarm Optimization is an optimization method based on the apparently intelligent social behavior of swarms such as schools of fish and flocks of birds. This method was proposed by Kennedy J. and Eberhart R. [37], [38]. In this algorithm, particles represent the individuals of the swarm as points in a multidimensional space, having no weight or volume. The particles are randomly initialized in a multidimensional space updating their velocity and position in each new iteration of the algorithm based on their own experience and the experience of the swarm. The position of a particle represents a potential solution to the optimization problem. Each particle evaluates its fitness based on an object function and has an individual memory by means of which it conserves its best position based on this function. Also, the swarm conserves the best global position of all individuals in every iteration. Based on these values each individual updates its position and velocity.

In the basic PSO algorithm with an N-dimensional search space and a number of S individuals, the i^{th} individual updates its j^{th} dimensional parameter according to the following two equations for velocity and position respectively:

$$v_{ij}^{t+1} = v_{ij}^{t} + c1\ U1_j[0,1](y_{ij}^{t} - x_{ij}^{t}) + c2\ U2_j[0,1]\ (y_{sj}^{t} - x_{ij}^{t})$$
$$x_{ij}^{t+1} = x_{ij}^{t} + v_{ij}^{t+1}$$

In these equations x_{ij} represents the current position of particle in the j^{th} dimension, y_i represents the best position of particle i and y_s the best overall position of all the individuals of the swarm. The constants c_1 and c_2 are the cognitive and social coefficients, representing the degree in which a particle relies on its own knowledge and the degree in which it relies on the collective knowledge of the swarm respectively. The constants c_1 and c_2 pull the particles towards the personal best and the global best positions respectively. Very low values cause the particles to roam further away of y_i and y_s while too high values will cause abrupt movements passing these regions. Based on past experience these values are often set to 2.0 [39], [40]. $U1$ and $U2$ are randomly distributed numbers between 0 and 1.

The basic PSO pseudocode is presented in figure 2.13. The particles are first randomly distributed across the boundaries of the N-dimensional space in which they exist, taking care that the position and velocity of each individual do not exceed x_{max} and v_{max}. If v_{max} is too small convergence may take longer and if it is too large the particles may fly too fast moving away from possibly good solutions. The values for $f(y)$ are also initially set to a vector of very high values that will be minimized during the optimization process. For the initial position and velocity values of each individual the object function value is then calculated and stored and y_s is then determined. In the nested for loops the velocities and positions of the individuals are updated for each dimension. The new fitness values are thus repeatedly calculated until they have reached a certain threshold value for the fitness function or the maximum number of iterations (Max_{iter}) has been reached. Every time the velocity and position of every particle is modified towards the personal best (y_i) and the global best position (y_s).

```
Inputs: S, N, c1, c2, $x_{max}$, $v_{max}$, $Max_{iter}$, threshold
Output: position of best individual x and its fitness f(x)
Begin
 Initialize Swarm
 repeat
   for k=1:S
     if $f(x_k) \le f(y_{ik})$ then
       $y_{ik} = x_k$
     endif
   endfor
   Calculate $y_s$ from the S fitness values $f(y_{ik})$
   for k=1:S
     for j=1:N
       $v_{kj} = v_{kj} + c1\ U1[0,1]\ (y_{kj} - x_{kj}) + c2\ U2\ [0,1]\ (y_{sj} - x_{kj})$
       $x_{kj} = x_{kj} + v_{kj}$
     endfor
   endfor
 until $f(y_s) <$ threshold
End
```

Figure 2.13 Basic PSO algorithm in pseudocode

2.6.PSO FOR PID PARAMETER OPTIMIZATION

PSO has been used in a number of PID parameter tuning applications. In [41] PSO is used for determining the PID parameters for velocity control of a BLDC motor designed in Matlab/Simulink. To determine the PID parameters of the controller PSO and Bacterial Foraging Optimization BFO techniques were applied to the BLDC motor design. For PSO, three dimensions, one for each PID parameter were used. Furthermore a swarm size of 50 individuals was used with a total of 100 iterations for performing PID parameter optimization. Simulation results have shown that both PSO and BFO can be used to determine PID parameters that perform efficiently for the controller, but with the parameters obtained through PSO the system presented better dynamic performance.

In [39] PID controller parameter optimization is performed using GA and PSO for industrial models. PSO has shown superior performance compared to GA. This paper among others presents the object function for PID parameter tuning with PSO.

In [42] Fractional Order PID controllers (FOPID), PID controllers and Fuzzy Logic Controllers (FLC) were optimized for trajectory control of a 2 DOF planar robot using PSO. In FOPID there are five parameters to be tuned, these are the usual PID controller parameters (K_p, K_i and K_d) and two additional ones λ and μ. The FOPID controller with its additional parameters adds flexibility to the controller design in achieving specified control objectives allowing real processes to be controlled more accurately. Since each FOPID controller has five parameters to be optimized, there are a total of ten dimensions in the case of the two DOF planar manipulator. The authors use three different cost functions to tune the FOPID controller, Mean of Root of Squared Error (MRSE), Mean of Absolute Magnitude of the Error (MAE) and Mean Minimum Fuel and Absolute Error (MMFAE).

In [43] the PID controller parameters for velocity control of a DC motor modelled in Matlab are tuned using PSO and compared with Fuzzy Logic Control (FLC). The authors argue that the frequency domain performance criteria IAE and ISE can result in relatively small overshoot but a long settling time because in these the error values are equally weighted independent of time. On the other hand Integral Time-weighted Square Error (ITSE) can overcome this problem but the formula is more complex and computationally more expensive. The authors instead use a time domain criteria that include the overshoot, the rise time, settling time and steady-state-error.

Nasri et al. [44] also presents a PSO-based approach to optimizing PID parameters for a BLDC motor simulated in Simulink. Compared to other control methods such as optimal control, variable structure control and adaptive control PID control offers the most simple and efficient form of control in many real world applications. Other methods for PID parameter optimization have been developed such as the LQR methods and GA-based optimization methods, the first one being computationally expensive and the second one presenting some deficiencies in object functions with highly correlated parameters. The authors argue in favor of PSO for its simple concept, easy implementation and computational efficiency. It was also found to solve optimization problems involving nonlinearity and non-differentiability, multiple optima and high dimensionality. A swarm size of 20 individuals and 20 iterations were used to execute the PSO algorithm. Comparison with the LQR and GA algorithms have shown that optimization with PSO presents superior dynamic performance over the two other optimization methods.

2.7.CONCLUSION

A wide range of exoskeletons have already been developed applying a very broad range of technologies for actuation, sensing and power. The emergence of new technologies has an impact on the development of exoskeletons. In the case of exoskeletons used for autonomy and force augmentation power supply and portability are very important issues. The field is relatively new and research is still in an early stage of development. Except from the exoskeletons presented in the review a considerable number of initiatives are under development for the construction of new exoskeletons.

The exoskeleton that is being developed at the LEIA laboratory will be specifically used for user mobility enhancement for persons who suffer from hemiplegia. For the control system PID control will be used. To host the control system, sensor data processing and motor drive a FPGA device will be used.

In the next chapter the exoskeleton physical design and kinematics model will be presented.

3. EXOSKELETON DESIGN

To enable simulation of the exoskeleton control system in Matlab, a design was created in Solidworks 2016. This Computer-aided Design (CAD) software package designed by Dassault Systèmes in 1995, utilizes a parametric feature-based approach to designing solid material models. The possibility to parameterize part measurements makes it relatively quick and easy to automatically adjust a design. In the case of the exoskeleton the lengths of the upper and lower leg are parameterized, allowing them to easily adjust the model measurements to the anatomy of a specific individual.

A Solidworks model is essentially a hierarchy of parts and assemblies of parts. The exoskeleton parts were designed with fictitious values which more or less resemble the measurements of a real individual. The parts were then mated together in such a way that they were able to move like a human leg with revolute joints in the hip and the knee. The exoskeleton leg model thus only moves in a plane.

3.1.PHYSICAL DESIGN

Figure 3.1 presents the proposed design for the exoskeleton. The model is comprised of an adjustable wearable hip with three parts that slide into each other. The hip and knee joints are each actuated by a motor with gear. For smoothness of movement bearings were added to the hip and the knee joints.

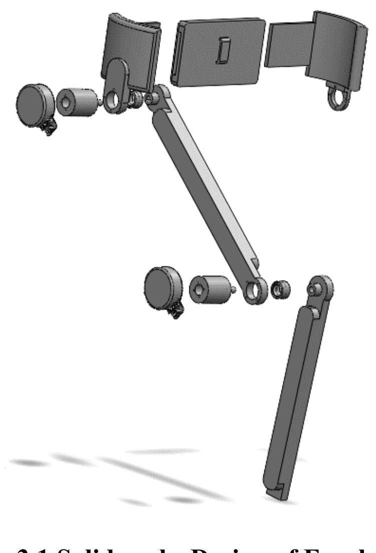


Figure 3.1 Solidworks Design of Exoskeleton

The Solidworks model was exported as an XML file, using the Simscape Multibody Link Add-in available from MathWorks [45]. The XML file was then imported in Matlab as a Simmechanics model (Figure 3.2). With Simmechanics physical systems can be designed from components, allowing them to integrate with Simulink blocks. The integration of Simmechanics components with Simulink blocks in this research has proven to be very useful to build and integrate the control system for the exoskeleton into the generated model.

The generated Simmechanics model can be visualized with Matlab's Mechanics Explorer. Mechanics Explorer offers a 3D visualization pane to view the model and a tree view pane to explore the model hierarchy. It also consist of a properties pane, presenting the parameters of each component of the design such as their weights and dimensions. The model can be visualized from any angle and can also be zoomed in or out.

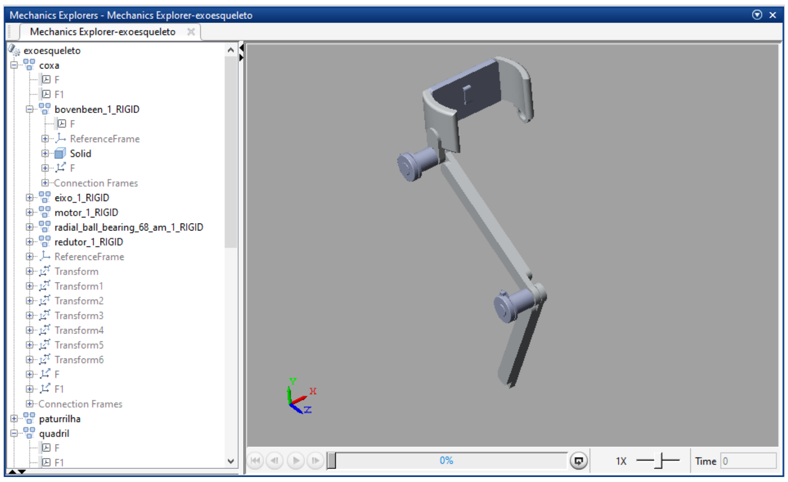


Figure 3.2 Exoskeleton Model imported visible in Mechanics Explorer

The Solidworks model (Figure 3.1) presents relatively significant detail including all mechanical components such as bearings, motors, gears, joints etc. For the purpose of designing and simulating the control system model less mechanical detail than present in the Solidworks model was necessary, and therefore the model was simplified, adding together and mating some components such as prismatic joints which were present due to the presence of bearings. On the other hand sensor outputs and actuator inputs from the motors were added to the two revolute joints of the model.

3.2.THE INVERSE KINEMATICS MODEL

Kinematics is the branch of classical mechanics that describes the motion of points, bodies (objects) and systems of bodies (groups of objects) without considering the causes of the motion [4]. In kinematics robots are modeled as chains of rigid bodies, connected to each other by joints that provide pure rotation and translation. The purpose of kinematics is to promote computer control, calculating forces and torques. The inverse kinematics calculates the joint positions and orientations given the end-effector position and orientation. To calculate the inverse kinematics there are two approaches, the algebraic and the geometric approach [29]. In the geometric approach the end-effector position and orientation is calculated by trigonometry in terms of joint angles and lengths. This approach works well for simple robot structures of up to two DOF's with revolute joints in two dimensions. To facilitate the calculation of the inverse kinematics a mathematical model of the exoskeleton is depicted in figure 3.3.

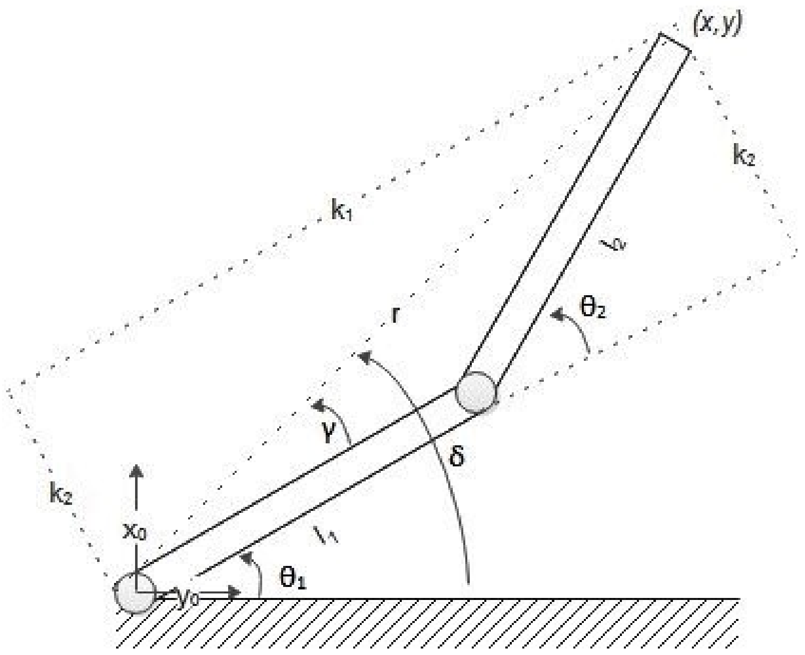


Figure 3.3 Simplified Exoskeleton Structure

The proposed exoskeleton consists of two links, one from hip to knee and the other from knee to foot, and two rotational joints, one at the hip and the other at the knee, each operating in the plane, therefore the trigonometric approach is used.

For better visualization the leg is positioned upside down (Figure 3.3) with reference frame *(x_0, y_0)*. The links l_1 and l_2, the upper and lower leg respectively, are connected by the knee joint. The hip joint is at the origin of the reference frame. Given the joint angles $\theta_1$and $\theta_2$of the hip and the knee joint, the end effector position *(x, y)* can be determined given the restriction that $\theta_2$is greater than 0° and no more than 180° in accordance with human leg anatomy. The Denavit-Hartenberg parameters of the exoskeleton are given in table 3.1.

Table 3.1 Denavit-Hartenberg parameters of the Exoskeleton

Link	**α**	**a**	**d**	θ
1	0	l_1	0	θ_1
2	0	l_2	0	θ_2

The inverse kinematics calculates the joint configuration (θ_1 and θ_2) given the end-effector configuration, i.e. the values of x and y which can be calculated by:

$$x = l_1c_1 + l_2c_{12} \tag{3.1.a}$$

$$y = l_1s_1 + l_2s_{12} \tag{3.1.b}$$

where c_1 and s_1 represent $\cos(\theta_1)$ and $\sin(\theta_1)$, and c_{12} and s_{12} represent $cos(\theta_1 + \theta_2)$ and $sin(\theta_1 + \theta_2)$.

From equations 3.1.a and 3.1.b follows that

$$x^2 + y^2 = (l_1c_1 + l_2c_{12})^2 + (l_1s_1 + l_2s_{12})^2 \tag{3.2}$$

$$= l_1^2.c_1^2 + l_2^2.c_{12}^2 + 2l_1c_1l_2c_{12} + l_1^2.s_1^2 + l_2^2.s_{12}^2 + 2l_1s_1l_2s_{12} \tag{3.3}$$

$$= l_1^2(c_1^2 + s_1^2) + l_2^2(c_{12}^2+s_{12)}^2 + 2l_1l_2(c_1c_{12} + s_1s_{12}) \tag{3.4}$$

$$= l_1^2 + l_2^2 + 2l_1l_2c_2 \tag{3.5}$$

Applying the trigonometric identity $cos(a - b) = cos(a)\,cos(b) + sin(a)\,sin(b)$, equation 3.5 can be derived from equation 3.4, finally yielding:

$$x^2 + y^2 = l_1^2 + l_2^2 + 2l_1l_2c_2 \tag{3.6}$$

To calculate the inverse kinematics our goal is to calculate θ_1 and θ_2. From equation 3.6 isolate

$$\cos(\theta_2) = \frac{x^2+y^2-l_1^2-l_2^2}{2.l_1l_2} \tag{3.7}$$

from which θ_2 can be directly derived by applying the inverse cosine function. For small angles however, this function is not very accurate.

$$sin(\theta_2)^2 + cos(\theta_2)^2 = 1 \tag{3.8}$$

$$\sin(\theta_2) = \pm\sqrt{1-\cos(\theta_2)^2} \tag{3.9}$$

Equation 3.9 corresponds to the two possible configurations for the planar manipulator, the elbow-up and the elbow-down configuration. In case of the exoskeleton leg only the elbow up configuration (Figure 3.3), where $\theta_2 > 0$ is relevant caused by the restrictions in the human knee joint. Thus the equation for $\sin(\theta_2)$ will be:

$$\sin(\theta_2) = \sqrt{1-\cos(\theta_2)^2} \tag{3.10}$$

$$\sin(\theta_2) = \sqrt{1-(\frac{x^2+y^2-l_1^2-l_2^2}{2.l_1l_2})^2} \tag{3.11}$$

$$\theta_2 = atan2(\sin(\theta_2), \cos(\theta_2)) \tag{3.12}$$

$$\theta_2 = atan2\left(\sqrt{1-\left(\frac{x^2+y^2-l_1^2-l_2^2}{2.l_1l_2}\right)^2}, \frac{x^2+y^2-l_1^2-l_2^2}{2.l_1l_2}\right) \tag{3.13}$$

where $atan2$ is the inverse tangent function. The $atan2$ function is different from the conventional arctangent function. $2.l_1l_2$The major difference is that the $atan2$ function also determines the quadrant of an angle, which is not the case of the arctangent function. To calculate $\theta_1$the angle γ and the lengths $k_1$and k_2 are introduced in figure 3.3.

$$\gamma = atan2(k_1, k_2) \tag{3.14}$$

$$k_1 = r.c_\gamma = l_1 + l_2c_2 \tag{3.15}$$

$$k_2 = r.s_\gamma = l_2s_2 \tag{3.16}$$

$$\theta_1 = \delta - \gamma = atan2(y,x) - atan2(l_2s_2, l_1 + l_2c_2) \tag{3.17}$$

3.3.CONCLUSION

The physical 3D Solidworks design of the exoskeleton offers a range of possibilities for simulation and ongoing development purposes. An XML model of the Solidworks design was generated using the Simmechanics add-in available at MathWorks. Simmechanics components generated by importing the XML file can be integrated with Simulink components making it possible to build control system models that can be used to perform simulations.

The exoskeleton being developed has a relatively simple structure with two DOF's. Nevertheless the calculation of the inverse kinematic involves some complex calculations involving the square root and *atan2* functions.

4. KINEMATICS MODEL ACCELERATED IN HARDWARE

The kinematics model, the control system, sensory data processing and actuation system of the exoskeleton will be implemented as a SoC on a FPGA. An important advantage of FPGA devices is portability in terms of weight and size. FPGA SoC's also present flexibility in system design integrating hardware components with software components, offering a number of support tools such as software/hardware co-design tools and interconnectivity with other tools such as Simulink.

Besides the performance benefits, SoC's offer portability and low energy consumption compared to PC's. PC's, though capable of offering great performance and flexibility for implementation of complex algorithms, consume far more energy than microcontrollers and SoC's and are not portable. Microcontrollers on the other hand, are very energy efficient but have very limited resources leading to a lack of computational performance for implementing complex algorithms as offered by PC's and SoC's.

The inverse kinematics of the exoskeleton presents a considerable amount of arithmetic and trigonometric calculations that need to be performed at a very high rate while the exoskeleton is in operation. Here, advantage can be taken of the parallelism capabilities of the FPGA while at the same time maximizing operator reuse as a trade-off. Thus the number of operators should be kept as low as possible to reduce FPGA resource usage.

In this research the floating-point precision of 27-, 32-, 45- and 64 bit VHDL hardware designs of the inverse kinematics of the exoskeleton are compared to one another. A 64 bit model of the inverse kinematics model in Simulink is therefore used as a reference model. Every VHDL hardware design is co-simulated with the Simulink model to calculate the Mean Square Error (MSE) [46] and the Average Relative Error (ARE) [47].

The precision analysis is performed by comparing hardware based inverse kinematics implementations in VHDL with a software based implementation in Simulink. For the kinematics implementation of the exoskeleton, the four bit-width representations were simulated and analyzed not only in terms of performance, but also in terms of FPGA resource consumption and power consumption.

4.1.HARDWARE IMPLEMENTATION OF THE ATAN2 FUNCTION

The atan2 function is not readily available so it was implemented using the units illustrated in table 4.1. The operators used for the design of *atan2* are IP floating-point operators developed at the LEIA lab. The *atan2* is a computationally expensive operator in terms of hardware area and machine cycles, thus a good candidate for design in hardware. For the implementation of the floating-point trigonometric functions the Coordinate Rotation Digital Computer (CORDIC) approach is used for its suitability and performance [48]. The CORDIC algorithm is a simple and efficient algorithm to calculate hyperbolic and trigonometric functions.

Without entering into further details, it is worth mentioning that the atan2 function is implemented as a Finite State Machine (FSM) with six states including the waiting state, reducing considerably the area in hardware. As can be seen in the table 4.1 it consists of only three operators. The optimized hardware model of the arctan2 function is given in figure 4.1.

Table 4.1 Number of Composing Operators for atan2 Operator

Floating-point Operator	Number of units
Arctangent	1
Divisor	1
Addition/Subtraction	1

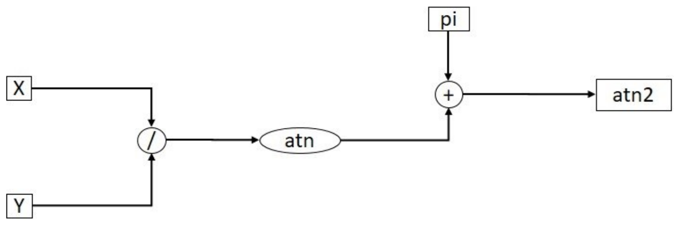


Figure 4.1 Optimized hardware model for atan2 function

4.2.HARDWARE IMPLEMENTATION OF INVERSE KINEMATICS

The hardware design of the inverse kinematics model has been carried out in VHDL using the Xilinx Integrated Software Environment (ISE) 14.7 development tool. The design consists of the several floating-point operators that were also designed in VHDL.

As stated by equation 3.17, the rigth-hand side of θ_1 is dependent on the value of $\sin(\theta_2)$ and $\cos(\theta_2)$ and can only be calculated after θ_2 is known. The calculation of the inverse kinematics can be drastically optimized by introducing constants stored in memory as illustrated by the input rectangles in figure 4.2. To save space on the SoC the least possible number of operators is used while maximizing their use in every cycle.

For the same reason as the *atan2* function the inverse kinematics is implemented in hardware as a FSM with twelve states including the *waiting* state, as illustrated in figure 4.3. The *waiting* state is a start and also an end state.

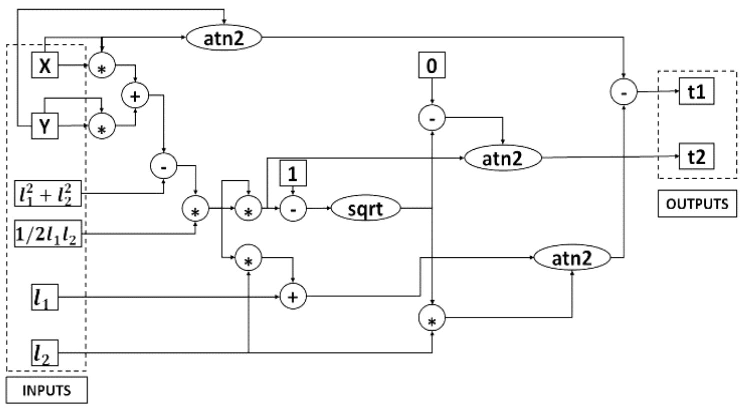


Figure 4.2 Optimized hardware model for calculating the Inverse Kinematics

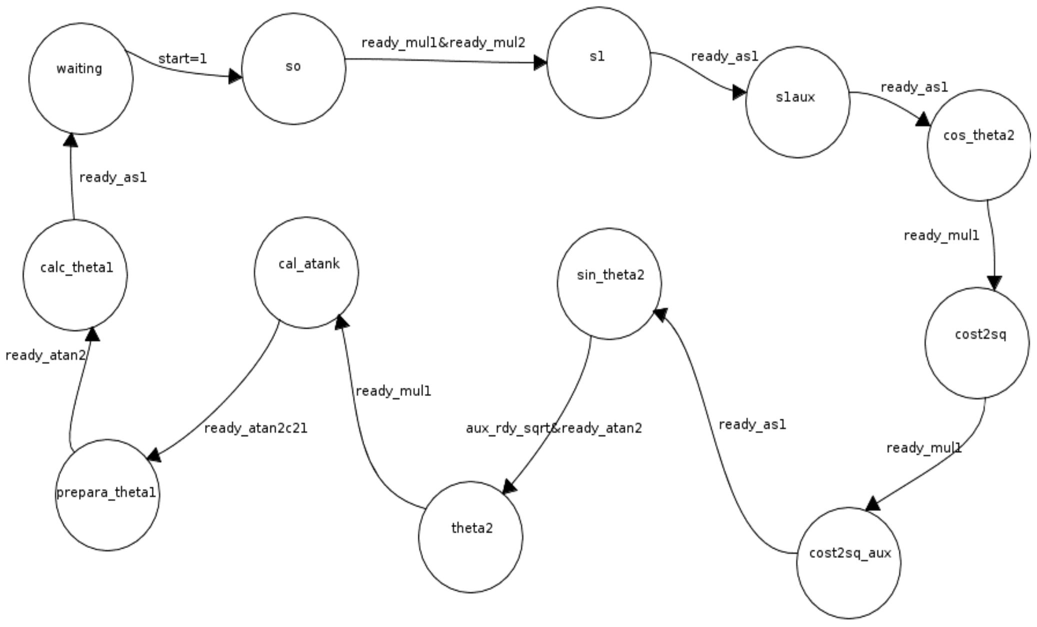


Figure 4.3 Finite State Machine for calculating the Inverse Kinematics

The inverse kinematic unit has two inputs, x and y, and two outputs, θ_1 and θ_2 (*t1* and *t2* in figure 4.2). All other input constants such as l_1 and l_2 are previously calculated and stored in the system, where l_1 and l_2 are the lengths of the links of the exoskeleton leg. This design decision reduces the number of states for the FSM and the number of operations to be performed. Table 4.2 presents the number of floating-point operators used for the inverse kinematics.

Table 4.2 Number of Composing Operators for calculating the Inverse Kinematics

Operator	**Number of units**
Addition/subtraction	1
Multiplier	2
Square root	1
Arctan2	1

Numerical design simulations were performed to verify the correctness of the design. Questa Sim 10.1 was used for this purpose. Three different hardware designs with different bit widths were developed and tested for the inverse kinematics model as illustrated in table 4.3.

4.3.CO-SIMULATION WITH QUESTA SIM

For the error analysis a 64 bit software model in Matlab is used. This model is created using Matlab/Simulink's CosimWizard tool and the Mentor Graphics Questa Sim 10.1 hardware simulator tool. Questa Sim is a hardware design and simulation tool for Hardware Description Language.

Table 4.3 Bit Widths of Hardware Design Representations Used

Bit width	**exponent**	**mantissa**
27	8	18
32	8	23
45	8	36
64	11	52

The four different bit width configurations were integrated into a Device Under Test (DUT) with a 64 bit software model developed in Matlab. A wrapper designed in VHDL was used to map the 27, 32 and 45 bit width representations onto the 64 bit inverse kinematics Matlab model. For each bit width representation an error analysis was performed using the co-simulation model depicted in figure 4.4.

Matlab's uses the IEEE standard double precision format defined by the ANSI/IEEE Standard 754-1985 for Binary Floating-Point Arithmetic [49]. This standard was adopted by the IEEE Standards Board and the American National Standards Institute in 1985. Nowadays this standard is used by all computers being designed. Since the standard offers some flexibility, computers will not always get exactly the same results.

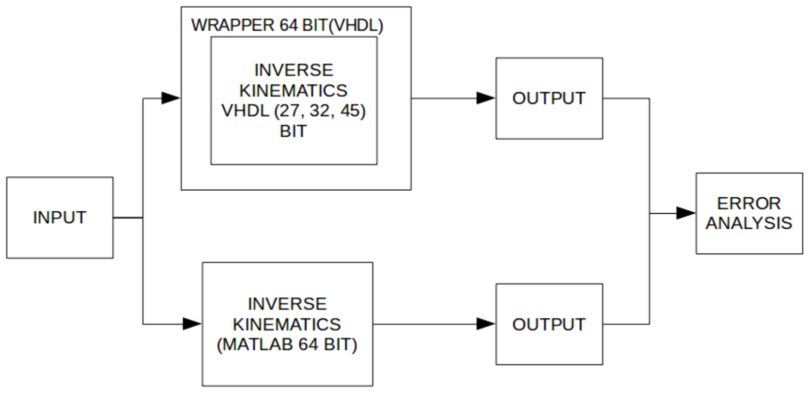

WRAPPER 64 BIT(VHDL)
INVERSE KINEMATICS VHDL (27, 32, 45) BIT
OUTPUT
INPUT
ERROR ANALYSIS
INVERSE KINEMATICS (MATLAB 64 BIT)
OUTPUT

Figure 4.4 64 bit wrapper for HDL co-simulation

For numerical conversions the ieee.float_pkg package, that is based on the IEEE 754 double precision floating-point standard was used [50].

The wrapper does the following transformations in VHDL using IEEE.float_pkg.ALL (in the example below for 27 bits):
std_logic (64) -> real -> float (27)
float (27) -> real -> std_logic (64)

Figure 4.5 illustrates the error analysis procedure. First the inputs *x* and *y* are generated in Matlab. The inputs are fed to the Hardware design and the Matlab functions, which represent the reference model for the DUT, generating outputs $\theta_1$and θ_2 (*t1* and *t2* in Figure 4.2). If the Hardware design is correct, its outputs should approximate that of the Matlab model. The generated Matlab inputs are 64 bit floating-point numbers. The MSE and the ARE values between the DUT and the Matlab model can then be calculated using the Matlab model as a statitical reference. The co-simulation model developed in Simulink is depicted in figure 4.4.

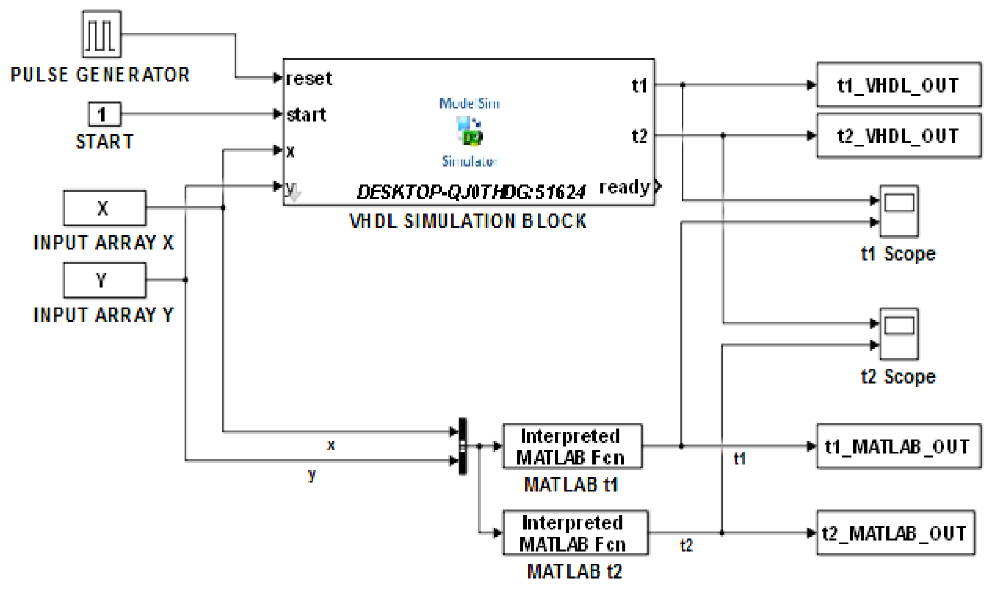


Figure 4.2 HDL co-simulation model in Simulink

The MSE and the ARE of 100 different input pairs between the Matlab model and the Hardware model were calculated for the outputs θ_1 and θ_2. The errors were calculated for 15 and 20 CORDIC iterations of the floating-point IP core of the *atan2* function.

The test data used was generated by letting θ_1 vary between 10 and 120 degrees and θ_2 between 10 and 179 degrees, incrementing θ_1 and θ_2 by three degrees for the next iteration, thus generating a total of 2016 input/output samples. The values for θ_1 and θ_2 for the error calculations were chosen to attend the possible values for the application domain. The generated MSE and ARE values are presented table 4.4 and 4.5.

Table 4.4 MSE and ARE values for 15 CORDIC Iterations (Error unit: radians)

Bit width	MSE t1	MSE t2	ARE t1	ARE t2
27	2,28E-05	6,48E-05	4,21E-01	5,85E-01
32	1,77E-05	1,85E-05	3,79E-01	3,96E-01
45	1,77E-05	1,85E-05	3,78E-01	3,97E-01
64	1,77E-05	1,85E-05	3,78E-01	3,97E-01

Table 4.5 MSE and ARE values for 20 CORDIC Iterations (Error unit: radians)

Bit width	MSE t1	MSE t2	ARE t1	ARE t2
27	1,50E-05	5,47E-05	1,77E-01	3,39E-01
32	1,77E-05	1,85E-05	3,79E-01	3,96E-01
45	7,16E-09	1,54E-08	6,60E-03	1,18E-02
64	7,16E-09	1,54E-08	6,60E-03	1,18E-02

4.4.RESOURCES CONSUMPTION ANALYSIS FOR THE INVERSE KINEMATICS ON FPGA

This section presents a hardware resources consumption analysis for the inverse kinematics mapped on a Artix7 XC7A100T FPGA device (Nexys4 Development Board). The resources consumption was estimated after logic synthesis (no physical implementation was performed).

In table 4.6 the trade-off between area on FPGA and performance can be observed. As the bit width increases, also the FPGA resource usage increases while the frequency decreases. The 64 bit representation presents far greater resource consumption than the three previous ones. There is a slight difference in resource consumption between the 27 and 32 bit representation while the former seems to have greater performance.

Table 4.6 FPGA area, Performance and Power Consumption trade-offs

	FF's		LUT's		DSP's		Max.freq.(MHz)
27	1224	1%	2647	4%	5	2%	112.16
32	1427	1%	3156	4%	8	3%	96.86
45	2011	1%	4789	7%	32	13%	64.07
64	2879	2%	7381	11%	57	23%	52.57

4.5.CONCLUSION

Tables 4.4 and 4.5 present the difference between errors when using 15 and 20 CORDIC iterations respectively for every bit width representation. The smaller bit widths present no significant difference when 15 or 20 CORDIC iterations are used. Analyzing table 4.4 (15

CORDIC iterations), one can observe that for the 32, 45 and 64 bit width representations the error values remain the same. Only the 27 bit width representation is slightly larger. In table 4.5 (20 CORDIC iterations) the error for the two smaller bit width representations (27 and 32) are significantly larger than that of the larger bit width representations, but nevertheless they are still negligible for their purpose. The two larger bit width representations show no error differences between one another.

Table 4.6 presents the resource consumption and performance of the four bit width representations. According to the trade-off analysis the 27 bit representation would be a good candidate for implementing the inverse kinematics of the exoskeleton, offering the highest performance of all, and far smaller resource usage than the 45 and 64 bit representations. Though, according to the error analysis the 27 bit representation presents the greatest error values, they are still very small and negligible for the purpose of calculating the inverse kinematics.

In the wrapper some numeric conversions are made. This, along with possible differences in implementation of the ANSI/IEEE Standard 754-1985 may introduce differences in numeric outcome.

5. CONTROL SYSTEM MODEL

Two control system models with integrated position and velocity control were developed for the exoskeleton. This chapter describes a system level implementation of both control system models and tuning, using the Ziegler-Nichols and PSO techniques. A human gait database was used to simulate and validate the exoskeleton control system models.

5.1.SIMMECHANICS MODEL

The Simmechanics model of the exoskeleton was generated using the Simmechanics import tool from Matlab (see Figure 5.1).

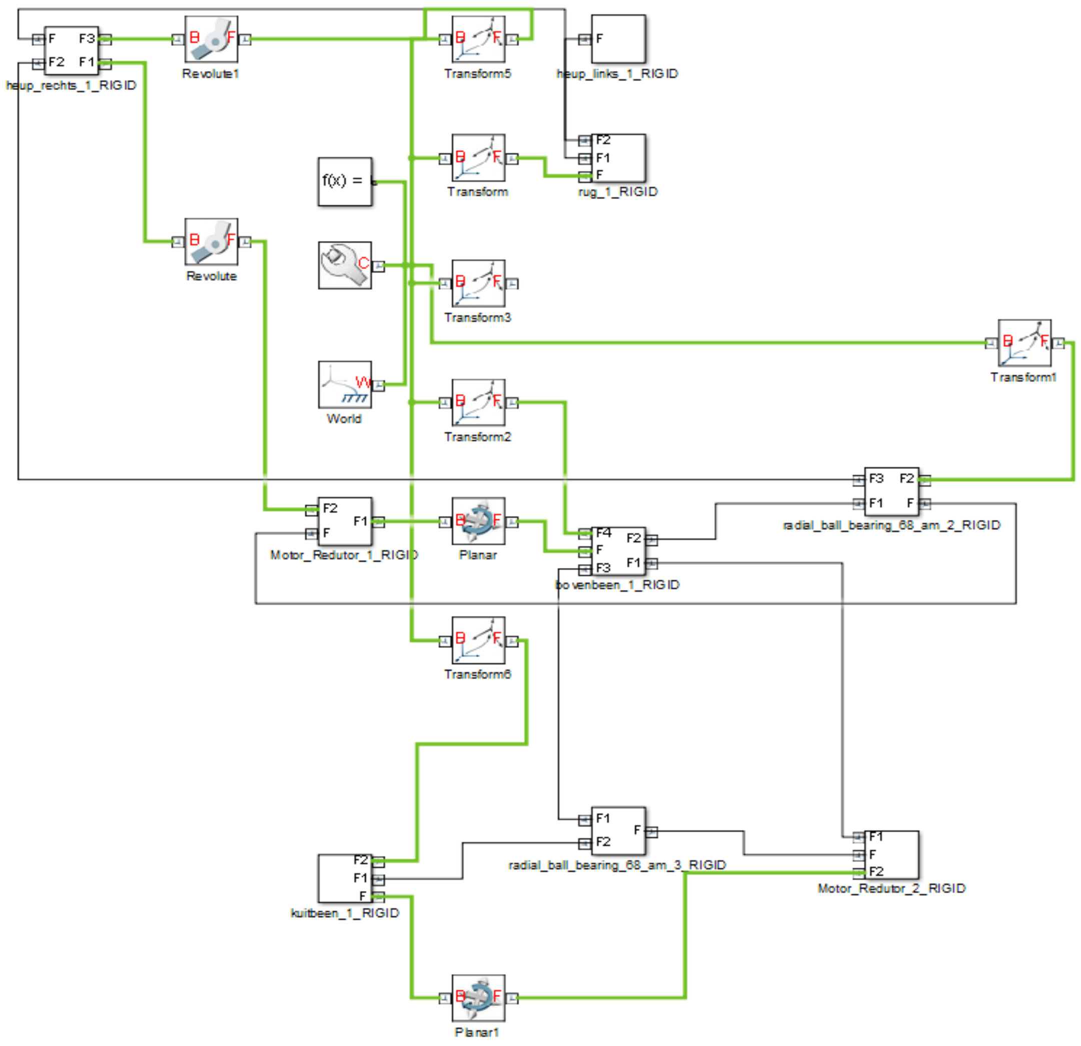


Figure 5.1 Simmechanics Model generated from Solidworks Exoskeleton Model

The initial Simmechanics model contained a great deal of detail caused by the presence of parts that slide into each other such as the bearings, links and joints generating a number of prismatic joints and other additional components. For the purpose of building the control system model in Simulink these details are not relevant, so the exoskeleton model was revised in Solidworks, mating parts that generated unnecessary prismatic and revolute joints together. The Simmechanics model was simplified removing the planar joints on the one hand, but on the other hand complexity was added by introducing actuator inputs and position- and velocity sensors outputs to the joints, generating the model in figure 5.2.

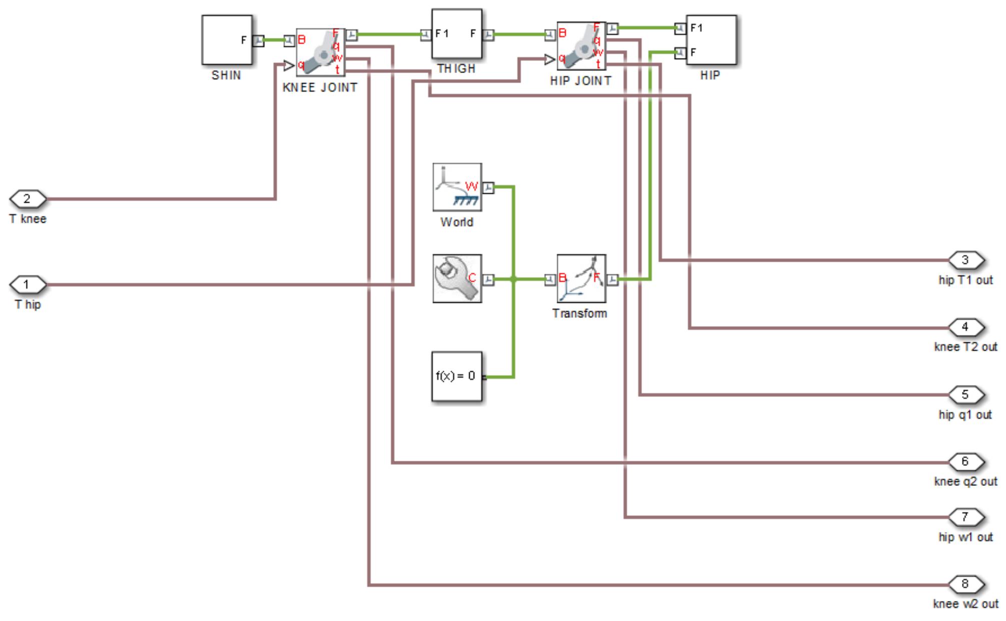


Figure 5.2 Simplified Simmechanics Model with Actuator Inputs and Sensor Outputs

To the model presented in figure 5.2 the necessary components and algorithms were then added to build and integrate the exoskeleton control system.

5.2.BLDC MOTOR MODEL IN MATLAB

The LEIA exoskeleton will consists of two motors [3], one in each joint for its actuation. Each motor will be controlled by an ESCON 70/10 [51] motor driver. This is a 4-quadrant pulse-width modulation (PWM) servo speed controller featuring open- and closed loop speed control and current control. Two approaches to the design of the combined velocity and position control of the exoskeleton were developed.

The exoskeleton motors are modeled as direct current (DC) Motor Simulink blocks. The values of the properties for these blocks are extracted from the datasheet of the motor. The motors used are Maxon EC 90 flat Ø90 mm, brushless 90 Watt motors equipped with Hall sensors. To generate the necessary torque a Maxon GP62 planetary gears with reduction ratio of 100 are also attached to each motor (figure 5.3).

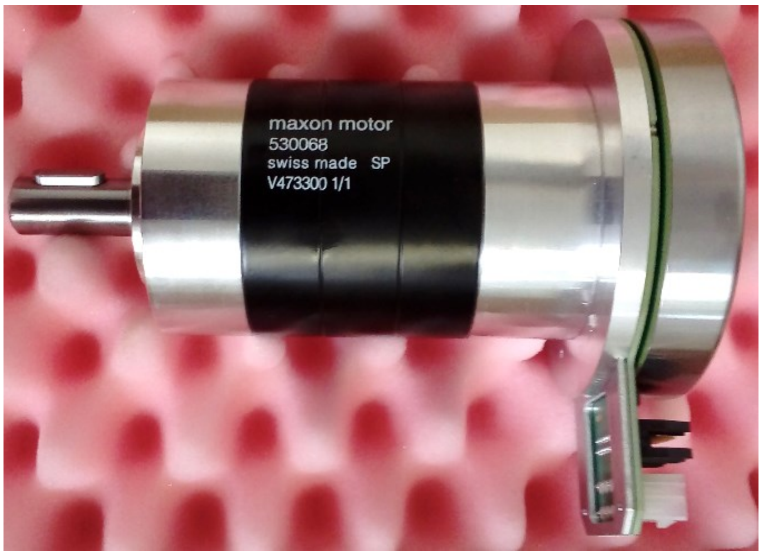


Figure 5.3 Maxon EC 90, brushless, 90 Watt motor with GP62 planetary gear

The DC Motor model is presented in figure 5.4 showing the low and high voltage inputs, and a number of sensor outputs. The outputs *Q1* and *W1* represent the motor angular position and angular velocity respectively.

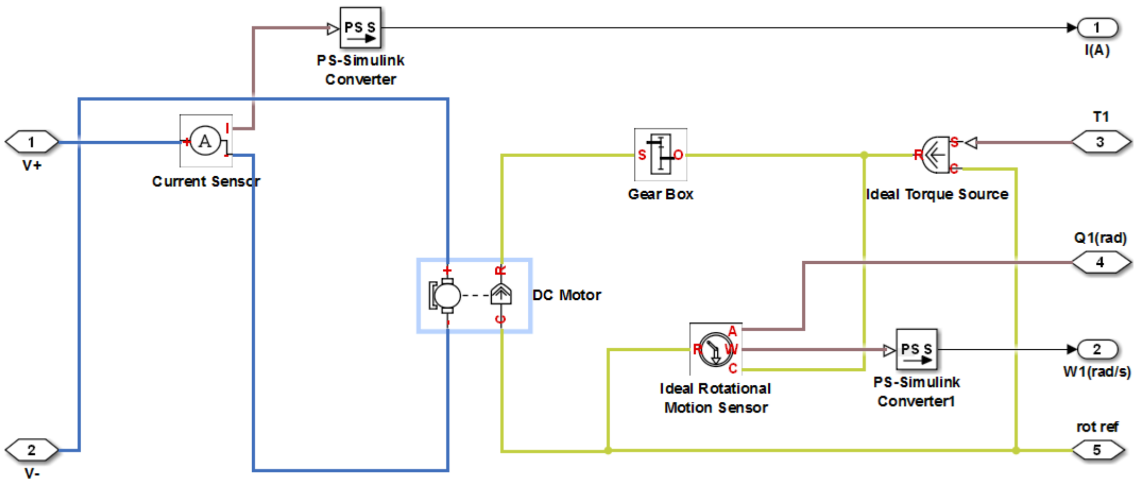
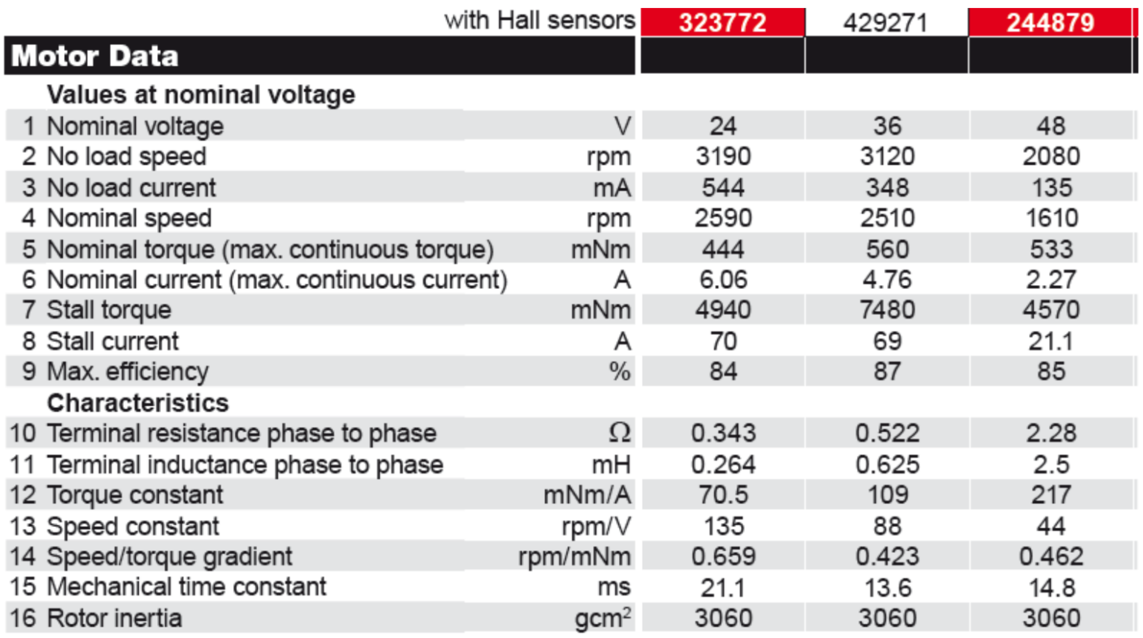


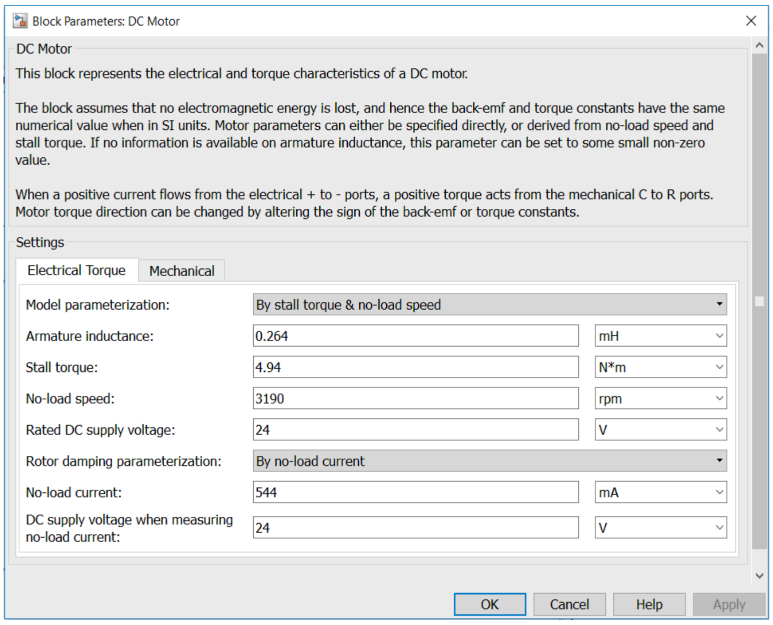
Figure 5.4 Simulink Model of the Exoskeleton Motor with Gear

Part of the data sheet of the motor is presented in figure 5.5 presenting the required parameters. These values were read into the DC Motor block of figure 5.6.



		with Hall sensors	323772	429271	244879
Motor Data					
	Values at nominal voltage				
1	Nominal voltage	V	24	36	48
2	No load speed	rpm	3190	3120	2080
3	No load current	mA	544	348	135
4	Nominal speed	rpm	2590	2510	1610
5	Nominal torque (max. continuous torque)	mNm	444	560	533
6	Nominal current (max. continuous current)	A	6.06	4.76	2.27
7	Stall torque	mNm	4940	7480	4570
8	Stall current	A	70	69	21.1
9	Max. efficiency	%	84	87	85
	Characteristics				
10	Terminal resistance phase to phase	Ω	0.343	0.522	2.28
11	Terminal inductance phase to phase	mH	0.264	0.625	2.5
12	Torque constant	mNm/A	70.5	109	217
13	Speed constant	rpm/V	135	88	44
14	Speed/torque gradient	rpm/mNm	0.659	0.423	0.462
15	Mechanical time constant	ms	21.1	13.6	14.8
16	Rotor inertia	gcm^2	3060	3060	3060

Figure 5.5 Maxon EC 90 BLDC Motor data sheet



Block Parameters: DC Motor

DC Motor

This block represents the electrical and torque characteristics of a DC motor.

The block assumes that no electromagnetic energy is lost, and hence the back-emf and torque constants have the same numerical value when in SI units. Motor parameters can either be specified directly, or derived from no-load speed and stall torque. If no information is available on armature inductance, this parameter can be set to some small non-zero value.

When a positive current flows from the electrical + to - ports, a positive torque acts from the mechanical C to R ports. Motor torque direction can be changed by altering the sign of the back-emf or torque constants.

Settings

Electrical Torque | Mechanical

Model parameterization:	By stall torque & no-load speed	
Armature inductance:	0.264	mH
Stall torque:	4.94	N*m
No-load speed:	3190	rpm
Rated DC supply voltage:	24	V
Rotor damping parameterization:	By no-load current	
No-load current:	544	mA
DC supply voltage when measuring no-load current:	24	V

Figure 5.6 DC Motor block settings for Maxon EC 90 BLDC motor

5.3.FIRST APPROACH FOR INTEGRATED VELOCITY AND POSITION CONTROL

In this approach the control system of the exoskeleton uses integrated speed and position control, with a separate control loop for each motor. Each control loop is comprised of a position controller and a speed controller within the position controller loop as stated in [52].

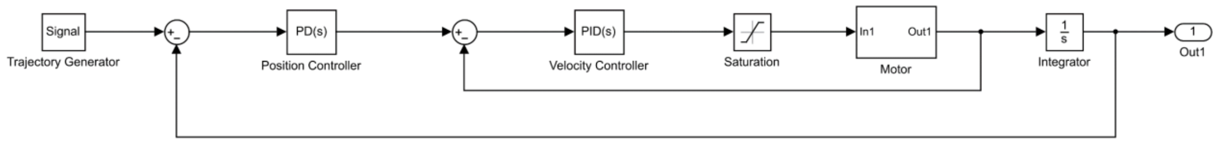


Figure 5.7 First approach to control system design for exoskeleton (adapted from [52])

The position controllers are both PD controllers while the velocity controllers are PID controllers. Figure 5.7 presents the control system model of the exoskeleton for the first approach. In this figure it can be observed that the Simmechanics model depicted in figure 5.8 is integrated into a subsystem called EXOSKELETON, around which the control system is built.

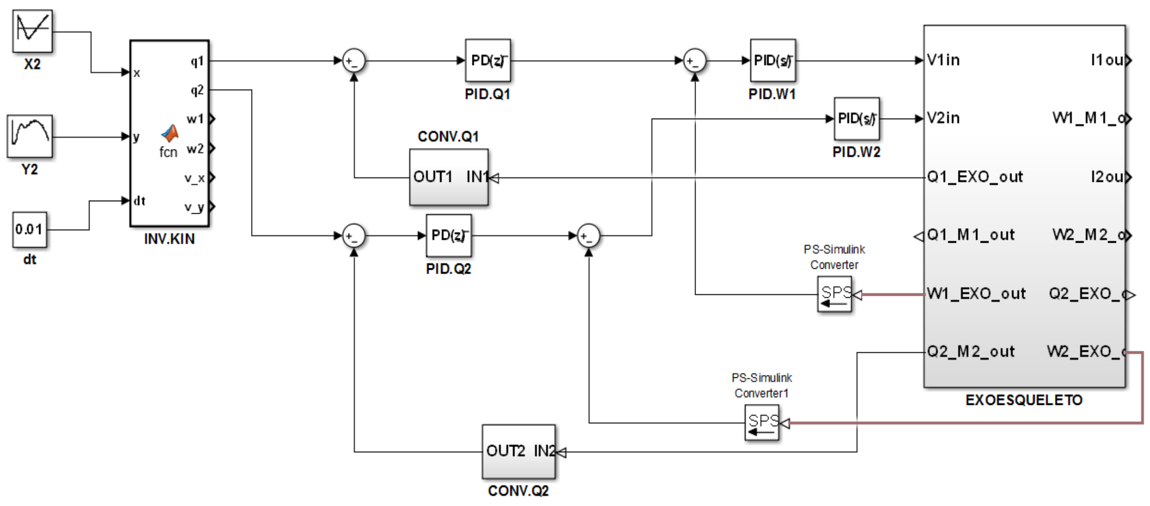


Figure 5.8 Control system design according to first approach

The exoskeleton motors each have one input to the exoskeleton, *V1in* and *V2in*, by which the two motors are powered, and a number of sensor outputs. The following outputs are used to calculate the error values for motor velocity (*rad/s*) and position (*rad*):

Q1_M1_out: angular position of the hip joint motor

Q2_M2_out: angular position of the knee joint motor

W1_M1_out: angular velocity of the hip joint motor

W2_M2_out: angular velocity of the knee joint motor

The control system of the exoskeleton receives the end-effector position as its input (*x* and *y*). The end-effector position is then calculated using the inverse kinematics in software (INV.KIN module). To evaluate the control system single input value pair (*x,y*) for the end-effector position are introduced to the input.

The PS Simulink converter blocks in figure 5.8 are unit conversion blocks which convert physical signal units from the exoskeleton block into the specified units from in these blocks. In this case the angular position is converted into radians.

The CONV.Q1 and CONV.Q2 blocks are used to set the initial angles of the hip and knee joints to a straight down position as presented in figure 5.9. The reference frame for the exoskeleton is located in the center of the hip joint, with the positive *x* axis pointing to the right and the positive *y* axis pointing upwards.

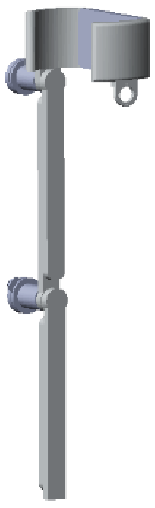


Figure 5.9 Exoskeleton leg straight down position

Table 5.1 presents the basic measurements of the upper and lower leg of the exoskeleton which are fictitious but nevertheless are more or less proportional to a real human anatomy, corresponding to an end-effector position of (0, -1.05) in the straight down position.

Table 5.1 Exoskeleton basic measurements

Part	**Length(m)**
Upper leg	0.5
Lower leg	0.55

The inverse kinematics then delivers the corresponding joint angles *q1* and *q2* for each motor. The angles *q1* and *q2* are thus the set points for each exoskeleton joint. The error signal is calculated by subtracting the desired end-effector position from the actual joint angle measured by the sensors at the exoskeleton.

The exoskeleton has a total of two PID controllers and two PD controllers (figure 5.8) as follows:

PD.Q1: PD controller for position control of the hip joint motor

PD.Q2: PD controller for position control of the knee joint motor

PID.W1: PID controller for the angular velocity of the hip joint motor

PID.W2: PID controller for the angular velocity of the knee joint motor

5.3.1. Control System Tuning for first approach

In this approach the velocity controllers are both first tuned separately using the Ziegler-Nichols method. To calculate the PID parameters for the velocity controllers in the first approach a unitary step input with step time of 0 seconds and simulation time of 2 seconds was presented to the plant according to the Ziegler-Nichols step response method yielding the following results:

Table 5.2 Results with Ziegler-Nichols step response method for velocity controller motor 1

a =	0.2564	**P**	**I**	**D**
L =	0.0036	0.30768	42.73333	0.000554

The results for the PID parameter tuning of both velocity controllers of model 1 appear to be essentially the same.

The PD position controllers of motor 1 and motor 2 were tuned using PSO since Ziegler-Nichols since this method does not define a procedure for the calculation of PD controller parameters. A swarm size of 10 individuals and 50 iterations were used. In this case *c1* and *c2* were chosen 2.05 which are the most common values encountered in literature [53] [54]. These values are commonly used to restrain velocities from attaining unacceptable levels.

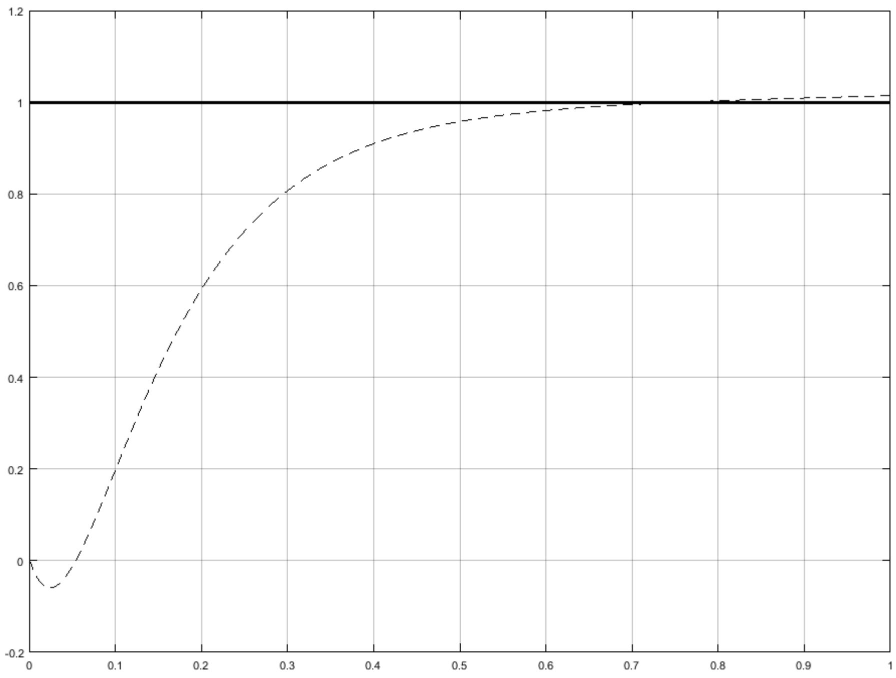


Figure 5.10 Step response with Ziegler-Nichols step response method for velocity controller motor1

Table 5.3 Results with Ziegler-Nichols step response method for velocity controller motor2

		P	**I**	**D**
a =	0.2335			
L =	0.0033	0.2802	42.45455	0.000462

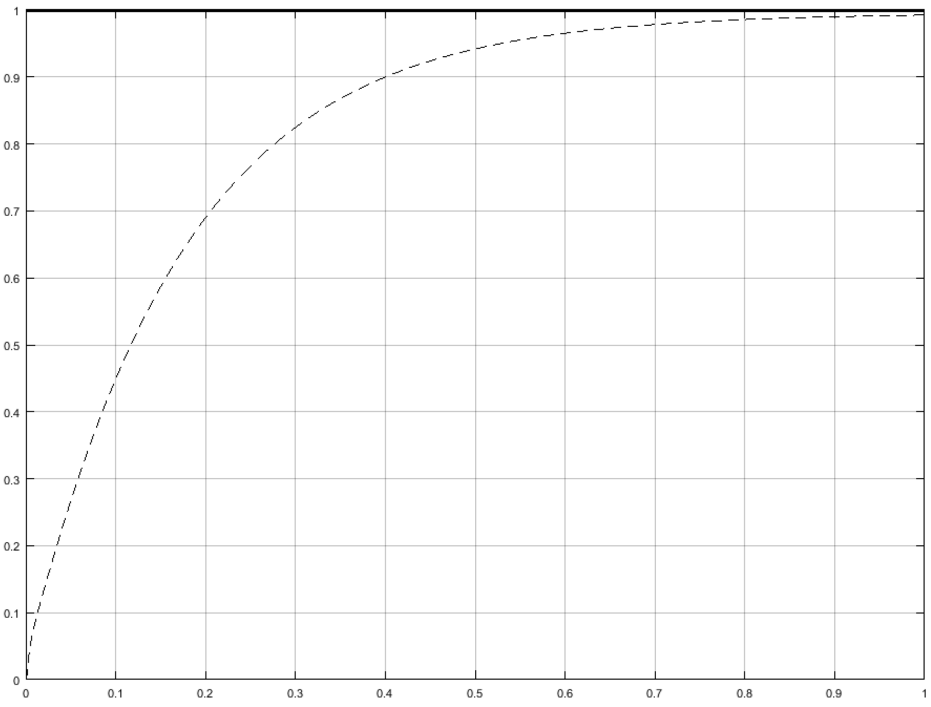


Figure 5.11 Step response with Ziegler-Nichols step response method for velocity controller motor2

The fitness function used was developed to minimize the error between the set-point and the output signal, to minimize the overshoot and minimize the rising time for each motor separately. Thus, the final fitness function is the sum of the two fitness functions.

$$F_{q1} = beta * e_{q1} + alpha * overshoot_{q1} + gamma * rising\_time_{q1} \quad (5.1)$$

$$F_{q2} = beta * e_{q2} + alpha * overshoot_{q2} + gamma * rising\_time_{q2} \quad (5.2)$$

$$F = F_{q1} + F_{q2} \quad (5.3)$$

Table 5.4.a PSO variables used in algorithm

Variable	**Identifier**	**Value**
Swarm size	*n*	10
Dimensions	*dim*	4
Search space	*(x,y)*	([0..100],[0..100])
Cognitive scaling parameter	*c1*	2.05
Social scaling parameter	*c2*	2.05
Momentum of Inertia	*w*	0.9
Iterations	*bird_steps*	50
Weighting factor overshoot	*alpha*	2
Weighting factor error	*beta*	1
Weighting factor rising time	*gamma*	1

The values of the variables in the PSO algorithm are presented in table 5.4.a and the tuning results for the PD parameters of motor1 and motor2 are presented in table 5.4.b. The convergence graph for the tuning the controller parameters with PSO is presented in figure 5.12 and the step responses with the obtained controller parameter values are presented in figure 5.13 and 5.14. From figure 5.12 it is clear that the fitness value decreases to a value of approximately 1.3 after 40 iterations. The step response of figure 5.13 presents a high settling time for the position controller of motor1 and figure 5.14 shows high overshoot for the position controller of motor2. From table 5.4.b it can be observed that the P value for the position controller of motor2 is a border value (See table 5.4.a).

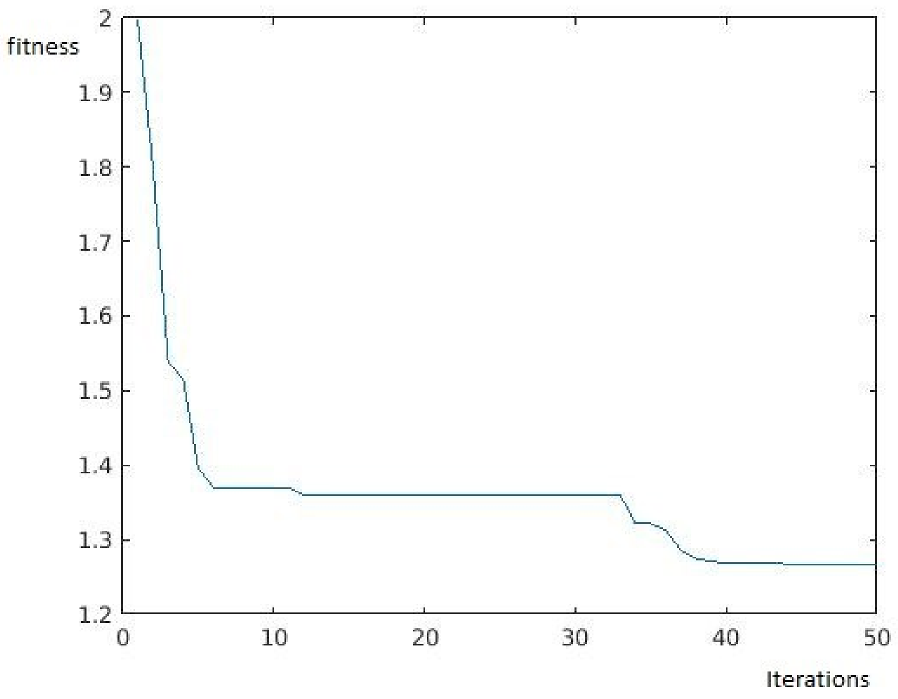


Figure 5.12 Convergence graph for PD controller tuning with PSO

Table 5.4.b PSO tuning results for the controller parameters of motor1 and motor2

	P	D
Motor1	76.3825	4.7374
Motor2	100	7.2391

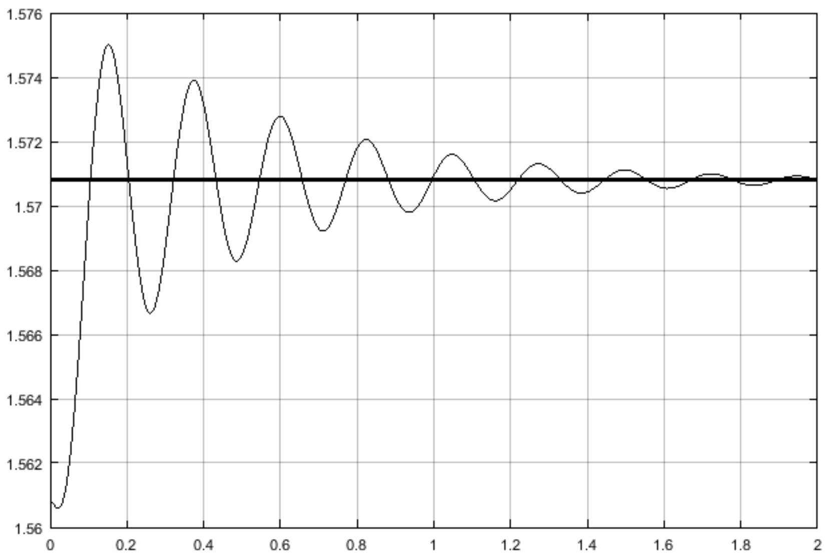


Figure 5.13 Step response for motor1 with PD parameters obtained with PSO

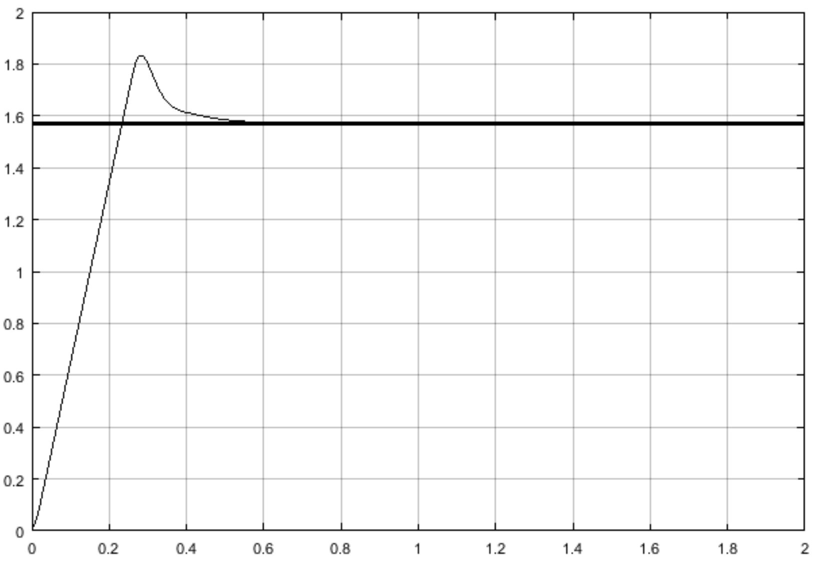


Figure 5.14 Step response for motor2 with PD parameters obtained with PSO

Considering the above, to improve control system performance the *PID tuner* tool available in Matlab was used to fine-tune the PD controllers. The step responses were greatly improved (figure 5.14 and 5.15). The fine-tuned PD controller parameters are presented in table 5.5.

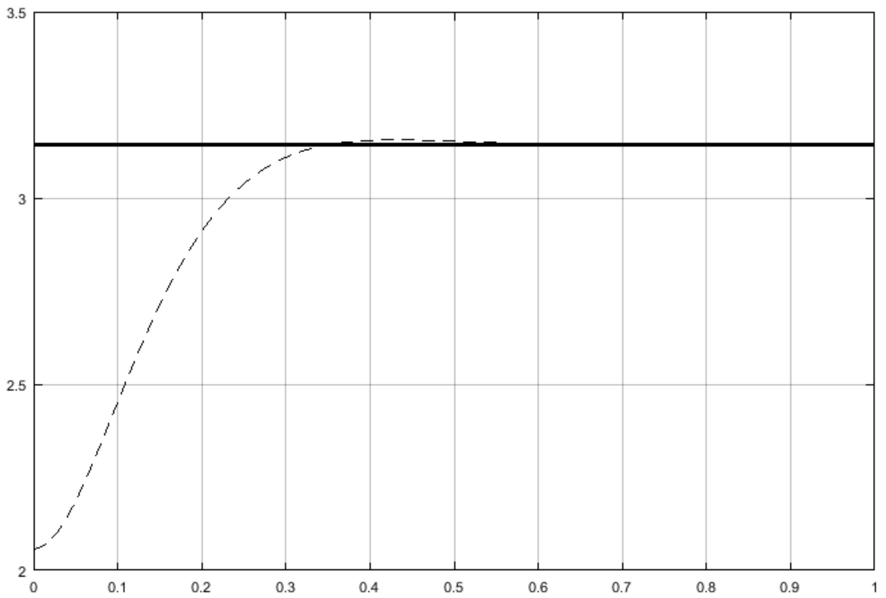


Figure 5.15 Step response for motor1 with PD parameters obtained with PID tuner

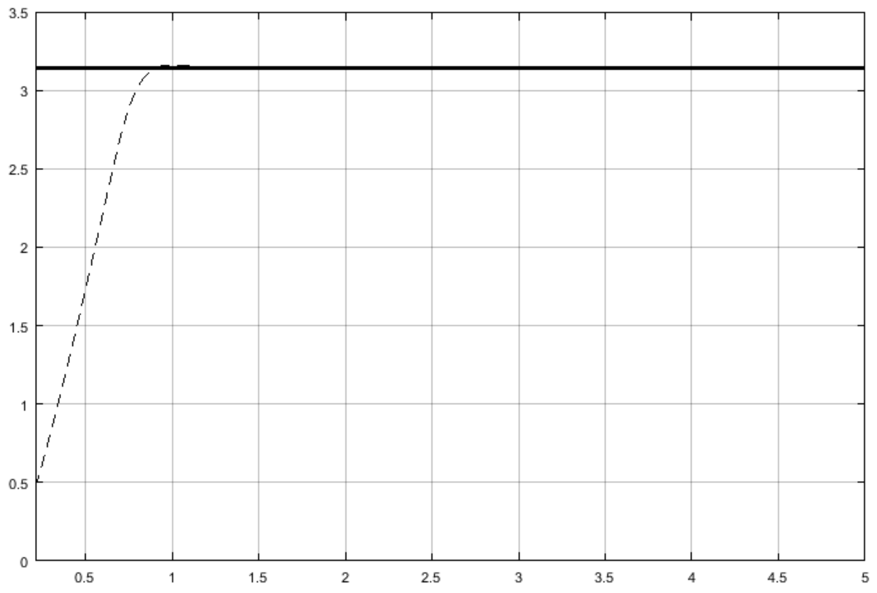


Figure 5.16 Step response for motor2 with PD parameters obtained with PID tuner

Table 5.5 PSO and PID tuner results for the controller parameters of motor1 and motor2

	PSO		**PID tuner**	
	P	**D**	**P**	**D**
Motor1	76.3825	4.7374	16.95962	1.640908
Motor2	100	7.2391	17.54065	1.571683

5.4.SECOND APPROACH FOR INTEGRATED VELOCITY AND POSITION CONTROL

In the second approach adapted from [55] another strategy to integrated position and velocity control is developed. The control system design of this approach is presented in figure 5.16. In this case two PID controllers are used for position control and two for velocity control, one for each motor. The top PID controller is for position control, receiving the angular position as its reference signal. The derivative block passes the derivative of the angular position, i.e. the angular velocity, as a reference signal to the second PID controller for velocity control. There is also a low pass filter present after the derivative block to attenuate frequencies higher than the cutoff frequency.

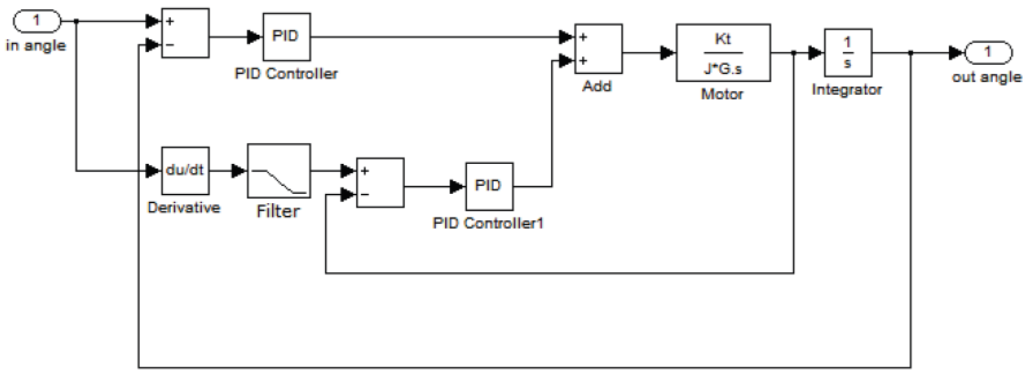


Figure 5.17 First approach to control system design for exoskeleton (source: [47])

The system setup for the exoskeleton control system is depicted in figure 5.17.

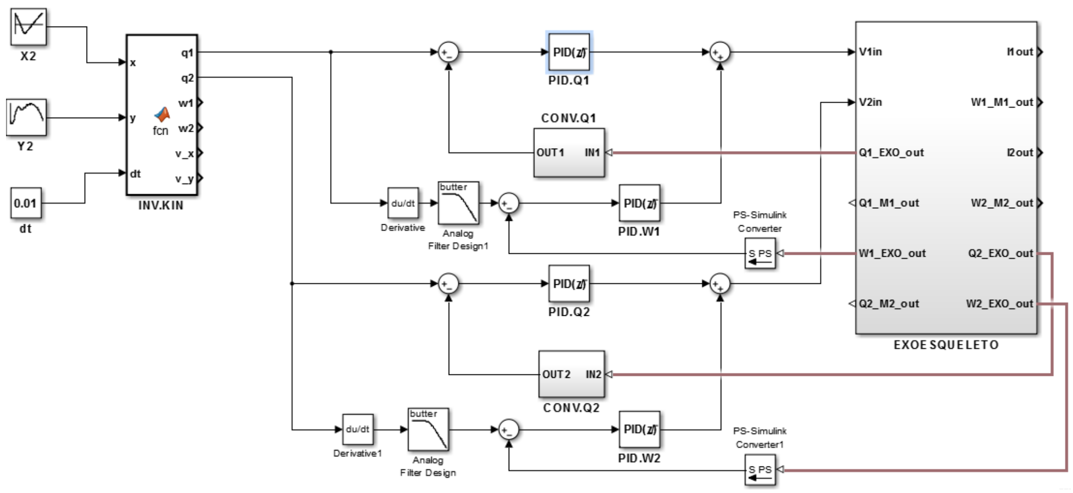


Figure 5.18 Control system design according to second approach

In this approach the contribution of the velocity controller imposes additional control on the system as a whole. Since the velocity controllers impose additional control the position controllers were first tuned and the velocity controllers afterwards.

As in the previous approach the exoskeleton inputs, outputs and other remaining blocks, except the control system blocks remain the same. In this approach the exoskeleton has a total of four PID controllers:

PID.Q1: PID controller for position control of the hip joint motor

PID.Q2: PID controller for position control of the knee joint motor

PID.W1: PID controller for the angular velocity of the hip joint motor

PID.W2: PID controller for the angular velocity of the knee joint motor

5.4.1.Control System Tuning for second approach

The position controllers are both first tuned separately using the Ziegler-Nichols method. To calculate the PID parameters a unitary step input with step time of 0 seconds and simulation time of 3 seconds was presented to the plant according to the Ziegler-Nichols step response method yielding the results of table 5.6. The I component is far higher than the P and D components.

Table 5.6 Results with Ziegler-Nichols step response method for position controller motor 1

a =	0.01129	**P**	**I**	**D**
L =	0.0115	106.2888	4621.25	0.61116

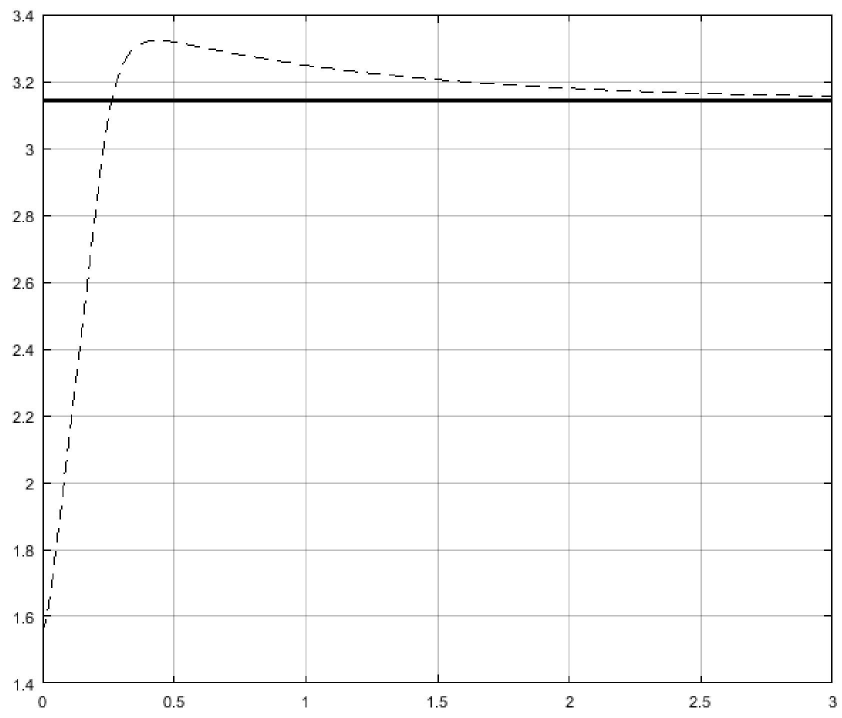


Figure 5.19 Step response with Ziegler-Nichols step response method for position controller motor1

For tuning the controller parameters for motor2 the Ziegler-Nichols self-oscillation method was used because of difficulties to determine the parameters *a* and *L* that are used in the step response method. The results are presented in table 5.7. The I component in this case is also very high compared to the P and D components.

Table 5.7 Results with Ziegler-Nichols self-oscillation method for position controller motor2

Ku =	2370	**P**	**I**	**D**
Tu =	0.052	1422	54692.31	9.243

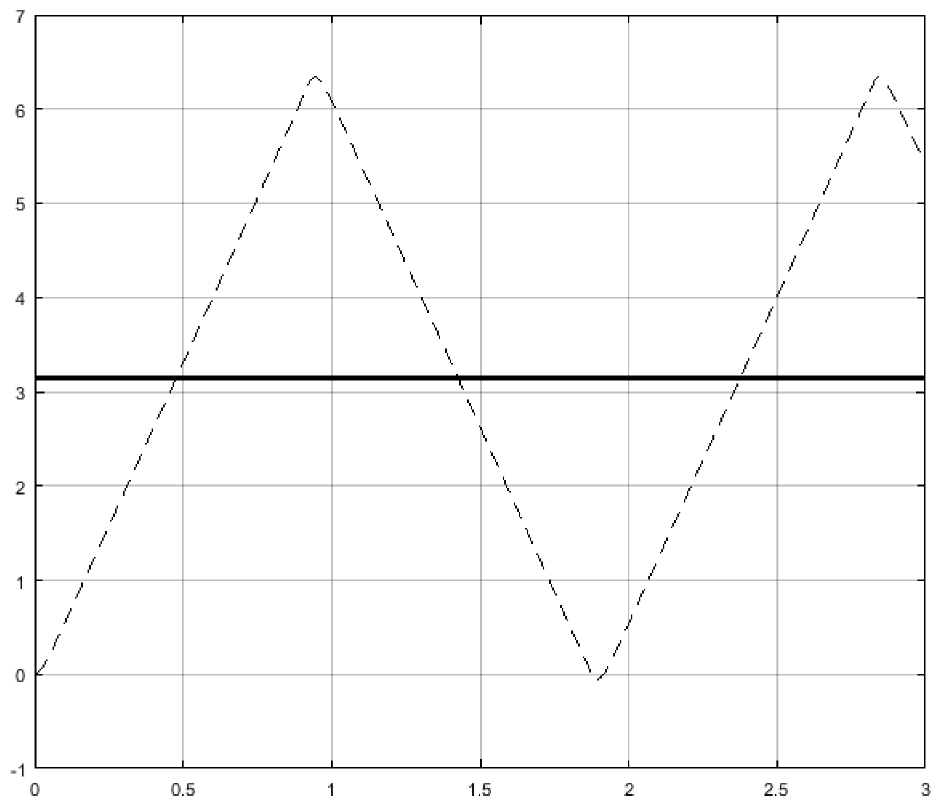


Figure 5.20 Step response with Ziegler-Nichols self-oscillation method for position controller motor 2

To obtain better system response the parameters are manually adjusted according to the closed loop step response properties of the PID controller parameters presented in table 5.8 [56].

Table 5.8 Effect of increasing PID parameters with closed loop step response

	Rise time	Overshoot	Settling time	Steady-state-error	Stability
Kp	Decrease	Increase	Small increase	Decrease	Degrade
Ki	Small decrease	Increase	Increase	Large decrease	Degrade
Kd	Small decrease	Decrease	Decrease	Minor change	Improve

Starting with the position controller of motor1 *I* was decreased to zero. This yielded the result finally adopted for the position controller of motor1. For the position controller of motor2 the I component was also first decreased to zero. Some finer tuning based on table 5.8 was applied. The final values are given in table 5.9 and the step responses for the controllers are given in figures 5.21 and 5.22.

Table 5.9 Ziegler-Nichols tuning results for the position controller parameters of motor1 and motor2

	Ziegler-Nichols			**After Manual adjustment**		
	P	I	D	P	I	D
Motor1	106.2888	4621.25	0.61116	106.2888	0	0.61116
Motor2	1422	54692.31	9.243	500	0	10

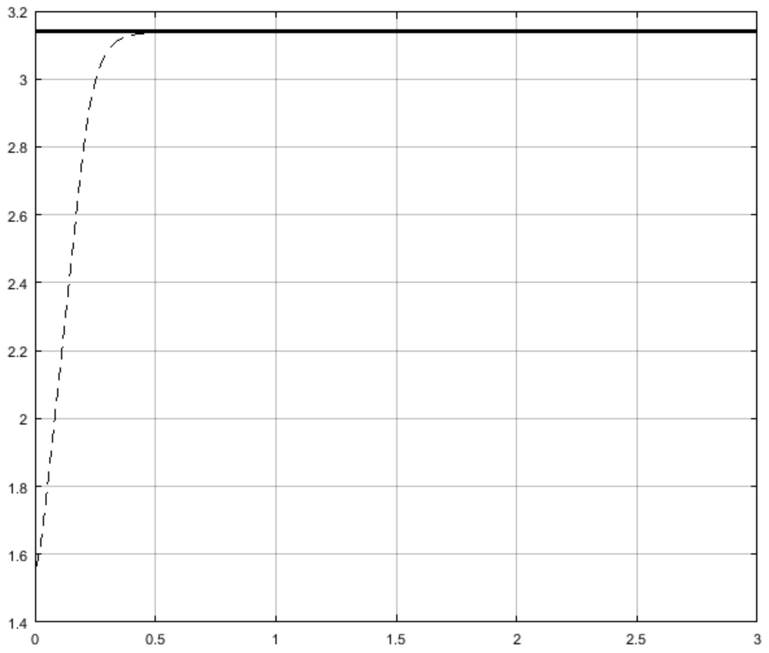


Figure 5.21 Manually tuned step response after Ziegler-Nichols step response method for position controller motor1

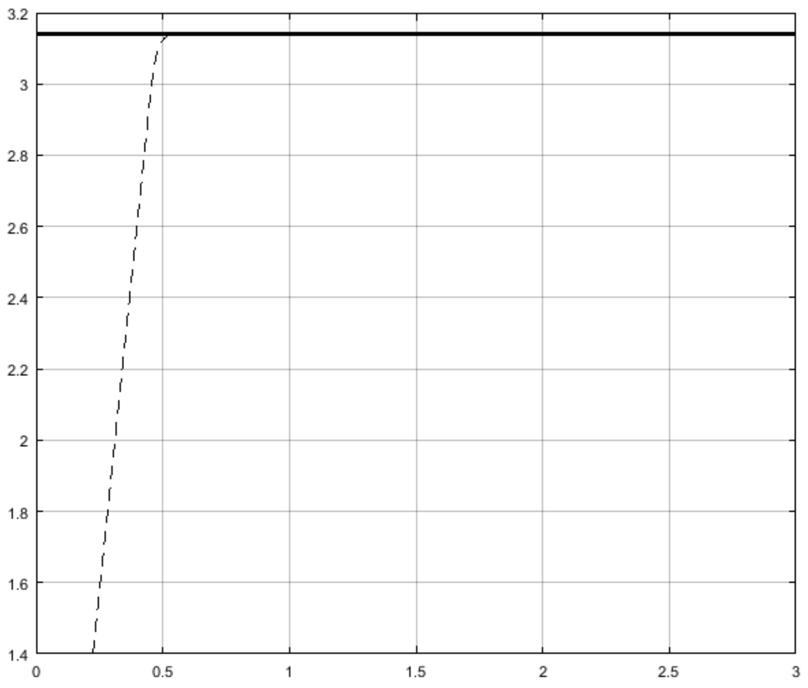


Figure 5.22 Manually tuned step response after Ziegler-Nichols step response method for position controller motor2

The velocity controllers were both tuned using the Ziegler-Nichols self-oscillation method. Figures 5.22 and 5.23 present the step responses for velocity control of motor1 and motor2 respectively.

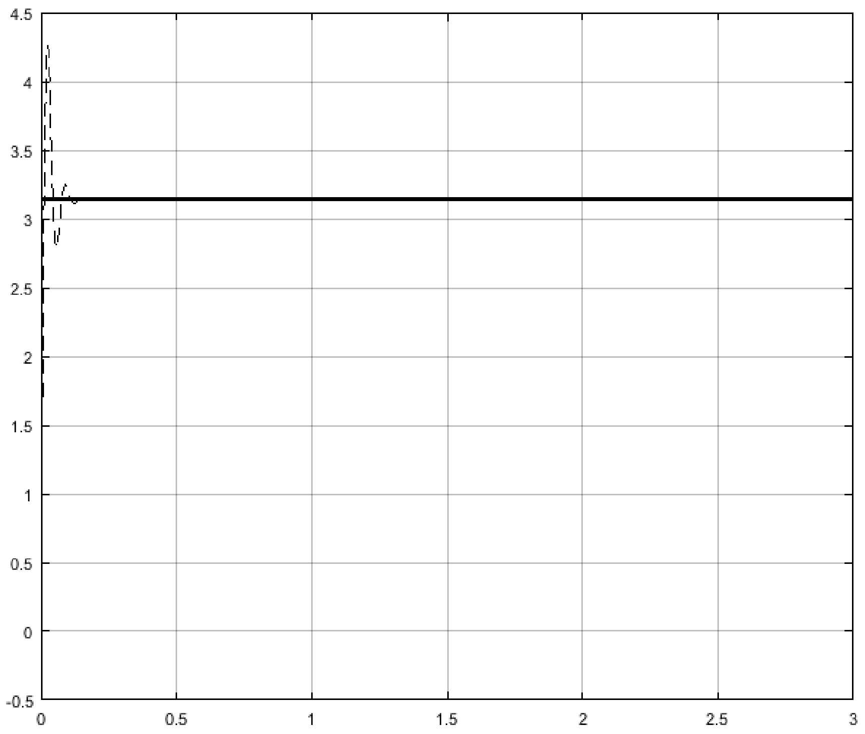


Figure 5.23 Step response for motor1 velocity controller obtained with Ziegler-Nichols self-oscillation method

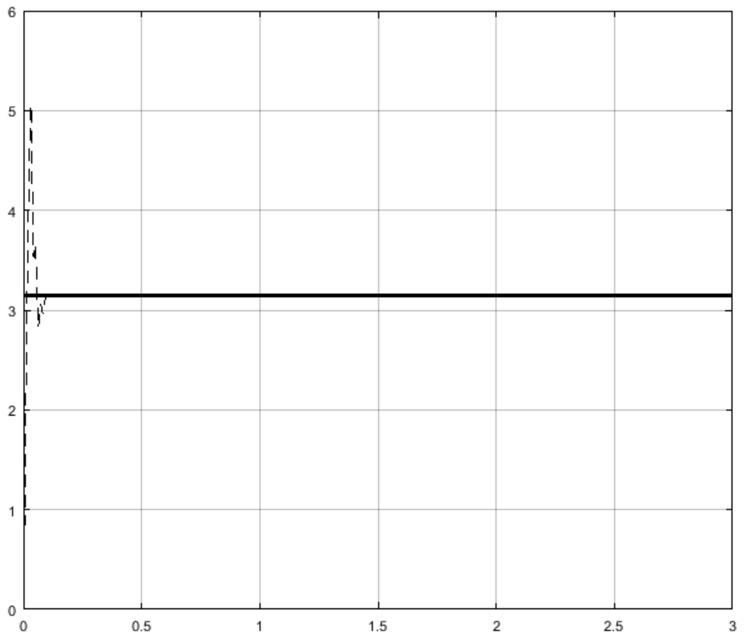


Figure 5.24 Step response for motor2 velocity controller obtained with Ziegler-Nichols self-oscillation method

The results of manual fine-tuning applying the properties in table 5.8 are given in table 5.10. Figures 5.25 and 526 present the final results of manually fine tuning the velocity controllers of motor1 and motor2 after using Ziegler-Nichols.

Table 5.10 Ziegler-Nichols tuning results for the velocity controller parameters of motor1 and motor2

	Ziegler-Nichols			**After Manual adjustment**		
	P	I	D	P	I	D
Motor1	16.2	1620	0.0405	0.7	100	0.03
Motor2	34.2	2280	0.12825	12	400	0.001

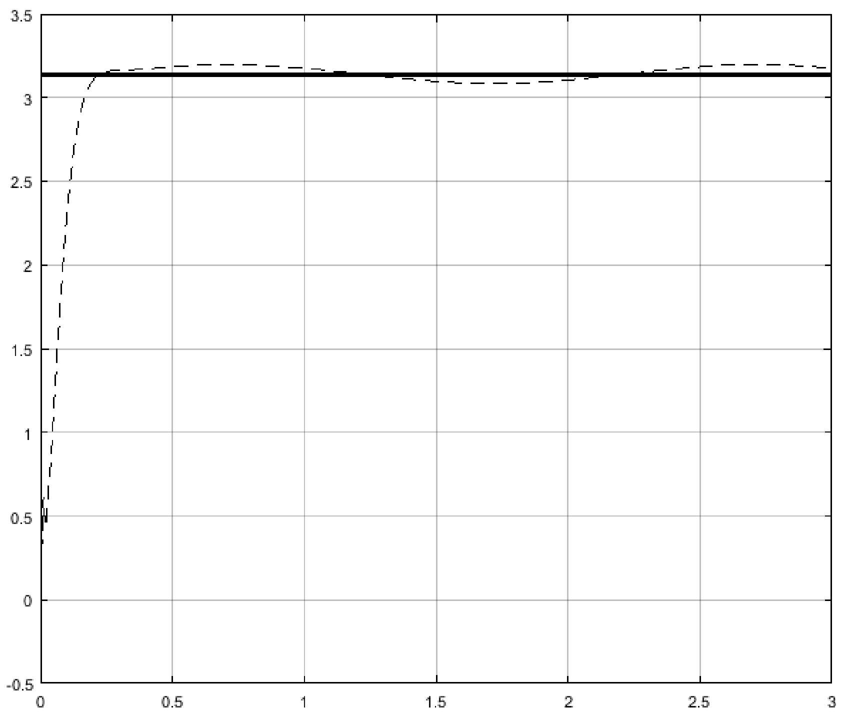


Figure 5.25 Step response for motor1 velocity controller obtained after manually fine-tuning Ziegler-Nichols obtained results

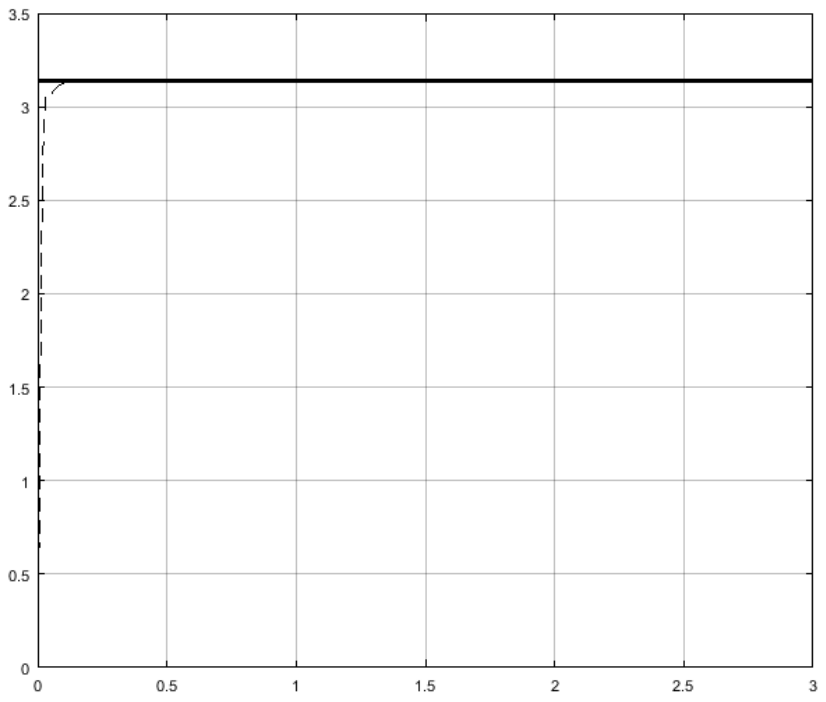


Figure 5.26 Step response for motor2 velocity controller obtained after manually fine-tuning Ziegler-Nichols obtained results

5.5.COMPARISON MODEL1 AND MODEL2

The step responses for the two combined position and velocity controller models were compared to one another for both motors. The Simulink setup to achieve this is depicted in figure 5.27. A step input of 2 radians is introduced to the inputs of both models. This value is beyond the human movement limits but is used here for test purposes for better visualization.

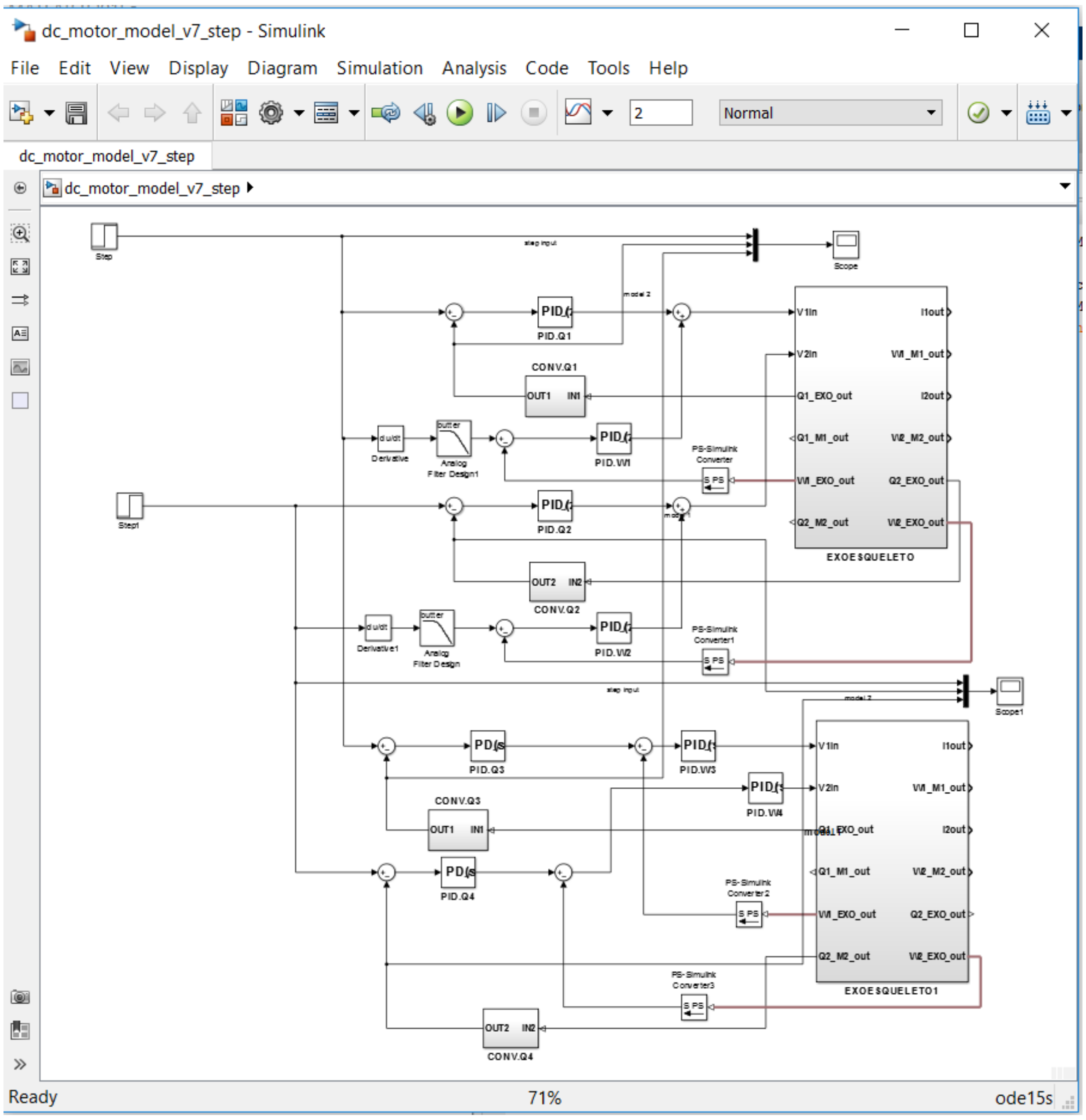


Figure 5.27 Step response setup in Simulink for control system model1 and control system model2

The step responses for both motors of model1 and model2 is presented in figures 5.28 and 5.29.

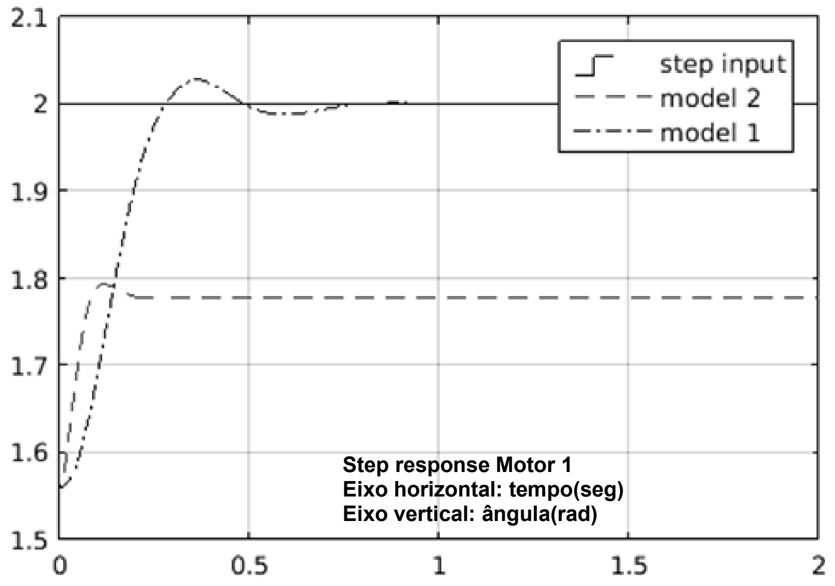


Figure 5.28 Step response for motor 1 of model1 and model 2

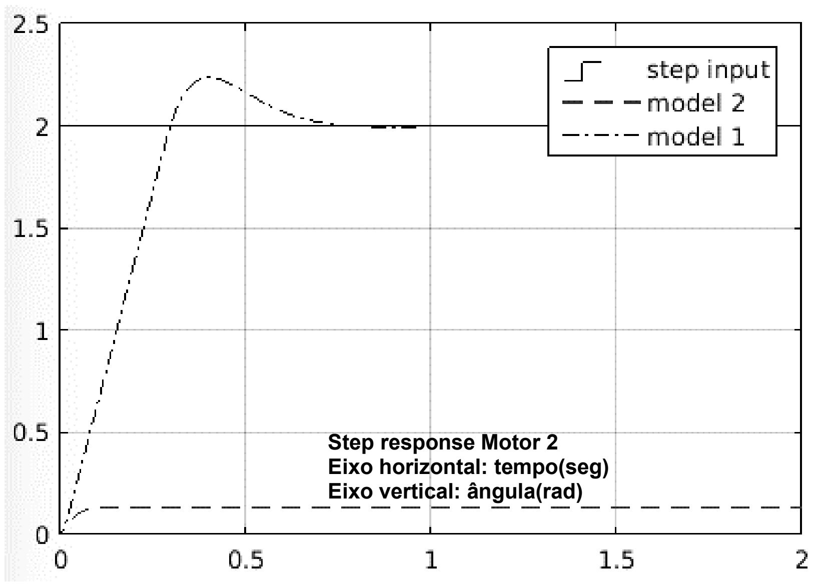


Figure 5.29 Step response for motor 2 of model1 and model 2

From the above results no efficient results are expected for model2 using the human gait database, because in this model the control system for both motor1 and motor2 underperform.

5.6.HUMAN GAIT SIMULATION DATA FOR THE EXOSKELETON

For the exoskeleton to simulate human gait, real human gait data was used. This data was extracted from Winter's human gait database available in [57]. It holds, among others, temporal angular position data for human gait movement. This data was extracted and adapted to the measurements of the exoskeleton, generating temporal angular position data for the exoskeleton leg. With this data the gait movement of the exoskeleton leg could then be observed.

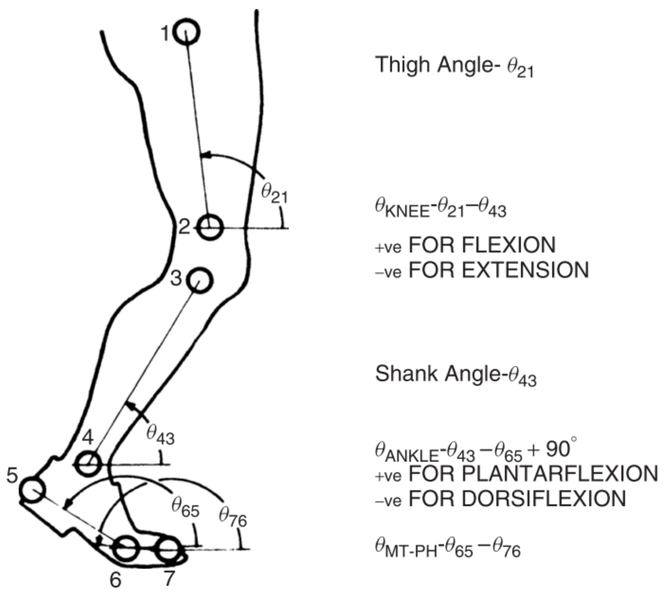
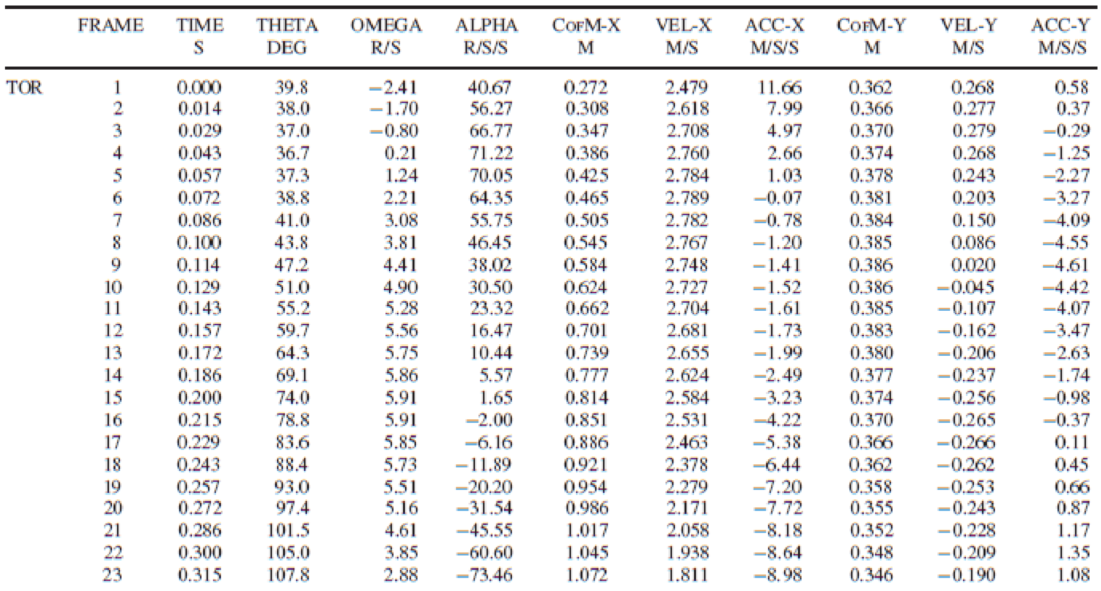

Thigh Angle- θ_{21}
θ_{21}
θ_{KNEE}-θ_{21}−θ_{43}
+ve FOR FLEXION
−ve FOR EXTENSION
Shank Angle-θ_{43}
θ_{43}
θ_{ANKLE}-θ_{43} −θ_{65} + 90°
+ve FOR PLANTARFLEXION
−ve FOR DORSIFLEXION
θ_{65}
θ_{76}
θ_{MT-PH}-θ_{65} −θ_{76}
1
2
3
4
5
6
7

Figure 5.30 Limb joint angle conventions for human gait data (source: [43])

Figure 5.30 presents the model used by Winter to collect the gait movement data. This model is different from the exoskeleton model, more specifically the exoskeleton kinematic model. Figure 5.31 depicts a portion of the Winter's data set used to simulate human gait movement.



	FRAME	TIME S	THETA DEG	OMEGA R/S	ALPHA R/S/S	CofM-X M	VEL-X M/S	ACC-X M/S/S	CofM-Y M	VEL-Y M/S	ACC-Y M/S/S
TOR	1	0.000	39.8	−2.41	40.67	0.272	2.479	11.66	0.362	0.268	0.58
	2	0.014	38.0	−1.70	56.27	0.308	2.618	7.99	0.366	0.277	0.37
	3	0.029	37.0	−0.80	66.77	0.347	2.708	4.97	0.370	0.279	−0.29
	4	0.043	36.7	0.21	71.22	0.386	2.760	2.66	0.374	0.268	−1.25
	5	0.057	37.3	1.24	70.05	0.425	2.784	1.03	0.378	0.243	−2.27
	6	0.072	38.8	2.21	64.35	0.465	2.789	−0.07	0.381	0.203	−3.27
	7	0.086	41.0	3.08	55.75	0.505	2.782	−0.78	0.384	0.150	−4.09
	8	0.100	43.8	3.81	46.45	0.545	2.767	−1.20	0.385	0.086	−4.55
	9	0.114	47.2	4.41	38.02	0.584	2.748	−1.41	0.386	0.020	−4.61
	10	0.129	51.0	4.90	30.50	0.624	2.727	−1.52	0.386	−0.045	−4.42
	11	0.143	55.2	5.28	23.32	0.662	2.704	−1.61	0.385	−0.107	−4.07
	12	0.157	59.7	5.56	16.47	0.701	2.681	−1.73	0.383	−0.162	−3.47
	13	0.172	64.3	5.75	10.44	0.739	2.655	−1.99	0.380	−0.206	−2.63
	14	0.186	69.1	5.86	5.57	0.777	2.624	−2.49	0.377	−0.237	−1.74
	15	0.200	74.0	5.91	1.65	0.814	2.584	−3.23	0.374	−0.256	−0.98
	16	0.215	78.8	5.91	−2.00	0.851	2.531	−4.22	0.370	−0.265	−0.37
	17	0.229	83.6	5.85	−6.16	0.886	2.463	−5.38	0.366	−0.266	0.11
	18	0.243	88.4	5.73	−11.89	0.921	2.378	−6.44	0.362	−0.262	0.45
	19	0.257	93.0	5.51	−20.20	0.954	2.279	−7.20	0.358	−0.253	0.66
	20	0.272	97.4	5.16	−31.54	0.986	2.171	−7.72	0.355	−0.243	0.87
	21	0.286	101.5	4.61	−45.55	1.017	2.058	−8.18	0.352	−0.228	1.17
	22	0.300	105.0	3.85	−60.60	1.045	1.938	−8.64	0.348	−0.209	1.35
	23	0.315	107.8	2.88	−73.46	1.072	1.811	−8.98	0.346	−0.190	1.08

Figure 5.31 Human gait data that was adapted to exoskeleton measures (source: [43])

From the temporal joint angles in the Winter data the end-effector position of the exoskeleton was calculated using a spreadsheet (figure 5.32). First the joint angles for θ_{21} and θ_{43} were projected onto θ_1 and θ_2 of the exoskeleton and then transformed from degrees to radians. The reference frame of the Winter model was then projected onto the reference frame of the exoskeleton and finally, using the equations for the forward kinematics (see equations 3.1.a and 3.1.b) the temporal end-effector positions of the exoskeleton were calculated.

			Ref:(0,0)	Ref:(0,0)	0.5	0.55				
			Thigh	Leg	Thigh	Leg				
STATE	FRAME	TIME	THETA1	THETA2	THETA1	THETA2	THETA1	THETA2	X	Y
		S	DEG	DEG	DEG	DEG	RAD	RAD	m	m
TOR	1	0	82.7	39.8	97.3	42.9	1.698205	0.748746	-0.48609	0.848008
	2	0.014	85.5	38	94.5	47.5	1.649336	0.829031	-0.47264	0.837072
	3	0.029	88.6	37	91.4	51.6	1.595231	0.90059	-0.45147	0.830849
	4	0.043	91.8	36.7	88.2	55.1	1.53938	0.961676	-0.42527	0.828447
	5	0.057	95.1	37.3	84.9	57.8	1.481785	1.0088	-0.39306	0.831314
	6	0.072	98.3	38.8	81.7	59.5	1.425934	1.038471	-0.35646	0.839395
	7	0.086	101.4	41	78.6	60.4	1.371829	1.054179	-0.31626	0.850968
	8	0.1	104.3	43.8	75.7	60.5	1.321214	1.055924	-0.27347	0.865187
	9	0.114	106.9	47.2	73.1	59.7	1.275836	1.041962	-0.22834	0.881958
	10	0.129	109.2	51	70.8	58.2	1.235693	1.015782	-0.18169	0.899618
	11	0.143	111.2	55.2	68.8	56	1.200787	0.977384	-0.13308	0.917794
	12	0.157	112.8	59.7	67.2	53.1	1.172861	0.92677	-0.08373	0.935799
	13	0.172	114	64.3	66	49.7	1.151917	0.867429	-0.03514	0.952365
	14	0.186	114.9	69.1	65.1	45.8	1.136209	0.799361	0.014312	0.967334
	15	0.2	115.6	74	64.4	41.6	1.123992	0.726057	0.064442	0.97961
	16	0.215	115.9	78.8	64.1	37.1	1.118756	0.647517	0.111572	0.989304
	17	0.229	115.9	83.6	64.1	32.3	1.118756	0.563741	0.157093	0.996351
	18	0.243	115.7	88.4	64.3	27.3	1.122247	0.476475	0.201473	1.000324
	19	0.257	115.2	93	64.8	22.2	1.130973	0.387463	0.241674	1.00166
	20	0.272	114.6	97.4	65.4	17.2	1.141445	0.300197	0.278978	1.000037
	21	0.286	113.7	101.5	66.3	12.2	1.157153	0.21293	0.310626	0.99679
	22	0.3	112.9	105	67.1	7.9	1.171116	0.137881	0.336912	0.991852
	23	0.315	112.1	107.8	67.9	4.3	1.185079	0.075049	0.356245	0.986935
	24	0.329	111.3	109.7	68.7	1.6	1.199041	0.027925	0.367028	0.983654
	25	0.343	110.8	110.7	69.2	0.1	1.207768	0.001745	0.371965	0.981907
	26	0.357	110.3	110.7	69.7	-0.4	1.216494	-0.00698	0.367879	0.983439
	27	0.372	110	109.8	70	0.2	1.22173	0.003491	0.357316	0.987331
HCR	28	0.386	109.6	108.4	70.4	1.2	1.228712	0.020944	0.341333	0.992911
	29	0.4	109.3	106.6	70.7	2.7	1.233948	0.047124	0.322386	0.998978
	30	0.415	109	104.6	71	4.4	1.239184	0.076794	0.301422	1.004999
	31	0.429	108.6	102.5	71.4	6.1	1.246165	0.106465	0.278521	1.010847
	32	0.443	108.4	100.3	71.6	8.1	1.249656	0.141372	0.256166	1.015575
	33	0.458	108.2	98.1	71.8	10.1	1.253146	0.176278	0.233663	1.019499
	34	0.472	107.9	95.9	72.1	12	1.258382	0.20944	0.210214	1.022884
	35	0.486	107.6	93.8	72.4	13.8	1.263618	0.240855	0.187636	1.025386
	36	0.5	106.9	91.9	73.1	15	1.275836	0.261799	0.163586	1.028104
	37	0.515	105.9	90.3	74.1	15.6	1.293289	0.272271	0.139859	1.030863
	38	0.529	104.5	88.9	75.5	15.6	1.317724	0.272271	0.114631	1.033972
	39	0.543	102.9	87.6	77.1	15.3	1.345649	0.267035	0.088593	1.036898
	40	0.558	101.2	86.5	78.8	14.7	1.375319	0.256563	0.06354	1.039452
	41	0.572	99.3	85.4	80.7	13.9	1.408481	0.242601	0.036693	1.041656
	42	0.586	97.6	84.4	82.4	13.2	1.438151	0.230383	0.012458	1.042983
	43	0.601	95.9	83.5	84.1	12.4	1.467822	0.216421	-0.01087	1.043816
	44	0.615	94.3	82.5	85.7	11.8	1.495747	0.205949	-0.0343	1.043887
	45	0.629	92.8	81.6	87.2	11.2	1.521927	0.195477	-0.05592	1.043503
	46	0.643	91.4	80.6	88.6	10.8	1.546362	0.188496	-0.07761	1.042465
	47	0.658	90	79.7	90	10.3	1.570796	0.179769	-0.09834	1.041137
	48	0.672	88.7	78.7	91.3	10	1.593486	0.174533	-0.11911	1.039209
	49	0.686	87.4	77.8	92.6	9.6	1.616175	0.167552	-0.13891	1.037064
	50	0.701	86	76.8	94	9.2	1.640609	0.16057	-0.16047	1.03425
	51	0.715	84.6	75.9	95.4	8.7	1.665044	0.151844	-0.18104	1.031211
	52	0.729	83.1	74.9	96.9	8.2	1.691224	0.143117	-0.20335	1.027389
	53	0.744	81.7	73.8	98.3	7.9	1.715659	0.137881	-0.22562	1.022924
	54	0.758	80.2	72.7	99.8	7.5	1.741839	0.1309	-0.24866	1.017822
	55	0.772	78.8	71.6	101.2	7.2	1.766273	0.125664	-0.27072	1.012359
	56	0.786	77.5	70.2	102.5	7.3	1.788962	0.127409	-0.29453	1.005632
	57	0.801	76.3	68.7	103.7	7.6	1.809906	0.132645	-0.31821	0.998205
	58	0.815	75.3	67.1	104.7	8.2	1.82736	0.143117	-0.3409	0.990286
	59	0.829	74.5	65.2	105.5	9.3	1.841322	0.162316	-0.36432	0.981093
	60	0.844	73.8	63.2	106.2	10.6	1.85354	0.185005	-0.38748	0.971069
	61	0.858	73.4	61.1	106.6	12.3	1.860521	0.214675	-0.40865	0.960667
	62	0.872	73.1	58.7	106.9	14.4	1.865757	0.251327	-0.43109	0.948359
	63	0.887	73.1	56.2	106.9	16.9	1.865757	0.294961	-0.45131	0.935448
	64	0.901	73.4	53.6	106.6	19.8	1.860521	0.345575	-0.46922	0.921853
	65	0.915	74.1	50.9	105.9	23.2	1.848304	0.404916	-0.48385	0.907696
	66	0.929	75	48.1	105	26.9	1.832596	0.469494	-0.49672	0.892334
	67	0.944	76.4	45.4	103.6	31	1.808161	0.541052	-0.50376	0.877595
	68	0.958	78.1	42.9	101.9	35.2	1.778491	0.614356	-0.506	0.863651
	69	0.972	80.2	40.7	99.8	39.5	1.741839	0.689405	-0.50208	0.851358
TOR	70	0.987	82.5	38.9	97.5	43.6	1.701696	0.760964	-0.4933	0.841102

Figure 5.32 Adapted human gait data for exoskeleton

The temporal data of figure 5.32 was introduced to the two developed control system models. In this way the gait pattern of the models could be observed in Simulink. Figure 5.33 presents four frames of the gait movement recorded during the gait simulation of model1.

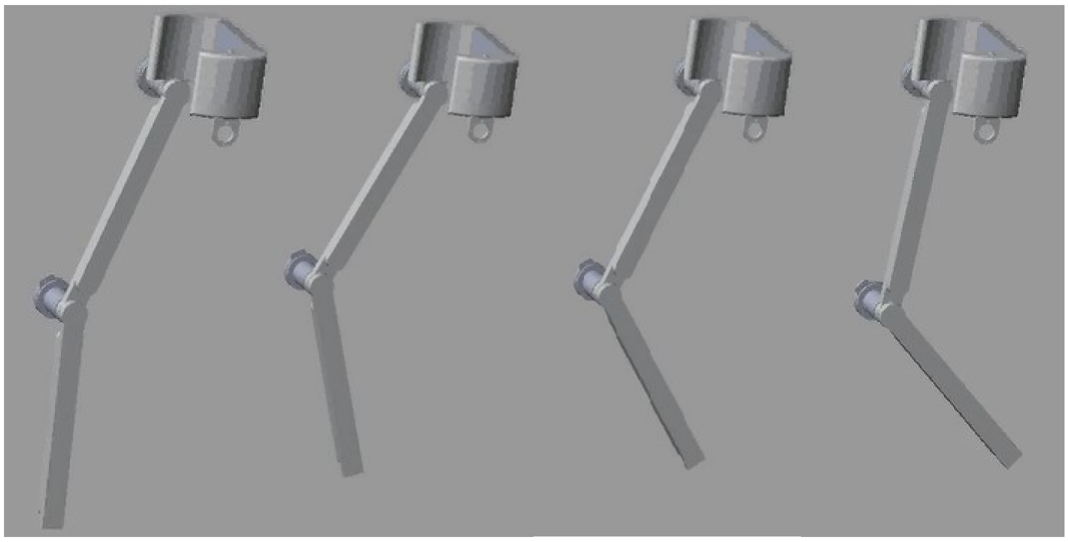


Figure 5.33 Human gait simulation for model1

5.7.CONCLUSION

The first control system model, model1, shows good step response for both motor1 and motor2. In model2 the step response for both motors is well beneath the reference value. The gait pattern observed by introducing the generated gait data showed that model1, in this test also performs far better than model2 as was expected by the step responses. The gait simulation of model1 resembles normal human gait pattern but model2 presented short stiff movements.

Model1 therefore is a better candidate was further developed to build the control system of the exoskeleton.

6. CONCLUSIONS, RECOMMENDATIONS AND FUTURE WORKS

This sections presents the conclusions and recommendations.

6.1.CONCLUSIONS

In this research the step-by-step development of the exoskeleton of the right lower limb and its control system with combined position and velocity control were designed. The design of the physical model of the exoskeleton was developed in Solidworks solid modeling CAD software. This tool has the ability to export models in XML format which can be imported in Matlab/Simulink as a Simmechanics model. The Simmechanics model can then be integrated with Simulink components giving a whole new dimension of possibilities to system design and simulation.

A hardware design model of the inverse kinematics model of the exoskeleton was developed. The inverse kinematics model was developed in VHDL using four different bit widths in floating-point arithmetics. To reduce hardware area maximizing hardware reuse a FSM was developed. For error analysis of the four bit width representations a co-simulation model with Questa Sim was developed in Matlab. The co-simulation has proven to be a valuable tool to perform error analysis. The four bit width representations (27, 32, 45 and 64) were evaluated in numeric precision, area on FPGA and performance. The 27 bit width representation presents the best overall advantages, achieving an error of MSE's of 2,28E-05 and 6,48E-05 for θ_1 and θ_2 respectively, a estiamted consumption of 2647 LUTs, 1224 FFs and 5 DSP blocks.

Two approaches to the exoskeleton control system for combined position and velocity control were developed. The controllers were tuned using a variety of methods such as Ziegler-Nichols, PSO and manual fine-tuning. The parameter values yielded with PSO presented results that eventually converge but needed fine-tuning. Since the PSO was implemented in Matlab, its execution is very expensive in computational time and can take very long (from a few hours to days) depending on the complexity of the object function, the swarm size and the number of iterations.

The Winter human gait dataset has proven to be very useful for gait simulation after being adapted to the configuration, reference system and units of the exoskeleton. Simulation of the human gait with the two combined position and velocity control models presented good resemblance in the first model as was expected by the results of step response analysis.

6.2.RECOMMENDATIONS AND FUTURE WORK

The exoskeleton will be a portable device so careful choices should be made in the choice of the composing components taking into consideration its weight, energy consumption and control system performance. System performance is an important aspect for smooth exoskeleton movements taking into consideration the complex kinematic and control system algorithms. For this purpose they will be designed in HDL and integrated in an FPGA device.

The PSO algorithms and especially the object functions used must be evaluated for their effectiveness. Multivariable optimization, though more complex can also be used to optimize the controller parameters. Root Mean Squared Error (MRSE), Mean Absolute Error (MAE) and other error functions can also be evaluated for use as optimization functions to tune the PID controllers for the exoskeleton.

For faster execution of the PSO, the algorithm and object functions can be developed for faster execution in programming languages such as C++ or java. This however will require a corresponding model of the exoskeleton in the chosen language.

The Winter dataset includes a great variety of human gait data such as linear and angular velocity and acceleration data that can be used in future research for more complex simulations. In this research the exoskeleton gait simulations were visually evaluated for resemblance of the human gait. MSE and other error analysis algorithms can be developed in future research to evaluate the exoskeleton movements against the Winter dataset.

The tuning methods used in this research should be more closely evaluated for their suitability in combined position and velocity control for each model.

REFERENCES

[1] “Hemiplegia : Defintion, Causes, Symptoms, Diagnosis And Treatment,” *Rayur*, 2012. .

[2] S. C. BOTELHO, T. S; NETO, C. D. M; ARAÚJO, F. L. C; ASSIS, “Epidemiologia do acidente vascular cerebral no Brasil,” *Temas em Saúde*, vol. 16, no. 2, pp. 361–377, 2016.

[3] “EC 90 Flat motor Æ 90 mm , brushless , 90 Watt maxon EC motor,” no. August. p. 2003, 2003.

[4] “Planetary Gearhead GP 62 A,” no. July. pp. 2004–2004, 2004.

[5] D. A. Winter, *Biomechanics and Motor Control of Human Movement*. 2009.

[6] J. L. Pons, “Rehabilitation exoskeletal robotics. The promise of an emerging field.,” *IEEE Engineering in Medicine and Biology Magazine*, no. June, pp. 57–63, 2010.

[7] E. Mikołajewska and D. Mikołajewski, “Exoskeletons in neurological diseases - Current and potential future applications,” *Adv. Clin. Exp. Med.*, vol. 20, no. 2, pp. 227–233, 2011.

[8] M. Cenciarini, M. Cenciarini, and A. M. Dollar, “Biomechanical considerations in the design of lower limb exoskeletons Biomechanical Considerations in the Design of Lower Limb Exoskeletons,” no. February, pp. 10–14, 2011.

[9] R. C. Sampaio, “Estado da arte na construção de exoesqueletos,” in *22nd International Congress of Mechanical Engineering*, 2013, no. Cobem.

[10] A. B. Zoss, H. Kazerooni, and A. Chu, “Biomechanical Design of the Berkeley Lower Extremity Exoskeleton (BLEEX),” *IEEE/ASME Trans. MECHATRONICS*, vol. 11, no. 2, pp. 128–138, 2006.

[11] S. K. S. Kim, G. Anwar, and H. Kazerooni, “High-speed communication network for controls with the application on the exoskeleton,” *Proc. 2004 Am. Control Conf.*, vol. 1, pp. 355–360, 2004.

[12] J. F. Veneman, R. Kruidhof, E. E. G. Hekman, R. Ekkelenkamp, E. H. F. Van Asseldonk, and H. van der Kooij, “Design and evaluation of the LOPES exoskeleton robot for interactive gait rehabilitation.,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 15, no. 3, pp. 379–86, Sep. 2007.

[13] K. N. Winfree, P. Stegall, and S. K. Agrawal, “Design of a minimally constraining, passively supported gait training exoskeleton: ALEX II,” *IEEE Int. Conf. Rehabil. Robot.*, pp. 0–5, 2011.

[14] dSPACE, “DS1103 PPC Controller Board RTI Reference,” pp. 1–8, 2010.

[15] U. Onen, F. M. Botsali, M. Kalyoncu, M. Tinkir, N. Yilmaz, and Y. Sahin, “Design and Actuator Selection of a Lower Extremity Exoskeleton,” *IEEE/ASME Trans. Mechatronics*, vol. 19, no. 2, pp. 623–632, Apr. 2014.

[16] Y. Sankai, “Leading edge of cybernics: Robot suit HAL,” *2006 SICE-ICASE Int. Jt. Conf.*, vol. 10, pp. 1–2, 2006.

[17] H. Kawamoto, S. L. S. Lee, S. Kanbe, and Y. Sankai, “Power assist method for HAL-3 using EMG-based feedback controller,” *SMC’03 Conf. Proceedings. 2003 IEEE Int. Conf. Syst. Man Cybern. Conf. Theme - Syst. Secur. Assur. (Cat. No.03CH37483)*, vol. 2, pp. 1648–1653, 2003.

[18] M. Hassan, H. Kadone, K. Suzuki, and Y. Sankai, “Wearable gait measurement system with an instrumented cane for exoskeleton control.,” *Sensors (Basel).*, vol. 14, no. 1, pp. 1705–1722, 2014.

[19] P. Beyl, “Design and control of a knee exoskeleton powered by pleated pneumatic artificial muscles for robot-assisted gait rehabilitation Chair :,” p. 232, 2010.

[20] P. D. Neuhaus, J. H. Noorden, T. J. Craig, T. Torres, J. Kirschbaum, and J. E. Pratt, “Design and evaluation of Mina: A robotic orthosis for paraplegics,” *IEEE Int. Conf. Rehabil. Robot.*, 2011.

[21] J. H. Kim, J. W. Han, D. Y. Kim, and Y. S. Baek, “Design of a walking assistance lower limb exoskeleton for paraplegic patients and hardware validation using CoP,” *Int. J. Adv. Robot. Syst.*, vol. 10, 2013.

[22] M. H. Rahman, M. Saad, J. P. Kenné, and P. S. Archambault, “Control of an exoskeleton robot arm with sliding mode exponential reaching law,” *Int. J. Control. Autom. Syst.*, vol. 11, no. 1, pp. 92–104, 2013.

[23] “3 Steps for Building a Real-Time System With NI Hardware and Software,” *National Instruments*, 2017. [Online]. Available: http://www.ni.com/white-paper/4040/en/. [Accessed: 30-Jul-2017].

[24] J. Kumar, N. Kumar, D. Pankaj, and A. Kumar, “Implementation of Real Time Control Algorithm for Gait Assistive Exoskeleton Devices for Stroke Survivors,” in *2014 International Conference on Electronic Systems, Signal Processing and Computing Technologies Implementation*, 2014, pp. 271–275.

[25] H. Kazerooni, “Exoskeletons for human power augmentation,” *2005 IEEE/RSJ Int. Conf. Intell. Robot. Syst. IROS*, pp. 3120–3125, 2005.

[26] R. Andrade, “Exoskeleton Control of a Programmable Robot Arm Using Field-Programmable Gate Arrays,” *Ohm.Ecce.Admu.Edu.Ph*, no. August, 2014.

[27] W. K. Lee and S. Jung, “FPGA design for controlling humanoid robot arms by exoskeleton motion capture system,” *2006 IEEE Int. Conf. Robot. Biomimetics, ROBIO 2006*, pp. 1378–1383, 2006.

[28] M. V. Mark W. Spong, Seth Hutchinson, *Robot Modeling and Control 1st ed*, vol. 9. 2013.

[29] S. Kucuk and Z. Bingul, “Robot kinematics: forward and inverse kinematics,” in *Industrial Robotics: Theory, Modeling and Control*, no. December, S. Cubero, Ed. Verlag, Germany: Pro Literatur, 2006, p. 964.

[30] S.-L. Chen, Y.-Q. Zhang, and R.-L. Chung, “Fully pipelined CORDIC-based inverse kinematic FPGA design for biped robots,” *Electron. Lett.*, vol. 51, no. 16, pp. 1241–1243, 2015.

[31] Y. S. Kung, K. H. Tseng, C. S. Chen, H. Z. Sze, and A. P. Wang, “FPGA-implementation of inverse kinematics and servo controller for robot manipulator,” *2006 IEEE Int. Conf. Robot. Biomimetics, ROBIO 2006*, no. vi, pp. 1163–1168, 2006.

[32] K. Gac, G. Karpiel, and M. Petko, “FPGA based hardware accelerator for calculations of the parallel robot inverse kinematics,” *IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA*, pp. 1–4, 2012.

[33] B. J. G. Ziegler and N. B. Nichols, “Optimum Settings for Automatic Controllers,” *Trans. ASME*, vol. 64, pp. 759–768, 1942.

[34] J. C. Basilio and S. R. Matos, “Design of PI and PID controllers with transient performance specification - Education, IEEE Transactions on - ieee-edu2002.pdf,” vol. 45, no. 4, pp. 364–370, 2002.

[35] K. Ogata, *Modern Control Engineering*, Fifth edit. Prentice Hall, 2009.

[36] C.-C. Yu, *Autotuning of PID Controllers; A Relay Feedback Approach*, Second edi.

Taipei: Springer, 2006.

[37] J. Kennedy and R. C. Eberhart, “Particle swarm optimization,” *Proc. IEEE Int. Conf. Neural Networks*, pp. 39–43, 1995.

[38] R. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory,” *MHS’95. Proc. Sixth Int. Symp. Micro Mach. Hum. Sci.*, pp. 39–43, 1995.

[39] O. Chao and L. Weixing, “Comparison between PSO and GA for parameters optimization of PID controller,” *2006 IEEE Int. Conf. Mechatronics Autom. ICMA 2006*, vol. 2006, pp. 2471–2475, 2006.

[40] F. Van Den Bergh and A. P. Engelbrecht, “A study of particle swarm optimization particle trajectories,” *Inf. Sci. (Ny).*, vol. 176, no. 8, pp. 937–971, 2006.

[41] H. E. A. Ibrahim, F. N. Hassan, and A. O. Shomer, “Optimal PID control of a brushless DC motor using PSO and BF techniques,” *Ain Shams Eng. J.*, vol. 5, no. 2, pp. 391–398, 2014.

[42] Z. Bingul and O. Karahan, “Tuning of fractional PID controllers using PSO algorithm for robot trajectory control,” *2011 IEEE Int. Conf. Mechatronics, ICM 2011 - Proc.*, pp. 955–960, 2011.

[43] B. Allaoua, B. Gasbaoui, and B. Mebarki, “Setting Up PID DC Motor Speed Control Alteration Parameters Using Particle Swarm Optimization Strategy,” *Leonardo Electron. J. Pract. Technol.*, vol. 14, no. 14, pp. 19–32, 2009.

[44] M. Nasri, H. Nezamabadi-pour, and M. Maghfoori, “A PSO-Based Optimum Design of PID Controller for a Linear Brushless DC Motor,” *Int. J. Electr. Robot. Electron. Commun. Eng.*, vol. 1, pp. 184–188, 2007.

[45] “Simscape Multibody Link Add-in.” [Online]. Available: https://www.mathworks.com/products/simmechanics. [Accessed: 14-Jun-2017].

[46] T. Weise and T. Weise, “Global Optimization Algorithms – Theory and Application –,” 2009.

[47] C. R. Stanley and D. Lawie, “Average relative error in geochemical determinations: Clarification, calculation, and a plea for consistency,” *Explor. Min. Geol.*, vol. 16, no. 3–4, pp. 267–275, 2007.

[48] C. C. Wong and C. C. Liu, “FPGA realisation of inverse kinematics for biped robot based on CORDIC,” *Electron. Lett.*, vol. 49, no. 5, pp. 332–334, 2013.

[49] C. Moler, “F l oa ting poi n t s.” MathWorks, pp. 2–4, 1996.

[50] B. D. Bishop, “Floating point package user ’ s guide,” no. February, 2008.

[51] H. Reference, “Escon 70/10,” no. November. 2015.

[52] V. Mhase, K. R. Sudarshan, O. Pardeshi, and V. Prasheel, “Integrated Speed – Position Tracking with Trajectory Generation and Synchronization for 2 – Axis DC Motion Control,” vol. 1, no. 6, pp. 61–66, 2012.

[53] R. Poli, J. Kennedy, and T. Blackwell, “Particle swarm optimization,” *An overview. Swarm*, vol. telligence, no. 1, p. 33—57, 2007.

[54] M. Clerc and J. Kennedy, “The particle swarm-explosion, stability, and convergence in a multidimensional complex space,” *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, 2002.

[55] I. Todić, M. Miloš, and M. Pavišić, “Position and Speed Control of Electromechanical Actuator for Aerospace Applications,” *Tech. Gaz.*, vol. 20, no. 5, pp. 853–860, 2013.

[56] K. H. Ang, G. Chong, and Y. Li, “PID control system analysis, design, and technology,”

IEEE Trans. Control Syst. Technol., vol. 13, no. 4, pp. 559–576, 2005.

[57] D. A. Winter, *Biomechanics and Motor Control of Human Movement*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2009.

APPENDIX

Title: Control System Design for an Exoskeleton of the Right Lower Limb

Marlon W. Koendjbiharie[1], Daniel M. Muñoz[1], Carlos H. Llanos[1]

[1]*Universidade de Brasília, Programa de Pós-graduação em Sistemas Mecatrônicos, Brasília, Brazil*
e-mail: m.koendjbiharie@gmail.com

Abstract: The control system design for position control of a wearable exoskeleton of the right lower limb is proposed using a speed controller. A great variety of exoskeletons has already been developed but research in this field is still under development because of the variety of applications and ongoing introduction of innovative technology [1], [2]. The proposed exoskeleton is designed for user mobility improvement for people with movement deficiency in the right leg. The mechanical design of the exoskeleton consists of a model developed in Solidworks, which was imported into MATLAB/Simmechanics, upon which the design of the control system is based. The control system, which is responsible for the behaviour of the exoskeleton is one of the most important aspects of this system. The control system uses feedback loops to adjust the movements of the two exoskeleton motors, one in the hip- and one in the knee joint. Each motor is equipped with a gear and the motor encoders deliver the actual values of speed and position of each joint as feedback to the control system.

Keywords: exoskeleton, kinematics, position control, velocity control.