Design and Development of an Anthropomorphic Hand Prosthesis

by

André Rui Dantas Carvalho
B.Sc., Instituto Superior Técnico, Lisboa, 2005
MASc., University of Victoria, 2007

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Mechanical Engineering

© André Rui Dantas Carvalho, 2011
University of Victoria

All rights reserved. This dissertation may not be reproduced in whole or in part, by photocopying or other means, without the permission of the author.

Design and Development of an Anthropomorphic Hand Prosthesis

by

André Rui Dantas Carvalho
B.Sc., Instituto Superior Técnico, Lisboa, 2005
MASc., University of Victoria, 2007

Supervisory Committee

Dr. A. Suleman, Supervisor
(Department of Mechanical Engineering)

Dr. D. Constatinescu, Departmental Member
(Department of Mechanical Engineering)

Dr. P. Agathoklis, Outside Member
(Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. A. Suleman, Supervisor
(Department of Mechanical Engineering)

Dr. D. Constatinescu, Departmental Member
(Department of Mechanical Engineering)

Dr. P. Agathoklis, Outside Member
(Department of Electrical and Computer Engineering)

ABSTRACT

This thesis presents a preliminary design of a fully articulated five-fingered anthropomorphic human hand prosthesis with particular emphasis on the controller and actuator design. The proposed controller is a modified artificial neural network PID-based controller with application to the nonlinear and highly coupled dynamics of the hand prosthesis. The new solid state actuator has been designed based on electroactive polymers, which are a type of material that exhibit electromechanical behavior and a liquid metal alloy acts as the electrode. The solid state actuators reduce the overall mechanical complexity, risk failure and required maintenance of the prosthesis.

Contents

List of Tables

List of Figures

Acronyms

JPL	Jet Propulsion Lab
MIT	Massachusetts Institute of Technology
DoF	Degree of Freedom
NASA	National Aeronautics and Space Administration
EMG	Electromyogram
ACT	Anatomically Correct Testbed
EAP	Electroactive Polymer
CMC	Carpometacarpal
MCP	Metacarpophangeal
IP	Interphalangeal Joint
PIP	Proximal Interphalangeal Joint
DIP	Distal Interphalangeal Joint
ABA	Articulated-body Algorithm
DCA	Divide-and-Conquer Algorithm
HDIA	Hybrid Direct/Iterative Algorithm
CM	Center of Mass
ANN	Artificial Neural Network
ADELINE	Adaptive Linear Element

MLP	Multi-layer *perceptron*
BPA	Back-propagation Algorithm
BFGS	Broyden-Fletcher-Goldfarb-Shanno
DFP	Davidon-Fletcher-Powell
L-BFGS	Limited memory BFGS
NOVEL	Nonlinear Optimization via External Lead
ARX	Autoregressive Model with External Input
ARMAX	Autoregressive Moving Average Model with External Input
NARX	Nonlinear ARX
NARMAX	Nonlinear ARMAX
LTI	Linear Time Invariant
ZOH	Zero-order Hold
TDL	Tapped Delay
SISO	Single Input Single Output
MIMO	Multiple Input Multiple Output
MISO	Multiple Input Single Output
SIMO	Single Input Multiple Output
LF	Linear Function
HT	Hiperbolic Tangent
PID	Proportional-Integral-Derivative
IMC	Internal Model Control
PZT	Lead Zirconate Titanate
EAC	Electroactive Ceramics

DC	Direct Current
RC	Resistance-Capacitor
AC	Alternated Current

Acknowledgements

I would like to thank:

Dr. Afzal Suleman for believing in me and giving this opportunity of a lifetime.

Fundação para a Ciência e Tecnologia for funding my graduate studies on Canada under the scholarship no. FRH / BD / 22861 / 2005.

Ricardo Paiva for being my friend and partner in crime during my time in Canada.

The Rocha Family - Bruno, Joana and Helena for their friendship and for being there when I most needed.

Kerem Karacoç, Bariş Ulutas and Casey Kuelen for being my friends and the life of our office.

Sandra Makosinski for being the most efficient person I know and without whom we would all be lost.

Art Makosinski for all the help setting up the experimental part in Canada.

Dr. Agostinho Fonseca for all the help setting up the experimental part in Portugal and making sure I would't die in the progress

Wayne and Veronica Psotka for welcoming me in their home and helping me settling in Victoria.

My friends in Portugal - Pedro Abrantes, Jorge Faria and Inês Gonçalves for helping me seeing that there is life outside my PhD, while nagging me to finish it.

Ana Castanhito Almeida, my "sister" for supporting me along the way and sharing the pain of being away from our family.

My parents - Rui and Ema Carvalho for the support and believing in me while bearing my long absences from home.

DEDICATION

To my parents...

Chapter 1

Introduction

Losing a limb is an extremely traumatic event that forever changes one's life, especially in the case of the hand. The human hand is our main manipulator and its shape and dexterity is one of the evolutionary characteristics that make us human. Apart from the obvious physical impairment, the loss of a hand has a profound impact in a person's social and emotional well-being as well.

When designing a prosthesis, one should try to attain the following objectives as close as possible [1]:

Lightweight – any device is worn by the operator on the end of a closely fitting external socket, hence the weight bears directly onto the skin of the stump. The lever-arm created is therefore large and the weight can obstruct blood flow in the underlying skin and results in symptoms ranging from discomfort to skin breakdown.

Compact – the user population varies widely in the length of their residual limb, so any device should retain all its drives and power sources within as small an envelope as possible, preferably within the hand profile.

Reversible – the use of a modular solution ensures that the largest number of people can use the device as well as providing simplicity of manufacture of both the left and right hands.

Quiet – the purpose of all prostheses is to provide functionality without attracting undue attention to the user. The sound of gears and motors or the escape of gas in a pneumatic system is therefore generally undesirable.

Appearance – the device must be aesthetically pleasing. Anthropomor-

phism is a key criterion that often contributes to a users acceptance of the device. Additionally it is desirable that it should be a non obtrusive solution. Finally, both the static as well as the dynamic appearance is important.

Price – the device must be useable by a wide range of possible wearers. The prosthetics are currently expensive. This means the device must be produced at a cost that will allow the hand to be priced competitively.

Another way to classify the performance of a prosthesis or a anthropomorphic manipulator is by the tasks it can do. There are six basic tasks a hand can do: cylindrical grasp, precision grasp, hook prehention, tip grasp, spherical grasp and lateral hip, Fig. 1.1.

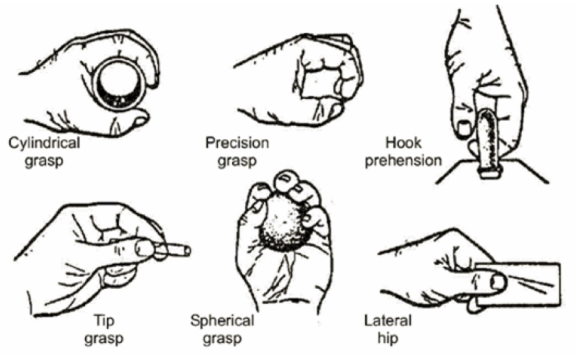


Figure 1.1: The six basic hand tasks [2].

Research in prostheses and anthropomorphic manipulators is inherently interconnected where usually the development of the latter precedes the former. In 1991, the research consisted mainly in mechanically artificial hands with three to five fingers [3]. At the same time, artificial hands, like the Standford/JPL, the Utah/MIT or the Belgrade/USC Dextrous hands, were already being commercialized. The Standford/JPL hand is composed of three fingers, in which one is a thumb-like finger, Fig. 1.2. This hand has twelve actuated joints, three for each finger and, because the fingers are actuated by a tendon system where each must be actuated by a set of four servomotors [3].

The Utah/MIT hand [3], has four 4-DOF fingers (one of them is a thumb) actuated by 32 independent tendons and pneumatic cylinders: 16 to close the hand and 16 to open, Fig. 1.3. This hand, although more anthropomorphic, has two disadvantages:

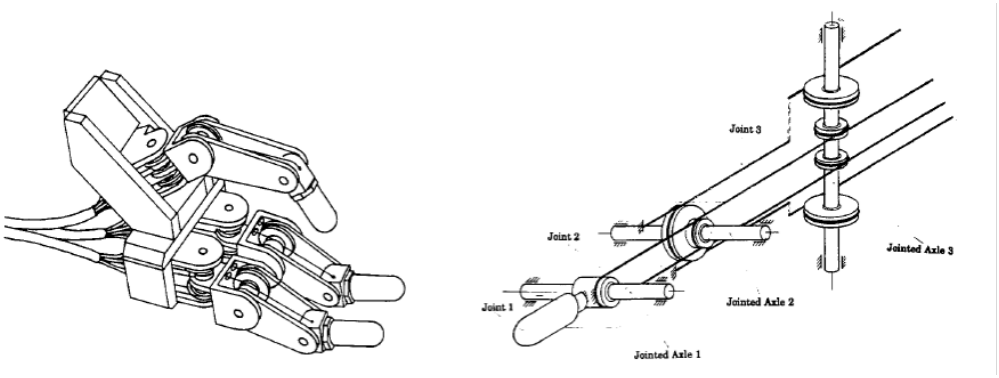


Figure 1.2: Standford/JPL hand [3].

it has a high number of actuators, and, because of the rigid tendons, the ulnar motion (the spreading of the fingers) affects the fingers flexion/extension movements.

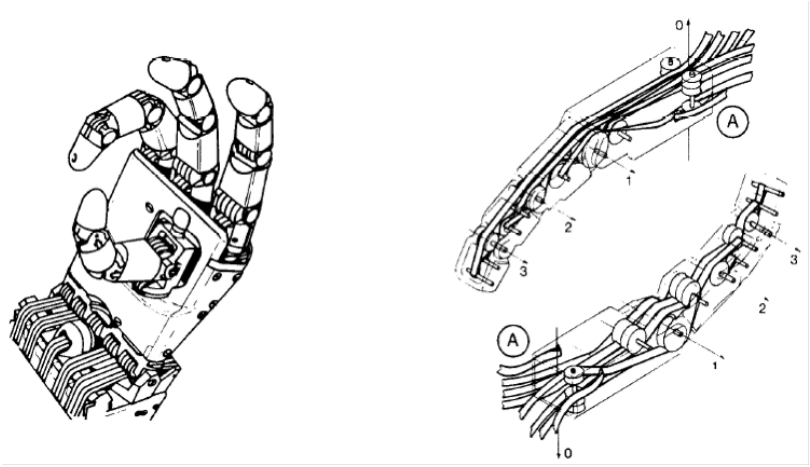


Figure 1.3: Utah/MIT hand [3].

The Belgrade/USC hand [3], has five fingers with four servomotors: one for each pair of fingers and two for the thumb, Fig. 1.4. This hand has a very limited motion, because only two servomotors actuate the four fingers and, consequently, the hand can only grasp objects [3]. The authors in [3] also have proposed another type of hand, a three-finger manipulator with 3-DOF in each finger, Fig. 1.5.

In reference [4], the authors introduce an anthropomorphic hand with five fingers and one thumb, actuated by pneumatic artificial muscles consisting in rubber sleeves that shorten their length when inflated, Fig. 1.6.

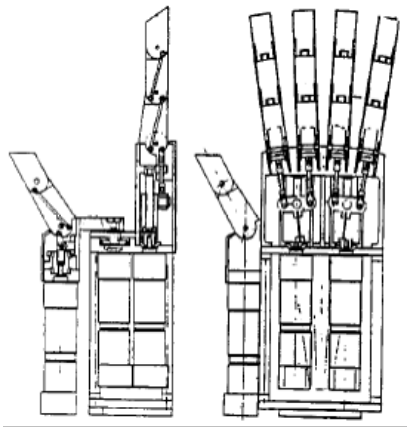


Figure 1.4: Belgrade/USC hand [3].

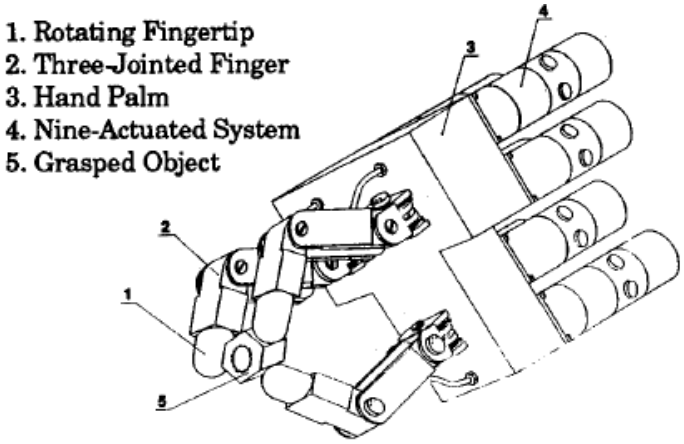


Figure 1.5: Three-finger manipulator [3].

These muscles are placed on a skeletal framework, similar to the human hand bone structure in approximately the same arrangement of the natural muscles. Although the final artificial hand has less muscles than the human counterpart, it can grasp, hook grip and finger stretch, Fig. 1.7.

The main disadvantages of this hand are its bulkiness (compared to the natural one) and the support system needed for the hand to function: electromagnetic valves and an air compressor.

The Graspar hand [5], has two 3-DOF fingers and one 2-DOF thumb and is mainly a grasping hand. This hand is actuated by electric servomotors and a tendon system, Fig. 1.8.

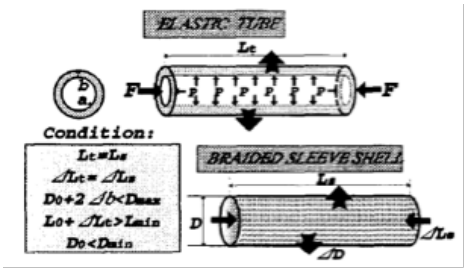


Figure 1.6: Pneumatic artificial muscle [4].

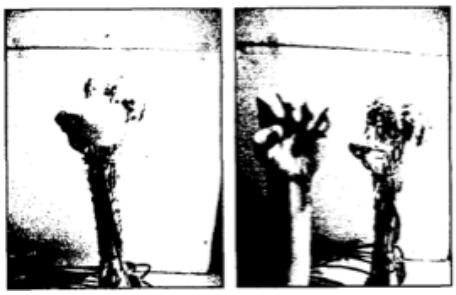


Figure 1.7: Skeletal pneumatic artificial hand [4].

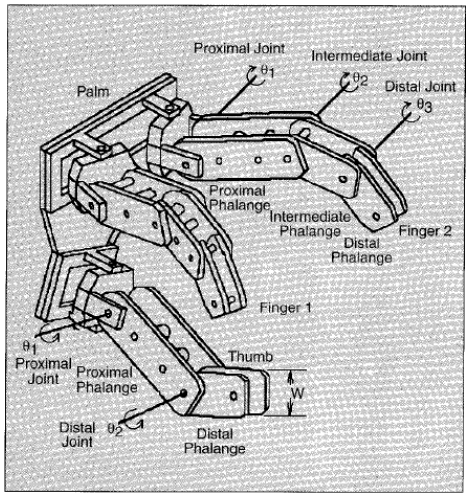


Figure 1.8: Graspar hand [5].

The Robonaut, is a space robot by NASA [6]. This robot is a one legged anthropomorphic robot made to resist the harsh conditions of space and have at least the same manipulation capabilities of an astronaut, Fig. 1.9. The hand has 14 actuated joints and 5 fingers (one of them a thumb), and is actuated using a tendon system

and 14 electric motors, Fig. 1.10.

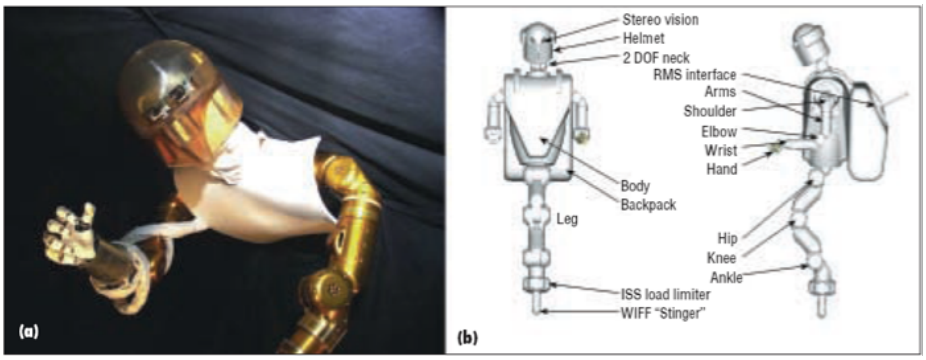


Figure 1.9: NASA's Robonaut [6].

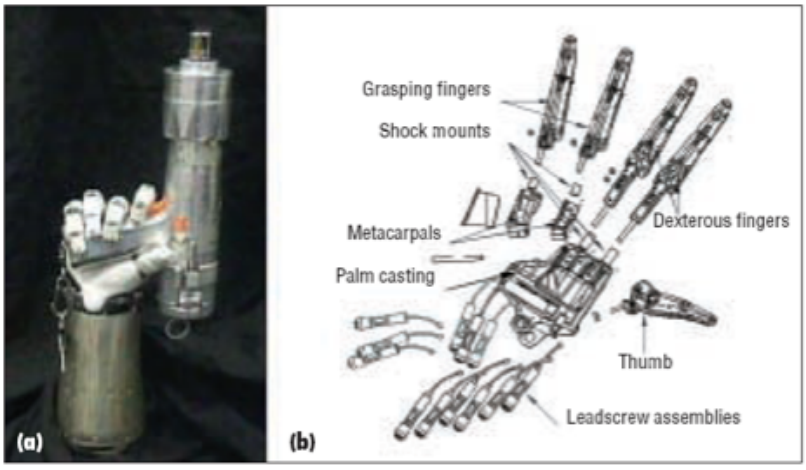


Figure 1.10: Robonaut robotic hand [6].

This is one of the most advanced artificial hands until today, and it is capable of grasping and manipulating objects, Fig. 1.10(a). The hand, along with the rest of the robot, is teleoperated [6].

The Oxford and Manus prostheses are 3-finger artificial hands, driven by electrical motors and a tendon system. The main difference between them is the way the tendon system is constructed, Fig. 1.11 [7].

The fingers in both hands are 2-DOF, but the Oxford hand, Fig. 1.11(a), uses rigid links as tendons, unlike the Manus hand, Fig. 1.11(b), and it uses a crossed-tendon mechanism with steel wires. To become more anthropomorphic, both have

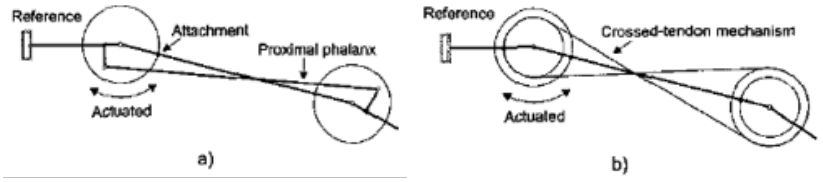


Figure 1.11: Oxford and Manus hand tendon system [7].

two passive fingers, rigid in the case of the Oxford hand, or manually deformable in the case of the Manus hand.

Alongside with the mechanical and structural parts, the control system also has an important part in the design of the artificial hand. Nowadays, the more advanced prosthesis control system is the Electromyogram (EMG) control. With the EMG, an amputee can control the prosthesis directly with his brain, like one would control the natural hand [8].

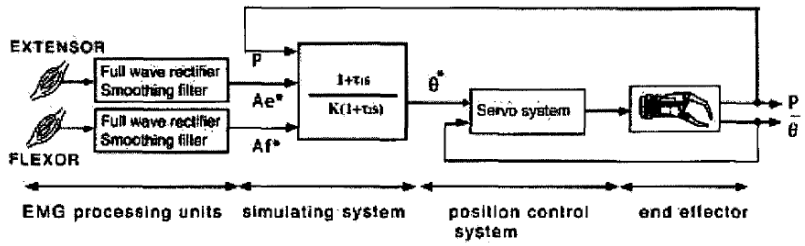


Figure 1.12: EMG control block diagram [8].

However, this type of prosthesis control, Fig. 1.12, imposes limitations on the prosthesis itself. Because of its complex design, EMG controlled hands only have a small number of actuated joints, thus limiting its capabilities. One example of an EMG controlled hand can be seen in Fig. 1.13.

Another prosthesis with three fingers (two fingers and a thumb) is the Leverhulme/Oxford Southampton Hand prosthesis [1]. This hand is a motor and gear driven manipulator that can perform grip movements, Fig. 1.14, and, although being mechanical-based, it only weights 964g while being capable of delivering a 45N grip force and being fully closed in less than 1.2s [1].

In 2004, the Anatomically Correct Testbed hand (ACT hand) was presented [9, 10]. This hand was designed to study the natural hand movements and dynamics, and not for an application in robotics or prostheses. Nevertheless, this hand is important,

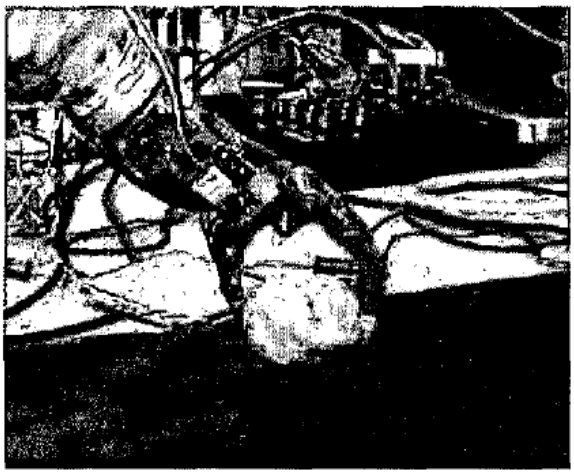


Figure 1.13: EMG controlled hand [8].

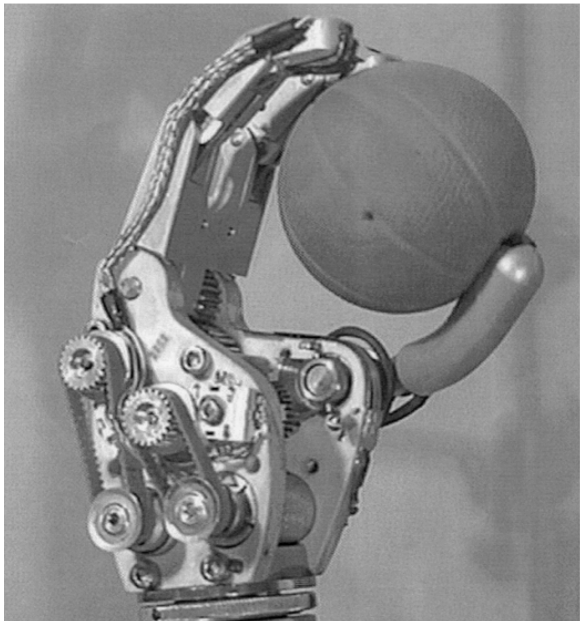


Figure 1.14: Leverhulme/Oxford Southampton Hand prosthesis (LO/ SH). A clinical version of the Southampton Hand series. It has two independent motions and is driven by permanent magnet dc motors [1].

since it tries to mimic the natural hand by constructing a similar muscular-skeletal structure. This hand uses servomotors and a tendon system as actuators. Both bone and tendon structures are made to be similar to the human hand, Fig. 1.15.

The Liquid-fueled hand (2006) [11], and the Jung-Kang-Moon hand (2008) [2],

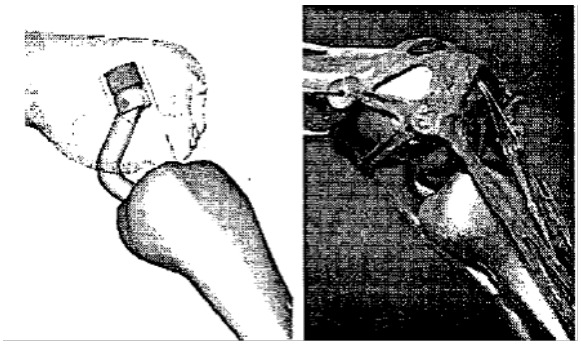


Figure 1.15: Finger joint ACT hand [9].

were proposed. The first one proposes a rather unique actuation method. While still being a fluid-based hand, moved by a set of parallel cylinders and valves, Fig. 1.16, it is in the fluids origin and storage that lies the novelty. The fluids (compressed Carbon Dioxide and monopropellant Hydrogen Peroxide) are stored on two small containers and are mixed in a catalyst chamber resulting in the decomposition of the Hydrogen Peroxide in a highly exothermical reaction. The resultant thermal energy is transduced to mechanical work via the expansion of the gaseous reaction products [11].

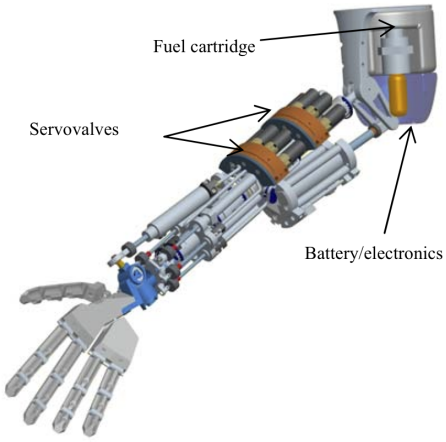


Figure 1.16: Solid model of the Liquid-fueled hand [11].

The Jung-Kang-Moon hand is a more conventional hand. It is a tendon-driven hand actuated by a set of servomotors located on the wrist, Fig. 1.17. The special characteristic of this hand is that, on par with the NASA Robonaut hand, it is a five

fingered dextrous hand. It can perform a high number of hand movements while only weighting 400.72g [2].

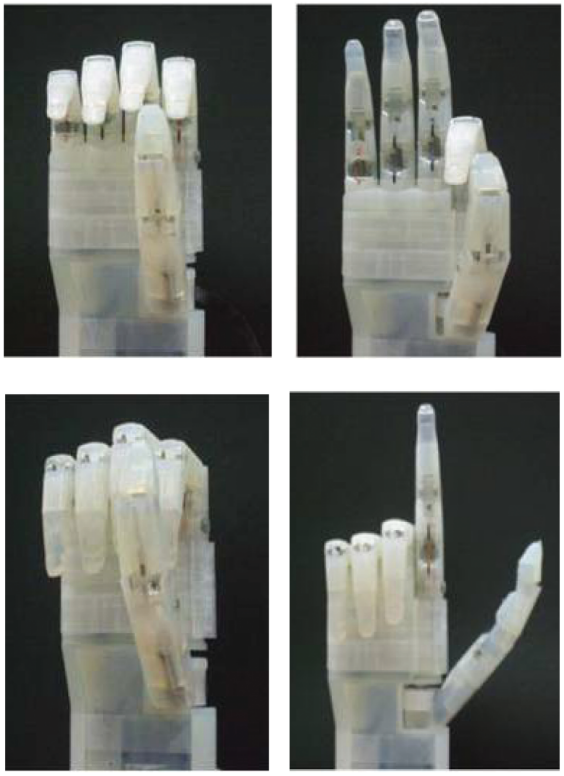


Figure 1.17: Jung-Kang-Moon hand in four positions [2].

More recently, a series of commercially available hands have been introduced in the market, most notably the Touch Bionics i-LIMB [12], and the BeBionics Hand [13]. Both hands are fully articulated five-fingered prosthetic hands powered by small servomotors that provide a relatively high movement speeds while being able to generate sufficient force and, for the BeBionics hand, weighing around 500g.

However, the greatest achievement of these hands is they are one of the first commercially available EMG based prosthesis.

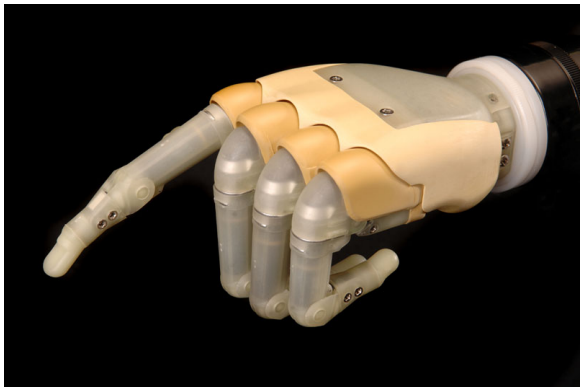


Figure 1.18: Touch Bionics i-LIMB [12].

Figure 1.19: BeBionics hand prosthesis [13].

1.1 Motivation

Analyzing the characteristics of each prosthesis, one can observe some limitations, such as limited movement range, limited generated force and having only three of four moveable fingers (usually the extra fingers are for aesthetical purposes only).

When analyzing dexterity, current prostheses are only able to perform a limited set of tasks (most commonly cylindrical grasp and precision grasp). The most recurrent cause for the lack of dexterity is the reduced number of active fingers (fingers that can move) that can be seen in prostheses such as the Standford/JPL, Utah/MIT, Graspar or Leverhulme/Oxford Southampton hands. Another cause for a reduced dexterity is a reduced number of actuators, leading to a smaller number of independently actuated joints. This is the case of the Belgrade/USC or the Jung-Kang-Moon hands.

Another limitation of the existing prostheses is the actuation principle. Servomotors are the most the popular actuation medium (servomotors are used on the Standford/JPL, Belgrade/USC, Guo-Gruver-Qian, Graspar, Robonaut, Leverhulme/Oxford, ACT, Jung-Kang-Moon, i-LIMB and the BeBionic hands) and exist in almost every size, shape, power and price, but all share the same disadvantages: they are mechanisms and as such prone to mechanical failure, and, unless directly applied to the joints or through a complex set of gears and belts, they are poor substitute for muscles mainly because of their actuation principle (servomotors are rotation-based while muscles are translation-based). Another popular actuation principle is the pneumatic/hydraulic piston, which is used prostheses such as the Utah/MIT hand [3], the Lee-Shimoyama hand [4], and the Liquid-fueled hand [11]. With the exception of the case of the latter, fluid-based actuators share the same set of disadvantages: they need a pressurized container to store the fluids, a pump to re-pressurize the fluid and they are very noisy systems (especially pneumatic actuators). The Liquid-fueled hand is a special case in the fluid-based actuators, because the pressurized fluid is generated on site, it solves the first disadvantage of fluid-based actuators. Also, as the authors claim, the resulting system is quiet [11], when compared to other hydraulic systems. However, this system comes with a disadvantage, since the fuel fluids are not readily available (unlike the air in pneumatic system) when they are spent, the user must look for resupplier or carry an extra fueling system.

The main objective of this work is to pave the way for the development of a human hand prosthesis able to perform the six basic tasks while trying to fulfill the requirements mentioned above. To achieve the desired dexterity, the prosthesis

has been designed based on the human hand anatomy (explained in more detail on section 2), with compressive translational actuators mounted accordingly with the muscle arrangement, which includes actuators both on the hand itself and on the forearm (in the case of the latter, if the amputee still has the forearm or part of it, they can be mounted around the stub). Also, a model of the real bone structure will be used as the supporting structure (similarly to what was done by [9, 10] on the ACT hand). The use of a model will not only provide an accurate approximation of the optimal support but will, also, reduce the manufacturing costs and time due to being commercially available. Another benefit of using a model is that it gives the prosthesis a more anthropomorphic look. This work is focused on the development of the two essential components of the hand prosthesis: the neural network controller design and the novel actuators.

Using Neural Networks, which are capable of modeling complex dynamic system with both nonlinearities and coupling, the resulting controller will be able to control the 17 joints (with some of them having 2 degrees of freedom) while dealing with the heavy coupling (created by the rigid-body dynamics of the skeleton and by the muscle arrangement) and variable saturations of the metacarpophalangeal joints.

The actuators have been specially designed to emulate the natural human hand muscles and they are based on the dielectric electoactive polymers (EAP). Dielectric EAPs are a special kind of polymers that can react to the presence of an electrical field, making compact actuators that can have a high range of compression rates and generated forces. These properties make the dielectric EAPs potential substitutes for the natural muscles. Another advantage of the proposed dielectric EAP actuators is that they are solid-state actuators, or in order words, they do not have moveable parts, which greatly reduces the complexity and failure probability.

1.2 Thesis Layout

This thesis is divided into five chapters, starting with the introduction. Chapter 2 will focus on the anatomy of the human hand and the development of a mathematical model. Next, the thesis document explains the basic mathematics and theory behind Artificial Neural Network and their capabilities, which are used in conjunction with the mathematical model of the hand, to create a nonlinear controller for the prosthesis. With the controller design completed and evaluated,the design of the actuators was performed to provide the motor capabilities to the prosthesis. Finally, the conclusions

are presented along with recommendations for future work.

Chapter 2

Dynamics of the Human Hand

2.1 The Hand

The human hand is the most complex muscular-skeletal system in the human body. Apart from being our main manipulator, the hand, is also responsible for the majority of input information from our touch sense. The hand can be divided into three subsystems: the skeletal system, the muscular system, and the dermo-neurological system [14]. This section will exclusively detail the first two.

2.1.1 Skeletal System

The bones are the structural basis of the hand: they support the muscles and tendons, and give form to the hand itself. A normal human hand has 26 bones divided into three categories: the carpal bones, the metacarpal bones and the phalanxes; and 17 joints divided into four categories: the wrist joint, the carpometacarpal joints, the metacarpophalangeal joint, and the digital joints [14, 15].

Bones

The carpals are located on the lower part of the hand, Fig. 2.1. Although independent from each other, the Carpal bones are fixed and act as a solid block, making a progressive transition between the two wrist bones and the five Metacarpal bones.

The five Metacarpal bones give form to the palm and are the largest bones in the hand, Fig. 2.2. Only the first and the fifth metacarpals (the leftmost and the rightmost, respectively) have active movements, whereas the second metacarpal has

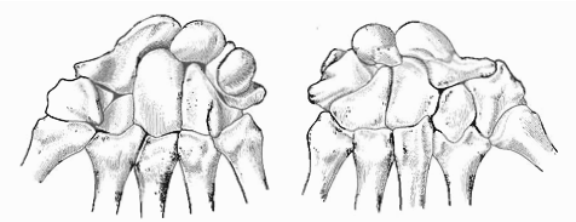


Figure 2.1: Carpal Bones [15].

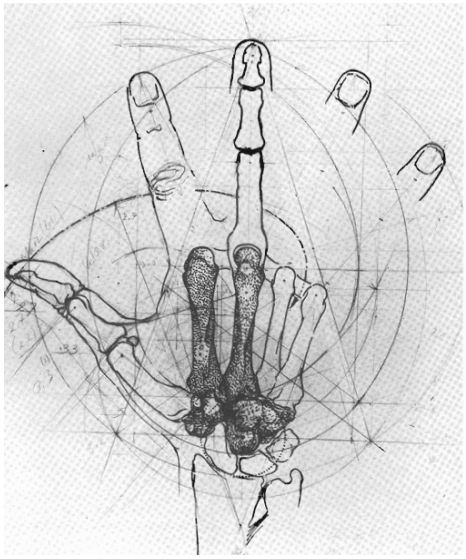


Figure 2.2: Metacarpal Bones [15].

a passive movement and the last two are fixed along with the Carpal bones. The Phalanges are the bones of the fingers and are denominated, from the base to the tip: proximal, middle and distal (with the exception of the thumb that only has the proximal and distal phalanxes) [14, 15].

Joints

The wrist joint has two degrees of freedom (dof): lateral movement of the hand, also called abduction/adduction (Fig. 2.3A and B, respectively), and extension/flexion. One characteristic of this joint is the coupling between the dofs when the hand flexes or extends [15].

The carpometacarpal (CMC) joints are mainly fixed joints except for the first and fifth joints. The first joint has two degrees of freedom: rotation around an axis

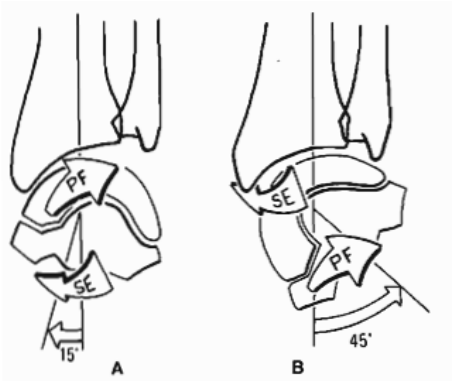


Figure 2.3: Wrist Joints [15].

formed by the second metacarpal bone, and flexion/extension. Both movements are depicted in Fig. 2.4. The fifth metacarpal joint has only a small rotation movement

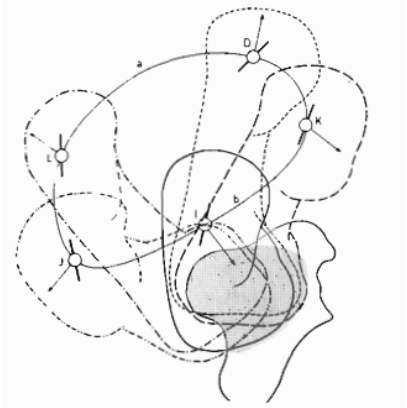


Figure 2.4: First carpometacarpal joint [15].

around an axis formed by the fourth metacarpal bone.

The metacarpophalangeal (MCP) joints (also called knuckles) are the first joints of the finger and all have two degrees of freedom (abduction/adduction and flexion/extension), with the sole exception of the thumb that has only one (flexion/extension). There is a small particularity in the second to fifth MPC joints, the extent of the abduction/adduction decreases with the increase of flexion [15]. The digital joints, the proximal joint (PIP) and the distal interphalangeal joint (DIP), are the main joint of the finger and only have flexion/extension movements.

2.1.2 Muscular System

All the hand muscles are skeletal muscles and can be divided into two categories: the extrinsic muscle and the intrinsic muscles. The extrinsic muscles are located on the wrist and forearm, Fig. 2.5 and Fig. 2.6, and are responsible for the wider movements of the fingers. They are, also, the muscles with more strength and are the last to be actuated. The functions, actuation characteristic and generated work of these muscles are detailed on Table 2.1[1] [14, 15, 16].

Table 2.1: Extrinsic Muscles: Actions and Generated Work [15, 14].

Name	Action	Active Joints	Generated Work (J)
Extensor Digitorum	Extends the four fingers simultaneously	Both IP joints	15.60
Extensor Indicis	Extends the index finger	Distal IP joint	4.59
Extensor Digiti Minimi	Extends the little finger	Distal IP joint	–
Extensor Carpi Radialis	Extends and abducts the hand	Wrist joint	10.10
Extensor Carpi Ulmaris	Extends and adducts the hand	Wrist joint	10.09
Palmaris Longus	Extends the hand	Wrist joint	0.92
Extensor Pollicis Longus	Extends the thumb	IP joint of the thumb	0.92
Extensor Pollicis Brevis	Extends the thumb	MCP joint of the thumb	0.92
Flexor Digitorum Profundus	Flexes the four fingers simultaneously	Distal IP joints	41.31
Flexor Digitorum Superficialis	Flexes the four fingers simultaneously	Proximal IP joints	44.06
Flexor Carpi Radialis	Flexes and abducts the hand	Wrist joint	7.34
Flexor Carpi Ulnaris	Flexes and adducts the hand	Wrist joint	18.36
Flexor Pollicis Longus	Flexes the thumb	IP joint of the thumb	11.02
Abductor Pollicis Longus	Abducts and extends the thumb	CMC joint of the thumb	0.92 - 3.67

The intrinsic muscles, Table 2.2, are located on the hand itself and are responsible for small amplitude and strength movements. They are also used to fine-tune the movements done by the extrinsic muscles [14, 15].

There are some remarks on the way the muscles actuate on the hand. The first is the intricate interaction between the Extensor Digitorum, the Dorsal and Palmar Interossei, and the lumbricals, called the Extensor Apparatus, Fig. 2.7 [14, 15].

[1]Please note that the generated work values are average values. Real values can vary greatly depending on the subject and how they were obtained [15].

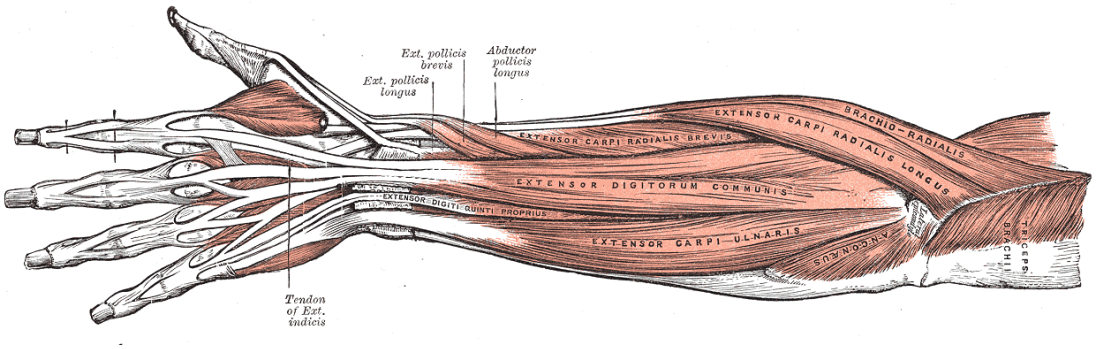


Figure 2.5: Extensor extrinsic muscles [14].

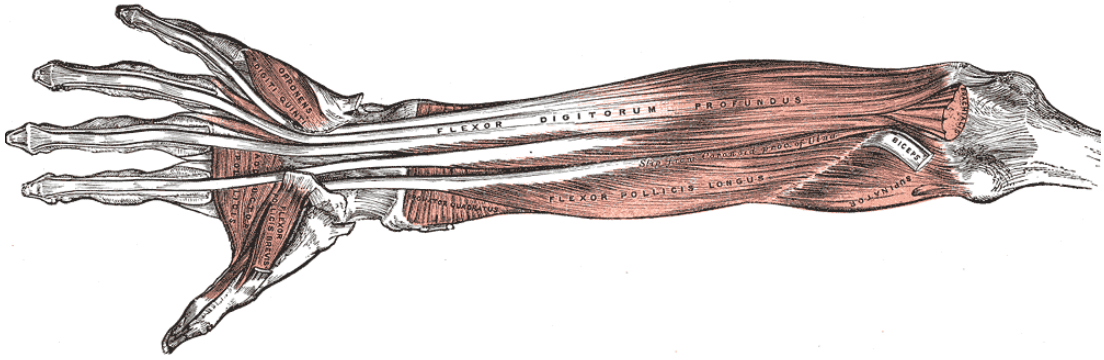


Figure 2.6: Flexor extrinsic muscles [14].

The way these muscles interact and act on the fingers is very complex: the Extensor Digitorum has the sole function of extending the fingers, but depending on the state of the interossei muscle, the Extensor Digitorum can actuate only the distal IP joint or only the proximal IP joints. As the fingers start to extend, the interossei hood (a ligament structure on top of the proximal phalanges connecting the opposing intesossei) moves in the direction of the MCP joint shifting the action of the lumbricals from extension of the MCP joints to flexion of the same joints, Fig. 2.8 [15]. Since the Interossei are directly connected to the tendon of the Extensor Digitorum, rather than to the bone, they assist also in the extension of the fingers.

Another remark is how the muscles are actuated. When the hand grabs and pulls something, the first bones to be actuated are the intrinsic muscles. If the brain perceives that the muscles are not producing sufficient movement, the extrinsic muscles start to be actuated in small groups of fibers, depending on the force needed.

Table 2.2: Intrinsic Muscles and their actions [15, 14].

Name	Action	Active Joints
Palmar Interossei	Adduct the index, ring, and little fingers toward the axial line through the third digit. Assist in flexion of MCP joints and extension of IP joints of the three fingers	MCP joints of the index, ring and little fingers
Lumbricals	Extend the IP joints and simultaneously flex the MCP joints of the second through fifth digits. The lumbicals also extend the IP joints when the MCP joints are extended	MCP and IP of the four fingers
Opponens Digiti Minimi	Opposes the little finger to the thumb.	CMC joint of the little fingers
Abductor Digiti Minimi	Abduct the little finger	MCP joint of the little finger
Flexor Digiti Minimi	Flexes the little finger and helps the Opponens Digiti Minimi	MCP joint of the little finger
Adductor Pollicis	Adduct the metacarp of the thumb and flex the thumb	CMC and MCP joints of the thumb
Flexor Pollicis Brevis	Flex the thumb	MCP joint of the thumb
Abductor Pollicis Brevis	Abducts and extends the thumb	MCP joint of the thumb
Opponens Pollicis	Opposes the thumb to the little finger.	CMC joint of the thumb
Dorsal Interossei	Abdutcs the index, ring and middle finger, adducts the middle finger and assists in flexion of MCP joints, and extension of IP joints of the three fingers	MCP joints of the index, ring and middle fingers

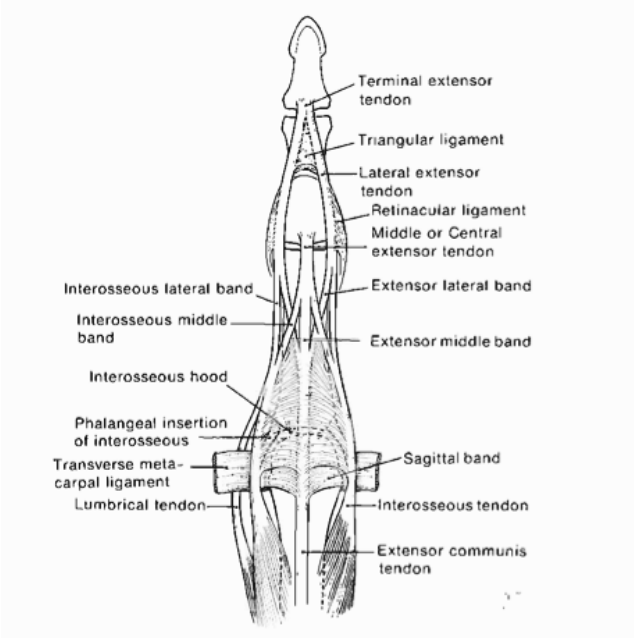


Figure 2.7: Extensor Apparatus [15].

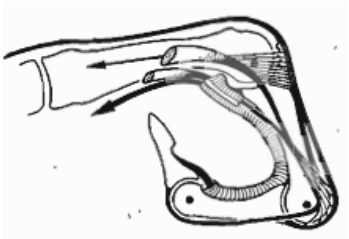


Figure 2.8: Lumbricals acting as flexors [15].

Since the extrinsic muscles are now handling the force, the intrinsic are used to fine-tune the movements of the fingers.

Finally, looking to Table 2.1 and Table 2.2, one can see that the muscular system is heavily coupled, because some muscles actuate several joints, simultaneously, and some joints are actuated by several muscles.

2.2 Dynamics

The human hand is a very complex and, due to its unique characteristics, it is extremely difficult to obtain the dynamic behavior. Because of this hindrance, a numerical model had to be created to serve as a first approximation of the hand dynamics for the training of the neural network model. Due of the extreme adaptability of the neural networks, one can use this numerical model to train a network and optimize its structure. When the full prosthesis is built one can use this initial model has a baseline and continue the training with the new data. To construct the numerical the Articulated-body algorithm (ABA) used.

2.2.1 The Articulated-body Algorithm

Basic equations

The Articulated Body Algorithm (ABA) is currently the fastest serial processing algorithm for a series of rigid-bodies inter-connected by joints. It is compared against other leading algorithms the Divide-and-Conquer Algorithm (DCA) [17], and the Hybrid Direct/Iterative Algorithm (HDIA) [18] in Table 2.3.

Table 2.3: Fastest Algorithms as function of number of bodies (NB) and number of processors (NP)

	$N_B = 10$	$N_B = 100$	$N_B = 1000$
$N_P = 1$	ABA	ABA	ABA
$N_P = 10$	HDIA	HDIA	HDIA
$N_P = 100$	–	HDIA/DCA	HDIA/DCA
$N_P = 1000$	–	–	DCA

Developed by Featherstone [19], this algorithm solves a linear system of n equations, while remaining $O(n)$, using the Newton-Euler vector dynamics relations.

The cornerstone of the ABA is the treatment of a structure link by link, rather than as a whole. Selecting a link from the structure depicted in Fig. 2.9 and defining the reference frames as in Fig. 2.10: the equations for the velocities and acceleration of the i-frame are given by Eq. 2.1 to Eq. 2.5:

$$\mathbf{v}_{i+1}^{O} = \mathbf{v}_{i}^{O} + \omega_i \times \mathbf{r}_{i}^{L} \tag{2.1}$$

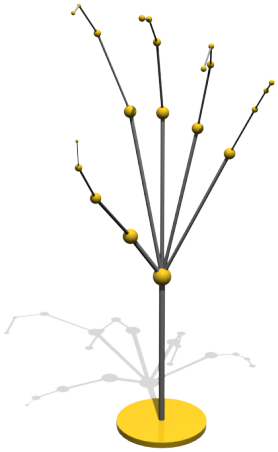


Figure 2.9: Joint-link model of a human hand.

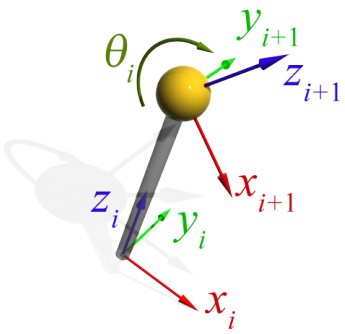


Figure 2.10: Link with joint and reference frames.

$$\omega_{i+1} = \omega_i + \dot{\theta}_i \tag{2.2}$$

$$\mathbf{a}^O_{i+1} = \mathbf{a}^O_i + \alpha_i \times \mathbf{r}^L_i + \omega_i \times \left(\omega_i \times \mathbf{r}^L_i\right) \tag{2.3}$$

$$\alpha_{i+1} = \alpha_i + \ddot{\theta}_i + \omega_i \times \dot{\theta}_i \tag{2.4}$$

$$\mathbf{a}^{CM}_i = \mathbf{a}^O_i + \alpha_i \times \mathbf{r}^{CM}_i + \omega_i \times \left(\omega_i \times \mathbf{r}^{CM}_i\right) \tag{2.5}$$

where ω_{i+1} and ω_i are the angular velocities of the $(i+1)$-frame (at the top of the link, Fig. 2.10) and i-frame (at the base of the link, Fig. 2.10), respectively. The (i)-frame is fixed to the link and the $i+1$-frame moves with the joint. $\mathbf{v}^O$ is the velocity of the origin of its frame and $\mathbf{r}^L_i$ is the position of the $(i+1)$-frame relatively to the i-frame. $\mathbf{a}^O_i$ and $\mathbf{a}^{CM}_i$ are the accelerations of the origin of the frame and of the Center of Mass (CM), respectively. α is the angular acceleration of the frame, $\dot{\theta}$

and $\ddot{\theta}$ are the angular velocity and acceleration of joint, and $\mathbf{r}_i^{CM}$ is the position of the CM relatively to the i-frame.

Based on Fig. 2.10, we can also define the force and moment balances, Eq. 2.6 and Eq. 2.7:

$$\sum \mathbf{F} = \mathbf{F}_{i+1} + \mathbf{F}_i = m_i \mathbf{a}_i^{CM} \tag{2.6}$$

$$\sum \mathbf{M} = \mathbf{M}_{i+1} + \mathbf{M}_i + \mathbf{r}_i^L \times \mathbf{F}_{i+1} = \dot{\mathbf{H}}_i^O \tag{2.7}$$

where: $\mathbf{F}_i$ and $\mathbf{M}_i$ are external forces and moments, respectively, m_i is the mass of the link, and $\dot{\mathbf{H}}_i^O$ is the time derivative of the angular momentum at the origin of the i-frame, given by Eq. 2.8:

$$\dot{\mathbf{H}}_i^O = \mathbf{I}_i^O \alpha_i + \mathbf{r}_i^{CM} \times m_i \mathbf{a}_i^{CM} + \omega_i \times \mathbf{I}_i^O \omega_i \tag{2.8}$$

where $\mathbf{I}_i^O$ is the inertia tensor of the link calculated at the origin of the i-frame.

Spatial Quantities and Articulated-body Inertia

The ABA introduces a new notation the spatial vectors. Spatial vectors assemble on a single vector both linear and angular parts of a quantity, Eq. 2.9:

$$\hat{\mathbf{v}}_i = \begin{bmatrix} \mathbf{v}_i^O \\ \omega_i \end{bmatrix} \quad \hat{\mathbf{a}}_i = \begin{bmatrix} \mathbf{a}_i^O \\ \alpha_i \end{bmatrix} \quad \hat{\mathbf{F}}_i = \begin{bmatrix} \mathbf{F}_i \\ \mathbf{M}_i \end{bmatrix} \tag{2.9}$$

where: $\hat{\mathbf{v}}_i$, $\hat{\mathbf{a}}_i$, $\hat{\mathbf{F}}_i$ are the spatial velocity, spatial acceleration and spatial force at the i-frame, respectively[2].

With this new notation, equations Eq. 2.1 and Eq. 2.7 become equations Eq. 2.10 to Eq. 2.12:

$$\hat{\mathbf{v}}_i = \hat{\mathbf{X}}_{i-1}^i \hat{\mathbf{v}}_{i-1} + \hat{\varphi}_i \dot{\theta}_i \tag{2.10}$$

$$\hat{\mathbf{a}}_i = \hat{\mathbf{X}}_{i-1}^i \hat{\mathbf{a}}_{i-1} + \hat{\varphi}_{i-1} \ddot{\theta}_{i-1} + \hat{\mathbf{C}}_i \tag{2.11}$$

$$\hat{\mathbf{F}}_i - \left(\hat{\mathbf{X}}_i^{i+1}\right)^T \hat{\mathbf{F}}_{i+1} = \hat{\mathbf{I}}_i \hat{\mathbf{a}}_i + \hat{\beta}_i \tag{2.12}$$

where $\hat{\mathbf{X}}_{i-1}^i$ is the spatial transformation matrix that converts spatial quantities from the (i-1)-frame to the i-frame, Eq. 2.13, $\hat{\varphi}_{i-1}$ is a unit vector six-dimensional vector defining the axis of the joint at the $(i-1)$-frame, $\hat{\mathbf{C}}_i$ is the centripetal component of

[2]Note that original definition of spatial vectors given by Featherstone [19], was reversed in this paper for convenience [20].

the spatial acceleration, Eq. 2.14, $\hat{\mathbf{I}}_i$ is the spatial inertia matrix, Eq. 2.15, and $\hat{\beta}_i$ is the spatial force bias, Eq. 2.16:

$$\hat{\mathbf{X}}_{i-1}^{i} = \begin{bmatrix} \mathbf{R}_{i-1}^{i} & -\mathbf{R}_{i-1}^{i}\mathbf{S}_{i-1}^{\mathbf{r}_{i-1}^{L}} \\ \mathbf{0} & \mathbf{R}_{i-1}^{i} \end{bmatrix} \tag{2.13}$$

$$\hat{\mathbf{C}}_i = \begin{bmatrix} R_{i-1}^{i}\left(\omega_{i-1} \times \left(\omega_{i-1} \times r_{i-1}^{L}\right)\right) \\ \omega_i \times \varphi_i \dot{\theta}_i \end{bmatrix} \tag{2.14}$$

$$\hat{\mathbf{I}}_i = \begin{bmatrix} m_i\mathbf{1} & -m_i\mathbf{S}_i^{\mathbf{r}_i^{CM}} \\ m_i\mathbf{S}_i^{\mathbf{r}_i^{CM}} & \mathbf{I}_i^{O} \end{bmatrix} \tag{2.15}$$

$$\hat{\beta}_i = \begin{bmatrix} m_i\omega_i \times \left(\omega_i \times \mathbf{r}_i^{CM}\right) \\ \omega_i \times \mathbf{I}_i^{O}\omega_i \end{bmatrix} \tag{2.16}$$

$\mathbf{S}_{i-1}^{\mathbf{r}_{i-1}^{L}}$ and $\mathbf{S}_i^{\mathbf{r}_i^{CM}}$ are the skew-symmetric matrices that represent the cross products of $\mathbf{r}_{i-1}^{L}$ and $\mathbf{r}_i^{CM}$, respectively. The spatial force bias, $\hat{\beta}_{i-1}$, comes from the fact that the balance of forces, Eq. 2.12, is no longer done at the CM, being done, instead, at the origin of the base frame.

The ABA also introduces the articulated-body inertia, $\hat{\mathbf{I}}_{i-1}^{A}$, and force bias, $\hat{\beta}_{i-1}^{A}$, defined by Eq. 2.17 and Eq. 2.18 [19] and [21]:

$$\hat{\mathbf{I}}_{i-1}^{A} = \hat{\mathbf{I}}_{i-1} + \left(\hat{\mathbf{X}}_{i-1}^{i}\right)^{T} \mathbf{N}_i \hat{\mathbf{X}}_{i-1}^{i} \tag{2.17}$$

$$\hat{\beta}_{i-1}^{A} = \hat{\beta}_{i-1} + \left(\hat{\mathbf{X}}_{i-1}^{1}\right)^{T} \left(\hat{\beta}_i^{A} + \mathbf{N}_i\hat{\mathbf{C}}_i + \mathbf{n}_i\mathbf{M}_i^{-1}\tau_i^{*}\right) \tag{2.18}$$

where τ_i^{*} is the applied torque to the joint adjusted for the dynamic forces (damping and elasticity of the joint), and $\mathbf{N}_i$, $\mathbf{n}_i$ and $\mathbf{M}_i^{-1}$ are auxiliary variables defined by:

$$\mathbf{N}_i = \hat{\mathbf{I}}_i^{A} - \mathbf{n}_i\mathbf{M}_i^{-1}\mathbf{n}_i^{T} \tag{2.19}$$

$$\mathbf{n}_i = \hat{\mathbf{I}}_i^{A}\hat{\varphi}_i \tag{2.20}$$

$$\mathbf{M}_i = \hat{\varphi}_i^{T}\hat{\mathbf{I}}_i^{A}\hat{\varphi}_i \tag{2.21}$$

Because the articulated-body inertia and force bias account for the dynamic behavior of the link in study and the effect of the movement of the upper links (all the links connected to the link top frame), the balance of forces defined by Eq. 2.12 is no

longer explicitly dependent on the upper links quantities, Eq. 2.22:

$$\hat{\mathbf{F}}_i = \hat{\mathbf{I}}_i^A \hat{\mathbf{a}}_i + \hat{\beta}_i^A \tag{2.22}$$

With equation Eq. 2.22 the system of equations formed by the equation of the dynamics changes from full to a triangular system. Hence, obtaining a solution requires $O(n)$ operations rather than $O(n^2)$ [19]. The last quantity to be determined and the goal of the ABA is the joint angular acceleration, which is given by Eq. 2.23:

$$\ddot{\theta}_i = \left(\hat{\varphi}_i^T \hat{\mathbf{I}}_i^A \hat{\varphi}_i\right)^{-1} \left(\left(\tau_i^* - \hat{\varphi}_i^T \hat{\beta}_i^A\right) - \hat{\varphi}_i^T \hat{\mathbf{I}}_i^A \left(\hat{\mathbf{X}}_{i-1}^i \hat{\mathbf{a}}_{i-1} + \hat{\mathbf{C}}_i\right)\right) \tag{2.23}$$

2.2.2 Dynamic Model of the Human Hand

The human hand can be divided into two parts: the skeletal and muscular systems. The latter will be mimicked by the actuators, described in Chapter 4, when assembled into the appropriate positions. However the former needs to be modeled in order to obtain a controller for the full system.

The are two major approaches when modeling a dynamic system: modeling from existing input/output data or from physical principles that govern the system. Obtaining the needed input/output data is very troublesome, because one would need live test subjects to record the joint angles, velocities and acceleration to get the output data. To obtain the input data, one would need to measure the muscles compression by means of EMG, which in turn only provides a limited amount of information [22]. On the other hand, modeling from the physical laws can be done on a purely theoretical basis, needing only some data regarding the average biometric data of a human hand. However, the resulting model is an approximation of the real system.

Using the physical principles to derive the model, one can choose one of two methodologies: the Lagrangean-based or Newton-Euler-based models. Both methods result in the differential equation that governs the model, however the Lagrangean leads to an analytical form, whereas the Newton-Euler-based can lead to an analytical or numerical form (one of the numerical forms is the ABA, presented in subsection 2.2.1). The main reason behind the choice of the numerical form is the impracticability of the analytical form of the dynamic model of the hand, due to the size and complexity of the coefficient matrices of the differential equation.

Using the biometric data provided by [15, 20] and by measures, the bones were

approximated as cylinders with a joint on the top end of the link. The values used for the lengths and radii are shown in Table 2.4[3] and the values for the planar angles between the axis of the metacarpal and the wrist axis at zero adduction are shown in Table 2.5.

Table 2.4: Approximate lengths and radii of the bones of the hand.

Bone Name	Length (m)	Radius (m)
Index Finger		
Metacarpal	0.105	0.0128
Proximal Phalanx	0.050	0.0073
Middle Phalanx	0.024	0.006
Distal Phalanx	0.015	0.006
Middle Finger		
Metacarpal	0.102	0.0134
Proximal Phalanx	0.046	0.008
Middle Phalanx	0.027	0.006
Distal Phalanx	0.019	0.006
Ring Finger		
Metacarpal	0.097	0.012
Proximal Phalanx	0.042	0.00715
Middle Phalanx	0.025	0.0054
Distal Phalanx	0.016	0.0054
Little Finger		
Metacarpal	0.091	0.0117
Proximal Phalanx	0.035	0.0062
Middle Phalanx	0.018	0.0054
Distal Phalanx	0.012	0.0054
Thumb		
Scaphoid and Trapezium	0.037	0.016
Metacarpal	0.048	0.011
Proximal Phalanx	0.031	0.007
Distal Phalanx	0.023	0.003

The mass of the bones was approximated using the average human bone density: 1600kg/m^3.

[3]For simplicity and because it does not, greatly, effect the final results, the length of the carpal bones were included in the metacarpus (with exception of the Scaphoid and Trapezium bones)

Table 2.5: Planar angles between the axis of the metacarpal and the wrist axis at zero adduction.

Bone Name	Angle (deg)
Index Metacarpal	-10
Middle Metacarpal	0
Ring Metacarpal	13
Little Metacarpal	27

Applied Forces on Joints

Another advantage of using the Newton-Euler approach provided by the ABA is that one can determine the torques (on the joints) generated by applied external forces. By knowing the relation between the external forces and the torque joints, one can calculate the part of the force that affects the hand and the fingers and the force that is transmitted to the rest of the arm (which will be supported by the elbow and shoulder joints).

The first step is to rotate the applied force from the inertial frame (the basis of the hand) to the frame of the link containing the application point. Next the force is moved to the basis (joint) of the corresponding link using Eq. 2.24:

$$\hat{\mathbf{F}}_i^{AF} = \left(\hat{\mathbf{X}}_i^P\right)^T \hat{\mathbf{F}}_P^{AF} \tag{2.24}$$

where $\hat{\mathbf{F}}_P^{AF}$ and $\hat{\mathbf{F}}_i^{AF}$ are the applied spatial force at the application point and the equivalent spatial force at the joint of the link, respectively, and $\hat{\mathbf{X}}_i^P$ is the spatial transformation from basis of the link to the application point

With the applied force moved to the joint, the torque can be calculated using Eq. 2.25:

$$\tau_i^{AF} = \hat{\varphi}_i^T \hat{\mathbf{F}}_i^{AF} \tag{2.25}$$

where τ_i^{AF} is the torque generated on the i joint by the applied force.

With the torque calculated the equivalent applied force is moved once again to the previous link, Eq. 2.26:

$$\hat{\mathbf{F}}_{i-1}^{AF} = \left(\hat{\mathbf{X}}_{i-1}^i\right)^T \left(\hat{\mathbf{F}}_i^{AF} - \hat{\varphi}_i \tau_i^{AF}\right) \tag{2.26}$$

The process is then repeated using equations 2.25 and 2.26 until the base link is

reached. The remaining force is the force transmitted to rest of the body.

Comparison of the ABA model with an Analytical model of the human hand

The analytical model of the human hand was obtained using the Lagrangean method. The resulting system of differential equations is extremely large and its solution is computationally expensive to obtain. Also, mainly due to the high level of couplings between the degrees of freedom and the non-linearity of the system, the process of obtaining the solution is numerically unstable, leading to completely inacurate solutions after some sample times.

The data shown in Fig. 2.11(a) was obtained using a sample time of 0.00005s and with no input forces (free fall). Even using a very small sample time did not prevent the numerical instability and the solution quickly tended to the saturation values after 0.35s of simulation. Nevertheless, by comparing Fig. 2.11(a) with (b) and by analyzing Fig. 2.12, one can see that both simulation have similar values. One can also see, in Fig. 2.12, the instability of the analytical model starting to develop (approximately at 0.25s of simulation).

Please note that, by reducing the sample time the analytical model becomes more stable. However, smaller sample times lead to longer simulations and even with a sample time as low as 0.5ms, each iteration took, approximately, 2s[4] of computation time(for a 3s simulation it would, approximately, take 33 hours to complete).

[4]The simulation was done in Wolfram Mathematica 7 using a 3.07GHz Corei7 CPU with 6GiB of RAM

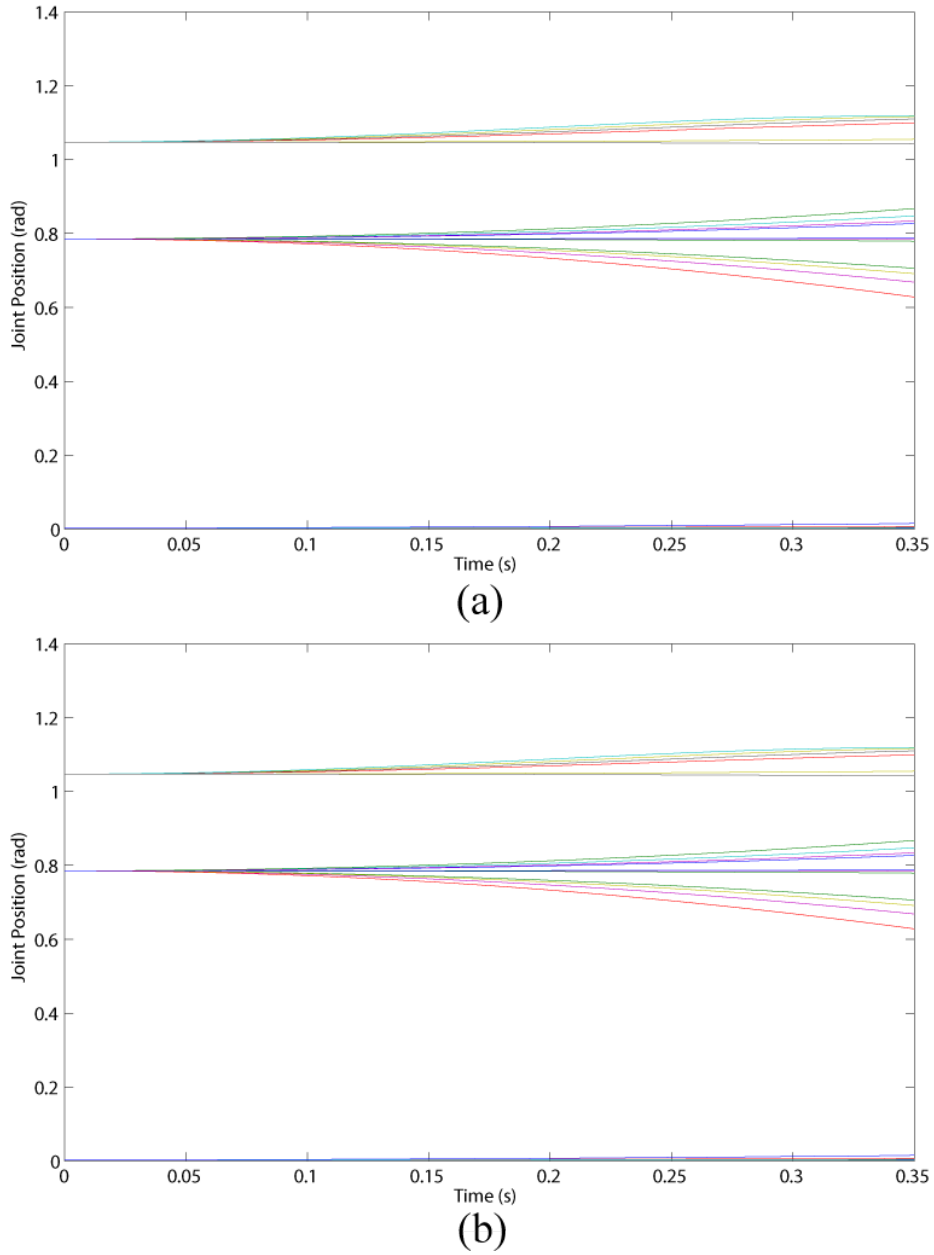


Figure 2.11: Results of the simulation of the human hand in free fall using the analytical model (a) and the ABA model (b).

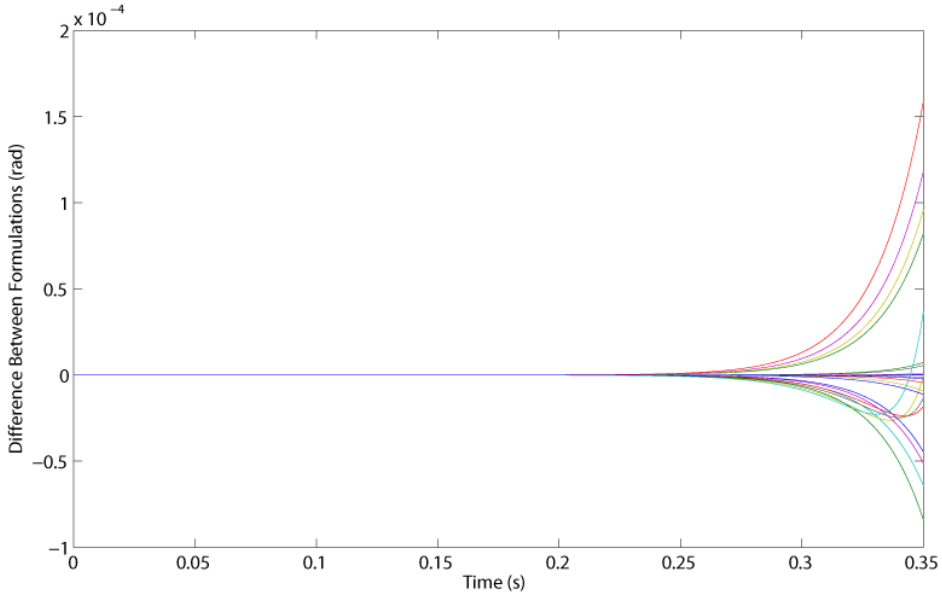


Figure 2.12: Difference between the simulations using the analytical model and the ABA model.

Chapter 3

Neural Network Control Strategy

3.1 Historical Background of Neural Networks

Artificial Neural Networks (ANN) were first theorized by Pitts and McCulloch in 1943,[23]. Based on their knowledge of the operation of organic brains, Pitts and McCulloch established several network configurations for logical neurons. In the years that followed, ANN were studied in great detail by the mathematical and computational analysis community. In turn, this led to major breakthroughs such as the development of the first learning rule by Hebb in 1949, the first *perceptron* by Rosenblatt in 1958, and the Adaptive Linear Element (ADELINE) by Widrow and Hoff in 1960 [24]. Nevertheless, in the following decades, there was a drastic decrease in interest in ANN since additional developments in the area would have required computational power not yet available at the time. This situation lasted until the early 1980s, when digital microprocessors began to see widespread use. In light of these technological achievements, ANN research regained some of the momentum it once had with the development of the associative memory network, by Hopfield, and with the Self-Organizing map, by Kohonen, both developed in 1982 [24, 25]. Nowadays there are over 20 different types of ANN used in a vast range of applications ranging from non-linear control to data mining.

3.2 Static Neural Networks

Static neural networks are the most generic neural networks. A static neural network is able to map a set of input data to its corresponding desired output data by

performing a nonlinear interpolation of the input-output space. There are several configurations for a static neural network, being the most common the Multi-layer perceptrons, Radial basis and Neuro-fuzzy neural networks [24, 23, 26].

3.2.1 The *Preceptron*

The basic unit of ANN is the artificial neuron, which in its simpler form is referred to as a *perceptron*. The *perceptron* (see Fig. 3.1) is a functional with two components: a weighted summation of the inputs (Eq. 3.1), and an activation function (Eq. 3.2),[24, 27]:

$$n = W_j x_j + \beta \tag{3.1}$$

$$\hat{y} = \sigma\left(n\right) \tag{3.2}$$

where x_j are the inputs of the *perceptron*, β is a bias and W_j are the weights of each input. σ and $\hat{y}$ are the activation function and the result of the functional, respectively.

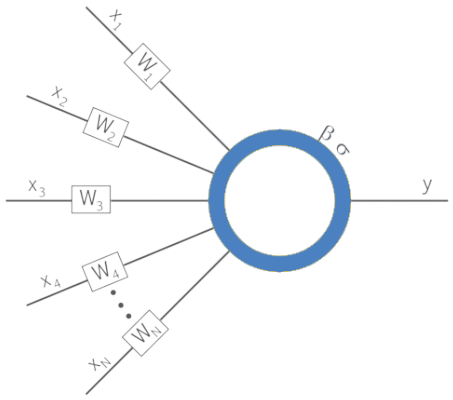


Figure 3.1: *Perceptron* - simplest form of the artificial neuron.

The activation function can be any monotonic function, but the following are the most commonly used: origin crossing linear functions, hyperbolic tangent, logistic functions or the sign function. Usually, linear functions are used in the input and output layers of a network; the sign function is used for biological inspired applications or in digital networks; the hyperbolic tangent is preferentially used in system identification and control; and the logistic function in data mining [24, 27].

3.2.2 Multi-layer *Perceptron*

The Multi-layer *perceptron* (MLP) is one of the configurations proposed by Pitts and McCulloch and is the most versatile and simple network structure [23]. As the name implies the MLP is comprised of a series of sequentially connected layers of *perceptrons*. A layer is defined as a group of *perceptrons* sharing a common set of inputs and, in the case of the MLP, there cannot be intra-layer connections between *perceptrons* (Fig. 3.2) [24].

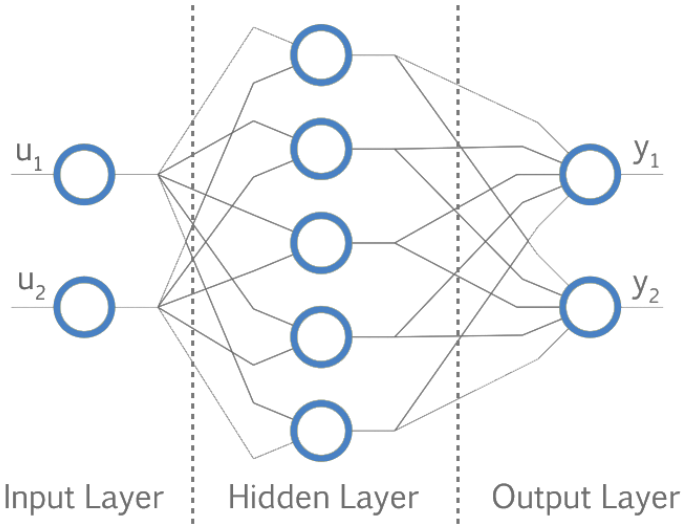


Figure 3.2: Multi-layer *Perceptron* showing the three types of layers

An MLP is divided into three functional regions, Fig. 3.2: the input layer, the hidden layers and the output layer. The input layer is the first layer of an MLP and is responsible for any preprocessing the input data might require. There is no adaptive learning in this layer, because the function of each *perceptron* is pre-established and is fixed. In parallel with natural neural networks, the input layer acts as the sensorial connections of the brain. In more simple cases this layer is not implemented and the inputs are simply distributed into the next layer.

The hidden layers are where the bulk processing of the network takes place. An MLP can have several hidden layers, however the required number of layers or the number of *perceptrons* per layer is dependent on the application. These parameters are very important, because they will define the overall performance of the network and must be selected accordingly to the complexity of the problem. There are no established methods to determine the parameters for a certain application of an MLP

and usually they must be set by a trial-and-error process or by discrete optimization capable methods (e.g. genetic algorithms).

Usually, for an MLP with a single hidden layer, if the number of *perceptrons* is too small, the network will not converge or will converge to a sub-optimal configuration. As one increases the number of *perceptrons*, the network will systematically converge to better configurations, until a point where the network becomes too complex and starts to over-fit the training data, losing quality. Because of this behavior, the Kolmogorov Theorem states that there is always an optimal set of parameters for which a neural network perfectly interpolates any given application [24]. In most cases a single hidden layer can be sufficient, but, in some cases, more layers can bring added stability to the network or more precision to the results. However, given the nonlinear nature of the MLP, the effect of extra layers is unpredictable and is highly dependent on the application. Typically MLPs with more than one hidden layer are used in feedback control, because, sometimes, the extra layers bring more stability to the closed-loop. On top of the number of *perceptrons* and layers, one can also specify different activation functions to each *perceptron* in the hidden layers, bringing some benefits in specific applications [26].

The last region of an MLP is the output layer. This layer is responsible for converting the outputs of the last hidden layer into the output space. In most cases, the *perceptrons* of the output layer only perform a weighted summation of the layer inputs. In this case the activation function is a linear function with a unitary slope, but any activation function can be used. The only restriction that is imposed on this layer is that the number of *perceptrons* must be equal to the number of outputs of the network.

The equations that describe the MLP are similar to Eq. 3.1 and Eq. 3.2:

$$\begin{aligned} x_k^{l+1} &= \sigma^{l+1}\left(W_{kn}^{l} x_n^{l} + \beta_k^{l}\right) \\ \hat{y}_i = x_i^{L} &= \sigma^{L}\left(W_{ij}^{L-1} x_j^{L-1} + \beta_i^{L-1}\right) \end{aligned} \tag{3.3}$$

where l denotes the layer and L is the output layer. In the case of the MLP the activation function σ^l is a vector function that maps a component of the input vector to the corresponding component of the output vector – $\sigma^l = \left[\sigma_i^l(n_i^l)\right]$.

3.2.3 The Back-propagation Algorithm

The main advantage of neural networks is their adaptive nature. The process by which an MLP learns is called the back-propagation algorithm. The back-propagation algorithm (BPA) systematically applies a gradient-based optimization algorithm to each layer of the MLP. The algorithm is divided in two parts: in the first the gradient of the cost function is determined and, in the second part, the weights are updated using an optimization algorithm that minimizes the cost function [24, 28].

There are two methods to calculate the BPA: epoch by epoch (where an epoch is defined as a set of input/output data points) or point by point. In the epoch by epoch BPA the weights and biases are updated after each epoch, whereas in the point by point BPA the update is done after each point. Both approaches yield the same results but the latter, while having a smoother convergence, is computationally more expensive.

The most common cost function used in the optimization of the network parameters (weights and biasses) is the mean square error between the desired output and the output of the network, Eq. 3.4:

$$C = \frac{1}{2N_P} \sum_{p=1}^{N_P} \sum_{i=1}^{N_O} (y_{ip} - \hat{y}_{ip})^2 \tag{3.4}$$

where C is the cost function value, N_P and N_O are the number of points in the epoch and the number of outputs of the plant, respectively. If a point by point BPA is to be used, N_P is 1. Finally y_{ip} and $\hat{y}_{ip}$ are the desired and network outputs, respectively. The major benefits of using this cost function to train neural networks is that it is always positive and it is a static map, meaning that the Cost function, for a given set of input/output data, is only function of the network parameters.

Network Gradient

The fist part of the BPA is, as stated, the determination of the gradient of the cost function, ∇C. The most common method to find the gradient is by using the chain-

rule, Eq. 3.5[1]:

$$\frac{\partial C}{\partial \mathbf{W}^l} = \sum_{p=1}^{N_P} \frac{\partial C}{\partial \hat{\mathbf{y}}_p} \frac{\partial \hat{\mathbf{y}}_p}{\partial \mathbf{W}^l} \tag{3.5}$$

The derivative of the cost function relatively to the network output, $\frac{\partial C}{\partial \hat{\mathbf{y}}_p}$, can be obtained from Eq. 3.4, resulting in Eq. 3.6:

$$\frac{\partial C}{\partial \hat{\mathbf{y}}_p} = -(\mathbf{y}_p - \hat{\mathbf{y}}_p) \tag{3.6}$$

The challenge lies, then, in finding the derivative of the network output relatively to the network weights, $\frac{\partial \hat{\mathbf{y}}_p}{\partial \mathbf{W}^l}$ (which is a third order tensor). In order to obtain this derivative the chain rule must be used once again, yielding for the last layer of the network, $l = L$, Eq. 3.7:

$$\frac{\partial \hat{\mathbf{y}}_p}{\partial \mathbf{W}^{L-1}} = \frac{\partial \sigma^L(\mathbf{n}^L)}{\partial \mathbf{n}^L} \frac{\partial \mathbf{n}^L}{\partial \mathbf{W}^{L-1}} = \sigma'^L \otimes \mathbf{x}_p^{L-1} \tag{3.7}$$

and for any arbitrary layer l, Eq. 3.8:

$$\frac{\partial \hat{\mathbf{y}}_p}{\partial \mathbf{W}^l} = \frac{\partial \sigma^L(\mathbf{n}^L)}{\partial \mathbf{n}^L} \frac{\partial \mathbf{n}^L}{\partial \mathbf{x}^{L-1}} \frac{\partial \sigma^{L-1}}{\partial \mathbf{n}^{L-1}} \frac{\partial \mathbf{n}^{L-1}}{\partial \mathbf{x}^{L-2}} \ldots \frac{\partial \sigma^{l+1}}{\partial \mathbf{n}^{l+1}} \frac{\partial \mathbf{n}^{l+1}}{\partial \mathbf{W}^l} = \sigma'^L \mathbf{W}^{L-1} \sigma'^{L-1} \mathbf{W}^{L-2} \ldots \sigma'^{l+1} \otimes \mathbf{x}_p^l \tag{3.8}$$

where $\mathbf{n}^l = \mathbf{W}^{l-1}\mathbf{x}^{l-1} + \beta^{l-1}$ and σ'^l is the Jacobian of the activation function.

From the equations 3.7 and 3.8, one can obtain a recursive relation that will reduce the number of calculations. For that, the sensitivity matrices, δ^l (Eq. 3.9), and the layer sensitivity, $\mathbf{s}_p^l$ (Eq. 3.10), will be defined:

$$\begin{cases} \delta^{L-1} = \sigma'^L \\ \delta^{l-1} = \sigma'^l \left(\mathbf{W}^l\right)^T \delta^l \end{cases} \tag{3.9}$$

$$\mathbf{s}_p^l = \delta^l \left(\mathbf{y}_p - \hat{\mathbf{y}}_p\right) \tag{3.10}$$

The reason behind this notation is that by pre-multiplying the Jacobians of the activation functions[2] and transpose of weight matrices (instead of what is done on Eq. 3.8) one can have, in the end, a simple outer product of two vectors instead of a

[1]Note that all quantities in bold are matrices or vectors. Also, all capital letters represent matrices and regular letters vectors

[2]Please note that because of the definition of σ^l, its jacobian matrix is a diagonal matrix, see subsection 3.2.2

product of a vector with a third order tensor (as defined in eq. 3.5), Eq. 3.11:

$$\frac{\partial C}{\partial \mathbf{W}^l} = \sum_{p=0}^{N_P} \frac{\partial C}{\partial \hat{\mathbf{y}}_p} \frac{\partial \hat{\mathbf{y}}_p}{\partial \mathbf{W}^l} = -\sum_{p=0}^{N_P} \mathbf{s}_p^l (\mathbf{x}_p^l)^T \tag{3.11}$$

Equation 3.11 can be further simplified by collecting all the layer sensitivities and layer inputs into two matrices, resulting in Eq. 3.12:

$$\frac{\partial C}{\partial \mathbf{W}^l} = -\mathbf{S}^l (\mathbf{X}^l)^T \tag{3.12}$$

where S^l and X^l are given by Eq. 3.13:

$$\begin{aligned} \mathbf{S}^l &= \begin{bmatrix} \mathbf{s}_0^l & \mathbf{s}_1^l & \cdots & \mathbf{s}_{N_P}^l \end{bmatrix} \\ \mathbf{X}^l &= \begin{bmatrix} \mathbf{x}_0^l & \mathbf{x}_1^l & \cdots & \mathbf{x}_{N_P}^l \end{bmatrix} \end{aligned} \tag{3.13}$$

The derivative of the cost function relatively to the *perceptron* biases, $\frac{\partial C}{\partial \beta^l}$ (Eq. 3.14), is almost identical to $\frac{\partial C}{\partial \mathbf{W}^l}$. The main difference is that $\frac{\partial \mathbf{n}^{l+1}}{\partial \beta^l}$ is the identity matrix:

$$\frac{\partial C}{\partial \beta^l} = \sum_{p=0}^{N_P} \frac{\partial C}{\partial \hat{\mathbf{y}}_p} \frac{\partial \hat{\mathbf{y}}_p}{\partial \beta^l} = -\mathbf{S}^l \mathbf{1}^T \tag{3.14}$$

where $\mathbf{1}$ is a matrix with the same dimensions of $\mathbf{X}^l$ but with every entry equal to one.

Collecting both partial derivatives, the gradient of the cost function can be obtained, Eq. 3.15:

$$\nabla C = \begin{cases} \frac{\partial C}{\partial \mathbf{W}^l} = -\mathbf{S}^l \left(\mathbf{X}^l\right)^T \\ \frac{\partial C}{\partial \beta^l} = -\mathbf{S}^l \mathbf{1}^T \end{cases} \tag{3.15}$$

Optimization Algorithms

With the gradient of the cost function determined, one can proceed to the optimization of the parameters. The most common optimization methods for neural network training are the steepest descent, the conjugate gradient and the quasi-Newton methods. Each method has its own strengths and weaknesses and they must be selected accordingly with the application in mind [24, 28].

The steepest descent is the simplest gradient-based optimization method. This method is a first-order optimization method (it makes a local first order approximation

of the cost function) and the optimization direction is proportional to the negative of the gradient, Eq. 3.16:

$$\begin{aligned} \mathbf{d}_k &= -\nabla C_k \\ \mathbf{P}_{k+1} &= \mathbf{P}_k + \alpha_k \mathbf{d}_k \end{aligned} \tag{3.16}$$

where $\mathbf{P_k}$ is a vector containing the parameters of the neural network (both $\mathbf{W}$ and β), α_k is the proportionality constant, commonly called step size, and ∇C is the gradient of the cost function in vector form. As it can be seen from Eq. 3.16, the greatest advantage of the steepest descent is the simplicity and ease of implementation. However, this method, has also the slowest convergence and it is easily trapped in local minima. The rate of convergence can always be adjusted by increasing or decreasing the α constant. It is even possible to adapt the α each step to get the best results using accurate line search algorithms such as the golden section, bisection or false position methods [29].

The conjugate gradient is an evolution of the steepest descent and, while still being a first-order optimization method, it generally has a better convergence rate. The improvement over the steepest descent comes for adding to the current descent the previous descent direction corrected by a constant, Eq.3.17:

$$\begin{aligned} \mathbf{d}_k &= -\nabla C_k + \max(0, \beta_k)\mathbf{d}_{k-1} \\ \mathbf{P}_{k+1} &= \mathbf{P}_k + \alpha_k \mathbf{d}_k \end{aligned} \tag{3.17}$$

where the β_k is the correction constant and can be obtained using one of the methods presented in Table 3.1:

Table 3.1: Methods to determine the correction constant for the Conjugate Gradient method

Method	β_k
Fletcher-Reeves	$\frac{\nabla {C_k}^T \nabla C_k}{\nabla {C_{k-1}}^T \nabla C_{k-1}}$
Polak-Ribière	$\frac{\nabla {C_k}^T (\nabla C_k - \nabla C_{k-1})}{\nabla {C_{k-1}}^T \nabla C_{k-1}}$
Hestenes-Stiefel	$\frac{\nabla {C_k}^T (\nabla C_k - \nabla C_{k-1})}{\nabla {C_{k-1}}^T (\nabla C_k - \nabla C_{k-1})}$

Being the most common method the Polak-Ribière.

The greatest advantage of this algorithm is the improved convergence rate and

tolerance to ill-conditioned functions when compared with the steepest descent. It, also, needs less available memory to perform the calculations than the quasi-Newton methods. This method is usually used in problems large enough to make quasi-Newton methods unusable.

The quasi-Netwon methods are the methods with the best convergence rate and best tolerance to ill-conditioned cost functions. The term "quasi-Newton" comes from the use of an approximation of the Hessian matrix rather than the actual Hessian matrix (as opposed to the Newton method). Although available analytically, the process of determining the Hessian matrix of a neural network has an extremely high cost in both processor load and memory consumption [30]. So, an approximation of the Hessian matrix must be used in order to have an usable method. Also, depending on the algorithm used, the approximation of the Hessian matrix can be made to be always a positive-definite matrix (even when the analytical Hessian is not), meaning that it is always possible to find a descent direction [29, 31, 32, 33]. Regardless of how the Hessian matrix, $\mathbf{H}_k$, is obtained, almost every quasi-Newton and Newton methods must follow the update equation in Eq 3.18

$$\begin{aligned} \mathbf{d}_k &= -\mathbf{H}_k^{-1}\nabla C_k \\ \mathbf{P}_{k+1} &= \mathbf{P}_k + \alpha_k \mathbf{d}_k \end{aligned} \tag{3.18}$$

Again, the step size, α_k, is obtained using a accurate line search[3].

The most well known quasi-Newton methods are the Broyden-Fletcher-Goldfarb-Shanno (BFGS), Eq. 3.19, and the Davidon-Fletcher-Powell (DFP), Eq. 3.20. The main difference between the two is that the BFGS approximates the Hessian matrix while the DFP approximates the inverse Hessian matrix [29, 33, 32].

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{\Delta\nabla C_k \Delta\nabla C_k^T}{\Delta\mathbf{P}_k^T \Delta\nabla C_k} - \frac{\mathbf{H}_k \Delta\mathbf{P}_k \Delta\mathbf{P}_k^T \mathbf{H}_k}{\Delta\nabla C_k^T \mathbf{H}_k \Delta\nabla C_k} \tag{3.19}$$

$$\mathbf{H}_{k+1}^{-1} = \mathbf{H}_k^{-1} + \frac{\Delta\mathbf{P}_k \Delta\mathbf{P}_k^T}{\Delta\mathbf{P}_k^T \Delta\nabla C_k} - \frac{\mathbf{H}_k^{-1} \Delta\nabla C_k \Delta\nabla C_k^T \mathbf{H}_k^{-1}}{\Delta\nabla C_k^T \mathbf{H}_k^{-1} \Delta\nabla C_k} \tag{3.20}$$

where $\Delta\nabla C_k = \nabla C_k - \nabla C_{k-1}$, $\Delta\mathbf{P}_k = \mathbf{P}_k - \mathbf{P}_{k-1}$ and $\mathbf{H}_0$ and $\mathbf{H}_0^{-1}$ are, usually, the Identity matrix. In order to reduce unnecessary calculation, the BFGS and DFP

[3]Please note that for $\mathbf{H}_0 = \mathbf{I}$, the quasi-newton methods become the steepest descent method

Hessian updates may only be done if the following condition is true:

$$\Delta \mathbf{P}_k^T \Delta \nabla C_k > \sqrt{\text{eps}} \|\Delta \mathbf{P}_k\| \|\Delta \nabla C_k\|$$

where "eps" is the machine floating point precision.

Both methods (the BFGS and DFP) generally yield positive-definite matrices, but if the cost function is too ill-conditioned, due to numerical errors, the resulting Hessian may have zero or even negative Eigenvalues. To solve this problem one can update the Cholesky factors of the Hessian directly, Eq. 3.21:

$$\begin{aligned} \mathbf{v}_k &= \sqrt{\frac{\Delta \nabla C_k^T \Delta \mathbf{P}_k}{\Delta \mathbf{P}_k^T \mathbf{L}_k \mathbf{L}_k^T \Delta \mathbf{P}_k}} \\ \mathbf{J}_k &= \mathbf{L}_k + \frac{(\Delta \nabla C_k - \mathbf{L}_k \mathbf{v}_k)\mathbf{v}_k^T}{\mathbf{v}_k^T \mathbf{v}_k} \\ \mathbf{J}_k &= \mathbf{Q} \left(\mathbf{L}_{k+1}\right)^T \end{aligned} \tag{3.21}$$

where $\mathbf{L}$ is the Cholesky factors and $(\mathbf{L}_{k+1})^T$ is obtained by performing the QR decomposition of $\mathbf{J}_k$. Once the Cholesky factors are updated, one can use Eq. 3.22 to get the full Hessian matrix.

$$\mathbf{H}_k = \mathbf{L}_k^T \mathbf{L}_k \tag{3.22}$$

Please note that while Eq. 3.21 and Eq. 3.22 are analytically equivalent to Eq. 3.19, the first two are numerically more accurate.

On top of the BFGS and DFP quasi-Newton methods, there is a special quasi-Newton method called the Levenberg-Marquardt method. The Levenberg-Marquardt method constrains the Newton method to force the descent direction to be inside a trust region, Eq. 3.23, and has, generally, a larger convergence rate than the BFGS or the DFP [33].

$$\begin{aligned} \mathbf{d}_k &= -\left(\mathbf{H}_k + \lambda_k \mathbf{I}\right)^{-1} \nabla C_k \\ \mathbf{P}_{k+1} &= \mathbf{P}_k + \mathbf{d}_k \end{aligned} \tag{3.23}$$

where λ_k is the Lagrangian parameter that constrains the resulting direction to the trust region [24, 29, 33].

There are several method to find the ideal λ, one of them is the Hook step method [33]. This method, determines if the regural quasi-Newton descent direction, Eq. 3.18, is outside the trust region. If it is outside, it starts increasing the λ until the descent direction is inside the trust region. This process is possible because the norm of

the descent direction always decreases monotonically with the increasing λ. Once it is found, the method must determine if the descent direction decreases the cost function: if the cost function decreases then the trust region radius is increased and the optimization continues, if the cost function increases the trust region radius is decreased and the method is restarted until it finds a suitable trust region. If the resulting trust region is too small, then the optimization process reached a local minimum or the function is too ill-conditioned to yield accurate results.

As one can see from equations 3.18 to 3.23, the quasi-Newton methods are much more complex than the steepest descent. Nevertheless, they can have, depending on the cost function, a quadratic convergence rate. They are, also, less likely to be trapped in local minima and are less sensitive to ill-conditioned cost functions. However, the biggest disadvantage of these methods is the amount of memory they require. For a neural network with N parameters, the system needs to store, at least, the parameters, the gradient and the Hessian matrix, requiring enough memory to store $2N + N^2$ values. For larger networks this requirement can render the quasi-Newton methods unusable. There are some quasi-Newton algorithms where this problem is lessened, being the most notable the Limited memory BFGS (L-BFGS) [34], where accuracy and convergence rate are traded for a smaller memory footprint.

Finally, there are other methods to optimize the network parameters, such as the Nonlinear Optimization via External Lead (NOVEL) [35], in which a heuristic is used in conjunction with one of the gradient-based methods to provide a global optimization of the network parameters; Simulated Annealing [36]; or Genetic Algorithms [37], and other Evolutionary Algorithms [38]. One can also use Genetic Algorithms to optimize the network structure itself [39], solving the cumbersome problem of finding the correct network structure for a given application.

Network Jacobian Matrix

One of the benefits of using neural networks is the network Jacobian matrix. The Jacobian matrix contains the partial derivatives of the network output relatively to the network inputs and can be derived from Eq. 3.8:

$$\mathbf{J}_p = \frac{\partial \hat{\mathbf{y}}_p}{\partial \mathbf{x}_p^0} = \frac{\partial \sigma^L\left(\mathbf{n}^L\right)}{\partial \mathbf{n}^L}\frac{\partial \mathbf{n}^L}{\partial \mathbf{x}_p^{L-1}}\cdots\frac{\partial \sigma^0\left(\mathbf{n}^0\right)}{\partial \mathbf{n}^0}\frac{\partial \mathbf{n}^0}{\partial \mathbf{x}_p^0} = \delta^0\mathbf{W}^0 \tag{3.24}$$

From the network jacobian, Eq. 3.24, one can obtain an instantaneous linear approximation of the modeled system, which can be used to determine the local stability of a nonlinear plant. Also, the network jacobian can be used to measure the "nonlinearity" of the plant using the average jacobian and the standard deviation of the matrix entries, in which a smaller the standard deviation means a "more linear" system for a given input. This can be very useful, because if the standard deviation is small, maybe it is more advantageous to model the system using classical linear identification techniques or using a neural network with only the output layer. Finally, the network jacobian is paramount for the training of recurrent neural network NARX, neural state space or neural network controllers.

3.2.4 Neural Network Modeling of Dynamic Systems

There are two ways to model a plant: from the physical laws that govern the plant or from input/output data [40]. The first method implies that a differential equation can be established and represented by a transfer function, state space model or by a numerical method (e.g., the Articulated-body Algorithm), in the second method a set of input/output data is obtained from the real plant and an algorithm to derive the model is used. Such algorithms can be based on classic linear system identification theory methods, e.g.: least square estimation methods (ARX, ARMAX, Box-Jenkins) [40], or black-box models, e.g.: ANN or Fuzzy logic [24, 26].

In linear system identification, all systems are a particular case of the following discrete-time linear time invariant (LTI) transfer function:

$$A\left(z^{-1}\right) y_k = \frac{B\left(z^{-1}\right)}{F\left(z^{-1}\right)} u_k + \frac{C\left(z^{-1}\right)}{D\left(z^{-1}\right)} e_k \tag{3.25}$$

where A,B,C and D are polynomials in the delay operator z^{-1}, and y_k, u_k and e_k are the system output, input and the exogenous noise at the time step k. The most common models used in system identification are the autoregressive model with external input (ARX), Eq. 3.26, and the autoregressive moving average model with external input (ARMAX), Eq. 3.27.

$$A\left(z^{-1}\right) y_k = B\left(z^{-1}\right) u_k + e_k \tag{3.26}$$

$$A\left(z^{-1}\right) y_k = B\left(z^{-1}\right) u_k + C\left(z^{-1}\right) e_k \tag{3.27}$$

Because of their properties, ANN can be adapted to be a non-linear version of the methods described above. If the network input is added to the previous outputs and inputs of the plant, the neural network will work as a non-linear ARX (NARX):

$$\hat{\mathbf{y}}_k = NN\left(\mathbf{u}_k, \mathbf{u}_{k-1}, ..., \mathbf{u}_{k-n_u}, \mathbf{y}_{k-1}, \mathbf{y}_{k-2}, ..., \mathbf{y}_{k-n_y}\right) + \mathbf{e}_k \tag{3.28}$$

where NN represents the ANN (given by Eq. 3.3), and n_u and n_y are the maximum lags for the inputs and outputs, respectively. If the previous exogenous errors are also added as an input to the NARX, a non-linear ARMAX (NARMAX) is obtained:

$$\hat{\mathbf{y}}_k = NN\left(\mathbf{u}_k, \mathbf{u}_{k-1}, ..., \mathbf{u}_{k-n_u}, \mathbf{y}_{k-1}, \mathbf{y}_{k-2}, ..., \mathbf{y}_{k-n_y}, \mathbf{e}_{k-1}, \mathbf{e}_{k-2}, ..., \mathbf{e}_{k-n_e}\right) + \mathbf{e}_k \tag{3.29}$$

where n_e is the maximum lag for the exogenous error.

Having a neural network model of a plant has several advantages: the ANN can model any non-linearities and couplings present in the plant that otherwise would be difficult or impossible to model accurately. Also, even if there is already a model of the system, ANN, after the initial overhead of the training, are faster and computationally lighter than most models. And, finally, one can obtain information about the system that normally would be hard to obtain such has the plant Jacobian matrix (which will be essential for an ANN controller).

When used to model dynamic systems, ANN are used to approximate and discretize the differential equation that governs the process. Because, ANN are discrete-time systems, special care is needed when collecting the input/output data, especially with the sample time. The effect of the sample time on ANN is similar to the effect on regular discrete-time LTI systems: if it is too small, the result will be more precise, but it will also require a larger computational power, and if it is to large the results are less precise with the probability of aliasing, but require less computational power. The optimal sample time can be obtained through the Nyquist sampling theorem that states that to avoid signal aliasing, the sampling frequency must be at least twice the plant bandwidth. The block diagram of a standard NARX training scheme can be seen in Fig. 3.3.

Where T_0 is the sample time; "ZOH" is a zero-order hold and "TDL" stands for

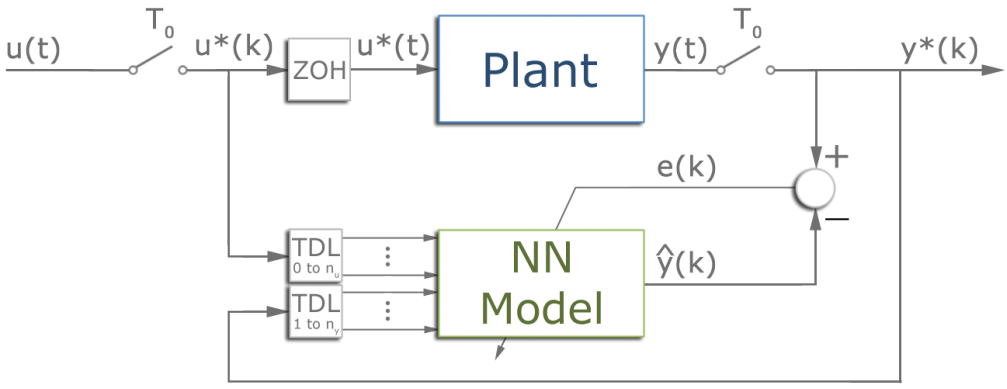


Figure 3.3: Training block diagram of a Static NARX Neural Network.

tapped delay; $u(t)$, $u^*(k)$ and $u^*(t)$ are the inputs, discretized inputs and holded inputs, respectively; $y(t)$ and $y^*(k)$ are the plant output and discretized output, respectively; $\hat{y}(k)$ is the neural network output , and $e(k)$ if the error between the plant and the network outputs.

The training scheme presented in Fig. 3.3 corresponds to a zero-order hold discretization of the plant and, though it is the only method to discretize the plant without prior knowledge of its transfer function, it adds a time delay in the inputs of the plant, which in turn will add extra complication to the controller. Also, from Fig. 3.3 and from Eq. 3.28, one can see that static neural network NARX models always needs the real Plant in parallel. To solve this problem Recurrent Neural Networks and neural state space models were created.

Training of a NARX Neural Network

The training process and data are at par with the network structure in terms of importance. A good training signal has to sweep the desired input space magnitude and frequency-wise. Also, the training signal must stimulate the system so that the desired output space is swept. It was found that the best signal for a SISO system is the chirp signal (Eq. 3.30), which is a sine with an increasing frequency that varies with time. Because the chirp is based on a sine, the signal is always bounded ($[-k\ k]$, where k is a constant) and because it has a variable frequency, the signal stimulates the system at both low and high frequencies, thus capturing the maximum amount

of information.

$$C(t) = \sin(f_0 t + \frac{f_1 - f_0}{\Delta t} t^2) \tag{3.30}$$

where $C(t)$ is the chirp signal, f_0 and f_1 is the start and end frequency, respectively. Finally, t is the time and Δt is the time span for the signal to go from f_0 to f_1.

When moving to MIMO systems, the problem becomes more complex. MIMO systems (and to some extent their particular cases: MISO (multi-input/single output) and SIMO (single input/multi-output) systems) have a larger input and output spaces, thus requiring a multidimensional chirp signal. Also, MIMO system usually have coupled states and inputs which need to be excited in order to obtain enough information about the system. The simpler solution is to feed the system to chirp signals, but the resulting signal is too predictable and fails to excite any existing couplings. The solution is to use a polynomial chirp signal and a frequency path.

A polynomial chirp signal, Eq. 3.31, is similar to the standard chirp signal, Eq. 3.30, however a polynomial interpolation of several frequencies is used instead of a linear interpolation of a start and end frequencies.

$$C(t) = \sin(p_n(t)\, t) = \sin(a_n t^{n+1} + a_{n-1} t^n + a_{n-2} t^{n-1} + ... + a_0 t) \tag{3.31}$$

For a MIMO system, the polynomial $p_n(t)$ is the frequency path, which can be obtained by performing a $n-$order polynomial interpolation over n random bounded points or using $n-1$ cubic splines, Fig. 3.4.

As a final note on the training, it is very important to choose the correct inputs because they have a high impact on the performance of the networks. For example, to obtain a model of a sine function one can use input/output data obtained directly from the sine function or take from the differential equation in Eq. 3.32[4]:

$$\ddot{y} + y = 1 \tag{3.32}$$

where y and $\ddot{y}$ are the output and its second derivative.

Using the sine function one needs a hidden layer and the final cost function value (which is the square of the root mean value of the error) depends on the number of neurons. The cost function values for different number of neurons can be seen in Table 3.2 and the outputs of the networks (with exception of the network with 8

[4]Note that the solution of Eq. 3.32 is y(x) = sin(x)+1

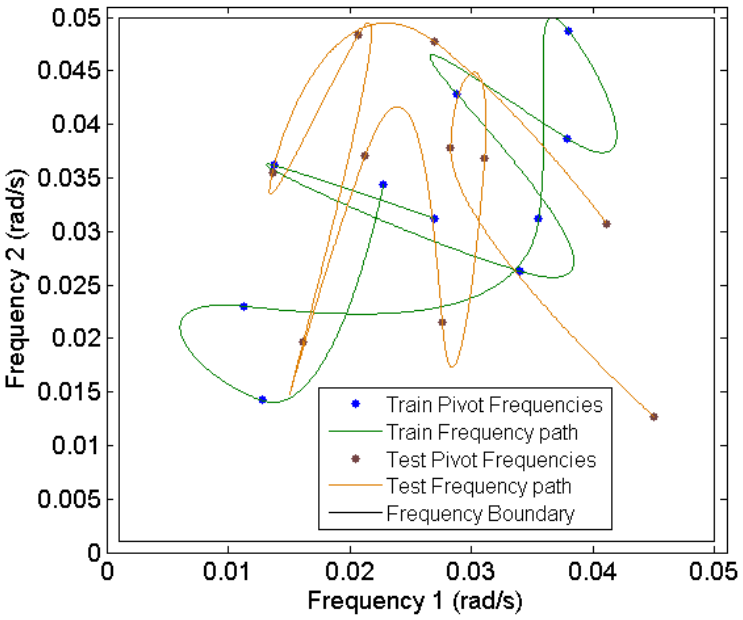


Figure 3.4: A spline frequency path for a 2Dof MIMO system.

neurons in the hidden layer because it is practically coincident with the sine) can be seen in Fig. 3.5.

Table 3.2: Neural network cost function values for the sine.

Number of neurons in hidden layer	Cost function value
2	0.0017
4	4.048×10^{-5}
8	4.626×10^{-11}

However, if one uses the differential equation in Eq. 3.32, with $y(x)$, $\ddot{y}(x)$ (which is $-y(x)$) and 1 as inputs, it is only necessary to use the output layer with one neuron and the resulting cost function value is 1.943×10^{-19} (which could be even lower using an optimization algorithm with a better performance than the Levenberg-Marquardt).

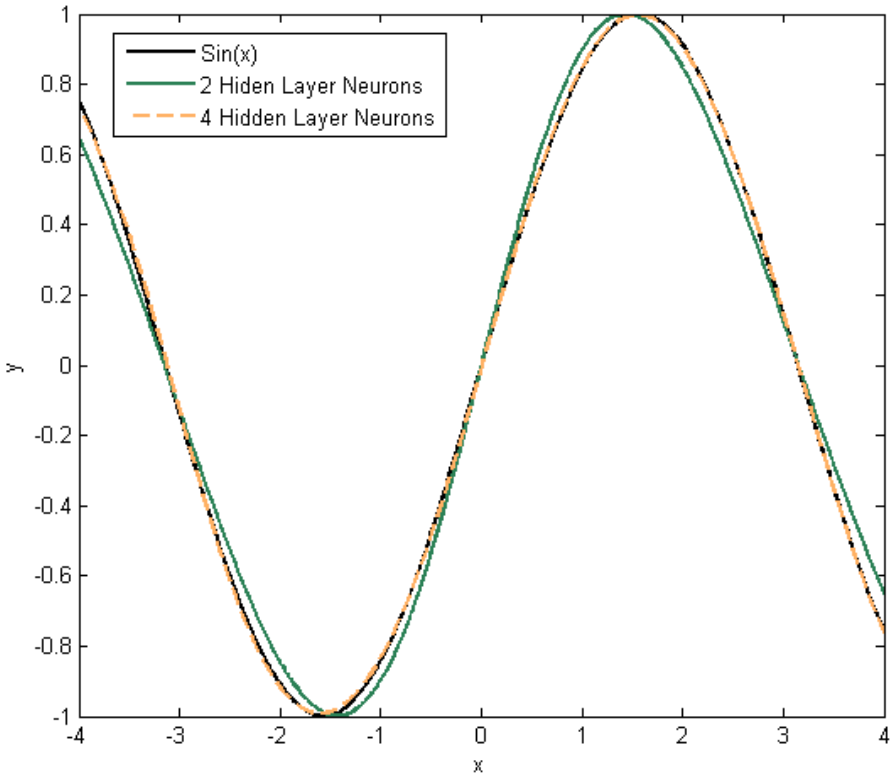


Figure 3.5: Comparison between the sine function and the neural network models.

3.3 NARX Recurrent Neural Networks

NARX Recurrent Neural Networks were created to free neural network NARX models from the system Plant. In term of structure, the only difference between recurrent and static NARX models is that in the former the previous outputs are taken directly from the neural network rather than from the Plant, Eq. 3.33, in opposition with what done in static NARX models, Eq. 3.28.

$$\hat{\mathbf{y}}_k = NN\left(\mathbf{u}_k, \mathbf{u}_{k-1}, ..., \mathbf{u}_{k-n_u}, \hat{\mathbf{y}}_{k-1}, \hat{\mathbf{y}}_{k-2}, ..., \hat{\mathbf{y}}_{k-n_y}\right) + \mathbf{e}_k \tag{3.33}$$

The corresponding block diagram can be seen in Fig. 3.6.

Another advantage of recurrent networks is that they require less training data, because they only need $u^*(k)$ as input and $y^*(k)$ as the desired output. However, this simple modification to the network structure, has an heavy impact in the calculation

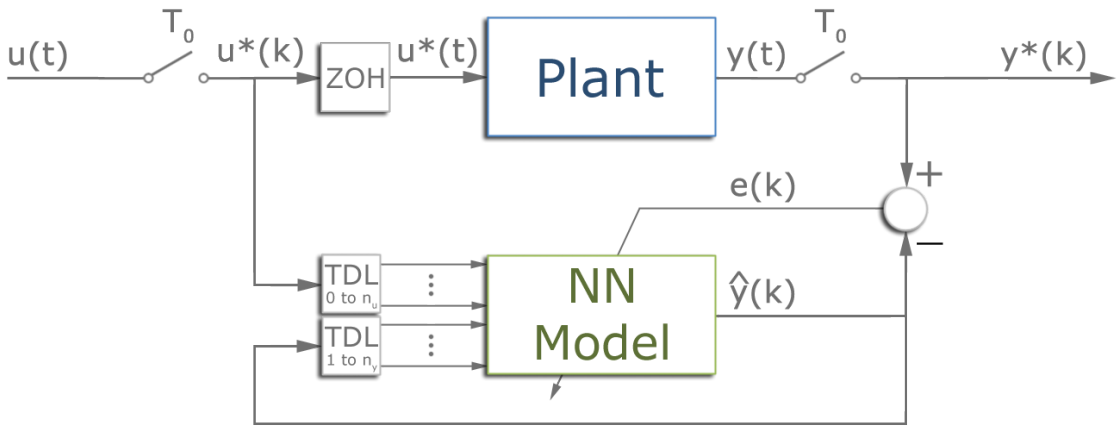


Figure 3.6: Training block diagram of a NARX Recurrent Neural Network.

of the gradient of the cost function.

3.3.1 Gradient of a Recurrent Neural Network

With the introduction of a recurrence in the neural network structure, the cost function is no longer a static map and becomes a dynamic map, meaning that the cost function now depends on the network parameters and on the path in the input/output space. One consequence of having a dynamic map is that a valid set of parameters of a static network can lead to an unstable recurrent network.

One of the first attempts in finding the analytical gradient of a NARX recurrent network was done in [41], with the Dynamic Back-progation Algorithm. The Dynamic Back-propagation was developed for first-order recurrent networks (recurrent networks with only one previous output as input) and neural state space (discussed in more detail in section 3.4), Eq 3.34.

$$\begin{cases} \frac{\partial \hat{\mathbf{y}}_{k+1}}{\partial p_i} = \frac{\partial \hat{\mathbf{y}}_{k+1}}{\partial \hat{\mathbf{y}}_k}\frac{\partial \hat{\mathbf{y}}_k}{\partial p_i} + \left\{\frac{\partial \hat{\mathbf{y}}_{k+1}}{\partial p_i}\right\} \\ \frac{\partial C}{\partial p_i} = \sum_{k=1}^{N_P} \frac{\partial C}{\partial \hat{\mathbf{y}}_k}\frac{\partial \hat{\mathbf{y}}_k}{\partial p_i} \end{cases} \tag{3.34}$$

where p_i is a parameter of the network (a component of the parameter vector $\mathbf{P}_i$), $\left\{\frac{\partial \hat{\mathbf{y}}_{k+1}}{\partial p_i}\right\}$ is the partial derivative of $\hat{\mathbf{y}}_{k+1}$ of its direct dependency on p_i and $\frac{\partial \hat{\mathbf{y}}_{k+1}}{\partial \hat{\mathbf{y}}_k}$ is the partial derivative of the network output relatively to the previous output, which

can be obtained from the network Jacobian matrix (Eq. 3.24). Although made for first-order systems, this algorithm can be expanded to handle second and higher-order systems.

Another approach to the problem is determine a corrected $\frac{\partial C}{\hat{\mathbf{y}}_p}$ so that one can use the same equations (and code) of regular static neural networks. This method also provides a mean to make the two types of neural network comparable.

Using the notation introduced in Eq. 3.34 for the derivative of the direct dependency, $\{\mathbf{.}\}$, Eq. 3.5 can rewritten as follows, Eq 3.35:

$$\frac{\partial C}{\partial \mathbf{W}^l} = \sum_{p=1}^{N_P} \frac{\partial C}{\partial \hat{\mathbf{y}}_p} \left\{ \frac{\partial \hat{\mathbf{y}}_p}{\partial \mathbf{W}^l} \right\} \tag{3.35}$$

The objective is to write the partial derivatives of the cost function of the recurrent network as Eq. 3.36:

$$\frac{\partial C}{\partial \mathbf{W}^l} = \sum_{p=1}^{N_P} \mathbf{E}_p \left\{ \frac{\partial \hat{\mathbf{y}}_p}{\partial \mathbf{W}^l} \right\} \tag{3.36}$$

where $\mathbf{E}_p$ is the corrected error.

To obtain the corrected error one must perform a mathematical manipulation on the expression of the partial derivative of the cost function. For a first order-system, the partial derivatives take the form of Eq.3.37:

$$\begin{aligned}
&\tfrac{\partial C}{\partial \mathbf{W}^l} = \tfrac{\partial C}{\partial \hat{\mathbf{y}}_{N_P}} \tfrac{\partial \hat{\mathbf{y}}_{N_P}}{\partial \mathbf{W}^l} + \tfrac{\partial C}{\partial \hat{\mathbf{y}}_{N_P-1}} \tfrac{\partial \hat{\mathbf{y}}_{N_P-1}}{\partial \mathbf{W}^l} + \cdots + \tfrac{\partial C}{\partial \hat{\mathbf{y}}_1} \tfrac{\partial \hat{\mathbf{y}}_1}{\partial \mathbf{W}^l} = \\
&= \tfrac{\partial C}{\partial \hat{\mathbf{y}}_{N_P}} \left(\left\{ \tfrac{\partial \hat{\mathbf{y}}_{N_P}}{\partial \mathbf{W}^l} \right\} + \tfrac{\partial \hat{\mathbf{y}}_{N_P}}{\partial \hat{\mathbf{y}}_{N_P-1}} \tfrac{\partial \hat{\mathbf{y}}_{N_P-1}}{\partial \mathbf{W}^l} \right) + \tfrac{\partial C}{\partial \hat{\mathbf{y}}_{N_P-1}} \left(\left\{ \tfrac{\partial \hat{\mathbf{y}}_{N_P-1}}{\partial \mathbf{W}^l} \right\} + \tfrac{\partial \hat{\mathbf{y}}_{N_P-1}}{\partial \hat{\mathbf{y}}_{N_P-2}} \tfrac{\partial \hat{\mathbf{y}}_{N_P-2}}{\partial \mathbf{W}^l} \right) + \\
&+ \cdots + \tfrac{\partial C}{\partial \hat{\mathbf{y}}_1} \left(\left\{ \tfrac{\partial \hat{\mathbf{y}}_1}{\partial \mathbf{W}^l} \right\} \right)
\end{aligned} \tag{3.37}$$

Rearranging Eq. 3.37 to explicit the derivative of the direct dependencies results in Eq. 3.38:

$$\begin{aligned}
&\tfrac{\partial C}{\partial \mathbf{W}^l} = \left(\tfrac{\partial C}{\partial \hat{\mathbf{y}}_{N_P}} \right) \left\{ \tfrac{\partial \hat{\mathbf{y}}_{N_P}}{\partial \mathbf{W}^l} \right\} + \left(\tfrac{\partial C}{\partial \hat{\mathbf{y}}_{N_P-1}} + \tfrac{\partial C}{\partial \hat{\mathbf{y}}_{N_P}} \tfrac{\partial \hat{\mathbf{y}}_{N_P}}{\partial \hat{\mathbf{y}}_{N_P-1}} \right) \left\{ \tfrac{\partial \hat{\mathbf{y}}_{N_P-1}}{\partial \mathbf{W}^l} \right\} + \\
&+ \left(\tfrac{\partial C}{\partial \hat{\mathbf{y}}_{N_P-2}} + \tfrac{\partial C}{\partial \hat{\mathbf{y}}_{N_P-1}} \tfrac{\partial \hat{\mathbf{y}}_{N_P-1}}{\partial \hat{\mathbf{y}}_{N_P-2}} + \tfrac{\partial C}{\partial \hat{\mathbf{y}}_{N_P}} \tfrac{\partial \hat{\mathbf{y}}_{N_P}}{\partial \hat{\mathbf{y}}_{N_P-1}} \tfrac{\partial \hat{\mathbf{y}}_{N_P-1}}{\partial \hat{\mathbf{y}}_{N_P-2}} \right) \left\{ \tfrac{\partial \hat{\mathbf{y}}_{N_P-2}}{\partial \mathbf{W}^l} \right\} + \cdots
\end{aligned} \tag{3.38}$$

From Eq. 3.38, one can take the following recurrent rule for the corrected error,

Eq. 3.39:

$$\begin{cases} \mathbf{E}_{N_P} = \frac{\partial C}{\partial \hat{\mathbf{y}}_{N_P}} \\ \mathbf{E}_p = \frac{\partial C}{\partial \hat{\mathbf{y}}_p} + \left(\frac{\partial \hat{\mathbf{y}}_{p+1}}{\partial \hat{\mathbf{y}}_p} \right)^T \mathbf{E}_{p+1} \end{cases} \tag{3.39}$$

It can be shown that for second and higher-order systems, Eq. 3.39 becomes Eq. 3.40:

$$\begin{cases} \mathbf{E}_{N_P} = \frac{\partial C}{\partial \hat{\mathbf{y}}_{N_P}} \\ \mathbf{E}_p = \frac{\partial C}{\partial \hat{\mathbf{y}}_p} + \sum_{k=p+1}^{p+n_y} \left(\frac{\partial \hat{\mathbf{y}}_k}{\partial \hat{\mathbf{y}}_p} \right)^T \mathbf{E}_k \end{cases} \tag{3.40}$$

With the corrected error determined, the gradient of the network is obtained using the same equations of the regular static neural networks, with the sole exception that Eq. 3.41 must be used instead of Eq. 3.10:

$$\mathbf{s}_p^l = \delta^l \mathbf{E}_p \tag{3.41}$$

3.3.2 Stability of Recurrent Neural Networks

Regular static neural networks are stable as long the modeled plant remains stable, however, due to the recurrences, recurrent neural network may not be stable even if the plant is. This problem arrises mainly in the choosing of the initial point.

If the initial point is inside the stability region[5] of the parameter space it will remain inside this region, because the cost function quickly tends to infinity when the optimization path reaches the border. However, this region is a finite region of the parameter domain and if the initial point is outside, the optimization will never converge because all the resulting systems will be unstable. There are two solutions for this problem, the first is decrease the number of points in the train/test data set, because the cost function tends faster to infinity outside the stability border with the increasing number of data points. However a smaller data set also decreases the amount of information to train the network, leading to poorer performances. The second solution is to use choose initial point closer to the origin of the domain, because the stability region is, usually, situated around the origin of the parameter domain. The disadvantage of bringing initial points closer to the origin is that it will take longer to train the network.

[5]The stability region is a region in the parameter domain in which the resulting recurrent neural network is a stable system

3.3.3 Comparison Between NARX Recurrent Networks and NARX Static Networks

NARX recurrent neural networks bring a set of advantages and disadvantages over NARX static neural networks. The main advantages are the need for less input/output training data, recurrent networks only need the input of the system and the desired output, while static networks need the input of system, the desired output and all the needed past inputs and outputs of the plant. Also, recurrent networks are independent of the plant and can work standalone, unlike static network that constantly need the plant, even when fully trained.

The disadvantages are mainly the more complex calculations needed to determine the cost function gradient and most of then cannot be parallelized (unlike static networks, which fully paralellizable) because of the time dependency. Also, the cost functions of recurrent networks are always more complex and ill-conditioned than similar static networks. For example, given the first order system in Eq. 3.42:

$$G(s) = \frac{1}{s+1} \xrightarrow[T_0=1\text{ms}]{\text{ZOH}} G(z) = \frac{0.0009995}{z-0.999} \tag{3.42}$$

the NARX models are $\hat{y}_k = \text{NN}(u_k, y_{k-1})$ and $\hat{y}_k = \text{NN}(u_k, \hat{y}_{k-1})$ for the static network and recurrent network[6], respectively. Since, the system to model is a first-order LTI transfer function, only the output layer is needed and, consequently, both networks only have two parameters. The cost function of both networks can be see in Fig. 3.7, for the static network, and in Fig. 3.8, for the recurrent network:

From both figures, it can be seen that both networks have the same global minimum at $\{p_0 = 0.0009995, p_1 = 0.999\}$, but the cost function of the recurrent network is much more complex and ill-conditioned than the static network and, on top, of having zones of the domain where the cost function tends to infinity (the parameters lead to an unstable system) the global minimum is located in the border of stability in an extremely narrow zone of the domain, which makes the optimization process more difficult and longer.

[6]Both NARX correspond to a ARX11 ($n_u = 1$ and $n_y = 1$)

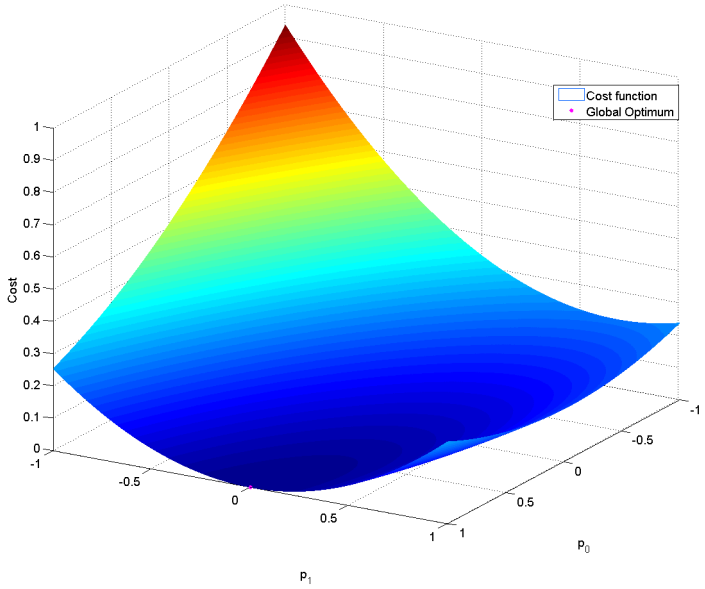


Figure 3.7: Cost function of a NARX11 static neural network model.

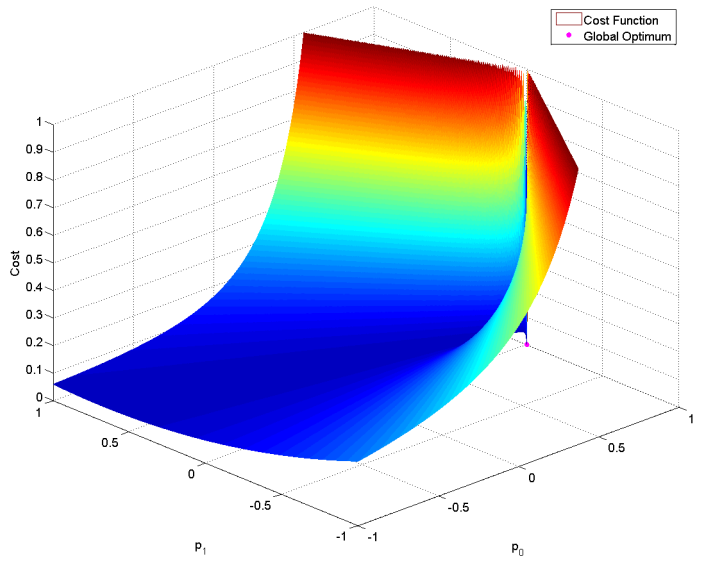


Figure 3.8: Cost function of a NARX11 recurrent neural network model.

In second-order systems the stability problem increases, because the recurrent neural network has two parameters that affect the stability of the system. Considering the following second-order system, Eq. 3.43:

$$G(s) = \frac{1}{s^2 + 0.5s + 1} \xrightarrow[T_0=1\text{ms}]{\text{ZOH}} G(z) = \frac{2.5003 \times 10^{-4} z + 2.49975 \times 10^{-4}}{z^2 - 1.999z + 0.9995} \tag{3.43}$$

the two NARX models are $\hat{y}_k = \text{NN}(u_k, u_{k-1}, u_{k-2}, y_{k-1}, y_{k-2})$ and $\hat{y}_k = \text{NN}(u_k, u_{k-1}, u_{k-2}, \hat{y}_{k-1}, \hat{y}_{k-2})$ for the static and recurrent networks, respectively. Since now, even with only output network, there are five network parameters, one has to select only two to be able to draw the cost function. The two most important parameters are the last two (the ones assigned to the previous outputs) because they affect the stability of the recurrent network. The static network cost function can be seen in Fig. 3.9 and the recurrent network cost function in Fig. 3.10.

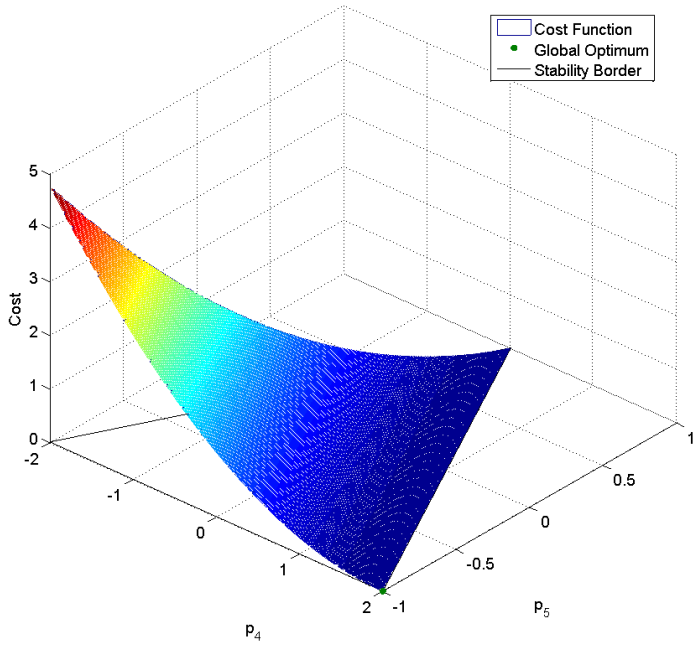


Figure 3.9: Cost function of a NARX22 static neural network model.

As one can see from Fig. 3.9 and Fig. 3.10, the cost functions are very different from each other, while the cost function of the static neural network remains smooth and with a clear descent towards the global optimum, the cost function of the recurrent network presents a highly ill-conditioned behavior remaining, almost, constant

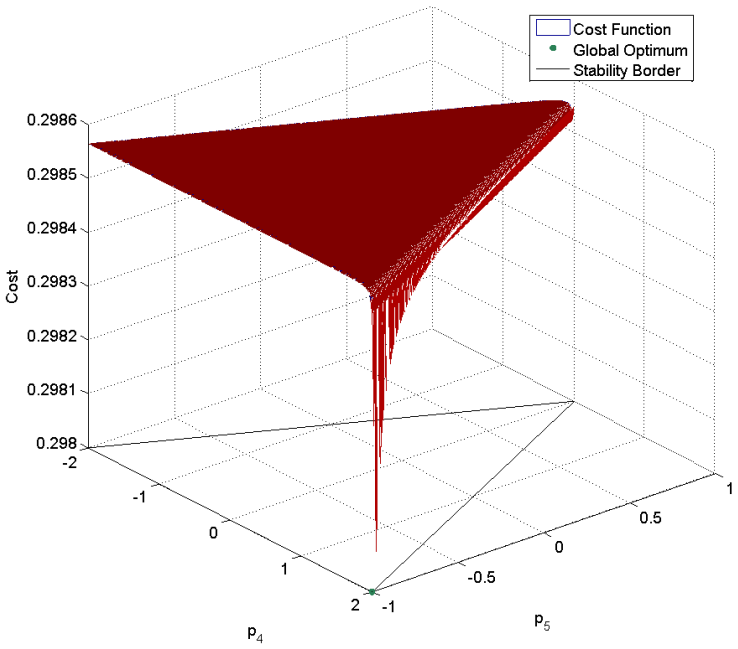


Figure 3.10: Cost function of a NARX22 recurrent neural network model.

in a large portion of the domain and then abruptly decreasing when is near global optimum. Another characteristic of the recurrent cost function is that outside the stability region it rapidly grows to infinity (the rate of growth depends on how many points the training data contains – more points lead to a larger rate), encasing the global minimum in a very narrow region of the domain. The stability region was also represented in the static neural network cost function for comparison, because, due to the network characteristics, it does not suffer from stability problems. With such different cost function, the optimization process is very different between both networks. In static networks the optimization process is smoother and the optimization paths are very similar to each other, Fig. 3.11.

On the other hand, the optimization paths of the training of recurrent networks are much more erratic and take longer to converge, Fig. 3.12

The convergence plots for the static and recurrent networks can be seen in Fig. 3.13 and Fig. 3.14, respectively.

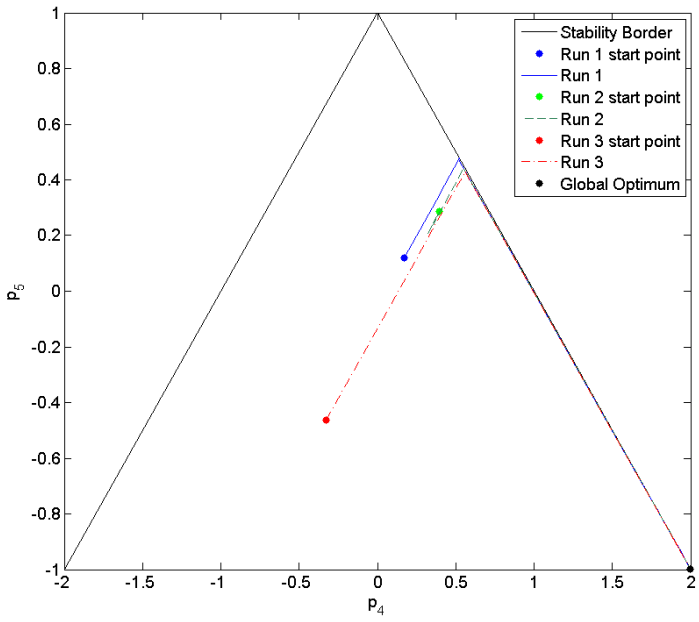


Figure 3.11: Optimization paths for three training runs of the NARX static neural network.

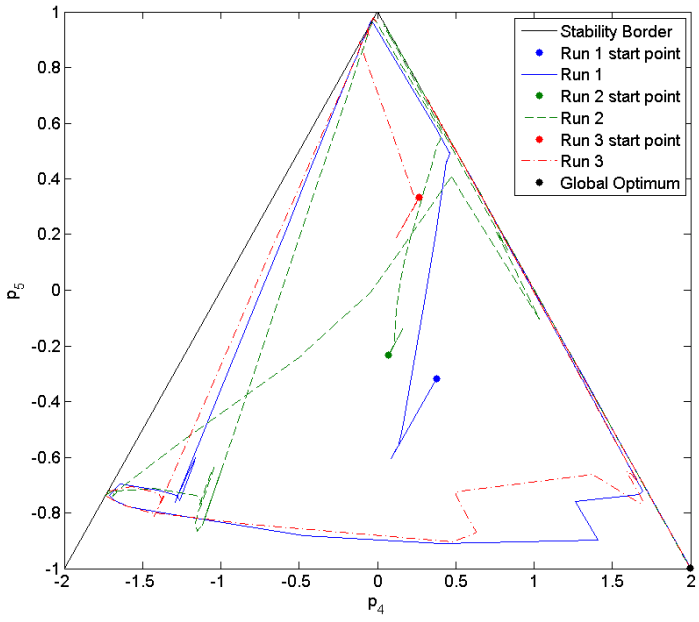


Figure 3.12: Optimization paths for three training runs of the NARX recurrent neural network.

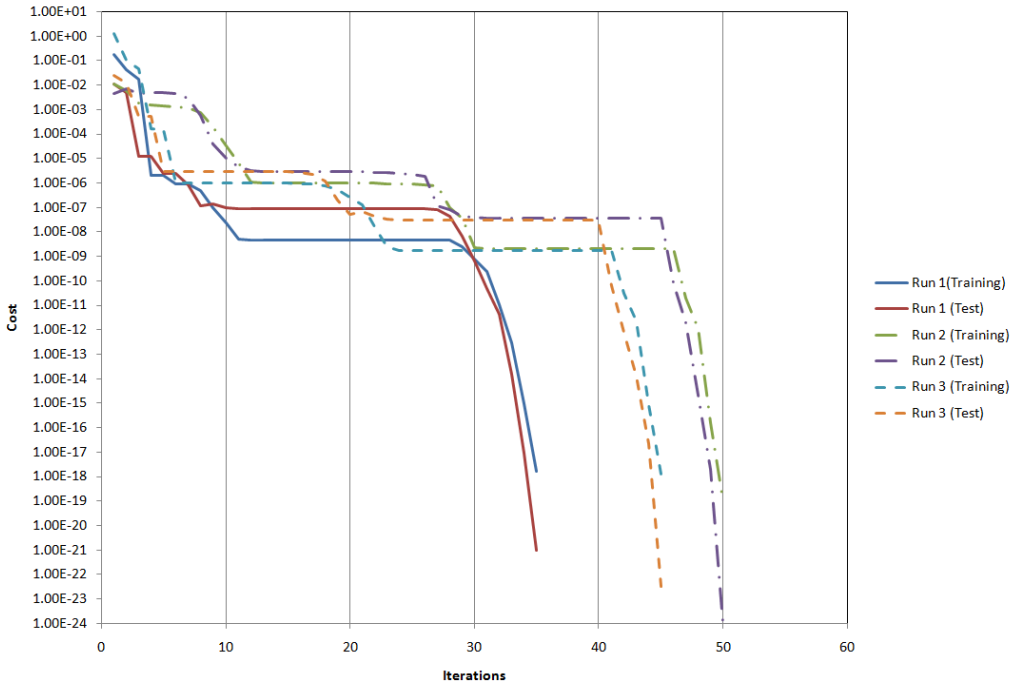


Figure 3.13: Cost function of a NARX22 static neural network model.

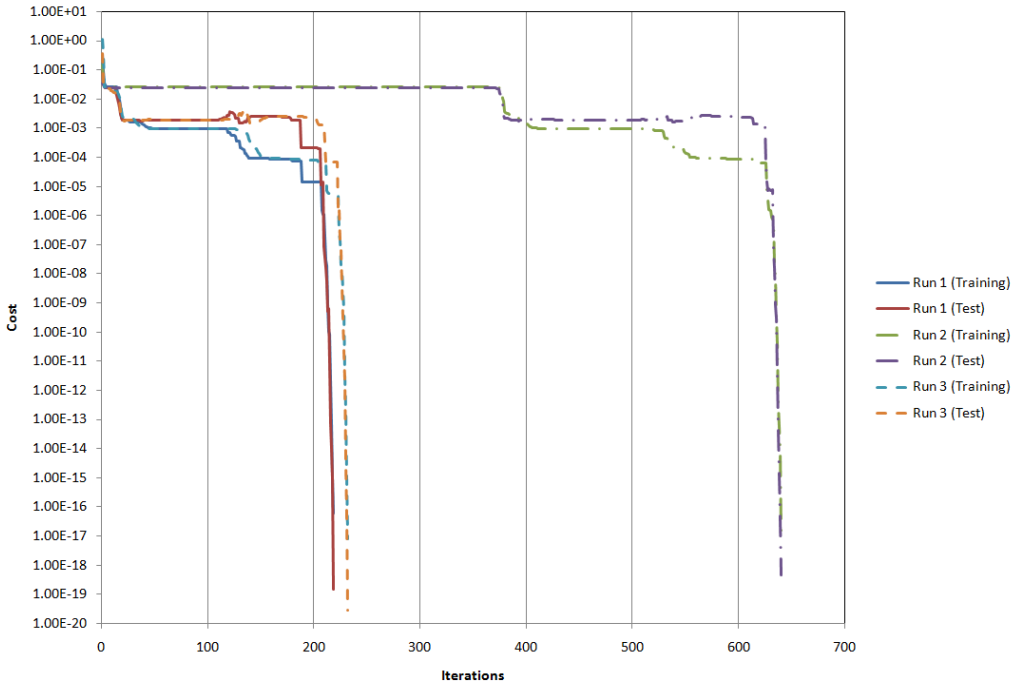


Figure 3.14: Cost function of a NARX22 recurrent neural network model.

When moving to nonlinear multi-input/multi-output (MIMO) systems, the training becomes slower and more difficult. The first system to be modeled will be a double pendulum, Fig. 3.15, which is a system with two inputs (torques on the joints) and outputs (joint positions) and with soft nonlinearities[7], Eq. 3.44.

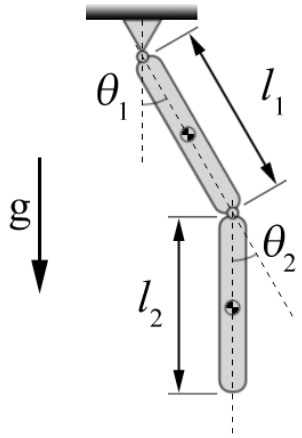


Figure 3.15: Double Pendulum.

$$\begin{bmatrix} I_{zz1}+I_{zz2}+m_1r_1^2+m_2(l_1^2+r_2^2)+m_2l_1l_2\cos(\theta_2) & I_{zz2}+m_2r_2^2+\frac{1}{2}m_2l_1l_2\cos(\theta_2) \\ I_{zz2}+m_2r_2^2+\frac{1}{2}m_2l_1l_2\cos(\theta_2) & I_{zz2}+m_2r_2^2 \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} +$$
$$+ \begin{bmatrix} -\frac{1}{2}m_2l_1l_2\sin(\theta_2)\dot{\theta}_2+c_1 & -\frac{1}{2}m_2l_1l_2\sin(\theta_2)\left(\dot{\theta}_1+\dot{\theta}_2\right) \\ \frac{1}{2}m_2l_1l_2\sin(\theta_2)\dot{\theta}_1 & c_2 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} +$$
$$+ \begin{bmatrix} \left(l_1m_2+\frac{1}{2}l_1m_1\right)g\sin(\theta_1)+\frac{1}{2}m_2gl_2\sin(\theta_1+\theta_2) \\ \frac{1}{2}m_2gl_2\sin(\theta_1+\theta_2) \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{3.44}$$

Where I_{zz} is the inertia moments about the z-axis, m is the mass, l is the length of the link, r is the distance between the joint and the link center of mass, c is the coefficient of friction, g is the acceleration of gravity and $\ddot{\theta}$, $\dot{\theta}$, θ and τ are the joint acceleration, velocity, position and torque, respectively.

Since the system in study is nonlinear the neural network models will need hidden layers and, as stated in subsection 3.2.2, the structure of the optimal network (number

[7]A soft nonlinearities are nonlinear functions with a continuity at least C^n, where n is the order of the system, e.g.: transcendental functions or powers of the output or state. In opposition, hard nonlinearities are functions with a continuity less than the order of the system, e.g.: saturations, dead-zones or coloumb friction; or path dependencies, e.g.: hysteresis or backlash.

of layers and neuron in each layer) can only be found by trial and error or by advanced discrete optimization methods. The following results will try to demonstrate the effects of a change in the network structure in the overall performance of a static neural network, Table 3.3 and Fig. 3.16:

Table 3.3: Results of the static neural network modeling of a Double Pendulum for different network structures (LF and HT are Linear and Hiperbolic Tangent activation functions, respectively, and the numbers represent the number of neurons in the layer: e.g. 5LF-10HT-2LF is a three layer network with 5 linear neurons in the first layer, 10 hyperbolic tangent neurons in the second and 2 linear neurons in the output layer).

Structure	Final Train Cost	Best Test Cost	Iterations	Designation
5LF-10HT-2LF	1.59×10^{-9}	6.87×10^{-8}	10000	Static1
10LF-10HT-2LF	2.17×10^{-9}	5.34×10^{-8}	10000	Static2
5HT-10HT-2LF	9.75×10^{-10}	5.03×10^{-8}	10000	Static3
5LF-10HT-10HT-2LF	1.64×10^{-9}	8.97×10^{-8}	10000	Static4

From both Fig. 3.16 and Table 3.3 it can be seen that extra hidden layers does not bring a better performance to the network and that the first layer is the one that has a larger impact on the overall performance, with the Static3 network having the better performance. Another observation is that, when dealing with nonlinear systems, neural networks can no longer perfectly model the plant and, consequently, the cost function values (and errors) are larger than the ones obtained by linear plants (which are in the order of $1\times^{-19}$). However, the larger errors brought by nonlinear plants are still small enough for the final neural network model to be a good model. Also, from Fig. 3.16, one can see that in spite of a decreasing exponential convergence for the training data the test data convergence tends to stay around 1^{-7}. This usually happens when the information contained in the training signal is insufficient for a further generalization. As a final remark, all the neural networks presented reached the maximum number of iterations allowed (10000) before reaching a minimum (local or global), which means that the training could be continued.

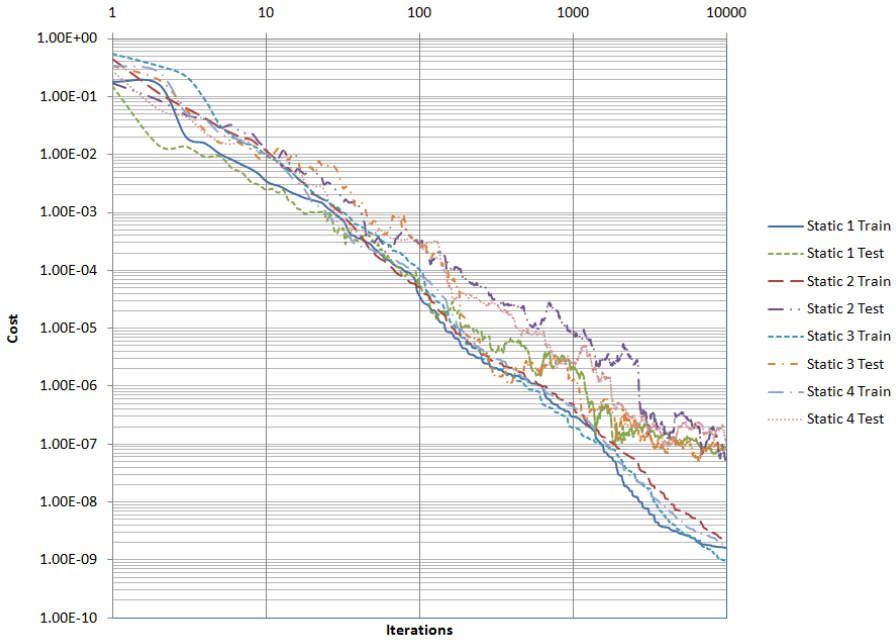


Figure 3.16: Convergence of the static neural networks for the model of the double pendulum.

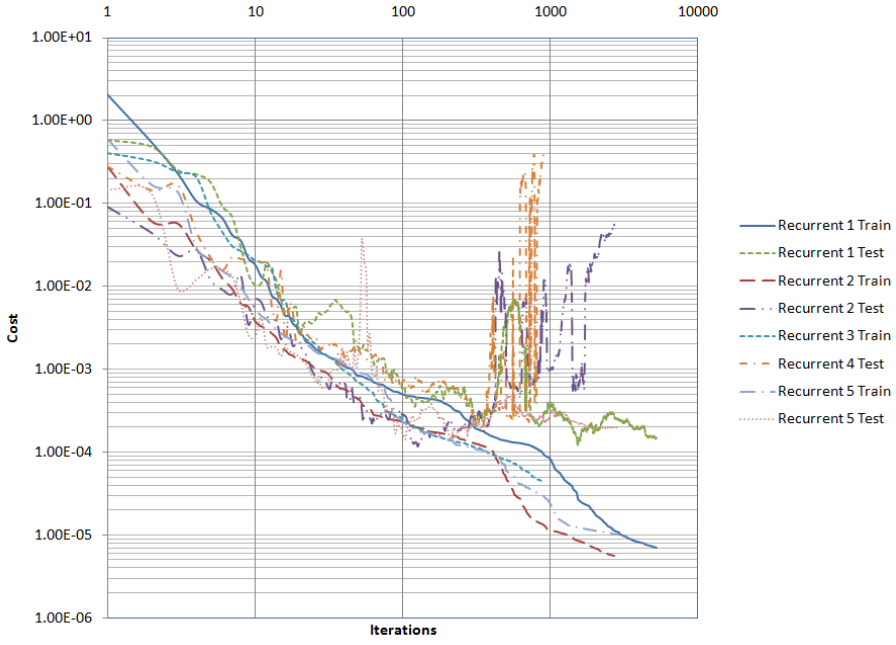


Figure 3.17: Convergence of the recurrent neural networks for the model of the double pendulum.

The results obtained for the recurrent neural network NARX are presented in Table 3.4 and Fig. 3.17.

Table 3.4: Results of the recurrent neural network modeling of a Double Pendulum for different network structures

Structure	Final Train Cost	Best Test Cost	Iterations	Designation
5LF-10HT-2LF	6.93×10^{-6}	1.19×10^{-4}	5228	Recurrent1
10LF-10HT-2LF	5.53×10^{-6}	1.18×10^{-4}	2705	Recurrent2
5HT-10HT-2LF	4.45×10^{-5}	1.98×10^{-4}	911	Recurrent3
5LF-10HT-10HT-2LF	1.01×10^{-5}	1.41×10^{-4}	2900	Recurrent4

As expected, the training of recurrent networks is much more complex, which in turn leads to larger cost function values and erratic convergence rates. The cost function is very ill-conditioned and the optimization process is stopped earlier because the optimizer finds local minima or extremely narrow descent paths (as seen in Fig. 3.8 for first order LTI system or Fig. 3.10 for a second-order LTI system) that are impossible to follow with the current optimization algorithms. This ill-condition comes from the fact that recurrent neural networks are much more sensible to the network parameters and structure: adding a third hidden layer (Recurrent4) or using hyperbolic tangent neurons in the first hidden layer (Recurrent3) worsens the performance of the network, and duplicating the number of neurons in the first hidden layer (Recurrent2) brings only a slight performance improvement, though with a faster convergence rate.

In terms of error between the model and plant outputs, the static neural network NARX, Fig.3.18a, has a better performance (due to smaller cost function values) than the recurrent neural network NARX, Fig.3.18b. Nevertheless both are good models when compared to the linear approximation, Fig.3.18c.

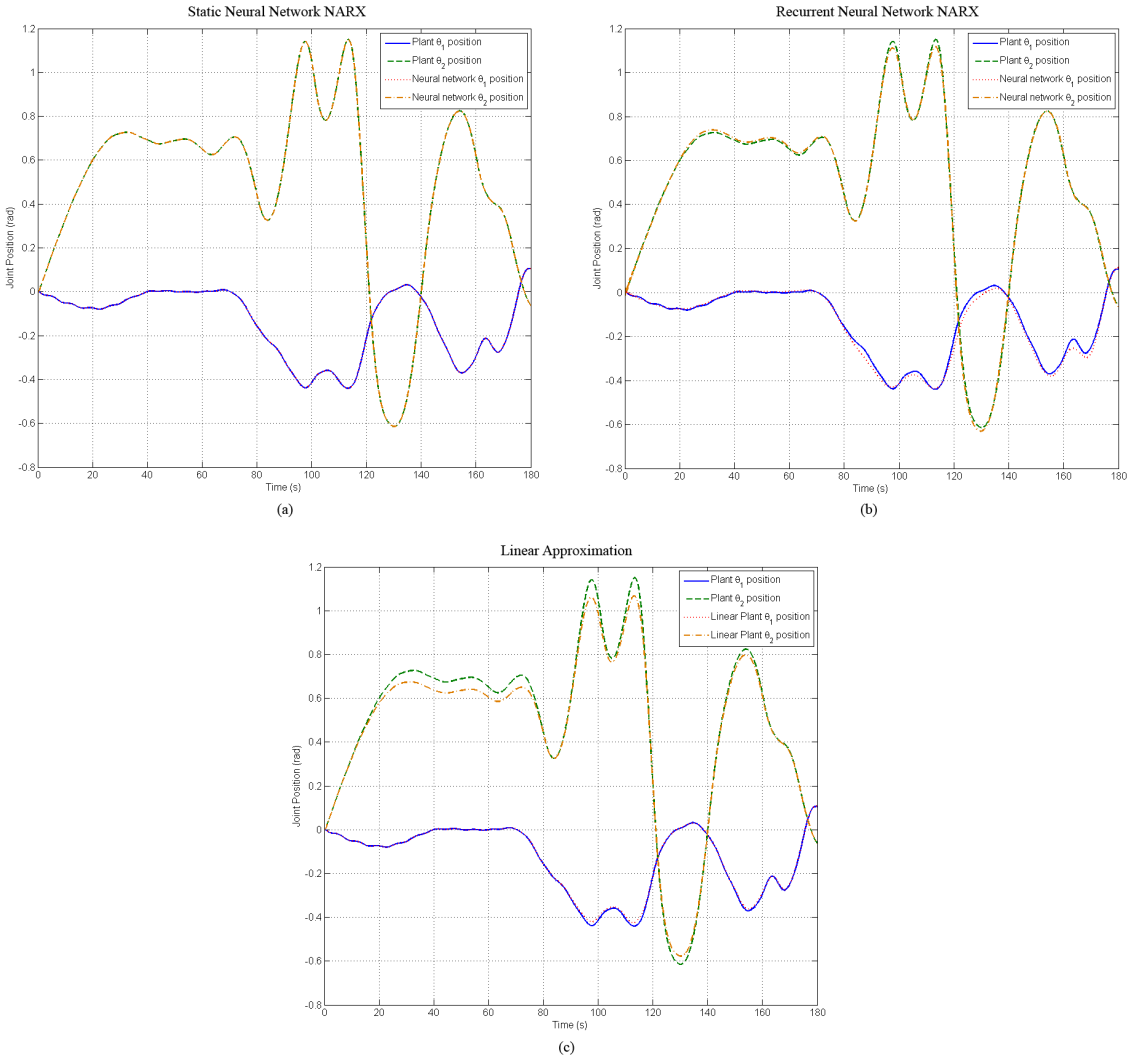


Figure 3.18: Output signal comparisson between the nonlinear plant and the: (a) static neural network NARX; (b) recurrent neural network NARX; (c) linear approximation.

3.4 Neural State Space

State space is a representation of a dynamical system. The state space representation tries to solve two shortcomings of the transfer function: representation of internal states and MIMO systems. The problem of the representation of internal states comes from the fact that a transfer function is designed to give the relation between the input and output of a system. However it is necessary, sometimes, to know how the derivative of the output changes with the inputs, especially in control theory, where the internal states can take prohibitive values even when the output is seemingly doable [40]. The problem of the representation of MIMO systems is the biggest limitation of transfer functions because they need to be grouped in a transfer matrix that has a transfer function in each entry (a 2x2 MIMO system would have four transfer functions), which brings an added complexity to stability analysis and control [40].

The state space solves both problem with a representation that shows the internal states while having a unified form that can be applicable to SISO, MIMO, SIMO and MISO system, Eq. 3.45:

$$\begin{aligned} \dot{\mathbf{s}}(t) &= \mathbf{A}\mathbf{s}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{s}(t) + \mathbf{D}\mathbf{u}(t) \end{aligned} \tag{3.45}$$

where $\mathbf{s}(t)$, $\mathbf{y}(t)$ and $\mathbf{u}(t)$ are the state, output and input vectors, and $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$ are the state, input, output and feedthrough matrices.

One can also have discrete time and nonlinear state space representations, Eq. 3.46 and Eq. 3.47:

$$\begin{aligned} \mathbf{s}(k+1) &= \mathbf{A_d}\mathbf{s}(k) + \mathbf{B_d}\mathbf{u}(k) \\ \mathbf{y}(k) &= \mathbf{C}\mathbf{s}(k) + \mathbf{D}\mathbf{u}(k) \end{aligned} \tag{3.46}$$

$$\begin{aligned} \dot{\mathbf{s}}(t) &= \mathbf{f}\left(t, \mathbf{s}(t), \mathbf{u}(t)\right) \\ \mathbf{y}(t) &= \mathbf{g}\left(t, \mathbf{s}(t), \mathbf{u}(t)\right) \end{aligned} \tag{3.47}$$

where $\mathbf{A_d}$ and $\mathbf{B_d}$ are the discrete versions of the state and input matrices, and $\mathbf{f}$ and $\mathbf{g}$ are the nonlinear state and output functions.

Apart from the advantages stated above, state space representation also brings extra advantages: it transforms an n-order differential equation into n first-order systems and by applying a rotation on the state vector, one can have an equivalent systems that can have special properties (e.g. a diagonal state matrix explicitly shows the poles of the system).

In neural networks system identification, static and recurrent neural network NARX are equivalent to a transfer function (or transfer matrix for MIMO systems), and hence inheriting the same problems. A neural space state has a representation similar to Eq. 3.47, but now the $\mathbf{f}$ and $\mathbf{g}$ functions are a first-order recurrent and a static neural networks, respectively, Eq. 3.48.

$$\begin{aligned} \hat{\mathbf{s}}(k+1) &= \mathbf{f}\left(\hat{\mathbf{s}}(k), \mathbf{u}(k), \mathbf{P}_f\right) \\ \hat{\mathbf{y}}(k) &= \mathbf{g}\left(\hat{\mathbf{s}}(k), \mathbf{u}(k), \mathbf{P}_g\right) \end{aligned} \tag{3.48}$$

where $\mathbf{P}_f$ and $\mathbf{P}_g$ are the network parameter vector for the $\mathbf{f}$ and $\mathbf{g}$ neural networks, respectively, and $\hat{\mathbf{s}}$ is the approximated state vector.

In spite of all the advantages there are two main disadvantages of using neural state spaces: two separate networks must now be tuned and neural state spaces have more parameters than static or recurrent neural network NARX, for an equivalent problem.

3.4.1 Gradient of a Neural Network State Space

The gradient of the neural state space is the aggregation of the gradients of a first-order recurrent network and of a static recurrent network. The gradient of the static neural network is calculated normally using Eq. 3.5 to Eq. 3.15. However, since one does not have the desired future states, part of the gradient of the recurrent network has to be changed from Eq. 3.39 to Eq. 3.49.

$$\begin{cases} \mathbf{E}_{N_P} = \left(\frac{\partial \hat{\mathbf{y}}_{N_P}}{\partial \hat{\mathbf{s}}_{N_P}}\right)^T \frac{\partial C}{\partial \hat{\mathbf{y}}_{N_P}} \\ \mathbf{E}_p = \left(\frac{\partial \hat{\mathbf{y}}_p}{\partial \hat{\mathbf{s}}_p}\right)^T \frac{\partial C}{\partial \hat{\mathbf{y}}_p} + \left(\frac{\partial \hat{\mathbf{s}}_{p+1}}{\partial \hat{\mathbf{s}}_p}\right)^T \mathbf{E}_{p+1} \end{cases} \tag{3.49}$$

where $\frac{\partial \hat{\mathbf{y}}_p}{\partial \hat{\mathbf{s}}_p}$ can taken from the jacobian of the static network and $\frac{\partial \hat{\mathbf{s}}_{p+1}}{\partial \hat{\mathbf{s}}_p}$ from the jacobian of the recurrent network.

Because it can be partially parallelized, the computation of the gradient of the neural state space is computationally lighter that a recurrent network NARX with the same number of parameters. Also, similarly to recurrent neural network NARX, neural state space are completely independent of the plant and can be used without requiring output of the plant.

3.4.2 Comparison Between NARX Neural Networks (Static and Recurrent) and Neural Space States

Due to their nature, neural state spaces only have the possibility of outperforming the NARX neural networks when the plant to the modeled is a MIMO system. The Double Pendulum problem presented in section 3.3.3 is an excellent candidate. The results for the best four network structures are presented in Table 3.5 and Fig. 3.19.

Table 3.5: Results of the neural state space modeling of a Double Pendulum for different network structures (the first structure is for the recurrent network and the second is for the static network

Structure	Final Train Cost	Best Test Cost	Iterations	Designation
6HT-4LF 6HT-2LF	3.96×10^{-6}	5.94×10^{-5}	10000	StateSpace1
3LF-6HT-4LF 3LF-6HT-2LF	7.80×10^{-6}	2.50×10^{-5}	5887	StateSpace2
6LF-6HT-4LF 6LF-6HT-2LF	3.11×10^{-6}	3.80×10^{-5}	10000	StateSpace3
3LF-6HT-6HT-4LF 3LF-6HT-6HT-2LF	5.70×10^{-6}	2.75×10^{-5}	8739	StateSpace4

From Table 3.5, one can see that the neural state spaces have better performance than recurrent neural network NARX. This happens, mainly, because of the processing power of neural state spaces is divided into a static network and a first-order recurrent network, instead of a full n-order recurrent NARXs. The division alleviates the ill-conditioning of the cost function, which becomes more apt for optimization. However this advantage comes with a cost, because of having two neural networks, neural state spaces are harder to tune and to find the optimal structure. When analyzing the convergence rate of the neural state spaces, Fig.3.19, it can be seen that the test convergence is less erratic than the convergence of recurrent neural network NARX. Due to the separation into a static and a simple recurrent network, the convergence can be seen as a transition between static neural network NARX and the recurrent neural network NARX.

Because of the smaller cost function values, the error in the modeling of the plant by neural state spaces is smaller, Fig. 3.20, than recurrent neural network NARX, Fig. 3.18b.

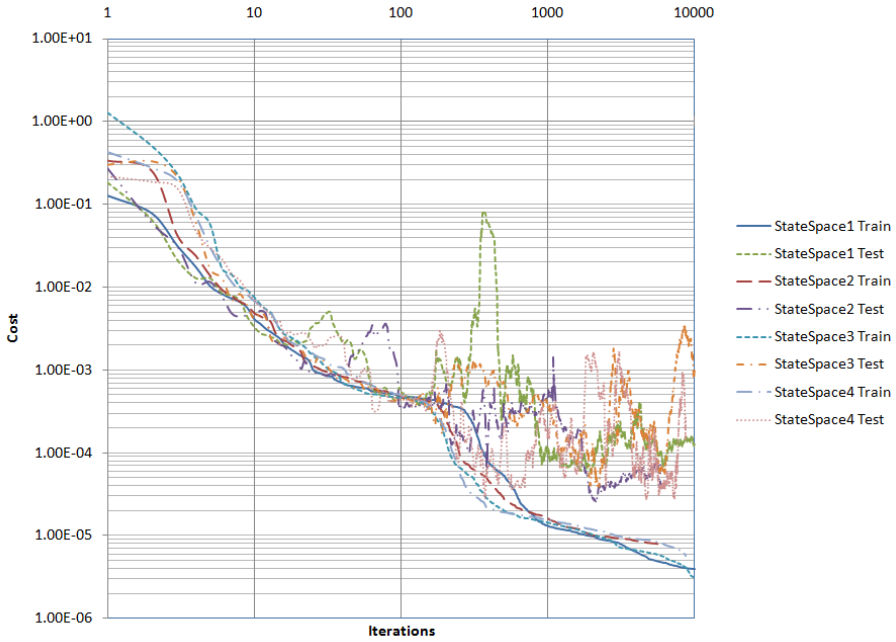


Figure 3.19: Convergence of the neural state space networks for the model of the double pendulum.

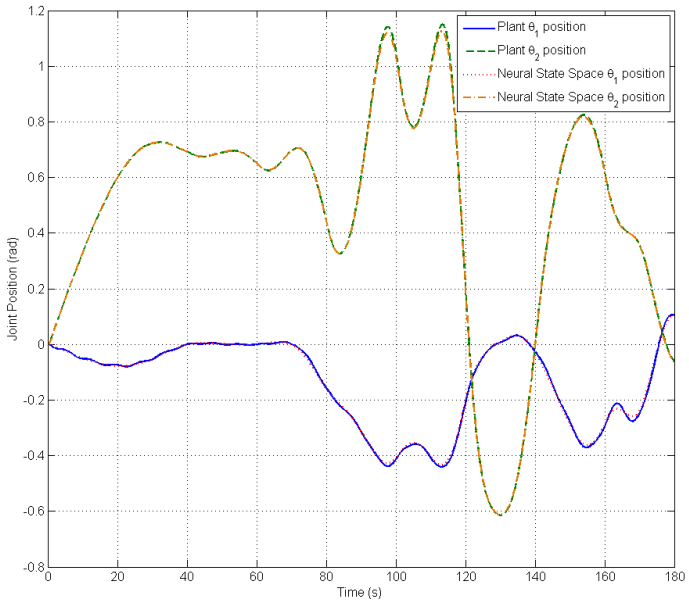


Figure 3.20: Output signal comparison between the nonlinear plant and the neural state space model.

3.5 Neural Network Control

Control is defined as a process that bring a system from its current state to a desired sate through a set of finite inputs. There are two types of control: feedforward and feedback control. In feedforward control, the controller corresponds to the inverse model of the plant. This kind of control is usually of theoretical interest only, because it is virtually impossible to obtain a exact mathematical model of a dynamic system and any discrepancy leads to a poor performance or even instability. In feedback control, the controller uses the error between a reference and the plant output to determine the next control action (plant input) and, thus, compensate the error of the plant. The most well known feedback control strategy is the proportional-integral-derivative (PID) controller, where the control action is the result of the pondered sum of the error, the derivative of the error and the integral of the error. Other linear feedback controllers are the lead and lag compensators, state space controller (the MIMO version of the PID), internal model control (IMC) and predictive controller [40].

When dealing with nonlinear plants, one can follow one of two paths: linearize the plant and use a linear control strategy or use a nonlinear controller. There are several types of nonlinear control strategies, being the most well known: the Feedback Linearization, Sliding Control, Adaptive Control (which includes Neural Network control), Gain-scheduling and Robust Control [42].

Being a subset of Adaptive Control, which also includes Fuzzy Control, Neural Network Control tries to adapt a neural network to compensate a nonlinear plant such that the output follows a predetermined reference. There are several neural network control strategies, mostly mimicking linear controllers: Neural Network Inverse Model Controller, Neural PID, Neural State Space Controller, Neural Internal Model Control and Neural Predictive controller. The first two control strategies will be described in detail in the following sections.

3.5.1 Neural Network Inverse Model Controller

Neural Network Inverse Model Control is a partially feedforward control strategy in which a neural network is trained to replicate the inverse of the plant. This control strategy, although more robust than its linear counterpart, is mainly of theoretical interest, mostly because of the same reasons of the impracticability of the inverse model linear controller, but also because neural network controllers are discrete time

controllers by nature[8] and a discrete time inverse model of a continuous plant can lead to causality problems [40].

Like all neural network controllers, the inverse model is a recurrent neural network with recurrence in the control actions and plant outputs (the recurrence of the plant outputs are an indirect recurrence), which means that the controller will share the ill-condition of the recurrent neural networks NARX. Also, because the inverse model cancels the effect the plant, one can any desired transfer function between the reference and the plant output (even 1).

The block diagram of a typical neural network inverse model controller can be seen in Fig. 3.21. where $R(t)$ and $R^*(t)$ are the reference in continuous and discrete

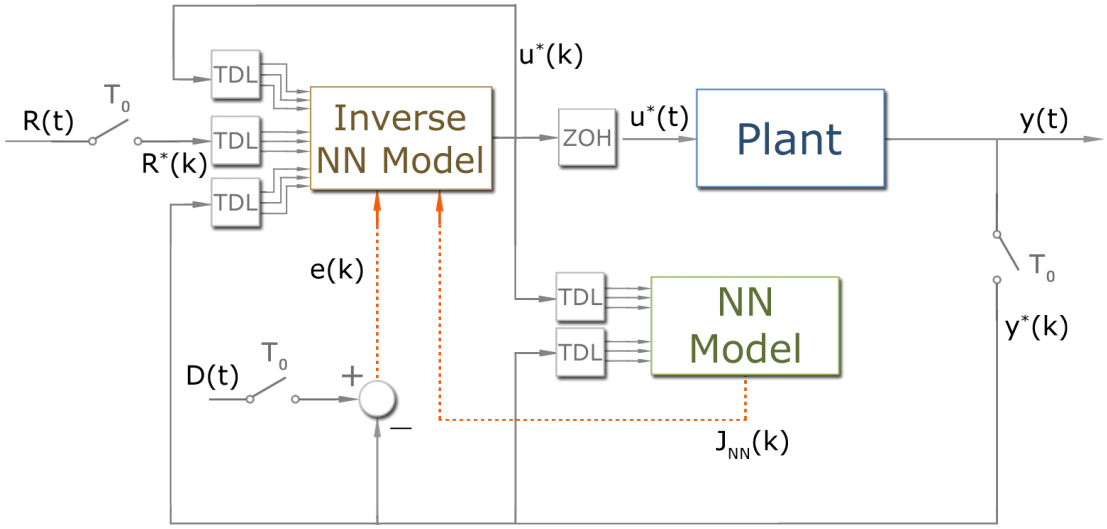


Figure 3.21: Block Diagram of a Neural Network Inverse Model Controller.

time signals, respectively; $u^*(t)$ and $u^*(k)$ are the control action in holded continuous and discrete time signals; $y(t)$ and $y^*(k)$ are the continuous and discrete time plant outputs; $D(t)$ is the desired plant output and $e(k)$ and $J_{NN}(k)$ are the training error and the approximated jacobian matrix of the plant, respectively (which will be necessary to calculate the gradient of the controller). Note that the dashed lines are only active during the training of the neural network inverse model.

[8]Continuous time neural networks also exist, but they are much more complex and require an analog implementation, which for large size networks can be prohibitive

Gradient of a Inverse Model Controller

The calculation of the gradient of the neural network that forms the inverse model controller is a good case-study, because it is the most generic case of a neural network controller. The objective is to have the gradient in the form of Eq. 3.36, reproduced in the following equation for convinience:

$$\frac{\partial C}{\partial \mathbf{W}^l} = \sum_{p=1}^{N_P} \mathbf{E}_p \left\{ \frac{\partial \hat{\mathbf{y}}_p}{\partial \mathbf{W}^l} \right\}$$

To accomplish this, one has to overcome two major hindrances: the existence of one extra indirect recurrence and the absence of the desired control action to determine the error of the controller. Both can be solved using the approximated jacobian matrix of the plant given by the neural network model, Fig .3.21.

Considering a generic second order plant to be controlled, Eq. 3.50, the cost function of the controller is given by Eq. 3.51:

$$G(z) = \frac{Y(z)}{U(z)} = \frac{b_2 z^2 + b_1 z + b_0}{z^2 + a_1 z + a_0} \tag{3.50}$$

$$\frac{\partial C}{\partial \mathbf{W}^l} = \frac{\partial C}{\partial \mathbf{y}_1}\frac{\partial \mathbf{y}_1}{\partial \mathbf{W}^l} + \cdots + \frac{\partial C}{\partial \mathbf{y}_p}\frac{\partial \mathbf{y}_p}{\partial \mathbf{W}^l} + \cdots + \frac{\partial C}{\partial \mathbf{y}_{N_P}}\frac{\partial \mathbf{y}_{N_P}}{\partial \mathbf{W}^l} \tag{3.51}$$

where $\mathbf{y}_p$ is the plant output, $\hat{\mathbf{u}}_p$ is the control action and $\frac{\partial \mathbf{y}_p}{\partial \hat{\mathbf{u}}_p}$ is the derivative of the plant outputs relatively to the control actions (taken from the plant jacobian matrix), $\frac{\partial \mathbf{y}_p}{\partial \mathbf{W}^l}$ is given by Eq. 3.52:

$$\begin{aligned} \frac{\partial \mathbf{y}_p}{\partial \mathbf{W}^l} = & \frac{\partial \mathbf{y}_p}{\partial \hat{\mathbf{u}}_p}\frac{\partial \hat{\mathbf{u}}_p}{\partial \mathbf{W}^l} + \frac{\partial \mathbf{y}_p}{\partial \hat{\mathbf{u}}_{p-1}}\frac{\partial \hat{\mathbf{u}}_{p-1}}{\partial \mathbf{W}^l} + \frac{\partial \mathbf{y}_p}{\partial \hat{\mathbf{u}}_{p-2}}\frac{\partial \hat{\mathbf{u}}_{p-2}}{\partial \mathbf{W}^l} \\ & + \frac{\partial \mathbf{y}_p}{\partial \mathbf{y}_{p-1}}\frac{\partial \mathbf{y}_{p-1}}{\partial \mathbf{W}^l} + \frac{\partial \mathbf{y}_p}{\partial \mathbf{y}_{p-2}}\frac{\partial \mathbf{y}_{p-2}}{\partial \mathbf{W}^l} \end{aligned} \tag{3.52}$$

and $\frac{\partial \hat{\mathbf{u}}_p}{\partial \mathbf{W}^l}$ is given by Eq. 3.53:

$$\frac{\partial \hat{\mathbf{u}}_p}{\partial \mathbf{W}^l} = \left\{ \frac{\partial \hat{\mathbf{u}}_p}{\partial \mathbf{W}^l} \right\} + \frac{\partial \hat{\mathbf{u}}_p}{\partial \hat{\mathbf{u}}_{p-1}}\frac{\partial \hat{\mathbf{u}}_{p-1}}{\partial \mathbf{W}^l} + \frac{\partial \hat{\mathbf{u}}_p}{\partial \hat{\mathbf{u}}_{p-2}}\frac{\partial \hat{\mathbf{u}}_{p-2}}{\partial \mathbf{W}^l} + \frac{\partial \hat{\mathbf{u}}_p}{\partial \mathbf{y}_{p-1}}\frac{\partial \mathbf{y}_{p-1}}{\partial \mathbf{W}^l} + \frac{\partial \hat{\mathbf{u}}_p}{\partial \mathbf{y}_{p-2}}\frac{\partial \mathbf{y}_{p-2}}{\partial \mathbf{W}^l} \tag{3.53}$$

To solve the direct recurrence, any explicit reference to $\frac{\partial \hat{\mathbf{u}}_p}{\partial \mathbf{W}^l}$ must be eliminated

from Eq. 3.52 and Eq. 3.53. The resulting gradient is given by Eq. 3.54:

$$\frac{\partial C}{\partial \mathbf{W}^l} = \frac{\partial C}{\partial \mathbf{y}_1}\left(\frac{\partial \mathbf{y}_1}{\partial \hat{\mathbf{u}}_1}\left\{\frac{\partial \hat{\mathbf{u}}_1}{\partial \mathbf{W}^l}\right\}\right) + \\ + \frac{\partial C}{\partial \mathbf{y}_2}\left(\frac{\partial \mathbf{y}_2}{\partial \hat{\mathbf{u}}_2}\left\{\frac{\partial \hat{\mathbf{u}}_2}{\partial \mathbf{W}^l}\right\} + \left(\frac{\partial \mathbf{y}_2}{\partial \hat{\mathbf{u}}_1} + \frac{\partial \mathbf{y}_2}{\partial \hat{\mathbf{u}}_2}\frac{\partial \hat{\mathbf{u}}_2}{\partial \hat{\mathbf{u}}_1}\right)\left\{\frac{\partial \hat{\mathbf{u}}_1}{\partial \mathbf{W}^l}\right\} + \left(\frac{\partial \mathbf{y}_2}{\partial \mathbf{y}_1} + \frac{\partial \mathbf{y}_2}{\partial \hat{\mathbf{u}}_2}\frac{\partial \hat{\mathbf{u}}_2}{\partial \mathbf{y}_1}\right)\frac{\partial \mathbf{y}_1}{\partial \mathbf{W}^l}\right) + \cdots \tag{3.54}$$

From Eq. 3.54 one can see that the coefficients of $\left\{\frac{\partial \hat{\mathbf{u}}_p}{\partial \mathbf{W}^l}\right\}$ and of $\frac{\partial \mathbf{y}_1}{\partial \mathbf{W}^l}$ can be rewritten in the following recurrent forms, Eqs. 3.55 and 3.56[9]:

$$\mathbf{A}_{ij} = \frac{\partial \mathbf{y}_i}{\partial \hat{\mathbf{u}}_j} + \sum_{k=j+1}^{i}\left(\mathbf{A}_{ik}\frac{\partial \hat{\mathbf{u}}_k}{\partial \hat{\mathbf{u}}_j}\right), j = 1, \cdots, i \tag{3.55}$$

$$\mathbf{B}_{ij} = \frac{\partial \mathbf{y}_i}{\partial \mathbf{y}_j} + \sum_{k=j+1}^{i}\left(\mathbf{A}_{ik}\frac{\partial \hat{\mathbf{u}}_k}{\partial \mathbf{y}_j}\right), j = 1, \cdots, i-1 \tag{3.56}$$

which simplifies Eq. 3.54 into Eq. 3.57

$$\frac{\partial C}{\partial \mathbf{W}^l} = \sum_{i=1}^{N_p}\left(\frac{\partial C}{\partial \mathbf{y}_i}\left(\sum_{j=1}^{i}\left(\mathbf{A}_{i\,j}\left\{\frac{\partial \hat{\mathbf{u}}_j}{\partial \mathbf{W}^l}\right\}\right) + \sum_{j=1}^{i-1}\left(\mathbf{B}_{i\,j}\frac{\partial \mathbf{y}_j}{\partial \mathbf{W}^l}\right)\right)\right) \tag{3.57}$$

With Eq. 3.57, only the indirect recurrence remains to be resolved. Rearranging Eq. 3.57 to remove any explicit reference to $\frac{\partial \mathbf{y}_j}{\partial \mathbf{W}^l}$, Eq. 3.58 is obtained:

$$\frac{\partial C}{\partial \mathbf{W}^l} = \left(\frac{\partial C}{\partial \mathbf{y}_{N_p}}\right)\sum_{k=1}^{N_p}\left(\mathbf{A}_{N_p\,k}\left\{\frac{\partial \hat{\mathbf{u}}_k}{\partial \mathbf{W}^l}\right\}\right) + \\ + \left(\frac{\partial C}{\partial \mathbf{y}_{N_p-1}} + \frac{\partial C}{\partial \mathbf{y}_{N_p}}\mathbf{B}_{N_p\,(N_p-1)}\right)\sum_{k=1}^{N_p-1}\left(\mathbf{A}_{(N_p-1)\,k}\left\{\frac{\partial \hat{\mathbf{u}}_k}{\partial \mathbf{W}^l}\right\}\right) + \cdots \tag{3.58}$$

Similarly to Eq. 3.54, one can see that the coefficients of the sums can be rewritten in the following recurrent form, Eq. 3.59:

$$\mathbf{D}_i = \frac{\partial C}{\partial \mathbf{y}_i} + \sum_{k=i}^{N_p}\left(\mathbf{D}_k\mathbf{B}_{ki}\right) \tag{3.59}$$

[9]Please note that if for a certain combination of i and j the derivatives $\frac{\partial \mathbf{y}_i}{\partial \hat{\mathbf{u}}_j}$ and $\frac{\partial \mathbf{y}_i}{\partial \mathbf{y}_j}$ do not exist, they are removed from the equations

which yields Eq. 3.60 when Eq. 3.59 is replaced back into Eq. 3.58:

$$\frac{\partial C}{\partial \mathbf{W}^l} = \mathbf{D}_{N_p} \sum_{k=1}^{N_p} \left(\mathbf{A}_{N_p\, k} \left\{ \frac{\partial \hat{\mathbf{u}}_k}{\partial \mathbf{W}^l} \right\} \right) + \mathbf{D}_{N_p-1} \sum_{k=1}^{N_p-1} \left(\mathbf{A}_{(N_p-1)\, k} \left\{ \frac{\partial \hat{\mathbf{u}}_k}{\partial \mathbf{W}^l} \right\} \right) + \\ + \cdots + \mathbf{D}_1 \mathbf{A}_{11} \left\{ \frac{\partial \hat{\mathbf{u}}_1}{\partial \mathbf{W}^l} \right\} \tag{3.60}$$

The final step is to rearrange Eq. 3.60 to remove the sums of the direct derivatives and obtain an equation similar to Eq. 3.36, resulting in the final recurrent form Eq. 3.61, which corresponds to the inverse model corrected error.

$$\mathbf{E}_i = \sum_{k=i}^{N_p} \mathbf{D}_k \mathbf{A}_{ki} \tag{3.61}$$

Comparing to the gradients of the other neural networks, the gradient of the cost function of the inverse model is the most complex and computationally heavy process. Nevertheless, and in par with recurrent networks, the gradient of the cost function could still be reduced to a series of matrix operations.

3.5.2 Neural PID

Proportional-integral-derivative (PID) control are the most well known and used control strategy. The objective of the controller is to compensate any deviation from a desired reference (due to external perturbations or variations in the reference itself) by feeding the plant the appropriate inputs. These inputs are determined by a set of basic actions done on the tracking error (the difference between the reference and the output): proportional action, integral action and the derivative action. Each basic action can be used standalone or by any combination of the three (the most common are the P, PI, PD and PID controllers), and each action has a specific effect on the plant output: the proportional action reduces the effects of external perturbations and reduces the tracking error, however, depending on the value of the constant, it can lead to instabilities in the closed loop (specially in higher-order plants) and it does not eliminate the static error; the integral action eliminates the static error, but it also reduces the stability of the control loop and can introduce unwanted oscillations in the output signal; finally, the derivative action creates a faster response to deviations from the reference and increases the stability of the closed loop, unfortunately

it will also increase the susceptibility of the closed loop to any noise in the input or output signals [40].

The standard PID controller is given by Eq. 3.62 and the block diagram can be seen in Fig. 3.22.

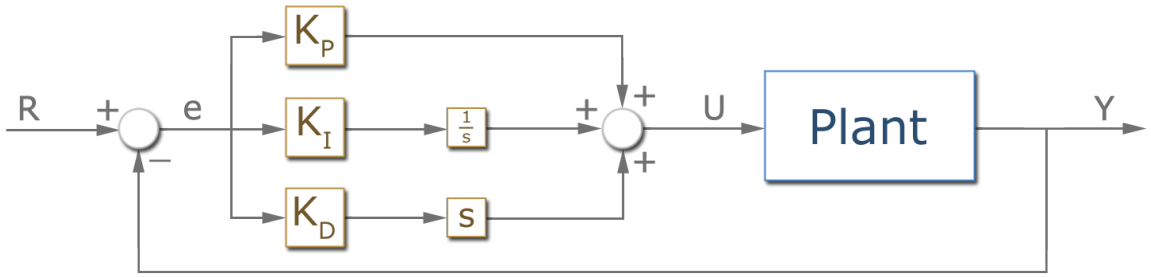


Figure 3.22: Block Diagram of a PID controller.

$$U(s) = K_p E(s) + K_i \frac{1}{s} E(s) + K_d s E(s) \tag{3.62}$$

where $U(s)$ and $E(s)$ are the control actions and the tracking error, respectively, and K_p, K_i and K_d are the proportional, integrative and derivative constants that equilibrate and regulate the influence of each action on the control loop.

Although Eq. 3.62 is a continuous-time transfer function, one can also have a discrete-time PID controller, which has the following transfer function, Eq. 3.63.

$$\frac{U(z)}{E(z)} = \frac{q_0 + q_1 z^{-1} + q_2 z^{-2}}{1 - z^{-1}} \tag{3.63}$$

The biggest challenge in discrete-time PID control lies on finding the appropriate values for the coefficients q_0, q_1, q_2. There are two method to determine these values: the approximation method and the direct method. In the approximation method, the continuous-time PID transfer function, Eq. 3.62, is discretized using the forward Euler, backward Euler or Tustin approximations, which yield the coefficients on Table 3.6.

In the direct method, the coefficients are found using the performance of the closed loop output, with tools such as the discrete-time root locus plot or the discrete-time Nyquist plot.

Applying the same structure of the discrete-time PID to a neural network, one

Table 3.6: Discrete-time PID coefficients using the approximation method

Approximation type	q_0	q_1	q_2
Forward Euler	$K_p + K_d\frac{1}{T_0}$	$K_iT_0 - K_p - 2K_d\frac{1}{T_0}$	$K_d\frac{1}{T_0}$
Backward Euler	$K_p + K_d\frac{1}{T_0} + K_iT0$	$-K_p - 2K_d\frac{1}{T_0}$	$K_d\frac{1}{T_0}$
Tustin	$K_p + K_d\frac{1}{T_0} + 2K_iT0$	$2K_iT_0 - K_p - 2K_d\frac{1}{T_0}$	$K_d\frac{1}{T_0}$

will have the following neural network controller, Eq. 3.64:

$$\hat{\mathbf{u}}(k) = \mathrm{NN}(\hat{\mathbf{u}}(k-1), \mathbf{e}(k), \mathbf{e}(k-1), \mathbf{e}(k-2)) \tag{3.64}$$

The cost function used to train the neural PID has to be slightly different. In the inverse model control, the objective of the controller is to transform the plant into another system with certain desired parameters, in the neural PID, the objective is to decrease the error between the plant output and a desired reference. The usual cost function (the sum of the squared errors) gives the same "importance" to each error between the desired and the obtained outputs, meaning that a point in the transient state has the same weight in the cost as a point in the steady state of the response. For a inverse model controller, this behavior is appropriate because, as stated above, the objective is to transform every point of the plant output into the output of another plant. However, in the Neural PID, one is mostly interested in the steady state response[10] of the control loop. If one was to use the sum of the squared error, as a cost function, when training the Neural PID, the optimizer will try to find a set of weights that tries equally minimize the error of the transient and steady states in the time interval represented in the input/output data, regardless of the stability of the control loop. In fact, when using this cost function to train PID controllers for simple LTI systems, most of them resulted in unstable control loops when run past the training time interval.

To solve this problem one has to use a different cost function and there are to possible solutions. The first one resides in the direct method of Lyapunov's stability theory. The direct method states that a system is stable around the origin of the state space if a scalar function, V, of the system states, $\mathbf{x}$, exists such that $V(\mathbf{x})$ is a

[10]Please note that when designing a regular PID, transient state parameters like the overshoot and raising time are also important, but they take a secondary importance when comparing to overall convergence of steady state.

continuous positive definite function, $\dot{V}(\mathbf{x})$ is a continuous negative definite function and $V(\mathbf{x}) \to \infty$ as $\|\mathbf{x}\| \to \infty$[11],[42]. If one defines Eq.3.65, to represent the control loop, any Lyapunov function (V) defined for it can also be used as a cost function for the Neural PID.

$$\mathbf{e} = (\mathbf{I} - \mathbf{C}\ \mathbf{P})^{-1}\,\mathbf{R} \tag{3.65}$$

where $\mathbf{C}$ and $\mathbf{P}$ are the controller and plant transfer functions and $\mathbf{R}$ and $\mathbf{e}$ are the reference and the error between the plant output and the reference.

The challenge to the optimizer lies then in finding the correct set of weights of the neural PID that makes a pre-determined function $V(\mathbf{e})$ into a Lyapunov function, which not only guarantees that the error is decreased in the time interval defined by the input/output training data set, but the resulting loop is also stable. On top of the stability, one also can make the guarantee that the resulting control loop is exponentially stable if the constrain defined in Eq. 3.66 is also added to the optimizer:

$$\|e(kT_0)\| \leq \alpha \|e(0)\| \exp(-\beta k T_0) \tag{3.66}$$

where k is the number of sample times, T_0 is the sample time, and α and β are positive scalar values.

The main disadvantages of using this cost function is that the Lyapunov function candidate is different for each system to be controlled. Also, to guarantee that $V(\mathbf{e})$ is a Lyapunov function, the optimizer has to minimize the cost function with points where $\dot{V}(\mathbf{e})$ is always negative and, if one is aiming for exponential stability, satisfying Eq. 3.66. The addition of constrains to the optimization of the weights brings extra complications to the whole process and needs special methods that are not contemplated in section 3.2.3.

The second solution to the cost function problem, which is not as powerful as the Lyapunov's direct method, is to make the summation of the squared errors for the steady state only, Eq. 3.67:

$$C = \frac{1}{2} \sum_{p=p_s}^{N_P} \sum_{i=1}^{N_O} (R_{ip} - y_{ip})^2 \tag{3.67}$$

where R is the reference signal, p_s is the desired steady state start time and N_p is the

[11]Please note that this theorem can be used to test the stability any point in the state space by applying a translation to the space to move its origin to the desired point.

last point in the data set, which acts as a surrogate for the response at infinity.

Because Eq.3.67 does not start a the beginning of the data set, it increases the freedom of optimizer and concentrates only on the steady state. The great difficulty of using this cost function is finding the appropriate steady state start time, p_s and N_p. The former cannot be too close to zero or falls into the problems of the normal cost function, Eq. 3.4, and N_p as to be far enough from p_s to guarantee that the system if completely in the steady state, however it cannot be to far or it will add too much data points with little information to be extracted from them. The correct equilibrium between both parameters must be found for each system to be controlled. Usually, from all the tests done, the best results are obtained by setting p_s at least twice the settling time of the plant to be controller and N_p ten times further from p_s. If the resulting system is stable, then one can start bringing p_s closer to zero (and N_p accordingly) until the resulting becomes unstable or the optimizer is unable to find a minimum (usually this happens because the cost function becomes too ill-conditioned). The biggest advantage of Eq. 3.67, when compared to the cost function based on the Lyapunov's method, is that a normal unconstrained optimizer can be used, although the tradeoff is the lack of a strong guarantee that the control loop is stable.

The training of a Neural PID also has to have special care. The first precaution to have is to verify that the amplitude of the control action needed to reach a certain reference is not outside of the training input/output space of the neural network model of the plant, because the further the amplitude is from the training space, the more difficult the optimization process becomes and the result will have a lower performance. This happens because the jacobian matrix calculated by the neural network model is only valid inside the training space.

Gradient of the Neural PID

The gradient of the neural PID cost function is a subset of the gradient of the inverse model cost function. Unlike the inverse model, where the number of recurrent control actions and plant outputs depends on the order and nonlinearity of the plant, in the neural PID one has, always, three recurrences in plant outputs and one control action recurrence, Eq. 3.64.

The equation of the gradient of the neural PID cost function is similar to Eq. 3.51 and Eq. 3.52, however $\frac{\partial \hat{\mathbf{u}}_p}{\partial \mathbf{W}^l}$ is now given by Eq. 3.68 instead of Eq. 3.53.

$$\frac{\partial \hat{\mathbf{u}}_p}{\partial \mathbf{W}^l} = \left\{ \frac{\partial \hat{\mathbf{u}}_p}{\partial \mathbf{W}^l} \right\} + \frac{\partial \hat{\mathbf{u}}_p}{\partial \hat{\mathbf{u}}_{p-1}} \frac{\partial \hat{\mathbf{u}}_{p-1}}{\partial \mathbf{W}^l} - \frac{\partial \hat{\mathbf{u}}_p}{\partial \mathbf{e}_{p-1}} \frac{\partial \mathbf{y}_{p-1}}{\partial \mathbf{W}^l} - \frac{\partial \hat{\mathbf{u}}_p}{\partial \mathbf{e}_{p-2}} \frac{\partial \mathbf{y}_{p-2}}{\partial \mathbf{W}^l} - \frac{\partial \hat{\mathbf{u}}_p}{\partial \mathbf{e}_{p-3}} \frac{\partial \mathbf{y}_{p-3}}{\partial \mathbf{W}^l} \quad (3.68)$$

From this point forward the demonstration of the neural PID gradient and of the inverse model gradient is done in a similar fashion, with some notable exceptions. Because, in neural PID, there is only one recurrence in the control action, the difference between j and i in Eq. 3.55 is always lesser or equal to one, which simplifies to Eq. 3.69:

$$\mathbf{A}_{ij} = \frac{\partial \mathbf{y}_i}{\partial \hat{\mathbf{u}}_j} + \mathbf{A}_{i(j+1)} \frac{\partial \hat{\mathbf{u}}_{j+1}}{\partial \hat{\mathbf{u}}_j} \quad (3.69)$$

The fixed recurrences also brings changes to Eq. 3.56, where the difference between j and i indexes is always lesser or equal to three. Also, because how the tracking error is calculated, there are some signal changes, Eq. 3.70:

$$\mathbf{B}_{ij} = \frac{\partial \mathbf{y}_i}{\partial \mathbf{y}_j} - \sum_{k=1}^{3} \left(\mathbf{A}_{i(j+k)} \frac{\partial \hat{\mathbf{u}}_{j+k}}{\partial \mathbf{e}_j} \right) \quad (3.70)$$

The remaining equations are equivalent to the ones belonging to inverse model gradient. This compatibility is very advantageous, because it enables a high code reusability.

Performance of Neural PID Controller

To test the performance of neural PID controllers several tests were done. The first one uses a second order SISO LTI transfer function, Eq. 3.71, as the plant to be controlled. The test begins with the training of a static neural network model of the plant, which yielded the final cost function value of 2.13×10^{-13}.

$$G(s) = \frac{1}{s^2 + 2s + 2} \xrightarrow[T_0=0.1\text{s}]{\text{ZOH}} G(z) = \frac{4.675 \times 10^{-3} z + 4.373 \times 10^{-3}}{z^2 - 1.8z - 0.819} \quad (3.71)$$

Several network configurations were tested as well several combinations of p_s and N_P constants. Table 3.7 shows some results obtained for a controller with one hidden layer with six hyperbolic tangent neurons.

The first observation one can take from Table 3.7 is that the optimization process is done with a very small number of iterations, which results from the fact that the cost function is extremely ill-conditioned (worst than the ill-condition observed in

Table 3.7: Some results for the training of the Neural PID for a SISO second order LTI plant.

Constants (p_s/N_P) (s)	Final Train Cost	Best Test Cost	Iterations
4/14	3.38×10^{-3}	1.24×10^{-3}	16
4/10	5.22×10^{-4}	3.15×10^{-4}	24
2/10	2.05×10^{-2}	1.74×10^{-2}	19

recurrent neural network NARX). The effects of the ill-condition of the cost function are more explicit in the next example. Nevertheless, all resulting controllers were stable and the response of the control loop using the 4/14 controller can be seen in Fig. 3.23.

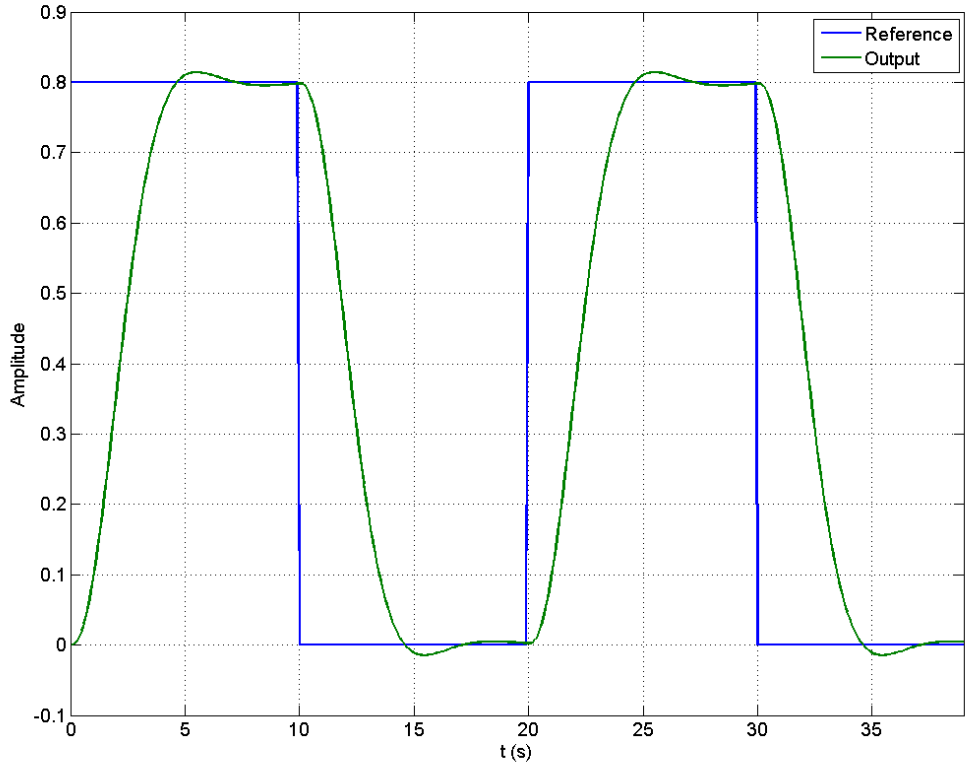


Figure 3.23: Time response to a squared wave reference of the 4/14 Neural PID controller.

The second test was done with the double pendulum, Eq. 3.44, as the plant to

be controlled, with both arms with a 10kg mass and a length of 1m. In next table, Table 3.8, the results for a neural PID with sixteen hyperbolic tangent neuron in the hidden layer are show for some combinations of the p_s and N_P constants.

Table 3.8: Some results for the training of the Neural PID for a MIMO second order LTI plant (the double pendulum).

Constants (p_s/N_P) (s)	Final Train Cost	Best Test Cost	Iterations
15/200	1.05	4.64×10^{-1}	45
20/200	2.55×10^{-1}	1.02×10^{-1}	23
60/300	9.73×10^{-6}	9.79×10^{-4}	19

Again, the number of iterations that the optimizer did is very low. Also, by changing the p_s and N_P constants very different results can be obtained and changing, even slightly, the start weights leads to different result, as it can be seen in Fig.3.24(a) and (b), taken for the same network structure and constants but with different starting points[12].

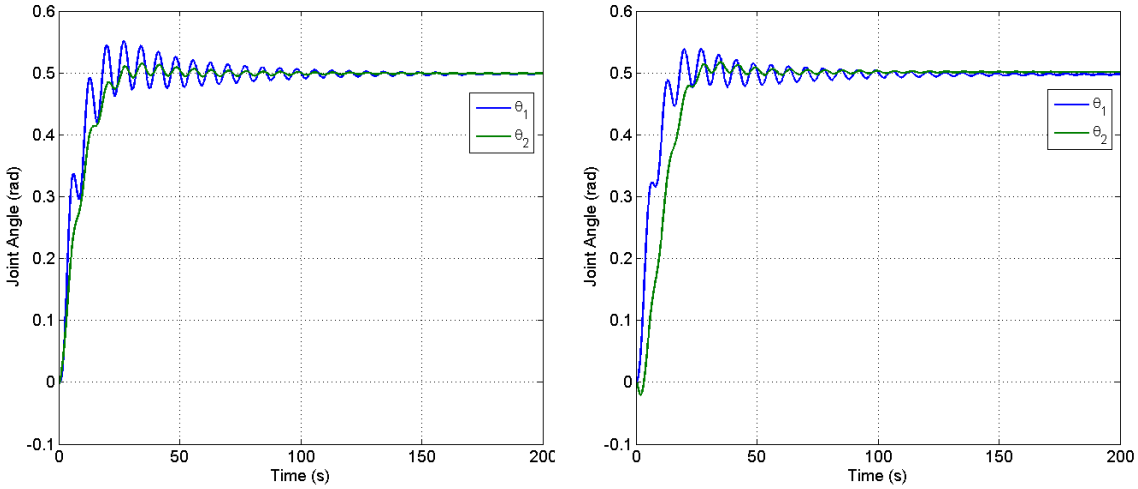


Figure 3.24: Time response to a step reference of .5 rad of two 20/200 Neural PID controllers with 16 hyperbolic tangent neurons in the hidden layer.

[12]The starting points of both controllers are taken from inside an hypercube with a side of 0.005

3.5.3 Other Possible Control Strategies

Apart from the neural model inverse model controller and the neural PID, one can have several other control strategies using neural networks based on traditional control strategies or in variations of PID controllers. One variation of the PID controller is a PID controller with the constants calculated by a neural network [43]. This control strategy is somewhat different from the Neural PID, presented in section 3.5.2, in the aspect that the Neural PID is a completely nonlinear controller whereas this controller is a linear time-variant PID controller. Also, on the same category of PID based neural network controllers, one can also have a LTI PID controller assisted by a neural network,[44]. In this controller, the control actions produced by the regular PID controller are added to a neural network with the error as input. In this configuration the neural network is only correcting the control action of an already stable and controlled loop. Beyond PID control-based neural network controllers, one can also apply neural networks to other linear control strategies, such as the internal model controller or predictive controllers [24].

3.6 Controller for the Hand Prosthesis

Using the dynamic model developed in Section 2.2, one can now proceed to the design of the controller. This process will be divided into three steps: the choice of the type of control strategy to be used, the modeling of the dynamics with a neural network in order to assist the training of the controller, and, finally, the training of the controller itself.

3.6.1 Control Strategies for the Human Hand Prosthesis

The dynamic system of the prosthesis is divided into the dynamics of the structure of the hand and by the actuation system, which is composed by each artificial muscle.

The are two paths that one can take to approach the problem and obtain a control strategy: the first, is to use a single controller for both the actuation system and the dynamics of the prosthesis, Fig. 3.25, while the second strategy consist in a master controller to control the overall dynamics of the system and slave controllers for each actuator, Fig. 3.26.

The major difference between the two strategies is the complexity of the main controller. In the first strategy, the controller has to deal, not only, with the dynamics

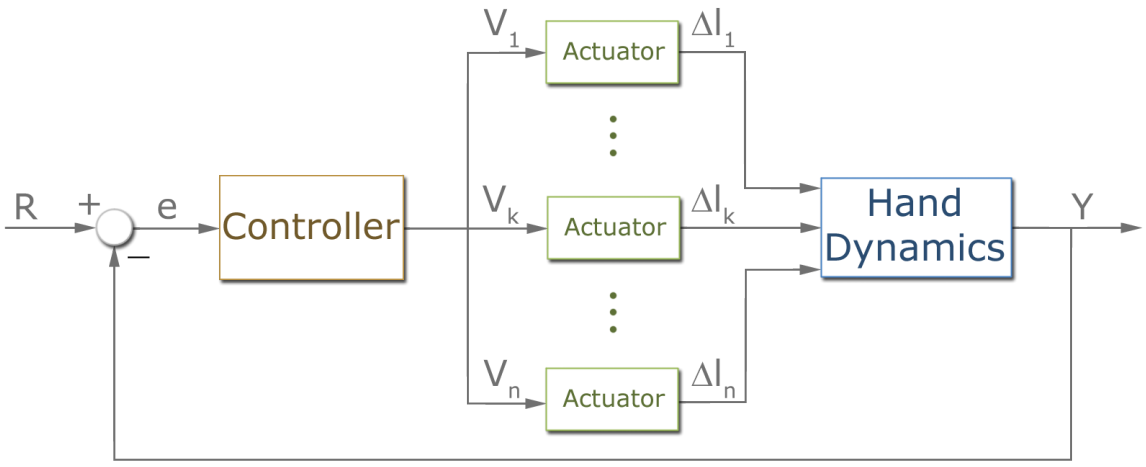


Figure 3.25: Single controller control strategy.

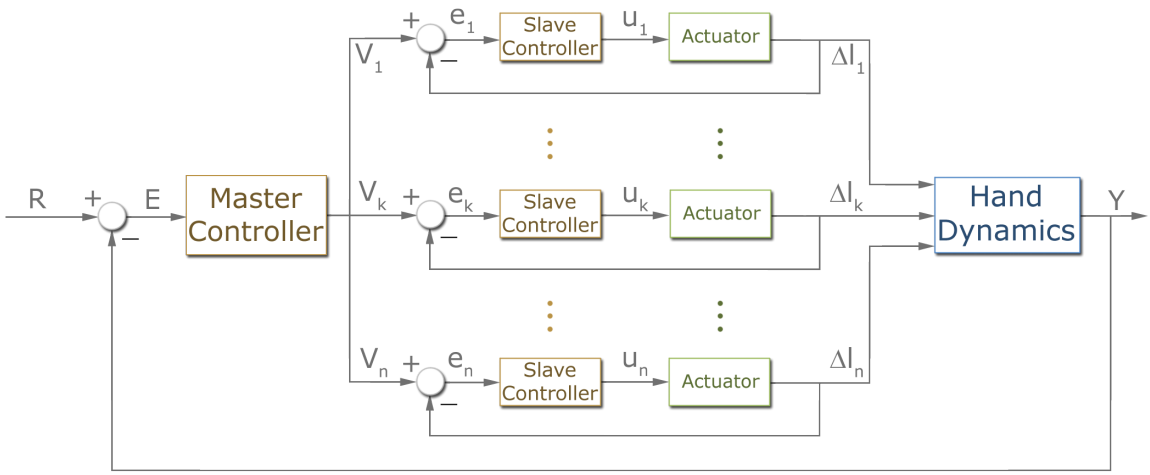


Figure 3.26: Master-slave controller control strategy.

of the hand, but also with the dynamics of each actuator, which leads to a longer and more difficult learning. However, the single controller strategy has the advantage of being centralized and easier to implement. On the other hand, the master-slave strategy has the master controller dealing only with the dynamics of the hand while delegating the control of each actuator to its controller. The main advantage is the simpler controllers and, consequently, easier to train. Also, because of the separation, each slave controller can be trained and tuned independently. On the disadvantage side, the master-slave strategy requires a more complex implementation.

3.6.2 Modeling the Human Hand with a Static Neural Network NARX

The starting point in neural network modeling are the static neural networks. This kind of network, as seen in section 3.2, is the simplest and easiest to train, making it perfect to establish an initial network configuration. Also, static neural networks can also be used in control as the plant model and, since, the actual plant has to be present, the main disadvantage of this type of network is lessened.

The first challenge in the training process is the acquisition of a correct training and testing data set. The hand, although inherently stable, has several nonlinearities, being the most severe the saturations. A special care must be taken when obtaining data from a system with saturations, because the output signals can saturate for long periods of time, which does not bring any extra information about the plant. A good training signal has to have a good proportion of saturated signals and unsaturated signals. Due to the high number of input and output signal, obtaining an optimal data set for the hand is extremely difficult and only after several attempts, an acceptable input signal was found.

The input signal was created using a cubic spline interpolation of thirty random frequency points between -0.1 to 0.1rad/s, which, in turn, is used in a polynomial chirp signal with a duration of 300s and a sample time of 0.01s. Once the chirp signal was obtained, it was scaled to have a better match between the amplitudes and the forces needed to mode each joint of the hand (e.g. the wrist flexion/extension joint needs larger forces that the joints of the fingers).

The best network structures and their results can be seen in Table 3.9.

Table 3.9: Results of the static neural network modeling of the human hand for different network structures

Structure	Final Train Cost	Best Test Cost	Iterations	Designation
(54LG/55HT)-22LF	3.87×10^{-4}	3.83×10^{-4}	20000	StaticHand1
(26LG/27HT)-22LF	5.45×10^{-4}	4.74×10^{-4}	20000	StaticHand2
5HT-10HT-2LF	4.45×10^{-5}	1.98×10^{-4}	911	StaticHand3

In Fig. 3.27 one can see the comparison between the outputs from the plant and the neural network model (StaticHand3) for four joints of the hand.

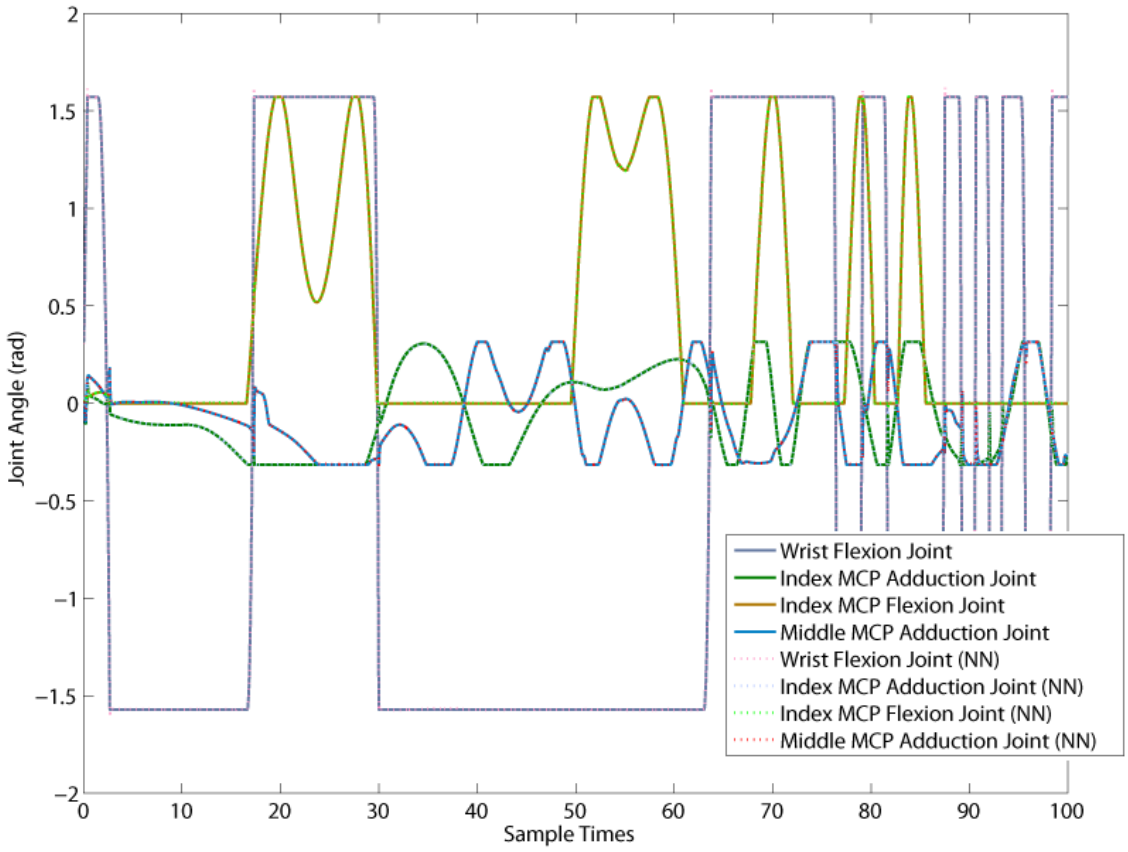


Figure 3.27: Comparison between the outputs of the plant and the StaticHand3 network for the wrist flexion joint, both DoFs of the metacarpophalangeal (MCP) joint of the index finger, and the adduction movement of the MCP joint of the middle finger.

3.6.3 Development of the Controller for the Human Hand Prosthesis

The dynamics of the human hand have several features that makes it a particularly difficult system to control, being the most notable the high degree of coupling between the joints and the presence of hard and soft nonlinearities. So, a special care must be taken in the training of the neural network controller.

The neural network controller for the human hand is based on the Neural PID, though some modifications must be done to simplify the problem. The first change is the separation of the controller into two separate controllers, one to perform the gravity control and the second to minimize the tracking error. This kind of controller strategy is common in control of robotic manipulators [40], because the gravity control takes the load of controlling the known effects of the gravity (these effects can be taken from the dynamics of the manipulator with very accurate results) from the tracking controller. For the human hand controller, a static neural network was used to map the inverse dynamics of the hand with no applied forces or torques. Because the rigid-body model is not the a real representation of the full system, it is expected a certain degree of difference between the model and the real plant, however, the neural network can be trained online and adapt to the new plant.

The second simplification resides in the training method of the tracking controller. The cost function of neural PID controller for the full hand is extremely ill-conditioned and as it can be seen in Fig. 3.28, which depicts the cost function for a linear PD controller for the flexion/extension joint of the wrist. It can be seen that the cost function for just one degree of freedom is plagued by local minima. This ill-condition is worsened when extra joints are added to the controller. To solve this problem, a controller for each joint must trained independently and by a certain order, and the resulting networks are merged into one neural network, with the new network parameters are initialized with very small values (in the order of 10^{-8} or less). The final controller is, then, retrained to optimized the new parameters. The difference between this method ad the one described in section 3.5.2, it that instead of using a random starting point for the controller (which has a very high probability of creating an unstable loop), the stating point is already stable and controller, leaving to the final optimization the fine-tuning of the controller.

The order in which the individual controller are to be trained depends on the importance of each joint. The joint that affects most the hand is the wrist joint and,

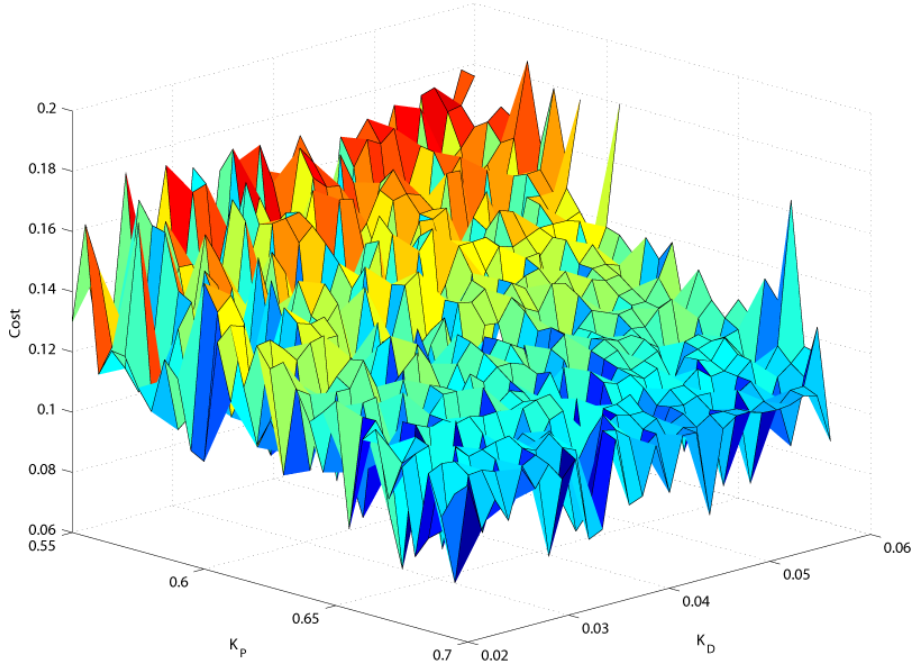


Figure 3.28: Cost function of a linear PD (where K_P and K_D are the proportional and derivative constants, respectively) controller for the flexion/extension joint of the wrist.

consequently, must be the first one to be trained. Because the wrist joint is a two degree of freedom joint and, also, because the two DoFs are coupled (mainly due to the variable saturation of the adduction/abduction DoF), the two controllers that will control the wrist must trained by turn, one after the other, until a sufficient performance is obtained. The initial wrist training for a grip movement can be seen in Fig. 3.29.

The next joints to be trained are the metacarpophalangeal joints (with exception of the thumb). They are also two DoF joints with characteristics similar to the wrist joint. The first to be trained are the index MCP joint and the little finger MCP joint. The reason for the priority of these two finger is that both have exclusive tendons that grant them a greater independence (and consequently less coupling) from the other fingers and from each other. From here the other fingers can be trained in a similar fashion, leaving for last the thumb, which is the most independent finger. At the end one will have a sub-optimal controller, Fig. 3.30, but with the guarantee that the initial weights for the merged neural network will result in a stable system.

The final tracking error for the hand, after the training of the merged neural

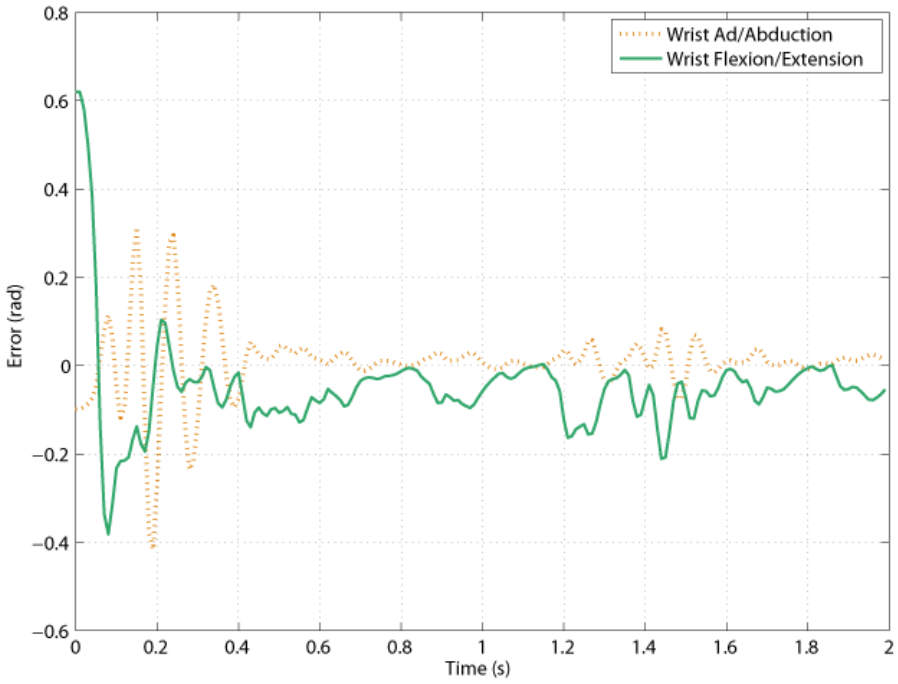


Figure 3.29: Tracking for the wrist joint for a grip movement.

network controller, the reference signal, the output and the control actions can be seen in Fig. 3.30, Fig. 3.31, Fig. 3.32, and Fig. 3.33, respectively[13].

A more detailed of the outputs of each joint and the control action for each muscle can be seen in Appendix A.

[13]Please note that these figures are only provided to give a general sense of the system, for a more detailed version, please refer to Appendix A

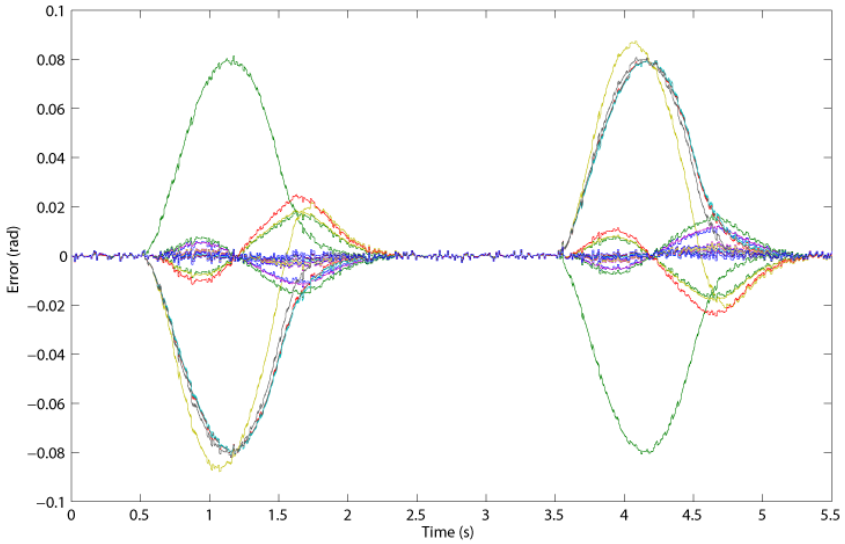


Figure 3.30: Final tracking errors of the hand joints.

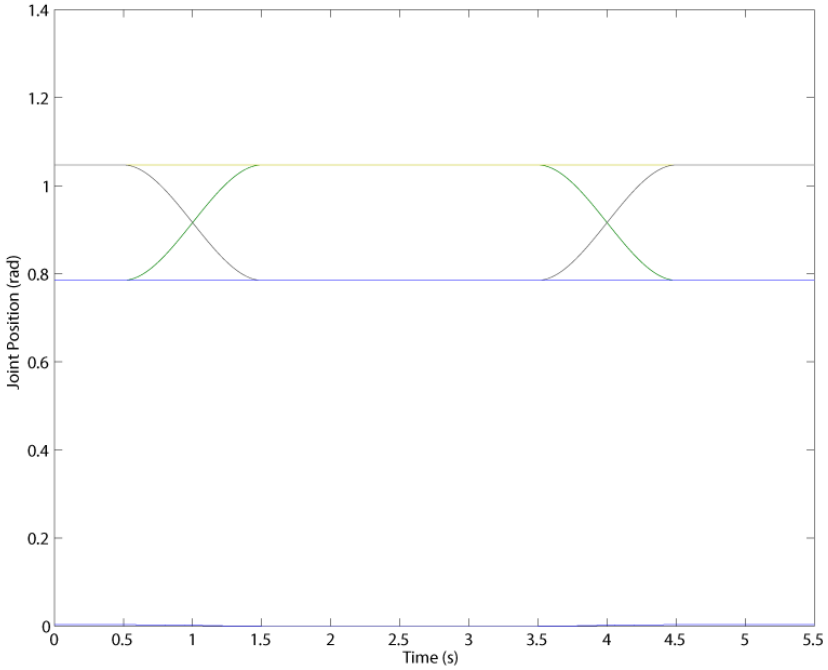


Figure 3.31: Reference signals for the grip movement.

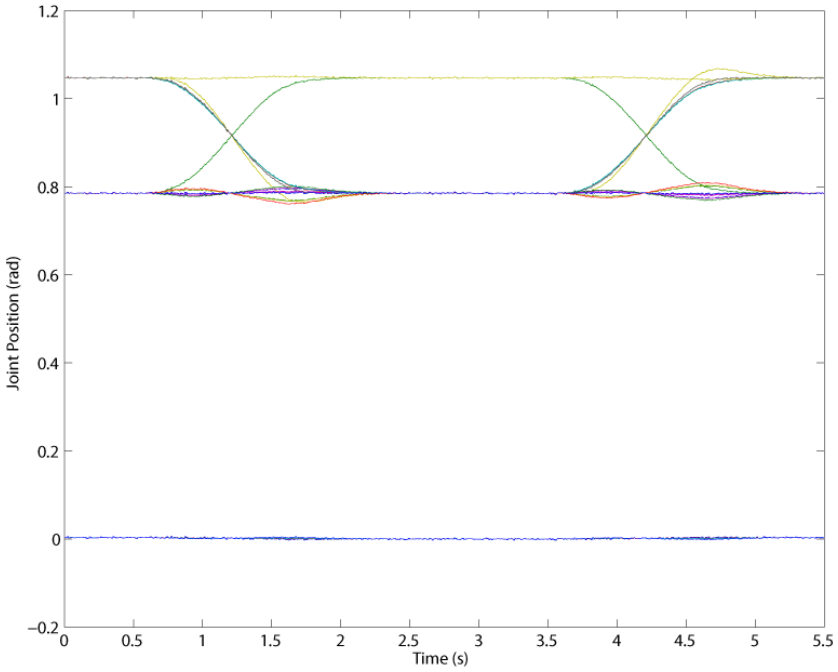


Figure 3.32: Joint positions for the grip movement.

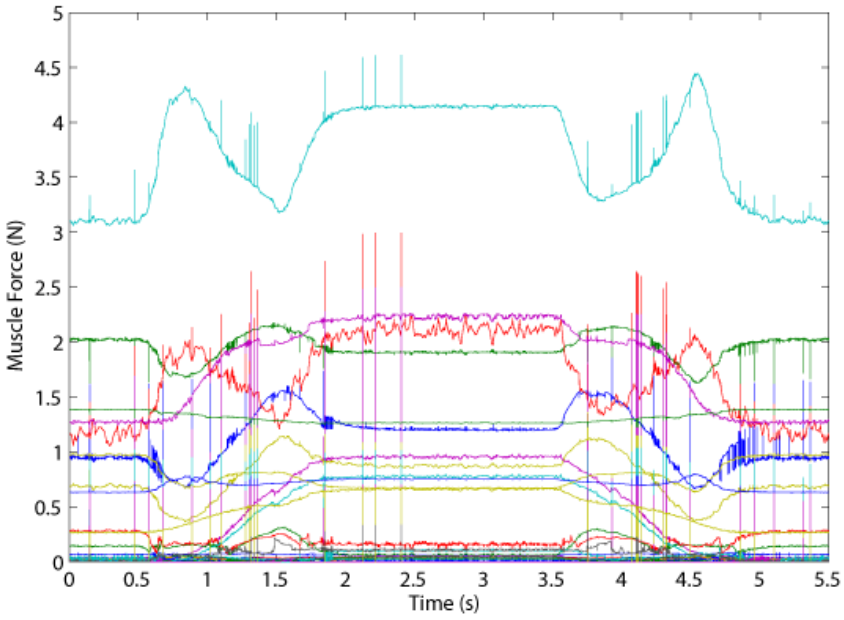


Figure 3.33: Control actions (muscle forces) for the grip movement

From figures 3.32 and 3.33, it can be verified that the neural network controller generates a significant noise. There are two sources for this noise: the first one are the inherent errors in the training of a neural network, the neural networks provides only an approximation and even if it is extensively trained, there will always be points where the network is sub optimal. The second source is the quality of the training signals, especially the Jacobian of the plant provided by the neural network model of the plant. The further the supplied Jacobian is from the real one, the less accurate the training of the neural network controller will be. One solution for this problem is presented by [24], where a Kalman filter is introduced into the control loop. Another observation that can be made from Fig. 3.33 is that the maximum required forces are well inside the projected maximum supplied force by the artificial muscles (as presented in section 4.5.

When an external force (commonly represented by the hand grabbing and pulling or pushing a weight) is introduced into the system will respond accordingly, as it will be seen in the next test.

The following test is divided into three parts: in the first the hand extends to a fully open configuration, then the hand closes to a grip position. During the grip position, an external vertical force of 50N is applied to the proximal phalanxes of the index and middle finger[14] and the extension/flexion joint of the wrist is flexed to the maximum flexion (which, for simplicity, is $\pi/2$ radians). In the last part, the grip is lessened until the external force is removed and the hand returns to the fully extended position. The final tracking error, reference, output, and control action are depicted in Fig. 3.34 to Fig.3.37, respectivelly.

[14]Note that the force is only applied to two fingers for simplicity sake, because if the force is applied to more finger it would result in a over-constrained system and, consequently, harder to solve with no increased value to the test.

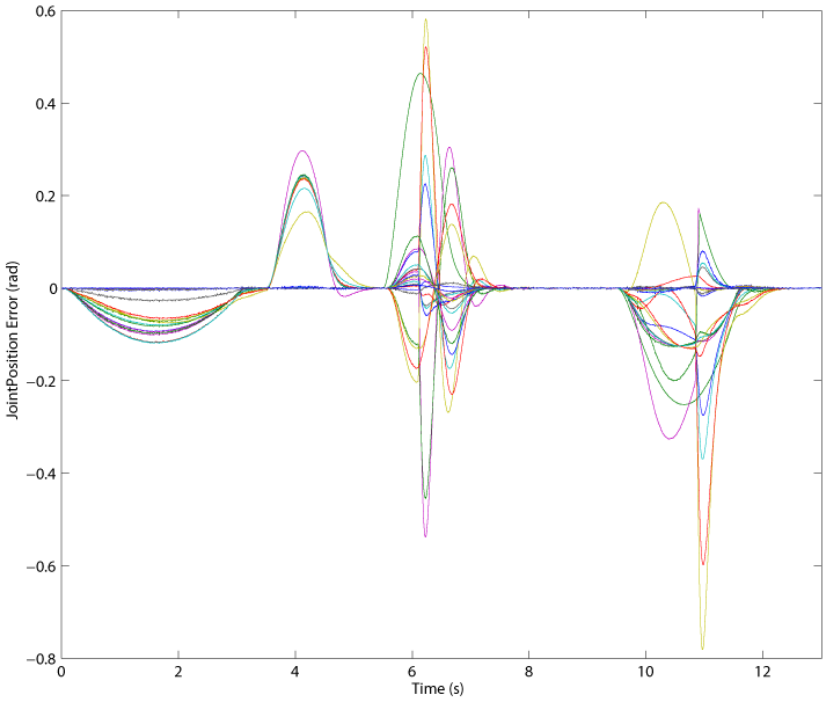


Figure 3.34: Final tracking errors of the hand joints.

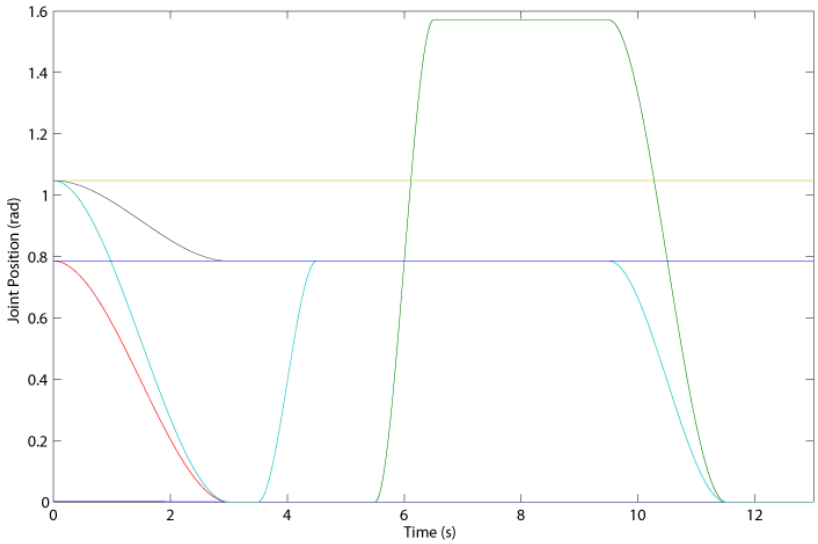


Figure 3.35: Reference signals for the grip movement with an external force.

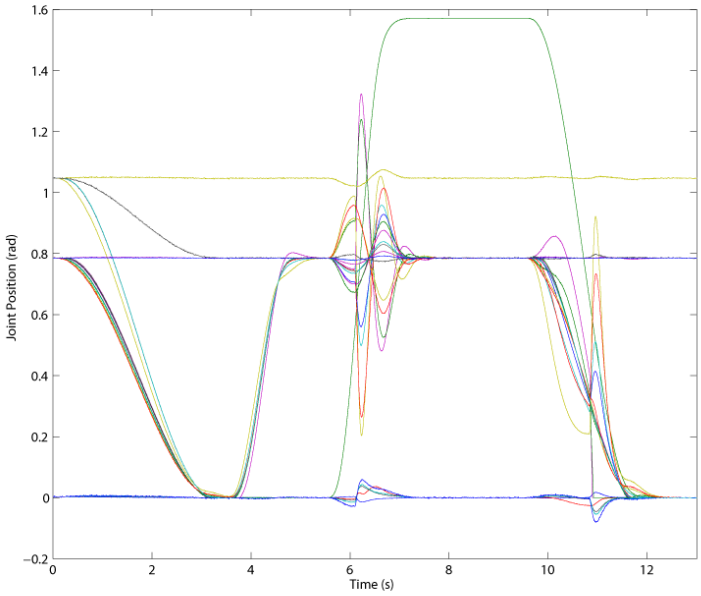


Figure 3.36: Joint positions for the grip movement with an external force.

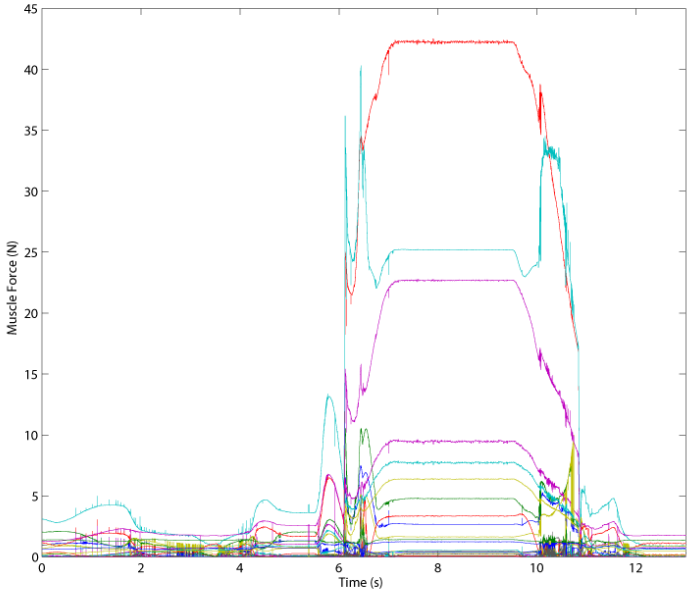


Figure 3.37: Control actions (muscle forces) for the grip movement with an external force.

Similarly to the previous test, a more detailed version of the results is presented in Appendix A.

From Fig. 3.37, one can see when the external force is applied to and removed from the fingers, because of the abrupt changes in the muscle forces. The external force can also be seen in the output and error (figures 3.36 and 3.34), where it affects not only the index and middle fingers, but whole hand. This can be explained by the heavy coupling of the muscles.

Again, there is a considerable amount of noise in the control action and output signal (though much lesser in the latter, because it is mostly absorbed by the dynamics of the hand). Nevertheless, the system remained stable, even with the unpredicted external force.

The last movement simulated to test the model is the cup movement. The cup movement is done by positioning the fingers as if they were to grasp a sphere with the wrist fully extended. This movement, although simple, requires large amounts of force from the muscles, especially from the large flexor and extensors and from the thumb muscles. The test is performed similarly to the first one, where the hand starts from the equilibrium position and is moved to and form the cup position in 8 seconds. The final tracking error, reference, output, and control action are depicted in Fig. 3.38 to Fig.3.41, respectively.

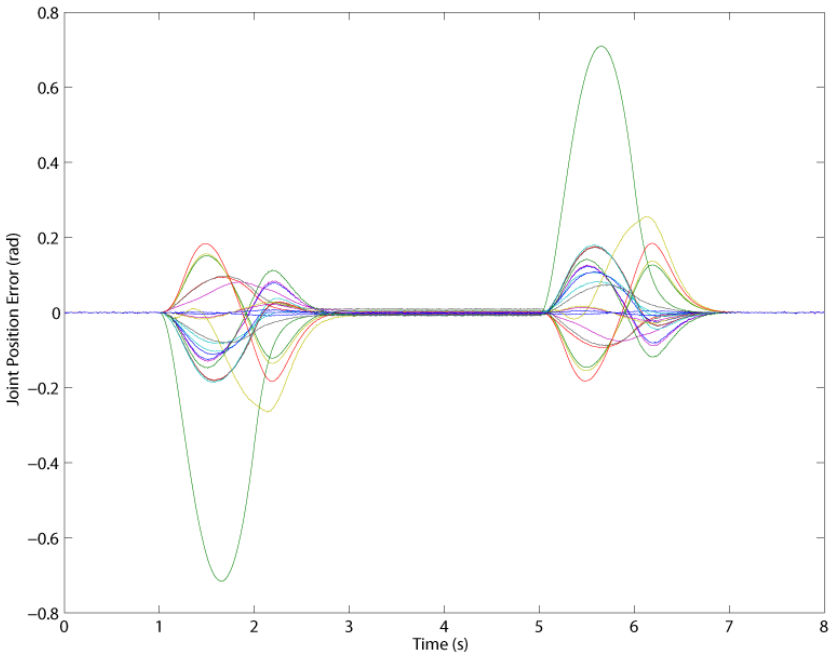


Figure 3.38: Final tracking errors of the hand joints.

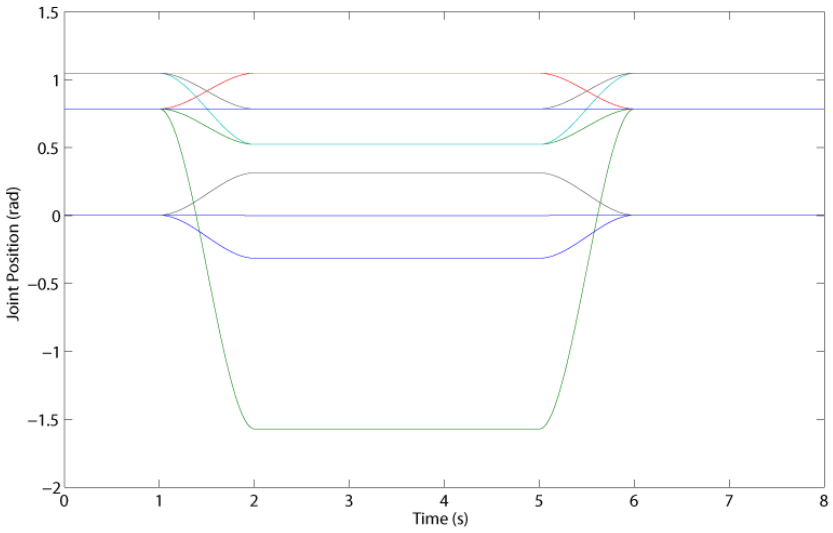


Figure 3.39: Reference signals for the cup movement.

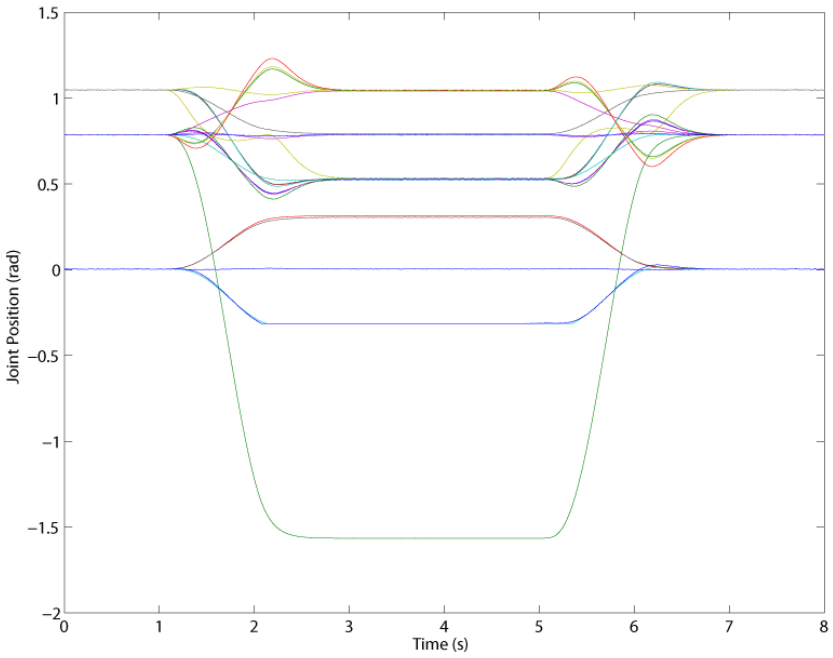


Figure 3.40: Joint positions for the cup movement.

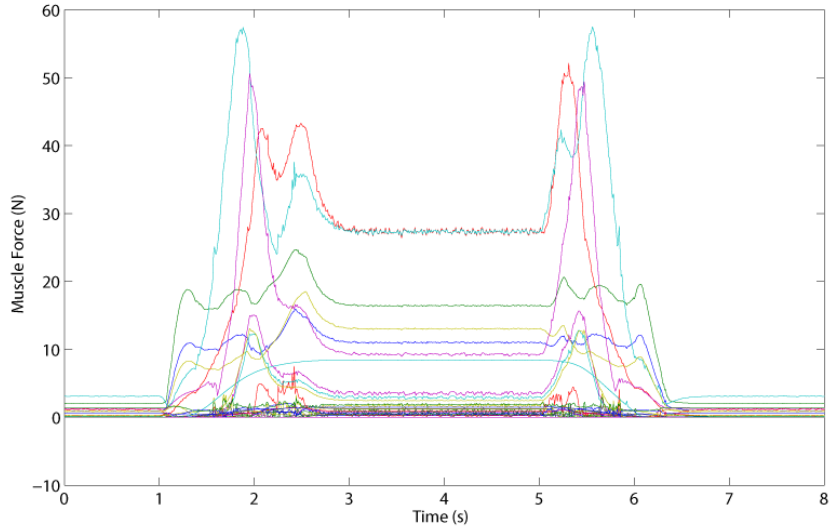


Figure 3.41: Control actions (muscle forces) for the cup movement.

Analyzing Fig. 3.38, it can be seen that the maximum error is larger than the other tests. This happens, mainly due to the fact that, to make this movement, most joints have to be fully extended or contracted. Pushing the joints to their limits brings the neural networks to the limits of the training region, which are normally the least trained points in the training region[15]. The solution to this problem is to create a training data that contains more points in the saturations. However, by doing so, the training data will have less points in the inner areas of the working region and, consequently, poorer performances in those areas.

[15]Usually to avoid this problem, the intended working region of the system is used for the test signal and a symmetrically larger region is used for the training signal, but because of the saturations, the training signal has to be contained in the working region, which in turn leads to poorer performances at the limits.

Chapter 4

Design and Development of an Artificial Muscle

4.1 Electroactive Polymers

Currently, most of the prostheses and anthropomorphic manipulators are basically mechanical grippers with three to five finger-like end-effectors that are driven by servo motors, applied directly to the joints or using an artificial tendon system. Some of the problems that arise from the use of servo motors include the noise and the failure probability inherent to the moving parts of the servo motors. Another problem is that, in order to be used in a prosthesis, the servo motors must be as small as possible while retaining a high torque, which increases somewhat the total cost of prosthesis.

The discovery of the electroactive polymers (EAPs) brought a new type of solid state actuators/sensors that can replicate the displacements and torques of the natural muscles. Electroactive polymers (EAPs) are a class of polymers that exhibit piezoelectricity, which is the ability to change shape and/or volume with the application of an external electrical field and vice versa [45]. There are several materials that also have this property, most noticeably the quartz and the Lead Zirconate Titanate (also known as PZT), which are Electroactive Ceramics (EAC).

Electroactive polymers can be divided in two main categories: Ionic EAPs and Electric EAPs [46]. Ionic EAPs require low voltages (usually between 1 and 5V) but because the electricity is transported by ions, this type of EAPs needs to be constantly submerged in a solution of ions. Additionally, it is very difficult to hold a displacement under a DC field. On the other hand, electronic EAPs require high

electrical fields, around 150V/μm, but they can sustain a displacement in a DC field and they need a small apparatus to be fully functional.

Other field of study that is intrinsically related to the design of an EAP-based actuator is the study of compliant electrodes. In order to function, the electrodes have to be coated on each side of the EAP, however they can only extend until a certain point, after which cracks start appearing and, consequently, the circuit is opened. Compliant electrodes have to be carefully designed, because they cannot be too tough or they will hamper the deformation of the polymer. Also, their electrical resistance must be as low as possible.

4.2 Dielectric Elastomers

Dielectric elastomers are rubbery electronic EAP that under an applied electrical field undergo large mechanical strains that can reach 350%. A polymer to be considered a dielectric elastomer must have a low elastic stiffness and high dielectric constants, because the electromechanical actuation is due to the Maxwell Stress.

4.2.1 Maxwell Stress

The Maxwell Stress is the pressure that two electrically charged masses (and consequentially any medium between them) feel because of the electromagnetic force generated by their charges. The Maxwell stress is given by Eq. 4.1, [46, 47]:

$$T_{ij}^{M} = -\left(E_i \epsilon_{jm} E_m - \frac{1}{2}\delta_{ij}\epsilon_{kl}E_l E_k\right) - \left(B_i \frac{1}{\mu_{jm}} B_m - \frac{1}{2}\delta_{ij}\frac{1}{\mu_{kl}}B_l B_k\right) \tag{4.1}$$

where T^M is the Maxwell Stress tensor, E is the applied electrical field, ϵ is the dielectric constant (also known as electrical constant), B is the magnetic field, μ is the material permeability and δ is the Kronecker delta.

Since the dielectric elastomer acts as parallel plate capacitor under direct current, there is no magnetic field and the electric field only has one direction (which is always perpendicular to the electrodes). Because of that, the only non-zero component of Eq. 4.1 is given by Eq. 4.2. Finally, combining Eq. 4.2 with Hooke's law, the

mechanical strain of the elastomer can be determined by Eq. 4.3:

$$T_{33}^{M} = -\frac{1}{2}\epsilon_{33}\epsilon_{0}E_{33}^{2} \tag{4.2}$$

$$S_{ij} = s_{ijkl}\left(T_{kl} + T_{kl}^{M}\right) \tag{4.3}$$

where S is the strain tensor, s is the stiffness tensor and T is the applied stress tensor.

4.2.2 Electromechanical properties of dielectic elastomers

To obtain high mechanical strains the elastomer must have a low elasticity modulus and a high dielectric constant. Other important performance parameters are the energy density, response time, viscoelastic losses and ease of fabrication,[46, 47, 48, 49, 50].

Dielectric elastomers can be characterized according to two important mechanical properties: the ability to sustain large elastic strains and the ability to maintain volume during the deformation. When compared to other electroactive materials such as piezoelectric and magnetostrictive ceramics, dielectric elastomers are capable of large strains with lower response times and high electromechanical efficiency [46, 48].

From all the available dielectric elastomers, Silicone-based poly(dimethyl siloxane) and the 3M 4910 VHB commercial acrylic have the largest energy density and maximum mechanical strain (more than 100% plane strain). Other good materials for dielectric elastomers can be seen on Table 4.1 [49, 50, 51].

Table 4.1: Properties of some dielectric elastomers

Polymer	Elastic energy density (MJ/m^3)	Maximum electromechanical strain (S_{ee})	Young's Modulus (MPa)	Applied electric field (MV/m)	Dielectric constant (1kHz)	Coupling efficiency
Silicone Nusil CF19-2186	0.15	-32%	1	235	2.8	54%
Silicone Dow Corning HS3	0.038	-41%	0.125	72	2.8	65%
Polyurethane Deerfield PT6100S	0.2	-11%	17	160	7	21%
Silicone Dow Corning Sylgard 186	0.1	-32%	0.7	144	2.8	54%
Fluorosilicone Dow Corning 730	0.051	-28%	0.5	80	6.9	48%

Since the required electrical fields are very high, the polymeric material can suffer from electrical breakdown. Electrical breakdown occurs when the applied electrical field is high enough to break the van der Waals connections that keep the linkage between the polymeric chains or, even, the connection between the chain atoms, resulting in the creation of free ions and making the polymer conductive [52]. One solution for this problem is pre-straining the material in order to decrease the initial thickness. The breakdown electrical field increases as the thickness decreases because the van der Waals connections become stronger as the polymeric chains come closer to each other, Fig.4.1.

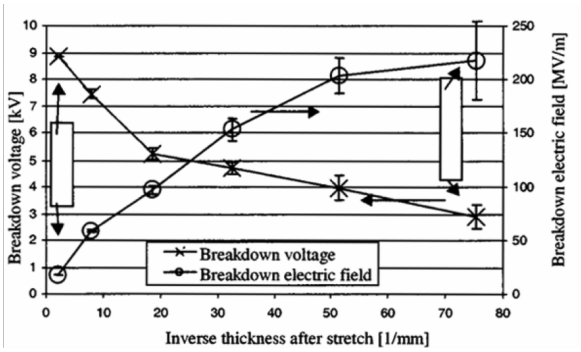


Figure 4.1: Electrical breakdown of an isotropically pre-strained 3M 4910 VHB acrylic tape [52].

Since the applied voltage is proportional to the distance between the electrodes, as the material is pre-strained the required electrical potential to obtain a certain strain value decreases. The type of pre-strain (linear or circular) can also affect the response of the dielectric elastomer as it can be seen on Table 4.2 [48].

One of most important characteristics of dielectric EAP materials is that they are highly viscoelastic by nature. Viscoelastic materials are a type of material that exhibit a nonlinear elastic region, and in most cases, with no evident frontier between the elastic region and the plastic region (or failure in a fragile material like the 3M VHB4905/4910). Another characteristic of viscoelastic materials is the hysteresis exhibited in compression/relaxation cycles (or strain/compression cycles).

Response time is also very important when designing an actuator. The response time is measured by determining when the maximum force is attained for a certain stimulus [53]. The results for two tests, one with the 3M VHB 4910 acrylic elastomer and the other with the CF19-2186 silicone, Fig. 4.2, show that the response agrees

Table 4.2: Circular and linear pre-strain test results for different types of elastomers

Polymer	Pre-strain (x,y)	Actuated relative thickness strain	Actuated relative area strain	Applied electric field (MV/m)	Effective compressive stress (MPa)	Estimated energy density (MJ/m^3)
			Circular Strain			
HS3	(68%,68%)	-48%	93%	110	0.3	0.098
	(14%,14%)	-41%	69%	72	0.13	0.034
CF19-2186	(45%,45%)	-39%	64%	350	3	0.75
	(15%,15%)	-25%	33%	160	0.6	0.091
VHB 4910	(300%,300%)	-61%	158%	412	7.2	3.4
	(15%,15%)	-29%	40%	55	0.13	0.022
			Linear strain			
HS3	(280%,0)	-54%	117%	128	0.4	0.16
CF19-2186	(100%,0)	-39%	63%	181	0.8	0.2
VHB 4910	(540%,75%)	-68%	215%	239	2.4	136

with a first-order dynamic system with a time constant of 20ms and 50ms, for the VHB 4910 and CF19-2186, respectively [53, 54].

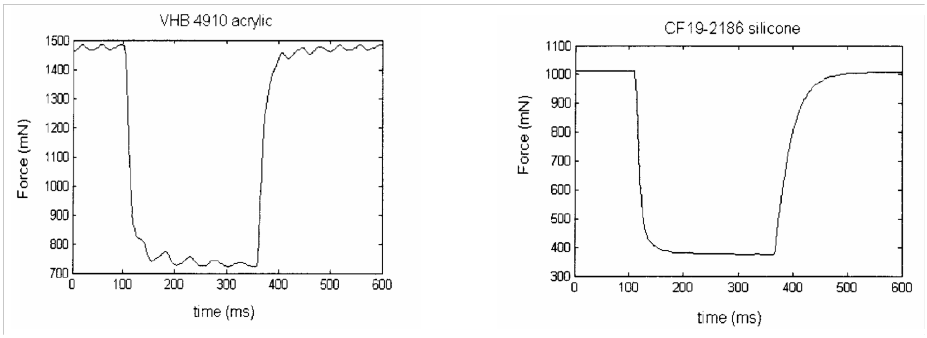


Figure 4.2: Maximal isomeric contraction for two dielectric elastomers [53].

4.3 Actuator Configurations

Dielectric elastomer actuators are available in a variety of configurations: stacks, diaphragm/membranes, rolls or helicoids. One of the more simple forms is the stack, which was derived from the piezoelectric technology [50, 55]. In stack formation, the actuator is made by stacking layers of elastomer alternated with layers of electrodes, Fig. 4.3 [46, 55].

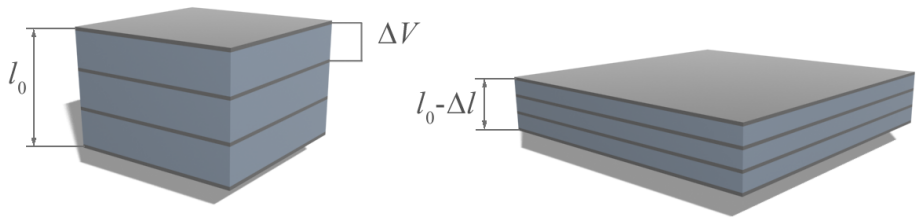


Figure 4.3: Stack dielectric elastomer actuator.

Actuators in this configuration can be used for linear contraction actuation (if the movement axis is perpendicular to the electrodes), linear expansion actuation (if the movement axis is coplanar to the electrodes and the in-plane perpendicular axis is constrained) or area expansion actuation (if the movement axes are coplanar to the electrodes). This type of actuator can be used in robotic applications such as manipulators [46].

The diaphragm or membrane type is a single stack version of the stack area expansion actuator with ring-like electrodes. The objective of this actuator type is to supply a vibration at a controlled frequency rather than an expansion movement, Fig. 4.4 and Fig. 4.5 [46, 56, 57].

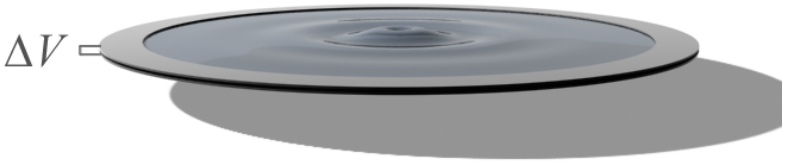


Figure 4.4: Diaphragm/membrane dielectric actuator.

Roll type dielectric elastomer actuators are linear actuators that consist in two alternating layers of elastomer and electrodes that rolled to form a cylindrical shaped actuator, Fig. 4.6 [49, 56, 57]. Experiments done with this configuration using the 3M VBH 4910 acrylic elastomer and carbon grease electrodes can be seen in Fig. 4.7 [49].

The last type of dielectric elastomer actuator is the helical. Helical actuators are made of two long layers of elastomer and electrodes alternately stacked that are rolled to form a helix, Fig. 4.8 [50]. Carpi et al. [50], experimented with this configuration,

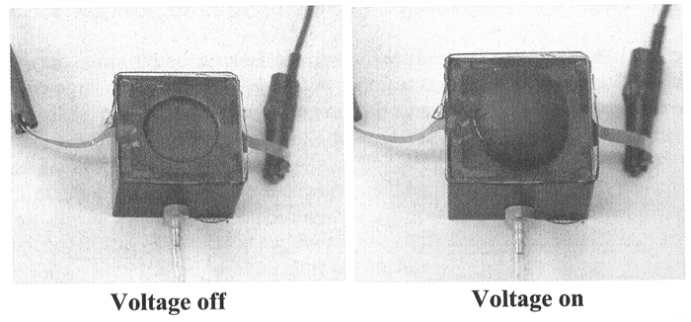


Figure 4.5: An acrylic diaphragm undergoing actuation under a DC field [46].

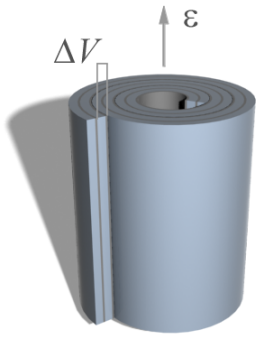


Figure 4.6: Roll type dielectric elastomer actuator.

using two elastomers: TC-5005 A/B-C silicone rubber from BJB Enterprises Inc. and a solution of the former with PZT powder, Fig. 4.9.

Figure 4.7: VBH 4910 roll dielectric elastomer [49].

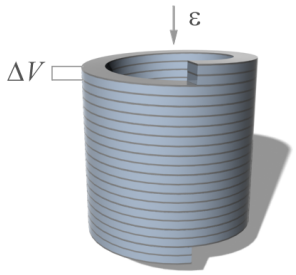


Figure 4.8: Helical dielectric elastomer actuator.

Figure 4.9: Helical dielectric elastomer [50].

4.4 Compliant Electrodes

Compliant electrodes are a very important aspect of electroactive polymers, especially for dielectric elastomers, because they must have a lower resistance as possible and at the same time they must be elastic enough to accommodate the high stains without restraining them and without cracking [48]. The compliant electrodes currently used can be categorized into three categories: grease/rubber electrodes, conductive polymers and metal electrodes.

4.4.1 Grease and Rubber Electrodes

The difference between grease electrodes and rubber electrodes is how they are prepared and applied. Grease electrodes are of powered graphite or metal particles embedded in a polymeric matrix. The preparation method consists in suspending the powered particles in a solvent (for the polymeric media) along with the polymer matrix. The mixture is them agitated using ultrasounds in order to promote the dispersion of the powered particles on the polymeric matrix. After the desired dispersion is attained, the mixture is left still to allow the solvent to evaporate. Finally, the evaporation process is interrupted when the grease has the desired viscosity. The grease is applied on the elastomer with a brush [58, 59]. Rubber electrodes have a similar preparation method, but, unlike grease electrodes, the rubber electrodes are not left for the solvent to evaporate, instead they are directly applied to the EAP using an airbrush [58, 59].

The advantage of grease and rubber electrodes is the elasticity of the polymeric matrix being similar to the elastomer, but, because the matrix is not a conductor, the conductivity of the electrode is proportional to the percentage of particles present in the matrix and the higher the concentration of powder is lower is the elasticity of the electrode. Normal percentage of powder is 5%wt to 6%wt, but if higher concentrations are used cracks start to appear on the electrode as the remaining solvent evaporates. The surface of the electrodes viewed with an optical microscope can be seen in Fig. 4.10 [48, 58, 59].

As the EAP undergoes mechanical strain the electrode the polymeric matrix of the electrode will extend to conform to the elastomer, but since the conductive media is discrete, more cracks will appear, thus increasing the resistance of the electrode and, consequently, increasing the losses, Fig. 4.11 [48].

The resistivity of the electrodes is measured using two gold probes with different

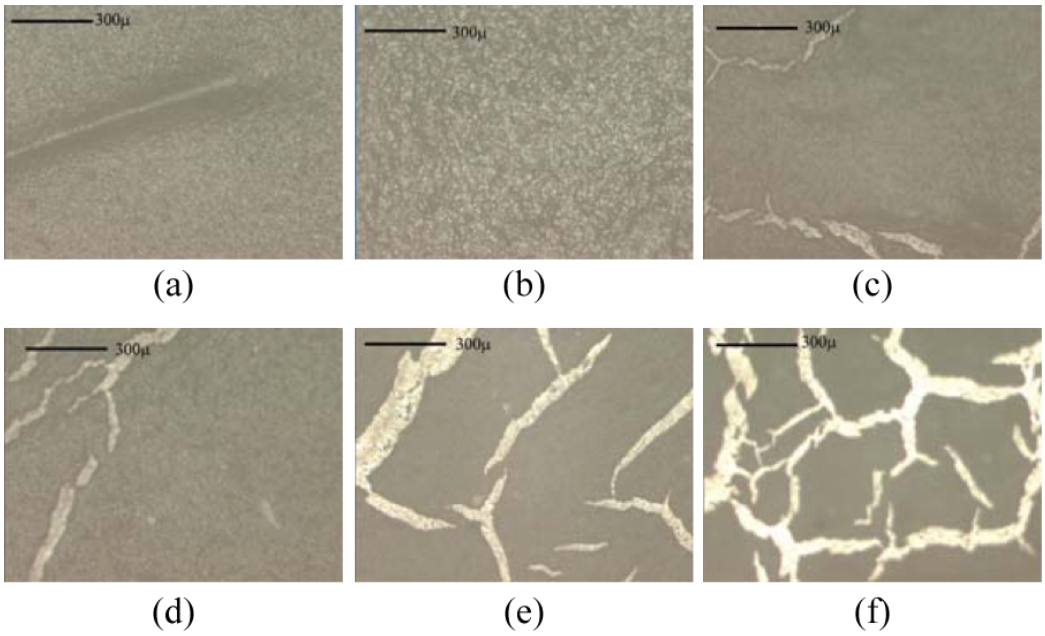


Figure 4.10: Sylgard 184- Fluid 200(r) FL 50 CST grease electrodes: (a) 5% graphite, (b) 6% graphite, (c) 7% graphite, (d) 8% graphite, (e) 9% graphite, (f) 10%graphite.

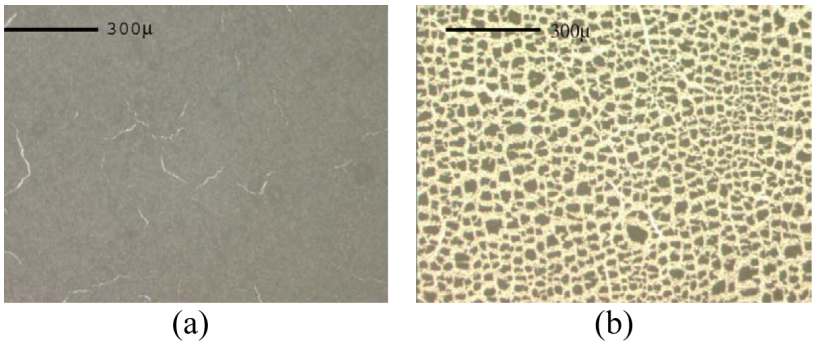


Figure 4.11: Sylgard 184 10% rubber electrode: (a) 0% stain, (b) 100% strain.

distances between them. As it can be seen on Fig. 4.12, the resistivity greatly increases as the actuator extends, mostly due to the formation of cracks.

4.4.2 Conductive Polymer Electrodes

Conductive polymers can, under certain conditions, have high conductivities values depending on the type and amount of doping. Polypyrole, a conductive polymer that

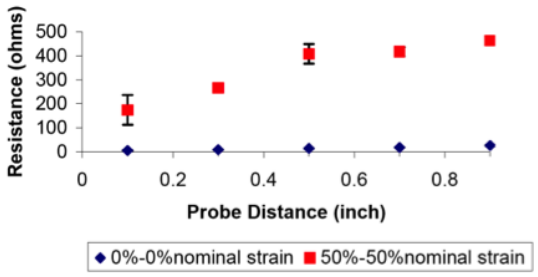


Figure 4.12: Resistance values for Sylgrad 184 10% graphite rubber electrode under 0 strain and 50% nominal strain.

can be used as an electrode and is easily polymerized using ferric chloride (FeCl3) in aqueous medium in the presence of dopants (a series of substance that act as catalyst for the conductive form of the conductive polymers) like sodium antraquinone-2-sufonic or 5-sulfoslicylic dehydrate. If the EAP elastomer is in contact with an aqueous solution of an oxidated polymerizable pyrrole compound, the polypyrrole will deposit on the elastomer surface, forming, thus, the electrodes [48, 60, 61].

When compared to other electrodes, like gold plated electrodes, conductive polymers have a more coherent interface with the elastomer and improve acoustic and optical transparency. They only lag behind gold when resistances are compared [60]. The compliance of the polypyrrole can be increased even more by wrinkling the electrode over the surface of the elastomer, because the polypyrrole has a lower elastic that most polymers [61]. The comparison between the performance of an unwrinkled electrode (undrawn) and various electrodes with different percentages of total area of the electrode relatively to the total elastomer surface area, Fig. 4.13 [61].

4.4.3 Metal Electrodes

Highly conductive metals like, gold, silver and aluminums can be used as electrodes for electroactive polymers. These materials have the main advantage of having low resistivity and, therefore, low electrical losses, but they impose mechanical constraints on the soft polymers reducing the actuation strain [48]. To decrease the mechanical constraint imposed by the high stiffness materials and because if the electrodes become too thin they crack easily, the metal electrodes can the applied in a pattern across the surface that works like a soft spring, Fig. 4.14. In order to disperse the

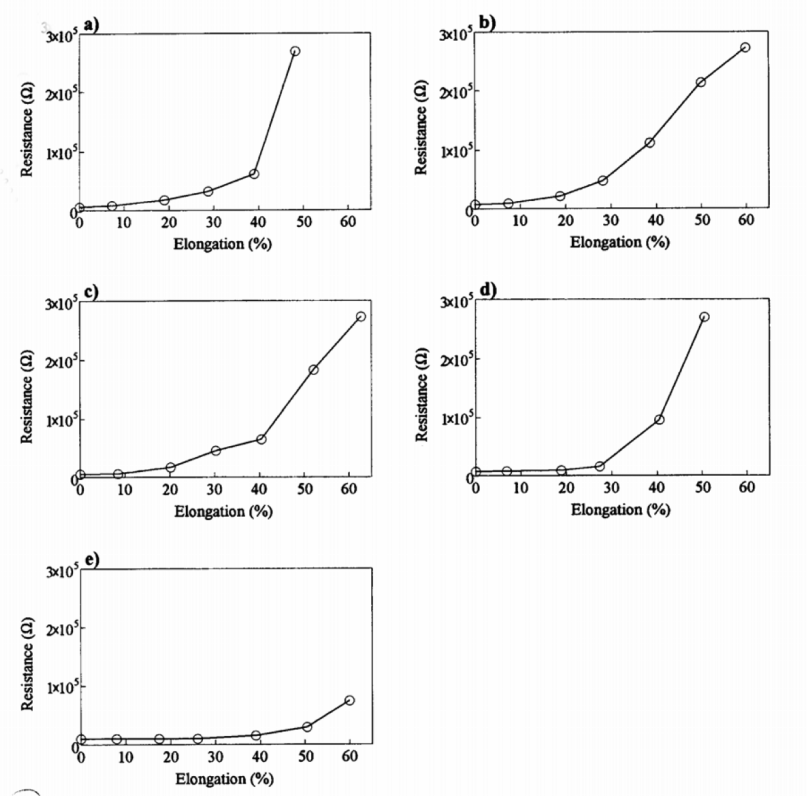


Figure 4.13: Resistance of polypyrrole electrode under elongation. The electrodes were prepared: a) undrawn, b) 9.8% drawn, c) 17.9% drawn, d) 28.2% drawn, e) 35% drawn.

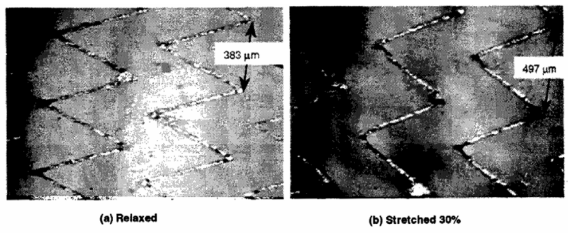


Figure 4.14: Zig-zag patterned metal electrodes.

electricity towards the elastomer surface, a low conductivity and elastic polymer can be used.

Another strategy to reduce the hampering effect of the metal electrodes is to make the surfaces of the elastomer rugged so that when extending the metal electrodes suffer a bending deformation, which is less constricting than linear mechanical deformation, Fig. 4.15 and Fig. 4.16 [48].

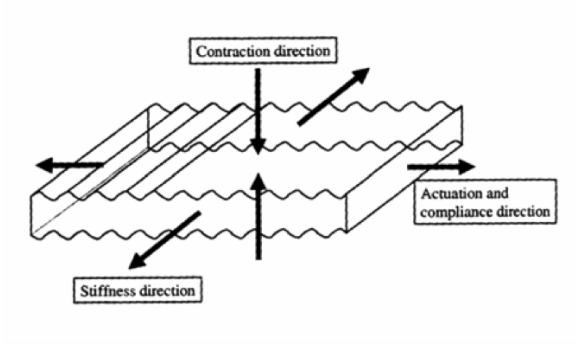


Figure 4.15: Rugged dielectric elastomer actuator schematics [48].

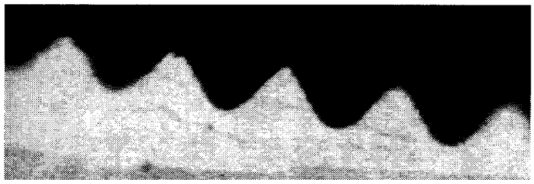


Figure 4.16: Optical microscope image of the cross-section of a metal electrode rugged actuator [48].

Although greatly increasing the actuation strain of the metal electrode actuators, up to 80%, the fabrication process needs complex and expensive technologies like photolithography and structures like the zig-zag patterned metal electrode can be fragile since their thickness can be as low as 5μm [48].

Liquid Metal Electrodes

A solution to the shortcomings of the metal electrodes is to use metals that can be easily deformed at low temperatures (room temperatures) while retaining high conductivity. Some of such metals are the naturally liquid metals: mercury, Hg; caesium, Cs; francium, Fr; gallium, Ga; and rubidium, Rb. These metals, because of their liquid state, can deform easily and do not impose any constraint on the elastomer actuation strain, while having high or very high conductivity. The greatest problem with liquid metals is that they are, with the sole exception of gallium, highly reactive and hazardous, e.g. Hg is extremely toxic, Fr is radioactive and Cs and Rb are highly reactive with water or any hydrated substance. The only safe liquid metal is gallium.

Pure gallium has a melting point, at sea level pressure, of 29.8°C (303.9K) and,

unlike Hg, it wets glass, which increases its adherence to surfaces. Because of its low toxicity, gallium can be used in medical application and it is currently used as a replacement for Hg in thermometers. Gallium also easily makes alloys with other metals giving it more properties. One alloy of special interest is Galinstan, a commercial gallium-indium-tin eutectic alloy, which has a melting temperature of -19°C and a boiling temperature of approximately 2200°C (2473.5K). The conductivities of Galinstan and other materials are shown in Table 4.3.

Table 4.3: Resistivity of various materials

Material	Resistivity (nΩ.m)
Gallinstan	435
Pure Gallium	14.74
Gold	22.12
Silver	25.87
Aluminum	26.51
Copper	16.77
Graphite	35000
Polypyrrole	10000 - 100000

The main disadvantage of liquid metal electrodes is that the fluid must be contained, otherwise the electrodes will spill.

4.5 Development of the Prosthesis Actuator

The development of the prosthesis actuator has to overcome several challenges imposed not only by the specificities of dielectric base actuators, but also the ones imposed by the highly nonlinear characteristics of the EAP, especially the 3M VHB4905, which will be used to build the actuator.

Using a commercially available material, like the VHB4905, has some advantages, however it also has disadvantages that cannot be ignored. Among the advantages are the availability of the material and that the fact the one does not have to manufacture the material or have the polymer and chemistry knowledge necessary for the process. Another advantage of using a commercial material is that the final cost of

the manufacturing process of the actuator can be more easier to calculate. The disadvantages have a deeper impact in the development of the actuator as whole. The most prominent disadvantage is that, since the chemical formula of the material is a trade secret, the properties of the material are harder to obtain, mainly because the primary function of the VHB4905 tape is to be an adhesive material and not an actuator component. Also, the form factor in which the tape is made is predetermined by the seller and, consequently will constrain the dimensions of the final actuator.

4.5.1 Test Apparatus

The tests performed to determine the mechanical properties and behavior of the VHB4905, were done using an Instron 5544 tensile strength test machine, Fig.4.17, with a charge cell of 2kN.

Figure 4.17: Tensile strength test machine used in the experiments.

To provide the high voltage required for the operation of the actuator, a 30W direct current (DC) high voltage converter from UltraVolt was used, Fig.4.18. This converter transforms a 24V input signal into a regulated, 6000V output signal.

4.5.2 The VHB4905 dielectric elastomer

The 3M VHB4905 tape is a silicon-based dielectric polymer. As said above, the chemical formula is unknown and the properties (mechanical and electrical) of the material supplied by the seller are (as specified) for reference purposes only and should not use for scientific calculations [62]. The main function of the tape is to work has a

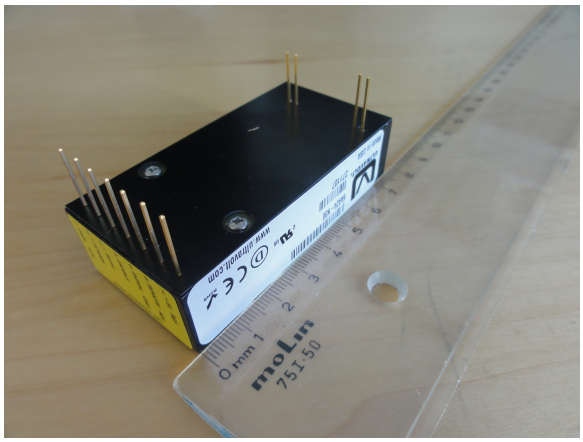


Figure 4.18: High voltage converter used to power the actuator.

highly general purpose adhesive material, having high adhesion to most metals, glass and a large selection of polymeric materials [62].

Mechanical Properties

The VHB4905 is a viscoelastic material and, consequently has a nonlinear behavior when subjected to any type of deformation.

Viscoelastic materials are defined as follows [63]:

> A material having [a viscoelastic] property is considered to combine the features of a so-called perfect elastic solid and a perfect fluid.

Which means that viscoelastic materials do not have a linear elastic region as regular elastic materials (such most metals and alloys) have. In elastic materials, the deformation depends linearly on the amount of stress the material is subjected to, thus obeying the Hooke's Law, Fig 4.19. However, viscoelastic materials not only nonlinearly depend on the stress, but also on the rate of its change, with energy losses that result in an hystereris loop, Fig 4.20.

The hysteresis of a viscoelastic material can have more complex behaviors than the more common elliptic path (where the relaxation path is symmetric to the extension path). A good example is the VHB4905 material which displays a "boomerang" shaped hysteresis when subjected to compression/relaxation test, as it can be seen in Fig 4.21:

The sharp corners on the hysteresis are originated by the triangular signal used to generate the deformation of the material (ideally it should one should use a sine,

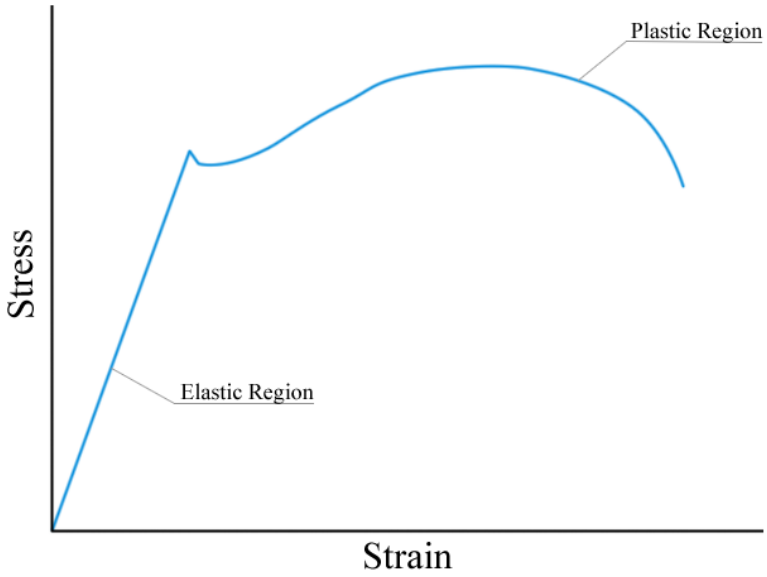


Figure 4.19: Regular linear elastic material behavior.

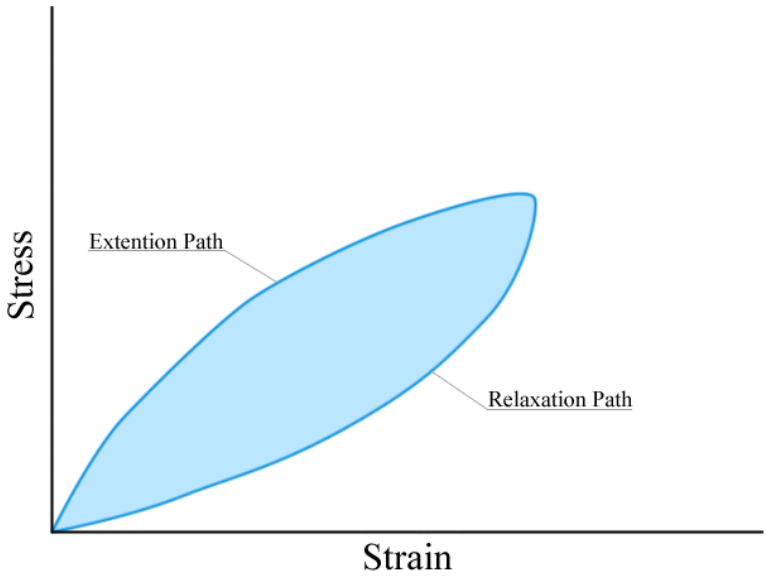


Figure 4.20: Viscoelastic material behavior, with the dissipated energy shaded.

but, due to equipment limitations, a triangular signal had to be used). The non-elliptical shape can be result of nonlinearities on the governing differential equation itself. Another particularity of the VHB4905 is that, as the deformation frequency in-

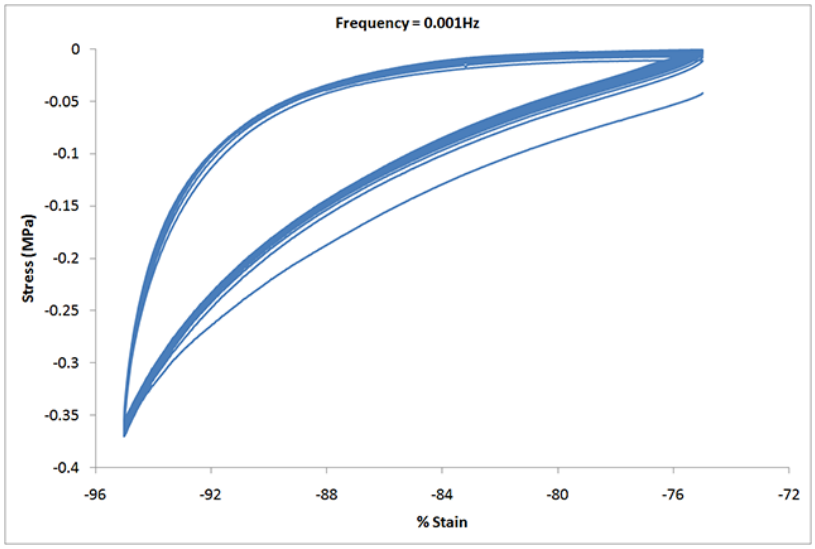


Figure 4.21: Viscoelastic behavior of the VHB4905 for a cyclic deformation with a frequency of 0.001Hz.

creases, instead of just changing the amplitude or rotating, the shape of the hysteresis gradually change into an even more complex form, as it can be seen in Fig 4.22

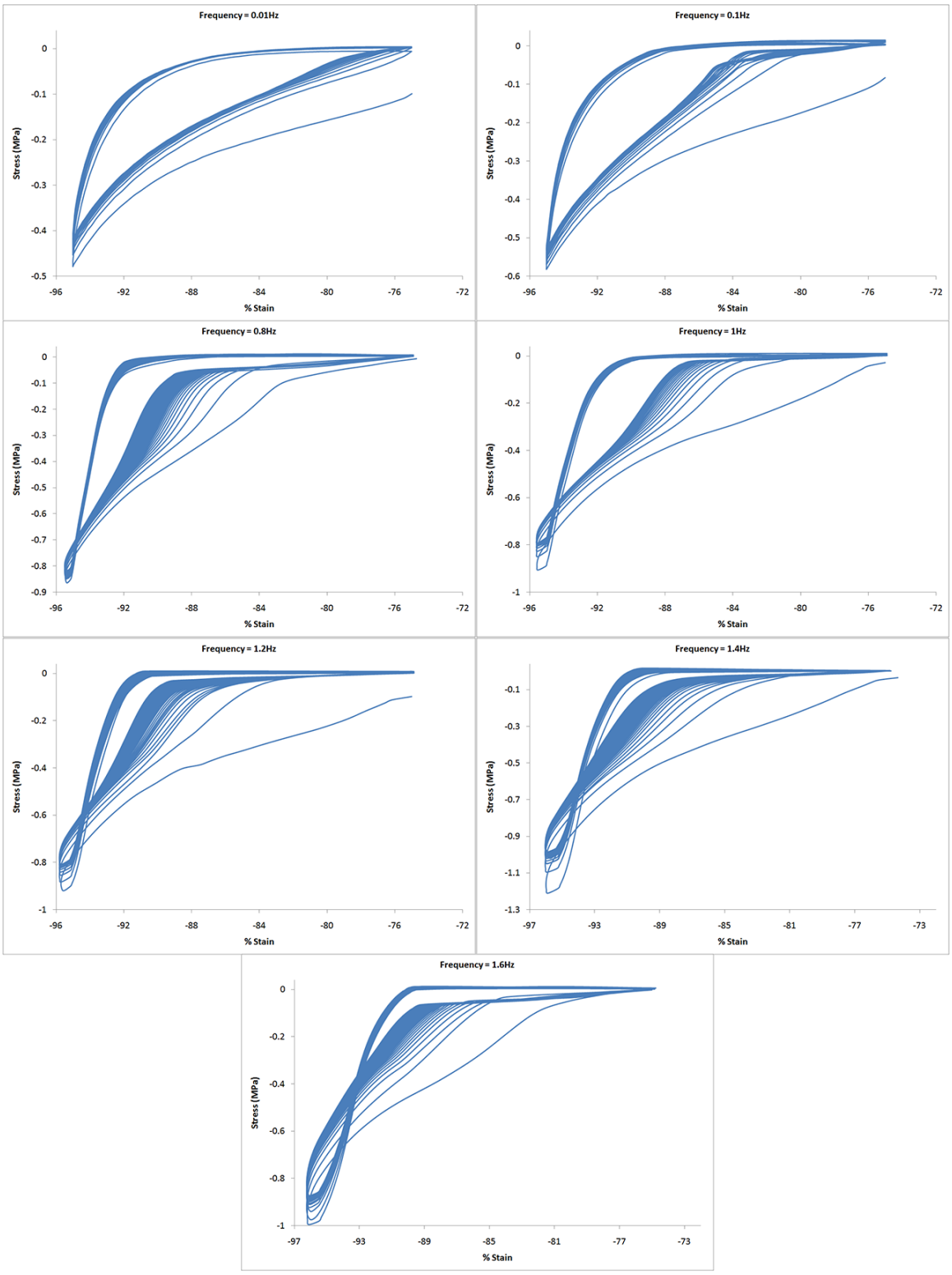


Figure 4.22: Viscoelastic behavior of the VHB4905 for a cyclic deformation with frequencies ranging from 0.01Hz to 1.6Hz.

From Fig. 4.22, it can be seen that as the deformation frequency increases, two distinct deformation zones start to appear: a zone, at the beginning, with a high variation of the deformation for a low variation of stress, and a zone with a low variation of the deformation with high variation of stress at the end. This, almost, abrupt change in the behavior has to taken in consideration when designing the actuator, especially for high speed movements.

Apart from analyzing the dynamic behavior of the elastomer, one also has to study the stationary behavior. Because the elastomer, to work has an actuator, has to be pre-compressed, one must know the amount of force (or stress) required to reach a stable compression rate. Also, as it can be seen in Eq. 4.1 or Eq. 4.2, the input of the mechanical part of the system is a stress rather than a strain. Most tensile test machines (including the one used in this work) are made to measure the stress when the material is subjected to a certain amount of deformation, meaning that the independent variable is the strain instead of the stress. In regular elastic materials, because of the linear relation, this problem can be readily overcome by solving a first order equation, however for viscoelastic materials the same cannot be done.

To obtain a strain vs. stress curve for the VHB4905, the material had to be compressed by steps of constant applied force, Fig. 4.23, (the machine can maintain a constant force by varying the deformation accordingly) and a point is taken when the deformation stabilizes (the response is considered stable when the variation of the deformation is less than 1%). The resulting curves, obtained from force domain of [-35 0] (N), can be seen in Fig.4.24.

The next step is to interpolate the data points for the true strains and stresses and obtain an equation to be used with Eq 4.2 to obtain an estimate of the final rate of compression of the actuator for a given voltage level. The best fit equation (with an $R^2 = 0.9968$) was achieved by an exponential interpolation function, Eq. 4.4, plotted in Fig. 4.25.

$$\epsilon(\sigma) = \epsilon_m + \left(A \exp\left(B \frac{\sigma - \sigma_m}{\sigma_M - \sigma_m} \right) + C \exp\left(D \frac{\sigma - \sigma_m}{\sigma_M - \sigma_m} \right) \right) \tag{4.4}$$

Where ϵ and σ are the true strain and stress, respectively, and ϵ_m, ϵ_M, σ_m and σ_M are, respectively, the minimum and maximum values of the true strain and true stress. The coefficients A, B, C and D are the interpolation coefficients and their values are presented in Table 4.4.

The numerical values resulting from these experiments can be see in Appendix B

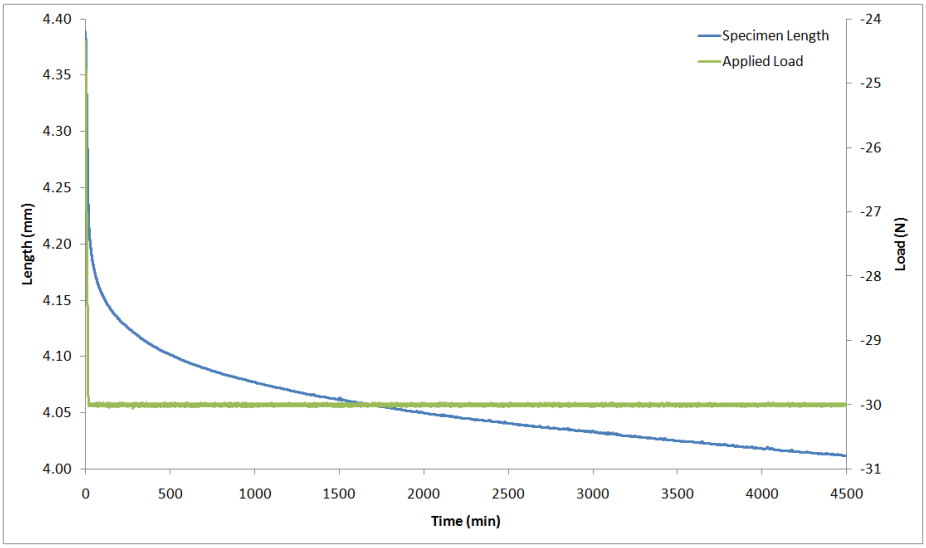


Figure 4.23: One step (-30N) of the series of test to find the stationary strain vs. stress behavior.

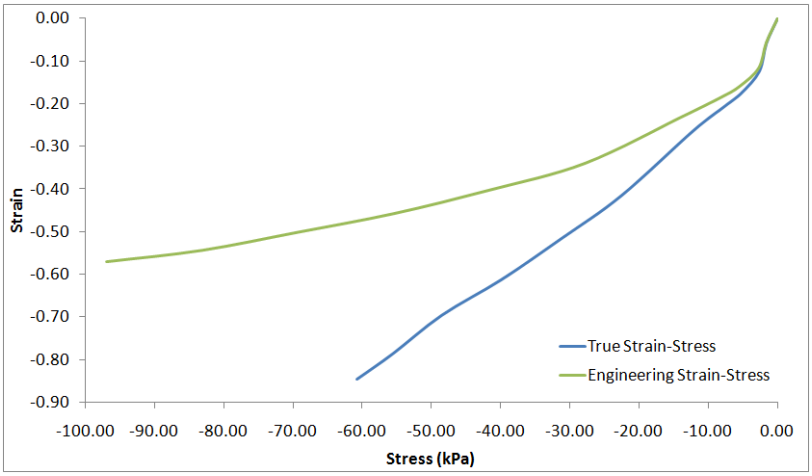


Figure 4.24: Stationary strain vs. stress (true and engineering) plot.

Electrical Properties

The electrical property that is of interest to the development of the actuator is the ability of the VHB4905 to act as a dielectric media, for which the dielectric constant, or stationary permittivity, is of paramount importance.

The dielectric constant is a property the material that relates the amount of

Table 4.4: Fitness function coefficients.

Coefficient	Value
A	5.5918×10^{-1}
B	6.0731×10^{-1}
C	5.1209×10^{-17}
D	1.7632×10^{1}

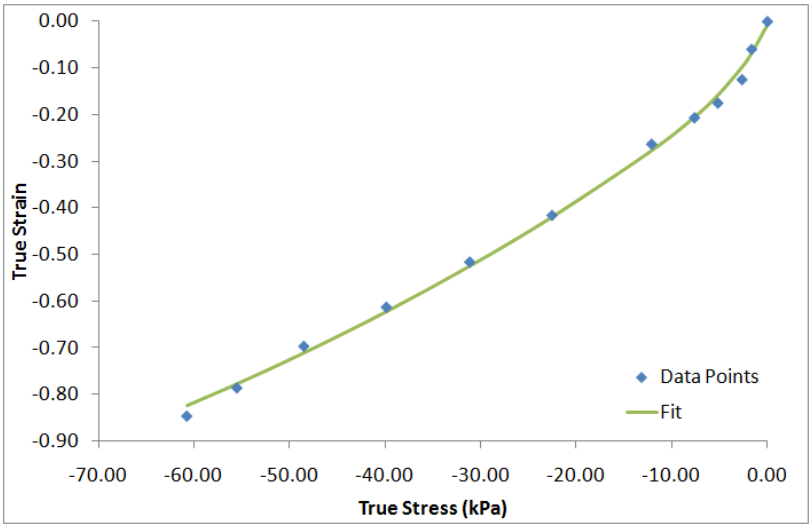


Figure 4.25: Fit for the true strain/stress data.

electrical energy stored in the material by and applied direct current (DC) voltage[1]. The dielectric constant is also the coupling factor in the Maxwell stress tensor, Eq. 4.1, or its simplification for isotropic materials only subjected to a unidirectional field represented by Eq. 4.2.

The manufacturer of the VHB4905 states in its datasheet [62], that the relative dielectric constant[2] of the tape is, approximately, 3.21. Tests done to the tape indicate that the average value for the relative dielectric constant is approximately 6.76. However, this value changes greatly with the deformation imposed to the material, as it can be seen in Fig. 4.26.

[1]The frequency of the applied voltage affects the permittivity. However, since the actuator is supposed to work with direct current, the frequency is irrelevant and only the stationary (0Hz) permittivity is needed

[2]The relative dielectric constant calculated relatively to the permittivity of the vacuum

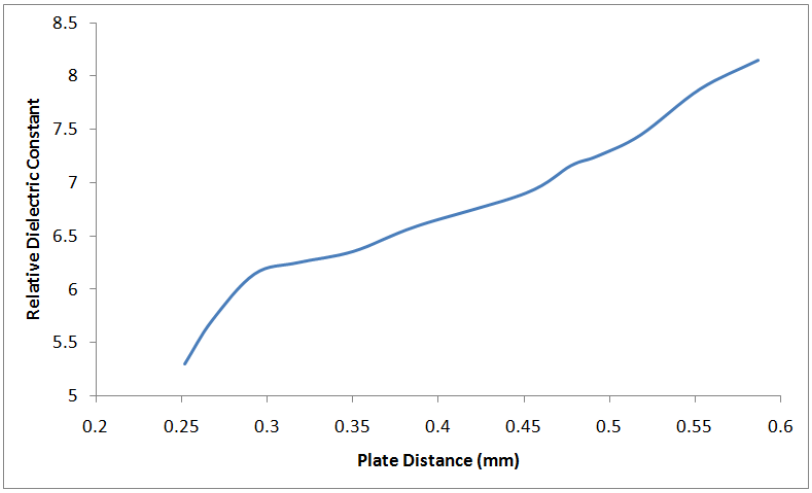


Figure 4.26: Variation of the dielectric constant of the VHB4905 tape with the plate distance[4].

The decrease of the dielectric constant comes in line with tests done with other actuator configurations (in the case of [56, 57], diaphragm and roll configurations) and electrodes (all references use a rubber/grease electrodes described in section 4.4.1). This decrease in the dielectric constant is not immediately visible during the experiments, because the capacity of the tape always increases with the reduction of the thickness, Fig. 4.27, which means the reduction of the distance between the electrodes is compensating the smaller ability of the VHB4905 to store electrical energy.

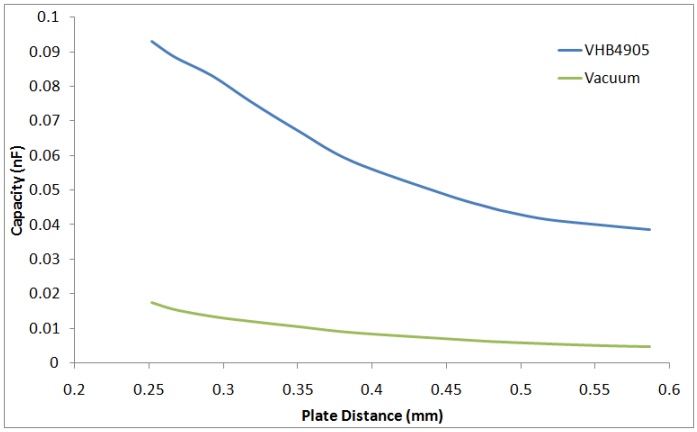


Figure 4.27: Variation of the capacity of the VHB4905 tape and Vacuum (theoretical values) with the plate distance.

4.5.3 Design Process

To start the design process of the actuator, one should establish the ideal configuration of the actuator and then adjust the design to the reality. Ideally, to achieve a maximum amount of electromechanical compression, the dielectric material should be as elastic and flexible as possible to reduce the resistance to the Maxwell stress. It should also have a high dielectric constant to maximize the magnitude of the Maxwell stress for any given electric field. On the electrode side, the flexibility should be as high as the dielectric material to ensure it follows the deformation, and it should have a low resistivity to minimize energy losses and to reduce the electrical time constant of the RC circuit (Resistance-Capacitor). With all these requirements in mind, the material selected for the dielectric medium was the 3M VHB4905 and the Galinstan alloy for the electrode.

The choice of the VHB4905 for the dielectric material was not only for its mechanical and electrical properties, but also for being commercially and readily available. However, the biggest disadvantage of using this material, as stated in the beginning of Section 4.5, is that it is only available in two thicknesses (0.5mm for the VHB4905 and 1mm for the VHB4910). In order to maximize the electrical field (and consequently the magnitude of the Maxwell stress) for a given voltage level, the thickness of the material must decreased. There are two ways to reduce the thickness of the VHB4905 tape: mechanically or chemically. The chemical compression is achieved by adding an additive to a mechanically pre-strained tape and when the reaction is complete the tape is released and a large percentage of the pre-strain is kept [64]. The greatest advantage of this method is the final actuator does not need a pre-straining apparatus to work, with the disadvantage of a reduced final dielectric constant and increased rigidity. Also the experimental setup to perform the reaction is complex, because the pre-strain must be kept while the process occurs and the additive must be added uniformly to avoid wrinkles and other subsequent deformations,[64].

The pure mechanical pre-strain option can achieved by the methods: bi-axial extension of the tape (planar deformation perpendicular to thickness of the tape) or by compression. The bi-axial deformation is the preferred method of pre-strain by various authors [50, 54, 55, 58, 57, 59], because it is the best method to achieve large area deformations (and consequently thickness reductions). However, due to the proportionality of the dimensions of the configuration chosen for the actuator, the stack configuration, it is difficult to deform the actuator bi-axially (most of the

literature regarding EAP actuators uses the single membrane diaphragm configuration, in which the planar dimensions are much larger than the membrane thickness). The only remaining option is to mechanically compress the actuator after the layers are assembled.

Compressing the actuator to decrease the thickness of the layers requires only a press or tensile test machine (for example the Instron 5544, Fig. 4.17). However, if not thoroughly controlled, compression can lead to mechanical instabilities and buckling. The former can be solved or reduced by adding a supporting structure to restrain movement on the perpendicular plane and the buckling can be reduced by the lubrication of the extremities of the actuator while it is being compressed.

Since the actuator needs to be compressed, ideally it should have a circular cross-section in order to have an uniform expansion of the area and to reduce warping of the actuator. Unfortunately, due to its extremely high flexibility and elasticity, the VHB4905 can only be cut by a blade of the grade of a scalpel or by application of heat. The application of heat to the material, also, degenerates the VHB into a gel-like substance that, when solidified, is more adhesive, less elastic, less resilient and more flexible than the original material[5]. Even with a very sharp blade, the VHB is very hard to manipulate because of its elasticity and adhesiveness. Due of these difficulties, it was decided to use a square cross-section.

A square cross-section has its own advantages, such as: one can take advantage from the fact that the material is sold in rolls and only has to perform two cut to have a square, and that, since the cuts must be done by hand, square forms have more probability of matching each other than circular shapes. The existence of mismatching between the cross-sections promotes mechanical instabilities.

The main concerns regarding the electrode liquid is how it is going to be applied to the surface. The are two methods to coat a surface with a liquid: by brush or by spray. Ideally the spray would be preferred because it provides a more uniform coating. The main problem is the amount of liquid that it is waisted and the high surface tension that creates large droplets. Also, due to the high corrosion reaction that the Galinstan has over aluminum (which can be an, somewhat high, exothermal reaction if in presence of water) [65], a normal valve and container cannot be used. On the other hand, paint brushes do not guarantee a coating as good as a spray (the quality of the coating depends on the quality of the brush, the surface tension of the liquid and the skill of the painter), but it makes it easier to apply the coating and it

[5]Changes in the electrical properties were not tested

is more economic regarding the amount of liquid necessary.

Another concern regarding the application of the Galinstan is the state of the surface of the dielectric medium. In order to avoid the formation of droplets instead of a coating, the surface must be free of oil (the Galinstan is both oleophobic and lipophobic), and water [65].

With all these constrains in mind and the objectives to be achieved, a first design, Fig. 4.28, was proposed.

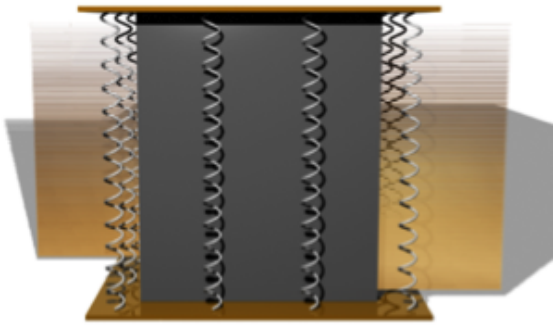


Figure 4.28: Initial design of the actuator.

In this design, the actuator was divided into two parts: the pre-strain apparatus and the EAP stack. The objective of the pre-strain apparatus is twofold: it pre-strains the EAP stack into a predetermined length, Fig. 4.29(a), and couples the EAP stack with the environment, Fig 4.29(b).

The coupling is needed, not only to protect the stack from extensions, but to provide the load needed. The stack can only work in compression and any extension applied to it will create permanent damages. The solution is to link the environment (in the case of the prosthesis the tendons) to the bases of the actuator, whose movement is restrained by the springs. This results in the actuation load being created by the spring rather than the stack itself. The stack only affects the length of the springs, Fig. 4.29(b) and Fig. 4.29(c).

Through testing, it was verified that the proposed design of Fig. 4.28 had two major shortcomings. The first is due to the tight sealing of the electrode fluid and the second due to the copper wires. Because the electrode fluid is sealed between layers of VHB4905 (the sealing is done by the adhesive properties of the VHB itself), the only way to connect the electrodes to the power source is by wires. However,

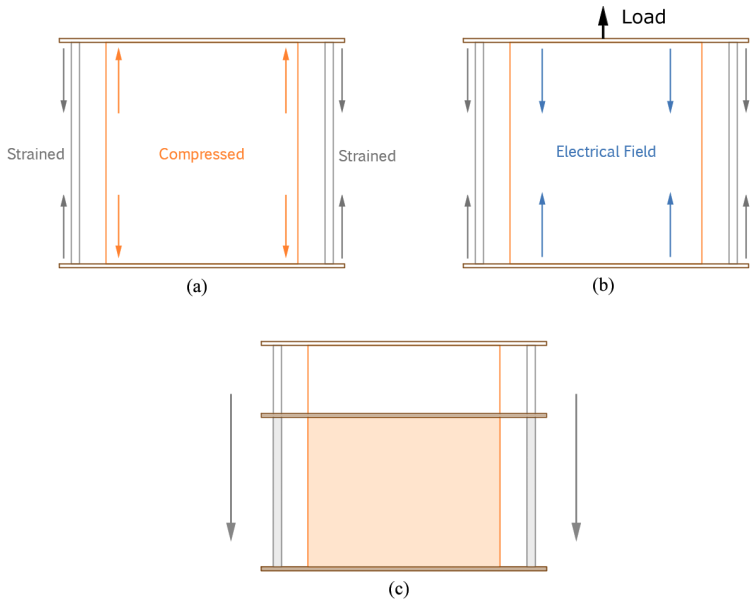


Figure 4.29: Working principle of the actuator.

due to the pre-compression, the liquid is kept under pressure and it was verified that the fluid always leaked, invariably leading to short-circuits. Also, the wires, although thin, add extra inactive length to the actuator, reducing the net electromechanical compression rate.

The solution to the detected problems lies in removing the copper wires and letting the fluid flow out of the stack in a controllable way, Fig. 4.30. By letting the fluid flow out of the stack on the correct side, reduces the pressure on the other three sides and avoids short-circuits. Also because the flow, the electrode coats the sides of the stack (the positive and negative sides), making a conductive surface and, thus, allowing a connection to the power supply. There are two main disadvantages in these changes in the design: one now has to have extra care in handling the actuator to avoid any contact between the two conductive surfaces and avoid overflowing of the fluid, which leaves the electrode surfaces (the surfaces between layers) without enough fluid to encompass the expansion of the cross-section. One method to avoid the overflow is to perform the pre-compression very slowly so that the fluid has time to adapt to the expansion of the cross-section while the excess flows outside the stack. By slowly compressing the stack, due to the viscoelastic nature of the actuator, one also has the advantage of needing smaller loads.

Although the uncontrolled leaking was solved, a new problem appeared during

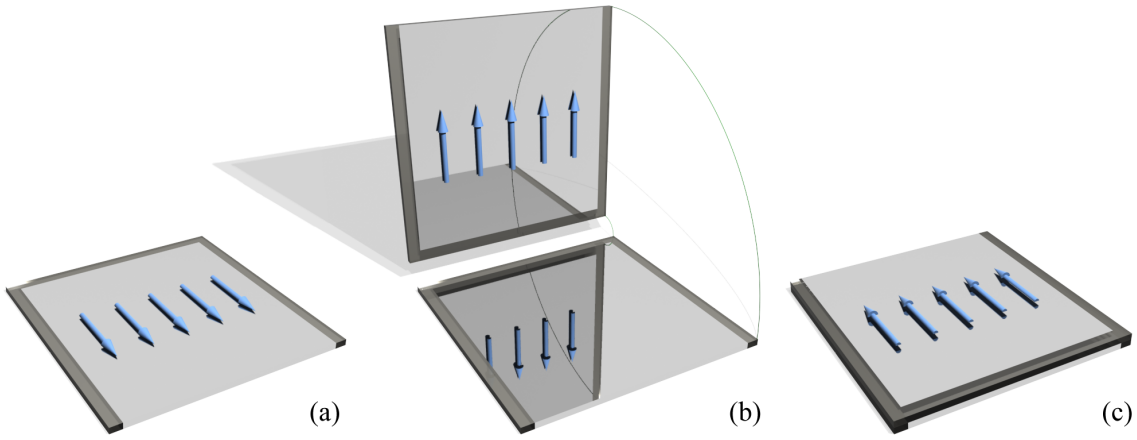


Figure 4.30: Assembly of the second design of the EAP stack: (a) One side of the layer is coated (the arrows represent the painting and flow direction); (b) The bottom side of the second layer is coated and sealed on top of the first one; (c) The process is repeated with the flow in the opposite direction and henceforward, always alternating the coating direction.

the testing of the new actuator design that was not seen on the first one. Due to the manual construction and assembly of the stack, the layers do not exactly match one on top of the other (whether by mismatch of the dimensions or by misplaced layers) and any irregularity leads to mechanical instabilities during the compression. Mechanical instabilities are unsolvable (even a perfect column made of an isotropic material can suffer from instabilities if the dimensions do not fit a certain criterium), but they can be minimized by introducing a guiding structure or by reducing build errors in the layers.

One possible solution to this problem is to decrease the number of cuts needed to make layer, by folding the VHB tape alternately in a shape of an harmonium, Fig. 4.31 and Fig. 4.32. This way, the number of cuts is reduced to half and one side of the stack (the side of the fold) does not need a sealing area, which in turn increases the active area. Also, the fold helps guiding the assembly of the layers, because it forces the tape to rotate around an axis, thus restricting the movement of layer while it is being assembled.

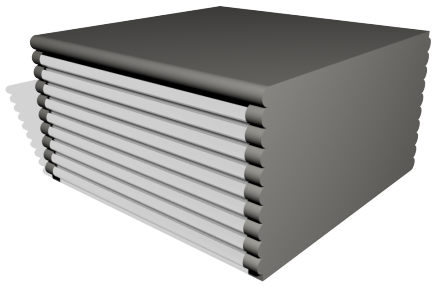


Figure 4.31: Third design of the EAP stack.

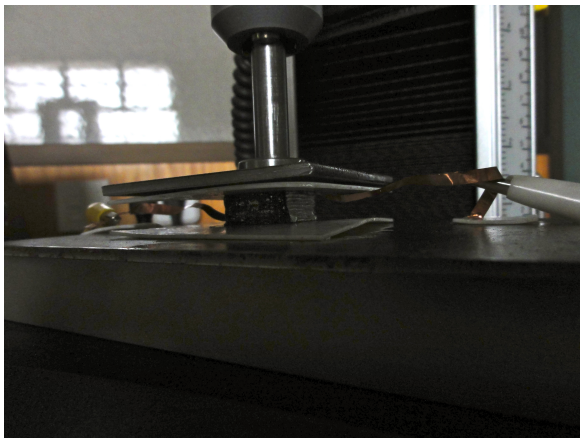


Figure 4.32: EAP stack (third design) during pre-compression.

4.6 Testing of the Prosthesis Actuator

4.6.1 Electrical Circuitry

The design of the powering electric circuit is paramount for the operation of the actuator. Special care must be taken when designing the circuit, not only because of the high voltage levels but also to minimize the effects of the underlying dynamic system of the circuit to ensure that the obtained results form the actuator are not eschewed.

The central component in the circuit is the high voltage converter described in section 4.5.1, Fig. 4.18. This component divides the circuit in two sub-circuits: a high voltage and a low voltage circuits. The low voltage sub-circuit is made of an AC-DV power-supply, a 50kΩ potentiometer (P) and the voltage converter, Fig. 4.33(a). The

high voltage sub-circuit is a traditional RC (resistance-capacitor), where the capacitor is the EAP stack, Fig. 4.33(b).

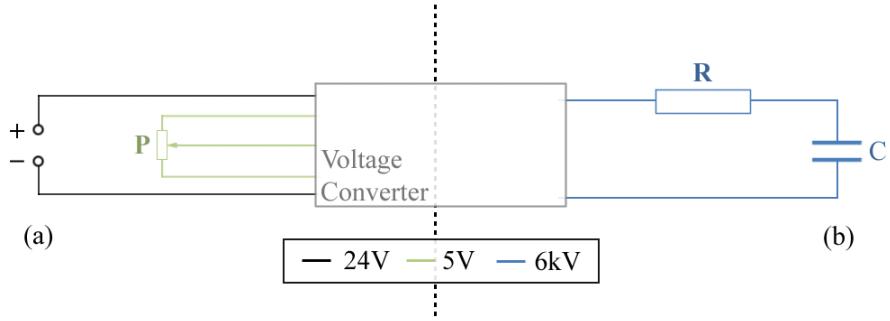


Figure 4.33: Electric Circuit to power the actuator: P is the 5kΩ potentiometer, R is the RC resistance and C is the EAP Stack .

The potentiometer is used to control the voltage supplied by the voltage converter (from 0 to -6kV). The only design variable of this circuit is the resistance (R) in the high voltage sub-circuit. The value of the resistance must high enough to avoid current spikes that could damage the voltage converter and to minimize the time constant of the RC dynamics, Eq. 4.5 to Eq. 4.7:

$$I = \frac{Cs}{1 + RCs}V \tag{4.5}$$

$$V_R = \frac{RCs}{1 + RCs}V \tag{4.6}$$

$$V_C = \frac{1}{1 + RCs}V \tag{4.7}$$

where I is the current intensity, V_R and V_C are the voltages at the terminals of the resistance and capacitor, respectively. R is the resistance, C is the capacity and V is the supplied voltage.

The time constant, τ, of the system described by Eq. 4.7 is given by:

$$\tau = RC \tag{4.8}$$

From Eq. 4.8 and using an average capacity of 1nF, the obtained resistance is 1MΩ. Solving the differential equation in Eq. 4.5 for a step of -6kV, the maximum current intensity verified is 6mA.

4.6.2 Testing and Results

To test the performance of the actuator, one needs to measure the length and the load the EAP stack is subjected to. Because of the high strain levels regular strain gauges cannot be used, so an alternative must be found. One alternative is the Instron tensile strength testing machine. The Instron machine can compress a material and maintain a level of applied load, while measuring and recording the applied loads and the length of the material through time.

The experiments started by applying a pre-compression to an EAP stack with a length of 14mm by 19 mm (sides). The stack was compressed until the applied load was -10.5N and left until the length stabilized. When the length stabilized a square signal of -6kV was applied[6], Fig. 4.34 and Fig. 4.35.

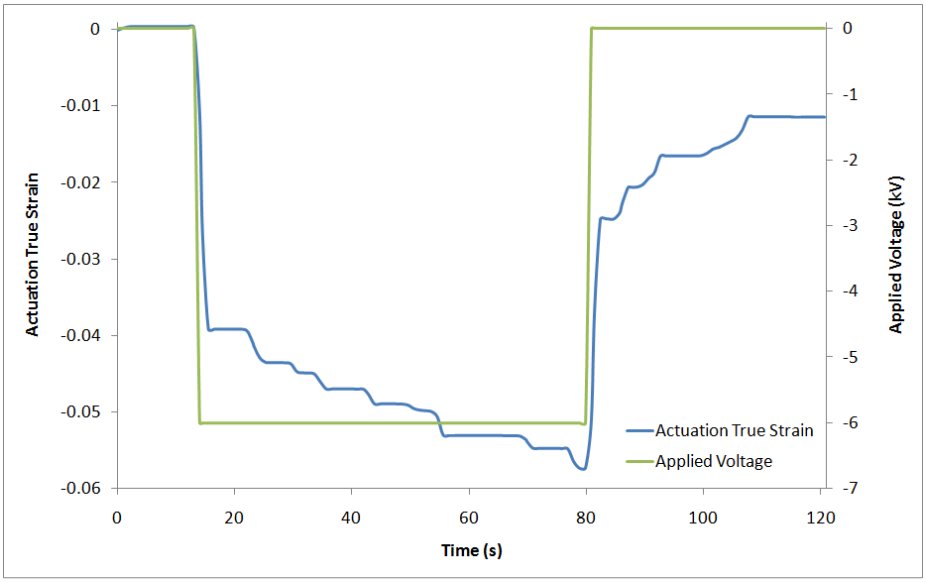


Figure 4.34: Comparison between a square wave application of voltage and the resulting actuation strain.

From Fig. 4.35, it can be seen the moment when the applied voltage was changed, to -6kV and back, by looking to the two spikes. The first spike represents a sudden decrease in the load, meaning that an external load (unaccounted by the machine) was applied. The machine tries then to return to the pre-determined applied load of -10.5N, resulting in a further compression of the stack, Fig. 4.34. This behavior mimics the springs of the pre-strain apparatus of the full actuator – if the resistance to the

[6]Due to the signal being applied manually, it was not possible to guarantee any frequency

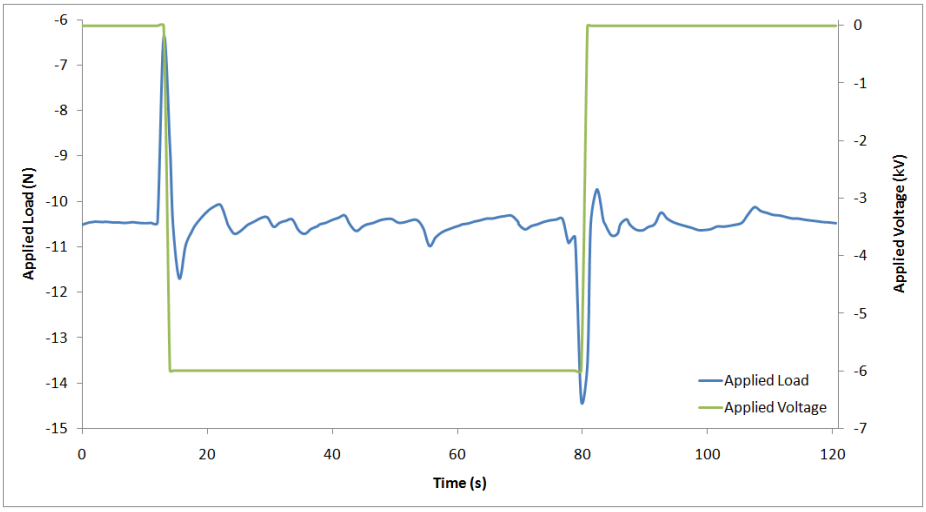


Figure 4.35: Comparison between a square wave application of voltage and the applied load maintained by the testing machine.

springs compression if reduced (through the electrically induced load), the springs will try to regain their initial size and, thus, compressing the EAP stack further, Fig. 4.29(b) and (c). The second spike, marks the removal of the applied voltage, which is recorded by the machine as a sudden increase in the applied load. Again, the machine will adjust the applied load back to the -10.5, which leads to an increase in the size of the actuator, Fig. 4.34. Calculating the difference between the spikes and the average load, yields an electrically induced load between 3.9N and 4.1N.

Using Eq. 4.4 and the data obtained from the test in section 4.5.2 and section 4.5.2, one can calculate the value of the stationary strain of the actuator when under an applied voltage. The results can be seen in Table 4.5, for the values concerning this experiment, or Table C.1 (Appendix C) for all initial applied loads.

Table 4.5: Predicted values for the stationary strain of the actuator when under an -6kV applied voltage.

Applied Load (N)	Dielectric Constant	Maxwell Stress (kPa)	Actuation Strain (True Strain)
-10.035	6.591	-5.262	-0.0647
-15.006	6.357	-6.198	-0.0830

With the theoretical stationary values, one can compare with the experimental data. However, the values obtained in the experiments do not reach stationary values[7]. A safe method to determine an approximated stationary value for the actuation strain is using an identified model based on the data. Through close inspection of the experimental data and through some trial and error steps, a discrete LTI model was found. This model has the structure of an ARMAX222 (please refer to section 3.2.4) with a sample time of 1s, Eq. 4.9:

$$\left(1 - a_1 z^{-1} + a_2 z^{-2}\right) \epsilon(z) = \left(b_1 z^{-1} + b_2 z^{-2}\right) V(z) + \left(1 - c_1 z^{-1} + c_2 z^{-2}\right) e(z) \quad (4.9)$$

where ϵ, V and e are the discrete actuation true strain, applied voltage and model perturbations, and z is the shift operator. The coefficient used by the model are shown in Table 4.6.

Table 4.6: Coefficients for the ARMAX222 model of the response of the actuator under an applied voltage.

Coefficient	Value
a_1	1.7612
a_2	0.7636
b_1	1.5257×10^{-6}
b_2	-1.4965×10^{-6}
c_1	-2.7401×10^{-1}
c_2	-7.2598×10^{-1}

The comparison between the experimental data the model is plotted in Fig. 4.36.

Although being an non-linear system, due to the relatively low strain, the linear ARMAX222 provides gives a good approximation.

From Eq. 4.9 and Table 4.6, the stationary value of the compression strain of the actuator can be approximated and compared with the predicted values form Table 4.5. Calculating the $\lim_{z\to1} \epsilon(z)$, yields a stationary value of -0.0733, which is in line with the predicted values.

[7]It is considered stationary when the output does not change more than 2%

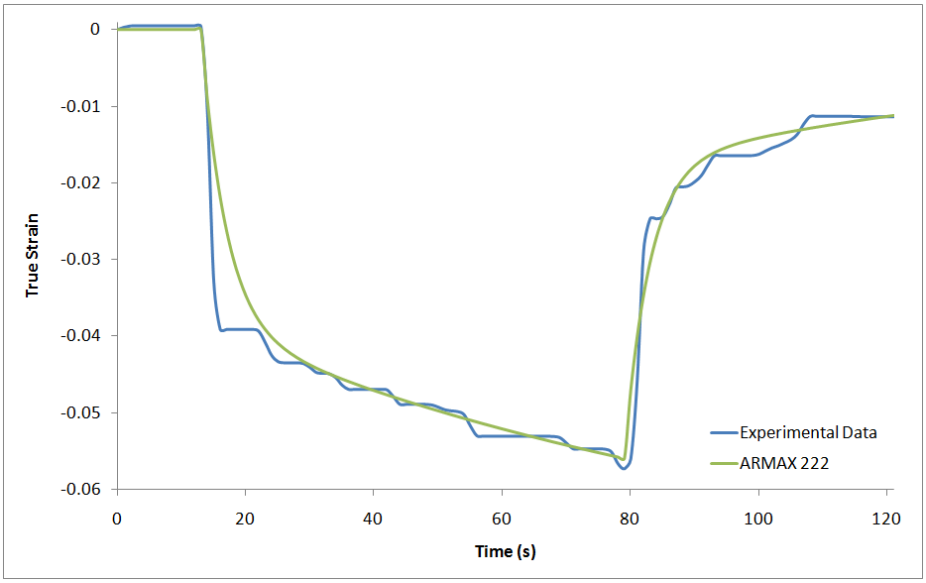


Figure 4.36: Comparison between the experimental data the ARMAX222 model.

When comparing the experimental results with the performance of natural muscles, there are points where the EAP actuators are highly promising. However there are still characteristics where the muscles largely excel the EAP actuators.

The compression rate of skeletal muscles of the hand, although varying highly from muscle to muscle and from person to person, can go from, approximately, 30% (for the long flexors) to around 18%, being the average around 20% [15]. When directly comparing with the experimental data, one sees that the EAP actuator, with its 7% compression strain, lags behind. However, the performance and compression rates of the actuator can be enhanced:

- By increasing the input voltage. To achieve the 7%, the actuator was supplied a voltage of 6kV[8], which results in an electrical field of 13.4MV/m. Comparing the applied electrical field with the ones used in the literature (Table 4.1 and Table 4.2), one can see that larger input voltages can still be applied. If the values of Tables 4.5 and C.1 are recalculated for an applied voltage of 9kV (22MV/m), the stationary compression raises up to a 18% compression rate[9];
- By improving the construction process. The manual-based assembly of the

[8]The actual applied voltage is -6kV, however, due to the quadratic nature of the Maxwell stress, the signal is irrelevant

[9]One could input even larger voltages to the model, although with the risk of extrapolation

actuator has an impact on the overall performance and by guaranteeing a better coating of conductive surfaces (e.g. using spray painting), a larger active area (e.g. optimize the seal area) or a better stacking of the layers (e.g. improve the matching between the cross-section dimensions), the performance can be improved. One way to apply these improvements could be done by specialized machinery;

- By using a different dielectric medium. Although one of the best, the VHB4905 is not the only material available to make an dielectric EAP actuator (please refer to Tables 4.1 and 4.2). One could even, with the correct knowledge in chemistry, develop a specially tailored polymer optimized for this application.

On the other hand, when comparing the response times and speed of compression of the muscles with the actuator, the former greatly outperform the latter. although having also, a large variance, the time for a muscle to go from fully relaxed to fully contracted is in the order of magnitude of $O(10^{-1})$ of a second[10] [66], which goes in stark contrast with the dynamic behavior of the EAP actuator. Analyzing the ARMAX222 model of the actuator one can determine the time domain characteristics, Table 4.7, yielding a settling time of 303.1s, which is four orders of magnitude larger than the settling time of muscles.

Table 4.7: Time domain characteristics of the ARMAX222 model.

Quantity	Value
Rise Time	147.5s
Settling Time	303.1s

As a remark, one should note that these values are somewhat affected by the relatively slow sample time of the instron testing machine, which is 1 second. This effect can be seen in the width of the load spikes in Fig. 4.35. Although the time constant of the dynamics of the electrical system is 1ms, it takes about five seconds for the machine to react to the application (or removal) of the electrically induced load, slowing the response of the whole system. In the actual actuator, where the machine is replaced by springs, the response is immediate.

[10]The maximum contraction velocity is only achieved when the muscle has no external load applied.

Nevertheless, even discounting the effect of the machine, the dynamic behavior of the VHB4905 is by nature slow, Fig. 4.23, which means there are only two alternatives to obtain better response times to reach a certain compression level: by applying a larger voltage than needed or by changing the dielectric material for one with faster dynamics.

Chapter 5

Conclusions and Future Work

The main objective of this work was to start the development of a five fingered fully articulated human hand prosthesis. The final goal is to have a prosthesis capable of performing the same range of movements as its natural counterpart, most notably the six basic movements: cylindrical grasp, precision grasp, hook prehention, tip grasp, spherical grasp and lateral hip. Although showing a big evolution in terms not only of performance but also in availability, current commercially available prosthesis can be divided in two categories: the aesthetic prosthesis, that are, relatively, inexpensive but only provide a cosmetic replacement, and the mechanical prosthesis, that can perform tasks and are created in several tiers of complexity and prices.

The mechanical prosthesis that are available commercially, go from the simple grip-only prosthesis, to complex five-fingered prothesis. These articulated five-fingered prosthesis, most notably the Touch Bionics i-LIMB and the Bebionics [12, 13], are made of complex servomotor-based mechanisms. These mechanisms, due to their complexity, not only are prone to mechanical failure (the more complex a mechanism is, more it is susceptible to failure) and difficult maintenances, but are also very expensive to the end-user.

With this, in mind, the objective of this work was to develop the two main components os a hand prosthesis in a manner that the final product is as simple (and low cost) as possible while maintaining the necessary dexterity. The two essential components developed in this thesis were the controller for the prosthesis and the actuators. The controller was design using artificial neural networks, because they (due to their universal interpolator capabilities) form a compact closed system that can handle the nonlinearities and heavy couplings of the dynamics of the hand. The neural networks proved to be capable of learning the particularities and complexities

of the hand dynamics and the developed controller. A fully neural network based nonlinear PID controller proved able to control the hand in a wide range movements. Unlike other neural network PID based controllers, that use the network to compute the PID parameters and use them in a regular PID controller [67, 43], or use an external generic neural network to improve the performance a PID [44], the neural PID used to control the prosthesis is itself a nonlinear PID.

The final advantage of artificial neural networks is that due to their structure, the neural networks can be implemented in a single microprocessor or, if one wants to take advantage of their parallel processing capabilities, multiple microprocessors. One can even implement simple on-line learning functionalities in the controller, so that it can continually adapt to the usage of the prosthesis, optimizing itself to the movements most commonly performed.

In order to avoid the complex mechanisms and their shortcomings, the actuator had to be as simple as possible while having a performance as close as possible to the natural muscles. With this in mind, it was decided develop solid state actuators, more precisely, dielectric electroactive polymer-based actuators. These actuators are the only type of solid state actuators that can reach high enough compression rates similar to the natural muscles. Throughout the course of the development of the actuator, several designs were tested, each one improving the previous. The final design can be seen in Fig. 4.31. This design is divided in two parts: a pre-straining apparatus, that reduces the initial thickness of the EAP to decrease the required voltage level and generate the actuation load, and the EAP stack that shortens when a voltage level is applied to the terminals.

The EAP stack, which is the core of the actuator, is composed of two materials, each with a very specific purpose: the dielectric EAP medium and the electrodes. For the dielectric medium the commercially available 3M VHB4905 was chosen. The main advantage of this material is its availability while being one of the materials with the best performances as an EAP. However, by being a commercial product, the form factor in which it is available is predetermined be the seller and it had an impact in the development of the actuator. Also, because the chemical components are unknown, a series of mechanical and electrical tests had to be done in order to determine its properties.

Galinstan alloy was selected for the electrodes. This alloy is also commercially available, however and unlike the VHB, its properties and composition are known. The Gallistan is a liquid metal alloy composed of Gallium, Indium and Tin, and it

is liquid at room temperature and it is non-toxic. The choice of a liquid electrode introduces a new alternative that contrasts with the commonly used solution in the literature – carbon-grease electrodes. Carbon-grease electrodes have the disadvantages of having high resistance which not only increases the losses but also slows down the dynamics of the electrical system that powers the stack. Also, due to its semirigid nature, the carbon-grease tends to generate cracks that reduce the active area of the electrode. By using a liquid electrode one does not have to be concerned with cracks because the fluid follows the deformation of the EAP, while having a very low resistivity. The only disadvantage of using a liquid is the containment, however this problem was greatly reduced by special considerations in the final design of the actuator.

During the performance tests of the actuator, the actuator performed as expected and according to the theoretical predictions. The test specimen showed a compression strain of, approximately, 4% in the first 3 seconds under a -6kV applied voltage, with a stationary compression strain of 7%. These strains are smaller then the ones attained by natural muscles, which average at 20%. However, it has been shown that the performance of the actuator can be enhanced by increasing the input voltage, that it is still far from the maximum the actuator can handle and by simply increasing the voltage to -9kV, the actuator can have an 18% compression strain. Also, improving the construction process, one can guarantee less defects and losses. Finally, the dielectric medium can be optimized by tailoring a custom polymer specifically for this purpose.

5.1 Synthesis of the Contributions to the State of the Art

The contributions of this thesis to the state of the art can be summarized as follows:

- Development of a new Artificial Neural Network controller - Neural PID. In this implementation, a fully nonlinear PID-based controller is completely implemented using neural networks and, consequently, not requiring of other controllers.

- Development of an Artificial Neural Network model and controller (based on the Neural PID) of a fully articulated human hand. The controller and model

can then be used to control the future prosthesis.

- Design and development of a new EAP-based actuator using a liquid metal alloy as electrodes. The resulting actuator has a high actuation strain, comparable to the natural skeletal muscles. This actuator can not only be used in the prosthesis, but also in other applications where solid states actuators are advantageous.

5.2 Future Work

The proposed future work should primarily focus on the improvements to the actuator design and the neural network controller. The controller has good overall performance, however the control signal is very noisy. This a characteristic of neural networks and further tuning of the parameters is necessary. Also, the proposed controller essentially is a nonlinear PID, and further controller configurations should be also tried, such as neural network predictive control. By using predictive control one could, possibly, decrease the response time of the prosthesis, lessening the speed requirements of the actuators. Finally, improved optimization algorithms should be tried and implemented to help the neural network model and the controller to reach better performance levels. One type of optimizers that could bring benefits to the training of the model and controller would be a global optimizer. Due to the ill-condition verified in the cost functions of the neural networks, a global optimization could improve the resistance of the optimization process to local minima.

On the side of the actuator, the proposed future work is mainly focused in improvements to the construction process and materials. The biggest source of improvements in the overall performance of the actuator resides in improving the assembly process of the EAP stack. The current process is an exclusively manual labour, which generates a large number of imperfections that decrease the final compression strain and it is difficult to carry out the pre-strain step. Analyzing the assembly process, one can see that it is highly repetitive and automated and specially designed machinery would greatly enhance the assembly of the EAP stack, quality and time-wise . Another area where further work is needed is in the EAP material itself. As stated above, designing a polymer tailored specifically to this application would also bring an improvement to the performance of the actuator. Although, there is still room for improvements in the total contraction strain (which could be attained by increasing the dielectric

constant of the material and/or decreasing the rigidity of the material), it is the speed of contraction that needs to be increased. To increase the speed of contraction, the new material should have faster dynamics or less viscosity.

Finally, when optimal performances in the actuator and controller have been reached, one can assemble everything in a supporting structure (a model of the bones human hand could be used with artificial joints, such as the ones proposed by [68]). With the prosthesis assembled, the last step should be to create the interface to the rest of the body through and EMG interface such as the one used in the i-LIMB and Bebionic hands.

Bibliography

[1] P. J. Kyberd, C. Light, P. H. Chappel, J. M. Nightingale, D. Whatley, and M. Evans, "The design of anthropomotphic prosthetic hands: A study of the southampton hand," *Robotica*, 2001.

[2] S. yoon Jung, S. kyun Kang, and I. Moon, "Design of biomimetic hand prosthesis with tendon-driven five fingers," in *Proceeding of the 2nd Biennial IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics*, pp. 895–900, 2008.

[3] G. Guo, W. A. Gruver, and X. Qian, "A robotic hand mechanism with rotating fingertips and motor-tendon actuation," *IEEE Robotics & Automation Magazine*, pp. 1023–1028, 1991.

[4] Y. K. Lee and I. Shimoyama, "A skeletal framework artificial hand actuated by pneumatic artificial muscles," *Proceedings of 1999 IEEE International Conference on robotics & Automation, Detroit*, pp. 926–931, 1999.

[5] J. D. Crisman, C. Kanojia, and I. Zeid, "Graspar: A flexible, easily controllable robotic hand," *IEEE Robotics & Automation Magazine*, pp. 32–38, 1996.

[6] R. O. Ambrose, H. Aldridge, R. S. Askew, R. R. Burridge, W. Bluethmann, M. Diftler, C. Lovechick, D. Magruder, and F. Rehmark, "Robonaut: Nasa's space humanoid," *IEEE Intelligent Systems*, pp. 57–63, July/August 2000.

[7] P. J. Kyberd and J. L. Pons, "A comparison of the oxford and manus intelligent hand prostheses," *Proceedings of the 2003 IEEE International Conference on Robotics & Automations*, pp. 3231–3236, 2003.

[8] R. Okuno, M. Fujikawa, M. Yoshida, and K. Akazawa, "Biomimetic hand prosthesis with easily programmable microprocessor and high torque motor,"

Proceedings ot the 25th Annual International Conference of the IEEE EMBS, pp. 1674–1677, 2003.

[9] M. V. Weghe, M. Rogers, M. Weissert, and Y. Matsuoka, "The ACT hand: Design ot the skeletal structure," *Proceedings of the 2004 IEEE International Conference on Robotics & Automation*, pp. 3375–3379, 2004.

[10] N. Gialias and Y. Matsuoka, "Muscle actuator design for the act hand," *Proceedings of the 2004 IEEE International Conference on Robotics & Automation*, pp. 3380–3385, 2004.

[11] K. B. Fite, T. J. Withrow, K. W. Wait, and M. Goldfarb, "Liquid-fueled actuation for an anthopomorphic upper extremety prosthesis," in *Proceedings of the 28th IEE EMBS Annual International Conference*, 2006.

[12] "The i-limb hand," May 2011.

[13] BeBionic, "Bebionic - product brochure," tech. rep., BeBionic, 2010.

[14] H. Gray, *Gray's Anatomy*. Senate, 16th ed., 2003.

[15] R. Tubiana, ed., *The Hand*, vol. 1. WB. Saunders, 1981.

[16] V. C. Mow and R. Huiskes, *Basic Orthopaedic Biomechanics and Mechano-Biology*. Lippincott Williams & Wilkins, 3 ed., 2005.

[17] R. Featherstone, "A divide-and-conquer articulated-body algorithm for parallel o(log(n)) calculation of rigid-body dynamics. part 1: Basic algorithm," *The International Journal of Robotics Research*, vol. 18, no. 9, pp. 867–875, 1999.

[18] K. S. Anderson and S. Duan, "Highly paralelizable low order algorithm for the dynamics of complex multi-rigid-body systems," *AIAA Jnl. Guidance, Control & Dynamics*, vol. 23, no. 2, pp. 355–364, 2000.

[19] R. Featherstone, *Robot Dynamics Algorithms*. Kluwer Academic Publishers, 1984.

[20] A. Carvalho, "Multibody dynamics modelling and analysis of the human hand," Master's thesis, University of Victoria, 2007.

[21] S. McMillan, D. E. Orin, and R. B. McGhee, "Efficient dynamic simulation of an underwater vehicle with a robotic manipulator," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 25, no. 3, pp. 1194–1206, 1995.

[22] M. Arvetti, G. Gini, and M. Folgheraiter, "Classification of emg signals through wavelet analysis and neural networks for controlling an active hand prosthesis," *Rehabilitation Robotics, 2007. ICORR 2007. IEEE 10th International Conference on*, 2008.

[23] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biology*, vol. 52, no. 1-2, pp. 99 – 115, 1990.

[24] J. A. K. Suykens, J. P. L. Vandewalle, and B. L. R. D. Moor, *Artificial Neural Networks for Modelling and Control of Non-linear Systems*. Kluwer Academic Publishers, 1996.

[25] R. Rojas, *Neural Networks*. Springler-Verlag, 1996.

[26] F. L. Lewis, J. Campos, and R. Selmic, *Neuro-Fuzzy Control of Industrial Systems with Actuator Nonlinearities*. Society for Industrial and Applied Mathematics, 2002.

[27] D. H. Nguyen and B. Widrow, "Neural networks for self-learning control systems," *International Journal of Control*, vol. 54, no. 6, pp. 1439 – 1451, 1991.

[28] K. S. Narendra and K. Parthasarathy, "Gradient methods for the optimization of dynamical systems containing neural networks," *IEEE Transactions on Neural Networks*, vol. 2, pp. 252–262, March 1991.

[29] J. S. Arora, *Introduction to Optimum Design*. McGraw Hill, 2004.

[30] C. M. Bishop, "Exact calculation of the hessian matrix for the multi-layer perceptron," *Neural Computation*, vol. 4, no. 4, pp. 494–501, 1992.

[31] H. X. Yin and D. L. Du, "The global convergence of self-scaling bfgs algorithm with nonmonotone line search for unconstrained nonconvex optimization problems," *Act Mathematica Sinica*, vol. 23, pp. 1233–1240, September 2007.

[32] C.-C. Peng and G. D. Magoulas, "Adaptive self-scaling non-monotone bfgs training algorithm for recurrent neural networks," *Proceedings of ICANN 2007*, vol. I, pp. 259–268, 2007.

[33] J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Inc., 1983.

[34] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical Programming*, no. 45, pp. 503–528, 1989.

[35] Y. Shang and B. W. Wah, "Global optimization for neural network training," *IEEE Computer*, vol. 29, pp. 45–54, March 1996.

[36] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[37] T. P. Caudell and C. P. Dolan, "Parametric connectivity: Training of constrained networks using genetic algorithms," in Schaffer [69], pp. 370–374.

[38] X. Yao, "Evolutionary artificial neural networks," *International Journal of Intelligent Systems*, vol. 4, pp. 539–567, 1993.

[39] G. F. Miller, P. M. Todd, and S. U. Hegde, "Designing neural networks using genetic algorithms," in Schaffer [69], pp. 379–384.

[40] K. Ogata, *Modern control engineering*. Upper Saddle River, New Jersey: Prentice-Hall, Inc., 4th ed., 2002.

[41] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, pp. 4–27, March 1990.

[42] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*. Upper Saddle River, New Jersey: Prentice-Hall, Inc., 1991.

[43] Y.-K. Yeo and T.-I. Kwon, "A neural pid controller for the ph neutralization process," *Industrial & Engineering Chemistry Research*, vol. 3, no. 38, pp. 978–987, 1999.

[44] A. Andrášik, A.Mészáros, and S. de Azevedo, "On-line tuning of a neural pid controller based on plant hybrid modeling," *Computers & Chemical Engineering*, no. 28, pp. 1499–1509, 2004.

[45] D. A. Skoog, F. J. Holler, and S. R. Crounch, *Principles of Instrumental Analysis*. Brooks Cole, 2006.

[46] Y. B. Cohen, *Electroactive Polymer (EAP) Actuators as Artificial Muscles*. SPIE, bellingham, washington ed., 2001.

[47] G. Kofod and P. Sommer-Larsen, "Silicone dielectric elastomer actuators: Finite-elasticity model of actuation," *Sensors and Actuators A*, pp. 273–283, 2005.

[48] M. Akbay, "Performance of compliant electrodes in electoactive polymers (eap) actuators," Master's thesis, North Carolina State University, 2004.

[49] A. Rajamani, M. Grissom, C. Rahn, Y. Ma, and Q. Zhang, "Wound roll dielectirc elastomer actuators: fabrication, analysis and experiments," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1520–1525, 2005.

[50] F. Carpi, A. Migliore, G. Serra, and D. de Rossi, "Helical dielectric elastomer actuators," *IOP Smart Materials and Structures*, vol. 6, pp. 1–7, 2005.

[51] R. Kornbluh and et. al., "High-field electrostriction of elastomeric polymer dielectrics for actuation," in *Smart Structures and Materials 1999: Electroactive Polymer Actuators and Devices, Proc. SPIE*, vol. 3669, pp. 149–161, SPIE, 1999.

[52] G. Kofod and et. al., "Actuation response of polyacrylate dielectric elastomers," in *Smart Structures and Materials 2001: Electroactive Polymer Actuators and Devices, Proc. SPIE*, vol. 4329, pp. 141–147, SPIE, 2001.

[53] K. Meijer, "Muscle-like actuators? a comparison between three electroactive polymers," in *Smart Structures and Materials 2001: Electroactive Polymer Actuators and Devices, Proc. SPIE*, vol. 4329, pp. 7–15, SPIE, 2001.

[54] R. Pelrine and et. al., "High-speed electrical actuated alastomers with strain greater than 100%," *Science*, vol. 287, pp. 836–839, 2000.

[55] R. Pelrine, R. Kornbluh, and J. Joseph, "Electrostriction of polymer dielectrics with compliant electrodes as mean of actuation," *Sensors and Actuators A*, vol. 64, pp. 77–85, 1998.

[56] M. Wissler and E. Mazza, "Electromechanical coupling in dielectric elastomer actuators," *Sensors and Actuators A*, vol. 138, pp. 384–393, August 2007.

[57] M. Wissler and E. Mazza, "Mechanical behaviour of an acrylic elastomer used in dielectric elastomer actuators," *Sensors and Actuators A*, vol. 134, pp. 494–504, March 2006.

[58] R. Pelrine and et. al., "High-field deformation of elastomeric dielectrics for actuators," *Materials Science and Engineering*, vol. 11, pp. 89–100, 2000.

[59] G. Kofod, *Dielectric elastomer actuators*. PhD thesis, The Technical University of Denmark, 2001.

[60] J. Su and et. al., "Preparation and characterization of electrostrictive polymer actuators," *Journal of Applied Physics*, vol. 9, pp. 317–321, 1998.

[61] M. Watanabe and et. al., "Wrinkled polypyrrole electrodes for electroactive polymer actuators," *Journal of Applied Physics*, vol. 92, pp. 4631–4637, 2002.

[62] M. I. Adhesives and T. Division, "3m vhb tapes technical data sheet," September 2010.

[63] D. V. Rosato and D. V. Rosato, *Plastics Engineered Product Design*. Elsevier, 1th ed., 2003.

[64] S. M. Ha, W. Yuan, Q. Pei, R. Pelrine, and S. Stanford, "Interpenetrating networks of elastomers exhibiting 300*Smart Materials and Structures*, vol. 16, pp. 280–287, March 2007.

[65] Geratherm Medical AG, *Safety Data Sheet acc, to Guideline 93/112/EC*, 2004.

[66] R. L. Lieber, "Skeletal muscle is a biological example of a linear electro-active actuator," in *Proceedings of SPIE's 6th Annual International Symposium and Materials*, March 1999.

[67] P. Gomathi, T. Manigandan, and N. Devarajan, "Design of neural pid controller for reduced order model," in *Proceedings of the International Conference on Man-Machine Systems (ICoMMS), 11-13 October 2009, Batu Ferringhi, Penang, Malaysia*, October 2009.

[68] Z. Xu, E. Todorov, B. Dellon, and Y. Matsuoka, "Design and analysis of an artificial finger joint for anthropomorphic robotic hands," in *2011 IEEE International Conference on Robotics and Automation*, 2011.

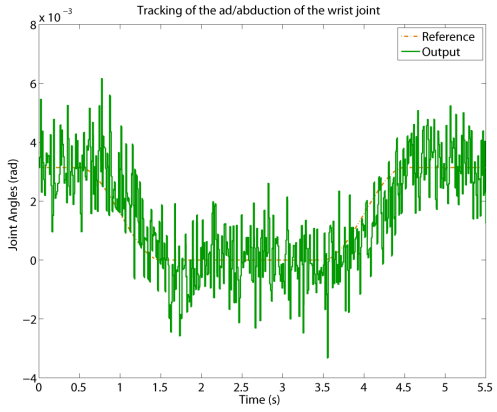
[69] J. D. Schaffer, ed., *Proceedings of the 3rd International Conference on Genetic Algorithms, George Mason University, Fairfax, Virginia, USA, June 1989*, Morgan Kaufmann, 1989.

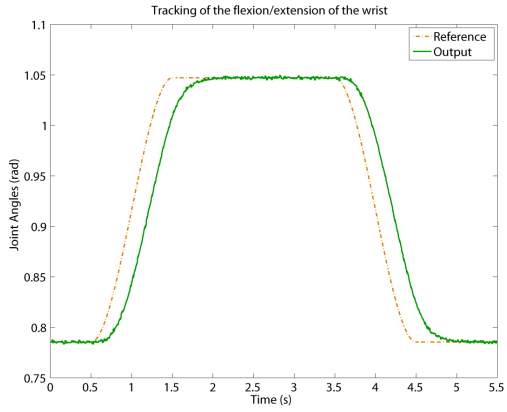
Appendix A

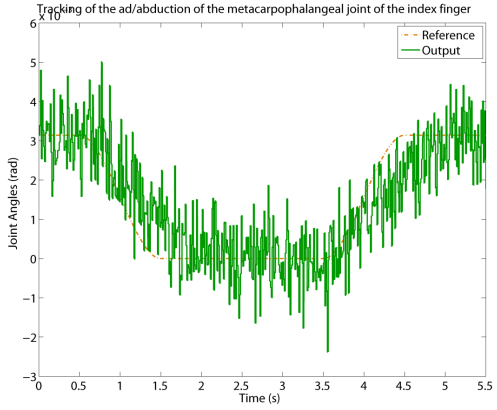
Detailed Results of the Controller Tests

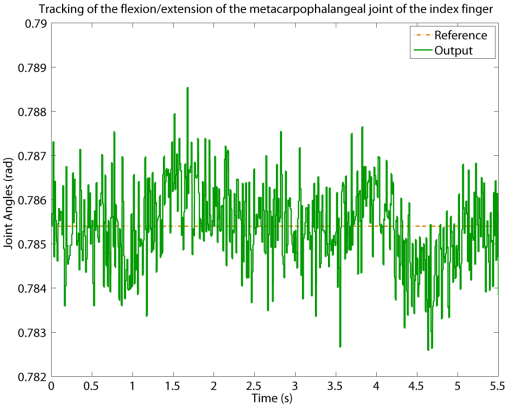
A.1 Results for the Grip Movement

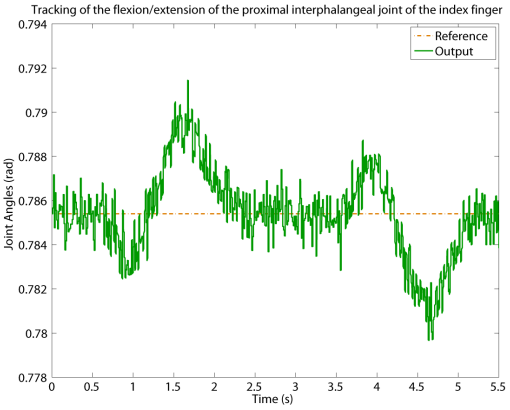
A.1.1 Output Signals

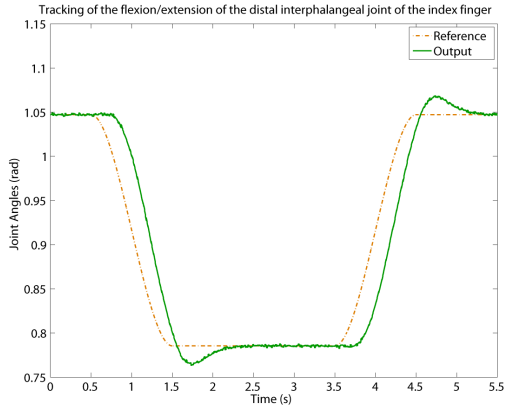

Tracking of the ad/abduction of the wrist joint
x 10^-3
Reference
Output
Joint Angles (rad)
Time (s)

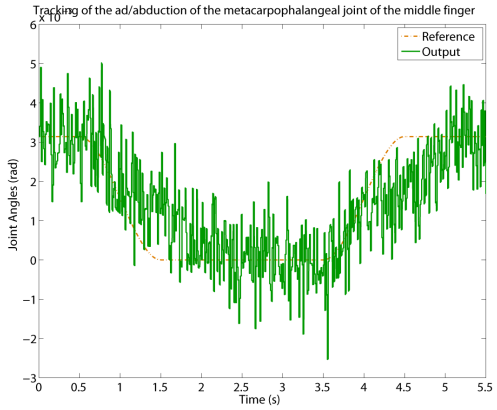

Tracking of the flexion/extension of the wrist
Reference
Output
Joint Angles (rad)
Time (s)

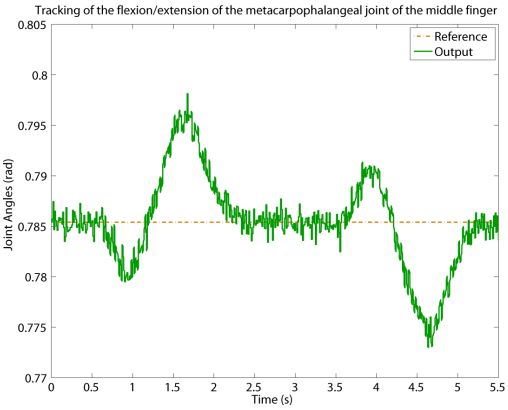

Tracking of the ad/abduction of the metacarpophalangeal joint of the index finger
x 10^-3
Reference
Output
Joint Angles (rad)
Time (s)

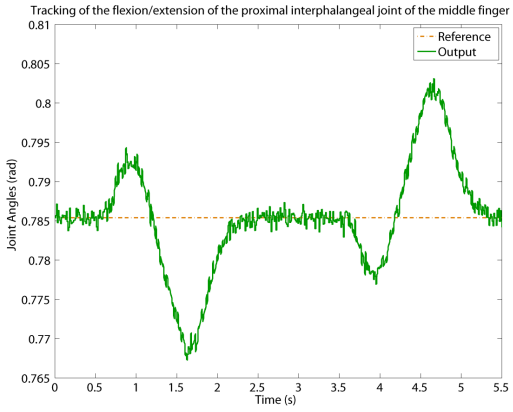

Tracking of the flexion/extension of the metacarpophalangeal joint of the index finger
Reference
Output
Joint Angles (rad)
Time (s)

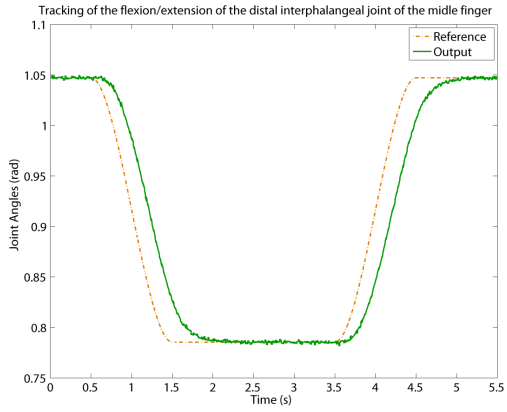

Tracking of the flexion/extension of the proximal interphalangeal joint of the index finger
x 10^-1
Reference
Output
Joint Angles (rad)
Time (s)

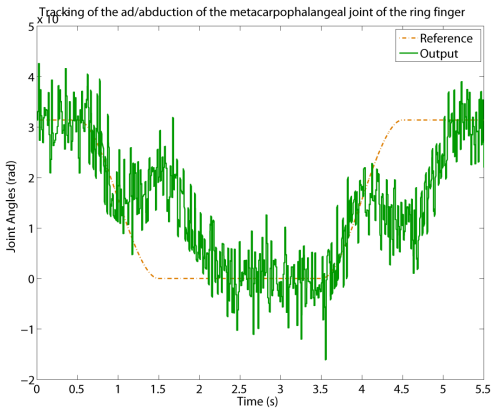

Tracking of the flexion/extension of the distal interphalangeal joint of the index finger
Reference
Output
Joint Angles (rad)
Time (s)

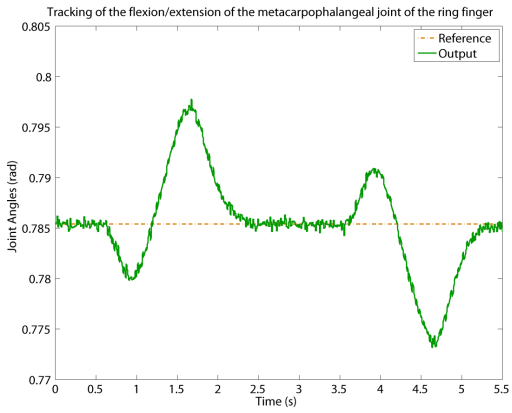

Tracking of the ad/abduction of the metacarpophalangeal joint of the middle finger
x 10^-3
Reference
Output
Joint Angles (rad)
Time (s)

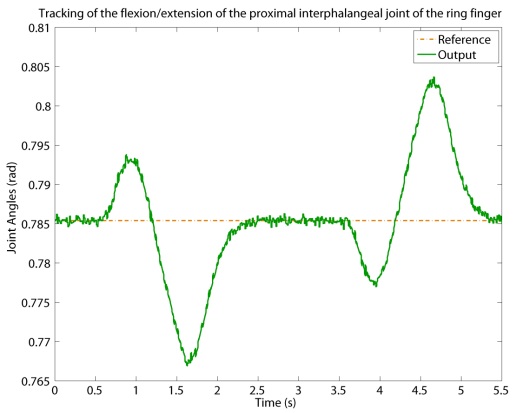

Tracking of the flexion/extension of the metacarpophalangeal joint of the middle finger
Reference
Output
Joint Angles (rad)
Time (s)

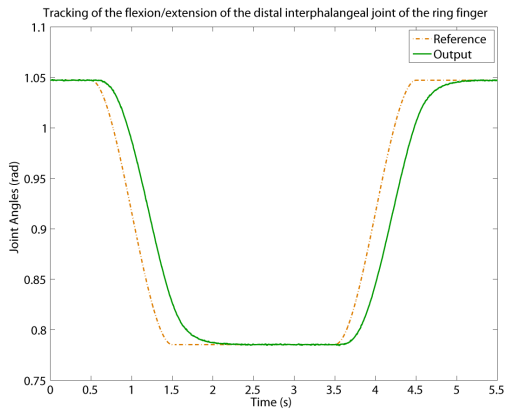

Tracking of the flexion/extension of the proximal interphalangeal joint of the middle finger
Reference
Output
Joint Angles (rad)
Time (s)

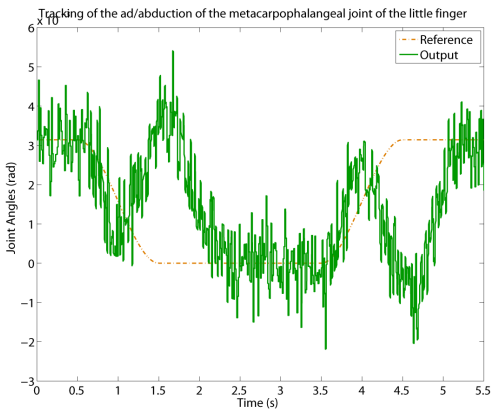

Tracking of the flexion/extension of the distal interphalangeal joint of the midle finger
Reference
Output
Joint Angles (rad)
Time (s)

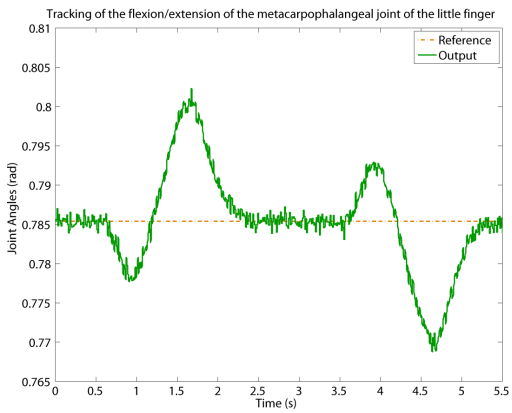

Tracking of the ad/abduction of the metacarpophalangeal joint of the ring finger
x 10
Reference
Output
Joint Angles (rad)
Time (s)

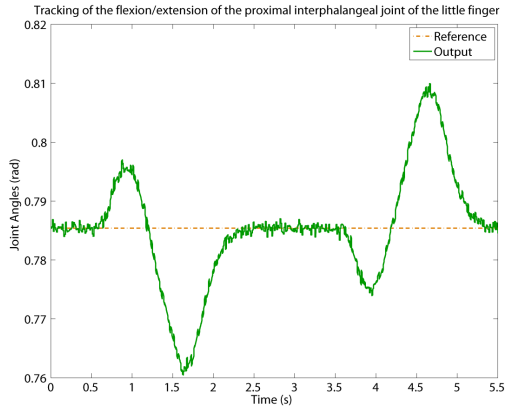

Tracking of the flexion/extension of the metacarpophalangeal joint of the ring finger
Reference
Output
Joint Angles (rad)
Time (s)

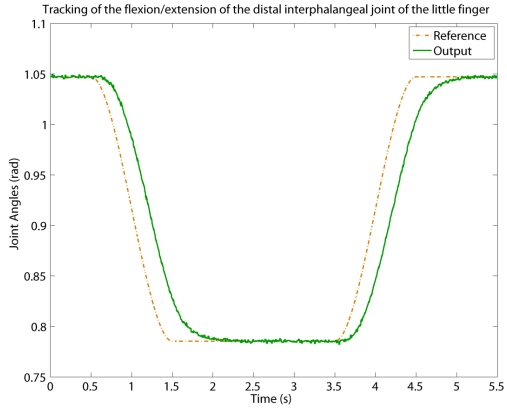

Tracking of the flexion/extension of the proximal interphalangeal joint of the ring finger
Reference
Output
Joint Angles (rad)
Time (s)

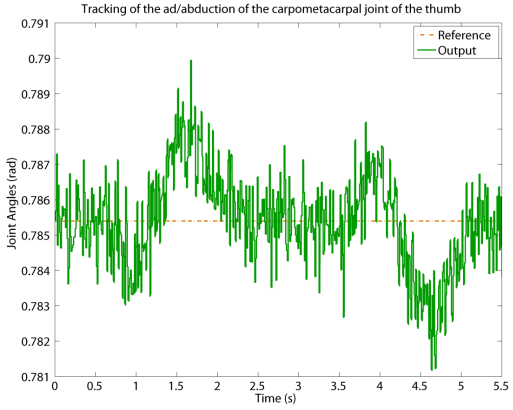

Tracking of the flexion/extension of the distal interphalangeal joint of the ring finger
Reference
Output
Joint Angles (rad)
Time (s)

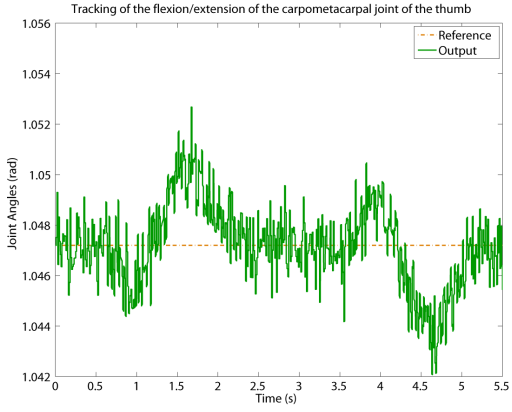

Tracking of the ad/abduction of the metacarpophalangeal joint of the little finger
x 10
Reference
Output
Joint Angles (rad)
Time (s)

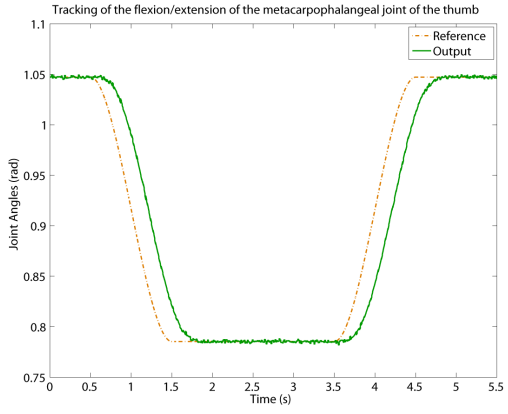

Tracking of the flexion/extension of the metacarpophalangeal joint of the little finger
Reference
Output
Joint Angles (rad)
Time (s)

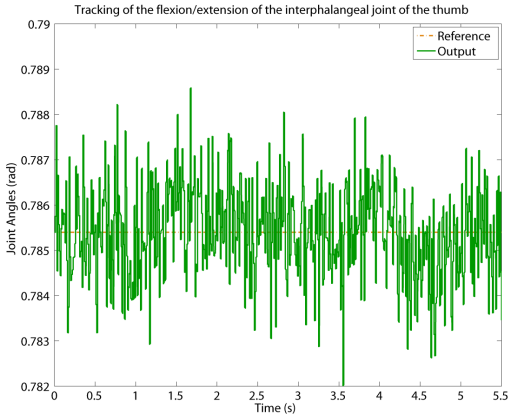

Tracking of the flexion/extension of the proximal interphalangeal joint of the little finger
Reference
Output
Joint Angles (rad)
Time (s)


Tracking of the flexion/extension of the distal interphalangeal joint of the little finger
Reference
Output
Joint Angles (rad)
Time (s)

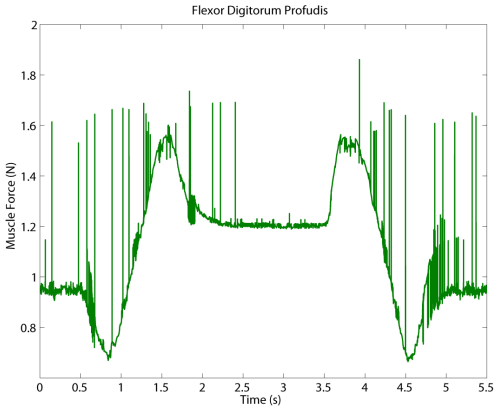

Tracking of the ad/abduction of the carpometacarpal joint of the thumb
Reference
Output
Joint Angles (rad)
Time (s)

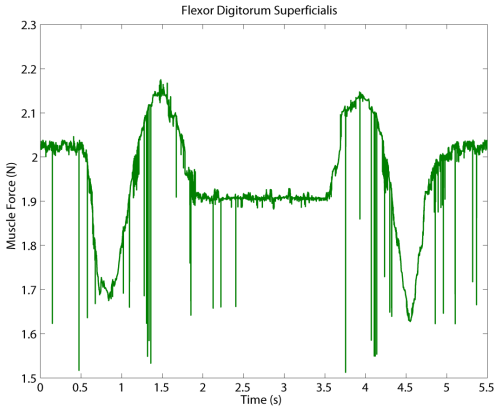

Tracking of the flexion/extension of the carpometacarpal joint of the thumb
Reference
Output
Joint Angles (rad)
Time (s)

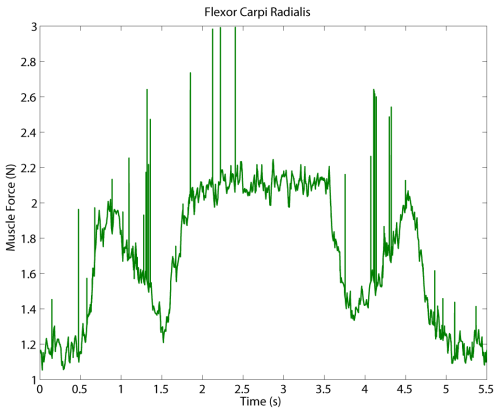

Tracking of the flexion/extension of the metacarpophalangeal joint of the thumb
Reference
Output
Joint Angles (rad)
Time (s)

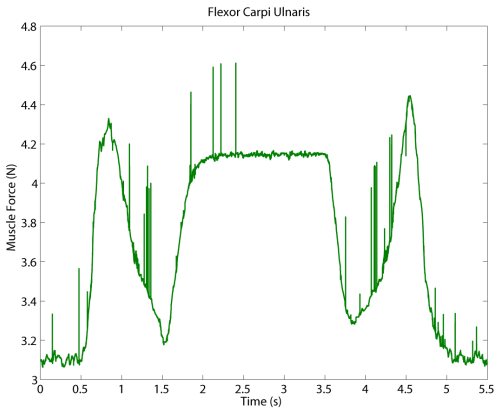

Tracking of the flexion/extension of the interphalangeal joint of the thumb
Reference
Output
Joint Angles (rad)
Time (s)

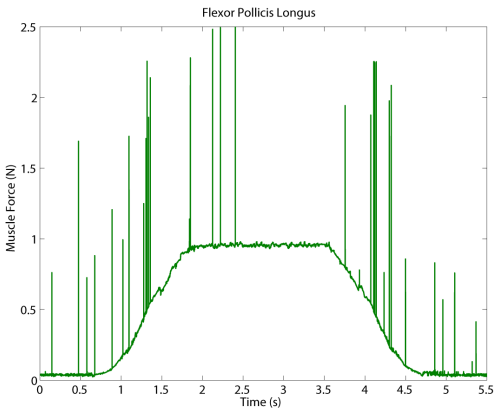
A.1.2 Muscle Forces (Control Actions)

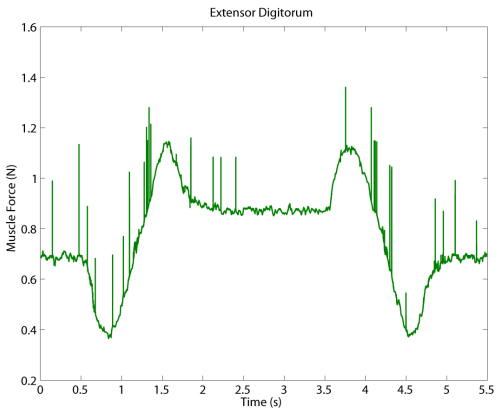

Flexor Digitorum Profudis
Muscle Force (N)
Time (s)

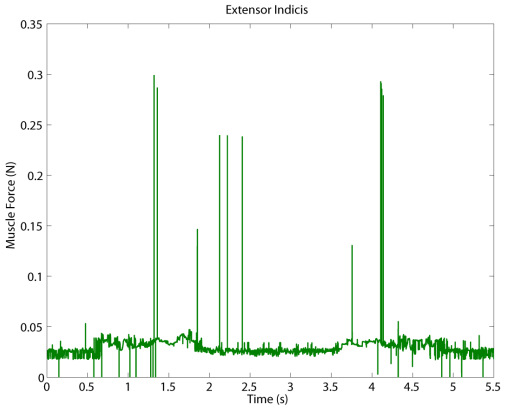

Flexor Digitorum Superficialis
Muscle Force (N)
Time (s)

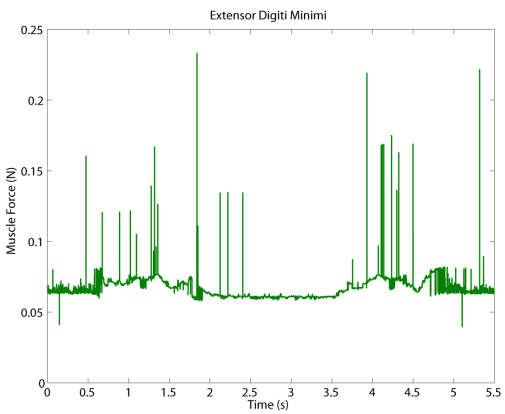

Flexor Carpi Radialis
Muscle Force (N)
Time (s)

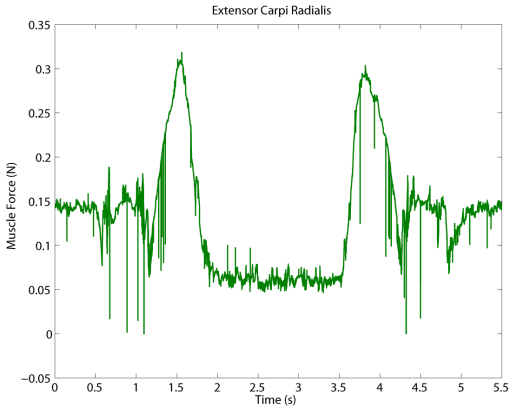

Flexor Carpi Ulnaris
Muscle Force (N)
Time (s)

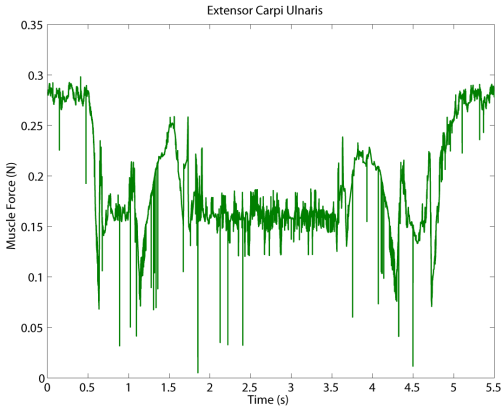

Flexor Pollicis Longus
Muscle Force (N)
Time (s)

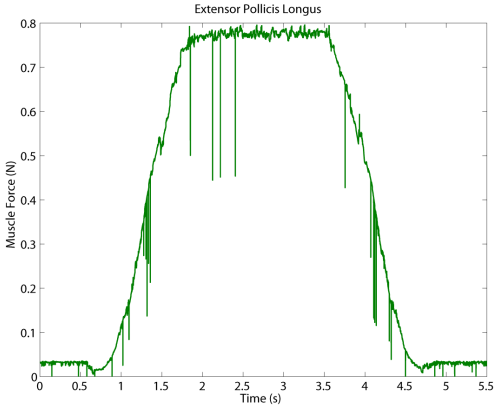

Extensor Digitorum
Muscle Force (N)
Time (s)

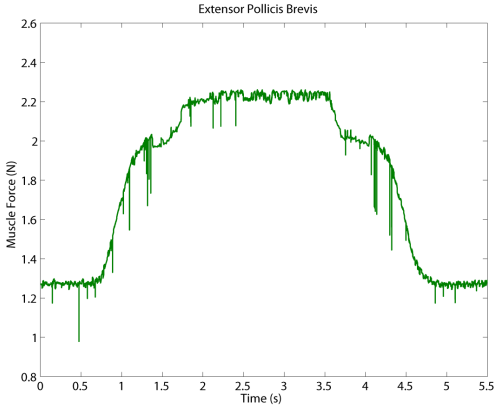

Extensor Indicis
Muscle Force (N)
Time (s)

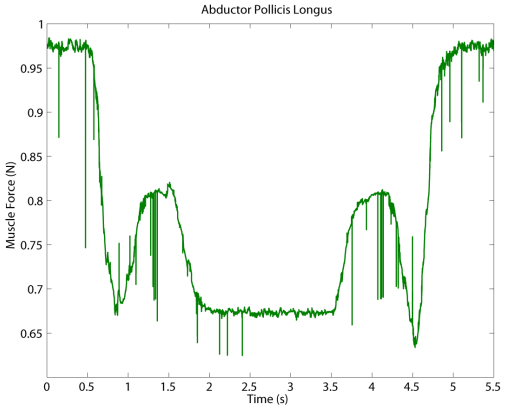

Extensor Digiti Minimi
Muscle Force (N)
Time (s)

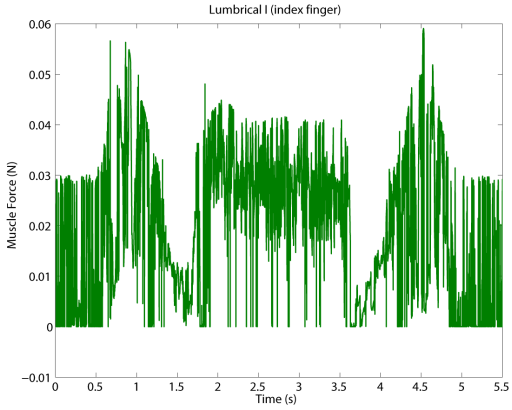

Extensor Carpi Radialis
Muscle Force (N)
Time (s)

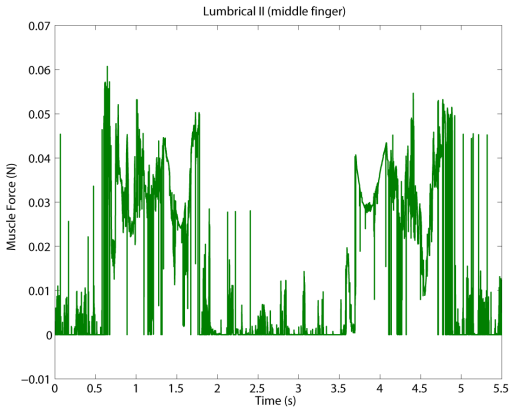

Extensor Carpi Ulnaris
Muscle Force (N)
Time (s)

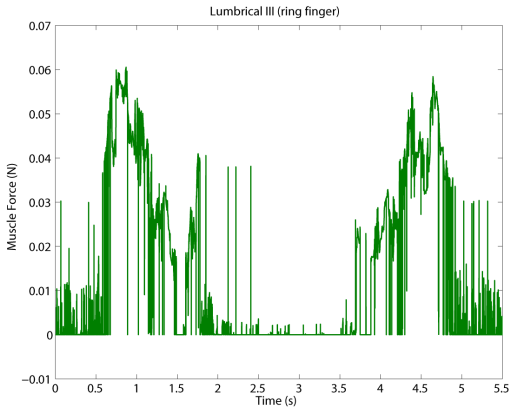

Extensor Pollicis Longus
Muscle Force (N)
Time (s)

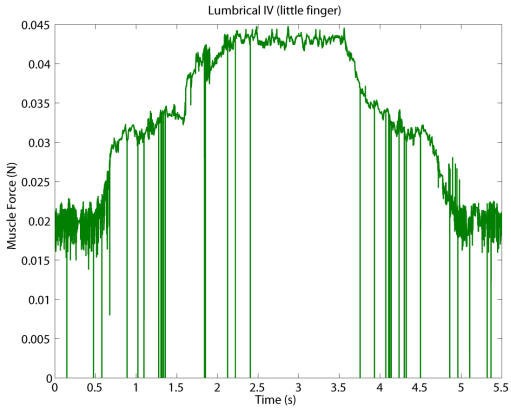

Extensor Pollicis Brevis
Muscle Force (N)
Time (s)

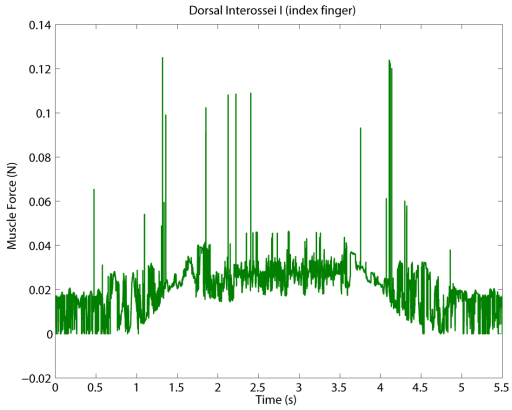

Abductor Pollicis Longus
Muscle Force (N)
Time (s)

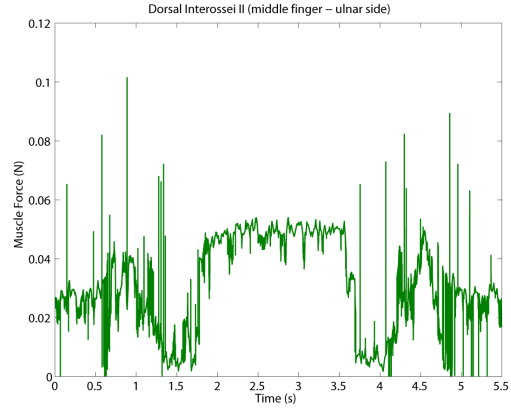

Lumbrical I (index finger)
Muscle Force (N)
Time (s)

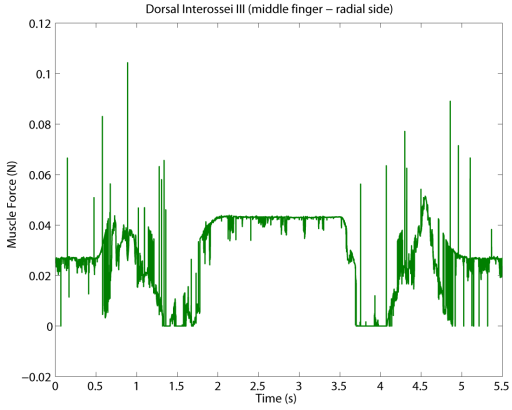

Lumbrical II (middle finger)
Muscle Force (N)
Time (s)

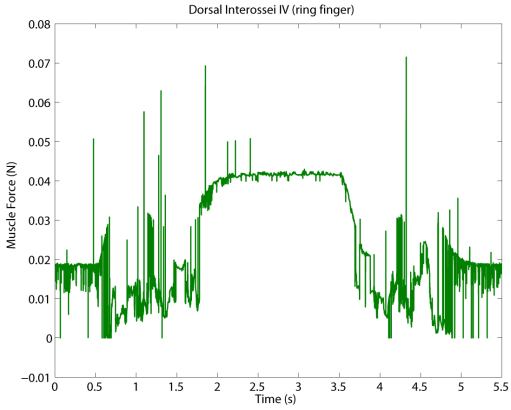

Lumbrical III (ring finger)
Muscle Force (N)
Time (s)

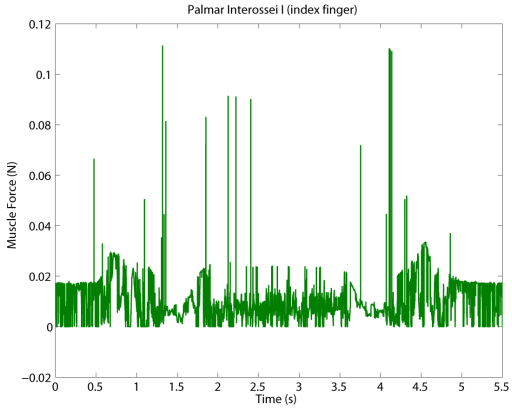

Lumbrical IV (little finger)
Muscle Force (N)
Time (s)

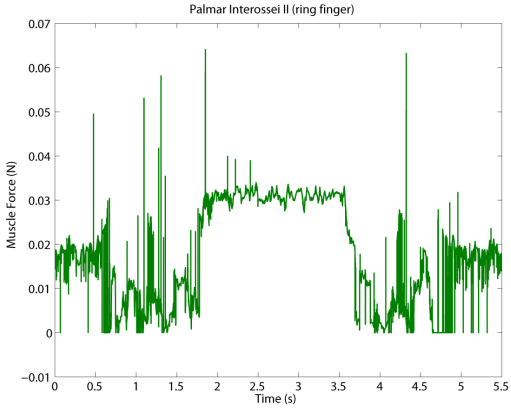

Dorsal Interossei I (index finger)
Muscle Force (N)
Time (s)

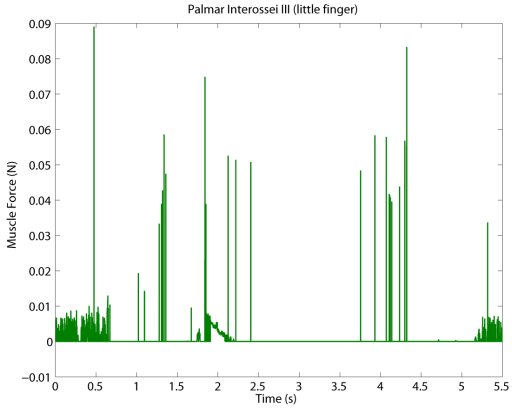

Dorsal Interossei II (middle finger – ulnar side)
Muscle Force (N)
Time (s)

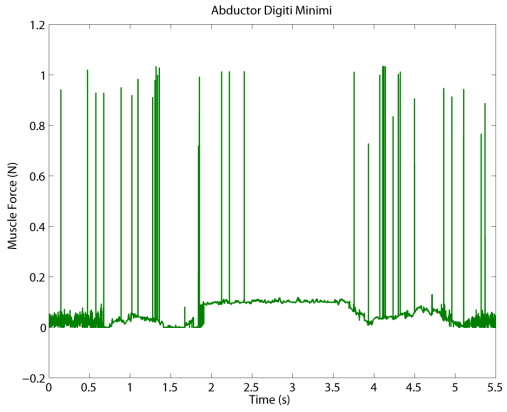

Dorsal Interossei III (middle finger – radial side)
Muscle Force (N)
Time (s)

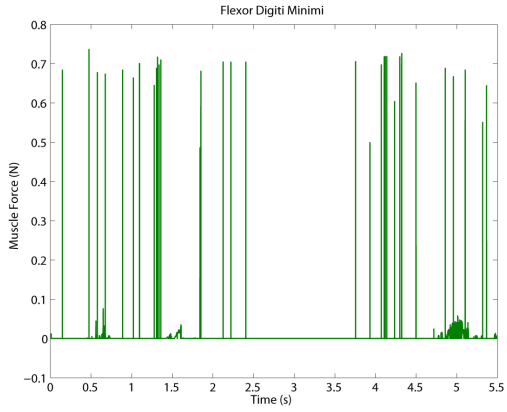

Dorsal Interossei IV (ring finger)
Muscle Force (N)
Time (s)

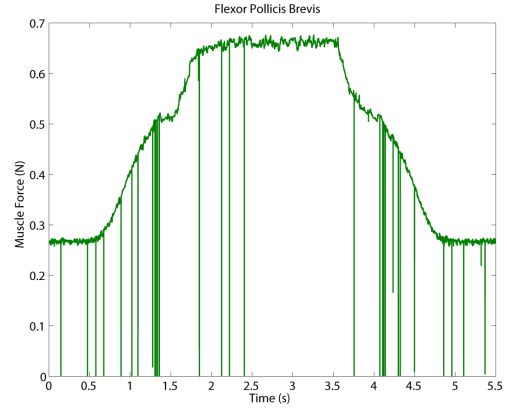

Palmar Interossei I (index finger)
Muscle Force (N)
Time (s)

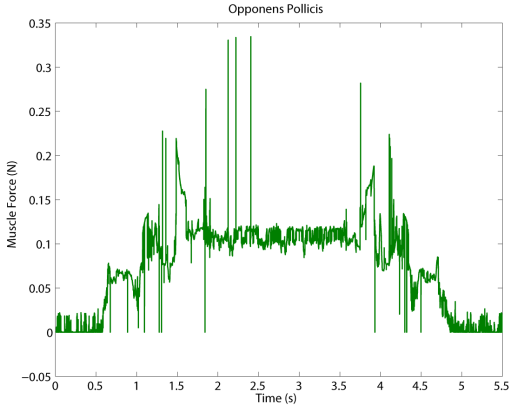

Palmar Interossei II (ring finger)
Muscle Force (N)
Time (s)

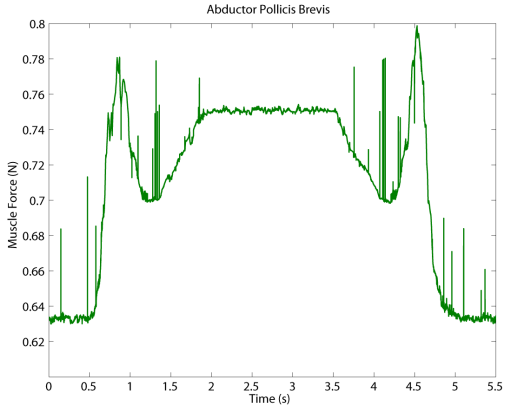

Palmar Interossei III (little finger)
Muscle Force (N)
Time (s)

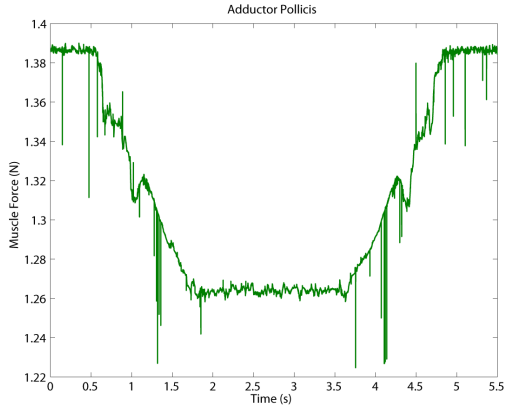

Abductor Digiti Minimi
Muscle Force (N)
Time (s)


Flexor Digiti Minimi
Muscle Force (N)
Time (s)


Flexor Pollicis Brevis
Muscle Force (N)
Time (s)

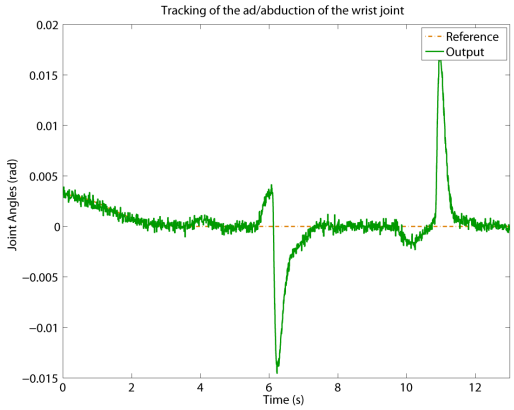

Opponens Pollicis
Muscle Force (N)
Time (s)

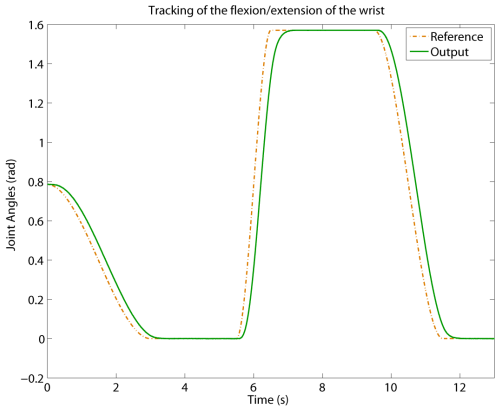

Abductor Pollicis Brevis
Muscle Force (N)
Time (s)

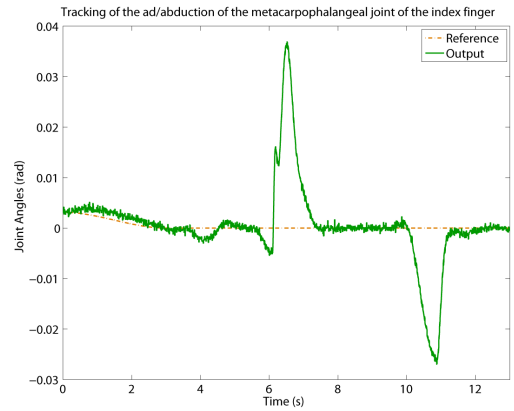

Adductor Pollicis
Muscle Force (N)
Time (s)

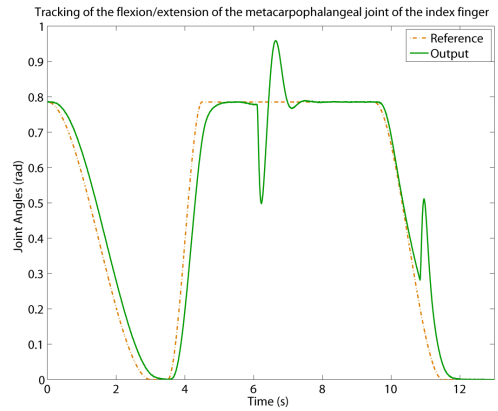
A.2 Results for the Grip Movement with an External Force

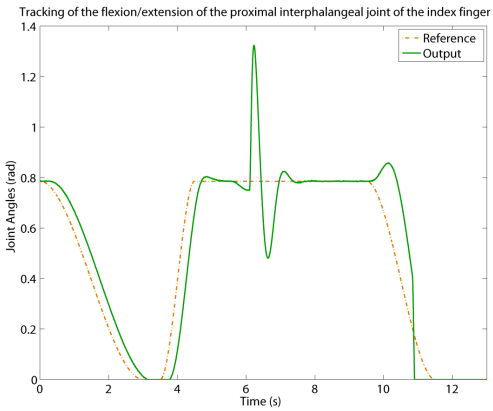
A.2.1 Output Signals

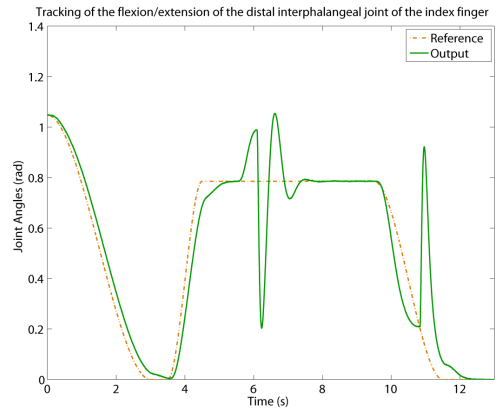

Tracking of the ad/abduction of the wrist joint
Reference
Output
Joint Angles (rad)
Time (s)

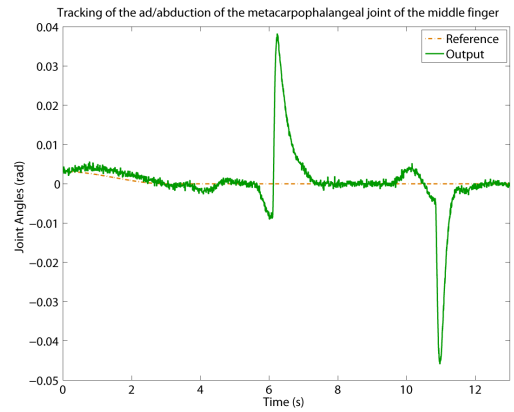

Tracking of the flexion/extension of the wrist
Reference
Output
Joint Angles (rad)
Time (s)

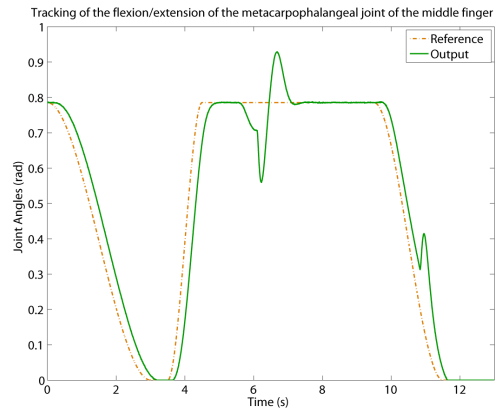

Tracking of the ad/abduction of the metacarpophalangeal joint of the index finger
Reference
Output
Joint Angles (rad)
Time (s)

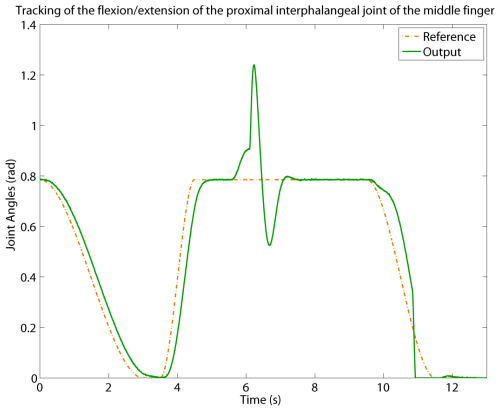

Tracking of the flexion/extension of the metacarpophalangeal joint of the index finger
Reference
Output
Joint Angles (rad)
Time (s)

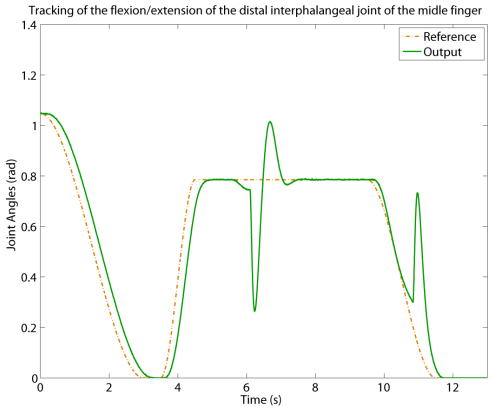

Tracking of the flexion/extension of the proximal interphalangeal joint of the index finger
Reference
Output
Joint Angles (rad)
Time (s)

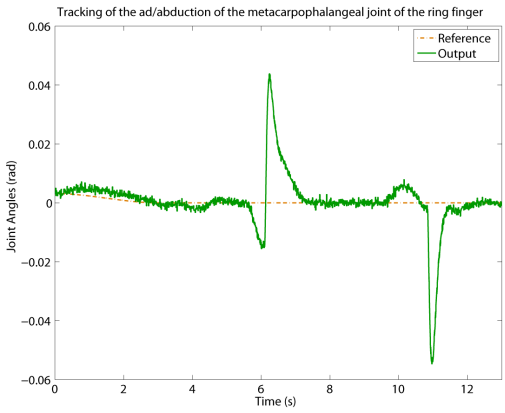

Tracking of the flexion/extension of the distal interphalangeal joint of the index finger
Reference
Output
Joint Angles (rad)
Time (s)

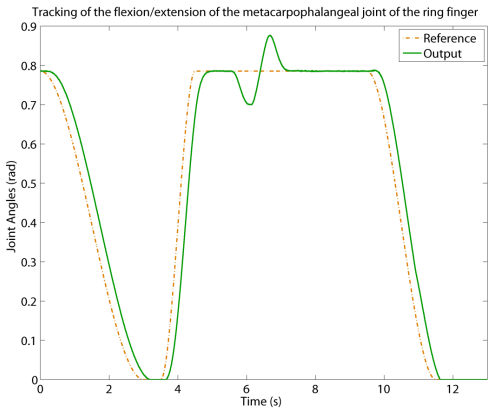

Tracking of the ad/abduction of the metacarpophalangeal joint of the middle finger
Reference
Output
Joint Angles (rad)
Time (s)

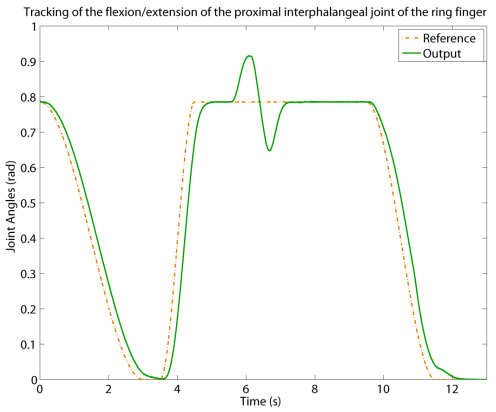

Tracking of the flexion/extension of the metacarpophalangeal joint of the middle finger
Reference
Output
Joint Angles (rad)
Time (s)

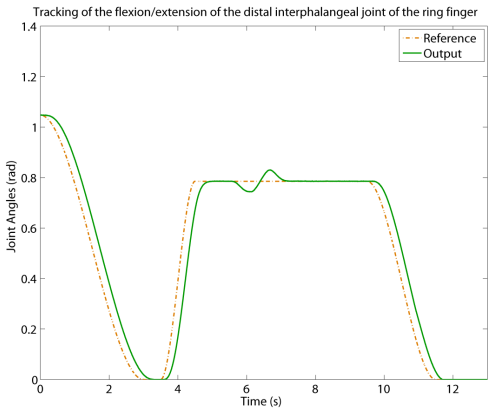

Tracking of the flexion/extension of the proximal interphalangeal joint of the middle finger
Reference
Output
Joint Angles (rad)
Time (s)

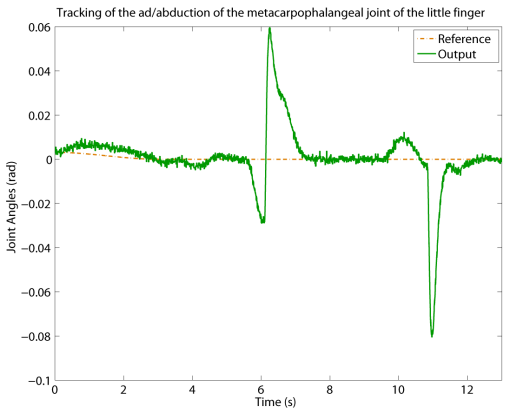

Tracking of the flexion/extension of the distal interphalangeal joint of the midle finger
Reference
Output
Joint Angles (rad)
Time (s)

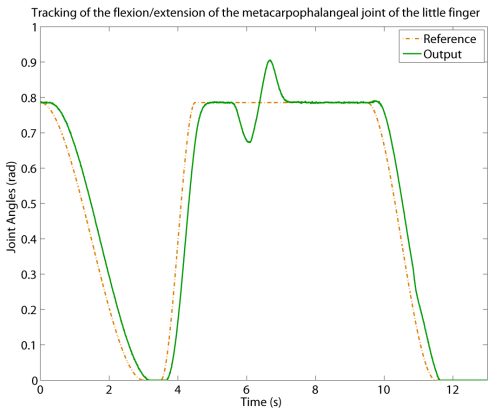

Tracking of the ad/abduction of the metacarpophalangeal joint of the ring finger
Reference
Output
Joint Angles (rad)
Time (s)

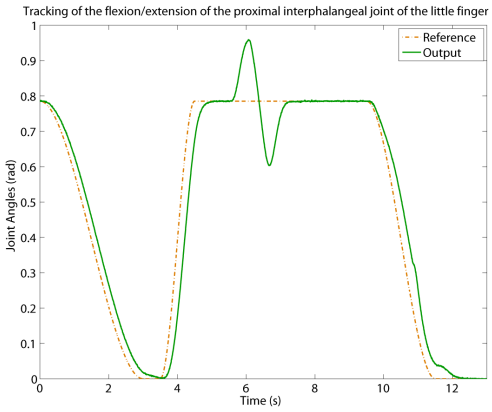

Tracking of the flexion/extension of the metacarpophalangeal joint of the ring finger
Reference
Output
Joint Angles (rad)
Time (s)

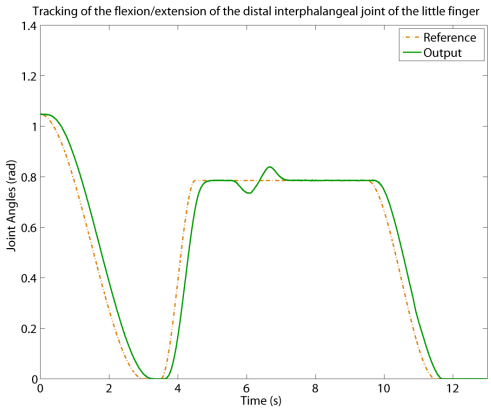

Tracking of the flexion/extension of the proximal interphalangeal joint of the ring finger
Reference
Output
Joint Angles (rad)
Time (s)

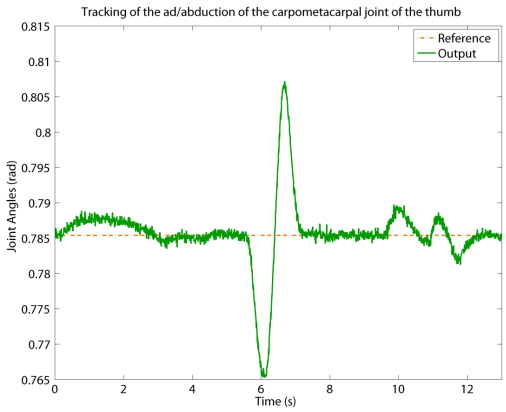

Tracking of the flexion/extension of the distal interphalangeal joint of the ring finger
Reference
Output
Joint Angles (rad)
Time (s)

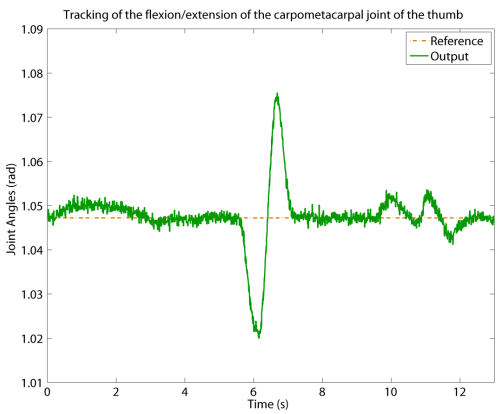

Tracking of the ad/abduction of the metacarpophalangeal joint of the little finger
Reference
Output
Joint Angles (rad)
Time (s)

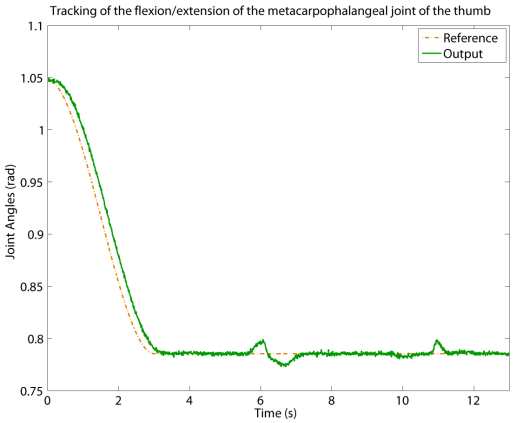

Tracking of the flexion/extension of the metacarpophalangeal joint of the little finger
Reference
Output
Joint Angles (rad)
Time (s)

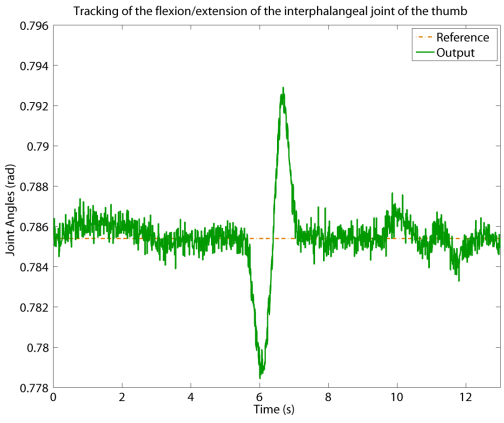

Tracking of the flexion/extension of the proximal interphalangeal joint of the little finger
Reference
Output
Joint Angles (rad)
Time (s)


Tracking of the flexion/extension of the distal interphalangeal joint of the little finger
Reference
Output
Joint Angles (rad)
Time (s)

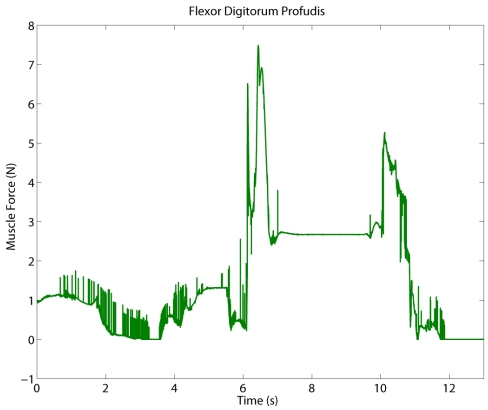

Tracking of the ad/abduction of the carpometacarpal joint of the thumb
Reference
Output
Joint Angles (rad)
Time (s)

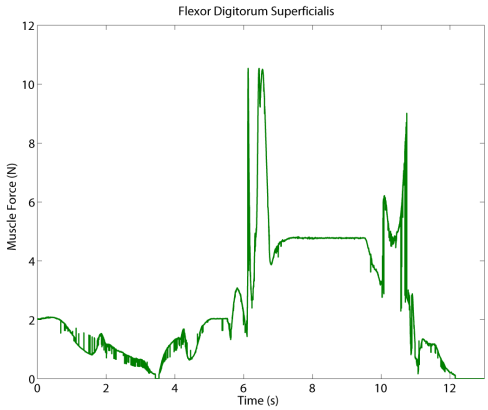

Tracking of the flexion/extension of the carpometacarpal joint of the thumb
Reference
Output
Joint Angles (rad)
Time (s)

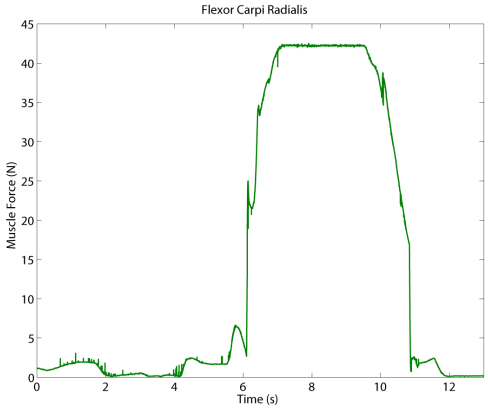

Tracking of the flexion/extension of the metacarpophalangeal joint of the thumb
Reference
Output
Joint Angles (rad)
Time (s)

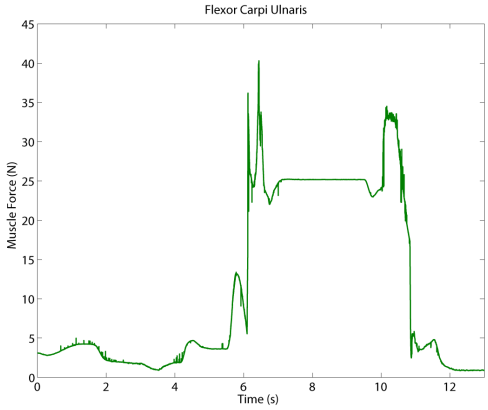

Tracking of the flexion/extension of the interphalangeal joint of the thumb
Reference
Output
Joint Angles (rad)
Time (s)

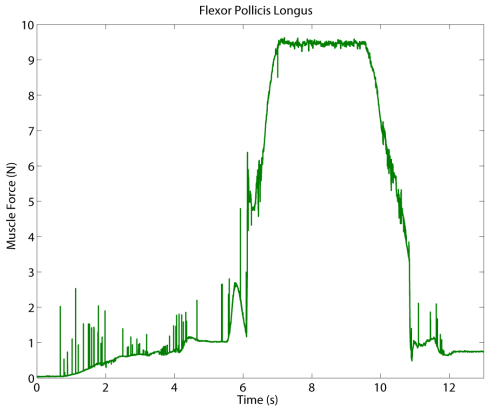
A.2.2 Muscle Forces (Control Actions)

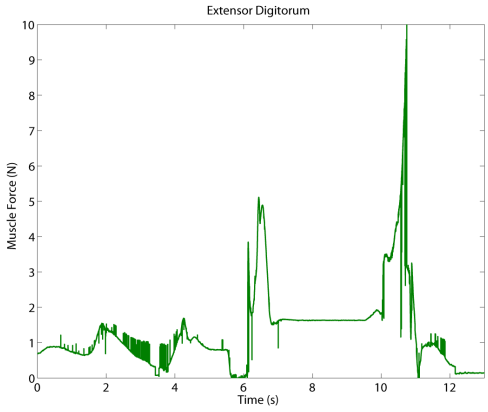

Flexor Digitorum Profudis
Muscle Force (N)
Time (s)

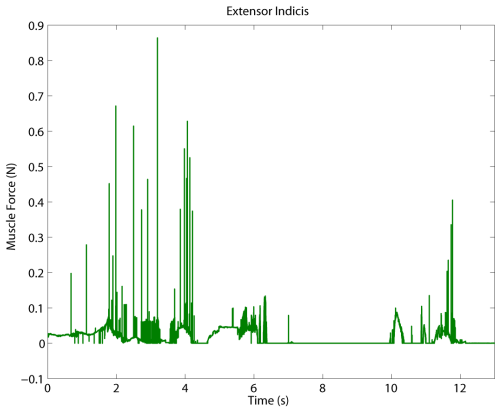

Flexor Digitorum Superficialis
Muscle Force (N)
Time (s)

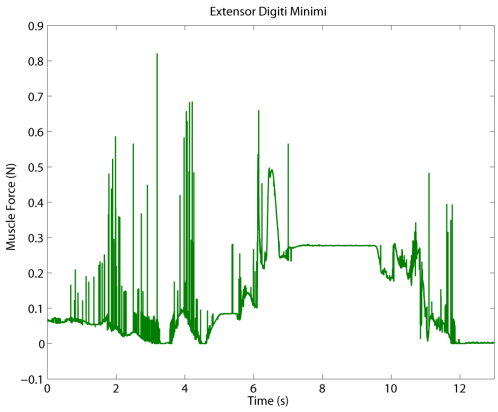

Flexor Carpi Radialis
Muscle Force (N)
Time (s)

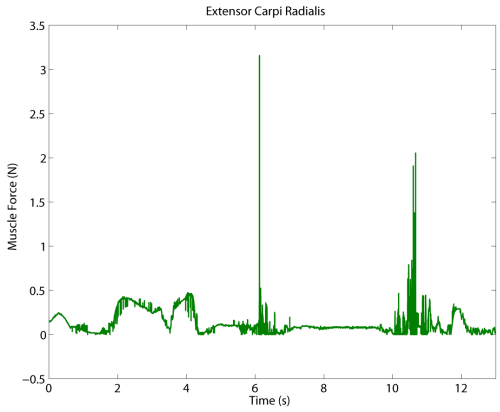

Flexor Carpi Ulnaris
Muscle Force (N)
Time (s)

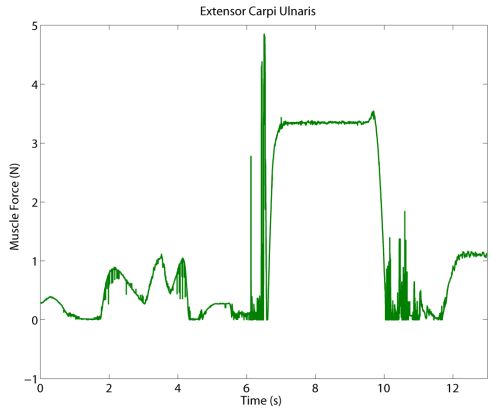

Flexor Pollicis Longus
Muscle Force (N)
Time (s)

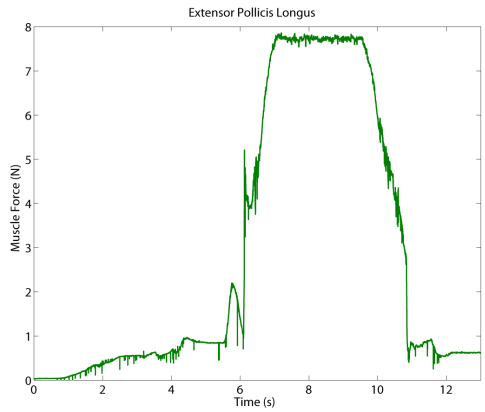

Extensor Digitorum
Muscle Force (N)
Time (s)

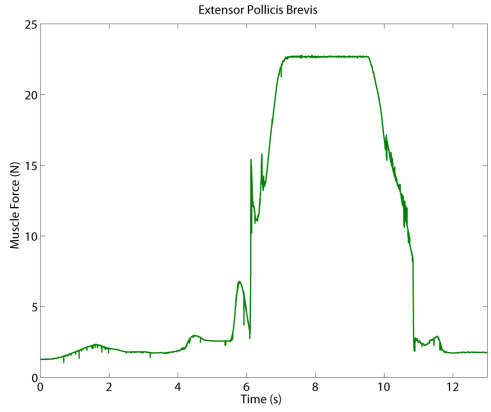

Extensor Indicis
Muscle Force (N)
Time (s)

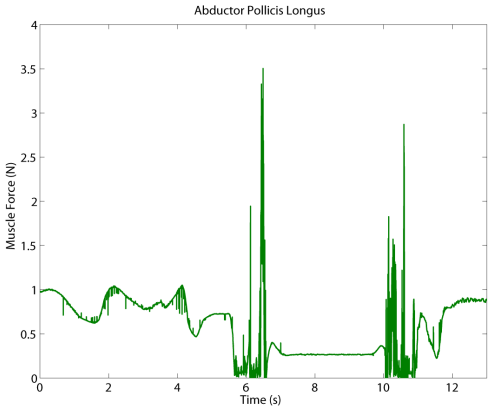

Extensor Digiti Minimi
Muscle Force (N)
Time (s)

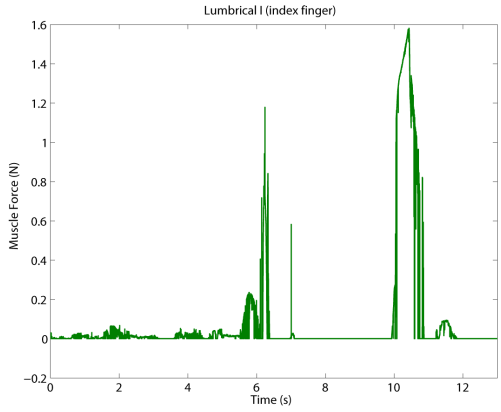

Extensor Carpi Radialis
Muscle Force (N)
Time (s)

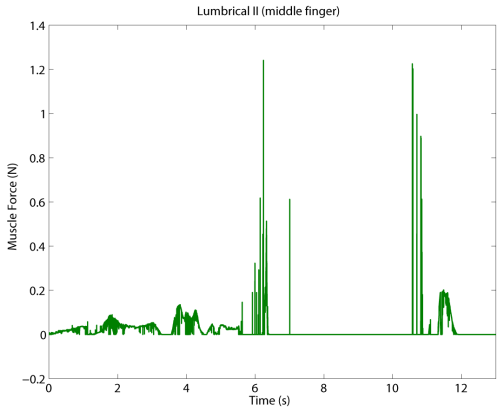

Extensor Carpi Ulnaris
Muscle Force (N)
Time (s)

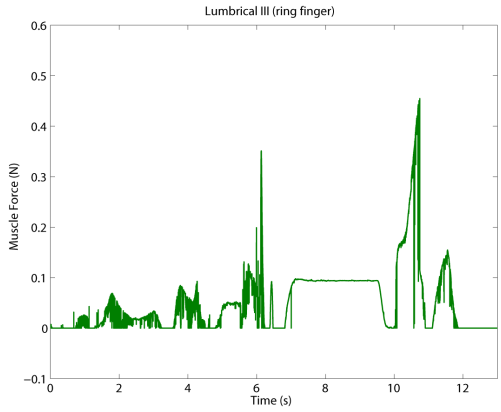

Extensor Pollicis Longus
Muscle Force (N)
Time (s)

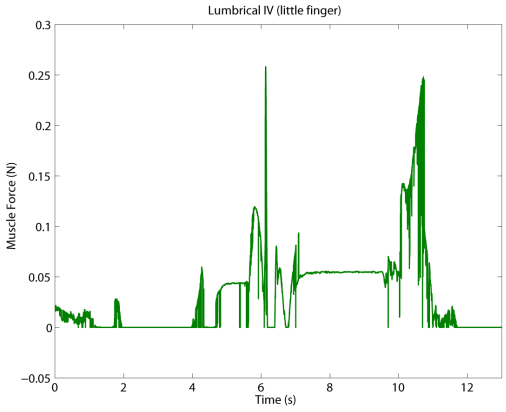

Extensor Pollicis Brevis
Muscle Force (N)
Time (s)

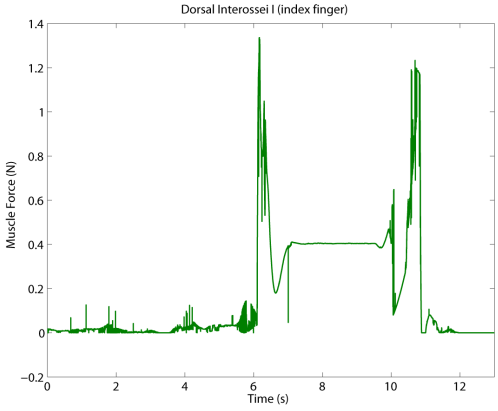

Abductor Pollicis Longus
Muscle Force (N)
Time (s)

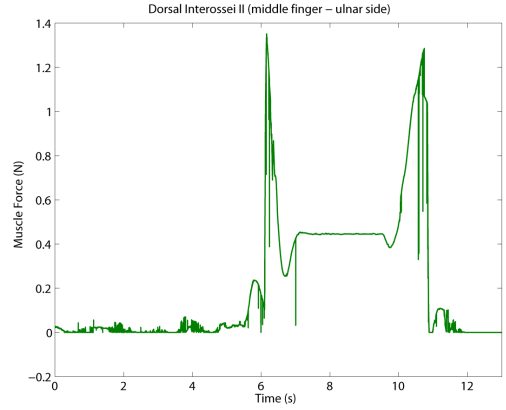

Lumbrical I (index finger)
Muscle Force (N)
Time (s)

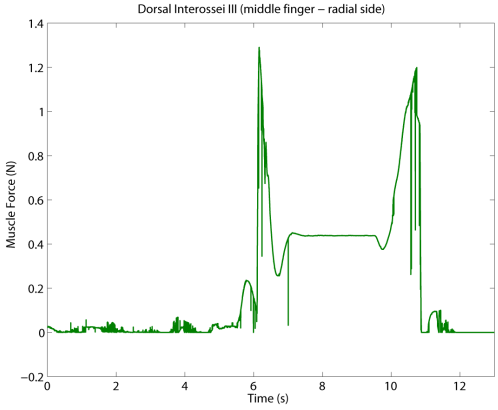

Lumbrical II (middle finger)
Muscle Force (N)
Time (s)

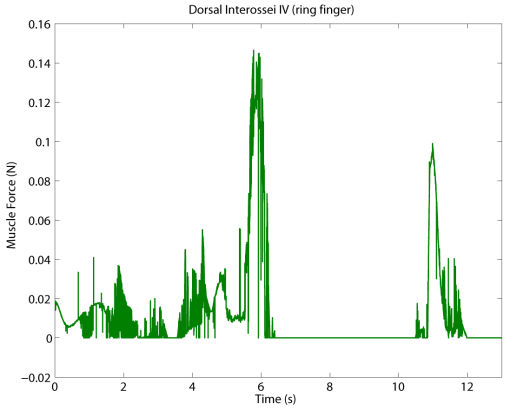

Lumbrical III (ring finger)
Muscle Force (N)
Time (s)

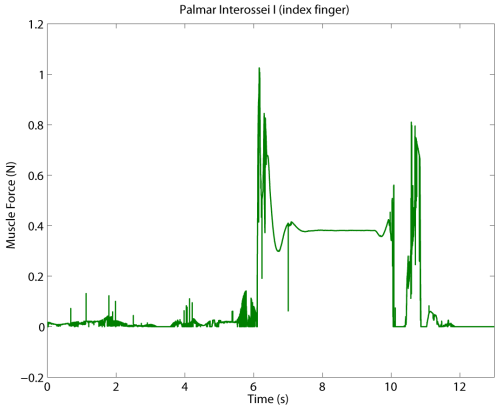

Lumbrical IV (little finger)
Muscle Force (N)
Time (s)

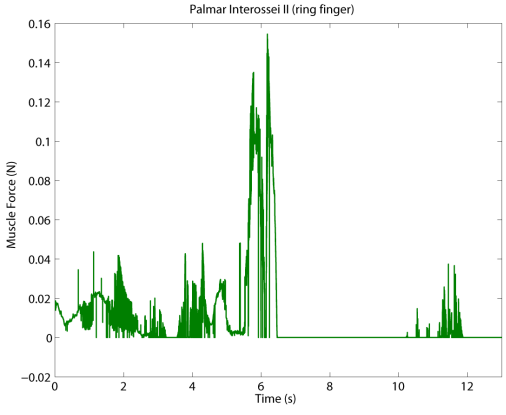

Dorsal Interossei I (index finger)
Muscle Force (N)
Time (s)

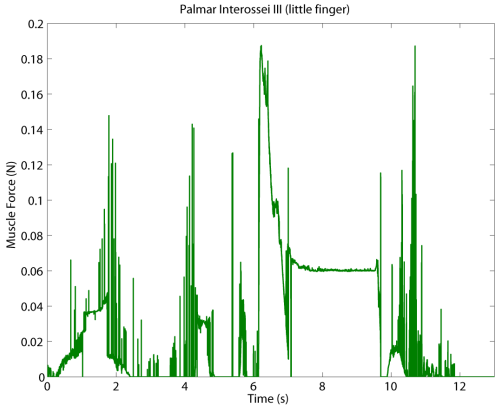

Dorsal Interossei II (middle finger – ulnar side)
Muscle Force (N)
Time (s)

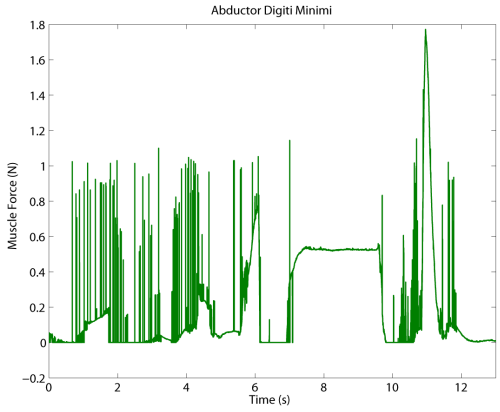

Dorsal Interossei III (middle finger – radial side)
Muscle Force (N)
Time (s)

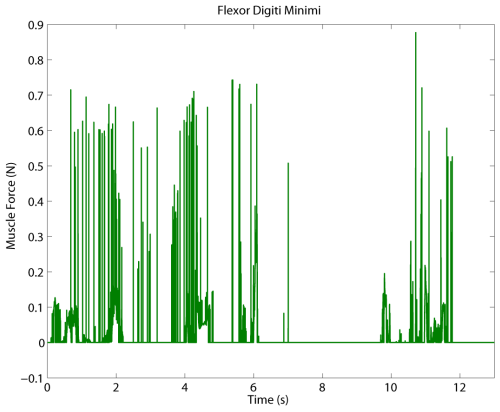

Dorsal Interossei IV (ring finger)
Muscle Force (N)
Time (s)

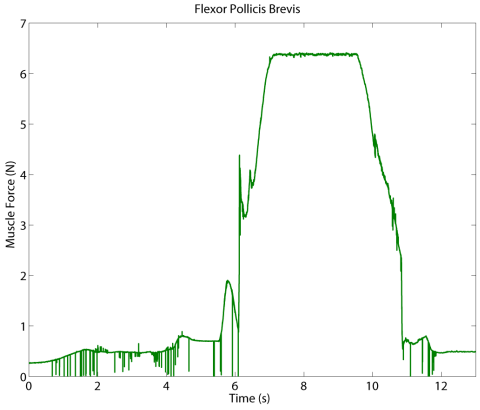

Palmar Interossei I (index finger)
Muscle Force (N)
Time (s)

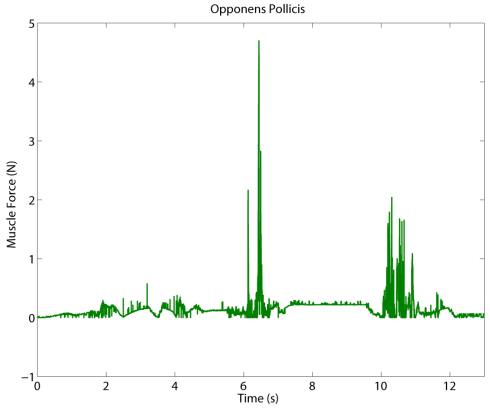

Palmar Interossei II (ring finger)
Muscle Force (N)
Time (s)

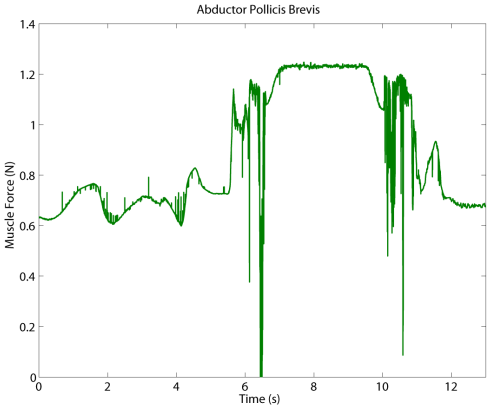

Palmar Interossei III (little finger)
Muscle Force (N)
Time (s)

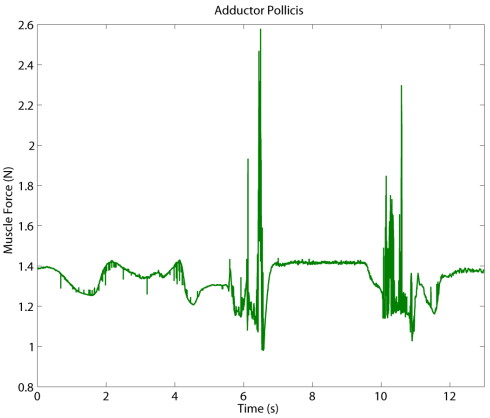

Abductor Digiti Minimi
Muscle Force (N)
Time (s)


Flexor Digiti Minimi
Muscle Force (N)
Time (s)


Flexor Pollicis Brevis
Muscle Force (N)
Time (s)

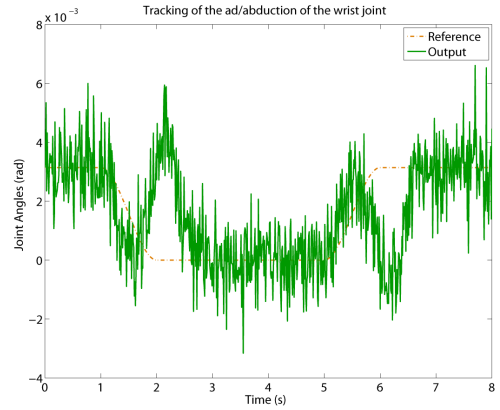

Opponens Pollicis
Muscle Force (N)
Time (s)

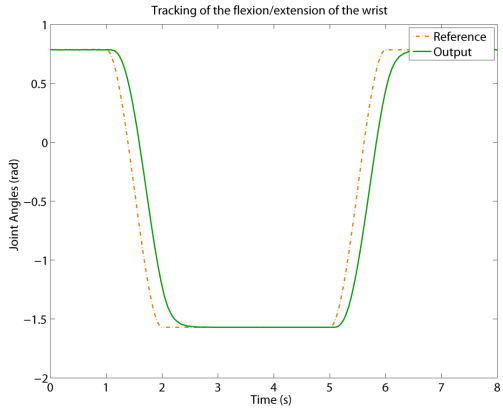

Abductor Pollicis Brevis
Muscle Force (N)
Time (s)

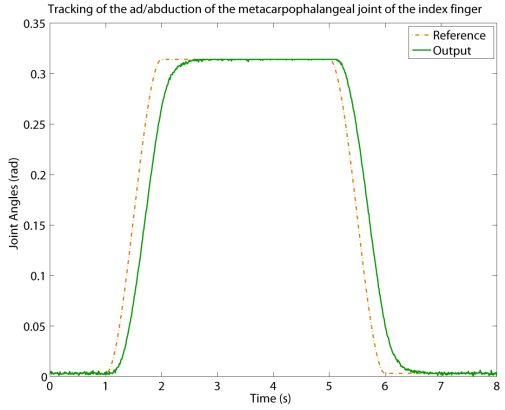

Adductor Pollicis
Muscle Force (N)
Time (s)

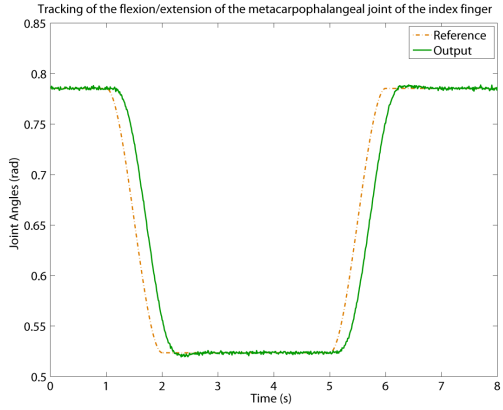
A.3 Results for the Cup Movement

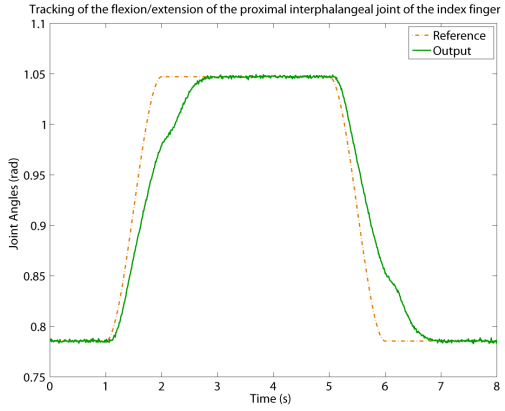
A.3.1 Output Signals

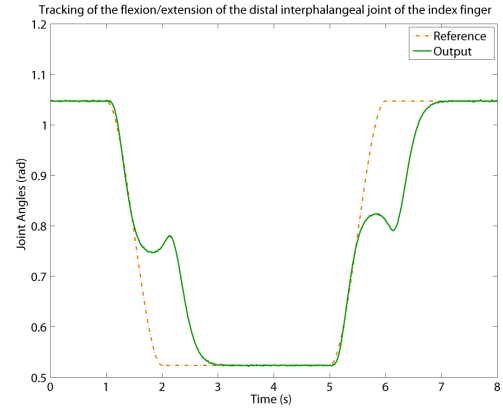

Tracking of the ad/abduction of the wrist joint
x 10^-3
Reference
Output
Joint Angles (rad)
Time (s)

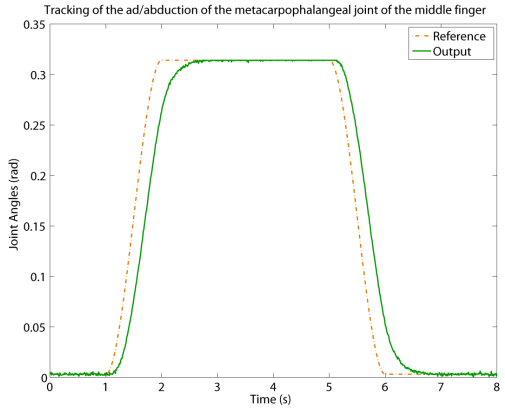

Tracking of the flexion/extension of the wrist
Reference
Output
Joint Angles (rad)
Time (s)

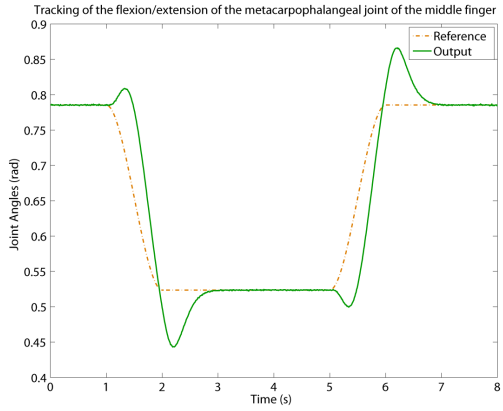

Tracking of the ad/abduction of the metacarpophalangeal joint of the index finger
Reference
Output
Joint Angles (rad)
Time (s)

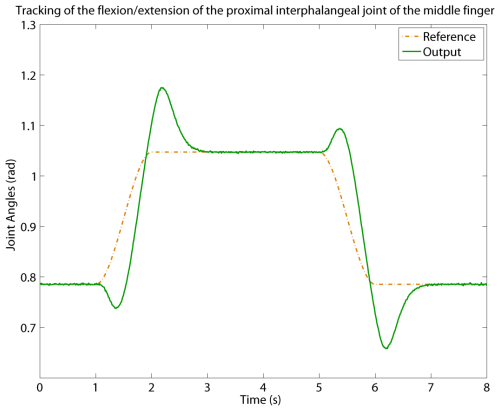

Tracking of the flexion/extension of the metacarpophalangeal joint of the index finger
Reference
Output
Joint Angles (rad)
Time (s)

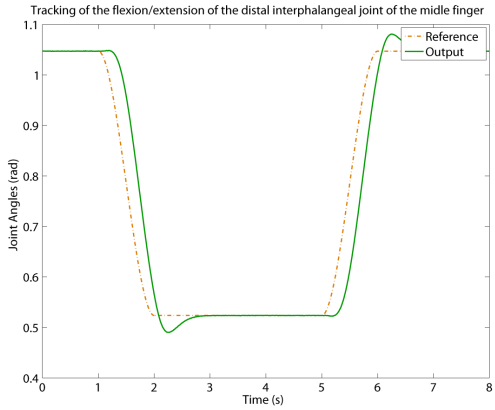

Tracking of the flexion/extension of the proximal interphalangeal joint of the index finger
Reference
Output
Joint Angles (rad)
Time (s)

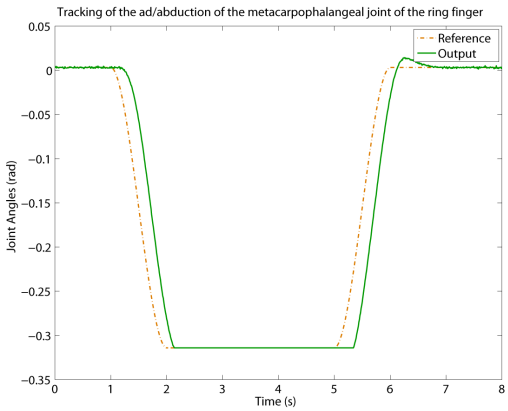

Tracking of the flexion/extension of the distal interphalangeal joint of the index finger
Reference
Output
Joint Angles (rad)
Time (s)

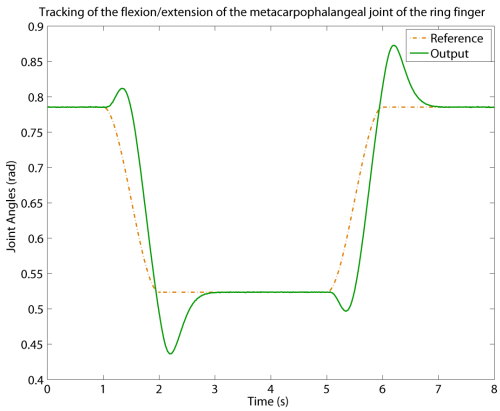

Tracking of the ad/abduction of the metacarpophalangeal joint of the middle finger
Reference
Output
Joint Angles (rad)
Time (s)

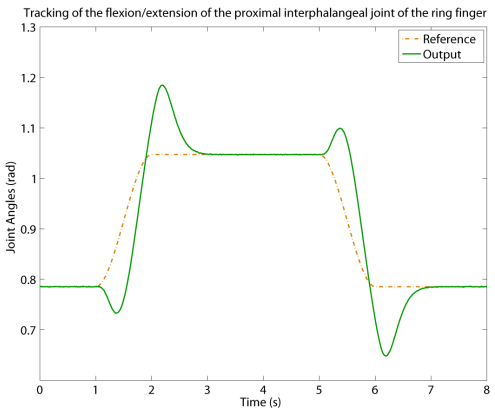

Tracking of the flexion/extension of the metacarpophalangeal joint of the middle finger
Reference
Output
Joint Angles (rad)
Time (s)

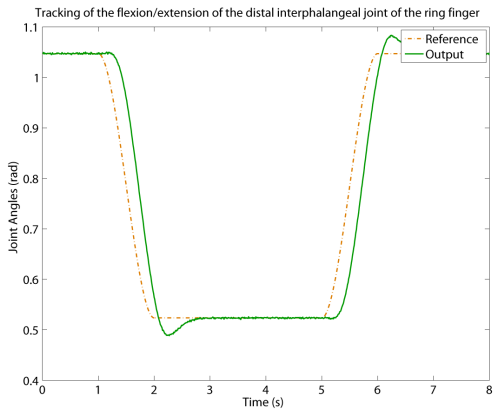

Tracking of the flexion/extension of the proximal interphalangeal joint of the middle finger
Reference
Output
Joint Angles (rad)
Time (s)

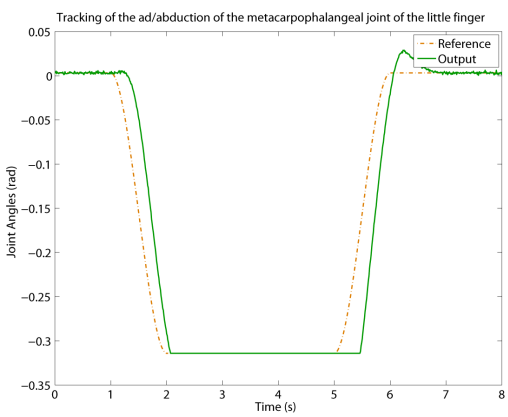

Tracking of the flexion/extension of the distal interphalangeal joint of the midle finger
Reference
Output
Joint Angles (rad)
Time (s)

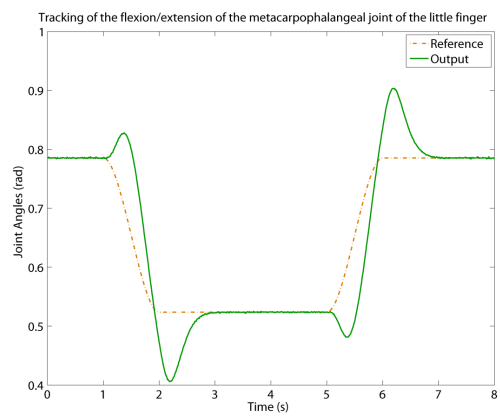

Tracking of the ad/abduction of the metacarpophalangeal joint of the ring finger
Reference
Output
Joint Angles (rad)
Time (s)

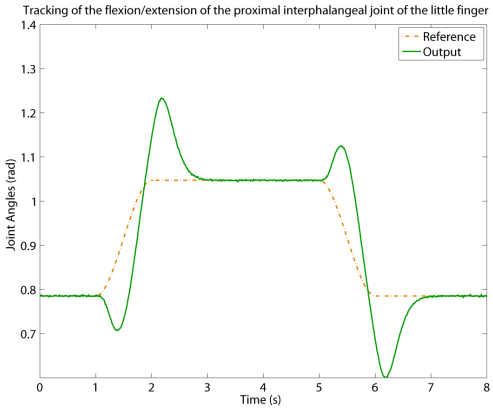

Tracking of the flexion/extension of the metacarpophalangeal joint of the ring finger
Reference
Output
Joint Angles (rad)
Time (s)

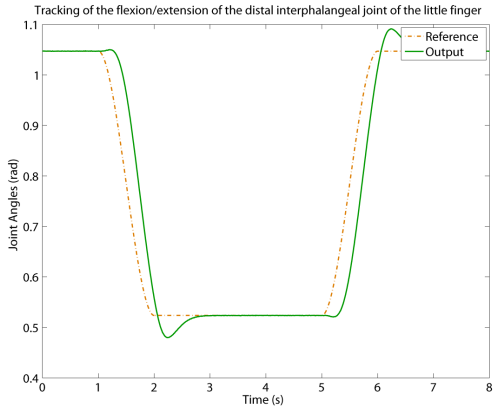

Tracking of the flexion/extension of the proximal interphalangeal joint of the ring finger
Reference
Output
Joint Angles (rad)
Time (s)

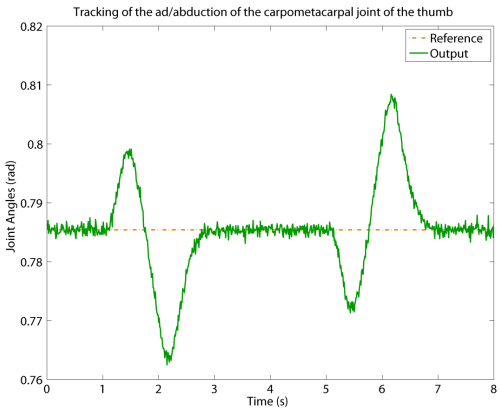

Tracking of the flexion/extension of the distal interphalangeal joint of the ring finger
Reference
Output
Joint Angles (rad)
Time (s)

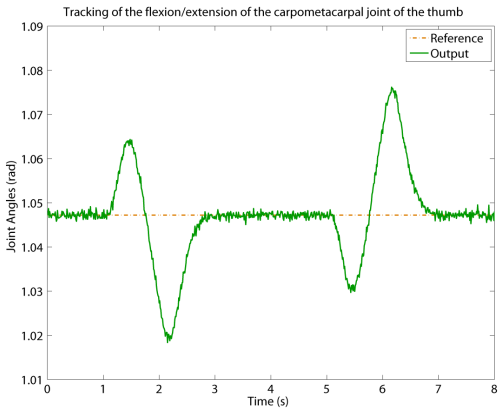

Tracking of the ad/abduction of the metacarpophalangeal joint of the little finger
Reference
Output
Joint Angles (rad)
Time (s)

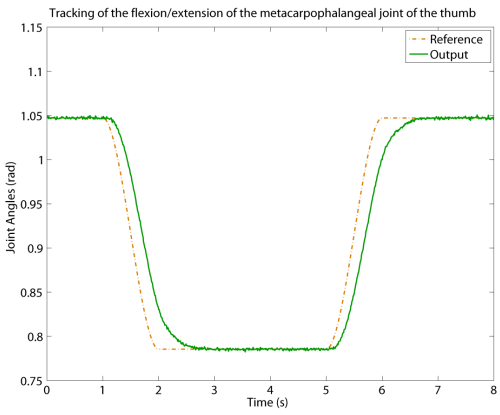

Tracking of the flexion/extension of the metacarpophalangeal joint of the little finger
Reference
Output
Joint Angles (rad)
Time (s)

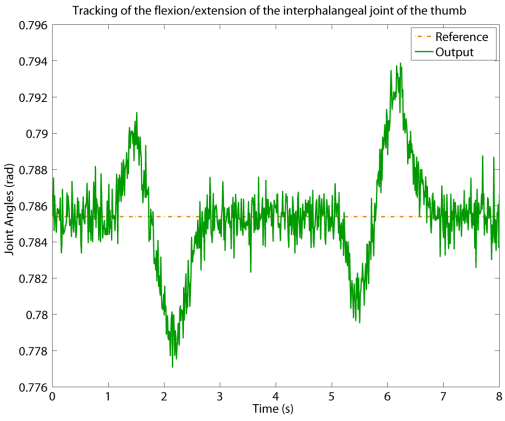

Tracking of the flexion/extension of the proximal interphalangeal joint of the little finger
Reference
Output
Joint Angles (rad)
Time (s)


Tracking of the flexion/extension of the distal interphalangeal joint of the little finger
Reference
Output
Joint Angles (rad)
Time (s)

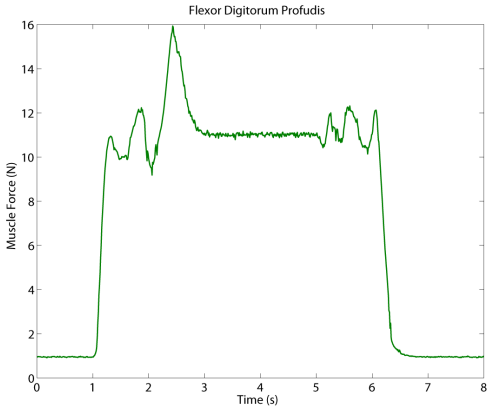

Tracking of the ad/abduction of the carpometacarpal joint of the thumb
Reference
Output
Joint Angles (rad)
Time (s)

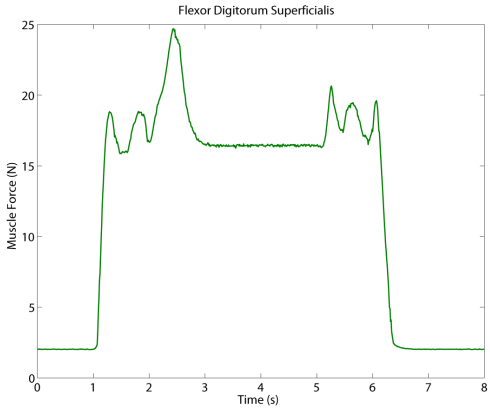

Tracking of the flexion/extension of the carpometacarpal joint of the thumb
Reference
Output
Joint Angles (rad)
Time (s)

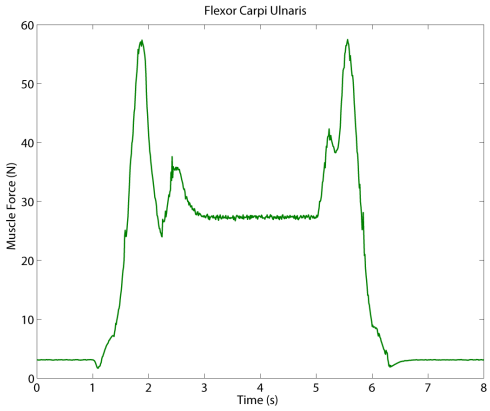

Tracking of the flexion/extension of the metacarpophalangeal joint of the thumb
Reference
Output
Joint Angles (rad)
Time (s)

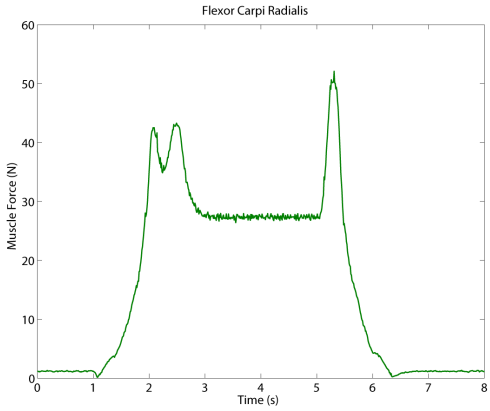

Tracking of the flexion/extension of the interphalangeal joint of the thumb
Reference
Output
Joint Angles (rad)
Time (s)

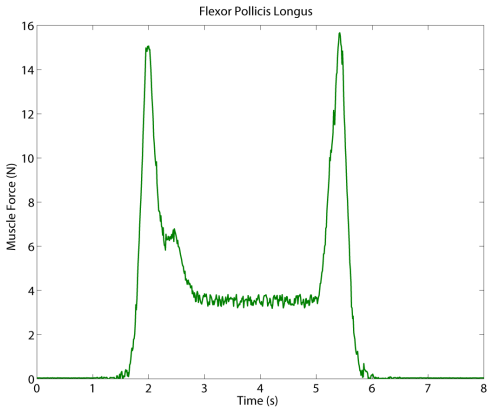
A.3.2 Muscle Forces (Control Actions)

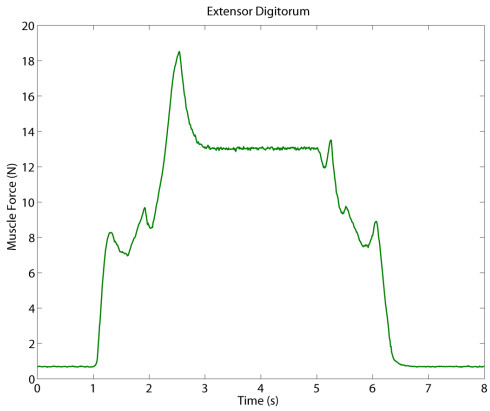

Flexor Digitorum Profudis
Muscle Force (N)
Time (s)

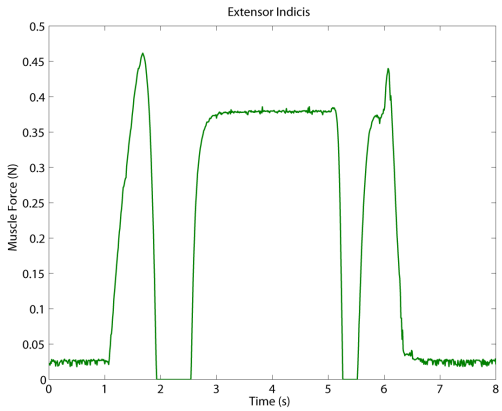

Flexor Digitorum Superficialis
Muscle Force (N)
Time (s)

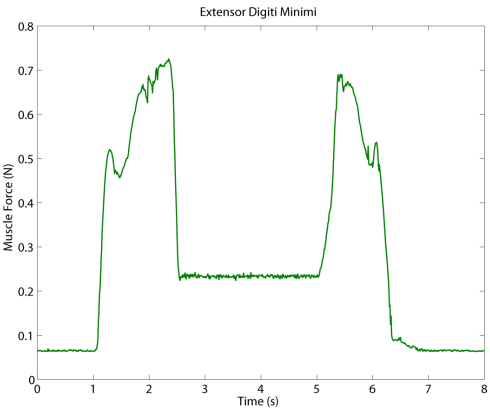

Flexor Carpi Ulnaris
Muscle Force (N)
Time (s)

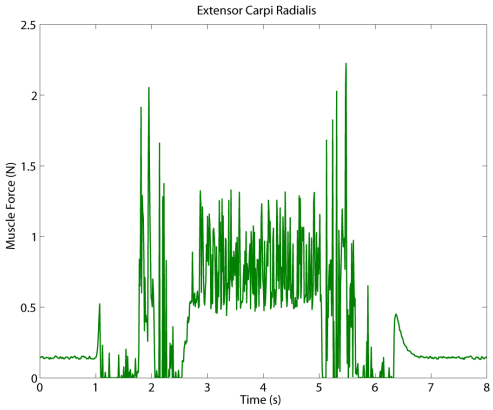

Flexor Carpi Radialis
Muscle Force (N)
Time (s)

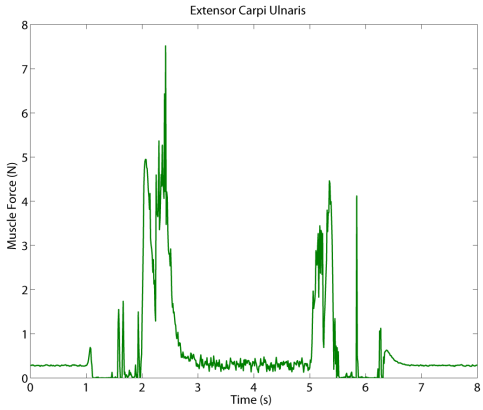

Flexor Pollicis Longus
Muscle Force (N)
Time (s)

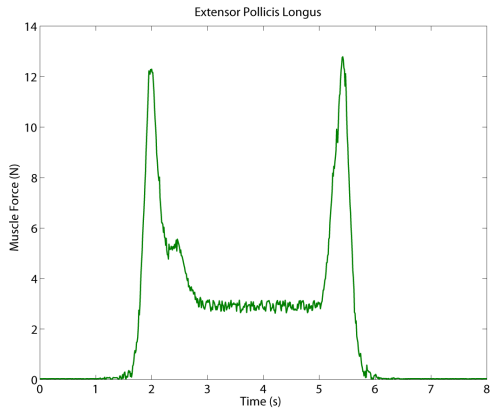

Extensor Digitorum
Muscle Force (N)
Time (s)

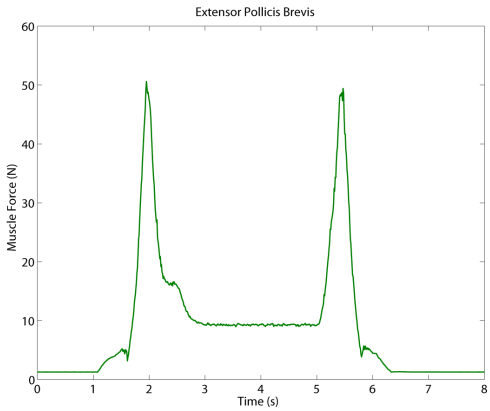

Extensor Indicis
Muscle Force (N)
Time (s)

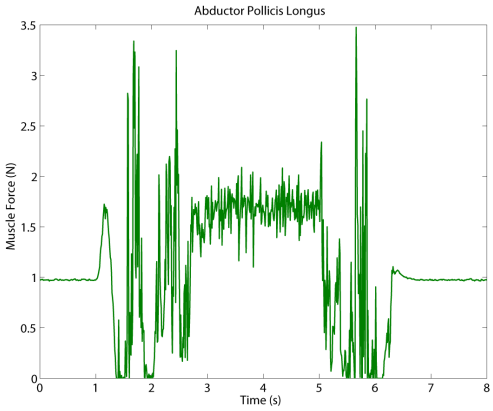

Extensor Digiti Minimi
Muscle Force (N)
Time (s)

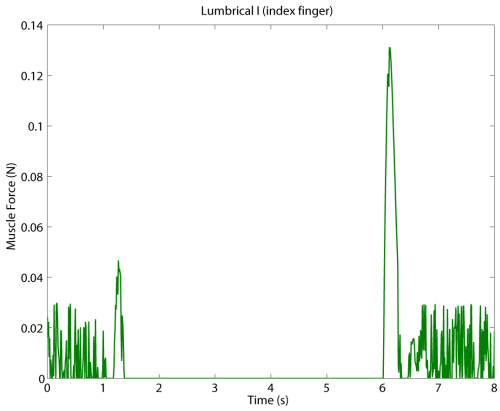

Extensor Carpi Radialis
Muscle Force (N)
Time (s)

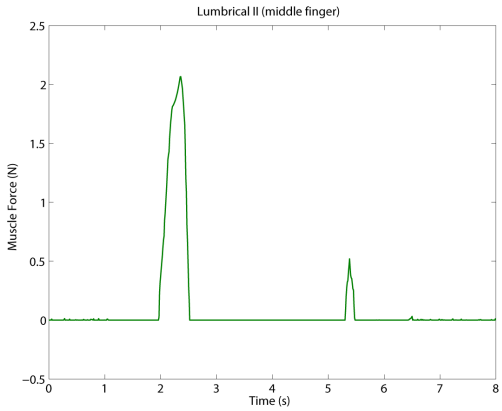

Extensor Carpi Ulnaris
Muscle Force (N)
Time (s)

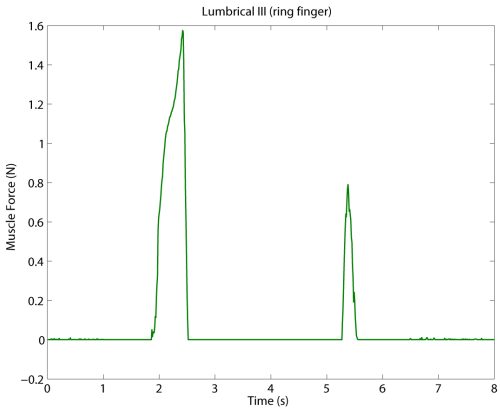

Extensor Pollicis Longus
Muscle Force (N)
Time (s)

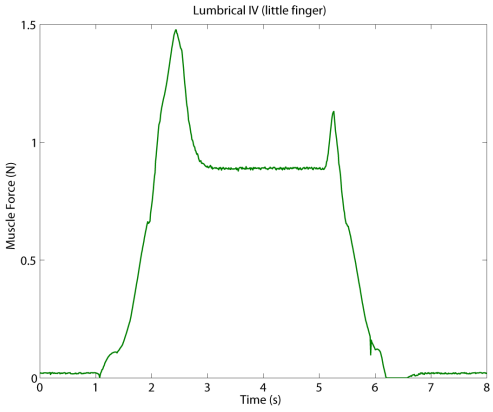

Extensor Pollicis Brevis
Muscle Force (N)
Time (s)

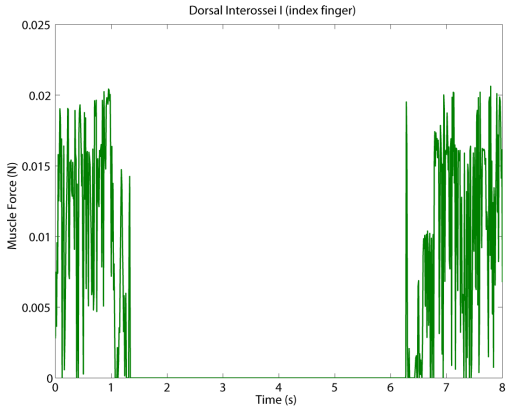

Abductor Pollicis Longus
Muscle Force (N)
Time (s)

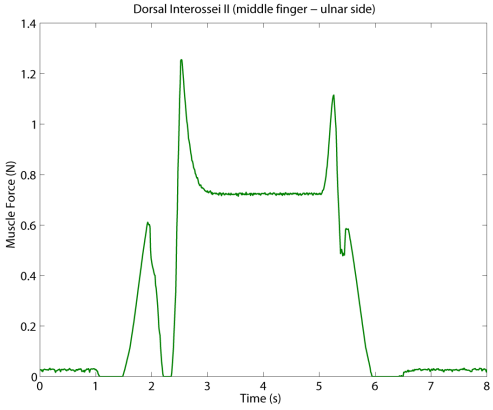

Lumbrical I (index finger)
Muscle Force (N)
Time (s)

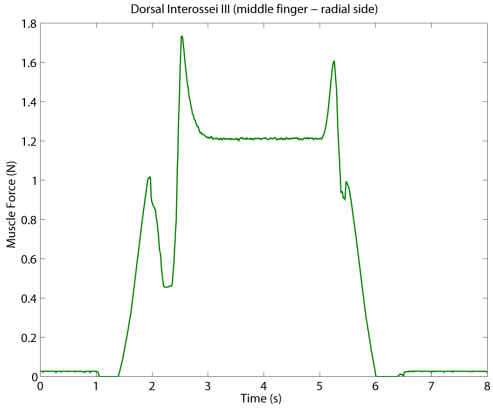

Lumbrical II (middle finger)
Muscle Force (N)
Time (s)

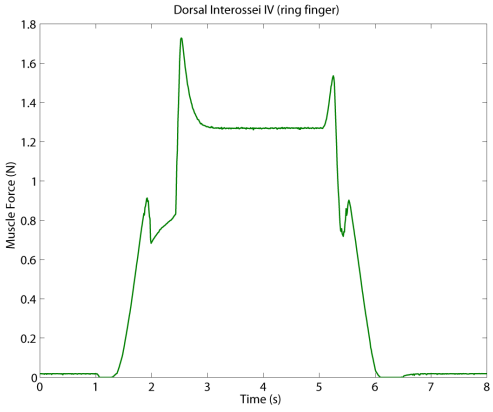

Lumbrical III (ring finger)
Muscle Force (N)
Time (s)

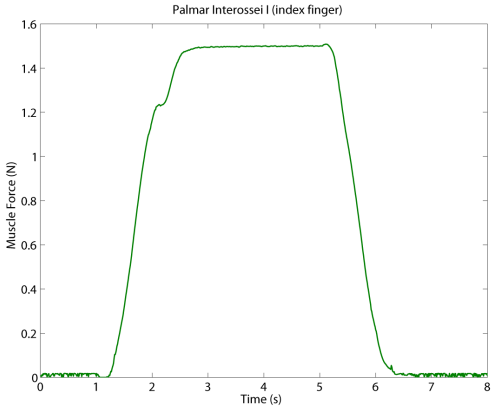

Lumbrical IV (little finger)
Muscle Force (N)
Time (s)

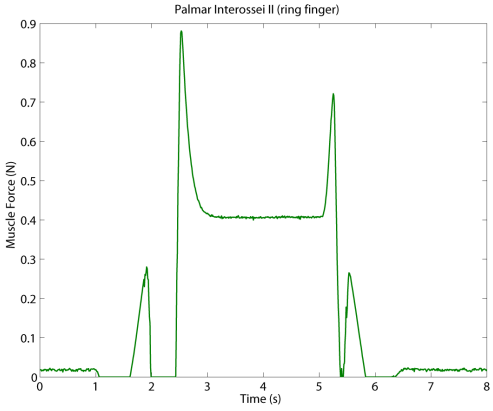

Dorsal Interossei I (index finger)
Muscle Force (N)
Time (s)

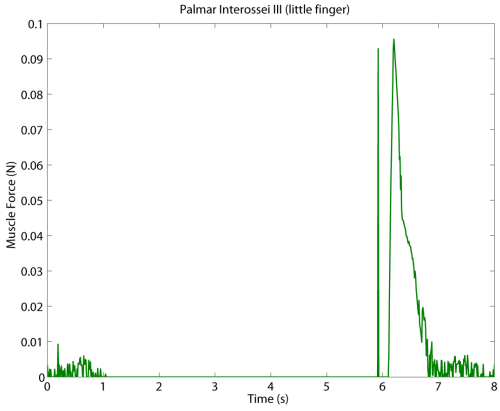

Dorsal Interossei II (middle finger – ulnar side)
Muscle Force (N)
Time (s)

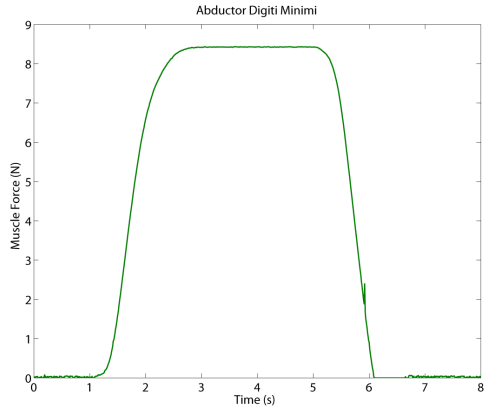

Dorsal Interossei III (middle finger – radial side)
Muscle Force (N)
Time (s)

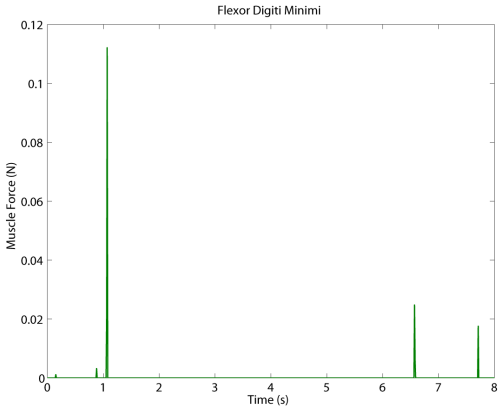

Dorsal Interossei IV (ring finger)
Muscle Force (N)
Time (s)

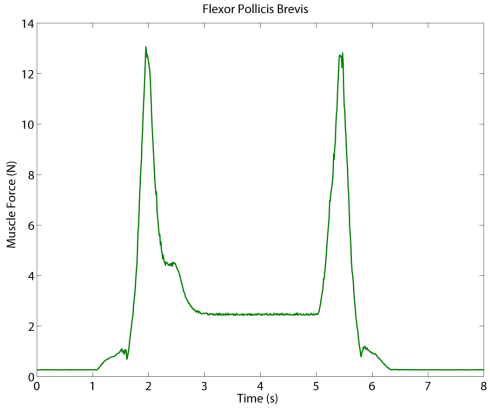

Palmar Interossei I (index finger)
Muscle Force (N)
Time (s)

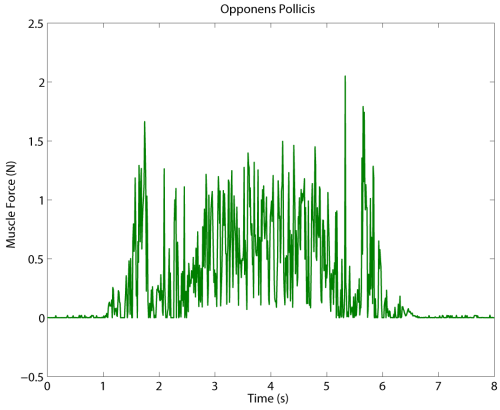

Palmar Interossei II (ring finger)
Muscle Force (N)
Time (s)

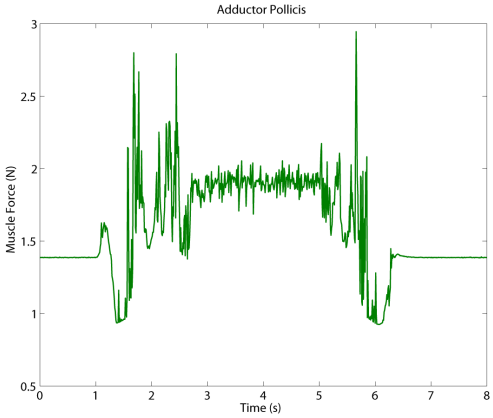

Palmar Interossei III (little finger)
Muscle Force (N)
Time (s)

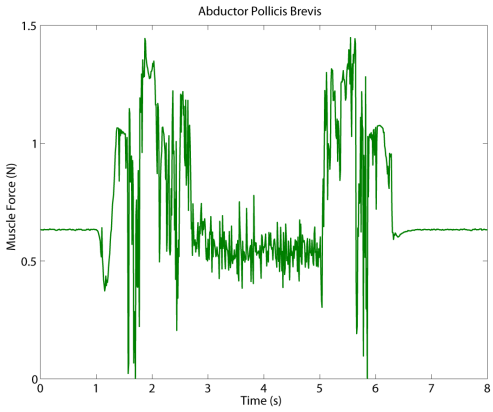

Abductor Digiti Minimi
Muscle Force (N)
Time (s)


Flexor Digiti Minimi
Muscle Force (N)
Time (s)


Flexor Pollicis Brevis
Muscle Force (N)
Time (s)


Opponens Pollicis
Muscle Force (N)
Time (s)


Adductor Pollicis
Muscle Force (N)
Time (s)


Abductor Pollicis Brevis
Muscle Force (N)
Time (s)

Appendix B

Results from the Stationary Material Tests

[1]The number of active layer in the stack was 15

Table B.1: Values obtained from the stationary tests to the EAP stack.

Applied Load (N)	Length (mm)	Capacity per layer (nF)[1]	Capacity per layer (Vacuum) (nF)[1]	Dielectric Constant	Cross-section (m^2)	True Stress (kPa)	True Strain	Engineering Strain
0	8.8000	0.0384	0.0047	8.146	3.61×10^{-4}	0	0	0
-0.60 25	8.2923	0.0398	0.0050	7.877	3.64×10^{-4}	-1.65	-0.0594	-0.0577
-0.9968	7.7702	0.0414	0.0056	7.446	3.75×10^{-4}	-2.66	-0.1245	-0.1170
-2.0000	7.3896	0.0435	0.0060	7.247	3.85×10^{-4}	-5.19	-0.1747	-0.1603
-2.9899	7.1602	0.0451	0.0063	7.159	3.92×10^{-4}	-7.63	-0.2062	-0.1863
-4.9945	6.7663	0.0484	0.0070	6.902	4.12×10^{-4}	-12.1	-0.2628	-0.2311
-10.035	5.8082	0.0581	0.0088	6.591	4.45×10^{-4}	-22.5	-0.4155	-0.3400
-15.006	5.2554	0.0671	0.0106	6.357	4.82×10^{-4}	-31.1	-0.5155	-0.4028
-20.006	4.7696	0.0757	0.0121	6.251	5.02×10^{-4}	-39.9	-0.6125	-0.4580
-25.001	4.3864	0.0831	0.0135	6.144	5.15×10^{-4}	-48.5	-0.6962	-0.5051
-30.010	4.0114	0.0885	0.0155	5.703	5.41×10^{-4}	-55.5	-0.7856	-0.5442
-34.996	3.7758	0.0931	0.0176	5.300	5.76×10^{-4}	-6.08	-0.8461	-0.5709

Appendix C

Predicted Stationary Strains for the Actuator under an Applied Voltage

Using Eq. 4.4 one can find an estimative of the stationary value of the strain when under an applied voltage. Combining Eq. 4.4 with Table B.1, one obtains the values in Table C.1.

From a quick analysis to the results, one can see that there is an optimum point (point with the greatest strain) for an applied load of -25N.

Table C.1: Predicted values for the stationary strain of the actuator when under an -6kV applied voltage.

Applied Load (N)	Length (mm)	Dielectric Constant	True Stress (kPa)	True Strain	Maxwell Stress (kPa)	Stationary True Strain	Actuation Strain (True Strain)	Actuation Strain (Engineering Strain)
-0.60 25	8.2923	7.877	-1.65	-0.0594	-3.085	-0.134	-0.0747	-7.200%
-0.9968	7.7702	7.446	-2.66	-0.1245	-3.321	-0.155	-0.0304	-2.999%
-2.0000	7.3896	7.247	-5.19	-0.1747	-3.574	-0.201	-0.0261	-2.572%
-2.9899	7.1602	7.159	-7.63	-0.2062	-3.760	-0.243	-0.0365	-3.584%
-4.9945	6.7663	6.902	-12.1	-0.2628	-4.060	-0.316	-0.0536	-5.219%
-10.035	5.8082	6.591	-22.5	-0.4155	-5.262	-0.480	-0.0647	-6.262%
-15.006	5.2554	6.357	-31.1	-0.5155	-6.198	-0.598	-0.0830	-7.962%
-20.006	4.7696	6.251	-39.9	-0.6125	-7.400	-0.706	-0.0936	-8.939%
-25.001	4.3864	6.144	-48.5	-0.6962	-8.600	-0.797	-0.1011	-9.618%
-30.010	4.0114	5.703	-55.5	-0.7856	-9.546	-0.860	-0.0741	-7.143%
-34.996	3.7758	5.300	-6.08	-0.8461	-10.014	-0.898	-0.0522	-5.082%

Appendix D

Additional Results from the Performance Tests of the Actuator

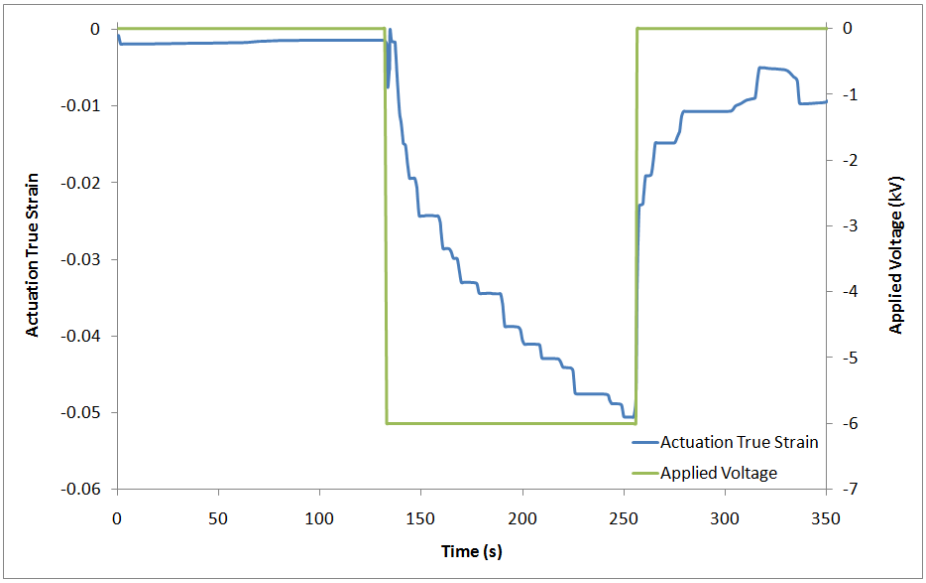


Figure D.1: Strain versus time results.

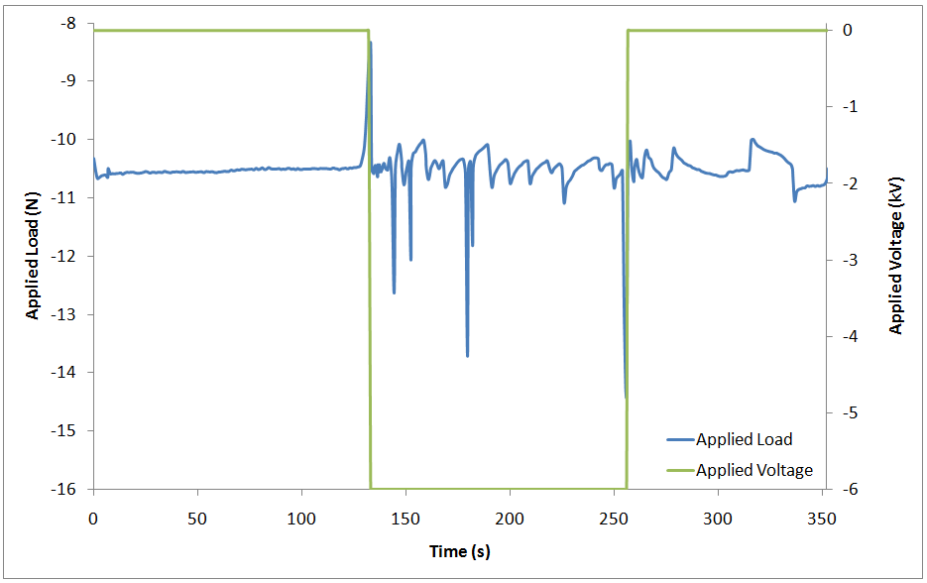


Figure D.2: Load versus time results.

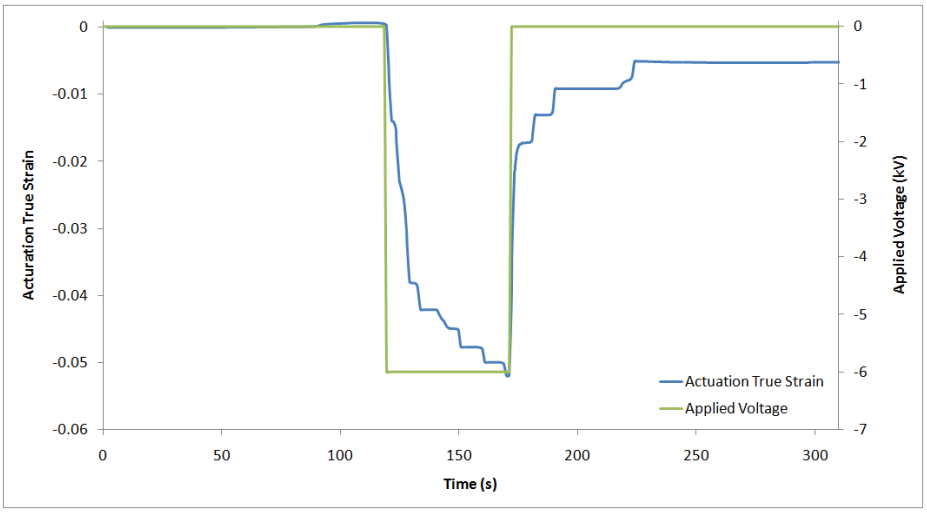


Figure D.3: Strain versus time results.

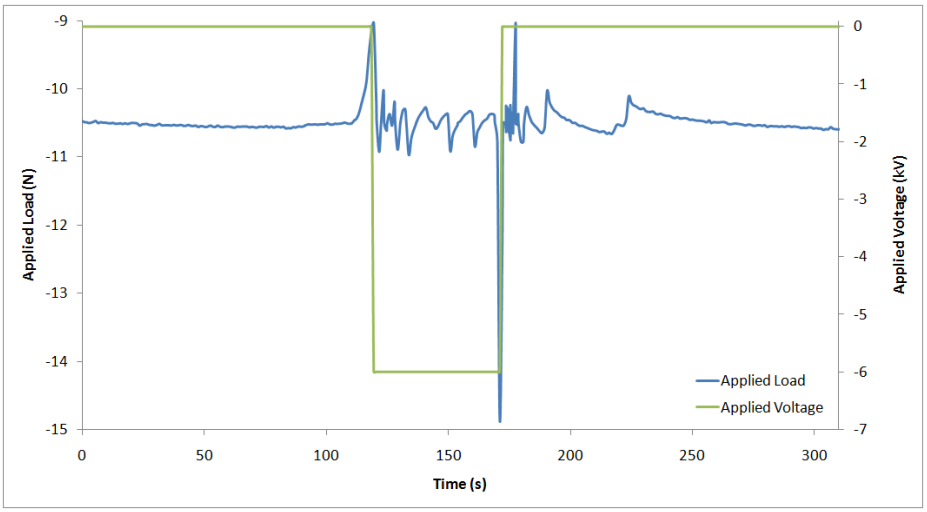


Figure D.4: Load versus time results.

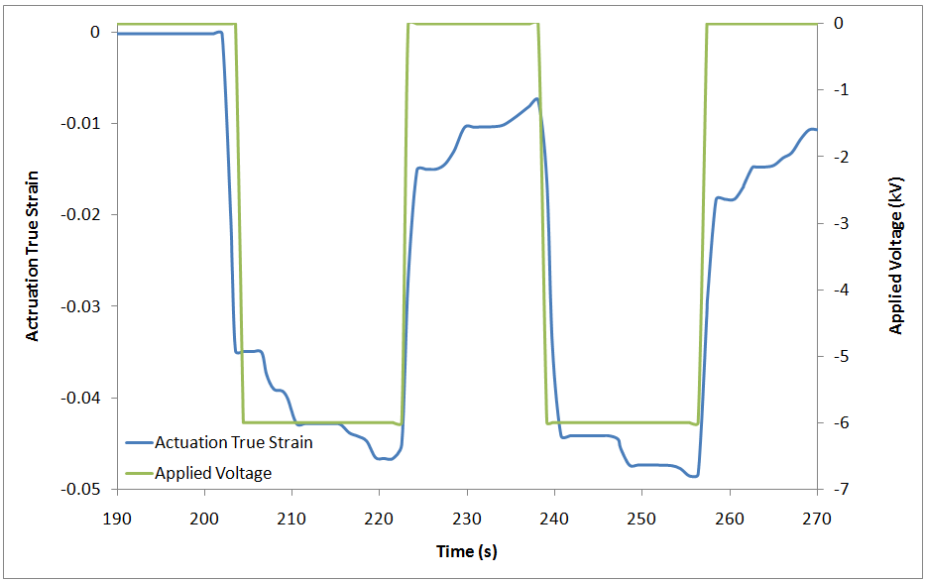


Figure D.5: Strain versus time results.

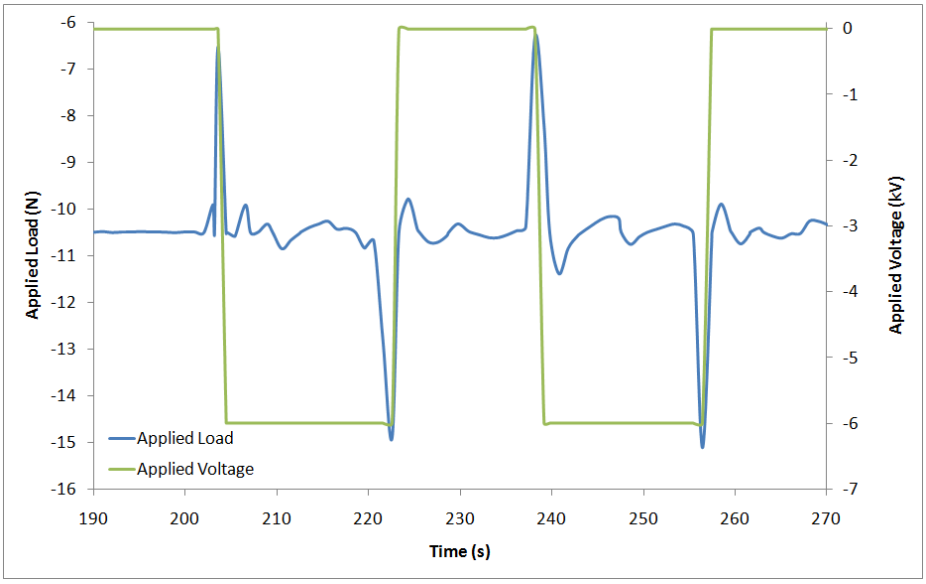


Figure D.6: Load versus time results.